Diseño e implementación de sistema de recepción para comunicaciones aéreas ACARS (aircraft communications addressing and reporting system) utlizando la tecnología de Radio Definido por Software para brindar información de la Geo-localización de las aeronaves en tiempo real.

Kevin Herney Rodríguez Gómez

Universidad Militar Nueva Granada
Facultad de Ingeniería
Bogotá, Colombia
2017

Diseño e implementación de sistema de recepción para comunicaciones aéreas ACARS (aircraft communications addressing and reporting system) utlizando la tecnología de Radio Definido por Software para brindar información de la Geo-localización de las aeronaves en tiempo real.

Kevin Herney Rodríguez Gómez

Trabajo de grado presentado como requisito parcial para optar al título de:
Ingeniero en Telecomunicaciones

Director(a):
M.Sc. Jose de Jesús de Rugeles

Universidad Militar Nueva Granada
Facultad de Ingeniería
Bogotá, Colombia
2017

Ante todo Dios y mis padres, motor de motivación e innegable apoyo.
A la Facultad de Ingeniería Telecomunicaciones por la formación que me han dado.
Es gracias a ustedes que es posible el presente trabajo.

En verdad, gracias.
***Kevin Herney**.*

Resumen

En el contexto de las comunicaciones aéreas, del constante desarrollo tecnológico, el aumento del número de aeronaves, la eficiencia del control de las actividades aéreas y la seguridad de las aeronaves se han vuelto requerimientos vitales; por ello la existencia de un sistema alterno que cumpla el papel de seguir monitoreando las actividades aéreas en caso de una emergencia. Actualmente el aeropuerto internacional el Dorado ubicado en la ciudad de Bogotá utiliza la tecnología ACARS como modelo de control y mantenimiento de las actividades de las aeronaves en la sabana Cundi-Boyacense. Por ello el presente trabajo contempla el diseño de un sistema de comunicación basado en un prototipo de recepción en la banda de frecuencias aéreas, una base de datos donde se almacena los mensajes ACARS recibidos y un aplicativo Web como plataforma interactiva en tiempo real de la información dispuesta en la base de datos. El prototipo de recepción utiliza tecnología de Radio Definido por Software cuyo dispositivo manipulado es un RTL-SDR junto con un programa de procesamiento de señales. El software para procesar las señales provenientes del espectro electromagnético es GNU-Radio. El aplicativo Web plasma la ubicación en tiempo real de los mensajes ACARS desde la base datos previamente diseñado.

Índice de figuras

Índice de cuadros

Índice general

Capítulo 1

Introducción

El presente trabajo se basa en el diseño de un prototipo de recepción ACARS que funcione como sistema alterno para caso de una emergencia, desarrollado en el *grupo de investigación* ***GISSIC*** del *semillero* ***MAXWELL*** de la universidad militar Nueva granada. La estructura del proyecto se desarrolla contextualizando el entorno del trabajo con un marco de referencia. Posterior a ello se introduce una prueba experimental para identificar el estado de operación del protocolo en la ciudad de bogotá. El proyecto está basado en IV fases y finaliza con las conclusiones.

El `marco de referencia` introduce los conceptos mas importantes de aviación civil y de Radio Definido por Software para entender el presente trabajo. De forma general el capitulo se divide en dos marcos. El primero detalla del concepto de comunicaciones Aéreas/Protocolo ACARS y el estado de dicha infraestructura en Colombia. El segundo marco explica la tecnología usada desde el punto de vista de Software y Hardware.

Debido a que no existe documentación de las frecuencias de operación ACARS en Colombia. En el capitulo denominado `Análisis experimental` abarca la pruebas necesarias para verificar el funcionamiento de la infraestructura ACARS en la ciudad de bogotá, con cercanía al *aeropuerto internacional ELDorado*. Para ello se realiza las pruebas para identificar las frecuencias de operación y analizar la trama ACARS. Todo ello se realiza a través de softwares externos denominados *SDRConsole* y **PlanePlotter**.

La `Fase I` se desarrolla la teoría referente a la demodulación ACARS y todos los parámetros requeridos para llevar a cabo la implementación de un sistema ACARS en GNURadio, teniendo en cuenta los conceptos de procesamiento de señales y protocolo ACARS. En la presente fase se debe lograr el diseño de un programa de recepción ACARS en GnuRadio a través del RTL-SDR.

La `Fase II` se basa en documentar el procedimiento de convertir los datos binarios en una trama ACARS según teoría de decodificación. Dicho proceso se desarrolla con el programa de GNURadio.

El resultado de la `Fase I` y la `Fase II` es un programa GNURadio capaz de entregar en un archivo de texto la información de la señal ACARS captado por un dispositivo RTL-SDR.

La `Fase III` constituye el diseño de la aplicación del usuario para visualizar los resultados de la trama ACARS en tiempo real. Los requerimientos para desarrollar la interfaz gráfica se basa en una base de datos, una aplicación Python para conectar el programa GNURadio con MySQL y una aplicación Web con tecnología de Google API.

La `Fase IV` se basa en analizar estadísticamente los límites del programa planteado desde el punto de cobertura, efectividad de Decodificación, eficiencia del de la interfaz gráfica y la veracidad de la Información presentada.

Capítulo 2

Preliminares

2.1. Objetivos

2.1.1. Objetivo General

Diseñar e Implementar un sistema de comunicación alterna de emergencia aérea ACARS (Aircraft Communications Addressing and Reporting System) mediante Radio Definida por Software (SDR), para brindar información de Geo-Localización de las aeronaves en tiempo real.

2.1.2. Objetivos Específicos

- Diseñar e implementar un sistema de demodulación MSK de las señales ACARS en una plataforma hardware de radio definido por software.
- Decodificar la trama ACARS para acceder a la información contenida según el tipo de mensaje.
- Desarrollar aplicación API de Google para la visualización de la información de los mensajes ACARS de las aeronaves en tiempo real.
- Realizar pruebas de reconocimiento de los mensajes ACARS de las aeronaves con el sistema desarrollado para identificar la eficiencia del prototipo.

2.2. Planteamiento del problema

En la vida moderna del hombre en pleno siglo XXI existe la necesidad de monitorear y analizar el tráfico aéreo de las aeronaves que las personas e instituciones gubernamentales (Ya sean militares, o civiles) demandan ante los desastres naturales que pueda incurrir los aviones.

Con el sistema de comunicación ACARS permite registrar la trayectoria, información común y mecánica (Combustible, Disponibilidad de Asientos, Estado de la tripulación, Condición Climática), información correspondiente a cada avión con el fin de evitar impactos y desastres aéreos, información de estado de la aeronave desde la última posición dada una emergencia.

Dicha sistema infraestructura es costosa y la envergadura es tan amplia que es difícil desarrollar. Dado que es una infraestructura compleja, existe una tecnología denominada Radio Definida por Software. Este permite desarrollar sistemas más económicos, que para un sistema alterno como un SDR-RTL cumple la necesidad del proyecto con total suficiencia. Además, el aprendizaje del funcionamiento y la implementación de sistema en comunicaciones aéreas con dispositivos SDR (USRP o RTL-SDR) es limitado, permiten bajo un rango de frecuencias, demodular y hacer todo el proceso de tratamiento de señales con algún tipo de programa que reconozca como lo es MATLAB o GnuRadio. Es por ello que analizar las señales provenientes del sistema ACARS es posible a un bajo costo.

Teniendo en cuenta todas las posibilidades de cualquier tipo desastre o una falla en la infraestructura de un sistema ACARS, no existe un sistema de comunicación de contingencia que permita conocer la última posición conocida de la aeronave y por consiguiente la trayectoria recorrida al instante de la emergencia.Actualmente la aeronáutica civil colombiana a través de la ARINIC puede disponer de una infraestructura de ACARS como soporte para los pilotos y la torre de control cuando aterriza y despega.

2.3. Estado del Arte

El artículo **ACARS Data Identification and Application in Aircraft Maintenance** [16] propone dos métodos relevantes para establecer la estructura de los formatos de los mensajes ACARS. Dichos métodos se utilizan para identificar las partes de un mensaje de descarga (DownLink) y mensajes de subida (Uplink). El autor denomina los métodos como de formato fijo y el posterior como formato modular.

Los mensajes ACARS se dividen según el tipo de formato: en mensajes downLink y mensajes Uplink. Los paquetes Uplink (Desde la estación base y la aeronave) difiere con los paquetes de Downlink (Desde la aeronave y la estación base). De manera que en dichos formatos contiene todos los parámetros del protocolo ACARS la cual dispone del campo texto en la cual contiene la información del mensaje.

El autor concluye, que el método de formato fijo es usado para conocer el inicio y el final del mensaje. Y el método de formato modular es óptimo para identificar el campo de texto de un mensaje.

El artículo **Experimental encryption of aircraft communication addressing and reporting system (ACARS) aeronautical operational control (AOC) messages.**[12] Curtis Risley, JamesMcMath y el capitán Brian Payne proponen realizar una simulación de un sistema para encriptar mensajes AOC con la finalidad de medir el grado de cumplimiento de los requisitos militares de seguridad y confiablidad en los mensajes AOC de un sistema ACARS.

La simulación se lleva a cabo en Boston en donde se tomaron los respectivos datos (Mensajes AOC). Utilizaron una encriptación comercial denominado FORTEZZA. Durante las pruebas se halló una situación crítica, puesto que el tamaño del mensaje aumentaba linealmente respecto al mensaje original. Consecuentemente la interacción del medio con las estaciones base en la comunicación también es un factor crítico al emplear CSMA, pues tiene que sensar el medio antes de transmitir, lo que causa retraso y por lo tanto congestión de mensajes. A manera de conclusión los autores recomiendan mejorar los estándares que puedan optimizar los procesos de encriptación en la CMU de la aeronave.

El investigador Jun Kitaori del instituto de investigaciones de navegaciones electrónicas, de la ciudad de Tokyo,Japón en el articulo **A performance comparison between VDL mode 2 and VHF ACARS by protocol simulator.**[7] justifica la necesidad de conocer los benéficos técnicos de las dos tecnologías VHF y VDL2.

El proyecto utiliza una simulación bajo los dos protocolos mencionados sobre el simulador OPNET para evaluar las eficiencias de los dos modelos. El tamaño del mensaje varía para algunos casos mayores a 660 Bytes como inferiores a dicho tamaño.

En las últimas décadas se ha visto la necesidad de mejorar el rendimiento de la velocidad de transmisión, procesamiento y enrutamiento de los mensajes. Esta mejora esta denominado como la tecnología VDL2.

Kitaori concluye que un sistema basado en VDL2 puede procesar hasta cuatro veces más rápido que un sistema VHF dado el caso extremo de tener una comunicación congestionada. De manera que actualmente la infraestructura ACARS existente, está migrando de VHF a VDL2 la cual se le está conociendo como la nueva generación de las comunicaciones aeronáuticas conocida como ACARS ATN.

Capítulo 3

Marco de Referencia

El presente capitulo contextualiza el entorno de las comunicaciones Aéreas, presentando los conceptos generales de la aviación civil. De igual forma se detalla la tecnología empleada en el proyecto.

En el inicio del capitulo explica de manera histórica y conceptual la estructura de la aviación civil. Como parte de toda esa gran estructura aérea, profundizamos el tema interés, El protocolo ACARS. Dicha profundización abarca desde su definición, entidades que soportan y regulan el protocolo, los detalles técnicos más importantes y su funcionamiento general. Posteriormente se da un panorama general del estado aéreo en el suelo colombiano, así como la aplicación de la infraestructura ACARS en el aeropuerto Internacional el Dorado de la ciudad de Bogóta.

Al final del capitulo introducimos el concepto de la tecnología de radio definida por software desde el punto de vista de hardware y software. Como Hardware explicamos la estructura electrónica del dispositivo RTL-SDR. Y como software detallamos la definición de GNURadio, en qué consiste y su aplicación en las redes Inalámbricas.

3.1. Navegación, Vigilancia y Comunicación Aérea

A lo largo de la historia del hombre las aeronaves han sido herramientas esenciales en el ambiente civil y bélico. Y fue solo en el escenario bélico el que permitió la evolución tecnológica para la comunicación, navegación y la vigilancia de las aeronaves con un centro de control aéreo. A finales de la segunda guerra mundial, la experiencia tecnológica obtenida fue aprovechada para la aviación civil con la creación de la ICAO la Organización de aviación civil en 1944.

A partir de ese momento se introdujo el concepto ATC(Control de tráfico Aéreo, en sus siglas en inglés Air Traffic control). Son todos los controles necesarios de las actividades aéreas que existen en un aeropuerto.

Desde entonces, se han mantenido tres aspectos importantes que permiten a los pilotos y las torres ATC ejercer un control sobre las actividades aéreas. Dichos aspectos han sido navegación, vigilancia y comunicaciones. Normalmente se le denominan CNS (Communication, Navigation and surveillance). [14]

La navegación se basa en soportar la propia posición y velocidad respecto al plan de vuelo establecido. Por ello que la planificación, rastreo, grabación y el control del movimiento del avión, son las cuatro funciones principales de la navegación.[14]

La vigilancia se fundamenta en determinar la posición y velocidad a través de medios externos de la aeronave. Los primeros medios externos que cumplían esta función fueron los radares. Existen dos tipos de radares. Los radares sobre la aeronave y los radares puesta en tierra. En los radares sobre la aeronave, el radar primario de navegación pasivo en que el radar puesta en tierra detecta la energía del radar pasivo para calcular la velocidad y la altitud de la aeronave. El radar secundario de navegación activa transmite una réplica de la señal originada por el radar puesta en tierra. La infraestructura de vigilancia principalmente radares ha permitido crear un sistema de prevención de coalición de aeronaves. Al mismo tiempo han servido como pronostico y prevención meteorológica para las aeronaves. [14]

En los últimos años existe un sistema alterno de navegación denominado ADSB (*Automatic dependent surveillance – Broadcast*). Es un sistema de vigilancia automático basado en la comunicación de datos. Este sistema permite uso GPS a través de satélites SATCOM para crear una visualización del tráfico aéreo de tiempo real [14]. La implementación de este tipo de tecnología ha sido benéfica en aeropuertos pequeños que no tienen una infraestructura de radares para las actividades de aterrizaje y despeje.

Las comunicaciones como parte de la CNS para soportar las actividades ATC se ha efectuado por comunicación por voz. Dicha comunicación por voz se ajusta a un rango de frecuencia específica para contactar con la línea de la torre de control. El gran problema era que la demanda de aviones era demasiada y generaba esperas para hacer contacto con la torre

de control. Tampoco se puede asignar largos rangos de frecuencias para suplir la necesidad de comunicación, pues como se sabe el espectro radioeléctrico es recuso limitado.

Fue por ello que en la década de los 70s se introdujo el concepto de la comunicación por datos para las actividades aéreas entre la aeronave y la torre de control. Desde entonces se le ha conocido como ACARS. Paralelo al sistema ACARS también se desarrolló sistemas alternos de Comunicación como sistema ATN (Aeronautical Telecommunication network).

Un sistema ATN es un modelo por capas OSI para comunicaciones de datos presentado por la ISO. Dicha infraestructura está basada en siete capas para trasmitir mensajes ATN. Es importante aclarar que na red ATN comparte la estructura tecnológica en tierra de ACARS.[14]

3.1.1. ACARS: Aircraft Communications Addressing and Reporting System

ACARS conocido como (Aircraft Communications Addressing and Reporting System) es una infraestructura de comunicación aérea de datos entre estaciones bases en tierra y aeronaves. Muchas aerolíneas utilizan dicha infraestructura para controlar el tráfico aéreo, comunicaciones con las autoridades aéreas y sus propios centros de operación (Aerolíneas). [2]

Existen principalmente cuatro proveedores que soporta un servicio ACARS, estos se clasifican por regiones continentales [1]:

- **ARINIC** (Rockell Collins) en Estados Unidos
- **CANADIAN** en Canada.
- **JAPANESE** en Japon
- **SITA** en otras regiones

Los mensajes ACARS que se transmiten en redes VHF operan en el rango de frecuencias (118 Mhz- 137Mhz)[1].

Rockell Collins es una compañía que soporta este tipo de infraestructura y en más de 150 países, y en Estados Unidos ejerce como ente de la ARINC (Aeronautical Radio Incorporated).[2]

Aclarado lo anterior, ACARS en términos técnicos es una comunicación que trabaja con un ancho de canal de separación de 25KHz y una tasa de bits de 2400bps. Trabaja con un esquema de modulación (MSK-AM). Como control de acceso utiliza CSMA de tipo no Persistente. CSMA No-persistente en sensar el canal para poder trasmitir un mensaje. Es decir,

verifica si el enlace esta libre para trasmitir, dado el enlace estuviera ocupado vuelve reiteradamente a sensar hasta que el enlace este libre para enviar el mensaje. Finalmente, ACARS no soporta ninguna corrección de errores (FEC). [8]

3.1.1.1. Mensajes

Hay dos tipos de mensajes:

1. **Mensaje "Uplink":** Se refiere a todos los mensajes enviados desde la estación base en tierra hacia la aeronave.
2. **Mensaje "Downlink":** Se refiere a todos los mensajes enviados desde la aeronave hacia la estación base en tierra.

Los mensajes ACARS principalmente tiene dos usos. Los **mensajes ATC** (*Control de tráfico Aéreo*, en sus siglas en inglés *Air Traffic control*) y **mensajes AOC** (*Trafico operacional aéreo*, en sus siglas en inglés *Air Operational Traffic*). [16]

Los mensajes ATC funcionan para comunicar la aeronave y la torre de control de un aeropuerto. Normalmente el contenido del paquete son solicitudes de despeje. Los mensajes AOC son comunicaciones entre el avión y centro de control de su aerolínea. Su contenido son reportes del estatus real de la aeronave en el trayecto de su vuelo[16]. Dichos mensajes son vitales para los trabajos de mantenimiento de las compañías aéreas, pues permiten anticipar los cambios pertinentes de una aeronave cuando aterriza.

Entre otras cosas, ACARS tiene la propiedad de pasar de un modo "Data Link" (Transmisión de mensajes digitales) a "Voice Communication" (Transmisión analógica), a través de un mensaje Downlink donde registra el Numero o ID de comunicación, que de acuerdo al ID de comunicación la estación sintoniza una frecuencia especifica para ejercer la transmisión por Voz. [11]

3.1.1.2. Infraestructura

El funcionamiento de una infraestructura ACARS esta soportado por satélites INSMART y SATCOM, Antenas VHF, VDLM2 o VDL2 (*VHF-Data LINK, Es un tipo de tecnología VHF con ventajas de velocidad de procesamiento y enrutamiento*), Torres de control que ejercen funciones de ATC y los centros de operación de las aeronaves. [2]

Dependiendo del lugar de la aeronave, se define qué tipo de tecnología conforma una red ACARS. Si la aeronave esta sobre tierra, los mensajes ACARS viajan sobre redes VHF y VDLM2. En cambio, Si el avión esta sobre un océano, dichos paquetes se enrutan sobre redes HFDL (High Frecuency data link) y satélites INSMART/SATCOM [2]. De esa manera una

infraestructura ACARS permite transportar mensajes ACARS sobre todo el globo terráqueo.

Como bien es sabido la necesidad de mejorar la eficiencia de la velocidad de enrutamiento y procesamiento para las redes VHF, existen las redes VDL2. Las redes VDL2 es una migración de tecnologías que evita congestión del tráfico aéreo entre los mensajes salientes y entrantes[8]. Por ello es importante aclarar que el presente trabajo se desarrolla en redes VHF. Para ello limitamos el termino de comunicaciones ACARS solo para redes VHF.

Conociendo la infraestructura ACARS en tierra, solo falta hablar acerca de los equipos que soportan el servicio ACARS sobre la aeronave. Existen principalmente dos equipos importantes la MU (Unidad de gestión en sus siglas en inglés Management Unit) y CDU (Unidad de visualización en sus siglas en inglés Control Management Unit).[16]

La MU está diseñado para recibir/enrutar Mensajes Uplink y transmitir Mensajes Downlink sobre equipos VHF. La MU cuando establece una comunicación satelital, lo realiza a través de una unidad de información satelital SDU (Satelite Data Unit). [16]

Dentro la CDU existe un componente que funciona como origen de los mensajes ACARS (Mensajes Downlink AOC). Dicho componente es conocido como FMS. Una FMS está ubicado en la cabina de piloto de la aeronave. [16]

3.1.1.3. Modos de operación ACARS

ACARS opera bajo cuatro modos importantes de operación [13]:

- **Off Mode:** EL Modo Apagado ocurre cuando la MU esta fuera de operación.
- **Demand Mode:** EL Modo Demanda entra en operación cuando tiene que: generar mensajes Downlink para reportar eventos pre-definidos y responder mensaje de petición (Mensaje Uplink).
- **Polled Mode:** El Modo Compartido ocurre cuando la MU recibe un mensaje Uplink con cabecera de tipo de mensaje la cual está contenida con el carácter "_ j". Este es utilizado en zonas de alto tráfico aéreo. La MU entrara en modo polled solo cuando sea requerido por la estación base. La estación base hace este tipo de petición periódicamente a alguna aeronave cada dos segundos. De manera que si la MU tiene un mensaje Downlink en cola este puede transmitir. Caso contrario envía un mensaje con cabacera "_DEL".
- **Failed Mode:** El Modo Fallido ocurre debido a las pruebas de error que ejerce la MU continuamente. De manera que, si detecta errores en la MU, entra en este modo de operación.
- **Voice Mode:** El Modo voz pude ocurrir bajo una petición desde la aeronave o la estación base. Se puede efectuar en el modo demanda o compartido.

3.1.1.4. OOOI

OOOI(OUT,OFF,ON,IN) son eventos basados en el status de la aeronave. OUT significa fuera de Puerta. OFF significa que la aeronave no esta en tierra. ON significa que la aeronave esta sobre tierra. IN significa que la aeronave esta en Puerta. [17]

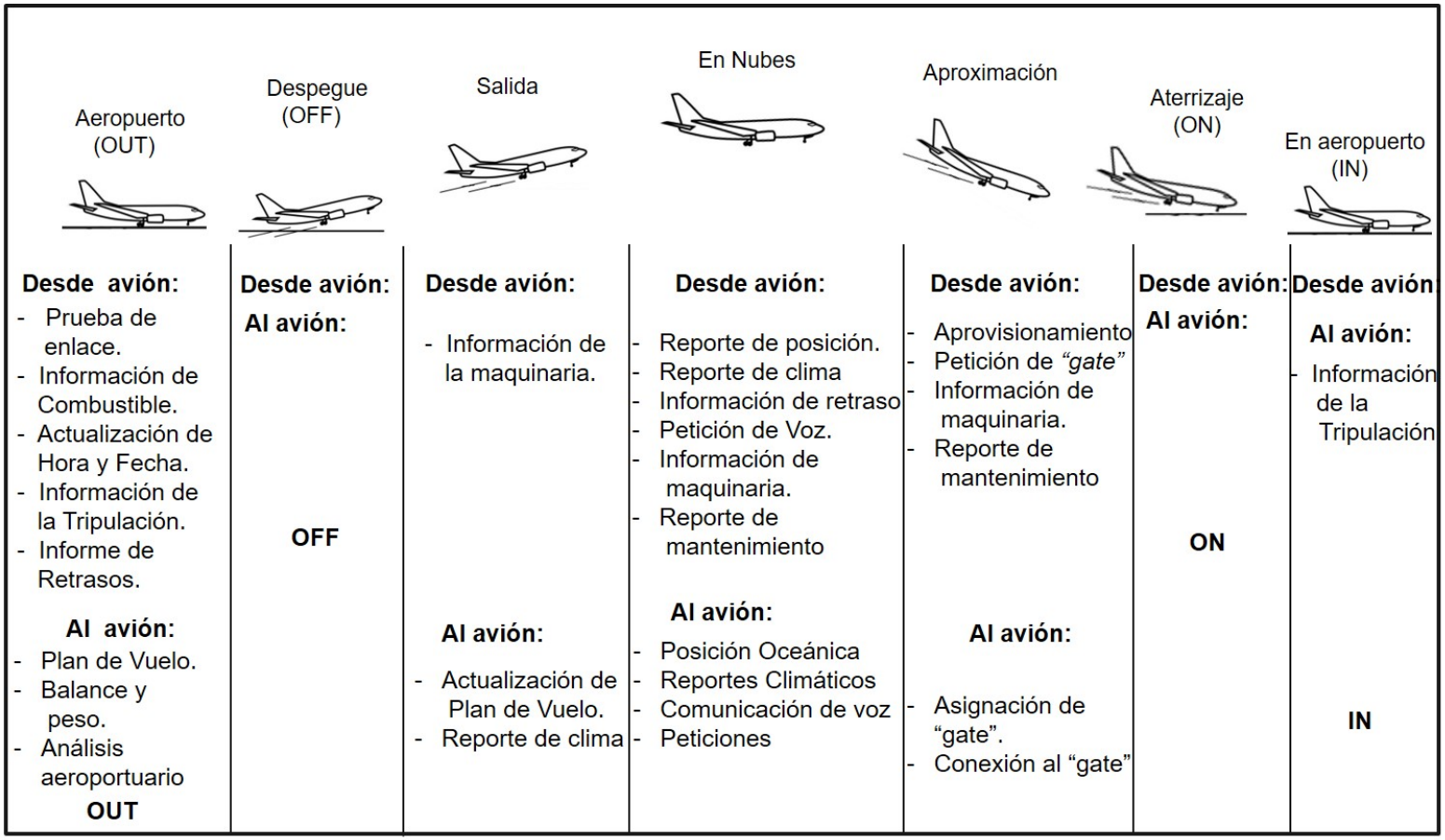

Figura 3.1: Diagrama general de *OOOI* ***ACARS***

3.2. Comunicaciones Aéreas en Colombia

El sistema de comunicaciones existente basado en una red VHF tiene una limitación fundamental, la cual es, que el intercambio de información entre el avión y la unidad de ATC se restringe solamente a mensajes de voz, por medio de ondas de radiofrecuencia utilizando modulación en amplitud AM-DBL. Resulta insuficiente para todo el flujo de trafico Aéreo que existe en el país [6]. Para superar esta limitación, en el Plan Nacional de Navegación Aérea presentado por la Dirección de Servicios a la Navegación Aérea de la Secretaría de Sistemas Operacionales de la *Unidad Administrativa Especial de Aeronáutica Civil*(U.A.E.A.C) ha desarrollado avances para la vigilancia aérea colombiana [4].

Para la vigilancia ATC sobre del espacio colombiano se realiza bajo el sistema RADAR primario y secundario, cuyo desarrollo ha sido importante en la última década. La cobertura RADAR está limitada a áreas continentales y costeras sin cobertura posible sobre las costas oceánicas. [10]

La red nacional de radares ha venido ampliándose desde 1990 que hasta el año 2010 cuenta con 20 sistemas de vigilancia radar. Once *Secondary Surveillance Radar (SSR)* y nueve *Primary Surveillance Radar (PSR)* instalados en el territorio nacional. [4]

La infraestructura de la aeronáutica civil colombiana también cuenta en el campo de navegación, una plataforma de radio ayuda instaladas en tierra. Dicha estructura permite una cobertura de baja densidad en lugares donde las condiciones son particulares, caso tal la amazonía y la región oceánica[6]. La infraestructura actual de los sistemas de radio ayudas cuenta con 49 Sistemas VOR, 51 Sistemas DME (49 Asociados a los VOR + 2 Sistemas DME Independientes), 11 Sistemas ILS Y 26 Sistemas NDB [10].

Los NDB *(Non Directional Beacon)* proveen guía de información horizontal de no precisión indicando el rumbo a las aeronaves en las fases de ruta y aproximación. Los VOR *(VHF Omni-directional Range)* proveen guía de información horizontal de no precisión indicando el radial de rumbo en todas las direcciones a las aeronaves en ruta y aproximación. Los DME *(Distance Measurement Equipment)* proveen la distancia oblicua en millas náuticas, desde la aeronave hasta la estación en tierra.Los ILS *(Instrument Landing System)* proveen guía de información horizontal y vertical de precisión a las aeronaves en la fase de aproximación y aterrizaje indicando a estas la alineación con el eje de la pista y el ángulo de descenso en la aproximación a la misma. Las aeronaves que cuentan con la aviónica de navegación GPS Básico (RAIM), pueden navegar procedimientos de no precisión en las fases de vuelo de ruta y aproximación. [6]

3.2.1. ACARS en Colombia

La frecuencia de operación del protocolo ACARS se basa de acuerdo a la organización del espectro radio eléctrico de cada país, que están sujetas a las políticas y las necesidades de la organización gubernamental encargada de las actividades aéreas. En Colombia dicha organización es la Aeronáutica Civil.

La aeronáutica civil de la unidad administrativa especial junto con la *dirección de servicios a la navegación aérea* desarrolla **un procedimiento para la autorización de salidas via Data Link en el aeropuerto Dorado de Bogotá** [3] que entró en vigor en noviembre de 2013. Este sistema notifica a los usuarios de la aviación colombiana la actualización del sistema de automatización para la expedición de la autorización de salidas conocida como *(DEPARTURE CLEARANCE)*. Dicha comunicación vía Data Link se realiza desde las aeronaves y la torre de control del *aeropuerto Internacional el Dorado de la ciudad de Bogotá*.

El servicio se soporta con equipos del protocolo ACARS. La finalidad de implementar este sistema se basa únicamente en solicitar y recibir la autorización de salida de manera automática. Para enviar los mensajes ACARS se realiza mediante un proveedor de servicios ACARS como `SITA` o `ARINIC`. El procedimiento para la autorización de salidas de las aeronaves se realiza en tres pasos.

3.2.1.1. Primer Paso: *Solicitud*

La tripulación debe solicitar la autorización mediante un mensaje RCD (DEPARTURE CLEARANCE REQUEST). La solicitud debe realizarse con 20 minutos de antelación ante el SLOT asignado por la oficina DFM. El contenido del mensaje contiene los siguientes datos:

- Distintivo de llamada.
- Aeródromo de despegue.
- Posición de parqueo.
- Aeródromo de destino.
- Designador de la información ATIS recibida.
- Tipo de aeronave.
- Registro de la aeronave.

3.2.1.2. Segundo Paso: *Sincronización*

Después de realizar la solicitud, el sistema recibe analiza el mensaje RCD :

- Sí el mensaje RCD se recibe con conformidad establecido en 3.2.1.1, entonces envía un mensaje **CLD** *(DEPARTURE CLEARANCE UPLINK MESSAGE)*
- Sí sistema detecta alguna inconsistencia en la información del mensaje RCD, entonces envía un mensaje **FSM** *(FLIGHT SYSTEM MESSAGE)*.

CLD El contenido del mensaje es:

- Identificación de la aeronave;
- Aeródromo de destino;
- Pista asignada para la salida;
- Procedimiento de salida (SID);
- Código Transponder;
- Siguiente frecuencia;
- Designador de la información ATIS vigente a la hora;
- SLOT asignado por la oficina DFM (EOBT);
- Información adicional

Este mensaje no es una autorización por parte del sistema. Es en cambio un acuerdo de ambas partes, en donde cada una de las partes se ajusta a las condiciones para que exista una coherencia.

Si la tripulación no acepta las condiciones del mensaje, deberá contactar al *controlador de Autorizaciones Eldorado*, en la frecuencia `121.6 MHz` para recibir la autorización de salida por este medio.

FSM El contenido del mensaje se basa en indicar un **REVERT TO VOICE PROCEDURES**. La tripulación debe ajustar en la frecuencia `121.6 MHz` para una *NUEVA AUTORIZACIÓN* de salida con el *controlador de Autorizaciones Eldorado*.

Las inconsistencias dadas para generar el mensaje FMS son:

- Errores de sintaxis del mensaje *RCD* `(ERROR IN MESSAGE)`
- El plan de vuelo no se encuentra en el sistema `FLIGHT PLAN NOT HELD` o existe más de un plan de vuelo `MULTIPLE FLIGHT PLANS HELD`
- La autorización ya fue dada *DCL* `(ALREADY GIVEN)`
- El servicio no se encuentra disponible *DCL* `(SERVICE CURRENTLY NOT AVAILABLE)`
- Incompatibilidad en el tipo de aeronave `AIRCRAFT TYPE MISMATCH`
- Posición de parqueo no enviada `STAND MISMACTH`
- Solicitud enviada muy pronto `REQUEST SENT TOO EARLY`
- Solicitud enviada muy tarde `REQUEST SENT TOO LATE`
- Autorización cancelada `CLEARANCE CANCELLED`

3.2.1.3. Tercer Paso: *Autorización*

Si la tripulación esta conforme con el mensaje pre-Autorización **CLD**, deberá enviar un mensaje **CDA***(DEPARTURE CLEARANCE READBACK)* a menor brevedad posible.

