PABLO E. BULLIAN

IMPLEMENTACIÓN DE RED CELULAR GSM CON HARDWARE SDR

IMPLEMENTACIÓN DE RED CELULAR GSM CON HARDWARE SDR

PABLO E. BULLIAN

Proyecto Final Ingeniería en Telecomunicaciones

Escuela de Ciencia y Tecnología – Universidad Nacional de San Martin

Septiembre 2017 – version 1.9

Pablo E. Bullian: *Implementación de Red Celular GSM con hardware SDR,*
Proyecto Final Ingeniería en Telecomunicaciones, © Septiembre 2017

TUTOR:
Mg. Ing. Rodolfo O. Salvatore

RESUMEN

El presente proyecto se centra en la aplicación de hardware de radiofrecuencia reconfigurado por software para darle un uso de estación base celular. Además se aprovechan algunas implementaciones de software libre para cumplir los roles de los distintos elementos de una red Core celular y asi poder tener un sistema autónomo completo. Sumado a esto último se investiga el rendimiento del mismo y sus posibles optimizaciones para actuar en un ambiente que no sea únicamente de laboratorio.

Toda la implementación se hace con software FOSS[1] (Software de código abierto y libre) con lo cual no queda reducido a ningún proveedor de software particular sino a un ecosistema mucho más amplio de posibles arreglos para lograr el mismo o similar resultado.

Por último, el presente documento desarrollado en LaTeX[2] se presenta bajo la licencia BY-SA 4.0 de Creative Common[3] para el libre uso del material presentado.

1 Free and Open Source Software: http://www.gnu.org.ua/philosophy/free-software-for-freedom.html.

2 LaTeX: https://www.latex-project.org/.

3 Reconocimiento-CompartirIgual 4.0 Internacional: https://creativecommons.org/licenses/by-sa/4.0/deed.es_ES.

ÍNDICE GENERAL

ÍNDICE DE FIGURAS

ÍNDICE DE CUADROS

Parte I

INTRODUCCION TEORICA

1

INTRODUCCIÓN

La Radio Definida por Software (SDR) es una tecnología en rápida evolución que está recibiendo enormes reconocimiento y generando un amplio interés en la industria de las telecomunicaciones. Sobre todo en los últimos años, los sistemas de radio analógicos están siendo reemplazados por sistemas de radio digital como algunas aplicaciones de radio en espacios militares, civiles y comerciales. Además de esto, los módulos de hardware programables se utilizan cada vez más en los sistemas de radio en diferentes niveles funcionales.

La tecnología SDR tiene como objetivo aprovechar estos desarrollos en hardware para construir un software de sistema de radio basado en arquitectura abierta. La tecnología SDR facilita la implementación de algunos de los módulos funcionales en un sistema de radio tales como modulación / demodulación, generación de señales, codificación y protocolos de capa de enlace (capa 2 del modelo OSI) en software. Esto ayuda a construir sistemas de radio de software reconfigurables donde es posible la selección de parámetros para cada uno de los módulos funcionales antes mencionados.

Aunque el concepto de SDR no es nuevo, la reciente evolución de la circuitería digital ha hecho posible desde el punto de vista práctico muchos de los procesos que tiempo atrás eran solamente posibles desde el punto de vista teórico.

Un sistema de radio basado en hardware tiene una utilidad limitada ya que los parámetros para cada los módulos funcionales son fijos ya que fueron establecidos en el momento del diseño y fabricación. Un sistema de radio construido utilizando la tecnología SDR amplía la gama de aplicaciones utilizando diferentes protocolos de capa de enlace y métodos de modulación / demodulación.

La industria de las comunicaciones inalámbricas comerciales se enfrenta actualmente a problemas de una rápida evolución de los estándares de protocolo de capa de enlace (2.5G, 3G y 4G), la existencia de tecnologías de red inalámbrica en diferentes países que inhiben el despliegue de roaming global, instalaciones y problemas en el despliegue de nuevos servicios / características debido a la amplia Gama de capacidades de los terminales de los usuarios. La tecnología

SDR promete resolver estos problemas mediante la implementación de módulos de software que se ejecutan en una plataforma genérica de hardware.

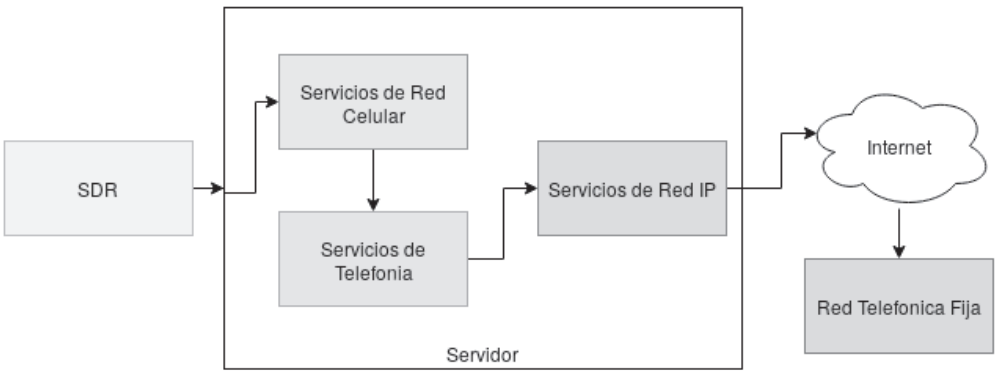


Figura 1: Esquema en bloques del sistema implementado

En la figura 1 se esquematiza a grandes rasgos el proyecto a realizar. Donde con la ayuda de la placa SDR actuando como transmisor/receptor se procesa la información para poder lograr una red celular autónoma que cuenta con los servicios básicos de una red de dichas características como puede ser la voz, sms, servicios adicionales de registración y control, y datos. Esta red también dispone por medio de la tecnología VoIP una conexión a la red telefónica tradicional, completando todas las partes que componen a un servicio de red celular.

A continuación revisaremos algunos conceptos teoricos para comprender mejor el desarrollo realizado.

1.1 PROCESAMIENTO DIGITAL DE SEÑALES

El procesamiento digital de señales (PDS, digital signal processing o DSP) es el tratamiento, análisis y manipulación de la información contenida en una o más señales que a su vez pueden ser representadas en funciones matemáticas específicas, con la finalidad de mejorar o modificar las mismas. En este sentido la señal está caracterizada por manejar la amplitud de forma discreta y por estar en función del dominio del tiempo discreto, las cuales son condiciones necesarias para que la señal pueda ser procesada por un microprocesador o un procesador DSP especializado.

1.1.1 *Teorema de Nyquist, muestreo y cuantificación.*

La señal de la voz es contínua en el tiempo y en amplitud. Para que pueda ser procesada por hardware(y software) digital es necesario convertirla a una señal que sea discreta tanto en el tiempo como en amplitud.

El teorema de muestreo de Nyquist-Shannon, es un teorema fundamental de la teoría de la información. El teorema trata con el muestreo, que no debe ser confundido o asociado con la cuantificación.

Una muestra es un valor numérico en función del tiempo. Este valor es parte de una señal continua o de una señal discreta y son extraídos de éstas para un procesado matemático mediante elementos electrónicos que nos permita realizar funciones tales como el filtrado de una señal. El muestreo con lo que a señales respecta, es el primer paso para el procesamiento de una señal y lo que hace en sí es tomar medidas a intervalos regulares de tiempo, llamadas muestras(samples), se graban temporalmente en un circuito de memoria (hold) y luego se hace con los pulsos resultantes lo que se tenga previsto. Finalmente, el muestreo permite que una señal analógica se convierte en una secuencia de números que normalmente están uniformemente espaciados en el tiempo

Para que dicho proceso tenga utilidad práctica es necesario elegir la tasa de muestreo adecuadamente de modo que esa secuencia de números identifique de forma única a la señal analógica original y que al reproducirla a una determinada velocidad la escuchemos como una señal continua.

Por ejemplo en la figura 2 se denota que si tomamos las muestras de las dos señales en los instantes marcados, ambas encajan con la representación obtenida, haciendo que nuestro muestreo no pueda diferenciar entre ambas dos.

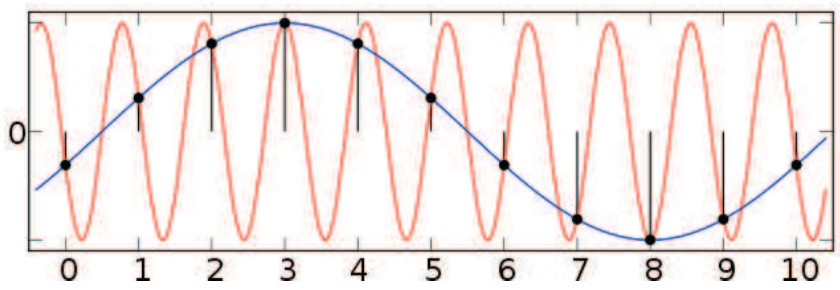


Figura 2: Esquema de sinusoides muestreadas

Si la frecuencia más alta contenida en una señal analógica $x_a(t)$ es $F_{max} = B$ y la señal se muestrea a una tasa $F_s > 2F_{max} \equiv 2B$, entonces $x_a(t)$ se puede recuperar totalmente a partir de sus muestras mediante la siguiente función de interpolación

$$g(t) = \frac{\sin 2\pi Bt}{2\pi Bt}$$

Así, $x_a(t)$ se puede expresar como:

$$x_a(t) = \sum_{n=-\infty}^{\infty} x_a\left(\frac{n}{F_s}\right) g\left(t - \frac{n}{F_s}\right)$$

donde $x_a\left(\frac{n}{F_s}\right) = x_a(nT) \equiv x(n)$ son las muestras de $x_a(t)$.

Segun el teorema una señal continua se puede muestrear correctamente sólo si no contiene componentes de frecuencia superiores a un medio de la frecuencia de muestreo. Si el criterio no es satisfecho, existirán frecuencias cuyo muestreo coincide con otras.

La cuantificación es la conversión de una señal discreta en el tiempo evaluada de forma contínua a una señal discreta en el tiempo discretamente evaluada. El valor de cada muestra de la señal se representa como un valor elegido de entre un conjunto finito de posibles valores.

Se conoce como error de cuantificación (o ruido), a la diferencia entre la señal de entrada (sin cuantificar) y la señal de salida (ya cuantificada), interesa que el ruido sea lo más bajo posible.

1.1.2 *Elementos de un sistema PDS*

La mayoría de las señales en ciencia e ingeniería tienen una naturaleza analógica, es decir, tanto las variables independientes de las funciones que las representan como sus valores son continuos. Matemáticamente estas señales se representan como funciones $f(t)$

$$f : \Re \to \Re$$

es decir, relaciones matemáticas que mapean valores reales en valores reales. Este tipo de señales pueden ser tratadas directamente utilizando sistemas analógicos, como por ejemplo filtros pasivos o analizadores de frecuencia (figura 3).

Figura 3: Procesamiento analógico de una señal

El procesamiento digital (figura 4) requiere transformar las señales de entrada a un formato digital, es decir, a funciones $f(n)$

$$f : \mathbb{Z} \to \mathbb{Z}$$

Esto ocurre en una etapa llamada conversión analógica-digital (A/D). La señal digitalizada es tratada luego en el procesador digital de señales, que puede ser desde una computadora, pasando por sistemas embebidos basados en microcontroladores, hasta circuitos digitales específicamente diseñados para realizar las tareas de procesamiento deseadas; sin embargo, las configuraciones programables tanto en software como en hardware son las que han brindado al procesamiento digital una flexibilidad inalcanzable con respecto a los sistemas analógicos equivalentes. El vertiginoso avance en la electrónica digital ha permitido el uso cada vez más generalizado de las técnicas digitales de procesamiento. En la actualidad hasta los más pequeños

teléfonos celulares utilizan algoritmos de alta complejidad que hace tan solo 15 años hubieran requerido computadores de gran tamaño y costo.

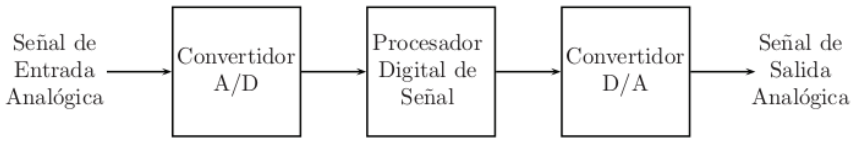


Figura 4: Procesamiento digital de una señal analógica

El último paso del procesamiento digital consiste en convertir la salida del bloque procesador a una señal analogica, lo que ocurre en el llamado convertidor digital-analogico (D/A). En aplicaciones de análisis de señal, la última etapa puede no ser necesaria, cuando la información a extraer se obtiene directamente de las representaciones digitales.

1.1.3 *Conceptos del Procesamiento Digital*

Conceptos algorıtmicos clásicos del procesamiento digital son: compresión, cifrado, reconocimiento, identificación, sintetización, eliminación de ruido, estimación espectral y filtrado, entre otros.

La compresión consiste en la reducción de capacidad necesaria para almacenar o transmitir una señal. En telefonía celular se utiliza la compresion para ahorrar tiempo de transmicion y recepcion de cada terminal para poder optimizar el canal.

El cifrado es necesario para proteger la confidencialidad de la información transmitida. Como la tecnología celular utiliza un medio inseguro (el aire) es importante que se cifre la información.

La sintetización permite producir señales artificiales similares a aquellas generadas a traves de fenomenos fisicos.

Las señales transmitidas por canales analogicos usualmente son perturbadas con ruido, es decir, con alteraciones indeseables que no contienen ninguna información relevante para la aplicación. Por medio del procesamiento digital es posible aplicar diversos algoritmos que

permiten reducir el efecto de dichas distorsiones.

El filtrado es un concepto básico del procesamiento digital que forma parte de prácticamente cualquier otro algoritmo. El sistema que realiza esta tarea se denomina filtro, y permite el paso de solo ciertas "componentes" de su señal de entrada, y bloqueando el paso de otras.

