UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

PROGRAMA DE MAESTRÍA Y DOCTORADO EN INGENIERÍA

ELÉCTRICA – TELECOMUNICACIONES

Implementación de una radiobase UMTS 3G utilizando Radios Definidos por Software

TESIS

PARA OPTAR POR EL GRADO DE:

MAESTRO EN INGENIERÍA

PRESENTA:

PETER MUÑOZ HERNÁNDEZ

TUTOR PRINCIPAL

VICTOR RANGEL LICEA, FACULTAD DE INGENIERÍA

CD. MX. SEPTIEMBRE 2018

JURADO:

Presidente: Dr. Matías Maruri José María

Secretario: Dr. Daniel Enrique Ceballos Herrera

Vocal: Dr. Rangel Licea Víctor

$1^{er.}$ Suplente: Dr. Gutiérrez Castrejón Ramón

$2^{do.}$ Suplente: Dr. Gómez Castellanos Javier

Lugar o lugares donde se realizó la tesis:

México, CDMX, UNAM, Facultad de Ingeniería.

TUTOR DE TESIS:

Dr. Víctor Rangel Licea

_____

FIRMA

AGRADECIMIENTOS

A la Universidad Nacional Autónoma de México… Por haberme brindado la oportunidad de complementar mi carrera profesional, en una de las mejores universidades de América. Por mi raza hablará el espíritu, eternamente gracias UNAM.

Al Consejo Nacional de Ciencia y Tecnología… Por el apoyo económico que me brindó, sin el cual me hubiera sido imposible cursar la maestría.

A mi tutor… Por todo su apoyo, consejos, enseñanzas, y por haber depositado en mí su confianza.

Al claustro de profesores del posgrado… Por sus excelentes enseñanzas y continua dedicación en cada lección de clases.

A mi esposa… Por estar siempre, por ser mi compañera de vida en las buenas y malas, y por todo el amor que me ha brindado.

A mis padres y mi familia… Por haberme ofrecido una buena educación, y aunque están lejos, siempre me brindan apoyo y cariño.

A mis amigos… Por su compañía, por su ayuda, por su cariño, y por ser para mí como una familia en México.

A México… Por haberse convertido en mi segunda patria.

RESUMEN

Con la constante evolución de las comunicaciones móviles, desde principios de los años 90 ha venido ganando terreno el concepto de Radios Definidos por *Software* (SDR, por sus siglas en inglés), introduciéndose dentro de los sistemas de radiocomunicaciones convencionales. El uso de las plataformas SDR permite reducir los componentes de *hardware*, implementando muchas de las funciones en *software* para el procesado digital de las señales.

Últimamente se ha venido utilizando el *software* libre en el sector de las telecomunicaciones como herramienta para desarrollar infraestructuras de telefonía celular. El proyecto OpenBTS (Open Base Transceiver Station) es basado en este tipo de software, y utiliza dispositivos programables para implementar una radiobase de telefonía celular.

También existe una necesidad creciente de establecer comunicaciones en lugares donde no hay servicio de telefonía celular, ya sea por encontrarse en zonas geográficas de difícil acceso, o por tener un número muy reducido de usuarios.

En este trabajo se presenta un sistema basado en SDR, para la implementación de una red de telefonía celular UMTS (Universal Mobile Telecommunication System) de costo relativamente bajo, utilizando el equipo de radio programable USRP (Universal Software Radio Peripheral) N210 y un ordenador externo.

Los componentes que conforman la red son de código abierto y se han instalado en la distribución de Linux Ubuntu 16.04 LTS. Estos componentes principales, *SIPAuthServe* y *OpenBTS-UMTS*, permiten emular una red UMTS operativa, usando una interfaz aérea WCDMA (Wideband Code Division Multiple Access) para la comunicación con los terminales móviles.

La potencia de salida del USRP N210 tiene como máximo valor 100 mW (20 dBm). Tomando en cuentas las mediciones de RSSI (Received Signal Strength Indicator) realizadas en la zona de la implementación, se hace una propuesta y simulación del alcance que tendría la radiobase 3G creada, aumentando la potencia para ampliar la cobertura del servicio.

ABSTRACT

By the constant evolution of mobile communications, since the beginning of the 90s, the concept of Software-Defined Radio (SDR) has been improved, introducing itself into conventional radiocommunication systems. The use of SDR platforms allows the reduction of hardware components, implementing many functions in software for the digital processing of signals.

Lately, free software has been used in the telecommunications sector, as a tool to develop cellular telephony infrastructures. The OpenBTS (Open Base Transceiver Station) project is based on this type of software and uses programmable devices to implement a cellular base station.

There is also a growing need to establish communications in places that have not cell phone service, either because they are in geographies that are difficult to access, or because they have a very small number of users.

This paper presents a system based on SDR, for the implementation of a UMTS (Universal Mobile Telecommunication System) cellular network of relatively low cost, using the programmable radio equipment USRP (Universal Software Radio Peripheral) N210 and an external computer.

The components that make up the network are open source and have been installed in the Ubuntu Linux 16.04 LTS distribution. These main components, SIPAuthServe and OpenBTS-UMTS, make it possible to emulate an operational UMTS network, using a WCDMA (Wideband Code Division Multiple Access) interface for communication with mobile terminals.

The power output of the USRP N210 has a maximum value of 100 mW (20 dBm). Considering the RSSI (Received Signal Strength Indicator) measurements made in the implementation area, a proposal and simulation is made of the scope that the 3G base station would have, increasing the power to extend the service coverage.

ÍNDICE

LISTA DE FIGURAS

LISTA DE TABLAS

GLOSARIO

2.5G	2.5 Generation
3G	Third Generation
3GPP	Third-Generation Partnership Project
4G	Fourth Generation
5G	Fifth Generation
ACC	Access Control Class
ADC	Analog-To-Digital Conversion
AKA	Authentication and Key Agreement
AMPS	Advance Mobile Phone Service
API	Application Programming Interface
APN	Access Point Name
ARFCN	Absolute Radio Frequency Channel Number
ART	Answer To Reset
AT&T	American Telephone & Telegraph
AUTN	AUthentication TokeN
AV	Authentication Vector
BB	BaseBand
BoD	Bandwidth on Demand
BTS	Base Transceiver Station
CDMA	Code Division Multiplex Access
CK	Ciphering Key
CN	Core Network
CPU	Central Processing Unit
CS	Coding Scheme
CS	Circuit Switching
DAC	Digital-To-Analog Conversion
D-AMPS	Digital AMPS
DDC	Digital Down Conversion
DL	DownLink
DNS	Domain Name Server
DUC	Digital Up Conversion
EDGE	Enhanced Data Rates for GSM Evolution
ETSI	European Telecommunications Standards
FCC	Federal Communications Commission
FDD	Frequency Division Duplex
FDMA	Frequency Division Multiple Access
FM	Frecuencia Modulada
FPGA	Field Programmable Gate Array
GGSN	Gateway GPRS Support Node

GMSC	Gateway Mobile Switching Centre
GMSK	Gaussian Minimum Shift Keying
GNU	GNU´s Not Unix
GPRS	General Packet Radio Service
GSM	Global System for Mobile communications
GSN	GPRS Support Node
HLR	Home Location Register
HSDPA	High-Speed Downlink Packet Access
HSPA+	High-Speed Packet Access Plus
HSUPA	High-Speed Uplink Packet Access
HTTP	HyperText Transfer Protocol
ICCID	International Circuit Card ID
IF	Intermediate Frequency
IK	Integrity Key
IoT	Internet of Things
IP	Internet Protocol
IMS	IP Multimedia Sub-system
IMSI	International Mobile Subscriber
IMT-2000	International Mobile Telecommunications - 2000
IMTS	Improved Mobile Telephone System
ISDN	Integrated Services Digital Network
LLC	Logical Link Control
LTE	Long Term Evolution
LTS	Long Term Support
MCC	Mobile Country Code
ME	Mobile Equipment
MIMO	Multiple-Input Multiple Output
MNC	Mobile Network Code
MS	Mobile Station
MSC	Mobile Switching Center
MSISDN	Mobile Station Integrated Services Digital Network
NMT	Nordic Mobile Telephone
NTT	Nippon Telegraph and Telephone
OFDMA	Orthogonal Frequency-Division Multiple Access
OpenBTS	Open Base Transceiver Station
PDC	Personal Digital Cellular
PLMN	Public Land Mobile Network
PS	Packet Switching
PSTN	Public Switched Telephone Network
P-TMSI	Packet - Temporary Mobile Subscriber Identity
QPSK	Quadrature Phase Shift Keying

RAND	RANDom number
RES	RESponse
RF	Radio Frequency
RNC	Radio Network Controller
RSSI	Received Signal Strength Indicator
RTMS	Radio Telephone Mobile System
RTP	Real-time Transport Protocol
Rx	Recepción
SDR	Software Defined Radio
SGSN	Serving GPRS Support Node
SIM	Subscriber Identity Module
SIP	Session Initiation Protocol
SMS	Short Message Service
SMSP	Short Message Service Parameters
SNR	Signal-to-Noise Ratio
SoC	System-on-Chip
SQL	Structured Query Language
TACS	Total Access Communications System
TCP	Transmission Control Protocol
TDD	Time Division Duplex
TDMA	Time-Division Multiple Access
TMSI	Temporary Mobile Subscriber Identity
TRX	Transceiver
Tx	Transmisión
UARFN	UMTS Absolute Radio Frequency Channel Number
UDP	User Datagram Protocol
UE	User Equipment
UHD	Usrp Hardware Driver
UL	UpLink
UMTS	Universal Mobile Telecommunications System
UNAM	Universidad Nacional Autónoma de México
USRP	Universal Software Radio Peripheral
UTRAN	UMTS Terrestrial Radio Access Network
UWC-136	Universal Wireless Communications - 136
VLR	Visitor Location Register
VoIP	Voice Over IP
WCDMA	Wideband Code Division Multiple Access
WiFi	Wireless Fidelity
WiMAX	Worldwide Interoperability for Microwave Access
XRES	eXpected RESponse

CAPÍTULO 1: INTRODUCCIÓN

1.1 Comunicaciones Móviles

3G, es la abreviación de tercera generación de transmisión de voz y datos a través de telefonía móvil. El estándar más conocido que caracteriza esta generación es UMTS [1].

Su antecesor GSM (Global System for Mobile communication), perteneciente a la segunda generación (2G), se desarrolló principalmente para transmisiones de voz con muy buena calidad, pero la transmisión de datos era lenta. El desarrollo de esta segunda generación fue impulsado principalmente por la necesidad de mejorar la calidad de transmisión y aumentar la capacidad y el área de cobertura del sistema. La tecnología digital en la telefonía fue bienvenida y con ella las ventajas de facilidad de señalización y niveles menores de interferencia [2].

Dados los cambios rápidos en las expectativas de los usuarios, GSM no cumplía las necesidades inalámbricas y fue evolucionado a lo que fue llamada la fase GSM 2+. Surgieron las tecnologías 2.5G GPRS (General Packet Radio Service) y 2.75 EDGE (Enhanced Data rates for GSM Evolution), y estas fueron la base para proveer servicio de multimedia de calidad en la siguiente generación 3G, que fue implementado a gran escala con el estándar UMTS [3].

La tecnología evoluciona rápidamente, y en este caso la tecnología de telefonía celular está experimentando avances cada vez mayores en las tasas de transferencia de información. La aparición de la tecnología 3G hizo que Internet estuviera totalmente disponible en cualquier teléfono compatible, permitiendo una variedad de funciones personales y comerciales [4]. Con ella fue posible la transferencia de datos similares a una computadora conectada a una red de banda ancha, y constituyó un paso de avance muy importante para el desarrollo de la próxima generación 4G [5].

El sistema UMTS fue promovido inicialmente por ETSI (European Telecommunications Standards Institute), su especificación actual corre a cargo del foro 3GPP (Third Generation Partnership Project), participado por varios organismos de normalización regionales [5].

UMTS se basa en el empleo de una interfaz radio WCDMA. Tiene como principal plataforma de lanzamiento a GSM por lo que muchos de los conceptos de ambas tecnologías son similares. Dentro de la red, en una primera fase se considera la utilización de los actuales elementos disponibles en las redes precedentes GSM, GPRS y EDGE, planteándose su evolución para fases posteriores. El primer servicio comercial utilizando UMTS fue lanzado en el año 2001 [6].

1.2 Problemática

La tecnología de la información juega un papel esencial en las actividades sociales, culturales y económicas. La forma en que hacemos compras, interactuamos con la banca, el acceso a servicios públicos, entre otras cosas, son ejemplos actuales que definen muy bien que estamos en la sociedad de la información. Hoy en día la telefonía celular es un elemento fundamental en la sociedad que nos desarrollamos. Esta juega un papel importante en la vida diaria de las personas, pues es posible realizar muchas operaciones importantes desde el terminal móvil, gracias a la posibilidad de acceso a Internet que ofrecen [7].

En este momento la cantidad de líneas móviles es similar al número de habitantes en el planeta. En el caso particular de México, se considera que la penetración de la telefonía móvil es baja, y hasta el tercer trimestre de 2016 era del 90% de la población [8].

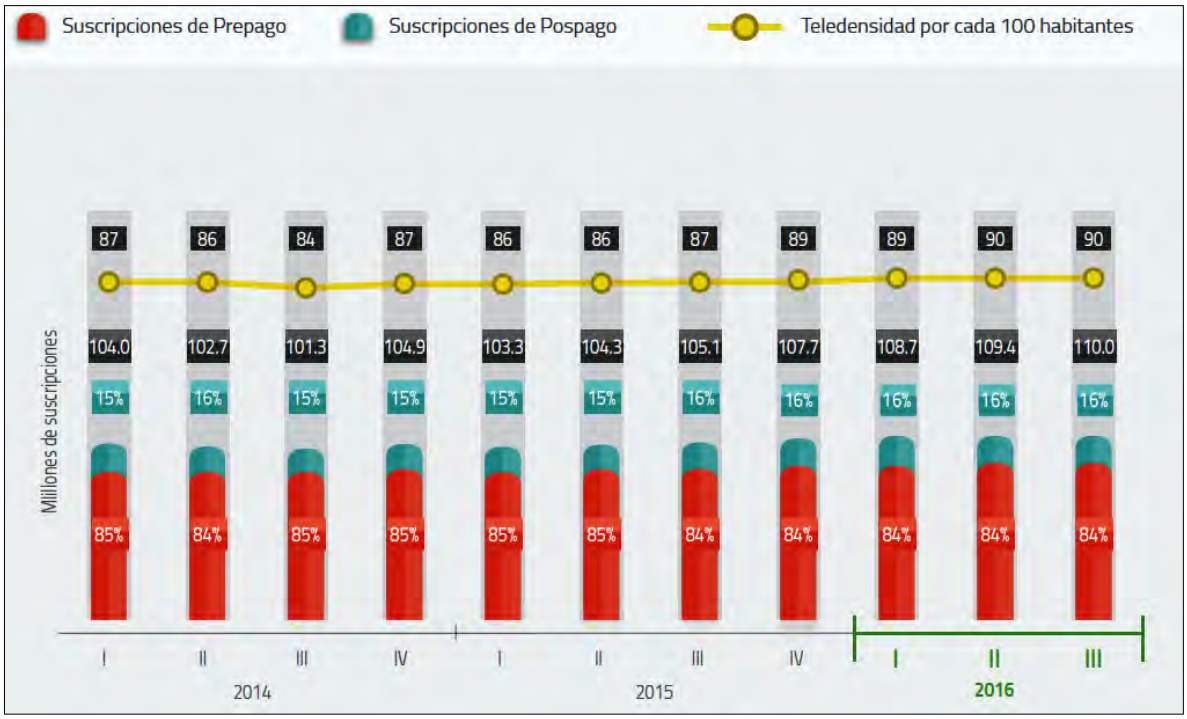


Figura 1.1 Crecimiento de líneas móviles en México [8].

En el campo de la telefonía y comunicaciones inalámbricas en general, se han llevado a cabo un gran número de investigaciones, que han dado lugar a la estandarización e implementación de protocolos cuyas tecnologías se han expandido globalmente. Es típico que estas tecnologías se desarrollen tradicionalmente en sistemas basados en hardware, haciendo que sea prácticamente imposible su modificación. Sin embargo, con el desarrollo de FPGAs (Field Programmable Gate Arrays) y SoC (System-on-Chip) programables para realizar funciones digitales, se pueden realizar prototipos de tecnologías inalámbricas cuyas configuraciones se realizan, desarrollan y modifican a través de *software*.

Esto se hace posible gracias a Radios Definidos por Software, concepto definido por Joseph Mitola [9]. Con SDR es posible la implementación de las características y funciones de la capa física de varias tecnologías inalámbricas, en una plataforma programable y reconfigurable. Esto sin lugar a duda brinda muchas ventajas y menor coste en el desarrollo de cualquier sistema.

La reducción de costes de los equipos SDR, junto con el aumento de las prestaciones de los procesadores, están marcando una nueva tendencia en los sistemas de comunicaciones. Se puede diseñar cualquier servicio de comunicaciones a partir de un *hardware*, estando el resto del sistema definido por el *software* que ejecutará un procesador [10].

Ya la tecnología SDR lleva algunos años en desarrollo, pero aún se encuentra en proceso de investigación y estudio. La implementación de la capa física de distintas tecnologías inalámbricas es uno de los principales desarrollos en el área, que permiten obtener sistemas de comunicaciones de una manera económica y eficiente [11].

En la actualidad, aunque la telefonía celular llega a muchos lugares, existen algunos de baja densidad de población que carecen del servicio. En los asentamientos poblacionales donde la cantidad de usuarios es pequeña, se hace difícil hacer una inversión tecnológica fuerte solamente para beneficiar pocas personas.

También existen los casos donde hay regulaciones gubernamentales que no permiten en determinadas áreas el levantamiento de torres, elemento indispensable para ofrecer servicios de telecomunicaciones. Uno de esos ejemplos es la UNAM, y por este motivo la cobertura que ofrecen los proveedores de servicios de telefonía celular es muy pobre, y todavía existen muchos lugares sin cobertura alguna en la Ciudad Universitaria.

La implementación de un sistema de comunicaciones de telefonía celular de bajo coste es una buena opción para cubrir estas zonas. Usando Radios Definidos por Software y sin muchos recursos, se puede instalar una radiobase para darle solución al problema, y la UNAM es un buen escenario para desarrollar este proyecto.

Dada esta flexibilidad de la tecnología de Radios Definidos por Software, se decide implementar la capa física de un sistema basado en UMTS. Este desarrollo será llevado a cabo utilizando la plataforma de código abierto OpenBTS, y como radio el equipo *Ettus* USRP (Universal Software Radio Peripheral) N210.

1.3 Objetivos

Objetivo General

- Implementar una estación base UMTS que permita que terminales móviles puedan conectarse a ella autenticándose, usando la plataforma de código abierto OpenBTS, una computadora y un equipo de Radios Definidos por Software.

Objetivos Particulares

- Entender la plataforma SDR y el principio de operación de los equipos USRP N210.
- Instalar y configurar una red UMTS sobre la plataforma OpenBTS.
- Poner a prueba en base al estándar UMTS, el alcance del equipo SDR para trabajar como *NodeB* (NB).
- Realizar una simulación de cobertura de servicio, ampliando la potencia con equipamiento adicional.

1.4 Metodología

OpenBTS.org es un proyecto de software libre desarrollado para revolucionar las redes móviles, y se piensa que va a sustituir los protocolos de telecomunicaciones, así como los sistemas de hardware complejos. OpenBTS-UMTS utiliza Radios Definido por Software para presentar una interfaz estándar WCDMA a los dispositivos. Usando esta plataforma se programará el equipamiento para implementar una red UMTS real.

Para la implementación de la radiobase 3G fueron necesarios los siguientes pasos:

- Familiarización y estudio del estado de este proyecto de software OpenBTS.org.
- Familiarización y estudio de las diferentes versiones de OpenBTS.
- Estudio de la plataforma OpenBTS para desarrollar prototipos de telefonía celular.

- Instalación de LINUX, librerías, drivers, y aplicaciones necesarias, en una computadora con buenos recursos de hardware, para la correcta instalación del programa OpenBTS-UMTS.
- Instalación y prueba de ejecución de la aplicación “OpenBTS-UMTS” que implementa la función de *NodeB*.
- Instalación y prueba de ejecución de la aplicación “SIPAuthServe” que implementa la función de autenticación y registro a la red.
- Adquisición de tarjetas SIM vírgenes compatibles y equipo programador “Smart Card Writer”, para adicionar usuarios a la nueva red.
- Definir y programar las diferentes variables y algoritmos para el correcto funcionamiento del sistema.
- Implementar en el equipo de Radios Definidos por Software el sistema programado.

Una vez implementado el sistema se comprueba su funcionamiento con el USRP N210 en un radio pequeño. Posteriormente se hace una propuesta de instalar un amplificador, duplexor y sistema radiante, brindando la cobertura teórica que refleje la simulación en la banda propuesta basados en los resultados obtenidos de cobertura real, con el USRP entregando 100 mW de potencia.

1.5 Estructura de la tesis

Este trabajo de investigación está dividido en 6 capítulos.

El capítulo 2 describe el estado del arte, donde se relata brevemente la historia de las comunicaciones móviles y el surgimiento la tecnología SDR. Se hace también una descripción general y amplia del estándar UMTS, y cómo este llega a implementarse en la plataforma OpenBTS, destacando los aspectos que difieren de sus versiones precedentes.

En el capítulo 3 se realiza una descripción del sistema a implementar, donde se tiene en cuenta el sistema operativo, marcos de desarrollo y arquitectura del USRP N210. Por otra parte, se describe la arquitectura por componentes en la red UMTS, destacando los elementos claves.

En el capítulo 4 se describe el desarrollo y la instalación de la red UMTS, la identificación de la red 3G y registro de los teléfonos móviles, la programación de la tarjeta SIM, y por último, la propuesta de implementación de la red con mayor potencia.

En el capítulo 5 se reflejan las pruebas y los resultados obtenidos en la implementación, y se hace una simulación de cobertura de la propuesta aumentando potencia a la estación base.

Finalmente, el capítulo 6 muestra las conclusiones y los trabajos futuros derivados de esta investigación.

CAPÍTULO 2: ESTADO DEL ARTE

2.1 Introducción

En los últimos años, en el mundo de las telecomunicaciones se han desarrollado numerosas y nuevas tecnologías. Dentro de las comunicaciones inalámbricas, la telefonía celular, constituye un gran éxito proporcionando a los dispositivos móviles una gran cantidad de nuevos servicios. Hoy en día la gente no sólo puede usarlo como radiotransmisor para hablar o leer mensajes cortos, sino también para navegar en Internet, administrar cuentas bancarias, ver programación de televisión, enviar fotografías y entretenerse con juegos electrónicos en línea, entre otras actividades.

Actualmente un teléfono celular es una herramienta que brinda enorme visibilidad, impone modas, es fuente de identidad para los jóvenes, es adictivo, se porta como parte de la vestimenta y sustituye en tiempos récord a otras tecnologías como la cámara fotográfica y grabadora; también es indispensable como reloj despertador, calculadora, agenda de actividades, *etc*. En el nuevo siglo, ha adquirido ya una importancia primaria para la sociedad y los individuos. Pertenece indudablemente al conjunto de artefactos que se han popularizado. La cultura es cada vez más influenciada por el incremento de estas sofisticadas tecnologías de comunicación e información, que incrementan las capacidades personales [12].

Al incorporar Internet, en cualquier lugar que se encuentren, las personas tienen oportunidad de involucrarse en todo tipo de interacciones bilaterales o multilaterales, para buscar información o publicar puntos de vista personales a escala mundial. Por su parte, con el teléfono móvil se ensancha la esfera de interacción micro-social [13]. Pero para entender estas nuevas tecnologías de comunicaciones de hoy en día es importante saber cómo han evolucionado.

2.2 Historia de la telefonía celular

En 1921 se realiza el primer experimento de comunicación trasatlántica por telefonía inalámbrica entre Reino Unido y Estados Unidos, utilizando una onda de 500 metros

de longitud. En el año 1946 se aprobó el servicio de telefonía móvil en Estados Unidos, donde la compañía AT&T introdujo el primer servicio con 6 canales en la banda de 150 MHz, espaciados a 60 kHz. En 1949, la FCC (Federal Communication Commission) dispuso más canales y los otorgó a varias compañías como la Bell System y la RCC (Radio Common Carriers), con la intención de crear la competencia y evitar los monopolios. En 1956, la Bell System comenzó a dar servicio en los 450 MHz [14]. En 1958, la Richmond Radiotelephone Co. mejoró su sistema de marcado conectando rápidamente las llamadas de móvil a móvil. A mediados de 1960 la Bell System introdujo el IMTS (Improved Mobile Telephone System) con características mejoradas. A finales de la década de 1960 comenzaron a trabajar los primeros sistemas de telefonía celular sin la reutilización de frecuencias. En 1969 la Bell System aplicó por primera vez el rehúso de frecuencias en un servicio comercial para teléfonos públicos [15].

En la década de 1980 comienza la primera generación de telefonía celular caracterizada por ser analógica y estrictamente para voz. A continuación, se listan ejemplos de algunos sistemas de telefonía celular empleados durante la primera generación.

Sistema	País	No. de Canales	Espaciado (kHz)
AMPS	EE.UU.	832	30
C-450	Alemania	573	10
ETACS	Reino Unido	1240	25
JTACS	Japón	800	12.5
NMT-900	Escandinavia	1999	12.5
NMT-450	Escandinavia	180	25
NTT	Japón	2400	6.25
Radiocom-2000	Francia	560	12.5
RTMS	Italia	200	25
TACS	Reino Unido	1000	125

Tabla 2.1 Sistemas de telefonía celular empleados en la Primera Generación [16].

Con el paso de los años avanza la tecnología y la Segunda Generación (2G) de telefonía da un salto a las técnicas de transmisión digitales. 2G surge en 1990, y a

diferencia de la primera, esta se caracteriza por ser un sistema digital que utiliza protocolos de codificación más sofisticados.

En esta Segunda Generación existen 4 estándares principales [17]:

- Global System for Mobile Telecommunications (GSM)
- Digital AMPS (D-AMPS)
- Code Division Multiplex Access (CDMA)
- Personal Digital Cellular (PDC)

La Segunda Generación continúa desarrollándose y surge la denominada 2G+. Los estándares más sobresalientes fueron GPRS (General Packet Radio System), HSCSD (High Speed Circuit Switched Data), y EDGE (Enhanced Data Rates for Global Evolution).

Sobre los años 2000 - 2001 comienzan a operar las redes 3G (Tercera Generación). Los protocolos empleados en 3G soportaron velocidades más altas de transferencia de información. Entre las tecnologías más importantes que compitieron en la tercera generación se encuentran UMTS, CDMA-2000, IMT-2000 (International Mobile Telecommunications - 2000), UWC-136 (Universal Wireless Communications - 136) [17].

Para aumentar las tasas de transferencias aparecen los estándares HSDPA (High-Speed Downlink Packet Access), HSUPA (High-Speed Uplink Packet Access), HSPA+ (High-Speed Packet Access Plus, o Evolved HSPA), WiMAX (Worldwide Interoperability for Microwave Access), hasta la llegada de LTE (Long Term Evolution). Entre 2007 - 2009 surge la cuarta generación de redes móviles donde se encuentran las tecnologías WiMAX y LTE-*Advanced*, siendo esta última la preferida pues utiliza OFDMA (Orthogonal Frequency-Division Multiple Access) como nueva técnica en la capa física, y además ha demostrado tener más eficiencia en las pruebas que se han realizado. Las redes del futuro que pertenecerían a la Quinta Generación (5G) van a abordar la evolución más allá del Internet móvil. Con ellas se piensa alcanzar el “Internet de las Cosas” (IoT, por sus siglas en inglés) masivo para comienzos de 2020 [18].

2.3 Surgimiento de la Tecnología Radios Definidos por Software (SDR)

Hoy en día las tecnologías evolucionan muy rápidamente y surge la problemática que los radios y protocolos basados en *hardware*, tienen muy pocas posibilidades de reprogramación y reconfiguración. Esta falta de flexibilidad provoca problemas para modificar el *firmware* y *hardware* si fuera necesario. Con el surgimiento de la tecnología SDR (Software-Defined Radio), se abre una puerta bien grande para solucionar estos problemas de incompatibilidad, interoperabilidad y actualización.

En el año 1991 se lleva a cabo la primera implementación SDR en Estados Unidos. Este se desarrolla como un proyecto de software militar denominado "SPEAKeasy", el cual implementaba comunicaciones inalámbricas con un equipo programable en un rango 2 MHz - 200 MHz [19].

Actualmente esta tecnología ha logrado propagarse globalmente. Sobre ella se realizan diferentes investigaciones y desarrollos por todo el potencial que posee, pues muchas funciones de la capa física son implementadas en el *software* [20].

2.4 Sistema universal de telecomunicaciones móviles. Estándar UMTS

UMTS es una de las tecnologías usadas por los móviles de tercera generación. A principios de 1998, la ETSI presentó la tecnología WCDMA como mejor opción a implementar en el estándar UMTS [21]**;** pero antes que esto sucediera tuvo que surgir el proyecto 3GPP.

2.4.1. Proyecto Asociación de Tercera Generación

A medida que se fueron estandarizando varias tecnologías inalámbricas similares en el mundo, se hizo evidente que para garantizar la compatibilidad de los equipos, sería muy difícil si se continuaba trabajando en paralelo sin llegar a un acuerdo común. A raíz de esto se toma la iniciativa de crear un foro único para la estandarización de WCDMA, así como sus especificaciones comunes. Es entonces cuando surge el Proyecto Asociación de Tercera Generación.

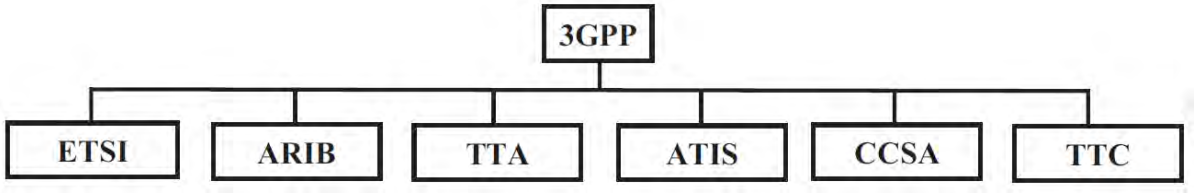


Figura 2.1 Composición organizativa y miembros de 3GPP [22].

A 3GPP pertenecen tanto empresas fabricantes de equipos y componentes como empresas operadoras de red. Para finales de 1999 surge la primera versión de la especificación común que fue llamada *Release-99*, y es la que se conoce como el estándar UMTS [22].

2.4.2 Acceso múltiple por división de código para banda ancha

El acceso múltiple por división de código para banda ancha, también conocida por WCDMA, es la interfaz aérea de tercera generación principal en el mundo. Este sistema utiliza el acceso por multiplexación de división de código (CDMA), proporcionando una amplia gama de servicios con características diferentes sobre una portadora común de 5 MHz [23].

Una de las principales ventajas que introduce el estándar UMTS con la modulación WCDMA, es la flexibilidad de la capa física para acomodar diferentes tipos de servicio simultáneamente, especialmente cuando las tasas de transferencia son relativamente bajas y medianas.

El ancho de banda de 5 MHz admite buena velocidad de transferencia de datos. WCDMA admite tasas de transferencia muy variables, en otras palabras, el concepto de obtener ancho de banda bajo demanda (BoD) está bien diseñado. La velocidad de transferencia de datos se mantiene constante durante cada trama de 10 ms, pero esta capacidad entre los usuarios puede cambiar de trama a trama. Esta asignación rápida de recursos de radio normalmente es controlada por la red para lograr un rendimiento óptimo en el servicio.

