UNIVERSIDAD TÉCNICA PARTICULAR DE LOJA

La Universidad Católica de Loja

ÁREA TÉCNICA

TITULACIÓN DE INGENIERO EN ELECTRÓNICA Y TELECOMUNICACIONES

Diseño e implementación de un Sistema de Conmutación de red para un sistema GSM basado en OpenBTS y Asterisk

TRABAJO DE FIN DE TITULACIÓN

AUTOR:
Tene Castillo, Juan Pablo

DIRECTOR:
Quiñones Cuenca, Manuel Fernando, Ing.

LOJA - ECUADOR

2013

2013

CERTIFICACIÓN

Ing.
Manuel Fernando Quiñones Cuenca,
DIRECTOR DEL TRABAJO DE FIN DE TITULACIÓN

CERTIFICA:

Que el presente trabajo, denominado: *Diseño e Implementación de un Sistema de Conmutación de red para un sistema GSM basado en OpenBTS y Asterisk*, realizado por el profesional en formación: *Tene Castillo, Juan Pablo*; cumple con los requisitos establecidos en las normas generales para la Graduación en la Universidad Técnica Particular de Loja, tanto en el aspecto de forma como de contenido, por lo cual me permito autorizar su presentación para los fines pertinentes.

Loja, Septiembre de 2013

F._____
Ing. Manuel Fernando Quiñones Cuenca

DECLARACIÓN DE AUTORÍA Y CESIÓN DE DERECHOS

Yo, *Juan Pablo Tene Castillo*; declaro ser autor del presente trabajo y eximo expresamente a la Universidad Técnica Particular de Loja y a sus representantes legales de posibles reclamos o acciones legales.

Adicionalmente declaro conocer y aceptar la disposición del Art. 67 del Estatuto Orgánico de la Universidad Técnica Particular de Loja, que en su parte pertinente textualmente dice: "Forman parte del patrimonio de la Universidad la propiedad intelectual de investigaciones, trabajos científicos o técnicos y tesis de grado que se realicen a través, o con el apoyo financiero, académico o institucional (operativo) de la Universidad".

F._____

Juan Pablo Tene Castillo

C.I.: 1104469406

DEDICATORIA

Dedicado a, Angelita y Juan.

AGRADECIMIENTO

Al ser a-temporal y a-espacial al cual pertenezco, en quien su voluntad confío, para encaminar mi vida hacia el bien.

A mis padres, por su apoyo incondicional y confianza. A mis hermanos, por alentar mi vida porque cada uno en particular ha ayudado a mantener siempre viva la llama de la esperanza.

A mi director de tesis: Ing. Manuel Quiñones, mi profesor y amigo, que siempre se mantiene alerta para el rescate de los jóvenes talentos que transitan por la titulación quien fue inmejorable guía para el desarrollo de este trabajo.

A todos quienes me esperaron.

Juan Pablo

Índice general

Índice de figuras

Índice de tablas

RESUMEN EJECUTIVO

En el presente trabajo de investigación, se emplean herramientas tanto de hardware como software de licencia libre con el objetivo de construir una estación base celular libre BTS, a bajo costo, y fácil implementación, iniciando por la indagación de los conceptos técnicos que rigen en los sistemas de telefonía móvil, así como en la tecnología de voz sobre ip VoIP. Para luego realizar una instalación del sistema denominado OpenBTS, el cual utiliza el hardware USRP, una radio definida por software SDR que permite desplegar una red análoga al estándar GSM, proporcionando el medio de comunicación con cualquier teléfono compatible con dicho estándar para que sea operado como extensión del protocolo de iniciación de sesiones SIP desde Asterisk, software que será utilizado como central de telefonía IP con varios módulos de configuración que integran un sistema de conmutación donde se ejecutan llamadas telefónicas entre los usuarios que ingresen a la red.

PALABRAS CLAVE: OpenBTS, SDR, software libre, GSM, USRP, VoIP, Asterisk.

ABSTRACT

In the present research, some open source tools of hardware and software are used in order to build a free cellular base station BTS, of low cost and easy implementation, starting with a investigation of the technical concepts about the mobile systems, and the voice over IP VoIP technology. Then is perform a installation of system called OpenBTS, which uses the USRP hardware a software defined radio SDR to deploy a free GSM network, providing the communication with any phone that supports this standard which is switching it as a user agent of the session Initiation protocol SIP from Asterisk computer program, this software is installed as a VoIP PBX system with several modules to integrate a switching system where telephone calls are executed between registered users.

KEYWORDS: OpenBTS, SDR, open source, GSM, USRP, VoIP, Asterisk.

INTRODUCCIÓN

La idea de construir una red celular abierta a partir de hardware y software libre, nace en vista de la necesidad de comunicación en ciertas comunidades que son aisladas por las empresas operadoras móviles al no completar el suficiente número de usuarios que generen rentabilidad económica; por otro lado también desarrollar un sistema que sirva como una red suplementaria en escenarios de emergencia donde se necesiten soluciones portables y de pronta instalación.

Para iniciar con el proyecto se emplea el sistema OpenBTS, que permite generar una interfaz de radiocomunicación por aire Um análogo al estándar de telefonía móvil Global System for Mobile Communications (GSM). Este utiliza como infraestructura de red el hardware USRP (Universal Software Radio Peripheral), que aprovecha la flexibilidad de la tecnología de Radio Definida por Software (SDR) para emular una Estación Base Transceiver BTS.

Debido a que el hardware USRP realiza la conversión de la señal analógica a digital mediante un procesamiento asistido por computador, los teléfonos móviles compatibles con el estándar GSM pueden ser manejados como extensiones del Protocolo de Iniciación de Sesión SIP desde Asterisk, un programa de código abierto que actúa como centro de conmutación de telefonía IP tanto para la red local de OpenBTS, como para realizar llamadas hacia operadoras móviles privadas.

En el primer capítulo se argumenta la divergencia existente entre la disponibilidad de dispositivos móviles con el servicio brindado a sectores desprotegidos, además son establecidas las fases principales de ejecución del proyecto. En el segundo capítulo son recogidos los conceptos técnicos que describen el funcionamiento de los sistemas de comunicación móvil, así como los involucrados en la tecnología de (VoIP). El tercer capítulo integra los conceptos adquiridos junto con los requerimientos necesarios para instalar la red celular, cuya configuración y verificación se detalla en el cuarto y quinto capítulo donde se realizan las pruebas necesarias para comprobar la funcionabilidad del sistema.

La base del hardware usado en el trabajo, es el USRP N210, que posee un procesador FPGA Xilinx Spartan 3A-DSP 3400 con una tarjeta de transmisión y recepción de tipo WBX, dos antenas VERT900 para conectarlas a cada puerto de la tarjeta, que cubren las bandas GSM 850/900 MHz. Además, se usa un computador personal de mediana capacidad, el cual debe tener instalado una tarjeta de red Gigabit-Ethernet para la programación y comunicación con el USRP.

1

ALCANCE DE LA INVESTIGACIÓN

1.1. Objetivos

Objetivo general

Diseñar e implementar un sistema de conmutación de llamadas para una red GSM usando Hardware y Software Libre.

Objetivos específicos

- Montar una red celular GSM de bajo costo y libre acceso usando el sistema OpenBTS.
- Instalar y configurar el programa de conmutación Asterisk, que posibilite la ejecución de llamadas telefónicas dentro y fuera de la red.
- Usar sistemas de conexión alternativos para que funcionen como puertas de enlace de la central montada por Asterisk.
- Realizar la comprobación de los servicios que ofrece la red abierta de telefonía móvil.

1.2. Justificación

Durante el primer cuarto del siglo XXI, la integración de las telecomunicaciones con la sociedad actual, se desenvuelve en conjunto con el incremento de los miles de usuarios que demandan diariamente servicios de comunicación móvil, de voz, datos y vídeo. Este incesante flujo de información ha pasado de ser una necesidad a un dilema social, tal como lo han mencionado algunos estudios científicos como por ejemplo los realizados por la Universidad Politécnica de Madrid (UPM) [9].
El estándar celular GSM se encuentra extendido en más de un 90 % alrededor de todo el mundo, contando con más de 3000 millones de usuarios en 219 países [22]. Por otro lado en Ecuador, existen más de 16 millones de dispositivos de comunicaciones móviles activados, de los cuales 14,5 millones son teléfonos [12], una cifra que sigue en aumento y supera incluso a la población del país que se acerca a 14,3 millones de habitantes [11].
Estas cifras pueden suponer hoy en día, que cualquier ciudadano tiene acceso a una red de comunicación por voz, pero la gran verdad es que todavía existen personas y comunidades a las que no llega ningún tipo de tecnología de comunicación, por tratarse de lugares poco rentables para las grandes empresas de telecomunicaciones que requieren de un cierto número de clientes para iniciar su operación.

En este trabajo, se propone usar equipos de bajo costo junto con la integración de software de licencia libre, con el fin de emular una plataforma de red celular GSM (Global System Mobile)[1] que permita brindar un acceso libre a sitios que carecen de cobertura celular, sin implantar una tarifa económica por cada llamada o mensaje realizado.

1.3. Metodología

Durante el desarrollo del proyecto de tesis, la metodología de investigación aplicada es un sistema de avance por fases, de esta forma se mantuvo un constante enfoque exploratorio de libros, documentos técnicos, artículos y publicaciones científicas, con el fin de extraer las ideas principales de sistemas de telefonía móvil de código abierto, y aplicarlos junto con un inherente componente heurístico, a fin de alcanzar los objetivos planteados dentro del trabajo de investigación.
Se inicio analizando la arquitectura que debe seguir cualquier operador de comunicaciones móviles en el caso de utilizar el estándar GSM para montar una red celular, presentando una breve descripción del rol que cumplen las partes más importantes del sistema de comunicación móvil. También, es mencionado el proceso que siguen las centrales telefónicas basadas en VoIP (Voz sobre IP)[2], enfocando la funcionalidad que presta el software de licencia libre Asterisk, gracias a la integración de sus componentes con las redes de comunicación.
Así mismo, para la comprensión de la transcendencia del tema, se expone el papel que juega la tecnología SDR en el ámbito de las telecomunicaciones modernas, estudiando su arquitectura, para terminar por comprender el procesamiento que ejecuta el dispositivo Universal Software Radio Peripheral (USRP) [3], con base a su utilización en la infraestructura de la red abierta GSM.
A partir de las etapas anteriores, se inicia el proceso de configuración y montaje de la red, tomando como guía el planteamiento del sistema por fases que se describe en la figura 1.1.

La primera fase, comprende el sistema de red GSM basado en OpenBTS [4], se analizan sus partes de forma que sea posible realizar una instalación y configuración exitosa, todo como parte de la infraestructura necesaria para implantar un sistema celular de aspecto portable.

[1] http://www.gsma.com/aboutus/gsm-technology/gsm
[2] http://es.wikipedia.org/wiki/Voz_sobre_Protocolo_de_Internet
[3] http://www.ettus.com/
[4] http://openbts.sourceforge.net/

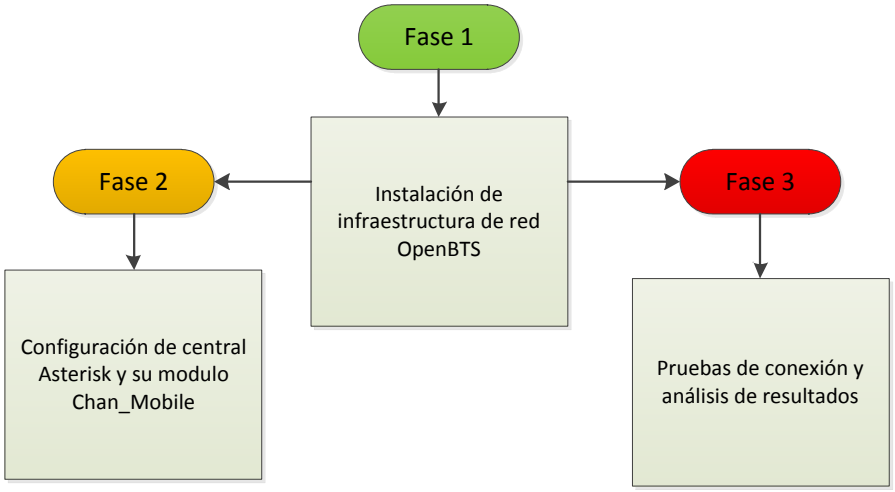

Figura 1.1: Fases de instalación y análisis

La segunda fase, comprende la instalación de una central telefónica que permita realizar la gestión y conmutación de llamadas entre estaciones móviles (MS), dentro y fuera de la propia red, por medio del plan de marcado y las aplicaciones de conexión con que cuente el software usado, Asterisk.
Por ultimo se proyecta realizar las pruebas de cada módulo del sistema, corrigiendo a su vez los errores que se presenten en cada etapa de la implementación de la red y su central de conmutación, estas pruebas servirán como guía para el análisis de los resultados que se obtengan al finalizar el proyecto.

2

ESTADO DEL ARTE

2.1. Introducción

El estándar GSM es una tecnología muy conocida al rededor del mundo actualmente usada por el 90 % de las operadoras móviles, además ha servido como base para el desarrollo de nuevos tipos de red celular de libre acceso usando herramientas open-source. La mayoría de estos proyectos son librerías de software, herramientas de hardware, y componentes de gestión de red desarrollados por profesionales y aficionados a las telecomunicaciones. Pero además, existen empresas como OSCOM[1] o Range Networks [2] que fabrican una plataforma tanto de hardware como de software para establecer toda una red celular que pueda ser configurada a medida de las necesidades del usuario.

OpenBTS desarrollado por Range Networks, implementa una red GSM con funcionalidades básicas de telefonía móvil, que se apoya de la flexibilidad que ofrece la tecnología de Voz sobre IP (VoIP) y la ubicuidad de la interfaz de aire GSM, estableciendo un servicio de comunicación entre los usuarios totalmente independiente de una operadora convencional.

En GSM, como se explicará más adelante, las llamadas desde y hacia los abonados móviles son conmutadas por la central de conmutación móvil, la cual provee las funciones de señalización y control necesarias para realizar con éxito dichas llamadas, su ubicación es independiente de la ubicación de la radiobase.

Por otro lado, el sistema OpenBTS no incluye un módulo de conmutación, razón por la cual sus creadores sugieren el uso de software gratuito Asterisk, que brinda la facilidad de emular una central telefónica de VoIP y así mismo sus funciones pueden ser fácilmente adaptables para integrar un sistema que permita conectar llamadas locales y con otras redes.

Durante este capítulo se describen los conceptos técnicos de los componentes que se involucran en una red de telefonía celular GSM, además de los parámetros que definen las comunicaciones de VoIP. También es discutida la relación existente entre las radios definidas por software SDR y los sistemas de telefonía móvil.

[1] `http://bb.osmocom.org`

[2] `http://rangenetworks.com/`

2.2. Sistema Global de Comunicaciones Móviles GSM

El organismo responsable por la estandarización del sistema de telefonía móvil GSM es el "European Telecommunications Standars Institute"(ETSI) [1], este se encarga de emitir especificaciones técnicas para fabricantes de equipos y operadores de red en la industria de las telecomunicaciones. ETSI ha divido en series todo el estándar GSM, estas series tratan de temas específicos sobre el funcionamiento de GSM, y se encuentran distribuidos tal como se muestra en la tabla 2.1.

La interfaz de aire que se la conoce como Um, describe los métodos usados por la

Serie	Tema
01.xx	Aspectos Generales
02.xx	Normas de servicio
03.xx	Características de red
04.xx	Interfaz y protocolos entre (MS - BS)
05.xx	Capa física de red (interfaz de radio)
06.xx	Codificación de la voz
07.xx	Adaptadores de terminales MS
08.xx	Interfaces BS-MSC
09.xx	Interfuncionamiento de redes
10.xx	Interfuncionamiento de servicios
11.xx	Especificaciones y homologación
12.xx	Operación y mantenimiento

Tabla 2.1: Series de especificaciones GSM. Modificado de [14]

estación móvil MS y la estación base BTS para establecer una comunicación bidireccional fullduplex. Um se encuentra definida en las series GSM 04.xx y 05.xx del estándar, donde se definen las frecuencias de operación más comunes que son 850 MHz, 900 MHz, 1800 MHz, y 1900 MHz.

GSM usa una combinación de FDMA y TDMA en un espectro total de 25 MHz. El multiplexado se realiza con Frequency Division Multiple Access (FDMA) [2] dividiendo esos 25 MHz en 125 canales de 200 kHz cada uno, y cada canal de 200 kHz a su vez es dividido en 8 ranuras de tiempo utilizando Time Division Multiple Access (TDMA) [2]. Además usando la técnica de Frequency Division Duplex (FDD) [3] se divide en dos la banda GSM, definiendo una banda de frecuencia ascendente, denominada "uplink" y otra banda de frecuencia descendente llamada "downlink", ambas son usadas por las estaciones móviles para su comunicación.

En la tabla 2.2 se enumeran las bandas de frecuencia downlink y uplink para GSM

[1] `http://www.etsi.org/`

[2] `http://en.wikipedia.org/wiki/Frequency-division_multiple_access`

[3] `http://en.wikipedia.org/wiki/Frequency_division_duplexing`

	GSM-850	GSM-900	GSM-1800	GSM-1900
Rango de Frecuencia *uplink*	824 - 849 (MHz)	890 - 915 (MHz)	1710 - 1785 (MHz)	1850 - 1910 (MHz)
Rango de Frecuencia *downlink*	869 - 894 (MHz)	935 - 960 (MHz)	1805 - 1880 (MHz)	1930 - 1990 (MHz)
ARFCN	128 - 251	1 - 124	512 - 885	512 - 810
Offset	45 MHz	45 MHz	95 MHz	80 MHz

Tabla 2.2: Bandas de Frecuencia GSM y Números de Canal ARFCN

recomendadas por la ETSI, además, son usadas en los países para designarlas a las operadoras que deseen brindar el servicio móvil.
Los canales de uplink y dowlink son especificados por un número llamado Absolute Radio Frecuency Channel Number (ARFCN) que cumple con la función de asignar los canales de transmisión y recepción que usaran los teléfonos en la red. Cada ARFCN tiene un ancho de banda de 270,833 Kbps y utilizan una separación entre canales de 200 kHz en cualquier banda de GSM.

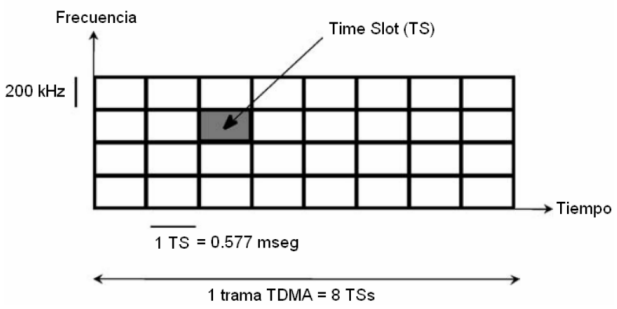

Figura 2.1: Esquema de acceso al medio en GSM. Tomado de [14]

Gracias al uso de TDMA[1] los usuarios en determinada área pueden compartir un simple canal de radio en diferentes intervalos de tiempo. Estos intervalos son denominados time slots (TS) con una duración de 0.577 ms. En la Figura 2.1 podemos ver un esquema en el que se divide la banda en función del tiempo y la frecuencia, donde el canal físico es asignado en un preciso ARFCN y un exacto time slot.

Arquitectura de GSM

La arquitectura de red definida por el estándar GSM se compone por tres subsistemas principales como se indica en el siguiente gráfico. La Estación Móvil (MS), el Subsistema de Estaciones Base (BSS), y el Subsistema de Red (NSS).

[1] `http://en.wikipedia.org/wiki/Time_division_multiple_access`

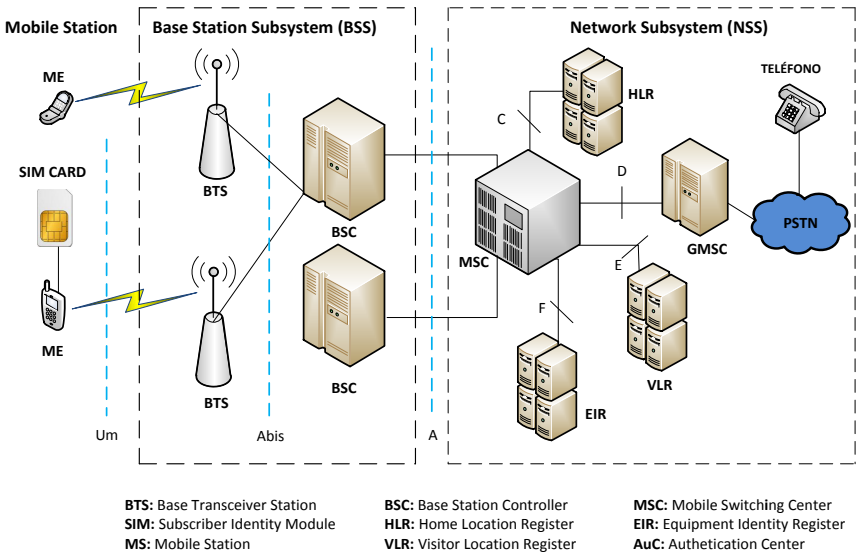

Figura 2.2: Subsistemas de una Red GSM. Modificado de [14]

Mobile Station MS

Se trata del teléfono móvil que esta provisto de una tarjeta de identificación de abonado Suscriber Identity Module (SIM)[1], esta permite identificar de manera única al usuario que se comunica con la red a través de la interfaz Um.

Figura 2.3: Estación Móvil. Tomado de [23]

La tarjeta SIM posee una clave de autenticación (Ki, Authentication key) asignada por el operador de red para validar el acceso único, se encuentra almacenada en la base de datos del Centro de Autenticación (AuC). Mientras que los terminales móviles se identifican por medio de un número denominado Identificador Internacional de Equipos Móviles (IMEI).

Dentro de la tarjeta SIM se almacena un número secreto de identificación denomi-

[1] `http://en.wikipedia.org/wiki/Subscriber_identity_module`

nado Identificador Internacional de Abonados Móviles (IMSI) como se indica en la tabla 2.3. Está constituido por, el MCC (Código del País), el MNC (Mobile Network Code) Código de Red y el MSIN (Número de Identificación del Suscriptor). El MCC

IMSI:740995295250378 6		
MCC	740	ECUADOR
MNC	99	CLARO
MSIN	52952503786	

Tabla 2.3: Estructura del número IMSI

identifica el país al cual pertenece la MS, el MNC indica el operador móvil al que esta registrado y el MSIN es un número único que identifica al abonado dentro de la red GSM [23].

Base Station Subsystem BSS

El sistema de estaciones base se encarga del control de la radiocomunicación con las estaciones móviles. Facilita la conexión con el subsistema de red (NSS, Network Subsystem), y con usuarios de la red PSTN [1]. Está compuesto de dos unidades:

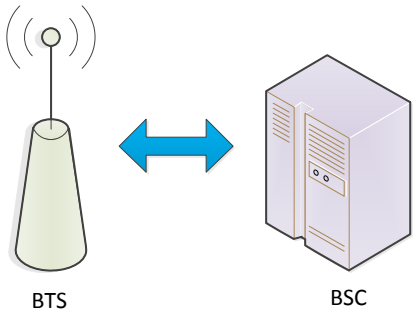

Figura 2.4: BSS Sistema de Estaciones Base.

- **Estación Base de transmisión (BTS, Base Transceiver Station):** Se compone de antenas y transceivers con las que gestiona la radiocomunicación de las estaciones móviles. Su potencia de transmisión determina el tamaño de la celda y proporciona un número de canales de radio a la zona a la que da servicio [1]. La MS puede distinguir las celdas por su BSIC (Base Station Identity Code), y su Cell ID (identificación de celda).

[1] `http://en.wikipedia.org/wiki/Public_switched_telephone_network`

- **Controlador de la estación (BSC, Base Station Controller):** Gestiona los recursos de radio de una o varias estaciones base (BTS). Se encarga del mantenimiento de la llamada de una celda a otra, mientras el equipo móvil se encuentra en movimiento lo que se conoce como "handover"[15].

Network Switching Subsystem NSS

Es responsable del establecimiento y terminación de cualquier llamada, así como la recopilación de información necesaria para el proceso de tarificación. El componente principal del subsistema de conmutación de la red (NSS), es el centro de conmutación de servicios móviles Mobile Services Switching Center (MSC), considerado el núcleo del sistema GSM.

En general el MSC es el responsable de gestionar una comunicación confiable entre

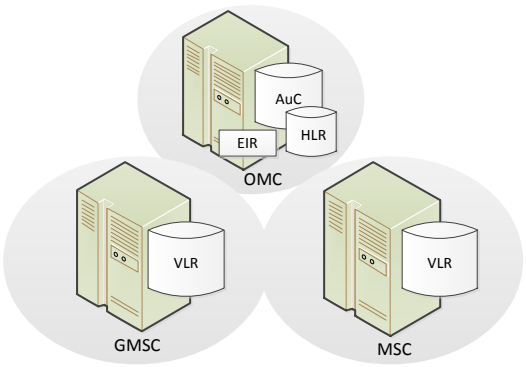

Figura 2.5: Subsistema de Conmutación de Red NSS. Modificado de [23]

la red GSM y las otras redes de telecomunicaciones ya sean móviles o fijas como la PSTN [23]. Dentro de la estructura del NSS hay una serie de subsistemas que se encargan de controlar diversas funciones del equipo móvil, estos son:

- **Registro de localización local (HLR, Home Location Register):** El HLR contiene información que describe los servicios contratados por el abonado y aquellas opciones a las que tiene acceso. Se almacena además la última localización conocida del abonado y el estado de su terminal móvil (si está fuera de servicio, encendido, apagado, en comunicación).