1.2 PRINCIPIO DE OPERACIÓN DE LOS SDR

1.2.1 *Esquema Ideal*

El esquema de receptor ideal sería conectar un convertidor analógico-digital a una antena. Un procesador de señales digitales leería el convertidor, y entonces su software transformaría la trama de datos del convertidor a cualquier otra forma programada en el software.

Un transmisor ideal sería similar. Un procesador de señal digital generaría un flujo de números. Éstos se enviarían a un convertidor digital-analógico conectado a una antena de radio.

El esquema ideal no es completamente realizable debido a los límites reales de la tecnología. El problema principal en ambas direcciones es la dificultad de conversión entre los dominios digital y analógico a una velocidad suficientemente alta y una precisión suficientemente alta al mismo tiempo y sin tener en cuenta procesos físicos como interferencia y resonancia electromagnética.

1.2.2 *Receptores SDR Comerciales*

Como se ve en la figura 5 un transmisor analogico utiliza un proceso de mezcla de frecuencias o heterodinación para convertir la señal recibida en una frecuencia intermedia fija. Cuenta además con filtros y amplificadores. El esquema de procesamiento está fijado al circuito que se realice físicamente.

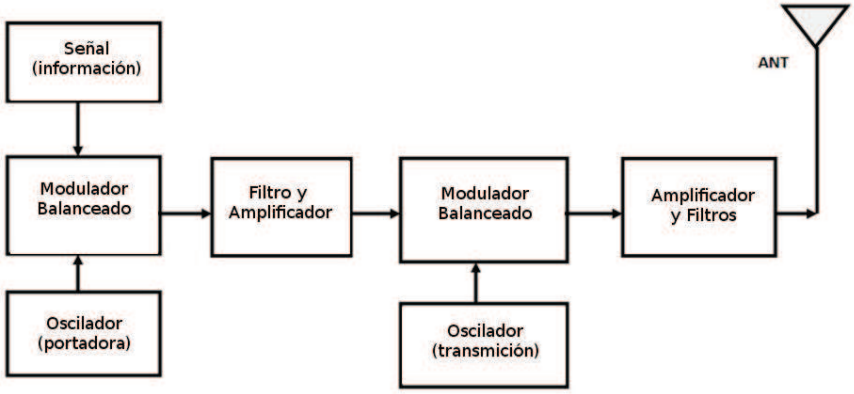


Figura 5: Esquema de un tradicional transmisor analogico

En cambio en un transmisor o receptor SDR como en la figura 6 tenemos una etapa de conversion analogica digital y un procesamiento digital que permite, con la ayuda de software como GNURadio, implementar todo tipo de amplificadores, filtros, mezcladores que pueden ser modificados incluso mientras se procesa la señal sin necesidad de cambiar nada en los circuitos físicos.

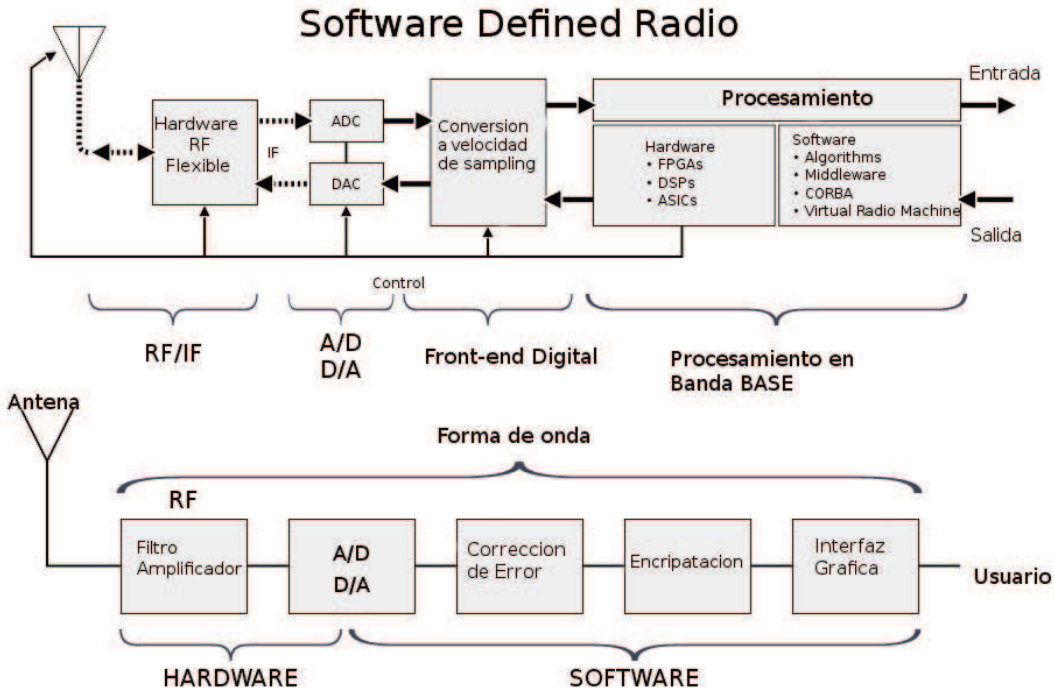


Figura 6: Esquema de un SDR receptor y transmisor

Aunque no todo es software en un SDR hay limitacion fisicas como que los convertidores analógico-digitales reales carecen del rango dinámico para captar señales de radio sub-microvolt o de potencia en los nanowatts. Por lo tanto, un amplificador de bajo ruido debe pre-

SIGLA	DEFINICIÓN
ADC ó A/D	conversor analogico a digital
DAC ó D/A	conversor digital a analogico
FPGA	matriz de puertas programables
DSP	Procesador digital de señales
ASIC	Circuito integrado de aplicación específica
CORBA	Common Object Request Broker Architecture
RF	Radiofrecuencia

Cuadro 1: Definición de las siglas de la figura 6

ceder al paso de conversión y este dispositivo introduce sus propios problemas. Por ejemplo, si hay señales espurias (lo que es típico, siendo las mismas un armonico de la señal deseada), estas compiten con las señales deseadas dentro del rango dinámico del amplificador. Pueden introducir distorsión en las señales deseadas, o pueden bloquearlas completamente. La solución estándar es colocar filtros de paso de banda entre la antena y el amplificador, pero estos reducen la flexibilidad de la radio. Los dispositivos comerciales a menudo tienen dos o tres filtros de canal analógico con diferentes anchos de banda para evitar la frecuencia imagen.

1.3 INTRODUCCIÓN A GNURADIO

Como se especifico en la sección anterior, los SDR se programan para hacer el procesamiento de la señal. Un set de herramientas para facilitar la programación a bajo nivel que se debe hacer es utilizar las bibliotecas de GNURadio que son implementaciones en software (lenguaje de programación Python) de diversos elementos físicos para el tratamiento de las señales digitales.

GNU Radio es un conjunto de herramientas de desarrollo de software libre y de código abierto que proporciona bloques de procesamiento de señales para implementar circuitos de radio por software. Puede utilizarse con hardware de RF externo de bajo costo disponible fácilmente para crear radios definidas por software o sin hardware en un entorno similar a la simulación. Es ampliamente utilizado por aficionados, en entornos académicos y comerciales para apoyar la in-

vestigación de comunicaciones inalámbricas y los sistemas de radio del mundo real.

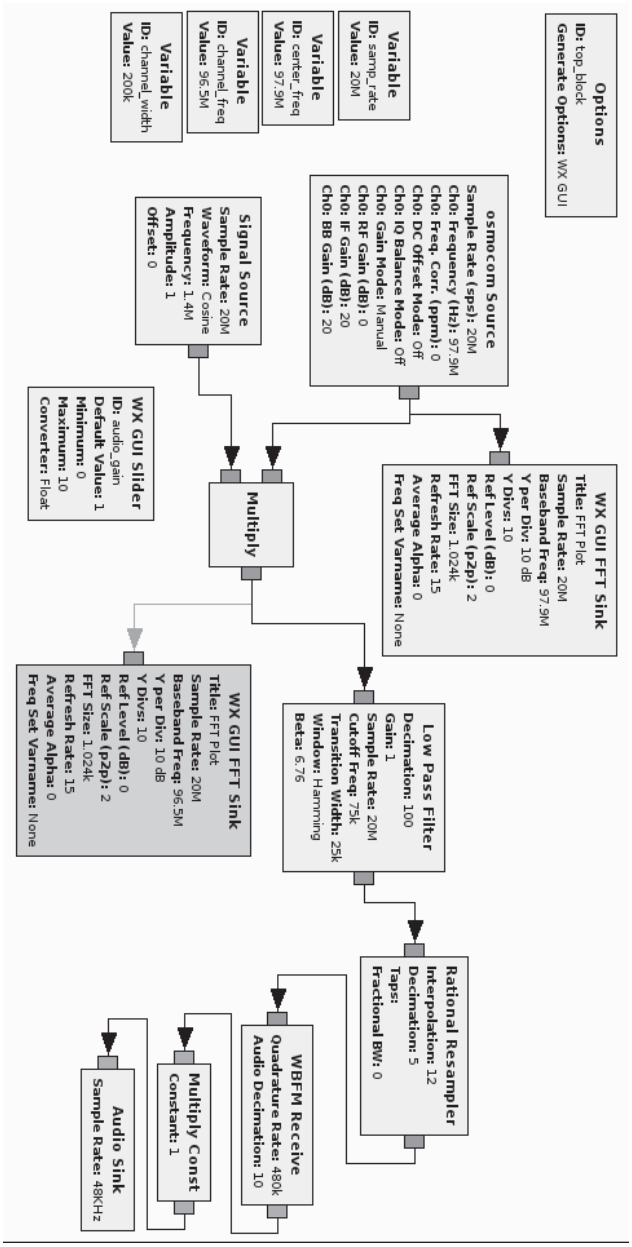


Figura 7: Ejemplo de diagrama de Bloques del entorno visual de GnuRadio

GNU Radio realiza todo el procesamiento de la señal. Puede utilizarse para escribir aplicaciones que reciban datos de flujos digitales o para enviar datos a flujos digitales, que luego se transmiten mediante hardware (SDRs). GNU Radio tiene filtros, códigos de canal, elementos de sincronización, ecualizadores, demoduladores, vocoders, decodificadores y muchos otros elementos (identificados en la aplicación como bloques de elementos, ver figura 7) que normalmente se encuentran en sistemas de radio. Más importante aún, incluye un

método para conectar estos bloques y luego gestiona cómo se pasan los datos de un bloque a otro.

Las aplicaciones de radio GNU se escriben principalmente utilizando el lenguaje de programación de Python[1], mientras que la ruta de procesamiento de señal crítica, que se suministra, se implementa en C ++[2] utilizando extensiones de punto flotante del procesador, cuando estén disponibles.

1.4 SOFTWARE OPENBTS Y RED GSM

El proyecto OpenBTS es una colección de componentes de software de código abierto que se pueden utilizar para construir una red celular con software utilizando las placas SDR con transmisores/receptores.

Las redes celulares tradicionales cuentan con numerosos elementos que en conjunto brindan una gama de servicios, pero estos componentes deben estar separados por ejemplo la conectividad de datos se realiza a través de un servidor que se denomina GGSN (Gateway GPRS Support Node) y la de voz a traves de otro denominado GMSC (Gateway Mobile Switching Center) (ver figura 8), OpenBTS permite que la interfaz de radio Um[3] de la red móvil tradicional se interconecte directamente con los protocolos de telefonía IP (ver figura 9 y cuadro 3).

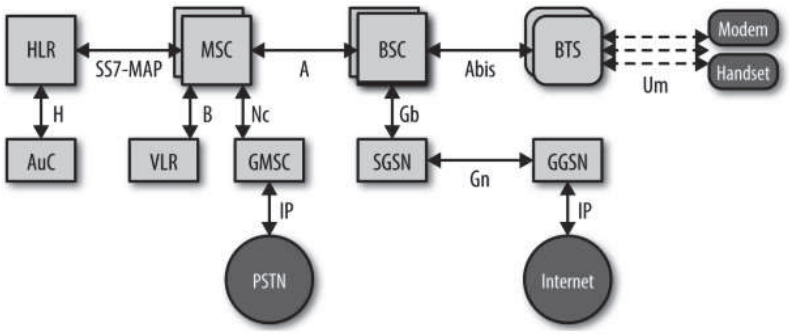


Figura 8: Esquema de una red Celular GSM tradicional. Siglas en Cuadro 3

La red tradicional GSM como se ve en la figura 8 se puede explicar mirando desde la derecha desde el handset o teléfono celular. Este se

1 Lenguaje de programación Python: https://www.python.org.
2 Lenguaje de programación C++: https://isocpp.org.
3 La interfaz Um es la interfaz de aire para el estándar de teléfono móvil GSM. Es la interfaz entre la estación móvil (MS) y la estación base de transmisión (BTS).

conecta inalámbricamente por medio de la interfaz denominada Um hacia una estación base o BTS. Esta es la encargada de recepcionar los datos recibidos del terminal, y orquestar la comunicación cuando hay multiples de estos elementos como veremos más adelante en tramas y canales lógicos.

Este se conecta por otra interfaz lógica llamada ABIS hacia un Subsistema de estación base o BSC que es el encargado de manejar la señalización de la red, los casos de handover (cuando un terminal pasa de una estación base a otra más cercana a el), y es quien separa la información a ser enviada a los demás elementos de la red discriminando la necesidad (voz, datos, registración con la red, etc.)