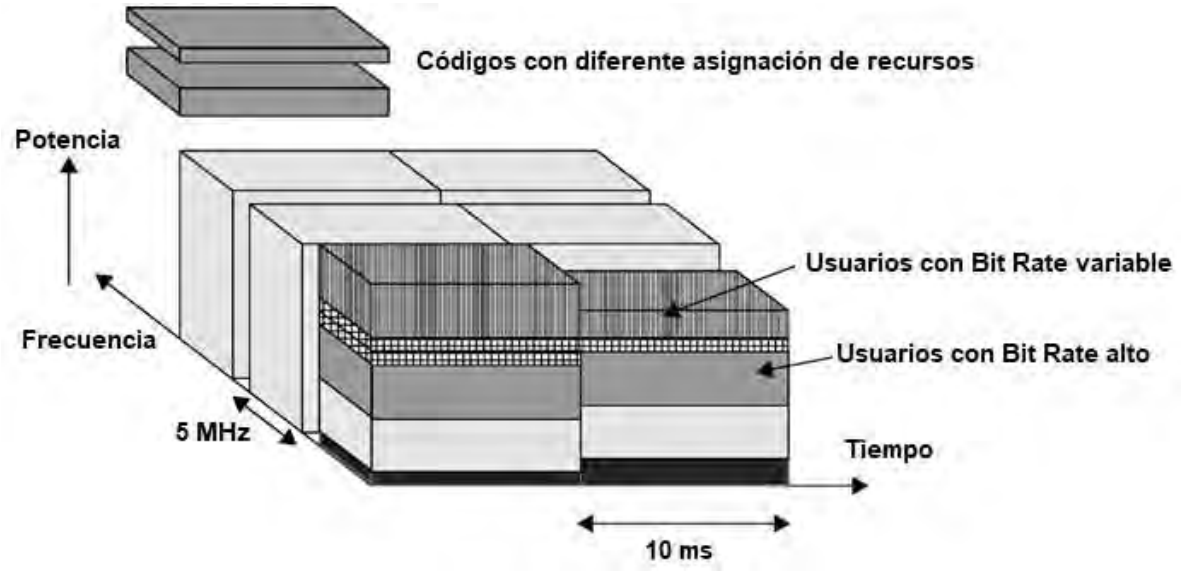


Figura 2.2 Asignación de ancho de banda en WCDMA [24].

WCDMA admite dos modos básicos de operación: FDD (Frequency Division Duplex) y TDD (Time Division Duplex). En FDD, se utilizan frecuencias portadoras separadas de 5 MHz para el UL (enlace ascendente o *Up Link*) y el DL (enlace descendente o *Down Link*), mientras que en TDD solo se comparte el tiempo en 5 MHz entre el UL y el DL.

2.4.3 Arquitectura de la red UMTS

El sistema UMTS consiste en una serie de elementos de red lógica que tienen una funcionalidad definida, pudiéndose estos agrupar según una funcionalidad similar o según la subred a la que pertenecen.

Otra forma de agrupar los elementos de la red UMTS es dividirlos en subredes. La posibilidad de tener varias entidades del mismo tipo permite la división del UMTS en subredes que funcionan por sí mismas, o junto con otras subredes, y que se distinguen entre sí por identidades únicas. Dicha subred se denomina UMTS PLMN (Public Land Mobile Network), y típicamente está conectada a otras PLMN y a otros tipos de red, como ISDN (Integrated Services Digital Network), PSTN (Public Switched Telephone Network), Internet, y más [25].

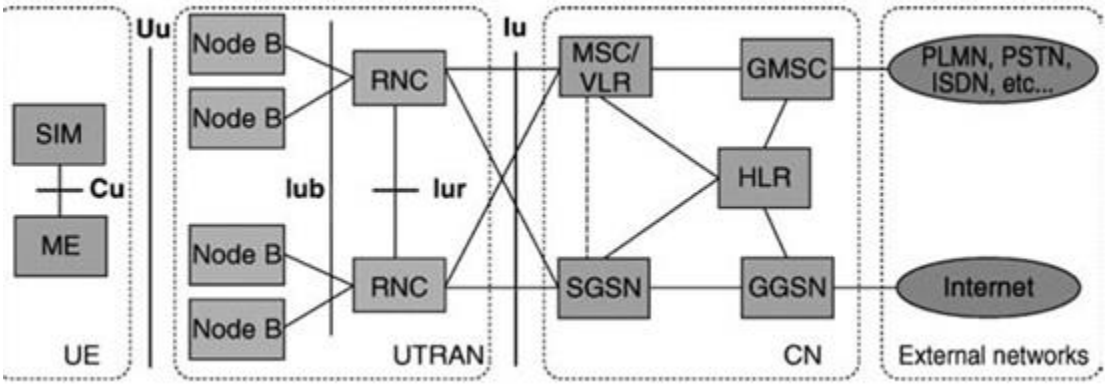


Figura 2.3 Arquitectura general de la red UMTS [26].

El bloque UE (User Equipment) está formado por un equipo, pero por dos componentes.

- El equipo móvil (ME) es el terminal de radio utilizado para la comunicación por radio a través de la interfaz Uu.
- La SIM (Subscriber Identity Module) es una tarjeta inteligente que contiene la identidad del suscriptor y realiza algoritmos de autenticación.
- Interfaz Uu: Esta es la interfaz de radio WCDMA a través de la cual el UE accede al sistema.

La estructura en el bloque UTRAN (UMTS Terrestrial Radio Access Network) se basa en el principio de que las capas y planos son lógicamente independientes entre sí, y si es necesario, partes de la estructura puede cambiarse en el futuro, mientras que otras partes pueden permanecer intactas.

- NodeB (estación base). Su función principal es realizar el procesamiento de la interfaz aérea. De forma general se puede ver como conversor del flujo de datos entre las interfaces Iub y Uu.
- Interfaz Iub: Esta interfaz es la que vincula el *NodeB* con el RNC.
- El RNC (Radio Network Controller) posee y controla los recursos de radio en su dominio, y es el punto de acceso para los servicios que UTRAN proporciona al CN.
- Interfaz Iu: Interfaz que conecta la UTRAN con el CN.
- Interfaz Iur. Esta interfaz permite el proceso de traspaso *handover* entre RNCs, incluso de diferentes fabricantes, complementando así la interfaz Iu.

El bloque CN (Core Network) contiene todos los elementos centrales encargados de administrar y procesar la trasferencia de información en la red. En este bloque, se separan los procesos de PS y CS.

Redes de conmutación de circuitos (CS, siglas en inglés). Estas proporcionan conexiones a base de circuitos conmutados, como el servicio de telefonía existente. ISDN y PSTN son ejemplos de estas redes.

Redes de conmutación de paquetes (PS, siglas en inglés). Estas proporcionan conexiones para servicios de intercambio de paquetes de datos. Internet es un ejemplo de una red basada en PS.

- Home Location Register (HLR). Es el elemento que almacena la base de datos del usuario y su perfil de servicio. Esta se crea cuando un nuevo usuario se suscribe al sistema y permanece almacenada mientras la suscripción esté activa.
- Mobile Switching Center / Visitor Location Register (MSC/VLR). MSC es el conmutador y VLR es la base de datos que sirve al UE en su ubicación actual para los servicios de CS. La parte de la red a la que se accede a través del MSC/VLR pertenece al dominio CS.
- Gateway MSC (GMSC). Es el conmutador que conecta UMTS PLMN a redes externas basadas en CS. Todas las conexiones CS entrantes y salientes pasan por GMSC.
- Serving GPRS Support Node (SGSN). Su función es similar a la de MSC/VLR, pero se usa para servicios basados en conmutación de paquetes. La parte de la red a la que se accede a través del SGSN pertenece al dominio PS.
- Gateway GPRS Support Node (GGSN). Su funcionalidad es similar a la de GMSC, pero relacionada con los servicios de PS.

2.4.4 Seguridad en UMTS. Proceso de registro y autenticación del usuario a la red.

La seguridad 3G se basa en la arquitectura de seguridad 2G porque esta demostró ser efectiva y eficiente en su momento. En esta generación se eliminan las deficiencias presentes 2G y se agregan nuevas funciones en cuanto a seguridad.
La seguridad de los sistemas 3G cuenta dos aspectos fundamentales: "Integridad de los datos" y "Cifrado" [27].
Integridad de datos: Es la función que garantiza que ninguna red desconocida pueda enviar mensajes de señalización innecesarios con malas intenciones, o que causen un efecto no deseado en un proceso de comunicación en curso.
Cifrado: Es la función que asegura que todos los mensajes de señalización y datos se cifren en la interfaz aérea, para que nadie pueda escucharlos o interpretarlos.
En el caso de UMTS, la protección de integridad es obligatoria mientras que el cifrado es opcional.
Cada usuario provisto con una tarjeta SIM cuenta con un IMSI (International Mobile Subscriber Identity). En la red UMTS nadie debe conocer qué servicios usa un IMSI en la interfaz aérea. Junto con la confidencialidad de la identidad del usuario se debe mantener la confidencialidad de la ubicación del usuario.
Para lograr esto se debe seguir el siguiente procedimiento:

- Asignar una identidad temporal al usuario para identificarlo (TMSI o P-TMSI).
- Después del transcurso de un tiempo determinado se debe cambiar esta identidad temporal.
- Cifrar los datos del usuario que pueden revelar su identidad.

La Identidad de Suscriptor Móvil Temporal o Temporary Mobile Subscriber Identity (TMSI) o *Packet* TMSI (P-TMSI), tiene un significado local que se aplica solo en el área de ubicación, o en el área de enrutamiento en la cual el usuario está registrado. Siempre que el TMSI/P-TMSI esté disponible, es usado para identificar al usuario para solicitudes de localización, solicitudes de actualización de ubicación, solicitudes de conexión y liberación, solicitudes de servicio, y restablecimiento de conexión.

2.4.4.1 Procedimiento de registro

El procedimiento de registro es la primera acción que ejecuta un usuario del sistema, en aras a poder recibir los servicios de la red. Normalmente se lleva a cabo por primera vez cuando se enciende el terminal móvil y este detecta la presencia de una red UMTS.

El llamado *attach* o *attachment*, es el proceso mediante el cual un terminal móvil se adjunta a la red para formar parte de ella, y beneficiarse de todos los servicios que esta ofrece. En el siguiente esquema se muestra el procedimiento general de *attachment* de la manera más simple.

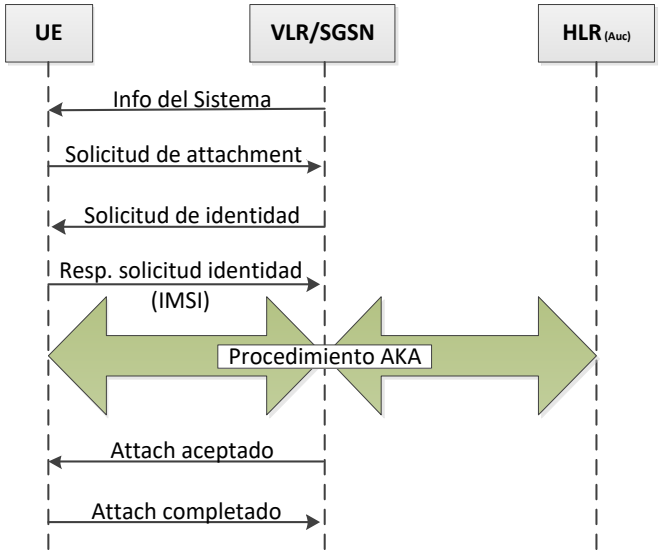


Figura 2.4 Esquema de procedimiento de attachment en UMTS [28].

Cuando el usuario ya se encuentra registrado se genera un valor TMSI, y si necesita realizar el proceso de *attachment* nuevamente, ya no es necesario enviar el IMSI como respuesta de identidad. Antes de que VLR inicie este procedimiento, genera un nuevo valor TMSI y almacena la asociación entre el IMSI y el TMSI en la base de datos. A continuación, se le envía el nuevo TMSI al usuario, y una vez que el móvil recibe este mensaje, borra el antiguo TMSI y envía una respuesta al VLR. Después de concluido este proceso, el VLR elimina la asociación entre el IMSI y el viejo TMSI. Este proceso se conoce como “procedimiento de reasignación de TMSI”.

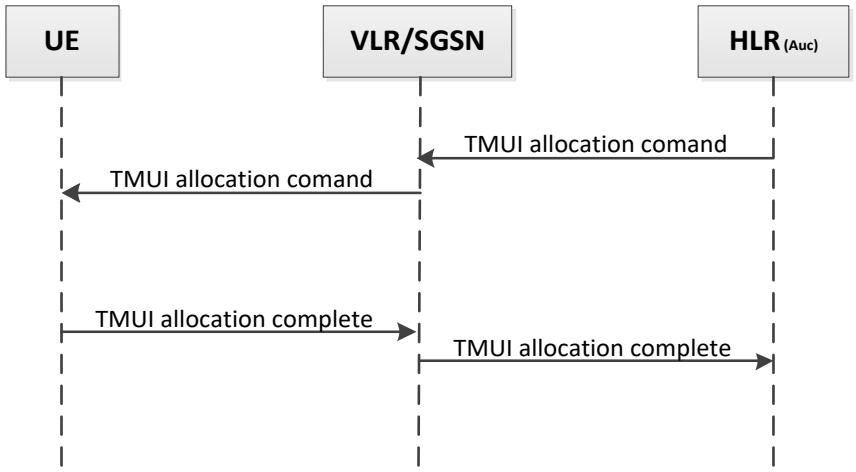


Figura 2.5 Esquema del procedimiento de reasignación de TMSI [28].

Para la asignación del nuevo valor de TMSI se usa el comando de asignación TMUI (Temporary Mobile User Identification). Cuando el VLR verifica la asignación exitosa mediante la respuesta "TMUI Allocation Complete", elimina entonces la asociación que había entre el IMSI y el antiguo TMSI.

2.4.4.2 Procedimiento de autenticación y establecimiento de claves de seguridad AKA (Authentication and Key Agreement)

Es muy importante que la red de servicio pueda verificar la identidad del usuario, y al mismo tiempo el usuario pueda verificar la autenticidad de la red a la que se está conectando. Para lograr esto, la autenticación se debe llevar a cabo en cada configuración de conexión entre el usuario y la red [29].
UMTS incluye dos mecanismos:

- Un mecanismo de autenticación que utiliza un vector de autenticación entregado por el usuario a la red.
- Un mecanismo de autenticación local que usa la clave de integridad establecida entre el usuario y la red. Este se lleva a cabo durante la ejecución previa de la autenticación, y el procedimiento de establecimiento de la clave.

El procedimiento AKA logra la autenticación mutua entre usuario y la red que muestra el conocimiento de la clave secreta K, pues esta clave solo se encuentra disponible para el usuario y el Centro de Autenticación (AuC) ubicado en el HLR.

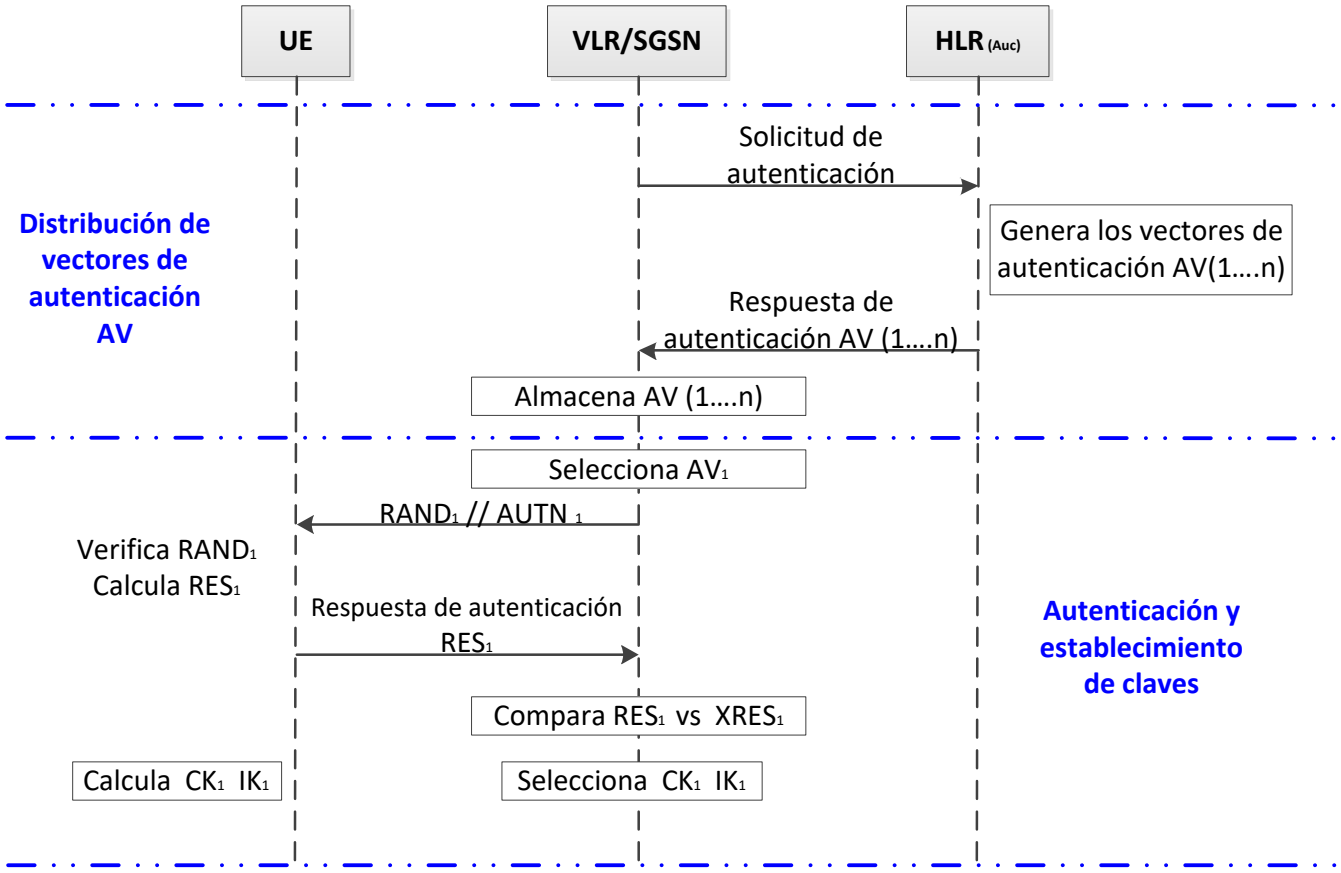


Figura 2.6 Esquema de procedimiento de autenticación y establecimiento de claves de seguridad AKA [30].

Después de la solicitud de *attachment* del usuario, para que sea debidamente registrado en la red UMTS, se debe realizar el procedimiento de autenticación AKA. El VLR solicita al HLR que envíe una matriz ordenada de n vectores de autenticación. Estos vectores de autenticación se derivan de la llave K que solamente es conocida por el usuario y el HLR, y se ordenan en función del número de secuencia.

Cada vector de autenticación consta de:

- Un número aleatorio **RAND**
- Una respuesta esperada **XRES** (HLR) o **RES** (UE)
- Una clave de cifrado **CK**
- Una clave de integridad **IK**
- Un *token* de autenticación **AUTN**

Cada vector de autenticación es válido para una autenticación entre VLR/SGSN y el UE. Cuando el VLR va a ejecutar un nuevo procedimiento AKA con el UE, selecciona el siguiente vector de autenticación de la matriz ordenada almacenada.
Al usuario se le envían los parámetros RAND y AUTN. El UE verifica si el AUTN es válido y, si es así, produce una respuesta RES que se envía de vuelta a VLR/SGSN. Cuando el VLR/SGSN recibe la respuesta RES, la compara con la respuesta esperada XRES. Si coinciden, el procedimiento es exitoso y se transfieren los valores CK e IK a las entidades que realizarán la protección de cifrado e integridad. Por parte del usuario, si el procedimiento es exitoso, se recuperan los valores CK e IK y se transfieren a las entidades que realizan funciones de cifrado e integridad dentro del equipo terminal.

2.5 Proyecto de plataforma de desarrollo colaborativo OpenBTS

OpenBTS ofrece una alternativa de código abierto para los protocolos de telecomunicaciones heredados, y los sistemas de hardware tradicionalmente complejos y propietarios. Es considerada una aplicación UNIX[1], que utiliza SDR para presentar una interfaz aérea de comunicación móvil a un teléfono móvil estándar, y se vincula con un softswitch[2] SIP (Session Initiation Protocol) o PBX (Private Branch eXchange) para interconectar llamadas [31].
La combinación de la interfaz aérea utilizada con una red VoIP (Voz sobre IP) de bajo costo, constituye la base de un nuevo tipo de red celular, que se puede implementar y operar a un costo sustancialmente más bajo que las tecnologías existentes en muchas aplicaciones. Esta solución es muy útil para las implementaciones celulares rurales, y las redes celulares privadas en áreas remotas.
Esta plataforma se puede decir que es una forma simplificada de IMS (IP Multimedia Subsystem), que en un principio fue diseñada para que funcionara con móviles de

[1] Unix (registrado oficialmente como UNIX®) es un sistema operativo portable, multitarea y multiusuario. [32]
[2] Softswitch: Dispositivo principal en la capa de control dentro de una arquitectura NGN (Next Generation Network), encargado de proporcionar el control y procesamiento de llamadas y otros servicios, sobre una red de conmutación de paquetes IP [33].

segunda generación GSM, y que ha continuado desarrollándose para implementar también los estándares GPRS y UMTS.

2.5.1 Estándar GSM en la plataforma OpenBTS

La implementación de GSM en OpenBTS se desarrolla similar al estándar original. La señal empleada se modula digitalmente utilizando GMSK (Gaussian Minimum Shift Keying, modulación por desplazamiento mínimo Gaussiano). Se trata de una modulación de fase, en la que la señal moduladora se filtra con un filtro gaussiano. De este modo se consigue que la mayor parte de la señal modulada ocupe menos de 200 kHz, facilitando así que la canalización en GSM se realice en radiocanales de 200 kHz de ancho de banda [34].

El software se instala en un ordenador con sistema operativo Linux. Un dispositivo de código abierto, *ej.* Universal Software Radio Peripheral (USRP), se conecta a la computadora, y juntos crean una señal igual a la de los teléfonos GSM.

El *software* más las herramientas del *hardware* USRP se conectan a una PBX de código abierto *Asterisk*, que es un servidor que actúa como mini-central telefónica para realizar llamadas. OpenBTS-GSM utiliza *Asterisk* no solamente para manejar las llamadas de voz sobre IP, sino también para autentificar los usuarios. Cada usuario debe ser registrado en un fichero definido con su correspondiente valor de IMSI.

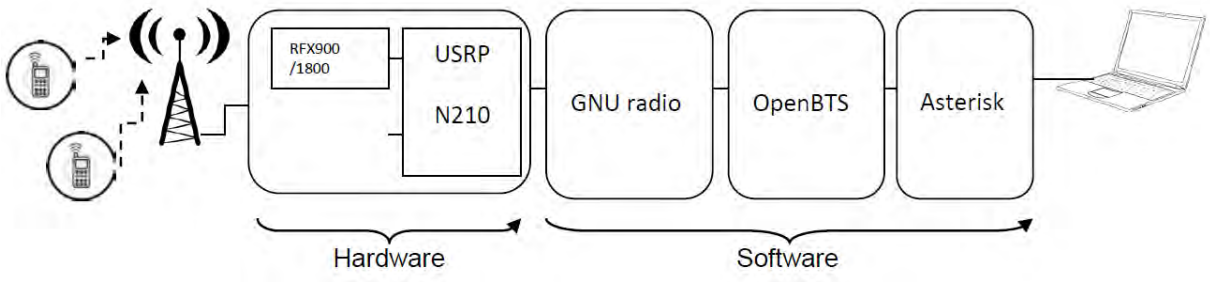


Figura 2.7 Módulos de OpenBTS [35].

OpenBTS reemplaza la infraestructura tradicional del Subsistema de Conmutación de Red del operador de GSM en las bandas de 900 y 1800 MHz. En vez de reenviar

el tráfico de la llamada a través del Centro de Conmutación Móvil (MSC), las llamadas son procesadas reenviando los datos sobre la PBX Asterisk, mediante el protocolo SIP y VoIP.

El sistema es implementado ayudándose de GNU-Radio[3] para hacer las interconexiones con *Asterisk*, y también requiere de algunas dependencias que son necesarias para operar. Estas necesitan instalarse independientes y son manipulables por el usuario.

- *Smqueue*: Servicio de almacenamiento y transmisión de mensajes cortos.
- *SIPAuthServe*: Servidor de registro y autorización de usuarios.
- *Subscriber Registry*: Base de datos de usuarios de la red.

2.5.2 Estándar GPRS en la plataforma OpenBTS

El estándar GPRS es elaborado en la versión 5 de OpenBTS, y con la llegada de este se abrió un nuevo interés en el diseño y programación de la tecnología SDR, para servicios de telefonía por conmutación de paquetes.

Está diseñado para compartir recursos de la capa física junto con GSM, por lo que su instalación es prácticamente igual. Sobre la capa física la pila de protocolos es distinta, y como una radio base 2.5G contiene dos sistemas independientes; uno para servicios de conmutación de circuitos, y otro para servicios conmutación de paquetes.

GPRS utiliza la misma estructura de GSM para formar los canales físicos. Los canales de frecuencia DL y UL tienen un ancho de banda de 200 kHz, y son definidos a través de FDMA (Frequency Division Multiple Access) por medio de un índice llamado ARFCN (Absolute Radio Frequency Channel Number) [36].

[3] GNU-Radio: proyecto de código abierto que proporciona una librería para implementar procesamiento de señales mediante programación gráfica por bloques [19].

Banda	Designación	ARFCN	F_{UL}	F_{DL}
GSM 850	GSM 850	128 - 251	824.2 + 0.2*(n – 128)	F_{UL} (n) + 45
GSM 900	P-GSM	1 – 124	890 + 0.2*n	F_{UL} (n) + 45
	E-GSM	0 – 124 975 – 1023	890 + 0.2*n 890 + 0.2*(n - 1024)	F_{UL} (n) + 45
	R-GSM	0–124 955 – 1023	890 + 0.2*n 890 + 0.2*(n – 1024)	F_{UL} (n) + 45
GSM 1800	DCS 1800	512 – 885	1710.2 + 0.2*(n - 512)	F_{UL} (n) + 95
GSM 1900	DCS 1900	512–810	1850.2 + 0.2*(n – 512)	F_{UL} (n) + 80

Tabla 2.2 Cálculo de frecuencias UL y DL [37].

GPRS cuenta con 4 Coding Schemes (CS), desde CS-1 a CS-4. El nivel de cada uno de los estos esquemas trae cambios en el nivel de protección de los datos y la tasa de transferencia efectiva (en inglés, *throughput*). El CS-1 ofrece mayor nivel de protección de datos y menor *throughput*, y CS-4 ofrece el mayor *throughput,* pero la menor protección contra errores a los datos. El esquema a ser usado depende de un mecanismo que toma en cuenta las condiciones del ambiente llamado "adaptación de enlace" [38].

La instalación de OpenBTS-GPRS básicamente es igual a la versión anterior. Esta desarrollado sobre la misma plataforma GSM agregando las bandas de 850 y 1900 MHz, solo que para agregar las funciones de GPRS hay que habilitarle la opción en la configuración después de estar instalado el OpenBTS.

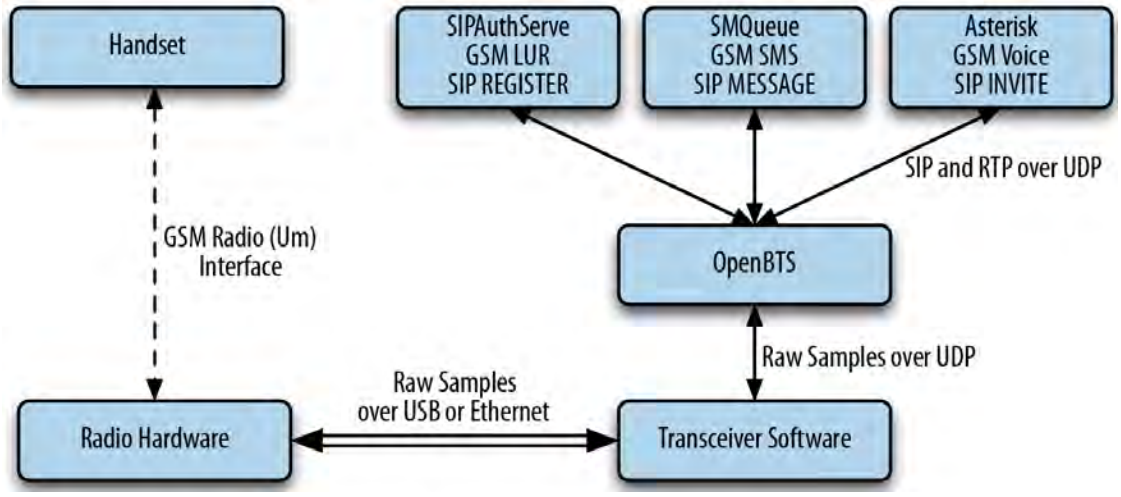


Figura 2.8 Componentes de la Arquitectura OpenBTS en GPRS [31].

Este sistema desarrolla un GPRS *attach* para autenticar el usuario en la red, pero para autenticar el usuario no es obligatorio obtener la clave secreta K. Existe una opción de autenticación parcial que se puede implementar conociendo solamente el valor IMSI del usuario.

También se puede habilitar una opción denominada "OpenRegistration". Esta es una función específica de OpenBTS en la versión 5 que proporciona una implementación sencilla para redes móviles, donde cualquier equipo puede acceder a la red, pero con acceso inicialmente restringido. Este tipo de red es muy útil en cualquier instalación donde los usuarios son temporales, o la red en sí misma solo se necesita temporalmente. Estos tipos de redes son más fáciles de implementar porque no se necesita administrador para crear cuentas y asignar números a los usuarios.

2.5.3 Estándar UMTS en la plataforma OpenBTS

OpenBTS-UMTS se basa en el marco OpenBTS, la comunicación es basada en conmutación de paquetes, donde el equipo terminal se trata como un punto final IP en la red creada. La interfaz aérea de radio está basada en modulación WCDMA, utilizando codificación QPSK a 5 MHz de ancho de banda como el estándar original.

2.5.3.1 Hardware compatible

La integración inicial de OpenBTS-UMTS se llevó a cabo con el *hardware* de la marca *Range Networks*, pero el soporte para dispositivos USRP recientes de *Ettus Research* ya está disponible, al igual que para dispositivos de *Fairwaves*.

Los productos de *Ettus Research* admitidos incluyen dispositivos USRP de tercera generación y modelos de segunda generación con ancho de banda compatible para UMTS. Los productos anteriores basados en *Ettus Research* USB 2.0 no son compatibles con OpenBTS-UMTS debido a las limitaciones en el bus de transporte.

Equipo	Bus de Trasporte
B200	USB 3.0
B210	
X300	1 o 10 Gigabit Ethernet
X310	
N200	1 Gigabit Ethernet
N210	
USRP2	

Tabla 2.3 Dispositivos de Ettus Research compatibles para UMTS [39].