Si pasados los 10 minutos después de la emisión del mensaje CLD, la tripulación no ha aceptado la pre-autorización. El sistema asumirá que se ha presentado un error y anulará el permiso mediante una generación de un mensaje *FSM*.

3.3. Radio Definido por Software

Las redes de comunicación de datos se ha vuelto un componente vital en la sociedad moderna que se ha extendido en todas las aplicaciones educativas, empresariales, militares, seguridad, comerciales y gubernamentales. Todos ellos soportados por sistemas cableados e inalámbricos como los servicios de telefonía fija y movil, Internet, Televisión, transferencia de archivos y servicio de streaming. Servicios que se han vuelto particularmente inalámbricos gracias al desarrollo de la tecnología (SDR) *Radio Definido por Software*.

El desarrollo de SDR se da gracias a la rápida evolución de la micro-electrónica donde los transmisores y receptores se han vuelto versátiles, potentes y portables. En SRD los radios transmisores por medio de Software en banda-base se basan en la Modulación/Demodulación, codificación, Corrección de errores y compresión de señales [18].

3.3.1. Hardware

Antes de describir la estructura de un radio definido por software, es necesario conocer un radio digital. Se sabe que un radio digital esta compuesto por:

Antena *Sección donde esta ubicado la antena* encargada de recibir o transmitir las señales electromagnéticas.

RF *La sección RF* es la responsable de transmitir/Recibir la señal en una frecuencia. Lo usual es volver una señal Banda-Base (Señal IF) en una señal Pasa-banda(Señal RF) (Transmisor).

ADC/DAC *El convertidor Análogo/Digital* se basa en hacer las conversiones pertinentes de una señal continua de radio frecuencia (IF) en una señal digital basada en secuencias binarias.

Modulación/demodulación *El modulo de Modulación* se basa en convertir la señal digital en una señal banda base o la señal original. En este modulo se aplica todos los componentes de procesamiento como filtros, Desmuestreo y remuestreo.

Banda-Base *La sección banda base son las* operaciones de tratamiento de la señal original como Codificación. Normalmente se usa para la implementación de algún tipo de protocolo de la capa de enlace de datos.

Entonces un radio digital SDR se base en implementar todos los componentes de RF, Modulación, ADC y Banda-Base en un solo componente electrónico programable. Dicho dispositivo son denominados FPGAs. La idea es ajustar todos esos componentes mediante un software, permitiendo diversas configuraciones para diferentes aplicaciones en un solo dispositivo SDR.

Actualmente hay muchos dispositivos SDR tales como las populares USRPs (*Universal software radio peripheral*), HackRF y RTL-SDR. Los RTL-SDR son dispositivos de procesamiento de señales de bajo costo, la cual puede programar cualquier aplicación del espectro Radioeléctrico usando tecnología SDR. Es importante aclarar que el diseño este tipo de dispositivos esta basado solo para recibir señales más no como radio transmisores.

La estructura de un dispositivo RTL-SDR basado en el chip MICRO-R82022T se estructura en convertir la señal RF en una señal IF mediante un multiplicador basado en un PLL-VCO.

$$F_{vco} = F_{RF} - F_{if} \tag{3.1}$$

Luego un filtro Pasa-abajo donde elimina el componente de alta frecuencia. Dicho filtro funciona como un decimador para bajar el numero de muestras la cual el dispositivo puede trabajar. Despues pasa por un convertidor ADC la cual opera a una frecuencia de 28.8Mhz. Luego de obtener una señal digital, se descompone en dos señales: En fase y Cuadratura. Lo anterior se logra a través de dos multiplicadores, uno de ellos desfasado 90 grados para lograr la señal en cuadratura. [15]

3.3.2. Software (GNU-Radio)

GNU-Radio es un software gráfico que utiliza un conjunto de herramientas en programación de alto nivel para el procesamiento de señales recreando y diseñando sistemas de comunicaciones. GNU-Radio ejecuta programas basado en lenguaje Python, cuyos programas internos se ejecutan en C++. Para exista coherencia entre los lenguajes GNURadio utiliza como API una interfaz SWIG.

Pero al nivel de usuario toda la estructura de GNU-Radio se basa en Pyhton, desde su ejecución de sus programas y la configuración de los bloques. Los bloques de GNURadio va desde las funciones de demodulación, Filtros, PLL hasta sistemas compactos ya realizados como de-moduladores para televisión digital terrestre, de-modulación GSM, corrección de errores que permiten diseñar sistemas de comunicaciones mas complejos.

El diagrama de flujo que usa GNU-Radio se basa en un grafo denominado Grafo Dirigido Aciclico (DGA) [5]. Es un Grafo basado en vertices cuyo bordes dirigidos NO son ciclicos. En GNU-Radio dichos vértices son los bloques, la cual deben ser por lo menos uno el origen de los datos. Los datos fluyen unilateralmente hacia una dirección donde finaliza en uno o varios vértices.

Para cada bloque, el flujo esta medido por el numero de items que puede procesar. La cantidad de Items depende de cuanto espacio puede haber en el buffer de salida y cuantos items están disponibles en el buffer de entrada. Para mantener ese control de flujo, GNU-Radio permite controlar entonces el máximo de items que puede salir un bloque o todos los bloques de un diagrama.[5]

Capítulo 4

Análisis Experimental: Frecuencia y Trama ACARS

La primera parte se basa en inspeccionar el espectro radio-eléctrico asignado para aplicaciones aéreas para establecer las frecuencias de operación ACARS. Consecuentemente, se hace un análisis de los mensajes ACARS interceptados en Matlab.

Luego en el segundo fragmento de prueba, se documenta el comportamiento espectral en la banda de frecuencia de las señales ACARS. Este análisis se basa en el driver para Matlab, desarrollado por un grupo de académicos de la universidad de Strathclyde en el libro "*Software Defined radio Using MATLAB y SIMULINK and the RTL-SDR*" [15] cuya referencia se encuentra al final del documento. La necesidad del presente análisis se argumenta en el transcurso del documento.

4.1. Identificación de frecuencia de Operación

Inicialmente el objetivo es identificar las frecuencias de operación para luego analizarlas. Para ello usaremos el software RADIO-SDR. Pero viene una pregunta importante, como se valida si efectivamente en esa frecuencia se están transmitiendo mensajes ACARS. Para ello se utiliza un programa de codificación PlanePloter que permite codificar los tonos de frecuencia cuando se trasmite un mensaje ACARS. Es decir, utiliza el mezclador estéreo para grabar la frecuencia de sintonización (En primera instancia a 131.50 MHz) y codificar el mensaje ACARS. La relación de ambos programas (RADIO-SDR y PLANEPLOTTER) permite de manera experimental identificar las frecuencias de operación.

El método es:

- Analizar desde la frecuencia de 131.50 Mhz como es el comportamiento espectral cuando se reciben los mensajes.

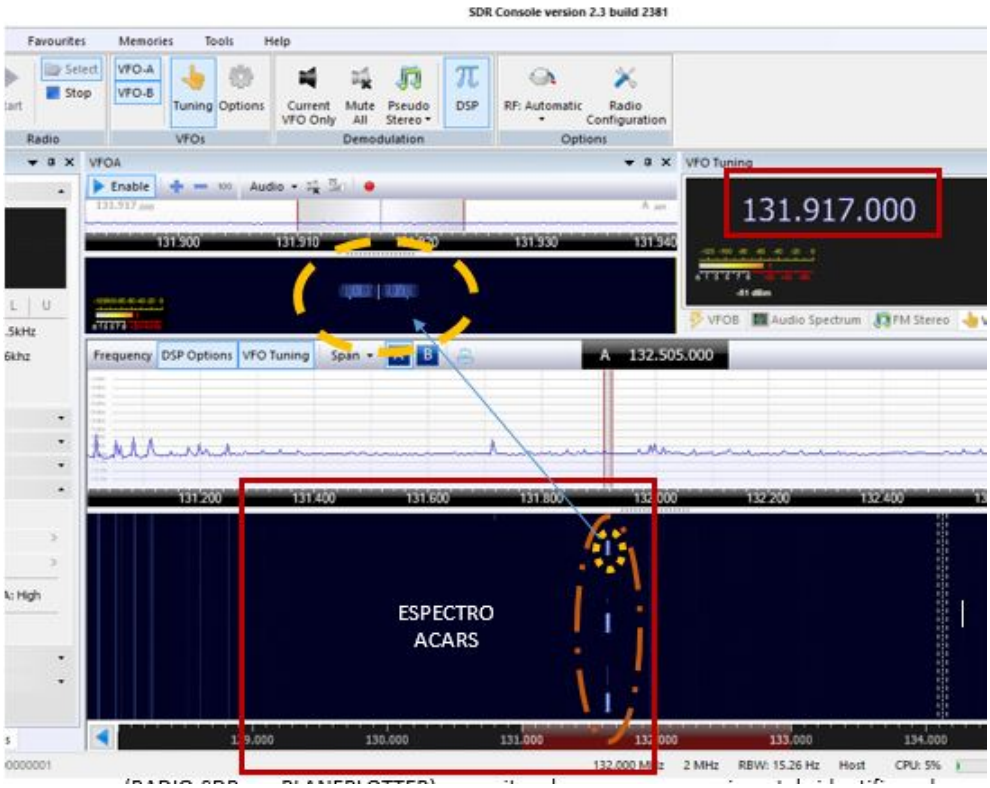

Figura 4.1: Ejemplo cuando se reciben en tiempo real los mensajes ACARS.

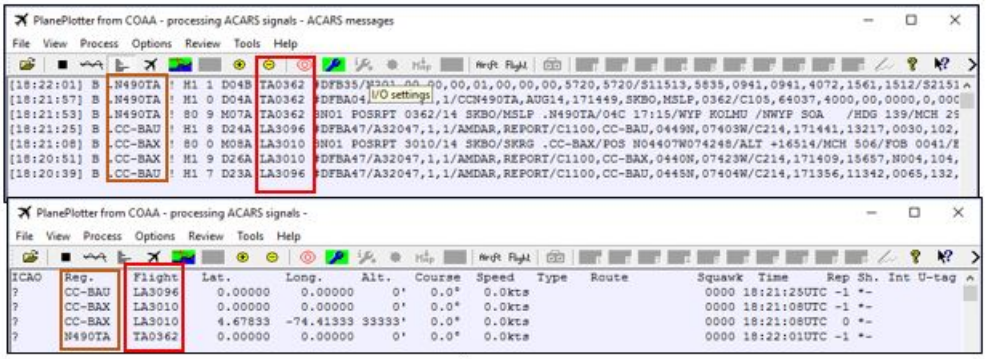

Figura 4.2: Mensajes ACARS de acuerdo a la sintonización de SDR-RADIO.

- Simultáneamente poner en modo escucha el programa PlanePloter para evidenciar efectivamente son frecuencia de operación de mensajes ACARS.

En la Figura 4.1 es un caso de la frecuencia de las diferentes pruebas que se ejecutaron. La cual se puso en modo escucha en diferentes frecuencias con una duración de un minuto y medio. Y simultáneamente en el programa se iba evidenciando las frecuencias que recibían mensajes ACARS. (Ver Figura 4.2).

Luego de simultaneas pruebas, en la tabla 4.1 se evidencia tres frecuencias de operación de señales ACARS.

Frecuencia (Mhz)	Características
131.550	En esta frecuencia no se registra mucha actividad.
131.719	En esta frecuencia se registra mayor cantidad de mensajes ACARS, pero con muy baja potencia.
131.917	No están reincidente como la frecuencia de 131.719 Mhz pero la potencia en que se reciben los mensajes es mayor a las dos frecuencias anteriores.

Cuadro 4.1: Características Frecuenciales - Frecuencias ACARS cercanas al aeropuerto Internacional el Dorado en la ciudad de Bogotá

4.2. Análisis de los mensajes ACARS en Matlab

Cuando se refiere a "Tono", se está referenciando a un mensaje ACARS que se identifica con un cambio de sonido como si escuchara una señal sinusoidal. Las grabaciones donde están contenidos los mensajes ACARS se cargan en Matlab como archivo de audio (Ver Tabla 4.2). La finalidad momentánea de este análisis, es observar el comportamiento de la señal.

Los tonos analizados son los siguientes:

Nombre (.wma)
Grabación
Un Tono
Tres Tonos
Multi-Tonos de Baja-potencia

Cuadro 4.2: Audios de prueba - Cuatro grabaciones (Audio) de las tres diferentes frecuencias de operación ACARS. Para analizar el comportamiento en Matlab

4.2.1. Audio Grabación

La señal de la figura 4.3, cuyo recuadros corresponden a los mensajes ACARS. Se tiene en cuenta que influye el componente de ruido en la señal.

4.2.2. Tres Tonos

El audio **Tres Tonos** tiene la particularidad de recibir señales de alta potencia y de larga duración de la frecuencia 131.917 Mhz. Cuando los mensajes son provenientes con características de un nivel de potencias altos, es fácil observar los mensajes ACARS de esa señal.

Ya al observar la figura 4.2 y la figura 4.3, existe un patrón de la variación del nivel de voltaje del mensaje ACARS, tal como se representa en recuadro azul de la figura 4.4.

4.2.3. Multi-Tonos de Baja-potencia

La señal de figura 4.5 es difícil de diferenciar un mensaje ACARS a comparación de las anteriores Figuras. Dicha señal se intercepto en la frecuencia de 131.719 MHz, donde los

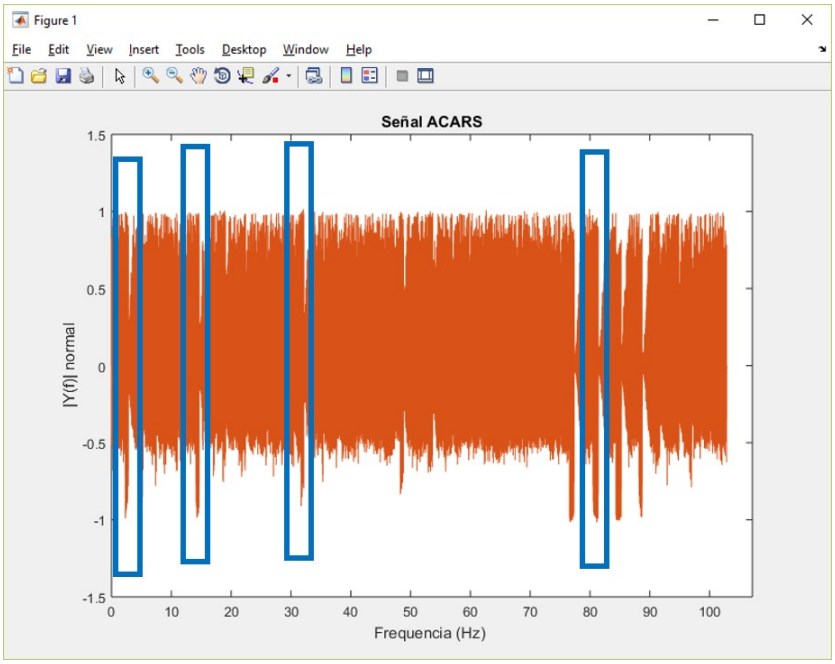

Figura 4.3: Mensajes ACARS del audio **Grabación**.

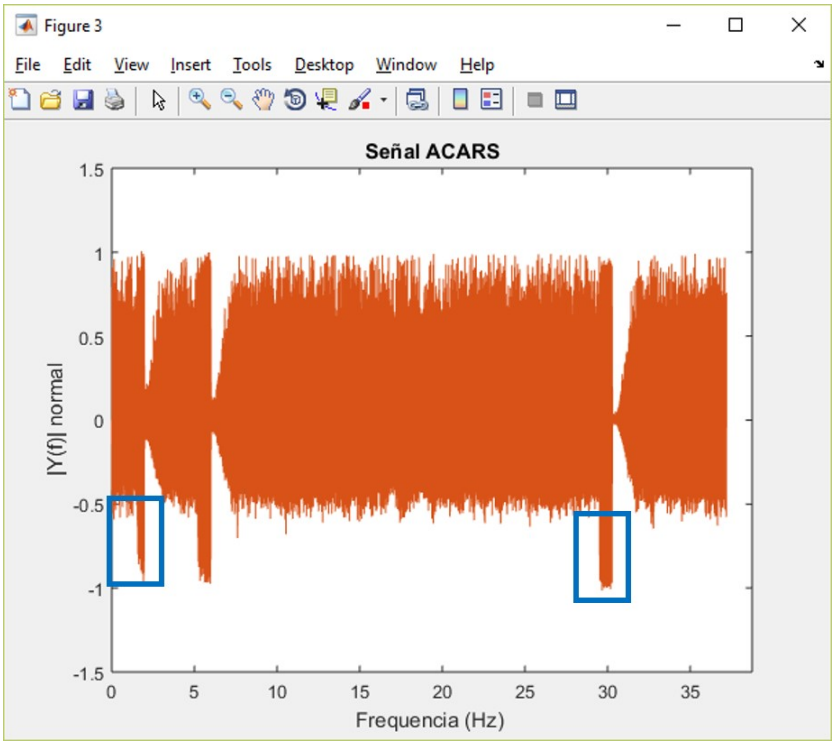

Figura 4.4: Mensajes ACARS del audio **Tres Tonos**.

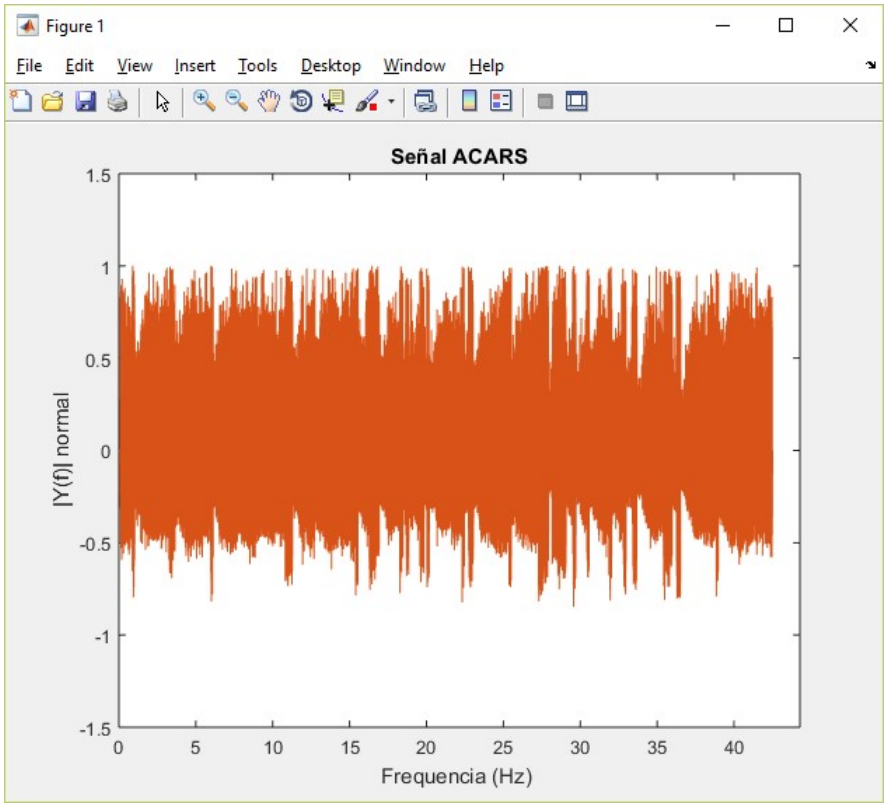

Figura 4.5: Mensajes ACARS del audio **Multi-Tonos de Baja-potencia**.

mensajes son muy reiterativos, pero con muy baja potencia.

Dada la escasez de información acerca de la frecuencia de operación del protocolo ACARS en Colombia, en el documento se comprueba e identifica dichas frecuencias de manera experimental. La metodología usada permite además de comprobar la frecuencia de operación, el comportamiento temporal de los mensajes ACARS. Pero no refleja información espectral acerca de la modulación(Demodulación) utilizada y tampoco es concluyente para establecer si la señal esta de-modulada.

Capítulo 5

I Fase: *Demodulación*

El presente capitulo desarrolla el programa de GNURadio para la demodulación de señales ACARS proveniente del espectro radioeléctrico por medio del dispositivo RTL-SDR. Inicialmente se contextualiza los conceptos de la modulación ACARS. Antes de empezar a diseñar el programa de GNURadio, Documentamos el procedimiento para ajustar el corrimiento de frecuencia que tienen los dispositivos RTL-SDR. El programa de GNURadio se basa en dos módulos: La recepción y la demodulación ACARS. El modulo de recepción son todos los componentes necesarios para ajustar todos los errores de frecuencia y fase que pueda incurrir en la señal recibida por el RTL-SDR. El modulo de Demodulación se basa en la conversión de la señal en bits ACARS mediante un bloque predeterminado en GNURadio. Luego de desarrollar todo el programa de GnuRadio, se le realiza pruebas en un escenario real para realizar los ajustes necesarios para optimizar el sistema de tal forma que pueda demodular los mensajes ACARS de manera correcta.

5.1. Demodulación ACARS

ACARS empezó a funcionar bajo el rango de frecuencias de VHF con una demodulación AM bajo la modulación digital MSK (No es más que la misma demodulación FSK pero con fase continua) o CPFSK (Modulación de fase continua) (Con k=0.5). La Modulación MSK es un caso especial de la Modulación FSK. Se basa en dos estados de frecuencias representando los bits '1' y '0' respectivamente. La diferencia se basa en que MSK es un caso de fase continua, es decir evita discontinuidades de fase o cambios bruscos en la forma de onda cuando se genera la transición de una frecuencia a otra frecuencia, y permite un ancho de banda más eficiente.

Bajo el estándar ARINIC 618 describe el funcionamiento básico de la comunicación aire/Tierra de las aeronaves usando el protocolo ACARS. De acuerdo con la teoría de la modulación MSK, la frecuencia asignada para el símbolo '1' es 1200HZ con fase 0 y para el símbolo '0' se asigna una frecuencia de 2400HZ con fase de 180. La velocidad de transmisión por el medio es 2400 bps. Además, tiene una señalización ACARS o "Señal ACARS", que en términos prácticos es un cambio de estados de los bits obtenidos por la demodulación MSK. [9]

$$S_{msk} = A\cos\left(2\pi f_c \pm \frac{\pi p_k}{2T_b} + \phi\right) \tag{5.1}$$

$$\begin{aligned} f_1 &= (f_c - \frac{1}{2T_b}) \\ f_2 &= (f_c + \frac{1}{2T_b}) \end{aligned} \tag{5.2}$$

La señalización ACARS funciona de la siguiente manera: Si el bit original se mantiene en un mismo estado (El anterior bit es igual al actual) el bit resultante se intercala. Caso contrario, si el bit original actual es diferente al bit anterior, el bit resultante mantiene el estado del bit resultante anterior [9]. Por ejemplo:

$$\begin{aligned} &P = \{0,1,0,0,1,1,1,0,0,1,0\} \quad Bits \quad originales \\ &P' = \{0,0,0,1,0,1,1,0,1,0,0\} \quad Mensaje \quad ACARS. \end{aligned} \tag{5.3}$$

La demodulación hace ese procedimiento inverso; de-modula la señal MSK, luego ejerce la señalización inversa para obtener los bits originales.

Para ejercer la demodulación MSK existe dos formas: de forma COHERENTE y NO-COHERENTE. La modulación NO-COHERENTE se basa en multiplicar por la portadora previo a un filtro pasa-bajo. Y luego ejerce una decisión en base a un envolvente en donde se hace una comparación y decisión. La forma COHERENTE se basa en un PLL retroalimentado con VCO, la cual detecta el cambio de fase y así determinar la salida digital de dicho

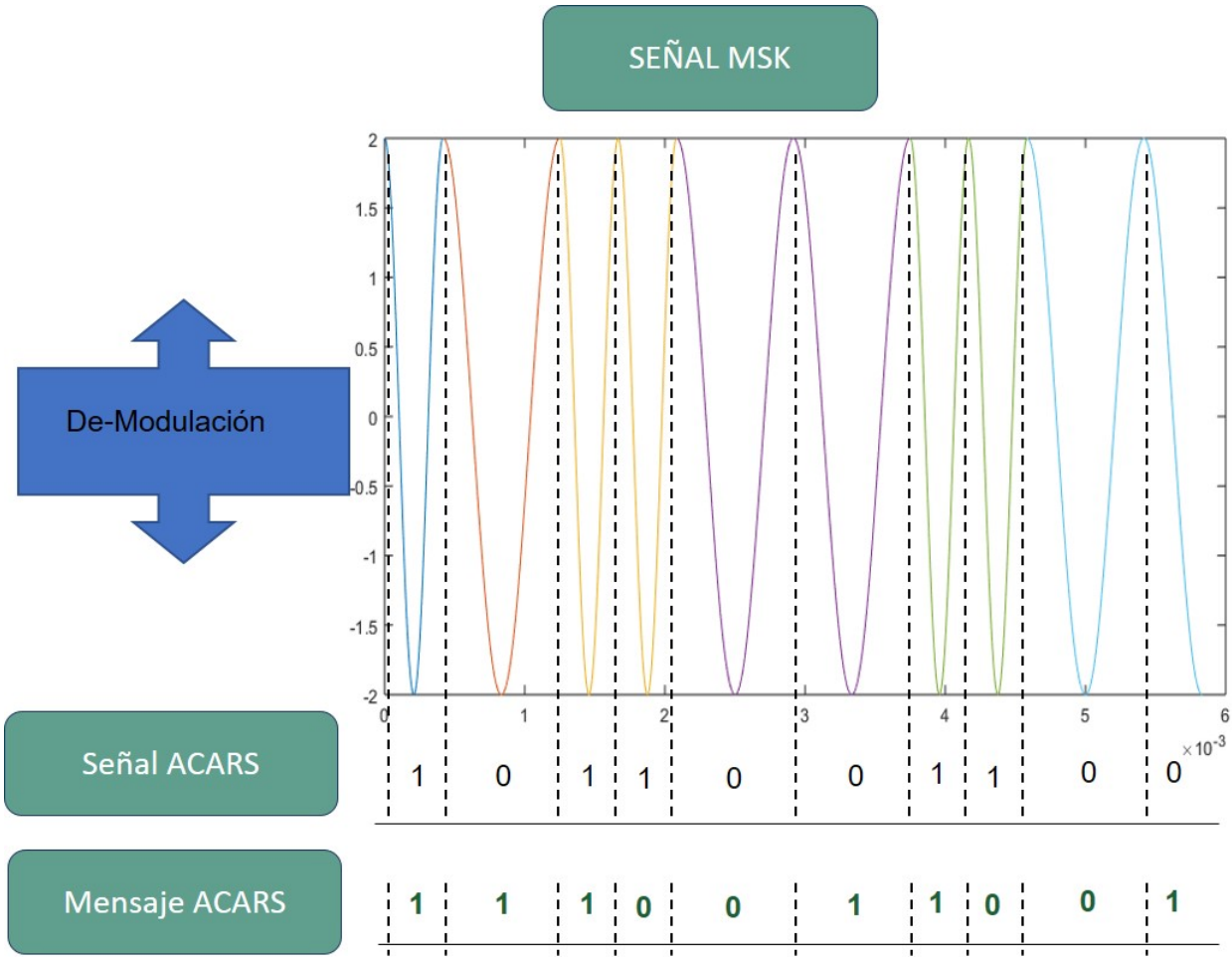

Figura 5.1: Resumen de la demodulación de mensajes ACARS.

demodulador. De igual manera existe un método para volver la forma NO-COHERENTE en una demodulación Coherente, mediante el método de recuperación de la portadora. Este permite entrar en sincronía en frecuencia y fase de acuerdo con la señal de entrada. Esta se le denomina como la demodulación OQPSK especial para MSK.

La comparación de las dos formas de demodulación frente a la relación de SNR y el BER, existe una mejor eficiencia con la forma COHERENTE. Existe menor probabilidad a mayor capacidad de existencia de ruido. Por ello implica usar una demodulación coherente, aunque la dificultad está en la complejidad circuito. Mientras que la demodulación NO-COHERENTE es más sencilla de ejercer, pero implica resultados inferiores respecto a la relación SNR y BER.

5.2. Ajuste PPM: Corrección del Offset de frecuencia

Un offset de frecuencia en un sistema de recepción puede ser crítico a medida que el ancho de banda de la señal a recibir sea estrecho. Además, se ha experimentado que dicho corrimiento no es fijo sino variable. Es decir, en un periodo de tiempo la frecuencia de la portadora del RTL-SDR oscila en una frecuencia (Fo) diferente a la frecuencia banda base (Fc). Por ello es importante hacer la corrección del corrimiento de frecuencia.

Todos los dispositivos RTL.SDR tienen un corrimiento de frecuencia que va desde los 5Khz hasta los 60Khz. Todos los dispositivos no tienen el mismo Offset de frecuencia, de manera que cada dispositivo RTL-SDR se debe realizar dicha prueba. El valor o la variable que corrige dicho defecto se denomina PPM (Partes por millón), y esta medido por la siguiente formula:

$$PPM = \frac{f_{offset}}{f_c * \quad 10^{-6}}' \tag{5.4}$$

Donde f_{offset} es la diferencia entre la frecuencia que está transmitiendo y la frecuencia que esta midiendo el RTL-SDR. Es muy importante mantener este cálculo, ya que dicha diferencia va cambiando a media que se va aumentando la frecuencia. Es decir, si el RTL-SDR recibe una señal con una frecuencia promedio a 100MHZ el offset aproximadamente será 7 KHz; Pero si se transmite a una frecuencia mayor a 500MHz el offset AUMENTA a 30KHz aproximadamente. De manera concluyente, a medida que aumentamos la frecuencia de operación del RTL-SDR, más crítico va a ser el offset de frecuencia.

De acuerdo con lo anterior, no es sensato definir una diferencia experimental o una corrección manual para cada frecuencia de recepción del dispositivo RTL. Más bien se debe definir un único valor para todo el rango de frecuencias de dicho dispositivo. La afirmación anterior se logra definiendo un PPM del offset de frecuencia.

Figura 5.2: Generador de Radio Frecuencia HAMEG. Con Frecuencia de 501MHz y una potencia de -37.5 dBm

La metodología de la presente prueba es la siguiente:

- Ajustar una frecuencia específica a un nivel de potencia cercana a los -35dBm en un generador de radio frecuencia.
- De forma experimental a través de SDR Console intentar aproximar una diferencia entre la Frecuencia RF (Fc) y la frecuencia que mide el dispositivo RTL-SDR (Fo). Calcular un PPM. *(Considerando el error de visualización del ojo humano).*
- Comparar el anterior resultado, con la herramienta de Matlab de RTL-SDR para calcular un PPM.
- Visualizar el efecto que tiene PPM en GNU Radio.

Al medir la diferencia entre Fc y Fo de manera espectral, se debe ajustar un desmuestreo de la frecuencia de muestreo menor a los 400Khz/s. Para obtener una mejor resolución espectral, para disminuir el error del ojo humano. (Tener en cuenta teoría de Nyquist, pues el ancho de banda de una señal está limitado por su frecuencia de muestreo).

Como primera instancia ajustamos una frecuencia mayor a los 400Mhz con la respectiva nivel de potencia (El nivel de potencia debe ser considerado según las interferencia que se encuentren en el espectro. Idealmente se debe hacer las pruebas donde no haya interferencia de señales en ese rango de frecuencia).

En otro lado de la ecuación, ajustamos en el dispositivo RTL-SDR a través de SDRConsole, exactamente la frecuencia que se encuentra en la imagen anterior.

