UNIVERSIDAD DE LA REPÚBLICA
FACULTAD DE INGENIERÍA

TELEMETRÍA PARA GLOBOSATS

FEBRERO 2009

Emiliano Pastorino
Natacha Reyes
Inés Scapinello

TUTORES
Ing. Juan Pechiar
Ing. Leonardo Steinfeld

CONTENIDO

1. INTRODUCCIÓN

Este proyecto está ligado al Proyecto Laí, el cual fue creado por el Grupo de Tecnología Aeroespacial perteneciente al Instituto de Ingeniería Eléctrica y cuyo objetivo es la construcción, puesta en órbita y operación de un satélite experimental. Debido al gran reto que implica el diseño de satélites, el paso inicial para el desarrollo de los mismos comprende la implementación de Globos Satelitales, puesto que estos permiten adquirir experiencia a bajo costo y de manera efectiva.

Los Globos Satelitales (GloboSats) consisten en una plataforma autónoma suspendida de un globo de tipo meteorológico la cual permite llevar a cabo ensayos a gran altura. Los ensayos conllevan recabar información atmosférica, de posicionamiento y de radiación aproximadamente a 30km de altura y los resultados se transmiten a frecuencias de radioaficionados en tiempo real. Una vez que el globo explota por expansión y cae a tierra amortiguado por un paracaídas se recupera el sistema de información.

Los programas de Globos Satelitales vienen siendo implementados hace ya varios años en diversas universidades de todo el mundo. Estos proyectos involucran años de desarrollo, en un esfuerzo conjunto de un equipo multidisciplinario de ingeniería de diversas áreas como la mecánica, eléctrica, computación y materiales, así como el inestimable apoyo de sus sponsors. Es de destacar que a nivel amateur, varios grupos de radioaficionados están utilizando éstos desarrollos para reportes de telemetría, dada su eficiencia y bajo costo.

Actualmente existen más de cuarenta instituciones a nivel mundial que han desarrollado la modalidad de “cubos satélites” que han desencadenado retos tecnológicos y científicos y que se encuentran en diferentes grados de avance.

Dentro del Instituto de Ingeniería Eléctrica, a la fecha existen otros tres proyectos de grado trabajando en la construcción de GloboSats: HWGloboSats, TelemetríaUHF y GloboSatIIE. Este último tiene como principal objetivo tomar todo el trabajo realizado hasta el momento en GloboSats y realizar una nueva versión más

efectiva, eficiente, reproducible, y con procedimientos documentados para construcción, integración, pruebas, liberación y rescate.

2. **OBJETIVO**

El objetivo principal del proyecto comprende el desarrollo del sistema de telemetría para Globos Satelitales. Ello implica determinar y desarrollar el sistema de comunicación entre el GloboSat y las estaciones terrestres, así como el software de control de los diferentes módulos para la plataforma. Este trabajo está estrechamente relacionado con el proyecto HWGloboSats, pues el éxito del mismo depende en parte del hardware que ellos desarrollen.

El sistema deberá prever telemetría / control para los siguientes módulos / sensores:

- sensores de temperatura
- sensor de presión (altímetro)
- sensor de estado de batería
- GPS
- datos de experimentos asociados a ensayos de radiación
- control de experimentos asociados (activación, comandos de configuración)
- control del suministro de energía
- control de separación del globo

Parte de la telemetría será transmitida en claro y en formatos estándar (ej. posición, GPS, temperaturas). La telemetría de los datos hacia tierra, serán cifrados en algún modo a determinar. Se estudiarán métodos de corrección de errores hacia adelante, o retransmisiones, de modo de no perder datos de experimentos asociados.

Se proveerá un software de toma de decisiones en caso de pérdida de telemetría. Por ejemplo, separación del globo si no hay telemetría y se escapa de una región predeterminada.

Los protocolos y modos de transmisión a utilizarse serán compatibles con el común de las estaciones terrenas de aficionados.

Como opcional, se propuso la implementación de un transmisor FM.

3. SOLUCIÓN IMPLEMENTADA

Para cumplir con los objetivos planteados se dividió el trabajo en tres bloques:

- Módulo Principal: software que se encarga de recabar información de telemetría, armar las tramas según el protocolo APRS, controlar el suministro de energía e implementar de un plan de vuelo para casos en que la carga se escape de un área predefinida.
- Modulación de los datos a transmitir según el protocolo Bell 202 (AFSK).
- Construcción de un transmisor FM en la frecuencia de 145.030Mhz.

Se trabajó conjuntamente con el grupo HWGoboSats para la realización del módulo principal. Ellos fueron los encargados de escoger el modelo de microcontrolador, los sensores de temperatura y presión y el GPS. Se acordó programar el micro en lenguaje C, donde el grupo HW programó los conversores A/D para leer los sensores y el voltaje de la batería, los puertos serie para obtener información proveniente del GPS, la placa FPGA (para los ensayos de radiación) y el puerto Usart por donde se envía la información de telemetría obtenida.

La programación del bucle principal involucró el armado de la trama con los datos obtenidos, encabezados y banderas, respetando el formato APRS para así poder ser interpretada por las estaciones terrenas destinatarias. Este programa incluye el diseño de una cerca electrónica para controlar la liberación de la carga del globo. Además, se controla la tasa de transferencia de datos hacia el exterior en función de la carga de la batería.

La modulación de los datos se realizó de manera independiente del programa principal, para ello se utilizó un microprocesador PIC16F887A de Microchip. A medida que se recibe la trama a través del puerto Usart, enviada por el Atmel, se le agregan las banderas restantes, se calcula el CRC para el control de errores, y se envía a través de un puerto hacia el transmisor de FM por medio de un conversor D/A. El código se escribió en lenguaje Assembler.

Si bien la implementación de un transmisor FM se definió como opcional en el plan de proyecto inicial, los tiempos permitieron avanzar en la construcción del mismo. En este momento se cuenta con un prototipo inicial, el cual requiere de ajustes para lograr el desempeño deseado.

En las siguientes secciones se presentan las herramientas utilizadas, así como las soluciones implementadas para satisfacer los requerimientos planteados en los objetivos.

4. ESPECIFICACIÓN DE TELEMETRÍA

En esta sección se introducen los conceptos teóricos utilizados como base para el desarrollo del software del sistema de comunicación entre la plataforma de telemetría y las estaciones de radio en tierra.

Se comienza haciendo una breve descripción del modelo de referencia OSI para luego situarse en las capas uno y dos específicamente donde se desarrolla este trabajo.

A continuación se especifican brevemente los protocolos implementados en las diversas capas. En primer lugar se detallan los protocolos X.25 y AX.25, los cuales proveen las pautas para el armado de los paquetes a nivel de capa de enlace.

Luego se describe el protocolo APRS (Automatic Packet/Position Reporting System), protocolo de comunicaciones basado en el AX.25 ampliamente utilizado por radioaficionados para la difusión de datos en tiempo real, de forma libre, a través de una red.

Finalmente se presentan los tipos de modulación de datos digitales más utilizados para la modulación de las tramas, haciendo hincapié en la señalización AFSK (Audio Frequency Shift Keying), utilizada en este trabajo.

4.1. MODELO DE CAPAS PARA COMUNICACIONES

El modelo de referencia OSI es un armazón alrededor del cual debe ser concebido el protocolo de comunicación.

La primera capa o capa física está compuesta por todo lo que afecta físicamente (tanto desde el punto de vista mecánico como desde el punto de vista eléctrico) al flujo de bits de un lugar hacia otro. La segunda capa o capa de enlace tiene como objeto colocar los datos en tramas y proporcionar una transferencia exenta de errores. Se añaden a esta trama informaciones de direcciones del expedidor y del destinatario. Se calcula un código ("Cyclic Redundancy Check" o CRC) para cada trama, este código es verificado por la estación destinataria con el código de la estación expedidora. Si no hay concordancia la trama es rechazada. La tercera capa o capa de red, concierne a la ruta de las tramas a través de una red. Esto se obtiene gracias a añadir información de ruta en cada trama. La misión de la cuarta capa o capa de transporte es la de mantener una conexión asegurando que los datos sean encaminados de la fuente hacia el destinatario. La quinta capa o capa de sesión, gestiona el log-on y la autentificación. La sexta capa o capa de presentación proporciona el medio de traducir o de interpretar los datos intercambiados. La séptima capa o capa de aplicación proporciona la interfaz entre el modelo de referencia y la aplicación del usuario.

4.2. PROTOCOLO X.25

X.25 es un conjunto de protocolos usados para establecer la conexión entre el equipo terminal de datos (Data Terminal Equipment o DTE) y el equipo de terminación de circuito de datos (Data Circuit Terminating Equipment o DCTE) de una red de conmutación de paquetes (packet switched data network o PSDN). Es decir, X.25 se utiliza como protocolo en el interfaz de acceso a una red de conmutación de paquetes.

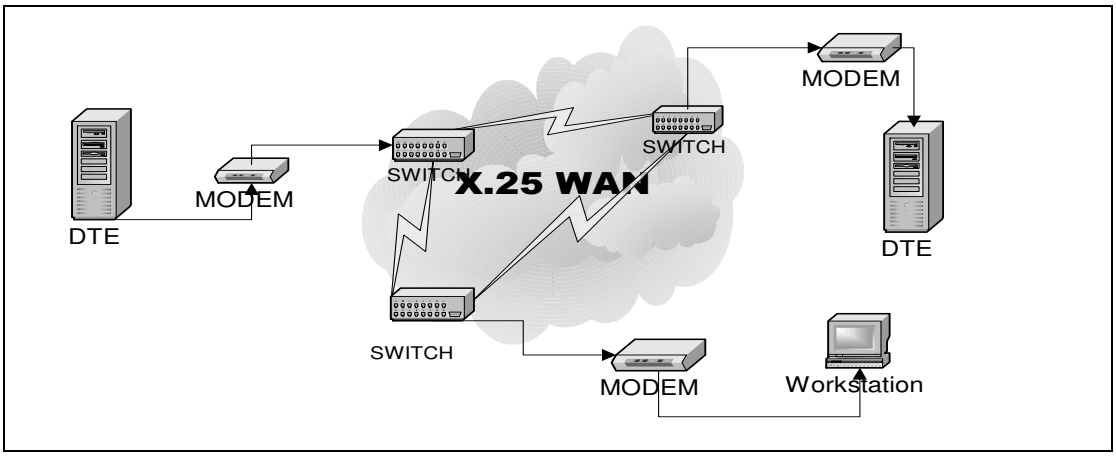

Figura 1 Esquema X.25

X.25 trabaja sobre servicios basados en circuitos virtuales (VC). Un circuito virtual o canal lógico es aquel en el cual el usuario percibe la existencia de un circuito físico dedicado exclusivamente al ordenador o equipo que el maneja, cuando en realidad ese circuito físico "dedicado" lo comparten muchos usuarios. Mediante diversas técnicas de multiplexado estadístico, se entrelazan paquetes de distintos usuarios dentro de un mismo canal. Las prestaciones del canal son lo bastante buenas como para que el usuario no advierta ninguna degradación en la calidad del servicio como consecuencia del tráfico que le acompaña en el mismo canal. Para identificar las conexiones en la red de los distintos DTE, en X.25 se emplean números de canal lógico (LCN). Pueden asignarse hasta 4095 canales lógicos y sesiones de usuario a un mismo canal físico.

4.3. PROTOCOLO AX.25

El AX.25 Amateur Packet Radio Link Layer Protocol, fue adoptado por la ARRL *(*American Radio Relay League) en octubre de 1984 y fue reconocido por la mayoría de las administraciones encargadas de controlar el servicio de aficionados. Actualmente está reconocido a través del mundo como "el protocolo estándar de packet radio".

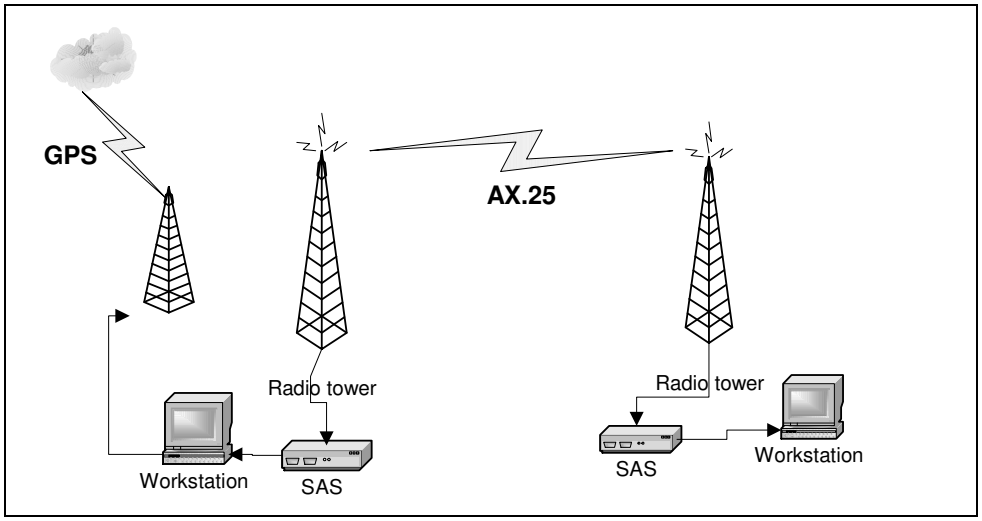

Figura 2 Esquema AX.25

La diferencia más relevante entre el protocolo X.25 comercial y el AX.25 radica en el formato de las direcciones. En X.25 son del tipo numérico, son números de teléfono, y en AX.25 son del tipo alfanumérico, son indicativos de radioaficionado que no son más que una combinación de letras y números que no superan los 6 caracteres. Además para distinguir distintas estaciones de un mismo indicativo, se definió el SSID (Secondary Station Identifier) que consiste en un número del 0 al 15.

Tal y como está definido, el AX.25 funciona tanto en semi- como en full-duplex. El AX.25 autoriza las conexiones múltiples y la conexión consigo mismo.

Una transmisión AX.25 se compone de pequeños bloques de información denominados tramas. Una trama está a su vez dividida en elementos más pequeños llamados campos. Existen tres tipos de tramas:

- Las tramas de información o tramas I, contienen la información enviada de una estación hacia otra.
- Las tramas de supervisión o de control o tramas S, realizan el control de la comunicación, por ejemplo acuse de recepción de una trama del tipo I o solicitud de repetición.
- Las tramas no numeradas o tramas U.

Cada trama comprende un campo bandera (flag), un campo de direcciones, un campo de control, un campo de información, un campo FCR y un campo con la bandera final.

El campo de flag o bandera indica el comienzo y el final de una trama. Una sola bandera puede ser compartida por dos tramas, en este caso la bandera indica el final de la trama precedente y el comienzo de la trama siguiente. La longitud del flag es de 8 bits (1 octeto) y su valor es único que vale:

0111 1110 o sea 7E en valor hexadecimal.

Cuando se dice que su valor es único, quiere decir que no se podrá, en ningún caso, encontrar una secuencia igual, es decir donde 6 bits "1" estén seguidos. Para realizar este imperativo, existe un procedimiento que se denomina bit stuffing (o inserción de cero) que consiste en añadir, en la emisión, un cero cada vez que se encuentren 5 bits consecutivos de valor 1. Por supuesto, se debe incorporar un procedimiento inverso en la recepción: cada vez que se encuentren 5 bits consecutivos de "1", hay que suprimir el "0" que sigue.

El campo de direcciones contiene el indicativo de la fuente y del destinatario de la trama. Puede también contener el indicativo de uno de los 8 digipeaters (repetidores) según AX.25 Versión 2. La longitud del campo de dirección varía de 14 a 70 octetos (112 a 560 bits). Los 7 primeros octetos del campo de direcciones contienen el indicativo y el SSID del destinatario. El indicativo tendrá un máximo de 6 caracteres, el codificado se hace sobre 7 bits, siendo siempre el octavo bit 0, excepto para el último

indicativo mencionado en el campo de dirección. El SSID permite utilizar el mismo indicativo varias veces. Para el octeto que contiene el SSID,

- El primer bit, llamado "H", - es 0 si se trata del expedidor o del destinatario, - y en el caso de indicativos de repetidores, es 1 si la trama ha sido repetida por el digipeater y 0 en caso contrario.
- Los bits 2 y 3 se denominan "R" y están reservados a un uso local o futuro
- Los bits 4 a 7 están reservados al SSID propiamente dicho, por lo que puede tomar todos los valores comprendidos entre 0 y 15,
- El bit 8 llamado bit de extensión es siempre 0 excepto si se trata del último indicativo (fuente o repetidor) mencionado en el campo de dirección.

Los octetos 8 a 14 contienen el indicativo y el SSID del que ha transmitido la trama (fuente). Los octetos 15 a 70 son facultativos y llevan pues los indicativos y SSID de los diferentes digipeaters. El campo de control tiene una longitud de un octeto, la tabla siguiente resume los diferentes valores que puede tomar:

Tipo de trama		Bits							
		7	6	5	4	3	2	1	0
I		N(R)			P	N(S)			0
S	Receiver ready: *RR*	N(R)			P/F	0	0	0	1
	Receiver not ready: *RNR*	N(R)			P/F	0	1	0	1
	Reject: *REJ*	N(R)			P/F	1	0	0	1
U	Set Asynchronous balanced mode: *SABM*	0	0	1	P	1	1	1	1
	Disconnect: *DISC*	0	1	0	P	0	0	1	1
	Disconnected mode: *DM*	0	0	0	F	1	1	1	1
	Unnumbered acknowledge: *UA*	0	1	1	F	0	0	1	1
	Frame reject: *FRMR*	1	0	0	F	0	1	1	1
	Unnumbered Information: *UI*	0	0	0	P/F	0	0	1	1

Tabla 1 Tipos de tramas

- N(S) número de secuencia en emisión es decir el número de la trama transmitida
- N(R) número de secuencia en recepción es decir el que espera recibir, gracias a este número se sabe que el receptor ha recibido correctamente todas las tramas hasta N(R)-1
- P/F bit poll/final solicita respuesta inmediata a una trama (poll) o responde a una solicitud de respuesta inmediata (final).

El PID (Protocol IDentifier) no está presente más que en las tramas I y UI e indica el tipo de red que es utilizada. Los valores de PID utilizados más comúnmente son:

- Tráfico packet radio normal AX.25 level 2 version 2 : F0
- La Net/Rom y TheNet : CF
- TCP/IP : CC y CD

El campo de información contiene los datos que son transmitidos por una trama de tipo I, de tipo UI o por una trama de rechazo de trama (FRMR). La longitud máxima es de 256 octetos.

El campo de Frame Check Sequence FCS se utiliza como detección de error, contiene un número de 16 bits (o sea 2 octetos) que se calcula según la Recomendación ISO 3309. De hecho se toma el número binario formado por todos los bits enviados, se divide por $x^{16} + x^{15} + x^2 + 1$ o sea 1100 000 000 0101 y el resto de la división proporciona el número que será utilizado como FCS. Para este campo se trasmite primero el bit más significativo, para todos los demás campos se trasmite primero el bit menos significativo.

En la recepción se pone en juego un procedimiento idéntico y el resultado del FCS calculado en la recepción, y el transmitido en la trama serán comparados, para ver si son idénticos, y se concluirá que la transmisión es correcta.

La utilización de los tipos de tramas de supervisión (S) es la siguiente:

- Una trama del tipo Receiver Ready (RR) señala al corresponsal que todas las tramas han sido recibidas correctamente y que puede continuar enviando nuevas tramas.
- Una trama del tipo Receiver Not Ready (RNR) señala que todas las tramas han sido recibidas correctamente, pero que no se puede de todas formas transmitir nuevas tramas.
- Una trama del tipo Reject (REJ) pide la repetición a partir de la trama N(R), habiéndose recibido sin error las tramas precedentes.

La utilización de las tramas de informaciones no numeradas (U) es la siguiente:

- Una trama del tipo Set Asynchronous Balanced Mode (SABM) inicia la conexión entre dos estaciones de packet. La estación destinataria puede responder con un UA o un DM.
- Una trama del tipo Unnumbered Ack (UA) se envía como respuesta a una trama del tipo SABM, cuando la conexión es realizable.
- Una trama del tipo Disconnect Mode Response (DM) se envía como respuesta a una trama del tipo SABM, cuando la conexión no es realizable (estación ocupada).
- La trama de Frame Reject (FRMR) se envía cuando la estación es incapaz de tratar la información y la expedición de una nueva trama no arregla las cosas. Este tipo de trama tiene en el campo informaciones, 3 octetos que especifican la naturaleza del error.
- Las tramas no numeradas del tipo Unnumbered Information (UI) permiten enviar informaciones sin que esté establecida la conexión. Como no puede haber corrección de error, no se puede dar ninguna garantía sobre la validez de esta trama.

La tramas UI contienen los campos PID y de información y transportan información a través del enlace por fuera de los controles normales de información. Esto permite el transporte de información por el enlace sin utilizar control de flujo. Como

estas tramas no necesitan respuesta (acknowledge), si una trama UI se pierde, no hay forma de recuperarla.

Una trama UI que es recibida con el bit P en 1 provoca que una respuesta sea trasmitida. Esta respuesta es una trama DM cuando se está en estado desconectado o RR si se está en estado de transferencia de información.

En este momento se tiene la secuencia de bits que se transmitirá por la radio. A continuación se realiza la modulación de los bits, tarea que corre a cargo del módem. Dependiendo de la velocidad utilizada, se suele realizar un tipo de modulación u otro, que puede llevar consigo un nuevo proceso de codificación de los bits. [1]

4.4. PROTOCOLO HDLC

El protocolo AX.25 utiliza a nivel de capa 2 del modelo OSI (capa de enlace) el protocolo HDLC (High-Level Data Link Control).

El Protocolo HDLC está diseñado para proporcionar un mecanismo de detección y corrección de errores de propósito general a los enlaces digitales; este protocolo es muy extenso, por lo que rara vez se utiliza la implementación completa; lo normal es que se utilicen subconjuntos.

La tarea principal del nivel de enlace consiste en, a partir de un medio de transmisión común y corriente, transformarlo en una línea sin errores de transmisión para la capa de red (capa 3).

Los protocolos del nivel de enlace definen, típicamente, reglas para: iniciar y terminar un enlace (sobre un circuito físico previamente establecido), controlar la correcta transferencia de información y recuperarse de anomalías.

El HDLC consiste en tramas de bits que están delimitadas por unas banderas de 8 bits de longitud que contienen el valor 01111110 binario. Cuando el receptor encuentra este valor en el canal, comienza la lectura de una trama, lectura que termina

cuando vuelve a encontrar este mismo valor. Nótese que una bandera puede indicar, simultáneamente, el final de una trama, y el comienzo de la siguiente. Puesto que dentro de una trama, en el campo de datos de usuario puede aparecer este valor, el transmisor insertará automáticamente un bit a 0 detrás de cada bloque de cinco bits a 1; el receptor, a su vez, eliminará cada bit a 0 que siga a un bloque de cinco bits a 1; con este esquema se garantiza que nunca aparecerá el valor de la bandera dentro de los bits de datos, es decir, el usuario puede colocar cualquier información dentro del paquete, la transmisión es totalmente transparente.

Las tramas incorporan una dirección, un código de control y unos números de secuencia. Los números de secuencia de recepción indican el número de secuencia de la siguiente trama que se espera recibir; así, si una trama es recibida correctamente, este valor se incrementará, haciendo que el emisor mande la siguiente trama; si la trama se pierde el valor permanecerá igual, con lo que el emisor la volverá a enviar.
Las tramas de control gestionan fundamentalmente el control de flujo y la notificación de errores. [2]

4.5. PROTOCOLO APRS

El APRS (Automatic Position Report System) es un sistema automático de información de posición, es decir que permite ver en un mapa la posición en la que está una estación fija o móvil de radioaficionado. También tiene otras capacidades como poder ver información meteorológica, señalización en el mapa de todo tipo de eventos (catástrofes, puntos de interés para el radioaficionado). En el seguimiento de estaciones móviles se aprovecha la tecnología que brindan los GPS, que conectados a un equipo de radio sirven para seguir en el mapa a un vehículo.

El APRS utiliza para transmitir los datos el protocolo AX.25, por lo tanto es compatible con cualquier módem sin suponer un coste añadido. Este protocolo fue creado con fines no comerciales en 1984 por Bob Bruninga WB4APR.

Desde su creación, este protocolo ha evolucionado, cambiando y mejorando para adaptarse a nuevas necesidades o utilidades. Esta evolución, llevó a la creación de un

comité que lidera la Tucson Amateur Packet Radio, que es una asociación Americana especializada en comunicaciones digitales. Este comité que reúne a los principales desarrolladores de APRS ha creado un documento en el que se definen el protocolo y todas las especificaciones del sistema APRS. Esto es muy importante para estandarizar y para los desarrolladores de software.

Para informar de su posición una estación transmite un paquete UI con las coordenadas geográficas en las que está ubicada. Por lo tanto se deberá acudir a un mapa de pequeña escala para decirle al programa de APRS en que coordenadas se está situado.

En estaciones móviles es el GPS el que mide la posición en grados segundos y minutos, la velocidad y el rumbo, la transmite al transceptor para ser enviadas por radio. Mientras que en estaciones fijas es conveniente transmitir una baliza cada 20-30 minutos en una estación móvil conviene transmitir cada 30 segundos o 1 minuto para poder hacer un buen seguimiento de la estación. En nuevos equipos hay otros métodos más eficaces como transmitir una baliza cuando el movimiento sea superior a una cifra, por ejemplo 100 metros. De esta forma se evita estar emitiendo continuamente cuando un móvil está parado.

Junto con la información de posición, el envío de mensajes es una de las características más importantes del APRS. Su uso es muy simple, simplemente hay que señalar en el mapa a que estación se quiere enviar el mensaje, escribirlo y enviarlo. En este punto el programa de APRS emite el mensaje en un paquete UI y espera a recibir una confirmación del destinatario también en un paquete UI, por lo que no es necesario realizar ninguna conexión. Si no se recibe la confirmación vuelve a emitir el mensaje hasta que se reciba. Si en cuatro intentos no se recibe el mensaje se descarta y se marca como no-enviado.

El APRS es práctico para transmitir mensajes pasando por hasta 4 repetidores, con lo que se pueden conseguir distancias de 400-500km dependiendo de la orografía de la región. Más lejos de estas distancias se hace poco práctico debido al retardo que se produce al ir pasando por muchos repetidores.

Cualquier equipo de VHF sirve para montar una estación APRS. De todo lo demás se encarga el software, por lo que sólo hace falta un PC. Existe un programa en MS-DOS para hacer APRS, pero conviene utilizar un PC con entorno gráfico, Windows, Linux, Mac. El programa más utilizado en Europa es el UIView. Se caracteriza por la facilidad de crear mapas personalizados, por lo tanto cualquiera puede escanear un mapa de su ciudad y utilizarlo en el programa. Es un programa muy fácil de utilizar y configurar. Existen además programas para MAC y para Linux.

Existen dos tipos de mapas para los programas de APRS, los vectoriales y los de imagen de bits. Los vectoriales son como los archivos de programas de CAD, es decir un conjunto de líneas que forman las carreteras, los ríos y toda línea que pueda tener un mapa. Tienen la ventaja de que la precisión es muy buena, ocupan poco y se puede hacer ampliaciones sobre ellos sin perder mucha calidad. También pueden estar formados por capas, por lo tanto podemos activar y desactivar la visualización de ciertos elementos como carreteras, vías de tren. El problema es que no son tan vistosos como los de mapa de bits. Los utiliza el programa WinAPRS muy utilizado en USA.

Los de imagen son cualquier mapa convertido a archivo gráfico mediante, por ejemplo, un scanner. Por lo tanto aquí se puede utilizar cualquier mapa que se disponga en papel o capturado de algún programa informático de mapas. La gran ventaja de estos mapas es la facilidad de obtención y su vistosidad. La desventaja es que no es práctico hacer ampliaciones sobre ellos ya que se pierde mucha calidad.

Se pueden distinguir dos tipos de estaciones: fijas y móviles. Entre las primeras se encuentran las de los radioaficionados, generalmente ubicadas en sedes de radioclubes, lugares aislados e incluso remotos, cumpliendo diversas funciones que se detallarán más adelante.

Los elementos mínimos e imprescindibles para disponer de una estación APRS, además del sistema transceptor y radiante, son:

- Módem para radiopaquete.
- Ordenador con programa específico para el sistema APRS.

4.5.1. DISTRIBUCIÓN DE LA INFORMACIÓN

Con una cadencia predefinida, las estaciones APRS emiten sus peculiares balizas conteniendo identificación e información adicional, que son repetidas por una o más digirrepetidoras especializadas. El resto de las estaciones recogen esta información balizada y la procesan para posicionar en sus mapas las nuevas estaciones o refrescar la información de las preexistentes. Cuando una estación queda inactiva, pasado cierto lapso de tiempo, desaparece de los mapas de sus corresponsales.

Cada estación individual puede convertirse en una estación digirrepetidora que dé acceso al sistema a otras a otras de su entorno con menor cobertura (p.e. móviles). Sin embargo el peso de esta operativa, para incorporar vastas extensiones de terreno, se encomienda a estaciones especializadas, anexas frecuentemente a otras analógicas.

La operativa habitual está en las bandas de V-UHF o superiores. Pero también hay frecuencias específicas en HF para aumentar cobertura y añadir al sistema lugares muy distantes entre sí. Se han efectuado experiencias con estaciones espaciales (caso de la MIR) y está prevista en el protocolo la utilización de satélites de radioaficionado para dar cobertura a estaciones móviles.

Otra habilidad, totalmente desarrollada y operativa, consiste en utilizar enlaces punto a punto a través de Internet. Es más, todos los programas aptos para APRS incorporan la posibilidad de conectarse a la red a través de sesiones Telnet, de forma que cualquier estación pueda disponer a la vez de uno o más puertos en radiofrecuencia y otro conectado a Internet, con sesiones simultáneas, trasvasando la información disponible todos ellos. Existen además programas especializados para servidores de este tipo. En USA está reglamentariamente contemplado, siempre que la materia del tráfico originado sea de radioaficionados para radioaficionados o servicios de emergencia civil.

Este sistema de enviar mensajes funciona en tiempo real, es decir que los mensajes llegan a su destinatario en unos 2 a 20 segundos, dependiendo del número de digipeaters por los que tenga que pasar el mensaje.

A parte de los mensajes entre estaciones también se pueden mandar anuncios o boletines generales para todas las estaciones. Los mensajes se transmiten línea a línea, siendo estas de unos 55 caracteres, por lo tanto a medida que se escriben los caracteres en el teclado se van transmitiendo contiguamente.

El programa de APRS asigna un número a cada línea para poder comprobar la confirmación del destinatario a cada una de ellas, y para poder mostrarlas en la ventana del destinatario en orden.

4.5.2. TELEMETRÍA, METEOROLOGÍA Y RADIOLOCALIZACIÓN.

La telemetría permite controlar permanentemente de forma cómoda por ejemplo, el sistema de alimentación de un determinado repetidor: la carga de sus baterías, el rendimiento de los paneles solares, si se está alimentando desde la red, la temperatura de un determinado circuito, etc. Es factible incorporar alarmas para avisar de situaciones tales como baja carga de batería o desconexión de red, apertura de puertas o detección de presencia, etc.

El protocolo para telemetría está bien desarrollado y especificado en el sistema APRS. Se debe contar con módems con interfaces especializados y conversores analógico/digitales.

Por sus características, APRS es muy apropiado para experimentación en lanzamientos de globos y sondas, puesto que la adición de un receptor GPS permite un fácil seguimiento. Dentro de este tipo, las estaciones más extendidas son sin embargo las meteorológicas.

Las estaciones de telemetría y las meteorológicas se identifican con su propio icono. Posicionadas en el mapa, se puede acceder a su información simplemente seleccionándolas con el "clic" del ratón sobre su icono. Se puede obtener en tiempo real temperatura, humedad, velocidad y dirección del viento, evaluación de máximas, mínimas, lluvia, etc.

Menos conocido, pero no por ello menos interesante, resulta otra habilidad del sistema APRS cual es la radiolocalización. Hay básicamente dos métodos: por intensidad de campo y triangulación. Existen utilidades para enlazar con ciertos medidores de campo. Resulta imprescindible que las estaciones participantes tengan perfectamente informadas las características de su sistema: potencia, ganancia, altura y direccionalidad.

4.5.3. FORMATO DE LAS TRAMAS APRS.

Como se mencionó anteriormente, a nivel de enlace (en la capa 2 del modelo OSI) APRS utiliza el protocolo AX.25, utilizando únicamente tramas UI (Unnumbered Information). Esto significa que APRS se ejecuta en modo sin conexión, por lo que las tramas AX.25 son trasmitidas sin esperar respuesta alguna, y la recepción al otro extremo del canal de comunicación no está garantizada. A un nivel superior, APRS soporta un protocolo de mensajería que permite a los usuarios enviar pequeños mensajes a estaciones en particular, y puede recibir acknowledgements de esas estaciones.

Todas las trasmisiones APRS utilizan tramas UI del protocolo AX.25:

Bandera	Destino	Origen	Repetidores (0 - 8)	Control	PID	Campo de informacion	FCS	Bandera
1 byte	7 bytes	7 bytes	0 – 56 bytes	1 byte	1 byte	1 – 256 bytes	2 bytes	1 byte

Tabla 2 Formato de las tramas APRS

- **Banderas** — La bandera al inicio y fin de la trama está definida en el protocolo AX.25, y vale 7Eh o 011111110b.
- **Destino** — Este campo contiene un distintivo APRS de destino o datos APRS. Cuando contiene datos APRS, la codificación es tal que se garantiza el cumplimiento del estándar para este campo definido en el protocolo AX.25 (6 caracteres más el SSID). Si el SSID es diferente de 0, se especifica un camino de digipiters genérico.