Este ultimo servidor se conecta por medio de la interfaz lógica denominada Gb al Serving GPRS support node o SGSN cuando el requerimiento del terminal son los datos (por ejemplo navegación por internet). Este es el encargado de mantener la información del terminal como en que celda se encuentra, de etiquetar correctamente los datos recibidos para entregar a cada terminal y de mantener la información del uso del servicio para poder ser tarifado. Una vez que se completa la información por una interfaz lógica llamada Gn se conecta al Gateway GPRS support node o GGSN que es que actúa como un router entre la red celular y una red IP como internet. Es este servidor el que interconecta ambos mundos.

1.4.0.1 *Proceso de establecimiento de una llamada de voz*

El establecimiento de una llamada de voz entre dos terminales GSM se define a continuación.

- La central telefónica local o de tránsito reconoce el código de destino nacional corresponde a una red de comunicaciones móviles y, por consiguiente, encamina la llamada hacia la pasarela GMSC más cercana
- La pasarela utiliza el número MSISDN para consultar el registro HLR (Home Location Registry) del terminal llamado, su localización, dato necesario para extender la llamada hasta el MSC donde se encuentra el terminal en ese momento. Esta informacion es especificada proporcionando el numero de itinerancia MSRN, el cual fue asignado por el VLR (Visitor Location Registry) y enviado al HLR durante la actualización de la localización del termina, proceso que ocurre continuamente.

- La llamada se extiende al msc, quien inicia el procedimiento de avisar al terminal de la existencia de una llamada destinada a el. Dicho procedimiento consiste en difundir un mensaje específico, concebido para esta función. La difusión se extiende al area de cobertura del MSC, ya que cabe la posibilidad de que no se conozca exactamente en qué subsistema de estaciones base BSS se halla el terminal. El canal a través del cual se efectúa la difusión es el canal PCH (Paging CHannel).

- La terminal responde al mensaje anterior determinando de este modo su ubicación precisa en ese instante.

- Seguidamente, se autentifica al usuario y se establecen los parámetros de cifrado de la comunicación posterior.

- El VLR comunica al MSC los parámetros necesarios para iniciar la llamada. En esta etapa, el VLR dispone de la opción de asignar al terminal un nuevo identificador temporal TMSI exclusivo para esta llamada

- El MSC envía un mensaje de solicitud de establecimiento de llamada al terminal. Este y los subsiguientes mensaje de señalización se transfieren sobre un canal dedicado SDCCH. (Stand-Alone Dedicated Control CHannel)

- El terminal confirma la recepción del mensaje y alerta al usuario de la existencia de una llamada entrante. Por otro lado, se transmite al abonado llamante una indicación de progreso de la llamada.

- Cuando finalmente el abonado llamado responde, su terminal envía un mensaje de conexión, a lo que la red responde adjudicando un canal para tráfico TCH y emitiendo el correspondiente mensaje informando de la conexión al usuario llamante. El proceso termina cuando desde la red se envia a la estación móvil una confirmación, que hace pasar la llamada al estado de activa.

En el caso de que la lamada sea hacia la red PSTN se trafica mediante la interfaz lógica NC a un Gateway Network switching subsystem o GMSC que al igual que el GGSN hace la interconexión entre la red celular y la PSTN.

SIGLA	DEFINICIÓN
A	Interfaz A
Abis	Interfaz Abis
AuC	Central de Autenticación
B	Interfaz B
BSC	Controlador de Estación Base
BTS	Estación Transductora Base
GB	Interfaz Gb
GGSN	Nodo de soporte de Gateway para GRPS
GMSC	Nodo de Gateway para red conmutada celular
Gn	Interfaz Gn
H	Interfaz H
IP	Internet Protocol
HLR	Registro de locación
MSC	Centro de conmutacion de red celular
Nc	Interfaz Nc
SGSN	Nodo de servicio de GPRS
SS7-MAP	Sistema de senalización 7
SGSN	Nodo de servicio de GPRS
Um	Interfaz de aire de red movil
VLR	Registro de servicio de locación

Cuadro 2: Definición de las siglas de una red celular

1.4.0.2 *Caso OpenBTS*

Son innecesarios cambios de software o configuración en el teléfono, ya que la interfaz de radio a la red móvil es idéntica a una red tradicional. El núcleo de la red, sin embargo, ya no se compone de una serie de protocolos y servidores complejos. Consiste en protocolos abiertos y utiliza la IP como su transporte. Ya existen muchos proyectos de software que implementan protocolos abiertos y tantos otros que hacen comunicacion VoIP como Asterisk[4]. También se desarrollaron nuevos componentes en OpenBTS para proporcionar funcionalidad, que todavía no estaban disponibles, para conectar los distintos servicios en el mundo de internet.

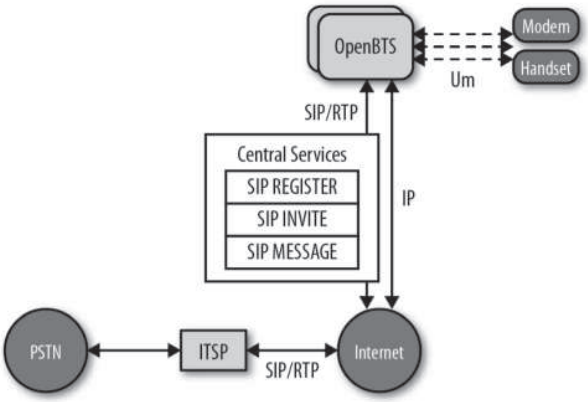


Figura 9: Esquema de una red Celular con transporte IP

Como se ve en la figura 9 el servidor OpenBTS nuclea los distintos elementos como BTS, BSC, SGSN, etc y se encarga de traducir como lo haría un GGSN los datos GPRS en IP y enviarlos a su destino como la voz en el protocolo SIP o RTP que son los que se comunican en servicios VoIP, por medio de los mensajes SIP que están especificados por el RFC (request for comments que es como la norma del protocolo y su especificación detallada) proximamente referido en la sección Software Asterisk.

1.5 TECNOLOGIA GPRS

GPRS se basa en la estructura GSM básica. Utiliza el mismo formato de señal que tiene anchos de banda de canal de 200 kHz. También tiene el mismo esquema de modulación siendo GMSK. Mantener el

4 Software VoIP Asterisk: http://www.asterisk.org

SIGLA	DEFINICIÓN
Um	Interfaz de aire de red GSM
SIP	Protocolo de iniciación de sesión
RTP	Protocolo de Transporte en Tiempo Real
DSP	Procesador digital de señales
PSTN	Red telefónica pública conmutada

Cuadro 3: Definición de las siglas de la figura 9

mismo esquema de modulación significa que se minimiza el nivel de actualización necesario para poder admitir GPRS además de GSM.

GMSK, Gaussian Minimum Shift Keying es una forma de modulación de fase que se utiliza en un sin número de elementos de comunicaciones portátiles y varias aplicaciones inalámbricas. Tiene ventajas en términos de eficiencia espectral así como una amplitud casi constante que permite el uso de amplificadores de potencia de transmisor más eficientes, ahorrando así el consumo de energia, un problema crítico para equipos a baterías.

1.5.1 *Estructura de Trama*

La interfaz inalambrica de GRPS emplea la misma estructura básica que la adoptada para GSM, que es una trama similar a la dividida en ranuras por tiempo (TDMA [5]). La estructura global de ranuras para este canal es la misma que la utilizada en GSM, que tiene el mismo perfil de potencia, y atributos de avance de sincronización para superar los diferentes tiempos de viaje de señal a la estación base dependiendo de la distancia que el móvil se encuentre. Esto permite que la ranura encaje perfectamente con la estructura GSM existente y no sea necesario cambiar nada en la estacion base o el terminal para soportarlo.

La unidad de datos del protocolo de la capa de red, denominada N-PDU o paquete, es recibida de la capa de red y es transmitida a través del interfaz de aire entre la estación móvil y el SGSN usando el protocolo LLC. Primero el protocolo encargado de la traduccion entre redes GPRS e IP denominado SNDCP transforma los paquetes en tramas denominadas logical link control o LLC caracteristicas de una red IP. El proceso incluye opcionalmente la compresión de la cabecera de datos, segmentación y encriptado. Una trama trama LLC es segmentada en bloques de datos denominados RLC, que son formados en la capa física, cada bloque consta de 4 bursts o rafaga normales que son similares a las de TDMA.

5 TDMA: https://www.motorolasolutions.com/content/dam/msi/docs/business/_documents/static_files/why_digital_white_paper_5_08.pdf

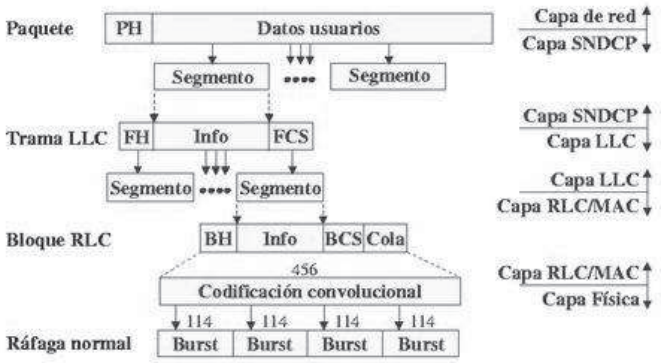


Figura 10: Esquema de transporte de datos a traves de tramas GPRS

1.5.2 *Estructura de un burst GPRS*

Cada GPRS de información es de 0.577 mS de longitud y es el mismo que el utilizado en GSM como ya mencionamos. La ráfaga o burst GPRS lleva dos bloques de 57 bits de información en línea con una ráfaga GSM, dando un total de 114 bits por ráfaga. Por lo tanto, requiere cuatro ráfagas GPRS para llevar cada 20 mS bloque de datos, es decir, 456 bits de datos codificados. Las ranuras pueden ser asignadas dinámicamente por el BSC a GPRS dependiendo de la demanda, siendo las restantes utilizadas para el tráfico GSM.

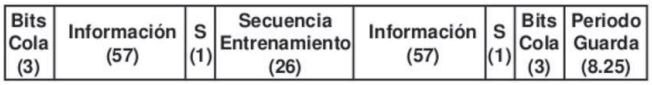


Figura 11: Estructura de un Burst GSM/GPRS normal

El BSC asigna canales logicos denominados PDCH a intervalos de tiempo particulares y habrá momentos en que el PDCH esté inactivo, permitiendo al móvil comprobar otras estaciones base y supervisar sus intensidades de señal para permitir a la red juzgar cuándo se requiere el traspaso o handover. La ranura GPRS también puede ser usada por la estación base para juzgar el retardo usando un canal lógico conocido como Canal de Control Avanzado de Tiempo de Paquetes (PTCCT).

1.6 TECNOLOGIA VOIP

Para poder implementar una red celular también debemos contar con una central telefonica digital que nos permita la administración de

las llamadas. Para ello utilizaremos un software que se basa en la tecnología de Voz sobre IP, siendo la voz digitalizada y procesada por software para ser transportada bajo el protocolo IP.

En mas detalle la Voz sobre Protocolo de Internet (Voz sobre IP, VoIP y telefonía IP) es una metodología y un grupo de tecnologías para la entrega de comunicaciones de voz y sesiones multimedia a través de redes IP[6], como Internet. Los términos telefonía por Internet, telefonía de banda ancha y servicio telefónico de banda ancha se refieren específicamente al suministro de servicios de comunicaciones (voz, fax, SMS, mensajes de voz) a través de la Internet pública, en lugar de la red telefónica pública conmutada (RTPC).

Los pasos y principios implicados en la emisión de llamadas tele-

Figura 12: Esquema de comunicación VoIP

fónicas VoIP son similares a la telefonía digital tradicional e implican la señalización, la configuración de canales, la digitalización de las señales de voz analógicas y la codificación. En lugar de transmitirse a través de una red de conmutación de circuitos; Sin embargo, la información digital se empaqueta y la transmisión se produce como paquetes IP a través de una red de conmutación de paquetes. Transportan flujos de audio utilizando protocolos de entrega de medios especiales que codifican audio y video con códecs de audio y codecs de vídeo. Existen varios códecs que optimizan el flujo de medios basado en los requisitos de la aplicación y el ancho de banda de la red; Algunas implementaciones se basan en la banda estrecha y el discurso comprimido, mientras que otros soportan códecs estéreo de alta fidelidad. Algunos codecs populares incluyen las versiones de la ley μ y de G.711[7], G.722[8], y muchos otros.

El codec G.711 codifica la voz desde la frecuencia de 50 hz a los 7 khz, utilizando un bitrate de 80 a 96 Kbps y es el mas compatible con

6 RFC de Internet Protocol: https://tools.ietf.org/html/rfc791
7 ITU G.711: https://www.itu.int/rec/T-REC-G.711-198811-I/en
8 ITU G.722: https://www.itu.int/rec/T-REC-G.722-198811-S/en

los sistemas de telefonía fija digital. El codec G.722 utiliza un bitrate de 24 a 32 Kbps y es preferido cuando se tiene un enlace con poca pérdida de tramas.

1.6.1 *Software Asterisk*

Asterisk es una implementación de software de una central telefónica privada (PBX); Permite a los teléfonos conectados hacer llamadas entre sí y conectarse a otros servicios telefónicos, como la red telefónica pública conmutada (PSTN) y los servicios de Voz sobre Protocolo de Internet (VoIP).

Asterisk es lanzado con un modelo de licencia dual, usando la Licencia Pública General GNU (GPL)[9] como una licencia de software libre[10] y una licencia de software propietaria para permitir a los licenciatarios distribuir componentes de sistemas propietarios e inéditos.

Asterisk fue creado en 1999 por Mark Spencer de Digium. Diseñado originalmente para Linux, Asterisk se ejecuta en una variedad de sistemas operativos, incluyendo NetBSD, OpenBSD, FreeBSD, macOS y Solaris, y otros sistemas enbebidos.

Hoy en día, hay más de un millón de sistemas de comunicaciones basados en Asterisk en uso, en más de 170 países. Es utilizado por casi toda la lista Fortune 1000 de clientes. Asterisk puede convertirse en la base de un sistema telefónico comercial completo o utilizarse para mejorar o ampliar un sistema existente o para superar una brecha entre sistemas.