5.2.1. SDR-CONSOLE

Evidentemente existe un offset de frecuencia negativa (Corrimiento hacia la izquierda). Para aumentar la resolución espectral disminuimos su frecuencia de muestreo para ajustar

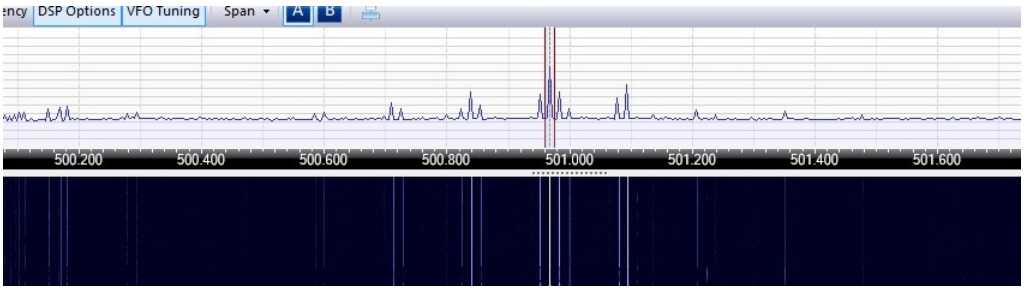

Figura 5.3: Señal del generador RF en el dominio de la frecuencia recibido por el RTL-SDR. A simple vista la señal no está ubicada exactamente en los 501 MHz

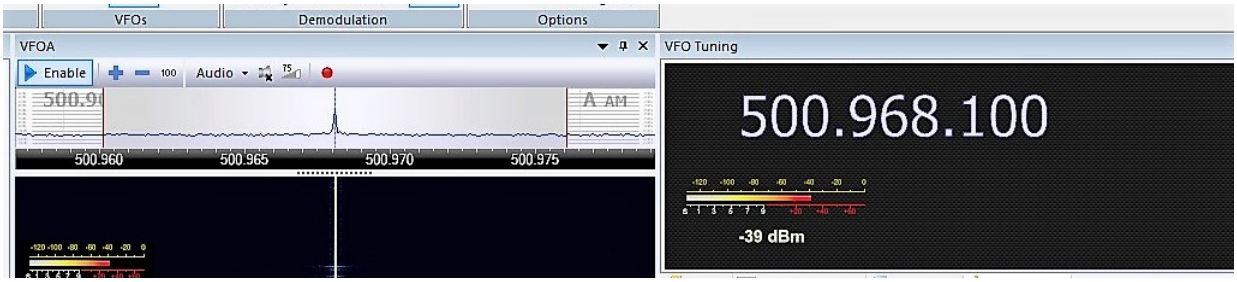

Figura 5.4: Una vista detallada del offset que existe en el RTL-SDR. Frecuencia real donde el RTL-SDR esta sintonizando la frecuencia de 501 MHz del generador HAMEG

mejor la frecuencia del dispositivo Fo.

Visiblemente al ajustar y medir la frecuencia de operación, existe una diferencia entre la frecuencia que se está transmitiendo (501 Mhz) y la frecuencia que está recibiendo en el RTL (500.9581 Mhz). Es decir, existe un offset igual a 31.9Khz. De manera que si calculamos el valor de PPM

$$PPM_{exp} = \frac{31900}{501 * 10^6 \quad * \quad 10^{-6}}$$
$$PPM_{exp} = 63{,}67 \tag{5.5}$$

5.2.2. Matlab *(Simulink)*

Para comparar y definir el alcance del cálculo anterior, realizamos la misma prueba, pero en un cálculo automatizado realizado en un programa Simulink en Matlab. Matlab realiza un calculo espectral en donde encuentra todos los picos de la señal de manera espectral, y luego

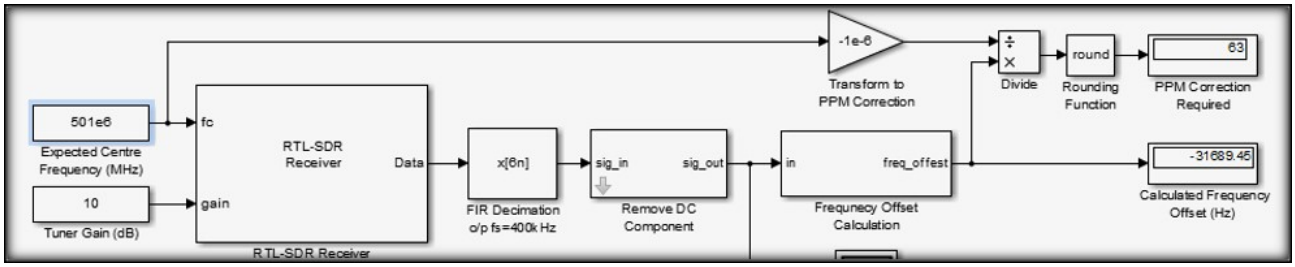

Figura 5.5: Calculo de PPM en Simulink para una frecuencia de 501 MHz

compara entre todos los valores picos, el valor más grande. Entonces el valor más grande será la frecuencia Fo del dispositivo RTL.

Evidentemente en un valor más exacto de offset de frecuencia, su PPM en un valor redondeado es exactamente igual al cálculo experimental realizado en SDRConsole. En la siguiente imagen espectral de Matlab se puede evidenciar más claramente el corrimiento de frecuencia.

Ahora para soportar la afirmación de que a medida que disminuimos la frecuencia de operación el offset de la señal disminuye, son directamente proporcionales, pero su valor PPM se mantiene. Para ellos ajustamos el generado RF a la frecuencia de operación ACARS hallado experimentalmente, es decir adicionando el error de offset frecuencial de131.719MHz.

Tal como se afirmó, el offset de frecuencia disminuye a 8.2KHz, pero su valor PPM se mantiene entre el rango de 62 y 63. Entonces se puede concluir que el valor nominal de la frecuencia ACARS no debe ser 131.719 MHz sino 131.7108MHz, que en un valor aproximado se maneja en 131.710Mhz.

Entonces al ajustar la simulación a 63 PPM, entonces el RTL se ajusta automáticamente a la frecuencia base del mismo dispositivo de manera correcta, solo con un offset de frecuencia de 100 Hz, corrimiento de frecuencia que el mismo dispositivo es capaz de controlar. Pues el dispositivo no es capaz de enganchar la portadora cuando el offset de frecuencia es mayor a 1 KHz.

$$PPM_{exp} = \frac{8200}{131{,}719 * 10^{6} \quad * \quad 10^{-6}}$$
$$PPM_{exp} = 62{,}65 \tag{5.6}$$

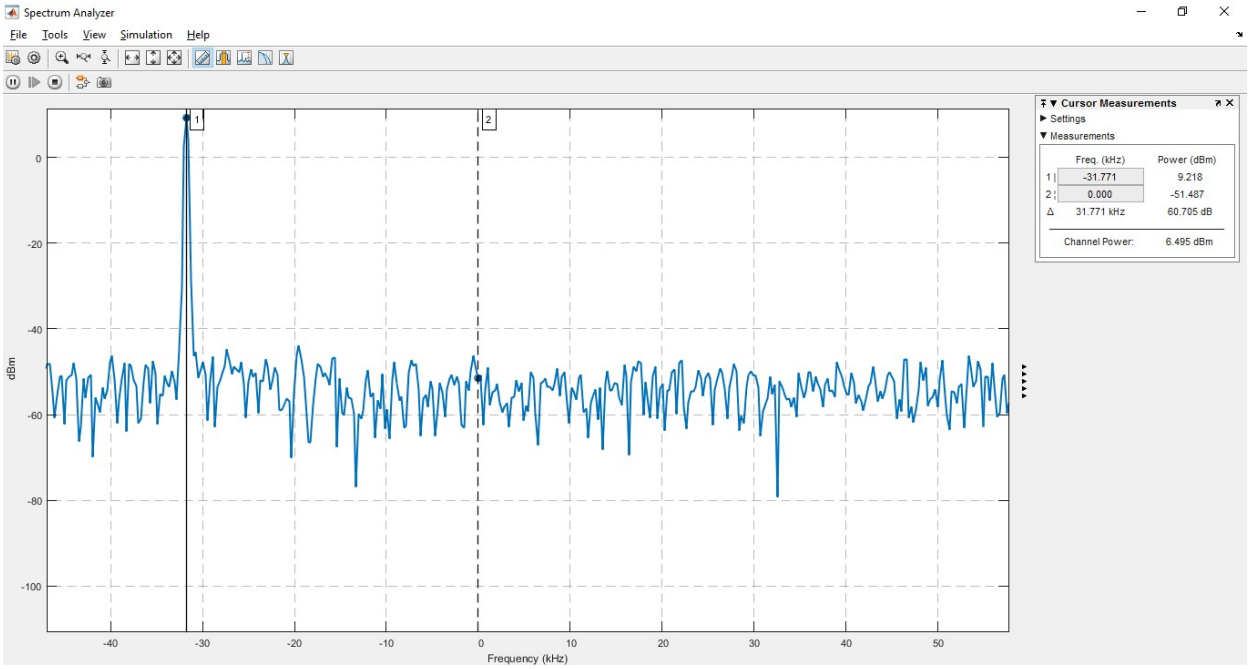

Figura 5.6: Imagen espectral del corrimiento de frecuencia de la señal recibida por el RTL-SDR.

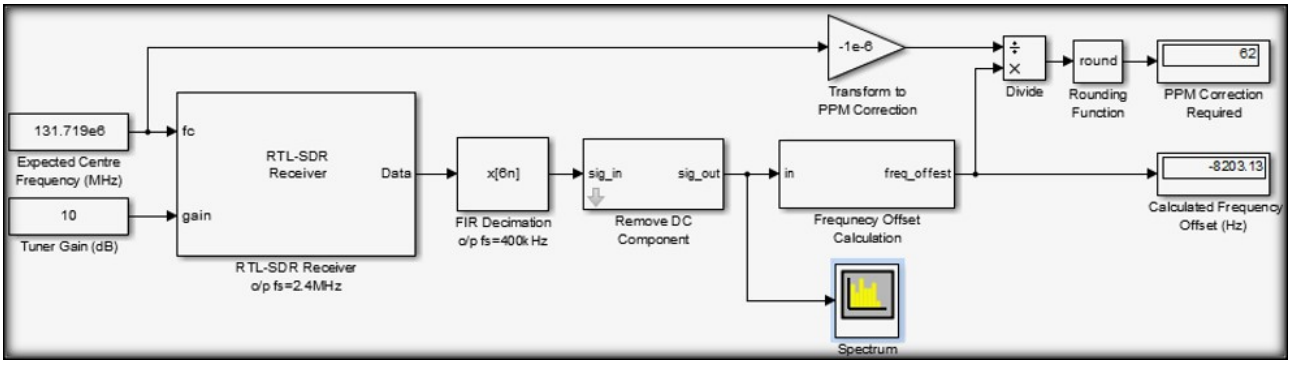

Figura 5.7: Calculo de PPM en Simulink para una frecuencia de 131.719 MHz. Dicha frecuencia se ajusta también en el henerador HAMEG

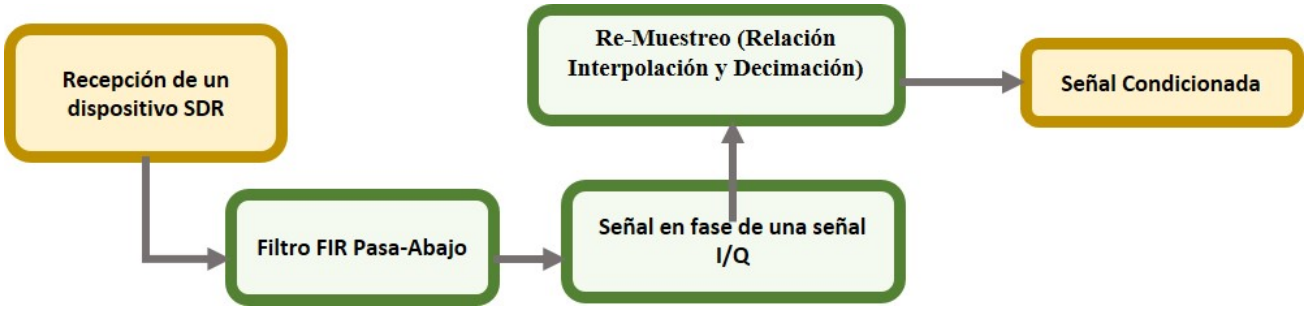

Figura 5.8: Procedimiento de recepción de señal ACARS del primer Sub-modulo.

5.3. Implementación de módulo ACARS: Recepción

Establecido ya las frecuencias de operación del protocolo ACARS-VHF en la ciudad de Bogotá y la teoría de demodulación digital banda-base y pasa-banda, Filtros FIR, Interpolación, Decimación y la definición de software definida por Software (SDR) para comprender este sistema de recepción ACARS.

El sistema de demodulación y decodificación se basa en los trabajos en GNU-Radio (gr-acars2) desarrollados por Antoine Neuenschwander. Estos desarrollos resumen el trabajo de un módulo de recepción ACARS mediante un dispositivo SDR en tiempo real presentado en el presente trabajo.

El módulo de recepción ACARS se pude dividir en dos sub-módulos. El primer sub-modulo se basa en acondicionar la señal ACARS recibida por el dispositivo SDR en condiciones óptimas para el siguiente sub-modulo (DSP). El segundo sub-módulo recibe la señal condicionada para ejercer la demodulación y la decodificación de una manera óptima.

El primer sub-modulo (Ver Figura No 5.8) cuando hablamos de condicionar la señal, se refiere hacer el procesado de señal para corregir errores de recuperación tal como la sincronización en fase y en frecuencia de la señal recibida por el dispositivo RTL. Y de igual manera efectuar un filtro pasa-abajo para procesar la señal sobre la banda seleccionada. Y aplicar una relación de submuestreo y desmuestreo para entregar una señal muestreada a 48KHz.

La idea de explicar cada sub-módulo se fundamenta en explicar en detalle cada funcionalidad de cada bloque funcional explicado en la Figura No 1, pero plasmado en GNU-Radio.

Inicialmente debe disponer de un bloque para adaptar el dispositivo SDR a GNURadio, que para el desarrollo de este proyecto se basa en un dispositivo RTL-SDR. Funcionalmente este se encarga de adaptar la señal recibida por el dispositivo para poderlo procesar en la plataforma de GNURadio. Los parámetros para destacar son la frecuencia de sintonización (Frecuencia de operación ACARS), la frecuencia de muestreo (Redondea desde los 500KHz

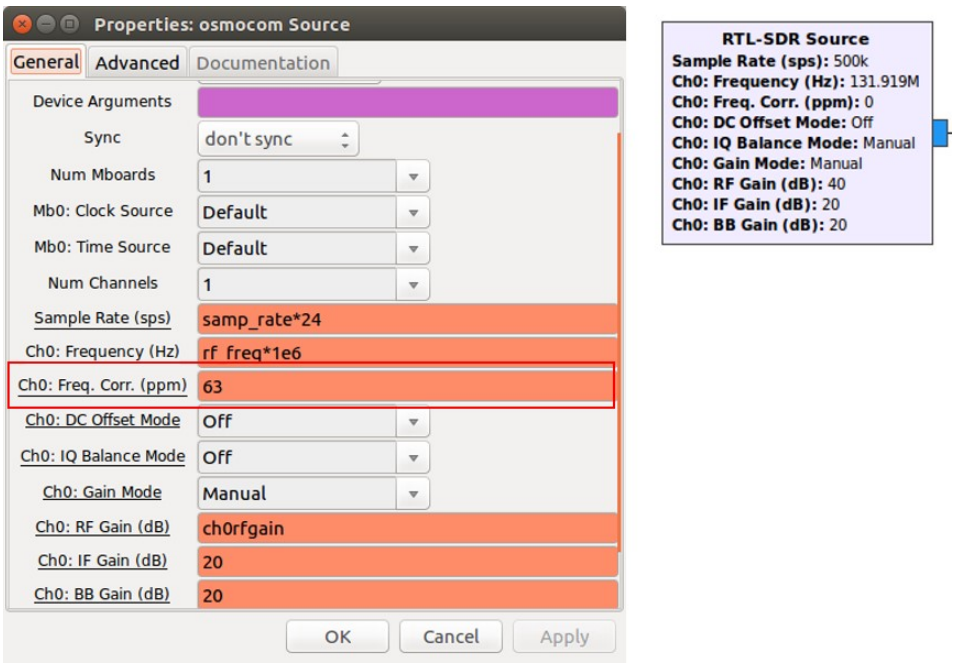

Figura 5.9: RTL-SDR Source con el ajuste de PPM.

a los 2MHz) y la ganancia o amplificación del ancho de banda a recibir. (Ver Figura 5.9

Ya con la señal captada, es necesario limitar el ancho de banda de trabajo a través de un filtro Pasa-Abajo. Dicho diseño del filtro pasa-bajo se puede facilitar a través Filter Firdes, que es un programa de GNURadio que se puede ejecutar a través de un emulador de python. La idea de usar la aplicación, es conocer la respuesta del filtro y aplicarlo en algún filtro FIR de GNU-Radio. (Ver Figura 5.10)

Los parámetros de diseño del filtro son la ganancia, la frecuencia de muestreo (Parámetro que define el ancho de banda de la señal a través de la teoría de Nyquist), la banda de transición que viene siendo la diferencia entre FIN DE PASABANDA e INICIO DE BANDA DE DETENCIÓN y la atenuación del filtro.

Luego de adecuar la señal en el ancho de banda adecuado, solo resulta separar la señal en fase de la señal IQ a través de un módulo que convierta un dato complejo a un dato real. (Ver Figura 5.11)

La relación de Re-muestreo es para ajustar dicha señal a un muestreo de 48KHz. Dicho muestreo es necesario para que la demodulación del segundo Sub-Modulo sea óptima. La relación funciona de la siguiente manera:

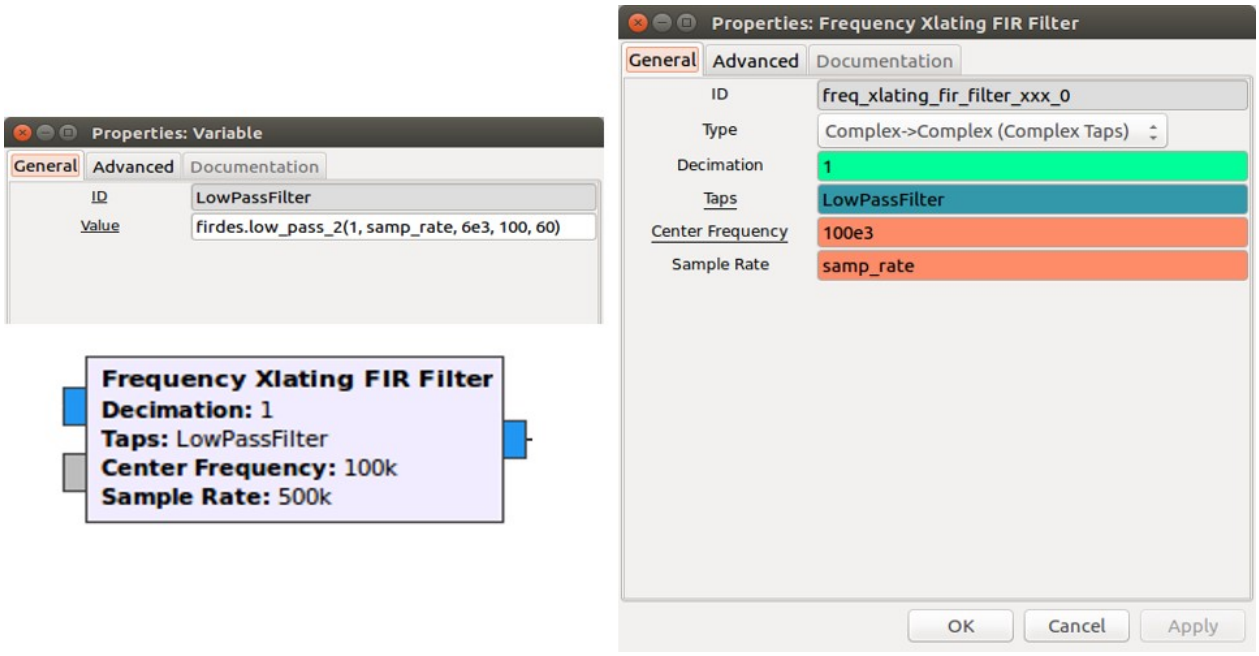

Figura 5.10: Bloque del filtro Pasa-Bajo.

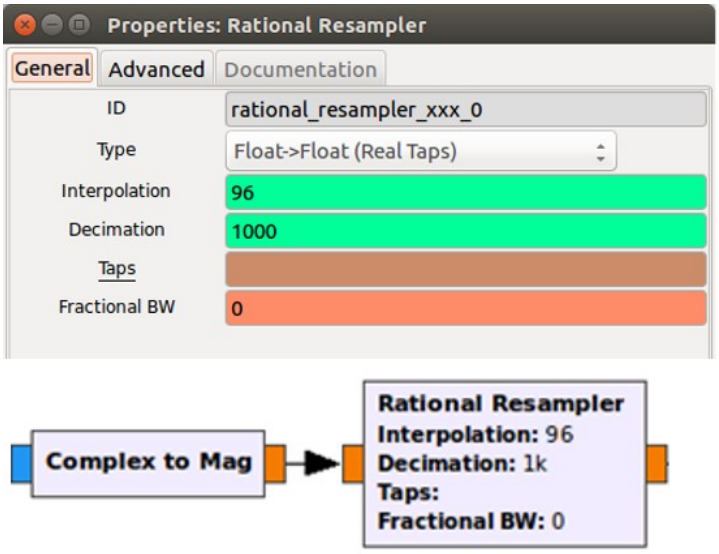

Figura 5.11: Conversión de una señal compleja *(Señal I/Q)* a una señal real o en fase. Remuestreo de la señal a través de la relación Decimación e interpolación *(Desmuestreo y Muestreo)*

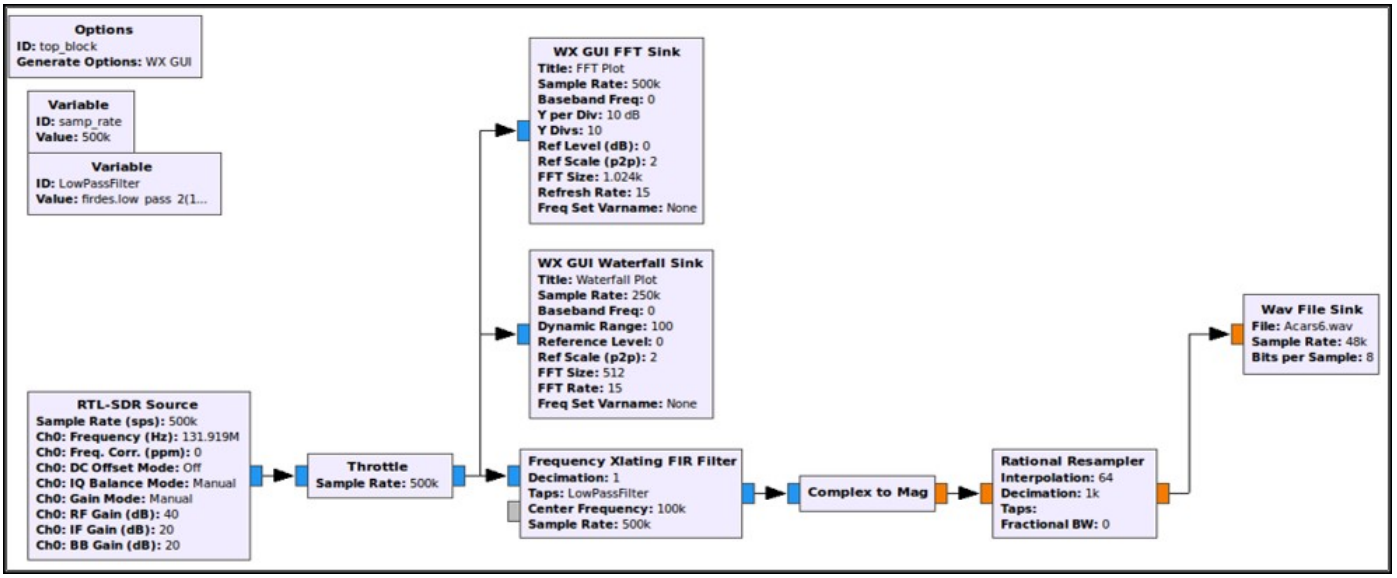

Figura 5.12: Función de filtro en el espectro de los mensajes ACARS simultáneamente.

$$RelationResampler = \frac{Interpolacion}{Decimacion} \tag{5.7}$$

$$Sampler_{48Khz} = Relation * SampleRate(in) \tag{5.8}$$

Si ***"Sample rate"*** es 500KHz entonces debe existir una relación tal que la señal quede muestreada a 48KHz(Aplicar la ecuación 5.7 y con el resultado usar la formula 5.8):

$$\begin{aligned} RelationResampler &= \frac{96}{1000} \\ Sampler_{48Khz} &= 0{,}096 * 500Khz \\ Sampler_{48Khz} &= 48Khz \end{aligned} \tag{5.9}$$

De manera que al simular el programa del Sub-módulo de la Figura No 5.12, se obtiene los siguientes resultados: (Ver Figura No 5.13)

En la figura 5.13, se puede evidenciar el funcionamiento del sub-módulo de recepción desde cuando capta la señal mediante el bloque de RTL-SOURCE en el diagrama de **Waterfall**. Igualmente, los dos diagrama de inferiores se evidencia los mensajes ACARS captados por el dispositivo SDR. Dichos mensajes pasan por un filtro pasa-abajo.

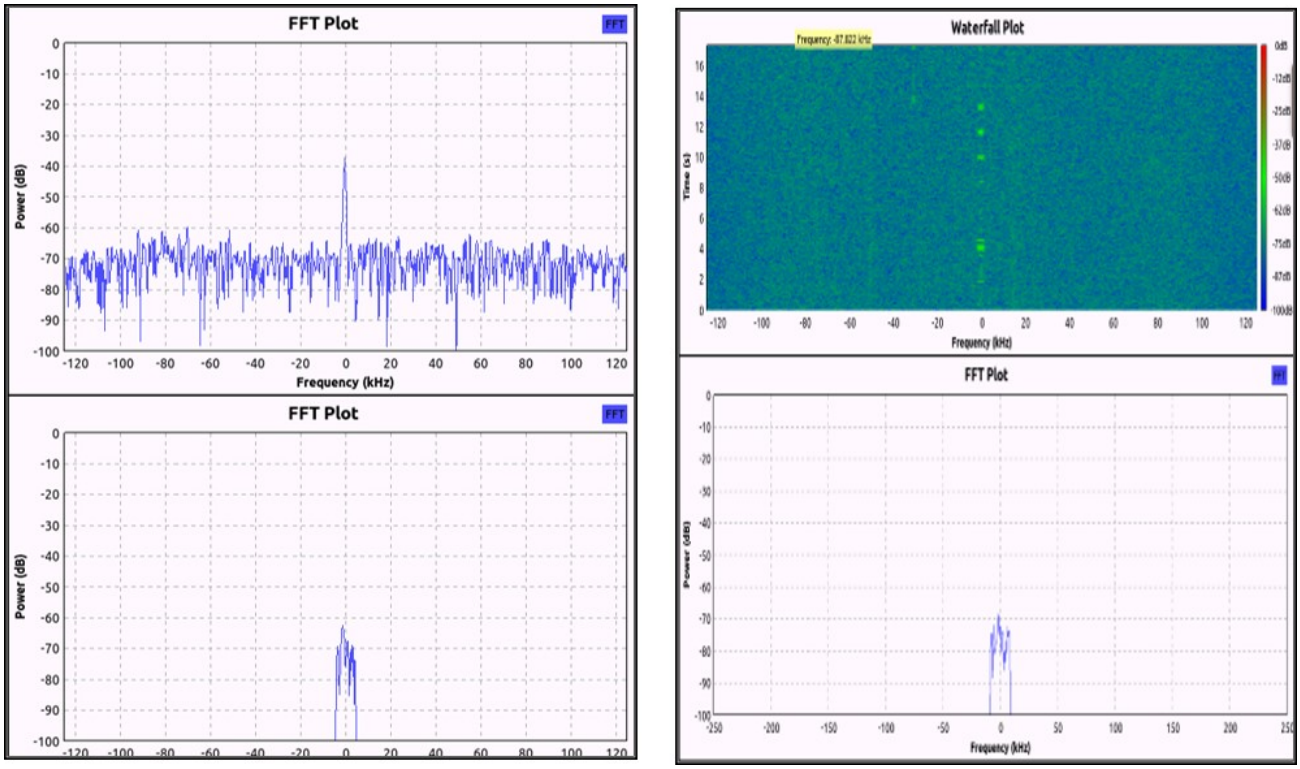

Figura 5.13: Sub-módulo de recepción ACARS

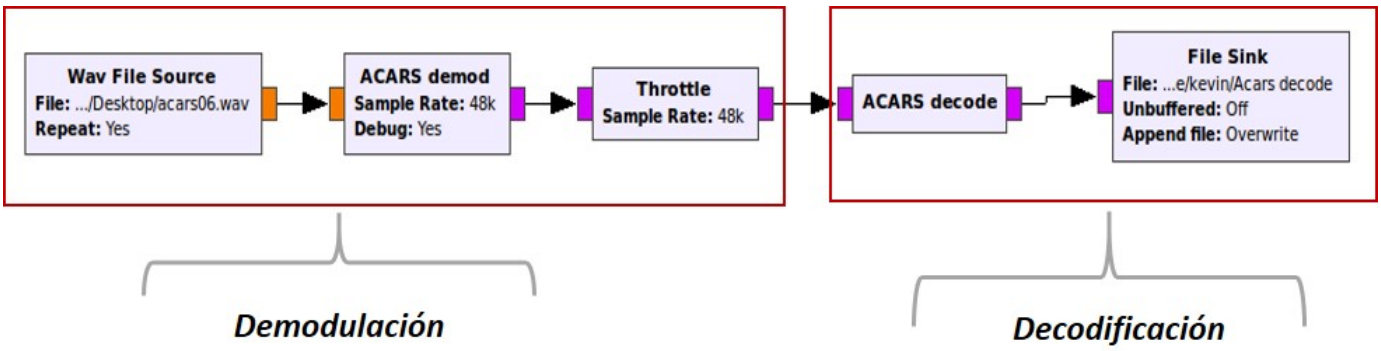

Figura 5.14: Diagrama de bloques de la demodulación ACARS en GNU-Radio.

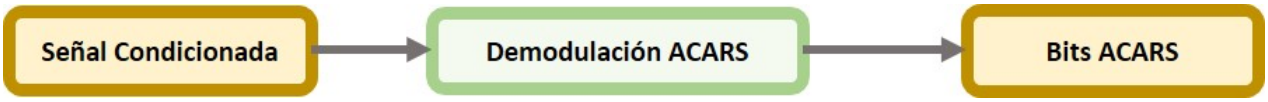

Figura 5.15: Procedimiento de demodulación de la señal ACARS del Segundo Sub-modulo.

5.4. Implementación de módulo ACARS: Demodulación

El segundo Sub-Módulo de la Figura 5.15 trata de hacer el procedimiento de demodulación de la señal ACARS de acuerdo con los parámetros de decisión de bits de la Demodulación MSK-FSK. El resultado de dicho de Sub-modulo es entregar paquetes de ocho bits *(Bytes)*.

De manera que la combinación de los dos sub-módulos suman la recepción de una señal ACARS con un dispositivo SDR en tiempo real. El proceso de Decodificación es el bloque faltante para el segundo sub-módulo. Cuando se habla de una señal condicionada, quiere decir que se entrega una señal limitada a un ancho de banda predefinido a un muestreo de 48Khz. Es el parámetro mediante por el cual la demodulación se puede considerar óptima. Los bloques necesarios para la demodulación en GNU-Radio se presenta en la Figura 5.14

La demodulación se basa en la modulación MSK-FSK. El proceso de Decodificación se explica en la siguiente fase, ya que esta se basa en una teoría que respalda dicha decodificación utilizada en este trabajo. El resultado se evidencia en la figura 5.16.

Por cada byte, hay 7 bits de información donde el LSB y MSB están invertidas. También existe un bit de paridad. (Ver Figura 5.17). En la Figura 5.16, se evidencia que la frecuencia de muestreo debe ser de 48KHz. Y la velocidad de bits en ACARS es equivalente a 2400Hz.

Los parámetros esenciales a tener en cuenta fueron los siguientes:

- La señal debe estar muestreada a 48Khz para poder ser de-modulado correctamente.

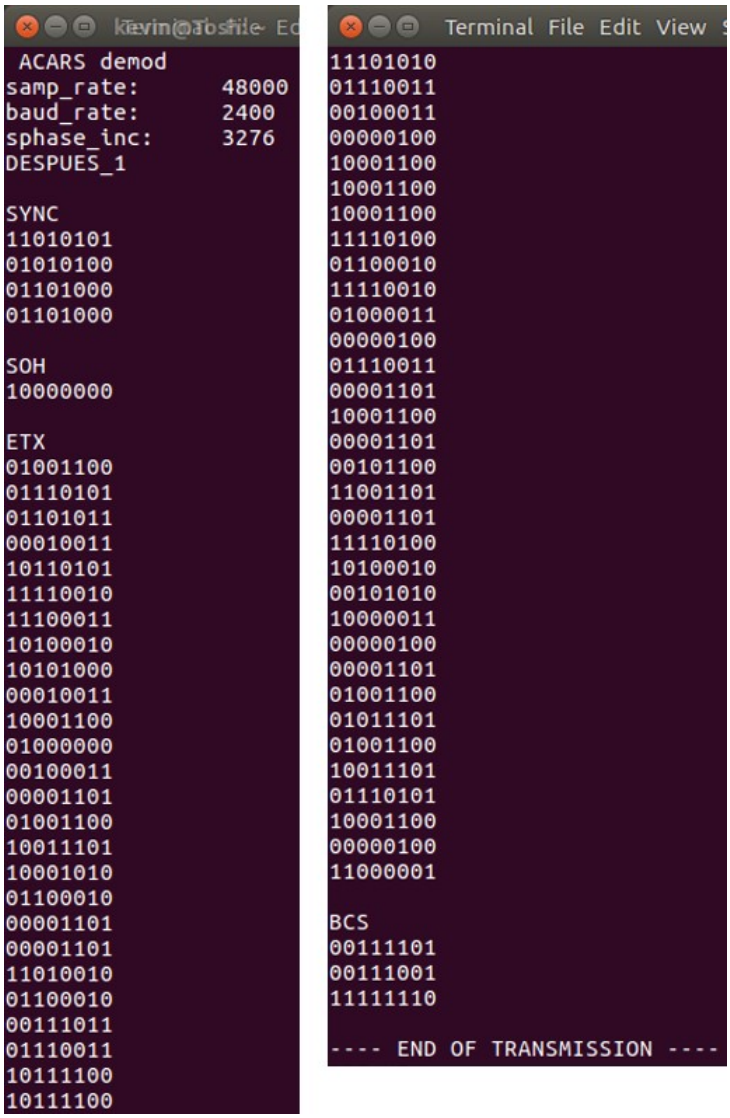

Figura 5.16: Bytes resultante de la señal ACARS.