- **Origen** — Este campo contiene el distintivo y SSID de la estación trasmisora. En algunos casos, si el SSID no vale 0, entonces especifica un símbolo APRS.
- **Repetidores** — Se pueden incluir hasta 8 repetidores en este campo.
- **Control** — Este campo vale siempre 03h (trama UI).
- **Protocol ID (PID)** — Este campo vale siempre F0h (no hay protocolo de capa 3).
- **Campo de información** — Este campo contiene datos APRS. El primer caracter de este campo es el Identificador de Tipo de Dato que especifica la naturaleza de la información que tiene a continuación.
- **FCS** — Es el CRC definido en el protocolo AX.25 para chequear la integridad de las tramas recibidas.

El campo de destino puede contener 6 tipos de información APRS:

- Una dirección genérica APRS.
- Una dirección genérica APRS con símbolo.
- Una versión de software APRS.
- Datos codificados en formato Mic-E.
- Un MGL (Maidenhead Grid Locator), obsoleto.
- Una dirección ALTNET.

En este proyecto se utilizó la dirección genérica BEACON, ya que es la más indicada para enviar tramas genéricas sin importar quién las reciba. Por lo tanto, en todas las tramas que se envíen, el campo de dirección de destino va a contener el distintivo “BEACON”, codificado según el protocolo AX.25 para este campo.[3]

4.5.3.1. DATOS APRS EN EL CAMPO DE INFORMACIÓN DE LAS TRAMAS AX.25

En general, el campo de información de la trama AX.25 puede contener alguno o todos los siguientes tipos de información:

- Identificador de Tipo de Datos APRS
- Datos APRS

- Datos de extensión APRS
- Comentario

Identificador	Datos APRS	Datos de extensión	Comentario
1 byte	n bytes	7 bytes	n bytes

Tabla 3 Campo de información

Todos los paquetes APRS contienen un identificador. Es el que determina el formato de los restantes datos de la trama. En este proyecto se utilizaron los identificadores ":" (mensaje) y "/" (posición con timestamp), pero hay definidos más de 30 en el protocolo APRS.

Existen 10 tipos principales de datos APRS:

- Posición
- Dirección
- Objetos e ítems
- Clima
- Telemetría
- Mensajes, boletines y anuncios
- Consultas
- Respuestas
- Estado
- Otros

De éstos, sólo se utilizaron datos del tipo posición y mensajes para este proyecto.

En general, todos los paquetes APRS pueden contener un comentario en texto plano en el campo de información, inmediatamente después de los datos APRS. No existe un separador entre datos y comentario, salvo que el tipo de dato lo especifique. En el comentario puede ir cualquier caracter imprimible excepto "|" y "~", que están reservados para otros usos. El largo máximo del comentario depende del tipo de dato

que se está enviando. Este largo está especificado para cada tipo de dato en el protocolo APRS.

En algunos casos, el campo de comentario puede contener más datos APRS, por ejemplo, en los mensajes de posición, la altura puede ir dentro del comentario, tal como se utilizó en el armado de las tramas para este proyecto. [3]

4.5.3.2. FORMATOS DE TIEMPO Y POSICIÓN

El tiempo puede ser representado de tres formas diferentes en APRS:

- Día / Horas / Minutos
- Horas / Minutos / Segundos
- Mes / Día / Horas / Minutos

Dado que la hora que se obtiene de las tramas GPGGA que envía el GPS se encuentran en el segundo formato, todos los paquetes APRS que se arman en el programa principal tienen este formato. Consiste en un campo de 6 bytes con la hora terminados con la letra “h” para indicar que los datos se encuentran en ese formato.

La latitud se expresa como un campo fijo de 8 bytes en grados y minutos decimales, hasta un máximo de precisión de centésimas de minuto seguidos de una letra N para norte o S para sur. El formato de latitud es el mismo que el de los mensajes GPGGA que envían los GPS. Por ejemplo, 4903.50N representa 40 grados, 3 minutos y 30 segundos norte.

La longitud se expresa como un campo fijo de 9 bytes en grados y minutos decimales, hasta un máximo de precisión de centésimas de minuto seguidos de una letra E para este o W para oeste. El formato de latitud es el mismo que el de los mensajes GPGGA que envían los GPS. Por ejemplo, 12001.75E representa 120 grados, 1 minuto y 45 segundos este.

Los datos de posición son una combinación de latitud y longitud, separados por un identificador de símbolo y terminados en un código de símbolo. Por ejemplo, en las tramas utilizadas en este proyecto, la posición se representa 3340.30S/05512.20WO . La barra “/” que separa latitud y longitud indica que se debe utilizar la tabla de símbolos

primaria (definida en el protocolo APRS) y la "O" del final quiere decir que de esa tabla se utiliza el símbolo de globo. Si el software que recibe estas tramas sabe decodificar esta información, puede mostrar en un mapa un símbolo de globo en la posición que corresponde.

Los datos de altitud se pueden representar en dos formas:

- Dentro del comentario
- En formato Mic-E

En caso de que la altura vaya en el comentario, se debe representar en la forma /A=aaaaaa donde "aaaaaa" es la altura en pies. Esta sentencia puede aparecer en cualquier posición dentro del comentario.[11]

4.5.3.3. REPORTES DE POSICIÓN

Los reportes de posición están contenidos en el campo de información de las tramas APRS AX.25. Las tramas de posición que se utilizan en este proyecto tienen el siguiente formato:

/	Hora	Latitud	Id. de tabla	Longitud	Símbolo	Comentario
1 byte	7 bytes	8 bytes	1 byte	9 bytes	1 byte	0 – 43 bytes

Tabla 4 Tramas de posición

Ejemplo:

/171941h3453.6987S/05609.6516WO/A=000147,Ti=21,Te=-5,H=79,P=873,UHX

Donde:

- 171941h : hora según el meridiano de Greenwich
- Latitud: 3453.6987S
- Longitud: 05609.6516W
- O indica que se trata de un globo de tipo meteorológico
- Altura (en pies): A=000147
- Temperatura interna (grados Celcius): Ti=21

- Temperatura externa (grados Celcius): Te=-5
- Humedad relativa: H=79
- Presión (en hPa): P=873

Todo lo que hay después de "…WO…" está dentro del comentario. Se puede ver que en el mismo se envía información de la altura, donde dice "/A=000147", el resto es información en texto plano que es interpretada por la persona que reciba la trama, no por el software, ya que no hay un formato definido para la misma.

También es posible enviar información en formato GPGGA directamente, lo cual es sencillo si se dispone de GPS, ya que la mayoría de los mismos envían mensajes en del protocolo NMEA. El campo de información en este caso contiene únicamente el mensaje GPGGA. Por ejemplo:
*$GPGGA,102705,5157.9762,N,00029.3256,W,1,04,2.0,75.7,M,47.6,M,,*62.* [3]

4.5.4. MENSAJES APRS

El otro tipo de datos que trasmite el GloboSat es el de mensajes APRS. Los mismos se utilizan para enviar texto sin ningún formato en particular para ser leídos por cualquier persona.

Un mensaje APRS es un texto cualquiera dirigido a un destinatario en particular. El destinatario se representa en un campo fijo de 9 caracteres (rellenados con espacios si sobra lugar) luego del identificador ":" y seguido de otro ":". Luego va el texto del mensaje. El mismo puede tener un máximo de 67 caracteres y puede contener cualquier caracter ASCII excepto "|", "~", o "{". Los mensajes pueden contener también un identificador de mensaje opcional al final del texto del mensaje. Consiste en una llave "{" seguida de hasta 5 caracteres alfanuméricos sin espacios que identifican el mensaje. Los mensajes que no contienen identificador no pueden ser respondidos con un acknowledge. Si tiene identificador, la estación trasmisora seguirá enviando el mensaje hasta recibir un acknowledge por parte del destinatario. Este no es el caso que compete a este proyecto, ya que el GloboSat no tiene capacidad para recibir mensajes, sólo envía información en una suerte de broadcasting.

Ejemplo:

:CV1LAI :Este es un ejemplo de un mensaje APRS sin identificador.

Por información adicional sobre APRS consultar [3] y [4].

4.6. MODULACIÓN DIGITAL

Las particularidades de las emisiones permitidas a las estaciones de radioaficionado, en lo referente a ancho de banda, a máximas potencias utilizables o a antenas han hecho que muchos radioaficionados diseñen e implementen nuevos modos de comunicación, que tratan de luchar contra algunas de estas limitaciones o los efectos de las mismas.

PSK31 es un modo para realizar contactos teclado a pantalla en tiempo real y sin protocolo a nivel de enlace punto a punto. El emisor y los receptores se sincronizan solos. Se basa en una modulación PSK a 31,25 baudios. De esta forma el ancho de banda ocupado es de 40Hz en vez de los 300-500 Hz de otros modos. Esto permite utilizar los filtros más estrechos del receptor con objeto de separarlo de otras emisiones. Estas características hacen que la calidad del enlace sea muy buena. El empleo de un alfabeto de longitud variable (Varicode) en el que los códigos correspondientes a las letras más corrientes son de tamaño más pequeño (como el código morse), nunca contienen más de un 0 seguido y son separados entre sí por dos ceros consecutivos, hacen que la resincronización en caso de error sea muy rápida. El cero se codifica como cambio de polaridad. Se consigue que velocidad real sea de 50 palabras por minuto.
La gran ventaja de las modulaciones PSK es que la potencia de todos los símbolos es la misma, por lo que se simplifica el diseño de los amplificadores y etapas receptoras (reduciendo costes), dado que la potencia de la fuente es constante.

FSK (Frequency Shift Keying) es también conocido como cambio de frecuencia de modulación de frecuencia y cambio de señalización. FSK es una señal de datos convertida en una determinada frecuencia o tono con el fin de transmitir a lo largo de hilo, cable, fibra óptica o medios inalámbricos a un punto de destino

El protocolo APRS, a nivel de capa física, utiliza el protocolo Bell202 para trasmitir datos a 1200 baudios. La modulación se hace en AFSK (Audio Frequency Shift Keying), es decir, que se utilizan tonos para codificar los unos y ceros. En el caso de Bell202, se utilizan dos tonos a 1200Hz y 2200Hz para trasmitir un 1 digital y un 0

respectivamente. El modo digital utilizado por APRS se conoce como PACKET, el cual hace una modificación al protocolo Bell202, y es que cambia de tono cuando trasmite un cero y mantiene el tono cuando trasmite un 1. Se debe aclarar que la fase del tono entre bit y bit trasmitido debe ser continua, detalle que debe ser tenido en cuenta al momento de programar. Como un 1 se representa manteniendo el tono, si se envían varios 1 seguidos, el receptor puede perder el sincronismo, lo cual está previsto por el protocolo mediante el "bit-stuffing", que consiste en insertar un cero luego de 5 unos consecutivos. El receptor sabe que luego de 5 unos seguidos viene un cero, el cual es eliminado al momento de recibir la trama. Si luego de 5 unos el bit siguiente no es cero, la trama se considera corrupta y se descarta, salvo que se esté recibiendo una bandera que indica inicio o fin de la trama, la cual se representa con la secuencia "01111110".
Por último, en PACKET se envía el bit menos significativo primero, excepto cuando se envía el CRC, donde sucede lo contrario.

Considerando la velocidad a la que se colocan los datos en el canal, control de errores, el ancho de banda requerido, luego de haber ensayado la performance del protocolo PSK31 se optó por utilizar a nivel de capa física la modulación AFSK utilizada por APRS.

5. MÓDULO PRINCIPAL

Con el objetivo de procesar la información proveniente de los dispositivos externos, se utilizó el microprocesador Atmega 2860 de la familia Atmel. El mismo es programado para leer los datos de: dos sensores de temperatura, uno colocado en el interior del cubo de poliuretano y el otro colocado en el exterior del mismo para así medir la temperatura externa a la que está sometida la plataforma; sensor de presión; GPS para obtener datos de posición y altura y placa FPGA para registrar los ensayos de radiación. Este microprocesador debe registrar información del consumo de energía que se le realiza a la batería para controlar la tasa de transferencia de información, disminuyéndola cuando la carga descienda un primer umbral, y pare la transmisión cuando llegue a un segundo. Para que el globo no se salga de una cierta región preestablecida, se activará un circuito "quema tanza" en caso de ser necesaria la liberación de la carga en vuelo.

5.1. HARDWARE

La colocación de los dispositivos periféricos, elección y evaluación de los mismos, diseño e implementación de circuito quema tanza, así como la administración de energía a la plataforma, quedó a criterio del grupo HWGloboSats.

5.2. SOFTWARE

Para realizar una mejor gestión del módulo principal, el software a implementar en el microprocesador Atmel se dividió en dos partes: 1) adquisición de datos de los dispositivos periféricos, 2) programa principal.

El grupo de hardware será el encargado de programar en el microprocesador la adquisición de los datos provenientes de los periféricos. El grupo de telemetría es el encargado de desarrollar el bucle principal y la cerca electrónica dentro de la cual podrá viajar el globo. La comunicación entre ambos grupos de trabajo se realiza mediante un acuerdo establecido en el archivo api.h incluido en el Apéndice.

5.2.1. PROGRAMA PRINCIPAL

El software embebido en el microcontrolador Atmega 2860 es el programa medular que se comunica con los diferentes dispositivos de hardware tales como el GPS, el modulador AFSK, los diversos sensores de la placa y además lleva a cabo el denominado "plan de vuelo", que rige todo comportamiento del GloboSat desde el momento de la liberación hasta que llega al suelo nuevamente.

Este documento no pretende entrar en detalle en cuanto a la implementación de las rutinas que controlan los diferentes dispositivos, sino que describe las interfaces con los mismos. Si se desea mayor detalle en dichas implementaciones, se debe consultar la documentación correspondiente al proyecto "HWGloboSats" que comenzó en Julio del 2007 y que se extendió hasta fines del 2008.

El programa principal deberá encargarse de:

1. Inicializar variables y configurar interfaces (puertos de comunicación, conversores A/D)
2. Recolectar los datos de los sensores de temperatura, presión y humedad.
3. Procesar las tramas GPGGA que envía el GPS.
4. Llevar a cabo el plan de vuelo.
5. Armar las tramas en formato APRS.
6. Enviar las tramas al modulador AFSK.

Las funciones que se encargan de enviar y recibir datos por los puertos UART, inicializar interfaces y recolectar datos de los sensores están definidas en un API (Application Programming Interface), donde los integrantes de los proyectos "TeleGloboSats" y "HWGloboSats" acordaron definir las interfaces para manejar dichas funciones.

5.2.1.1. INICIALIZACIÓN

La función HW_init() es la encargada de configurar los puertos de entrada/salida del microcontrolador, los puertos UART y la interfaz TWI para la comunicación con las memorias externas.

Los puertos UART 2 y 3 se utilizan para la comunicación con el GPS y el modulador AFSK respectivamente. Para configurarlos, la función HW_init() llama a la función UART_init(puerto, bps, tx, rx) que recibe como parámetros el número de puerto, la velocidad del mismo y dos parámetros más que habilitan o deshabilitan la trasmisión y/o recepción del puerto. Por ejemplo, para configurar el puerto 3 (comunicación con el modulador) se ejecuta UART_init(3,4800,1,0) , lo que setea la velocidad del puerto 3 a 4800 bps (la misma que debe tener configurado el modulador) y habilita sólo la trasmisión por ese puerto dado que el modulador no envía información hacia el microcontrolador.

Las configuraciones de los puertos de entrada/salida y la interfaz TWI para la comunicación con las memorias son funciones internas a la función HW_init() y no se van a detallar en este documento, pero si se desea más información se puede consultar la documentación correspondiente al proyecto “HWGloboSats”.

5.2.1.2. SENSORES

En el API se definen las siguientes funciones para manejar los sensores:

- *float get_temperatura (enum sensor)* – Recibe como parámetro una variable del tipo enum sensor que puede tomar los valores sensor_temp_int y sensor_temp_ext y devuelve un valor de tipo float con la temperatura interna o externa respectivamente en grados Celsius.
- *float get_presion(void)* – No recibe ningún parámetro, devuelve un valor de tipo float con la presión en hPa.
- *float get_humedad(void)* - No recibe ningún parámetro, devuelve un valor de tipo float con el porcentaje de humedad.

- *float get_Vss(void)* - No recibe ningún parámetro, devuelve un valor de tipo float con el voltaje de la fuente en [V].

5.2.1.3. GPS

Una vez inicializado el puerto UART al que se encuentra conectado al GPS, se puede utilizar la función *int get_gps_frame(char *buffer , int max_size)* para recibir las tramas GPGGA que envía el GPS. A dicha función hay que pasarle como parámetros el lugar de memoria donde se va a guardar el mensaje y la cantidad máxima de bytes que puede escribir a partir de esa dirección. Devuelve un entero con la cantidad de bytes recibidos del GPS o 0 si no se recibió ningún dato.

Suponiendo que se obtuvo una trama GPGGA completa, se utiliza la función *int parse_gpgga(char *dato_gps, char *hora, char *latitud, char *card_lat, char *longitud,char *card_lon,char *altura)* definida en el archivo *funciones.h* y que se encarga de extraer de la trama sólo los valores que interesan, a seber: timestamp de la trama (hora a la que se trasmitió), latitud, longitud y altura.

5.2.1.4. PLAN DE VUELO

En el plan de vuelo se establecen las especificaciones que rigen el comportamiento del GloboSat desde su liberación hasta la recuperación. Esto quiere decir que el programa principal no es un software “tonto”, sino que debe tomar decisiones en cada instante dependiendo del estado en que se encuentre. El estado queda definido por variables tales como la altura, posición geográfica, altura, presión, temperatura y carga de la batería, pero podrían ser más según lo que se pretenda.

El primer requerimiento para el plan de vuelo es evitar que el GloboSat caiga fuera del territorio nacional o en el Río de la Plata o el Océano Atlántico. Para ello se implementó un mecanismo bautizado como “cerca electrónica”, el cual se encarga de liberar la carga útil del globo si el GloboSat se va por fuera de los límites establecidos.

El siguiente diagrama de bloques representa la porción del plan de vuelo que implementa la cerca electrónica:

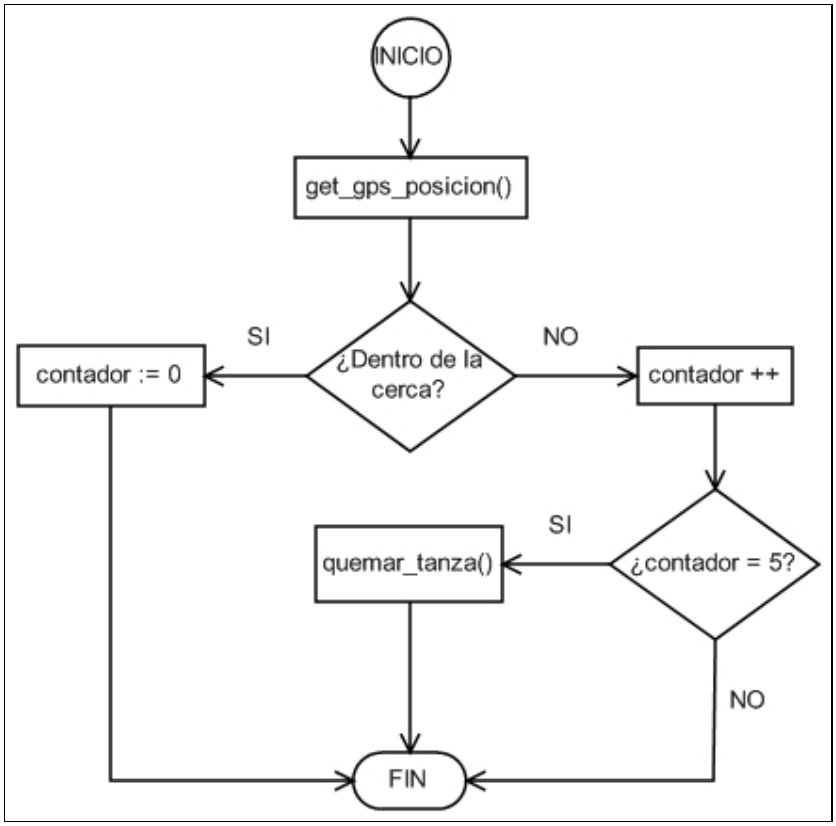

Figura 3 Diagrama de bloques del plan de vuelo

Se desprende de este diagrama que para que esta implementación cumpla su fin es imprescindible que el GPS reporte siempre la posición correcta del GloboSat. Si el mismo falla y no puede reportar la posición, no se pueden realizar los cálculos necesarios para determinar si el GloboSat está dentro o fuera de la cerca electrónica. De todas formas se debe tomar una decisión, por lo que se resolvió que si no se conoce la posición, no se libere la carga. Esto se debe a que no se puede saber de antemano si el GPS va a fallar o no en plena subida del globo y, en caso de fallar, la carga igualmente va a caer cuando el globo explote por expansión. Para la recuperación en ese caso donde no se conoce la posición, lo único que se puede hacer es esperar a que alguien encuentre la carga y se comunique con alguno de los responsables, por lo que en la caja del GloboSat siempre se escribe algún número telefónico de contacto y también en un papel dentro de la misma.

Suponiendo que el GPS no falla y se puede obtener la posición, con ese dato el programa determina si el GloboSat se encuentra dentro del polígono predefinido. De ser así, no se realiza ninguna acción en particular. Cuando el GPS reporta una posición que resulta externa al polígono, se incrementa un contador. Cuando el contador llega a 4, es decir, 4 reportes consecutivos fuera del polígono (el contador vuelve a 0 si se está dentro del polígono), el programa llama a una función que se encarga de calentar un alambre que está enrollado a la tanza que une la carga con el globo, lo que provoca que la misma se rompa y se libere la carga, comenzando el descenso.

Otro requerimiento del plan de vuelo es que el GloboSat reporte la posición con mayor frecuencia cuando está cerca de tocar el suelo. Se estableció que una altura de 1000m dispara este evento. Al igual que en el caso de la cerca electrónica, el dato de la altura se obtiene únicamente del GPS, por lo que también es necesario el correcto funcionamiento del mismo para que esto pueda ser llevado a cabo. En ese momento se supone que el GloboSat estuvo en el aire unas 3 horas, por lo que seguramente no le quede mucha carga a la batería. Como se aumentó la frecuencia de envío de las tramas, hay un mayor consumo de energía por unidad de tiempo por parte del trasmisor. Para contrarrestar esta consecuencia, se decidió que en el momento que se aumenta la frecuencia de trasmisión también se apaguen los dispositivos secundarios y quede únicamente encendido el GPS para reportar la posición.

Lo que hay que tener en cuenta en este evento es que no alcanza con establecer una altura mínima para considerar que la carga está por tocar el suelo, ya que en el ascenso, hasta no superar dicho límite, se sigue estando por debajo de la misma. Hay que determinar de alguna manera en qué momento el GloboSat superó ese límite para que cuando se encuentre por debajo del mismo nuevamente, se apaguen los dispositivos y se comience a trasmitir la posición con mayor frecuencia. Lo más sencillo es encender una bandera en el programa cuando se superan los 1000 m de altitud, para indicar que el GloboSat “subió”, y si la bandera está encendida y el GloboSat está por debajo de los 1000 m, entonces se dispara el evento. El error en esta solución es que no se tiene en cuenta que por imprecisiones en el GPS al determinar la altura (la altura en los GPS tiene mayor incertidumbre que la posición) o que debido a ráfagas de vientos el GloboSat pueda moverse de forma brusca y descender repentinamente antes de ascender nuevamente, al superar los 1000 m se encendería la bandera, pero debido a alguna de las

causas anteriores el GloboSat descendería unos pocos metros, y entonces el programa consideraría de forma errónea que la carga está por tocar el suelo y desactivaría los dispositivos secundarios. Para evitar esto, lo único que se resolvió fue agregar cierta histéresis a este mecanismo, por lo que la bandera se enciende no a los 1000 m, sino a los 2000 m. De esta manera, cambios bruscos en la altura a los 2000 m no activaría este evento.

Otro requerimiento del plan de vuelo es monitorear la carga en la batería y modificar la frecuencia de trasmisión de acuerdo a la misma. Se definen 3 niveles para la carga en la batería: alto, medio y bajo. Si el nivel es alto, entonces se trasmite a cierta frecuencia, por ejemplo, cada 5 segundos. Si el nivel es medio, entonces se trasmite a menos frecuencia para que la batería dure más tiempo. Si el nivel de batería es bajo, entonces se apagan los dispositivos secundarios y se trasmite únicamente los datos de posición. Este requerimiento tiene conflictos con el anterior, que aumentaba la frecuencia de trasmisión cuando se está por debajo de los 1000 m. Al ser más crítico conocer la posición cuando la carga está por tocar el suelo, se le da mayor prioridad a ese evento, por lo que sin importar la carga en la batería la frecuencia de trasmisión aumenta por debajo de los 1000m siempre. De todas formas, este requerimiento no se implementó para GloboSat01, ya que las rutinas que determinan la carga restante en la batería no estaban implementadas.

5.2.1.5. ARMADO DE LAS TRAMAS

Para explicar el proceso de armado de una trama, tómese como ejemplo la porción de código encargada de armar una trama cuando el GPS no tiene suficientes satélites utilizables:

```
1  frame_buffer[0]='\0';
2  strcatmaxn(frame_buffer,FRAME_INIT_BYTE, FRAME_BUFFER_SIZE);
3  strcatmaxn(frame_buffer,ENCABEZADO_AX25, FRAME_BUFFER_SIZE);
4  strcatmaxn(frame_buffer,MENS_NOSAT, FRAME_BUFFER_SIZE);
5  strcatmaxn(frame_buffer,"\nTi=", FRAME_BUFFER_SIZE);
6  strcatmaxn(frame_buffer,temperatura_int,FRAME_BUFFER_SIZE);
```

```
7  strcatmaxn(frame_buffer,"\nTe=", FRAME_BUFFER_SIZE);
8  strcatmaxn(frame_buffer,temperatura_ext, FRAME_BUFFER_SIZE);
9  strcatmaxn(frame_buffer,"\nHum=", FRAME_BUFFER_SIZE);
10 strcatmaxn(frame_buffer,humedad, FRAME_BUFFER_SIZE);
11 strcatmaxn(frame_buffer,"\nP=", FRAME_BUFFER_SIZE);
12 strcatmaxn(frame_buffer,presion, FRAME_BUFFER_SIZE);
13 if (strcatmaxn(frame_buffer,FRAME_END_BYTE, FRAME_BUFFER_SIZE)
   == -1){
14 frame_buffer[FRAME_BUFFER_SIZE - 1]=FRAME_END_CH;}
```

En la primera línea se asigna el valor NULL al primer elemento del buffer. Dado que en el lenguaje ANSI C las cadenas de caracteres (el buffer de la trama en este caso) finalizan siempre con un caracter NULL, el asignarle este valor al primer elemento es una manera de limpiar el buffer. Si bien los restantes elementos no son nulos, al ser el primero nulo, el buffer queda "vacío".

En las líneas de la 2 a la 12 se puede ver cómo se va armando la trama utilizando la función strcatmaxn definida en *funciones.h* . Esta función recibe como parámetros el buffer donde se van a agregar los datos, el dato que se quiere agregar, que debe ser una cadena de caracteres, y el tamaño del buffer. Luego, si queda suficiente espacio en el buffer, la función concatena el dato nuevo con lo que había previamente en el buffer. Si no hay suficiente espacio, devuelve el entero "0". De aquí se deduce que tiene que haber una declaración previa del buffer, especificando un tamaño fijo para el mismo. En este caso, el tamaño del buffer es de 98 bytes, dado que el buffer de recepción del modulador es de 96 bytes, a los que se le agregan los bytes de inicio y fin de trama que no son guardados en el buffer del modulador. Al comienzo del programa principal, la sentencia char frame_buffer[FRAME_BUFFER_SIZE]; inicializa el buffer.

La segunda línea de código, al igual que la primera, debe ser siempre la misma. Lo que se hace es agregar como primer dato en el buffer el caracter de inicio de trama, según como está definido en el modulador. De esta manera, el modulador sabe que de ahí en adelante se van a enviar datos para su posterior trasmisión, tal como está explicado en el capítulo sobre el modulador de esta documentación. Como los datos que se envían son solamente caracteres ASCII, se tomó como byte de inicio de trama el

valor 02h, ya que al no representar ninguna letra, número o signo de la tabla ASCII, ese valor no va a aparecer más en toda la trama (no debería si la trama está bien armada, es responsabilidad del programador).

La sentencia en la tercera línea puede cambiar, pero en este proyecto es siempre la misma. Lo que hace es agregar el encabezado AX.25 como se describe en el capítulo sobre dicho protocolo de esta documentación.

Existe un punto importante a tener en cuenta cuando se genera el encabezado AX.25. Como se dijo anteriormente, la trama se arma con un byte de inicio y otro de fin cuyos valores no se pueden repetir dentro de la trama. Para ello se eligieron dos números que no representan ningún caracter ASCII imprimible, el 02h y 04h. El problema surge de que en el encabezado AX.25 no se utilizan caracteres ASCII, por lo que se debe estar seguro de que no aparezcan los valores 02h o 04h dentro del encabezado. En este caso, el encabezado se define en las primeras líneas del programa principal:

#define ENCABEZADO_AX25
"\x84\x8A\x82\x86\x9E\x9C\x60\x86\xB0\x60\x86\x8C\x92\x61\x03\xF0"

Como se puede ver, el 02h y el 04h no aparecen en el encabezado. Para recuperar las direcciones de origen y destino del encabezado, rotamos cada byte un lugar a la derecha tal como está explicado en el capítulo sobre el protocolo AX.25. El encabezado decodificado quedaría entonces: BEACON'60h'CX0CFI'61h''03h''F0h'. Los dos últimos bytes, el 03h y F0h no cambian mientras se envíen solamente tramas UI. Como los mensajes van a estar en formato APRS, sólo se envían tramas UI, según el protocolo APRS.

Desde la línea 4 a la 12 simplemente se arma el mensaje que contiene la información útil de la trama. En el ejemplo, como se supone que el GPS no está viendo ningún satélite utilizable, se envía un mensaje definido que informa este hecho, por ejemplo, "NO SAT". En planvuelo.h se define este mensaje:

#define MENS_NOSAT ":CV1LAI : NO SAT"

Como se puede observar, el mensaje comienza con el indicativo ":CV1LAI :". Esto corresponde a un mensaje del tipo "APRS message", tal como se explica en el capítulo sobre el protocolo APRS en esta documentación. Luego del indicativo, se puede poner la información que se desee en cualquier formato. Entonces, además de reportar que el GPS no ve ningún satélite utilizable, se envía información de los sensores, ya que al no tener datos de posición y altura, con las temperaturas, humedad y presión se puede estimar a qué altura se encuentra el globo.

Por último, resta agregar al final de la trama el byte de fin de trama para que el modulador sepa donde finaliza la misma y comience su trasmisión. Puede suceder que luego de armar la trama, no quede espacio en el buffer para agregar el último byte. Dado que es imprescindible que se inserte este byte al final, si no hay lugar, se debe sobreescribir el último lugar con el byte de fin de trama. De esta manera se pierde el último carácter de la información útil, pero es preferible eso a perder la trama completa. Esto es lo que se hace en las dos últimas líneas de código, que deben ser siempre las mismas. En la primera se intenta agregar al final del buffer el caracter de fin de trama, si no hay más espacio, en la segunda línea se sobreescribe el último valor con dicho caracter.

En este ejemplo se armó una trama del tipo "APRS message". El mismo formato es utilizado para mandar mensajes de error cuando la comunicación serie entre el GPS y el microcontrolador falla, para enviar la baliza cada 5 mensajes y es el mismo formato de mensaje que utiliza el modulador para enviar mensajes de error cuando falla la comunicación entre el mictrocontrolador y el modulador mismo. Además de este formato, también se utilizan los mensajes del tipo "Position Report Format with Timestamp" cuando hay información completa del GPS. Un ejemplo de una trama en este formato sería el siguiente:

/171941h3453.6987S/05609.6516WO/A=147,Ti=21,Te=-5,H=85,P=1040,UHX

Este formato está explicado en detalle en el capítulo sobre APRS de esta documentación. La información que lleva esta trama es:

- **Timestamp** – hora GMT a la que se envió la trama, obtenida desde el GPS
- **Latitud**
- **Longitud**
- **Altura** – Luego de la longitud, dentro del comentario de la trama como se especifica en el protocolo APRS, debe estar en pies, la altura en metros se obtiene del GPS y luego se multiplica por 3.28 para hacer la conversión.
- **Datos de los sensores:** temperatura interna y externa en grados Celsius, humedad en porcentaje y presión en hPa
- **Estado del GloboSat:** se representa con 2 letras. La primera indica si el globo sube ("U"), baja ("D"), se mantiene a altura constante ("=") o está llegando al suelo ("G"), tal como se explica en el plan de vuelo. La segunda letra indica si el globo está dentro ("O") o fuera ("X") de la cerca electrónica.

5.2.1.6. ENVÍO DE TRAMAS AL MODULADOR

Una vez que la trama fue armada, sólo resta enviarla al modulador para generar la señal en AFSK que será trasmitida en FM en la banda de 2m. Para ello se utiliza la función *void send_paq(char *buffer)* definida en el API, que sólo toma como parámetro un puntero a la dirección de memoria donde se encuentra guardada la trama ya armada. Internamente, la función pone en el puerto UART 3 cada byte para ser trasmitido al modulador a una velocidad de 4800 bps como se describe el la sección sobre inicialización de interfaces.

El programa principal no hace ningún chequeo para saber si la trama fue trasmitida correctamente, simplemente la envía al modulador y asume que éste se encargará de modular la señal correspondiente y enviarla al trasmisor FM.

6. MODULACIÓN

En esta sección se detallará el funcionamiento de los principales módulos (tanto de hardware como de software) responsables de la modulación, centrándose en la implementación de los protocolos de capa física (Bell 202) y capa de enlace (AX.25).

6.1. HARDWARE

En la Figura 4 se puede apreciar un diagrama de bloques del modulador.