9 Licencia GPL Version 3: https://www.gnu.org/licenses/gpl.html
10 Codigo Abierto: https://es.opensuse.org/Software_libre_y_de_código_abierto

Parte II

IMPLEMENTACION

2

IMPLEMENTACIÓN

2.1 ESQUEMA INTRODUCTORIO

Para comprender mejor las subsiguientes secciones se muestra en la figura 13 el esquema completo de la implementación.

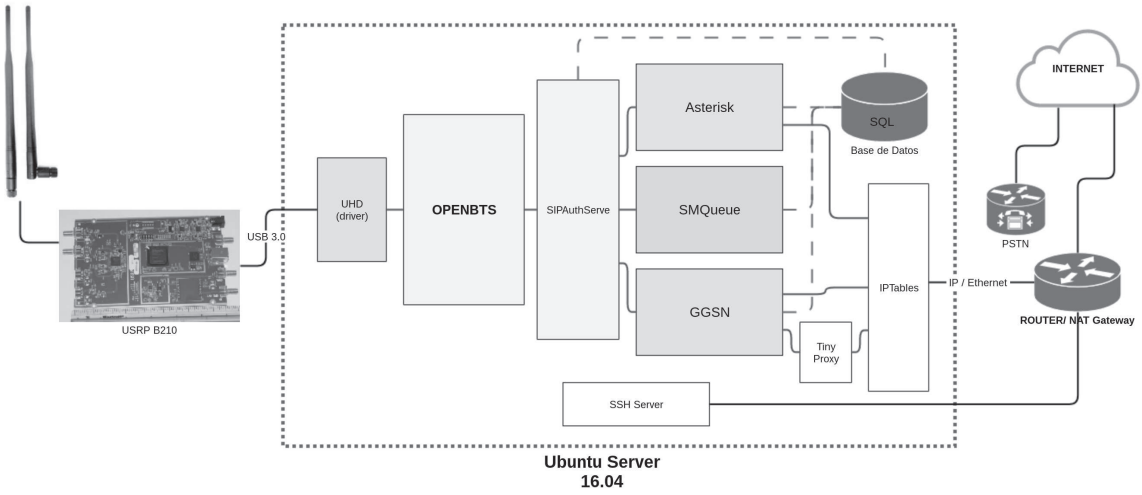


Figura 13: Esquema en bloques del sistema implementado

Se puede observar desde la izquierda que los módulos que residen en el servidor son primeramente el software del driver de la placa, en este caso el **UHD** para las placas USRP como la utilizada (modelo B210). Este driver envía la información al **OpenBTS** que paquetiza (más información en las subsiguientes secciones) lo recibido y se conecta con el **SIPAuthServe** que es el encargado de la la autenticación y manejo de sesiones como el BSC que se explica en el capitulo 1. Este también es encargado de redireccionar al igual que en una red GSM convencional al software **Asterisk** que cumple el rol del MSC para la comunicación de voz. En el caso de SMS se envía la información al **SMQueue** que procesa y responde a los mensajes recibidos o los redirige al OpenBTS si es un SMS de terminal a terminal.

En el caso de datos se envía al **GGSN** que cumple los roles de SGSN y GGSN el cual se conecta directamente con el "stack" de red del sistema GNU/Linux para ser ruteado por **IPTables** a internet o con un

paso intermedio por medio del **TinyProxy** para el caso de comunicaciones que tienen ya marcados un proxy como los terminales BlackBerry. Todos ellos se conectan a **bases SQL** donde guardan la información de abonados, tarifaciones, datos históricos de transacciones e incluso las configuraciones. El ruteo externo a internet es independiente, pero la conexión con la **PSTN** se realiza por medio de este camino con la voz paquetizada en IP.

Cuenta tambien para la administración remota con un servidor **SSH** (Secure SHell) para el acceso al terminal del sistema.

2.2 HARDWARE Y SOFTWARE UTILIZADO

2.2.1 *Servidor con GNU/Linux*

El primer requisito es una PC con GNU/Linux. Aunque varias arquitecturas de procesadores son compatibles se utilizó un procesador x86 con software para 32 bits. Esta computadora debe ser una máquina fisica separada por los requisitos mínimos para la potencia de procesamiento y la RAM. Aunque no están claramente definidas las características, las numerosas variables implicadas, como el número de señales portadoras concurrentes, la carga, tipo de uso de red, entorno de radio, etc. cada uno afectará a los recursos requeridos.

Una única señal portadora requiere que el software OpenBTS genere formas de onda de enlace descendente para transmitir al teléfono móvil y demodular las formas de onda de enlace ascendente recibidas desde el handset. OpenBTS soporta la creación de múltiples señales portadoras simultáneas en una única radio física para aumentar linealmente la capacidad de la red, pero las demandas de procesamiento son muy altos. Para una configuración de laboratorio estable con una sola señal portadora (máximo de siete canales de voz concurrentes), una Intel i5[1] o algo comparable con 2 GB de RAM es recomendado. También debe tener al menos una interfaz USB3.

En un entorno productivo pueden utilizarse múltiples señales portadoras simultáneas, lo que aumenta drásticamente el ancho de banda de muestra requerido. Además, los algoritmos usados para demodular señales puede configurarse para funcionar de una manera más

1 Intel i5 Broadwell: https://ark.intel.com/products/85212/Intel-Core-i5-5200U-Processor-3M-Cache-up-to-2_70-GHz

robusta (por ejemplo, para rectificar la distorsión de la señal debido al entorno de despliegue). Por lo tanto, la potencia de procesamiento necesaria para generar y demodular estas señales podría ser un orden de magnitud mayor que en una configuración de laboratorio.

2.2.2 *Software Defined Radio*

La radio definida por software (SDR) es el avance clave que hace OpenBTS posible desde una perspectiva de hardware. Los SDRs se han utilizado en aplicaciones militares durante unos 20 años. Sólo recientemente se han puesto a disposición de un público más amplio debido a la disminución del costo de la tecnología.

El SDR utilizado se denomina USRP B210 de la empresa laboratorio ETTUS. Sus características principales son:

- Dispositivo USRP de dos canales totalmente integrado con cobertura continua de RF de 70 MHz a 6 GHz.
- Funcionamiento dúplex completo, MIMO (2 Tx y 2 Rx) con hasta 56 MHz de ancho de banda en tiempo real (61,44MS / s en cuadratura).
- Conectividad USB 3.0 de SuperSpeed rápida.
- GNURadio y soporte de OpenBTS a través de la fuente abierta USRP Hardware Driver (UHD).
- Chip FPGA reconfigurable Spartan 6 XC6SLX150 FPGA.

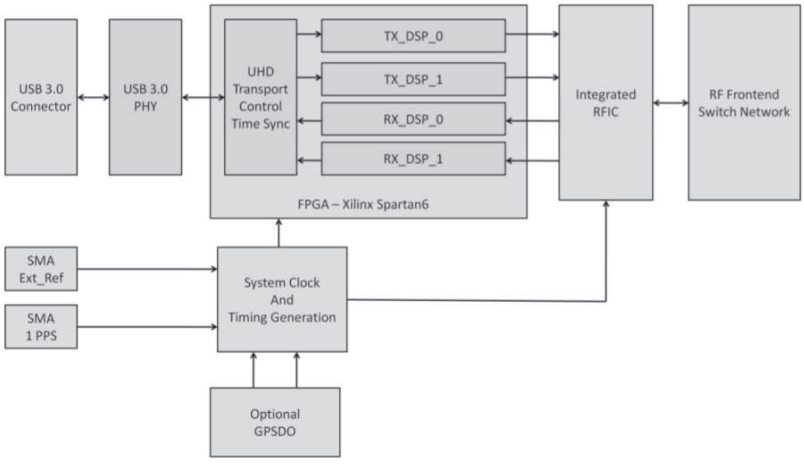


Figura 14: Esquema de arquitectura interna del SDR B210

En la figura 14 se observa la arquitectura del SDR utilizado, donde, desde la izquierda, tenemos la conexión USB 3.0 de alta velocidad y su puerto físico en el servidor. Estos datos son procesados por el driver UHD el cual divide en colas las dos salidas Tx y la información de las entradas en Rx, todo procesado por el FPGA. Esta informacion es procesada en paralelo por el circuito integrado de radiofrecuencia o RFIC que referencia un conjunto de elementos para el procesamiento analogico saliendo luego a una interfaz de radio. También cuenta con un reloj integrado que se le puede ser anexado un reloj externo para más precisión, como así también un valor de referencia de frecuencia externo.

El hardware B210 cubre frecuencias RF de 70MHz a 6 GHz, tiene una Spartan6 FPGA y conectividad USB 3.0. Esta plataforma permite la experimentación con una amplia gama de señales incluyendo FM y difusión de TV, celular, Wi-Fi, y más. El USRP B210 ofrece un total de dos canales de recepción y dos de transmisión, un GPIO, e incluye una fuente de alimentación externa. Utiliza un Analog Devices RFIC para ofrecer una plataforma de experimentación de RF rentable y pueden transmitir hasta 56 MHz de ancho de banda instantáneo a través de un bus USB 3.0 de alto ancho de banda.

2.2.3 *Antenas*

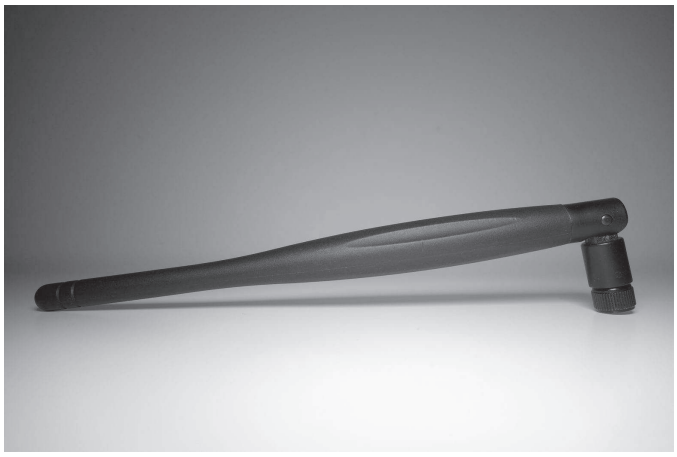


Figura 15: Antena de uso interno para frecuencias de GSM

Muchos SDRs tienen suficiente sensibilidad de transmisión y recepción para operar sin antenas en un entorno pequeño. Típicamente, se puede conseguir un área de cobertura con un radio de 1 m. Esta es una configuración deseable para un entorno de laboratorio. Las áreas

de cobertura no se superponen e interfieren entre sí. Además, la red no interferirá con ningún operador en el área.

Incluso si el área de cobertura es muy pequeña, el regulador nacional muy probablemente requiere que se obtenga una licencia de prueba antes de usar frecuencias GSM. La adición de un par de pequeñas antenas de 5 dBi puede aumentar considerablemente, hasta un radio de 25 m en un ambiente sin obstrucciones. Estas son por lo general antenas de goma estilo pato con un conector SubMiniature versión A (SMA), similar en apariencia a una típica antena de enrutador WiFi doméstico. Un ejemplo se muestra en la Figura 15.

Las antenas se instalan para una frecuencia específica, así que se elije una que más se aproxima a la banda GSM que va a utilizar (850, 900, 1800 o 1900 MHz). La frecuencia también juega un papel en el tamaño del área de cobertura. Las bandas de baja frecuencia (850 y 900 MHz) propagan distancias mayores que las bandas de alta frecuencia, a veces casi el doble.

2.2.4 *Telefonos de prueba y SIMs*

El SIM utilizado en los teléfonos GSM no es más que una tarjeta inteligente recortada. Las tarjetas inteligentes también se conocen como tarjetas de chip o tarjetas de circuitos integrados (ICC) y se pueden encontrar en muchas aplicaciones de autenticación e identificación. Una tarjeta inteligente de tamaño completo tiene las mismas dimensiones que una tarjeta de crédito; Las tarjetas de crédito más nuevas utilizan la tecnología de tarjetas inteligentes para aumentar la seguridad. Las tarjetas SIM se programan utilizando escritores estándar de tarjetas inteligentes y, una vez escritas, se salen del marco de la tarjeta de tamaño completo para que encajen en un teléfono GSM.

Para fines de prueba, cualquier tarjeta SIM que se adapte físicamente en el terminal probablemente funcionará. Sin embargo, el uso de una SIM existente, una tarjeta SIM expirada o SIM de un país extranjero tiene concesiones, principalmente en relación con las características de seguridad. Hay una clave secreta, llamada Ki, almacenada en el SIM, que sólo el emisor de la SIM sabe (generalmente el proveedor de telefonía celular junto al fabricante). Se almacena en una parte ilegible de la memoria en la tarjeta SIM y no es recuperable después de que el fabricante lo coloca allí. Este secreto compartido entre el ope-

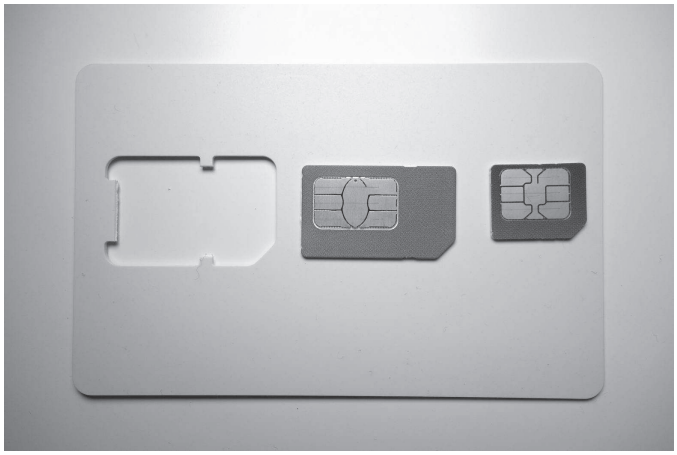


Figura 16: Sims de Prueba

rador y el teléfono es lo que permite a la red determinar otra clave, denominada Kc, que se utiliza para dar soporte al cifrado de llamadas. Al no contar con estos datos esta característica de seguridad se perdera. La otra opción es utilizar un grabador de tarjetas inteligentes en el cual podemos insertar nuestro Ki personalizado sobre una tarjeta virgen con el cual se podra calcular el Kc (Figura 17) en ambos lados.