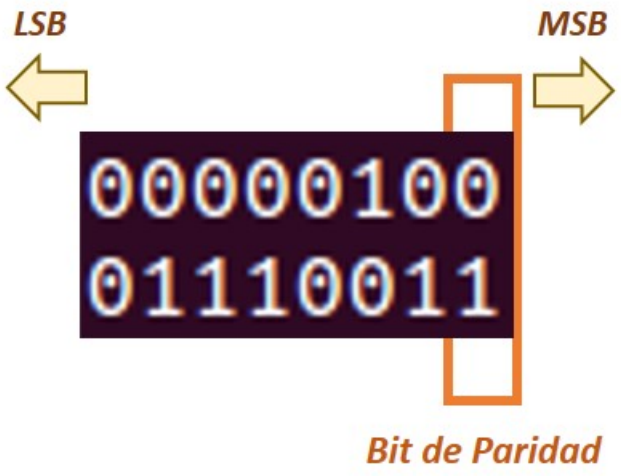

Figura 5.17: Bytes resultante de la señal ACARS.

- La señal captada debe estar limitada por un filtro, de tal manera que pueda ser eficientemente espectral sin necesidad de perder información.

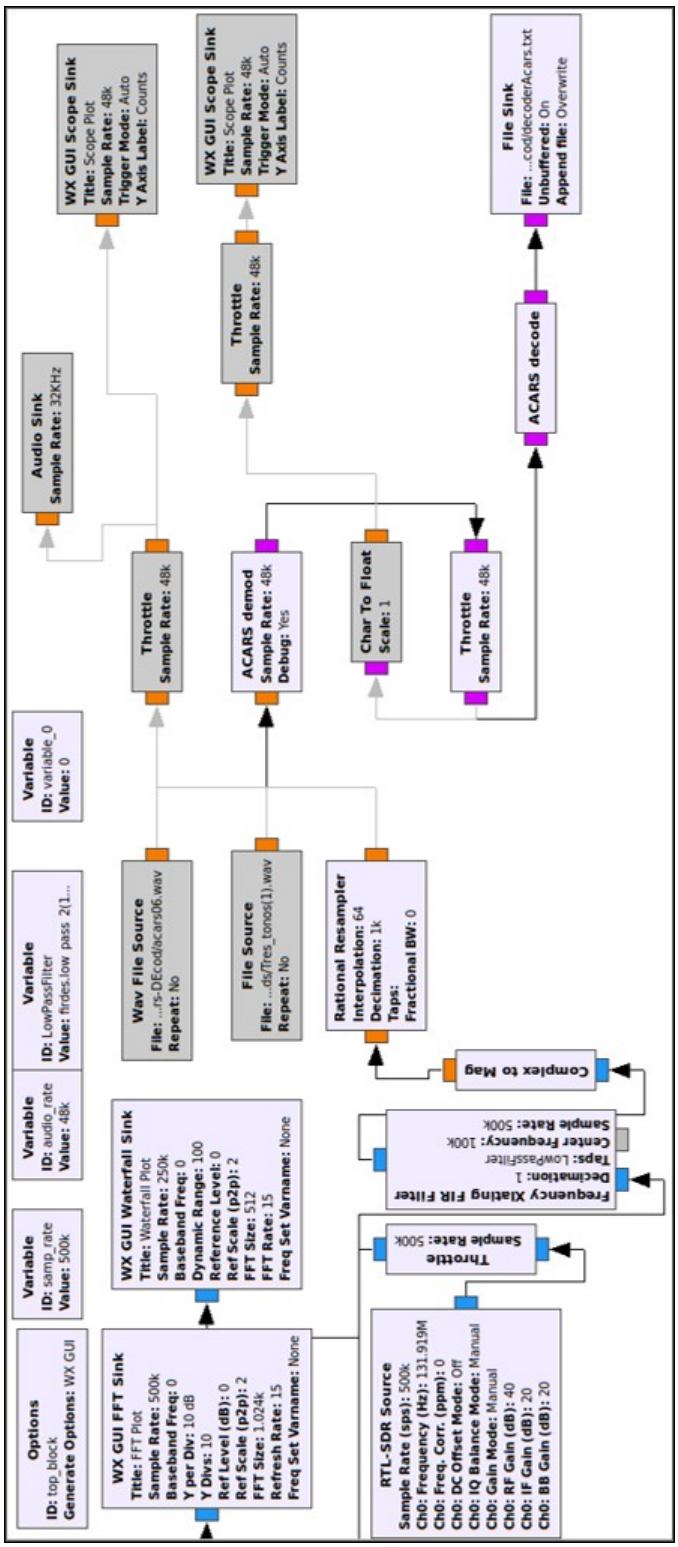

Figura 5.18: Prototipo Final para captar señales ACARS a través de un RTL-SDR en tiempo real .

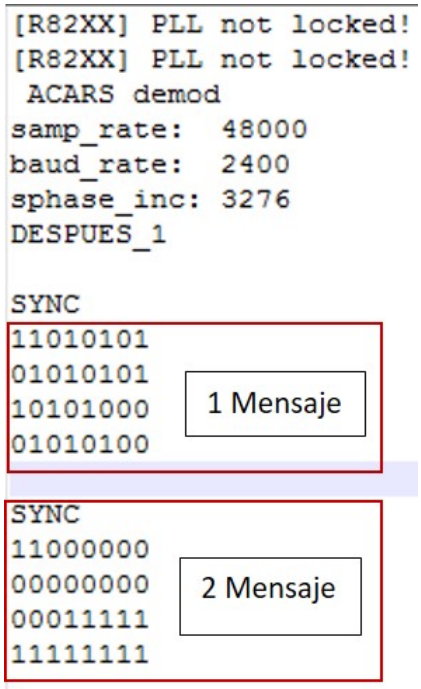

Figura 5.19: Resultado del proceso de recepción y demodulación en tiempo real con el dispositivo RTL-SDR.

5.5. Demodulación en tiempo real del prototipo: Archivos .WAV

En la anterior prueba se verifica la funcionalidad del sistema de cumplir la demodulación a partir de archivos .WAV. La finalidad de la presente prueba analizar el comportamiento del prototipo para Demodular mensajes ACARS en tiempo real.

Al realizar el proceso de recepción y demodulación de manera continua y en tiempo real. Evidentemente alcanza a captar mensajes, pero solo logra Demodular solo un campo de todo el mensaje ACARS. Es decir, en dicha imagen logra captar dos mensajes.(Ver Figura 5.19)

El parámetro SYNC, se cuenta para sincronizar el inicio del mensaje, de manera que, sí detecta el inicio del mensaje pero no de-modula el resto mensaje. Entonces, se identifica las dificultades del prototipo, para definir si son errores de la recepción o errores de la demodulación del sistema.

La metodología es la siguiente: El resultado de la recepción de la señal, es decir antes de Demodular dicha señal se guarda en un archivo .WAV. Para luego Demodular dicho archivo .WAV en otra instancia de la prueba. Como se puede apreciar en la figura 5.20.

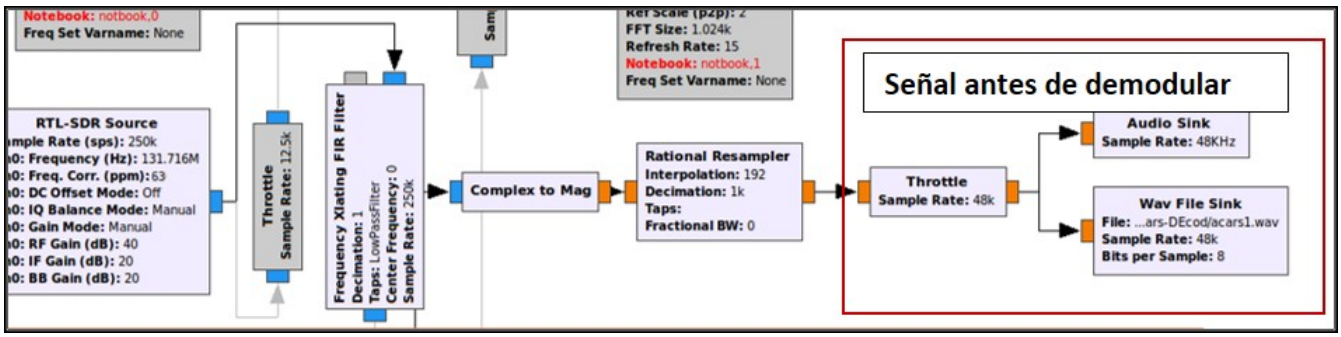

Figura 5.20: La señal captada por el RTL-SDR se guarda en un archivo .WAV, con la finalidad de analizar los problemas de demodulación en tiempo real.

A partir del mismo archivo .WAV obtenido en la anterior prueba, se realiza la demodulación en otra prueba (Ver Figura 5.21):

Se logra Demodular el archivo .WAV en el programa de la Figura 5.21, cuando en tiempo real no logro el prototipo Demodular los mensajes ACARS de manera completa. Entonces se logra mediante la presente prueba definir que existe problemas en el módulo de recepción, es decir, el proceso de adecuación de la señal a través del filtro pasa-abajo y desmuestreo.

En conclusión, se deja abierto un análisis más abierto desde el punto de vista de los problemas que puedan estar pasando en términos de sincronización de la portadora, frecuencia y fase de la señal recibida. También hay que analizar el efecto de la tarjeta sonido en el prototipo, ya que al guardar un archivo .WAV se está utilizando la tarjeta de sonido. Pues la tarjeta de sonido si logra adaptar la señal a 48KS/s, función que no logra el proceso de desmuestreo del prototipo de la figura 5.18en el bloque Resampler.

5.6. Prueba de desmuestreo de la tarjeta de sonido

Teniendo en cuenta la dificultad del presente prototipo Demodular completamente un mensaje ACARS en tiempo real. Los resultados de la anterior prueba, concluye que solo es capaz de Demodular los mensajes si convierte la señal entrante en un archivo .WAV mediante la tarjeta de sonido del equipo.

En esta prueba es determinar cuál debe ser el factor de desmuestreo más óptima para una tarjeta de sonido. Es decir, cada cuanto “samples” se deben eliminar desde la frecuencia de muestreo del RTL (En orden de los MS/s) hasta la frecuencia de la tarjeta de sonido (48KS/s) para que pueda Demodular en tiempo real.

Sabiendo que una mayor cantidad de muestras implica una mayor cantidad de combina-

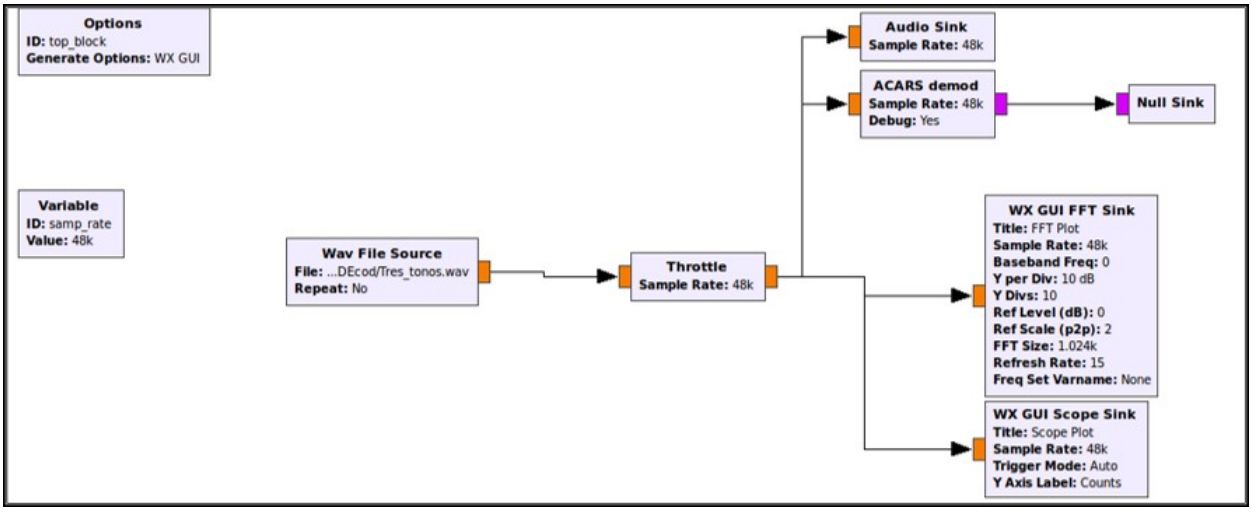

Figura 5.21: Programa de GNU-Radio para demodular la misma señal captada por el RTL-SDR por medio de un archivo .WAV

ciones (Un número mayor de niveles) o un rango dinámico mayor, es decir, existe una mayor resolución de bits. Entonces la pregunta que deja la afirmación anterior es ¿Cuál es mi resolución del prototipo? y ¿Cuál debe ser la resolución que debe recibir la tarjeta de sonido?

Para contestar las dos preguntas anteriores, se plantea el siguiente análisis:

Si la frecuencia de muestreo del dispositivo RTL de la señal entrante es de 2.5 MS/s y la frecuencia de salida de la tarjeta de sonido para la demodulación ACARS es 48KS/S, la relación de desmuestreo (Decimación) es aproximadamente 50. Es decir, para lograr bajar la frecuencia de muestreo a 48KS/s debe quitar 50 muestras periódicamente hasta llegar a las 2 500 000 muestras de la señal entrante en un segundo.

$$I_r = \frac{f_{RTL}}{f_{Tsonido}} \qquad (5.10)$$

$$I_r \cong 50$$

Pero, entonces cual debe ser el factor ideal para que pueda la tarjeta de sonido operar de la manera más óptima. La teoría de articulo [2], define que un desmuestreo o submuestreo debe ocurrir cada 4 muestras (2), es decir, quitar cuatros muestras y mantener una muestra.

$$f_{RTL} = \quad 4^n * f_{Tsonido} \qquad (5.11)$$

Figura 5.22: Diagrama del flujo de desmuestreo de una tarjeta de sonido

En la ecuación define la relación de sobre-muestreo en términos de la resolución de bits. Entonces para determinar cuál es la resolución que entrega mi prototipo:

$$\frac{f_{RTL}}{f_{Tsonido}} = 4^n \tag{5.12}$$

Entonces la resolución de entrada (n) se adapta a la ecuación No 3. De manera que despejando n, queda de la siguiente manera:

$$\log\left(\frac{f_{RTL}}{f_{Tsonido}}\right) = n\log(4)$$

$$n = \log\left(\frac{\log 50}{\log 4}\right) \tag{5.13}$$

$$I_r \cong 3$$

El resultado anterior significa que dicho sobre-muestreo, implica un aumento de resolución de 3 bits. Si la tarjeta de sonido opera a partir de 8 bits entonces:

$$\begin{aligned} N &= 2^{8+n} \\ N &= 2^{8+3} \end{aligned} \tag{5.14}$$

La resolución que recibe el sistema son del formato de 11 bits y la tarjeta de sonido opera bajo 8 bits, entonces debe se debe ajustar el prototipo de tal manera que pueda bajar la resolución de bits aproximadamente 3 bits con la relación 4^n de la tarjeta de sonido.

En conclusión, cuando se quiere adaptar alguna aplicación con la tarjeta de sonido de un ordenador, debe al final del proceso de desmuestreo ajustar una escala de Decimación de 4 muestras.

A partir del análisis anterior, en la siguiente prueba se debe ajustar de acuerdo con el análisis anterior el proceso de desmuestreo. Con la finalidad de mejorar la demodulación en tiempo real de los mensajes ACARS.

Figura 5.23: Nuevo Prototipo de Recepción-Demodulación de señales ACARS en tiempo real a través del RTL-SDR

Teniendo en cuenta las características de un convertidor Análogo-Digital (ADC), la frecuencia de muestreo no es tan importante para las aplicaciones de un dispositivo RTL-SDR, pues solo basta con la frecuencia de muestreo de la tarjeta de sonido.

5.7. Optimización de Prototipo ACARS: Demodulación AM aplicando concepto de desmuestreo de tarjeta de sonido.

Teniendo en cuenta el análisis de la anterior prueba, se realiza una modificación al prototipo de la Figura 5.18, de tal manera que pudiera cumplir con los requerimientos de la tarjeta de sonido. El nuevo prototipo se evidencia en la Figura 5.24.

Se mantiene el bloque de recepción del dispositivo RTL, así como también el filtro. Se cambia la relación de desmuestreo y la aplicación de una demodulación AM.

5.7.1. Relación de Desmuestreo

Si utilizamos una frecuencia de muestre de 1.15MS/s, entonces la relación de desmuestreo al valor más aproximado es 24:

$$I_r = \frac{1{,}15MS/s}{48KS/s}$$
$$I_r \cong 24 \tag{5.15}$$

Pero como la tarjeta debe hacer un desmuestreo en factor 4, entonces anterior a ello debe existir un factor de Decimación tal que N x 4 =24. Dicho factor debe ser igual a 6.

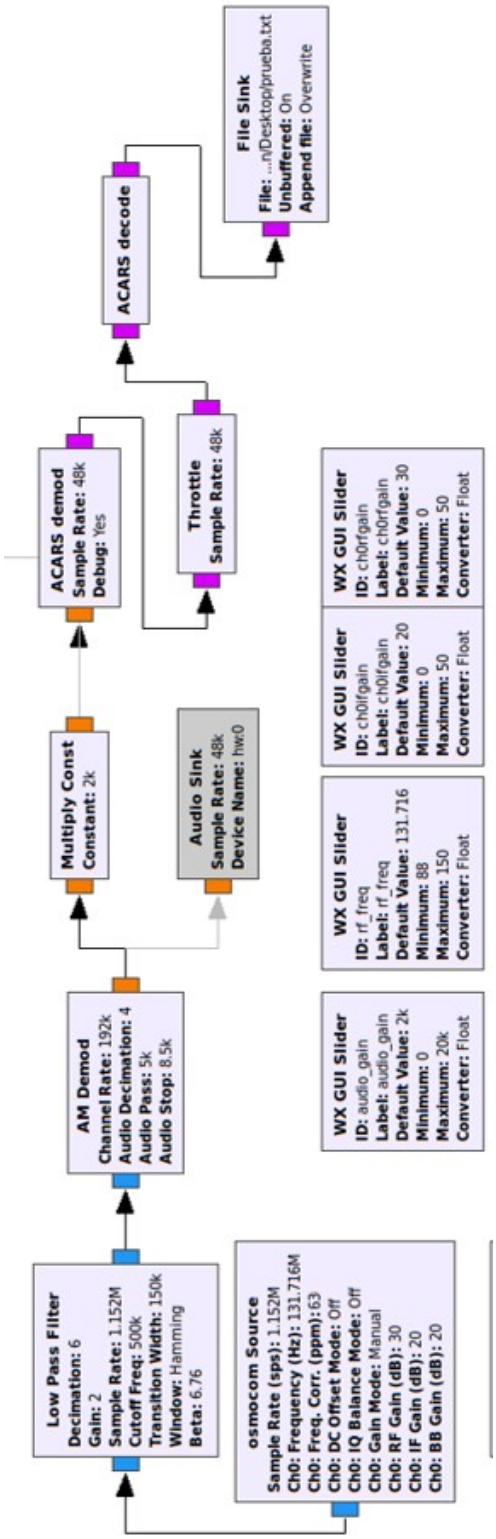

Figura 5.24: Nuevo Prototipo de Recepción-Demodulación de señales ACARS en tiempo real a través del RTL-SDR

Figura 5.25: Configuración del Filtro PasaAbajo en Gnu-Radio

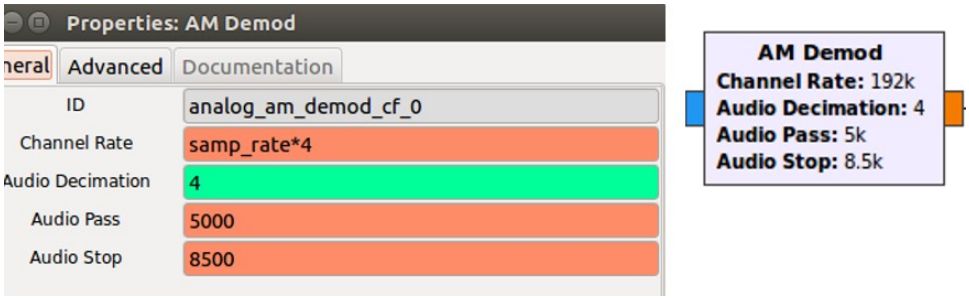

Figura 5.26: Configuración de una De-modulación AM en GNURadio

En GNURadio en el prototipo quien satisface la Primera Decimación con factor de 6 es el bloque de filtrado.Luego para satisfacer la segunda Decimación con factor 4 es el bloque de demodulación AM.

5.7.2. Demodulación AM

La finalidad de forzar una Decimación en una demodulación AM, es para que pueda des-muestrear la señal conforme la envolvente que produce una demodulación AM. La envolvente está controlada por la velocidad del canal de la tarjeta de sonido, así como la Decimación de audio. Es por ello que este bloque se adapta mejor para la finalidad de des-muestrear la señal en factor de 4 muestras a través de la tarjeta de sonido.

El análisis del resultado se obtiene que por primera vez logra Demodular un mensaje completo en tiempo real, pero con una probabilidad de reconocer mensajes ACARS. Es decir, tienen que pasar muchos mensajes para que el prototipo pudiera Demodular completamente el mensaje.

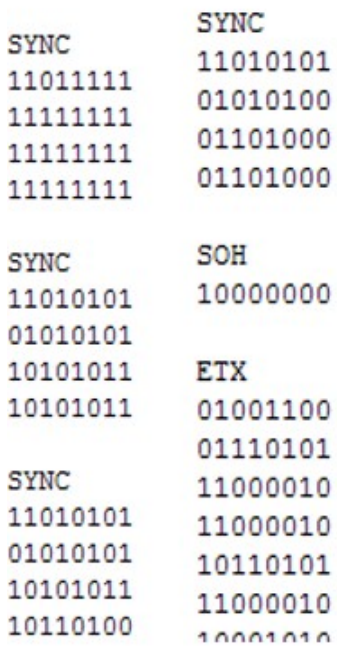

Figura 5.27: Resultado de la Demodulacion del prototipo optimizado

Capítulo 6

II Fase: *Decodificación*

En este capitulo desarrolla la teoría pertinente de la estructura de una trama ACARS, dependiendo si es un mensaje Downlink o mensaje Uplink. Para ello se identifica la estructura de un Byte (Byte entregado por el programa de GnuRadio) y conocer la tabla de traducción ASCII para pasar de un Byte a un dato tipo CHAR. Cumpliendo con las dos características anteriores se evidencia la decodificación ACARS a través del programa de GnuRadio en desarrollo de la Fase I. Como última prueba se realiza una comparación de decodificar un mismo mensaje ACARS bajo el programa pago PlanePloter y el programa de GnuRadio.

6.1. Trama ACARS

- **BIT "Sync":** Son dos bytes "+" y " *" la cual habilita transmisión del Bloque ACARS donde está contenido el mensaje.
- **SYN:** Son dos Bytes que ejercen la sincronización del bloque ACARS.
- **SOH:** Habilita el comienzo del bloque ACARS.
- **Modo:** Registra la información acerca de la información de la estación base. Si la cobertura es de tipo B, el indicador será 2.
- **Dirección:** Define el ID del aeronave o MU.
- **ACK-NACK:** Parámetro para notificar el estado del anterior mensaje. ACK notifica una recepción positiva y NACK notifica una retransmisión del mensaje anterior ante una notificación negativa.
- **Label:** Indicador que define el tipo de mensaje.
- **Block ID:** Es identificador numérico secuencial que sirve como parámetro para duplicar un mensaje ante un NACK o cuando no se responde una petición. De manera que cuando se duplica el mensaje se deja el mismo número secuencial.
- **STX:** Indicador de comienzo del mensaje.
- **Mensaje:** Está contenida la información nativa del MU. Máximo puede contener 220 Bytes.
- **ETX-ETB:** Fin del mensaje.
- **BCS:** Fin de transmisión del bloque ACARS.

Cabeceras	BIT "Sync"	SYN	SOH	Mode	Address	ACK NACK	Label	Block ID	STX	Mensaje	ETX-ETB	CRC	BCS
Bytes	2	2	1	1	7	1	1	1	1	220	1	2	1

Figura 6.1: Estructura de un mensaje ACARS.

	MSB	000	001	010	011	100	101	110	111
	LSB	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**
0000	**0**	NULL	DLE	SP	0 (Zero)	@	P	`	p
0001	**1**	SOH	DC1	!	1	A	Q	a	q
0010	**2**	STX	DC2	"	2	B	R	b	r
0011	**3**	ETX	DC3	#	3	C	S	c	s
0100	**4**	EOT	DC4	$	4	D	T	d	t
0101	**5**	ENQ	NAK	%	5	E	U	e	u
0110	**6**	ACK	SYN	&	6	F	V	f	v
0111	**7**	BEL	ETB	'	7	G	W	g	w
1000	**8**	BSC	CAN	(	8	H	X	h	x
1001	**9**	<TAB>	EM	)	9	I	Y	i	y
1010	**10**	LF	SUB	*	:	J	Z	j	z
1011	**11**	VT	ESC	+	;	K	[	k	{
1100	**12**	FF	FS	,	<	L	\	l	|
1101	**13**	CR	GS	-	=	M	]	m	}
1110	**14**	SO	RS	.	>	N	^	n	~
1111	**15**	SI	US	/	?	O	_	o	DEL

Figura 6.2: Tabla ASCII para el protocolo ACARS

La estructura del bloque ACARS está definido por una tabla de comparación de Bits Vs Bytes (Ver Figura 6.2). De manera que la traducción de los bits ACARS a datos CHAR (Byte) se realiza a través de la tabla presentada.

6.2. Bloque de decodificación ACARS en GNU-Radio

Por cada byte, hay 7 bits de información donde el LSB y MSB están invertidas. También existe un bit de paridad. (Ver Imagen 6.3). En la tabla de la figura 6.2, la columna corresponde a los Bits menos significativo y las filas referente a los bits más significativos.

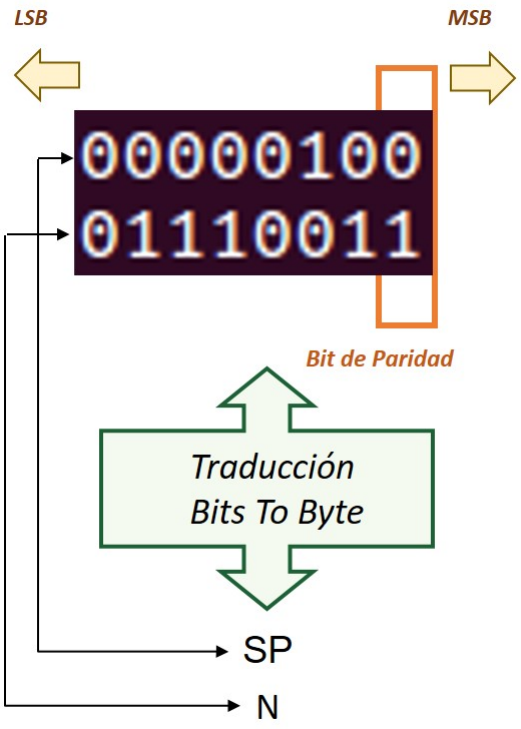

Figura 6.3: El primer Byte de la Figura donde sus primero 4 bits menos significativos son *(0000)* y los 4 bits mas significativos son *(010), al cruzar fila y columna de la figura 6.2* el resultado es ***SP***. ***El segundo Byte****, sus bits 4 bits menos significativos (0111) y sus 4 bits mas significativos (100), al cruzar fila y columna de la figura 6.2* el resultado es ***N***.

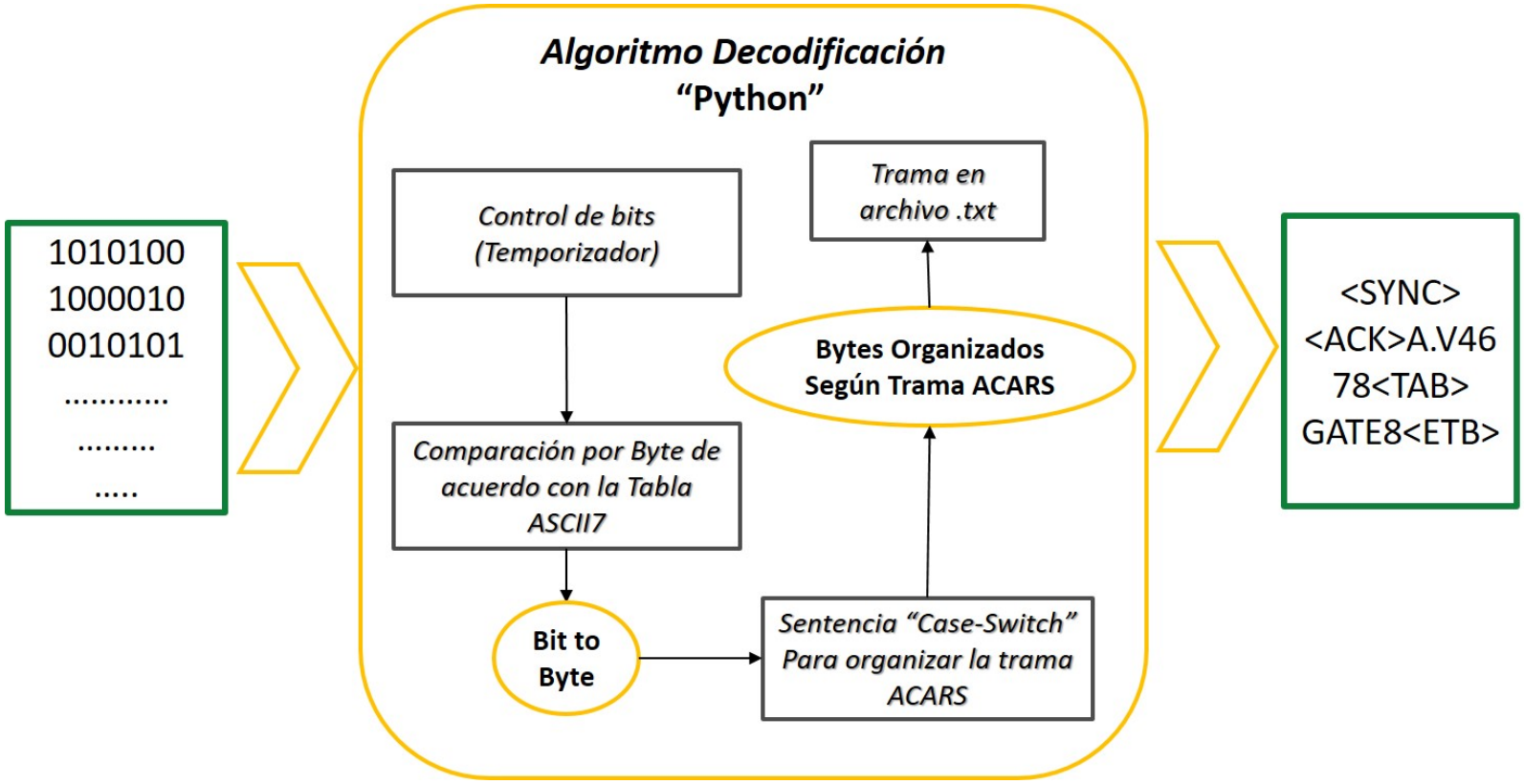

Figura 6.4: Los bits entrantes, son los Bytes entregados por el bloque de GNU-Radio de la demodulación ACARS. Para cada Byte entrante tiene que identificar los bits MSB y los bits LSB para convertir dicho Byte en una dato CHAR a través de la tabla ASCII. En programación *Python* realiza un*Case:Switch* para establecer un orden entrada coherente a la trama ACARS de la figura 6.1. Este por ultimo sobrescribe el resultado *(Mensaje ACARS)* en un texto plano

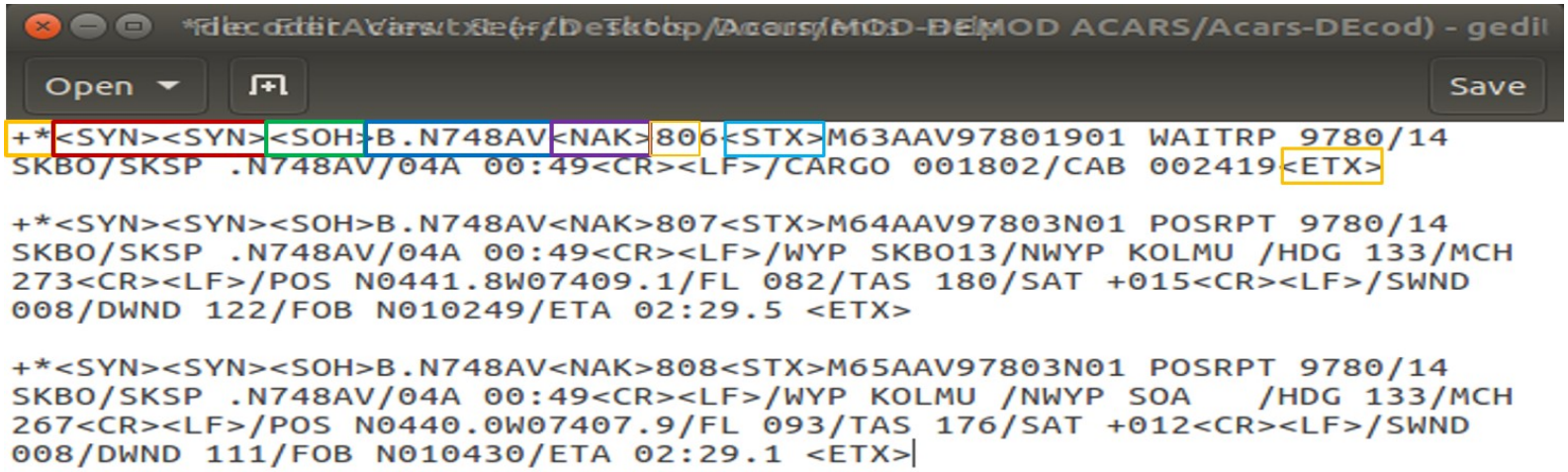

Figura 6.5: De acuerdo con la teoría de la composición de la trama de un mensaje ACARS, se puede verificar en primera instancia los bits de SYN (* +), y los demás bloques explicados en la teoría preliminar de este capitulo.