2.5.3.2 Requisitos de CPU (Central Processing Unit)

OpenBTS-UMTS es una aplicación más intensiva en procesamiento que las anteriores versiones de OpenBTS, ya que el ancho de banda del canal UMTS es aproximadamente 13 veces más grande que un canal GSM. Por lo general, se requiere una CPU de alto rendimiento, como *Intel Core i3, i5 o i7* a más de 1.6 GHz. Los procesadores *Intel Atom* son demasiado débiles para admitir la implementación de UMTS [31].

2.5.3.3 Tarjeta SIM y autenticación

OpenBTS-UMTS exige autenticación mutua entre el UE y el *NodeB*. Este es un cambio importante con respecto a la autenticación en las versiones precedentes, donde este proceso es mucho más sencillo.

El registro del suscriptor necesitará conocer el valor de la clave secreta de la tarjeta SIM denominada *Ki*, para realizar la autenticación y habilitar la protección de la integridad. Sin la autenticación adecuada y la protección de la integridad, el UE no podrá realizar el proceso de *attachment* ni registrarse con OpenBTS-UMTS. Para la mayoría de los usuarios, esto significa que deben adquirir las tarjetas SIM y escribirles valores conocidos de *Ki,* para los que los UE puedan registrarse en la red. La única forma de usar tarjetas SIM de otro proveedor es obtener el valor *Ki.*

Esto también significa que algunas de las características que eludieron la autenticación en OpenBTS, como el “OpenRegistration”, o la autenticación solamente por IMSI, no son posibles con OpenBTS-UMTS.

Las tarjetas USIM (UMTS Subscriber Identify Module) actualmente no son compatibles con la implementación de OpenBTS-UMTS. Estas requieren diferentes

algoritmos de autenticación que las tarjetas SIM, y no son compatibles con la versión pública de OpenBTS-UMTS que usa el algoritmo COMP128[4].

La instalación de esta versión de OpenBTS también difiere mucho de las anteriores. Para su correcto funcionamiento necesita dependencias adicionales, pero no requiere de una instalación independiente de *Asterisk* para gestionar llamadas, ni tampoco GNU-Radio para la interconexión de bloques programables.

[4] Los algoritmos COMP128 son implementaciones de varios algoritmos. El algoritmo A3 se usa para autenticar el UE a la red. El algoritmo A8 se usa para generar la clave de sesión, que luego utiliza otro algoritmo A5 para encriptar los datos transmitidos entre el UE y la estación base [40].

CAPÍTULO 3: DESCRIPCIÓN DEL SISTEMA IMPLEMENTADO

En este capítulo se explican las claves esenciales para entender que función desempeñan los componentes *hardware* y *software* utilizados para crear una red UMTS definida por *software*.

3.1 Plataforma para el desarrollo del sistema de Radios Definidos por Software

Para realizar este proyecto se ha escogido el USRP N210 como plataforma SDR, por ser una de las compatibles para soportar el estándar UMTS. Este permite crear una interfaz de radio WCDMA, para recrear un escenario de red de telefonía celular de tercera generación.

Se conocen en el mercado otras plataformas compatibles con OpenBTS, que resultan ser soluciones muy competitivas, y no dependen de un ordenador externo para la ejecución del sistema radio. Pero en este caso contamos con el ordenador externo (*core i7*) que cumple los requerimientos de *hardware*, lo que hace que el procesamiento de datos de la red no dependa de una plataforma SDR portable con recursos muy reducidos.

El sistema SDR implementado estaría compuesto entonces por una computadora y un equipo USRP N210. Este es relativamente económico en relación a lo que costaría implementar una estación base convencional, principalmente porque permite diseñar e implementar rápidamente sistemas de Radios Definidos por Software flexibles, y pudiendo aumentar el alcance con algunos recursos más.

Figura 3.1 USRP N210 conectado a computador externo.

3.2 Arquitectura de la implementación SDR.

Esta tecnología opera sobre un transceptor digital compuesto por tres bloques básicos [19]:

- Sección de Frecuencia Intermedia (IF) (tarjeta madre)
- Sección de Radio Frecuencia (RF) *Front End* (tarjeta auxiliar)
- Sección Banda Base (BB) (Computadora)

En un sistema SDR son programables tanto el procesamiento banda base como los módulos DDC/DUC (Digital Down Converter / Digital Up Converter). Los protocolos de capa de enlace y capa física son implementados en software. En la figura 3.2 se observa la arquitectura genérica.

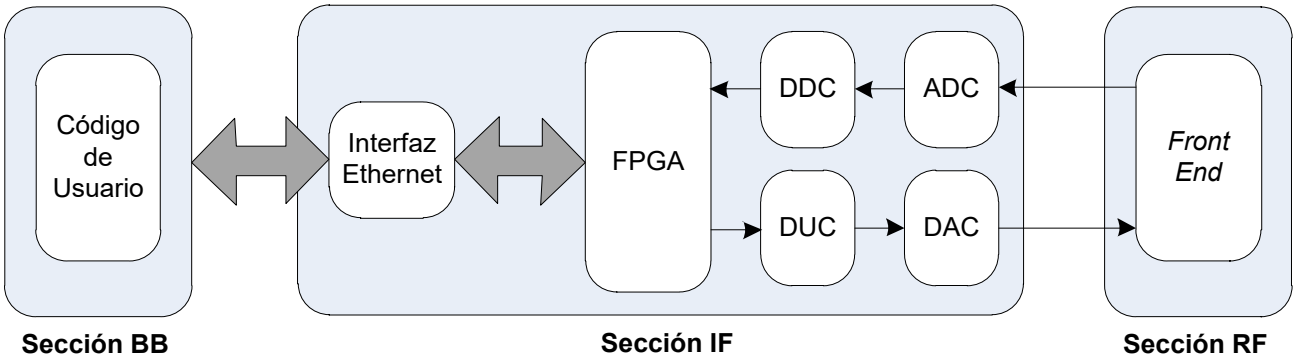


Figura 3.2 Diagrama de bloques SDR [41].

- La sección *Front End* emplea un circuito RF analógico para la transmisión y recepción de señales a una frecuencia definida.
- La sección IF realiza la conversión digital y analógica a través de convertidores DAC (Digital to Analog Converter) y ADC (Analog to Digital Converter).
    - ADC/DAC: Encargada del procesamiento digital que reemplaza el tradicional procesamiento de radio.
    - DDC/DUC: Estos elementos son los encargados de los procesos de modulación y demodulación de la señal.
- La sección BB se encuentra en la computadora donde el código es implementado.

3.3 Hardware Ettus Research USRP N210

El USRP N210 es un dispositivo desarrollado por Ettus Research LLC [42], creado con el propósito de desarrollar plataformas flexibles de SDR, a través de un ordenador externo. El USRP es montado sobre un equipo de *hardware* que genera varios tipos de señales de radio. Este dispositivo junto con OpenBTS, se emplean para enviar y recibir las transmisiones entre una estación base de telefonía celular y equipo terminal.

Los beneficios de este dispositivo abarcan: usuario, fabricante, sistemas de acceso inalámbrico, interoperabilidad, consideración de espacio, consideración de potencia, entre otros.

Principales características del modelo USRP N210 [42]:

- 50 MHz de ancho de banda con muestras de 8 bits
- 25 MHz de ancho de banda con muestras de 16 bits
- Conectividad *Gigabit Ethernet*
- Soporta conexión MIMO – Requiere 2 o más dispositivos USRP N210.
- Procesamiento FPGA en la misma tarjeta madre
- FPGA: *Xilinx Spartan* XC3SD3400A
- Convertidores Analógico-Digital/Digital-Analógico 14-bits 100 MS/s
- Capaz de conectar un reloj externo de 5 o 10 MHz de referencia
- TCXO Frecuencia de Referencia
- Opcional GPS interno.

3.3.1 Motherboard (Tarjeta Madre)

La tarjeta madre es la que contiene los elementos pertenecientes a la sección de frecuencia intermedia IF. Es el corazón del *hardware* USRP que contiene convertidores ADC, DAC, DDC, DUC y antenas. En este caso del N210, posee una interfaz *Gigabit Ethernet* que permite la comunicación con el ordenador externo, con una tasa de transferencia de datos elevada.

3.3.2 Daughterboard (tarjeta auxiliar)

Estas son las encargadas del manejo de RF en un determinado rango de operación. Actualmente existen diversos tipos de tarjetas auxiliares en el mercado. En muchos casos, para la selección de una *daughterboard* se deben tomar en cuenta los requisitos de la aplicación para la cobertura de frecuencia que se desee. En este caso que se usan las frecuencias de UMTS, se suelen ocupar varias bandas de frecuencias.

En la tabla siguiente se muestra la posible gama de frecuencias de funcionamiento (desde DC a 5,9 GHz), dependiendo del tipo de tarjeta auxiliar que se vaya a utilizar.

Modelo	Rango de Frecuencia	Función
BasicRX	1-250 MHz	Receptor
BasicTX	1-250 MHz	Transmisor
LFRX	DC - 30 MHz	Receptor
LFTX	DC - 30 MHz	Transmisor
WBX	50 - 2200 MHz	Transceptor
SBX	400 – 4400 MHz	Transceptor
RFX900	750-1050 MHz	Transceptor
RFX1200	1150-1450 MHz	Transceptor
RFX1800	1.5-2.1 GHz	Transceptor
RFX2400	2.3-2.9 GHz	Transceptor
XCVR2450	2.4-2.5 GHz	Transceptor de doble banda
	4.9-5.9 GHz	

Tabla 3.1 Lista de tarjetas auxiliarles y rango de frecuencias de operación [42].

El dispositivo con el que se despliega la interfaz de aire WCDMA es el *Ettus Research* USRP N210, y la tarjeta auxiliar que se eligió fue la SBX para que pudiera operar en las distintas bandas de frecuencia de UMTS.

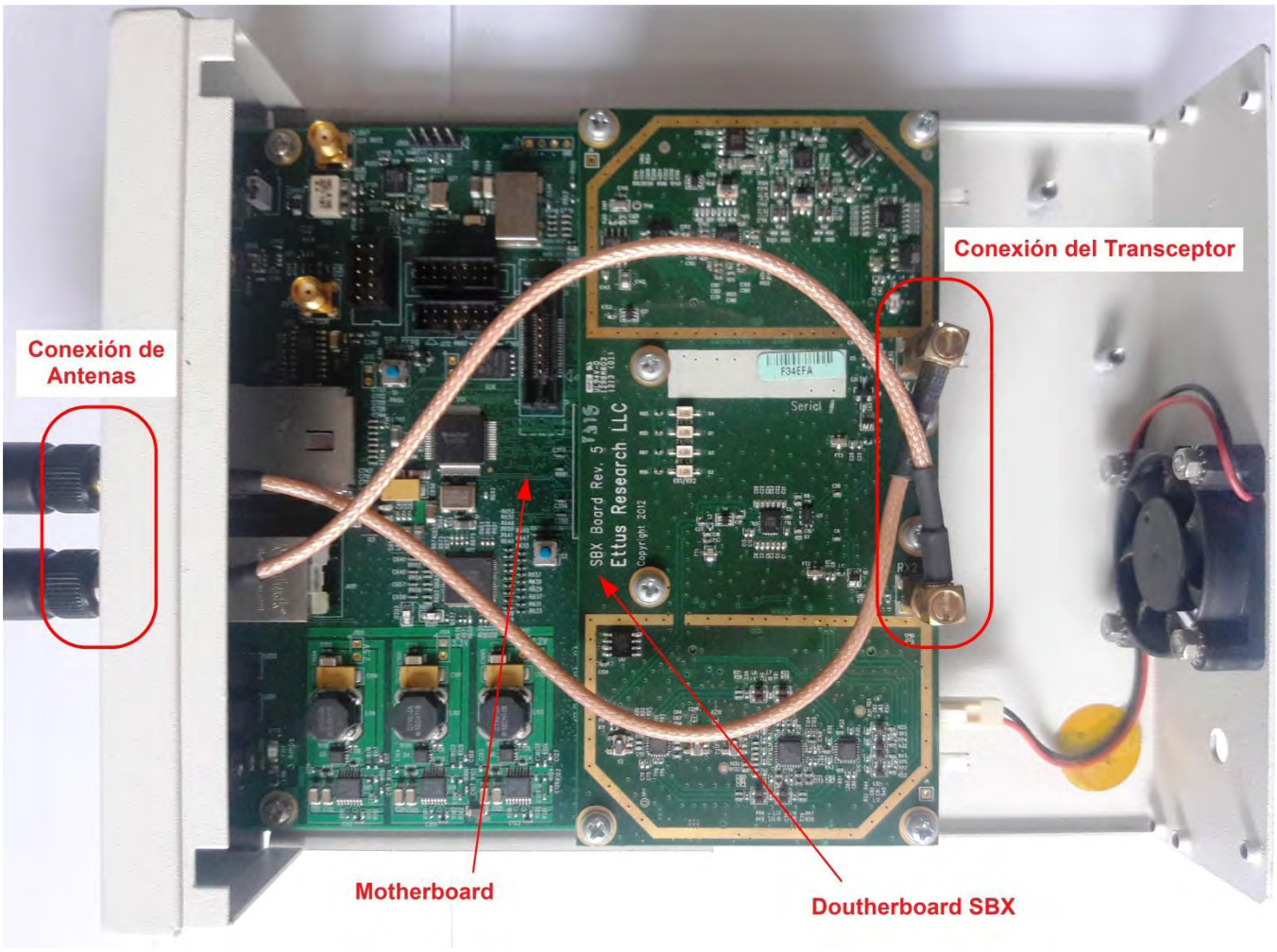


Figura 3.3 Vista interna del dispositivo USRP N210.

La tarjeta SBX funciona como un transceptor que proporciona un amplio ancho de banda, entregando hasta 100 mW de potencia de salida, y una cifra de ruido típica de 5 dB. Estos tipos de tarjetas son ideales para aplicaciones que requieren acceso a una variedad de bandas en el rango de 400 MHz a 4400 MHz [43].

Para poder desplegar la interfaz Uu en el aire se puede seleccionar cualquiera de las bandas en que trabaja OpenBTS-UMTS (850, 900, 1700, 1800, 1900 o 2100 MHz), pues todas están dentro del rango en que trabaja la tarjeta SBX.

3.3.3 Sistema radiante

Para que la tarjeta SBX pueda enviar y recibir las señales eficientemente, se conectan antenas VERT900 al dispositivo. Esta antena tiene una ganancia de 3 dBi y trabaja en el rango de frecuencias de 824-960 MHz y 1710-1990 MHz. La banda

UMTS de 2100 MHz es la única que no entra dentro de este rango, pero igual podemos usarla para operar en 5 bandas de frecuencia (850, 900, 1700, 1800 y 1900 MHz).

Figura 3.4 Antena Omni-direccional de 3dBi.

Una recomendación para la instalación de las antenas es inclinarlas para que formen un ángulo de 90 grados. Si las antenas se ubican paralelas entre sí, la señal puede fluir de manera eficiente desde la antena de transmisión a la antena de recepción, y esto es perjudicial. Si forman un ángulo de 90 grados, se evita que se alimenten tan fácilmente entre ellas, pues el patrón de radiación y la polarización en la transmisión se encuentra en un plano diferente al de recepción. En la práctica este ajuste simple reduce el ruido hasta 10 dB.

Figura 3.5 Vista del dispositivo N210 con antenas instaladas.

3.4 Especificaciones del software OpenBTS

El software OpenBTS es una aplicación para sistemas operativos *Linux* que requiere una plataforma SDR para presentar a los usuarios una interfaz radio de telefonía celular, y a la vez presentar los UE como puntos finales a una red IP [44]. El código

fuente para OpenBTS-UMTS presenta por primera vez una interfaz 3G con una plataforma programada en C++, principalmente con el objetivo de transmisión de datos. Para el presente proyecto se ha utilizado la versión pública que fue liberada recientemente libre de costos.

El código de OpenBTS contiene una colección muy amplia de componentes de código abierto que pueden utilizarse para implementar una red moderna mucho más ligera que la convencional. Se reduce así la complejidad del sistema y el aspecto económico. La versión pública tiene el soporte de una comunidad de desarrolladores y especialistas en comunicaciones móviles, dedicados al mantenimiento y desarrollo del código. En ocasiones la estructura de la información es confusa y no está actualizada.

OpenBTS crea y opera una red celular reemplazando en *software* la estación base. Para OpenBTS-UMTS no es obligatorio implementar servidor *Asterisk* externo, pues la entidad *Subscriber Registry* y sus aplicaciones internas suplen las funciones necesarias para la red UMTS.

La red creada posee una arquitectura híbrida IP donde los UE van a poder observar una interfaz radio WCDMA compatible, y a su vez, el *Core Network* podrá observar puntos finales SIP [31]. La clave para entender la integración de OpenBTS-SIP, es que en la comunicación entre un UE y la estación base, el UE aparece para el *Core Network* como un punto final SIP con el nombre de usuario “IMSIxxxxxxxxxxxxxxx”, donde las x’s son los 15 dígitos del IMSI almacenado en la SIM del UE. Por otro lado, la dirección IP del usuario SIP coincide con la dirección IP de la estación base, una dirección IP asignada al USRP N210.

En resumen, OpenBTS cumple con todos los requisitos para el desarrollo de este proyecto [44]:

- Está abierto para la investigación, siendo completamente *software*.
- Todo el *software* corre sobre una distribución *Linux* y conecta con protocolos IP, de manera que permite tener el *Core Network* en la nube.
- Permite la interconexión directa de la interfaz radio “Uu” con los protocolos de Internet.

- Los UE´s aparecen como dispositivos SIP sin la necesidad de disponer de *software* especial para los propios dispositivos.
- Los subsistemas de una red convencional pueden ser reemplazados por aplicaciones de *software* libre.

En la figura 3.6 se muestra la arquitectura *software* que sustituye la arquitectura convencional de la figura 3.7.

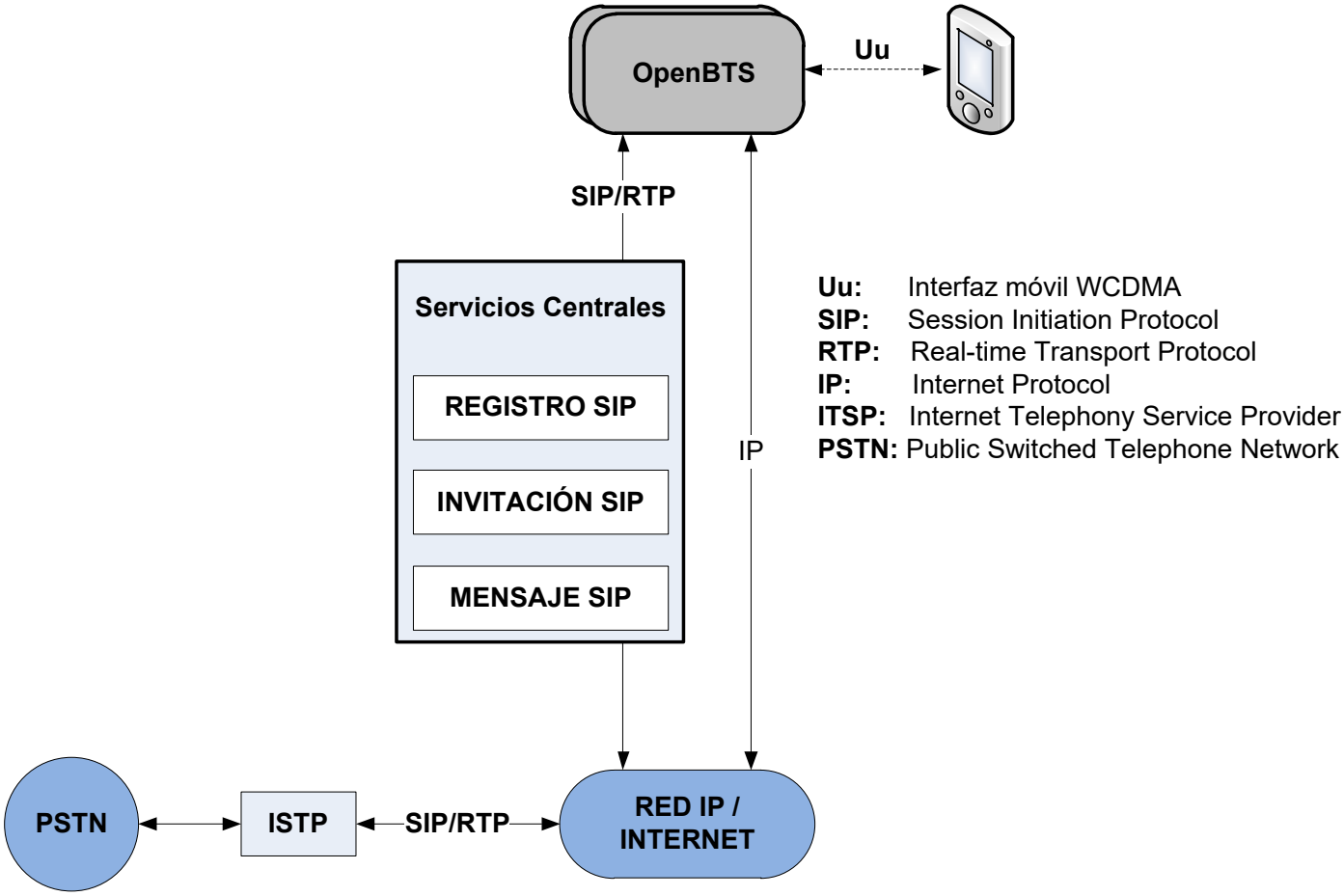


Figura 3.6 Arquitectura Híbrida IP [31].

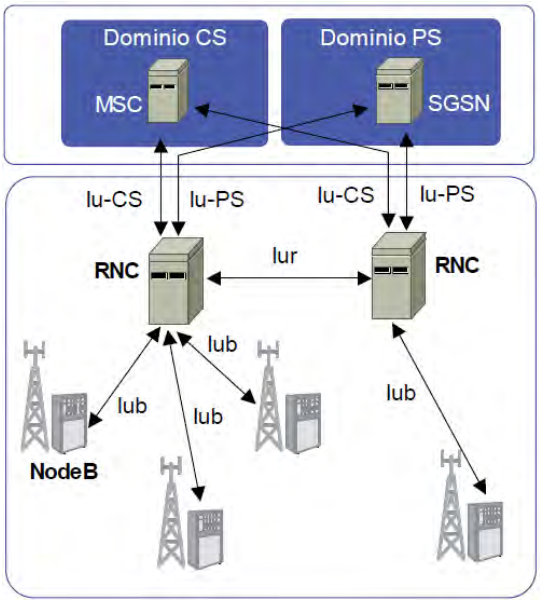


Figura 3.7 Arquitectura convencional de una red UMTS [45].

3.5 Especificaciones del software complementario "Subscriber Registry", aplicación SIPAuthServe.

SIPAuthServe es una aplicación que implementa la subscripción del usuario dentro de *Subscriber Registry*. La base datos donde los diferentes componentes almacenan y actualizan la información de los subscriptores, se denomina "sqlite3.db" y puede ser ubicada a conveniencia del administrador.

Subscriber Registry reemplaza en funciones, tanto al registro SIP de *Asterisk,* como el HLR/AuC y VLR que se encuentra en una red UMTS convencional [31] [35]. En la base de datos denominada "sipauthserve.db" se encuentran los parámetros de configuración de *Subscriber Registry*. Es posible editar dichos parámetros de la base de datos "sipauthserve.db" con la herramienta *sqlite3*, que actualmente es la única opción posible para la versión pública de OpenBTS-UMTS.

SIPAuthServe se encarga de procesar las solicitudes de registro SIP generadas por OpenBTS cuando un UE intenta registrarse a la red. Se puede observar dicha relación en la anterior figura 3.6 expuesta. Por tanto, se encarga de procesar las solicitudes y respuestas SIP necesarias entre OpenBTS y *Subscriber Registry*. Cuando un UE se ha autenticado en la red, *SIPAuthServe* actualiza la base de datos

del *Subscriber Registry* (sqlite3.db), con la dirección IP de la estación base OpenBTS con la que se ha conectado el UE.

3.6 Componentes fundamentales en el funcionamiento de la red UMTS implementada

En la siguiente figura se observan todos los componentes que juegan un papel fundamental para poner en marcha una red UMTS operativa, y poder así registrar los terminales móviles y establecer la comunicación.

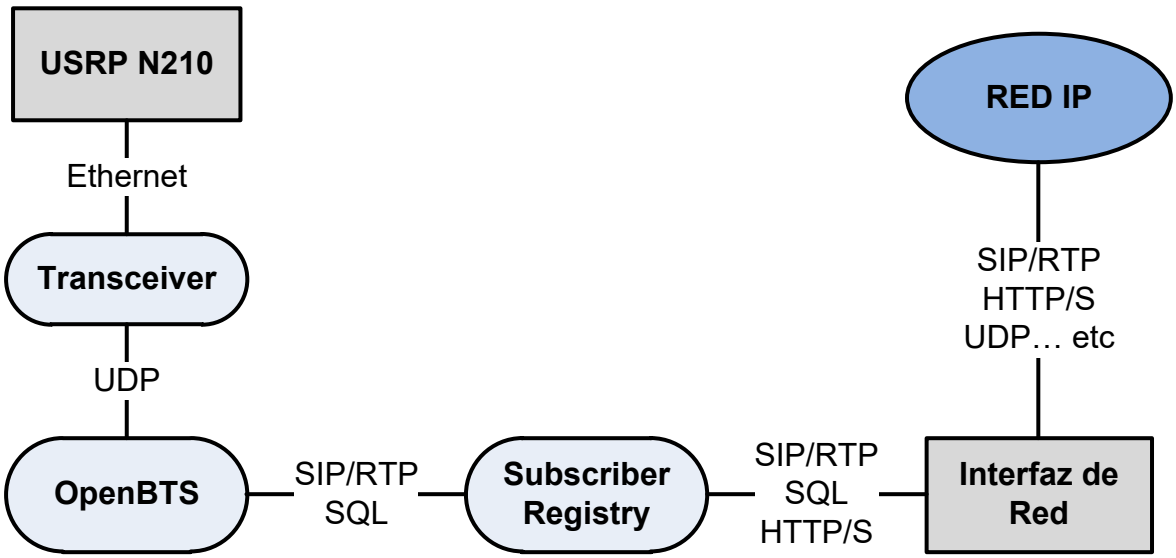


Figura 3.8 Arquitectura por componentes.

En este caso formamos la red más simple con un único punto de acceso, pues la versión pública de OpenBTS-UMTS no soporta *handover*. Todas las aplicaciones del conjunto se ejecutan dentro del punto de acceso en la misma computadora incorporada.

Si se quisieran separar las funciones por algún motivo de infraestructura, las aplicaciones *OpenBTS* y *Transceiver* se deben ejecutar dentro del mismo punto de acceso; es decir, en la computadora que está conectada al USRP. En el caso de *Subscriber Registry* (con *SIPAuthServe*), este realiza la comunicación a través del sistema de archivos, pero puede estar instalado en una computadora remota desde el punto de acceso. Si se quisiera instalar un servidor *Asterisk* para desarrollar un

sistema que pudiera manejar un mayor tráfico de llamadas VoIP, también pudiera ejecutarse en cualquier lugar y pueden tener varias instancias, pero siempre debe tener conectividad con el punto de acceso de la radiobase UMTS.

Los componentes fundamentales que componen la estación base serían los siguientes:

USRP N210: Dispositivo principal de *hardware* que hace la función de transceptor digital de radio en las bandas de frecuencia de UMTS (850, 900, 1700, 1800, 1900 y 2100 MHz)

Transceiver: Aplicación que realiza la función de radio-modem y gestiona la interfaz de comunicación *ethernet* con USRP

OpenBTS: Aplicación que se encarga del procesado de los protocolos de comunicación con diferentes entidades, y también utiliza Lenguaje de Consulta Estructurada (SQL, siglas en inglés) para consulta de base de datos.

- UDP en la comunicación con la aplicación Transceiver
- SIP, RTP y SQL en la comunicación con *Subscriber Registry*

Subscriber Registry: Es una de las entidades más importantes en la versión OpenBTS-UMTS, pues es el centro de toda la comunicación para el intercambio de datos. Maneja la información referente a consulta y actualización de bases de datos mediante SQL, pero también utiliza para la comunicación los protocolos SIP, RTP y HTTP/S.

Interfaz de Red: Dispositivo interno de la computadora mediante el cual se accede a la red IP (*ej.* Internet). En este caso es una tarjeta de red inalámbrica porque se accede a través de una conexión *WiFi*.

Red IP: Red de área local desde donde se puede tener acceso a servicios habilitados y recursos compartidos.

Componentes Adicionales: Existe un componente adicional que viene incorporado en los códigos de *Subscriber Registry* y de OpenBTS-UMTS denominado “NodeManager”. Esta es una interfaz de control común utilizada para la

configuración y supervisión del sistema. Las API (Application Programming Interface) de *NodeManager* se dividen en dos categorías principales: solicitud/respuesta y transmisión. Ambos tipos usan mensajes con un formato específico, y se comunican a través de una biblioteca que crea conexiones de *socket*[5] simples y robustas entre procesos [46]. Las API de *NodeManager* se implementaron para simplificar la administración remota de múltiples instancias de OpenBTS, así como otros servicios centrales. En el caso de esta red, es utilizado para visualizar, agregar y/o eliminar los usuarios que se registran en la red 3G implementada.

3.7 Tarjeta SIM y dispositivo programador

Un aspecto importante para este escenario donde se plantea una red UMTS, es la forma en que el usuario se registra a la red. Para esto se requiere contar con una tarjeta SIM. Esta tarjeta proporcionará una conexión cifrada con la estación base, lo cual le dará seguridad a cada uno de los usuarios habilitados para usar la red UMTS. Con el desarrollo de los estándares de la telefonía celular hasta la fecha, se han implementado los siguientes algoritmos de seguridad, que se encuentran listados en orden de complejidad y seguridad:

- COMP128 v1
- COMP128 v2
- COMP128 v3
- MILENAGE
- KASUMI

COMP128 v1 es el más sencillo y KASUMI el más complejo. Cada uno de los ellos agrupan una serie de modelos matemáticos que se encargan de cifrar, tanto el registro del usuario a la estación base, como la comunicación en su totalidad [47].

[5] Socket: Es un punto final interno que también se define como una forma de recurso del sistema. Se usa como referencia para enviar o recibir datos dentro de un nodo en una red informática. Generalmente se representa con un número IP seguido del puerto de comunicación, *ej*. 172.0.0.1::5064 [48].

Todos los algoritmos complementan una serie de normativas que regulan la estandarización de las tecnologías de telefonía celular. El software OpenBTS al tratarse de un software en pleno desarrollo, trabaja con parámetros de seguridad básica como lo es el algoritmo COMP128 v1. Es por esto que las tarjetas SIM que no implementan este tipo de algoritmo, no son compatibles con la red OpenBTS.

3.7.1 Selección del tipo de tarjeta SIM.

Una de las partes cruciales de este proyecto es la correcta selección de los componentes a utilizar, y entre estos se encuentran la selección del tipo de tarjeta SIM. Para lograr con éxito el registro del UE a la red UMTS, se requiere una SIM virgen de determinadas especificaciones, pues como se menciona anteriormente, OpenBTS-UMTS solo es compatible con tarjetas SIM que usen el algoritmo COMP128 v1.

La tarjeta elegida es la llamada “Super SIM X-Sim”.

Figura 3.9 Tarjeta SIM elegida para la autenticación en la red.

Esta SIM específica es únicamente compatible con el algoritmo COMP128 v1, y también debe de ser configurada en su totalidad, ya que no posee parámetros programados por defecto [49]. Otra característica que posee, aunque en este proyecto no la utilicemos, es que tiene soporte para 16 números telefónicos diferentes, particularidad que puede ser aprovechada en implementaciones futuras. Algo importante que hay que tener en cuenta, es que prácticamente todos los celulares de hoy en día, el tipo de tarjeta que usan son “nano SIM” o “micro SIM”. Por este motivo se debe disponer de un cortador de SIM para ajustarlas a la medida indicada, y poderlas insertar correctamente en el terminal móvil.