- **Registro de localización de visitante (VLR, Visitor Location Register):**Contiene información del estado de todos los usuarios que provienen de

otras operadoras que necesiten acceso a la red GSM (roaming)[1]. Además conoce si el usuario está o no activo, a efectos de evitar retardos y consumo de recursos innecesarios cuando la estación móvil está apagada.

- **Centro de Autenticación (AuC, Authentication Center):** Base de datos que almacena información confidencial de cada abonado de la red. Contiene claves de seguridad que la red utiliza para verificar la identidad del usuario y provee seguridad a los operadores de probables fraudes.
- **Registro de Identidad del Equipo (EIR, Equipment Identity Register):** Es una base de datos que puede deshabilitar el acceso a la red a teléfonos defectuosos o clonados.

Canales de transmisión

En GSM cada canal físico de radio para transmisión por el aire puede contener dos tipos de canales lógicos diferenciados por su funcionalidad: los de trafico (TCH: TrafficChannel) y los de control (CCH: control Channel).

Tipo de Canal	Denominación	Descripción
Canales de tráfico TCH	TCH/FS	Voz a velocidad de 13 Kbps
	TCH/F9.6	Datos a 9600 bps
	TCH/F4.8	Datos a 4800 bps
	TCH/F2.4	Datos a 2400 bps
	TCH/HS	Voz a 7 Kbps
	TCH/H4.8	Datos a 4800 bps
	TCH/H2.4	Datos a 2400 bps

Tabla 2.4: Canales de Tráfico en GSM

Los canales de tráfico contienen información de voz y datos existente en un canal de comunicación. En la tabla 2.4 se nombra los tipos de canales en función a la tasa de transmisión que soporta el sistema.

Por otra parte, los canales de control son utilizados para enviar y recibir información de señalización de los diferentes procesos entre los equipos. Existen varios tipos de canales de control algunos comunes a todos los móviles, y otros dedicados a un terminal específico, entre los que se incluyen los de la tabla 2.5.

[1] `http://en.wikipedia.org/wiki/Roaming`

Tipo de Canal	Denominación		Descripción
Canales de Control (CCH)	Canales de Broadcast	BCCH (Broadcast Control Channel)	Canales de control utilizados para permitir el enganche de estaciones móviles a la red y el monitoreo de las potencias de celdas vecinas.
		FCCH (Frequency Correction Channel)	Envian información del sistema usando una referencia de frecuencia y otra de tiempo.
		SCH (Syncronization Channel)	
	Canal Común de Control	PCH (Pagging Channel)	Canales usados para reservar y asignar los recursos de radio y el acceso a canales de control.
		RACH (Random Access Channel)	
		AGCH (Access Grant Channel)	
	Canales de Control Dedicados	SDCCH (Stand-Alone Control Channel)	Canales de control bidireccionales utilizados para prestar servicios de señalización y autenticación de usuario.
		SACCH (SlowAssociates Control Channel)	
		FACCH (Fast Associated Control Channel)	

Tabla 2.5: Canales de Control en GSM

2.3. Voz Sobre IP (VoIP)

Es la transmisión de datos de voz en tiempo real, sobre redes basadas en el protocolo de Internet IP [25]. Posibilita la convergencia de varias redes con soporte de IP, la figura 2.6 muestra un diagrama del alcance al que puede llegar la tecnología de VoIP al integrar varios medios de comunicación.
Este sistema permite establecer un nuevo concepto de comunicación llamado telefonía IP. En donde, las líneas telefónicas convencionales de la red (PSTN) entrantes, pueden ser convertidas a VoIP, a través de una pasarela (Gateway) que permite recibir y hacer llamadas en la red telefónica normal. Los elementos fundamentales de una red VoIP son:

- *Terminales:* Teléfonos IP que pueden ser hardware o software.
- *Central de conmutación:* Controla y gestiona los procesos de toda la comunicación de VoIP.
- *Gateway:* Dispositivo que sirve de enlace con otras redes telefónicas. Actúa de forma transparente al usuario.

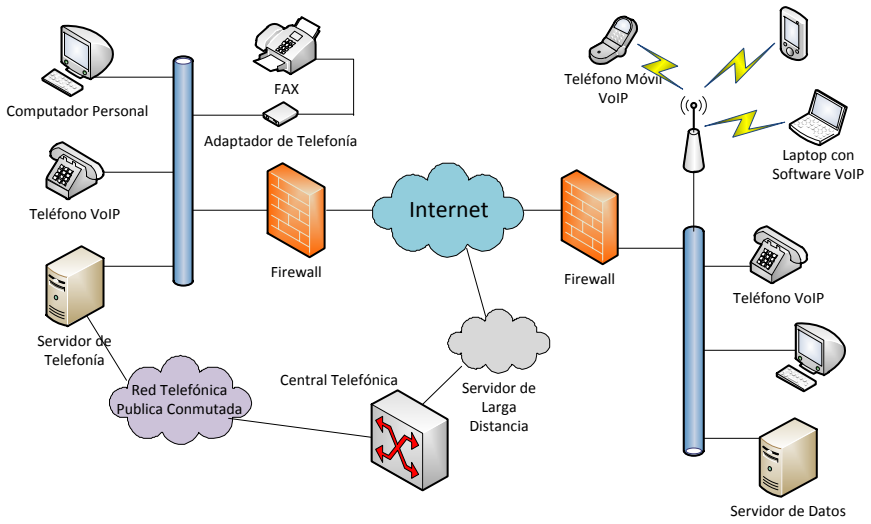

Figura 2.6: Diagrama de Red VoIP. Modificado de [24]

PBX

Se trata de un tipo de central que no depende directamente del proveedor de telefonía convencional. La central telefónica privada PBX posibilita el manejo de las llamadas que se realicen dentro de un edificio o área de trabajo local, realizando la conmutación, control y señalización de manera autónoma, además utiliza líneas externas para conectarse con la red pública de telefonía (PSTN) [1]. Las centrales que utilizan el protocolo IP (Internet Protocol) para transmitir información de voz se han sintetizado en programas de software con capacidad de telefonía IP, como por ejemplo Asterisk [37].

Durante el establecimiento de una conversación se necesita de un conjunto de reglas que permitan establecer, mantener, administrar y finalizar los elementos de red involucrados. Estas reglas se denominan protocolos y son directamente responsables del funcionamiento de la comunicación, uno en particular es el Protocolo de Iniciación de Sesiones (SIP) el cual ha sido parte integral dentro de este proyecto.

Protocolo de VoIP

SIP

Fue desarrollado por el IETF. Define los parámetros de señalización para crear, modificar y terminar sesiones con uno o más participantes. Estas sesiones incluyen llamadas telefónicas por Internet, distribución de datos multimedia, y conferencias multimedia [25].

SIP ofrece la posibilidad de programar nuevos servicios no definidos por la propia

recomendación. Esta ventaja de su flexibilidad es por lo que actualmente es mas usado que otros protocolos. Algunas funciones de señalización son:

- Establecer, modificar y finalizar llamadas.
- Registrar y localizar participantes.
- Gestión del conjunto de participantes y componentes del sistema.

Los clientes SIP llamados *peers* o agentes de usuario usan los protocolos TCP (Protocolo de Control Transmisión) y UDP (Protocolo de Datagramas de usuario) para conectar con los servidores SIP. SIP es usado simplemente para iniciar y terminar sesiones multimedia. Todas las comunicaciones en SIP usan el Protocolo de Transporte en Tiempo Real (RTP)[1].

Códecs de Audio

Para poder transmitir la voz sobre una red IP, necesitamos transformar una señal analógica en digital y viceversa este proceso se conoce como codificación y decodificación. El software/hardware que realiza este proceso se denomina códec, y dependiendo de cual utilicemos variará la calidad de voz, el ancho de banda ocupado y la carga computacional, en la tabla 2.6 podemos ver una comparación de los códecs más utilizados en VoIP.

Códec	**Ancho de Banda (Kbps)**	**Retardo (ms)**	**Estándar**	**Descripción**
G.711	64	20 - 30	ITU-T	PCM
G.726	16/24/32/40	20 - 30	ITU-T	ADPCM
G.729	8	15	ITU-T	Combina código de predicción de linea y estructura algebraica.
GSM	13	20	ETSI	Predicción de excitación con pulso regular a largo plazo

Tabla 2.6: Tabla de comparación de Códecs. Modificado de [26]

G.711

En redes locales LAN, es el códec que más se utiliza. La calidad de audio es óptima y el consumo es moderado. Proporciona un flujo de datos de 64 Kbps [26]. Existen dos modelos:

[1] `http://es.wikipedia.org/wiki/Real-time_Transport_Protocol`

- **ulaw**: codifica cada 14 muestras en palabras de 8 bits. Usado en EE.UU. y Japón.
- **Alaw**: codifica cada 13 muestras en palabras de 8 bits. Usado en el resto del mundo.

Se toman muestras a una frecuencia de 8 kHz y utiliza PCM (Pulse Code Modulation) para comprimir, descomprimir, codificar y decodificar.

Códec GSM

Este códec es muy reconocido dado que se utiliza regularmente en los canales de líneas móviles. Realmente no prima en la calidad sino en cantidad, un ejemplo claro es el flujo de datos en una conexión Full-Rate que tan solo alcanza los 13 kbps, esta cantidad es la que se utiliza en el sistema de red OpenBTS [35].
Por otro lado, representa un buen método para ahorrar ancho de banda puesto que consume cinco 5 veces menos que el códec G.711, aunque si trabajamos en conexiones de mala calidad a nivel de tiempo de respuesta (tiempo que tarda en ir y volver un paquete enviado) o de jitter (desviación estándar del tiempo de respuesta), la voz podría llegar a distorsionarse tanto hasta ser incomprensible.

2.4. SDR como plataforma GSM

La tecnología que utilizan los sistemas de telefonía móvil ha migrado de analógica a digital con el propósito de ofrecer flexibilidad, eficiencia, e intuición en los servicios que los usuarios esperan de sus portadoras. Las radiocomunicaciones definidas por software, son parte de la tecnología digital que hace posible implementar una interfaz de aire GSM, de esta forma, estaciones base BTS y terminales MS, pueden interactuar a nivel de hardware y software logrando por ejemplo evadir múltiples interferencias de aire durante los periodos de transición de su continua comunicación.

Definición y Aspectos Fundamentales de SDR

Los desarrollos en radios inteligentes y adaptativos se han enmarcado a lo que hoy es un Radio Definido por Software, el cual es definido, según el Wireless Inovation Forum[1], de la siguiente manera:
"Radio en el cual algunas o todas las funciones de la capa física son definidas mediante software".

[1] `http://www.wirelessinnovation.org/`

SDR es una tecnología creada para mejorar la interoperabilidad entre diferentes servicios; está compuesta de software y hardware, y puede ser reconfigurada dinámicamente para habilitar comunicaciones entre una amplia variedad de normas de comunicaciones y protocolos [5]. SDR permite crear dispositivos inalámbricos, equipos de redes multibanda y multifuncionales, que pueden ser dinámicamente reconfigurados, a través de actualizaciones de software y reconfiguraciones de hardware.

Arquitectura de Hardware SDR

La arquitectura de transmisores basados en software consiste en un subsistema digital y un subsistema analógico. Las funciones analógicas son restringidas a aquellas que no pueden ser mejoradas digitalmente, que son: antena, filtrado RF, combinación RF, preamplificación en recepción, transmisión de potencia de amplificación y generación de frecuencia de referencia [3]. A continuación se enumeran las partes que conforman un transceiver de radio basado en software:

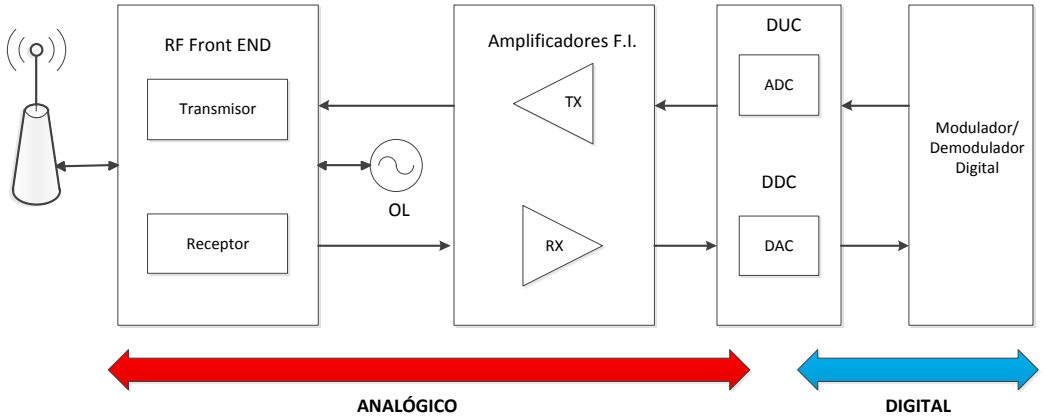

Figura 2.7: Arquitectura hardware de SDR

Antena

La transmisión de la información se realiza mediante ondas electromagnéticas que son transmitidas al aire. El elemento que radia estas ondas se denomina antena, puede considerarse como un transductor y un adaptador de impedancia al medio de transmisión.

Front End de RF

Este bloque cuenta con dispositivos electrónicos de estado sólido que adaptan el nivel de las señales de entrada para que sea adecuado en las siguientes etapas del

SDR. Por ejemplo en el caso de la telefonía celular GSM las señales se procesan a la frecuencia de transmisión/recepción del orden de 1 GHz en la banda de 850 MHz [4].
En el transmisor, se produce una amplificación de la señal entregada por las etapas de procesamiento hasta el nivel de potencia suficiente para su transmisión por el medio físico.

Oscilador Local

Genera las frecuencias apropiadas para convertir la frecuencia de RF en la frecuencia intermedia FI, mediante una mezcla no lineal que produce frecuencias de suma y resta. Se selecciona la frecuencia deseada mediante filtros analógicos para su amplificación en los amplificadores de frecuencia intermedia.

Bloque de Frecuencia Intermedia

En este bloque de FI se realiza la selectividad y ganancia del receptor, la FI siempre tiene menor frecuencia que la RF debido a que es mas fácil y menos costoso fabricar amplificadores estables para señales de baja frecuencia [2]. Por razones similares, también se procesa la señal para la transmisión a una frecuencia inferior para luego convertirla al valor final y amplificarla hasta el nivel permitido en la antena.

Conversión AD/DA

Tomando en cuenta que la transmisión por el medio físico se realiza mediante señales analógicas, pero el procesamiento en el Transceiver es de índole digital, se hace imprescindible realizar una conversión analógica/digital en el receptor y digital/analógica en el transmisor. A continuación se describen las partes más importantes de este bloque:

- **ADC:** El convertidor analógico digital (ADC) es un dispositivo que es capaz de ofrecer un valor binario de salida a partir de una entrada analógica de voltaje. Esta definición involucra los siguientes procesos:
  - Muestreo: Consiste en tomar muestras periódicas de la amplitud de la señal analógica. La velocidad en que se toman las muestras se llama frecuencia de muestreo.
  - Cuantificación: Mide el nivel de voltaje de cada muestra y le asigna un valor numérico de salida. Cuando no coincide el valor de salida con el de entrada, se dice que existe ruido de cuantificación.

- Codificación: La codificación consiste en traducir los valores obtenidos durante la cuantificación en código binario.

- **DDC:** El conversor Digital Down Converter se encarga de convertir una señal digital de FI en una señal de banda base. La figura 2.8 ilustra su composición:

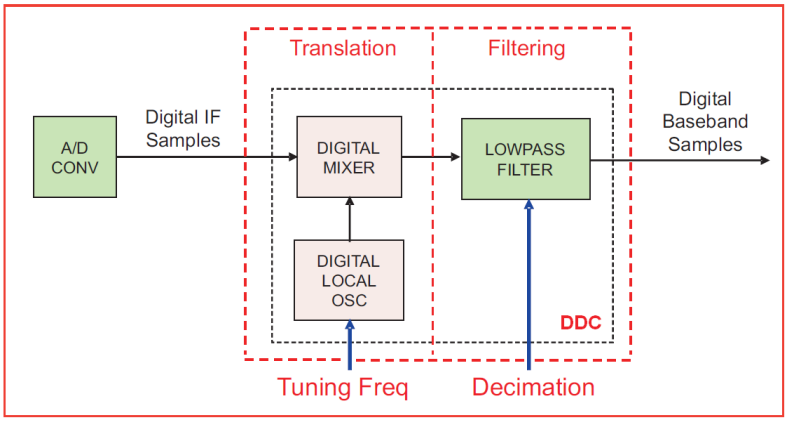

Figura 2.8: Digital Down Converter. Tomado de [5]

El DDC se compone de un mezclador digital, un oscilador local digital y un filtro digital pasabajos. El mezclador y el oscilador trasladan las muestras digitales de FI en banda base. El filtro limita el ancho de banda de la señal realizando la función de decimación de muestras a un rango menor de muestreo.

- **DUC:** Digital Up converter es un conversor que traslada la señal de banda base en frecuencia digital intermedia IF. Luego, es transformada en FI analógica por el convertidor digital analógico (DAC) y esta señal es a su vez convertida en señal RF por el transmisor. El DUC se compone como muestra la figura 2.9.
  El mezclador y el oscilador local trasladan las muestras de banda base a frecuencia FI. El filtro de interpolación resuelve la diferencia entre la frecuencia de muestreo del oscilador y la frecuencia de entrada de la señal en banda base.

Modulador/Demodulador

Para que pueda transmitirse una información útil mediante una onda electromagnética que se propaga, es necesario imprimir de algún modo esta información sobre una señal portadora. Esto se logra modificando alguno de los parámetros que la definen de acuerdo con el valor de la información a transmitir. Este proceso se

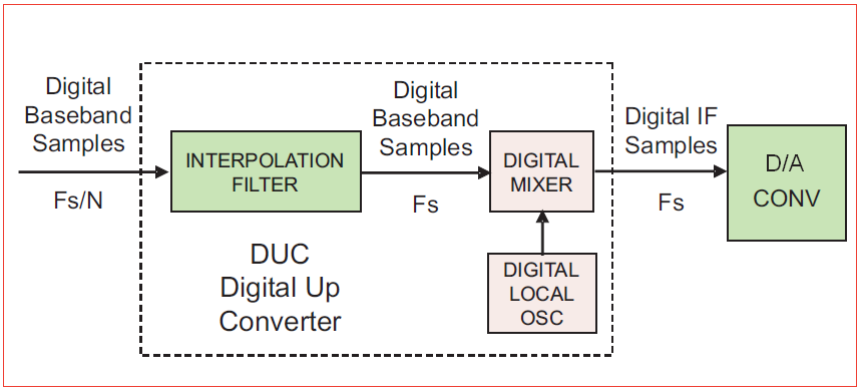

Figura 2.9: Digital Up Converter. Tomado de [5]

denomina modulación y el proceso inverso para recuperar la información es la demodulación, es por eso que algunos llaman ha este bloque "MODEM".
Estas funciones son actualmente muy complejas y totalmente digitales. Generalmente son realizadas por Procesadores de Propósito General GPP, pero para que las tareas de Modulación/Demodulación puedan ser fácilmente programables se utilizan procesadores como DSPs (Digital Signal Processors) o FPGAs (Field Programmable Gate Arrays). Los parámetros que se modifican para que la onda transmita información útil son típicamente la frecuencia o la fase/amplitud de la señal, utilizando modulaciones de múltiples niveles denominadas “en cuadratura” [2] mQAM, mPSK, etc.
GSM utiliza la modulación digital 0.3GMSK (Gaussian Minimun Shift Keying). Es un tipo especial de modulación donde 0.3G describe la banda de Filtro Gaussiano de premodulación usado para reducir el espectro de la señal modulada. MSK (Minimun Shift Keying) se deriva de la modulación digital en frecuencia FSK (Frecuency Shift Keying). Cuando la tasa de bits de la señal moduladora es exactamente cuatro veces la traslación de la frecuencia de la portadora se consigue minimizar el espectro y la modulación es llamada MSK (Minimun Shift Keying) [8].

3

MATERIALES Y MÉTODOS

3.1. OpenBTS

El sistema OpenBTS, es definido como una aplicación de software libre desarrollado bajo el sistema operativo multitarea Unix y que utiliza el Hardware USRP (Universal Software Radio Peripheral) para construir la interfaz inalámbrica de radio Um emulando al estándar de comunicaciones móviles GSM. Lo que posibilita que los teléfonos celulares circundantes detecten una completa red GSM, y a su vez, estos sean vistos como extensiones del protocolo SIP , permitiendo montar *in-situ* un sistema de conmutación o central telefónica gracias al software de licencia libre Asterisk [33].

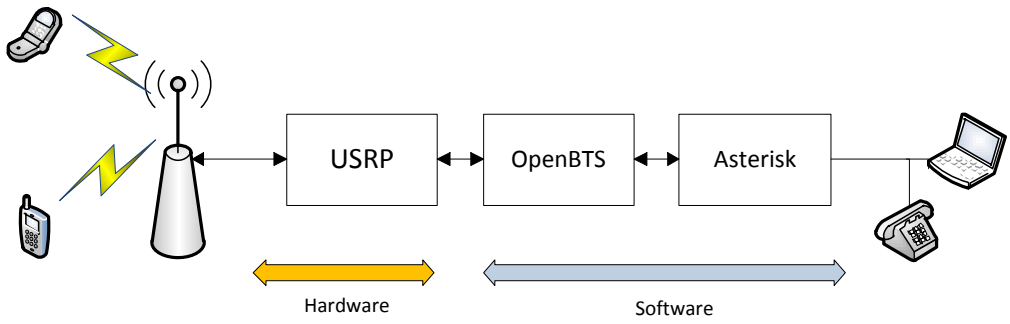

Figura 3.1: Sistema OpenBTS. Tomado de [35]

La Figura 3.1 muestra en bloques los componentes del sistema OpenBTS. En primera instancia la Antena y el USRP conforman la sección hardware de la BTS, luego están Asterisk y la aplicación OpenBTS constituyendo el software que se instala en un computador formando la otra parte del proyecto. A lo largo de este capítulo se explicará el funcionamiento de estas secciones, junto con la configuración básica de cada módulo que cumple con una función específica en el sistema.
OpenBTS forma la base de un nuevo tipo de red celular que puede ser desarrollada y operada a un costo más bajo que las tecnologías existentes en muchas aplicaciones, incluyendo zonas rurales y redes privadas de celular en áreas remotas [36]. Esto es posible, porque la arquitectura de OpenBTS es diferente de la arquitectura jerárquica GSM convencional, donde la BTS de la red GSM es manejada externamente por la BSC.
Hay que recalcar que, OpenBTS no corresponde a una implementación completa de una red GSM, sino que es la implementación de hardware y software de una BTS. De este modo es posible ofrecer servicios básicos de telefonía móvil de voz y mensajes cortos de texto SMS.

Arquitectura de OpenBTS

Una instalación completa del sistema OpenBTS versión P2.8 requiere de varios componentes de software que interactuan entre si con la finalidad de emular la arquitectura GSM. En el diagrama 3.2 se puede observar la distribución que tienen

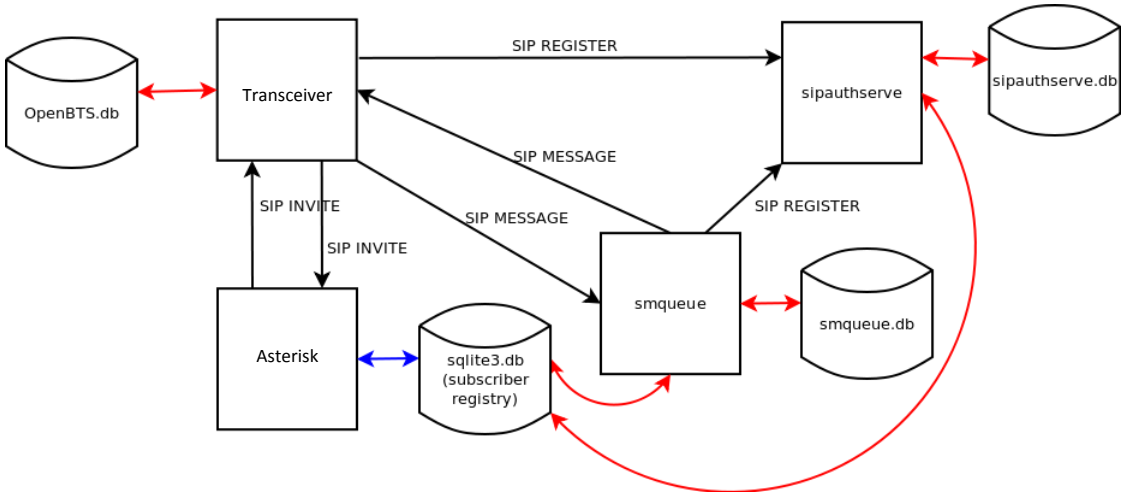

Figura 3.2: Arquitectura de red OpenBTS. Fuente Range Networks

cada una de las partes en la instalación. Se debe señalar que las conexiones de color negro son las conexiones realizadas por medio del protocolo (SIP). Las de color rojo son las conexiones del sistema (sqlite3), y las conexiones azules son establecidas por el sistema OBDC (tablas de datos de red local).