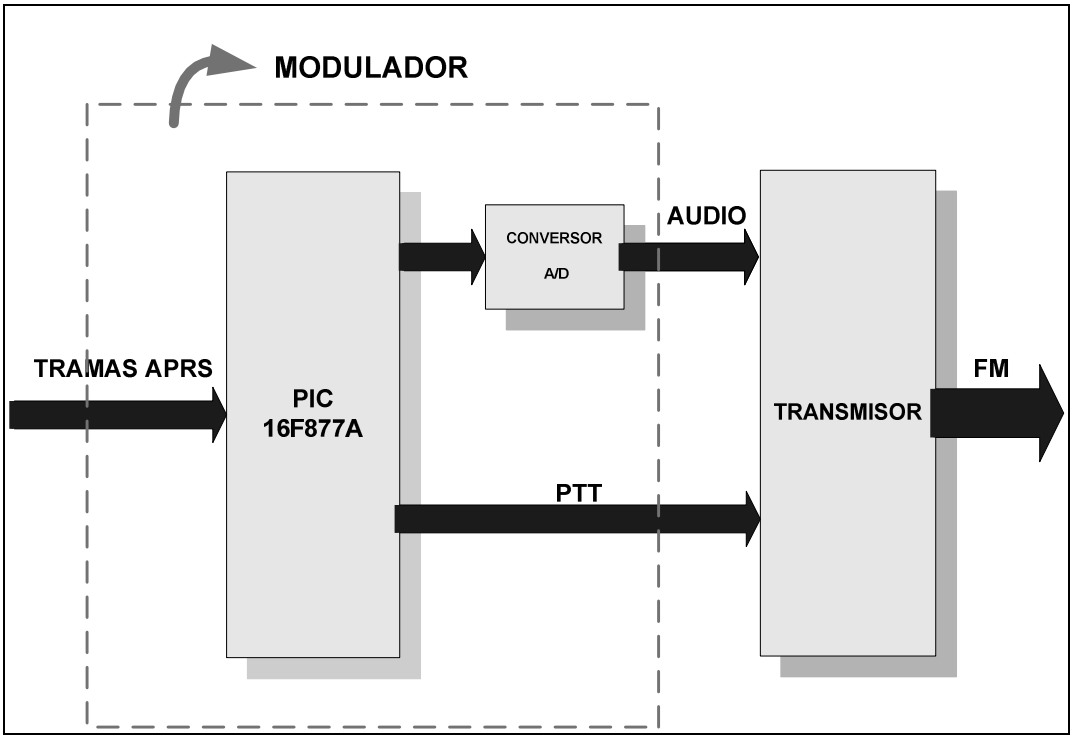

Figura 4 Diagrama de bloques del modulador

El modulador corresponde al área encerrada por la línea punteada, el mismo está compuesto básicamente por un microcontrolador encargado de realizar la modulación y un conversor digital analógico para convertir la señal digital en una señal de audio que será enviada al transmisor.

El PTT (Push to talk) es un método para hablar en líneas half-duplex de comunicación. Cuando la señal PTT proveniente del PIC se activa el transmisor se energiza, transmitiendo así la señal de audio. La señal PTT se activa unos instantes antes que se comience la transmisión, de esta forma se permite el ahorro de batería, pues el transmisor se enciende solamente cuando hay datos disponibles.

Los datos de telemetría son recibidos a través del puerto serie del microcontrolador, el mismo los modula según el protocolo Bell 202 AFSK (Audio Frequency Shift Keying) a 1200 bps, calcula el CRC de la trama y agrega las banderas de inicio y fin de la misma de acuerdo al protocolo HDLC (High-Level Data Link Control). El resultado es una señal en el rango de frecuencias audibles que por último es enviada al transmisor FM.

En una primera instancia se desarrolló la plataforma detallada en el Anexo 1, que fuera utilizada en la liberación del primer GloboSat. La misma estuvo compuesta dos sensores de temperatura, GPS, microcontrolador y transmisor. Con la misma se pudo evaluar el desempeño de la modulación de datos y armado de trama, resultando satisfactoria. El software encargado de la transmisión será reutilizado en posteriores lanzamientos.

6.1.1. CONVERSOR DIGITAL ANALÓGICO

Una vez armadas las tramas según el protocolo APRS, éstas son enviadas al exterior a través de los 8 bits del puerto B del PIC. Esta salida es digital, por lo tanto es necesaria una etapa de conversión antes de enviar la señal al transmisor. Dicha etapa consiste en un conversor digital analógico, en este caso el conversor elegido es el de tipo escalera R/2R. Para la implementación del mismo se utilizaron resistencias de $10K\Omega$ y $20K\Omega$ al 1%. La elección de dicho conversor se basa principalmente en la rapidez de conversión y en su simple arquitectura. A continuación se hará una breve reseña del mismo.

Sea el conversor de la Figura 5:

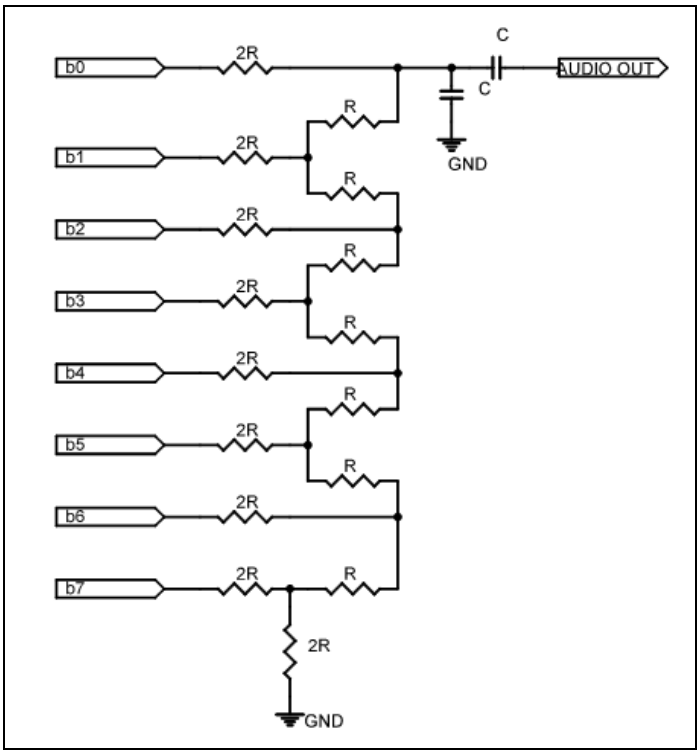

Figura 5 Conversor digital - analógico

Los extremos de las resistencias 2R están conectados al puerto paralelo (Puerto B) del PIC, y estarán en nivel bajo cuando el bit correspondiente esté apagado (0), o conectados a un voltaje de referencia (Vref = 5V) cuando el bit correspondiente esté en nivel alto (1).

La corriente de salida será I, para calcular el valor de la misma se estudiará la contribución de cada bit del circuito. Supóngase que todos los interruptores están a cero salvo el que actúa por la acción del MSB del dato de entrada, que se supondrá está a 1. En ese caso los extremos de las resistencias 2R anteriores estarán todas a tierra. Si se calcula el equivalente Thevenin en el nodo A se tendrá el siguiente circuito, teniendo en cuenta que todo el circuito anterior se reduce a R.

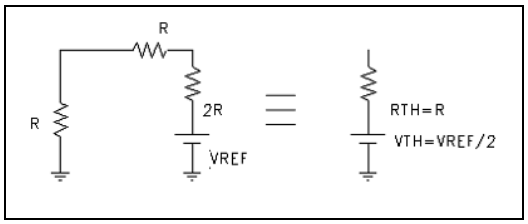

Figura 6 Equivalente Thévenin

Este es el modelo equivalente para el MSB. Supóngase ahora que todos los interruptores están a cero, incluido el MSB, salvo el segundo más significativo. Si se calcula nuevamente su equivalente Thevenin en el nodo A se tiene que:

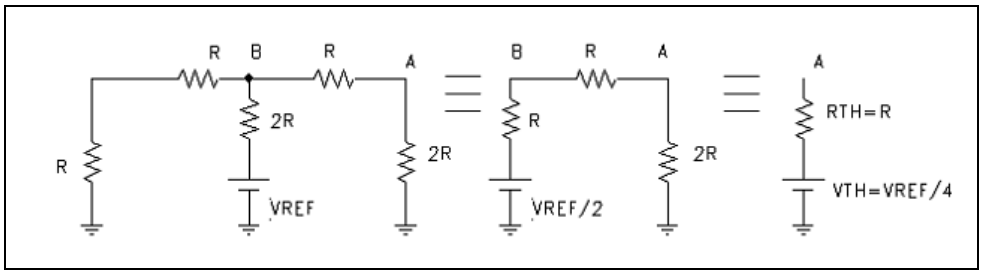

Figura 7 Equivalente Thévenin

Primero se calcula el equivalente en B de las dos ramas de la izquierda y con él se obtiene el equivalente total en A.

Si se repite el proceso con los demás interruptores del circuito se puede ir obteniendo la contribución de cada uno de ellos: obsérvese que la resistencia Thevenin es en todos los casos Rth=R y la tensión Thevenin Vth = Vref /8, Vref/16, Vref /32, etc. Con estos datos se calcula la intensidad total como suma de las intensidades que aporta cada rama:

$$I = \frac{Vref}{R}\left(\frac{b_{n-1}}{2} + \frac{b_{n-2}}{4} + ... + \frac{b_0}{2^n}\right), \text{ con } n = 8$$

Fuente: [7]

De esta forma se puede estimar que la corriente de salida es como máximo 0.5mA.

A la salida del conversor se agregaron dos capacitores de desacople de valor $C = 0.01\mu F$ para eliminar posibles componentes de continua.

6.1.2. CIRCUITO FINAL

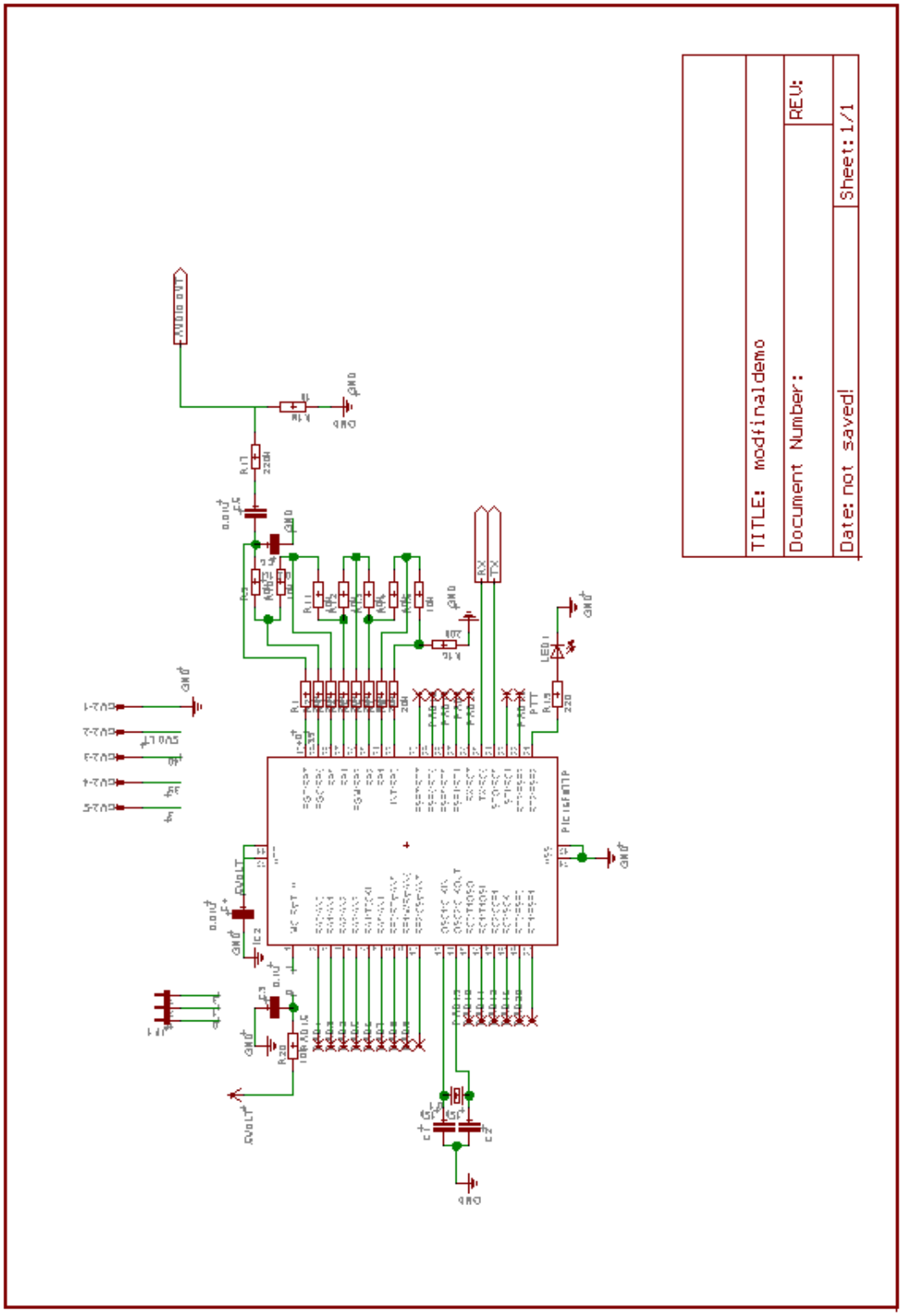

Figura 8 Esquemático del modulador

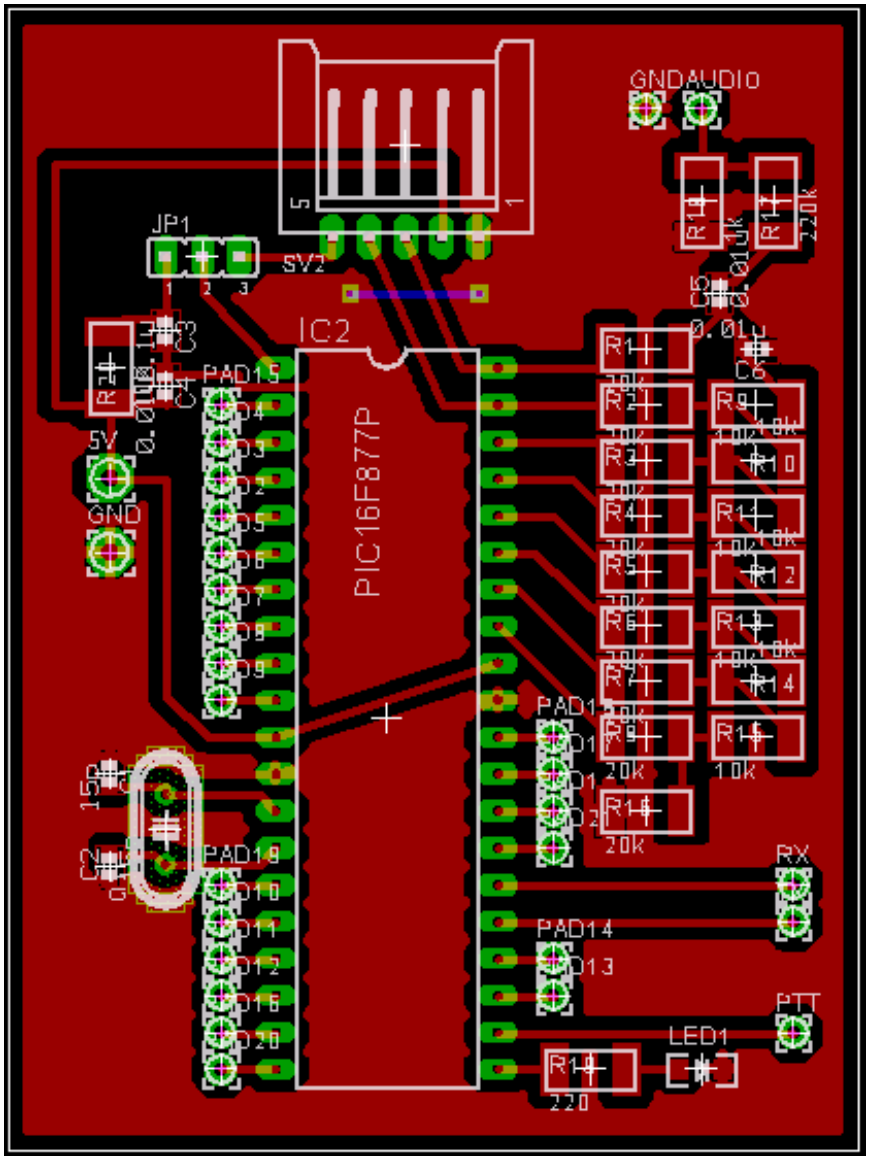

Figura 9 Layout del modulador

El layout y el esquemático del modulador se encuentran en el CD adjunto bajo el nombre Modulador.

6.2. SOFTWARE

6.2.1. PROTOCOLO BELL 202 AFSK 1200 bps

AX.25 utiliza el protocolo Bell 202 AFSK para modular la señal a nivel de capa física. El mismo consiste en modular la señal con 2 tonos audibles, que para una

velocidad de 1200 bps son 1200 Hz para marca y 2200 Hz para espacio, a diferencia de una modulación FSK estándar donde la portadora misma es modulada entre las dos frecuencias. La ventaja de utilizar AFSK es que cualquier receptor de radio diseñado para voz puede ser utilizado para recibir señales moduladas en AFSK. Un detalle a tener en cuenta es que la fase se la señal modulada debe ser continua, no puede haber saltos en la fase cuando se cambia de tono.

AX.25 no utiliza este protocolo en su estándar, en realidad, hace una modificación al utilizar una codificación NRZI (Non Return to Zero Inverted), que consiste en cambiar de tono cuando se recibe un "0" y mantenerlo cuando se recibe un "1".

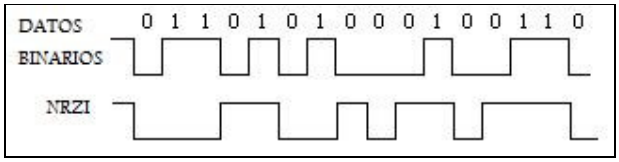

Figura 10 Forma de onda de codificación NRZI

Con esta codificación, una secuencia de varios "1" seguidos implicaría que la señal no cambie de frecuencia en ese lapso, por lo tanto, para que el receptor no pierda sincronismo con el emisor, el protocolo recurre al "bit stuffing", que consiste en intercalar un "0", es decir, un cambio de tono, cada cinco "1" seguidos, inmediatamente después del quinto "1". Vale hacer una aclaración en este punto: en la especificación oficial del protocolo AX.25, que se puede obtener de [8] o del CD que viene junto con esta documentación, en el párrafo donde se menciona el uso de bit stuffing, dice que el "0" hay que intercalarlo después del primer "1". Eso es erróneo, el "0" debe ir después del quinto "1" únicamente, de lo contrario, la señal no va a poder ser decodificada correctamente por el receptor.

Cuando el receptor recibe la señal, al ver cinco "1" seguidos sabe que inmediatamente después viene un "0". De no ser así, descarta el paquete inválido. La elección de cinco "1" seguidos se debe a que en la capa 2 del modelo OSI (Data Link Layer), AX.25 utiliza el protocolo HDLC, que como se verá más adelante, establece como bandera de inicio o fin de trama el byte "01111110". Para las banderas no se debe aplicar el método de bit stuffing, de esta forma, el único momento en el que el receptor recibe seis "1" seguidos es en las banderas de inicio y fin de trama.

Por último, hay que mencionar que el orden en que se envían los bits es el menos significativo primero para cada byte. Hay que aclarar que en este punto hay otra diferencia con la especificación oficial, que menciona que al enviar los 2 bytes del CRC al final de la trama, se debe enviar el bit más significativo primero. Esto no es así, los bytes del CRC se envían como cualquier otro byte de la trama.

6.2.2. PROTOCOLO HDLC

El protocolo AX.25 utiliza a nivel de capa 2 del modelo OSI (Data Link Layer) el protocolo HDLC (High-Level Data Link Control). En el circuito modulador, lo único que está implementado de este punto es el agregado de las banderas de inicio y fin de trama y el cálculo del CRC de la trama.

Como se mencionó en el párrafo anterior, el protocolo HDLC utiliza un carácter especial para delimitar las tramas (banderas), en binario el número 01111110. Antes de enviar la trama que recibió a través del puerto serie, el modulador envía una secuencia de varias banderas a las cuales no se les aplica bit stuffing. luego sí envía la trama, a la cual le calcula el CRC a medida que la va enviando, luego el CRC calculado y por último envía algunas banderas más para indicar el fin de la trama.
El detalle del encabezado HDLC no será expuesto en esta sección dado que no es implementado por el modulador, sino que ya es parte de la trama que recibe por el puerto serie.

Lo que sí está implementado en el modulador es el cálculo del CRC (Código de Redundancia Cíclica), llamado FCS (Frame Check Sequence) en el protocolo HDLC. Otra vez hay que hacer una aclaración en este punto con respecto a la documentación oficial del protocolo AX.25, y es que en la misma no se menciona cómo calcular el CRC, sólo se limita a mencionar que hay que utilizar el algoritmo CRC-CCITT. Dicho algoritmo tiene diversas variantes, cada una produciendo un resultado diferente al calcular el CRC. Luego de varios intentos fallidos de encontrar el algoritmo correcto, se llegó al correcto gracias a un comentario en [9], donde menciona el seudo-código para calcularlo bit a bit. El mismo dice así: “Comenzar con los 16 bits del FCS seteados en

0xFFFF. Por cada bit enviado, rotar a la derecha el FCS un bit. Si el bit que salió no es igual al que se está enviando, hacer un XOR entre el FCS y el número 0x8408, si son iguales, no se hace nada. Luego del último bit, tomar el complemento a uno del FCS y enviar el byte bajo primero". Si se implementa este código, los dos bytes del FCS se envían con el bit menos significativo primero, que fue lo que se hizo en el firmware de este modulador.

6.2.3. FUNCIONAMIENTO

El firmware del modulador fue escrito en su totalidad en lenguaje assembler, por lo que se recomienda estudiar el set de instrucciones del circuito integrado que se utilizó, el PIC16F877A, cuya hoja de datos se encuentra en el CD de la documentación o se puede descargar del sitio web del fabricante, [10]
A continuación se muestra un diagrama de flujo del bucle principal:

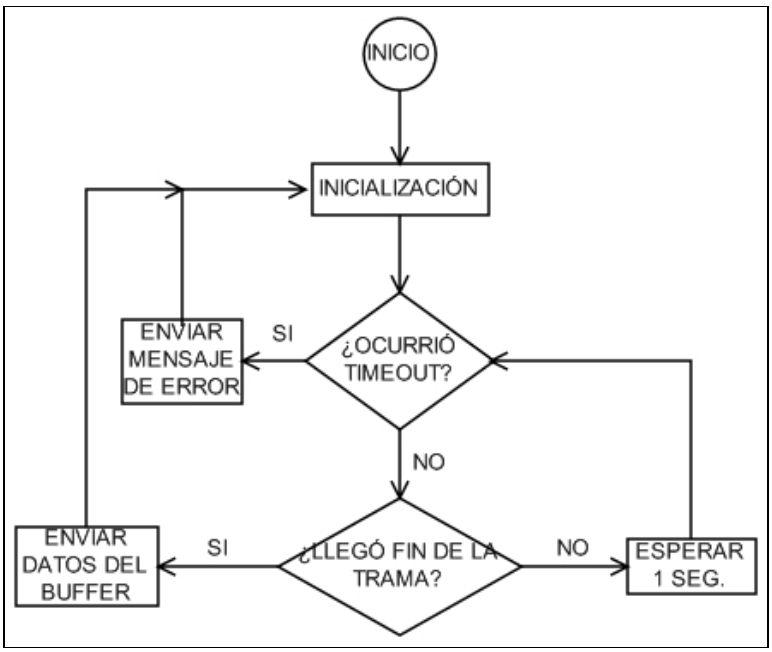

Figura 11 Diagrama de flujo del bucle principal del modulador

El bloque de inicialización realiza las siguientes funciones:

- Limpiar el buffer de entrada
- Configurar los puertos de entrada/salida

- Inicializar variables
- Configurar el puerto UART para comenzar a recibir datos

Una vez finalizadas estas tareas, el dispositivo está listo para recibir datos a través del puerto serie. El programa principal primero verifica que no haya ocurrido un timeout, es decir, que no haya pasado más de cierto tiempo (configurable) sin haber recibido una trama válida a través del puerto serie. De ser así, envía un mensaje de error predefinido y vuelve a comenzar. Mientras no se haya superado el tiempo de espera, verifica si se recibió el byte que indica el fin de la trama. Si no se recibió aun, hay una espera de 1 segundo y luego vuelve al punto donde verifica si ha ocurrido un timeout. Si el byte de fin de trama se recibió dentro del tiempo establecido, se procesan los datos guardados en el buffer para su posterior trasmisión.

A continuación se muestra el diagrama de flujo de la rutina de interrupión cuando se recibe un dato por el puerto serie:

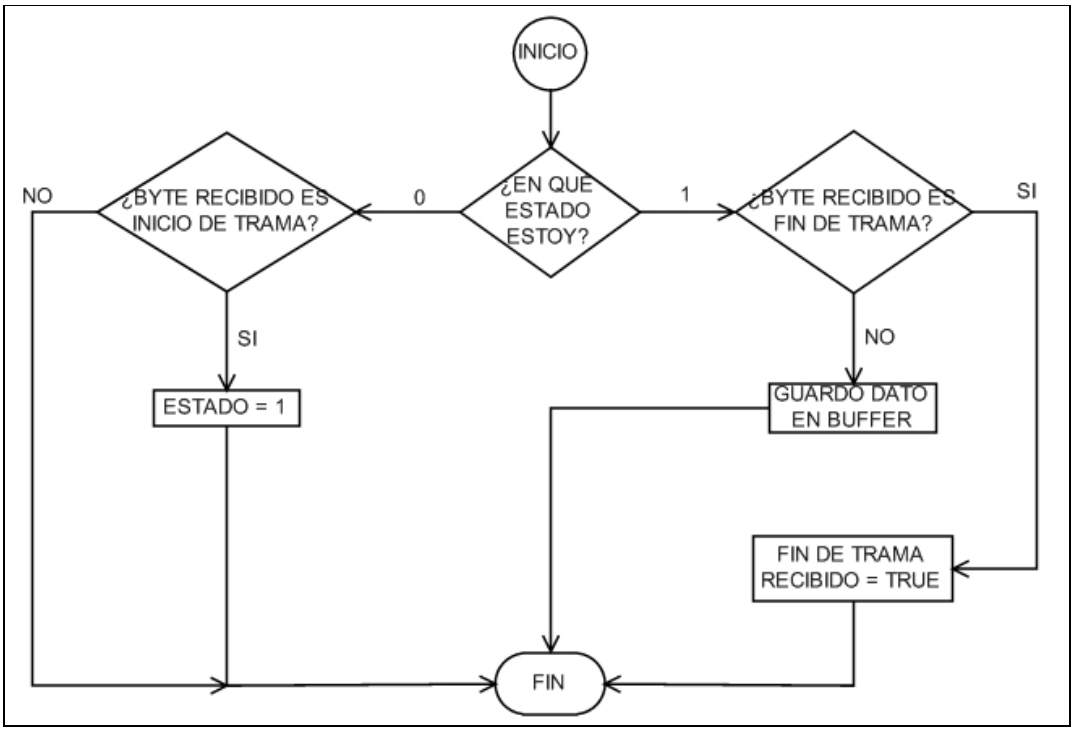

Figura 12 Diagrama de flujo de rutina de interrupción UART

Cada vez que un dato llega al buffer de la UART se genera una interrupción que es atendida por la rutina que se describe en el diagrama anterior.

Lo primero que hace la rutina es verificar en qué estado se está actualmente. Estos estados son solamente 2:

Estado 0: No se recibió el byte de inicio de trama aun.
Estado 1: Ya se recibió el byte de inicio de trama.

En el estado 0 la única función que se lleva a cabo es la de verificar si el byte recibido es el de inicio de trama o no. Si es, se pasa al estado 1, si no, el dato simplemente se descarta.

Si se está en el estado 1, entonces el byte de inicio de trama fue recibido anteriormente, por lo tanto se deben guardar todos los bytes que se reciben en un buffer hasta encontrar el byte de fin de trama. Cuando esto ocurre, se levanta una bandera que le indica al programa principal que se recibió una trama completa, como se puede observar en el diagrama de flujo del programa principal. Luego, el programa principal desactiva todas las interrupciones ya que de ocurrir alguna durante la modulación, la señal quedaría mal modulada (es decir, no tendría la forma de onda deseada) y no se podría recuperar durante la recepción. Si el byte de fin de trama no llega a ser recibido por cualquier razón, entonces ocurrirá un timeout y el ciclo vuelve a empezar, esperando por un nuevo byte de inicio de trama.

Una vez recibido el byte de fin de trama y deshabilitadas las interrupciones, el siguiente paso es generar la señal modulada en AFSK a 1200 bps, agregar las banderas de inicio y fin de trama y calcular el CRC de la trama completa según las especificaciones del protocolo HDLC. Para lograr dicho propósito, se implementaron dos rutinas que en el código se encuentran bajo los nombres de “Rutina de interrupción por desborde de timer0” y “Rutina para enviar la trama” o “ENVIARMSG”. Además de estas dos, hay varias rutinas pequeñas de ayuda que no se van a detallar en este documento dado que su funcionamiento se encuentra comentado en el código mismo.

La rutina para enviar la trama es la rutina que va leyendo los datos a enviar del buffer de transmisión. Se encarga de pasarle el próximo bit a enviar a la rutina de interrupción por desborde del TIMER0. También se encarga de calcular los 2 bytes del CRC que debe ir en el último campo de la trama AX.25 antes de la bandera de

finalización de trama y de implementar el bit-stuffing. Está compuesta por 4 bloques o loops principales con una estructura interna muy similar y que es posible generalizar con el siguiente diagrama:

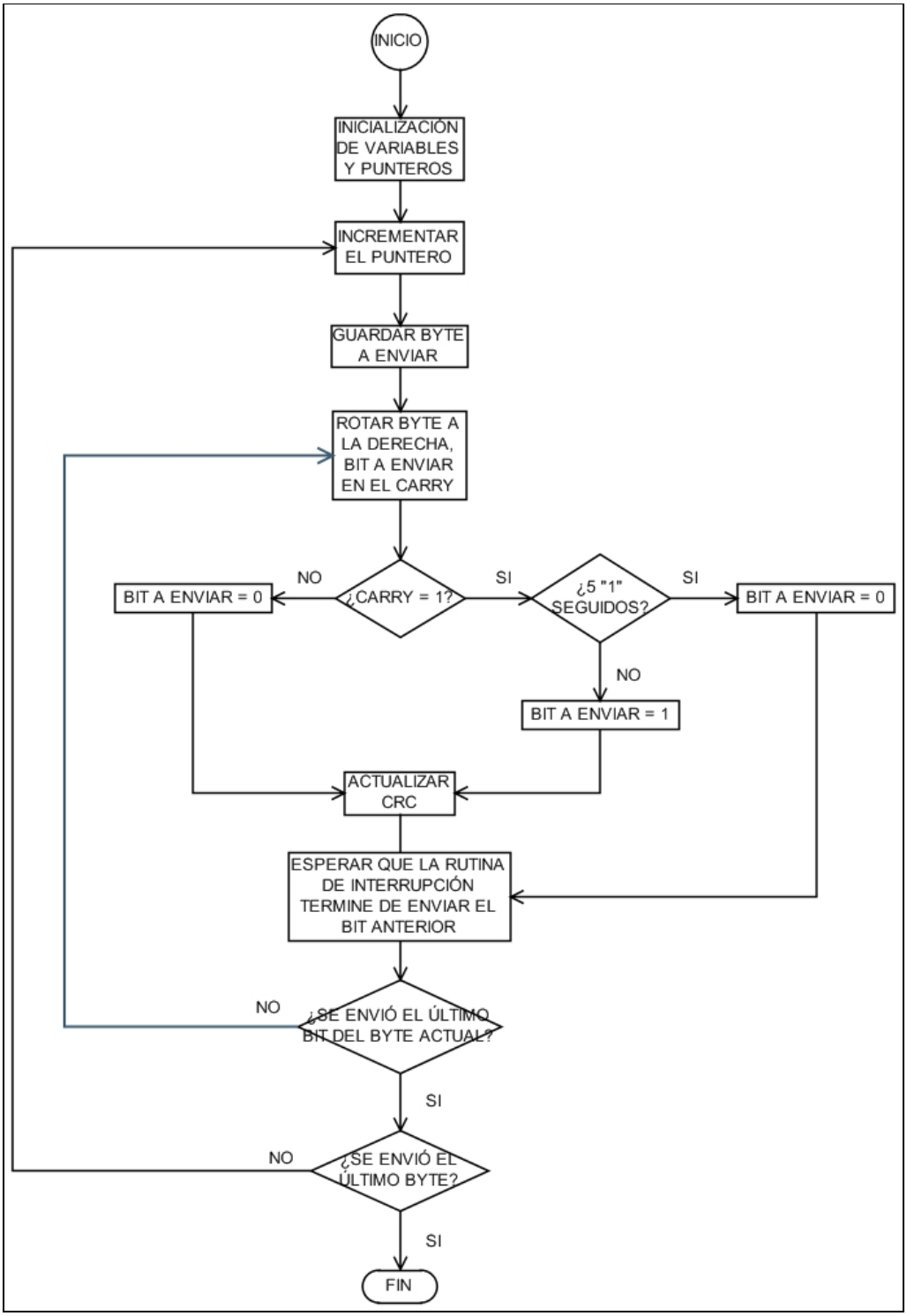

Figura 13 Rutina para enviar la trama

Esta división en bloques similares se debe a que en cada uno se procesan distintos tipos de información, pero el procedimiento para enviar los bytes que las conforman es casi el mismo. En el primer y cuarto bloque se generan las banderas de inicio y fin de trama del protocolo HDLC. Como se mencionó anteriormente, el único caso en que el receptor recibe seis "1" seguidos es en estas banderas. Por lo tanto, en estos bloques se debe eliminar el control del bit-stuffing. Además, como lo especifica el protocolo HDLC, el cálculo del CRC se debe hacer con los bytes de la trama completa exceptuando las banderas, por lo que también se elimina el bloque "Actualizar CRC" del diagrama anterior.