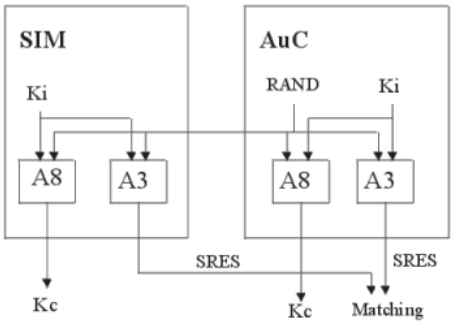


Figura 17: Calculo de Kc en terminal y Estacion base

Para este proyecto se utilizó SIMCards de proveedores comerciales de distintas empresas los cuales se los registra en el SIPAuthServe y conectan a la red como si fuera una conexión de una red externa (roaming).

2.3 SOFTWARE UTILIZADO EN EL SERVIDOR

Se optó por una distribución de GNU/Linux[2] denominada UBUNTU[3] en su versión LTS Server 16.04 para la instalación del OS del

2 GNU/Linux: https://en.wikipedia.org/wiki/GNU/Linux_naming_controversy
3 Ubuntu GNU/Linux: https://www.ubuntu.com

servidor en donde va a estar el software de manejo de la placa SDR y el procesamiento de los datos. Estos datos van a ser tratados luego por el software OpenBTS que fue compilado en el mismo servidor en base al código fuente.

En una primera instancia se instaló el Servidor VoIP Asterisk dentro del mismo servidor y se lo conectó a la red IP por medio de su interfaz gigabit ethernet. El mismo procesa los requerimientos de llamada y los reenvía al gateway de red telefónica IP brindado para la salida externa.

2.4 INSTALACIÓN OS Y MODULOS PRINCIPALES

2.4.1 *Instalación UBUNTU*

El sistema operativo fue instalado siguiendo un proceso de automatización. Esto permite que varias copias del mismo sistema puedan ser recreadas de forma rápida con mínima intervención humana. Esa opción además permite una fácil actualización de las subsiguientes versiones de software. El sistema para la automatización se llama ansible[4]. Ansible corre por conexiones SSH[5] ejecutando scripts de Python[6] que a su vez ejecutan y corroboran ejecutables del sistema.

El programa Ansible para la instalación se publicó al sistema ansible-galaxy bajo la licencia GPL versión 3[7]. El sistema se divide en instalación de paquetes, configuración de los mismo, descarga del código fuente de OpenBTS, compilación de OpenBTS y asterisk.

2.4.2 *Compilacion OpenBTS e instalación de bibliotecas requeridas*

Para la construccion del entorno es necesario contar con algunas herramientas basicas para la compilacion de paquetes, un gestor de GIT (sistema de versionamiento de codigo) y algunas herramientas especificas, en este caso para ubuntu, para manejar la instalacion de los paquetes previamente compilados (APT[8].).

4 Ansible: https://www.ansible.com
5 SSH: https://linux.die.net/man/1/ssh
6 Python: https://www.python.org
7 GPL V.3: https://www.gnu.org/licenses/gpl-3.0.en.html
8 Gestor de paquetes APT: https://help.ubuntu.com/lts/serverguide/apt.html

Las bibliotecas utilizadas para la compilacion son:

- autoconf
- libtool
- libosip2
- libortp
- libusb-1.0
- g++
- sqlite3
- libsqlite3-dev
- libreadline6-dev
- libncurses5-dev

El paquete de servicios que se instala se puede visualizar en la siguiente figura

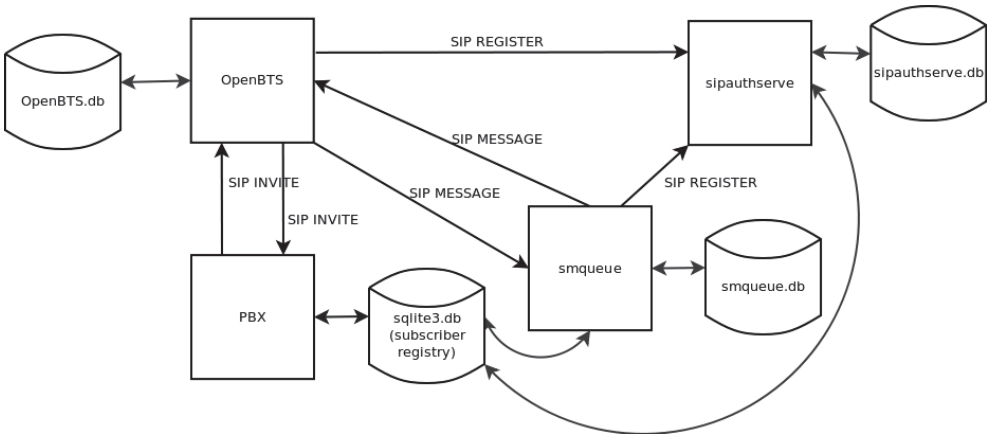


Figura 18: Esquema de servicios en OpenBTS

Los elementos del sistema se van a dividir principalmente en el **Transreceiver**, que es el software que va a programar la capa 1 de radio, implementandose sobre el hardware SDR, en este caso con los detalles necesarios para la placa de Ettus Research. Este módulo es controlado por OpenBTS.

El software **OpenBTS** corre la implementación del TDMA en capa 1 y luego todos los demás elementos de la capa 2, 3 y la comunicación necesaria con la 4. El mismo también controla las llamadas de usuario para definir configuraciones.

La **PBX** que en este caso se utiliza Asterisk, tiene todos los componentes de un servidor VoIP y la conexión a POTS. El mismo tiene una

base de datos conectada con los datos de los subscriptores habilitados.

El servidor **SIPAuthServe** es el encargado de la registración y autorización SIP, quien maneja la actualizacion de ubicación de los terminales que llegan del OpenBTS para actualizar los datos en ambas bases de datos.

El servidor de mensajes de texto SMS el **SmQueue** es un servicio adicional que no es totalmente necesario pero complementa el servicio de telefonía. El mismo tiene conexión a una base propia para manejar colas de mensaje, historiales y una conexión a la base de subscriptores.

El servidor de Datos **GPRS** se controla dentro de OpenBTS pero es separado en el caso de UMTS donde es llamado **OpenBTS-UMTS** y se conecta a al SIPAuthServer para controlar el acceso de los subscriptores. Ambos sistemas, GPRS y UMTS utilizan el módulo de red de GNU/Linux para derivar los paquetes IP. La herramienta utilizada para el ruteo de los mismos es IPTables.

2.4.3 *Instalación Modulos OpenBTS*

Para la instalación se tomó el codigo fuente de OpenBTS desde github[9] el cual se compilo con el script armado en el mismo denominado build.sh para la arquitectura B210 que es la placa utilizada.

La compilación y el armado de los .deb se hizo modulo por modulo, primero el de la librería a53, para la encriptación de la comunicación en caso de que se cuente con la Kc, luego el subscriber Registry que consta de la API para cargar los suscritos a la red y el sipathserve, el smqueue que es un SMS gateway , el cliente de asterisk con algunas configuraciones pre-armadas para openbts y por ultimo el modulo propiamente openbts con el transreceiver y las herramientas de interfaz con el administrador.

Los Archivos generados fueron los siguientes:

```
root@openbts:/home/pbullian/dev/BUILDS/installed# ls | grep .deb
liba53_0.1_i386.deb
libcoredumper1_1.2.1-1_i386.deb
libcoredumper-dev_1.2.1-1_i386.deb
```

9 Repositorio de OpenBTS dev: https://github.com/RangeNetworks/dev

```
openbts_5.0_i386.deb
range-asterisk_11.7.0.5_i386.deb
range-asterisk-config_5.0_all.deb
range-configs_5.1-master_all.deb
SIPAuthServe_5.0_i386.deb
smqueue_5.0_i386.deb
```

Los cuales se instalaron junto algunos servicios adicionales.

Uno de ellos es bind9[10], una implementación de servidor DNS para poder brindar una resolución de nombres local a la red. Otro paquete es tinyproxy[11], el cual se utiliza com proxy para aquellos celulares que tengan configurado el mismo en el APN y no pueda ser cambiado como el caso de los blackberry. Se instalaron el set de herramientas y driver de ettus-research uhd, el cual se utiliza para cargar el código de la b210 al FPGA de la placa y hacer algunas pruebas de transmisión y recepción. Se instalo la herramienta kalibrate [12] ya que al no contar con un reloj externo se necesita tener una idea de la desviación para ajustarla en el transreceiver. Esta es una herramienta que permite obtener diferencias tomando como referencia antenas de GSM de alguna empresa de telefonía celular. Algunas librerías adicionales como zeroqm3 y libcoredumper[13] de google fueron instaladas también para poder compilar algunas herramientas de openbts.

En la Figura 19 podemos observar como queda el esquema completo de conexionado entre los modulos. En el se visualiza la conexión de radio Um contra el hardware de radio USRP, el cual por el usb pasa la informacion para ser procesada por el software de transreceiver. Esta informacion se paquetiza en UDP (protocolo de capa 4 de red montado sobre IP) para ser tratada por OpenBTS y hacer el direccionamiento a cada módulo necesario, como el SIPAuthServe para la registración y control, el SMQueue para el manejo de SMS o el Asterisk para el establecimiento de llamadas de voz.

10 Documentacion Bind9: https://wiki.debian.org/Bind9
11 Documentacion Tinyproxy: https://tinyproxy.github.io/
12 Repositorio de Kalibrate: https://github.com/ttsou/kalibrate
13 Repositorio de paquetes precompilados de libcoredumper: https://code.google.com/archive/p/google-coredumper/downloads

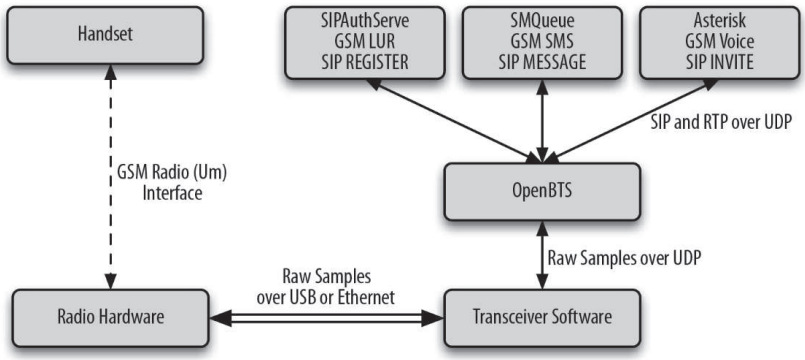


Figura 19: Arquitectura de conexión de los modulos

2.5 PRUEBAS DE FUNCIONAMIENTO

En la presente sección se harán las pruebas para corroborar el correcto funcionamiento de la placa, la conexión contra el driver UHD y la recepcion de informacion. También se prueba la ganancia de salida del transmisor para lograr que no se genere una retroalimentación en el receptor que sature y genere ruido a la señal recibida. Por último se hacen pruebas del SNR recibido con transmisiones de prueba.

2.5.1 *Placa SDR*

Para las primeras pruebas se utiliza las herramientas provistas por Ettus-Research. En una primera instancia utilizamos **uhd_images_downloader** el cual detecta la placa conectada por USB y descarga de sus servidores la imagen a copiar en el FPGA de la placa. Para enviar la imagen descargada se utiliza **uhd_image_loader** o el mismo openbts lo hace al iniciar el transreceiver.