Transceiver

Se trata del radiomodem basado en software, e implementa la capa física L1. Se lo llama transceiver considerando que cumple con las funciones de envío y recepción de la señal de radio Um. El transceiver OpenBTS implementa la red GSM comenzando por la plataforma TDMA de la capa L1 subiendo hasta las capas L3/L4 de aplicación. Su interfaz SIP es normalmente ejecutada en el puerto 5062. El proceso que ejecuta sus funciones se encuentra en el directorio openbts/trunk/apps/.

Sipauthserve

Este es el servidor de registro y autorización de usuarios, usado para procesar las solicitudes de actualización de ubicación que realiza OpenBTS y ejecuta las correspondientes actualizaciones en la base de datos de usuarios (sqlite3.db). Su interfaz de protocolo SIP, normalmente utiliza el puerto 5064 para su comunicación.

Los archivos del servidor, como su proceso de ejecución se alojan en el directorio subscriberRegistry/trunk.

Smqueue

Es el servidor de envío y almacenamiento de mensajes cortos de texto (SMS), los datos de su configuración se encuentran almacenados en la base de datos smqueue.db localizada en etc/OpenBTS. Puede ser inicializado independientemente del transceiver OpenBTS. La interfaz SIP normalmente se ejecuta en el puerto 5063. Si smqueue no es instalado no afecta al funcionamiento del sistema OpenBTS. Sus procesos se encuentran en smqueue/trunk.

Bases de datos

- **OpenBTS.db** En esta base de datos se encuentran todos los parámetros de configuración de OpenBTS, tales como la frecuencia de operación, ARFCN, potencia, puertos, mensajes de aviso, etc. Aquí se registra cualquier modificación que se realice a las características de la red desde su consola de configuración OpenBTSCLI. Se encuentra archivada en el directorio etc/OpenBTS/.
- **Sqlite3.db (subscriber registry)** Es la base de datos donde se alojan todos los usuarios que se registran en la red OpenBTS. Registra tanto los códigos IMSI de las estaciones móviles como los números de extensión asignados a los mismos, el puerto que utilizan, códec usado, dirección ip, etc.
- **sipauthserve.db** Es el archivo usado por el servidor de registro Sipauthserve para alojar sus datos de configuración.

Pila de Protocolos

OpenBTS se basa en la misma pila de protocolos que usa GSM definida en la sección 7 de la especificación GSM 04.01 [15], siguiendo el modelo de capas OSI (Open System Interconnection)[1]. Se encuentra estructurada en tres capas (Layers) que dependen de la interfaz de la arquitectura GSM (Um, Abis[2], y A) a cual corresponda la comunicación, tal como se muestra en la figura 3.3.
La implementación de la arquitectura de OpenBTS comprende una versión paralela de los canales lógicos del estándar GSM definidos en GSM 04.03 [16], enfocándose en la interfaz de la estación móvil MS y la de la estación base BTS por lo que las

[1] `http://es.wikipedia.org/wiki/Modelo_OSI`
[2] `http://en.wikipedia.org/wiki/Application_binary_interface`

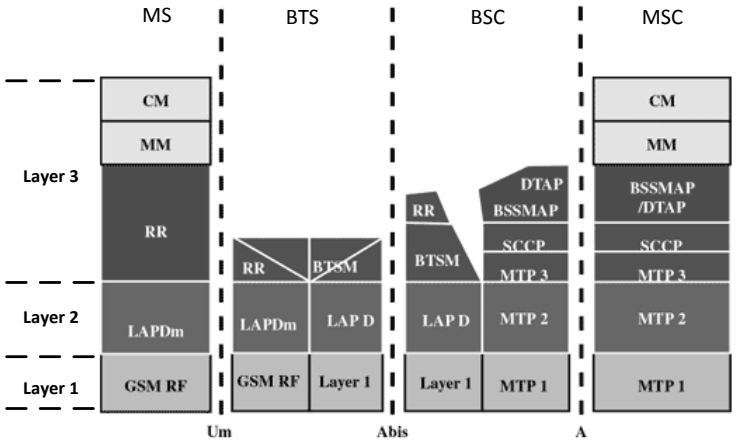

Figura 3.3: Pila de protocolos GSM. Modificado de [15]

capas del protocolo GSM en las interfaces Abis (BSC) y A (MSC) no son tomadas en cuenta. Cada canal de OpenBTS, es construido siguiendo las capas de GSM que a continuación nombraremos.

Capa 1

Se trata de la interfaz de radio RF, también llamada interface Um. Transmite y recibe tramas de control de 184 bits de longitud o tramas de tráfico de 260 bits de longitud sobre la interfaz de radio en ráfagas de 148 bits, una ráfaga por timeslot. La capa se subdivide en tres subcapas:

- **Radiomodem:** Es el componente de hardware de radio USRP o transceiver definido en la especificación GSM 05.04. Implementa la interfaz de radio Um.
- **Multiplexado y Sincronización:** Como ya se ha señalado GSM usa TDMA para subdividir cada canal de radio en un máximo de 16 canales de tráfico o un máximo de 64 canales de control. Las especificaciones de multiplexación se definen en GSM 05.02 [19].
  En GSM la sincronización, es dirigida por la estación base BTS a través de los canales de sincronización SCH y corrección de frecuencia FCCH. Todos los relojes en la estación móvil MS, incluyendo el oscilador local y el reloj de símbolo, se acoplan a las señales recibidas desde la BTS, como se describe en la especificación GSM 05.10.
- **Codificación:** OpenBTS tanto como GSM utilizan codificación FEC. Esta subcapa proporciona una detección y corrección de errores de bit en cada

canal, está definida en GSM 05.03 [20]. Como regla general, cada canal GSM utiliza un código de bloque con bit de paridad, una taza de compresión 1:2, un código convolucional de 4to orden, y un intercalador de 4-rafagas a 8-rafagas de bits. Exceptuando para los canales SCH y RACH que no usan intercalador puesto que transmiten en una sola ráfaga. En OpenBTS el decodificador FEC, valiéndose de un decodificador Viterbi basado en software, puede recuperar tramas íntegras con una taza de borrado de bits de exceso del 25 % [34].

Capa 2

Es la capa de enlace de datos, la cual provee tres funciones básicas:

- Establecer, mantener y finalizar un enlace.
- Control de flujo
- Detección de errores

Además realiza la segmentación de los mensajes de las capas superiores en tramas de L2 para luego ordenar y retransmitir dichas tramas. La capa de enlace de datos es llamada LAPDm y es documentada en las especificaciones GSM 04.05 [17] y 04.06. Es una versión modificada de la interfaz LAPD (Link Access Protocol for D-channel), que es un protocolo para control de enlace en canales D usados para transportar información de control y señalización en redes ISDN[1].
Ambos LAPD y LAPD-m (móvil) son derivadas del protocolo HDLC (High Level Data Link Control) (ISO-13239) [41], usado en frame relay[2], X.25, etc. LAPDm es la mayor de las veces activado en los canales FACCH (Fast Associated Control Channel) y SDCCH (Stand Alone Control Channel), los cuales tienen las mismas funcionalidades del canal D en ISDN (control y señalización). Está también presente en un canal SACCH (Slow Associated Control Channel) cuando un mensaje de texto es enviado o una llamada es iniciada.

Capa 3

En la capa 3 el protocolo de señalización GSM se subdivide en 3 subcapas:

- **Administración de Recursos de radio (RR):** Se encarga del establecimiento del enlace de radio entre la estación móvil y la BTS. Una sesión RR es iniciada por un teléfono cuando este solicita el acceso a la red, o bien para

[1] `http://es.wikipedia.org/wiki/Red_Digital_de_Servicios_Integrados`
[2] `http://es.wikipedia.org/wiki/Frame_Relay`

realizar una llamada. Controla el establecimiento, mantenimiento y terminación de los canales de radio, incluyendo los handovers [18].
Un handover puede ser el cambio de MS a otra BTS, pero también existe el handover intracelular y consiste en cambiar el canal en que se realiza la comunicación dentro de una misma BTS. Este proceso lo ejecuta la BTS midiendo la señal en todos los canales posibles y no solamente el que está activo, determinando así cual es el que tiene menos interferencia.

- **Administración de movilidad (MM)** Dirige la actualización de localización y los procedimientos de registro, tanto como la seguridad y autenticación.
- **Administración de conexión (CM)** Controla en general la llamada, similar a la recomendación CCITT Q.931[1]. Maneja además los servicios suplementarios y de SMS (Short Message Service).

Secuencia de Acceso

A continuación se describe el proceso que siguen la estación móvil MS y la estación base BTS, cuando el usuario de un teléfono realiza una petición de acceso a la red OpenBTS.

1. La estación móvil MS es encendida y empieza a escanear las bandas de GSM en búsqueda de un ARFCN con la mejor señal. La señal producida por el USRP debe estar activa para que el teléfono la detecte.

2. Cuando el ARFCN ha sido seleccionado, la MS busca el canal lógico de sincronización SCH. Si no lo encuentra continúa intentando con otro ARFCN de acuerdo a la potencia que este pueda ofrecer.

3. Luego la MS decodifica el SCH para obtener sincronización con el reloj de la BTS y los relojes de tramas. El contenido del mensaje SCH es descrito en la especificación GSM 04.08 [18].

4. Teniendo la correcta sincronización, la MS puede demultiplexar el canal de control broadcast BCCH. El BCCH identifica la portadora y provee información detallada acerca de los servicios ofrecidos por la BTS y sobre la configuración de multiplexación en el canal común de control (CCCH).

[1] `http://www.itu.int/rec/T-REC-Q.931`

5. Luego, la MS empieza a decodificar el CCCH y envía peticiones de acceso por medio del canal de acceso aleatorio RACH. Cada mensaje al RACH ocupa una sola ráfaga de radio y usa una etiqueta aleatoria. La MS envía máximo 8 ráfagas al canal RACH separadas por retrasos aleatorios de 1 a 2 segundos, mientras que revisa si el canal CCCH ha dado una respuesta.

6. La BTS recibe una ráfaga del tipo RACH enviada por la MS y responde con un mensaje de asignación de canal usando el CCCH. En este mensaje la BTS devuelve la etiqueta y el tiempo asignado para la ráfaga tipo RACH correspondiente, de esta forma la MS podrá reconocer que ráfaga ha sido aceptada. Este mensaje también asigna el canal SDCCH para realizar una comunicación.

7. La estación móvil recibe la asignación inmediata sobre el canal CCCH y cambia su multiplexación y posiblemente su ARFCN de acuerdo a los parámetros recibidos en el mensaje anterior.

8. La MS conmuta sobre el canal SDCCH asignado, espera un corto tiempo hasta detectar un SDCCH en estado de reposo. Al verificar el SDCCH, la MS envía una Petición de Actualización de Localización (LUR) lo que se puede considerar como una identificación temporal [18].

9. Asumiendo que la BTS acepta a la estación móvil MS, la primera responde con un mensaje de Actualización de Localización Aceptado (LUA). La MS y BTS cierran el canal SDCCH y liberan los recursos de radio utilizados.

10. Ahora la estación móvil esta asociada a la BTS y continua monitoreando el CCCH por mensajes de petición de acceso asociados con las tentativas de llamada o el envío de mensajes de texto SMS.

Requerimientos de OpenBTS

Se han nombrado algunos de los principales componentes que conforman el sistema de comunicación OpenBTS, pero también es necesario detallar los requerimientos tanto de software como de hardware que se deben completar antes de desplegar cualquier comunicación.

Requerimientos de software

Es necesario contar con la instalación del sistema operativo S.O. “Linux”, en un computador personal. Debido a que el software libre usado en el presente trabajo se ejecuta bajo dicho sistema, se utilizó la versión de Linux Ubuntu 12.04.2 LTS

(Precise Pangolin)[1].
También se recomienda las versiones Ubuntu 10.04 LTS o Ubuntu 10.10 por la facilidad que presentan estas distribuciones para el manejo de las dependencias de software, como son las librerías y paquetes que compilan e instalan los sistemas OpenBTS, UHD, y Asterisk. En la tabla 3.1 se enumeran los paquetes que deben ser instalados previo a cualquier configuración de los sistemas anteriormente nombrados.

OpenBTS	**Asterisk**	**UHD Driver**
autoconf	Kernel-devel	Libboost-all-dev
libtool	Bluez	Libboost-1.0-0-dev
libosip2	Bluez-libs	Python-cheetah
libortp	Usbutil	Doxygen
libusb-1.0	Pygobject2	Python-docutils
Sqlite3	asterisk-addons-trunk	
libsqlite3-dev	dahdi-linux-complete	
libboost-all-dev	libpri	
Libreadline6-dev		

Tabla 3.1: Librerías de software necesarias en el sistema

Al tener completamente instaladas las dependencias de software, se puede realizar la instalación del software OpenBTS y de los servidores Sipauthserve, Smqueue y OBDC.
Así mismo, como ya se ha visto el software Asterisk es otro de los programas que deben estar instalados. Algunos autores recomiendan el uso de versiones basadas en Asterisk 1.4 o 1.6. por recomendación de los programadores de OpenBTS, las versiones basadas en Asterisk 1.8 presentan un problema a nivel de SIP trabajando con OpenBTS, el cual provoca que las llamadas se terminen a los 32 segundos [33]. Para este trabajo se utilizó la versión 11.

Requerimientos de Hardware

El equipo a utilizar consta de un radio USRP N210, manufacturado por la empresa Ettus Research, e incluye una tarjeta madre motherboard con procesador del tipo FPGA, 2 cables de RF con conectores SMA, una fuente de poder independiente, un cable Ethernet, y componentes para el ensamble del equipo. Conectado a este equipo se utiliza un computador personal PC de mediana capacidad que posea una tarjeta PCI Gigabit Ethernet, que permite la conexión con la interfaz de red del

[1] `http://releases.ubuntu.com/precise/`

USRP.
Además, se usan adaptadores USB-Bluetooth, de marca Cambridge Silicon Radio, que se comportan como dispositivos FXO y/o FXS si fuera el caso para la conexión con redes portadoras privadas, gracias a la aplicación ChanMobile.

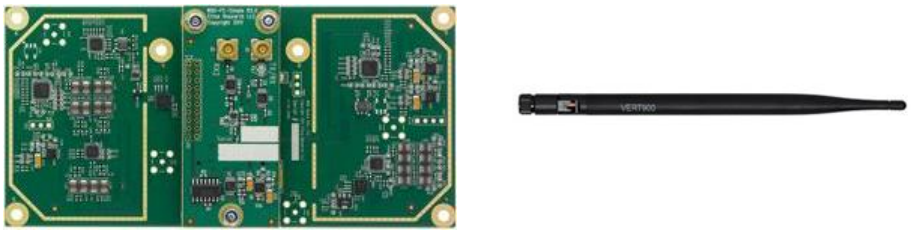

Figura 3.4: Equipamiento Front End de USRP N210: a) Tarjeta daughterboard WBX 2.2 GHz, b) Antena VERT900

Adicionalmente se debe adquirir una tarjeta con soporte de RF y dos antenas que constituyan el "Front-End" del USRP necesarias para cubrir las bandas GSM, ver figura 3.4. En este trabajo se utilizan la tarjeta transmisora y receptora (transceiver) WBX, con una figura de ruido de 5dB que provee una capacidad de ancho de banda de 40 MHz y puede trabajar en frecuencias desde los 50 MHz hasta los 2.2 GHz [30].

Tarjetas RF	WBX	RFX900
Rango de frecuencia	50 MHz a 2.2 GHz	750 MHz a 1050 MHz
Potencia de transmisión	15 dBm	23 dBm

Tabla 3.2: Características de tarjetas RF daughterboards WBX y FRX900

Existen múltiples tarjetas RF de este tipo, como las RFX900 que cubren la banda de 800 a 900 MHz o del tipo RFX1800 que cubren el rango de frecuencias de 1800 a 1900 MHz para GSM. Es recomendable usar dos tarjetas hijas para minimizar la diafonía entre la trasmisión y la recepción; de esta forma se obtiene una mejor calidad de la señal y cobertura.[33]. En la tabla 3.2 se muestran las características de frecuencia y potencia de la tarjeta daugtherboard.

El módulo de transmisión y recepción utiliza dos antenas VERT900 que poseen las características listadas en la tabla 3.3. Su ganancia no supera los 3 dBi por lo que el alcance que la red tendrá no superará los 84 mts, si se aplica en un espacio libre. Usando la siguiente ecuación calculamos esta distancia.

$$Ml = PIRE - Pathloss + Grx - THrx \tag{3.1}$$

Parámetros	Características
Ganancia	3 dBi
Frecuencia	824 MHz a 960 MHz 1710 MHz a 1990 MHz
Banda	Cuadribanda Celular/PCS
Compatibilidad	Daughterboards WBX, RFX900, RFX1800

Tabla 3.3: Características de la antena VERT900

- En donde PIRE es la potencia isotrópica radiada
- Grx es la ganancia del receptor.
- THrx es la potencia mínima que el receptor puede recibir, y que en base a las especificaciones GSM 05.05 para un receptor clase 4 es 105 dBm.

$$PIRE = 15dB + 3dBi = 18dB \tag{3.2}$$

Reemplazando en la primera ecuación los valores de distancia en metros y frecuencia en MHz tenemos.

$$Ml = 18 - 20log(d) + 20log(f) + 27,55 + 3dBi - (-105dB) \tag{3.3}$$

$$20log(d) = 18 - 20log(900) - 27,55 + 3dBi - (-105dB) \tag{3.4}$$

$$log(d) = \frac{38,5}{20} \tag{3.5}$$

$$d = 10^{1,9} \tag{3.6}$$

$$d = 84,1m. \tag{3.7}$$

Por otro lado como interfaces de usuario, es imprescindible el uso de teléfonos celulares compatibles con el estándar GSM y que cuenten con tarjetas SIM. Además los teléfonos gateways FXO, deben poseer capacidad de conexión Bluetooth 2.0; y tener saldo disponible para realizar llamadas en sus respectivas operadoras.

3.2. Configuración de OpenBTS

Antes de poner en marcha la red de aire GSM, algunos parámetros importantes de OpenBTS son modificados con el objetivo de no interferir ninguna otra red cercana, e implementar el sistema sin infringir ninguna de las normas técnicas que la entidad reguladora CONATEL[1] emite para el funcionamiento de redes de telefonía móvil en el país [13].

[1] `http://www.regulaciontelecomunicaciones.gob.ec/`

Banda de Operación

OpenBTS puede usar un amplio rango de frecuencias de GSM como por ejemplo GSM850, GSM900, GSM1800 y GSM1900 [34]. Este será el primer parámetro a definir en las opciones de configuración. Para la implementación del proyecto se seleccionó la frecuencia de 900 MHz usada en Ecuador para los sistemas de radiolocalización, telefonía móvil aeronáutica y no para servicio GSM. De la misma forma se establece una banda de operación para OpenBTS esta banda es definida por el Número de Canal de Radio-Frecuencia Absoluta ARFCN, el cual define el método de acceso múltiple en GSM y provee los canales dowlink y uplink de la estación móvil (MS) para establecer la radiocomunicación con la Estación Base Transceiver (BTS).

Lo siguiente es definir un sistema de numeración para la red. En el estándar GSM se utiliza el código IMSI, el cual permite identificar de forma única una Estación Móvil MS dentro de la red, este se compone del MCC, MNC y el MSIN, como se explicó en la tabla 2.3. OpenBTS maneja este mismo sistema, por lo que los IMSI de los teléfonos a usar dentro de la red son manipulados temporalmente para su registro como extensiones SIP dentro de Asterisk.

Para proceder ha realizar estos cambios en la versión de OpenBTS P2.8, se debe inicializar el sistema OpenBTS desde el directorio de instalación </openbts/trunk/apps>. Y haciendo uso de la consola **OpenBTSCLI** se pueden realizar las modificaciones del sistema usando códigos específicos para cada parámetro. Para ingresar a dicha consola se ejecuta la siguiente orden en un terminal de Linux.

```
/openbts/trunk/apps/ ./OpenBTSCLI
```

Esto ejecutará la consola de OpenBTS, que mostrará algunas opciones de comandos para ingresar como por ejemplo la opción ayuda **help** que indica todas las opciones de entrada disponibles. Una de estas opciones es la sentencia **config** la cual muestra la lista de las configuraciones que el sistema posee por defecto. Además nos ayudará a modificar el primer item planeado que es la frecuencia de operación, junto con la variable **GSM.Radio.Band** que contiene dicho valor de frecuencia.

Hay que recalcar que OpenBTS realiza una asignación automática de frecuencia según lo soporte el hardware en uso o USRP, por lo que si observamos en la lista de configuración existirá ya un valor establecido. Para ingresar la frecuencia de GSM 900 MHz se usa la siguiente sentencia [27].

```
>config GSM.Radio.Band 900
```

Código IMSI

A continuación, debemos identificar los códigos que conformarán el IMSI de los terminales en la red, podemos definir dentro de la configuración de OpenBTS el número MCC, que en este caso es el 740 que corresponde a Ecuador según la recomendación de la Unión Internacional de Telecomunicaciones ITU [10]. Para MNC elegimos un número de dos o tres dígitos diferente al que usan los operadores Claro, Movistar o CNT. En este caso se seleccionó el número 101, quedando de la siguiente manera:

```
>config GSM.Identity.MCC 740
>config GSM.Identity.MNC 101
```

De igual forma hay que indicar los canales de uplink y downlink que van a ser usados dentro de la banda GSM que previamente se programó. Para ello hay que definir el valor de ARFCN más conveniente para el sistema, según la tabla 2.2. Definimos este número usando la siguiente configuración.

```
>config GSM.Radio.C0 51
```

Registro de usuario

Para entregar una experiencia mas interactiva con el usuario, configuramos el mensaje de bienvenida que se recibirá una vez que la MS se registre en la red. Para ello hay que modificar las opciones con las siguientes líneas, el mensaje entregado provendrá del numero 101:

```
>config Control.LUR.OpenRegistration.Message
Bienvenido, registrate enviando al 101 una extension entre 1001 a
1012
>config Control.LUR.OpenRegistration.ShortCode 101
```

Por último, para establecer la característica de una red celular abierta se verifica que la opción de registro abierto este activado usando la siguiente declaración.

```
>config Control.LUR.OpenRegistration .*
```

Realizada esta configuración, cualquier teléfono celular con una tarjeta SIM puede ingresar a la nueva red GSM si es seleccionada. Lo siguiente es proporcionar un encaminamiento de las llamadas que se generen entre las extensiones. Para lo cual se utiliza la herramienta de conmutación por software Asterisk.

3.3. Asterisk

Asterisk es un proyecto de código abierto que agrupa la funcionalidad de una central telefónica privada PBX dentro de un paquete de software, permitiendo realizar el registro, conexión y comunicación de terminales telefónicos conectados al servidor Asterisk. Puede ser instalado en un computador personal de medianas características, al cual se le añaden tarjetas electrónicas telefónicas FXS, que se conectan con teléfonos analógicos y a teléfonos software basados en (VoIP) usando protocolos como SIP, H.323, IAX o MGCP.
A pesar de que se trata de una aplicación software ofrece las mismas características y servicios que un moderno equipo de hardware PBX. Soporta una variedad de tecnologías para hacer y recibir llamadas telefónicas, muchos protocolos VoIP, así como también conectividad analógica y digital a las redes de telefonía tradicional, o a una red digital de servicios integrados (RDSI) [36]. La instalación de Asterisk y algunos de sus detalles de configuración usados para el desarrollo del proyecto de sistema de conmutación para OpenBTS se encuentran detallados en el anexo C.

Arquitectura de Asterisk

En su núcleo Asterisk se encuentra formado por un conjunto de módulos que brindan una gran capacidad de abstracción de los protocolos, códecs e interfaces utilizados en cada conexión esto es lo que hace de Asterisk una herramienta totalmente flexible y adaptable. Existe una variedad de módulos en función de la aplicación que se vaya a desarrollar, pero Asterisk siempre mantiene cuatro módulos principales denominados APIs (Application Programming Interface), en la figura 3.5 podemos observar el núcleo de Asterisk.