Algo similar ocurre en el tercer bloque, que se encarga de leer el CRC que fue generado con los datos que estaban guardados en el buffer. En este caso sí se debe implementar bit-stuffing, pero el bloque "Actualizar CRC" también es eliminado como en los bloques primero y cuarto por razones obvias.

El segundo bloque es entonces el único que implementa todas las funciones que aparecen en el diagrama anterior. Primero se inicializan contadores y punteros para la lectura de los datos del buffer. Se guarda el primer dato del buffer en un registro de la memoria y se va rotando el mismo hacia la derecha. En cada rotación, sale el bit que se encontraba en la primera posición del registro el cual queda guardado en un bit del registro de estado del PIC16F877A (bit de carry). Si el bit que salió fue un "0", se actualiza el CRC y se levanta una bandera que le avisa a la rutina de interrupción por desborde de timer 0 que hay un nuevo bit para enviar. Si el bit que sale es un "1", hay que tener en cuenta el bit stuffing. Para la implementación del bit stuffing, la rutina lo que hace es incrementar un contador cada vez que lee un "1" y borrar dicho contador cuando lee un cero. Si el contador llega a 5, se enviaron 5 unos seguidos, por lo que le debe informar a la rutina que genera los tonos que el próximo bit a enviar debe ser un "0". Cuando este bit "0" es insertado no se actualiza el CRC ya que en recepción se eliminan todos los bits "0" que aparezcan luego de cinco "1" consecutivos.

Cuando la rutina de interrupción por desborde de timer 0 lee el nuevo bit a enviar, levanta una bandera para avisarle a la rutina para enviar la trama que puede continuar con el procesamiento de los datos. En cada rotación del byte que se está enviando se decrementa un contador que se inicializa en 8 (1 byte = 8 bits). Cuando el

contador llega a 0 es porque en el carry quedó guardado el último bit (el más significativo) del byte que se está procesando. Cuando este último bit es enviado, se decrementa otro contador que fue inicializado con la cantidad de bytes que hay para enviar en el buffer. Mientras dicho contador sea distinto de 0, se va incrementando el puntero al próximo byte que será guardado en el registro mencionado anteriormente. Cuando este contador llega a 0 es porque ya se procesó el último bit del último byte que había en el buffer y se procede a enviar el CRC que fue calculado en esta etapa.

Sobre el cálculo del CRC, como se mencionó al principio de este capítulo, la especificación en el protocolo es muy escueta. La única mención sobre este campo es que el CRC debe ser calculado según el polinomio de la CRC-CCITT. El problema es que hay diferentes variantes para calcularlo, y la especificación no aclara cuál de todas esas variantes hay que utilizar. El algoritmo que está implementado en este modulador es el que se menciona en la sección 6.2.2.

Por último, resta mencionar cómo se implementa la generación de la señal en AFSK. Como se mencionó al principio del capítulo, en AFSK a 1200 bps se utiliza un tono a 1200 Hz y otro tono a 2200 Hz para representar las marcas y los espacios. Un 1 lógico se representa manteniendo constante la frecuencia del tono y un 0 lógico como un cambio en la frecuencia. A modo de ejemplo, si el tono actual es de 1200 Hz y se reciben dos “0” consecutivos, con el primero se cambia la frecuencia de la señal a 2200 Hz y con el segundo cero vuelve a cambiar a 1200 Hz.

El diagrama siguiente puede ayudar a comprender mejor el código de generación de los tonos:

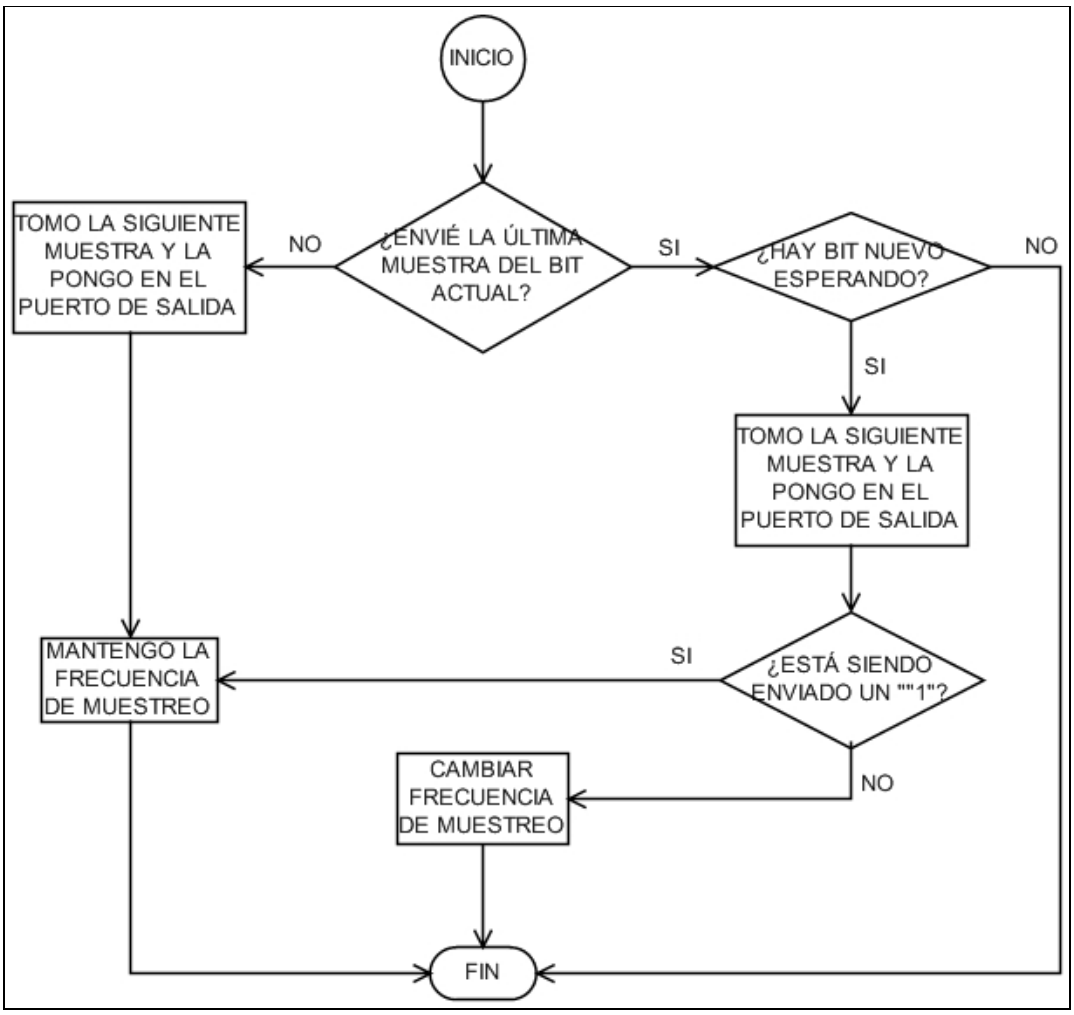

Figura 14 Rutina de interrupción por desborde del timer 0.

Este diagrama representa lo que en el código aparece bajo el nombre de “Rutina de interrupción por desborde de timer 0”. El timer 0 es un contador propio del PIC16F877A que una vez seteado su valor inicial y al ser habilitado, comienza un conteo regresivo a partir de dicho valor inicial. Cuando llega a 0, se genera una interrupción y se ejecuta el código de generación de tonos. Lo que se consigue al utilizar interrupciones con este timer para generar los tonos es que la frecuencia de muestreo va a ser la misma ya que las muestras se envían en intervalos de tiempo iguales y además el programa principal puede seguir realizando operaciones entre interrupción e interrupción.

A grandes rasgos, lo que hace esta rutina es tomar un valor de una tabla que contiene muestras de una señal sinusoidal y ponerla en el puerto de salida que cuando se le conecta el conversor D/A genera la señal modulada en AFSK.

Un detalle muy importante que no hay que dejar pasar por alto al generar los tonos es que entre bit y bit que se trasmite la fase de la onda debe ser contínua. Para mantener la fase entre cada tono, se utiliza una "lookup-table", la cual consiste en una tabla con los valores de las muestras de una sinusoide. En cada interrupción, se consulta la tabla y se toma el valor siguiente a trasmitir. Cuando se termina de trasmitir un tono y llega un nuevo bit, si es un 1, el tono se mantiene, por lo que se mantiene la frecuencia de muestreo y se toma el valor siguiente. Ahora, si el nuevo bit a trasmitir es un 0, se debe cambiar de la frecuencia del tono. Lo que la rutina de generación hace para mantener la fase y cambiar la frecuencia del tono es cambiar la frecuencia de muestreo. Es decir, el valor con el que se carga el contador del timer 0 es menor cuando se quiere trasmitir una frecuencia más grande (menor valor en el contador implica más interrupciones por unidad de tiempo). De esta forma, utilizando una única tabla con las muestras de una sinusoide y cambiando la frecuencia con la que se leen los datos, se logra generar una señal de fase continua. El valor con el que se setea el contador del timer 0 lo establecen los bloques que en el diagrama anterior aparecen como "Mantener (cambiar) la frecuencia de muestreo".

Cuando la rutina manda la última muestra de un bit y ya no hay mas bits para trasmitir, el programa principal deshabilita las interrupciones por desborde del timer 0, deshabilita el timer y vuelve al estado inicial donde espera recibir nuevos datos a trasmitir a través del puerto serie.

7. TRANSMISOR

La construcción de un transmisor FM se definió como tarea opcional en el Plan de Proyecto. Sin embargo, dado que esta tarea resultó de gran interés se decidió avanzar en la construcción del circuito y en este momento se cuenta con un prototipo inicial, el cual requiere determinadas modificaciones para lograr el desempeño requerido. La implementación de este transmisor requirió de cierto aprendizaje en el área de la radiofrecuencia y en la construcción de circuitos, puesto que sólo se contaba con conocimientos básicos al inicio del proyecto. Se construyeron varios prototipos a medida que se avanzaba en el aprendizaje y se fue mejorando el desempeño de los mismos, sin embargo aún no se llego a un resultado óptimo.

A continuación se presentan los criterios de diseño de transmisores FM y el desarrollo del circuito implementado.

7.1. INTRODUCCIÓN FM

La modulación en frecuencia (FM) se usa extensamente para la radiodifusión de radio comercial, transmisión de audio de televisión, radio móvil de dos sentidos, radio celular y los sistemas de comunicaciones por microondas y satélite. Corresponde a una modulación donde tanto las señales de transmisión como las señales de datos son analógicas y es un tipo de modulación exponencial.

Existen varias ventajas en utilizar la modulación en frecuencia en vez de la modulación en amplitud (AM), como son la reducción de ruido, la fidelidad mejorada del sistema y el uso más eficiente de la potencia.

En FM la señal modulada mantendrá fija su amplitud y el parámetro de la señal portadora que variará es la frecuencia, y lo hace de acuerdo a como varíe la amplitud de la señal moduladora.

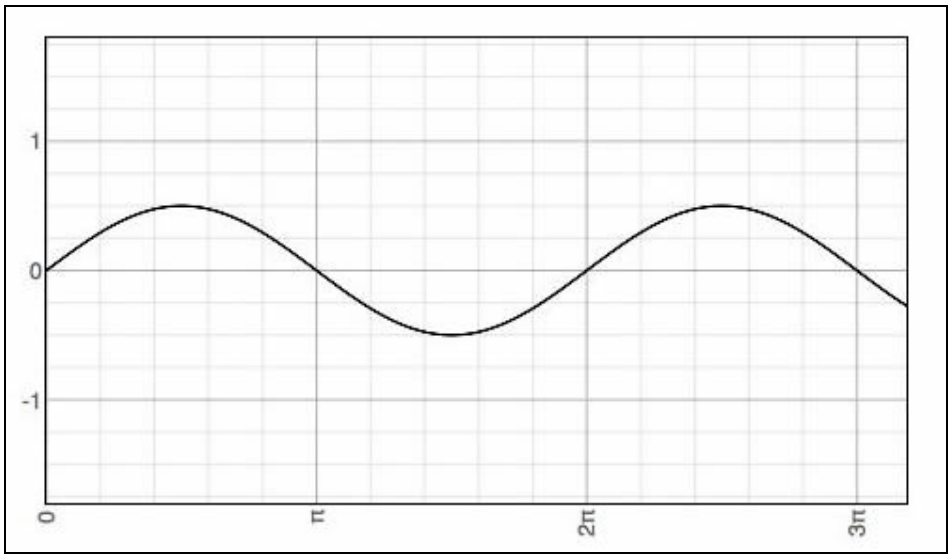

Figura 15 Señal moduladora

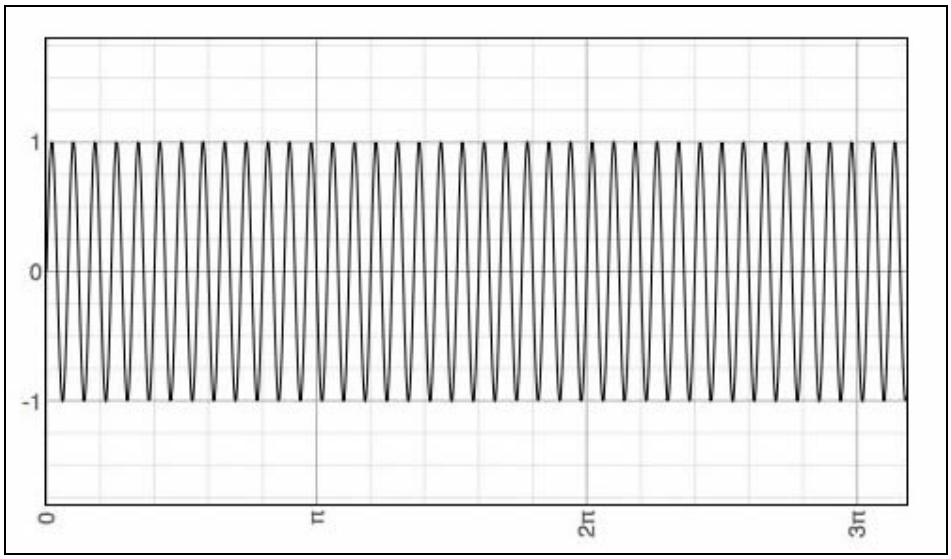

Figura 16 Señal portadora

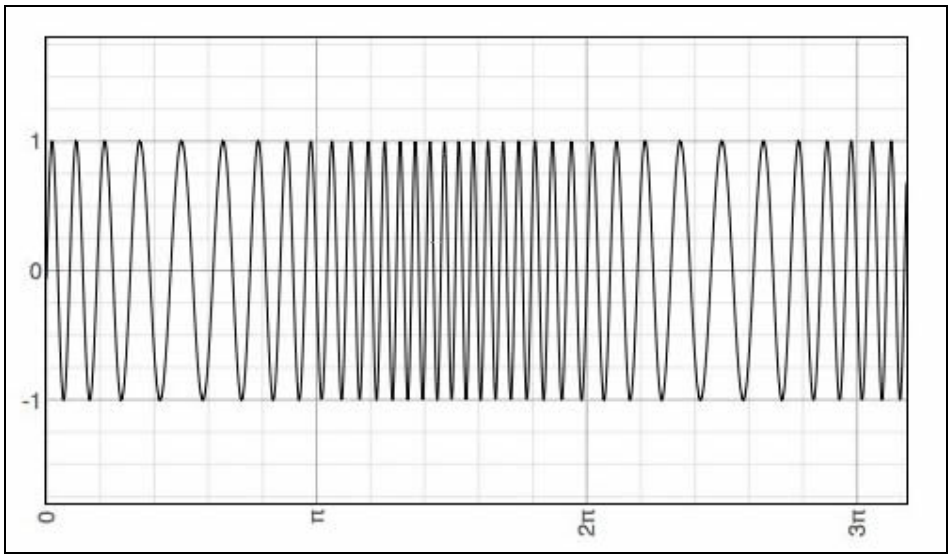

Figura 17 Señal Modulada

Sea $x_m(t)$ la señal en banda base a transmitir, con la restricción en amplitud $|x_m(t)| < 1$, la expresión matemática de la señal portadora estará dada por:

$$x_c(t) = A_c \cos(2\pi f_c t) \quad (1)$$

Donde A_c es el valor pico de la señal portadora y f_c es la frecuencia de la misma.

Al momento de elegir el tipo de modulación y la frecuencia en la cual transmitir se debió tener en cuenta que los datos recibidos en tierra serán decodificados principalmente por radioaficionados, los cuales utilizan ampliamente la banda de dos metros (144Mhz-174Mhz). Por lo tanto se decidió adoptar una frecuencia de portadora de 145.030MHz.

El modulador combina la señal de la portadora con la señal a transmitir en banda base para generar la señal transmitida:

$$y(t) = A_c \cos\left(2\pi \int_0^t f(\tau) d\tau\right) = A_c \cos\left(2\pi \int_0^t [f_c + f_\Delta x_m(\tau)] d\tau\right) = A_c \cos\left(2\pi f_c t + 2\pi f_\Delta \int_0^t x_m(\tau) d\tau\right)$$

Siendo $f(\tau)$ la frecuencia instantánea del oscilador y f_Δ la frecuencia de desviación, la cual representa el máximo corrimiento respecto a la frecuencia de la portadora, asumiendo que $|x_m(t)| < 1$. De esta forma se define $\Delta f = f_\Delta |x_m(t)|$ como la desviación de frecuencia y es el máximo cambio de frecuencia que puede experimentar la frecuencia de la señal portadora. A la variación total de frecuencia desde la más baja hasta la más alta, se la conoce como oscilación de portadora.

En este caso la señal moduladora es la generada por el microcontrolador PIC, y corresponde a una señal AFSK de frecuencia 1200Hz y 2200Hz.

Se denomina índice de modulación a $m_f = \frac{\Delta f}{f_m}$, donde f_m es la mayor frecuencia de modulación de $x_m(t)$.

Se denomina porcentaje de modulación a la razón entre la desviación de frecuencia efectiva respecto de la desviación de frecuencia máxima permisible.

$$\text{Porcentaje de modulación} = \frac{\Delta f_{efectiva}}{\Delta f_{máxima}}.100$$

Al analizar el espectro de frecuencias de una señal modulada en frecuencia, se observa que se tienen infinitas frecuencias laterales, espaciadas en f_m, alrededor de la frecuencia de la señal portadora f_c; sin embargo la mayor parte de las frecuencias laterales tienen poca amplitud, lo que indica que no contienen cantidades significativas de potencia.

El análisis de Fourier indica que el número de frecuencias laterales que contienen cantidades significativas de potencia, depende del índice de modulación de la

señal modulada, y por lo tanto el ancho de banda efectivo también dependerá de dicho índice.

Un estudio matemático detallado indica que el ancho de banda necesario para transmitir una señal FM para la cual $m_f < \frac{\pi}{2}$, depende principalmente de la frecuencia de la señal moduladora y es totalmente independiente de la desviación de frecuencia. Un análisis más completo demostraría que el ancho de banda necesario para transmitir una señal de FM, en la cual $m_f < \frac{\pi}{2}$, es igual a dos veces la frecuencia de la señal moduladora.

$$BW = 2f_m \text{ para } m_f < \frac{\pi}{2}$$

De igual manera que en AM ya a diferencia de lo que ocurre para FM con $m_f > \frac{\pi}{2}$, por cada frecuencia moduladora aparecen dos frecuencias laterales, una inferior y otra superior, a cada lado de la frecuencia de la señal portadora y separadas en f_m de la frecuencia de la portadora. Dado lo limitado del ancho de banda cuando $m_f < \frac{\pi}{2}$, se la denomina FM de banda angosta, mientras que las señales de FM donde $m_f < \frac{\pi}{2}$, se las denomina FM de banda ancha.

Los espectros de frecuencia de AM y de FM de banda angosta, aunque pudieran parecer iguales, por medio del análisis de Fourier se demuestra que las relaciones de magnitud y fase en AM y FM son totalmente diferentes

En FM de banda ancha se tiene la ventaja de tener menor ruido.

En FM el contenido de potencia de las señal portadora disminuye conforme aumenta m_f, con lo que se logra poner la máxima potencia en donde está la información, es decir en las bandas laterales.

7.2. REQUERIMIENTOS

Un transmisor debe generar una señal con el tipo correcto de modulación, con suficiente potencia, en la frecuencia portadora correcta y con razonable eficiencia. La señal de salida tiene que estar acoplada a una antena. La modulación debe hacerse con la suficiente exactitud de manera que cuando la recupere un receptor, sea una copia razonablemente fiel de la señal modulada original.

Todos los transmisores generan señales espurias, es decir, generan señales a frecuencia distinta de la portadora. A menudo, estas señales son armónicas de la frecuencia de operación o del oscilador de la portadora. Todas las frecuencias, excepto la asignada para la transmisión, deben ser filtradas para evitar interferencias con otras transmisiones.

Los transmisores reales son de una variedad infinita, pero la mayoría son variaciones de las estructuras básicas.

En todos los casos se genera y se transmite una señal modulada en radiofrecuencia (RF). Un sintetizador de frecuencias genera la portadora y luego se amplifica a su potencia de salida. Se utiliza opcionalmente un multiplicador de frecuencias si, la frecuencia requerida de portadora fuera más alta de lo que puede generar en forma adecuada el sintetizador.

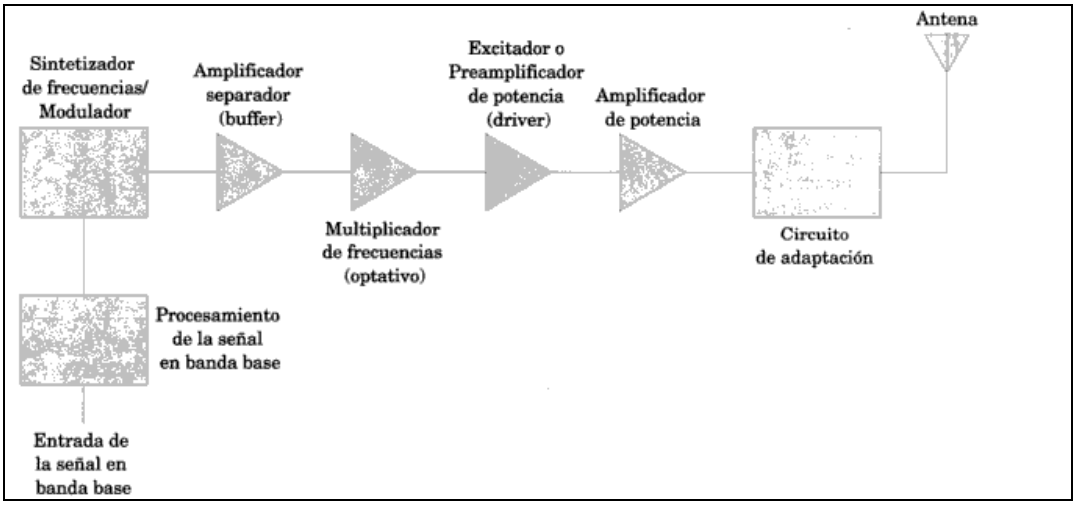

Figura 18 Diagrama de bloques de un transmisor

Los diseños de transmisores terminan con un amplificador de varias etapas. Se muestran dos etapas: el excitador o preamplificador de potencia (driver) y el amplificador de potencia. Podrían ser necesario mas, sobretodo para niveles altos de potencia de salida. El circuito de adaptación acopla al amplificador de potencia a la impedancia de carga, que es casi siempre de 50 ohms, y también elimina armónicos y otras señales espúreas a la salida del transmisor.

El siguiente es un diagrama de bloques de un transmisor de FM representativo.

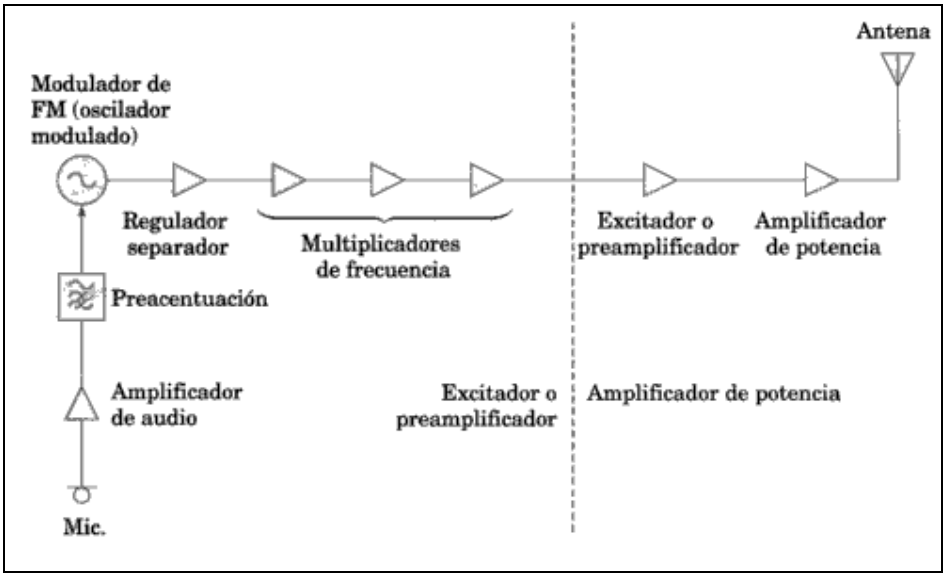

Figura 19 Etapas de un transmisor

Las señales de FM se generan en forma directa cuando se varía la frecuencia del oscilador de portadora, o indirecta, cuando se convierte la modulación de fase en modulación de frecuencia.

En la FM directa se requiere que la frecuencia de oscilación de portadora varíe en forma correspondiente con la amplitud instantánea de la señal moduladora. Una manera de lograrlo es usando un modulador de reactancia. Para su funcionamiento utiliza la señal moduladora para modificar una reactancia en el circuito que determina la frecuencia. Esto se logra utilizando un varactor, o diodo de reactancia variable dentro del circuito que determina la frecuencia de oscilación de portadora. El varactor tiene polarización inversa y ésta varía con la señal moduladora.

Los multiplicadores de frecuencia son en general amplificadores clase C con salida sintonizado con una armónica de la frecuencia de entrada. El los transmisores de FM, multiplican la desviación de una señal en FM por el mismo factor que la frecuencia portadora. Si se supone que una portadora no modulada con frecuencia f_c se aplica a la entrada de un circuito que multiplica la frecuencia por N . Entonces, la frecuencia de la señal de salida es Nf_c .

7.3. OSCILADOR/MODULADOR

En modulación de frecuencia se suele modular directamente sobre el oscilador, por esta razón se incluyeron el oscilador y el modulador en un mismo bloque.
En primer lugar se explicará el funcionamiento del oscilador para luego pasar a la etapa de modulación.

7.3.1. CONDICIONES DE OSCILACIÓN

Un oscilador puede ser pensado como un sistema de lazo cerrado compuesto por un amplificador y una red de realimentación (que contendría al cristal) que determina la frecuencia de operación. En la Figura 20 se puede apreciar un esquema simplificado del sistema.

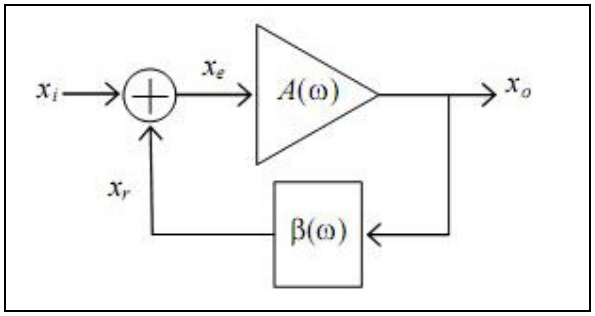

Figura 20 Diagrama de bloques de un circuito lineal con realimentación positiva

x_i y x_o son las señales de entrada y salida, mientras que x_r y x_e son, respectivamente, la señal de realimentación y la señal de error. A es la ganancia del amplificador inicial, o ganancia en lazo abierto, β es el factor de realimentación y $A\beta$ es la ganancia de lazo cerrado. La ganancia del circuito realimentado es

$$\frac{x_o}{x_i} = \frac{A}{1 - A\beta}$$

El comportamiento del circuito se puede predecir conociendo el módulo, $|A\beta|$ y la fase, $\varphi_{A\beta}$, de la ganancia de lazo.

- Si $|A\beta| < 1$, el circuito es estable sea cual sea el valor de $\varphi_{A\beta}$.
- Si a una frecuencia determinada $A\beta = 1$, es decir $|A\beta| = 1$ y $\varphi_{A\beta} = 0$, cualquier oscilación presente en la entrada a esa frecuencia se mantiene indefinidamente, a la misma amplitud.
- Si a una frecuencia determinada $A\beta > 1$, es decir $|A\beta| > 1$ y $\varphi_{A\beta} = 0$, cualquier oscilación presente en la entrada a esa frecuencia se amplifica indefinidamente hasta que la saturación del amplificador lo devuelva a la condición anterior. Como la saturación es un fenómeno no lineal, la misma provoca la aparición de armónicos.

Si el circuito tiene $A\beta > 1$ se puede prescindir de la señal de entrada puesto que el ruido, siempre presente, contiene componentes a todas las frecuencias. La componente de ruido a la frecuencia en la que se cumpla esta condición, conocida como condición de arranque, se amplifica indefinidamente hasta la saturación del amplificador o hasta que un circuito auxiliar consiga que para esa frecuencia $A\beta = 1$. A partir de entonces la amplitud de la oscilación se mantiene, por eso a la condición $A\beta = 1$ se la denomina condición de mantenimiento. Estas condiciones para que un circuito oscile se conocen como criterio de Barkhausen.

7.3.2. OSCILADORES LC

Un oscilador LC utiliza una red de realimentación compuesta por tres impedancias, ya sean inductancias o capacidades. El circuito equivalente de un oscilador LC se ve a continuación:

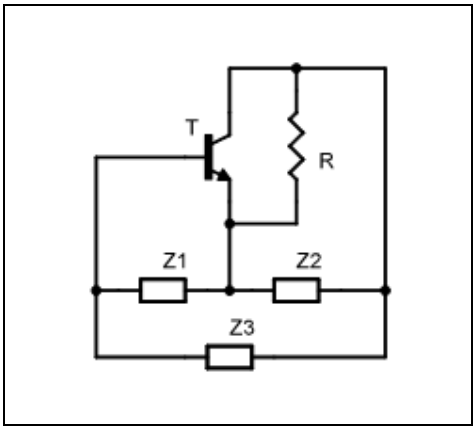

Figura 21 Oscilador LC genérico

La red de realimentación está constituida por Z1, Z2 y Z3. Se pueden distinguir dos tipos de osciladores según la combinación LC seleccionada. Si bien existen distintas combinaciones LC, por ejemplo: tres capacitores o tres inductancias o dos capacitores con una inductancia o dos inductancias con un capacitor, de todas estas solamente dos combinaciones LC podrán constituir un oscilador. Se puede demostrar matemáticamente que la combinación Z1 y Z2 del mismo tipo y Z3 de signo contrario es la adecuada. Si Z1 y Z2 son inductores con acoplamiento mutuo y Z3 es un capacitor, al oscilador que constituye se lo llama Hartley. Pero si Z1 y Z2 son capacitores y Z3 es un inductor, al oscilador que constituye se lo llama Colpitts, siendo este el más utilizado en la actualidad.

Las dos impedancias Z1 y Z2 con la impedancia Z3 conforman un circuito resonante capaz de oscilar a la frecuencia deseada. La realimentación se produce debido a que el circuito resonante a esta frecuencia produce un desfasaje de 180º entre la señal de entrada y la señal de salida. Esto significa que la realimentación produce un desfasaje de 180º solo a la componente cuya frecuencia coincide con la de resonancia, siendo distinto de 180º para las otras componentes.

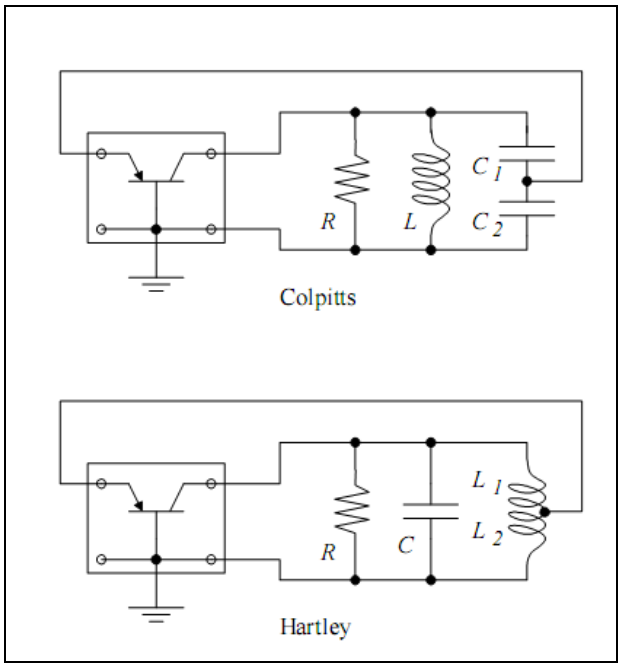

Figura 22 Esquemáticos de los osciladores Colpitts y Hartley

En la figura se representan los osciladores Colpitts y Hartley. El oscilador de Colpitts es uno de los más comúnmente utilizados, ya que la derivación en el condensador es más económica, en general, que la derivación en la autoinducción.

La figura 23 muestra el diagrama esquemático de un oscilador Colpitts. El amplificador transistorizado (Q1) proporciona la amplificación necesaria para una ganancia de voltaje de lazo unitaria a frecuencia de resonancia. El capacitor de acoplamiento (Cc) proporciona la ruta para la realimentación regenerativa, L, C1a y C1b son los componentes que determinan la frecuencia, y V es la fuente de voltaje de c.c.