La prueba de conexion y propiedades de la placa se corre con **uhd_usrp_probe**, que da la siguiente información.

```
root@openbts:/home/pbullian/dev/BUILDS/installed# uhd\_usrp\
    _probe
linux; GNU C++ version 5.3.1 20151219; Boost\_105800; UHD\_003
    .009.002-0-unknown

-- Detected Device: B210
-- Operating over USB 3.
-- Initialize CODEC control...
-- Initialize Radio control...
-- Performing register loopback test... pass
```

```
-- Performing register loopback test... pass
-- Performing CODEC loopback test... pass
-- Performing CODEC loopback test... pass
-- Asking for clock rate 16.000000 MHz...
-- Actually got clock rate 16.000000 MHz.
-- Performing timer loopback test... pass
-- Performing timer loopback test... pass
-- Setting master clock rate selection to 'automatic'.
  ____________________________________________________
  /
  |       Device: B-Series Device
  |     ______________________________________________
  |    /
  |   |       Mboard: B210
  |   |   revision: 4
  |   |   product: 2
  |   |   serial: 3111809
  |   |   name: MyB210
  |   |   FW Version: 8.0
  |   |   FPGA Version: 13.0
  |   |
  |   |   Time sources: none, internal, external, gpsdo
  |   |   Clock sources: internal, external, gpsdo
  |   |   Sensors: ref_locked
  |   |
      ____________________________________________________
  |   |    /
  |   |   |       RX DSP: 0
  |   |   |   Freq range: -8.000 to 8.000 MHz
  |   |
      ____________________________________________________
  |   |    /
  |   |   |       RX DSP: 1
  |   |   |   Freq range: -8.000 to 8.000 MHz
  |   |
      ____________________________________________________
  |   |    /
  |   |   |       RX Dboard: A
  |   |   |
      ____________________________________________________
  |   |   |    /
  |   |   |   |       RX Frontend: A
  |   |   |   |   Name: FE-RX2
  |   |   |   |   Antennas: TX/RX, RX2
  |   |   |   |   Sensors: temp, rssi, lo_locked
  |   |   |   |   Freq range: 50.000 to 6000.000 MHz
```

```
|   |   |   |   Gain range PGA: 0.0 to 76.0 step 1.0 dB
|   |   |   |   Bandwidth range: 200000.0 to 56000000.0 step
    0.0 Hz
|   |   |   |   Connection Type: IQ
|   |   |   |   Uses LO offset: No
|   |   |
    ----------------------------------------------------
|   |   |    /
|   |   |   |       RX Frontend: B
|   |   |   |   Name: FE-RX1
|   |   |   |   Antennas: TX/RX, RX2
|   |   |   |   Sensors: temp, rssi, lo_locked
|   |   |   |   Freq range: 50.000 to 6000.000 MHz
|   |   |   |   Gain range PGA: 0.0 to 76.0 step 1.0 dB
|   |   |   |   Bandwidth range: 200000.0 to 56000000.0 step
    0.0 Hz
|   |   |   |   Connection Type: IQ
|   |   |   |   Uses LO offset: No
|   |   |
    ----------------------------------------------------
|   |   |    /
|   |   |   |       RX Codec: A
|   |   |   |   Name: B210 RX dual ADC
|   |   |   |   Gain Elements: None
|   |
    ----------------------------------------------------
|   |    /
|   |   |       TX DSP: 0
|   |   |   Freq range: -8.000 to 8.000 MHz
|   |
    ----------------------------------------------------
|   |    /
|   |   |       TX DSP: 1
|   |   |   Freq range: -8.000 to 8.000 MHz
|   |
    ----------------------------------------------------
|   |    /
|   |   |       TX Dboard: A
|   |   |
    ----------------------------------------------------
|   |   |    /
|   |   |   |       TX Frontend: A
|   |   |   |   Name: FE-TX2
|   |   |   |   Antennas: TX/RX
|   |   |   |   Sensors: temp, lo_locked
|   |   |   |   Freq range: 50.000 to 6000.000 MHz
```

```
|   |   |   |   Gain range PGA: 0.0 to 89.8 step 0.2 dB
|   |   |   |   Bandwidth range: 200000.0 to 56000000.0 step
    0.0 Hz
|   |   |   |   Connection Type: IQ
|   |   |   |   Uses LO offset: No
|   |   |
    ----------------------------------------------------
|   |   |    /
|   |   |   |       TX Frontend: B
|   |   |   |   Name: FE-TX1
|   |   |   |   Antennas: TX/RX
|   |   |   |   Sensors: temp, lo_locked
|   |   |   |   Freq range: 50.000 to 6000.000 MHz
|   |   |   |   Gain range PGA: 0.0 to 89.8 step 0.2 dB
|   |   |   |   Bandwidth range: 200000.0 to 56000000.0 step
    0.0 Hz
|   |   |   |   Connection Type: IQ
|   |   |   |   Uses LO offset: No
|   |   |
    ----------------------------------------------------
|   |   |    /
|   |   |   |       TX Codec: A
|   |   |   |   Name: B210 TX dual DAC
|   |   |   |   Gain Elements: None
```

En nuestro laboratorio vamos a utilizar el Frontend A.

Para las pruebas de transmisión podemos utilizar **benchmark_rate –rx_rate 10e6 –tx_rate 10e6**, el cual transmite y recibe en la misma frecuencia y nos devuelve los resultados de ambos.

Tambien tenemos una prueba mas visual de receptor con **rx_ascii_art_dft –freq 930e6 –rate 5e6 –gain 20 –bw 5e6 –ref-lvl -30**[14], como se ve en la figura 21

2.6 CONFIGURACIÓN DEL SERVIDOR E INICIALIZACIÓN

Cuando se pone en funcionamiento un servidor, ya sea para correr estos programas o un servidor WEB/SMTP, etc, es necesario controlar y administrar como actúa ante los reinicios del sistema y en el caso de ser necesario cuál es la priorización de inicio de un servicio sobre otro, como por ejemplo no podemos inicializar el servicio Asterisk

14 Herramientas de Prueba UHD: https://kb.ettus.com/Verifying_the_Operation_of_the_USRP_Using_UHD_and_GNU_Radio

Figura 20: Laboratorio con elementos de medición

sin antes contar con una conexión a internet para el acceso a la PSTN. Para ello hay herramientas en GNU/Linux que pueden ser utilizadas.

2.6.1 *Inicio de los modulos*

El sistema que controla hoy en dia el inicio y orquestación de demonios en linux es llamado **systemd**[15], quien reemplazó al antiguo programa llamado **sysv**[16]. Openbts ya tiene configuraciones para iniciar con **upstart**[17], un sistema muy similar a sysv que se utilizaba en ubuntu. Al ser simple para manejar los módulos y los pocos servicios adicionales se reemplaza por el mismo instalando el paquete **upstart-sysv**. Para el inicio de los módulos como primer paso recomiendo iniciar el asterisk y corroborar su correcta ejecucion, para luego iniciar los módulos de smqueue, y SIPAuthServe los cuales si también inician correctamente puede llevarse a cabo la ejecución de openbts.

15 Systemd: https://wiki.debian.org/systemd
16 SysV: http://glennastory.net/boot/init.html
17 Upstart: http://upstart.ubuntu.com/

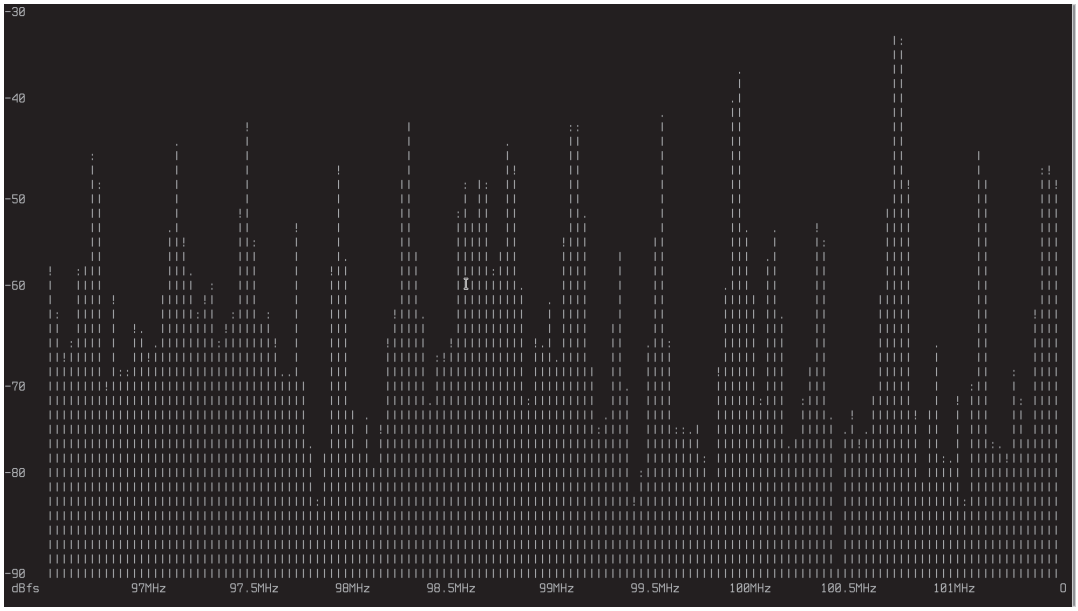


Figura 21: Arte ascii como interfaz de usuario

Upstart utiliza el prefijo start o stop seguido del nombre del servicio para iniciar o detenerlos. Los logs los nuclea en **/var/log/upstart/nombre_del_servicio.log**.

Luego de un inicio correcto del transreceiver podemos visualizar los leds de la salida de antena de Tx y Rx y podemos corroborarlo si sabemos el canal y frecuencia utilizada (ARFCN), con otro SDR o un analizador de espectro. En la figura22 se visualiza una captura del espectro tomado con el software GNURadio y un sdr HackRF en una frecuencia 945.2 Mhz que es el Uplink del **ARFCN 51**[18] el cual estaba configurado el OpenBTS.

2.6.2 *Configuraciones Principales OpenBTS*

Para poder utilizar la placa USRP es necesario en base a las pruebas previamente realizadas establecer la ganancia de salida que OpenBTS va a necesitar informar al driver UHD, así también como algunas configuraciones de qué frecuencia utilizar y como tratar algunas señales de radio.

Algunas configuraciones básicas del sistema se acceden desde la interfaz OpenBTSCLI, la cual se encuentra dentro de la carpeta dev/-NodeManager. Al ejecutar el comando entramos a la configuración del OpenBTS.

18 Tabla de ARFCN: http://niviuk.free.fr/gsm_arfcn.php

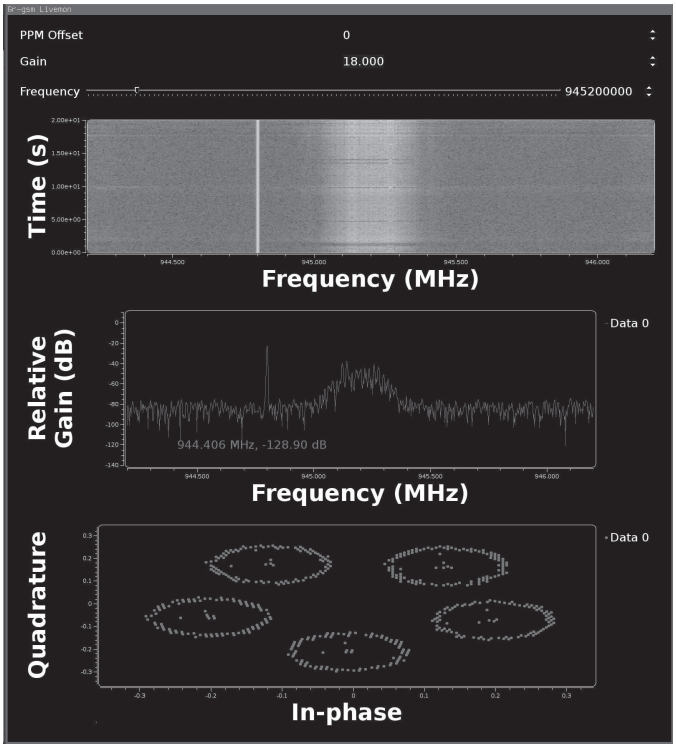


Figura 22: Captura con GNURadio del ARFCN 51 con visualizacion cascada, espectro y de constelación

Primero luego de calibrar la placa es necesario setear la ganancia de la misma, por defecto el valor es muy alto para la placa B210 y genera que se sature la entrada por Rx así que se disminuye a 10 Db

```
OpenBTS> devconfig GSM.Radio.RxGain 10
```

Luego seteamos un nombre de fantasía para la radio base, la cual figurara en los terminales una vez que se registren, con

```
OpenBTS> config GSM.Identity.ShortName UNSAMnet
```

Podemos también configurar algunos mensajes SMS que llegan al móvil dependiendo de la modalidad de registración que utilizamos, siguiendo el mismo criterio podemos listar estas configuraciones con

```
OpenBTS> config Registration.Message
```

La cual nos dara una lista para modificar a gusto.

Luego vamos a setear algunos datos como el MCC y el MNC[19] de la estación con el primero MCC en 722 que es el codigo para Argentina, y el MNC tiene que ser uno distinto a los MNC de los proveedores de telefonía celular en el area, ya que si no los terminales se conectan a la misma confundiendolo con la red de su SIM.

19 MCC y MNC: http://mcc-mnc.com/

```
OpenBTS> config GSM.Identity.MCC 722
OpenBTS> config GSM.Identity.MNC 001
```

Continuando seleccionaremos el rango de frecuencia a utilizar, por ejemplo GSM900

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

Y seteamos el ARFCN a un valor que no sea igual a otras redes, por ejemplo 51

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

En las subsiguientes secciones se configuraran los datos necesarios para una conexión por GPRS.

2.6.3 *Configuracion para el ruteo de llamadas en Asterisk*

Como se explico en el capítulo 1 es necesario contar con un MSC que dirija el establecimiento de llamadas y soporte la conexión y el direccionamiento hacia la PSTN en caso de que se quiera salir fuera de la red celular. El servicio de ASterisk también consulta la base de datos SQL para obtener información de los terminales en la red, su MSISDN e informa y controla la tarifación en las llamadas.