Tipos de módulos

Los módulos en Asterisk proveen de una funcionalidad especifica, como puede ser un controlador de canal (p. ejm. chan_sip.so), o un recurso que permite la conexión a una tecnología externa (como func_odbc.so), el cual es utilizado en la gestión de base de datos de usuarios en la red OpenBTS.
Asterisk carga los módulos al iniciar su aplicación, pero además se puede especificar exactamente que módulos cargar y en que orden, modificando un archivo de configuración localizado en el directorio /etc/asterisk/modules.conf, con esto se reduce carga en memoria y se obtienen beneficios de seguridad [6]. Los tipos de módulos se clasifican en subapartados, los mas básicos son:

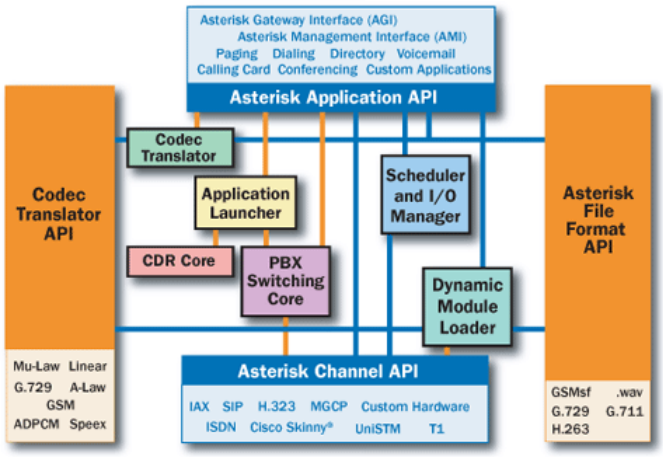

Figura 3.5: Arquitectura de Asterisk. Modificado de [37]

- **Aplicaciones:** Son acciones que se aplican al manejo de las llamadas dentro del "DialPlan" plan de marcado y pueden ser modificadas dentro del archivo extensions.conf. Desde donde se procesan las llamadas entre extensiones que identifican a cada dispositivo conectado a la red. Este proceso será explicado en la parte de configuración de Asterisk.
  Una aplicación comúnmente usada es identificada como "Dial", la cual lanza una llamada a un canal en función de las propiedades que señalemos para su ejecución. El formato de este tipo de modulos es **app_<aplicación>.so**.

- **Controladores de canal:** Los módulos controladores de canal - Channel Drivers aportan la posibilidad de gestionar los canales de manera homogénea. Sin estos módulos, Asterisk no tendría forma de realizar llamadas [6]. Cada llamada entra a Asterisk a través de un controlador de canal, el cual verifica el plan de marcado y asigna un canal de Asterisk a la llamada. El formato de estos módulos es **chan_<controlador>.so**

- **Funciones de plan de marcado:** Añaden y obtienen determinada información específica de cada canal. Son complementarias a las aplicaciones del plan de marcado y ofrecen determinadas mejoras para determinados aspectos del sistema, como el manejo de cadenas de caracteres o la conectividad ODBC (Open DataBase Connectivity, Conectividad Abierta de Bases de Datos), la cual facilita la conexión de la base de datos de usuario con Asterisk. Las funciones no pueden ser llamadas directamente en el plan de marcado, estas son llamadas dentro de las aplicaciones y con letras mayúsculas [38].

- **Traductor de códec y formato:** Como se ha visto Asterisk soporta muchos códecs y formatos de audio, como los de la tabla 3.4, pero también puede utilizar los APIs de traducción y formato de audio cuando sea necesario. Esto

Tipo	Soportados
Códecs de audio	ulaw, alaw, gsm, ilbc, speex, g722, g723, g726, g729.
Formatos de sonido	GSM, PCM, WAV, OGG, SLINEAR, MP3.

Tabla 3.4: Códecs y formatos de audio soportados por Asterisk

  le permite a Asterisk convertir formatos de audio entre llamadas en tiempo real. Por ejemplo, si una llamada se establece desde un circuito PRI (Primary Rate Interface) usando el códec G.711 y necesita ser pasada a un canal SIP usando por ejemplo G.729, el correspondiente traductor de códec será encargado de su conversión.

Estructura de Archivos

Los recursos que Asterisk utiliza en cada módulo, necesitan estar alojados en directorios específicos dentro del sistema de archivos de Linux. De esta forma se pueden realizar varias tareas de administración y almacenamiento como por ejemplo las grabaciones de voz, archivos de configuración y contestación automática.

- **Archivos de configuración:** Todos se encuentran en la ruta por defecto **/etc/asterisk** que se crea en la instalación de Asterisk. Algunos archivos de configuración son **extensions.conf, sip.conf, manager.conf**, además de otros archivos que definen parámetros para una funcionalidad específica. Una parte de los archivos de configuración pueden ser editados de forma dinámica a través de un sistema de Base de datos (sqlite3). A esta funcionalidad se la denomina Asterisk Realtime, y se utiliza dentro del proyecto para brindar una funcionalidad mas intuitiva durante el acceso a la red OpenBTS.

- **Archivos de Módulos:** Son usualmente instalados en el directorio /asterisk/modules desde donde Asterisk realiza su carga. Si un módulo no es cargado se debe revisar su existencia en este directorio, caso contrario habrá que instalarlo.

- **Almacenamiento temporal:** Asterisk usa el almacenamiento temporal para guardar una información transitoria, como mensajes de voz, grabaciones de llamadas generadas por algunas aplicaciones [36]. Para este fin Asterisk utiliza el directorio /var/spool/asterisk.

También existen módulos mas específicos que integran tecnologías no desarrolladas por la empresa Digium, propietaria de Asterisk, lo que permite agregar medios de conexión no convencionales pero que cumplen la misma función. Para el sistema planteado en este trabajo se han utilizado los siguientes tipos:

3.4. Módulo Chan Mobile

Como controlador de canal de Asterisk permite a los teléfonos móviles ser usados como dispositivos FXO (Foreign Exchange Office), mediante una conexión Bluetooth [39]. Chan_Mobile se inicio con el desarrollo del software chan_cellphone que fue

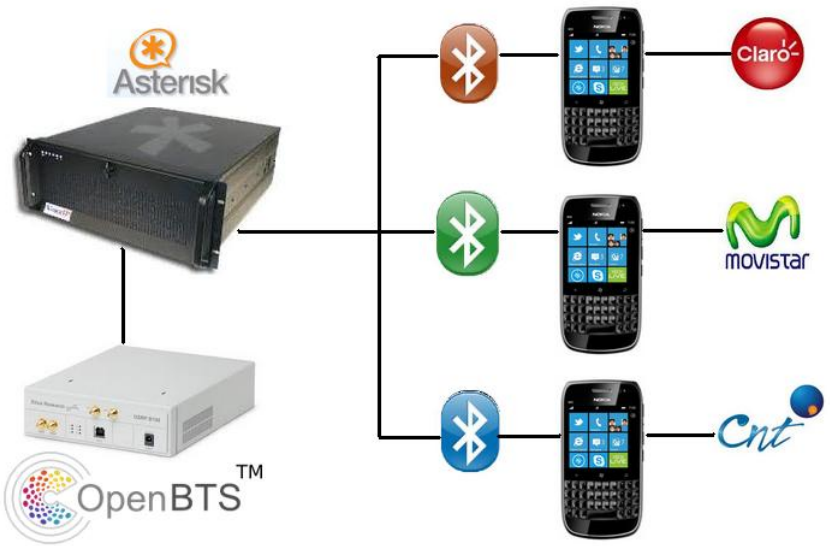

Figura 3.6: Diagrama de conexión Chan_Mobile

desarrollado con el objetivo de enlazar los teléfonos al servidor Asterisk, de forma que se puedan gestionar las llamadas con la operadora celular. El módulo se ha actualizado a la versión chan_mobile e integra varios servicios de conexión mediante el protocolo SIP, además es integrado en el paquete de Asterisk desde su version 1.6.

Características

El módulo posee algunos servicios de interés para el desarrollo del presente trabajo. Permitiendo establecer enlaces salientes con operadoras móviles convencionales

externas al sistema OpenBTS, usando un esquema como el mostrado en la figura 3.6. De esta forma se puede realizar la comunicación móvil a móvil con usuarios externos, algunos de los mencionados servicios se listan a continuación [39].

- Múltiples teléfonos celulares pueden ser conectados.
- Múltiples adaptadores bluetooth pueden ser conectados.
- Asterisk se conecta automáticamente a cada teléfono celular cuando este se encuentra dentro del rango de cobertura RF del dispositivo bluetooth.
- Chan Mobile utiliza comandos específicos para detectar dispositivos cercanos. Son útiles para la configuración.
- Cada conexión utiliza un adaptador Bluetooth desocupado. Si se requiere la conexión a una operadora distinta se debe usar otro dispositivo Bluetooth o "dongle"
- Las llamadas entrantes desde la red operadora a los teléfonos celulares son manejadas por Asterisk.
- Identificación de llamadas entrantes.
- Se debe declarar el plan de salida en un teléfono celular con acceso telefónico, dentro del dialplan de Asterisk.
- La aplicación CellStatus permite comprobar si un teléfono se encuentra en estado conectado o si esta siendo usado por el canal de Asterisk.
- Existe la aplicación MobileSMS para enviar mensajes cortos de texto desde Asterisk a través de un teléfono que sirva como FXO [39].

Sin embargo se debe tener en cuenta que no todos los teléfonos que poseen conexión bluetooth son compatibles con el módulo, en esta pagina [1] se puede encontrar una lista de compatibilidad de los celulares, además de toda la documentación del módulo.

3.5. Módulo Google Voice

OpenBTS se puede comunicar con las aplicaciones de Google Voice y google talk, por medio del uso del módulo de Asterisk **chan_motif** un controlador de canal y el módulo de recursos **res_xmpp** (ver figura3.7).

[1] `http://www.voip-info.org/wiki/view/chan_mobile`

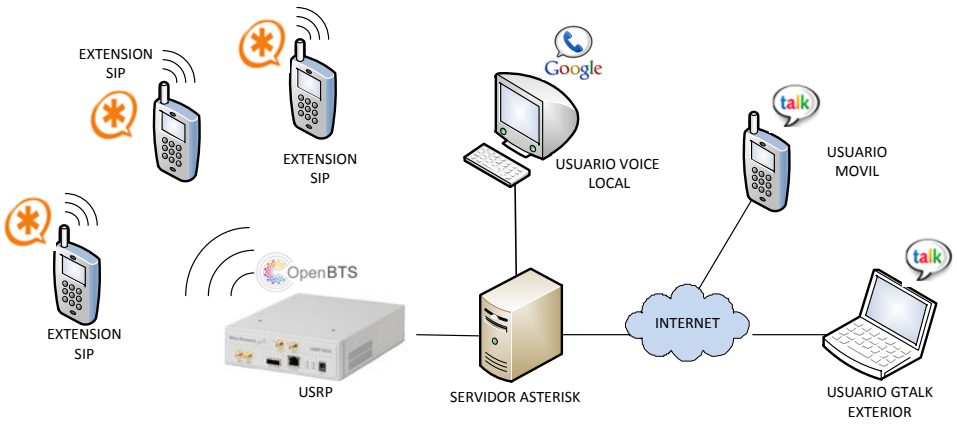

Figura 3.7: Diagrama de red OpenBTS-GoogleTalk

Estos módulos deben ser configurados durante la instalación de asterisk, para lo cual primero debemos de asegurarnos de haber instalado la librería denominada "iksemel", a continuación se muestran los pasos seguidos para descargar e instalar esta librería junto con Asterisk. Cabe indicar que necesitamos contar con una conexión de acceso libre a Internet, para este proyecto inicialmente se utilizó una tarjeta Ethernet adicional modelo TP-link TF3200 con bus PCI compatible con un computador de escritorio, sin embargo también ha sido posible usar la conexión inalámbrica que provee la tarjeta 11 b/g/n Wireless LAN PCI Express en un computador portátil.

Paso 1

Desde una terminal Linux establecemos el directorio de descarga por ejemplo /usr/src, luego introducimos la siguiente línea, que instalará y configurará la librería mencionada.

```
apt-get install libiksemel-dev libssl-dev
```

Paso 2

El módulo res_xmpp depende de los módulos chan_motif y chan_jingle. Por lo tanto debemos agregarlos en la compilación de asterisk, que se realiza desde el directorio donde se encuentra instalado, una vez aquí ejecutamos el siguiente comando:

```
make menuselect
```

A continuación, observaremos un menú de selección de módulos y aplicaciones que podemos agregar en la instalación de Asterisk, dentro de este debemos seleccionar el módulo chan_motif y chan_jingle mediante el uso de la barra espaciadora que presentará un asterisco " * " cuando ha sido seleccionado el componente, ver la figura 3.8.

Luego también seleccionamos dentro de Resource modules el campo res_xmpp. A partir de esto podemos realizar la reinstalación de Asterisk con los nuevos compo-

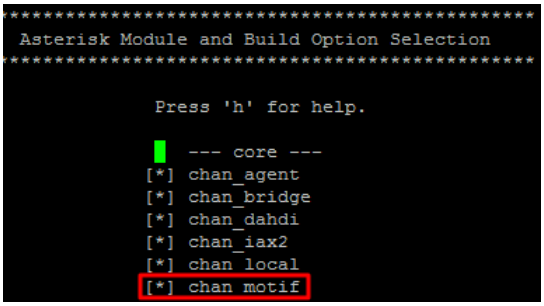

Figura 3.8: Selección del módulo Chan_Motif

nentes, para lo cual ejecutamos la siguiente orden dentro del directorio de instalación de asterisk /usr/src/asterisk.

```
make & make install
```

Esto proporciona a Asterisk nuevas funciones de conexión y recursos de comunicación con Google. Es importante recordar que debemos tener activada una cuenta en Google voice, para lo cual ingresamos desde un navegador a la cuenta de Gmail y nos dirigimos al ícono del servicio, que nos indicará que debemos de instalar el plugin de Google-voice de acuerdo a los pasos establecidos en la página, si este procedimiento se cumple el navegador mostrará una interfaz como la de la figura 3.9.

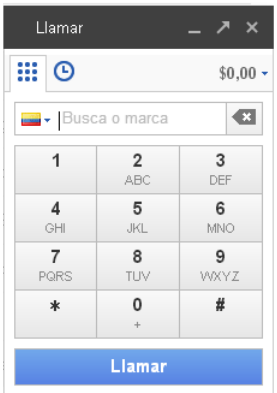

Figura 3.9: Interfaz de usuario Google - Voice

Esta interfaz no es más que un teléfono software con un teclado numérico y una sección para búsqueda de contacto en función del país que indica la bandera adjunta. Además se indica el saldo disponible que tenemos para realizar llamadas desde Google.

Paso 3

Ahora hay que configurar todos los componentes que intervienen en el canal de

comunicación Asterisk-Google. El primero a configurar es el protocolo Real-time Transport Protocol (RTP)[1], el cual permitirá el intercambio de voz en tiempo real.

- **Configuración del protocolo (RTP):** Modificamos las siguientes líneas en el archivo de configuración rtp.conf localizado en /etc/asterisk. Aquí habilitamos el soporte para ICE (Interactive Connectivity Estalishment)[2]

```
general
icesupport=yes
```

  Si esta opción no es habilitada recibiremos el siguiente mensaje

```
Unable to add Google ICE candidates as ICE
support
not available or no candidates available
```

- **Configuración del contexto Motif:** El controlador de canal Motif es configurado en el archivo motif.conf, y es alojado en /etc/asterisk. Se configura de la siguiente manera.

```
[google]
context=incoming-motif
disallow=all
allow=ulaw
connection=google
```

  Con las líneas anteriores se han declarado las siguientes condiciones.

  1. Las llamadas serán terminadas u originadas desde el contexto incoming-motif.
  2. Todos los códecs son explícitamente deshabilitados en el canal.
  3. Se activa solamente el códec G.711 ulaw.
  4. Se utilizará la conexión llamada "google" del tipo XMPP (Extensible Messaging and Presence Protocol) [3].

  Las llamadas con Google pueden soportar algunos códecs los cuales son listados en la pagina `https://developers.google.com/talk/open_communications`.

- **Configuración del módulo XMPP:** El recurso de Asterisk res_xmpp es configurado modificando el archivo xmpp.conf, también encontrado en /etc/asterisk/.

[1] `http://es.wikipedia.org/wiki/Real-time_Transport_Protocol`
[2] `http://en.wikipedia.org/wiki/Interactive_Connectivity_Establishment`
[3] `http://es.wikipedia.org/wiki/Extensible_Messaging_and_Presence_Protocol`

Una configuración normal debe de llevar lo que se muestra a continuación.

```
serverhost=talk.google.com
username=example@gmail.com
secret=examplepassword
priority=25
port=5222
usetls=yes
usesasl=yes
status=available
statusmessage=Imavailable
timeout=5
```

Cada termino anterior corresponde a las siguientes opciones:

- La conexión es tipo cliente al solicitar a Google el servicio.
- Especifica el servidor de gtalk.
- Declara el usuario de gtalk que se usa para conectarse (usuario@gmail.com).
- La contraseña del usuario gtalk.
- Indica la prioridad de conexión.
- El puerto por el cual opera gtalk.
- Habilita la seguridad de capa de transporte (TLS) requerida por Google.
- Activa la capa de seguridad y autenticación simple (SASL) usada por Google.
- Muestra el estatus "available" de gtalk.
- El mensaje que se mostrará en gtalk.
- El tiempo de espera para recibir un mensaje con el tiempo de espera en caso se retrase la red.

Paso 4

Hasta aquí tenemos todo listo para la ejecución de llamadas, únicamente hace falta configurar el plan de marcado que identifique la marcación de una extensión.

```
[incoming-motif]
exten =>s,1,NoOp()
same =>n,Wait(1)
same =>n,Answer()
same =>n,SendDTMF(1)
same =>n,Dial(SIP/1001,20)
```

A continuación creamos un contexto de Asterisk, donde la extensión "s" conteste la llamada entrante y a su vez envíe un tono DTMF al servidor Google, para luego ejecutar la llamada a la extensión sip 1001. Para realizar llamadas salientes a la aplicación de Google voice, utilizamos la siguiente declaratoria.

```
exten =>100,1,Dial(Motif/google/jpablo10e@gmail.com,,r)
```

Aquí se indica que la tecnología usada es el módulo "motif", la extensión a marcar es "google" definida en xmpp.conf, y la cuenta de Google que recibirá la llamada se indica con el correo del usuario. Se usa además la opción de re-conexión "r" debido a que Google no posee reglas propias de marcado [1].

3.6. Conexión con PSTN

Como ya se había mencionado existen interfaces del tipo FXO y FXS en una central de conmutación. La primera permite conectar una línea telefónica analógica perteneciente a la red PSTN (Public Switched Telephone Network). Y la interfaz FXS permite conectar dispositivos analógicos de telefonía para que sean reconocidos como extensiones IP, el objetivo de nuestra red móvil es realizar llamadas a teléfonos fijos de la operadora de telefonía convencional, para esta tarea necesitamos un medio físico de conexión FXO.

Inicialmente Asterisk solo funciona con tarjetas fabricadas por la empresa "Digium" para lo cual se instalan los driver DAHDI (Digium/Asterisk Hardware Device Interface), estas tarjetas brindan el puerto físico de las interfaces FXO y FXS, podemos encontrar múltiples modelos que se ajustan a cualquier aplicación de VoIP que se necesite implementar. Una tarjeta Digium de menores prestaciones es la TDM400P de la figura 3.10, que consta de 4 puertos que funcionan como FXO/FXS según se configure. Pero también, la interfaz FXO la puede proporcionar una tarjeta Módem PCI de bajo presupuesto que sea compatible con los drivers anteriores, esta se debe instalar en el Computador donde se encuentra funcionando Asterisk. El modelo de tarjeta que se ha utilizado en este proyecto es FAX Modem PCI Trendnet V92A.

Para que sea reconocida por Asterisk realizamos el siguiente procedimiento:

Primero, instalamos los drivers - controladores DAHDI este paquete puede ser descargado desde los repositorios de asterisk `www.asterisk.org/downloads`, descargaremos la última versión, descomprimimos el archivo y a través de un terminal de Linux ingresamos a ese directorio, para lo que utilizamos los siguientes comandos.

[1] `https://wiki.asterisk.org/wiki/display/AST/Calling+using+Google`

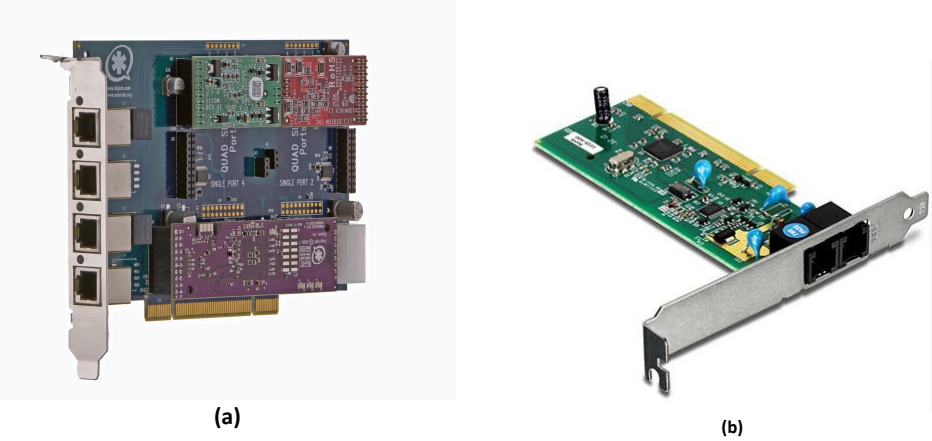

Figura 3.10: Interfaces FXO - FXS: a) Tarjeta Digium TDM400P, b) Modem Trendnet V92A

```
# tar xvfz dahdi-linux-complete-2.5.0.2+2.5.0.2.tar.gz
# ln -s /usr/src/dahdi-linux-complete-2.5.0.2+2.5.0.2/
dahdi
# cd dahdi
# make all
# make install
# make config
```

Los primeros descomprimirán y enviarán el paquete al directorio /usr/src y luego se crea una carpeta llamada dahdi donde se procede a la configuración e instalación de los controladores.

Si la instalación se realizó con éxito la tarjeta Modem PCI será automáticamente detectada por asterisk e imprimirá un mensaje como el siguiente:

```
DAHDI has been configured.
List of detected DAHDI devices:
pci:0000:07:01.0 wcfxo+ 1057:5608 Wildcard X100P
run 'dahdi_genconf modules' to load support for only the
DAHDI hardware installed in this system. By default support
for all DAHDI hardware is loaded at DAHDI start.
```

Lo cual indica que ha sido reconocida como una "wildcard" X100P, si no se muestra nada entonces el hardware no es compatible con dahdi, en la siguiente dirección `http://docs.tzafrir.org.il/dahdi-linux/README.html` se puede verificar la compatibilidad del hardware.

El siguiente paso es ejecutar el comando dahdi_cfg, dentro de la consola de asterisk, y luego el comando dahdi_genconf. De esta forma se generan los archivos de configuración para la tarjeta instalada. Ahora modificaremos uno de esos archivos llamado dahdi_channels.conf, aquí se definen el contexto y callerid que usaremos en

nuestra red. Una muestra de esta configuración es:

```
Span 1: WCTDM/1 ‘‘Wildcard Wildcard X100P’’ (MASTER)
;;; line=‘‘1 WCTDM/1/0’’
signalling=fxo_ls
callerid=‘‘Channel 1’’ <4001>
mailbox=4001
group=5
context=from-internal
channel =>1
callerid=cnt
mailbox=cntbox
group=phones
context=FXO
```

Es importante señalar que Asterisk no reconoce directamente la configuración realizada en el archivo dahdi_channels.conf, sino que se debe copiar esta configuración dentro del archivo chan_dahdi.conf al cual Asterisk tiene como referencia de configuración.

El último paso es configurar el contexto definido en dahdi para que funcione dentro del plan de marcado de Asterisk, en el archivo extensions.conf. En el caso de tarjetas Digium estas poseen varios puertos, que se configuran uno a uno indicando el canal que utilizan para realizar una comunicación. Pero en nuestro caso la tarjeta Modem PCI solo posee un canal de salida, entonces, la configuración en el plan de marcado sería.

```
[FXO]
exten=>_2X.,1,DIAL(DAHDI/1/$EXTEN:1)
exten=>_2X.,2,Hangup()
```

Donde indica que hay que anteponer el 2 a cualquier numeración de teléfono convencional, para utilizar la línea FXO conectada a la tarjeta. El contexto CNT permite que las llamadas de salida o entrada sean enrutadas desde el canal FXO definido en dahdi_channels.conf. Ahora se podrá marcar desde la red OpenBTS a un número de teléfono convencional. El contenido de los archivos de esta configuración están ilustrados en el anexo F.

3.7. Dialplan de Asterisk

La configuración de Asterisk en este apartado se enfoca a las funciones que posee el plan de marcado “DialPlan” con la finalidad de procesar cualquier llamada que se genere dentro del sistema OpenBTS.

También debe aclararse que, toda la configuración que se realice en el Dialplan es guardada en los archivos **sip.conf** y **extensions.conf** que se alojan en el directorio /etc/asterisk/. El primero define los canales SIP que se usan para cursar las llamadas,

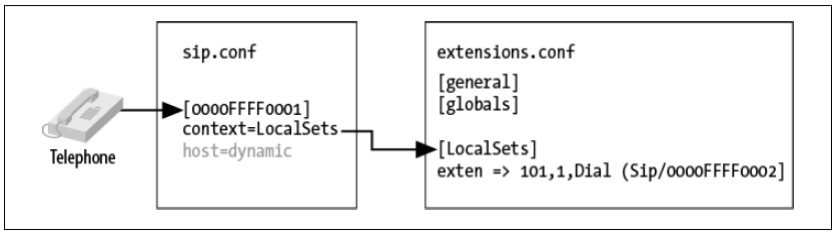

Figura 3.11: Relación entre los archivos de configuración y el dialplan. Tomado de [6]

especificando las características tales como (contexto, puerto, tipo de usuario, códec, etc.). El segundo archivo utiliza los canales definidos en sip.conf y los utiliza dentro de un segmento del plan de marcado llamado contexto, la relación entre estos dos se esquematiza en la figura 3.11.