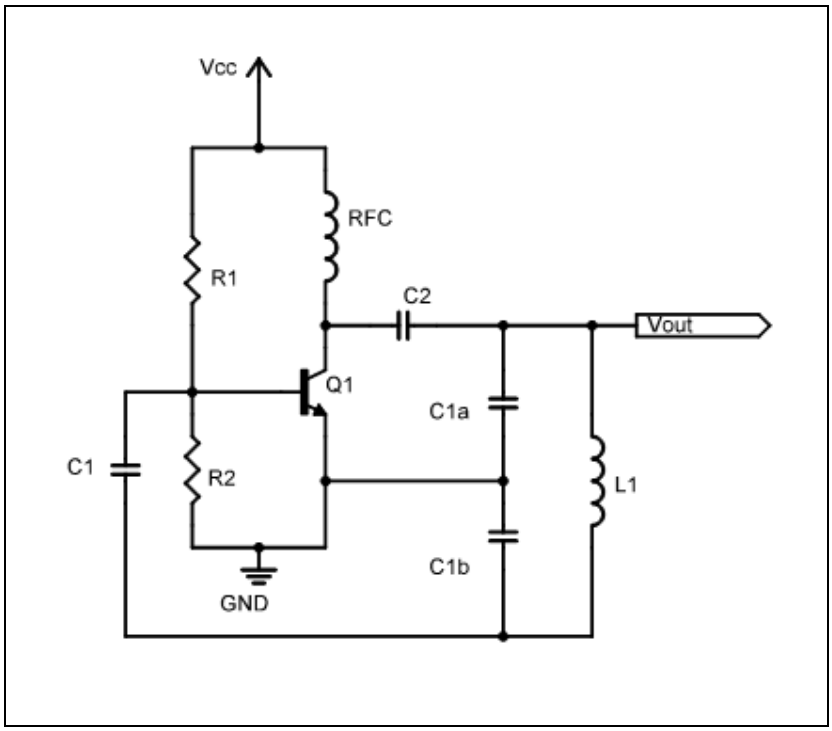

Figura 23 Oscilador Colpitts

La operación del oscilador Colpitts es la siguiente: en el arranque inicial, aparecen una multitud de frecuencias en el colector de Q1 que se acoplan a través de C2 dentro del circuito tanque, haciendo que empiece a oscilar. C1a y C1b constituyen un divisor de voltaje en ca. El voltaje que se deja caer a través de C1b se realimenta a la base de Q1 por medio de Cc. La señal amplificada aparece en el colector 180º fuera de fase con la señal de base. Se realiza un desplazamiento adicional de fase de 180º a través de (C1a + C1b). En consecuencia, el cambio total de fase es de 360° y la señal de

realimentación es regenerativa manteniendo así las oscilaciones sin señal de entrada externa.

La proporción de energía oscilatoria que se realimenta a la base de Q1 se determina por la razón de C1a a C1a + C1b. Si se realimenta insuficiente energía, las oscilaciones se amortiguan. Si se realimenta energía en exceso, el transistor se satura. Por lo tanto, los valores de C1a y C1b se ajustan hasta que la cantidad de energía de realimentación sea exactamente la requerida para una ganancia de voltaje de lazo unitario, garantizando así la continuidad de las oscilaciones.

Con la siguiente fórmula se obtiene una aproximación cercana a la frecuencia de oscilación del oscilador Colpitts:

$$f_0 = \frac{1}{2\pi\sqrt{(LC)}} \qquad C = \frac{C_{1a}C_{1b}}{C_{1a} + C_{1b}}$$

La relación de la reactancia de la bobina o del condensador, a la frecuencia de resonancia con la resistencia se denomina Q (factor de calidad) del circuito, y determina la agudeza de la curva de resonancia (corriente versus frecuencia).

Los osciladores LC difícilmente permiten alcanzar factores de calidad superiores a 200. Para conseguir factores de calidad superiores y una mejor estabilidad de la frecuencia de oscilación (frente a variaciones de la temperatura, de la tensión de alimentación o de la resistencia de carga) se pueden utilizar cristales de cuarzo.

7.3.3. OSCILADOR A CRISTAL DE CUARZO

Los cristales de cuarzo se pueden utilizar para que el circuito pueda resonar únicamente a la frecuencia de resonancia serie del cristal (aprovechando el hecho de que a esta frecuencia el cristal presenta una impedancia mínima) o a la frecuencia de resonancia paralelo (a esta frecuencia presenta una impedancia máxima).

La figura 24 muestra el circuito eléctrico equivalente para un cristal. Cada componente eléctrico es equivalente a una propiedad mecánica del cristal. Cp es la capacitancia real formada entre dos electrodos del cristal, mientras que el cristal en sí es el dieléctrico. Cs es equivalente al relajamiento mecánico del cristal (también se llama resistencia o elasticidad) L, es equivalente a la masa del cristal en vibración, y R es la pérdida por fricción mecánica. En un cristal, es bastante alta la relación de masa a fricción mecánica (L/R).

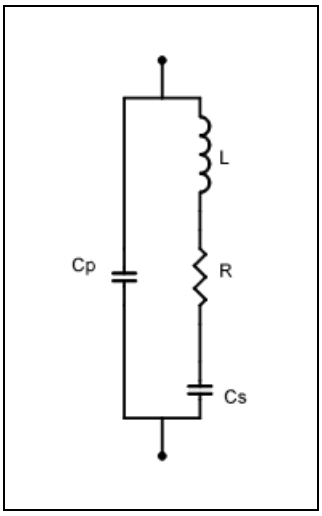

Figura 24 Circuito eléctrico equivalente de un cristal

La frecuencia de resonancia serie queda determinada por el circuito serie LRCs y se expresa como:

$$f_s = \frac{1}{2\pi\sqrt{LC_s}}$$

A esta frecuencia la impedancia que presenta el cristal es mínima.

Para obtener la frecuencia de resonancia paralelo se debe tener en cuenta el capacitor Cp, que queda conectado en serie con el capacitor Cs, la capacidad total esta dada por el capacitor equivalente, la expresión en este caso es:

$$f_p = \frac{1}{2\pi\sqrt{\frac{C_s C_p}{C_s + C_p}}}$$

La gráfica de variación de la reactancia y de la resistencia en función de la frecuencia para un cristal es la siguiente:

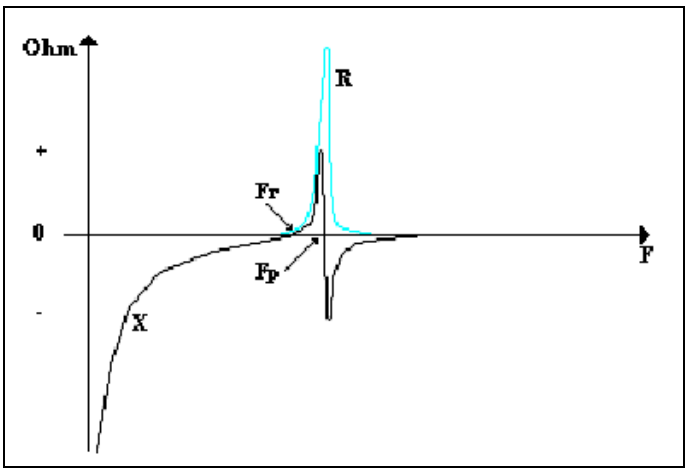

Figura 25 Reactancia y resistencia versus frecuencia

A continuación se muestra un oscilador Colpitts controlado por cristal, similar al utilizado en el circuito implementado.

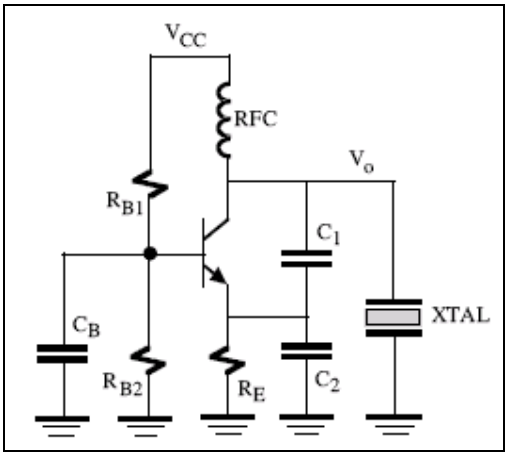

Figura 26 Oscilador Colpitts controlado por cristal

En este caso el cristal funcionaría como la inductancia del oscilador. Para que esto ocurra el cristal debe funcionar a la frecuencia de resonancia paralelo.

7.3.4. MODULADOR

Como se mencionó anteriormente, en modulación de frecuencia se suele modular directamente sobre el oscilador. En este caso para obtener la señal modulada en frecuencia se debe modificar la frecuencia de oscilación con la señal modulante. En un oscilador controlado por cristal el corrimiento de frecuencia será bajo, pero lo suficiente como para obtener la desviación en frecuencia necesaria. Para un cristal de 20 MHz se puede llegar a un corrimiento de su frecuencia del orden de 1 kHz. Esta desviación en frecuencia es baja y no suficiente para ser emitida, esto hace necesario utilizar etapas multiplicadoras a continuación del oscilador que eleven la frecuencia de operación y aumenten la desviación en frecuencia. Por ejemplo, si se desvía 1 KHz la frecuencia del cristal y posteriormente se la multiplica por 9, se obtiene una desviación de frecuencia de 9 KHz, lo que supera la desviación de frecuencia normalizada Δf y cuyo valor es igual a ± 5K Hz como máximo. Este valor normalizado no se puede superar, entonces cuando se modula se debe llegar como máximo a esa desviación. Para poder obtener en la salida ± 5K Hz de desviación si se multiplica por 9, se deberá desviar la frecuencia de oscilación ± 555 Hz.

Para modificar la frecuencia de oscilación se suele incorporar al circuito un elemento que permita obtener automáticamente una variación de su capacidad con la señal modulante. Este elemento se llama Varicap, el mismo es un diodo polarizado inversamente y que produce una variación de su capacidad según la tensión aplicada en sus extremos.

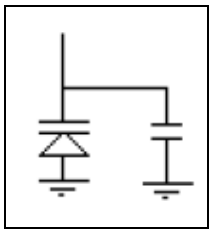

Figura 27 Modelo del Diodo Varicap

Si al diodo se lo polariza en sentido directo se comportará como un simple diodo. Se construyen diodos varicap para distintas aplicaciones, esto significa que difiere el rango de variación de capacidad. Por ejemplo, para los diodos a ser utilizados en equipos de FM en la banda de VHF la variación de capacidad puede ser de 5 a 20 pF,

para el caso de UHF la variación podría ser del orden de 1 a 5 pF, mientras que para HF la variación puede ser del orden de 20 a 400 pF.

Las variaciones de tensión pueden ser del orden de 0 a 15 V aproximadamente.

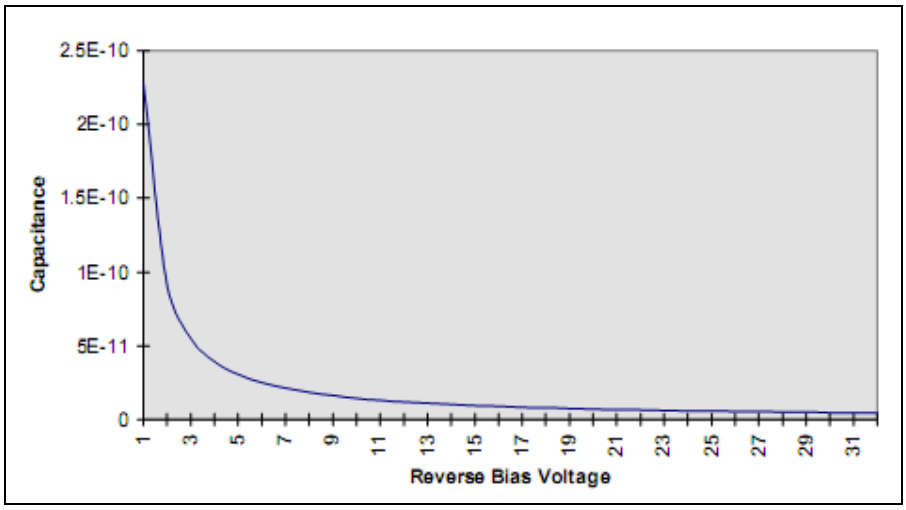

Figura 28 Curva de variación capacidad – tensión para un Diodo Varicap

No conviene utilizar los extremos de la curva debido a la falta de linealidad que presenta, lo usual es utilizar una pequeña porción de la curva donde esta sea lo más lineal posible. Se polariza entonces al diodo en ese sector y con esta tensión continua se inyecta la modulación, ésta producirá variaciones del valor de la capacidad dentro de ese entorno, lo que se traduce en variaciones de frecuencia.

En algunos casos, para obtener una variación de frecuencia adecuada se requieren grandes variaciones de la capacidad del varicap. Para mejorar esto y facilitar la modulación se coloca al cristal con un circuito resonante serie en serie con el.

7.4. AMPLIFICADORES DE POTENCIA

En los transmisores se usan tres tipos básicos de amplificadores de potencia: lineales, clase C y por conmutación. Los amplificadores lineales proporcionan una señal de salida que es una réplica idéntica y aumentada de la entrada; su salida es directamente proporcional a su entrada y, por lo tanto, reproducen de manera fiel una entrada, pero a un nivel de potencia más alto. Todos los amplificadores de audio son lineales. Para elevar el nivel de potencia de señales de RF de amplitud variable, como las señales de AM o BLU de bajo nivel, es necesario usar amplificadores RF lineales. Las señales de frecuencia modulada no varían en amplitud y, por lo tanto, pueden ampliarse con amplificadores clase C o de conmutación no lineal, los cuales tienen una eficiencia mayor.

Los amplificadores se clasifican de acuerdo con la porción del ciclo de entrada durante el cual, el dispositivo activo conduce la corriente. A esto se llama ángulo de conducción y se expresa en grados. El dispositivo activo podría ser un transistor bipolar, un FET o un tubo de vacío. Tienen la particularidad de que en su salida tenemos ganancia de tensión y de corriente con respecto a la señal de entrada.

Los amplificadores de clase A conducen la corriente todo el tiempo para un ángulo de conducción de 360°. Con esto se logra una operación lineal, donde la salida es una copia fiel de la entrada, excepto por la amplitud. Un amplificador de potencia funciona en clase A cuando la tensión de polarización y la amplitud máxima de la señal de entrada poseen valores tales que hacen que la corriente de salida circule durante todo el período de la señal de entrada.

En un amplificador clase B, el transistor se polariza para operar en la región de corte, así conduce durante 180° del ciclo de entrada.

Los amplificadores de clase AB son, por así decirlo, una mezcla de los dos anteriores. Un amplificador de potencia funciona en clase AB cuando la tensión de polarización y la amplitud máxima de la señal de entrada poseen valores tales que hacen

que la corriente de salida circule durante menos de un período y más de un semiperíodo de la señal de entrada.

Un amplificador de clase C conduce por menos de 180° de su ciclo de entrada. El amplificador puede ser asimétrico (single-ended) o en contrafase (push-pull) al igual que el clase B. De todas maneras, la señal de salida para una entrada sinusoidal se asemejará mas a una serie de pulsos que a la señal original Estos pulsos de corriente, pueden transformarse nuevamente en ondas seno, mediante un circuito sintonizado de salida. El dispositivo estará en corte la mayor parte del ciclo. Este tipo de operación reduce la disipación de potencia en el transistor. Este es el tipo de amplificadores que se usaron en el circuito implementado.

7.5. MULTIPLICADORES DE FRECUENCIA

Una forma especial del amplificador clase C es el multiplicador de frecuencia. Cualquier amplificador clase C tiene la capacidad de realizar tal multiplicación si el circuito sintonizado en el colector entra en resonancia a un múltiplo entero de la frecuencia de entrada. Por ejemplo, un duplicador de frecuencia se puede construir con sólo conectar un circuito sintonizado paralelo que entre en resonancia al doble de la frecuencia de entrada, en el colector de un amplificador clase C. Cuando ocurre el pulso de corriente del colector, el circuito sintonizado se excita, y éste repica al doble de la frecuencia de entrada.

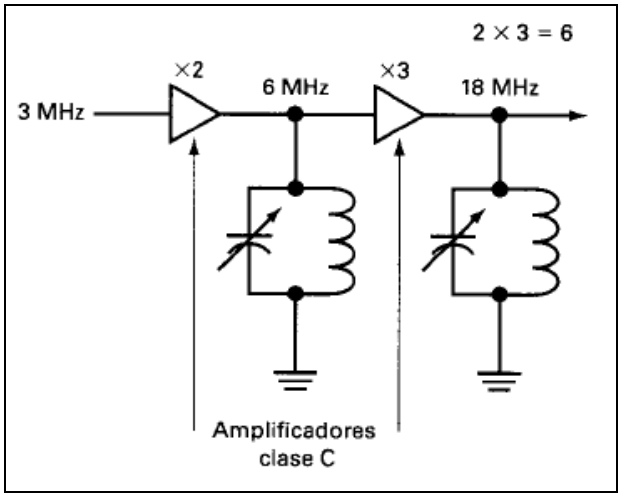

Figura 29 Multiplicadores de frecuencia con amplificadores clase C

Un circuito triplicador se conforma de la misma manera, excepto que el circuito sintonizado entra en resonancia al triple de la frecuencia de entrada. De este modo, el circuito sintonizado recibe un pulso de entrada por cada tres ciclos de oscilación que produce. Pueden elaborarse multiplicadores para incrementar la frecuencia de entrada en un factor entero de hasta aproximadamente 10. Cuando el factor de multiplicación aumenta, la potencia de salida del multiplicador se reduce. En la mayoría de las aplicaciones prácticas, los mejores resultados se obtienen con multiplicadores de valor 2 y 3.

Otra forma de visualizar la operación de los multiplicadores de frecuencia clase C es recordar que el pulso de corriente no senoidal es rico en armónicas. Cada vez que ocurre el pulso se generan las armónicas segunda, tercera, cuarta, quinta, y demás. El propósito del circuito sintonizado activo en el colector es actuar como filtro para seleccionar la armónica deseada.

En muchas aplicaciones se requiere un factor de multiplicación mayor que el que puede conseguirse con una sola etapa de multiplicación. En estos casos se conectan dos o más multiplicadores en cascada. El factor de multiplicación total es el producto de los factores de multiplicación de las etapas individuales.

7.6. IMPLEMENTACIÓN DEL TRANSMISOR FM /VHF

Se construyó un transmisor FM de 145.030MHz. El circuito implementado es una adaptación del utilizado por el Proyecto Argentino LU GLOBO [11]. El módulo adoptado es justamente el módulo transmisor, aunque se debieron realizar algunos ajustes en los componentes para sintonizar la frecuencia de salida en 145.030MHz.
Dado que la potencia a la salida del transmisor es de 100mW, se debe amplificar la misma hasta llevarla a 1W aproximadamente con una etapa de potencia.

Adicionalmente fue necesaria una etapa de acondicionamiento de la señal de audio anterior a la etapa transmisora, pues la señal de audio de salida de la placa moduladora era muy débil.

A continuación se detallan los circuitos implementados.

7.6.1. CIRCUITO PREAMPLIFICADOR DE AUDIO

A la salida de la placa moduladora se implementó el siguiente circuito:

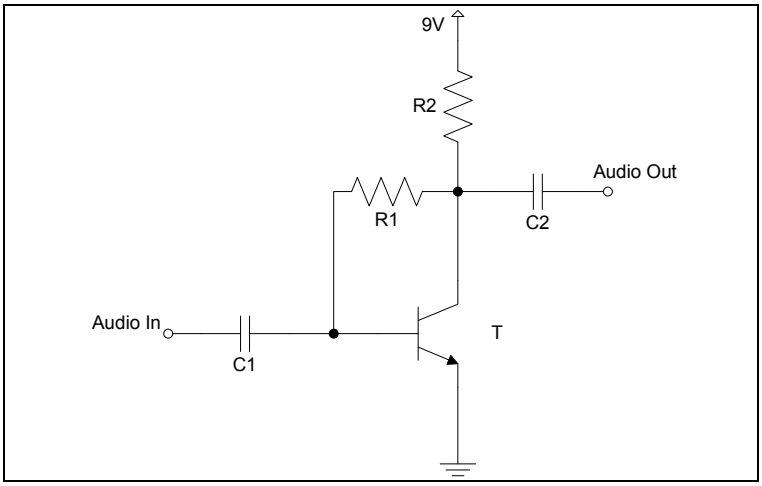

Figura 30 Esquema del Preamplificador

Sin embargo, la ganancia no mejoró lo suficiente como para aportar la potencia mínima necesaria a la entrada de las etapas amplificadoras.

Los valores de los componentes son:

- $C1 = 0.1\mu F$
- $C2 = 0.1\mu F$
- $R1 = 100k\Omega$
- $R2 = 10k\Omega$
- T = BC546

7.6.2. CIRCUITO TRANSMISOR

El transmisor funciona en la banda de 145.030 MHz y es alimentado por una batería de 9V. Según el esquema de la Figura 31 el circuito consta básicamente de cuatro etapas. En la primera etapa se tiene un oscilador LC controlado por un cristal en sobretono (hasta TRIM1), la cual provee una frecuencia de oscilación de 145.030MHz.

Las siguientes etapas corresponden a etapas amplificadoras, donde se produce además un filtrado de las señales espúreas. Al final de la última etapa se produce la adaptación de impedancias entre el transmisor y la antena. Adicionalmente el circuito cuenta con PTT ("Push to talk") lo cual permite el ahorro de energía.
En síntesis el funcionamiento del transmisor es el siguiente:

La señal del oscilador está generada por T1 (BF259) en configuración Colpitts y controlada por el cristal X1. El transistor T1 actúa como amplificador, y junto con el divisor capacitivo formado por C3 y C4 se encarga de proveer la realimentación positiva de voltaje necesaria para generar las oscilaciones (Criterio de Barkhausen).
La frecuencia de oscilación queda determinada por la frecuencia de resonancia del circuito sintonizado paralelo (también conocido como circuito tanque porque se comporta como un depósito para almacenar energía) compuesto por los capacitores C3 y C4 y la bobina L1. Esta última tiene un núcleo de ferrite ajustable, lo que permite ajustar la frecuencia de oscilación:

$$f_o = \frac{1}{2\pi\sqrt{L_1\left(\frac{C_3 C_4}{C_3 + C_4}\right)}}$$

El cristal utilizado (X1) es de 36MHz, el circuito tanque actúa como un filtro pasabanda, permitiendo que el cristal oscile en su cuarto armónico (144 MHz), obteniéndose así una frecuencia de oscilación más estable. Con el ajuste adecuado de la bobina L1 se logra una frecuencia de oscilación de 145.030MHz.

Las resistencias R4, R5 y R6 proveen la polarización deseada al transistor, mientras que los capacitores C1, C2 y C6 sirven a efectos de filtrar el ruido del circuito. La modulación en frecuencia se realiza por medio del diodo varicap (BB112), el mismo transforma los cambios en la amplitud de la señal modulante a cambios en la frecuencia de oscilación. Cuando no hay señal de audio las resistencias R2 y R3 proveen cierta polarización al varicap, la que produce una capacitancia nominal (y por tanto la frecuencia de oscilación de la portadora). Cuando se aplica una señal de audio, el voltaje de la señal modulante externa produce cambios en la polarización inversa, lo cual cambia la capacitancia del diodo y como consecuencia cambia también la frecuencia de oscilación. Los cambios positivos de la señal modulante incrementan la polarización inversa sobre el varicap, la cual disminuye su capacitancia e incrementa la frecuencia de oscilación. Al contrario, los cambios negativos de la señal modulante disminuyen la frecuencia de oscilación.

La inductancia L2 permite desacoplar el colector de T1 de la fuente a la frecuencia de oscilación, permitiendo así que se produzcan las oscilaciones, mientras que TRIM1 sirve para acoplar la primera etapa con la segunda.

En la segunda etapa se efectúa una amplificación de la señal por medio del transistor T2 (MRF901). Luego de que la señal es amplificada se filtra el armónico de interés (145.030MHz) mediante el filtro compuesto por TRIM2, TRIM3, L3 y C10. Las resistencias R7, R8 y R9 proveen la polarización adecuada al transistor.

En la siguiente etapa se amplifica nuevamente la señal mediante el transistor T3 (BFR96) y se filtra nuevamente el armónico correspondiente a 145.030 MHz mediante el filtro compuesto por C13, TRIM4, TRIM5 y L4.

La última etapa es idéntica a la anterior, amplificándose nuevamente la señal. Luego de en esta etapa se produce una adaptación de impedancias entre la salida y la antena (50 ohm).

Los chokes RFC1, 2, 3 y 4 impiden que la señal de RF afecte la fuente

Para limitar el consumo de energía y aprovechar al máximo el tiempo de vida de la batería el circuito cuenta con PTT, el cual consiste en el transistor T7 (BC327). Cuando se dispone de una señal a transmitir el microcontrolador envía un pulso a la base del transistor energizando así las etapas dos, tres y cuatro del circuito permitiendo la transmisión.

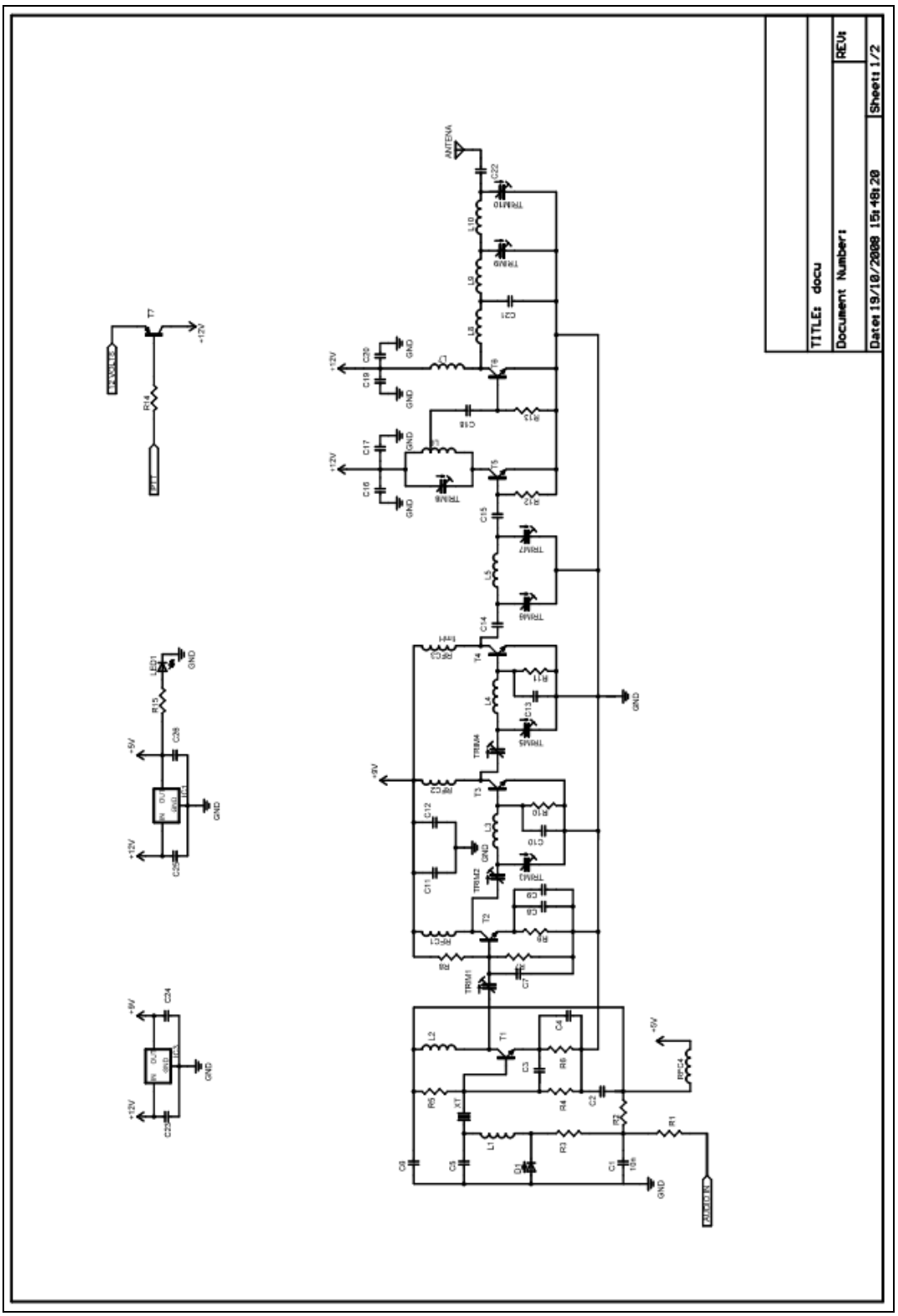

Figura 31 Esquemático del transmisor

Lista de componentes

Componente	Valor/Modelo
R1, R6, R7, R12, R13	1kΩ
R2, R4	12kΩ
R3	47kΩ
R5	15kΩ
R8	10kΩ
R9	10Ω
R10, R11	270Ω
C1, C9, C12, C16, C19, C22	10nF
C2, C6, C8, C11, C18, C20	1n0F
C3,C5	2p7F
C4, C7, C15, C18	33pF
C10, C13	22pF
C14	8p2F
C21	15pF
TRIM1, 7	10-50pF
TRIM2, 3, 4, 5, 6	5-20pF
TRIM8	2-22pF
TRIM9, 10	40pF
T1	BF254
T2	MRF901
T3, T4	BFR96
T5	2N2222
T6	2N4427
T7	BC327
X1	36MHz
D1	BB112

Tabla 5 Lista de componentes del transmisor

Bobinas:

Las bobinas L1 a L4 se construyeron con un diámetro interno de bobinado de 5mm y utilizando alambre esmaltado de 0.6mm de diámetro. En todos los casos las espiras se bobinaron juntas (sin espaciado entre espiras).

- L1: 15 espiras en núcleo roscado de ferrite.
- L2: 12 espiras en núcleo de aire.
- L3 y L4: 6 espiras en núcleo de aire.
- L5: 5 espiras en núcleo de aire.

Las bobinas L6 a L10 se construyeron de 5 espiras en núcleo de aire con un diámetro interno de bobinado de 4mm y largo de 7.5mm, utilizando alambre esmaltado de 0.7mm de diámetro.

7.6.3. AMPLIFICADOR DE SALIDA

Con el objetivo de obtener una potencia de salida de 1W se construyó un amplificador de potencia, el cual se acopló a la salida del transmisor. El mismo está compuesto por un transistor 2N2222 como driver y un transistor 2N4427 como amplificador de potencia, la alimentación es de 12V. A la salida del amplificador se cuenta con una serie de filtros de adaptación.

El circuito es el de la Figura 32, este modelo de amplificador fue adoptado del kit "1 Watt FM transmitter for the 2 mtr VHF band", de Dick Smith Electronics [12].

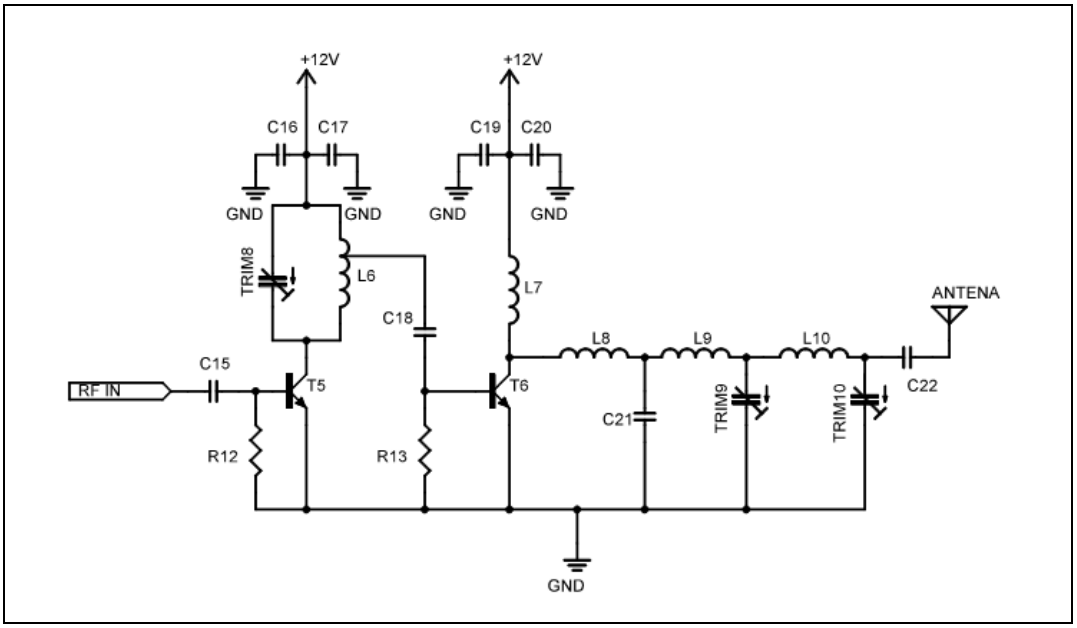

Figura 32 Esquema del amplificador

7.6.4. CONSTRUCCIÓN Y AJUSTE

Los primeros prototipos se construyeron mediante placas de circuitos impresos (PCB, por Printed Circuit Board), diseñando el layout de los mismos con el programa Eagle y luego imprimiendo la placa con percloruro de hierro.

Sin embargo, el prototipo final se construyó dibujando las islas necesarias en el PCB con un marcador y luego aislándolas del resto del cobre con la ayuda de un taladro. De esta forma se evitaron los agujeros para la instalación de los componentes, logrando una mayor rigidez mecánica y mayor flexibilidad a la hora de cambiar componentes. Tanto el transmisor como el amplificador se instalaron en el mismo PCB.