Figura 3.10 Herramienta cortadora de tarjetas SIM (SIM cutter).

3.7.2 Dispositivo Lector/Programador "Smart Card Writer"

Todas las tarjetas SIM del mercado son construidas mediante un estándar internacional denominado ISO/IEC 7816 [50]. Este estándar define todas las características tanto de *hardware* como de *software*, especificando de manera conjunta como vienen distribuidos: los contactos metálicos, las dimensiones físicas, los protocolos de transmisión de datos y seguridad, distribución eléctrica, registros, comandos, entre otros [51].

Para realizar la programación de la tarjeta SIM, se debe contar con un dispositivo lector/programador que sea compatible con el estándar antes mencionado. Este dispositivo se encargará de escribir los datos necesarios en la tarjeta SIM, de manera que pueda ser reconocida por la red UMTS.

En este proyecto se utilizó como dispositivo programador el *Smart Card Reader/Writer* PC/SC USB-MCR3512, que soporta la norma ISO 7816 clase A y AB para tarjetas SIM [52].

Figura 3.11 Dispositivo programador de Tarjetas SIM (Smart Card Writer).

3.7.3 Software para programar tarjetas SIM, "pySIM"

Para realizar la programación de la tarjeta SIM, es posible hacer uso de cualquier *software* que sea compatible con la ISO 7816. En este trabajo se hace uso de uno que está desarrollado para sistemas operativos *Linux,* denominado "pySim". Este *software* es basado en lenguaje de programación *Python*, resultando muy simple realizar cambios y actualizaciones en su estructura. Mediante una sencilla línea de código se ejecuta la escritura de los datos requeridos en la tarjeta SIM.
Las tarjetas compatibles con este *software* son las siguientes [53]:

- sysmoUSIM-SJS1
- GrcardSIM
- GrcardSIM2
- MagicSIM

El tipo de tarjeta SIM escogida para el proyecto (*Super SIM X-Sim*), clasifica dentro de la categoría *MagicSIM*, y por lo tanto es compatible con el *software*. Para su instalación se requiere la previa instalación de dependencias, requeridas tanto para el funcionamiento del *Smart Card Writer*, como del software *pySIM*.

CAPÍTULO 4: DESARROLLO E IMPLEMENTACIÓN

El propósito de este proyecto de tesis es la implementación de una radiobase de telefonía celular que permita recrear redes de telecomunicaciones de tercera generación, basada específicamente en el estándar UMTS (Release 99). Para lograr esto se hace uso de la técnica SDR por medio de la plataforma libre OpenBTS, y el *hardware* USRP N210 en el sistema operativo Ubuntu 16.04 LTS.

El escenario consiste en una red híbrida de telefonía celular, donde el *software* OpenBTS es el encargado de hacer que el USRP N210 se comporte como un *NodeB,* y genere una interfaz aérea Uu. Una vez instalado el controlador UHD (USRP Hardware Driver) para la comunicación con el USRP, y verificado su conexión con el computador, se procede a la compilación del código de OpenBTS, cuyo proceso genera los programas necesarios para poder implementar la red UMTS completa.

Con el fin de que los terminales móviles se puedan registrar en la red, se presenta un proceso de programación de la tarjeta SIM. A esta se le escribe un valor determinado de clave secreta *Ki*, que es obligatorio que el sistema lo conozca para así lograr generar un registro completo de los equipos móviles en la red 3G.

Antes de iniciar la descarga e instalación de los programas para establecer este escenario, primero es necesario asegurar que dentro del sistema operativo se encuentren las dependencias, controladores y librerías para la correcta implementación y funcionamiento.

4.1 Dependencias Generales

En las versiones anteriores de OpenBTS este proceso de instalación de dependencias es mucho más sencillo. Al descargar el código existe un archivo de compilación que, al ejecutarlo, instala todas las dependencias necesarias y descarga cada programa que forma parte de la red. Para el caso de OpenBTS-UMTS que aún está en su primera versión, se instala cada componente y sus dependencias por separado.

Para realizar la instalación tenemos que tener privilegios administrativos y estar en modo super-usuario, así que para no tener que escribir el comando *sudo* antes de cada línea que se vaya a ejecutar, se escribe entonces lo siguiente en la terminal:

```
~$ sudo su
[sudo] password for user: *contraseña*
~#
```

A partir de este momento estamos en modo super-usuario.
Primeramente, se prepara el sistema operativo para evitar que pueda tener algún problema de actualización:

```
~# apt-get -y update
~# apt-get -y upgrade
```

El comando "-y" se usa para no tener que confirmar cada instalación. Este se puede omitir, pero hay que estar atento a la instalación para confirmar cada vez que se vaya a instalar un componente.
Luego se instalan las librerías:

```
~# apt-get -y install git autoconf libortp-dev libusb-1.0-0-dev g++ sqlite3 libsqlite3-dev erlang libreadline6-dev libncurses5-dev libboost-dev ntp screen autoconf libtool debhelper dh-virtualenv python-all-dev libboost-dev libortp-dev libfontconfig1-dev libxrender-dev libpulse-dev swig libsdl1.2-dev git-core guile-1.8-dev libqt4-dev python-numpy ccache python-opengl libgsl0-dev python-cheetah python-lxml qt4-dev-tools libqwt5-qt4-dev libqwtplot3d-qt4-dev pyqt4-dev-tools python-qwt5-qt4 ssh g++ libxml2-dev gawk libxi-dev
```

Puede ser que en la distribución de Linux que se use, ya estén instaladas algunas de estas librerías, pero hay dos en específico que no pueden faltar, de lo contrario cuando compilemos el código de OpenBTS arroja un error.

```
~# apt-get install libtool-bin
~# apt-get install libboost-all-dev
```

Se añaden los siguientes repositorios también:

```
~# add-apt-repository ppa:chris-lea/zeromq
~# add-apt-repository ppa:git-core/ppa
```

Algunas dependencias no aparecen en el repositorio de Ubuntu, por eso es necesario agregarlas al repositorio o usar administradores de paquetes. En este caso se usó *synaptic*.

```
~# apt-get install synaptic
~# synaptic
```

Como se observa en la Figura 4.1, al ejecutar *synaptic* aparece una ventana. En la barra de filtro rápido (recuadro azul) se buscan las siguientes dependencias, y luego se marcan para instalación.

- libdbi-dev
- libdbd-pgsql
- bind9

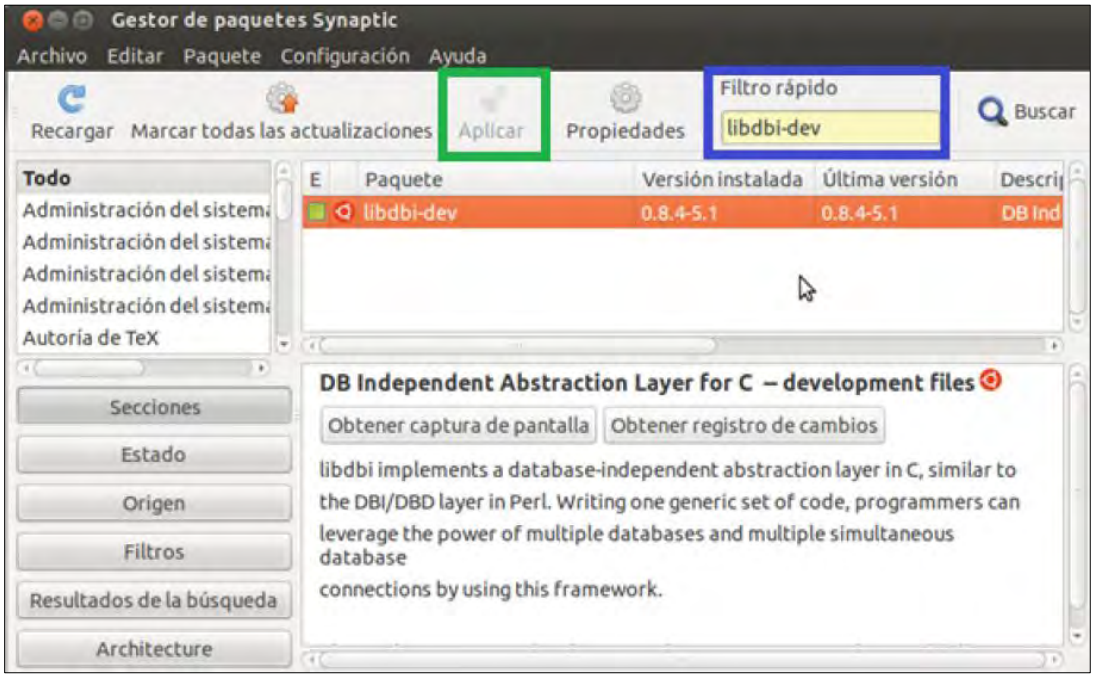


Figura 4.1 Administrador de paquetes synaptic.

4.2 Dependencias específicas

Existen también algunos componentes que son considerados requisitos fundamentales para el correcto funcionamiento y operabilidad de la red.

4.2.1 Libosip

Esta es una librería que tiene como objetivo proporcionar la iniciación y control de las sesiones interactivas en diferentes usuarios para servicios multimedia. Se usa como protocolo de señalización para voz sobre IP y es fundamental para la interoperatividad.

Se puede instalar de modo sencillo con el comando:

```
~# apt-get install libosip2-dev
```

pero en dependencia del sistema operativo, presenta algunos errores. En este caso se instaló a través de su código fuente descargado, y se usó la versión 3.5.0:

```
~# wget http://ftp.gnu.org/gnu/osip/libosip2-3.5.0.tar.gz
~# cd ~carpeta-de-descarga/
~# tar -xzvf libosip2-3.5.0.tar.gz    (descomprimir)
~# cd libosip2-3.5.0
~# ./configure
~# make
~# make install
```

4.2.2 UHD - USRP Hardware Driver

Las librerías del UHD se encargan de controlar y detectar el USRP conectado al ordenador. En este proyecto se usó la versión 3.10.2.0 compatible con el USRP N210 (USRP2 series), y para su funcionamiento se requieren las siguientes dependencias:

```
~# apt-get -y install cmake sdcc libaudio-dev libboost-all-dev python-mako
doxygen python-docutils build-essential
```

De todas las versiones publicadas del UHD desarrolladas por Ettus Research, se selecciona la que sea compatible con el USRP y se descarga el archivo. Para este caso se escogió *uhd-3.10.2.0.tar.gz* como se observa en la figura 4.2 (recuadro azul).

Figura 4.2 Descarga del controlador UHD.

Una vez descomprimido el archivo se crea una carpeta llamada “build” en donde se compila e instala el controlador UHD:

```
~# cd ~carpeta-de-descarga/
~# tar -xzvf uhd-3.10.2.0.tar.gz    (descomprimir)
~# cd uhd-3.10.2.0
~# mkdir build                     (crea carpeta “build”)
~# cd build
~# cmake ../
~# make
~# make install
```

Al ejecutar el comando “cmake ../” aparte de construir los ficheros para poder instalar el UHD, también permite ver el campo “UHD enabled components” el cual contiene los USRP compatibles con esta versión, ver figura 4.3. A continuación se verifica la compatibilidad del USRP N210 (serie USRP2) con la versión 3.10.2.0, y se prosigue a su instalación con los comandos “make” y “make install”.

Figura 4.3 Componentes habilitados del UHD.

Una vez instalado el UHD, es necesario descargar las imágenes en donde se encuentran cada uno de los *firmwares* que usan las FPGA internas del USRP. Para esto se escriben los siguientes comandos:

```
~# /usr/local/share/uhd/utils/uhd_images_downloader.py
```

o en otros casos:

```
~# /usr/local/lib/uhd/utils/uhd_images_downloader.py
```

Al terminar la descarga de las imágenes se debe verificar que la carpeta “images” se encuentre en las siguientes direcciones:

- /usr/local/share/uhd
- /usr/local/lib/uhd

4.2.3 Libcoredumper

OpenBTS usa la biblioteca compartida de *coredumper,* para producir información de depuración significativa si OpenBTS falla. En realidad, hay dos paquetes *libcoredumper:*

- *libcoredumper-dev:* Contiene los archivos de desarrollo necesarios para compilar programas que utilizan la biblioteca *coredumper.*
- *libcoredumper:* contiene la biblioteca compartida que carga las aplicaciones al momento de ejecutarlas

Para instalar este complemento se ejecutan los siguientes comandos:

```
~# git clone https://github.com/RangeNetworks/libcoredumper.git
~# cd libcoredumper
~# ./build.sh
~#  dpkg  -i  libcoredumper1_1.2.1-1_amd64.deb  libcoredumper-dev_1.2.1-
1_amd64.deb
```

4.2.4 Enlace de python PyZMQ

Esta es una librería de mensajería asíncrona de alto rendimiento, destinada al uso en aplicaciones distribuidas o concurrentes. Posee una API que proporciona *sockets* que representan conexiones entre puntos finales dentro de un código, y requiere que se use un patrón de mensajes determinado. En OpenBTS es necesaria para poder agregar los usuarios a la red a través del componente *Nodemanager,* aspecto que se abordará en la sección 4.6.

La distribución binaria de *PyZMQ* debe venir con la instalación de la librería *libzmq*, pero luego en el momento de adicionar un usuario suele fallar. Se recomienda entonces instalarla independiente para mayor seguridad mediante los siguientes comandos:

```
~# apt install python-pip
~# git clone https://github.com/zeromq/pyzmq.git
~# pip install pyzmq
```

4.3 Verificación del hardware USRP N210

Es importante destacar que para la correcta comunicación entre el USRP N210 y la computadora, el enlace debe tener una capacidad de 1 Gbps. Esto quiere decir que la interfaz de red del ordenador debe ser al menos *Gigabit Ethernet.* Con interfaz *Fast Ethernet* no se puede lograr la correcta comunicación.

Para verificar esta conexión se usan dos comandos básicos que se emiten desde la terminal:

```
~# uhd_find_devices
~# uhd_usrp_probe
```

El primer comando muestra el tipo de USRP, la versión del UHD y la dirección IP asignada. El segundo muestra características del *hardware* como *motherboard*, *daughterboard*, rango de frecuencia, *etc*.

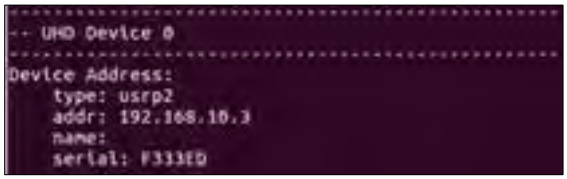


Figura 4.4 Salida al emitir el comando uhd_find_devices en la terminal.

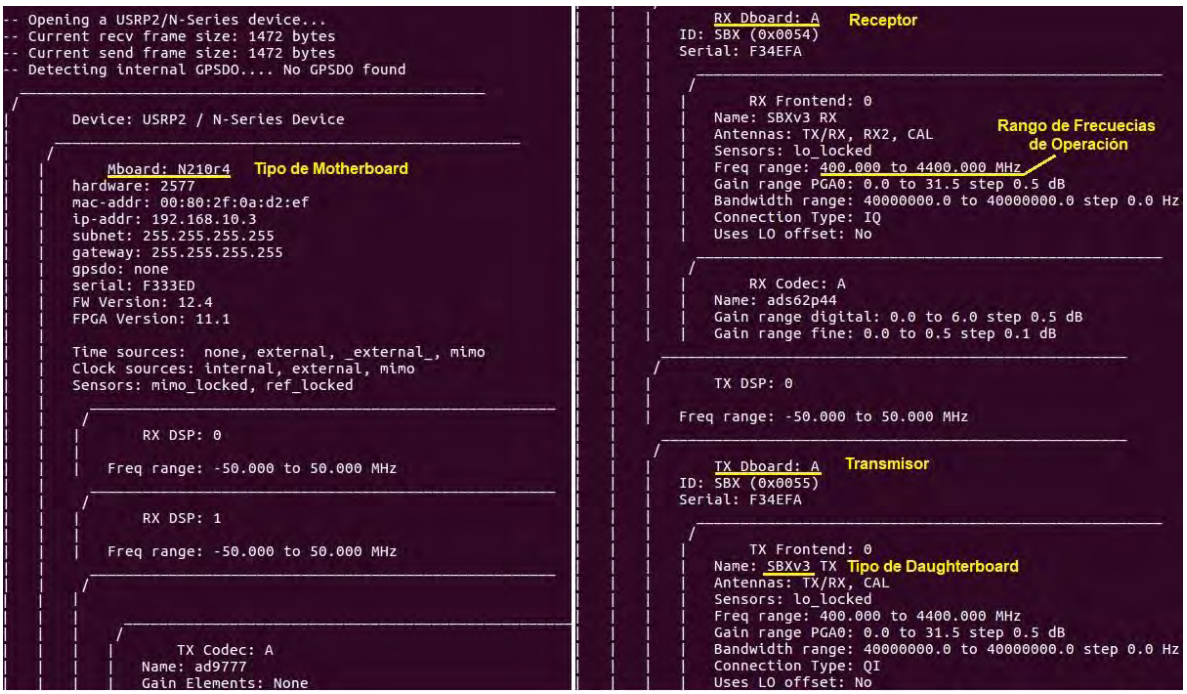


Figura 4.5 Salida al emitir el comando uhd_usrp_probe en la terminal.

4.4 Instalación de OpenBTS-UMTS y compilador ASN.1-C

El código de la versión 1.0 de OpenBTS-UMTS es libre de costos y está disponible en los repositorios de *GitHub:*
https://github.com/RangeNetworks/OpenBTS-UMTS.git
Para tener acceso a este sitio tenemos que crear una cuenta, y para descargarlo hay que tener instalada la librería *git*, actualizada con una versión superior a la 1.8.2. Esta es una de las dependencias generales que ya está instalada, y para comprobar la versión se emite el comando:

```
~# git version
```

que devuelve la versión que está instalada:

```
git version 2.17.0
```

El siguiente paso es descargar el código y comenzar su instalación:

```
~# git clone https://github.com/RangeNetworks/OpenBTS-UMTS
~# cd OpenBTS-UMTS
~# git submodule init
~# git submodule update
```

Para la correcta instalación y funcionamiento debemos instalar un compilador denominado “ASN.1-C”. Esta herramienta la necesitamos para convertir las especificaciones ASN.1[6] en códigos de programación C.
Existen muchas versiones de este compilador, pero la que está probada que funciona con OpenBTS-UMTS es la 0.9.23. Cuando se descarga el código de OpenBTS, este compilador se encuentra dentro en una carpeta comprimida llamada “asn1c-0.9.23.tar.gz”:

```
~# tar -zxvf asn1c-0.9.23.tar.gz              (descomprimir)
~# cd vlm-asn1c-0959ffb/
~# ./configure
~# make
~# make install
```

[6] ASN.1: Abstract Syntax Notation One (notación sintáctica abstracta 1) es una norma para representar datos independientemente de la máquina que se esté usando, y sus formas de representación internas. Es un protocolo de nivel de presentación en el modelo OSI [54].

Ahora todo está listo para compilar el código de OpenBTS-UMTS. Se recomienda que la compilación se realice con el USRP conectado a la computadora:

```
~# cd ..                    (de vuelta al directorio OpenBTS-UMTS)
~# ./autogen.sh
~# ./configure
~# make
~# make install
```

4.5. Creación de bases de datos y preparación del software

Después de culminado el proceso anterior se deben crear algunas carpetas y bases de datos de trabajo.

Se crea una carpeta para las variables temporales:

```
~# mkdir /var/log/OpenBTS-UMTS
```

Para cargar la base de datos con los parámetros que se definan, hay que hacerlo a través de un archivo llamado “OpenBTS-UMTS.example.sql”. Este archivo se encuentra en la carpeta *apps*, y se recomienda copiarlo para el directorio donde va a estar esta base de datos:

```
~# cp ~/OpenBTS-UMTS/apps/OpenBTS-UMTS.example.sql /etc/OpenBTS/
```

Nos situamos dentro de la carpeta y cargamos los datos:

```
~# cd /etc/OpenBTS
~# sqlite3 OpenBTS-UMTS.db ".read OpenBTS-UMTS.example.sql"
```

Cada vez que se quieran modificar parámetros en la configuración, la recomendación que se hace es:

- Modificar los parámetros en el fichero OpenBTS-UMTS.example.sql
- Cargar el fichero en la base de datos:

  ```
  sqlite3 OpenBTS-UMTS.db ".read OpenBTS-UMTS.example.sql"
  ```
- Reiniciar OpenBTS

  Existen formas dentro de la consola OpenBTS que permiten modificar algunas configuraciones, pero muchas de ellas arrojan errores que para solucionarlos, hay que borrar la base de datos y volverla a cargar.

  En un principio se configuran los siguientes valores en este fichero:

  ```
  ‘GGSN.DNS’ ajustar a ‘8.8.8.8’ para habilitar el DNS de Google.
  ```

'GGSN.Firewall.Enable' ajustar a '0' para deshabilitar el *firewall*.
'UMTS.Radio.Band' - configurar la banda que se va a usar.
'UMTS.Radio.C0' – configurar el valor de UARFCN.
'GGSN.MS.IP.Base' – configurar valor '192.168.99.1'. IP inicial para asignar a los usuarios
'SIP.Local.IP' – configurar valor '127.0.0.1. IP que hace función de proxy para OpenBTS-UMTS
'SIP.Local.Port' – configurar valor '5062'. Puerto que OpenBTS-UMTS usa para la interfaz SIP.
'SIP.Proxy.Registration' – configurar valor '127.0.0.1:5064'. Socket que se usa para el registro y autenticación.
'SubscriberRegistry.A3A8' – configurar valor '/OpenBTS/comp128'. Ruta de la aplicación que implementa el algoritmo de autenticación y cifrado.
'SubscriberRegistry.Port' – configurar valor '5064'. Puerto usado por el servidor de autenticación SIP.
'SubscriberRegistry.db' – configurar valor '/var/lib/asterisk/sqlite3dir/sqlite3.db'. Ruta donde se encuentra la base de datos del suscriptor.
'TRX.IP' – configurar valor '127.0.0.1'. IP de aplicación transceiver.
'TRX.Port' – configurar valor '5700'. Puerto usado por la aplicación transceiver.
'UMTS.Identity.MCC' – configurar valor '001'. Debe coincidir con el MCC del IMSI.
'UMTS.Identity.MNC' – configurar valor '01'. Debe coincidir con el MNC del IMSI.

El siguiente paso es copiar la aplicación *transceiver* en varios directorios:

```
~# cp ~/OpenBTS-UMTS/TransceiverUHD/transceiver ~/OpenBTS-UMTS/
~# cp ~/OpenBTS-UMTS/TransceiverUHD/transceiver ~/OpenBTS-UMTS/apps/
```

Esto se hace porque dentro del código la aplicación es llamada en esos directorios, y solamente se encuentra en la carpeta TransceiverUHD. La otra forma de solucionarlo es sustituyendo la dirección de ubicación dentro del código, pero esto requiere modificar las líneas una por una.

También se debe establecer la configuración para reenviar correctamente los datos entre los dispositivos, el equipo host y las redes IP:

```
~# iptables -t nat -A POSTROUTING -j MASQUERADE -o wlp3s0
~# echo 1 > /proc/sys/net/ipv4/ip_forward
```

En este caso "wlp3s0" es el nombre de la interfaz WiFi de la computadora que tiene acceso a la red IP. Si la conexión es a través de otra interfaz (*ej*. ethernet), se debe cambiar wlp3s0 por el nombre de la interfaz aplicable (*ej*. eth0). Para conocer las interfaces del ordenador se debe escribir en la terminal el comando:

```
~# ifconfig
```

Hay que tener en cuenta que con esta configuración, cada vez que el equipo es reiniciado, hay que volver a escribir los comandos en la terminal.

Para que este reenvío de datos sea efectivo en todo momento, sin importar si el equipo es reiniciado, se debe crear un fichero llamado “iptables.rules” en el directorio */etc/OpenBTS/* e insertarle las siguientes líneas:

```
*nat
:PREROUTING ACCEPT [0:0]
:POSTROUTING ACCEPT [0:0]
:OUTPUT ACCEPT [0:0]
-A POSTROUTING -o wlp3s0 -j MASQUERADE
COMMIT
# Generated by iptables-save v1.4.4
*filter
:INPUT ACCEPT [0:0]
:FORWARD ACCEPT [0:0]
:OUTPUT ACCEPT [0:0]
COMMIT
```

Para que cada vez que se inicie la computadora tenga efecto el reenvío de datos, hay que editar el archivo “interfaces”, ubicado en el directorio */etc/network/* insertando al final la siguiente línea:

```
pre-up iptables-restore < /etc/OpenBTS-UMTS/iptables.rules
```

4.6 Instalación de Subscriber Registry y suscripción de terminales móviles

Mediante este proceso se habilita la API del registro del suscriptor y el servidor de autenticación SIP, para poder iniciar OpenBTS-UMTS. Las funciones más importantes que desempeña *Subscriber Registry* son: Controlar la base de datos de suscriptores, y funcionar como HLR en la red UMTS.

El proceso para su instalación es parecido al de OpenBTS-UMTS. Los comandos se muestran a continuación:

```
~# git clone https://github.com/RangeNetworks/subscriberRegistry.git
~# cd subscriberRegistry
~# git submodule init
~# git submodule update
~# NodeManager/install_libzmq.sh
~# autoreconf -i
~# ./configure
```

```
~# make
~# make install
```

Luego de la compilación del código se procede a crear el directorio para las bases de datos:

```
~# mkdir /var/lib/asterisk/
~# mkdir /var/lib/asterisk/sqlite3dir/
```

A continuación, se copia la aplicación *comp128* en diferentes directorios para computar el proceso de autenticación de los usuarios:

```
~# cp ~/subscriberRegistry/apps/comp128 ~/OpenBTS-UMTS/
~# cp ~/subscriberRegistry/apps/comp128 ~/OpenBTS-UMTS/apps/
~# cp ~/subscriberRegistry/apps/comp128 /OpenBTS/
```

Al igual que en la instalación de OpenBTS-UMTS se carga una base de datos inicial, con la diferencia de que esto se hace solamente una vez.
El archivo “subscriberRegistry.example.sql” es configurado con los siguientes valores:

```
‘SubscriberRegistry.A3A8’ – configurar valor '/OpenBTS/comp128'. Ruta de
la aplicación que implementa el algoritmo de autenticación y cifrado.
‘SubscriberRegistry.Port’ – configurar valor '5064'. Puerto usado por el
servidor de autenticación SIP.
‘SubscriberRegistry.db’ – configurar valor
'/var/lib/asterisk/sqlite3dir/sqlite3.db'. Ruta donde se encuentra la
base de datos del suscriptor.
```

Luego se cargan los valores en la base de datos:

```
~# cd /etc/OpenBTS/
~# sqlite3 -init subscriberRegistry.example.sql sipauthserve.db
```

Después de ejecutar este comando se entra a una interfaz de comandos de base de datos (>), de donde se sale con el comando:

```
>  .quit
```

Al código particular del archivo “sipauthserve.cpp” se le realizaron varios cambios, incluyendo la visualización del proceso de autenticación del usuario (anexo 1) .
Una vez realizado esto se procede a añadir los usuarios que se pueden autenticar en la red. Esto se hace a través del componente “NodeManager”:

```
~# cd ~/subscriberRegistry/NodeManager/
```

El comando que se emite para adicionar al usuario tiene el siguiente formato:

```
./nmcli.py sipauthserve subscribers create “name” imsi msisdn ki
```

Para añadirlo se escribe con los valores que se van a asignar:

```
~# ./nmcli.py sipauthserve subscribers create "Magic" IMSI001010000000021
1021 1108306a830921b09469c910226dcafe
```

Donde “Magic” es el nombre asignado, **IMSI001010000000021** es el valor IMSI de la tarjeta SIM, **1108306a830921b09469c910226dcafe** es el valor de la clave secreta Ki, y 1021 es el *msisdn,* que viene siendo el número telefónico que se le está asignando. Si los valores se guardaron correctamente, como respuesta del comando emitido se debe obtener lo siguiente:

```
raw request:
{"command":"subscribers","action":"create","fields":{"name":"Magic","imsi
":"IMSI001010000000021","msisdn":"
1005","ki":"1108306a830921b09469c910226dcafe"}}
raw response: {
      "code" : 200,
      "data" : "both ok"
```

Nota importarte: El IMSI se tiene que asignar con las letras **IMSI** delante del valor, de lo contrario el UE no va a poder registrarse en la red. Otro detalle es que estos comandos en *NodeManager* se tienen que ejecutar con la aplicación *SIPAuthServe* activa, de lo contrario no se hace efectivo hasta que no se ejecute la aplicación.
Para comprobar los usuarios que están en la base de datos se debe emitir el siguiente comando:

```
~# ./nmcli.py sipauthserve subscribers read
```

Deben aparecer los usuarios que se han añadido:

```
            {
                 "imsi" : "IMSI001010000000021",
                 "msisdn" : "1021",
                 "name" : "Magic"
            },
```

Para eliminar un usuario se puede proceder como en el siguiente ejemplo:

```
~# ./nmcli.py sipauthserve subscribers delete imsi IMSI001010000000021
```

4.7 Programación de la tarjeta SIM

Como se describe en la sección 3.7.3, el software que se ocupa para la programación de la tarjeta SIM es el *pySim*. Primeramente, se deben instalar las dependencias para el correcto funcionamiento del *software*, y para que la computadora haga el reconocimiento del *Smart Card Writer*:

```
~# apt-get install pcscd pcsc-tools libccid libpcsclite-dev python-
pyscard
```

Finalizada la instalación de las dependencias, el ordenador será capaz de reconocer el lector/programador. Si se conecta el *Smart Card Writer* con la tarjeta SIM insertada a un puerto USB, se puede ingresar el siguiente comando para verificar la conectividad:

```
~# pcsc_scan                          (Ctrl+c para abandonar la lectura)
```

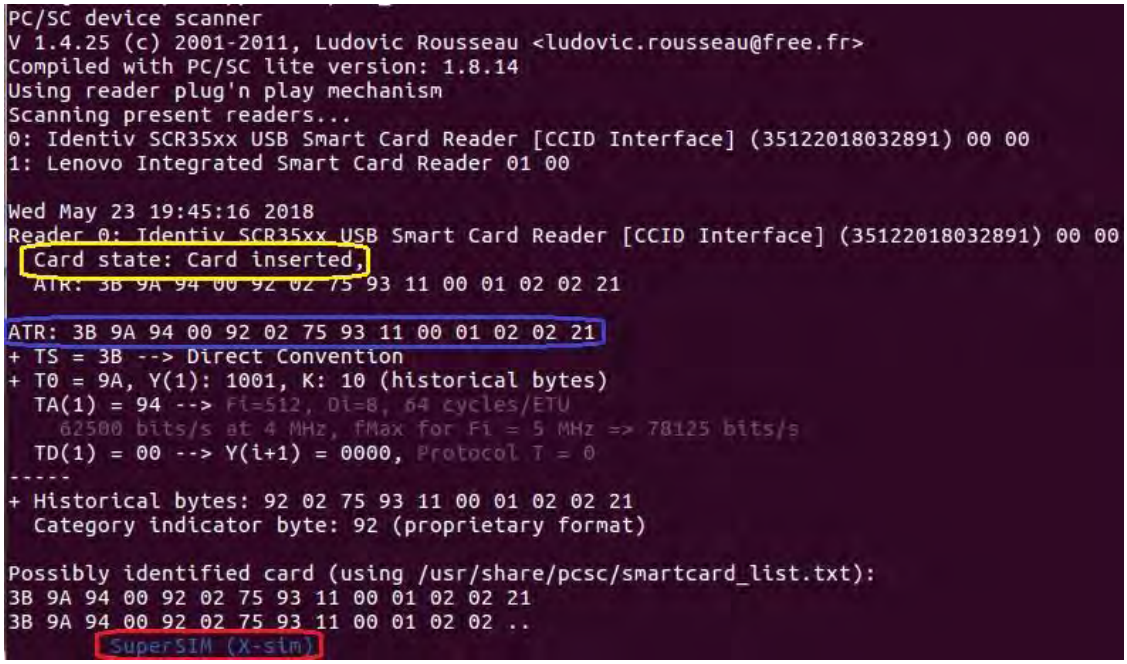


Figura 4.6 Conexión y lectura del Smart Card Writer.