Contexto

Define los espacios del plan de marcado donde las extensiones son procesadas dinámicamente para cumplir una determinada función. Una extensión que es definida dentro de un contexto es completamente aislada de extensiones en cualquier otro contexto, a menos que la interacción sea específicamente permitida [6].
Para definir un contexto se coloca el nombre entre corchetes **[ ]**, el nombre puede estar compuesto de letras minúsculas o mayúsculas de la A a Z y números del 0 al 9. Hay que asegurarse de no dejar espacios en blanco. Por ejemplo para definir el contexto "phones" se utiliza la siguiente línea.

```
[phones]
```

A partir de la definición del contexto, todas las instrucciones puestas a continuación son parte del contexto, hasta que el siguiente contexto sea definido.
Al inicio del plan de marcado hay dos contextos especiales llamados **[general]** y **[globals]**. El contexto [general] contiene una lista de configuraciones generales del dialplan. Es importante saber que ninguno de estos dos son realmente contextos, por lo que se debe evitar usar [general], [default], y [globals] como nombres de contextos para evitar conflictos del sistema [6].

Extensiones

En asterisk a diferencia del sistema de telefonía fija, una extensión no se refiere solamente a una serie numérica para timbrar un teléfono; sino que a más de cumplir con esa función, define una serie única de pasos a través de los que Asterisk llevará a cabo esa llamada.
Dentro de cada contexto, podemos definir tantas extensiones como sean requeridas. Asterisk no tendrá complicaciones en seguir los pasos definidos para una extensión cuando esta sea activada.
Ahora definimos una extensión con la palabra **"exten"** seguido de una flecha formada por un signo igual y mayor que, tal como se muestra.

```
exten =>
```

Luego de esta definición se coloca el nombre (o número) de la extensión. Asterisk permite que los nombres de una extensión puedan ser una combinación de letras o números. Cada paso en una extensión está compuesto por tres componentes [40]:

- El nombre (o número) de la extensión.
- La prioridad, que es el número que corresponde al siguiente paso al que Asterisk debe procesar, y.
- La aplicación (o comando) que se realizará en ese paso. Estos tres componentes son separados por comas, como se puede ver.

```
exten =>name,priority,aplicatión()
```

Aquí hay un ejemplo sencillo de como se vería una extensión real.

```
exten =>123,1,Answer()
```

El nombre de la extensión llamada es 123, la prioridad es 1, y la aplicación **Answer()** contestará la llamada.

Prioridades

Cada extensión puede tener múltiples pasos, llamados prioridades. Las prioridades se enumeran secuencialmente empezando en 1, y cada una ejecuta una aplicación específica. Por ejemplo en la prioridad uno se contesta la llamada, y en la siguiente prioridad se cuelga.

```
exten =>123,1,Answer()
exten =>123,2,Hangup()
```
Para evitar confusión entre prioridades Asterisk desde la versión 1.2 agregó la prioridad n (next), así cada vez que Asterisk encuentra una prioridad llamada n, toma el número anterior y lo aumenta en 1. Un ejemplo de esto puede ser:

```
exten =>123,1,Answer()
exten =>123,n,hacer algo
exten =>123,n,hacer algo más
exten =>123,n,hacer una ultima cosa
exten =>123,n,Hangup()
```

Se debe tener en cuenta que siempre debe existir la prioridad 1, pues de lo contrario la extensión dejará de existir para Asterisk, pues no encontrará donde empezar su plan de marcado [6].
Para simplificar el código, se dispone de otra función para crear extensiones, se trata del operador "same =>" permite que no sea necesario escribir el número de la extensión, siempre que ésta permanezca igual a la de la línea anterior, reemplazando al operador "exten =>".

```
exten =>123,1,Answer()
same =>n,do something
same =>n,do something else
same =>n, do one last thing
same =>n,Hangup()
```

Los espacios no son necesarios, pero facilita la lectura del plan de marcado. Existen además etiquetas que se pueden asignar a las prioridades de una extensión, esto sirve para realizar saltos de algún lugar de un plan de marcado a otro, permitiendo asignar una aplicación determinada a una extensión que cumpla con una condición [40]. Por ejemplo si un teléfono marca a otro que no esta registrado en OpenBTS. La sintaxis que sigue este complemento es:

```
exten =>123,n(etiqueta),aplicación()
```

Se debe evitar insertar el signo de coma entre la n y el paréntesis de la etiqueta.

Aplicaciones

Las aplicaciones, son encargadas de realizar acciones específicas en el canal seleccionado tales como marcar, colgar, la reproducción de un archivo de sonido, aceptar tonos de entrada, búsqueda en base de datos, entre muchas otras [38].
En los ejemplos anteriores del dialplan se introdujeron dos aplicaciones sencillas:

Answer() y **Hangup()**, las cuales contestan y cuelgan el canal actual respectivamente. Estas funciones no necesitan argumentos, pero la mayoría de aplicaciones si requieren recibir información, estos argumentos se colocan dentro del paréntesis, y se separa usando comas. Por ejemplo la función **Dial()**, puede contener lo siguiente.

```
Dial(SIP/$ARG1,20,rt)
```

Lo que marcará una extensión SIP después de esperar 20 segundos. Otra función básica muy común es **Playback()**, la cual recibe como parámetro la ruta de un archivo de audio para ser reproducido.

Asterisk ejecuta secuencialmente los comandos asociados a cada extensión. Esos comandos son realmente aplicaciones que controlan el comportamiento de la llamada y del sistema en sí. Por ejemplo al marcar la extensión 200 se reproducirá un sonido de saludo "hola mundo" usando la aplicación **Playback()**, y luego se colgara la llamada[40].

```
exten =>200,1,Answer()
same =>n,Playback(hello-world)
same =>n,Hangup()
```

Durante la instalación de asterisk se puede elegir la instalación de algunos sonidos de ejemplo para reproducirlos en el plan. Existen una variedad de sonidos para usarlos durante una condición específica, es común que estos se encuentren en el directorio /var/lib/asterisk/sounds.

Otra función importante para realizar las funciones de conmutación es **Goto()**, la cual sirve para trasladar el estado de la llamada a otra parte del plan de marcado, por ejemplo en el caso que el teléfono de un usuario que está registrado en OpenBTS no puede contestar.

```
same =>n,Goto(context,extention,priority)
```

Como ya se ha visto la función Dial() sirve para marcar la extensión deseada. Su sintaxis tiene el siguiente orden.

```
exten =>n,Dial(Tecnología/usuario,timeout,opciones)
```

En el argumento Tecnología se asigna la abreviatura del tipo de canal que se usará, Asterisk puede manejar algunas tecnologías como SIP, DAHDI e IAX2 [6]. OpenBTS aplica la tecnología SIP por lo tanto usaremos este argumento.

Timeout indica la cantidad de segundos que se deben esperar para recibir una respuesta del equipo llamado. Si la llamada se contesta antes del timeout se desviará la comunicación y el plan de marcado se dará como terminado. Si el destinatario no contesta, está ocupado o no está disponible, Asterisk asigna una variable llamada DIALSTATUS con el estado del llamado y luego continuará con la siguiente prioridad del plan de marcado [40]. Esta función permite terminar las tentativas de llamadas no completadas entre usuarios móviles.

Existe una gran cantidad de opciones para usarse como argumento de la función Dial(). **r y t** son dos funciones usadas en nuestro sistema para brindar una mejor solución a la conmutación de llamadas. La opción **r** envía tonos de espera al llamante y la opción **t** permite al llamado transferir la llamada, si es necesario, usando una secuencia de tonos DTMF[1].
Dentro de la configuración del Sistema de Conmutación para OpenBTS, también utilizamos la función **SET()** la cual permite manejar variables dentro del plan de marcado. Estas variables pueden extraer información de una base de datos perteneciente a OpenBTS. Para realizar este procedimiento se configura a Asterisk en modo Realtime, lo cual es explicado en el anexo C. Un ejemplo en el que se aplica la función Set() sería asignar a la variable "VAR" la extensión SIP 1001.

```
exten =>301,1,Set(VAR=SIP/1001)
same =>n,Dial(VAR,20)
```

Hay que señalar que Asterisk diferencia entre mayúsculas y minúsculas. En el caso de las variable **CHANNEL** y **EXTEN** son reservadas por el sistema, por ejemplo utiliza EXTEN para almacenar el número que ha marcado el usuario llamante [38]. Lo anterior se toma en cuenta cuando declaramos variables globales, estas son definidas dentro del contexto [globals] sin usar Set(), además deben escribirse en mayúsculas, por ejemplo (NOMBREDEVARIABLE). Las variables usadas en el canal se pueden escribirse combinando minúsculas y mayúsculas, por ejemplo (nombreDeVariable)[6].

```
[globals]
var=SIP/1001
```

Una variable global también puede ser definida dentro de un contexto por medio de la función **GLOBAL()**, por ejemplo para definir "VAR" usamos:

```
exten =>301,1,Set(GLOBAL(VAR=SIP/1001))
```

En Asterisk se pueden utilizar patrones de marcado para verificar las extensiones que han sido llamadas, de esta forma se clasifican los tipos de extensiones dentro del plan de marcado, por ejemplo si un número de teléfono móvil empieza por 09 podemos abreviar los siguientes números que le siguen usando la letra X, por ejemplo 09XXXXXXX. Se pueden usar diferentes letras tales como.

- X coincide con un dígito del 0 al 9
- Z coincide con un dígito del 1 al 9

[1] http://es.wikipedia.org/wiki/Marcaci%C3%B3n_por_tonos

- N coincide con un dígito del 2 al 9
- . coincide uno o más caracteres, no importa cual.

Así podremos por ejemplo formar el dialplan de manera que para llamadas locales, el patrón sería NXXXXXX, pues en los números locales no se usa el 1 en la primera cifra. Un patrón para llamadas nacionales sería por ejemplo 0N2XXXXXX.
La función **GotoIf()** también es usada dentro del dialplan de OpenBTS, esta función realiza un salto condicional, es decir que verifica una condición y dependiendo de su validez salta a una determinada etiqueta.

```
GotoIf(condición?etiqueta1:etiqueta2)
```

En caso de cumplirse la condición lógica la llamada será cambiada a la etiqueta 1 que le corresponda, de lo contrario se cambiará a la etiqueta 2. Para facilidad de la construcción del código de marcación uno de las dos etiquetas pueden obviarse, por ejemplo:

```
exten =>345,1,Set(TEST=1)
same =>n,GotoIf([TEST = 1]?:Quito)
same =>n,Playback(Hola-Loja)
same =>n,Hangup()
same =>n(Quito),Playback(Hola-Quito)
same =>n,Hangup()
```

Macros

A lo largo del plan de marcado se pueden utilizar subrutinas que ayudan a desbordar las peticiones de un canal de comunicaciones, estas subrutinas son llamadas "macros".
Se deben de transferir los argumentos válidos para que una macro pueda identificar la función que debe realizar. Una macro se define de igual forma que un "contexto" pero se debe anteponer la palabra "macro" seguida de un guion "-" y luego de éste el "nombre" que se le va a asignar, por ejemplo para definir la macro "claro":

```
[macro-claro])
```

Ahora para ejecutar esta macro, se debe usar el siguiente argumento:

```
exten =>101,1,Macro(claro,ocupado)
```

Lo que hará que al marcar la extensión 101 esta pase a la macro claro, manteniendo el argumento ocupado. Una macro posee unas características específicas que determinan el origen y los argumentos con que se inicia una petición de llamada.

- MACRO_CONTEXT Contexto original desde donde la macro fue llamada.

- MACRO_EXTEN Extensión original desde donde la macro fue llamada.
- MACRO_PRIORITY Prioridad original desde donde la macro fue llamada.
- ARG n Son los argumentos que se pasaron al llamar la macro.

En el ejemplo anterior el segmento de la macro ARG 1 sería igual a ocupado.

Funciones adicionales

Para cumplir algunos de los parámetros necesarios dentro del sistema de conmutación desarrollado en este proyecto, se han utilizado las siguientes funciones específicas.

- LEN(): esta función devuelve el valor numérico del tamaño de la variable o extensión que se coloque entre los paréntesis.
- NoOp: La función NoOp, sirve para mostrar mensajes en la consola de Asterisk (CLI), esto ayudará a verificar el estado del plan de marcado y de las llamadas que se cursen en el sistema.
- MusicOnHold(): Permite la reproducción de un tono de espera para indicar al usuario que llama que debe mantenerse en la línea.

Todas las definiciones anteriores se aplican en el plan de marcado utilizado para realizar el sistema de conmutación para OpenBTS, este dialplan que describe el comportamiento de cada llamada realizada en el sistema se explica en el anexo F.

3.8. Universal Software Radio Peripheral USRP

Se trata de un SDR diseñado por la empresa "ETTUS Research" como propuesta de hardware libre, donde los microprocesadores convencionales pueden actuar como dispositivos de radio bajo un gran ancho de banda, convirtiéndose en una plataforma flexible de bajo costo que permite implementar y diseñar potentes sistemas de radiocomunicaciones con aplicaciones en tiempo real [7]. En esencia, sirve como procesador digital de banda base y conversor de frecuencia intermedia FI en un sistema de radiocomunicación.

La gran comunidad de científicos y usuarios han contribuido a la filosofía de diseño básico detrás del USRP que tiene como objetivo realizar todo el procesamiento de señales específicas como modulación, demodulación e interpolación. Todo lo anterior con la ayuda de un computador personal sin tener que adquirir software especializado o pagar una licencia.

Hardware USRP - N210

El N210, es parte de la familia de equipos SDR que ofrece Ettus Research, en la figura 3.12 podemos observar su arquitectura de hardware. En el lado de recepción posee dos ADCs de alta velocidad con entradas a 14 bits por muestra con una tasa de 100 mega-muestras por segundo (100 MSPS) [28]; en teoría se podría muestrear una señal de hasta 50 MHz.

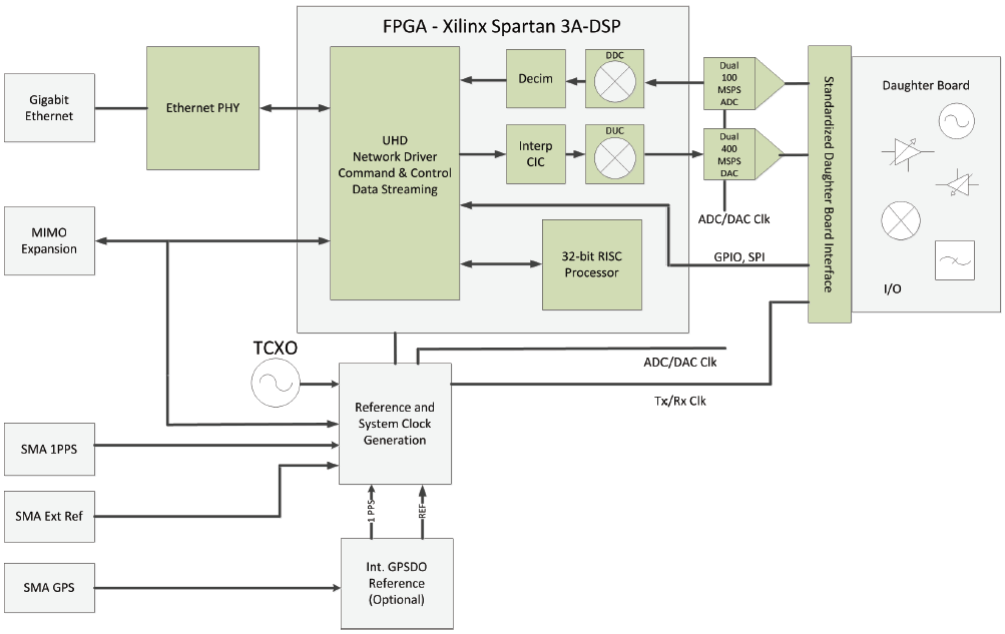

Figura 3.12: Arquitectura de hardware USRP. Tomado de [28]

Cuenta con un PGA (Programable Gain Amplifier, amplificador de potencia programable) antes de los ADCs para amplificar la señal de entrada y utilizar el rango completo en caso de que la señal sea débil. En la parte de transmsión el USRP tiene dos DACs de alta velocidad a 16 bits por muestra y una tasa de 400 mega-muestras por segundo (400 MSPS), contando de igual forma con un PGA después de los DACs que proporciona hasta 10 mW de ganancia [29].

Estos canales de entrada y salida son conectados a un FPGA (Field-Programable GateArray, Matriz de Compuertas Programable en campo) de la familia Xilinx Spartan 3A-DSP 3400 FPGA, al cual se conecta un chip Gigabit Ethernet que sirve como interfaz de conexión al computador con un ancho de banda de 50 MHz usando una cuantización de 8 bits.

La FPGA realiza un procesamiento a alta velocidad y reduce la tasa de datos para que puedan ser enviados a través de la interfaz Gigabit-Ethernet hasta el computador. En el N210, el procesamiento con alta frecuencia de muestreo se realiza en la

FPGA, mientras el procesamiento con baja frecuencia de muestreo se realiza en el computador.
La configuración básica de la FPGA incluye dos DDCs completos, pero también es posible la implementación de 4 DDCs sin filtros de media banda. Esto permite tener 1, 2 o 4 canales de recepción separados. Las salidas de los ADCs van conectadas a la entradas de los DDC. Estos mezclan, filtran y diezman las señales de entrada en la FPGA. Se utilizan en la recepción, esencialmente por dos razones:

- Para convertir la señal en banda de frecuencia FI a una señal en banda base.
- Para diezmar la señal, logrando que la tasa de datos pueda ser adaptada a la interfaz Gigabit Ethernet y que sea acorde a la capacidad de procesamiento del computador.

En la transmisión se realiza el proceso inverso, donde es necesario convertir una señal banda base a una señal de frecuencia intermedia, y enviarla a través de los DACs. Esto proceso lo realizan los DUC. En la transmisión se usan filtros interpoladores CIC (CascadedIntengrator-Comb, Peine Integrador en Casca) que interpolan las muestras antes de trasladar la señal digital a la frecuencia intermedia por el DUC. Los DDC y DUC combinados con altas tasas de muestreo simplifican en gran medida los requerimientos de filtrado analógico.

Panel Frontal

En la parte frontal del equipo N210 se han colocado 6 leds enumerados y de color verde, que indican un determinado estado del dispositivo como se observa en la figura 3.13. Según las letras que se han asignado a los leds pueden indicar los siguientes estados si están encendidos:

- **LED A:** El equipo está transmitiendo Tx.
- **LED B:** Indica si está conectado el cable tipo MIMO[1].
- **LED C:** Recepción.
- **LED D:** El firmware ha sido cargado.
- **LED E:** Reloj de referencia.
- **LED F:** CPLD (Complex Programmable Logic Device) está activo.

[1] `http://es.wikipedia.org/wiki/MIMO`

En el panel también encontraremos los puertos tanto de la interfaz Gigabit Ethernet que será conectada con el computador, el puerto de expansión MIMO utilizado para realizar una conexión en paralelo con otros USRPs si es necesario para sincronizarlos con el mismo reloj.

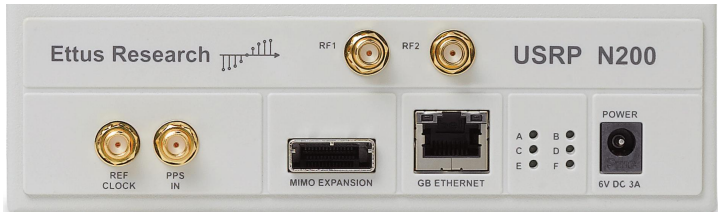

Figura 3.13: Panel frontal del equipo N210

Así mismo tenemos los puertos que ya se han nombrado para conectar las antenas de transmisión y recepción estos puertos están etiquetados como "RF1" para transmisión y "RF2" para recepción. Y por ultimo están los puertos "REF clock" y "PPS in" que como indican sirven para conectar un reloj de referencia en el primero o un Multivibrador[1] en el segundo ambos con la finalidad de realizar la sincronización necesaria para las aplicaciones que se desarrollen. La siguiente tabla 3.5 indica los parámetros permitidos de reloj o multivibrador.

Puerto	**Tolerancia de entrada**
REF Clock	Max 10 MHz Reference CLock con 0 - 15 dBm nivel de potencia.
PPS in	3.3 a 5 Vpp (voltaje pico-pico).

Tabla 3.5: Valores de entrada permitidos en el panel frontal

GPSDO

El dispositivo Global Positioning System Disciplined Oscillator (GPSDO)[2], permite una sincronización precisa de muestras para transmisión Tx y recepción Rx en el USRP, generando una frecuencia de reloj de 10 MHz en su oscilador de salida con una precisión de 0.01 ppm[3], acercándose a la frecuencia de 13 MHz con 0.02 ppm requerido en la mayoría de las implementaciones GSM tanto para la generación de frecuencia como para la sincronización del reloj [32].

El USRP N210 tiene incorporado en su tarjeta principal un Oscilador de Cristal de Temperatura Compensada (TCXO) de 64 MHZ con precisión de 2.5 ppm, muy

[1] `http://es.wikipedia.org/wiki/Multivibrador`
[2] `http://en.wikipedia.org/wiki/GPS_disciplined_oscillator`
[3] Partes por millón

baja para trabajar como reloj principal en la red. Por lo que se recomienda usar la señal del GPSDO como entrada de reloj de referencia REF en la tarjeta motherboard del USRP. Los pasos correctos para la conexión del GPSDO con el equipo USRP los podemos encontrar en la guía desarrollada por el fabricante [31].

UHD

UHD (USRP Hardware Driver) se trata de el software controlador o driver, que se debe instalar en el ordenador para que pueda interactuar con el radio USRP, es desarrollado por Ettus Research para el desarrollo de aplicaciones con los equipos de la familia USRP. Puede trabajar en los sistemas operativos Linux, Windows, y Mac.
Posibilita una Interfaz de Programación de Aplicaciones (API) para la investigación de nuevos servicios que se pueden adaptar a los radios basados en software SDR. Los usuarios de UHD pueden realizar la comunicación del hardware usrp bajo plataformas de desarrollo de software como:

- GNU Radio
- LabVIEW
- MATLAB
- OpenBTS

4

FUNCIONAMIENTO Y PRUEBAS

4.1. Conexión del Equipo N210

Antes de iniciar el sistema de telefonía móvil, necesitamos realizar una prueba de funcionamiento y conectividad del equipo USRP con el computador. Todas las pruebas que se realizaron durante este trabajo han sido desarrolladas dentro de un ambiente de laboratorio donde el nivel de señal GSM de otras operadoras es muy bajo. Esto favorece la detección y acceso de los teléfonos móviles hacia la red GSM implementada.

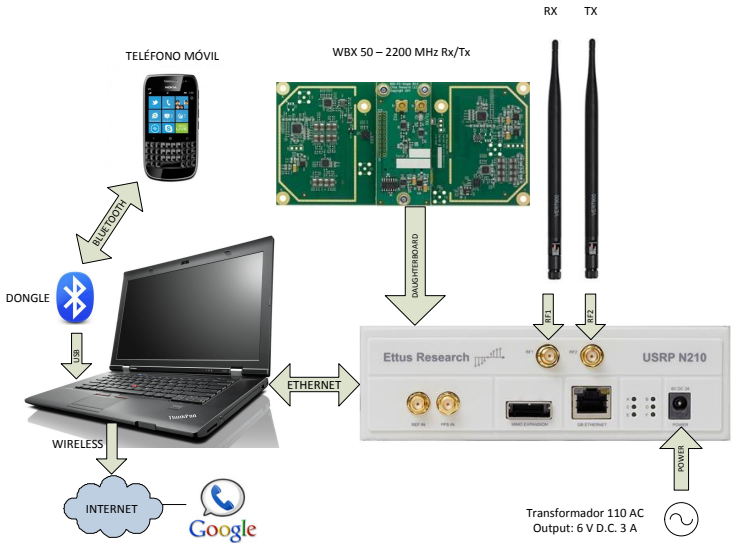

Figura 4.1: Diagrama de conexión del equipo

Es indispensable tener disponible una toma de energía de 110 V en AC (corriente alterna) para facilitar la alimentación tanto del SDR como el computador. A partir de esto se puede empezar a conectar físicamente todas las partes como se detalla en la figura 4.1. Primero adaptamos cuidadosamente la tarjeta RF WBX dentro del USRP para luego colocar en los conectores las antenas VERT900 de Tx y Rx, este proceso es detallado en el anexo D. También mediante un cable Ethernet categoría 6, conectamos el computador con el SDR y además realizamos el enlace bluetooth de los teléfonos gateway con el servidor asterisk. Por último verificamos si disponemos de una conexión a Internet para comunicarnos con las cuentas Google que deben estar activas.

Seguido comprobamos si una nueva conexión en la interfaz Ethernet se ha activado, Ubuntu la detectará automáticamente y tendremos que introducir una dirección IP estática para comunicación con el USRP que posee la dirección por defecto **192.168.10.2**

Una vez realizado lo anterior, ahora se puede ejecutar dentro de un terminal "Linux"

dos aplicaciones que se instalan junto con el UHD la primera "uhd_find_devices" detecta el dispositivo USRP conectado, y la segunda "uhd_usrp_probe " examina el estado de todas las tarjetas que se encuentren activas presentando las características como rango de frecuencia, sensores activos, y nivel de ganancia que posee cada una ellas. Este procedimiento se lleva a cabo con la finalidad de comprobar que el equipo se encuentre en buen estado y disponible para ejecutar el sistema.