Para el diseño y la construcción se trató de respetar los siguientes criterios de diseño en radiofrecuencia:

- Identificar las señales de RF y DC. Tratar de mantener la señal de RF en una línea recta y lo más corta que sea posible, si la pista que lleva la señal de RF se cruza o da muchas vueltas el circuito puede resultar inestable. Las pistas de las señales DC no son críticas, por eso conviene dejar su diseño para el final.
- Dividir el circuito en módulos bien definidos (ej: oscilador, amplificadores, etc.) para luego aislarlos fácilmente. La construcción en módulos mejora la estabilidad en RF y hace que los módulos individuales sean más fáciles de construir y testear. Esto permite también realizar pequeños cambios en cada módulo sin tener que alterar todo el circuito.
- Mantener las etapas que manejan distintas frecuencias bien separadas para evitar el acoplamiento entre etapas.
- Usar una cara de la placa de cobre enteramente como plano de tierra. Esto asegura mayor estabilidad y desempeño de la señal de RF.
- Mantener las entradas y salidas de cada etapa (y del circuito entero) bien separadas.
- Blindar cada módulo.

- Llenar con planos de tierra toda la parte de la placa que no lleve pistas. Utilizar una tierra común para todo el circuito, y conectar las tierras de las dos caras de la placa en varios puntos.
- Tratar de mantener las señales de RF y DC en lados opuestos de la placa para evitar que la señal DC se vea afectada por los campos generados por la RF.
- Evitar acoplamiento indeseado entre etapas, para esto aislar las bobinas y cuando sea posible ubicar las bobinas con núcleo de aire adyacentes perpendiculares entre sí de forma de minimizar las inductancias mutuas.
- Las patas de los componentes deben ser lo más cortas que sea posible
- Es conveniente que el circuito resultante sea compacto, pero si los puntos anteriores llevan a un diseño más amplio es más importante respetarlos.

El ajuste del circuito requirió de cierta iteración, y se realizo con la ayuda de analizador de espectro, osciloscopio digital y medidor de potencia. Las etapas del circuito se fueron armando y ajustando por módulos, a medida que se soldaban los componentes. En primer lugar se ajustó el transmisor con el criterio de obtener máxima potencia y el menor contenido de señales espúreas y luego se acopló y ajustó el amplificador siguiendo los mismos criterios.

La frecuencia de oscilación se ajustó mediante la bobina de núcleo variable (L1), mientras que los trimmers se ajustaron con el criterio de maximizar la potencia de salida y la eliminación del contenido espúreo en los múltiplos de la frecuencia del cristal.
Finalmente se obtuvo una señal de salida de 145.030MHz de frecuencia, con una potencia que osciló entre los 100mW y los 800mW.

El layout y el esquemático del transmisor y amplificador se encuentran en el CD adjunto bajo el nombre Transmisor.

Por información adicional sobre diseño y construcción de circuitos en radiofrecuencia consultar [13]

7.7. CONCLUSIONES Y TRABAJO A FUTURO

La señal generada por el oscilador resultó ser estable y libre de frecuencias espúreas. Sin embargo, el desempeño del circuito completo no fue el esperado dado que la potencia de salida no llegó a estabilizarse en un valor aceptable, oscilando entre los 100mW y 800mW. Se realizaron varios intentos para mejorar la situación implementando nuevos prototipos con mejoras respecto a los puntos señalados en los criterios de diseño de RF y agregando etapas de amplificación, pero lamentablemente no se llegó a obtener la potencia de salida deseada.

Entre las posibles causas del bajo desempeño del circuito se podrían encontrar la falta de experiencia, así como la mala calidad de determinados componentes, principalmente los trimmers, lo cual derivó en que el ajuste del circuito resultara bastante tedioso puesto que las variaciones en la señal al variar la capacidad de los trimmers eran muy abruptas.

Cabe destacar que la implementación de este transmisor constituyó un primer paso en el aprendizaje para el diseño y desarrollo de circuitos en radiofrecuencia, puesto que no se contaba con conocimiento alguno en esta área al inicio del proyecto. Por lo tanto se considera que los resultados obtenidos no son menores, si bien el desempeño no es el óptimo sería un tanto ambicioso lograr diseños perfectos desde el primer proyecto, en especial trabajando en un área tan compleja como lo es la radiofrecuencia.

Como trabajo a futuro se sugiere la implementación del transmisor y amplificador mediante el uso de chips integrados lo cual evita en gran parte el proceso de ajuste y por ende mejora la estabilidad.

8. VALIDACIÓN DEL PROTOTIPO FINAL

En esta sección se pretende mostrar los procedimientos realizados para validar los módulos más importantes que componen el prototipo final. Dichos módulos son:

1. Modulador
2. Programa Principal
3. Transmisor

8.1. VALIDACIÓN DEL MODULADOR

Pruebas realizadas para validar el modulador:

1. Encender el modulador sin conectar el puerto serie. En este caso se debe recibir un mensaje del tipo "*No serial data"* cada 30 segundos.
2. Encender el modulador con el puerto serie conectado. Cuando se envían datos por el puerto serie a 4800 bps se debe encender el led azul que indica que se están recibiendo datos correctamente. En el caso que algún dato llegue corrupto se enciende el led rojo y permanece encendido hasta que se vuelva a recibir algún dato correctamente.
   Si los datos recibidos tienen el formato descripto la sección 6.2.3 (flag de inicio: 02H y flag de fin: 04H) al recibir el flag de fin se apagan los leds (si había alguno encendido) y se enciende el led verde correspondiente al PTT indicando que se está generando el audio, el cual se puede escuchar si se conecta el modulador a un PC.

8.2. VALIDACIÓN DEL PROGRAMA PRINCIPAL

Se realizaron dos tipos de pruebas:

1. Se escribió un programa en C (hecho para linux) el cual ejecuta el programa principal recibiendo los datos del GPS a través de la lectura de un archivo de texto y los datos que simulan la salida de los sensores a través de funciones que devuelven valores “tontos”. Este programa imprime en pantalla información como: datos a enviar al modulador, salida de funciones, mensajes informativos para identificar que parte del programa se está ejecutando, etc.
   Editando el archivo de texto es posible simular la trayectoria del globo verificando así el correcto funcionamiento de la cerca electrónica y de los mensajes que se transmiten.
   Este programa se encuentra en el CD adjunto a esta documentación.
2. Pruebas de campo. La primera prueba consiste en ensamblar todos los módulos y realizar un recorrido de un área predefinida. Para esto se debe modificar los límites de la cerca electrónica de manera tal que se cubra el área a recorrer. Para realizar esta prueba se debe contar con una estación donde se reciban los datos para así verificar que los mismos lleguen correctamente. Con esta prueba se logra observar el comportamiento de la plataforma completa.
3. Para comprobar la estabilidad a lo largo del tiempo se debe realizar una segunda prueba de campo. La misma consiste en dejar la plataforma completa encendida durante 48 horas (conectada a una fuente de corriente en lugar de la batería), y recibiendo los datos desde una estación fija.

8.3. VALIDACIÓN DEL TRANSMISOR

Las pruebas a realizar en este caso van de la mano con las pruebas de campo de la validación del programa principal y se pueden dividir en dos partes:

1. Integrar el transmisor al resto de la plataforma y realizar un recorrido de un área predefinida. Con esta prueba se logra tener una idea del alcance, si bien el comportamiento no es el mismo en cuanto a altura, temperatura y línea vista, de esta forma se podrá estimar a grandes rasgos la estabilidad del transmisor.
2. Para comprobar la estabilidad a lo largo del tiempo se debe recurrir al punto 3 de la parte anterior.

9. CONCLUSIONES

Mediante este proyecto se intentó dar los primeros pasos hacia la construcción de un satélite experimental. Si bien los ensayos con globos satelitales se vienen realizando desde hace muchos años en otros países, esta fue la primera experiencia para Uruguay.

Es de destacar que los resultados de este proyecto fueron el mérito del trabajo en conjunto de dos grupos, lo cual significó una dificultad extra a la hora de coordinar tareas. Sin embargo la experiencia resultó positiva, pues se construyó un ambiente de trabajo muy ameno e incentivador.

El desarrollo del hardware y del software de las plataformas implementadas para ambas liberaciones aportó conocimientos y experiencia útiles para el diseño y construcción de futuros prototipos con más funcionalidades, siendo esta documentación una referencia para el desarrollo de los mismos.

10. **REFERENCIAS**

[1] http://lostrego.uvigo.es/articulos/radiopaq.html – *Protocolo X.25 y AX.25.* Visitada en 11/08

[2] http://es.wikipedia.org/wiki/HDLC – *Protocolo HDLC.* Visitada en 11/08

[3] http://www.tapr.org/aprs_working_group.html - *Especificación del protocolo APRS*

[4] http://www.digigrup.org/aprs/quees.htm - ¿Qué es APRS? Autor: Toni Planas. Visitada en 11/08

[5] http:// www.monografias.com. - *Descripción del PIC 16F877,* Visitada en 2/08

[6] www.redeweb.com/_txt/articulos/800303.pdf – *Protocolo de programación de dispositivos Microchip PIC*. Visitada en 11/08

[7] www.dte.us.es/ing_inf/ins_elec/temario/Tema%207.%20Convertidores%20D-A.pdf - *Conversores D/A*, Visitada en 2/08.

[8] http://www.tapr.org/pub_ax25.html - *Especificación del protocolo AX.25*

[9] http://n1vg.net/packet/index.php - *Detalles del modo Packet a 1200 bps*

[10] http://www.microchip.com – *Página principal del fabricante de los microcontroladores PIC*

[11] http://www.lu5egy.com/ - *Página personal de uno de los integrantes del proyecto LUGLOBO*. Visitada en 11/08

[12] http://www.dse.com.au – *Página principal de Dick Smith Electronics.* Visitada en 11/08

[13] RF Circuit Design – Chris Bowick. Segunda Edición, Octubre 2007. ISBN-13: 9780750685184

[14] http://www.datasheetcatalog.org/datasheet/nationalsemiconductor/DS005516.PDF – *Hoja de datos del sensor de temperatura LM35*- Visitada en 11/08

[15] http://iie.fing.edu.uy/twiki/bin/view.cgi/Satelite/GloboSat01Informe - *Informe de la liberación de GS01*. Visitada en 11/08

[16] http://f6cte.free.fr/index_anglais.htm – *Página principal del programa Multipsk.* Visitada en 11/08.

[17] Antenna Basics, Christof Rohner (Rohde&Schwarz), Noviembre 1999

[18] http://fermi.la.asu.edu/w9cf/yagipub/index.html - *Página principal de la aplicación "Yagi Modeler".* Visitada en 2/09.

11. **ANEXO I – GLOBOSAT01**

El 24 de Abril se liberó el primer Globosat desde la Base Aérea de Santa Bernardina. La carga útil consistió en la plataforma de telemetría, un transmisor, una cámara de video y una antena tipo monopolo de $\lambda/4$ con 3 elementos de plano de tierra.

Los datos de telemetría consistieron en: temperatura interna (dentro de la caja), temperatura externa (fuera del globo) y posición. Las temperaturas fueron proporcionadas por sensores de temperatura y la posición por un GPS. La plataforma de telemetría era la encargada de generar los paquetes APRS en base a las temperaturas obtenidas de los sensores y de los datos del GPS y enviarlas al transmisor. El elemento principal de la misma fue el microcontrolador PIC16F877A, el cual manejó los datos de telemetría.

Se debió tener en cuenta en todo momento que el peso de la carga útil del globo no superara los 1.5Kg, por lo tanto la principal restricción fue realizar la plataforma lo más liviana posible.

La plataforma de telemetría estuvo compuesta por:

- Microcontrolador PIC16F8774
- 2 sensores de temperatura, modelo LM35D
- GPS marca Trimble, modelo Lassen IQ
- Conversor A/D tipo R2R
- PTT
- Quematanza
- Leds indicadores

La frecuencia de transmisión se fijó en 145.030MHz. El transmisor utilizado en esta etapa consistió en un handy comercial y la potencia entregada por el mismo fue de aproximadamente 1W.

Se transmitieron dos tipos de mensajes según el protocolo APRS. El primero consistió en un reporte de posición con timestamp conteniendo los datos de telemetría y el segundo en un boletín identificando el globosat. El reporte de posición se transmitió con una cadencia de 5 segundos, mientras que el boletín se transmitió con una cadencia de 10 segundos.

El formato del reporte de posición fue el siguiente:

/092345h4903.50N/07201.75WO/A=123456,Ti=-25,Te=+35

Donde:

- 092345h: hora en hhmmss
- 4903.50N: latitud en formato GGMM.mm (grados, minutos, y centésimas de minuto)
- 07201.75W: longitud en formato GGMM.mm (grados, minutos, y centésimas de minuto)
- O indica que se trata de un globo
- A = 123456: altura en pies
- Ti = -25: temperatura interna
- Te = +35: temperatura externa

El formato del boletín fue el siguiente:

CX0CFI: Baliza de CX0CFI
QSL a cx0cfi@fing.edu.uy

TNX, 73

El primer campo identifica la baliza, el segundo proporciona una dirección de correo electrónico donde reportar la recepción de datos y el tercero corresponde a “gracias” y “saludos”.

11.1. HARDWARE

Diagrama de Bloques de la Plataforma

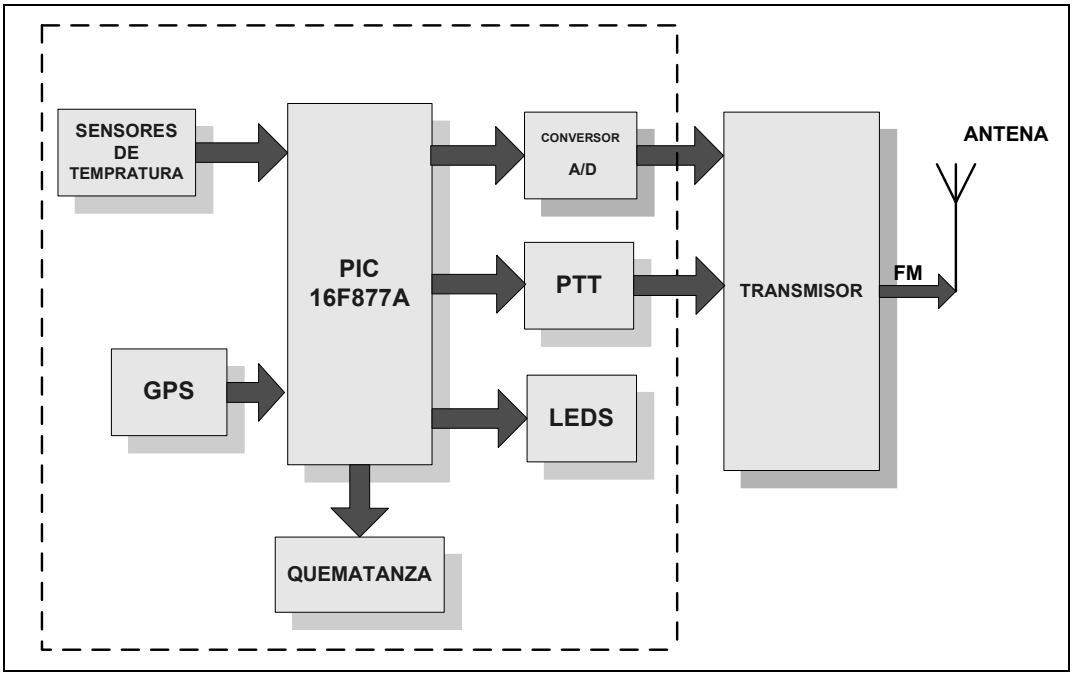

Figura 33 Diagrama de bloques de la plataforma de telemetría del GS01

Como se puede apreciar en la Figura 33, la plataforma implementada está compuesta por los bloques que se encuentran dentro de la zona punteada, los mismos corresponden a 7 módulos que realizan tareas independientes:

- Módulo 1 - Sensores de Temperatura
- Módulo 2 - GPS
- Módulo 3 - Microcontrolador PIC 16F877A
- Módulo 4 - Conversor digital analógico
- Módulo 5 - PTT para comunicación con el transmisor
- Módulo 6 - LEDs indicadores
- Módulo 7 - Quematanza

A continuación se detallarán los módulos mencionados arriba.

Módulo 1 - Sensores de Temperatura

El modelo de sensor elegido fue el LM35DH-ND de National Semiconductor. Esta elección se basó en varios factores. En primer lugar el amplio rango dinámico que va de -55ºC a +180ºC, en segundo lugar la linealidad de la señal de salida, la cual equivale a 10mV/ºC, por lo tanto el voltaje de salida puede describirse mediante la siguiente expresión: $Vo = 0.01T$ donde Vo esta dado en voltios y T en grados Celsius. Por último, la precisión que se logra con este sensor es de ± 1⁄4˚C a temperatura ambiente y ± 3⁄4˚C en todo el rango de temperaturas (−55ºC a +150˚C) sin necesidad de calibración externa.

El conexionado de los sensores fue el sugerido en la hoja de datos [14]:

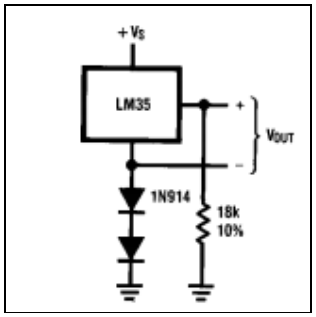

Figura 34 Conexionado de los sensores

Donde Vs corresponde a un voltaje de referencia de 5V y Vout es la señal que se conecta al conversor analógico digital del PIC.

El sensor de temperatura externa se conectó al bit 0 del Puerto A del PIC (pin 2) mediante un cable blindado de 0.5 metros para evitar ruidos en la señal, mientras que la salida del sensor de temperatura interna (el cual se encontraba fijo en la placa) se conectó al bit 1 del mismo puerto (pin 3). Para el procesamiento de la señal se hizo uso del conversor analógico digital que viene integrado en el PIC.

Módulo 2- GPS

El modelo de GPS elegido fue el “Lassen® iQ GPS module” de Trimble, que viene con antena incorporada y es de muy bajo consumo. La alimentación es de 3.3V a 5V.

Los datos son generados por el mismo según el protocolo MNEA 0183, y son recibidos por el puerto serie del PIC (pin 26) mediante el estándar RS232 a una velocidad de 4800 baudios.

Módulo 3 - Microcontrolador PIC 16F877A

La descripción del microcontrolador se encuentra detallada en la sección 6.1.1 de esta documentación.

Módulo 4 - Conversor digital analógico

La descripción del conversor se encuentra detallada en la sección 6.1.2 de esta documentación.

Módulo 5 - PTT para comunicación con el transmisor

El conexionado entre la plataforma de transmisión y el transmisor se hizo a través de 3 señales: la señal de audio, la tierra y el PTT ("Push to talk"). El PTT es un método para hablar en líneas half-duplex de comunicación, cuando la señal PTT baja a tierra, el transmisor libera la portadora y se transmite la señal de audio.

Para hacer posible la comunicación descrita se debió generar una señal que bajara a tierra unos instantes antes de que comenzara la transmisión y subiera a Vcc (5 voltios) una vez terminada la misma. Para lograr el comportamiento deseado se utilizó un optoacoplador, este es un dispositivo que se compone de un diodo LED y un fototransistor, de manera de que cuando el diodo LED emite luz, ilumine el fototransistor y conduzca. Con este dispositivo se logra aumentar la velocidad de conmutación con respecto a la de un conmutador normal. El circuito que se implementó es el siguiente:

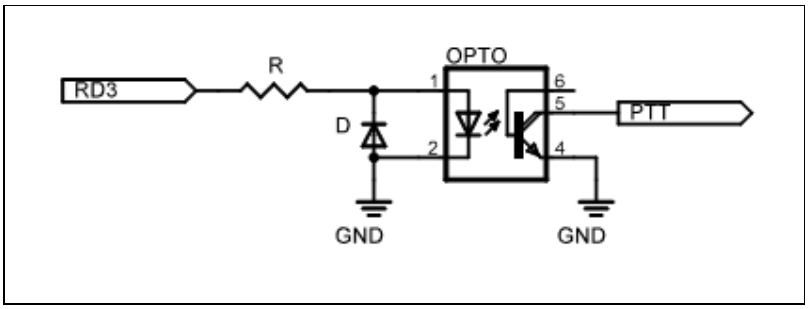

Figura 35 Circuito PTT

La señal RD3 es generada por el PIC (pin 22), la misma sube a Vcc unos instantes antes de iniciar la transmisión y baja a tierra al finalizar, de esta forma cuando RD3 está en nivel alto el fototransistor conduce y la señal PTT se pone a tierra.

Con este comportamiento se asegura que el transmisor estará encendido solamente durante los intervalos de transmisión, reduciendo así el consumo.

Módulo 6 - LEDs indicadores

Se utilizaron 3 LEDS indicadores de color rojo, amarillo y verde. La función de los mismos es la siguiente:

- LED amarillo - Se enciende cada vez que comienza la transmisión de datos. Se conecta al puerto RD2.
- LED rojo - Se enciende cuando comienza una transmisión serie. Se conecta al puerto RD0.
- LED verde- Se enciende si ocurren errores en la transmisión. Se conecta al puerto RD1.

El conexionado de los LEDs es el siguiente:

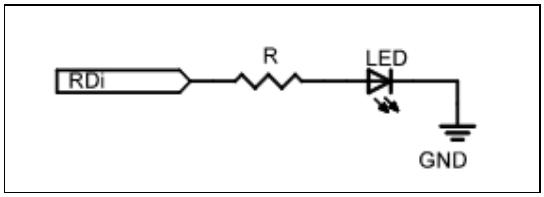

Figura 36 Conexionado LEDs

Donde RDi corresponde al puerto RD0 para el LED rojo, RD1 para el LED verde y RD2 para el LED amarillo. La resistencia R es de $2.2K\Omega$.

Módulo 7 – Quematanza

Este módulo fue desarrollado por el grupo HWGloboSats, el circuito del mismo estuvo compuesto por un buffer modelo 74HCT14D y un transistor mosfet modelo 05N03LA.

A continuación se muestran los esquemáticos de los módulos diseñados y el layout de los mismos así como una foto de la plataforma final. Los archivos correspondientes al layout y al esquemático se encuentran en el CD adjunto bajo el nombre GS01.

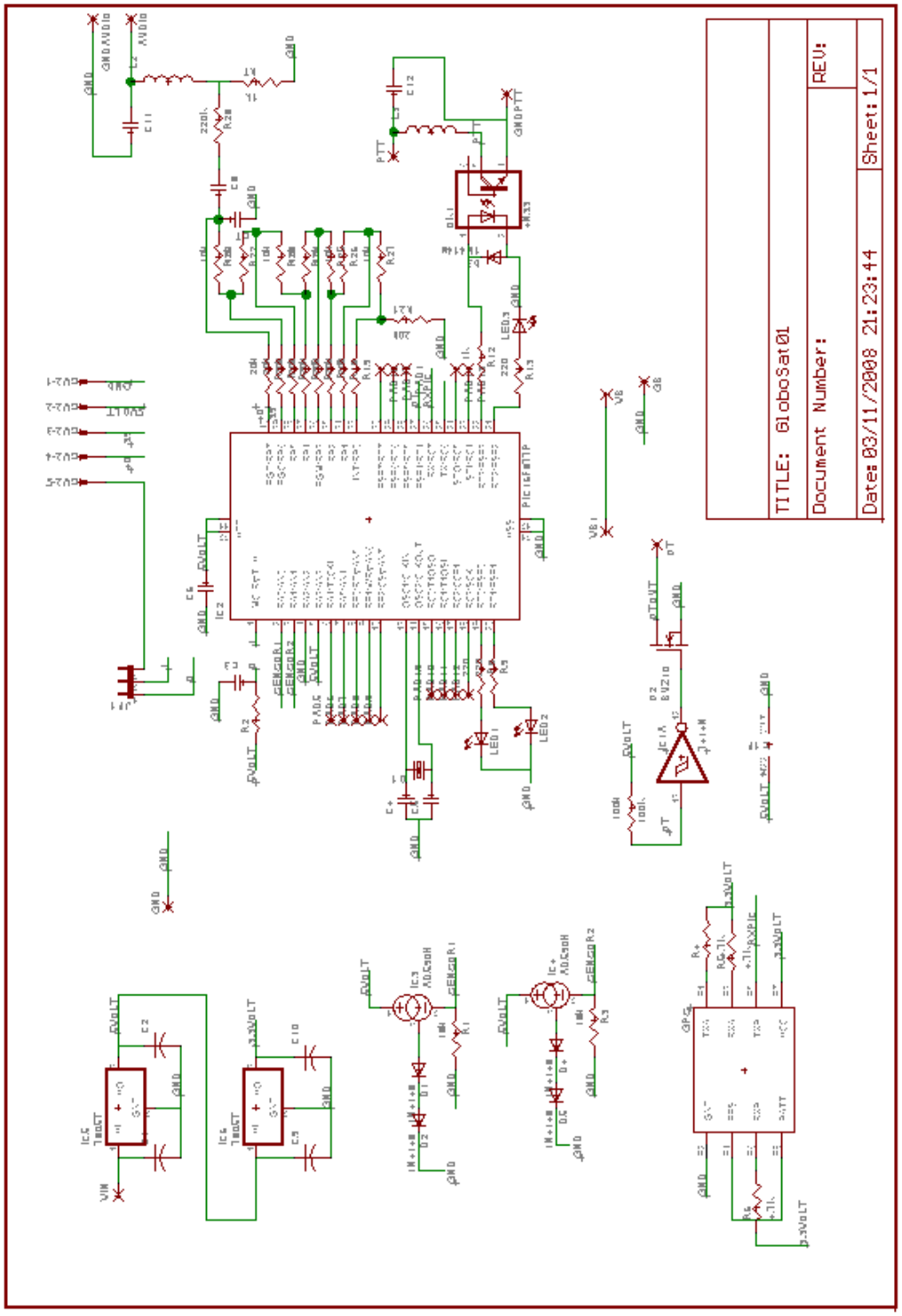

Figura 37 Esquemático de la plataforma de telemetría del GloboSat01

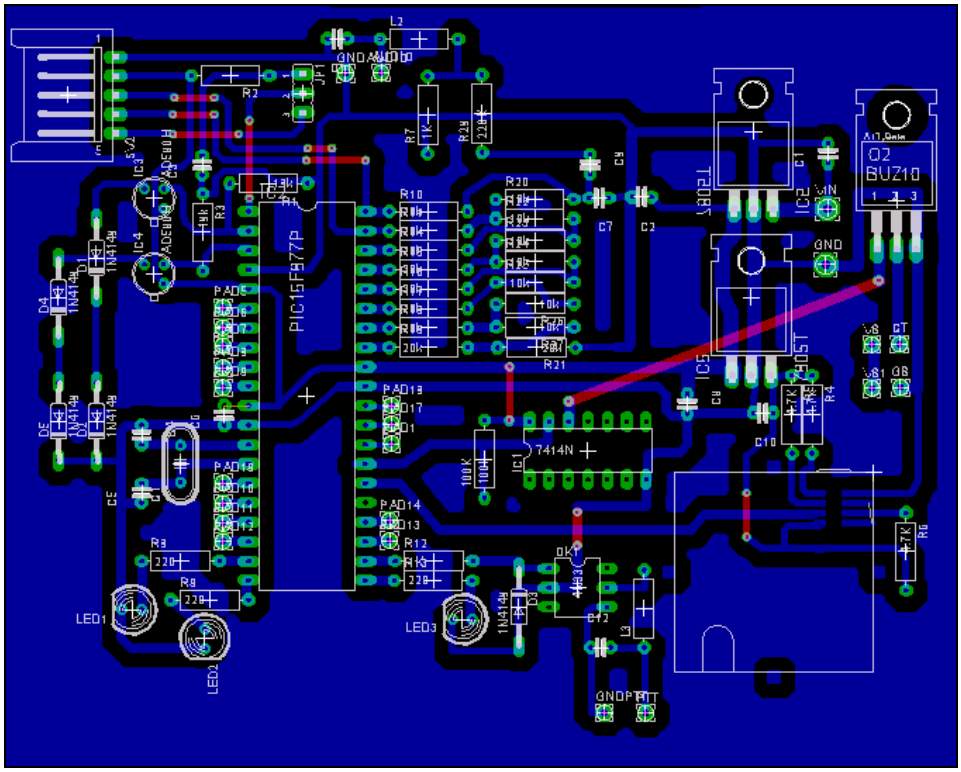

Figura 38 Layout de la plataforma de telemetría del GloboSat01

La plataforma final se construyó en tres PCBs distintos:

1. Modulador y sensores de temperatura
2. Quematanza
3. GPS

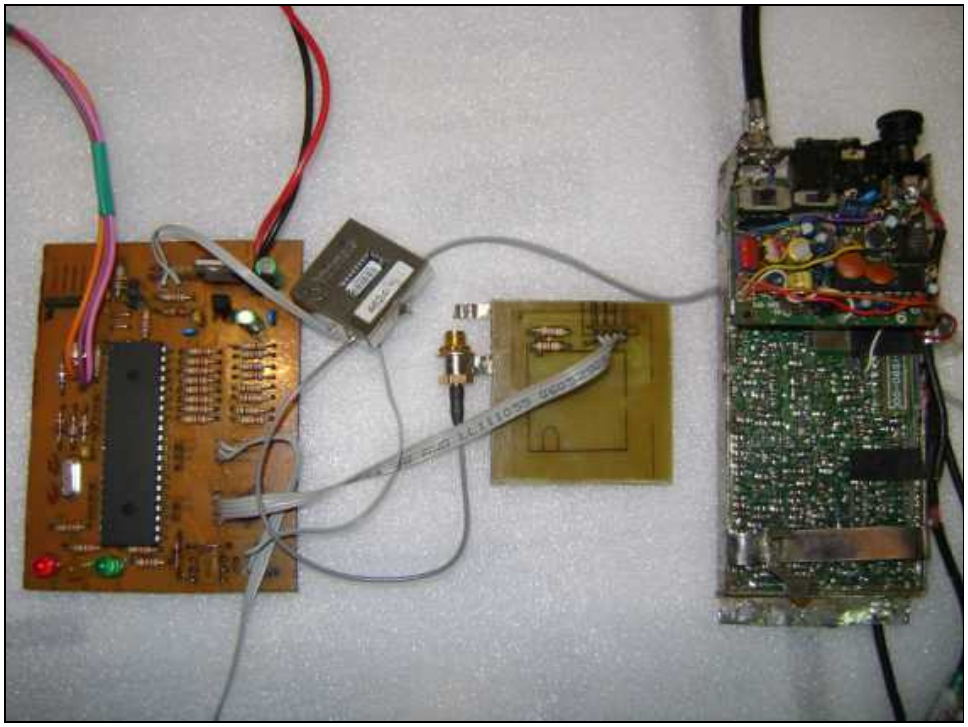

Figura 39 Plataforma de telemetría

11.2. Desarrollo del Software de control para la Telemetría

En esta sección se dará una explicación sobre cómo funcionan los principales módulos del firmware en el PIC16F877A para GLOBOSAT01.

El código fuente se puede consultar en el CD que viene junto con esta documentación. Cabe aclarar que el mismo fue escrito en su totalidad por los integrantes del grupo en lenguaje assembler, no se tomó código prestado de otras fuentes, exceptuando una rutina de multiplicación que fue creada con una herramienta generadora de código assembler para este propósito.

11.2.1. MÓDULO PRINCIPAL

El módulo principal del firmware consiste en un loop en el cual se hace lo siguiente:

- Espera a recibir los datos del GPS a través del puerto serie.
- Luego de 5 segundos, si no recibió ningún dato, arma una trama con un mensaje de error y la trasmite.
- Si los datos fueron recibidos correctamente, se procesa esa información para verificar que el globo se encuentra dentro de los límites preestablecidos (cerca electrónica).
- Se obtienen los datos de los sensores de temperatura y se arma la trama en el formato explicado al comienzo de este apéndice.
- Cada 5 mensajes envía la baliza.
- Vuelve al inicio.

El código del módulo principal puede ser consultado en el código fuente bajo el nombre de “Programa principal”.

11.2.2. IMPLEMENTACIÓN DE LA CAPA FÍSICA (Bell202)

El código que implementa el protocolo Bell 202 es el mismo que está documentado en la sección sobre el software del capítulo 6 de la documentación, no hay variantes entre las dos liberaciones.

11.2.3. RUTINA DE TRANSMISIÓN

Al igual que la implementación de la capa física, la rutina de trasmisión es la misma tanto en el firmware de GLOBOSAT01 como de GLOBOSAT02, y se encuentra documentada en el capítulo 6.

11.2.4. PLAN DE VUELO

No estaba previsto para el primer prototipo, pero dada la efectividad en cuanto a los tiempos planteados al comienzo del proyecto, se decidió realizar una pequeña rutina para el plan de vuelo, conocida como "cerca electrónica". Consiste en tomar los datos de coordenadas recibidos por GPS y verificar que el globo se encuentre dentro de ciertos límites. Si se va de estos límites, se debe liberar la carga del globo y abrir el paracaídas. Para el primer prototipo, la cerca electrónica es un rectángulo definido por las latitudes mínima y máxima y por las longitudes mínima y máxima. Para evitar que se libere la carga debido a errores en los datos recibidos, se incrementa un contador cuando se constata que se está fuera de los límites. A las 5 lecturas consecutivas (configurable) que den fuera de los límites, se activa el dispositivo de liberación.

11.3. RESULTADOS DE LA LIBERACIÓN

Desde la estación terrestre se pudo recibir telemetría hasta el final de la trayectoria, cuando la cápsula cayó debajo del horizonte.

Aunque el día de la liberación el receptor GPS no captó señal satelital en ningún momento, se pudo hacer una estimación de la trayectoria en base a: el boletín de

meteorología de la Fuerza. Aérea, la dirección de recepción de la señal, datos sobre isotermas provistos por meteorología de la Fuerza. Aérea y cálculos de trayectoria basados en peso de la carga e inflado del globo.