En recuadro amarillo de la figura 4.6 se visualiza el ingreso de la tarjeta SIM en el *Smart Card Writer*, donde automáticamente se arrojan algunos parámetros que la identifican como el ATR (Answer To Reset) visto en el recuadro azul. El ATR entrega la información sobre todos los parámetros de los protocolos de comunicación de la

tarjeta, así como su estado de conexión con el dispositivo programador, todo esto basado en la ISO 7816. El recuadro rojo indica el modelo de tarjeta SIM usado, siendo este valor también identificado por medio del parámetro ATR.
El siguiente paso sería obtener el software pySim:

```
~# git clone https://git.osmocom.org/pysim
```

Antes de realizar la programación de la SIM se debe verificar si se realizó correctamente el proceso anterior. Para ello se ejecuta el *script* "pySim-read.py" con el objetivo de leer el contenido de esta. Se debe tener en cuenta que para la ejecución del *script* se debe habilitar el programador mediante el parámetro "-p0", como se muestra a continuación:

```
~# cd pysim
~# ./pySim-read.py -p0
```

La salida en pantalla debe mostrar los parámetros que pueden ser leídos de la SIM:

```
Reading ...
ICCID: 89860082190313860529
IMSI: 460003113237934
SMSP:
ffffffffffffffffffe9ffffffffffffffffffff0891683108705505f0000e0000ffa7
ACC: ffff
MSISDN: Not available
Done !
```

Aquí pueden comprobarse dos cosas. La primera, es que se verifica el correcto funcionamiento de *pySim* con el *Smart Card Writer*. La segunda, es observar que al leer la SIM, *pySim* arroja la lectura de los principales parámetros usados en el ámbito de la telefonía celular: ICCID (International Circuit Card ID), IMSI, SMSP (Short Message Service Parameters), ACC (Access Control Class) y el MSISDN (Mobile Station Integrated Services Digital Network).
Para realizar ya la programación se usa otro script llamado "pySim-prog.py" al cual se le deben definir los valores a programar. Para visualizar estas opciones podemos consultar la ayuda:

```
#./pySim-prog.py -help
```

Estas opciones son de mucha utilidad para definir los valores que se van a escribir en la tarjeta SIM (anexo 2). Empleando esta ayuda se procede a programar la tarjeta SIM con la siguiente línea de comando:

```
~#  ./pySim-prog.py  -p  0  -x  001  -y  01  -i  001010000000021  -s
8309219000000110000 -k 1108306A830921B09469C910226DCAFE
```

Los valores que no fueron definidos y son obligatorios, el *software* los genera aleatoriamente o usa los que tiene definidos por defecto. La salida en pantalla después de emitir esta línea de comando aparece a continuación:

```
Insert card now (or CTRL-C to cancel)
Autodetected card type fakemagicsim
Generated card parameters :
 > Name    : Magic
 > SMSP    : e1ffffffffffffffffffffff0581005155f5fffffffffff000000
 > ICCID   : 8309219000000110000
 > MCC/MNC : 1/1
 > IMSI    : 001010000000021
 > Ki      : 1108306A830921B09469C910226DCAFE
 > OPC     : 75952a71255c38ee3453ea3cfd77ea8c
 > ACC     : None
Programming ...
Done !
```

Si volvemos a ejecutar el script para leer los nuevos valores de la SIM, se puede verificar que los valores cambiaron:

```
~# ./pySim-read.py -p0
Reading ...
ICCID: 8309219000000110000
IMSI: 001010000000021
SMSP:
ffffffffffffffffffe1ffffffffffffffffffff0581005155f5fffffffffff00000
0
ACC: ffff
MSISDN: Not available
Done !
```

Al tener el celular preparado con la SIM programada, ya es posible registrar el móvil con los datos definidos anteriormente. Como se puede apreciar son los mismos valores que se usan en la sección 4.6.

Un dato importante a tener en cuenta para que la SIM que se programó pueda registrarse en la red UMTS, es que los valores MCC y MNC deben coincidir con los programados en la configuración “UMTS.Identity.MCC” y “UMTS.Identity.MCC”, tema abordado en la sección 4.5.

4.8 Propuesta de implementación de la radiobase UMTS usando mayor potencia.

El USRP N210 puede entregar una potencia de salida de hasta 100 mW. Esto limita un poco el alcance de la radiobase 3G. Para extender efectivamente la cobertura de la red se necesita un amplificador. Debido a que los teléfonos móviles usualmente tienen picos de potencia de transmisión de alrededor de 2 W, la estación base será el factor que limite el área de cobertura. La señal de UL puede alcanzar a la estación base, pero la señal de DL desde la estación base al teléfono es demasiado débil para que el UE la reciba. La asimetría de potencia relativa entre la estación base y el teléfono móvil se ilustra en la siguiente figura [31].

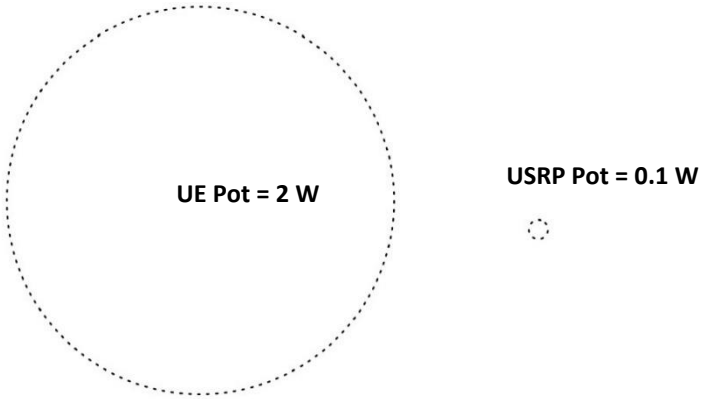


Figura 4.7 Asimetría de potencia entre el UE y la estación base [31].

4.8.1 Propuesta de Amplificador RF

Para ampliar la cobertura del servicio, es necesario aumentar la potencia de la radiobase con un amplificador que la eleve hasta un valor 2 W o mayor. Desafortunadamente los amplificadores RF se diseñan para operar en un ancho de banda específico, por lo que no se puede pretender cubrir todas las bandas.

No.	Banda (MHz)	F_{DL_Offset}	Banda UL (MHz)	Banda DL (MHz)	Canales absolutos RF de UTRAN	
					UL	DL
I	2100	0	1920 - 1980	2110 – 2170	9612 – 9888	10562 – 10838
II	1900	0	1850 – 1910	1930 – 1990	9262 – 9538	9662 – 9938
III	1800	1575	1710 – 1785	1805 – 1880	937 – 1288	1162 – 1513
IV	1700	1805	1710 – 1755	2110 – 2155	1312 – 1513	1537 – 1738
V	850	0	824 – 849	869 – 894	4132 – 4233	4357 – 4458
VIII	900	340	880 – 915	925 – 960	2712 – 2863	2937 – 3088
IX	1700	0	1749,9 – 1784,9	1844,9 – 1879,9	8762 – 8912	9237 – 9387
X	1700	1490	1710 – 1770	2110 – 2170	2887 – 3163	3112 – 3388

Tabla 4.1 Bandas de frecuencias y anchos de banda de canal UMTS-FDD [55].

Se propone entonces adquirir uno que trabaje en la banda de 900 MHz por los siguientes motivos:

- Buena propagación por ser de las bandas bajas de UMTS.
- No se detecta actividad en esta banda en zonas aledañas al Posgrado Ingeniería (Anexo 3).
- Existe un rango de frecuencias libres en esta banda que pudiera ser aprovechado en trabajos similares.

El espectro libre en la banda de 900 MHz abarca desde 902-928 MHz [56]. Las frecuencias del DL de la estación base UMTS comienzan a partir de 925 MHz, pero con los 5 MHz de ancho de banda se va más arriba de los 928 MHz, por lo que este rango libre no se puede usar para la implementación de telefonía celular. Pero teniendo en cuenta que no habrá interferencia en la zona de Posgrado Ingeniería, se pueden hacer pruebas con estas frecuencias en la zona.

Se propone un amplificador que cumpla los requisitos para la implementación de la red UMTS, y que se encuentre disponible en el mercado.

Figura 4.8 Amplificador RF en la banda de frecuencia de 900 MHz [57].

Especificaciones [57].

- Amplificador ajustable hasta + 85 dB de ganancia
- Procesamiento de señales inteligente y funcionamiento adaptativo en tiempo real
- Rango de frecuencias UL: 890 - 915 MHz
- Rango de frecuencias DL: 935 - 960 MHz
- Temperatura de trabajo: -10 ~ 50 °C
- Fuente de alimentación: AC 110 – 220 V
- Impedancia: 50 ohm
- Máxima potencia de salida: 4 dBW

$$P(dBW) = 10 * logP(W)$$

$$P(W) = 10^{P(dBW)/10}$$

$$P = 2.51\ W$$

Con este dispositivo se puede alcanzar hasta 2.5 W de potencia de salida según sus especificaciones, y también se puede manejar este valor a través de la consola de OpenBTS usando las facilidades que este ofrece para variar parámetros:

```
OpenBTS> power          (maneja la potencia de salida del USRP)
OpenBTS> txatten        (maneja la atenuación de la transmisión del USRP)
```

4.8.2 Propuesta de Duplexor de Cavidad

Cuando se agrega un amplificador de 2 W a la cadena de transmisión aumenta el área de cobertura, pero se debe tomar una precaución adicional con respecto a la cadena de recepción. La antena receptora recibiría mucha energía de la antena transmisora que puede dañar los circuitos del USRP. Como mínimo, haría imposible la demodulación de una señal limpia, porque los 2 W adicionales de la energía de transmisión local del amplificador actuarían como ruido y afectarían cualquier dispositivo remoto. Para resolver esto se necesita un duplexor de cavidad.

De forma general, el duplexor se debe conectar a los puertos de transmisión y recepción de la antena. Luego se conecta una sola antena al duplexor como se muestra en la figura 4.9. El sistema transmite y recibe en diferentes frecuencias, y

los duplexores están diseñados para dividir limpiamente las señales de transmisión y recepción.

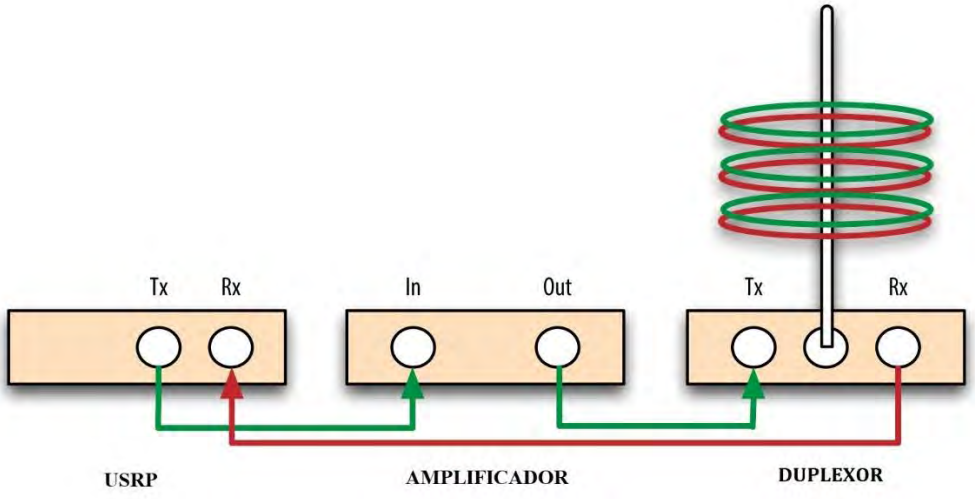


Figura 4.9 Esquema general de conexión dúplex [31].

Existen esquemas donde la antena que se utiliza cuenta con dos puertos dedicados, uno para la transmisión y otro para la recepción, pero esto sería un poco complicado a la hora de adquirirla en el mercado por estas características específicas. El duplexor que se propone entonces debe contener tres puertos, permitiendo que las dos cadenas de radiofrecuencia del sistema dual utilicen una sola salida a la antena. El esquema estructural está compuesto por dos filtros paso banda sintonizados a las bandas de transmisión y recepción, y un divisor a modo de elemento de unión entre la antena y los filtros. Por lo tanto, el puerto de salida se debe corresponder con el puerto de la antena, y los dos puertos de entrada con las cadenas receptora y transmisora [58].

Teniendo esto en cuenta el dispositivo que se propone es el duplexor EGSM-900, Modelo 14540.

Figura 4.10 Duplexor EGSM-900, Modelo 14540 [59].

Especificaciones [59].

- Rango de frecuencias UL: 880 - 915 MHz
- Rango de frecuencias DL: 925 - 960 MHz
- Aislamiento entre Tx y Rx: 40 dB mínimo
- Potencia máxima soportada: 20 W
- Temperatura de trabajo: 0 - 50 °C
- Humedad relativa soportada: por encima de 90%
- Impedancia: 50 ohm

4.8.3 Antena propuesta

La antena del sistema debe operar en la banda de frecuencia de 900 MHz. La propuesta es la Antena Omnidireccional 6dBi 900MHz, de polarización vertical, Modelo HGV906U [60].

Figura 4.11 Antena Omnidireccional 6dBi 900MHz, Modelo HGV906U [60].

Especificaciones.

- Rango de frecuencias de operación: 824 - 960 MHz
- Ganancia: 6 dBi
- Longitud: 23.6" (600mm)
- Diámetro: 1.3" (33mm)
- Polarización: Vertical
- Potencia máxima soportada: 100 W
- Temperatura de trabajo: -40 ~ 85 °C
- Patrón de radiación: Omnidireccional
- Impedancia: 50 ohm

4.8.4 Diseño del sistema completo.

La radiobase 3G creada a través de OpenBTS-UMTS utiliza duplexado por división en frecuencia para separar los canales de UL y DL. Esta técnica implica que los terminales han de ser capaces de seleccionar entre las dos bandas, y de esta manera establecer una comunicación bidireccional mediante una sola antena. La utilización de interruptores temporales no es viable dada la técnica de duplexado.
El sistema queda diseñado como se muestra en la siguiente figura.

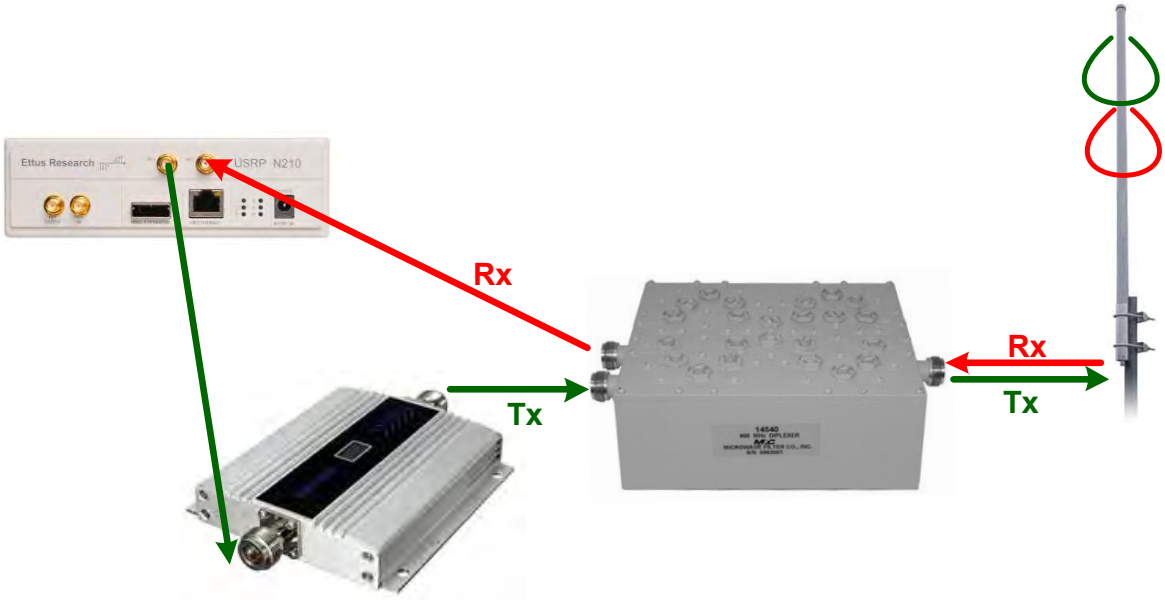


Figura 4.12 Esquema del sistema a implementar.

Este diseño presenta una estructura capaz de implementar una comunicación limpia separando las frecuencias usadas. De esta forma se puede conseguir la máxima transferencia posible en las bandas de interés, y la máxima atenuación en el resto de las bandas, cuidando además el hardware del USRP N210.

CAPÍTULO 5: RESULTADOS OBTENIDOS.

Después de todo el proceso de instalación explicado en el capítulo anterior se procede a poner en funcionamiento la estación base UMTS. Se comprobará cómo un terminal móvil accede a la red y se autentica. Además, se visualizará el espectro en cada banda de frecuencia a la que puede operar la radiobase creada, y se analizará la cobertura que ofrece el USRP N210 funcionando como *NodeB*. Posteriormente, se realiza una simulación de cobertura teniendo en cuenta el sistema propuesto en la sección 4.8.

5.1 Puesta en funcionamiento de la estación base.

Para hacer que la radiobase 3G comience a brindar servicio de telefonía celular se deben ejecutar dos aplicaciones:

- *OpenBTS-UMTS*: Es la que genera la interfaz aérea Uu a través de la cual los terminales móviles pueden acceder a la red.
- *SIPAuthServe*: Sistema de archivos que funciona como el servidor SIP de registro y autorización de usuarios. Se usa para procesar las peticiones de actualización de localización por parte de *OpenBTS*, y realiza los correspondientes cambios en la base de datos del registro de subscriptores.

Para esto se deben abrir dos terminales de Ubuntu por separado. En una terminal emitimos los siguientes comandos para ejecutar la aplicación *OpenBTS-UMTS*:

```
~# cd ~/OpenBTS-UMTS/apps
~# ./OpenBTS-UMTS
```

Si todo está funcionando bien la salida en pantalla debe ser como la siguiente figura:

Figura 5. 1 Salida en pantalla de la consola OpenBTS.

En la otra terminal de Ubuntu se ejecuta lo siguiente:

```
~# cd ~/subscriberRegistry/apps
~# ./sipauthserve
```

Esta aplicación utiliza el puerto 5062 para la comunicación con OpenBTS y el puerto 5064 para la autenticación SIP, elementos configurados en el capítulo anterior. Se encuentra dentro del componente *Subscriber Register*, y su función es autenticar el usuario en la red. Sin esta aplicación el usuario puede ver la red y tratar de acceder a ella, pero no podrá realizar el *attach*. La salida en pantalla debe ser como sigue:

Figura 5.2 Salida en pantalla de la consola SIPAuthServe.

Una vez hecho lo anterior se inicializa el sistema y la interfaz Uu queda desplegada en el aire. La señal será visible por los terminales móviles que estén dentro de su alcance, pero solo se podrán registrar los usuarios que fueron añadidos correctamente como se explicó en la sección 4.6.

5.2 Reconocimiento de la radiobase UMTS

Cuando ya se encuentra activo el sistema OpenBTS-UMTS, automáticamente se crea una interfaz virtual llamada “sgsntun” para facilitar la comunicación con los usuarios registrados. Esto se puede comprobar emitiendo el comando “`ifconfig`” en otra terminal de Ubuntu, y la interfaz virtual se debe visualizar en pantalla:

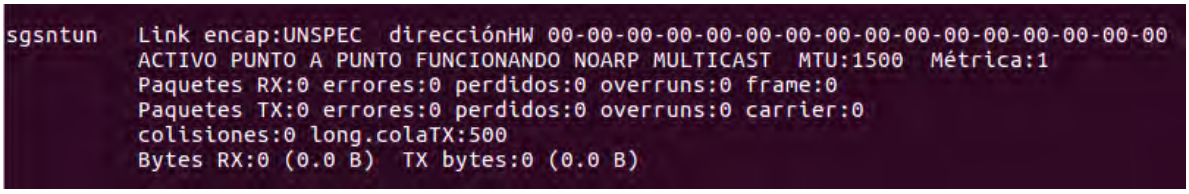


Figura 5.3 Visualización de la interfaz virtual “sgsntun”.

Luego de comprobar que todo se encuentra en orden, se puede escanear la red UMTS creada mediante el terminal móvil. Este proceso es diferente dependiendo

de la marca del celular que se tenga, pero generalmente se encuentra en las opciones de configuración.

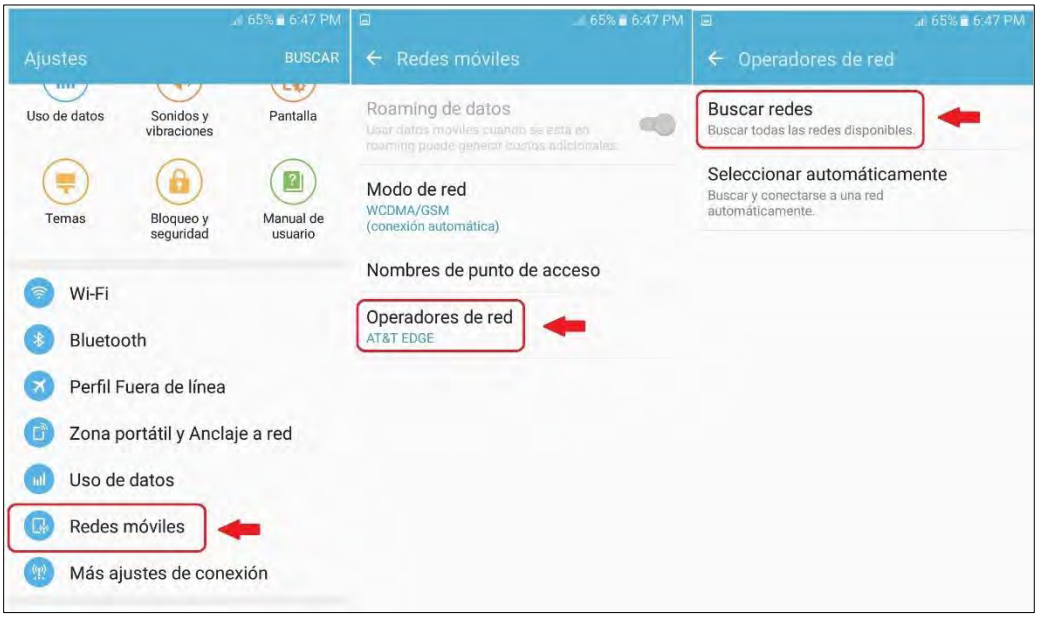


Figura 5.4 Escaneo de redes móviles en el celular.

Una vez hecho esto debe aparecer la red UMTS creada con cualquiera de los siguientes nombres:

- 00101
- 001-01
- Test PLMN 1-1

Cuando se selecciona la red 01001 en este caso, comienza el proceso de acceso y autenticación con la estación base hasta que se realiza el *attach*.

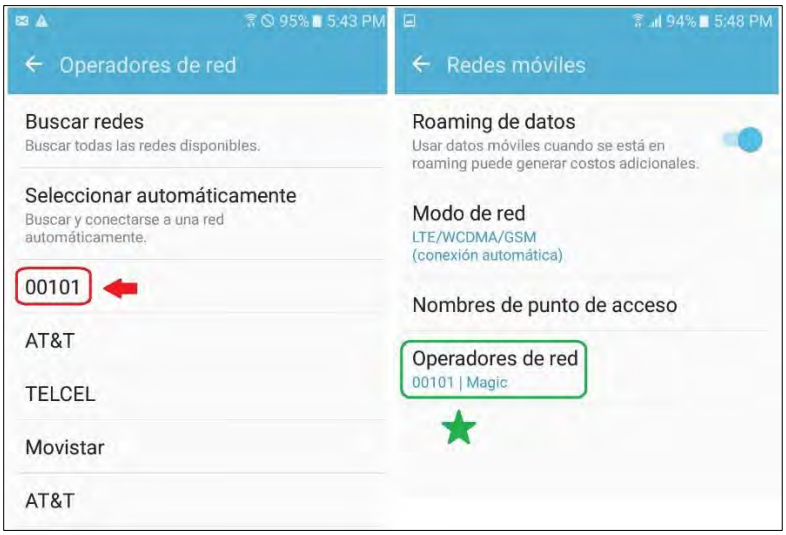


Figura 5.5 Selección de la red creada en el terminal móvil.

Este proceso (explicado en la sección 2.4.4.1, Figura 2.4) se puede observar mientras se encuentra en ejecución a través de la consola OpenBTS como se muestra en la siguiente figura:

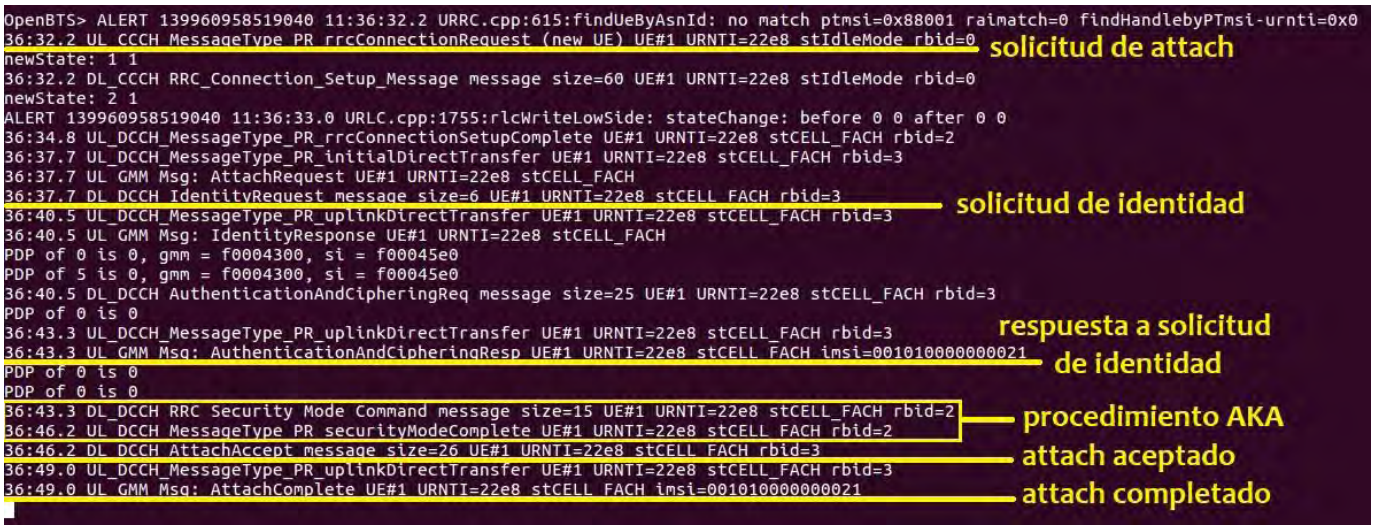


Figura 5.6 Proceso de registro en la red visto a través de la consola OpenBTS.

Al fichero *sipauthserve.cpp* se le hicieron modificaciones para que también mostrara en su consola detalles relacionados con este proceso.

```
ALERT 5467:5467 2018-06-04T00:23:46.1 sipauthserve.cpp:337:main: ./sipauthserve (re)starting
checando IMSI Vs Database
... ok ... IMSI conocido. Paso al procedimiento AKA
procesando solicitud de autenticacion
generando vectores de autenticacion
ordenando vectores de autenticacion
seleccionando vector de autenticacion a verificar
seleccionado RAND y AUTN... eviar al UE
comparando RES con XRES
seleccion y distribucion de CK IK
```

Figura 5.7 Proceso de autenticación visto a través de la consola SIPAuthServe.

Para conectarse con los datos 3G en la red, hay que tener activo el *roaming* en el celular y agregar un punto de acceso, preferiblemente, que tenga la dirección IP del servidor de DNS principal de la computadora donde está instalado OpenBTS.

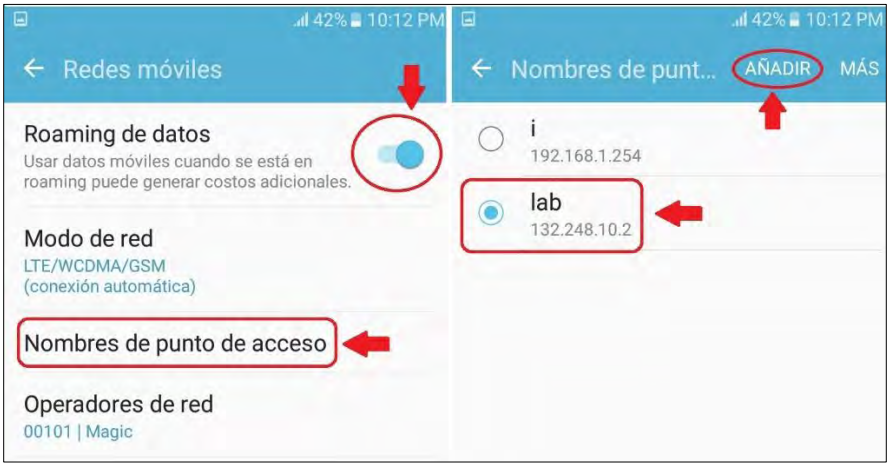


Figura 5.8 Activación de roaming y punto de acceso IP.

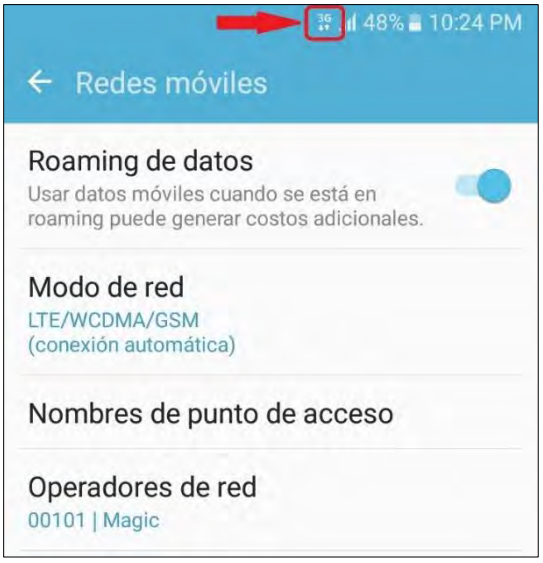


Figura 5.9 Acceso por datos a la red UMTS.

5.3 Mediciones del Espectro

Para seleccionar la frecuencia en la que la radiobase va a trabajar no se puede hacer desde la consola, pues de esta forma arroja errores que alteran el funcionamiento de la radiobase, y para resolverlo se debe reiniciar la base de datos. Para poder hacer esto correctamente se debe parar los procesos de *OpenBTS-UMTS* y *SIPAuthServe*, y cargar la base de datos externamente. Para detener estos procesos en cada una de las consolas se presiona en el teclado `Ctrl+C`.