A continuación hacemos uso del programa denominado **Kalibrator** que permite escanear las bandas de frecuencia GSM activas junto con los identificadores ARFCN usados las redes móviles que operen dentro del área. Para ejecutar este programa escribimos en un terminal Linux la siguiente línea de comandos:

```
kal -s GSM850 -F 52000000 -R B
```

Indicando la banda de 850 MHz la cual vamos a escanear, también se ingresa la frecuencia de muestreo necesaria para procesar la señal requerida, luego el resultado debe ser similar al mostrado en la figura 4.2. Las opciones de banda a escanear son la GSM850, GSM900, DCS y PCS.

El programa muestra el número de canal ARFCN usado junto con la potencia y

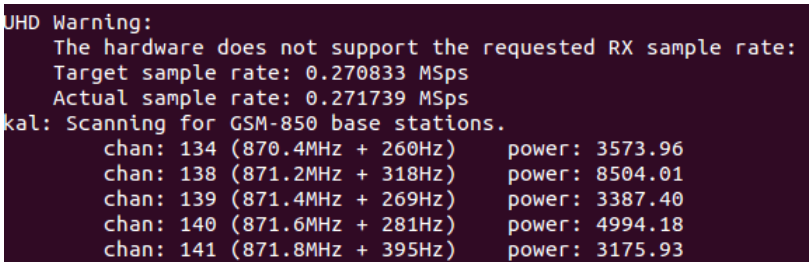

Figura 4.2: Frecuencias usadas en la banda GSM850

la frecuencia de cada canal, con esta información podremos evitar usar un mismo número de ARFCN en nuestra red y causar una interferencia a esa operadora.

4.2. Ejecución del sistema

Realizada la conexión y alimentación de los equipos procedemos al inicio del sistema OpenBTS, el sistema comprende la ejecución de los siguientes procesos, todos en el mismo computador:

1. El primer proceso es el servidor de registro (Sipauthserve), el cual se encuentra en el directorio /SubscriberRegistry/Trunk, lo ejecutamos usando la sentencia **./sipauthserve**

2. Iniciamos el servidor de mensajes de texto SMS (smqueue), ingresamos al directorio /smqueue/trunk/smqueue y lo iniciamos mediante **./smqueue**

3. Luego de los servidores anteriores se ejecuta el transceiver (OpenBTS) que está alojado en /openbts/trunk/apps con el comando **./OpenBTS**

4. Por último se inicia el servidor Asterisk el cual lo podemos llamar desde una consola de Linux por medio de **asterisk -vvvvvvc**.

Cuando la red se encuentra en funcionamiento y la transmisión está activa, los teléfonos dentro de la cobertura de la celda celular podrán notar la existencia de una nueva red dentro de la lista de redes operadoras disponibles en su menú de configuración, la que pueden acceder de forma manual si así lo deciden para registrarse. Al registrarse los teléfonos en OpenBTS, este les asigna una Identidad de Usuario Móvil (IMSI) hasta que se registren de forma completa como extensión SIP dentro de Asterisk, si se borran estas identificaciones temporales OpenBTS obliga al teléfono a que nuevamente se registre, mas adelante se explica este procedimiento.
Luego, tenemos la opción de abrir la interfaz de líneas de comando (CLI) que ya habíamos usado antes, desde aquí podemos generar llamadas a los usuarios y enviar mensajes de texto de prueba, además podemos observar los identificadores temporales (TIMSI) de las estaciones móviles registradas. Para enviar mensajes a un terminal se utiliza la siguiente sentencia:

```
OpenBTSCLI>sendsms 740000207538464 101
enter text to send: Hola desde Consola
message submitted for delivery
```

Donde:

- 740000207538464 es el número IMSI del teléfono al que se envía el mensaje
- 101 es el número usado para enviar el mensaje
- El texto: “Hola desde consola” es el mensaje en si

En el terminal se mostrará una notificación que el mensaje ha sido remitido para ser enviado, luego el mensaje es recibido en el teléfono. Esta función permite realizar un chequeo de la conexión de OpenBTS con los terminales y así mismo probar la implementación de las capas del estándar GSM.
Por otro lado Asterisk esta configurado para aceptar llamadas desde cualquier usuario registrado en la red, usando el identificador internacional de usuario IMSI. Dado que el identificador de usuario es demasiado extenso para marcar, se usa extensiones mas cortas dentro del plan de marcado de Asterisk, por lo que se ha determinado

conveniente usar números mas cortos para cada teléfono en un rango de “1001” a “1012” que dan como base un límite de registro de 12 usuarios, tomando en cuenta que OpenBTS al usar un solo ARFCN soporta siete llamadas simultaneas [34].
Todos los teléfonos que envían la petición de registro son inicialmente almacenados dentro de la base de datos “sqlite3.db” con todas las características de una extensión SIP tales como contexto, callerid, tipo de códec, puerto, dirección IP, etc. Esta información es utilizada simultáneamente por el servidor de registro (sipauthserve), el servidor de mensajes (smqueue), y Asterisk, al cual se configura para que maneje la base de datos en tiempo real. Esto quiere decir, que cuando un usuario realice una llamada o envíe un SMS a otro usuario, tanto Asterisk como los otros servidores usaran los datos registrados dentro la base. Cada usuario SIP usa el puerto 5062 y la dirección 127.0.0.1 como dominio para iniciar una llamada, esta dirección pertenece al servidor OpenBTS.

4.3. Registro de terminales en OpenBTS

El usuario que desee acceder a la red OpenBTS debe poseer un teléfono celular con su respectiva tarjeta SIM, la tarjeta puede ser libre o de otra operadora la diferencia está en que si se encuentra registrada en una operadora la información guardada en su memoria debe ser reiniciada para que tome la información de la red OpenBTS, esto se realiza de forma automática cuando se reinicia el teléfono y se selecciona de forma manual la red preferida en este caso se selecciona la red de nombre **UTPL**.

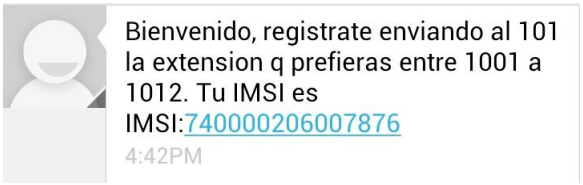

Figura 4.3: Mensaje de Bienvenida a OpenBTS

Al ingresar a la red el teléfono, automáticamente recibirá un mensaje de bienvenida enviado desde la extensión 101 que OpenBTS tiene configurado como número “outbounb”. En el mensaje se solicita que envíe una serie de números que servirán como extensión para el usuario; el mensaje en un teléfono se ve como en la figura 4.3.
La respuesta que el usuario debe dar al mensaje es el número a solicitar, en caso de que este número este registrado con el IMSI de otro teléfono, el usuario recibirá una notificación aclarando que el número ya se encuentra en uso, y que además puede

intentar enviando otro número dentro del rango establecido como se observa en la figura 4.4.

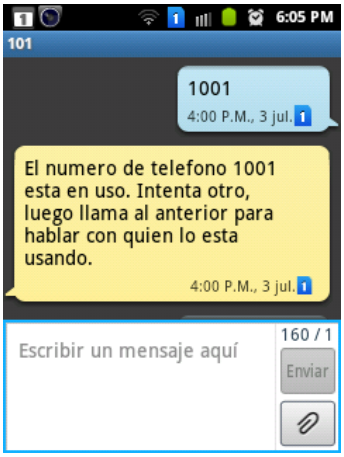

Figura 4.4: Mensaje de Aviso, número de extensión esta en uso

Al enviar la petición de registro, cumpliendo con la condición de que el número no está siendo usado por otro usuario, entonces inmediatamente recibirá un mensaje de haber completado el registro junto con el número aceptado por el sistema, este mensaje esta representado en la figura 4.5. El usuario entonces podrá usar su número tanto para realizar y recibir llamadas como para enviar o recibir mensajes de texto. Los mensajes de texto poseen un límite de 160 caracteres por mensaje, para ser usados en cada comunicación.

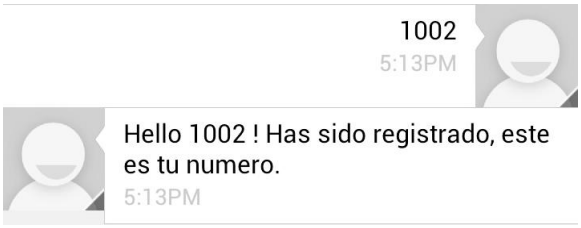

Figura 4.5: Mensaje que recibe usuario al ser registrado

En caso de que el usuario no recibió el mensaje de confirmación, el administrador deberá revisar si el número enviado está completo y se encuentre dentro del rango establecido. El usuario en ambos casos recibirá un mensaje aclarando su estado, ya sea por no haber enviado el número completo o porque está en uso, como en el caso anterior.

4.4. Interfaz Gráfica de Usuario GUI

Con el objetivo de facilitar el uso del sistema OpenBTS y sus aplicaciones, se ha desarrollado una Interfaz Gráfica de Usuario, dentro del sistema operativo Ubuntu. El programa que facilitó el desarrollo de la GUI, se denomina "Gambas" `http://gambas.sourceforge.net/en/main.html` que es una plataforma dentro de Unix, que permite la programación visual orientada a objetos similar al conocido lenguaje Visual Basic.

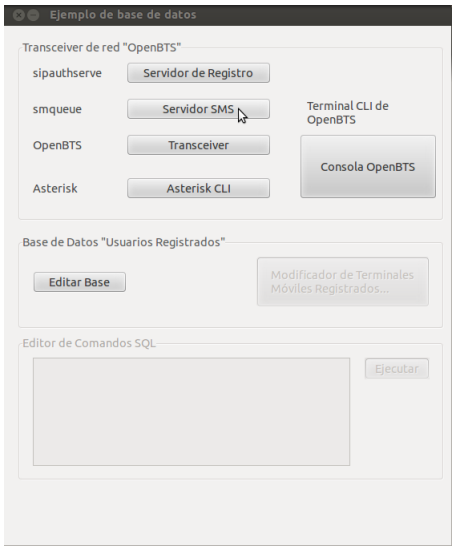

Figura 4.6: Ventana de Interfaz Gráfica de Usuario

La interfaz consta de un módulo de arranque del sistema, donde el usuario debe iniciar obligatoriamente los servidores de Registro (sipauthserve) y Mensajes (smqueue). Todos los procesos del sistema se ejecutan de manera automática en terminales del sistema Unix. Donde podremos controlar tanto su inicio como su ejecución.
Luego de ejecutar los servidores, podemos levantar la estación base abriendo el proceso Transceiver desde la interfaz. A partir de este evento, la red celular de OpenBTS ya puede ser detectada por los teléfonos móviles circundantes en el área.
El siguiente componente que debemos iniciar es el servidor Asterisk, que se ejecuta también dentro de una consola Unix. Este programa, iniciara sus módulos de conexión con los teléfonos Gateway (FXO), y a su vez ejecuta el sistema de gestión de base de datos ODBC que manejará la tabla de registro de usuarios, y por último cargará el plan de marcado para que las extensiones registradas puedan realizar sus llamadas.
A este nivel el sistema se encontrará levantado completamente, por lo que estamos

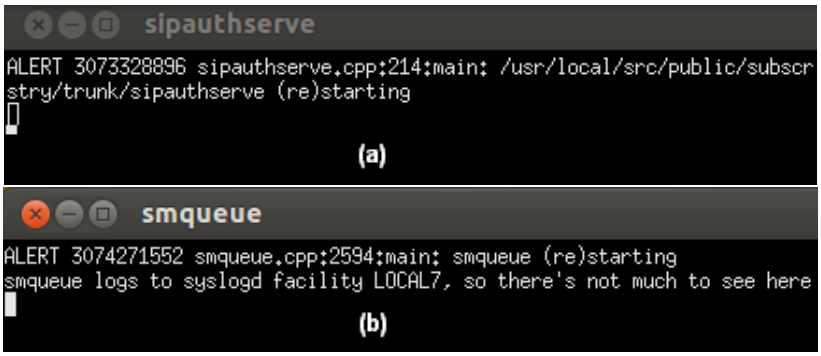

Figura 4.7: Servidores de OpenBTS: a) Servidor de Registro, b) Servidor de Mensajes de texto SMS

en posición de gestionar el tráfico generado en nuestra red y administrar sus componentes. Una primera tarea de administración que podemos realizar es analizar los parámetros de los usuarios registrados en la base de datos (sqlite3.db), para lo cual inicializamos la edición de la base en el segundo módulo de la Interfaz GUI, esto conectará la base de datos y podremos dirigirnos al segmento Modificador de terminales/Móviles registrados. Dentro de este segmento, observaremos en la parte superior, los números de los usuarios registrados y sus números IMSIs, al seleccionar uno ellos podremos editar cada uno de los parámetros que definen a ese usuario usando el lenguaje SQL de modificación de base de datos [1].

4.5. Vinculación de teléfonos (FXO)

Los teléfonos que funcionan como puertas de enlace hacia operadoras móviles externas, deben estar conectados primeramente al servidor de Asterisk para que este reconozca el puerto físico y lógico que proporcionan estos teléfonos. Lo primero que debemos configurar es la dirección física MAC tanto del dispositivo "dongle" como del teléfono, así como el puerto que está usando para la conexión con Asterisk. Cada parámetro es obtenido como se explica en los siguientes segmentos.

MAC del adaptador Bluetooth

Desde la terminal de Linux Ubuntu es posible utilizar la herramienta "hcitool dev", que permite visualizar los elementos conectados mediante bluetooth y entrega la información necesaria para la configuración final, en el siguiente caso el dispositivo "hci0" posee la MAC que se muestra a continuación.

[1] `http://www.sql.org/`

```
root@utpl:# hcitool dev
Devices:
hci0 00:1B:10:00:2A:EC
```

MAC del teléfono móvil

La dirección física MAC del adaptador como la del teléfono móvil, son necesarias para la configuración del modulo chan_mobile. Dicho módulo es configurado por medio de la modificación del archivo mobile.conf, que será posteriormente cargado en el servidor de Asterisk.
Para obtener la dirección MAC del teléfono ejecutamos desde el terminal ubuntu, el comando "hcitool scan", el cual mostrará el nombre del teléfono y su dirección MAC.

```
root@utpl:# hcitool scan
Scanning ...
C8:19:F7:3A:9F:5E GT-S6102
```

El siguiente parámetro también importante es el puerto que utiliza cada dispositivo para su conexión, este número de puerto lo obtenemos desde la consola de asterisk. Como ya se ha visto para entrar a la consola de asterisk ejecutamos en un terminal la sentencia "asterisk -r". Dentro de la consola se ejecuta el comando "mobile search" que despliega una lista de los puertos y dispositivos acoplados al sistema de asterisk. Tal como muestra la figura 4.8.

```
*CLI> mobile show devices
ID              Address           Group Adapter         Connected State
GT-S6102        C8:19:F7:3A:9F:5E 0     dong            Yes       Free
```

Figura 4.8: Dispositivos móviles conectados a Asterisk

Una vez obtenidos los datos anteriores, ya podemos configurar el fichero mobile.conf. La edición del archivo es realizado mediante el editor de textos ejecutado desde el terminal "gedit /etc/asterisk/mobile.conf". Dentro de este colocamos los parámetros que se han obtenido en los pasos anteriores y procedemos a guardar la configuración.

```
[adapter]
address = 00:1B:10:00:2A:EC
id = dong
[GT-S6102]
address = C8:19:F7:3A:9F:5E
port = 1
context = phones
adapter = dong
```

De vuelta a la consola de Asterisk, recargamos el modulo de chan_mobile mediante los comandos "module load chan_mobile.so" que actualizará los cambios realizados en la configuración. Una vez terminado esto, tendremos una petición en nuestro teléfono para conectarnos hacia la máquina que ejecuta Asterisk. Nos solicitará que confirmemos la clave de conexión que permite la vinculación de ambos dispositivos la cual debemos aceptar.
Dentro de Asterisk se muestra que la conexión se ha realizado con éxito. Nuevamente, ejecutamos el comando "mobile show devices", con el que podremos observar los teléfonos asociados al servidor, además del estado de su conexión y si soportan el envío de SMS.

4.6. Prueba de servicios del sistema

El último paso a comprobar es el comportamiento del sistema frente al uso de los servicios de red que los usuarios realicen una vez que sus teléfonos se encuentren dentro de la red OpenBTS, el servicio de llamadas es administrado por el plan de marcado "dialplan" que hemos configurado en Asterisk y el cual es explicado en el Anexo F. Por otro lado, el sistema de mensajes cortos de texto es manejado por el centro de mensajería smqueue. Los servicios a los cuales el usuario puede acceder son listados a continuación:

- Llamadas locales entre usuarios móviles de OpenBTS.
- Llamadas desde un terminal OpenBTS hacia otra operadora móvil (Claro, Movistar, CNT).
- Llamadas desde un terminal OpenBTS hacia una cuenta Google activa y configurada en Asterisk.
- Llamadas desde un terminal OpenBTS hacia un teléfono fijo.
- Mensajes de texto SMS entre terminales registrados en la red.

- Mensajes SMS entre el centro de mensajería y el usuario.

Para que un usuario pueda llamar a otro primero debe marcar el número que posee el usuario llamado de otra forma su tentativa de llamada será descartada y colgada. El responsable de que se registren y se ejecuten estas llamadas es Asterisk, y gracias a su consola CLI podemos monitorear el inicio, ejecución y terminación de una llamada. Por ejemplo si el usuario que posee el terminal "1001" realiza una tentativa de llamada al usuario "1002", se completa cuando el último contesta esta llamada, todo el proceso que se lleva a cabo puede ser observado dentro de la consola de Asterisk tal como se muestra en la figura 4.9.

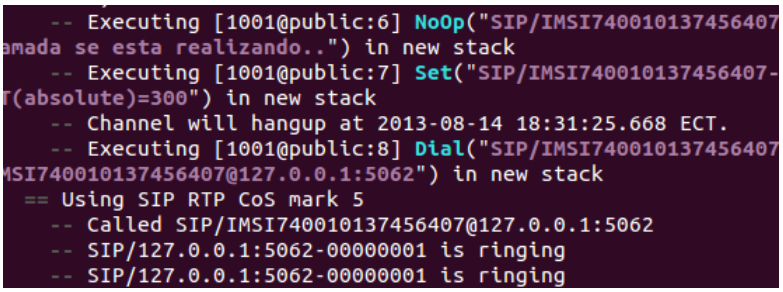

Figura 4.9: Ejecución de llamada entre usuario 1001 y usuario 1002

Si algún usuario requiere realizar una llamada a un teléfono de una operadora móvil privada lo puede hacer marcando desde su terminal el número que corresponda al usuario. Esta llamada será aceptada únicamente si el canal montado por chan_mobile está libre, es decir, solamente se puede realizar una sola llamada por canal debido a que estos canales son construidos por teléfonos celulares ordinarios registrados en esas operadoras y no pueden realizar llamadas simultáneas.

En cuanto a la comunicación con los servicios de Google, se ha asignado extensiones específicas para cada cuenta de google a marcar, por ejemplo si se quisiera comunicar con la cuenta de Google "jpablo10e@gmail.com" un usuario de celular OpenBTS debe de marcar la extensión "400". Esto puede ser un poco complicado desde el punto de vista del usuario móvil ya que no sabrá que esa extensión existe salvo que se comunique previamente mediante mensajes desde consola a todos los usuarios del sistema. Esta tarea la debe de cumplir el administrador de la red para evitar confusiones y congestionamiento en la red. Durante las pruebas realizadas para comprobar la ejecución de llamadas telefónicas se ha utilizado un esquema de conexión básico tal como se muestra en la figura 4.10. De esta forma ha sido posible cursar las llamadas realizadas entre los usuarios registrados en la red, en este caso usando dos teléfonos se mantiene una comunicación de prueba cerca de nuestra

Figura 4.10: LLamada entre dos teléfonos del sistema OpenBTS implementado

BTS, evitando colocar a una distancia menor a los 20 cm. ambos celulares ya que se produciría una retroalimentación en la llamada terminando por distorsionarla.

La comunicación tiene un tiempo de duración máximo de cinco minutos pasado este tiempo será finalizada por Asterisk, además si un usuario llama a otro que no se encuentre registrado, que esté fuera del alcance de la red o haya apagado su télefono, el llamante recibirá un tono de aviso alertando de estas condiciones, todo este comportamiento se lo configura en el plan de marcado de Asterisk permitiendo establecer números de servicios específicos que el administrador desee proporcionar en la red.

5

CONCLUSIONES Y RECOMENDACIONES

CONCLUSIONES

Cumplidos finalmente los objetivos propuestos y finalizada la investigación y ejecución del trabajo de fin de titulación, se exponen las siguientes conclusiones y recomendaciones.

1. Queda demostrado mediante la práctica y experimentación, que es posible la construcción de una red telefónica móvil usando componentes de software libre y hardware de bajo costo.

2. Se ha logrado con satisfacción la integración de la tecnología GSM, VoIP, SDR, con la finalidad de despertar el interés en la utilización de las redes abiertas de telefonía celular como solución a la necesidad de comunicación en sectores rurales y escenarios de emergencia.

3. El sistema OpenBTS reduce la infraestructura especializada que utilizan las redes montadas por las operadoras móviles comerciales, minimizando el costo de implementación en un 90 % para un área de cobertura no mayor a 35 Km. con una potencia de 10 Watts.

4. El impacto positivo que se intenta generar con la penetración del teléfono móvil en zonas desprotegidas, está encaminado al uso de este dispositivo como plataforma de desarrollo en el ámbito de educación, salud, económico, y social.

5. Los equipos móviles de los usuarios, deben tener compatibilidad con el estándar GSM y la banda usada en la red, esto significa un gasto menor si se adquiere un modelo de teléfono económico o reciclado.

6. Las tarjetas SIM que han sido usadas en el sistema son propiedad de las operadoras ecuatorianas “Claro” y “Movistar”, razón por la cual cada tarjeta posee un valor de autenticación Ki ya establecido por estas operadoras, esto representa un impedimento al tratar de registrar completamente el teléfono con la red implementada.

7. El uso de la comunicación de voz sobre IP (VoIP) y las características del protocolo SIP, facilita enormemente el manejo de los terminales de usuario y la conmutación de las llamadas de móvil a móvil, prestando un servicio telefónico de buena calidad tanto en voz como en tiempo de respuesta.

8. Para facilitar el registro de los usuarios, Asterisk tiene configurada la opción de auto-crear las extensiones, y junto con el sistema ODBC permite agregar automáticamente un nuevo usuario en la base de datos de registros “sqlite3.db”,

esta característica evita realizar manualmente el registro de cada usuario en el archivo de configuración “sip.conf”.

9. La mayoría de implementaciones GSM usan un reloj de referencia de 13 MHz llamado “stratum 3” con una precisión en la frecuencia del orden de 20 ppb. El reloj que usa el equipo USRP por defecto es un oscilador de cristal a una frecuencia de 64 MHz, lo que significa que existe un error entre el reloj principal del sistema y el número de símbolos por segundo (symbol rate) que GSM requiere. Para tener un acercamiento al reloj de 13 MHz se utilizó el módulo GPSDO el cual posee una frecuencia de oscilación de 10 MHz.

RECOMENDACIONES

1. Hay que evitar utilizar el sistema de OpenBTS para realizar cualquier acto de índole fraudulenta, debido a que este presenta las herramientas suficientes como para confundir a cualquier persona que utilice las redes móviles celulares como medio de comunicación.

2. La versión de Asterisk 11 es usada en este sistema, por poseer las funciones necesarias para cada conexión entre los sistemas de red, además brinda la facilidad para que las extensiones SIP se configuren en la red de forma autónoma.

3. Para una conexión con los servicios de Google el servidor o computador debe de tener una conexión a internet y por lo menos una cuenta de google-voice activada.

4. Es recomendable estudiar previamente cada parte que compone el sistema OpenBTS para así reconocer cualquier error de funcionamiento que se produzca durante su ejecución y puesta en funcionamiento.

5. Cada llamada que se realiza mediante el protocolo SIP necesita del establecimiento de puertos de comunicación usados tanto por asterisk como por OpenBTS, se recomienda revisar la configuración del firewall si se encuentra presente y del comportamiento de asignación de puertos que se use en la red en la que estamos trabajando.

6. El sistema ODBC que permite el acceso a la base de datos de usuarios debe de poseer los permisos de usuario necesarios que faciliten a Asterisk la ejecución de llamadas.

7. Se recomienda manipular con precaución los componentes del equipo USRP, con el fin de evitar una avería que sea irreparable y que termine por arruinar el funcionamiento del sistema.

8. En vista de que se manejan varias bases de datos tanto para la configuración y el registro de usuarios en el sistema celular OpenBTS, es recomendable repasar los comandos del sistema SQL que permitan editarlas.

9. El diseño de un sistema de amplificación de potencia e implementación de enlaces entrantes, constituye un requerimiento esencial para futuros adelantos en el prototipo de la estación base BTS presentado en el presente proyecto.