La configuración de Asterisk respecto a la integración con OpenBTS se resuelve con la configuracion de extensiones. Los datos en tiempo real de los usuarios conectados y sus msisdn asignados se tomas de la base de datos. La base de datos es sqlite3[20] . Podemos ver la configuración con las consultas pertinentes

```
[phones](HangupCause) ;define un contexto
exten => _[*#+0-9]!,    1,Set(CDR(B-IMSI)=\${ODBC_SQL(select dial
     from dialdata_table where exten=\"\${CDR(B-Number)}\")})

same => n,GotoIf(\${EXISTS(\${CDR(B-IMSI)})}?B-IPAddr)  ;el
    numero es conocido

same => n,GoSub(to-e164,\${CDR(B-Number)},1)     ;pasarlo a e164
    y volver a probar

same => n,Set(CDR(B-Number)=\${GOSUB_RETVAL})       ;update B-
    Number
```

20 Base de Datos SQLite: https://www.sqlite.org/

```
same => n,Set(CDR(B-IMSI)=\${ODBC_SQL(select dial from dialdata_
    table where exten=\"\${CDR(B-Number)}\")})

same => n,GotoIf(\$["\${CDR(B-IMSI)}"=""]?to-pstn,\${CDR(B-Number
    )},1)             ;No es un numero en la celda, envia el
    pedido a la PSTN

same => n(B-IPAddr),Set(CDR(B-IPAddr)=\${ODBC_SQL(select ipaddr
    from sip_buddies where username=\"\${CDR(B-IMSI)}\")})

same => n,ExecIf(\$["\${CDR(B-IPAddr)}"=""]?Set(CDR(B-IPAddr)
    ="127.0.0.1"))       ;setea el puerto

same => n,Set(CDR(B-Port)=\${ODBC_SQL(select port from sip_
    buddies where username=\"\${CDR(B-IMSI)}\")})

same => n,ExecIf(\$["\${CDR(B-Port)}"=""]?Set(CDR(B-Port)=5062))
     ;setea el puerto

same => n,GoSub(to-openBTS,\${CDR(B-Number)},1(\${CDR(B-IMSI)
    },\${CDR(B-IPAddr)},\${CDR(B-Port)}))

same => n,Goto(h-\${HANGUPCAUSE},1)
```

2.6.4 *Salida de llamadas fuera de la red celular*

Para la salida a la PSTN se debe configurar un Trunk SIP para derivar las llamadas. En este caso podemos ver la configuración de un servicio accesible por internet.

```
[onsip]
type=peer
host=sip.onsip.com ; direccion del servidor
username=unsam ;usuario del trunk
fromuser=pablo ;usuario de salida en el trunk
fromdomain=unsam.onsip.com ;dominio del origen
secret=password
dtmfmode=rfc2833 ;modo de dtmf
context=incoming-context
insecure=invite
srvlookup=yes
```

Para ello se debe llevar a cabo una registración, que se configura dependiendo del trunk. En este caso el pedido es el siguiente

```
register => pablo@unsam.onsip.com:password:unsam@sip.onsip.com
```

Para las comunicaciones hay dos datos necesarios, el From y la cabecera del Invite[21] para inicializar un pedido.

```
exten => _X.,n,Set(CALLERID(name)=15135555555) ;from
exten => _X.,n,Dial(SIP/000\${EXTEN}@onsip) ;cabecera del invite
```

Para probar que el registro sea exitoso podemos hacerlo desde el asterisk CLI[22] , que se ejecuta con asterisk -r y corriendo el comando como se ve en la figura 23

Figura 23: Registro de SIP exitoso

Vemos que la registración es exitosa, luego podemos tratar de establecer llamadas y podemos ver los registros CDR[23]

```
"","IMSI722010003412733","i","to-pstn","IMSI722010003412733","SIP
    /00101100010-00000010","SIP/onsip-00000011","","","2017-08-01
     22:53:32","2017-08-01 22:53:33","2017-08-01
    22:54:14","42","41","ANSWERED","DOCUMENTATION
    ","1501638812.16","","A","","","1511111111","*98","IMSI
    722010003412733",""
```

Donde se ve que se ruteo al contexto “to-pstn” desde el msisdn que se registro 151111111 al numero *98 que es un sistema de voicemail externo.

2.7 CONFIGURACION DE RED IP PARA GPRS

GPRS es un servicio de datos (2.5G) que soporta teóricamente hasta 30kb/s con el esquema adecuado y los recursos asignados necesarios. Para la configuración de GPRS primero es necesario activar el segmento que va a levantar el GGSN y el SGSN simulados dentro de openbts.

21 RFC 3261 SIP: https://tools.ietf.org/html/rfc3261
22 Asterisk CLI: https://www.voip-info.org/wiki-Asterisk+CLI
23 Registros CDR: http://www.asteriskdocs.org/en/3rd_Edition/asterisk-book-html-chunk/asterisk-SysAdmin-SECT-1.html

```
OpenBTS> config GPRS.Enable 1
```

Luego vamos a elegir un rango de Ips para asignar a los terminales, esta red va a ser enmascarada (nateo, que es el enmascaramiento de redes privadas hacia las redes publicas por una direccion de la misma) con IPTables, y al ser una nueva interfaz que se va a crear necesitamos que sea un rango de IP distinto para no generar problemas en el ruteo interno. Par ejemplo una red de 254 terminales (clase A subneteada, pero para ruteo en LAN[24]) 10.10.10.0/24

```
OpenBTS> config GGSN.MS.IP.Base 10.10.10.1
OpenBTS> Config GGSN.MS.IP.MaxCount 254
```

También especificamos un DNS externo publico como el de google por ejemplo con

```
OpenBTS> config GGSN.DNS 8.8.8.8
```

El ruteo es manejado por iptables. Como se puede ver en la figura 24 se ponen la table filter en aceptar todo y luego en la tabla nat sobre la política de POSTROUTING se agrega un filtro MASQUERADE para que tome todo lo que entre a dicha regla y haga un NAT por la interfaz seleccionada. (ver anexo 5.2)

Como se ve en la figura 24, IPtables cuenta con muchas tablas diferentes que procesan el paquete IP recibido o a enviar. Se divide en grandes razgos como PREROUTING (antes de ser ruteada o direccionada), INPUT (aquellos paquetes entrantes y el tratamiento que se le da), FORWARD (aquellos paquetes que deben ser redireccionados a otro lado, OUTPUT (para paquetes que saldrán del equipo hacia el exterior y fueron generados dentro del mismo) y POSTROUTING (para todas aquellas reglas a aplicar antes de la salida por la interfaz de red).

Para el manejo de aquellos celulares con proxy se armo un script (ver anexo 5.1) para generar una interfaz “dummy” con la ip del proxy que se consulta por el celular. Esto se puede ver haciendo un tcpdump del tráfico por la interfaz o activando el logging de GPRS a debug. Luego se configura el tinyproxy para escuchar por todas las interfaces y rutear el tráfico por la salida a internet (ver anexo 5.2)

24 RFC Address Allocation for Private Internets: https://tools.ietf.org/html/rfc1918

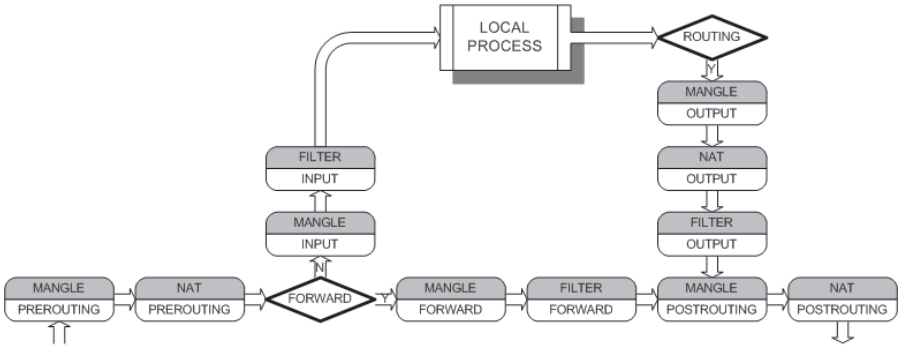


Figura 24: Esquema de ruteo de IPTables

2.8 CONFIGURACIÓN DE LOS TERMINALES

La configuración de los terminales (MS) es bastante simple. Aquellos terminales que tengan una SIM de un operador comercial pueden conectarse a la red, pero por la configuración de la SIM y su red de operación nativa, la red de OpenBTS figurara como Roaming. Con solo buscar la red el terminal debería listar la misma como se ve en la figura 25. Depende de como tenga la configuración en la SIM puede ser que los nombres que muestra varien, en este caso muestra el MCC y MNC juntos como identificador ya que en su lista no existe dicha combinacion.

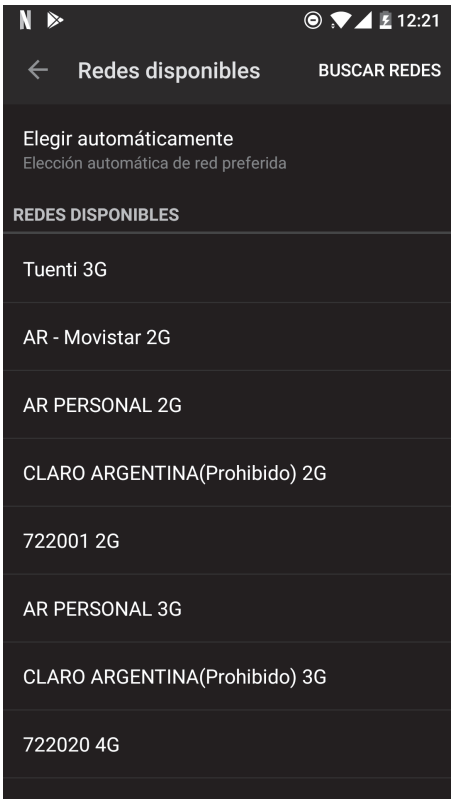


Figura 25: Busqueda de redes mobiles con una SIM de operador comercial

La primera interacción del terminal con la red ejecuta un LUR[25] (location update request) cuando seleccionamos la red en la pantalla como se ve en la figura 26. En ese momento en la estación base esa petición es interpretada y se envía al SIPAuthServe para ser tratada. El SIPAuthServe busca en su base de datos si existe la información del IMSI que solicita la registración, si no existe rechazara la registración, pero va a quedar registrado en una tabla de intercambios de información que se denomina TMSI[26] (formalmente Temporary mobile subscriber identification aunque no se le asigne uno en este momento.), como se ve en la figura 27. Esta tabla indica en la columna Auth si existe o no en la base de datos.

Figura 26: Registración con una SIM comercial

Otra opción es que la red sea de registración abierta, sin un previo chequeo. Esto es configurable en el SIPAuthServe y podemos utilizar RegEx para filtrar por ejemplo por SIM de cierto operador, o de cierto país, etc..

25 Location Update Request: http://www.eventhelix.com/RealtimeMantra/Telecom/GSM_Location_Update_Sequence_Diagram.pdf

26 Temporary Mobile Subscriber Identity: http://www.efort.com/r_tutoriels/GSM1_EFORT.pdf

```
OpenBTS> tmsis
IMSI            TMSI IMEI            AUTH CREATED ACCESSED TMSI_ASSIGNED
722310312232176 -    004401138059320 1    35m     35m      0
722341261254616 -    013020000000870 1    69m     65m      0
722010003412733 -    867290025172250 1    70m     70m      0
```

Figura 27: Lista de TMSIs

2.8.1 *Creacion de suscriptores en SIPAuthServe*

Para registrar un numero en el SIPAuthServe podemos escribir en la base de datos o podemos utilizar una herramienta que proporciona openBTS para cargar los datos. Cada terminal que se conecte a la red debe tener asignado un MSISDN (Mobile Station Integrated Services Digital Network) por lo tanto es un dato que tenemos que generar. La herramienta es nmcli, escrita en python que interacciona con el servicio zeroMQ para ejecutar comandos contra el sistema pot TCP. La ejecución para agregar un nuevo subscriptor es la siguiente

```
./nmcli.py SIPAuthServe subscribers create name imsi msisdn ki
```

El imsi lo podemos obtener de la tabla y se puede verificar en los terminales ingresando el código *#06# para verificar que se el terminal que queremos agregar. El ki es opcional, las comunicaciones no estarán cifradas sin el, en las SIM que sean de un proveedor comercial no podemos averiguar este codigo, por lo tanto lo dejaremos vacío.

Una vez que se ingresa el suscriptor, podemos reintentar la registración y vemos como en la figura 28 que el mismo se puede registrar correctamente. Podemos ver tambien la registración exitosa desde el log de openbts.

2.8.2 *Registracion contra el SGSN*

Una vez registrado comenzará a enviar peticion de registración contra el SGSN para la conexión de datos con su TMSI y otros datos adicionales. Para esto el terminal necesita tener configurado un APN, por más que no tenga valores reales, sin un APN configurado el software del terminal no intenta ninguna registración contra un SGSN.

En los terminales más modernos la registración de datos puede tardar varios minutos, ya que la tecnología 2G es muy lenta para una conexión de datos para el uso actual, y el terminal prefiere probar muchas

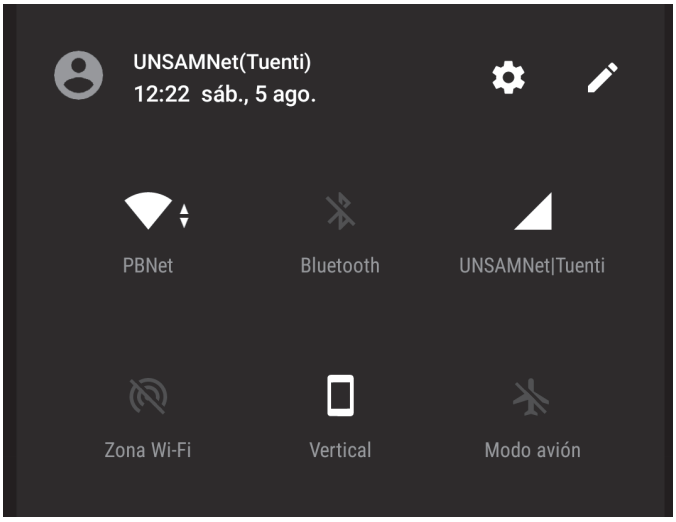


Figura 28: Registración correcta con una SIM de operador comercial

otras combinaciones contra la estación base antes de terminar en una conexión GPRS[27].