Los rangos de temperatura recibidos oscilaron entre los -25ºC – 28ºC para la temperatura externa y entre los 0ºC – 28ºC para la temperatura interna. Se estima en base al tiempo de vuelo y al peso de la carga e inflado del globo que la altura máxima alcanzada rondó los 34.000 metros, lo que en base a las isotermas para dicha altura correspondería a una temperatura exterior de aproximadamente -60ºC. La diferencia entre la temperatura exterior mínima registrada y la estimada se debe a un error en la elección de los sensores. El rango de temperatura de los sensores utilizados (LM35D) es de 0ºC – 100ºC, por lo cual los datos de temperatura registrados por debajo de 0ºC no son de fiar.

Fue fundamental ir ajustando cuidadosamente la dirección de las antenas para poder decodificar la señal.

La señal en tierra fue decodificada con tres tipos de software: MixW, Multipsk y uno hecho a medida para Mac OS-X.

Otros radioaficionados reportaron recepción de datos desde ciudad de Treinta y Tres, o recepción de señal sin poder decodificarla desde Colonia

La cápsula se recuperó en las proximidades de la Quebrada de los Cuervos (Departamento de Treinta y Tres) gracias al reporte de las personas que la encontraron, habiendo recorrido una trayectoria de aproximadamente 200Km. La carga útil fue recuperada en perfecto estado.

A pesar de que el GPS no funcionó correctamente, se concluye que la liberación fue exitosa dado que se logró poner el equipo en el aire, recibir los datos transmitidos y recuperar la carga sin daños.

Información adicional así como material gráfico se puede encontrar en [15].

12. ANEXO II – GLOBOSAT02

La liberación del segundo GloboSat se llevó a cabo el 24 de Setiembre, nuevamente el lugar de partida fue la Base Aérea de Santa Bernardina, Durazno.

El objetivo de esta liberación fue poner a prueba la plataforma final diseñada en conjunto con el grupo HWGloboSats, así como medir los efectos de la radiación sobre memorias electrónicas. La placa conteniendo las memorias a testear fue diseñada y construida en el Instituto Politécnico de Grenoble, Francia. El layout y el esquemático de la misma se encuentran en el CD adjunto bajo el nombre GS02.

12.1. Carga útil

En esta ocasión la carga útil estuvo compuesta por:

- placa diseñada por el grupo HWGloboSats (conteniendo los sensores de temperatura y presión, GPS, microcontrolador Atmega 2860)
- plataforma de telemetría diseñada para el lanzamiento anterior
- DUT (Device Under Test) conteniendo las memorias a testear
- fuente switcheada también diseñada y construida por el grupo HWGloboSats
- batería de 14V
- handy marca Yaesu
- antena tipo monopolo de $\lambda/4$ con tres elementos de plano de tierra.

La frecuencia de transmisión se fijó en 145.030MHz. Si bien la idea fue utilizar el transmisor construido, esto no fue posible debido a la inestabilidad del mismo, debiéndose utilizar nuevamente un handy comercial.

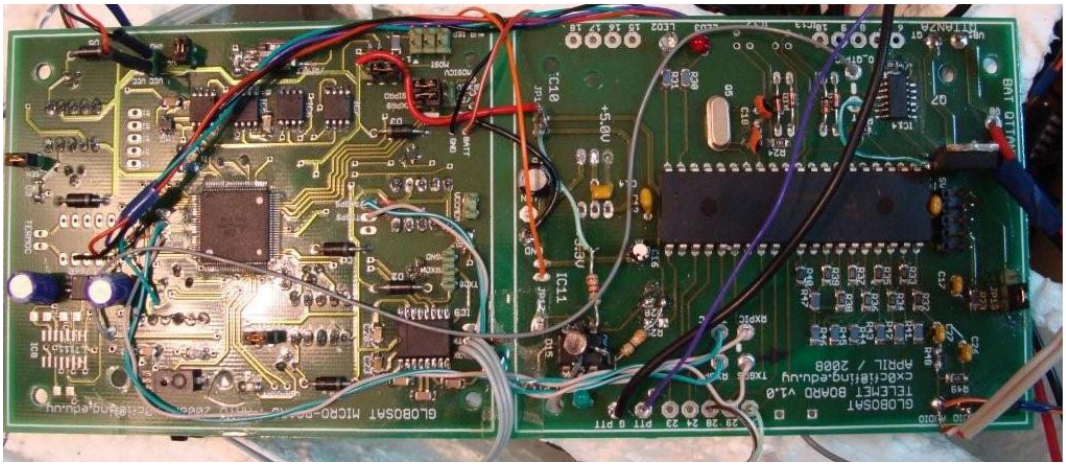

Figura 40 Plataforma de telemetría

12.2. Formato de las tramas

Al igual que en la liberación del GloboSat01, se transmitieron dos tipos de mensajes según el protocolo APRS, uno reportando posición con timestamp conteniendo los datos de telemetría y otro identificando el globosat. El reporte de posición se transmitió con una cadencia de 5 segundos, mientras que el boletín se transmitió con una cadencia de 50 segundos.

El formato del reporte de posición fue el siguiente:

/171941h3453.6987S/05609.6516WO/A=147,Ti=21,Te=-5,H=85,P=1040,UHX

La información que lleva esta trama es:

- Timestamp – hora GMT a la que se envió la trama, obtenida desde el GPS
- Latitud - en formato GGMM.mm (grados, minutos, y centésimas de minuto)
- Longitud - en formato GGMM.mm (grados, minutos, y centésimas de minuto)
- Altura – Luego de la longitud, dentro del comentario de la trama como se especifica en el protocolo APRS, debe estar en pies, la altura en metros

se obtiene del GPS y luego se multiplica por 3.28 para hacer la conversión.

- Datos de los sensores: temperatura interna y externa en grados Celsius, humedad en porcentaje y presión en hPa
- Estado del globosat: se representa con 2 letras. La primera indica si el globo sube ("U"), baja ("D"), se mantiene a altura constante ("=") o está llegando al suelo ("G"), tal como se explica en el plan de vuelo. La segunda letra indica si el globo está dentro ("O") o fuera ("X") de la cerca electrónica.

El formato del boletín fue el siguiente:

CX0CFI: Baliza de CX0CFI
QSL a cx0cfi@fing.edu.uy
TNX, 73

12.3. Resultados de la liberación

La base terrestre estuvo compuesta por un tandem de 2 Yagi de 5 elementos de alta ganancia y 2 laptops conectadas a un handy Yaesu para la decodificación de las tramas. Se logró recibir telemetría durante las 3 horas de vuelo, hasta que la capsula aterrizó en los alrededores de Molles de Godoy, Lavalleja. Se reportó recepción de datos por parte de radioaficionados desde Montevideo y desde la estación móvil de rastreo en Sarandí del Yí.

Tanto la información referente a la altura como a la hora (ambas procedentes del GPS) fueron recibidas sin problemas durante los primeros 10.000 metros de altura. Superada esta altura, ambos parámetros comenzaron a decodificarse con errores. Se logró constatar que la falla estuvo en el procesamiento de los datos en el GPS, dado que superada esta altura el GPS comenzó a reportar alturas a partir de los -10.000 metros. Esto afectó el procesamiento de los datos en el programa principal, siendo imposible detectarlo y por lo tanto corregirlo de antemano. Una vez detectado este problema se reconstruyeron las tramas corruptas.

Se obtuvieron los siguientes resultados

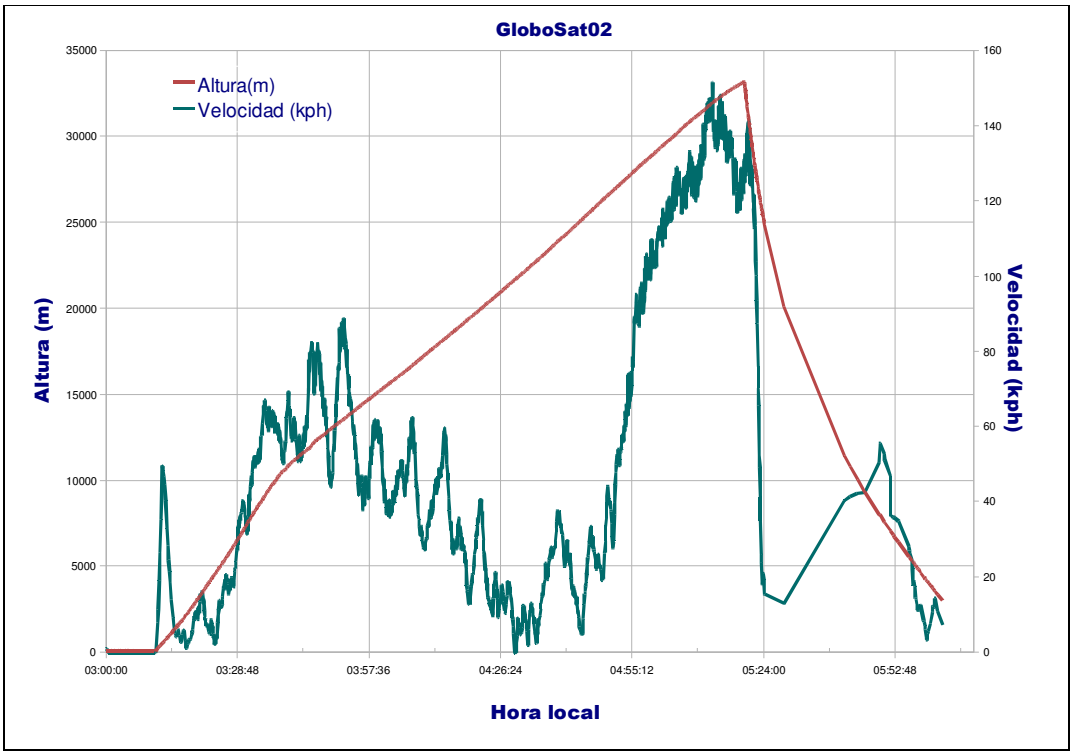

Figura 41 Resultados experimentales

Según la gráfica de la Figura 40 se observa que la altura máxima alcanzada por el GloboSat (en rojo) fue 33.172 metros. En verde se encuentra graficada la velocidad de los vientos.

Los resultados de esta liberación fueron exitosos dado que el software de control funcionó como se esperaba brindando una interfaz de comunicación robusta a la placa de hardware. Asimismo la plataforma de modulación que se reutilizó de la primera liberación también funcionó como se esperaba.

13. **ANEXO III – Estudio de Radioenlace**

En esta sección se estudiará el radioenlace entre el Globosat y las bases terrestres de Montevideo y Durazno. El objetivo de este estudio es estimar la potencia mínima del transmisor para poder recibir los datos correctamente en tierra. Para estimar dicha potencia se hará uso de la ecuación de Friis:

$$\frac{P_R}{P_T} = \left(\frac{\lambda}{4\pi d}\right)^2 G_T(\theta_T,\phi_T) G_R(\theta_R,\phi_R)\left(1-|\rho_T|^2\right)\left(1-|\rho_R|^2\right)$$

Donde:

- P_T es la potencia del transmisor
- P_R es la sensibilidad del receptor
- $\left(\frac{\lambda}{4\pi d}\right)^2$ son las pérdidas de propagación en el espacio libre
- $G_T(\theta_T,\phi_T)G_R(\theta_R,\phi_R)$ son las pérdidas por desapuntamiento
- $\left(1-|\rho_T|^2\right)\left(1-|\rho_R|^2\right)$ son las pérdidas por desadaptación

A continuación se analizarán más en detalle los factores de pérdidas.

13.1. Pérdidas de propagación en el espacio libre

Las mismas corresponden al término $\left(\frac{\lambda}{4\pi d}\right)^2$, donde λ es la longitud de onda a la frecuencia de trabajo $(145.030MHz)$ y d es la distancia entre la base terrestre y el GloboSat. Para calcular la distancia d se tomará el caso en que el globo se encuentra a mayor altura, pues es en ese punto donde se requerirá la mayor potencia de transmisión. Este cálculo es muy relativo, ya que depende mucho de la trayectoria que siga el GloboSat, y esta última depende de la velocidad y dirección de los vientos el día de la liberación, por lo tanto se realizará una estimación en base a las tres últimas liberaciones. Según los registros que se tienen la distancia (terrestre) entre el Globosat y

Montevideo cuando se alcanzó la mayor altura fue aproximadamente $x_{Montevideo} = 200km$ y entre el Globosat y la base terrestre de Durazno fue aproximadamente $x_{Durazno} = 112km$.

Según el esquema de la Figura 42 y tomando como altura máxima alcanzada 30 km se tiene que la distancia a la base terrestre de Montevideo fue $d_{Montevideo} = 202km$ y la distancia a la base terrestre de Durazno fue $d_{Durazno} = 116km$.

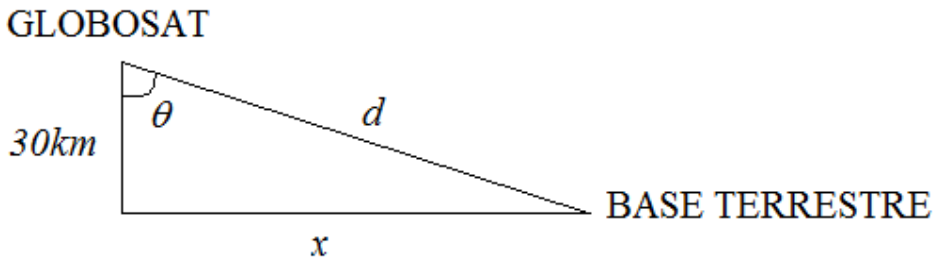

Figura 42 Esquema de distancias

$$d(km) = \sqrt{x(km)^2 + 30^2}$$

Por lo tanto las pérdidas en el espacio libre serán (en dB):

$$L_{FSMontevideo} = 20\log\left(\frac{\lambda}{4\pi d_{Montevideo}}\right) = -122dB$$

$$L_{FSDurazno} = 20\log\left(\frac{\lambda}{4\pi d_{Durazno}}\right) = -117dB$$

13.2. Pérdidas por desapuntamiento

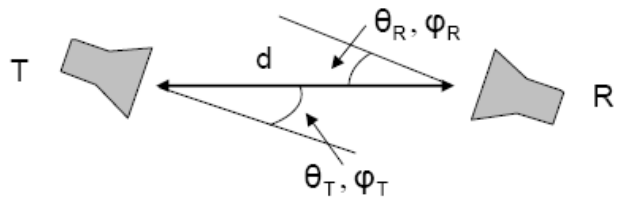

Figura 43 Pérdidas por desapuntamiento

Cuando las antenas no están orientadas en la dirección que presentan su máxima ganancia, en la ecuación de Friis debe sustituirse la ganancia por la ganancia de potencia en la dirección del enlace.

Como se verá mas adelante en este caso se tienen dos antenas receptoras direccionales y una antena transmisora omnidireccional, por lo tanto se supondrá que la antena receptora apuntará en todo momento en la dirección de máxima ganancia, reduciéndose el término de ganancia en recepción a G_R.

Para el caso de la antena transmisora, la ganancia de la misma es:

$$G_T = 1{,}64 sen^3(\theta)$$

Según el patrón de radiación que se muestra a continuación se deberá calcular el ángulo θ en base a la distancia a la estación terrestre

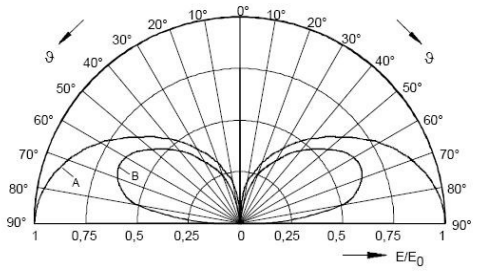

Fig. 2.1 Radiation pattern of vertical monopole above ground of perfect (A) and average (B) conductivity

Figura 44 Patrón de radiación antena omnidireccional [17]

Según el esquema de la Figura 42 y tomando como altura máxima alcanzada 30 km se tiene que: $\theta = Arctg\left(\frac{x}{30}\right)$.

Entonces $\theta_{Montevideo} = 81{,}5^{\circ}$ y $\theta_{Durazno} = 75^{\circ}$. Por lo tanto las pérdidas por desapuntamiento para la antena receptora se reducen a $G_{T\,Montevideo} = 1{,}58$ y $G_{T\,Durazno} = 1{,}48$.

13.3. Pérdidas por desadaptación

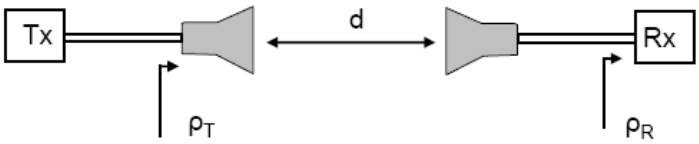

Figura 45 Pérdidas por desadaptación

Cuando existe desadaptación de impedancias en el transmisor o en el receptor, hay que calcular lo que se pierde por reflexión. En este caso se supondrá una ROE de 1,5 para el caso del transmisor, siendo este un límite de seguridad aceptable para transmisores modernos. Se supondrá perfecta adaptación en el receptor.

Se tendrá entonces que $\rho_T = \frac{ROE-1}{ROE+1} = 0{,}2$, $\rho_R = 0$. Por lo tanto las pérdidas por desadaptación (en dB) serán: $10\log\left(1-\left|\rho_T\right|^2\right) = -0{,}17dB$, lo cual es un valor despreciable en comparación con los demás factores de pérdidas.

Ahora sólo resta calcular la ganancia de las antenas receptoras.

13.4. Cálculo de la ganancia de las antenas receptoras

Tanto en la base terrestre de Montevideo (azotea de Facultad de Ingeniería) como la de Durazno (Base Aérea de Santa Bernardina) se utilizaron antenas direccionales del tipo Yagi-Uda. Sin embargo las características de ambas antenas son bastante diferentes.

13.4.1. Estación terrestre de Durazno

En este caso se utilizó un arreglo de dos antenas Yagi de 5 elementos enfasadas con dipolo plegado y en polarización vertical. El diámetro de cada elemento es de $6mm$ y la frecuencia de operación es de $145.030MHz$.

En base a estos parámetros y haciendo uso de la aplicación “Yagi Modeler” [18] se obtiene la ganancia para una antena:

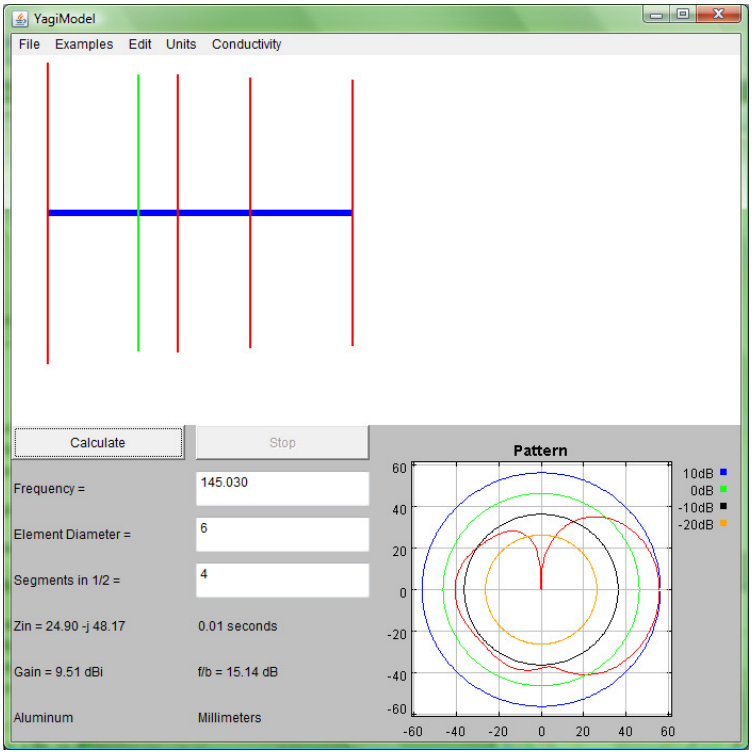

Figura 46 Cálculo de ganancia

Sabiendo que en un array de Yagis si se agrega una antena la ganancia sube 3dB, entonces la ganancia del array utilizado será $G_{R\,Durazno} = 12.51dB$

13.4.2. Estación terrestre de Montevideo

La antena instalada en la azotea de Facultad de Ingeniería es una Yagi de 8 elementos de ganancia $G = 15dB$ y de polarización circular. La diferencia de polarización supone una pérdida de 3dB.

En este caso la ganancia será entonces $G_{R\,Montevideo} = 15dB - 3dB = 12dB$

13.5. Cálculo de la potencia mínima del transmisor

Para el cálculo de la potencia mínima se separarán los dos casos (base terrestre de Montevideo y base terrestre de Durazno) y a partir de la ecuación de Friis se tomará el valor de P_T más restrictivo. En ambos casos se considerará un margen adicional de pérdidas $(L_{adicional})$ de -15dBi.

La ecuación de Friis en (en dB) que se debe cumplir es:

$$P_T = P_R - L_{FS} - G_T - G_R + L_{adicional}$$

Base terrestre de Durazno

Se tienen los siguientes valores numéricos:

- $P_R = -100dBm = 1x10^{-10}mW$ (sensibilidad del equipo receptor)
- $L_{FSDurazno} = -117dB$
- $G_R = 12.5dBi$
- $G_T = 1.48$
- Márgen adicional de pérdidas: 15dBi

Entonces $P_T = 63mW$

Base terrestre de Montevideo

Se tienen los siguientes valores numéricos:

- $P_R = -100dBm = 1x10^{-10}mW$ (sensibilidad del equipo receptor)
- $L_{FSMontevideo} = -122dB$
- $G_R = 12dBi$
- $G_T = 1.58$
- Margen adicional de pérdidas: 15dBi

Entonces $P_T = 220mW$

Se concluye que la potencia mínima de transmisión para hacer posible la recepción de la telemetría tanto en la base terrestre de Montevideo como en la de Durazno es aproximadamente $P_T = 220mW$.

14. ANEXO III – Posibles mejoras para los GloboSat

De las experiencias en las liberaciones de GloboSat01 y GloboSat02, así también como del trabajo realizado en el período que duró el proyecto, se pueden sacar diversas conclusiones que permitirían mejorar futuras experiencias.

A continuación se mencionan algunas de ellas. Es importante que sean consideradas y estudiadas ya que podrían resultar de suma utilidad en el futuro.

14.1. Modulador AFSK.

Tanto para GloboSat01 como para GloboSat02, el módulo que se encarga de generar la señal audible modulada según el protocolo Bell 202 AFSK fue implementado utilizando un microcontrolador PIC16F877A, con un conversor D/A R2R a la salida de uno de sus puertos de 8 bits para generar la señal analógica. Dado que el tamaño y peso de la carga útil de los GloboSat son varibales que hay que intentar minimizar ya que determinan, por ejemplo, el tipo de globo, cantidad de helio y la posibilidad de agregar más funcionalidades a la carga útil, una de las mejoras que se puede realizar es intentar reducir el tamaño del módulo AFSK.

Una primera solución para reducir el tamaño de este módulo puede ser implementar todas sus funcionalidades directamente en el procesador principal. De esta manera se maximiza el ahorro de tamaño y peso debido al mismo. Sin embargo, este ahorro en hardware implica un sacrificio en el software del procesador principal. La señal que se genera debe ser estable en frecuencia para que pueda ser decodificada correctamente en a recepción. Para asegurar que esto sea así, es necesario que durante la generación de la señal sólo se destine tiempo de procesador para esa tarea. Esto quiere decir que no se deberían generar interrupciones tanto internas como externas ni tampoco realizar otras tareas durante la generación de la señal. También hay que asegurarse que en caso de producirse algún error, el programa debería poder continuar su normal funcionamiento desde un estado conocido. Si el tiempo que se demora en generar la señal no es importante, esta solución sería válida. Sin embargo, hay funciones en el

programa principal que también necesitan ser ejecutadas en intervalos regulares de tiempo mediante interrupciones, por lo que esta solución no sería tan efectiva como parece.

Como se concluye del párrafo anterior, la mejor opción en cuanto a software se refiere es mantener un módulo externo encargado de generar la señal modulada en AFSK. De esta forma el procesador principal sólo tiene que enviar la trama al modulador mediante algún protocolo de comunicación (por ejemplo, UART) y mientras la señal es generada puede continuar con su hilo principal. Pero entonces, para conseguir el objetivo principal de reduciar tamaño y peso hay que buscar una manera de implementar un nuevo módulo con las mismas funcionalidades que el que ya está hecho.

Este módulo debe contar por lo menos con lo siguiente:

- Un puerto de comunicación USART
- Más de 300 bytes de memoria RAM
- Puerto de salida para generar la señal audible.
- Trabajar con un cristal de 20 MHz.
- Debe ser más pequeño que el PIC16F877A
- Disponer de algún pin de salida para otras señales (PTT, leds indicadores, etc.)

Estas características son necesarias para poder reutilizar código ya desarrollado en el modulador actual y no tener que diseñar un modulador desde cero. Por esta razón, lo primero que se debe investigar es la existencia de microcontroladores de la misma familia del PIC16F877A que cumplan con estos requerimientos. Estudiando el catálogo de productos de la empresa que fabrica estos microcontroladores (Microchip) se encontró que el PIC16F87 y el PIC16F88 son los más pequeños que cumplirían con estos requerimientos. Tienen 18 pines contra los 40 del PIC16F877A. Cuentan con un puerto de comunicación UART, 368 bytes de memoria RAM, trabajan con cristales de 20 MHz y cuentan con un módulo PWM (Pulse Width Modulation) para generar la señal audible.

En cuanto a tamaño, el PIC16F87 es 4 veces más pequeño que el PIC16F877A. Además, si se utiliza el módulo PWM para generar la señal audible, sólo un pin del PIC16F87 es necesario (el que genera la señal modulada en ancho de pulso) contra los 8 pines que se utilizan en el modulador basado en el PIC16F877A. El conversor D/A en este último modulador es un R2R que consta de 18 componentes entre resistencias y condensadores. Al utilizar PWM, basta utilizar tan sólo 4 componentes (2 resistencias y 2 condensadores) para construir un filtro pasabajos de segundo orden que convierte la señal PWM en la señal AFSK deseada. Este cambio implica una modificación al código, pero es válido el esfuerzo dadas las ventajas que se ganan en tamaño y cantidad de componentes del nuevo modulador.

El PIC16F87, al igual que el PIC16F877A, soporta ICSP (In Circuit Serial Programming), es decir que el mismo puede ser reprogramado una vez ubicado en la placa final.

Teniendo reservados los pines correspondientes al puerto UART y al módulo PWM, siguen sobrando pines de salida, uno de los cuales debe ser utilizado para el PTT (Push To Talk), necesario para que el trasmisor FM comience a trasmitir.

En conclusión, sería sencillo modificar el diseño del modulador basado en el PIC16F877A para conseguir un modelo 4 veces más pequeño, con algunos cambios en el código original.

La figura siguiente es un esquemático del posible modulador mejorado:

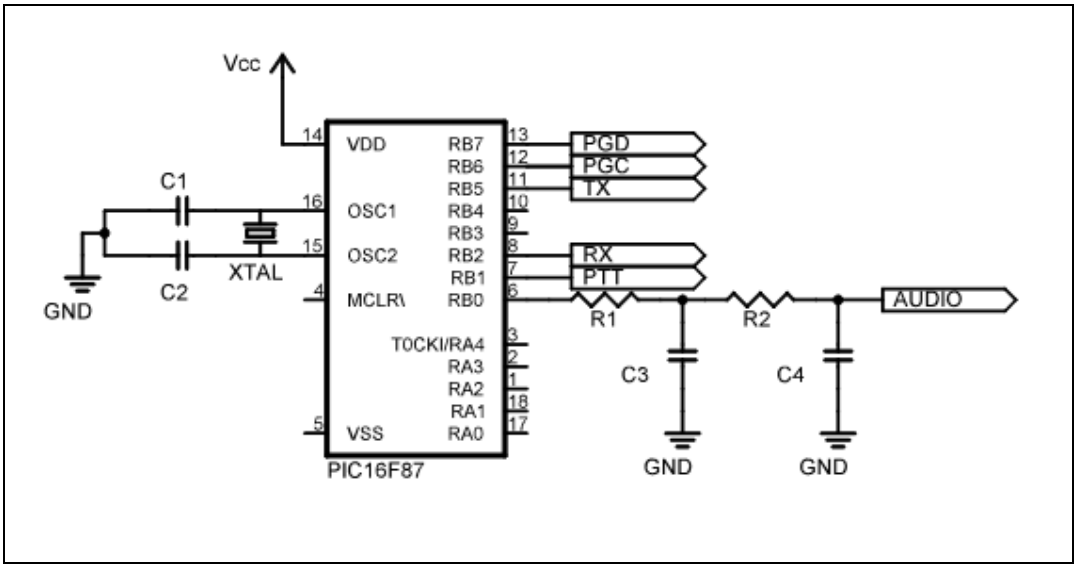

Figura 47 Modulador mejorado

Donde:

- $C1 = 15\,pF$
- $C2 = 15\,pF$
- $C3 = 0.1\mu F$
- $C4 = 0.1\mu F$
- $R1 = 2.5k\Omega$
- $R2 = 2.5k\Omega$
- XTAL es un cristal de $20MHz$

14.2. Modos digitales de trasmisión

Luego de un estudio de los diferentes modos de trasmisión digitales, se resolvió que para las dos primeras liberaciones se utilice el protocolo APRS, cuyas tramas son trasmitidas según las especificaciones del protocolo Bell 202 a nivel de capa física y AX.25 en capa de enlace.

Puede resultar conveniente utilizar más de un modo de trasmisión por las siguientes razones:

Difusión – las tramas con información referente a los datos de los sensotes, GPS, DUT, etc., se envían en formato APRS sobre AX.25, mientras que las balizas y otros mensajes de notificación pueden ser moduladas, por ejemplo, en CCW (Morse), PSK31, PSK63. La ventaja radica en que si no llegó a difundirse masivamente la información sobre una liberación, un radioaficionado con el oído entrenado puede interpretar los mensajes trasmitidos en CCW, o reconocer que se está trasmitiendo información en PSK31 sólo por el sonido. En esos mensajes se puede dar aviso, por ejemplo, que se está trasmitiendo información APRS en cierta banda para que el radioaficionado pueda configurar sus equipos y comenzar a recibir dicha información.

SNR – Los modos digitales de trasmisión de bitrate razonable como PACKET a 1200 bps (AX.25) requieren una buena relación señal a ruido (-3 dB) para que los datos puedan ser decodificados correctamente por el software de recepción. Para otros modos como PSK31, si bien no cuenta con código de corrección de errores, la SNR puede alcanzar los -11.5 dB, lo que permite recibir información en situaciones donde no lo soportaría otros modos. Si se utiliza CW, una persona entrenada podría interpretar un mensaje con una SNR de hasta -20 dB, dependiendo de la velocidad de trasmisión.

Ancho de banda – Si bien en Uruguay no significa un problema dada la escasa actividad de radioaficionados en comparación con otros países, el ancho de banda disponible para trasmitir podría llegar a ser un problema. Utilizando AX.25, sólo se puede trasmitir una única señal a determinada frecuencia dado que las frecuencias de los tonos (1200 Hz y 2200 Hz), y por ende el ancho de banda (3400 Hz) están predefinidos. El ancho de banda de PSK31 es de 40 Hz nada más, lo que permite la existencia de varias trasmisiones en ese modo a una misma frecuencia. Lo mismo sucede con CW o CCW.

Sería posible implementar más de un modo de trasmisión en el firmware del modulador. Se propone el siguiente formato para las tramas que recibe el modulador:

	BANDERA DE INICIO	MODO	LARGO DE LA TRAMA	INFORMACIÓN
Bytes	1	1	1	0 - 255

Tabla 6 Formato de tramas para el nuevo modulador

La bandera de inicio le indica al modulador que va a comenzar la trasmisión de datos. El byte de "modo" selecciona el modo de trasmisión (AX.25, PSK31, etc.). Por último el largo de la trama indica cuántos bytes de largo tiene el campo de información.

Para poder implementar diferentes modos digitales en el firmware se necesita más memoria para el programa y más memoria RAM, por lo que hay que estudiar si los microcontroladores utilizados para GLOBOSAT01 y GLOBOSAT02 tienen capacidad suficiente.

15. ANEXO IV – Microcontrolador PIC

En este proyecto se utilizo el PIC 16F877A. Este microcontrolador es fabricado por MicroChip familia a la cual se le denomina PIC. El modelo 16F877A posee varias características que hacen a este microcontrolador un dispositivo muy versátil, eficiente y practico para ser empleado en la aplicación que posteriormente será detallada. Por información acerca del mismo dirigirse a la hoja de datos, en [10].

Programación

Una gran ventaja de los microcontroladores PIC es que se pueden programar sin necesidad de sacar el microcontrolador del circuito donde esta montado, a esta característica Microchip la llama programación in circuit o ICSP (In Circuit Serial Programming).

Se trata de un sistema de programación serie síncrona en el que intervienen 2 señales, una de entrada y salida para la transmisión y recepción de datos y otra de entrada para la sincronización de la transmisión y recepción de los datos. Las líneas utilizadas se ubican en el pin RB6/PGC para la señal de sincronización (reloj) y en el RB7/PGD para los datos.

El protocolo trabaja con dos tensiones, una de alimentación (VDD), cuyo rango de valores está comprendido entre 4.5 y 5.5 voltios, y otra de programación (VPP) cuyo rango oscila entre un mínimo y un máximo de 12 y 14 voltios respectivamente.

Los PIC integran en el chip interno un protocolo de programación. Para accionar este modo de programación la tensión del pin MCLR debe variar su valor desde 0 voltios hasta el valor de la tensión de programación Vpp. El chip entraría entonces en un modo denominado por Microchip “Program/Verify”. Una vez en este modo se puede acceder a la memoria de programa, de datos y a la de configuración para programarla y verificarla. En este modo RB6 y RB7 son entradas Trigger Schmitt. El tiempo para

pasar del nivel 0 al Vpp debe ser inferior a 72 ciclos de reloj para asegurar que el dispositivo a entrado en este modo.