Bibliografía

[1] José M. Huidobro & Rafael Conesa *Sistemas de Telefonía - 5ta Edición* , pp. 170-178, (2006). 10, 14

[2] Tomasi, Wayne *Sistemas de Comunicaciones Electrónicas - 5ta Edición*, pp. 864-901 ,(2005) 7, 18, 20

[3] Bruce A. Fette *Cognitive radio technology - 1ra Edición*, pp. 1-17 ,(2006) 17

[4] Burns, Paul *Software defined radio for 3G - 1ra Edición*, pp. 3-25 ,(2003) 18

[5] Hüseyin Arslan *Cognitive Radio, Software Defined Radio, and Adaptive Wireless Systems - 1ra Edición*, pp. 129-138 ,(2007) x, x, 17, 19, 20

[6] Leif Madsen, Jim Van Meggelen, Russell Bryant *Asterisk: The Definitive Guide - 3ra Edición*, pp. 9 - 81, (2011) x, 34, 35, 46, 48, 49, 50

[7] Tong, Z.& Arifianto M.S.& Liau, C. F. *Wireless transmission using universal software radio peripheral; Space Science and Communication,* International Conference on , vol., no., pp.19,23, 26-27 Oct. (2009),[en linea],disponible en:`http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=5352678&isnumber=5352625` 52

[8] Murota, K.& Hirade, K. *GMSK Modulation for Digital Mobile Radio Telephony,* Communications, IEEE Transactions on, vol.29, no.7, pp.1044,1050,Jul 1981,[en línea], disponible en: `http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=1095089&isnumber=23940` 20

[9] Rodríguez S. Francisco, Sáez V. Fernando *El teléfono móvil producto estelar de la red universal digital*,[en línea]. Disponible en: `http://web.dit.upm.es/~jsr/ElTelefonomovilproductoestelardelaRUDv2.2.pdf` 2

[10] ITU-T Telecommunication Standarization Sector *List of Mobile Country or Geographical Area Codes*, Ene. (2004),[en linea], disponible en:`http://www.itu.int/itudoc/itu-t/ob-lists/icc/e212_685.pdf` 33

[11] Byron Villacis, Daniela Carrillo *Nueva cara sociodemográfica del Ecuador*, [en línea]. Disponible en:`http://www.inec.gob.ec/publicaciones_libros/Nuevacarademograficadeecuador.pdf` 2

[12] Superintendencia de Telecomunicaciones del Ecuador *Evolución de telefonía móvil en Ecuador*, [en línea]. Disponible en:`http://www.supertel.gob.ec/pdf/publicaciones/revista_supertel_16_final.pdf` 2

[13] Superintendencia de Telecomunicaciones del Ecuador *Reglamento para el servicio de telefonía móvil celular*[en linea], disponible en:`http://www.conatel.gob.ec/site_conatel/?option=com_content&view=article&catid=335:todos&id=115:-reglamento-para-el-servicio-de-telefonia-movil-celular` 31

[14] Marc Kahabka *Pocket Guide for Fundamentals and GSM Testing*, [en línea]. Disponible en :http://web.itu.edu.tr/~pazarci/WandelGoltermann_gsm.pdf x, x, xii, 7, 8, 9

[15] ETSI *Mobile Station -Base Station System (MS - BSS) interface; General aspects and principles*, [en linea]. Disponible en: http://www.etsi.org/deliver/etsi_gts/04/0401/05.00.00_60/gsmts_0401v050000p.pdf x, 11, 24, 25

[16] ETSI *Mobile Station -Base Station System (MS - BSS) interface; Channel structures and Access capabilities*, [en linea]. Disponible en: http://www.etsi.org/deliver/etsi_gts/04/0403/05.01.00_60/gsmts_0403v050100p.pdf 24

[17] ETSI *Mobile Station -Base Station System (MS - BSS) interface;Data Link Layer*, [en linea]. Disponible en: http://www.etsi.org/deliver/etsi_gts/04/0405/03.01.05_60/gsmts_0405sv030105p.pdf 26

[18] ETSI *Mobile radio interface layer 3 specification*, [en linea]. Disponible en: http://www.etsi.org/deliver/etsi_gts/04/0408/05.03.00_60/gsmts_0408v050300p.pdf 27, 28

[19] ETSI *Multiplexing and multiple access on the radio path*, [en linea]. Disponible en: http://www.etsi.org/deliver/etsi_gts/05/0502/05.00.00_60/gsmts_0502v050000p.pdf 25

[20] ETSI *Channel coding*, [en linea]. Disponible en: http://www.etsi.org/deliver/etsi_gts/05/0503/05.02.00_60/gsmts_0503v050200p.pdf 26

[21] ETSI *Radio Transmission and Reception*, [en linea]. Disponible en: http://www.etsi.org/deliver/etsi_gts/05/0505/05.00.00_60/gsmts_0505v050000p.pdf

[22] Grupo GSM (gsma) *About us*, [en linea]. Disponible en: http://www.gsma.com/aboutus/gsm-technology/gsm 2

[23] Nokia Networks *Nokia GSM Architecture*, [en linea]. Disponible en: http://www.roggeweck.net/uploads/media/Student_-_GSM_Architecture.pdf x, x, 9, 10, 11

[24] J. Xiao, Liu, Dang, Zhao *Analysis and Simulation for VoIP Capacity in IEEE 802.11*, [en línea]. Disponible en:http://www.jofcis.com/publishedpapers/2012_8_19_7955_7962.pdf x, 14

[25] Jose I. Moreno, Ignacio Soto, David, Larrabeiti *Protocolos de Señalización para transporte de voz sobre IP*, [en línea]. Disponible en:http://www.it.uc3m.es/~jmoreno/articulos/protocolssenalizacion.pdf 13, 14

[26] Hira Sathu, Mohib A. Shah *Performance Comparision of VoIP codecs on IPv4 and IPv6*, [en línea]. Disponible en:http://www.ijeeee.org/Papers/093-Z00076E00126.pdf xii, 15

[27] Abul Azad *OpenBTS Implementation with Universal Software Radio Peripheral*,[en línea]. Disponible en:http://mathinstitution.com/ECE/SDR/sdr_openBTS_aazad.pdf 32

[28] *USRP N210 datasheet*,[en línea]. Disponible en:http://www.ettus.com/content/files/kb/Ettus_Networked_Series.pdf x, 53

[29] George Fredric Eichinger *Cognitive Radio Universal Software Hardware*, [en línea]. Disponible en: http://www.coe.neu.edu/Research/rcl/theses/eichinger-msthesis2012.pdf 53

[30] *daughterboard WBX*,[en línea]. Disponible en:https://www.ettus.com/product/details/WBX 30

[31] Ettus Research *Installing the GPSDO kit for USRP N200 Series & E100 Series*, [en línea]. Disponible en: https://www.ettus.com/content/files/gpsdo-kit_4.pdf 56

[32] Mälardalens Högskola *Evaluation of Software Defined Radio Platform with respect to implementation of 802.15.4 ZibBee*, [en línea]. Disponible en: http://www.idt.mdh.se/utbildning/exjobb/files/TR1155.pdf 55

[33] BURGESS, David A. & SAMRA, Harvind S. *The Open BTS Project*,[en línea].3 Ago. 2008. Disponible en:http://www.ahzf.de/itstuff/papers/OpenBTSProject.pdf[consulta: 2 Feb. 2013]. 22, 29, 30

[34] Range Networks, Inc. *OpenBTS P2.8 User´s Manual*,[en línea].Disponible en: http://wush.net/trac/rangepublic/attachment/wiki/WikiStart/SoftwareP2.8Manual.pdf 26, 32, 61

[35] Dario Flores *Manual de uso e instalación de OpenBTS*, [en linea]. Disponible en: http://wush.net/trac/rangepublic/attachment/wiki/WikiStart x, 16, 22

[36] Vasquez, J. & Santa I. & Restrepo J. *Prototipo de una Estación Celular Portátil Para atención de Emergencias*, [en línea]. Disponible en: http://wush.net/trac/rangepublic/attachment/wiki/WikiStart/Prototipo%20De%20Una%20Estaci%C3%B3n%20Celular%20Port%C3%A1til%20Para%20Atenci%C3%B3n%20De%20Emergencias.pdf 22, 34, 37

[37] Spencer Mark & Mack Allison & Cristopher Rhodes *The Asterisk Handbook*, [en línea]. Disponible en: http://bananabread.net/AsteriskHandbk_w_Bkmarks.pdf x, 14, 35

[38] Nefta Anaya *Fundamentos de Telefonía IP e Introducción a Asterisk/Elastix*, [en línea]. Disponible en: http://listas.asteriskbrasil.org/pipermail/asteriskbrasil/attachments/20130308/40376b0c/attachment.pdf 35, 48, 50

[39] Bowerman David *Asterisk Channel Driver to allow Bluetooth Cell/Mobile Phones to be used as FXO devices, and Headsets as FXS devices*, [en linea]. Disponible en: http://svnview.digium.com/svn/asterisk-addons/branches/1.6.2/doc/chan_mobile.txt?view=markup 37, 38

[40] Malcolm Davenport & Russell Bryant *A Beginners Guide to Asterisk*, [en línea]. Disponible en: https://wiki.asterisk.org/wiki/display/AST/Getting+Started 47, 48, 49

[41] ISO & IEC *Information exchange between systems - High-level data link control (HDLC) procedures*, [en linea]. Disponible en: http://webstore.iec.ch/preview/info_isoiec13239%7Bed3.0%7Den.pdf 26

ANEXOS

ANEXO A

Instalación del driver UHD

El primer paso: previo a instalar los programas GNURadio y OpenBTS es la instalación del software driver del equipo que posteriormente utilizaremos, en nuestro caso se trata del equipo USRP de marca ETTUS modelo N210.

Para poder gestionar la conexión del Hardware con los programas que vamos a

Figura A.1: Equipo USRP N210

utilizar, es necesario instalar el USRP Hard Drive UHD, que se trata de librerías que permiten el manejo de los recursos de este material.

A.1. Configuración de librerías

Para empezar dentro del sistema operativo Ubuntu, abrimos un Terminal de Linux usando la combinación de teclas Ctrl+Alt+t, donde ingresaremos como usuario root, para esto usamos los comandos "sudo su" que nos pedirá la clave del sistema, y luego usamos los siguientes comandos seguidos de un Enter:

```
# sudo apt-get install libboost-all-dev
# libusb-1.0-0-dev python-cheetah doxygen python-docutils
```

Este procedimiento instalará las librerías necesarias para configurar y compilar los archivos del paquete UHD.

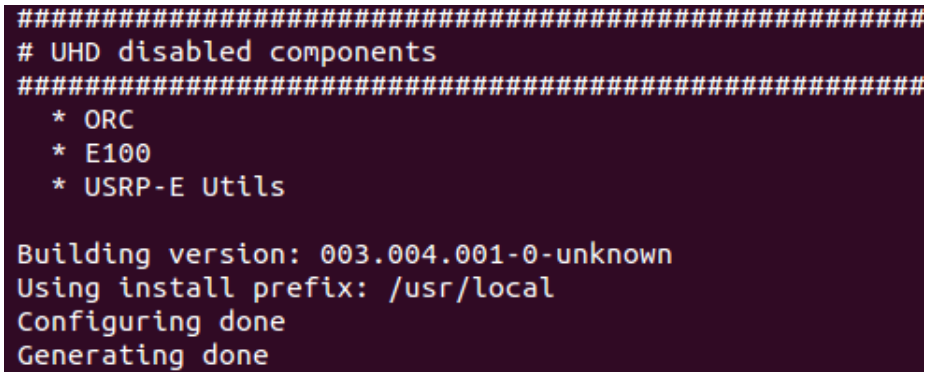

Figura A.2: Instalación librerias UHD

A.2. Configuración del controlador

El siguiente paso es descargar los archivos de configuración del driver UHD que lo podemos hacer desde la siguiente página: `https://github.com/EttusResearch/UHD-Mirror/tags`.
Aquí descargaremos un paquete comprimido en formato tar.gz, existen varias versiones de este software, desde versiones de hace un año hasta la última versión, debemos escoger una que se ajuste a nuestro fin, en este caso se utilizará la versión UHD-3.4.1 puesto que esta versión es estable y ya se encuentra "parchada"de algunos errores de compilación. En fin, ya descargado el paquete lo descomprimimos en alguna ruta o directorio de fácil acceso, y luego desde el terminal accedemos a esta carpeta, los comandos a ejecutar son:

```
# cd <uhd-repo-path>/
# cd host
# mkdir build
# cd build
# cmake ../
# make & make test
# sudo make install & sudo ldconfig
```

El primer comando “cd” permite entrar en la carpeta descomprimida que tendrá por nombre UHD-Mirror-release-003-004-001, en el caso de que la versión de UHD sea 3.4.1, una vez aquí escribimos el siguiente comando “cd host” que permite dar atributos de usuario a esta carpeta para luego crear un directorio llamado “build” para eso se usa “mkdir”, ya en este directorio podremos ejecutar el programa de creación “cmake”.

Nota: si la instalación de Ubuntu no contiene el comando de configuración cmake lo

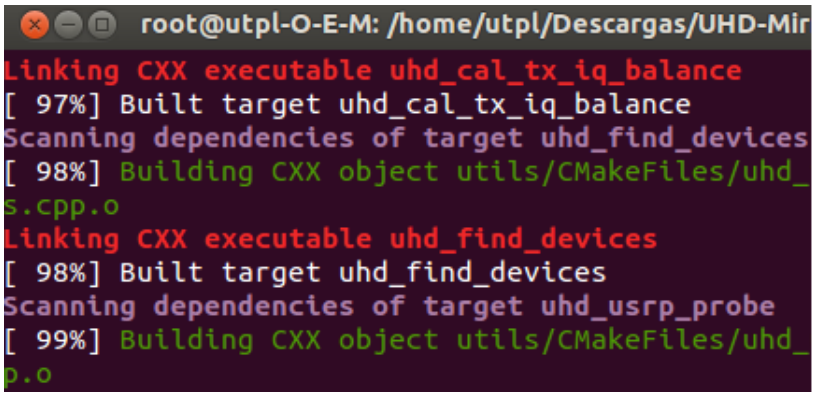

Figura A.3: Creación de archivos de configuración

podemos instalar usando el comando:

```
# sudo apt-get update ->actualizará el repositorio de
Ubuntu
# sudo apt-get install cmake ->instalará la libreria cmake
```

Utilizando el comando **cmake** se crearán los archivos que contienen la configuración del programa y los ejecutables del mismo, esta configuración se tarda un poco dependiendo de la velocidad de nuestro equipo, por tanto hasta aquí estamos en condiciones para crear programas que puedan interactuar con el hardware USRP.

Así mismo la instalación por consola nos permite realizar una prueba de nuestra

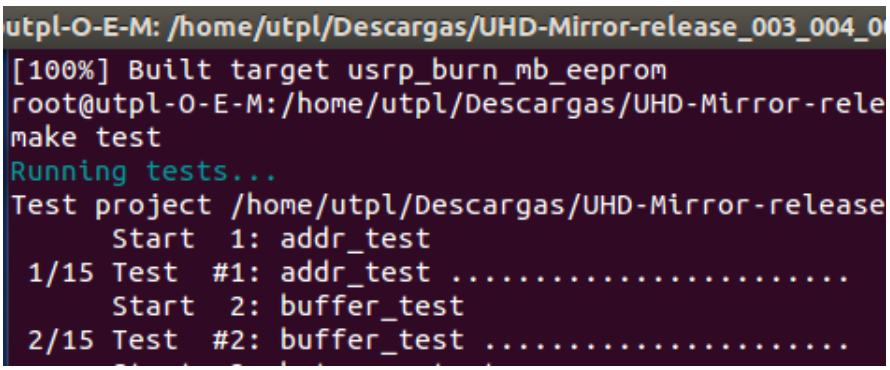

Figura A.4: Instalación completa UHD

instalación que la podemos ejecutar usando el comando "make test", esto nos asegura que nuestros procesos están listos para ser ejecutados cuando se los requiera.

Por último ejecutamos el comando "ldconfig" que creará los vínculos en memoria cache para las librerías recientemente compartidas en el directorio de instalación en la línea de comandos, es decir, que podemos ejecutar estas librerías cuando las llamemos desde un programa que contenga el directorio del driver UHD como ruta de comandos, como por ejemplo el proceso transceiver del sistema OpenBTS.

ANEXO B

Instalación del Sistema OpenBTS

OpenBTS es distribuido por Range Networks, esta empresa ha creado un sitio web donde su contenido puede ser editado por usuarios voluntarios a través de sus navegadores web. La dirección de esta página es `http://wush.net/trac/rangepublic/wiki` aquí podremos encontrar todo lo relacionado a OpenBTS tanto su funcionamiento como instalación. En las siguientes páginas se explicaran los pasos ejecutados por el autor para el levantamiento de la red móvil.

B.1. Configuración del software

OpenBTS puede, en principio, ser configurado y ejecutado en cualquier Sistema Operativo basado en Unix. Sin embargo, la mayoría del desarrollo es hecho en Ubuntu 12.04 LTS, así que a partir de este punto se considera que se poseen los conocimientos básicos de este sistema.

1. Descargar el paquete de OpenBTS: La mejor alternativa para obtener el paquete de utilidades OpenBTS es descargarlo directamente desde el repositorio de código fuente oficial, autenticándose como usuario anónimo. Para esto ejecutamos la siguiente línea de comandos, dentro de un terminal del S.O Linux.

   ```
   # svn co http://wush.net/svn/range/software/public
   ```

   Previamente a obtener el fichero del código fuente de OpenBTS, se debe crear una carpeta donde se puedan almacenar los archivos a descargar, esto se puede realizar desde el terminal de Ubuntu con el comando <mkdir openbts>en el directorio de nuestra preferencia, por ejemplo el escritorio.
   Luego de descargado OpenBTS, procedemos a instalar cada componente, el primero será el transceiver, para esto nos dirigimos al directorio de descarga /public/ y utilizamos los siguientes comandos:

```
# cd openbts/trunk  # par autoreconf -i
# ./configure
# Make
```

A continuación, se procede con la configuración de los controladores de software transceiver apropiados para el USRP N210. Para los modelos actuales del hardware ETTUS Research (USRP2, B100, N200, y E100 series) se usan las librerías UHD (Universal Hardware Device), la instalación de estas librerías es descrita en el anexo A, estas permiten ejecutar los siguientes comandos.

```
# cd openbts/trunk
# autoreconf -i
# ./configure -with-uhd -with-resamp
# Make
```

El siguiente paso es la ejecución del transceiver configurado para el USRP N210, para esto nos dirigimos al directorio raíz de openbts, desde la terminal, este directorio es <openbts/trunk/apps>. Y se utiliza el siguiente comando para realizarlo.

```
# ln -s ../Transceiver52M/transceiver
```

B.2. Configuración de OpenBTS

Con el OpenBTS construido, se debe realizar una configuración inicial para el funcionamiento del sistema en modo básico. Existen dos archivos claves que deben ser creados para que todo funcione.

El primero es el archivo <OpenBTS.db>, que contiene una base de datos que almacena toda la configuración hecha a OpenBTS. Debe ser creado e instalado en el directorio </etc/OpenBTS>, el cual todavía no existe. Así que, se lo pude crear de la siguiente forma desde el directorio de OpenBTS.

```
# sudo mkdir /etc/OpenBTS
# sudo sqlite3 -init ./apps/OpenBTS.example.sql
```

Para comprobar que se realizó completamente la instalación de la base de datos, podemos comprobarlo usando la siguiente sentencia.

```
# sqlite3 /etc/OpenBTS/OpenBTS.db .dump
```

Lo cual mostrará una lista de variables de configuración, esto quiere decir que la instalación se ha completado correctamente.

B.3. Ejecutando OpenBTS

En este punto, se puede realizar una revisión del funcionamiento de OpenBTS, ejecutando la siguiente orden dentro de la carpeta donde se encuentra la instalación.

```
# cd apps
# sudo ./OpenBTS
```

Al realizar esto, nuestro hardware USRP se convertirá en la Estación Base Transceiver (BTS) GSM, que despliega la interface de aire Um. Ahora es posible desde un teléfono escanear las redes móviles disponibles, desde donde se podrá observar la red existente con nombre UTPL o Test-TMLN, que es la que hemos montado. Si tratamos de ingresar a esa red desde el mismo teléfono seremos rechazados, debido a que OpenBTS por defecto solo permite la conexión con terminales registrados en su base de datos.
Por otro lado, aún no está siendo ejecutado nuestro servidor de registro llamado (Sipauthserve), por tanto ningún teléfono podrá ingresar. Es necesario entonces, abordar algunas variables de configuración de OpenBTS a las que se puede acceder únicamente desde la interfaz de comandos OpenBTSCLI, que se inicia desde el siguiente directorio de OpenBTS.

```
# cd apps
# sudo ./OpenBTSCLI
```

Al ejecutarse la consola de configuración OpenBTS>, en esta podemos introducir la opción <config>que mostrará una lista de opciones de configuración de OpenBTS. Existen múltiples opciones de configuración, que modifican el comportamiento del sistema las cuales se describen en la pagina de Range Networks http://wush.net/trac/rangepublic/wiki/openBTSConfig. Sin embargo, para la operación en modo básico necesitamos solamente modificar las más importantes que se enumeran a continuación:

- GSM.Radio.Band Esta variable contiene la banda de frecuencia GSM en la que trabaja el USRP
- GSM.Radio.C0 este parámetro es el ARFCN. Debe ser el apropiado para la frecuencia

- Control.LUR.OpenRegistration es la opción que permite rechazar o aceptar el registro de los teléfonos en la red. Facilita la compilación e inicialización del sistema.

Una instrucción básica que debemos modificar, es la forma en que acceden a la red los terminales móviles sin que tengan dificultad en su ingreso. Para realizar esto se utiliza la siguiente sentencia:

```
# config Control.LUR.OpenRegistration .*
```

Lo que brinda un acceso sin restricción a cualquier estación móvil MS cercana a la red OpenBTS. Se debe tener precaución con interferir los teléfonos de usuarios que no desean ingresar a la red porque serán conectados a la Estación Base de nuestro sistema.

B.4. Creación del servidor de registro (Sipauthserve)

OpenBTS depende de la instalación del servidor de autorización SIP Sipauthserve para efectuar la conmutación de tráfico en la red. Este servidor es indispensable para obtener el funcionamiento completo del sistema. A su vez, el servidor necesita inicialmente de que configuremos primeramente la base de datos del Registro de Abonados que se lo maneja desde el archivo <sqlite3.db>, usando las siguientes líneas:

```
# subscriberRegistry/trunk/configFiles/
# sudo mkdir -p /var/lib/asterisk/sqlite3dir
# sudo sqlite3 -init subscriberRegistryInit.sql
sudo mkdir /var/run/OpenBTS
```

A continuación, se procede a la creación de los archivos ejecutables de Sipauthserve. Dentro de estos archivos existe un proceso demonio que se ejecuta continuamente para facilitar los servicios de autenticación en el protocolo SIP. Inicialmente, se ejecuta dentro del directorio <subscriberRegistry>el comando de creación <make>que crea los ejecutables del servidor Sipauthserve. (from svn root)

```
# subscriberRegistry/trunk
make
```

Luego, continuando con la creación de este servidor debemos configurar el acceso a la base de datos del servidor <sipauthserve.db>, por lo que usamos la siguiente sentencia para crear la base de datos dentro del directorio </etc/OpenBts>.

```
# sudo sqlite3 -init sipauthserve.example.sql
/etc/OpenBTS/sipauthserve.db ".quit"
```

De esta forma, tenemos la posibilidad de ejecutar sin problemas el servidor sipauthserve en el directorio de <subscriberRegistry>, escribiendo en un terminal Linux, lo siguiente:

```
# sudo ./sipauthserve
```

El servidor no posee una interfaz de programación por comandos, por lo que podremos observar un mensaje de ejecución como por ejemplo:

```
ALERT 139639310980928 sipauthserve.cpp:214:main:
./sipauthserve (re)starting
```

Si todos los pasos anteriores se completaron satisfactoriamente, entonces el sistema OpenBTS está listo para ser ejecutado y levantado, con la ayuda del hardware N210. Lo siguiente que se debe hacer es conectar los equipos, PC computador personal y USRP entre sí, por medio de un cable de red entre las interfaces Gigabit-Ethernet para realizar la comunicación. Es necesario seguir una serie de pasos antes de ejecutar una llamada entre dos terminales móviles registrados, los cuales se describen en el Anexo C.

ANEXO C

Configuración de Asterik Real-Time

C.1. Compilación de sqlite3 y asterisk

Habitualmente asterisk y la libreria sqlite3 son instaladas en el sistema gracias a la utilidad del sistema "apt" que permite la descarga e instalación de estos programas desde el repositorio de Linux. Si realizamos la instalación de Asterisk para que funcione en tiempo-real debemos asegurarnos de remover completamente estos dos paquetes, con el fin de que permitan la construcción de los nuevos. Se ejecutan los siguientes comandos en la terminal de Ubuntu.

```
# sudo apt-get remove asterisk sqlite3
# sudo apt-get autoremove
```

C.2. Instalando el software apropiado

Una vez que eliminamos la distribución de los paquetes anteriores, procedemos a realiza la configuración e instalación personalizada de los nuevos programas. A continuación se menciona los pasos para su efecto.