Capítulo 7

III Fase: *Desarrollo interfaz gráfica ACARS*

El capitulo *Interfaz Gráfica* se desarrolla un aplicativo capaz de visualizar los mensajes en una mapa de Google Maps. Para llevar a cabo lo anterior, empezamos diseñando un aplicativo Web basado en HTML5/CSS3. En el aplicativo web se anida el API de google Maps a través de JavaScript. Posteriormente, conviene toda la programación de la API de Google Maps en JavaScript para plasmar la trayectoria de la aeronave y el historial actual de la aeronave. Luego se realiza el procedimiento para diseñar y desarrollar la base de datos. Después programar la conexión con la base de datos y hacer las respectivas consultas con PHP y JavaScript obteniendo los mensajes estructurados en archivos con XML. Habiendo desarrollado la base de datos y el aplicativo Web solo falta adaptar la conexión entre GNU-Radio-Python-SQL(Base de datos).

7.1. Análisis y Planteamiento

Las API incluyentes en la anterior imagen hacen parte de la librería de **servicios web de google maps API**. Cada API realiza una función específica:

- **Directions API:** Es un servicio que permite calcular la ruta más óptima de un mapa estático (No en tiempo real. El resultado se resuelve a través de una petición http.
- **Distance Matrix API:** Es un servicio que permite calcular el tiempo y la distancia dada una ruta.
- **Elevation API:** Es un servicio que permite determinar una elevación promedio dada una ubicación geográfica.
- **Geocoding API:** Es un servicio de conversión entre una dirección (Calle. Carrera Avenida) y una ubicación Geográfica. Dicho de otra manera, como una caja negra, si la entrada es una dirección obtienes una localización geográfica, y viceversa, si la entrada es una localización geográfica la salida es una dirección.
- **Time Zone API:** Es un servicio que proporciona la diferencia horaria entre dos ubicaciones geográficas de la tierra. Este tipo de solicitudes son HTTPS.
- **Roads API:** Es un servicio que permite la mejor aproximación geométrica de una ruta y el límite de velocidad de un vehículo a través de la geolocalización GPS.
- **Places API:** Es un servicio que proporciona información de lugares como negocios, restaurantes etc., a través de la base de datos de Google.

Todos los API-Google mencionados son servicios web, cuya utilidad es el intercambio de información basado en JSON. Es decir, hace una petición http a través de una URL y una llave google maps, y este a su vez devuelve una respuesta en texto plano con datos organizados y estructurados. De modo que no existe API Google-Python capaz de desplegar una interfaz gráfica tal como un mapa para diseñar la trayectoria de una aeronave.

En conclusión, no satisface las necesidades y requerimientos del proyecto. GoogleMaps-Python es un servicio web, es decir solo entrega una respuesta de una solicitud WEB. Entrega un ACK de un servicio API.

Teniendo en cuenta conclusión anterior, el planteamiento resulta en utilizar un mapa dinámico basado en WEB a través de API-JAVASCRIPT Y ANDROID MAP, soportado por una base de datos (BD). Dicha DB están almacenados toda la información de los mensajes ACARS capturados por el programa receptor (RTL). La inserción de los datos de los mensajes a la DB se ejecuta con una API Python-Mysql. La anterior estructura se resume en la figura 7.1

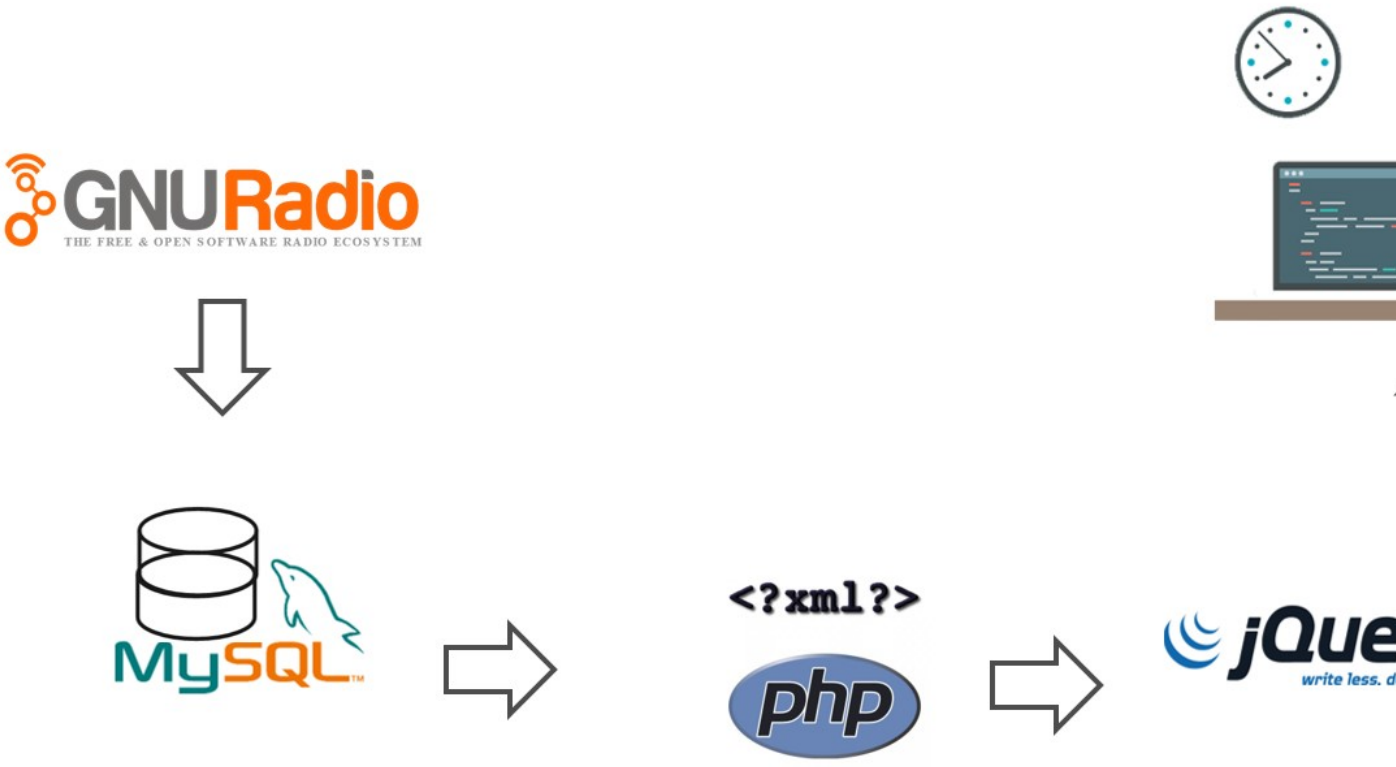

Figura 7.1: Diagrama de la estructura de la interfaz gráfica del usuario, con las tecnologías a emplear

7.1.1. API MySQL-Python

Dicho aplicativo cubre la funcionalidad de insertar los datos en la base de datos de una manera eficiente. Dicha eficiencia se mide con la disponibilidad de la información en el flujo de datos entre el programa receptor y la base de datos. Se debe cumplir el requisito de ¿Cómo adaptar la conexión entre GNU-Radio (python y C) y una base de datos en MySQL?

7.1.2. Base de Datos (MySQL)

El diseño de la base de datos es el requerimiento más importante de este planteamiento, pues es el soporte para que la interfaz pueda funcionar. El diseño de la base de datos está sujeto a los requerimientos de la interfaz gráfica y los tipos de mensajes ACARS Downlink y Uplink que existen.

La funcionalidad más importante de la interfaz gráfica es trazar la trayectoria o conocer la ubicación de una aeronave, y la representación histórica de una aeronave (Estado en tiempo real de la aeronave). La representación histórica se lleva a cabo a través de la clasificación de los tipos de mensajes ACARS. Para ello se debe hacer un trabajo de investigación de, cuáles son los tipos de mensajes existentes y desarrollar el aplicativo SQL para organizar dicha información.

En base de los requerimientos del párrafo anterior se deben diseñar las tablas relacionales de la base de datos de tal manera que las consultas sean sencillas y los recursos de programación sean mínimas.

7.1.3. JAVASCRIPT- Google Maps

En esta sección del planteamiento se desarrolla todo el soporte Web para poder contener el aplicativo de Google Maps en JavaScript. Para ello primeros se debe configurar un servidor WEB (Apache 2), luego el soporte HTML para el diseño sencillo de un página WEB, después la conexión con la base de datos y hacer las respectivas consultas (PHP y JavaScript) para procesar la información. Posteriormente, conviene toda la programación de la API de Google Maps en JavaScript para plasmar la trayectoria de la aeronave y el historial actual de la aeronave.

7.2. Metodología

La Metodología se desarrolla desde la interfaz gráfica hacia el programa Python-MySQL. Los pasos a seguir son los siguientes:

- Diseñar página Web (HTML5-CSS3) y API Google Maps.

- Desarrollar el aplicativo WEB para la conexión a la base de datos con PHP-AJAX. (La base de datos simula mensajes de posición de la aeronave para trazar la trayectoria).
- Programar Aplicativo para insertar los datos de manera ordenada a la base de datos a través de C-Python(GNURadio)-MySQL.
- Optimización de la conexión del DB y el aplicativo web para trazar varias trayectorias en tiempo real.

7.3. Diseño de página Web con (HTML5-CSS3) y API Google Maps

7.3.1. Página Web

El diseño de la página se divide en dos componentes; la página de inicio y el archivo de JavaScript la cual soporta el API de google Maps. Inicialmente, se tiene como criterio de diseño, que la página web presente a manera de introducción de qué se trata la aplicación Web y posteriormente el mapa en otra página. El resultado de aplicación web se muestra en la figura 7.2.

La página ofrece dos servicios. El primer servicio soporta la trayectoria de la aeronave en tiempo real sobre un mapa. El segundo servicio presenta el historial de las actividades de los aviones que haya recibido el RTL-SDR. Como primera instancia solo se desarrolla el primer servicio como el principal aplicativo del proyecto.

7.3.2. API Google Maps

En esta sección se debe logra programar el archivo JavaScript donde esta contenido el mapa y lograr anexar dicho archivo a la página Web. En primer lugar, definimos la configuración básica del API Google maps. Luego aumentamos la complejidad del código para conocer la ubicación del dispositivo del usuario (Celular o computador) a través de GPS o red Móvil/WIFI. Por último, agregamos el código necesario para trazar trayectorias sobre el mapa.

7.3.2.1. Creación de un Mapa con *Google Maps*

La función **initMap** se ejecuta implícitamente por el elemento **"map"** en la etiqueta **¡div¿**del cuerpo HTML. Dicha función es utilizada para solicitar autorización a GOOGLE MAPS para ejecutar el contenido del mapa a través de KEY. dicho KEY es administrado por Google API MANAGER.

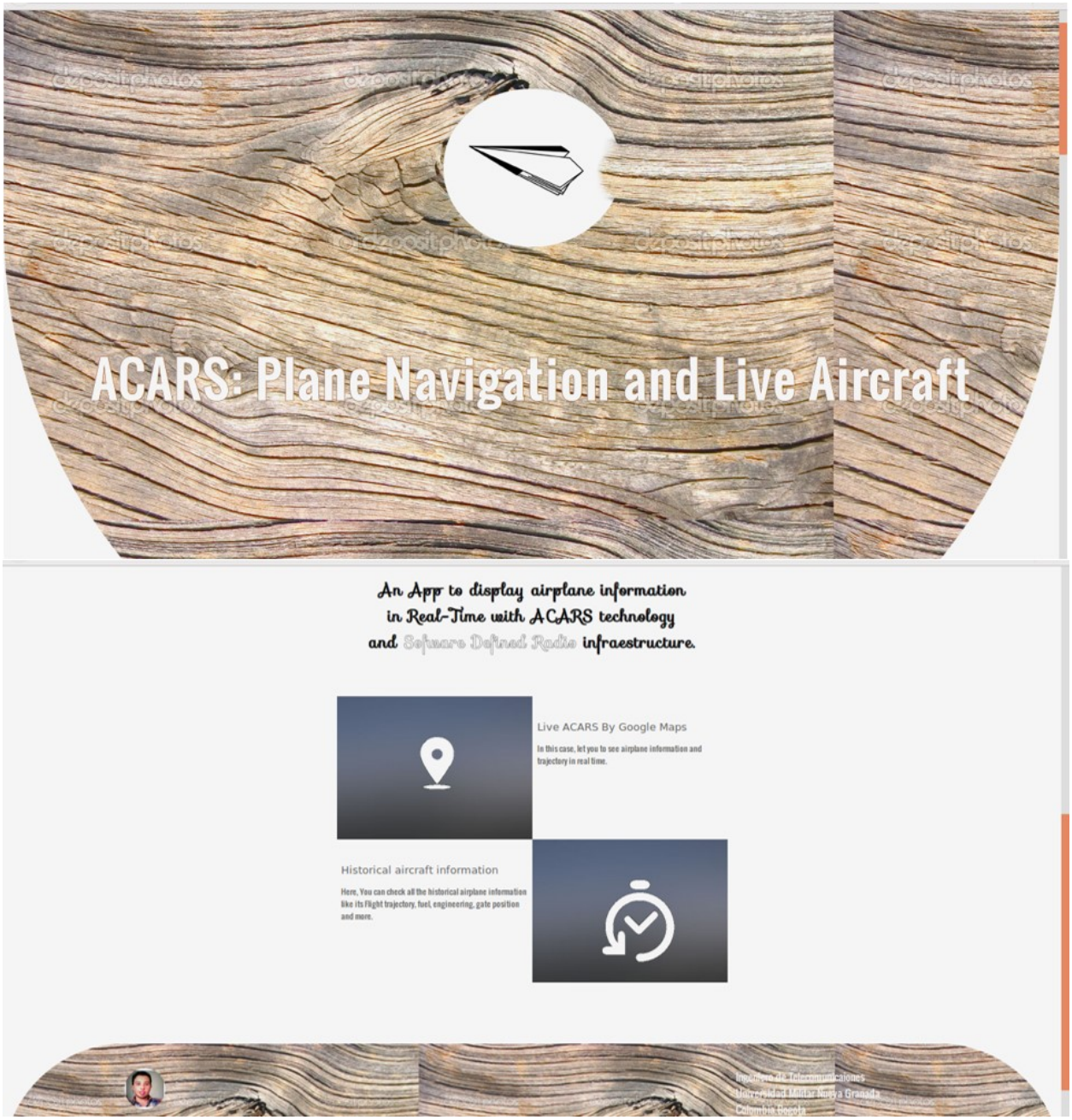

Figura 7.2: Aplicación web, basado en HTML5 y CSS3

Para generar un mapa es necesario crear una instancia a través de "new google.maps.**Map**(PARAMETROS)". *A cuantos Google maps se ejecuten así mismo son el número de instancias que se deben realizar.*

```
function inMainMap() {
  var centerLoad= {lat: 4.697558, lng: -74.098894 };
```

La siguiente función específica donde centrara el map la cargarse en el archivo HTML. El centro se especifica con la Latitud y longitud en decimales. y el zoom está la escala desde 0 (Nivel tierra)

```
var Mainmap = new google.maps.Map(document.getElementById('map'), {
    zoom: 10,
    center:centerLoad
});
```

La instancia **Marker** permite agregar marcadores al mapa. La ubicación está controlada por la variable **centerLoad** (*Dicha ubicación corresponde a la ciudad de Bogotá.*)

```
var marker= new google.maps.Marker({
  position:centerLoad,
  map:Mainmap
});
```

El resultado del código completo se refleja en la figura 7.3.

7.3.2.2. Activación de la ubicación de GPS del usuario

El método navigator.geolocation.**getCurrentPosition**(OBJECT 1, OBJECT 2) entrega dos atributos en forma de objetos. Cada objeto es una instancia que contiene información de la solicitud. OBJECT1 obtiene información respecto a la ubicación del dispositivo como altura, velocidad. OBJECT2 contiene la información de errores del navegador al rechazar la geolocalización. La instancia **InfoWindow** es la clase de la ventana de información para desplegar datos de algún evento.

```
var infoWindow = new google.maps.InfoWindow({map: map});
```

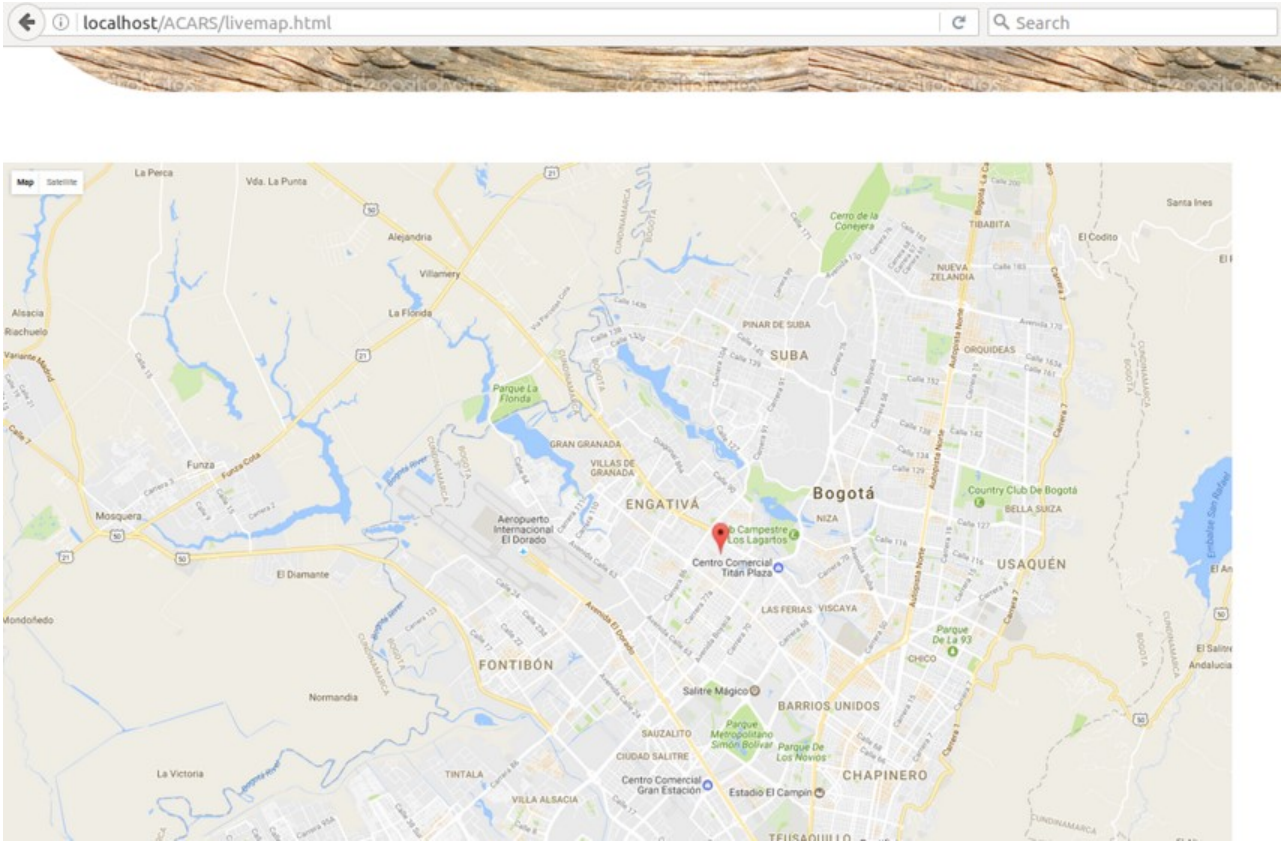

Figura 7.3: Creación de un Mapa centrado en la ciudad de Bogotá con un *Marker* de *Google Maps*

Solicita permiso de localización al navegador a través de **navigator.geolocation**.

```
1  if (navigator.geolocation) {
2  navigator.geolocation.getCurrentPosition(
3  function(position) {
4  var pos = {
5  lat: position.coords.latitude,
6  lng: position.coords.longitude
7  };
8  infoWindow.setPosition(pos);
9  infoWindow.setContent('Location found.');
10 map.setCenter(pos);
11 },
12 function() {
13 handleLocationError(true, infoWindow, map.getCenter());
14 });
15 }
16 else{handleLocationError(false, infoWindow, map.getCenter());}
17 function handleLocationError(browserHasGeolocation, infoWindow, pos){
18 infoWindow.setPosition(pos);
19 infoWindow.setContent(browserHasGeolocation ?
20 'Error: The Geolocation service failed.' :
21 'Error: Your browser doesn\'t support geolocation.');
22 }
```

El resultado del código anterior se refleja en la figura 7.4.

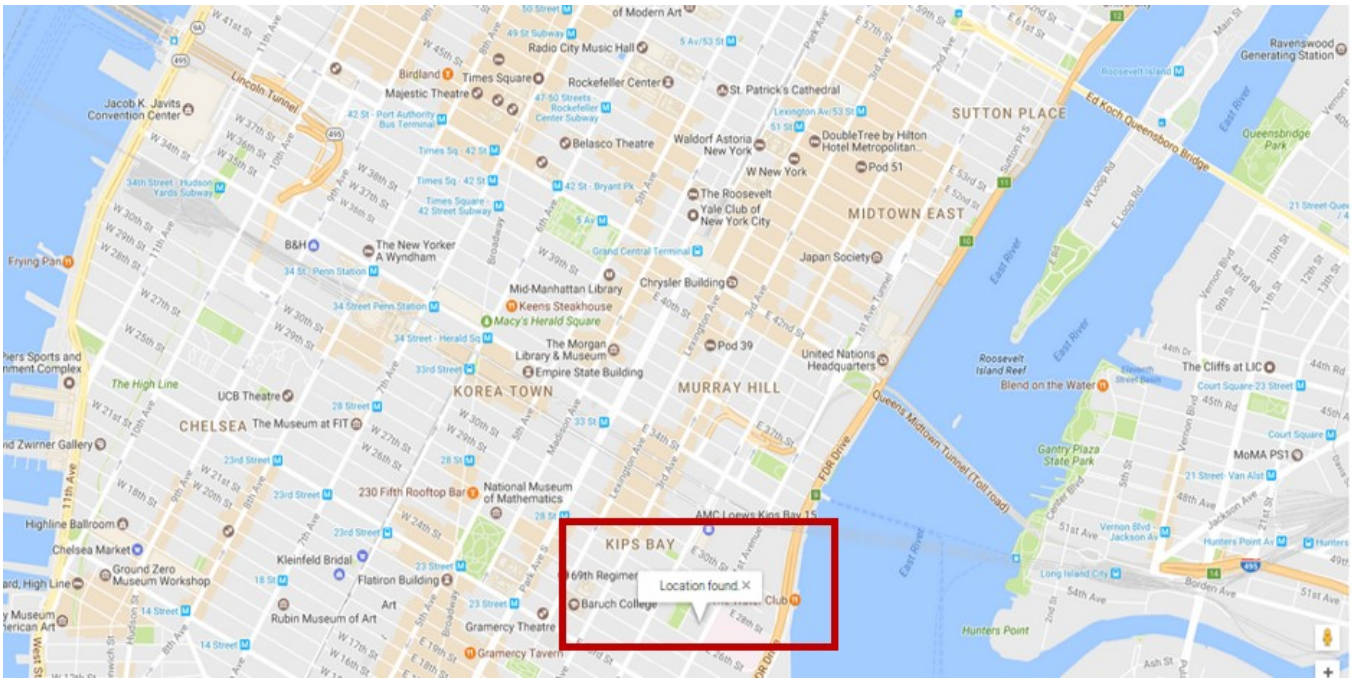

Figura 7.4: La ubicación real en la ciudad de New York cuando se ejecutó el código en el lado del cliente.

7.3.2.3. Programación de la trayectoria en el mapa

Ubicaciones de la primera trayectoria:

```
var trajec=[
  {lat:4.633911, lng:-74.176484},
  {lat:4.649833, lng:-74.155068},
  {lat:4.662140, lng:-74.136945},
  {lat:4.691128, lng:-74.081322},
  {lat:4.745104, lng:-74.031218},
  {lat:4.830095, lng:-73.965087},
  {lat:4.907013, lng:-74.016900}
];
```

Ubicaciones de la segunda trayectoria:

```
var trajec2=[
  {lat:4.608863, lng:-74.022617},
  {lat:4.643611, lng:-74.143302},
  {lat:4.695503, lng:-74.196541},
  {lat:4.777298, lng:-74.267555}
]
```

Luego creamos las instancias Polyline para crear las líneas en base de las ubicaciones.

```
var flightTraject= new google.maps.Polyline({
  path: trajec,
  geodesic: true,
  strokeColor: '#FF0000',
  strokeOpacity: 1.0,
  strokeWeight: 2
});

var flightTraject2= new google.maps.Polyline({
  path: trajec2,
  geodesic: true,
  strokeColor: '#000000',
```

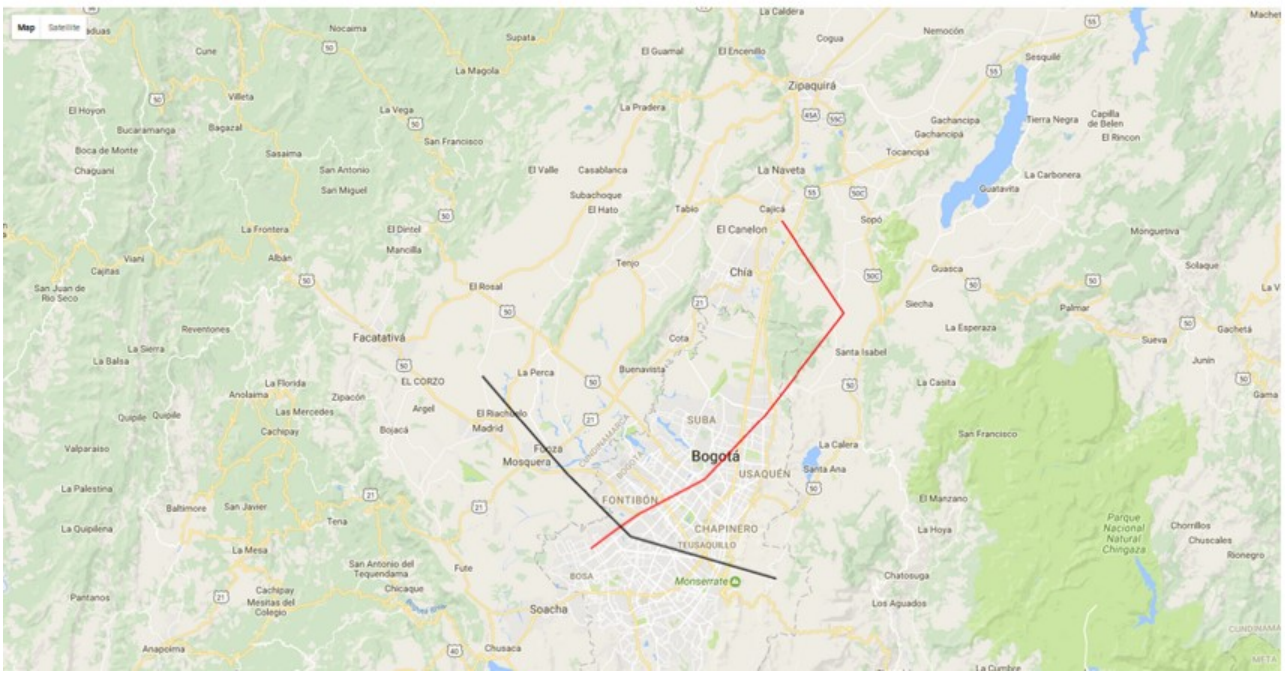

Figura 7.5: La ubicación real en la ciudad de New York cuando se ejecutó el código en el lado del cliente.

```
13    strokeOpacity: 1.0,
14    strokeWeight: 2
15  });
```

Luego de crear y configurar los atributos de los objetos creados a partir de las instancias anteriores lo implementamos en el objeto map, donde se encuentra el mapa a través de la función **setMap**.

```
1     flightTraject.setMap(map);
2     flightTraject2.setMap(map);
```

El resultado del código anterior se evidencia en la Figura 7.5

En la siguiente sección, se realiza el codigo para obtener dichas ubicaciones desde una base de datos MySQL y generar un script .sh donde periódicamente este insertando datos a la DB (Simula el papel funcional de GNURadio). El resultado de esta prueba es que se simula en tiempo real el movimiento de la aeronave en google Maps.

7.4. Desarrollo del aplicativo WEB para la conexión a la base de datos con PHP-AJAX.

En esta fase del proyecto ya existe una aplicación Web capaz de mostrar de una manera amable al usuario el contexto del proyecto. También existe el archivo .js capaz de manipular un mapa con GPS y trazar trayectorias sobre la misma.

El objetivo de esta fase es hacer peticiones periódicas al servidor a través de AJAX por medio de JQUERY. La finalidad es obtener los datos desde la DB de las ubicaciones para poder trazar una trayectoria. Dichas peticiones no son más que consultas que se realizan a través de PHP.

La ventaja de usar AJAX sobre JavaScript es obtener los datos del servidor sin necesidad de recargar toda la página, de manera que solo se actualiza el mapa. Dicha ventaja permite que sea más rápido el programa y menos carga de bajada para el servidor. Hay que tener en cuenta que AJAX es una extensión de JAVASCRIPT, es decir hereda la sintaxis de JavaScript, pero con funcionalidades potentes para aplicaciones en tiempo real.

En primer lugar, se debe instalar y configurar un servidor Web (APACHE 2) para alojar los archivos de la página web, archivos de JavaScript y PHP. La ruta para alojar dichos archivos en el servidor en Ubuntu se encuentra en la ruta *var*/*www*/*html*/*TU_CARPETA*. Inicialmente la base de datos se diseña e implementa una tabla donde contiene el Número de vuelo y la posición Geográfica (Latitud y Longitud) a través de un archivo sql.