Los valores a configurar se deben modificar en el archivo que se configuró en la sección 4.5 “OpenBTS-UMTS.example.sql”.

‘**UMTS.Radio.Band**’ – se selecciona la banda en la que se va a operar (850, 900, 1700, 1800, 1900 o 2100) *ej.* 900.
‘**UMTS.Radio.C0**’ – Se configura un valor de UARFCN válido dentro de la banda elegida, *ej.* para la banda de 900 se selecciona 3050.

Una vez hecho esto, se borra el archivo de base de datos y se vuelve a cargar con los nuevos valores configurados, utilizando la herramienta *sqlite3*:

```
~# cd /etc/OpenBTS
~# rm -R OpenBTS-UMTS.db
~# sqlite3 OpenBTS-UMTS.db ".read OpenBTS-UMTS.example.sql"
```

En UMTS, para realizar el cálculo de la frecuencia en el DL y del valor UARFCN (canal absoluto RF de UTRAN, siglas en inglés) se deben utilizar las siguientes fórmulas [25].

$$N_{DL} = 5(F_{DL} - F_{DL\_Offset})$$
$$F_{DL} = F_{DL\_Offset} + (N_{DL})/5$$

Donde:

N_{DL} *: Valor de UARFCN*
F_{DL_Offset} *: Valor predefinido para la banda en cuestión. Tabla 4.1, sección 4.8.1*
F_{DL} *: Frecuencia central en el DL (MHz)*

Los valores con los cuales se inicializó el sistema fueron:

```
INSERT OR IGNORE INTO "CONFIG" VALUES('UMTS.Radio.Band','900',1,0,…)
INSERT OR IGNORE INTO "CONFIG" VALUES('UMTS.Radio.C0','3050',1,0,…)
```

Realizando el cálculo se obtiene:

$\mathbf{N_{DL} = 3050}$

F_{DL_Offset} = 340
F_{DL}: = 950 MHz

Con un analizador de espectro podemos visualizar el resultado:

Figura 5.10 Espectro de la radiobase 3G en la banda de 900 MHz.

Se procede a continuación a realizar esta misma medición del espectro en la banda de 850 MHz. Realizando el cálculo se obtiene:

N_{DL} = 4425
F_{DL_Offset} = 0
F_{DL}: = 885 MHz

Se modifica el fichero con el editor de texto de preferencia, en este caso se utilizó el *geany*.

```
~# cd /etc/OpenBTS
~# geany OpenBTS-UMTS.example.sql
```

Se cambian los valores en el archivo:

```
INSERT OR IGNORE INTO "CONFIG" VALUES('UMTS.Radio.Band','850',1,0,…)
INSERT OR IGNORE INTO "CONFIG" VALUES('UMTS.Radio.C0','4425',1,0,…)
```

Después de guardar los cambios y cerrar el archivo, se elimina la base de datos y se carga otra vez:

```
~# rm -R OpenBTS-UMTS.db
~# sqlite3 OpenBTS-UMTS.db ".read OpenBTS-UMTS.example.sql"
```

El siguiente paso es reiniciar las aplicaciones *OpenBTS-UMTS* y *SIPAuthServe* para que los cambios surtan efecto. Para esto, se detienen los procesos individualmente en cada terminal donde se ejecutan con `Ctrl+c`, y se vuelven a ejecutar.

`Terminal de OpenBTS`:

```
> Ctrl+c
~# ./OpenBTS-UMTS
```

`Terminal de SIPAuthServe`:

```
> Ctrl+c
~# ./sipauthserve
```

Hecho esto se puede visualizar el resultado con un analizador de espectro:

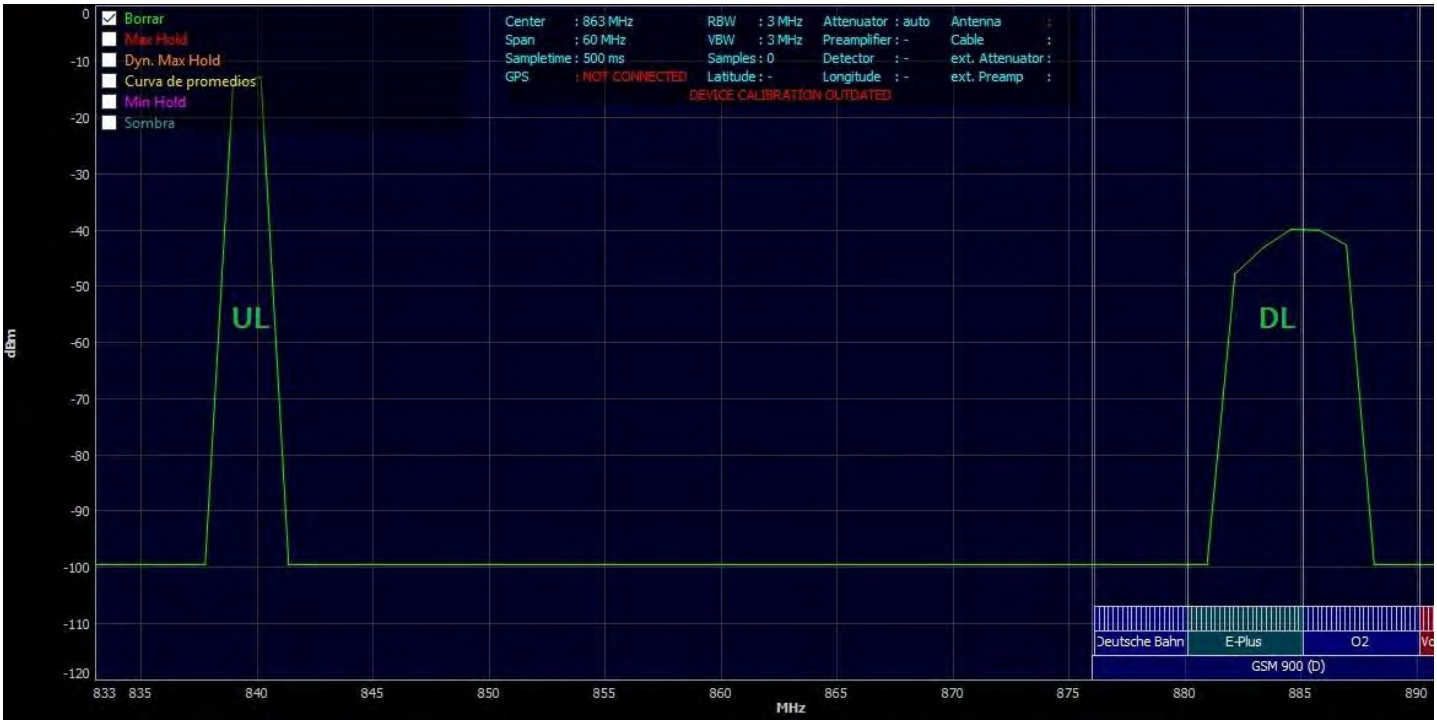


Figura 5.11 Espectro de la radiobase 3G en la banda de 850 MHz.

Este mismo procedimiento se repite cada vez que se vaya a operar en otra frecuencia, y así se obtiene también el espectro para la banda de 1900 MHz.

N_{DL} = 9905
F_{DL_Offset} = 0
F_{DL}: = 1981 MHz

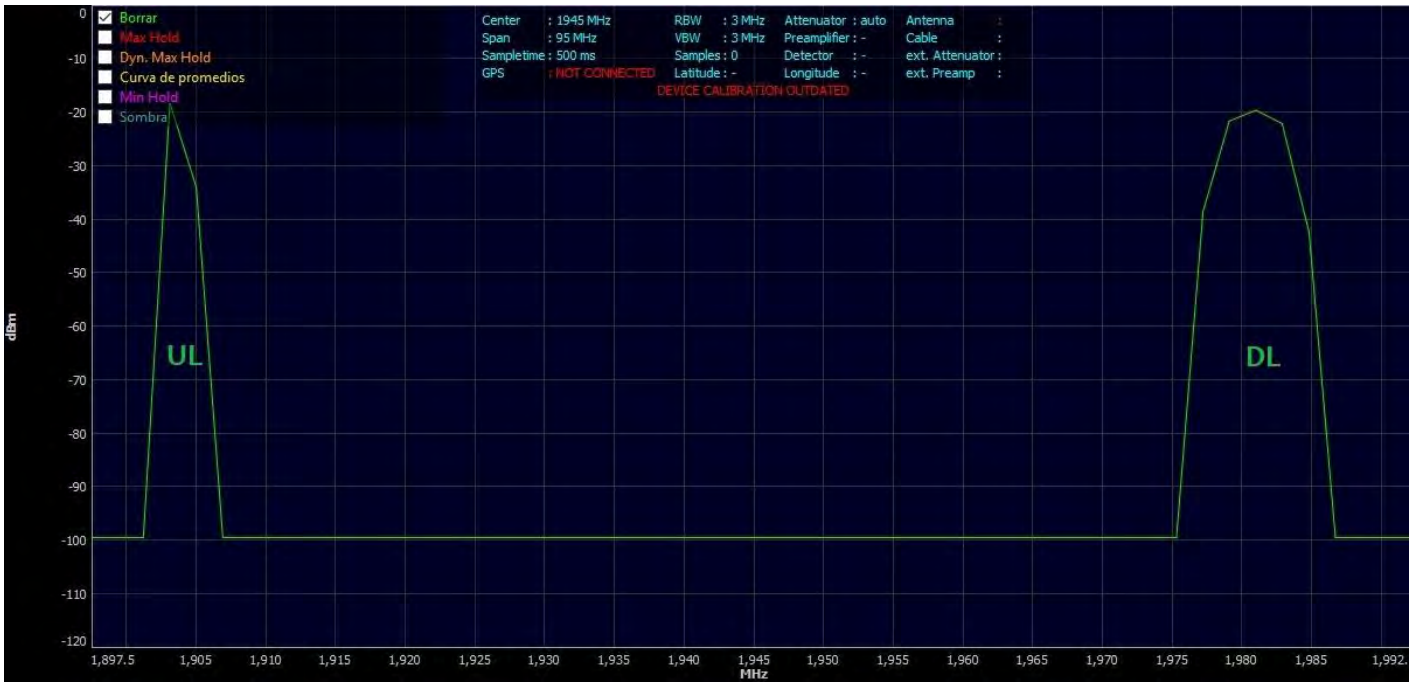


Figura 5.12 Espectro de la radiobase 3G en la banda de 1900 MHz.

Se decide no medir en las bandas de 1700 y 1800 MHz porque el celular que se utiliza no está habilitado para operar en estas frecuencias con tecnología 3G (anexo 4). Tampoco se realizaron mediciones de espectro en la banda de 2100 MHz debido a que en esta frecuencia hay actividad en la zona de Posgrado Ingeniería (anexo 3).

5.4 Mediciones de intensidad de la señal, RSSI

El indicador de intensidad de señal recibida RSSI, es una escala de referencia en relación a 1 mW, para medir el nivel de potencia de las señales recibidas por un dispositivo inalámbrico (típicamente WiFi o telefonía móvil) [61]. La escala tiene al valor cero como centro (0); representa 0 RSSI, o 0 dBm. Aunque teóricamente puede darse el caso de medirse valores positivos, generalmente la escala se expresa dentro de valores negativos que abarcan de 0 a -120 dBm; cuanto más negativo, mayor pérdida de señal.

Para los cálculos teóricos de cobertura hay que tener en cuenta también la sensibilidad del receptor, también llamado umbral del receptor [62]. Este parámetro se define como el nivel mínimo de la señal de RF que se puede detectar a la entrada

del receptor y producir una señal útil de información demodulada [63]. Para la interfaz aérea de UMTS se define como promedio un valor de -99 dBm, y si tenemos en cuenta la atenuación del cuerpo humano (3 - 4 dB), se puede definir como valor a tener en cuenta -95 dBm [64].

Como ya se explicó en la sección 4.8, el alcance de la estación base será el factor que limite la distancia a la que el terminal móvil pueda tener cobertura. Por este motivo, aunque la comunicación sea bidireccional, basta que la cobertura se analice solamente en el enlace descendente, frecuencias de DL.

Existen diversas formas de expresar las pérdidas de intensidad de señal como modelos empíricos y determinísticos. Para este trabajo se decide implementar el modelo empírico OKUMURA-HATA, que es uno de los más usados en telefonía celular. Las fórmulas usadas corresponden a la especificación de ciudades medianas [65].

$$L = 69.55 + 26.16Log(f) - 13.82Log(hte) - a + \{44.9 - 6.55Log(hte)\}Log(d)$$

$$a = \{1.1Log(f) - 0.7\}hre - \{1.56Log(f) - 0.8\}$$

$$RSSI = Ptx + Gtx - L$$

Donde:

d : Distancia (km)

f : Frecuencia (MHz)

L : Pérdidas de propagación (dB)

a: Factor de corrección para la altura efectiva de la antena móvil (dB).

$hre = 1\ m$: Altura de la antena receptora

$hte = 12\ m$: Altura de la antena transmisora

$hre = 1m$: Altura de la antena receptora

$Gtx = 3\ dBi$: Ganancia de antena del USRP (no se considera la ganancia del celular)

$Ptx = 100\ mW = 20\ dBm$: Potencia del USRP

$RSSI$: Nivel de intensidad de la señal (dBm)

Las pérdidas en cables y conectores no se tienen en cuenta por ser prácticamente despreciables. Igualmente, se asume que la antena del equipo terminal no tiene ganancia. Realizando los cálculos se obtienen los valores teóricos de RSSI para cada banda de frecuencia en la que se van a realizar las mediciones reales.

Ptx (dBm)	Gtx (dBi)	f (MHz)	hte (m)	a (dB)	d (m)	L (dB)	RSSI (dBm) teórico
20	3	950	12	-0.89	30	75.81	-52.81
					40	80.54	-57.54
					50	84.20	-61.20
					60	87.20	-64.20
					70	89.73	-66.73
					80	91.93	-68.93
					90	93.86	-70.86
					100	95.59	-72.59
					110	97.16	-74.16
					120	98.59	-75.59
					130	99.90	-76.90
					140	101.12	-78.12
					150	102.26	-79.26
					160	103.32	-80.32
					170	104.31	-81.31
					180	105.25	-82.25

Tabla 5.1 Calculo de RSSI teórico para la banda de 900 MHz.

Ptx (dBm)	Gtx (dBi)	f (MHz)	hte (m)	a (dB)	d (m)	L (dB)	RSSI (dBm) teórico
20	3	881	12	-0.88	30	74.94	-51.94
					40	79.67	-56.67
					50	83.33	-60.33
					60	86.33	-63.33
					70	88.86	-65.86
					80	91.06	-68.06
					90	92.99	-69.99
					100	94.72	-71.72
					110	96.29	-73.29
					120	97.72	-74.72
					130	99.03	-76.03
					140	100.25	-77.25
					150	101.38	-78.38
					160	102.44	-79.44
					170	103.44	-80.44
					180	104.38	-81.38

Tabla 5.2 Calculo de RSSI teórico para la banda de 850 MHz.

Ptx (dBm)	Gtx (dBi)	f (MHz)	hte (m)	a (dB)	d (m)	L (dB)	RSSI (dBm) teórico
20	3	1981	12	-1.04	30	84.31	-61.31
					40	89.03	-66.03
					50	92.70	-69.70
					60	95.70	-72.70
					70	98.23	-75.23
					80	100.42	-77.42
					90	102.36	-79.36
					100	104.09	-81.09
					110	105.66	-82.66
					120	107.08	-84.08
					130	108.40	-85.40
					140	109.62	-86.62
					150	110.75	-87.75
					160	111.81	-88.81
					170	112.81	-89.81
					180	113.75	-90.75

Tabla 5.3 Calculo de RSSI teórico para la banda de 1900 MHz.

El terminal móvil con el cual se realizaron las mediciones es un Samsung Galaxy J7, que permite ver los parámetros de intensidad de la señal recibida, por lo que no fue necesario instalar ninguna aplicación adicional.

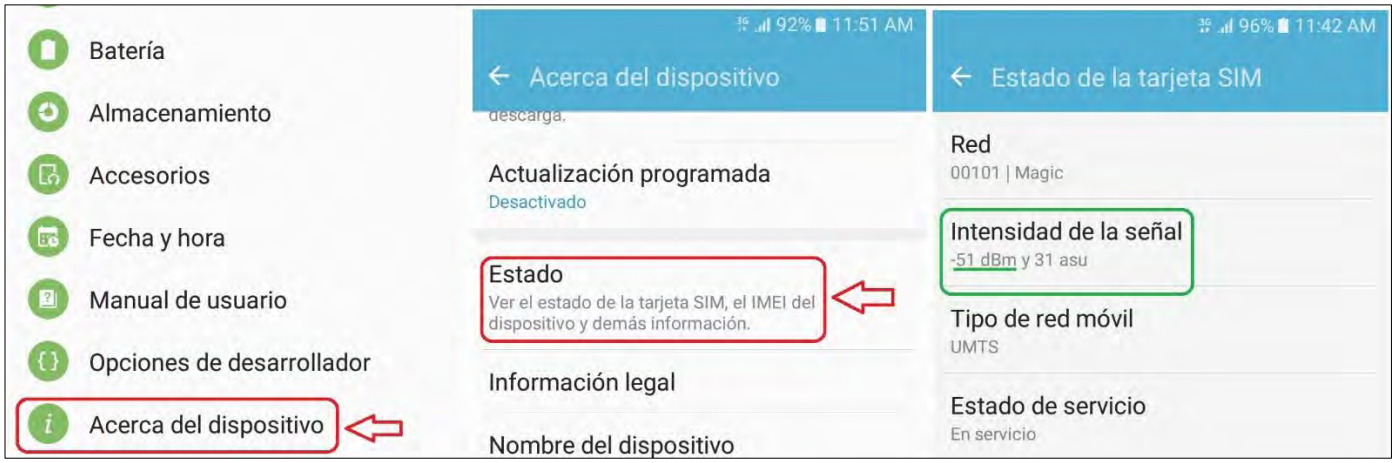


Figura 5.13 Medición de intensidad de la señal con el terminal móvil.

Se procede entonces a realizar las mediciones reales de la radiobase 3G, y se realiza una comparación con los valores teóricos calculados.

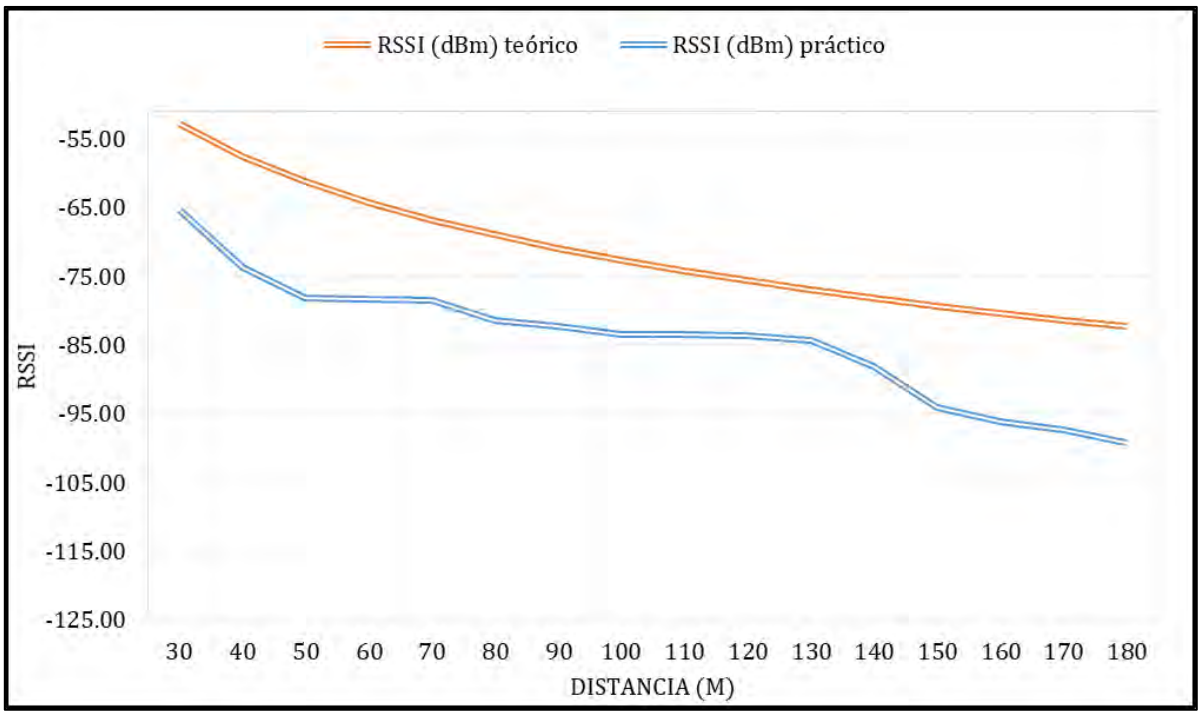


Figura 5.14 Comparación de RSSI teórico Vs RSSI práctico, banda 900 MHz.

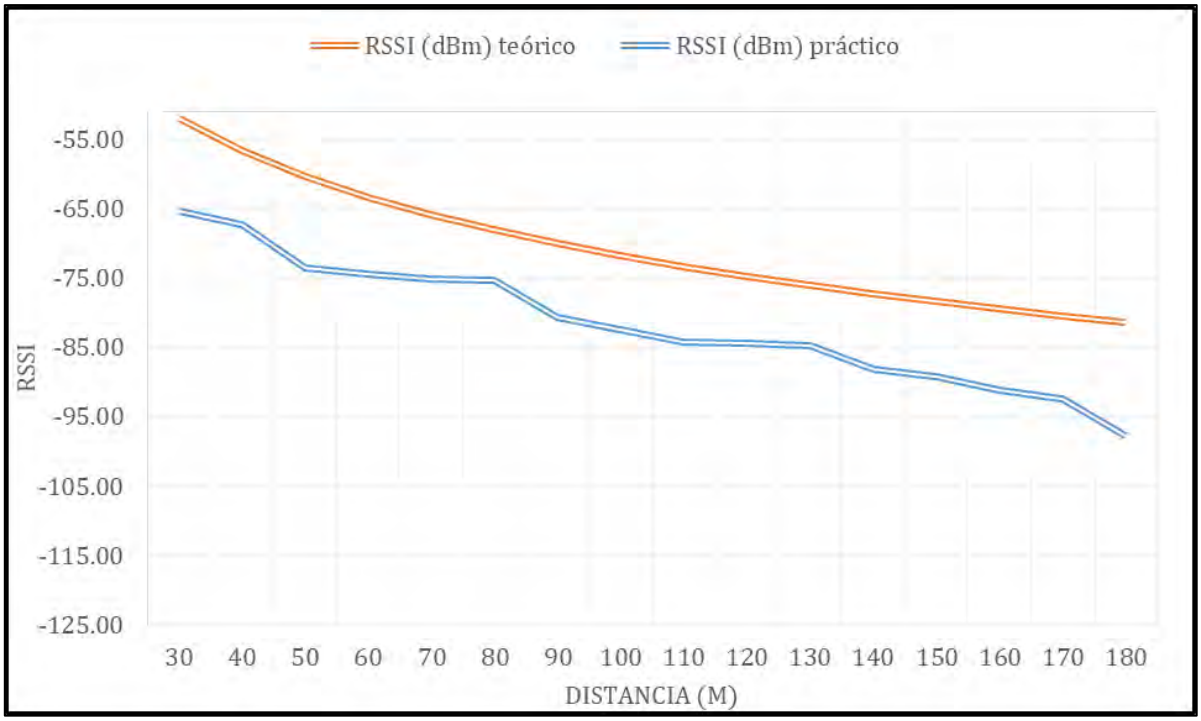


Figura 5.15 Comparación de RSSI teórico Vs RSSI práctico, banda 850 MHz.

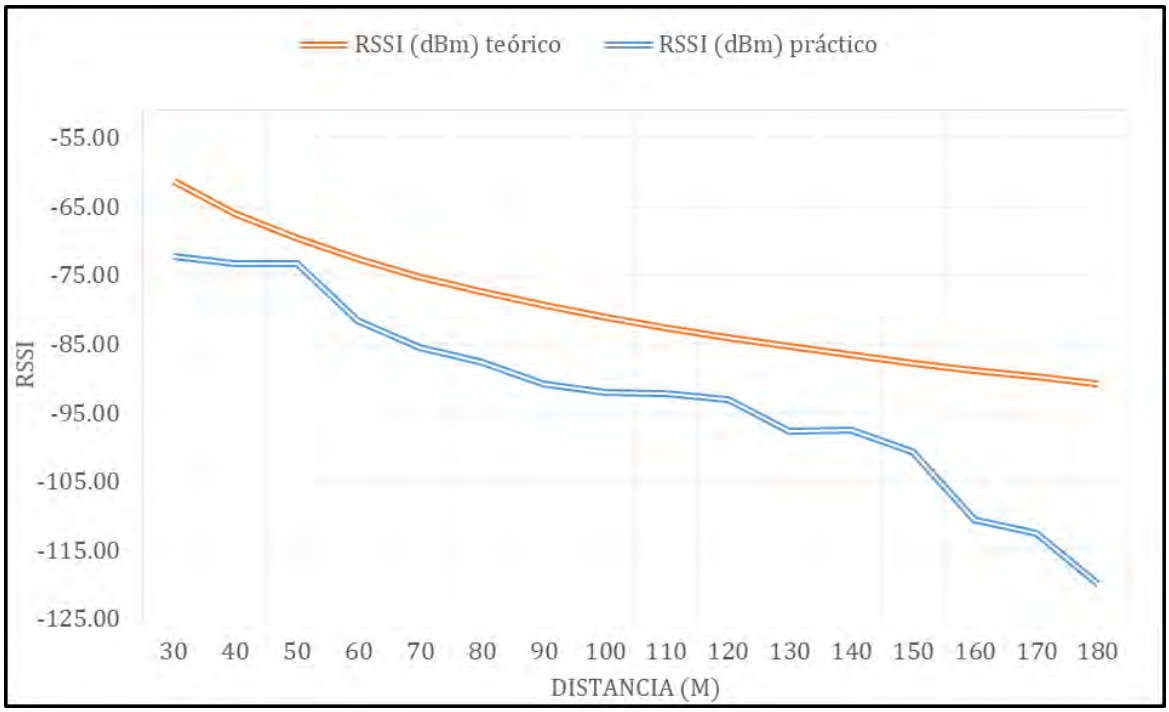


Figura 5.16 Comparación de RSSI teórico Vs RSSI práctico, banda 1900 MHz.

En las mediciones reales se pudo observar en las bandas bajas (850 y 900 MHz), que aproximadamente a una distancia 100 metros, el terminal móvil perdía la conexión de datos con la radiobase 3G. En la banda de 1900 MHz esto sucedía alrededor de los 60 metros de distancia. Para todos los casos coincidió, que la intensidad de señal oscilaba entre los valores de -83 ~ -85 dBm, cuando ocurría la pérdida de conexión por datos.

Esto resulta un poco contradictorio, pues si tenemos en cuenta que la sensibilidad del receptor tiene un valor de -95 dBm, la comunicación no debería experimentar fallos con esos niveles de recepción. Pero para que la comunicación sea efectiva no es suficiente un buen valor del umbral de recepción. Para tener una buena calidad de la señal, y determinar si es útil o no, es necesario contar con una buena relación señal a ruido [66].

5.5 Relación señal a ruido de la señal recibida, SNR.

La relación señal a ruido SNR (Signal-to-Noise Ratio), impacta en el rendimiento de una comunicación con calidad. Un valor de SNR alto representa que la intensidad de la señal es mayor que los niveles de ruido a la entrada del receptor, lo que

permite mantener un mejor rendimiento [67]. Si por el contrario la SNR es menor, disminuye el rendimiento y la calidad de la comunicación. Por lo general se expresa en dB y abarca un rango de valores que va desde 0 hasta 120 [61].

El ruido N (Noise), lo determinan varios elementos que degradan la señal en la propagación hasta su destino, y es imposible eliminarlo. Se define como la mezcla de señales no deseadas que se combinan con la señal útil que se quiere recibir, y que por tanto la distorsionan. Para realizar los cálculos de la SNR recurrimos a las siguientes fórmulas [68]:

$$N = 10Log(TBK) + N_f$$
$$SNR = RSSI - N - 30$$

En esta última fórmula, el valor de RSSI viene dado en unidades dBm, es por esto que para obtener la SNR en dB debemos restarle 30 al final de la ecuación.

Donde:

$T = 298\ K$:	Temperatura (grados Kelvin)
$B = 5000000\ Hz$:	Ancho de banda (5 MHz)
$K = 1.38 * 10^{-23}\ {}^{J}/_{K}$:	Constante de Boltzman
$N_f = 5\ dB$:	Figura de ruido del USRP
N :	Potencia de ruido (dB)
SNR :	Relación señal a ruido (dB)

Para la telefonía celular se considera una buena relación señal a ruido cuando el valor está por encima de los 20 dB [69]. Utilizando los valores teóricos de RSSI obtenidos en la sección 5.4 y el valor calculado anteriormente de N, se calculan los valores de SNR a los que se le denominan en la gráfica “SNR Teórico”. De la misma forma, el “SNR práctico”, es obtenido con este mismo valor de potencia de ruido calculado, pero con los valores reales de RSSI medidos en el terreno.

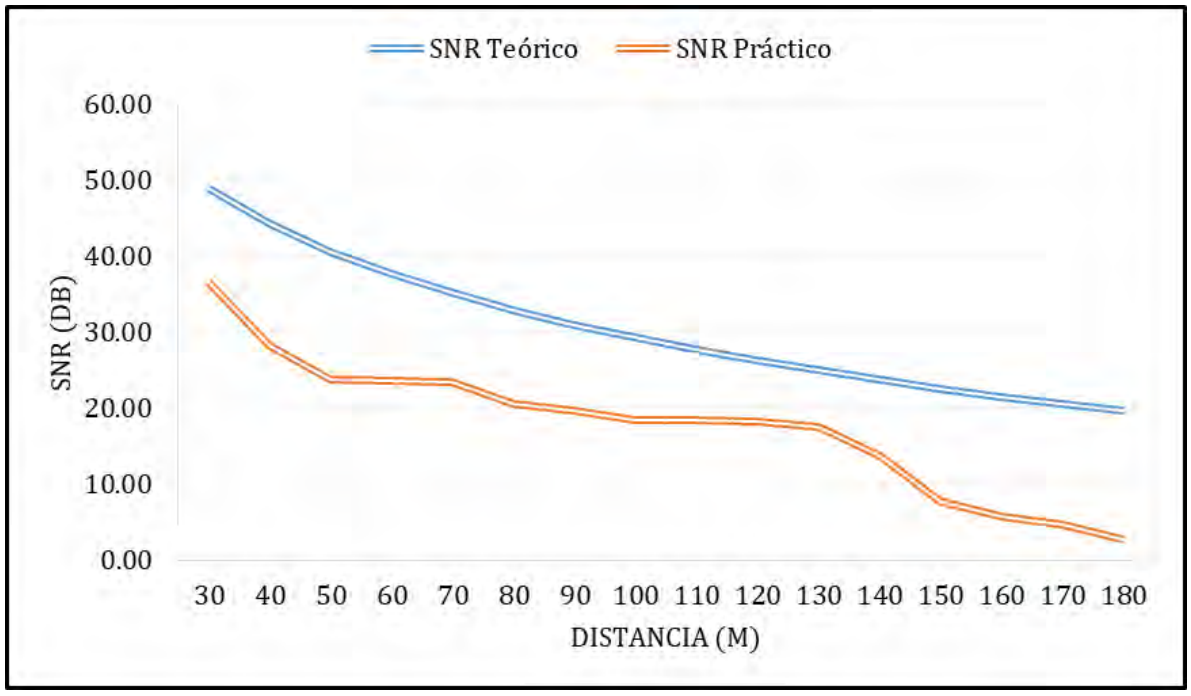


Figura 5.17 Gráfica SNR teórico Vs SNR práctico, banda 900 MHz.