- Descargar sqlite3, desde el repositorio oficial [1], luego de descargar el paquete lo descomprimimos y nos dirigimos a su directorio desde la terminal; aquí configuramos, creamos e instalamos, usando las siguientes sentencias de manera secuencial.

  ```
  # configure, make, make install
  ```

  Sqlite3 debe ser compilado con el atributo definido por

[1] http://www.sqlite.org/download.html

SQLITE_ENABLE_COLUMN_METADATA, por lo que lo configuramos usando
"./configure CFLAGS=DSQLITE_ENABLE_COLUMN_METADATA -02"; el siguiente paquete que funciona como editor de base de datos los instalamos por apt-get.

```
# sudo apt-get install libsqlite3-dev
```

- Ahora necesitamos el driver que permite la conexión con las bases de datos implicadas en el sistema, se trata del programa ODBC Open Database Connectivity que brinda conectividad a las bases de datos de cualquier sistema de gestión de datos; y del driver mismo "sqliteodbc", al primero lo descargamos desde su página oficial `http://www.unixodbc.org/`, y el segundo desde el repositorio `http://www.ch-werner.de/sqliteodbc/`, ambos los configuramos e instalamos tal como a sqlite3, usando las siguientes referencias.

  ```
  # configure, make, make install
  ```

  Hasta este punto hemos configurado el sistema de gestión de base de datos. Lo siguiente es conectarlo con asterisk.

C.3. Reinstalando Asterisk

Las versiones de Asterisk se pueden descargar gratuitamente desde la página principal de la empresa "Digium" `http://www.asterisk.org/`, una vez descargada la descomprimimos e instalamos normalmente. La diferencia para que funcione junto a ODBC es en el punto de su configuración, por lo que de acuerdo a la página de Range Networks debemos configurar con la siguiente orden: "./configure –disable-xmldoc".
Luego de esto realizamos la selección de los paquetes que deseamos se ejecuten junto con Asterisk, para esto usamos la opción: "make menuselect". Veremos un menú gráfico, como el de la figura C.1 en el seleccionamos los elementos Sqlite3 y ODBC, dentro de los módulos "aplications", "resource modules" y "dialplan functions".
Debemos asegurarnos que no existen campos ODBC o sqlite3 seleccionados con los signos "XXX" en el menú de configuración. Si observamos "XXX" para cualquiera de los componentes sqlite3 - odbc, quiere decir que los paquetes no se encuentran correctamente instalados. Podemos comprobar los errores en el fichero config.log que crea asterisk en su instalación. Por otro lado si no tenemos problema debemos seleccionar los siguientes componentes:

- En "applications" seleccionamos, "app_db"

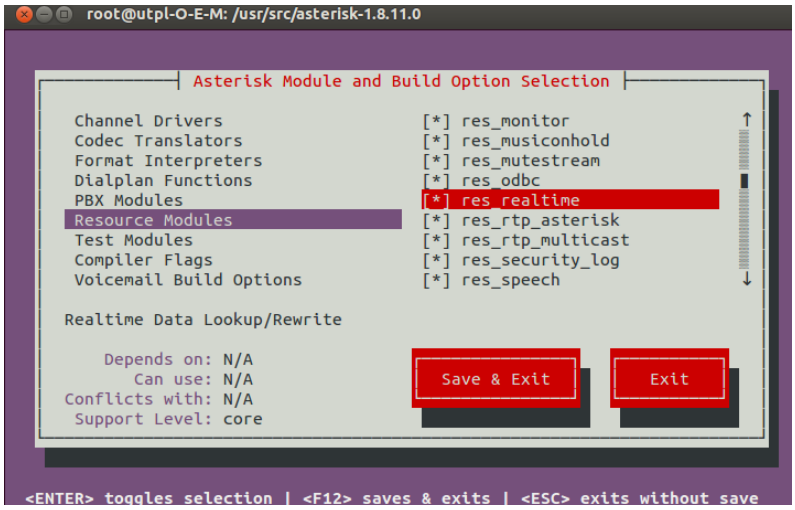

Figura C.1: Menú de selección de componentes Asterisk

- En "Dialplan Modules", seleccionamos, "func_odbc"
- En "Resource Modules", seleccionamos, "res_config_odbc", "res_odbc" y "res_realtime". Si existe "XXX" en "res_config_sqlite" significa que no se ha tomado el componente del viejo sqlite no del sqlite3.

Además dentro del menú aprovechamos para comprobar que en "Codec Translators" se encuentran seleccionados "codec_gsm", "codec_alaw" y "codec_ulaw".

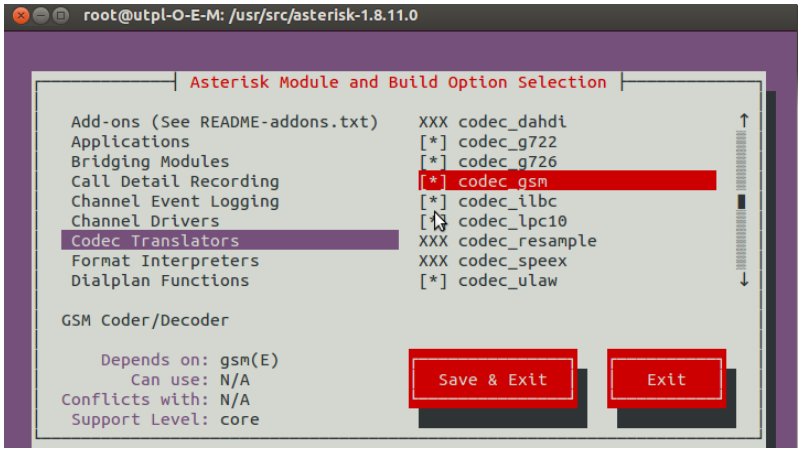

Figura C.2: Menú de selección de códecs

Luego de todo esto estamos listos para instalar completamente la central Asterik, usando los comandos "make & make install".

C.4. Archivos de configuración ODBC

Luego de nuestra instalación de Asterisk Real-time, configuramos algunos parámetros de ODBC dentro de los ficheros "odbcinst.ini" y "odbc.ini".

- En el primero declaramos los módulos con los que asterisk puede gestionar las bases de datos, por lo que editamos lo siguiente:

```
[SQLite3]
Description=SQLite3 ODBC Driver
Driver=/usr/local/lib/libsqlite3odbc.so
Setup=/usr/local/lib/libsqlite3odbc.so
Threading=2
```

- En el fichero localizado en /etc/odbc.ini declaramos el nombre de nuestra base de datos junto con el directorio en el que se aloja, en el caso de OpenBTS la base "sqlite3.db" es donde se encuentran todos los datos de los usuarios registrados. Entonces las líneas quedan como se muestra a continuación:

```
[asterisk]
Description=SQLite3 database
Driver=SQLite3
Database=/var/lib/asterisk/sqlite3dir/sqlite3.db
Timeout=2000
```

  De esta forma establecemos todos los datos de configuración con el servidor ODBC.

C.5. Archivos de configuración en Asterisk

Ya dentro de asterisk debemos tener habilitados algunos parámetros confinados a la ejecución de los módulos de conexión ODBC, lo cual permita el acceso a los datos de nuestra base "sqlite3.db".
Estos archivos se los puede editar con el comando "gedit" desde el terminal, usando el directorio "/etc/asterisk". El primer archivo a configurar es "modules.conf", este habilita la carga automática de los módulos odbc, su contenido debe quedar como sigue:

```
[modules]
autoload=yes
preload =>res_odbc.so
preload =>res_config_odbc.so
; noload =>res_config_odbc.so
```

El siguiente archivo se trata de “extconfig.conf” el cual señala a asterisk a usar el servicio de odbc para el funcionamiento del protocolo SIP en tiempo real. El archivo queda:

```
[settings]
sipusers =>odbc,asterisk,sip_buddies
sippeers =>odbc,asterisk,sip_buddies
```

A esta altura podemos declarar un contexto de conexión a la base de datos, este contexto define si se habilita la conexión a la base de datos, en este caso asterisk llamará al contexto [asterisk]el cual ya está declarado en el archivo “odbc.ini“. El ejemplo queda como sigue:

```
[asterisk]
enabled =>yes
dsn =>asterisk
pre-connect =>yes
```

Por último establecemos la función que llamará a el gestor ODBC para que ejecute cualquier orden en lenguaje SQL dentro del “dialplan” de asterisk.

```
[SQL]
dsn=asterisk
readsql=$ARG1
```

En este caso la función que se ejecutará en el dialplan sera “ODBC_SQL”, esta función siempre debe de ser declarada con mayúsculas de otra forma asterisk no la reconocerá en su plan de marcado. Si todo este proceso se ha completado satisfactoriamente entonces tenemos camino libre para usar la configuración de asterisk en tiempo-real; es decir, podremos seleccionar cualquier dato dentro de la base de datos que registra los usuarios en la red OpenBTS, tal como su IMSI, extensión, puerto, dirección ip, códec, etc.

ANEXO D

Proceso de Conexión y Ejecución

D.1. Conexión del equipo USRP N210

Los siguientes son los pasos recomendados para realizar una correcta conexión del equipo USRP con el computador, y los dispositivos usados en el sistema.

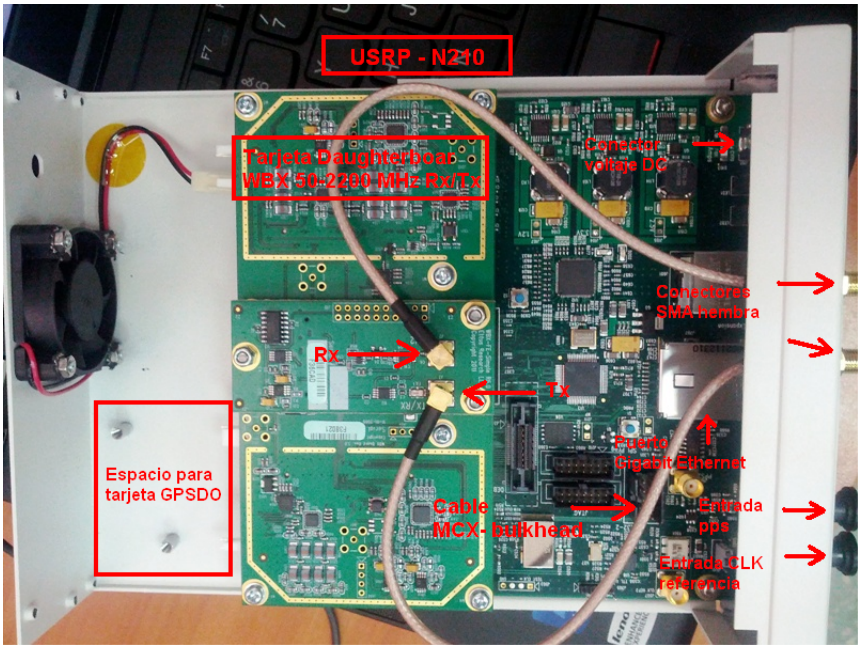

Figura D.1: Hardware de equipo N210

1. Preparar el USRP N210, conectamos la tarjeta (daugtherboard) WBX con la motherboard FPGA utilizando las ranuras de energía, y puenteamos mediante los cables MCX los conectores de salida y entrada RF de la daughterboard con los conectores bnc de la parte frontal del N210 como indica la figura D.1.

2. Luego podemos realizar la conexión del módulo GPSDO que puede servir como un reloj de referencia de 10 MHz, para realizar esta conexión se coloca

el módulo en el espacio indicado en la figura D.2 y se procede a conectar los respectivos cables tanto de alimentación como comunicación.

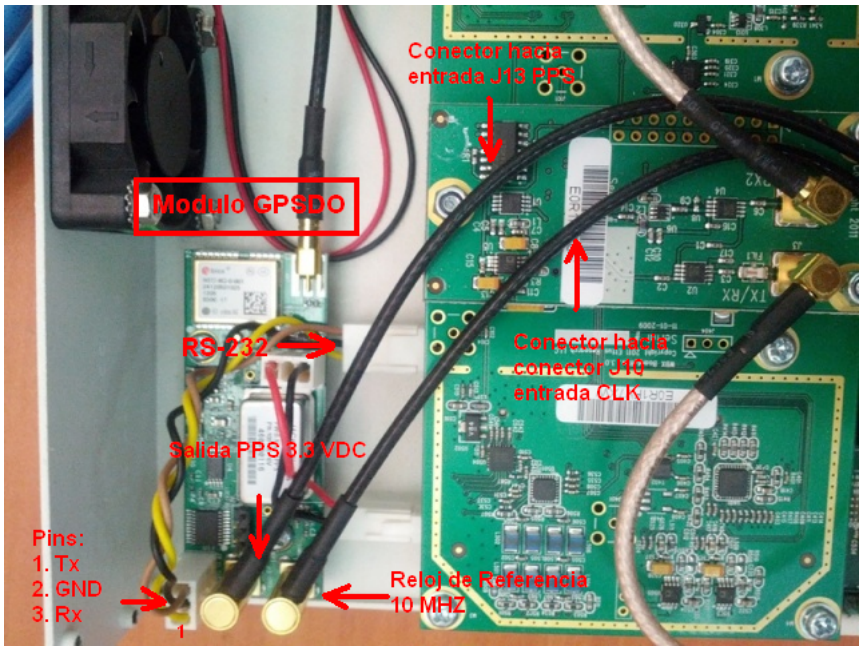

Figura D.2: Módulo GPSDO

3. Integramos dos antenas VERT900 a los conectores rf1 y rf2 del radiotransceiver.

4. Energizamos el equipo USRP desde una fuente de voltaje continuo de 9 Voltios - 3 Amperios. Podemos utilizar el adaptador de 110 voltios alternos a voltaje continuo que se incluye en el paquete ETTUS.

5. Luego conectamos mediante el cable Ethernet la interfaz del computador Personal PC, con la interfaz de red que posee el USRP.

6. Revisamos las conexiones Ethernet, de Internet y de Bluetooth presentes en el computador, configurando cada interfaz con los pasos anteriormente descritos. Lo que indica que la conexión básica del sistema esta listá y que podremos arrancar los servidores **OpenBTS** y **Asterisk**.

D.2. Inicio del Sistema

1. Antes de iniciar el sistema debemos asegurarnos de que nuestras bases de datos con los registros de usuarios "sqlite3-db" estén totalmente vacías, por lo que utilizando cualquier editor de bases de datos, y junto con el uso de

comandos SQL podemos borrar la información de IMSI o estaciones móviles que deseamos purgar de nuestro sistema. Por ejemplo si deseamos borrar el

Full View | Item View | Script Output

	id	exten	dial
1	1	2100	IMSI001010000000000
2	2	0001003	IMSI740020120051383
3	3	1001	IMSI740010137456407

Figura D.3: Base de datos con usuarios registrados

1001 ejecutamos la siguiente frase: DELETE FROM dialdata_table WHERE id = 3.

2. Abrimos un terminal Linux en el Computador, y ejecutamos el servidor <Sipauthserve>desde el directorio </subscriberRegistry/trunk>.

3. Ejecutamos un nuevo terminal Linux usando Ctrl+T, para ser usado por el sistema <OpenBTS>, entramos al directorio </openbts/ trunk/apps>, seguido en otro terminal ejecutamos también la aplicación <OpenBTSCLI>, desde el mismo directorio.

4. Iniciamos también el servidor Asterisk, desde una consola ejecutamos los comandos: asterisk -vvvvvvc, lo cual indica que asterisk mostrara con detalle o en modo verbosity de nivel 6 su ejecución.

5. Encendemos los teléfonos móviles con tarjetas SIM previamente instaladas. Se recomienda que estas tarjetas no pertenezcan a proveedores móviles locales. Si el teléfono no accede automáticamente a la red, debemos seleccionar que busque de forma manual las redes móviles locales, donde registrará la red OpenBTS y podrá ingresar a ella.

6. Realizamos una llamada de prueba, para comprobar la conexión de voz. Marcamos al número 600 desde el teléfono para llamar al servicio “echo”. Podremos escuchar nuestra propia voz en el terminal.

7. Si ejecutamos con satisfacción esta prueba, quiere decir que el sistema está listo.

ANEXO E

Configuración SIP

E.1. sip.conf

```
; Opciones de Configuración del Protocolo SIP (Session Initiation Protocol) en Asterisk
; Autor: Juan Pablo Tene,
;----------------------- General settings----------------------------------------------
[general]
context=default
type=peer
insecure=very
canreinvite=no
addr=127.0.0.1
allowguest=no

udpbindaddr=0.0.0.0:5060    ; Dirección IP que enlaza paquetes UDP
bindaddr=127.0.0.1:5060     ; Dirección opcional para enlace local

disallow=all                ; Deshabilita todas los codecs
allow=ulaw                  ; Habilita el codec ulaw
allow=alaw ; codec alaw
allow=gsm ; Activa el codec gsm

relaxdtmf=yes

progressinband=yes           ; Puede generar señalización en la banda usada
 ; Existen las opciones: yes, no, never
 ; Se usa never para nunca usar señalización in-band

dtmfmode = auto              ; El modo de señalización es por Default: rfc2833
                             ; Otras Opciones:
                             ; info : SIP INFO messages (application/dtmf-relay)
                             ; shortinfo : SIP INFO messages (application/dtmf)
                             ; inband : Inband audio (requires 64 kbit codec -alaw, ulaw)
                             ; auto : Usa rfc2833 si es necesario, o elige otra opción
use_q850_reason = yes


autocreatepeer=persist         ; Permite a cualquier usuario registrarse dentro de la red
                               ; sin autenticación previa. Ayuda en gran medida a OpenBTS
                               ; tomar el concepto de red abierta.
                               ; Puede usar la opción "yes", pero al reiniciar el plan de
                               ; marcado los usuarios se borraran.
```

```
                                ; Si se usa la opción "persist", los pares creados no son
                                ; borrados en el reinicio.

;--------------------------------------- NAT SUPPORT ------------------------

externaddr = 12.34.56.78:9900          ; define la ip publica usada en internet

icesupport = yes ; habilita el soporte de ICE (Interactive Connectivity Establishment)

;--------------------------------------- REALTIME SUPPORT ------------------------
rtcachefriends=yes  ; Añade usuarios sip del tipo "friend" a la lista interna de
; la configuración de asterisk realtime

rtupdate=yes ; Envia actualizaciones de registro a la base de datos

ignoreregexpire=yes ; No se toma en cuenta si la información de la base de datos ha expirado
```

ANEXO F

Plan de Marcado - Dialplan

A continuación se explica el plan de marcado (dialplan) y los contextos usados en el mismo. El dialplan se encuentra alojado dentro del archivo "extensions.conf".

F.1. extensions.conf

El primer contexto que debemos modificar es el "general" que describe las funciones del plan de marcado desde la consola de Asterisk, brindando la facilidad de guardar los cambios hechos desde consola y de borrar las variables globales en el reinicio del dialplan.

```
; Configuración del Dialplan (Plan de Marcado) en Asterisk
; Autor: Juan Pablo Tene,
;----------------------- General settings----------------------------------------
[general]
static=yes
writeprotect=no
autofallthrough=yes
clearglobalvars=yes

; El contexto demo, tiene por finalidad ejecutar la prueba de eco, al marcar desde un
; teléfono el número 600.

[demo]
exten => 600,1,Playback(demo-echotest)
exten => 600,n,Echo
exten => 600,n,Playback(demo-echodone)
exten => 600,n,Goto(s,6)

; El siguiente contexto es el que refleja toda la configuración realizada en el fichero
; sip.conf, si es eliminado entonces se borrara también la configuración SIP.

[default]
include => demo
include => phones
include => google
```

```
include => celular

; Para conmutar las llamadas hacia otras operadoras móviles se usa el contexto celular,
; funciona cuando se marca un numero de celular desde OpenBTS, primeramente verifica el
; estado del teléfono Gateway y realiza una prueba lógica para asegurarse que el teléfo-
; no este libre y pueda realizar una llamada, luego si marca el número caso contrario
; corta la llamada e indica el estado de fuera de servicio.

[celular]
exten => _X.,1,MobileStatus(${EXTEN},stado)
exten => _X.,2,GotoIf($["${stado}" = "2"]?3:5)
exten => _X.,3,Dial(Mobile/GT-S6102/${EXTEN},15)
exten => _X.,4,Hangup
exten => _X.,5,NoOp(El estado es ${DIALSTATUS})
exten => _X.,6,Playback(ss-noservice)
exten => _X.,7,Hangup

; Los Estados de MovileStatus pueden ser:
; 1 = Desconectado. El dispositivo esta fuerar de rango o apagado,etc.
; 2 = Conectado y en estado espera, esta Libre
; 3 = Conectado y llamando. Ocupado
; Como ya se explicado Asterisk puede conectarse al servicio de Google para realizar
; llamadas VoIP, estas llamadas pueden ser de entrada o salida, si un número es marcado
; desde una cuenta google conectada con Asterisk, entonces sera receptada por la exten-
; sion s, luego la descolgara y enviara a la extensión correspondiente.

[google]
include => phones
exten => s,1,NoOp()
 same => n,Wait(1)
 same => n,Answer()
 same => n,SendDTMF(1)
 same => n,Dial(SIP/${EXTEN},20)

exten => 100,1,Dial(Motif/google/jptene@utpl.edu.ec,,r)

; Pero si se quiere realizar una llamada a una cuenta google desde OpenBTS se debe de
; marcar la extensión 100 la que marcará la cuenta configurada previamente.
; Al contexto phones le compete toda la conmutación que se realice dentro de la red
; OpenBTS, este comunica las llamadas que inicien marcando el número de 4 digitos asig-
; nado los teléfonos celulares registrados. Utiliza la función ODBC_SQL cuando se marca
; el número que puede estar entre 1001 a 1012, lee el número de IMSI que contiene la
; tabla dialdata_table en función de la extensión llamada. Una vez que tiene el IMSI lee
; la dirección IP que le corresponde a ese número, asigna como CALLERID el número que
; inicia la llamada y ejecuta esa llamada con los datos anteriores.

[phones]
;carga d datos
;switch => Realtime/[phones]@
include => default
include => google
icclude => celular

exten => _Z.,1,Set(Name=${ODBC_SQL(select dial from dialdata_table where exten=${EXTEN})})
exten => _Z.,n,Set(IPAddr=${ODBC_SQL(select ipaddr from sip_buddies where name=${Name})})
exten=>_Z.,n,Set(num=${ODBC_SQL(selectexten from dialdata_table where dial=${CALLERID(num)})
exten => _Z.,n,Set(CALLERID(name)=num)
exten => _Z.,n,Set(STOP=0)
exten => _Z.,n,GotoIf($[[${LEN(${Name})} = ${STOP}]]?STOPCALL)
exten => _Z.,n,NoOp(${CALLERID(name)})
```

```
exten => _Z.,n,NoOp(${CALLERID(num)})
exten => _Z.,n,Dial(SIP/${Name}@${IPAddr}:5062,,r)
exten => _Z.,n,Set(TIMEOUT(absolute)=180)
exten => _Z.,n,NoOp(El estado es ${DIALSTATUS})
exten => _Z.,n,NoOp(El estado es ${DIALSTATUS})
exten => _X.,n,Set(s-${DIALSTATUS})

; Se utiliza una variable llamada STOP para comprobar que la variable name contenga
; un número de IMSI, si este es igual a 0 entonces la llamada no se podrá realizar
; en vista de que ese usuario no existe. Entonces el plan se dirige a la extension
; STOPCALL donde se colgara la llamada y mostrara un mensaje que indica el número que
; no esta registrado

exten => s,n(STOPCALL),NoOp(El número marcado no esta registrado)
same => n,Playback(silence/2)
same => n,Set(TIMEOUT(absolute)=15)
same => n,Answer()
same => n,MusicOnHold()
same => n,Hangup()

; Si se realiza la llamada se inicia la etiqueta DIALSTATUS la cual obtiene el estado
; de la llamada en ese momento puede presentarse los estados:
; CANCEL, cuando la llamada ha sido cancelada por el emisor.
; NOASWER, sin respuesta.
; BUSY, ocupado.
; CONGESTION, si existe un exceso de peticiones de llamada.
; CHANUNAVAIL, cuando el teléfono ha sido apagado o desconectado.
; y se inicia un contador que limita la llamada a 3 minutos o 180 segundos, en el caso
; de que el estado sea ocupado, cancelado, no responda la llamada esta se colgara.

exten => s-CANCEL,1,Hangup()
exten => s-NOANSWER,1,Hangup()
exten => s-BUSY,1,Busy(30)
same => n,Hangup()

; Si DIALSTATUS es congestión, se esperarán 10 segundos, luego de esto, si no cuelga,
; se enviará la llamada al estado no disponible CHANUNAVAIL.

exten => s-CONGESTION,1,Congestion(10)
same => n,NoOp(la extensión esta congestionada)
same => n,Goto(s-CHANUNAVAIL,1)
same => n,Hangup()

; Si DIALSTATUS entrega el valor CHANUNAVAIL, o si DIALSTATUS entrega CONGESTION
; se eliminan los datos de ese valor IMSI ya que el estado indica que el usuario
; se a desconectado o esta fuera de rango de cobertura

exten => s-CHANUNAVAIL,1,NoOp(EL teléfono esta fuera de rango)
same => n,Playback(silence/2)
same => n,Playback(ss-noservice)
same => n,Goto(s-${EXTEN},1)

exten => s-${EXTEN},1,Set(OBDC_SQL(DELETE FROM sip_buddies WHERE callerid = ${EXTEN}))
same => n,Hangup()
```