– COMIENZO DEL FICHERO .sql –

```
1 DROP database ACARS;
2 CREATE database ACARS;
3 USE ACARS;
4 CREATE TABLE Vuelo
5 (
6 ID INT NOT NULL AUTO_INCREMENT,
7   Identificacion CHAR (9),
8   Latitud FLOAT(6,3),
9   Longitud FLOAT(6,3),
```

```
  PRIMARY KEY (ID)
) ENGINE=InnoDB;

INSERT INTO Vuelo VALUES (null,'AV965','4.633911','-74.176484');
INSERT INTO Vuelo VALUES (null,'AV965','4.649833','-74.155068');
INSERT INTO Vuelo VALUES (null,'AV965','4.662140','-74.136945');
INSERT INTO Vuelo VALUES (null,'AV965','4.691128','-74.081322');
INSERT INTO Vuelo VALUES (null,'AV965','4.745104','-74.031218');
INSERT INTO Vuelo VALUES (null,'AV965','4.830095','-73.965087');
INSERT INTO Vuelo VALUES (null,'AV965','4.907013','-74.016900');

SELECT * FROM Vuelo;
```

Ya existiendo una DB, los pasos a seguir hacer es la conexión con PHP desde la página web para realizar la consulta de la longitud y la latitud de la tabla VUELO. Luego de poder hacer consultas desde la página Web, se procede a realizar dichas peticiones desde el archivo JavaScript a través de AJAX. Y por último realizar el código necesario para que dichas solicitudes se hagan periódicamente de tal manera que se cargue solamente el mapa. Finalmente insertar datos a la DB para comprobar el funcionamiento de la trayectoria en tiempo real sin necesidad de recargar totalmente la página.

7.4.1. Consultas al BD desde la página Web con PHP.

La función del archivo PHP es realizar la conexión a la DB y extraer los datos a través de consultas. Dichos datos se organizan de una manera estructurada a través de un archivo XML. Desde la perspectiva de una caja negra, la entrada son peticiones SQL y la salida es un archivo XML donde los datos son la respuesta del servidor de la petición SQL.

Antes de comenzar se debe precisar que la para la versión 7 de PHP o superiores la clase MYSQL ya no existe. Para ello se debe utilizar MYSQLi. En PHP existen dos formas de programar, en forma procedimental y la programación orientado a objetos. En este proyecto el estilo de programación usado es la Orientado a objetos. Cabe resaltar que las dos formas de programación tienen la misma eficiencia de código. En primer lugar, se debe realizar una conexión a la DB.

```
<?php

//CONNECTION.PHP
```

```
4
5  // Carga la configuracion
6  $config = parse_ini_file('conf.ini');
7
8  // Conexion con los datos del 'config.ini'
9
10 $connect= new mysqli(
11   'localhost',
12   $config['username'],
13   $config['password'],
14   $config['dbname']);
15
16 if ($connect->connect_errno) {
17   echo "Fallo al conectar a MySQL: " . $connect->connect_error;
18 }
19 /*else {
20 echo 'Conectado a la base de datos';
21 }*/
22 ?>
```

El archivo conf.ini es un texto plano donde contiene un ARRAY. Dicho ARRAY está la información de la Base de Datos, tal como el usuario, la contraseña y el nombre de la DB. La función parse_ini_file convierte el ARRAY en una variable PHP. A partir de la clase MYSQLi se crea el objeto CONNECT a través de la instancia new mysqli. El condicional permite controlar la conexión según si fue exitosa o falló la conexión.

En segundo lugar, se realiza la consulta y organiza los datos en un archivo XML.

```
1
2
3  <?php
4  //LOCATION.PHP
5  //Archivo de conexion a la base de datos
6  require('connection.php');
7
8  $result= $connect->query("SELECT Identificacion,Latitud,Longitud FROM
9                            Vuelo ORDER BY ID ASC");
```

```
10 if(!$result){
11   echo "Lo sentimos, este sitio web esta experimentando problemas.";
12 }/*else {
13   echo 'Consulta correcta';
14 }*/
15
16 // Start XML file, create parent node
17 $dom = new DOMDocument("1.0");
18 $node = $dom->createElement("traject");
19 $parnode = $dom->appendChild($node);
20 header("Content-type: text/xml");
21
22 // Add to XML document node
23 while ($row = $result->fetch_assoc()){
24   $node = $dom->createElement("traject");
25   $newnode = $parnode->appendChild($node);
26   $newnode->setAttribute("Identificacion", $row['Identificacion']);
27   $newnode->setAttribute("Latitud", $row['Latitud']);
28   $newnode->setAttribute("Longitud", $row['Longitud']);
29 }
30 echo $dom->saveXML();
31 ?>
```

La función require('connection.php') carga el archivo connection.php, realizado en el código anterior. La variable $connect− >query es un llamado de la función query al objeto connect declarado en el archivo connect.php. Los parámetros de la función query es un petición SQL. En este caso se realizar una consulta a la tabla Vuelo de la base de datos ACARS creados anteriormente. La petición SQL es: ("SELECT **Identificacion,Latitud,Longitud** FROM **Vuelo** ORDER BY **ID ASC**") cuyo resultado lo entrega de manera ascendente según el ID. (ID es un campo de la tabla Vuelo de la Base de Datos ACARS).

Con la sentencia $dom =new DOMDocument("1.0"); creamos el archivo XML. La idea es anexar el resultado de la base de datos en archivo XML. Un archivo XML está compuesto de nodos, este a su vez hereda hijos. Para diferenciar un nodo de otro nodo es a través de sus atributos. En este ejemplo solo usaremos nodos con atributos.

A través de un While se crear un nodo diferente con su respectivo atributo cada fila de la

tabla consultada. Dicha funcionalidad se ejecuta con while ($row = $result− >fetch_assoc() . Primero se asigna un nombre a cada nodo con la sentencia $node = $dom− > createElement(”traject”);. Se crea una herencia del nodo creado con $newnode = $parnode − > append Child ($node);. Y a cada Sub-nodo se le asigna un valor en forma de un atributo con $newnode− >setAttribute(”Latitud”, $row ['Latitud']);. El primer parámetro es el nombre que se le asigna en el archivo XML y el según parámetro es el nombre del campo en la tabla consultada en la base de datos.

Por ultimo exportamos el archivo XML con la sentencia echo $dom-¿saveXML(); . Para acceder al resultado del código anterior se debe indicar la ubicación del archivo Location.php según el servidor que se esté usando.

7.4.2. Peticiones al servidor con programación AJAX.

Manteniendo el mismo código JavaScript de los ejemplos anteriores, cuyas funciones han sido crear un Mapa de Google en archivo HTML a través de una llave de GOOGLE API. Dibujar marcadores y trayectorias sobre el mapa creado. Y por el ultimo la ubicación por GPS del usuario. El código a implementar está basado en JQUERY, es decir, se escribe AJAX con Jquery. En resumidas palabras JQUERY es una forma de escribir JavaScript de una manera más sencilla y eficiente. El código adicional es el siguiente:

```
1  var filename = 'xxxxxxxx.php';
2  function request(){
3    $.ajax({
4      type: "GET",
5      cache: false,
6      url: filename ,
7      dataType: "xml",
8      success: parseXML,
9      error : onXMLLoadFailed
10   });
11 }
```

La variable var filename = 'http://10.0.1.52/ACARS/include/location.php' ;es la ubicación del archivo XML, creado a partir de PHP. La IP es la ubicación del servidor. Para evitar problemas es recomendable escribir la dirección IP del servidor en vez de LOCALHOST. El inicio de *JQUERY* se denota por la variable peso ($) $.ajax() . El primer parámetro type: es el tipo de petición. *”GET”* es para obtener un resultado y *”POST”* para insertar algún valor. El

segundo parámetro `cache:` es para indicarle que no guarde información adicional del servidor, para evitar sobrecarga en el servidor. El tercer parámetro indicamos la ubicación del archivo en el servidor. El cuarto parámetro a pesar de que es un archivo PHP el resultado es un documento XML, se debe especificar el resultado de dicha petición. Los dos parámetros siguientes son las acciones a seguir en caso en que la petición haya sido correcta `(success: parseXML)`, caso contrario cuando haya ocurrido algún error `(error : onXMLLoadFailed)`. Vale la pena aclarar que los dos anteriores parámetros son funciones.

En caso que haya ocurrido algún error se ejecuta la función onXMLLoadFailed(), solo muestra un error. Tal como está escrito:

```
1 function onXMLLoadFailed(){alert("Servicio no disponible. Contacte
2                           con el Administrador para mas informacion")}
```

Dado el caso que la petición haya sido exitosa, *AJAX* ejecuta la función `parseXML()`. Esta función se debe leer el archivo XML y extraer dichos valores. Después, hacer las instancias respectivas para insertar los marcadores y la trayectoria con las ubicaciones geográficas obtenidas desde el *XML* mediante la base de datos. El código correspondiente es el siguiente:

```
1  function parseXML(xml){
2    var markerCoords=[];
3    var marker=[];
4
5    $(xml).find("traject").each(function(i){
6    var lat = $(this).attr('Latitud');
7    var lng = $(this).attr('Longitud');
8    markerCoords[i] = new google.maps.LatLng(parseFloat(lat),parseFloat(lng));
9      marker[i]= new google.maps.Marker({
10       map:map
11       });
12   marker[i].setPosition(markerCoords[i]);
13   });
14   var flightTraject= new google.maps.Polyline({
15     path: markerCoords,
16     geodesic: true,
17     strokeColor: '#FF0000',
```

```
18      strokeOpacity: 1.0,
19      strokeWeight: 2
20    });
21    flightTraject.setMap(map);
22    //alert(markerCoords);
23  }
```

La función parseXML(xml) es un parámetro que recibe como argumento el resultado de la petición *Ajax*, en este caso es un archivo *XML*. Es decir, la función recibe como variable o un objeto denominado *xml*. Para fines posteriores declaramos dos arrays. En este array var marker=[]; se almacena los objetos instanciados de cada marcador y var markerCoords=[]; recibe las ubicaciones geográficas para dibujar la trayectoria y los marcadores sobre el mapa.

Luego con la función **find** sobre el objeto *XML* busca el nombre de cada nodo *traject*, a través de un bucle mediante **each** $(xml).find("traject").each(function(i) . Con la función attr('Latitud'); extrae el valor del atributo para cada nodo en una variable var lat = $(this).attr('Latitud');. La variable markerCoords[i] guarda todas las ubicaciones para cada ciclo que encuentra un nodo en el archivo XML. Luego realiza una instancia para convertir las variables geográficas (de variables STRING a variables FLOAT con parseFloat) en objetos a través de new google.maps.LatLng.

Posterior, con marker[i].setPosition(markerCoords[i]) ; plasma los marcadores según las coordenadas en un mapa. Y flightTraject.setMap(map); dibuja la trayectoria de todas las ubicaciones almacenadas en el array flightTraject .

7.4.3. Optimización de código AJAX para trayectoria en tiempo real.

Ahora la cuestión es cada cuanto debe realizar las peticiones de consulta al servidor. Se pensaría en implementar un while controlado al código *AJAX* escrito. De tal manera que realiza el mismo código indefinidamente cada cierto tiempo. Sin embargo, en *JavaScript* existe un método de la clase *JQUERY* que permite ejecutar una misma función cada cierto tiempo. Teniendo en cuenta el método request() para la petición AJAX, el "Bucle" se define de la siguiente manera:

```
1  setInterval(request, 30000);
```

Figura 7.6: Como la función es cada 30 segundos, aún no ha enviado la petición al servidor (Archivo PHP).

Cuyo primer parámetro es una función, en este caso la función donde está contenido el código AJAX. El segundo parámetro es el tiempo reiterativo (en ms) que va ejecutar la función definido en el primer parámetro.

Al ejecutar el código completo, la secuencia se define de la siguiente manera:

- Antes de hacer la consulta (Antes de 30segundos). Ver Figura 7.6
- Despues de hacer la consulta (Despues de 30 Segundos) Ver Figura 7.7
- Para simular una trayectoria en tiempo real, simulamos dicho comportamiento agregando una nueva ubicación desde la Base de datos. Ver Figura 7.8

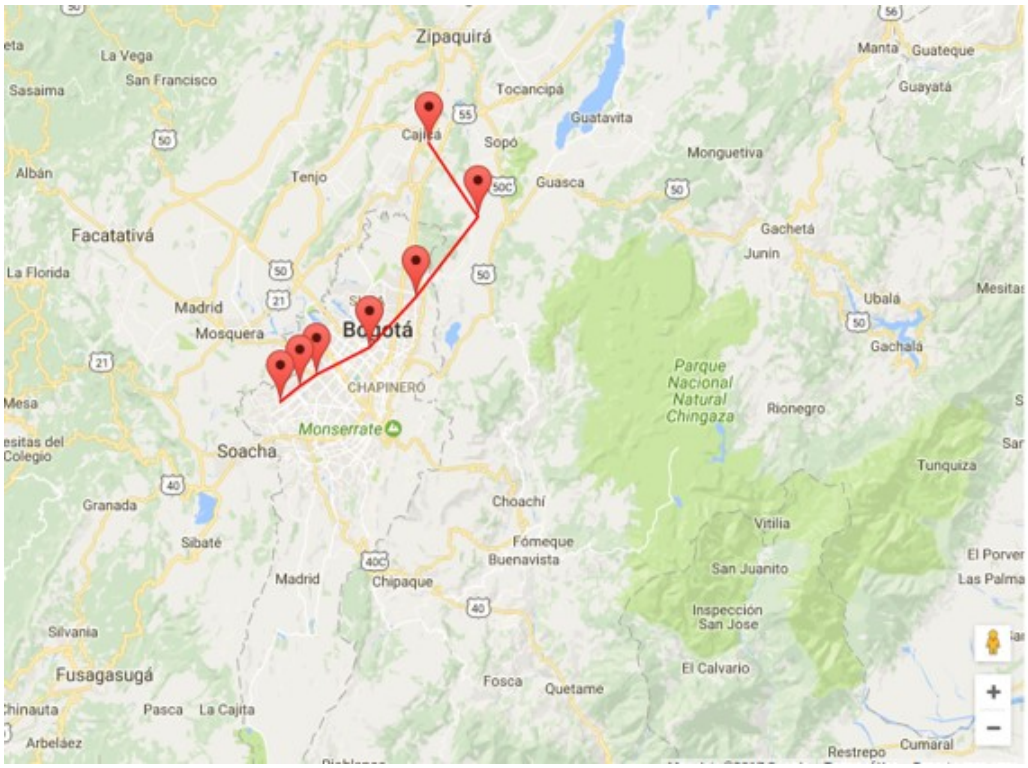

Figura 7.7: Pasado los 30 segundos obtiene la respuesta del servidor y lo plasma en el mapa. Hay que tener en cuenta que los marcadores son las ubicaciones geográficas y la trayectoria es la unión de los mismas. La idea de cada marcador es que al ocurrir el evento de un click sobre un marcador despliegue el mensaje ACARS.

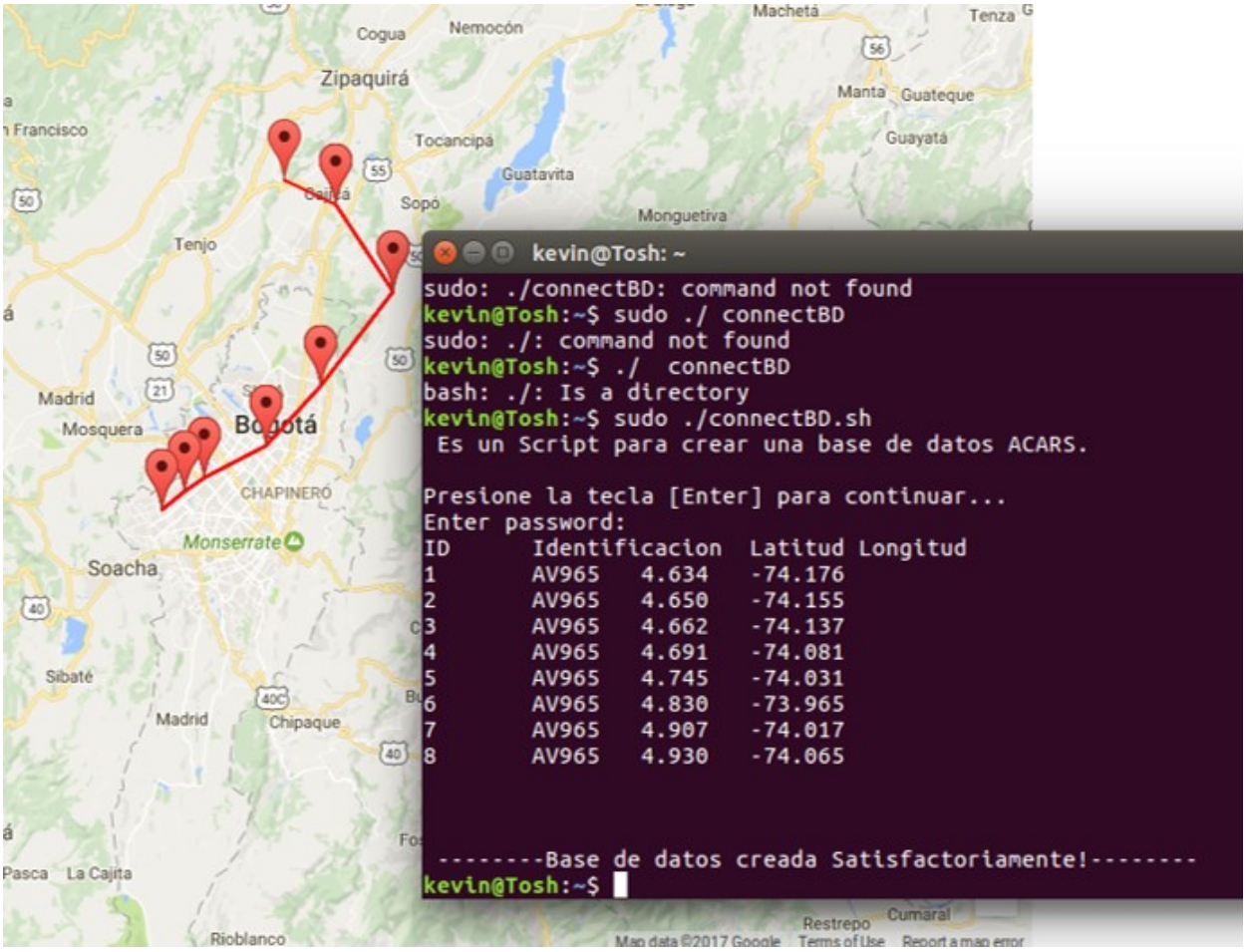

Figura 7.8: Al agregar un nuevo punto geográfico a la base de datos, el mapa, pasado los 30 segundos actualiza el nuevo marcador.

7.5. Programación de aplicativo para insertar los datos de manera ordenada a la base de datos a través de Python (GNURadio)-MySQL.

Teniendo el resultado de la demodulación y la decodificación en GNU-Radio, el paso a seguir es insertar los datos en una base de datos MySQL desde GNU-Radio. Desde un archivo Python actúa como un buffer para identificar un mensaje ACARS e inmediatamente insertar los datos a la base de datos. El Buffer esta implementado con un control de flujo de datos. Dicho control permite establecer la longitud del mensaje, ya que esta es variable según el contenido del paquete. Dicho archivo Python se ejecuta simultáneamente a través de un bloque de GNU-Radio denominado ***embedded de GNU-Radio.***

7.5.1. Trabajando con block Python embedded de GNU-Radio

El bloque Python embedded permite interactuar los datos provenientes de otro bloque de GNURadio, para realizar cualquier actividad escrita en Python y obtener una nueva salida de datos; en otras palabras, una caja blanca. La estructura del código por defecto de este bloque se refleja en la Figura 7.9

El diagrama de la Figura 7.9 resume la estructura del código que entrega por defecto GNURadio. La cuestión es cómo adaptar nuestro código en la estructura dada. Para empezar, si se necesita anexar un API adicional o una libraría externa, **Inicio** es el espacio ideal para declarar librerías, API, Archivos .PY o bloques de GNURadio.

Por la Clase GnuRadio puede interpretar el bloque dado. Existen varios tipos de bloques como SycBlock y BasicBlock. De acuerdo a la clase heredada por GnuRadio se pueden instanciar los parámetros y funciones contenidos dentro del método work() .

Los parámetros iniciales se refieren a los tipos de variables que se van a manipular. Dichas variables se deben declarar con la librería numpy. Para declarar el tipo de variable sea entrada o salida se define como in_sig= [np.complex64] , donde `complex64` es una variable compleja. Para identificar qué tipos de variables existen y cuál es el adecuado según la aplicación a programar, se debe consultar la referencia de la librería numpy.

En la función def work(self, input_items, output_items) es donde ocurre toda la magia. Acá definimos el tratamiento que se le va a dar a los datos entrantes. Por ejemplo , establecer el flujo de datos entre la salida con los datos entrantes output_items[0][:]=input_items[0]. Vale aclarar que la función work() se ejecuta cada vez que la variable es diferente de cero, es decir, es un Bucle controlado por los datos que existan en la entrada del bloque Python embedded.

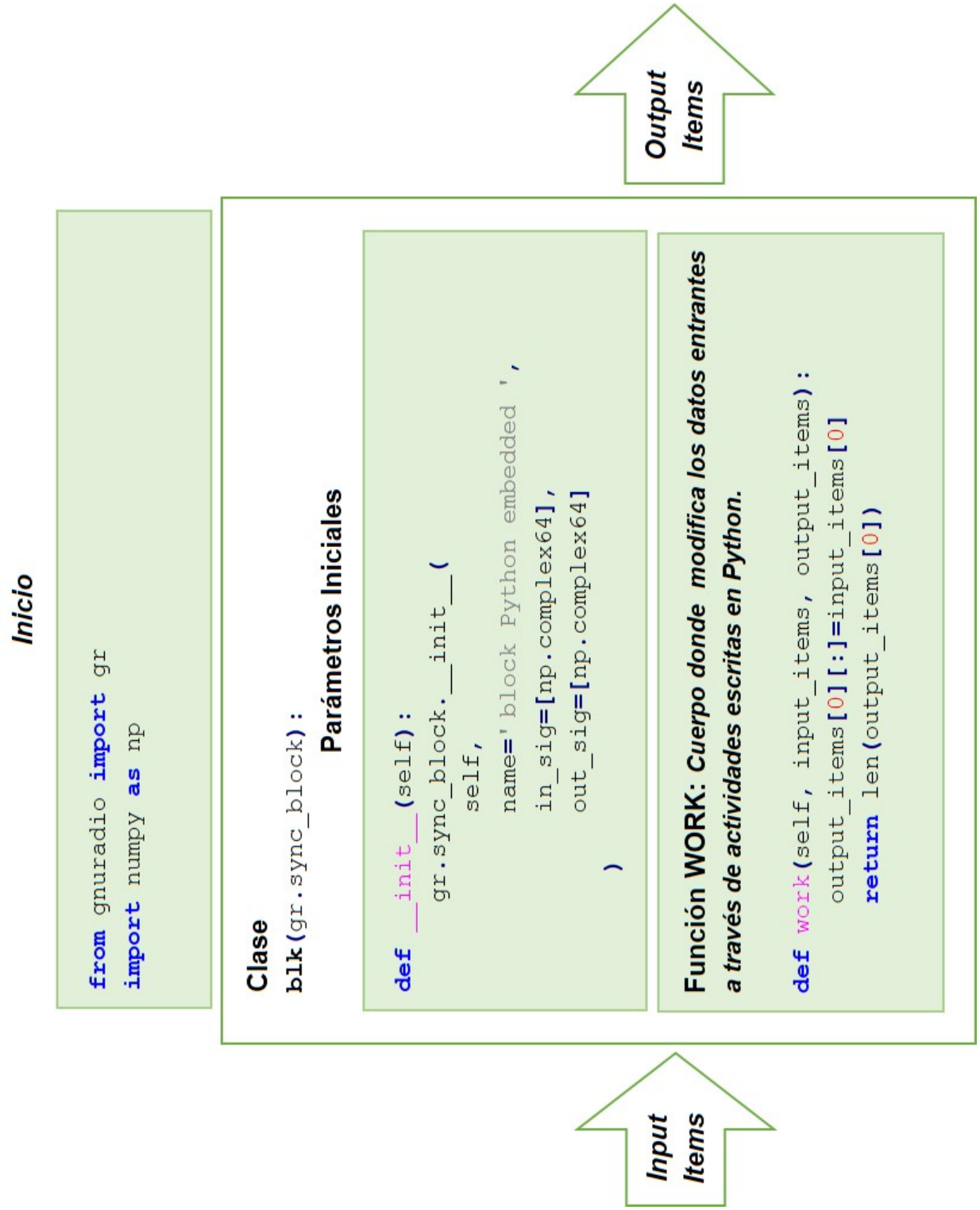

Figura 7.9: Diagrama por defecto del bloque ***embedded Python de GNU-Radio.***

7.5.2. Conexión a la DB con Python y MySQL desde GNU-Radio

En primer lugar, hay que descargar la librería MySQL para Python que se encuentra en la página oficial de MySQL. En la página oficial se encuentra la referencia de las clases, métodos y ejemplos para conectar a la base de datos desde Python. En este apartado realiza una guía de cómo adaptar la librería MySQL en GNURadio, guía que difícilmente se encuentra en algún documento.

Al principio de este documento ya habíamos creado nuestra base de datos ACARS con una tabla denominada Vuelo, cuyo propósito había servido para insertar ubicaciones geográficas para comprobar el funcionamiento del aplicativo de Google Maps. Ahora, se crea otra tabla en la base de datos para almacenar los mensajes ACARS provenientes de GNURadio. Dicha tabla se define como mensaje, el código SQL se define de la siguiente manera:

```
1  CREATE TABLE mensaje
2  (
3      IDm INT NOT NULL AUTO_INCREMENT,
4      fecha TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
5      iniM VARCHAR(3),
6      modo VARCHAR(2),
7      rgo VARCHAR(8),
8      ack VARCHAR(4),
9      tipo CHAR(2),
10     numero VARCHAR(6),
11     iniT VARCHAR(4),
12     vuelo VARCHAR(10),
13     textI text,
14     finT VARCHAR(4),
15     PRIMARY KEY (ID)
16 ) ENGINE=InnoDB;
```

Todos los campos de la tabla mensajes son respectivamente la estructura de la trama de un mensaje ACARS. A excepción del campo **IDm INT**y el campo **fecha TIMESTAMP**. Estos dos campos se generan de manera automática. El primero es un identificador y el segundo registra la hora y fecha en que fue agregado dicho mensaje a la base de datos.

Habiendo instalado la librería de MySQL en Python, es posible empezar a programar nuestro código en *Python embedded* para conectar nuestro flujo de datos de GNU-Radio en

Inicio

```
import numpy as np
from gnuradio import gr
import mysql.connector
from mysql.connector import errorcode
```

Clase

```
class blk(gr.sync_block):

    try:
        cnx = mysql.connector.connect(user='xxxx',
password='xxxxxx',host='127.0.0.1', database='xxxx')
        print "Data Base ACARS Connnected "
    except mysql.connector.Error as err:
        if err.errno == errorcode.ER_ACCESS_DENIED_ERROR:
            print "Something is wrong with your user name or password"
        elif err.errno == errorcode.ER_BAD_DB_ERROR:
            print "Database does not exist"
        else:
            print err
```

Parámetros Iniciales

```
def __init__(self):
     gr.sync_block.__init__(
            self,
            name=' ACARS Mysql ',
            in_sig=[np.int8],
            out_sig=[np.int8]
        )
```

Función WORK

```
def work(self, input_items, output_items):
       output_items[0][:]=input_items[0]
        print map(str,map(unichr,output_items[0]))
        data_G={
        "IniM":"",
        "Modo":"",
        "RGO":"",
        "Ack":"",
        "Tipo":"",
        "Numero":"",
        "IniT":"",
        "Nsge":"",
        "Vuelo":"",
        "TextI":"",
        "FinT":""
        }
        for i in (output_items[0][0:3]):
            data_G["IniM"]=data_G["IniM"]+str(unichr(i))
        print data_G["IniM"]

        for i in (output_items[0][18:19]):
            data_G["Modo"]=data_G["Modo"]+str(unichr(i))
        print data_G["Modo"]

        for i in (output_items[0][19:26]):
            data_G["RGO"]=data_G["RGO"]+str(unichr(i))
        print data_G["RGO"]
```

Figura 7.10: Código del bloque de GNU-Radio para insertar los mensajes ACARS en MySQL a través de ***embedded Python de GNU-Radio.*** I Parte

Figura 7.11: Bloque de ACARS MySQL con ***embedded Python de GNU-Radio.*** I Parte

nuestra base de datos. Con la estructura anterior del bloque Python embedded de la Figura 7.9, vamos a explicar el código de la Figura 7.10 para conectar e insertar un mensaje ACARS a la DB.

En el recuadro Inicio se agrega la librería MySQL import mysql.connector. El Objeto permite insertar los valores de usuario, contraseña, Dirección IP y nombre de la Base de datos. Y además, la librería import errorcode que es un método que retorna un error dado caso qué ocurriera.

Pasando a la clase del bloque de Python, adicionamos el código para conectarse a la DB. Para ello recurrimos a un Try para intentar la conexión y Except dado caso que no se pueda ejecutar la conexión. En Try ejecutamos el método connect()a través del objeto cnx = mysql.connector para insertar los parámetros de conexión. Y ejecutamos el mensaje que la conexión fue exitosa con print Data Base ACARS Connnected . En Except ejecuta dos IF donde la primera se ejecuta cuando los parámetros indicados no coinciden con la Base de Datos. La segunda se ejecuta cuando la base de datos no existe.

En **Parámetros Iniciales** establecemos el tipo de variable a utilizar. Dado que se manipulan datos tipo Byte, establecemos `out_sig=[np.int8]` para los datos entrantes y salientes. En este momento el bloque resultaría como la figura 7.11.

Luego de establecer las condiciones iniciales del bloque Python, ahora solo falta programar las funcionalidades al bloque ACARS MySQL. Esencialmente la idea es obtener un mensaje a través de la variable `output_items[0]` , extraer y organizar los datos en variables de un diccionario. Posteriormente realizar la instancia de la conexión SQL para insertar los datos en la Base de Datos. Dentro de la función **Works**, el diccionario `data_G` inicializa las variables de los campos de la tabla **mensaje** creado anteriormente.

7.5.3. Control de flujo de datos de los Mensajes ACARS

GnuRadio tiene bloques predeterminados que determinan el flujo de datos del flujograma. Este flujo de datos esta dado por ítems o muestras contenidos en objetos denominados Array.

De modo que el flujo de datos en GnuRadio está dada por el tamaño o longitud de los arrays. En Python un array es un objeto de la librería Numpy cuyo atributos y métodos permiten la manipulación de los datos.

Existen generalmente dos bloques predeterminados en GnuRadio que en la práctica son clases constructoras programados en Python. La clase SyncBlock y Bloques Variables. Los bloques variables son el tipo de clases que permiten configurar con un escalar la correlación que existe con los datos entrantes y los datos salientes. Es decir, permite un tamaño variable del array saliente con respecto a longitud del array entrante. La clase SyncBlock no permite que exista array variable entre los datos entrantes y salientes, es decir la longitud del objeto Array para in y out debe ser la misma.

Si se tiene en cuenta que los mensajes ACARS no son mensajes con longitud fija sino variable, no existe un mecanismo que permita de manera automática identificar el comienzo y el termino de un paquete ACARS. El resultado sería que se inserten datos de manera aleatoria sin lograr identificar un mensaje ACARS. Por esa razón debe existir dentro del programa de GnuRadio un mecanismo que permita identificar la longitud de un mensaje ACARS y posteriormente capturar dicho mensaje de manera inmediata antes de insertar los datos en la base de datos.

Dado que el bloque Python Embbeded utiliza la clase SyncBlock, no permite cambiar la longitud de un array. Por ello se diseña otro bloque Python Embbeded para calcular la longitud o el número de ítems que tiene el mensaje ACARS en tiempo real junto con un Módulo Python como clase constructora. En el bloque MySQL ACARS obtiene el mensaje ACARS de acuerdo con la longitud del mensaje calculado previamente y posteriormente insertarlo en la base de datos.

7.5.3.1. Control de Flujo: Cálculo de número de items del mensaje

Antes de describir el código para hallar la longitud de un mensaje ACARS es pertinente programar la clase constructora o un constructor que permita manipular el resultado de dicho cálculo. La manipulación se refiere a "*transportar*" el resultado de la clase *Control Flujo* hacia la otra clase *MySQL ACARS* como un objeto.

El constructor se programa bajo el bloque de GnuRadio denominado Python Module, cuya funcionalidad permite ejecutar el código escrito e importar dicha programación en cualquier bloque de GNURadio. Para este caso sirve como puente entre los dos bloques Python Block, el primero donde se calcula la longitud denominado Flujo ACARS y el segundo ya denominado MySQL ACARS donde captura el mensaje según la longitud calculado, dado por el objeto de la clase constructor y pertinentemente la inserción del mensaje en la Base de datos.