En la Tabla 9 aparece una descripción de los pines necesarios para llevar a cabo una programación serie. [6]

Pin	Función	Tipo de pin	Descripción
RB6	CLOCK	Entrada	Entrada de reloj
RB7	DATA	Entrada / salida	Entrada y Salida de Datos
MCLR	VPP	Programación	Tensión de Programación
VDD	VDD	Alimentación	Tensión Positiva
VSS	VSS	Alimentación	Tensión Negativa

Tabla 7 Pines utilizados en la programación in circuit

16. ANEXO V – Multipsk

El MULTIPSK es un transceiver digital para Windows que soporta una gran variedad de modos digitales. Además de ser utilizado para recibir los mensajes enviados por el GLOBOSAT, es una herramienta muy útil a la hora de debuguear. Esto se debe a que tiene la opción de mostrar en pantalla los mensajes recibidos aunque la trama esté mal armada. Este documento pretende ser una guía rápida para utilizar este programa con dicho fin.

Al ejecutar el programa por primera vez, se abre una ventana como la siguiente:

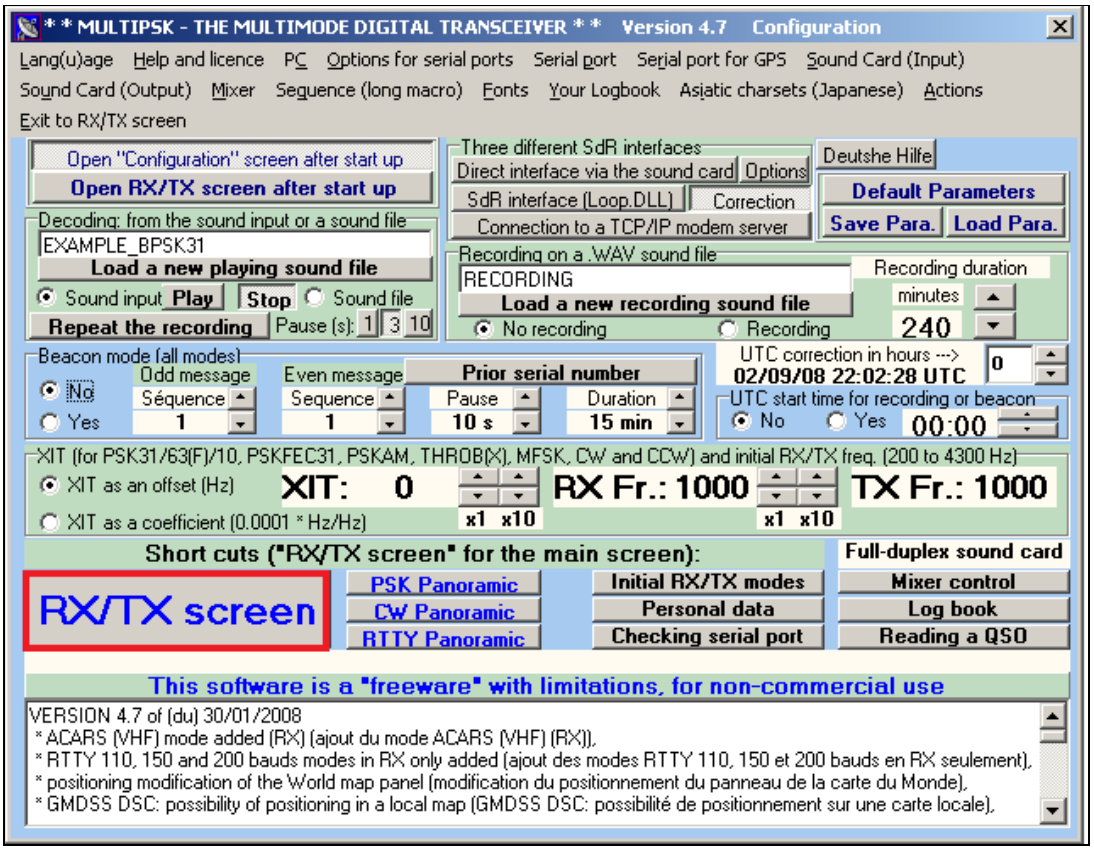

Figura 48 Pantalla de configuración

Las opciones por defecto se dejan como están, simplemente hay que hacer click en el botón “RX/TX screen” que aparece recuadrado en la imagen.

Esto lleva a la ventana principal, tal como aparece en la próxima imagen:

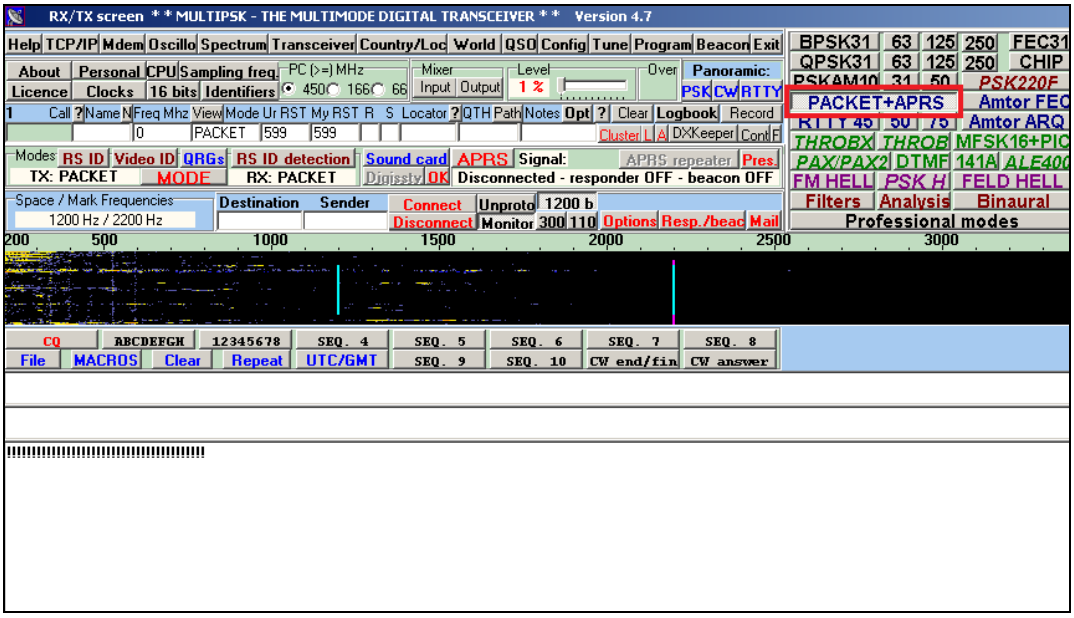

Figura 49 Ventana principal

Se pueden distinguir 5 partes principales en la ventana principal. Arriba a la derecha se encuentran los modos digitales soportados por el programa. El que interesa en este caso es el que dice “PACKET + APRS”. Al hacer click en ese botón, aparecen arriba a la izquierda todas las opciones que se pueden utilizar con el protocolo seleccionado.

Inmediatamente debajo de estos paneles, se encuentra un analizador de espectro temporal, que va mostrando en el tiempo la potencia de la señal recibida en el espectro de frecuencias audibles.

Debajo de éste, se encuentra el panel de trasmisión, que no interesa en este caso, ya que nunca se envía información al GLOBOSAT, sólo se recibe.

Por último, abajo del todo se tiene el panel de recepción. En éste se muestran los mensajes recibidos decodificados.

Si se conecta el cable de audio del receptor FM a la PC, el MULTIPSK comenzará a procesar el audio recibido e intentará decodificar la señal en el modo

digital seleccionado, mostrando la información en el panel de recepción. Por defecto, sólo se muestran la información de las tramas que llegan con el CRC correcto, las tramas cortadas o que tienen un CRC inválido no se muestran. Para cambiar este comportamiento, hay que hacer click en el botón “Options” del panel de opciones del modo PACKET + APRS, como se muestra en la figura:

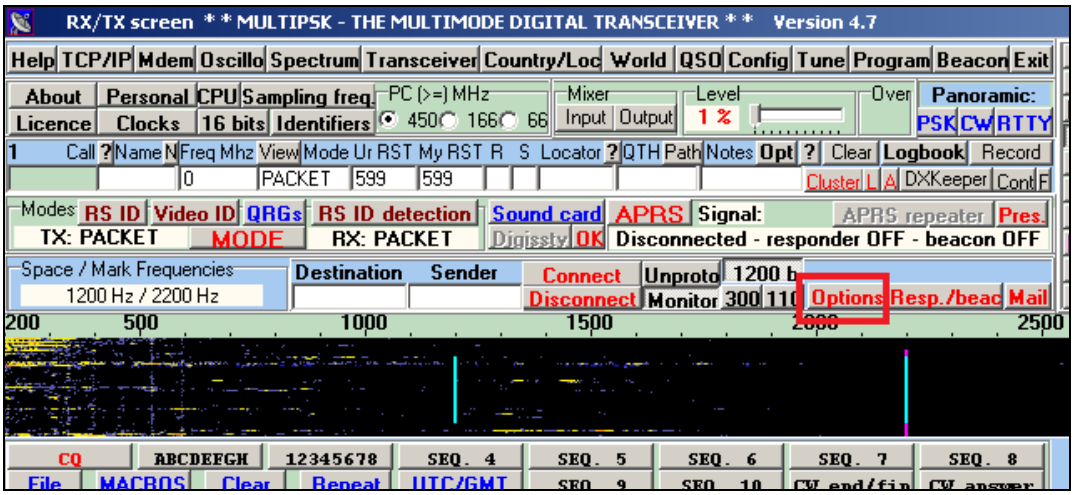

Figura 50 Opciones

Esto abre una ventana que tiene los siguientes campos:

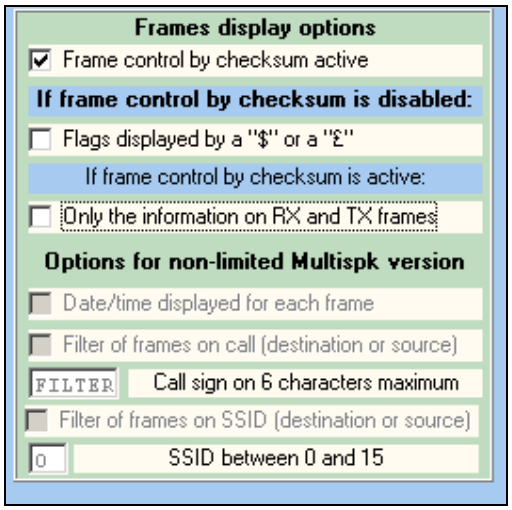

Figura 51 Configuración para la vista de las tramas

Por defecto, la primera opción (“Frame control by checksum active”) está habilitada. Si se deshabilita, el programa no va a hacer ningún chequeo de la trama recibida, y mostrará en pantalla lo que pueda ir decodificando.

También se puede habilitar la segunda opción, que sirve para mostrar con símbolos las banderas al inicio y fin de las tramas AX.25.

Por último, si la tercera opción está marcada, sólo se mostrará el contenido de los paquetes sin el encabezado AX.25.

Algo que hay que tener presente al momento de buscar errores es que a pesar de que no se hagan los chequeos de FCS sobre las tramas recibidas, por lo menos el encabezado de la trama tiene que “parecerse” al del protocolo AX.25, sino el programa no mostrará nada en el panel de recepción.

El MULTIPSK guarda automáticamente los mensajes que se muestran en el panel de recepción, no hay opción para guardar. El log se genera cuando se cierra el programa y se guarda en la carpeta “QSO”, que se encuentra en la misma carpeta donde está el ejecutable.

Hay que tener sumo cuidado con lo siguiente: el tamaño de los logs no puede exceder los 64 KB y sólo se guardan los primeros 64 KB, por lo que es necesario cerrar y abrir el programa cada cierto tiempo para que no se pierdan datos. [16]

17. ANEXO VI – API del programa principal

Documentación de archivos

Referencia del Archivo api.h

Funciones de I/O.

Enumeraciones

- enum **sensor**

Funciones

- void **reset_energia** (void)
- float **get_energia** (void)
- void **send_paq** (char *buffer)
- void **reset_GPS_port** (int BPS)
- int **get_gps_frame** (char *buffer, int max_size)
- float **get_temperatura** (enum **sensor**)
- float **get_presion** (void)
- float **get_humedad** (void)
- int **init_sensores** (void)
- float **get_Vss** (void)
- float **get_Iss** (void)
- void **quemar_tanza** (void)
- void **UART_Init** (int UART_num, int BPS, int ena_TX, int ena_RX)
- void **UART_Transmit** (unsigned char data, int UART_num)
- unsigned char **UART_Recieve** (int uart_num)
- int **get_DUT_status** (unsigned char *buffer)
- void **HW_Init** (void)
- void **apagar_DUT** (void)
- void **apagar_memorias** (void)
- void **apagar_sensores** (void)
- void **hex2ascii** (unsigned char origen, char *destino)
- void **TIMER1_Init** (void)
- void **EE_PTR_Init** (void)

Descripción detallada

Funciones de I/O.

Encabezados de las funciones que controlan las I/O implementadas por el grupo HWGLOBOSATS

Definición en el archivo **api.h**.

Documentación de las enumeraciones

enum sensor

Lista de los sensores de temperatura y humedad.

Definición en la línea 11 del archivo api.h.

Documentación de las funciones

void apagar_DUT (void)

Apaga el DUT

Definición en la línea 5 del archivo apagar_DUT.c.

void apagar_memorias (void)

Apaga las memorias

Definición en la línea 5 del archivo apagar_memorias.c.

void apagar_sensores (void)

Apaga los sensores de temperatura y presion

Definición en la línea 5 del archivo apagar_sensores.c.

void EE_PTR_Init (void)

Inicializa los punteros a las memorias donde se guardan los datos de caja negra

Definición en la línea 26 del archivo EE_PTR_Init.c.

Hace referencia a EE_PTR_Init().

Referenciado por EE_PTR_Init(), y main().

int get_DUT_status (unsigned char * *buffer*)

Devuelve:

- -3 si ocurre un DUT timeout
- -2 si ocurrieron mas cambios en la memoria que el maximo establecido
- -1 si ocurre un error inesperado
- 0 si no hubo cambios en las memorias
- n donde n es la cantidad de cambios en las memorias. Las direcciones de los cambios se guardan en *buffer*

Definición en la línea 5 del archivo get_DUT_status.c.

Hace referencia a get_DUT_status(), UART_Recieve(), y UART_Transmit().

Referenciado por get_DUT_status(), y main().

float get_energia (void)

Devuelve la energia que le queda a la bateria en mAh como float.

Definición en la línea 46 del archivo api.c.

int get_gps_frame (char * *buffer*, int *max_size*)

Llena buffer con datos recibidos, Devuelve largo de buffer o 0 si no hay datos

Definición en la línea 7 del archivo get_gps_frame.c.

Hace referencia a get_gps_frame(), y UART_Recieve().

Referenciado por get_gps_frame(), y main().

float get_humedad (void)

Recibe como parámetro el sensor (enum) y devuelve un float con el valor del mismo en porcentaje.

Definición en la línea 7 del archivo get_humedad.c.

Hace referencia a get_humedad().

Referenciado por get_humedad(), y main().

float get_Iss (void)

Devuelve un float con la corriente entregada en mA.

Definición en la línea 138 del archivo api2.c.

Hace referencia a get_Iss().

Referenciado por get_Iss().

float get_presion (void)

Devuelve un float con la presion en hPa.

Definición en la línea 114 del archivo api2.c.

Hace referencia a get_presion().

Referenciado por get_presion(), y main().

float get_temperatura (enum *sensor*)

Recibe como parámetro el sensor (enum) y devuelve un float con el valor del mismo en grados Celsius.

Definición en la línea 98 del archivo api2.c.

Hace referencia a get_temperatura().

Referenciado por get_temperatura(), y main().

float get_Vss (void)

Devuelve un float con el voltaje de la batería en Volts.

Definición en la línea 132 del archivo api2.c.

Hace referencia a get_Vss().

Referenciado por get_Vss(), y main().

void hex2ascii (unsigned char *origen*, char * *destino*)

Convierte el byte *origen* en su valor hexa como string y lo guarda en *destino*

Definición en la línea 5 del archivo hex2ascii.c.

Hace referencia a hex2ascii().

Referenciado por hex2ascii(), y main().

void HW_Init (void)

Inicializa todos los dispositivos de HW

Definición en la línea 22 del archivo HW_Init.c.

Hace referencia a HW_Init(), y UART_Init().

Referenciado por HW_Init(), y main().

int init_sensores (void)

Inicializa los sensores. Devuelve -1 si hubo error.

Definición en la línea 127 del archivo api2.c.

Hace referencia a init_sensores().

Referenciado por init_sensores().

void quemar_tanza (void)

Quema la tanza.

Definición en la línea 149 del archivo api2.c.

Hace referencia a quemar_tanza().

Referenciado por main(), y quemar_tanza().

void reset_energia (void)

Resetea el contador de energia

Definición en la línea 40 del archivo api2.c.

Hace referencia a reset_energia().

Referenciado por reset_energia().

void reset_GPS_port (int *BPS*)

Setea la velocidad del puerto al que esta conectado el GPS.

Definición en la línea 68 del archivo api2.c.

Hace referencia a reset_GPS_port().

Referenciado por reset_GPS_port().

void send_paq (char * *buffer*)

Envía por el puerto serie el string buffer.

Definición en la línea 57 del archivo api2.c.

Hace referencia a send_paq().

Referenciado por main(), y send_paq().

void TIMER1_Init (void)

Inicializa el timer 1 para determinar timeouts

Definición en la línea 9 del archivo timer1_init.c.

Hace referencia a TIMER1_Init().

Referenciado por TIMER1_Init(), y UART_Recieve().

void UART_Init (int *UART_num*, int *BPS*, int *ena_TX*, int *ena_RX*)

Inicializa el puerto *UART_num* a una velocidad *BPS* y habilita o deshabilita la trasmisión y recepción con "1" o "0" en los parametros *ena_TX* y *ena_RX*

Definición en la línea 4 del archivo UART_Init.c.

Hace referencia a F_CPU, y UART_Init().

Referenciado por HW_Init(), y UART_Init().

unsigned char UART_Recieve (int *uart_num*)

Devuelve el char recibido en el puerto *uart_num*

Definición en la línea 7 del archivo UART_Recieve.c.

Hace referencia a TIMER1_Init(), y UART_Recieve().

Referenciado por get_DUT_status(), get_gps_frame(), y UART_Recieve().

void UART_Transmit (unsigned char *data*, int *UART_num*)

Envia el char *data* por el puerto *UART_num*

Definición en la línea 11 del archivo Uart_transmit.c.

Hace referencia a UART_Transmit().

Referenciado por get_DUT_status(), y UART_Transmit().

Referencia del Archivo compic.h

Configuracion del modulador.

Definiciones

- #define **FRAME_BUFFER_SIZE** 128
- #define **FRAME_INIT_BYTE** "\x02"
- #define **FRAME_INIT_CH** '\x02'
- #define **FRAME_END_BYTE** "\x04"
- #define **FRAME_END_CH** '\x04'

Descripción detallada

Configuracion del modulador.

Parametros relacionados con la configuracion del modulador AFSK (PIC). Deben coincidir con los parametros del firmware del modulador.

Definición en el archivo **compic.h**.

Documentación de las definiciones

#define FRAME_BUFFER_SIZE 128

Tamaño maximo de la trama que se puede enviar al modulador

Definición en la línea 13 del archivo compic.h.

Referenciado por main().

#define FRAME_END_BYTE "\x04"

Bandera de fin de trama para el modulador (string)

Definición en la línea 28 del archivo compic.h.

Referenciado por main().

#define FRAME_END_CH '\x04'

Bandera de fin de trama para el modulador (char)

Definición en la línea 33 del archivo compic.h.

Referenciado por main().

#define FRAME_INIT_BYTE "\x02"

Bandera de inicio de trama para el modulador (string)

Definición en la línea 18 del archivo compic.h.

Referenciado por main().

#define FRAME_INIT_CH '\x02'

Bandera de inicio de trama para el modulador (char)

Definición en la línea 23 del archivo compic.h.

Referencia del Archivo funciones.h

Encabezados de funciones auxiliares.

```
#include <stdlib.h>
#include <string.h>
```

Funciones

- float **gps2grados** (char *dato_gps, char *punto_cardinal)

  Convierte coordenadas GPS en grados y fraccion.

- int **en_poligono** (float x, float y, int cant_puntos, float array[][2])

Verifica si un punto se encuentra dentro de un poligono.

- int **parse_gpgga** (char *dato_gps, char *hora, char *latitud, char *card_lat, char *longitud, char *card_lon, char *altura)

  Parsea mensajes GPGGA.

- void **str2ax25** (char *orig)

  Rota 1 lugar a la izquierda los elementos de un array.

- int **strcatmaxn** (char *dest, char *orig, int max_size)

  Concatena dos strings si el tamano del destino lo permite.

- int **verificar_poligono** (float vertices[][2], int cant_vertices)

  Verifica que la cerca electronica este bien definida.

- int **es_convexo** (int cant_puntos, float array[][2])

  Determina si un poligono es convexo.

Descripción detallada

Encabezados de funciones auxiliares.

Encabezados de funciones que son utilizadas por el programa principal y que no forman parte del API.

Definición en el archivo **funciones.h**.

Documentación de las funciones

int en_poligono (float *x*, float *y*, int *cant_puntos*, float *array*[][2])

Verifica si un punto se encuentra dentro de un poligono.

en_poligono recibe un punto (x,y), una lista de puntos *array* [][2] que contiene los vertices que definen el poligono, siendo el ultimo punto igual al primero, y la cantidad de puntos definidos. Devuelve "1" si el punto (x,y) esta dentro del poligono definido por *array* [][2] o "0" si esta fuera.

Referenciado por main().

int es_convexo (int *cant_puntos*, float *array*[][2])

Determina si un poligono es convexo.

es_convexo recibe un array con las coordenadas de los vertices de un poligono en *float* array[][2], debiendo ser el ultimo vertice del array igual al primero, la cantidad de puntos del array en *int* cant_puntos y devuelve 1 si el poligono es convexo o 0 si no lo es. *

Definición en la línea 210 del archivo funciones.c.

float gps2grados (char * *dato_gps*, char * *punto_cardinal*)

Convierte coordenadas GPS en grados y fraccion.

gps2grados recibe **dato_gps* conteniendo un dato de latitud o longitud en formato GPS (grados.minutos,fraccion de minuto), **punto_cardinal* que puede ser N,S,E,W y devuelve un float con signo en formato (+/-)(grados,fraccion de grado). En caso de error devuelve 0.

Referenciado por main().

int parse_gpgga (char * *dato_gps*, char * *hora*, char * *latitud*, char * *card_lat*, char * *longitud*, char * *card_lon*, char * *altura*)

Parsea mensajes GPGGA.

parse_gpgga recibe un string dato_gps en formato NMEA (GPGGA) y punteros a hora, latitud, card_lat, longitud, card_lon y altura donde guarda los datos parseados. ATENCION: el string dato_gps se destruye durante el proceso, no se puede reutilizar. Si no se desea perder dato_gps, hay que pasar como parametro una copia del string que se quiere parsear.

Referenciado por main().

void str2ax25 (char * *orig*)

Rota 1 lugar a la izquierda los elementos de un array.

str2ax25 recibe un string y a cada elemento del mismo le hace un shift hacia la izquierda de 1 lugar. Se utiliza para generar el campo de direcciones del encabezado de las tramas UI. El ultimo bit del campo de direcciones debe ser 1, esto se debe realizar a mano. El string se sobreescribe, asi que si se desea conservar el original, pasar una copia del mismo o hacer un

shift de 1 lugar hacia la derecha de cada elemento del string. Tener en cuenta que el string DEBE ser UNSIGNED CHAR para que esto funcione.

int strcatmaxn (char * *dest*, char * *orig*, int *max_size*)

Concatena dos strings si el tamano del destino lo permite.

strcatmaxn recibe un puntero a un buffer de tamano maximo conocido, un string para guardar en dicho buffer y el tamano maximo que puede tener ese buffer. La funcion se asegura de que el tamano del string resultante de concatenar lo que estaba guardado en el buffer con el string orig sea menor que el el tamano maximo permitido max_size. Si es asi, concatena los strings y devuelve 0. Si el tamano resultante es mayor, no concatena y devuelve -1.

Referenciado por main().

int verificar_poligono (float *vertices*[][2], int *cant_vertices*)

Verifica que la cerca electronica este bien definida.

verificar_poligono recibe como parametros el poligono que define la cerca electronica. Si esta mal definido, devuelve 1, si no, devuelve 0.

Referenciado por main().

Referencia del Archivo planvuelo.h

Parámetros del plan de vuelo.

Definiciones

- #define **DELAY_BATERIA_CARGADA** 5000
- #define **DELAY_BATERIA_MEDIA** 5000
- #define **DELAY_BATERIA_BAJA** 5000
- #define **NIVEL_BATERIA_MEDIA** 400
- #define **NIVEL_BATERIA_BAJA** 200
- #define **MENS_NOSAT** ":CV1LAI : NO SAT"
- #define **MENS_NO232** ":CV1LAI : NO SERIAL DATA"
- #define **LIMITE_VECES_AFUERA** 4

- #define **POLIGONO** {{-32.210197, -56.245111}, {-32.294619, -56.873594}, {-33.200661, -57.276739}, {-33.892714, -56.800664}, {-34.033275, -56.003383}, {-33.399608, -55.173256}, {-32.604622, -55.3647}, {-32.210197, -56.245111}};

Descripción detallada

Parámetros del plan de vuelo.

Definiciones de algunos parametros involucrados en el plan de vuelo.

Definición en el archivo **planvuelo.h**.

Documentación de las definiciones

#define DELAY_BATERIA_BAJA 5000

Milisegundos entre mensajes con bateria baja

Definición en la línea 21 del archivo planvuelo.h.

#define DELAY_BATERIA_CARGADA 5000

Milisegundos entre mensajes con bateria cargada

Definición en la línea 11 del archivo planvuelo.h.

Referenciado por main().

#define DELAY_BATERIA_MEDIA 5000

Milisegundos entre mensajes con bateria media

Definición en la línea 16 del archivo planvuelo.h.

#define LIMITE_VECES_AFUERA 4

Cantidad de lecturas consecutivas fuera de la cerca electronica para liberar el globo

Definición en la línea 48 del archivo planvuelo.h.

Referenciado por main().

#define MENS_NO232 ":CV1LAI : NO SERIAL DATA"

Mensaje que se envia cuando no se reciben datos por el puerto serie al que esta conectado el GPS

Definición en la línea 42 del archivo planvuelo.h.

Referenciado por main().

#define MENS_NOSAT ":CV1LAI : NO SAT"

Mensaje que se envia cuando no hay senial de satelites

Definición en la línea 36 del archivo planvuelo.h.

Referenciado por main().

#define NIVEL_BATERIA_BAJA 200

Nivel a partir del cual se considera baja bateria, en mAh

Definición en la línea 31 del archivo planvuelo.h.

#define NIVEL_BATERIA_MEDIA 400

Nivel a partir del cual se considera media bateria, en mAh

Definición en la línea 26 del archivo planvuelo.h.

#define POLIGONO {{-32.210197, -56.245111}, {-32.294619, 56.873594}, {-33.200661, -57.276739}, {-33.892714, -56.800664}, {-34.033275, -56.003383}, {-33.399608, -55.173256}, {-32.604622, -55.3647}, {-32.210197, -56.245111}};

Coordenadas de los vertices del poligono de la cerca electronica en formato { {lat_1,lon_1} , {lat_2,lon_2} , ... , {lat_n,lon_n} }; con lat_1 = lat_n y lon_1 = lon_n. lat y lon se expresan en grados y fraccion de grado. SUR y ESTE son NEGATIVOS NORTE y OESTE son POSITIVOS

Definición en la línea 58 del archivo planvuelo.h.

Referenciado por main().

Referencia del Archivo teleglobosat.c

Rutina principal.

```
#include "api.h"
#include "funciones.h"
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include "planvuelo.h"
#include "compic.h"
#include <avr/io.h>
#include <avr/wdt.h>
#include <avr/eeprom.h>
#include <avr/interrupt.h>
#include <util/delay.h>
```

Definiciones

- #define **F_CPU** 8000000
- #define **GPS_BUFFER_SIZE** 90
- #define **ENCABEZADO_AX25**
  "\x84\x8A\x82\x86\x9E\x9C\x60\x86\xB0\x60\x86\x8C\x92\x61\x03\xF0"

Funciones

- int **main** (void)

Descripción detallada

Rutina principal.

Contiene la funcion *main* , donde se implementa el plan de vuelo y se llevan a cabo todas las funciones de recolectar datos de los diversos dispositivos, armar las tramas y enviarlas.

Definición en el archivo **teleglobosat.c**.

Documentación de las definiciones

#define ENCABEZADO_AX25 "\x84\x8A\x82\x86\x9E\x9C\x60\x86\xB0\x60\x86\x8C\x92\x61\x03\xF0"

Encabezado AX.25 de la trama APRS

Definición en la línea 35 del archivo teleglobosat.c.

Referenciado por main().

#define F_CPU 8000000

Frecuencia del oscilador en Hz

Definición en la línea 24 del archivo teleglobosat.c.

Referenciado por UART_Init().

#define GPS_BUFFER_SIZE 90

Tamaño del buffer para el mensaje recibido desde el GPS

Definición en la línea 30 del archivo teleglobosat.c.

Referenciado por main().

Documentación de las funciones

int main (void)

Rutina principal

Definición en la línea 55 del archivo teleglobosat.c.

Hace referencia a DELAY_BATERIA_CARGADA, EE_PTR_Init(), en_poligono(), ENCABEZADO_AX25, FRAME_BUFFER_SIZE, FRAME_END_BYTE, FRAME_END_CH, FRAME_INIT_BYTE, get_DUT_status(), get_gps_frame(), get_humedad(), get_presion(), get_temperatura(), get_Vss(), gps2grados(), GPS_BUFFER_SIZE, hex2ascii(), HW_Init(), LIMITE_VECES_AFUERA, MENS_NO232, MENS_NOSAT, parse_gpgga(), POLIGONO, quemar_tanza(), send_paq(), strcatmaxn(), y verificar_poligono().

18. ANEXO VII – Gestión de tiempos

Tarea	Tiempo Estimado	Tiempo Real
Especificación de Telemetría	Julio 2007 - Noviembre 2007	Julio 2007 - Noviembre 2007
Desarrollo de SW del Controlador	Noviembre 2007 - Mayo 2008	Octubre 2007 - Setiembre 2008
Elaboración del Plan de Vuelo	Diciembre 2007 - Enero 2008	Febrero 2008 - Abril 2008
Transmisor de FM VHF	Enero 2008 - Febrero 2008	Mayo 2008 - Setiembre 2008
Integración final	Febrero 2008 - Julio 2008	Febrero 2008 - Setiembre 2008

Tabla 8 Tiempos estimados y reales de las tareas

Especificación de telemetría – Se logró cumplir esta tarea de acuerdo al cronograma original.

Desarrollo de software del controlador – Esta tarea comprendió la programación del programa principal de los microcontroladores (PIC y Atmel). Para el microcontrolador PIC los tiempos estimados y reales coincidieron. Sin embargo, en el caso del microcontrolador Atmel se produjeron retrasos debido a que tanto el microcontrolador como la placa principal llegaron a Uruguay en el mes de Julio de 2007.

Elaboración del Plan de Vuelo – En este caso se produjeron desviaciones respecto al cronograma original debido a que el mismo se elaboró en dos partes: una más sencilla para la primera liberación (Abril de 2008) y otra más compleja para la segunda (Setiembre 2008), este hecho no había sido considerado al momento de crear el cronograma original.

Transmisor de FM VHF – Esta tarea llevó bastante más tiempo del estimado, pues resulto ser de una complejidad mucho mayor a la supuesta. La estimación original de un mes de trabajo no se ajustó a la realidad debido a que el modelo de transmisor elegido no funcionó como se esperaba. Además se debió contar con una etapa previa de aprendizaje, la cual no se contempló en la estimación de tiempos original.

Integración final – La integración de todos los módulos se retrasó tres meses porque al igual que en la tarea "Desarrollo de software del controlador" se debió esperar por el microcontrolador Atmel y la placa de telemetría provenientes de Francia.

A continuación se adjunta el diagrama de Gantt elaborado para el primer entregable donde se especifican las tareas mencionadas.

19. **ANEXO VIII – Contenido del CD**

```
|-- \
|   |-- Documentación.pdf
|
|   |-- GS01
|   |   |-- LANZAMIENTO1.asm
|   |   |-- LANZAMIENTO1.HEX
|   |
|   |   |-- GS01\pcbs_y_esquematicos
|   |   |   |-- gs01.brd
|   |   |   |-- gs01.sch
|   |   |
|   |-- GS02
|   |
|   |   |-- GS02\pcbs_y_esquematicos
|   |   |   |-- gs02.brd
|   |   |   |-- gs02.sch
|   |   |
|   |   |-- GS02\prog_modulador_pic
|   |   |   |-- telesatv2.asm
|   |   |   |-- telesatv2.HEX
|   |   |
|   |   |-- GS02\prog_principal_atmel
|   |
|   |-- GS03
|   |
|   |   |-- GS03\prog_modulador_pic
|   |   |   |-- telesatv2.asm
|   |   |   |-- telesatv2.HEX
|   |   |
|   |   |-- GS03\prog_principal_atmel
|   |
|   |-- hojas_de_datos
|   |   |-- PIC16F87X.pdf
|   |
|   |-- pcbs_y_esquematicos
|   |
|   |   |-- pcbs_y_esquematicos\modulador
|   |   |   |-- modulador.brd
|   |   |   |-- modulador.sch
|   |   |
|   |   |-- pcbs_y_esquematicos\transmisor
|   |   |   |-- transmisor.brd
|   |   |   |-- transmisor.sch
|   |   |
|   |-- programas_extra
|   |   |-- MULTIPSK.ZIP
|   |
|   |   |-- programas_extra\teleglobosat_sim
|   |   |
|   |-- protocolos
|   |   |-- APRS101.PDF
|   |   |-- AX25.2.2.pdf
|   |
```