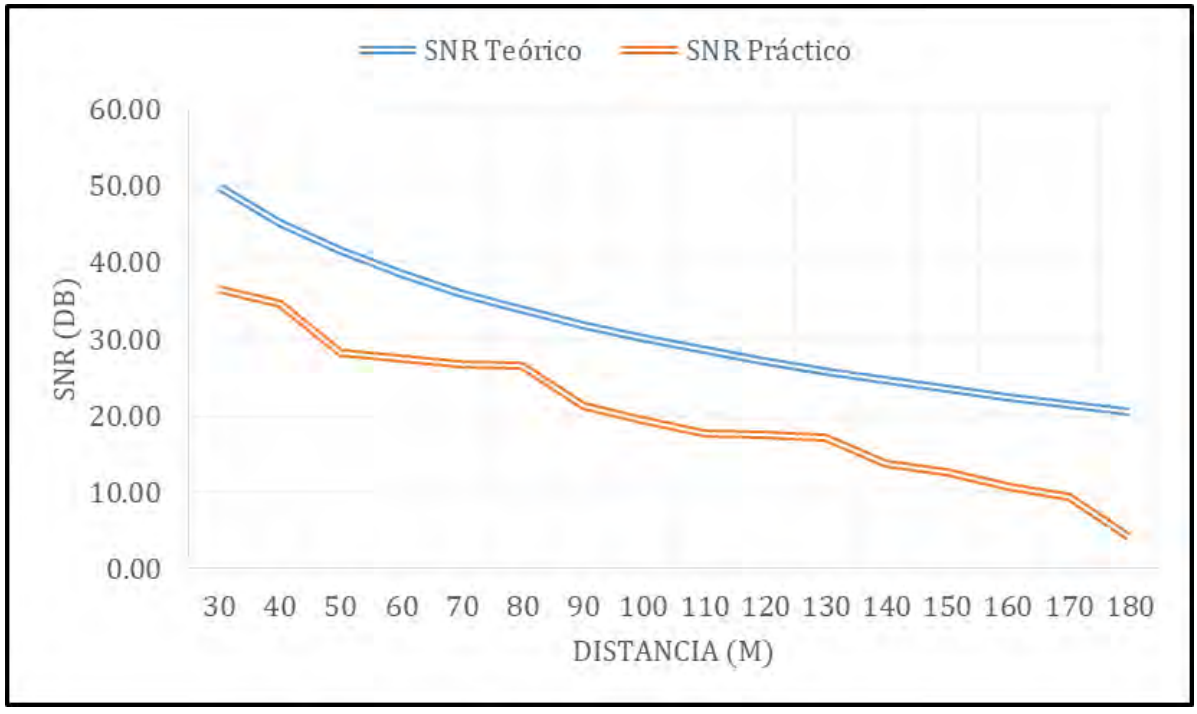


Figura 5.18 Gráfica SNR teórico Vs SNR práctico, banda 850 MHz.

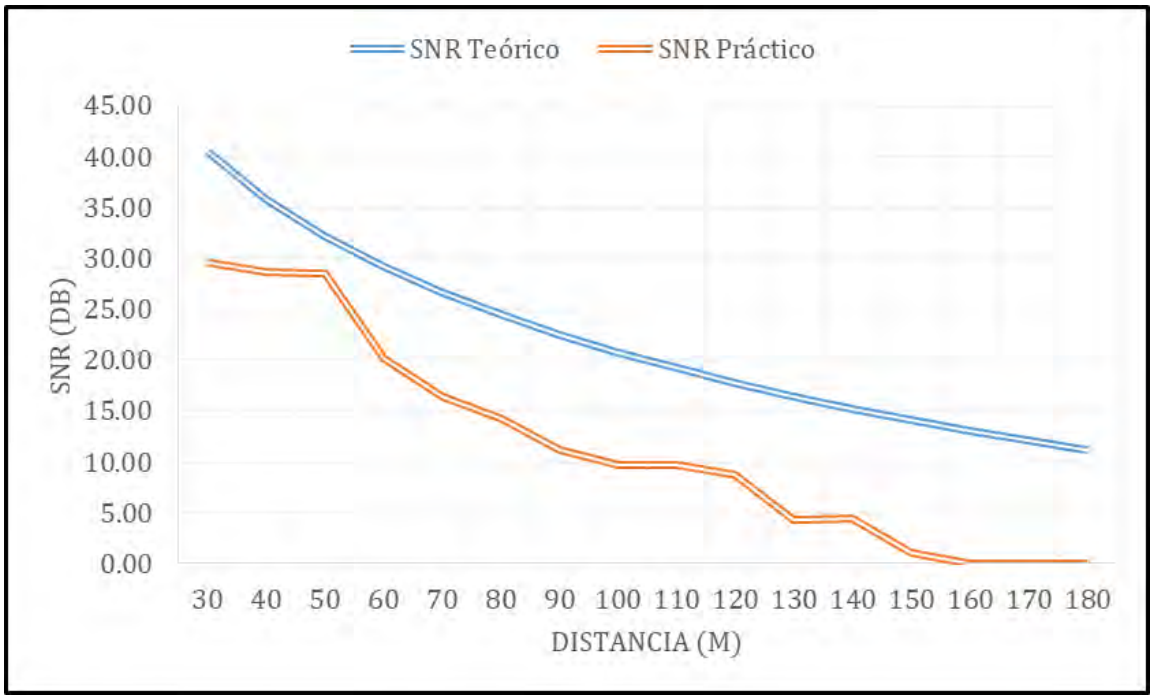


Figura 5.19 Gráfica SNR teórico Vs SNR práctico, banda 1900 MHz.

En las gráficas se puede observar que las mediciones reales están por debajo de los cálculos teóricos. Esto se debe a las irregularidades del terreno que no hay forma de contemplarlas mediante el cálculo matemático que se realiza. Para realizar una predicción más cercana a la realidad, lo más común es utilizar un software al que se le puedan cargar mapas con las irregularidades del terreno, y teniendo esto en cuenta, se realice entonces una predicción de cobertura mediante simulaciones.

5.6 Simulación de cobertura implementando mayor potencia a la estación base

Si se lleva a cabo la propuesta realizada en la sección 4.8, se puede ampliar la cobertura de la red UMTS creada. Se procede entonces a realizar una simulación de cobertura con el programa *Radio Mobile,* para analizar el área aproximada que se puede cubrir con esta radiobase 3G.

Radio Mobile constituye una herramienta de uso libre que permite simular enlaces de radio. Auxiliándonos de este *software* podemos realizar determinados análisis de un sistema de radiocomunicaciones, su comportamiento, la distancia entre

transmisor y receptor, las ganancias de las antenas, polarización de onda, la frecuencia, la potencia, *etc* [70].

La estación base estaría ubicada sobre el edificio Valdés Vallejo en Posgrado Ingeniería, UNAM. La banda de frecuencia que vamos a ocupar es de la 900 MHz según la propuesta.

Ubicación de la radiobase 3G 19.328103°N - 99.182129°W

Figura 5.20 Ubicación de la radiobase 3G.

Datos:

Límites de frecuencia: 880 – 960 MHz (Tabla 4.1, sección 4.8)

Modo de variabilidad: Móvil 90% del tiempo 50% de situaciones [67]

Refractividad de la superficie: 301 N-Units

Conductividad del suelo: 0.005 S/m

Tipos de antenas: Polarización Vertical, Patrón Omnidireccional

Ganancia de antena de radiobase: 6 dBi

Ganancia de antena de terminal móvil: 0 dBi

Altura de la antena transmisora: 15 m

Altura de la antena receptora: 1 m

Pérdidas en cableado: 0.3 dB

Potencia máxima del Transmisor: 2.5 W

Potencia máxima de transmisión en el terminal móvil: 2 W

En los resultados obtenidos en las mediciones reales se puede ver que aunque los valores de RSSI estuvieran por encima del umbral de recepción cuando la SNR no superaba los 20 dB, la conexión de datos fallaba. Para hacer la simulación de cobertura se debe tener en cuenta entonces un valor de RSSI que satisfaga una SNR al menos de 20 dB.

Como el valor de la potencia de ruido es constante (sección 5.5):

$$N = 10Log(TBK) + N_f = -131.87\ dB$$

La relación señal a ruido solamente va a depender de la variación de RSSI. Si se fija a 20 dB la SNR, se puede calcular entonces el valor mínimo de RSSI con el cual la comunicación va a ser efectiva:

$$SNR = RSSI - N - 30$$
$$SNR > 20\ dB$$
$$RSSI - N - 30 > 20\ dB$$
$$RSSI > 20\ dB + N + 30$$
$$RSSI > 20\ dB - 131.87\ dB + 30$$
$$\boldsymbol{RSSI > -81.87\ dBm}$$

Por este motivo la sensibilidad del receptor que se use en la simulación debe tener este valor.

Sensibilidad del receptor: **- 81.87 dBm.**

Teniendo todos estos datos definidos se procede a simular la cobertura de la red.

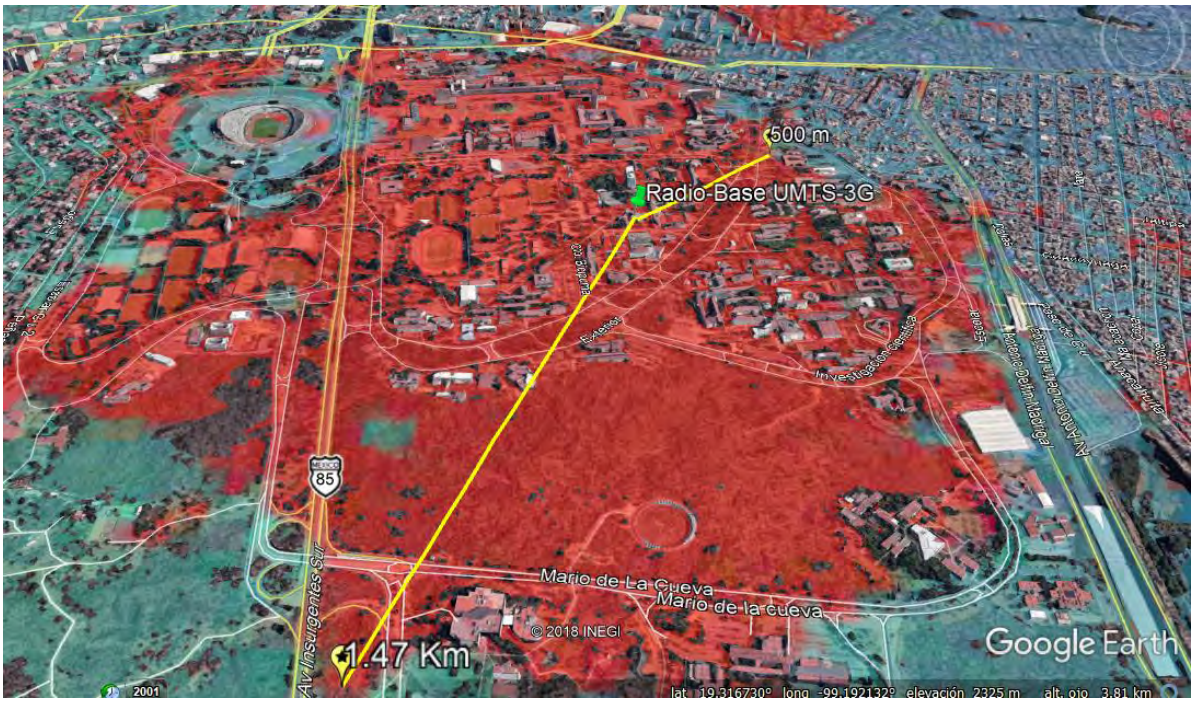


Figura 5.21 Cobertura de la radiobase 3G.

El programa *Radio Mobile* utiliza el modelo matemático de propagación *Longley Rice*. Este es diferente al modelo empírico que se utilizó para realizar los cálculos comparativos con las mediciones reales, pero el objetivo de la simulación es visualizar una cobertura aproximada del sistema propuesto, y no la comparación con mediciones reales. La figura 5.21 muestra una cobertura irregular pues el *software* toma en cuenta las irregularidades del terreno, donde su mayor alcance está a una distancia de 1.47 km, y el trayecto menor es de 500 m.

Es importante destacar que los modelos de propagación utilizados para el cálculo de cobertura en este trabajo no abarcan los espacios interiores de las edificaciones. Para esto habría que realizar un trabajo más profundo, utilizando especificaciones particulares del modelo de propagación que se utilice.

CAPÍTULO 6: CONCLUSIONES

La contribución del proyecto es implementar en *software* una estación base UMTS a un precio relativamente bajo en comparación con los sistemas que se usan en la actualidad, utilizando el USRP N210 como equipo de Radios Definidos por Software. En él se describieron los requerimientos de *hardware* y *software* necesarios para su implementación, así como la configuración para poner la red en operación.

Este trabajo amplía una línea de investigación en el desarrollo de las telecomunicaciones utilizando tecnología SDR, y proporciona un panorama del potencial de esta tecnología y del *hardware* utilizado.

La versión pública de OpenBTS-UMTS 1.0 fue la empleada en este desarrollo por ser totalmente gratuita. Este código abierto está totalmente disponible por lo que se puede aprovechar, y obtener beneficio del soporte que brinda la comunidad de OpenBTS, un foro donde se puede consultar e intercambiar con programadores que también lo utilizan.

Con esta implementación se pudo crear una interfaz aérea WCDMA que permite establecer comunicación con un terminal móvil. Gracias al amplio ancho de banda de operación de la tarjeta auxiliar SBX, fue posible trabajar en varias frecuencias de UMTS. Se pudieron utilizar distintos canales UARFCN en diferentes bandas, logrando exitosamente la autenticación y registro en la red creada, y estableciendo la comunicación mediante los datos con la radiobase 3G.

Se realizaron diversas mediciones reales para evaluar la cobertura del USRP N210 funcionando como *NodeB*, comprobando la distancia límite a la que el terminal móvil cuenta con servicio de datos. La estación base transmite a una potencia de 20 dBm, y con el uso de dos antenas VERT900 (orientadas 90º entre sí), se alcanzaron distancias aproximadas de 100 metros en las bandas bajas (850 y 900 MHz), y de 60 metros en la banda de 1900 MHz.

Para lograr mayor alcance es necesario incorporar al sistema un amplificador y duplexor, motivo por el cual se realiza una propuesta a implementar en la banda de 900 MHz. Al realizar la simulación, teniendo en cuenta los valores de las mediciones reales, esta nos brinda resultados de cobertura bastante satisfactorios.

Después de comprobar los resultados obtenidos, se llega a la conclusión de que la implementación de una radiobase UMTS 3G utilizando Radios Definidos por Software es una solución fiable y segura para desarrollar proyectos ubicados en zonas donde no hay una infraestructura preestablecida para las comunicaciones. La propuesta de ampliar la potencia de la estación base constituye una solución muy completa, pues lograría una cobertura aceptable en estos lugares, y se brindaría un servicio seguro ya que se puede tener el control de los usuarios registrados. Esta implementación podría funcionar brindando servicios de telefonía celular de tercera generación en lugares donde no existe cobertura, y también en casos de emergencia cuando la red convencional no esté en funcionamiento. Al ser un sistema relativamente pequeño, de pocos recursos y fácil ejecución, se puede desplegar tecnológicamente en muy poco tiempo.

6.1 Trabajos Futuros

El desarrollo de una red UMTS con OpenBTS es muy reciente todavía. El código fuente de la versión pública aún presenta algunas inconsistencias en la adquisición de número IP independiente, y problemas de compatibilidad con varios modelos de celulares. Por estos motivos no fue posible realizar análisis de tráfico de datos y llamadas en tiempo real. En este momento la versión OpenBTS-UMTS 1.0 cuenta con un total de 21 errores, de los cuales 5 se han solucionado, uno se encuentra en proceso de verificación y 15 no tienen propuesta de solución todavía. Sin embargo, en la plataforma de desarrollo colaborativo GitHub se continúa trabajando para superar las deficiencias, así como para optimizar los algoritmos de acceso al medio y calendarización de los recursos.

Este proyecto de tesis brinda una perspectiva inicial para posteriormente, con la mejora del código, realizar estudios más profundos en diferentes tipos de escenarios. El número de terminales móviles soportados y el volumen de tráfico que pudiera manejar el equipo USRP N210 dependerá mucho de los recursos de hardware que tenga la computadora donde se encuentra instalado OpenBTS. Pero también es un estudio abierto que resulta importante para el análisis de viabilidad en zonas rurales y proyectos comunitarios.

La comprensión de la plataforma OpenBTS, proporciona las bases para trabajos futuros de implementación de SDR en otras tecnologías como LTE. Hoy en día se encuentran en desarrollo la próxima versión de OpenBTS-UMTS y también la plataforma OpenLTE, lo que resulta una motivación para continuar investigando y aportando mejoras a la tecnología de Radios Definidos por Software.

Como trabajo futuro concreto, se propone implementar en la práctica la propuesta realizada en la sección 4.8, que fue simulada en la sección 5.6. La implementación de esta radiobase UMTS 3G, sin duda puede aportar conocimiento a los estudiantes de la carrera de telecomunicaciones, donde podrán interactuar y comprender estas tecnologías en la práctica.

BIBLIOGRAFÍA

[1] Chen. H. H. Next generation CDMA technologies. Sep 2007.

[2] ETSI TC SMG, «EDGE Feasibility Study Work Item 184; Improved Data Rates through Optimised Modulation Version 0.2». 13-oct-1997.

[3] T. Halonen, J. Romero, y J. Melero, Eds., GSM, GPRS, and EDGE performance: evolution towards 3G/UMTS, 2nd ed. Chichester, West Sussex, England ; Hoboken, NJ, USA: J. Wiley, 2003.

[4] Martínez. E. (2001). La evolución de la telefonía móvil, la guerra de los celulares. Revista RED, 11.

[5] Marcellini, C., Martínez, D. Quiroz, S. y González, A (2013). La próxima generación en comunicaciones móviles. Universidad Técnica Federico Santa María Departamento de Electónica , 4G LTE.

[6] Dubendorf. V A. Wireless data technologies, reference handbook. 2003.

[7] Primera demostración pública del sistema de telefonía celular PAN-Europea GSM, La Vanguardia, p. 55, July 1991.

[8] J. Mitola, "Software radios-survey, critical evaluation and future directions," in Telesystems Conference, 1992. NTC-92, National, 1992, pp. 13/15-13/23.

[9] Instituto Federal de Telecomunicaciones (IFT) de México. Primer Informe Trimestral Estadístico 2016.

[10] Shankar, G., Venkita, G., Gregory, A. P. y Seyed, Z. (2014). Generic SDR Architecture: Vendor Independent Implementation. IEEEAC, 978 (1).

[11] Shome, P., Yan, M., Najafabad, S. M., Mastronarde, N. y Sprintson, A. (2015). CrossFlow: A Cross-layer Architecture for SDR Using SDN Principles. IEEE Conference on Network Function Virtualization and Software Defined Networks 2015 Demo Track, 4673 (1).

[12] Ruelas. A.L. (2010). El teléfono celular y las aproximaciones para su estudio. Comunicación y Sociedad, 143(167).
http://www.adecom.biz/pdf/pdf_agosto2005/La%20evolucion%20de%20la%20telefonia%20movil.pdf

[13] Tesis: Introducción a la Telefonía Celular, IPN. Robledo Ramos Carlos, 2007. Link: http://tesis.ipn.mx/bitstream/handle/123456789/6895/ice%20181.pdf?sequence=1

[14] Rey. E. “Comunicaciones móviles”. Marcombo, 1998.

[15] Gibson, Stephen W. “Cellular Mobile Radiotelephones System”. Prentice Hall, 1987

[16] Dahlman. E. 4G: LTE/LTE-Advanced for Mobile Broadband, 1st ed. 2011.

[17] Garg, Vijay K, Wilkes, Joseph E. “Wireless and Personal Communications Systems”. Prentice Hall, 1996.

[18] Gemalto, «Security to be free: Characteristics and uses of 5G networks». Gemalto, 2014.
Link: https://www.gemalto.com/brochures-site/download-site/Documents/tel-5G-networks-QandA-es.pdf
[19] BOSE, V. (2004). A Software Driven Approach to SDR Design. The Journal of Military Electronics & Computing. 864 (617).
[20] A. P. S. Kalsi, «SMT-8036 based implementation of secured adaptive Software Defined Radio system for LDPC coded modulation techniques», THAPAR UNIVERSITY, 1956.
[21] Dahlman. Erick., Parkvall. S., Scöld. J., Beming. P. 3G Evolution: HSPA and LTE for Mobile Broadband. May 2009.
[22] 3GPP, «3rd Generation Partnership Project; Technical Specification Group GSM/EDGE Radio Access Network; Radio transmission and reception (Release 8) ». 3GPP TS 45.005, mar-2010.
[23] Holma. H., Toskala. A. WCDMA for UMTS - HSPA Evolution and LTE, 4th Edition. September 2007.
[24] Sesia. S., Baker. M. LTE - The UMTS Long Term Evolution, From Theory to Practice. 2011.
[25] Rangel, Víctor. "Temas Selectos de Telecomunicaciones: LTE 4G", Posgrado Ingeniería UNAM, 2017.
[26] Andreas F. Molisch. Wireless Communications, Second Edition.
[27] Fox. D. Testing UMTS: Assuring Conformance and Quality of UMTS User Equipment. May 2008.
[28] Carlevaro. P., Vázquez. M. UMTS: Estándar de Tercera Generación de Telefonía Celular. Facultad de Ingeniería, Universidad de la República, Agosto 2001.
[29] 3GPP TS 23.121: Architectural Requirements for Release 99.
[30] 3GPP TS 21.133: 3G Security; Security Threats and Requirements.
[31] M. Iedema, H.Samra, Getting Started with OpenBTS, January 2015.
[32] Bell Labs. The Creation of the UNIX Operating System, 2002.
[33] Franklin D. Ohrtman, Jr. 2003. SOFTSWITCH Architecture for VoIP.
[34] GSM 04.04 (ETS 300 553): "Digital cellular telecommunications system (Phase 2); layer 1 General requirements", 1999.
[35] Range Networks, OpenBTS Application Suite, http://openbts.org/site/wp-content/uploads/2014/07/OpenBTS-Manual.pdf [Acceso 20/5/18].
[36] 3GPP, «3rd Generation Partnership Project; Technical Specification Group GSM/EDGE Radio Access Network; Radio transmission and reception (Release 8)». 3GPP TS 45.005, mar-2010.
[37] Sanchez. J., Thioune. M. UMTS. Edition January 2010.
[38] E. Seurre, EDGE for mobile Internet. Boston: Artech House, 2003.

[39] Caracteristicas y especificacion de OpenBTS-UMTS, disponible en: http://openbts.org/w/index.php?title=OpenBTS-UMTS.
[40] Brumley. B. A3/A8 & COMP128, (2004).
[41] Mao. S., Huang. Yingsong., Li. Yihan. «Introducing Software Defined Radio into Undergraduate Wireless Engineering Curriculum through a Hands-on Approach», presentado en 120th ASEE Annual Conference and Exposition, Atlanta Georgia, Estados unidos, 2013.
[42] Ettus Research a National Instruments Company.
Disponible en: http://www.ettus.com/home.
[43] «Ettus Research SBX - Product Detail». [En línea]. Disponible en:
http://www.ettus.com/product/details/SBX. [Accedido: 19-mayo-2018].
[44] http://openbts.org/ [Acceso 19/5/18].
[45] Gorricho. M., Gorricho J. L. Comunicaciones Móviles, 1st ed. Barcelona, España: Ediciones UPC, 2002.
[46] Arun Murthy, Vinod Vavilapalli, Douglas Eadline, Joseph Niemiec, Jeff Markham. Apache Hadoop YARN: Moving beyond MapReduce and Batch Processing with Apache.
[47] S. Gindraux, "From 2G to 3G: a guide to mobile security", ISBN 0537-9989, publicado 10 de mayo de 2002 en IEEE Xplore, disponible en: http://ieeexplore.ieee.org/abstract/document/1032044/
[48] Gaughan. K. Client-Server Programming with TCP/IP Sockets, 2003.
[49] Super SIM X-Sim specifications "grcard", disponible en:
https://es.aliexpress.com/store/product/16-in-1-Max-SIM-Cell-Phone-Magic-Super-Card-Integrate-Backup-all-your-Sims/2007066_32810074014.html
[50] ISO/IEC 7816-4:2013 Identification cards - Integrated circuit cards - Part 4: Organization, security and commands for interchange. 2013.
[51] ISO/IEC 7816-4:2013, "Identification cards, Chip Cards and Biometrics" consultado 20 de agosto del 2017 disponible en:
https://www.iso.org/standard/54550.html
[52] USB-MCR3512 Smart Card Reader specifications, disponible en:
http://www.bplus.com.tw/CardReader/USB-MCR3512.html
[53] "Controladores UHD ", Ettus Research, consultado 17 de agosto del 2017 disponible en: http://files.ettus.com/binaries/uhd/
[54] ITU-T Rec. X.680 | ISO/IEC 8824-1
[55] Actas Finales de la Conferencia Administrativa Mundial de Radiocomunicaciones para examinar la atribución de frecuencias en ciertas partes del espectro (CAMR-92). Ginebra, Suiza: Unión Internacional de Telecomunicaciones. 1992. p. 278. ISBN 92-61-04663-0.
[56] Acuerdo SCT 130306, DOF 13/03/2006. Inventario de bandas de frecuencias de uso libre, IFT. Última actualización 25/04/2014.

[57] «BANG GOOD GSM900 Amplifier». [En línea]. Disponible en: https://www.banggood.com/High-Gain-GSM-900Mhz-Signal-Booster-Amplifier-with-Cable-Antenna-p-948790.html?cur_warehouse=CN. [Accedido: 09-may-2018].
[58] Microstrip Filters for RF/Microwave Applications. M. J. Lancaster. John Wiley & Sons, 2001.
[59] «EGSM-900 Duplexer Model 14540». [En línea]. Disponible en: http://www.microwavefilter.com/(E)GSM-900_diplexer.htm. [Accedido: 06-may-2018].
[60] « EMPRETEL L-com 900-MHZ-OMNIDIRECCIONALES Modelo HGV906U ». [En línea]. Disponible en: http://empretel.com.mx/900-mhz-omnidireccionales/733-antena-omnidireccional-6dbi-900mhz-conector-n-hembra-polarizacion-vertical.html. [Accedido: 11-may-2018].
[61] Tran. D. Ch., Rosdiazli. I., Vijanth. S. A., Nordin. S., Sabo. M. H. WirelessHART^{TM}: Filter Design for Industrial Wireless Networked Control Systems. November 16, 2017.
[62] Matías, José María. "Temas Selectos de Telecomunicaciones: Radiodifusión Digital Sonora", Posgrado Ingeniería UNAM, 2017.
[63] Recomendación UIT-R P.525-3. Cálculo de la atenuación en el espacio libre, 2016.
[64] Revista del Consumidor No. 284, Octubre 2010. Estudio de calidad de teléfonos celulares. Disponible en: https://www.profeco.gob.mx/revista/pdf/est_00/telcelu.pdf
[65] Rangel, Víctor. "Redes y Servicios Integrados". Posgrado Ingeniería UNAM, año 2016.
[66] Tomasi. W. Sistemas de Comunicaciones Electrónicas, Cuarta Edición. 2003.
[67] Recomendación FCC/OET-74. OET BULLETIN, Oct 26, 2015.
[68] Seybold. J.S. Introduction to RF Propagation. 2005.
[69] Pérez. C., Zamanillo. J. M., Casanueva. A. Sistemas de Telecomunicaciones, Textos Universitarios No 7 Ingenierías, 2007.
[70] «Tutorial de Radio Mobile». [En línea]. Disponible en: http://www.eslared.net/walcs/walc2011/material/track1/Manual%2520de%2520Radio%2520Mobile.pdf. [Accedido: 15-may-2018].