La clase constructor Items queda determinado de la siguiente manera:

```
1  class Items:
2
3    def __init__(self,NoItems=0,ack=0):
4      self.NoItems=NoItems
5      self.ack=ack
6
7    def get_NoItems(self):
8      return self.NoItems
9    def get_ack(self):
10     return self.ack
11   def set_NoItems(self,n):
12     self.NoItems=n
13   def set_ack(self,a):
14     self.ack=a
15
16   AcarsObject=Items()
```

Un constructor permite inicializar variables cuya manipulación ejecuta métodos inherentes al objeto al instanciar una clase. El método constructor en Python está determinado por la palabra reservada __init__ cuya funcionalidad consiste en inicializar los variables al instanciar la clase Items. Dichas variables son: el No de ítems donde contiene el tamaño de array donde cabe un mensaje ACARS y la confirmación para indicar la existencia de un mensaje. Para cada variable existe un método SET y un método GET. El método set permite insertar o actualizar el valor de la variable. El método GET permite realizar consultas del valor actual de la variable. La instancia ocurre con la sentencia AcarsObject Items(), cuyo objeto es AcarsObject instancia del clase Items().

Creamos un nuevo Python Block y manteniendo la misma estructura del bloque MySQL en la Figura 7.9 configuramos el parámetro de entrada para lograr inicializar el objeto AcarsObject. El objeto AcarsObject debe ser parámetro de entrada de los dos bloques embebidos: Flujo ACARS y MySQL ACARS (Ver Imagen 7.12). Tal requerimiento permite la existencia de un puente de control entre esos dos bloques.

Luego de detallar el constructor e inicializar las variables y el objeto AcarsObject, se describe el algoritmo para calcular la longitud del mensaje ACARS. (Ver Imagen 7.13)

El algoritmo presentado en la figura 7.13 explica de forma general el funcionamiento del

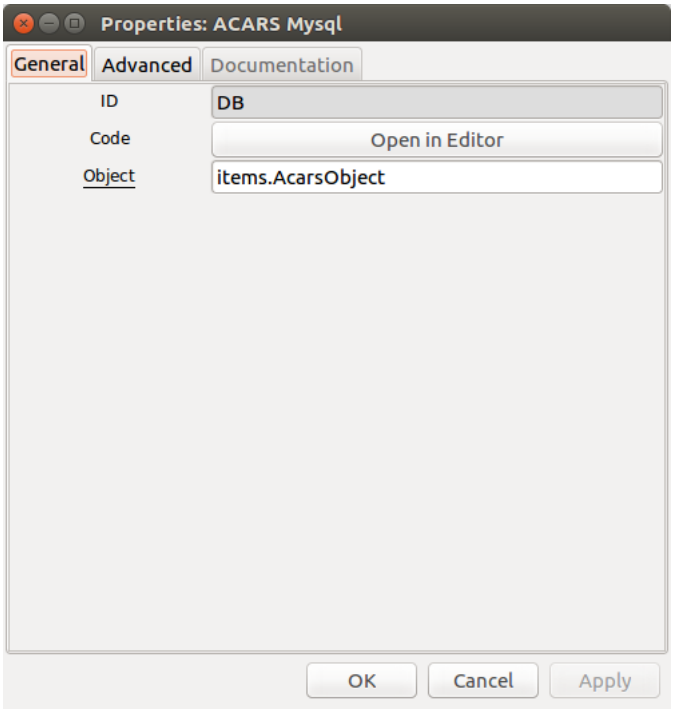

Figura 7.12: El objeto `ObjectACARS` como parametro de entrada del Bloque de ACARS MySQL

bloque de Flujo ACARS. Por cada array entrante en el bloque se recorre todos los elementos del vector a través de un FOR. Al mismo tiempo un contador para cada contar cada elemento del Array recorrido. De acuerdo con la teoría de la estructura de una trama ACARS, los caracteres *+ identifica el inicio de un mensaje ACARS y el carácter ¡ETX¿define el fin del mensaje. El algoritmo es sencillo, si al recorrer el Array existen dos caracteres +* consecutivos inicializa el contador en cero. Si al recorrer el Array existe el carácter ¡ETX¿entonces el contador ha contado N veces desde el inicio hasta el final del mensaje, es decir la longitud del paquete. Al cumplirse la anterior condición define la longitud del mensaje y el valor "on" ACK a través del objeto AcarsObject con el método SET para cada variable.

7.5.3.2. Control de Flujo: Control de Items del mensaje

En realidad, se realiza una modificación en la estructura en el código escrito en el bloque MySQL ACARS. En primera instancia agregamos un parámetro inicial al bloque, tal como se hizo en el bloque de Flujo ACARS en la Figura 7.12.

Después de inicializar el objeto ObjectACARS, en la Figura 7.14 describe el algoritmo para capturar el mensaje ACARS de acuerdo con la longitud calculado. Este último código es aún más sencillo que el anterior. En vez de un contador, este es un acumulador del Array

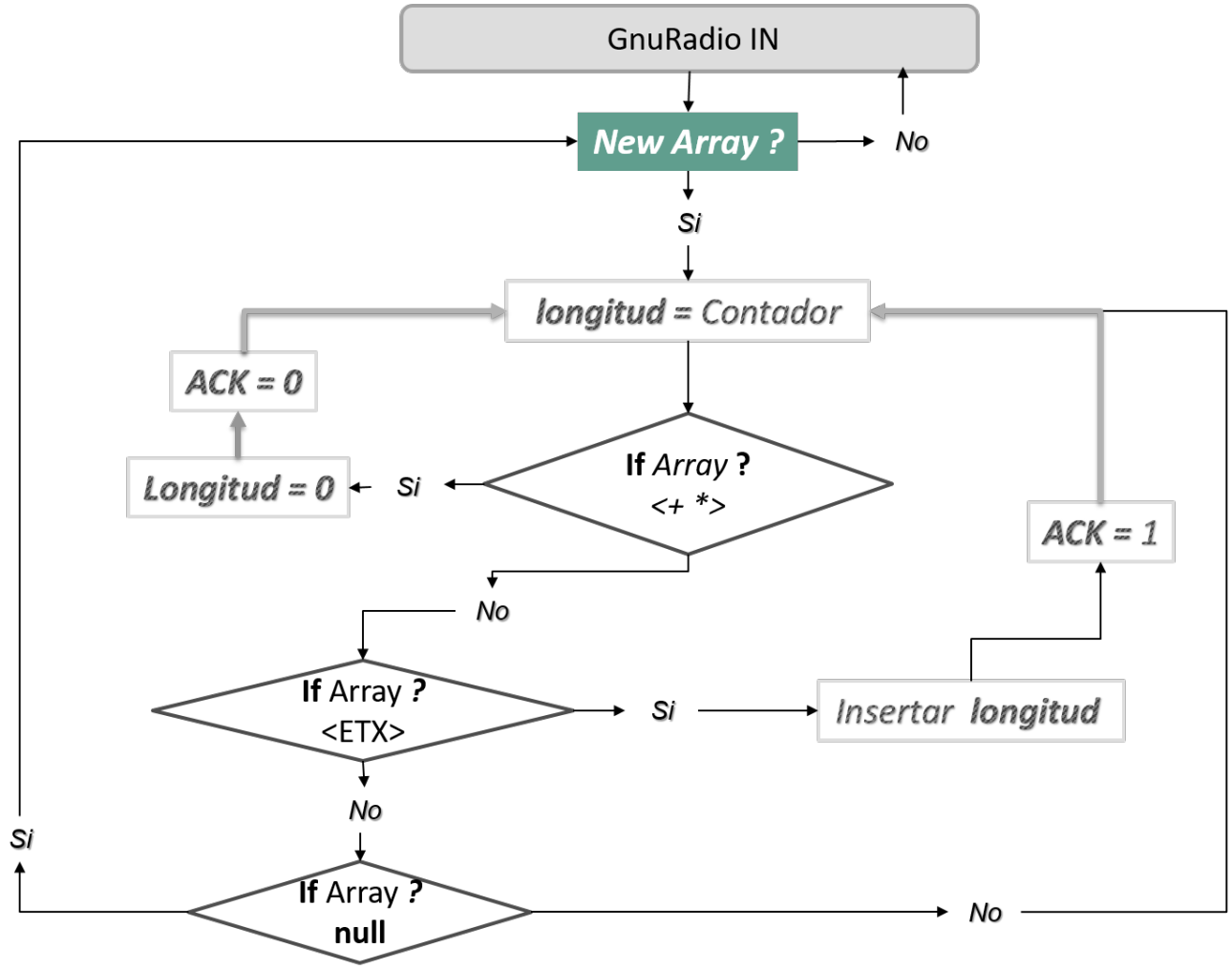

Figura 7.13: Algoritmo para hallar la longitud de un mensaje ACRS

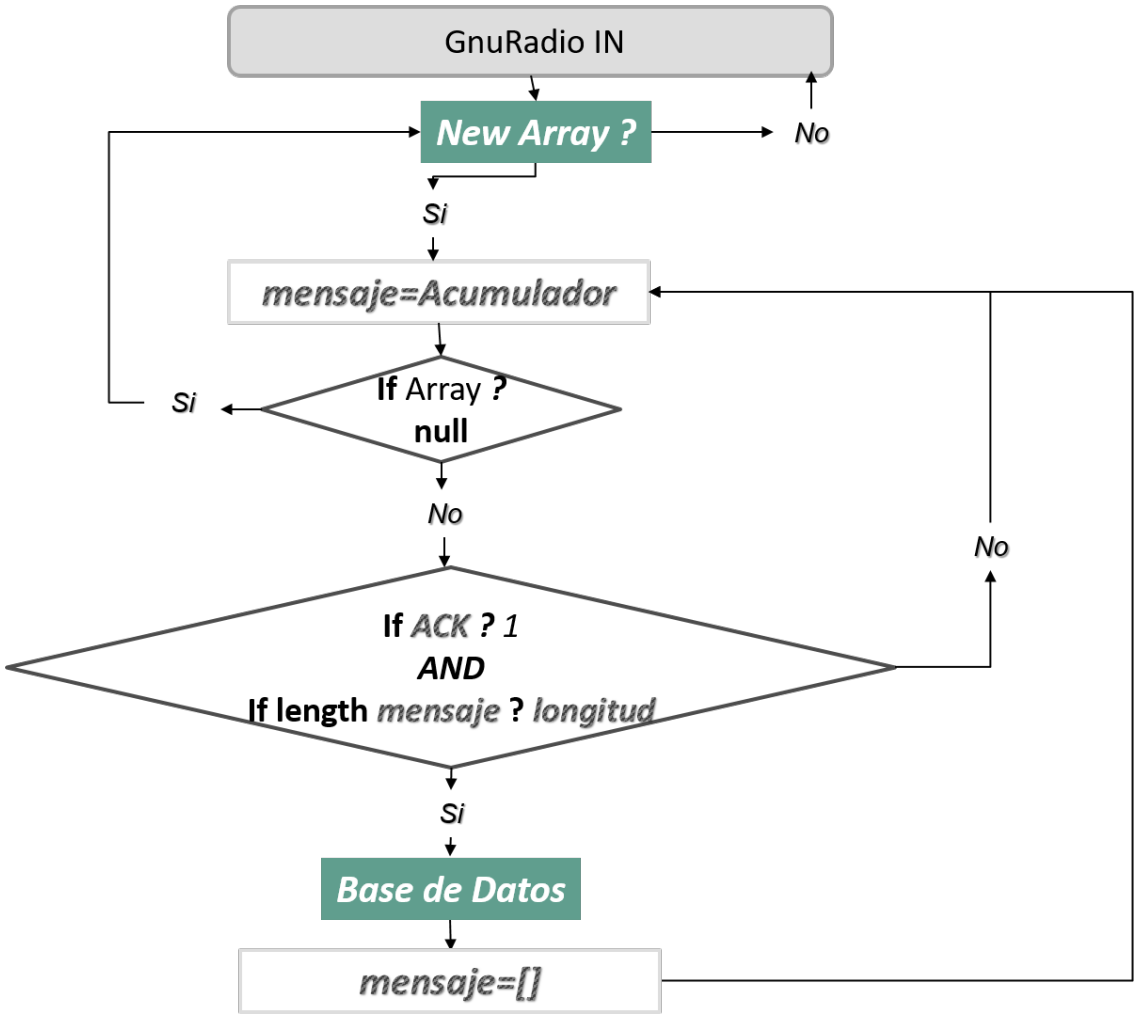

Figura 7.14: Algoritmo para captar un Mensaje ACARS

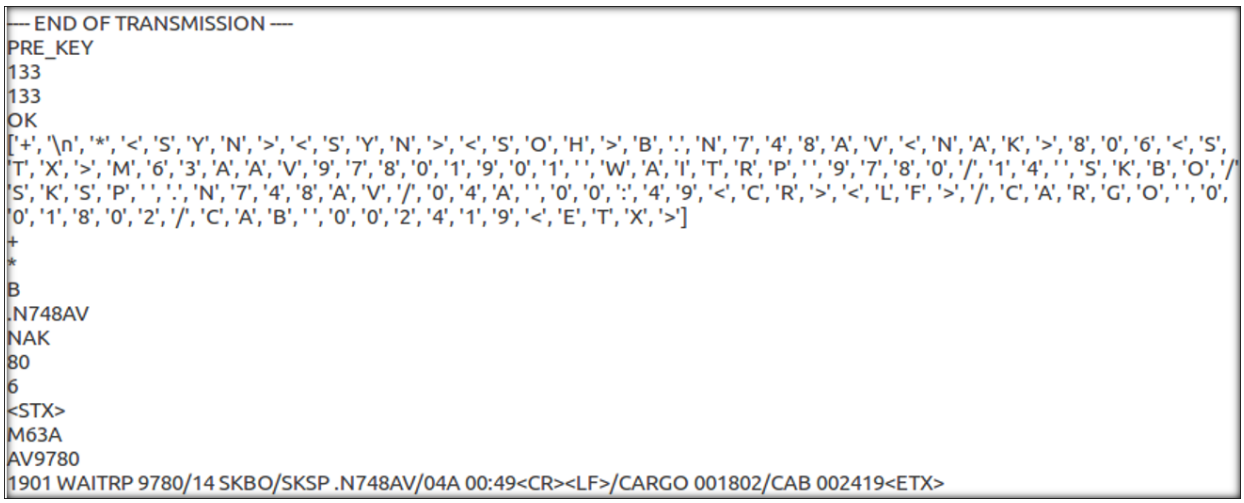

Figura 7.15: Resultado de I mensaje de los algoritmos desarrollados

entrante del bloque. El acumulador esta contantemente condicionado a un estado lógico. El estado lógico debe cumplir que la longitud del acumulador deber ser exactamente a la longitud del mensaje calculado previamente en el bloque y el reconocimiento (ACK) debe ser igual a 1. Cuando lo anterior se cumple, el acumulador se vuelve una lista para ser tratado por el código de inserción a la base de datos ya programada. Es importante señalar que después de haber insertado el mensaje a la DB, volver NULL o vaciar el acumulador para contener el próximo mensaje.

Los resultados de los algoritmos de la Figura 7.13 y Figura 7.14 se puede apreciar en la Figura 7.15 y la Figura 7.16.En la Figura 7.15 corresponde a un mensaje ACARS cuya longitud es calculado y es totalmente diferente al tamaño del paquete ACARS DE LA Figura 7.16, logrando así identificar cada mensaje de manera correcta e independiente para lograr insertar dichos mensajes en la base de datos.

7.6. Optimización del sistema para visualizar los mensajes ACARS entre DB y aplicativo Web

Por ahora existe la aplicación de GNU-Radio para insertar los mensajes demodulados y decodificados en una base de datos, y la plataforma web para conocer la ubicación de la aeronave como también la interacción de los mensajes ACARS con el usuario. Solo resta adaptar el sistema para mostrar los mensajes ACARS en una tabla dinámica y las coordenadas geográficas por medio de PHP, JavaScript y HTML.

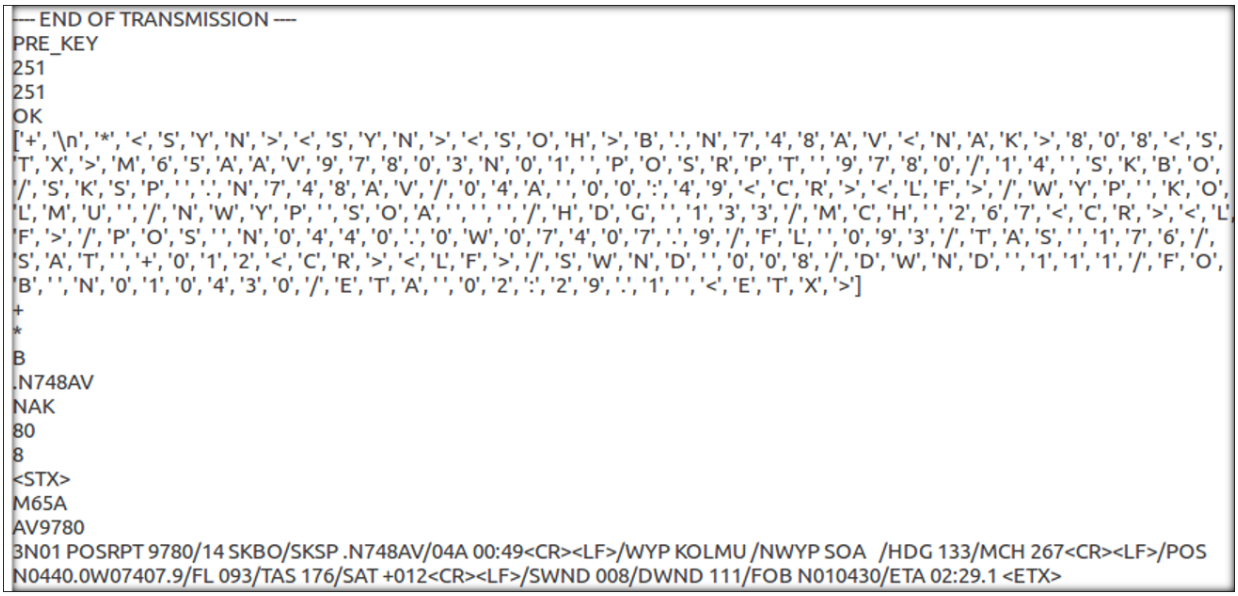

Figura 7.16: Resultado de II mensaje de los algoritmos desarrollados

PHP como la programación del servidor para realizar la consulta de los mensajes ACARS provenientes del programa GNURadio en la base de datos y organizar los datos en archivos XML. Como agregado PHP tiene la obligación de revisar e identificar en cada mensaje consultado la existencia de una coordenada geográfica para indexarlo en otro archivo XML para el API de Google Maps. JavaScript para realizar peticiones personalizadas al servidor para la tabla dinámica y la inclusión de los puntos geográficos en Google Maps. Por último, HTML para diseñar la tabla dinámica, el contenedor de los mensajes ACARS.

7.6.1. Código PHP: Consultas de los mensajes ACARS con PHP.

Manteniendo la estructura del código explicado en la sección 7.4.1 para realizar la consulta en la base de datos y organizar los datos en un archivo XMl. En esa sección explica como realizar la conexión a la DB como la consulta a una base de datos. El único cambio es la estructura de la consulta, debido a que la tabla contiene mas campos pero con la misma estructura de programación.

Solo es necesario definir y describir de que forma se organiza la información del mensaje ACARS en el archivo XML. Vale la pena resaltar que se introduce nuevos comando XMl DOM basado en PHP para obtener el resultado de la Figura 7.17

```
1 <?php
2 //Archivo de conexion a la base de datos
```

```
3
4  require('connection.php');
5
6  $result= $connect->query("SELECT IDm,fecha,modo,rgo,ack,tipo,numero,Nsge,
7  vuelo,textI,finT FROM mensaje ORDER BY IDm ASC");
8
9  if(!$result){
10   echo "Lo sentimos, este sitio web esta experimentando problemas.";
11 }
12
13 // Start XML file, create parent node
14 $dom = new DOMDocument("1.0");
15 $node = $dom->createElement("ACARS");
16 $parnode = $dom->appendChild($node);
17
18 header("Content-type: text/xml");
19
20 // Add to XML document node
21
22 $i=0;
23 while ($row = $result->fetch_assoc()){
24 $i=$i+1;
25 $nod = $dom->createElement("Mensaje");
26 $newnode = $parnode->appendChild($nod);
27 $newnode->setAttribute("Mensaje", $i);
28 $newnode->setAttribute("Fecha", $row['fecha']);
29
30 $ID = $dom->createElement("Identificacion");
31 $ID = $newnode->appendChild($ID);
32 $ID->setAttribute("Vuelo",$row['vuelo']);
33
34 $RGO = $dom->createElement("RGO");
35 $RGO = $ID->appendChild($RGO);
36 $text = $dom->createTextNode($row['rgo']);
37 $text = $RGO->appendChild($text);
38
```

```
39 $ACK = $dom->createElement("Acknowledgement");
40 $ACK = $ID->appendChild($ACK);
41 $text = $dom->createTextNode($row['ack']);
42 $text = $ACK->appendChild($text);
43
44 $Nmsge = $dom->createElement("No_Mensaje");
45 $Nmsge= $ID->appendChild($Nmsge);
46 $text = $dom->createTextNode($row['Nsge']);
47 $text = $Nmsge->appendChild($text);
48
49 $Con = $dom->createElement("Contenido");
50 $Con = $newnode->appendChild($Con);
51 $Con->setAttribute("Tipo",$row['tipo']);
52
53 $Mod = $dom->createElement("Modo");
54 $Mod= $Con->appendChild($Mod);
55 $text = $dom->createTextNode($row['modo']);
56 $text = $Mod->appendChild($text);
57
58 $Sec = $dom->createElement("Secuencia");
59 $Sec= $Con->appendChild($Sec);
60 $text = $dom->createTextNode($row['numero']);
61 $text = $Sec->appendChild($text);
62
63 $tex = $dom->createElement("Informacion");
64 $tex= $Con->appendChild($tex);
65 $text = $dom->createTextNode($row['textI']);
66 $text = $tex->appendChild($text);
67 }
68 echo $dom->saveXML();
69 ?>
```

En la sección 7.4.1 establecimos que la función `fetch_assoc()` emula una sentencia FOR sistemático. Es decir por N consultas halla en la variable `result` N ciclos va a realizar. Teniendo en cuenta lo anterior, para contar los mensajes existentes para el final de todos los

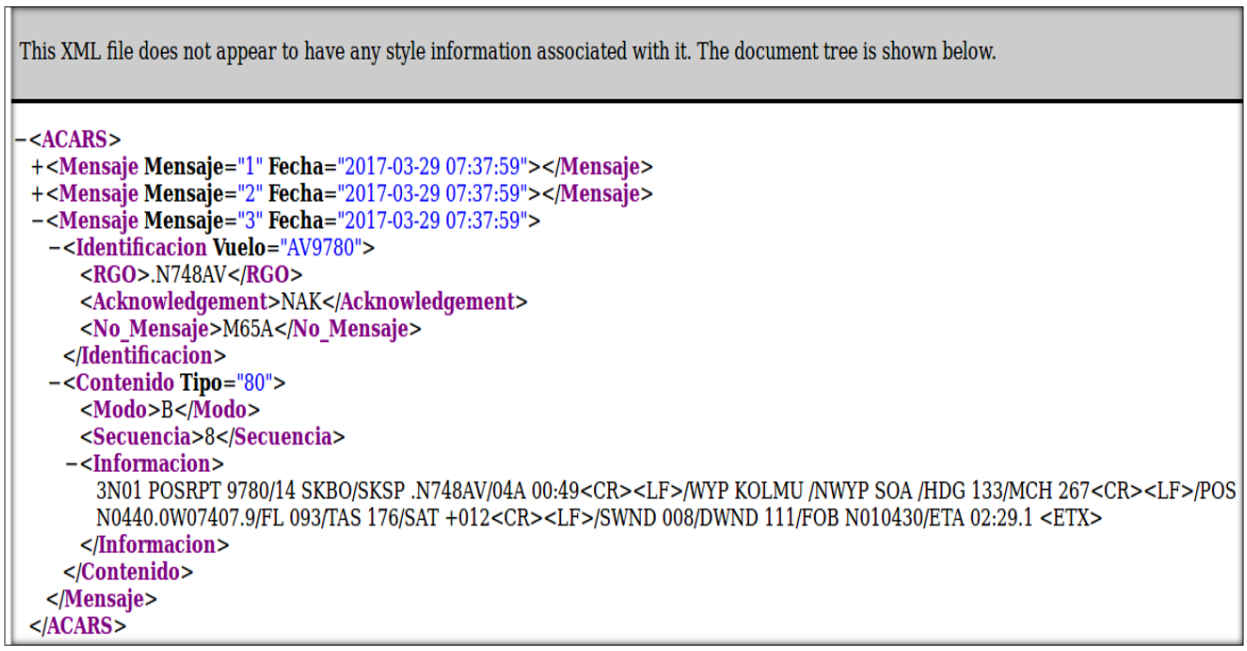

Figura 7.17: Archivo XML de los mensajes ACARS contenidos en la base de datos por medio de PHP

ciclos se logra por medio de la variable i=i+1 . Un archivo XML es un texto plano cuya organización es un modelo jerárquico, compuesto por nodos, Sub-Nodos y atributos donde los subnodos son nodos hijos del nodo padre. Esa jerarquía se logra con la sentencia NodoPadre −> appendChild(NodoHijo) .En la Figura 7.17 el nodo **Mensaje** es nodo hijo del nodo padre **ACARS**. Así como el nodo *Identificación* y *Contenido* son hijos nodos del Sub-nodo padre **Mensaje**. El procedimiento para organizar los datos jerárquicamente es el siguiente, Primero crea el nodo a partir del nodo `PADRE` que es **ACARS**. Luego indexar dicho elemento en algún *Subnodo Padre*.

Luego de haber construido todo el árbol genealógico del archivo XMl, solo resta insertar los datos. En XML existen dos formas: por Atributo y por Texto. En la Imagen 7.17, el Sub-nodo padre **Mensaje** tiene dos Atributos: *Mensaje* y *Fecha*. El valor de los Nodos Hijos como *NoMensaje* o *Información* no tienen atributos sino valores por Texto. Para insertar un valor tipo texto es necesario crear un nodo tipo Texto a partir del nodo padre **ACARS** y la variable de la consulta MySQL, para luego insertar el Nodo tipo Textual en el nodo Hijo. Para valores tipo Atributo solo es necesario indexar en el nodo hijo la variable del atributo y la variable MySQL, con la sentencia serAttribute .

7.6.2. Código PHP: Extracción de la ubicación geográfica de las consultas.

En la anterior sección exportamos en archivos XML por medio de PHP los mensajes ACARS, listos para ser tratados en el aplicativo WEB. Solo queda exportar otro tipo de archivo XML donde contenga los mensajes cuya información este contenida una ubicación geográfica. Es decir, si de los N mensajes contenidos en el archivo XML solo un pequeño grupo de N son mensajes que contienen una información geográfica. El algoritmo debe ser capaz de seleccionar solo ese pequeño grupo y extraer la ubicación geográfica para disponerla en otro archivo XML.

Entonces, hay dos archivos XML, el primero están contenidos todos los mensajes ACARS que existan en la base de datos para ser dispuesto en la Tabla Dinámica del aplicativo Web. Y el segundo archivo XMl son las coordenadas geográficas de cada mensaje (No todos los mensajes contienen información acerca de la ubicación geográfica de la aeronave) para ser utilizado en el API de Google Maps programado en el aplicativo Web.

Primero se debe filtrar los mensajes cuya información se relacione con una coordenada Geográfica. Para ello existe la etiqueta **/POS**, que identifica una coordenada sexagesimal(Grados, Minutos y Segundos). La ubicación referente es N (Norte) y W (Oeste). La estructura para la coordenada N esta determinada por (DDMMS)y W (DDDMMS) donde D son grados, M Minutos y S segundos. Conociendo la estructura de una ubicación en el mensaje ACARS, el primer algoritmo (Ver Figura 7.18) describe determinar la ubicación de la coordenada dentro del mensjae ACARS y el tamaño de la misma. Y el segundo algoritmo es extraer la ubicación N y W para convertirlo en sistema decimal e insertarlo en un archivo XML.

El algoritmo de la Figura 7.18 se constituye en un FOR en la cual recorre todo el array donde esta contenido la consulta MySQL. Si al recorrer el Array no encuentra la palabra **/POS** pasa a la siguiente consulta. Al encontrar la etiqueta **/POS** captura la posición en el array para fines del segundo algoritmo. Al mismo tiempo genera dos contadores incremental, esto con la finalidad de calcular la longitud de cada coordenada.

El segundo algoritmo (Ver Figura 7.19) se detalla en código en PHP, cuyo función es volver a recorrer el Array y extraer las dos coordenadas a partir de la longitud y la ubicación de la etiqueta **/POS** con la variable *(init)*.

```
1 <?php
2 $Lagrados=$latitud[0].$latitud[1];
3 $Laminutos=$latitud[2].$latitud[3];
4 $Lasegundos=$latitud[5];
5
6 $Lngrados=$longitud[0].$longitud[1].$longitud[2];
```

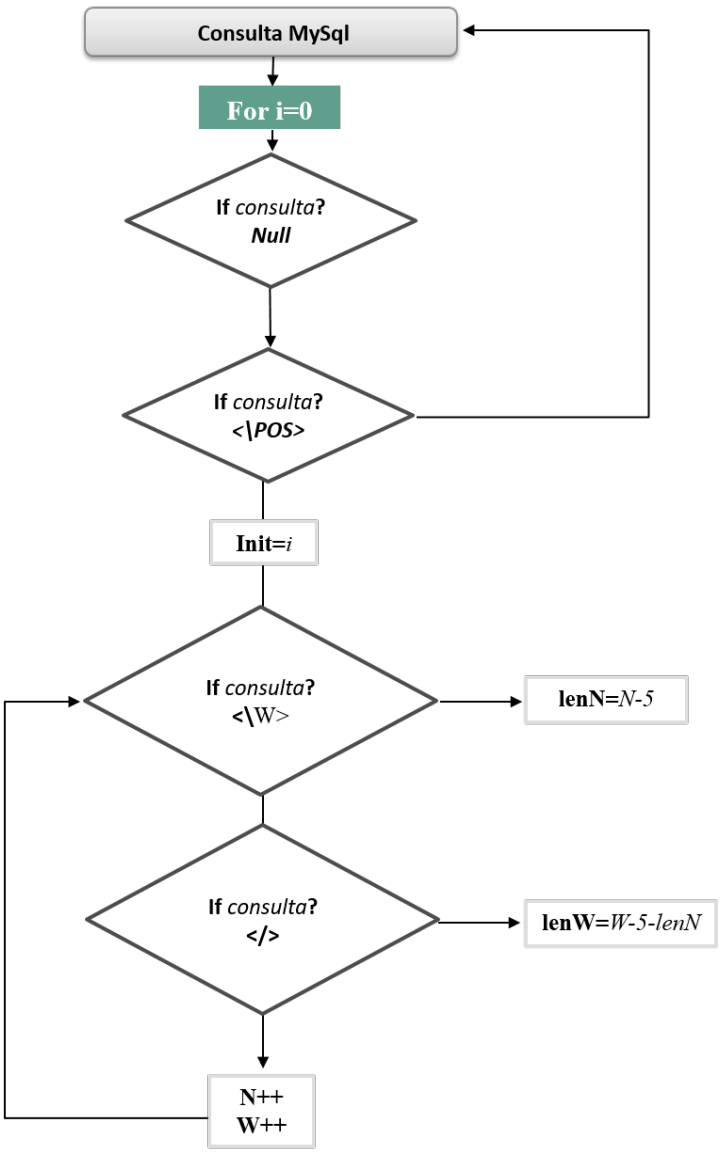

Figura 7.18: I Algoritmo para determinar la longitud y ubicación dela coordenada geográfica de cada mensaje ACARS

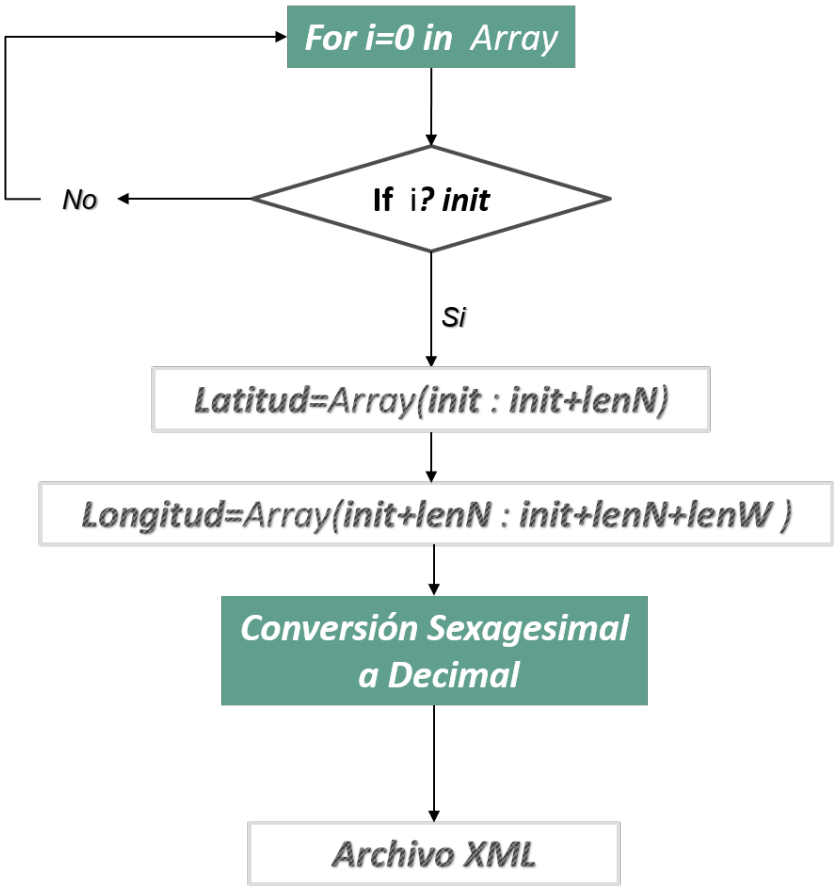

Figura 7.19: II Algoritmo para extraer la longitud y latitud del mensaje ACARS

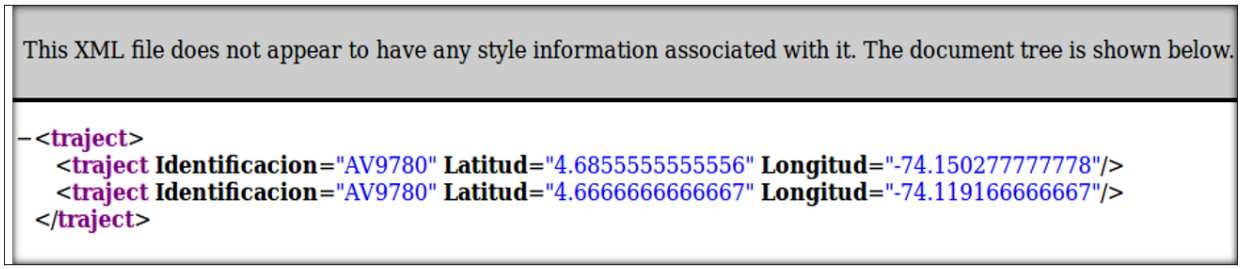

Figura 7.20: Ubicación geográficas de los mensajes ACARS dispuestos en la base de datos

```
7  $Lnminutos=$longitud[3].$longitud[4];
8  $Lnsegundos=$longitud[6];
9
10 $latituds= $Lagrados+($Laminutos/60)+($Lasegundos/3600);
11 $longituds=-($Lngrados+($Lnminutos/60)+($Lnsegundos/3600));
12 ?>
```

La conversión de sexagesimal a decimal de la Figura 7.19 es la sumatoria de la división de los grados en 1, los minutos en 60 y los segundos en 3600. Para efectuar la anterior operación es necesario extraer de la variable Latitud y Longitud los grados, minutos y segundos manteniendo la estructura (DDMMS) para latitud y (DDDMMS) para longitud. En la Figura 7.20 esta resultado al insertar la variable `latituds` y `longituds` como atributo del nodo padre traject.

7.6.3. Código JavaScript: Visualización de mensajes ACARS en una tabla dinámica con AJAX

Cuando se dice de una tabla dinámica se refiere a una tabla basada en AJAX, es decir peticiones al servidor reiteradamente por un lapso constante de tiempo. Esto permite una visualización de los mensajes ACARS en tiempo real. En la sección 7.4.2 explica el código para realizar peticiones personalizadas al servidor con AJAX por medio de JQUERY. La única sentencia adicional es `$(".ola").empty()` cuya funcionalidad permite vaciar el contenido de la tabla HTML para evitar que se repitan los mensajes de la petición anterior. La etiqueta `.ola` hace referencia a la clase HTML donde esta programada la tabla.

```
1
2  function Message(){
```

```
3  var filename = 'xxxxxx.php';
4
5  setInterval(request, 10000);
6
7  function request(){
8    $(".ola").empty();
9    $.ajax({
10     type: "GET",
11     cache: false,
12     url: filename ,
13     dataType: "xml",
14     success: parseXML,
15     error : onXMLLoadFailed
16   });
17 }
```

Cuando la ptición AJAX es sastidfactoria, el paso a seguir es extraer la información del archivo XMl. Como bien habíamos explicado en XML existen dos formas de insertar datos, en JavaScript tambien existe esos mismos dos modos. **Por atributo**, la forma de extraer los datos se logra con la sentencia `.attr(Atributo)`. **Por Nodo textual**, se realiza con la sentencia `.text()`.

```
1  function parseXML(xml){
2    $(xml).find("Mensaje").each(function(i){
3      var NoMensaje = $(this).attr('Mensaje');
4      var fecha = $(this).attr('Fecha');
5      var id = $(this).find('Identificacion').attr('Vuelo')
6      var rgo = $(this).find('Identificacion').find('RGO').text();
7      var ack = $(this).find('Identificacion').find('Acknowledgement').text();
8      var SecMensaje = $(this).find('Identificacion').find('No_Mensaje').text();
9      var tipo = $(this).find('Contenido').attr('Tipo');
10     var modo = $(this).find('Contenido').find('Modo').text();
11     var secuencia = $(this).find('Contenido').find('Secuencia').text();
12     var mensaje = $(this).find('Contenido').find('Informacion').text();
```

Cada nodo del archivo XML esta almacenado en las variables de JavaScript, es decir, ya están almacenados en el dispositivo del usuario, solo falta mostrar dichas variables en el aplicativo WEB. Para ello, seleccionamos la etiqueta de HTML `$(".ola")` donde está la tabla para lograr insertar los datos en cada campo de la tabla. Lo anterior se cumple gracias al comando `append(Codigo HTML)` . Esa sentencia permite indexar algún código HTML, en este caso vamos a insertar una fila de la tabla para cada mensaje ACARS. Donde `<tr>` indica una fila de la tabla y `<td>` un campo de de la fila.

```
1      $(".ola").append("<tr><td>"+NoMensaje+"</td><td>"+fecha+"</td><td>"
2      +modo+"</td><td>"+rgo+"</td><td>"+ack+"</td><td>"+tipo+"</td><td>"
3      +secuencia+"</td><td>"+SecMensaje+"</td><td>"+id+"</td><td>"
4      +mensaje+"</td></tr>");
5    });
6  }
```

Pasando al cuerpo HTML, la etiqueta `<table>` enmarca todo en contenido de la tabla. Con la etiqueta `<thead>` declaramos el encabezado de la misma. Tenga en cuenta que utiliza las mismas sentencias `<tr>` y `<th>` para definir la fila y los campos. Y por ultimo la clase .ola en la etiqueta `<tbody>` que a través de JavaScript insertan las filas en la tabla de manera implicita. El resultado del metodo y el codigo empleado para materializar la tabla dinámica se refleja en la Figura 7.21.

```
1      <h1 id ="tiB">Live ACARS Messages</h1>
2      <table border="1" id="same">
3        <thead>
4          <tr>
5            <th class="1">N</th>
6            <th class="2">Date</th>
7            <th class="3">Mode</th>
8            <th class="4">RGO</th>
9            <th class="5">Acknowledgement</th>
10           <th class="6">Label</th>
11           <th class="7">Sequence</th>
12           <th class="8">Seq Message</th>
13           <th class="9">Flight</th>
14           <th class="10">Message</th>
15         </tr>
```

```
16            </thead>
17            <tbody class="ola">
18            </tbody>
19        </table>
```

Live ACARS Messages

N°	Date	Mode	RGO	Acknowledgement	Label	Sequence	Seq Message	Flight	Message
1	2017-03-29 07:37:59	B	.N748AV	NAK	80	6	M63A	AV9780	1901 WAITRP 9780/14 SKBO/SKSP .N748AV/04A 00:49/CARGO 001802/CAB 002419
2	2017-03-29 07:37:59	B	.N748AV	NAK	80	7	M64A	AV9780	3N01 POSRPT 9780/14 SKBO/SKSP .N748AV/04A 00:49/WYP SKBO13/NWYP KOLMU /HDG 133/MCH 273/POS N0441.8W07409.1/FL 082/TAS 180/SAT +015/SWND 008/DWND 122/FOB N010249/ETA 02:29.5
3	2017-03-29 07:37:59	B	.N748AV	NAK	80	8	M65A	AV9780	3N01 POSRPT 9780/14 SKBO/SKSP .N748AV/04A 00:49/WYP KOLMU /NWYP SOA /HDG 133/MCH 267/POS N0440.0W07407.9/FL 093/TAS 176/SAT +012/SWND 008/DWND 111/FOB N010430/ETA 02:29.1

Figura 7.21: Tabla Dinámica como contenedor de los mensajes ACARS en tiempo real

Capítulo 8

III Fase: *Pruebas Experimentales*

El actual capitulo describe el prototipo completo puesta en marcha desde la recepción hasta el aplicativo Web. Esta sección presenta dos escenarios de pruebas, el primer escenario es en cercanía al aeropuerto internacional el Dorado de Bogotá DC. El segundo escenario el prototipo estará ubicado en la ciudad de Cajica. Para el segundo escenario y de forma permanente, el sistema de recepción estará dispuesto en las instalaciones de WIRID LAB en el Campus de la Universidad Militar Nueva Granada ubicado en la ciudad de Cajicá, Cundinamarca.

Antes de comenzar con la pruebas es importante realizar un resumen de la estructura del prototipo así como el listado de los códigos desarrollados para el funcionamiento de todo el sistema.Desde la estructura del sistema es claro que se basa en tres bloques. Respectivamente los bloques son: el componente de recepción y procesamiento de señales ACARS, Los componentes para adaptar el sistema con GNU-Radio y MySQL (Base de Datos), y el servicio Web para la visualización de los mensajes ACARS.

El primer Bloque esta contenido todos los componentes de GNU-Radio de procesamiento de señales para adaptar la señal recibida por dispositivo RTL-SDR sin errores de fase, frecuencia y tiempo de la señal ACARS. Después de haber mitigado los errores de recepción, el sistema se adapta una demodulación ACARS para entregar mensajes decodificados. El segundo bloque sin salir de GNU-Radio se diseña, desarrolla e implementa un par de algoritmos con la capacidad de controlar el flujo de los mensajes ACARS tal que la inserción del contenido de los mensajes en la base de datos sea eficiente.

I Bloque	
Código	**Características**
`ACARS.grc`	Prototipo de recepción, De-Modulación, Codificación, Control de Flujo e inserción de datos en la base de datos.
II Bloque	
Código	**Características**
`ControlFlujo.py`	Algoritmo de control de flujo ACARS desarrollado en **Python** e implementado como bloque de GNURadio dentro del archivo `ACARS.grc`
`ACARSMySQL.py`	Algoritmo de inserción de datos ACARS en MySQL desarrollado en **Python** e implementado como bloque de GNURadio dentro del archivo `ACARS.grc`

Cuadro 8.1: Software Desarrollado - Archivos y aplicaciones desarrollados durante el bloques I y el Bloque II. Todos los desarrollos se basan totalmente en el entorno de GNURadio.

El tercer bloque es el aplicativo web cuyos servicios son la visualización de todos los mensajes ACARS en una tabla y el servicio de Google Maps para observar la ubicación de los mensajes cuyo contenido tenga una posición gráfica. Los dos servicios se ejecutan en tiempo real. Este es el bloque donde mas desarrollo a nivel de software contiene, Pues se trabaja con programación del lado del servidor PHP, Servicios de tiempo real para servidores Jquey-Ajax y los servicios WEB implementados en HTML5-CSS-JavaScript.

III Bloque	
Código	**Características**
`LocationACARS.php`	Algoritmo de consulta y estructura de los mensajes cuyo contenido exista una coordenada geográfica en archivos XML desarrollado en **PHP**.
`mensajeACARS.php`	Algoritmo de consulta y estructura de los mensajes consultados en archivos XML desarrollado en **PHP**.
`AcarsMess.js`	Algoritmo para procesar archivo XML de `mensajeACARS.php` y estructurarlo en la tabla HTML. Archivo basado en **JavaScript-Jquery-Ajax**.
`geolocation.js`	Algoritmo para procesar archivos XML de `locationACARS.php` y proyectarlo en la API de Google Maps. Archivo basado en **JavaScript-Jquery-Ajax**.
`index.html`	Archivo HTML como presentación del aplicativo Web ACARS.
`liveMap.html`	Archivo HTML contenedor del API de google maps y script `geolocation.js`
`liveTable.html`	Archivo HTML contenedor de la tabla dinámica y script `AcarsMess.js`

Cuadro 8.2: Software Desarrollado - Archivos desarrollados durante en el III bloque.

Figura 8.1: Visualización de la ubicación de uno de los mensajes de la tabla dinámica de la Figura 8.2

8.1. I Escenario: Aeropuerto *El Dorado*

La prueba se desarrolla en el aeropuerto internacional El Dorado cuya metodología consiste en captar las señales ACARS en la frecuencia 131.917MHZ. En la Figura 8.1 visualiza tres mensajes ACARS codificados en la tabla dinámica donde solo uno de ellos contiene información geográfica de la aeronave. El mensaje donde esta contenido la coordenada geográfica se plasma sobre el API de google Maps de la Figura 8.2.

Live ACARS Messages

N°	Date	Mode	RGO	Acknowledgement	Label	Sequence	Seq Message	Flight	Message
1	2017-04-11 17:08:24	B	.CC-BAU	NAK	H1	7	D23A	LA3096	#DFBA47/A32047,1,1/AMDAR,REPORT/C1100,CC-BAU,0445N,07404W/C214,171356,11342,0065,132,016/C311342,226,/
2	2017-04-11 17:08:25	B	.CC-BAU	NAK	H1	8	D24A	LA3096	#DFBA47/A32047,1,1/AMDAR,REPORT/C1100,CC-BAU,0449N,PHKOL((C214,171441,13217,0030,102,016 /C313217,NLNS
3	2017-04-11 17:08:25	B	.N490TA	NAK	80	9	M07A	TA0362	3N01 POSRPT 0362/14 SKBO/MSLP .N490TA/04C 17:15/WYP KOLMU /NWYP SOA /HDG 139/MCH 290/POS N0440.8W07407.8/FL 094/TAS 190/SAT +013/SWND 013/DWND 161/FOB N009977/ETA 19:40.6

Figura 8.2: Tabla de la primera señal ACARS De-Modulada.

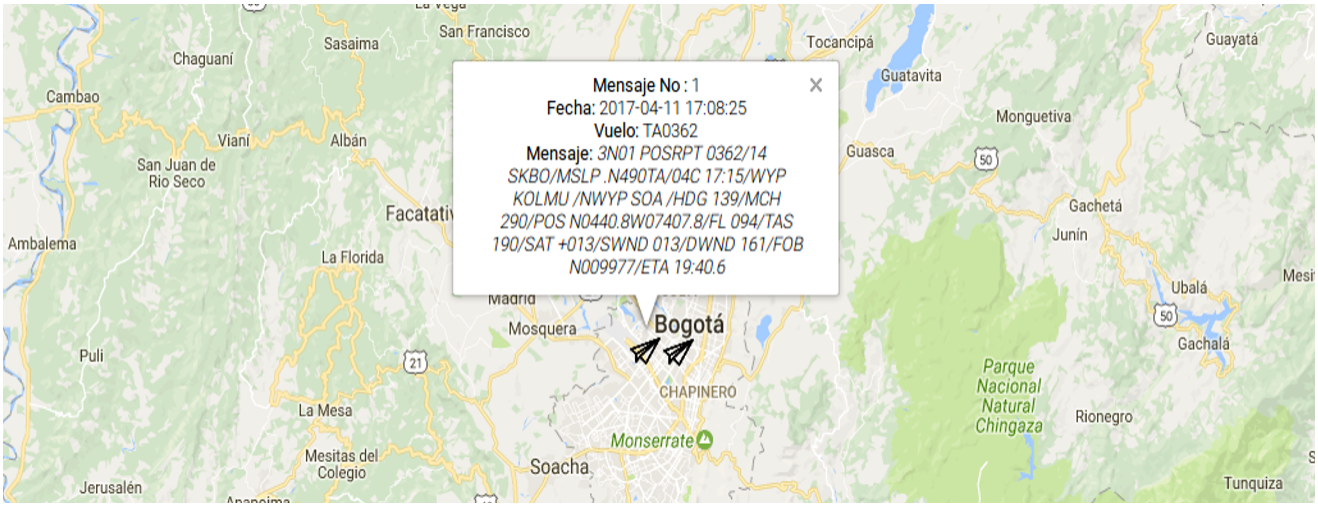

Figura 8.3: Visualización del mensaje de la tabla dinámica de la Figura 8.2 sobre el mismo mapa de la Figura 8.1

Cada punto de google maps debe desplegar información complementaria como parte del mensaje, tal como se refleja en la Figura 8.3. El mensaje No 3 de la Figura 8.2tiene como referencia `TA0352` como número de vuelo, la cual coincide con la información que despliega en el cuadro de dialogo de la Figura 8.3. Al mismo tiempo el aplicativo Web identifica otro mensaje ACARS y los plasma de manera automática, en tiempo real y sin interrumpir o actualizar la pagina.(Ver Figura 8.3). El nuevo mensaje corresponde a los nuevos mensajes de la tabla dinámica de la Figura 8.4. Y desde luego el cuadro de dialogo en Google Maps tal como se ve en la Figura 8.5.

Live ACARS Messages

N°	Date	Mode	RGO	Acknowledgement	Label	Sequence	Seq Message	Flight	Message
1	2017-04-11 17:13:06	B	.N596EL	NAK	H1	4	D37C	AV8568	#DFB00/V209,20,096,152,00,00,00000 /V3048,089,1,1,00060,00010,2221/V4048,083,1,1,00060,00010,2221 /X111187,0237,099,037,1006/X211265,0232,100,037,0998,/
2	2017-04-11 17:13:07	B	.N596EL	NAK	H1	5	F46A	AV8568	#M1BPRG/DTSKBG,35O,98,0135066FF8
3	2017-04-11 17:13:07	B	.N596EL	NAK	80	6	M75A	AV8568	3N01 POSRPT 8568/15 SKBO/SKBG .N596EL/04A 01:02/WYP R /NWYP GUXUN /HDG 125/MCH 271/POS N0440.4W07405.0/FL 100/TAS 177/SAT +010/SWND 008/DWND 067/FOB N005424/ETA 01:35.3

Figura 8.4: Tabla de la segunda señal ACARS De-Modulada.

Figura 8.5: Visualización de la ubicación y el mensaje de la tabla dinámica de la Figura 8.4

Siguiendo la secuencia de la prueba, la tabla dinámica de la Figura 7.6 dos de los tres mensajes disponibles, contienen información geográfica cuyo ID de Vuelo es el mismo. Quiere decir que el aplicativo web debe reconocer que cuando existan varios puntos geográficos de un mismo vuelo debe trazar una sola trayectoria entre los mismos. El resultado de la trayectoria en Google maps se refleja en la Figura 8.7.

Live ACARS Messages

N°	Date	Mode	RGO	Acknowledgement	Label	Sequence	Seq Message	Flight	Message
1	2017-04-11 17:17:23	B	.N748AV	NAK	80	6	M63A	AV9780	1901 WAITRP 9780/14 SKBO/SKSP .N748AV/04A 00:49/CARGO 001802/CAB 002419
2	2017-04-11 17:17:24	B	.N748AV	NAK	80	7	M64A	AV9780	3N01 POSRPT 9780/14 SKBO/SKSP .N748AV/04A 00:49/WYP SKBO13/NWYP KOLMU /HDG 133/MCH 273/POS N0441.8W07409.1/FL 082/TAS 180/SAT +015/SWND 008/DWND 122/FOB N010249/ETA 02:29.5
3	2017-04-11 17:17:24	B	.N748AV	NAK	80	8	M65A	AV9780	3N01 POSRPT 9780/14 SKBO/SKSP .N748AV/04A 00:49/WYP KOLMU /NWYP SOA /HDG 133/MCH 267/POS N0440.0W07407.9/FL 093/TAS 176/SAT +012/SWND 008/DWND 111/FOB N010430/ETA 02:29.1

Figura 8.6: Tabla de tercera señal ACARS De-Modulada.

Figura 8.7: Visualización de la ubicación y el mensaje de la tabla dinámica de la Figura 8.6

Ahora, el prototipo tiene una deficiencia cuando existe errores de codificación que son provocados por los errores de recepción del sistema. Un ejemplo se puede ver en la figura 8.8 como resultado la de-modulación y la codificación. Visiblemente existe incoherencia en el contenido de los mensajes donde los algoritmos del aplicativo Web son ineficientes para comprender que acción tomar e identificar los mensajes ACARS como control del flujo de datos. El resultado se refleja en la Figura 8.9.

```
+
*<SYN><SYN><SOH>b.N723?)N <NUL>=|L*<NUL>[V<SO>L=<SOH>;kRy<BS>~<LF><DC1>p|{<SYN><DC2>}<DC2>#&{<GS>6<STX>:<SOH><ETB>+
*<SYN><SYN><SOH>b.N74X~V3_<DEL>K<ETX>+
*<SYN><SYN><SOH>b.N748AV5_<DEL>M<ETX>+
*<SYN><SYN><SOH>B.N723>)j7NM};OO>>)OOOO\;9=>LMP><QFLMS0,1/C1XXXXXXS''''SRRRRRRS''''S'''W,&'_X/C201,770SMS0010,XXXXXX,(/'X $-X$/
C3XXXX,'WX_X,XXX<DC2>XX'<DC3>'''SB520JLPB4X$'SO00,RFI-/<ETX>+
*<SYN><SYN><SOH>b.N748AV7_<DEL>O<ETX>+
*<SYN><SYN><SOH>b.N748AV8_<DEL>P<ETX>+
*<SYN><SYN><SOH>B.N748AVQ_<DEL>9<STX>S35AAV9780<ETX>+
*<SYN><SYN><SOH>b.N723AV3_<DEL>$|Ab<NUL>7V<STX><SOH>KU1(<ENQ><VT><DC2>f]m5<NUL>X"<ENQ>8^-<SI>" <VT>l~Z<SO><BS>f"V?-<SUB>R"<CAN>tMQ
+<ACK>RMeUq<ENQ>:<US>AW<NAK>k+Z<STX><SYN>"'<4^ZgN\5bhE%7U2<CR><LF>z<ACK>ab<GS>J;U<DC1><oNEU>&#o<SI>/<DC4>M<VT>\^<DLE>:RI\<SYN>M,*#u
(<LF>)<BS>`<ENQ>dP 8<ENQ>5<EOT><US>@kEB<SI>])p<DC3><ENQ>>VD4{uL:<STX><DC1>R_<DC2>N<SYN>%4_;"QYX]euOS)J<SO>i!Zc4)Bf<SI>I<ENQ><DC4>UJZVm?HG=h@
(e<BS><SOH>o+<DLE>0<EM>|'Z2V*4=^<ACK>[k{,UZ<LF>[}0-*rI\a4rFE7I<LF>hSovy<ENQ>^e<DC2><RS>'a<RS>PK<DLE>z=h4p~`<TAB><US><ENQ>$iF?
Y<LF>"E<DC2>3<RS><LF>5<STX>y+.Q)v:o@Q#<GS>;h'5="W0d<ETB><FS><DC4>Y&S<NUL>*1<BEL>R*M}gUtC<GS>%H~_1U<ETX>+
*<SYN><SYN><SOH>B.N748AV<NAK>H11<STX>F23AAV9780#M1BRESPWI/AK?>55MSHN;I;=|Pn<NUL>m5U<SO>?<SOH>vwhhuu4#Jqru~j.j67<SI>TAk<BS>S'g<ACK>P)<DEL>s]%J%
+i<DC1><ETB>
```


Figura 8.8: Resultado de una mala decodificación del prototipo de recepción.

Live ACARS Messages

N°	Date	Mode	RGO	Acknowledgement	Label	Sequence	Seq Message	Flight	Message
1	2017-04-11 17:21:53	b	.N723?)		L>	=	UL>[	VL	=;kRy~p|{}#&{6:
2	2017-04-11 17:21:53	b	.N74X~V	_	L>	K			
3	2017-04-11 17:21:53	b	.N748AV	_	L>				

Figura 8.9: Visualización de los mensajes erráticos decodificados de la Figura 8.8 en la Tabla Dinámica.

Capítulo 9

Conclusiones

El protocolo ACARS está basado en un esquema MSK basado en una demodulación AM cuyo ranfo de frecuencia en la zona de Cundinamarca va desde los 130 Mhz hasta los 132Mhz. El funcionamiento de OOOI de ACARS en Colombia se evidencia en las pruebas realizadas en el Aeropuerto Internacional el Dorado de la ciudad de Bogotá cuyas actividades reportadas fueron la posición de la aeronave próximo a aterrizar, Numero de Gate o Sala de espera donde tenía que permanecer la aeronave, la confirmación o asignación del vuelo y la autorización de despeje. Las otras pruebas realizadas en la ciudad de Cajicá, Cundinamarca cuya ubicación es alejada de algún punto de control aéreo, reporta información de la ubicación geográfica y altitud de la aeronave, detalles técnicos y mecánicos del avión y el origen/destino del vuelo. Los dos lugares de prueba confirman que el funcionamiento de OOOI del protocolo ACARS clasifica el tipo de información que puede enviar una aeronave.

En términos teóricos la demodulación FSK se vuelve un esquema MSK siempre en cuando la diferencia de frecuencia de operación sea la mitad de la tasa de bits. La magnitud del vector distancia desde el origen y un símbolo en un diagrama IQ, siempre es contante pero variable en fase en más o menos 90 grados, cuya dependencia está dada por la disposición de los bits que componen un mensaje ACARS.

El desarrollo del prototipo de recepción ACARS evidencia que los sistemas de recepción prácticos son muy sensibles a los errores de frecuencia, tiempo y fase. Para mitigar el error de fase y de tiempo se utilizan métodos de sincronización como la detección por correlación empleado por el prototipo como mecanismo de sincronismo para recuperar los símbolos en la demodulación MSK. Los errores de frecuencia y de fase está muy relacionado por la recuperación de la portadora del Hardware de recepción utilizado. El ajuste de corrección de frecuencia PPM utilizado en el dispositivo de recepción RTL-SDR cumple con satisfacción la corrección de los errores de frecuencia y fase.

En un dispositivo RTL-SDR el corrimiento de frecuencia es directamente proporcional a la frecuencia RF o frecuencia de recepción. Es decir, la fluctuación frecuencial del cristal del RTL-SDR se vuelve inestable a medida que la frecuencia de recepción aumenta o disminuye.

Ese comportamiento permite una variación dinámica del corrimiento de frecuencia. Por ello es muy importante realizar el cálculo PPM para sistemas de recepción basados en RTL-SDR.

GNU-Radio es un sistema de procesamiento de señales flexible para implementar API's gracias a los bloques Python Block Embedded y Python Module. En base que GNU-RADIO funciona con programación Python, existen infinidad de API para desarrollar aplicaciones más complejas en GNU-Radio. Gracias a ello se logró desarrollar adaptar una API de MySQL para Python en GNU-Radio para poder insertar los datos del flujograma en la base de datos MySQL en tiempo real. La capacidad de mantener los mensajes en una base de datos, permite generar cualquier tipo de desarrollo de software como aplicaciones web (Utilizados en el prototipo del presente trabajo), desarrollo de aplicaciones móviles y desarrollo de aplicaciones en la nube.

El alcance de este proyecto no logra una simulación real de una infraestructura ACARS dada la limitación de la cobertura del dispositivo RTL-SDR. Para que efectivamente se logre un sistema ACARS como infraestructura alterna para casos de emergencia es necesario disponer una red de RTL-SDR conectados a una central. La central es la base de datos cuyos nodos están conectados a ella con la finalidad de clasificar los mensajes. La clasificación de los mensajes debe estar basado en el funcionamiento OOOI para que pueda abarcar grandes extensiones espacio manteniendo las actividades aéreas por más tiempo. La programación para recolectar y clasificar los mensajes provenientes de los sistemas de recepción debe lograrse con programación de lado del servidor. El desarrollo obtenido en la visualización de los resultados de la demodulación/Decodificación en el proyecto debe ser optimizado para que el contenido de interno de los mensajes ACARS sea un lenguaje explicito, legible y entendible para el ojo humano.

Bibliografía

[1] EASA (European Aviation Safety Agency). **Capitulo 15. Técnicas digitales / Sistemas de instrumentos electrónicos.** techreport, EASA, 2005. Modulo 5 de la EASA Parte 66. Modulos necesarios para la obtención de la Licencia de Mantenimiento de Aeronaves. 9

[2] Rockwell Collins. ARINC Aviation. **Aircraft Communications Addressing and Reporting System (ACARS)**. Rockwell Collins, 2551 Riva Road, Annapolis, MD 21401. Available from: https://www.rockwellcollins.com/Services_and_Support/Information_Management/~/media/DA843DB0792946C58740F613328E5022.ashx. 9, 10

[3] Aeronautica Civil. *AUTORIZACIÓN DE SALIDAS VIA DATA LINK EN EL AEROPUERTO ELDORADO DE BOGOTA*. Aeronautica Civil, Centro Nacional de Aeronavegación CNA Av. El Dorado No. 112-09 Bogotá D.C., November 2013. 15

[4] Unidad Administrativa Especial de Aeronáutica Civil U.A.E.A.C. *Plan de Navegación Aérea para Colombia*. Aeronáutica Civil Colombiana, Bogotá, volumen i: requerimientos operacionales edition, 2010. 14

[5] The Free & Open Software Radio Ecosystem. *GNU Radio Manual and C++ API Reference 3.7.10.1*. GNU-RADIO PROJECT, reference 3.7.10.1 edition. Available from: http://gnuradio.org/doc/doxygen/. 19

[6] Leonardo Gómez. **Vigilancia DepenDiente automática (aDS-b) en colombia**. *Ciencia y Poder Aéreo*, **10**(1):21–32, 2015. 14

[7] Jun Kitaori. **A performance comparison between VDL mode 2 and VHF ACARS by protocol simulator**. In *Digital Avionics Systems Conference, 2009. DASC'09. IEEE/AIAA 28th*, pages 4–B. IEEE, 2009. 5

[8] Jun Kitaori. **A performance comparison between VDL mode 2 and VHF ACARS by protocol simulator**. In *Digital Avionics Systems Conference, 2009. DASC'09. IEEE/AIAA 28th*, pages 4–B. IEEE, 2009. 10, 11

[9] Yun Lin, Chao Lv, Xiaochun Xu, and Bin Lin. **An Improved Method of Demodulation for Air-Ground Data Link Communication System**. *International Journal of Future Generation Communication and Networking*, **7**(2):1–8, 2014. 28

[10] Héctor Matamoros. **Estudio de CNS para Colombia**. *Francia: Ecole Nationale de l'a Aviation Civile ENAC*, 1999. 14

[11] Lionel K. Anderson Msc. *ACARS - A User's Guide*. Las Atalayas, first edition, 2010. 10

[12] Curtis Risley, James McMath, and B Payne. **Experimental encryption of aircraft communications addressing and reporting system (ACARS) aeronautical operational control (AOC) messages**. In *Digital Avionics Systems, 2001. DASC. 20th Conference*, **2, pages 7D4–1. IEEE, 2001.** 5

[13] ARINC Specification**. 618-5. *Air/ground character-oriented protocol specification*, 2000.** 11

[14] Cary R Spitzer and Cary Spitzer**. *Digital Avionics Handbook*. CRC press, 2000.** 8**,** 9

[15] Robert W Stewart, Kenneth W Barlee, Dale SW Atkinson, and Louise H Crockett**. *Software Defined Radio using MATLAB & Simulink and the RTL-SDR*. Strathclyde Academic Media, 2015.** 18**,** 21

[16] Shuguang Sun**. Acars data identification and application in aircraft maintenance. In *Database Technology and Applications, 2009 First International Workshop on*, pages 255–258. IEEE, 2009.** 4**,** 10**,** 11

[17] Thomas H Wright and James J Ziarno**. System and method of providing OOOI times of an aircraft, November 28 2000. US Patent 6,154,636.** 12

[18] Alexander M Wyglinski, Maziar Nekovee, and Y Thomas Hou**. Cognitive radio communications and networks. *IEEE Communications Magazine, Guest editorial*, 2008.** 17