ANEXOS

Anexo 1: Código sipauthserve.cpp

```
#include <arpa/inet.h>
#include <cstdlib>
#include <ctype.h>
#include <errno.h>
#include <fcntl.h>
#include <fstream>
#include <iostream>
#include <netinet/in.h>
#include <osip2/osip.h>
#include <osipparser2/osip_message.h>
#include <stdlib.h>
#include <string.h>
#include <sys/socket.h>
#include <sys/stat.h>
#include <sys/time.h>
#include <sys/types.h>
#include <time.h>
#include <unistd.h>
#include <Logger.h>
#include <Globals.h>
#include <NodeManager.h>
#include <JSONDB.h>
#include "servershare.h"
#include "SubscriberRegistry.h"
using namespace std;
ConfigurationTable gConfig("/etc/OpenBTS/sipauthserve.db", "sipauthserve", getConfigurationKeys());
Log dummy("sipauthserve", gConfig.getStr("Log.Level").c_str(), LOG_LOCAL7);
int my_udp_port;
SubscriberRegistry gSubscriberRegistry;
/** The remote node manager. */
NodeManager gNodeManager;
/** The JSON<->DB interface. */
JSONDB gJSONDB;
/** Application specific NodeManager logic for handling requests. */
JsonBox::Object nmHandler(JsonBox::Object& request)
{
        JsonBox::Object response;
        std::string command = request["command"].getString();
        std::string action = request["action"].getString();
        if (command.compare("subscribers") == 0) {
                request["table"] = JsonBox::Value("subscribers");
                response = gJSONDB.query(request);
        } else {
                response["code"] = JsonBox::Value(501);
        }
        return response;
}
void prettyPrint(const char *label, osip_message_t *sip)
{
        char *dest=NULL;
        size_t length=0;
        int i = osip_message_to_str(sip, &dest, &length);
        if (i!=0) {
                LOG(ERR) << "cannot get printable message";
        } else {
                LOG(INFO) << label << ":\n" << dest;
                osip_free(dest);
        }
}
string imsiFromSip(osip_message_t *sip)
{
        char *dest;
        osip_uri_t *fromUri = osip_from_get_url(sip->from);
        if (!fromUri) {
                LOG(ERR) << "osip_from_get_url problem";
                return "";
                        }
        osip_uri_to_str(fromUri, &dest);
        string imsi = dest;
        osip_free(dest);
        printf("procesando IMSI\n");
        return imsi;
```

```
}
string imsiToSip(osip_message_t *sip)
{
        char *dest;
        osip_uri_t *toUri = osip_to_get_url(sip->to);
        if (!toUri) {
                LOG(ERR) << "osip_to_get_url problem";
                return "";
        }
        osip_uri_to_str(toUri, &dest);
        string imsi = dest;
        osip_free(dest);
        return imsi;
}
bool imsiFound(string imsi)
{
        printf("checando IMSI Vs Database\n");
        string x = gSubscriberRegistry.imsiGet(imsi, "id");
        return x.length() != 0;
}
string imsiClean(string imsi)
{
        if (0 == strncasecmp(imsi.c_str(), "sip:", 4)) {
                imsi = imsi.substr(4);
        }
        size_t p = imsi.find("@");
        if (p != string::npos) {
                imsi = imsi.substr(0, p);
        }
        if (0 == strncasecmp(imsi.c_str(), "imsi", 4)) {
                imsi = imsi.substr(4);
        }
        return imsi;
}
char *processBuffer(char *buffer)
{
        int i;
        osip_message_t *sip;
        i=osip_message_init(&sip);
        if (i!=0) {
                LOG(ERR) << "cannot allocate";
                osip_message_free(sip);
                return NULL;
        }
        i=osip_message_parse(sip, buffer, strlen(buffer));
        if (i!=0) {
                LOG(ERR) << "cannot parse sip message";
                osip_message_free(sip);
                return NULL;
        }
        prettyPrint("request", sip);
        osip_message_t *response;
        osip_message_clone(sip, &response);
        osip_from_t * contact_header = (osip_from_t*)osip_list_get(&sip->contacts,0);
        osip_uri_t* contact_url = contact_header->url;
        printf("procesando autenticación\n");
        char *remote_host = contact_url->host;
        char *remote_port = contact_url->port;
        ostringstream newvia;
        const char *my_ipaddress = "localhost";
        newvia << "SIP/2.0/UDP " << my_ipaddress << ":" << my_udp_port << ";branch=1;received="
                << "string_address@foo.bar";
                printf("comprobando identidad \n");
        osip_message_append_via(response, newvia.str().c_str());
        osip_message_set_method(response, NULL);
        string imsi = imsiClean(imsiFromSip(sip));
        string imsiTo = imsiClean(imsiToSip(sip));
        if ((imsi == "EXIT") && (imsiTo == "EXIT")) exit(0);
        if (!imsiFound(imsi)) {
                LOG(NOTICE) << "imsi unknown";
                printf("no encuentra IMSI\n");
                osip_message_set_status_code (response, 404);
                osip_message_set_reason_phrase (response, osip_strdup("IMSI Not Found"));
        } else if (gConfig.defines("SubscriberRegistry.IgnoreAuthentication")) {
               osip_message_set_status_code (response, 200);
               osip_message_set_reason_phrase (response, osip_strdup("OK"));
               LOG(INFO) << "success, imsi " << imsi << " registering for IP address " << remote_host;
               printf("parece que le da un IP success imsi\n");
               gSubscriberRegistry.imsiSet(imsi,"ipaddr", remote_host, "port", remote_port);
        } else {
                string randx;
                string sres;
```

```
                char *RAND = strcasestr(buffer, "nonce=");
                printf("checando RAND y AUTN\n");
                char *SRES = strcasestr(buffer, "response=");
                printf("comprobando XRES\n");
                if (RAND && SRES) {
                        RAND += 6;
                        while (!isalnum(*RAND)) { RAND++; }
                        RAND[32] = 0;
                        int j=0;
                        while(isalnum(RAND[j])) { j++; }
                        RAND[j] = '\0';
                        SRES += 9;
                        while (!isalnum(*SRES)) { SRES++; }
                        int i=0;
                        while(isalnum(SRES[i])) { i++; }
                        SRES[i] = '\0';
                        LOG(INFO) << "rand = /" << RAND << "/";
                        LOG(INFO) << "sres = /" << SRES << "/";
                }
                if (!RAND || !SRES) {
                        LOG(NOTICE) << "imsi " << imsi << " known, 1st register";
                        printf("RES = XRES registrar usuario\n");
                        osip_message_set_status_code (response, 401);
                        osip_message_set_reason_phrase (response, osip_strdup("Unauthorized"));
                        osip_www_authenticate_t *auth;
                        osip_www_authenticate_init(&auth);
                        printf("autenticando usuario\n");
                        string auth_type = "Digest";
                        osip_www_authenticate_set_auth_type(auth, osip_strdup(auth_type.c_str()));
                        string randz = generateRand(imsi);
                        osip_www_authenticate_set_nonce(auth, osip_strdup(randz.c_str()));
                        i = osip_list_add (&response->www_authenticates, auth, -1);
                        printf("autenticando usuario\n");
                        if (i < 0) LOG(ERR) << "problem adding www_authenticate";
                } else {
                        string kc;
                        printf("distribucion de CK IK\n");
                        bool sres_good = authenticate(imsi, RAND, SRES, &kc);
                        LOG(INFO) << "imsi " << imsi << " known, 2nd register, good = " << sres_good;
                        if (sres_good) {
                                osip_message_set_status_code (response, 200);
                                osip_message_set_reason_phrase (response, osip_strdup("OK"));
                                if (kc.size() != 0) {
                                        osip_authentication_info *auth;
                                        osip_authentication_info_init(&auth);
                                        osip_authentication_info_set_cnonce(auth,
osip_strdup(kc.c_str()));
                                        i = osip_list_add (&response->authentication_infos, auth, -1);
                                        if (i < 0) LOG(ERR) << "problem adding authentication_infos";
                                }
                                static string calleridstr("callerid");
                                string callid = gSubscriberRegistry.imsiGet(imsi,calleridstr);
                                if (callid.size()) {
                                        char buf[120];
                                        snprintf(buf,120,"<tel:%s>",callid.c_str());
                                        osip_message_set_header(response,"P-Associated-URI",buf);
                                }
                                LOG(INFO) << "success, registering for IP address " << remote_host;
                                gSubscriberRegistry.imsiSet(imsi,"ipaddr", remote_host, "port",
remote_port);
                                printf("Registro exitoso\n");
                        } else {
                                printf("no autorizado\n");
                                osip_message_set_status_code (response, 401);
                                osip_message_set_reason_phrase (response, osip_strdup("Unauthorized"));
                        }
                }
        }
        prettyPrint("response", response);
        size_t length = 0;
        char *dest;
        int ii = osip_message_to_str(response, &dest, &length);
        if (ii != 0) {
                LOG(ERR) << "cannot get printable message";
        }
        osip_message_free(sip);
        osip_message_free(response);
        return dest;
}
#define BUFLEN 5000
int
main(int argc, char **argv)
```

```
{
        if (argc > 1) {
                for (int argi = 0; argi < argc; argi++) {
                        if (!strcmp(argv[argi], "--version") ||
                            !strcmp(argv[argi], "-v")) {
                                cout << gVersionString << endl;
                        }
                        if (!strcmp(argv[argi], "--gensql")) {
                                cout << gConfig.getDefaultSQL(string(argv[0]), gVersionString) << endl;
                        }
                        if (!strcmp(argv[argi], "--gentex")) {
                                cout << gConfig.getTeX(string(argv[0]), gVersionString) << endl;
                        }
                }
                return 0;
        }
        sockaddr_in si_me;
        sockaddr_in si_other;
        int aSocket;
        char buf[BUFLEN];
        LOG(ALERT) << argv[0] << " (re)starting";
        srand ( time(NULL) + (int)getpid() );
        my_udp_port = gConfig.getNum("SubscriberRegistry.Port");
        gSubscriberRegistry.init();
        gNodeManager.setAppLogicHandler(&nmHandler);
        gNodeManager.start(45064);
        osip_t *osip;
        int i=osip_init(&osip);
        if (i!=0) {
                LOG(ALERT) << "cannot init sip lib";
                exit(1);
        }
        if ((aSocket = socket(AF_INET, SOCK_DGRAM, IPPROTO_UDP)) == -1) {
                LOG(ALERT) << "can't initialize socket";
                exit(1);
        }
        memset((char *) &si_me, 0, sizeof(si_me));
        si_me.sin_family = AF_INET;
        si_me.sin_port = htons(my_udp_port);
        si_me.sin_addr.s_addr = htonl(INADDR_ANY);
        if (bind(aSocket, (sockaddr*)&si_me, sizeof(si_me)) == -1) {
                LOG(ALERT) << "can't bind socket on port " << my_udp_port;
                exit(1);
        }
        LOG(NOTICE) << "binding on port " << my_udp_port;
        while (true) {
                gConfig.purge();
                socklen_t slen = sizeof(si_other);
                memset(buf, 0, BUFLEN);
                if (recvfrom(aSocket, buf, BUFLEN, 0, (sockaddr*)&si_other, &slen) == -1) {
                        LOG(ERR) << "recvfrom problem";
                        continue;
                }
                LOG(INFO) << " receiving " << buf;
                char *dest = processBuffer(buf);
                if (dest == NULL) {
                        continue;
                }
                if (sendto(aSocket, dest, strlen(dest), 0, (sockaddr*)&si_other, sizeof(si_other)) == -
1) {
                        LOG(ERR) << "sendto problem";
                        continue;
                }
                osip_free(dest);
        }
        close(aSocket);
        return 0;
}
```

Anexo 2: Ayuda para programar la tarjeta SIM con el software pySIM

```
root@lenovo:/home/peter/pysim# ./pySim-prog.py -help
Usage: pySim-prog.py [options]

Options:
  -h, --help            show this help message and exit
  -d DEV, --device=DEV  Serial Device for SIM access [default: /dev/ttyUSB0]
  -b BAUD, --baud=BAUD  Baudrate used for SIM access [default: 9600]
  -p PCSC, --pcsc-device=PCSC
                        Which PC/SC reader number for SIM access
  -t TYPE, --type=TYPE  Card type (user -t list to view) [default: auto]
  -a PIN_ADM, --pin-adm=PIN_ADM
                        ADM PIN used for provisioning (overwrites default)
  -e, --erase           Erase beforehand [default: False]
  -S SOURCE, --source=SOURCE
                        Data Source[default: cmdline]
  -n NAME, --name=NAME  Operator name [default: Magic]
  -c CC, --country=CC   Country code [default: 1]
  -x MCC, --mcc=MCC     Mobile Country Code [default: 901]
  -y MNC, --mnc=MNC     Mobile Network Code [default: 55]
  -m SMSC, --smsc=SMSC  SMSP [default: '00 + country code + 5555']
  -M SMSP, --smsp=SMSP  Raw SMSP content in hex [default: auto from SMSC]
  -s ID, --iccid=ID     Integrated Circuit Card ID
  -i IMSI, --imsi=IMSI  International Mobile Subscriber Identity
  -k KI, --ki=KI        Ki (default is to randomize)
  -o OPC, --opc=OPC     OPC (default is to randomize)
  --op=OP               Set OP to derive OPC from OP and KI
  --acc=ACC             Set ACC bits (Access Control Code). not all card types
                        are supported
  --read-imsi           Read the IMSI from the CARD
  -z STR, --secret=STR  Secret used for ICCID/IMSI autogen
  -j NUM, --num=NUM     Card # used for ICCID/IMSI autogen
  --batch               Enable batch mode [default: False]
  --batch-state=FILE    Optional batch state file
  --read-csv=FILE       Read parameters from CSV file rather than command line
  --write-csv=FILE      Append generated parameters in CSV file
  --write-hlr=FILE      Append generated parameters to OpenBSC HLR sqlite3
  --dry-run             Perform a 'dry run', don't actually program the card
root@lenovo:/home/peter/pysim#
```

Anexo 3: Mediciones realizadas antes de la implementación

Anexo 4: Especificaciones Generales del terminal móvil Samsung Galaxy J7

Samsung

Galaxy J7

telcel

SM-J700M

Especificaciones

- Dimensiones: 152.4 x 78.6 x 7.5 mm
- Peso: 171 grs.
- Duración de la batería:

Tiempo de espera 4G:	Hasta 30 hrs
Tiempo de espera 3G:	Hasta 37 hrs
Tiempo de conversación 3G:	Hasta 18 hrs
Tiempo de espera GSM:	Hasta 57 hrs
Tiempo de conversación GSM:	Hasta 27 hrs

Otras características

Ranura para Memoria Micro SD	Micro SD, expandible hasta 64 GB.
Cliente de Correo electrónico	Consulta tu cuenta de correo electrónico desde tu teléfono.
Tipo de SIM	MicroSIM
Módem	Puedes conectar tu teléfono a una PC y utilizarlo como módem.

Bandas y Frecuencias disponibles

Tecnología	Bandas y Frecuencias disponibles
GSM	3/1800+, 5/850, 8/900, 2/1900
3G	5/850, 8/900, 2/1900, 1/2100
4G	4/1700-2100, 3/1800+, 28/700Mhz (APT), 17/700bc, 2/1900, 1/2100, 7/2600, 5/850

Anexo 5: Hoja de datos USRP N210

Ettus Research

A National Instruments Company

USRP™ N200/N210 NETWORKED SERIES

FEATURES:

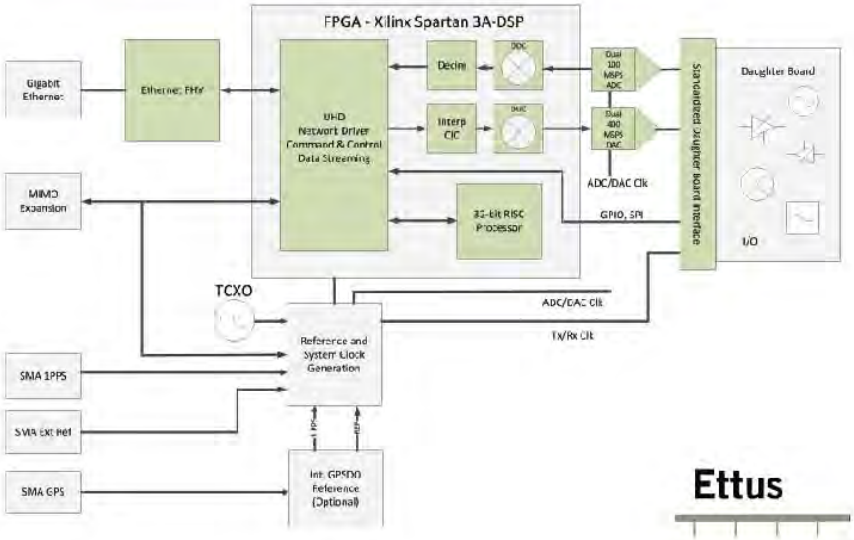
- Use with GNU Radio, LabVIEW™ and Simulink™
- Modular Architecture: DC-6 GHz
- Dual 100 MS/s, 14-bit ADC
- Dual 400 MS/s, 16-bit DAC
- DDC/DUC with 25 mHz Resolution
- Up to 50 MS/s Gigabit Ethernet Streaming
- Fully-Coherent MIMO Capability
- Gigabit Ethernet Interface to Host
- 2 Gbps Expansion Interface
- Spartan 3A-DSP 1800 FPGA (N200)
- Spartan 3A-DSP 3400 FPGA (N210)
- 1 MB High-Speed SRAM
- Auxiliary Analog and Digital I/O
- 2.5 ppm TCXO Frequency Reference
- 0.01 ppm w/ GPSDO Option

SPECIFICATIONS

Spec	Typ	Unit
POWER		
DC Input	6	V
Current Consumption	1.3	A
w/ WBX Daughterboard	2.3	A
CONVERSION PERFORMANCE AND CLOCKS		
ADC Sample Rate	100	MS/s
ADC Resolution	14	bits
ADC Wideband SFDR	88	dBc
DAC Sample Rate	400	MS/s
DAC Resolution	16	bits
DAC Wideband SFDR	80	dBc
Host Sample Rate (8b/16b)	50/25	MS/s
Frequency Accuracy	2.5	ppm
w/ GPSDO Reference	0.01	ppm

Spec	Typ	Unit
RF PERFORMANCE (W/ WBX)		
SSB/LO Suppression	35/50	dBc
Phase Noise (1.8 Ghz)		
10 kHz	-80	dBc/Hz
100 kHz	-100	dBc/Hz
1 MHz	-137	dBc/Hz
Power Output	15	dBm
IIP3	0	dBm
Receive Noise Figure	5	dB
PHYSICAL		
Operating Temperature	0 to 55°	C
Dimensions (l x w x h)	22 x 16 x 5	cm
Weight	1.2	kg

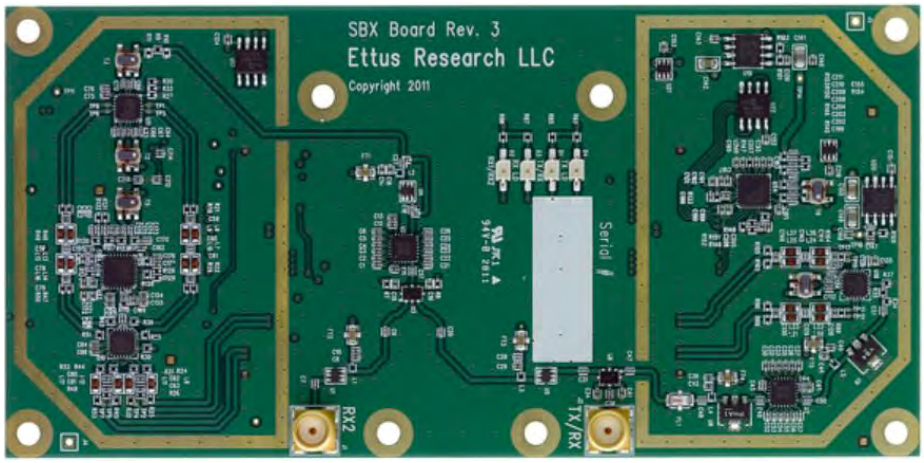
* All specifications are subject to change without notice.



Anexo 6: Hoja de datos Daughterboard SBX

Ettus Research™ A National Instruments Company

SBX 400-4400 MHz Rx/Tx (40 MHz)



Features:

- 2 quadrature frontends (1 transmit, 1 receive)
  - Defaults to direct conversion
  - Can be used in low IF mode through lo_offset with **uhd::tune_request_t**
- Independent receive and transmit LO's and synthesizers
  - Allows for full-duplex operation on different transmit and receive frequencies
  - Can be set to use Integer-N tuning for better spur performance with **uhd::tune_request_t**

Frequency Range: 400 MHz to 4.4 GHz
Transmit Antennas: **TX/RX**
Receive Antennas: **TX/RX** or **RX2**

- **Frontend 0:** Complex baseband signal for selected antenna
- **Note:** The user may set the receive antenna to be TX/RX or RX2. However, when using an SBX board in full-duplex mode, the receive antenna will always be set to RX2, regardless of the settings.

Transmit Gains: **PGA0**, Range: 0-31.5dB
Receive Gains: **PGA0**, Range: 0-31.5dB
Bandwidths:

- **SBX**: 40 MHz, RX & TX
- **SBX-120**: 120 MHz, RX & TX

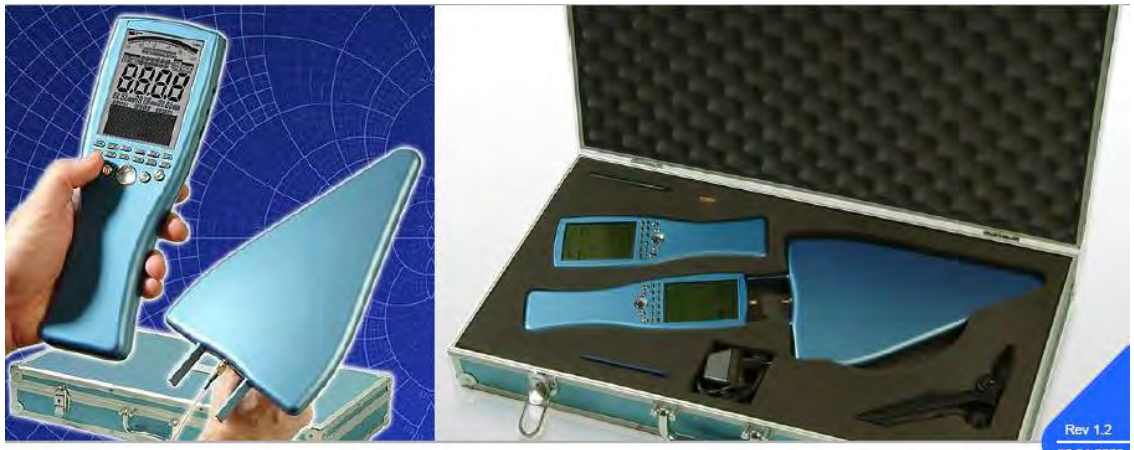
Sensors:

- **lo_locked**: boolean for LO lock state

LEDs:

- All LEDs flash when daughterboard control is initialized
- **TX LD**: Transmit Synthesizer Lock Detect
- **TX/RX**: Receiver on TX/RX antenna port (No TX)
- **RX LD**: Receive Synthesizer Lock Detect
- **RX1/RX2**: Receiver on RX2 antenna port

Anexo 7: Hoja de datos del Analizador de Espectros



Rev 1.2
06.04.2009

Handheld RF Spectrum Analyzer series SPECTRAN® 40xx incl. EMC test antenna

VECTOR Spectrum Analyzer for the semi professional

Frequency range Min	100MHz
Frequency range Max	6GHz
Optional PEAK Power-Detector (Maximum usable frequency)***	6GHz
AVG Noise Level (1Hz)	-90dBm
AVG Noise Level (1Hz) **with PreAmp**	-
Maximum Level	0dBm
Filter bandwith **(RBW) Min**	100kHz
Filter bandwith **(RBW) Max**	50MHz
EMC-Filter (RBW) 9kHz, 120kHz, 5MHz; 20MHz; 40MHz	-
Accuracy Base unit (typical)	+/-3dB
Vector power measurement (I/Q) and True RMS	✓
Lowest possible **SampleTime**	100mS

SPECTRAN® HF-4060 Rev.3

- Frequency range: 100MHz to **6GHz***
- Typ. level range: -90dBm to 0dBm*
- Lowest possible SampleTime: 100mS
- Typ. accuracy: +/- 3dB*
- Filter bandwith (RBW) Min: 100kHz
- Filter bandwith (RBW) Max: 50MHz
- Vector (I/Q) / True RMS level measurement
- High performance DSP (Digital Signal Processor)
- USB 2.0 interface
- Direct RF spectrum display
- Frequency and signal strength display
- Enhanced triple multi-function display
- Advanced HOLD function
- Switchable PULS mode
- Exposure limit calculation according to DIN/VDE 0848
- AM / FM Demodulation
- DECT & TimeSlot Analyser
- Realtime PEAK power detector (option)
- **1MB memory expansion (option)**
- Internal datalogger (64K)
- Internet software updates
- Incl. battery pack and charger
- Incl. HyperLOG **7060** EMC antenna
- Incl. aluminum carrycase

Enlace: http://www.wifi-shop.cz/img.asp?attid=3865

Anexo 8: Mediciones de RSSI

Banda de 900 MHz										
dist. (m)	*1ra med. (dBm)*	*2da med. (dBm)*	*3ra med. (dBm)*	*4ta med. (dBm)*	*5ta med. (dBm)*	*6ta med. (dBm)*	*7ma med. (dBm)*	*8va med. (dBm)*	*desviación estándar*	*valor prom. (dBm)*
30	-65	-66	-66	-65	-64	-65	-66	-65	0.71	-65.25
40	-73	-74	-73	-75	-73	-76	-72	-73	1.30	-73.63
50	-78	-79	-78	-77	-78	-78	-79	-78	0.64	-78.13
60	-78	-79	-77	-78	-78	-79	-78	-79	0.71	-78.25
70	-79	-79	-78	-78	-77	-79	-78	-79	0.74	-78.38
80	-81	-82	-81	-82	-83	-81	-81	-80	0.92	-81.38
90	-82	-83	-84	-82	-82	-83	-80	-82	1.16	-82.25
100	-83	-82	-85	-83	-83	-84	-83	-84	0.92	-83.38
110	-83	-83	-84	-83	-83	-85	-83	-84	0.76	-83.50
120	-83	-85	-85	-83	-82	-83	-85	-83	1.19	-83.63
130	-84	-84	-85	-84	-83	-86	-84	-84	0.89	-84.25
140	-88	-88	-89	-85	-89	-89	-88	-89	1.36	-88.13
150	-94	-94	-92	-94	-95	-96	-94	-94	1.13	-94.13
160	-96	-97	-98	-96	-96	-95	-96	-95	0.99	-96.13
170	-97	-98	-96	-97	-99	-97	-96	-98	1.04	-97.25
180	-99	-99	-100	-101	-98	-99	-98	-99	0.99	-99.13

Banda de 850 MHz										
dist. (m)	*1ra med. (dBm)*	*2da med. (dBm)*	*3ra med. (dBm)*	*4ta med. (dBm)*	*5ta med. (dBm)*	*6ta med. (dBm)*	*7ma med. (dBm)*	*8va med. (dBm)*	*desviación estándar*	*valor prom. (dBm)*
30	-65	-66	-64	-65	-67	-65	-65	-66	0.92	-65.38
40	-67	-66	-67	-68	-66	-67	-69	-68	1.04	-67.25
50	-73	-74	-73	-75	-73	-76	-72	-73	1.30	-73.63
60	-74	-75	-74	-76	-74	-73	-74	-75	0.92	-74.38
70	-75	-75	-76	-75	-76	-74	-75	-75	0.64	-75.13
80	-74	-75	-76	-76	-77	-75	-74	-75	1.04	-75.25
90	-80	-81	-83	-79	-80	-79	-84	-79	1.92	-80.63
100	-82	-83	-81	-86	-82	-83	-82	-81	1.60	-82.50
110	-84	-86	-84	-83	-84	-85	-84	-84	0.89	-84.25
120	-84	-83	-85	-84	-86	-84	-86	-83	1.19	-84.38
130	-83	-84	-86	-84	-86	-87	-84	-84	1.39	-84.75
140	-88	-87	-89	-90	-87	-88	-89	-87	1.13	-88.13
150	-89	-89	-88	-89	-91	-91	-88	-89	1.16	-89.25
160	-91	-90	-91	-92	-91	-93	-91	-90	0.99	-91.13
170	-92	-91	-93	-94	-95	-92	-92	-91	1.41	-92.50
180	-97	-98	-99	-97	-98	-97	-99	-97	0.89	-97.75

Banda de 1900 MHz										
distancia (m)	*1ra med. (dBm)*	*2da med. (dBm)*	*3ra med. (dBm)*	*4ta med. (dBm)*	*5ta med. (dBm)*	*6ta med. (dBm)*	*7ma med. (dBm)*	*8va med. (dBm)*	*desviación estándar*	*valor prom. (dBm)*
30	-71.00	-72.00	-71	-73	-72	-71	-73	-75	1.39	-72.25
40	-73.00	-74.00	-73	-74	-75	-72	-72	-73	1.04	-73.25
50	-73.00	-73.00	-74	-73	-72	-75	-73	-74	0.92	-73.38
60	-81.00	-82.00	-81	-83	-81	-80	-83	-82	1.06	-81.63
70	-85.00	-86.00	-85	-86	-85	-86	-84	-87	0.93	-85.50
80	-87.00	-88.00	-87	-87	-88	-86	-90	-88	1.19	-87.63
90	-90.00	-91.00	-90	-91	-90	-93	-91	-90	1.04	-90.75
100	-92.00	-93.00	-91	-92	-92	-93	-92	-92	0.64	-92.13
110	-92.00	-91.00	-92	-93	-93	-94	-92	-91	1.04	-92.25
120	-93.00	-94.00	-93	-95	-93	-92	-95	-90	1.64	-93.13
130	-97.00	-98.00	-97	-98	-99	-97	-96	-99	1.06	-97.63
140	-97.00	-98.00	-97	-99	-97	-98	-96	-98	0.93	-97.50
150	-100.00	-102.00	-102	-100	-101	-101	-99	-101	1.04	-100.75
160	-110.00	-111.00	-109	-110	-112	-113	-108	-112	1.69	-110.63
170	-113.00	-113.00	-113	-112	-112	-113	-113	-112	0.52	-112.63
180	-120.00	-120.00	-120	-120	-120	-120	-120	-120	0.00	-120.00

Anexo 9: Tablas comparativas de SNR

Banda de 900 MHz

d (m)	N (dB)	SNR (dB) teórico	SNR (dB) práctico
30	-131.87	49.06	36.62
40		44.33	28.24
50		40.66	23.74
60		37.67	23.62
70		35.14	23.49
80		32.94	20.49
90		31.01	19.62
100		29.28	18.49
110		27.71	18.37
120		26.28	18.24
130		24.97	17.62
140		23.75	13.74
150		22.61	7.74
160		21.55	5.74
170		20.56	4.62
180		19.62	2.74

Banda de 850 MHz

d (m)	N (dB)	SNR (dB) teórico	SNR (dB) práctico
30	-131.87	49.93	36.49
40		45.20	34.62
50		41.54	28.24
60		38.54	27.49
70		36.01	26.74
80		33.81	26.62
90		31.88	21.24
100		30.15	19.37
110		28.58	17.62
120		27.15	17.49
130		25.84	17.12
140		24.62	13.74
150		23.49	12.62
160		22.43	10.74
170		21.43	9.37
180		20.49	4.12

Banda de 1900 MHz

d (m)	N (dB)	SNR (dB) teórico	SNR (dB) práctico
30	-131.87	40.56	29.62
40		35.83	28.62
50		32.17	28.49
60		29.17	20.24
70		26.64	16.37
80		24.45	14.24
90		22.51	11.12
100		20.78	9.74
110		19.21	9.62
120		17.78	8.74
130		16.47	4.24
140		15.25	4.37
150		14.12	1.12
160		13.06	0.00
170		12.06	0.00
180		11.12	0.00

Anexo 10: Componentes del código que fueron modificados o añadidos para la correcta implementación de la radiobase UMTS 3G.

Enlaces:

- https://github.com/RangeNetworks/OpenBTS-UMTS/commit/d528034ac8dc4eddfc063806350be96a7290fb51
- https://github.com/RangeNetworks/OpenBTS-UMTS/commit/1df760735bfca782d3be42b409a120a74e9f8cbb
- https://github.com/RangeNetworks/OpenBTS-UMTS/commit/f27c5d90ec08bc11506d03205d7cfa8b26387a8b
- https://github.com/RangeNetworks/OpenBTS-UMTS/commit/45193085454229e1a5a63aaf98edf5291f09d574
- https://github.com/RangeNetworks/OpenBTS-UMTS/commit/d0bae40d8b291cbaff11fd2f1ea29d24e9cf76a8
- https://github.com/RangeNetworks/OpenBTS-UMTS/commit/6cb7498e64fb34dcc9a45ea57515016e22485eda
- https://github.com/RangeNetworks/OpenBTS-UMTS/commit/6e9711c0ff917e50dbfb571d6b7f3a074ed02174
- https://github.com/RangeNetworks/OpenBTS-UMTS/commit/89ec9e60463ca5f931362426e67f4c1884698cf2
- https://github.com/RangeNetworks/OpenBTS-UMTS/commit/bb2784c75e508c8e9ac61bbb3f7ebe01120e7cec
- https://github.com/RangeNetworks/OpenBTS-UMTS/commit/00e302f32a4200a7c0ad7cc56033ece9ed3f7ccf
- https://github.com/RangeNetworks/OpenBTS-UMTS/commit/274bda43a81c08a2a521751f6b4dffba3d4e24e1
- https://github.com/RangeNetworks/OpenBTS-UMTS/commit/19724e36a8a39d4558665765db9eafe12e241589
- https://github.com/RangeNetworks/OpenBTS-UMTS/commit/ef76ef005be94019ac63d88ae6b517f9563f0e5a
- https://github.com/RangeNetworks/OpenBTS-UMTS/commit/59292692565ff1a34478f77a4ab8355161c63839
- https://github.com/RangeNetworks/OpenBTS-UMTS/commit/d7ce14cbe7718904bdd51e6d2362e61b91cd44c9
- https://github.com/RangeNetworks/OpenBTS-UMTS/pull/6/commits/fc71e6b8d02f765121552cdc1c0ecb4cb842e1ba
- https://github.com/RangeNetworks/OpenBTS-UMTS/pull/6/commits/bee54f00a25a693dd8c4b0aa518b0ed558435dd0
- https://github.com/RangeNetworks/OpenBTS-UMTS/pull/6/commits/2b37707bc95c917a6af42782c700fb2135ce93c4
- https://github.com/RangeNetworks/OpenBTS-UMTS/blob/master/UMTS/URRC.cpp