Podemos observar en la figura 29 un terminal registrado y en línea contra el sgsn

```
OpenBTS> gprs list
 MS#1,TLLI=c00ee001,ba22d09d rrmode=PacketIdle Bytes:819up/462down Utilization=0%
        GMM Context: imsi=722010003412733 ptmsi=0xee001 tlli=0xc00ee001 state=GmmRegisteredNormal age=1506 idle=708 IPs=none
        TimingError=(-0.96 min=-1.50 max=-0.92 avg=-1.18 N=2346) RSSI=(-56 min=-59 max=-20 avg=-52.89 N=2346) CV=(62 min=48 max=63 avg=60.00 N=27) ILe
v=(0) RXQual=(0 min=0 max=7 avg=1.00 N=14) SigVar=(0) ChCoding=(0)
        dataER:.7% (390) recent:0% (0) tbfER:.12% (26)
        rrbpER:.02% (125) recent:0% (0) cchER:0% (0) recent:0% (0)
 MS#2,TLLI=c00ee002,7bb16d22 rrmode=PacketIdle Bytes:1196up/77down Utilization=0%
        GMM Context: imsi=722341261254616 ptmsi=0xee002 tlli=0xc00ee002 state=GmmRegisteredNormal age=142 idle=3 IPs=192.168.99.2
        TimingError=(-0.41 min=-1.15 max=-0.28 avg=-0.76 N=3809) RSSI=(-39 min=-39 max=-38 avg=-38.88 N=3809) CV=(13 min=0 max=14 avg=12.50 N=12) ILev
=(0) RXQual=(0 min=0 max=7 avg=3.50 N=2) SigVar=(0) ChCoding=(3)
        dataER:.7% (201) recent:0% (0) tbfER:0% (13)
        rrbpER:0% (144) recent:0% (0) cchER:0% (0) recent:0% (0)
 PDCH ARFCN=51 TN=1 FER=0%
 PDCH ARFCN=51 TN=2 FER=0%

OpenBTS> sgsn list
 GMM Context: imsi=722010003412733 ptmsi=0xee001 tlli=0xc00ee001 state=GmmRegisteredNormal age=1508 idle=710 MS#1,TLLI=c00ee001,ba22d09d IPs=none

 GMM Context: imsi=722341261254616 ptmsi=0xee002 tlli=0xc00ee002 state=GmmRegisteredNormal age=144 idle=2 MS#2,TLLI=c00ee002,7bb16d22 IPs=192.168.99.2
```

Figura 29: lista de usuarios activos de SGSN

27 Problemas recurrentes en OpenBTS: https://github.com/RangeNetworks/openbts/blob/master/GPRS/todo.txt

Parte III

MEDICIONES Y RESULTADOS

3

MEDICIONES Y OPTIMIZACIÓN

3.1 MEDICIONES CON HACKRF Y GNURADIO

Las mediciones se realizaron con una placa SDR HackRF, la cual consta de una antena de similares características a las utilizadas con la placa USRP B210. En primera instancia se utilizó el software GQRX para verificar la correcta transmisión de la señal en la Banda de Upload y Download. Siendo el ARFCN 51 las bandas como vemos en la tabla 4 nos centramos cerca de la frecuencia a monitorear y podemos ver como en la figura 30 se observa el espectro de la misma.

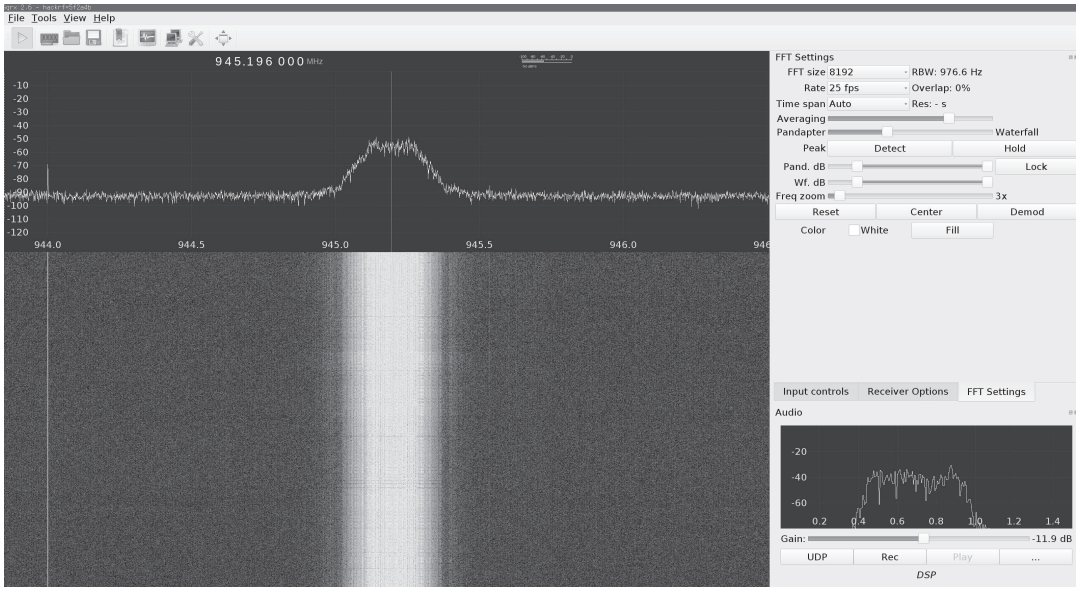


Figura 30: Espectro de Downlink

También podemos ver como solo en momentos de recepción de datos transmitidos desde un terminal vemos que hay un espectro en la banda de 900.2 Mhz. En la figura 31 se puede observar en el grafico de cascada los momentos de no transmisicion.

Cuadro 4: Tabla de Frecuencias para ARFCN 51

Banda	ARFCN	Downlik (Mhz)	Uplink (Mhz)
900 P	51	945.2	900.2

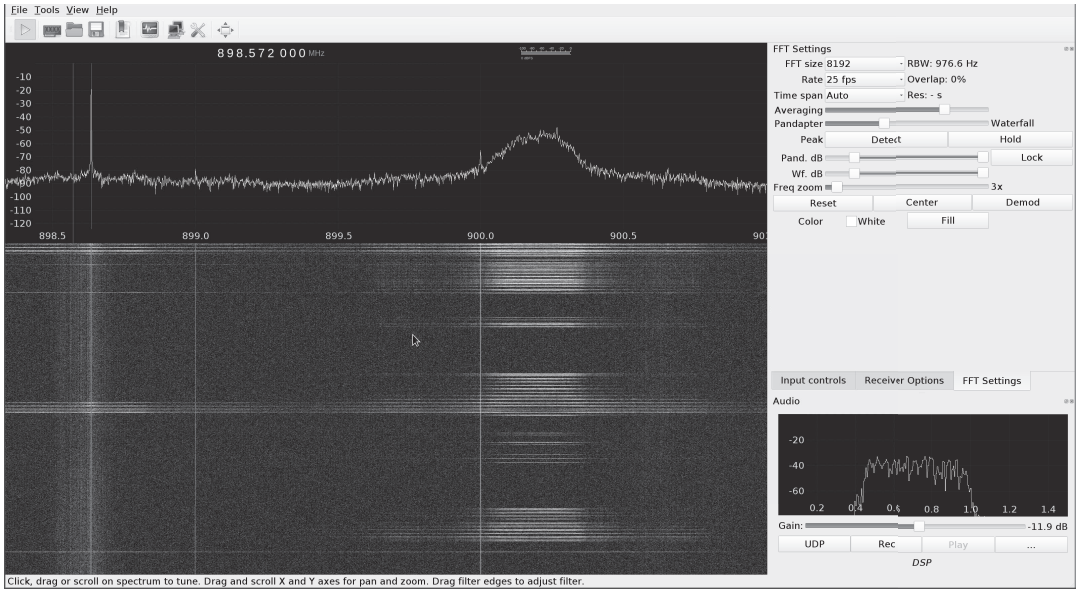


Figura 31: Espectro de Uplink

3.1.1 *Esquema en GNURadio*

Los datos capturados con gqrx pueden ser guardados, pero más eficientemente podemos realizar un esquema en GNU que los procese. Para ello se realizó un esquema con módulos para el tratamiento y de codificación de señales GSM para SDR de bajo costo[1]. El diagrama completo lo podemos ver en el anexo 5.5

El primer módulo es el ingreso de la información del SDR a través de un "RTL-SDR source" como se ve en la figura 32 donde se setean algunos valores. Entre los valores están la frecuencia central donde la cadena RF se va a situar, La corrección de Frecuencia en PPM[2] (partes por millón) la cual podemos obtener con al software kalibrate probando contra celdas conocidas, la ganancia, el ancho de banda entre otros.

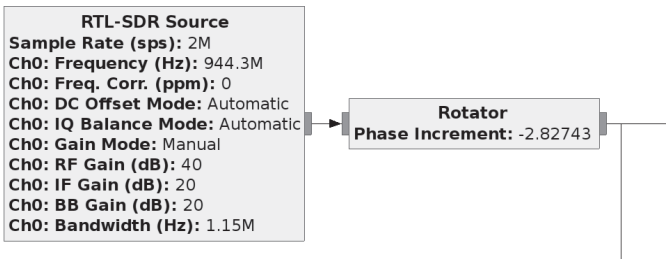


Figura 32: Origen SDR

1 Modulo GR-GSM: https://github.com/ptrkrysik/gr-gsm
2 Calculo de PPM http://digital.ni.com/public.nsf/allkb/2A0B9D3F365DEDEF86256BDB007354ED

Luego tenemos un adaptador de burst GSM el cual tiene un control de lazo cerrado para la frecuencia que es muy necesario para sincronizar las secuencia ya que se necesita tener una precisión de 1 ppm para poder leerlas correctamente, y algunas placas RTL-SDR por efectos de calor y relojes de baja calidad llegan a 80 ppm. Luego del control de sincronización tiene un filtro pasa bajos antes de terminar como se ve en la figura 34.

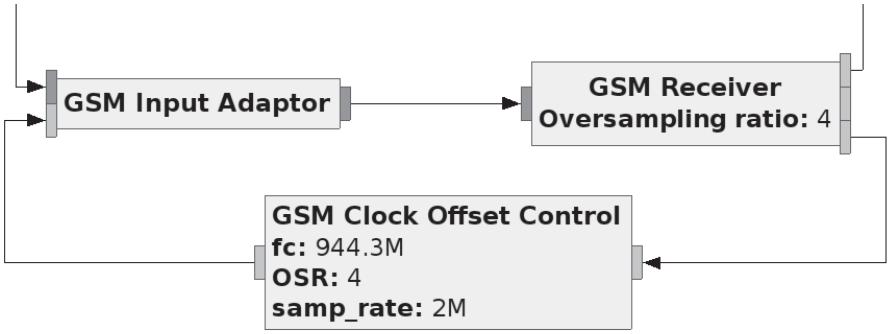


Figura 33: Receptor GSM

Luego del adaptador tenemos el receptor el cual resta el oversampling de la red GSM y es quien se encarga de calcular la corrección siguiendo los burst de corrección de frecuencia y de sincronización. A parte de informar corrige su sincronización y devuelve un burst GSM correcto a los mapeadores.

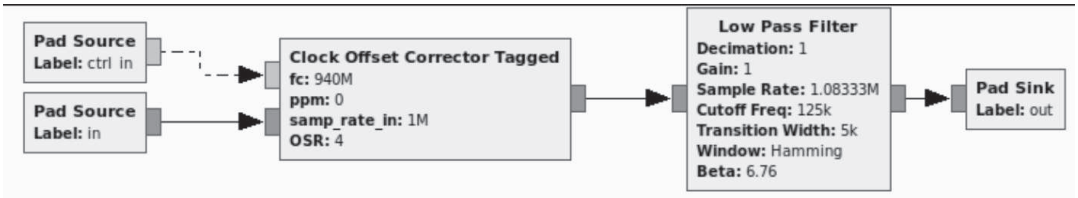


Figura 34: Adaptador GSM

Los mapeadores revisan la cabecera del burst GSM para filtrar por time-slot que es un parámetro que se puede definir, y lo mapea al canal correcto. Agrupan en los distintos tipos de canales lógicos y tipos dentro de los mismos. Se puede ver en la figura 35 que podemos mapear canales de tráfico denominados THC/F (donde va la voz, los datos en si, etc..) en GSM y canales de control como los denominados BCCH (informacion de estacion base en modo broadcast o para todos), SDCCH, CCCH (es el canal común de control, para informar a los terminales y a la estación base del establecimiento de llamadas, sms, registraciones, etc..), SACCH (informa datos de la relacion señal a ruido para el canal de bajada o downlink) y FACCH (encargado

de informar autenticación, handover, o que el terminal debe pasar a escuchar otro canal logico como si recibiera por ejemplo un sms).

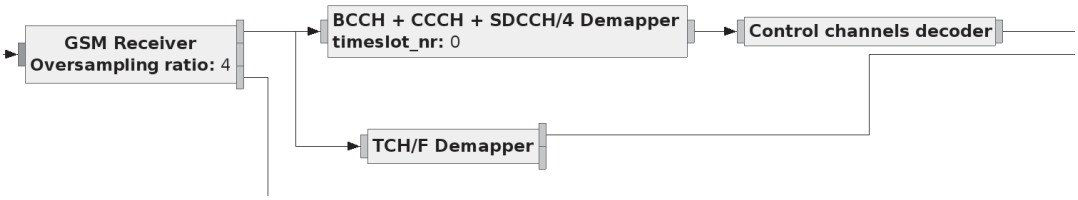


Figura 35: Mapeadores de Canales

Si tuvieramos el Ki de la SIM y las comunicaciones irían cifradas podemos insertar un módulo de descifrado para obtener los datos planos.

Por ultimo enviamos estos paquetes que pueden ser leídos por software como GNU-Radio a través de un Socket PDU como se ve en la figura 36.

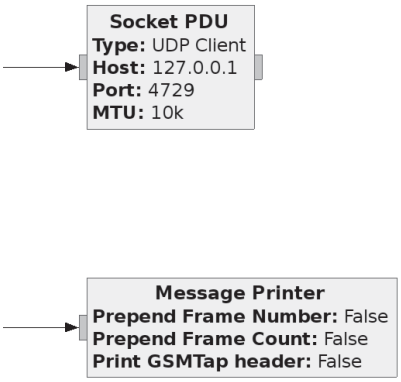


Figura 36: Salida a socket PDU

Además se agregan unas visualizaciones y unos deslizables para el fácil seteo de los parámetros como se ve en la figura 37.

3.2 CAPTURAS DE TRÁFICO CON GNU-RADIO

Para las capturas de tráfico se diseñó una prueba con envíos de SMS de un terminal a otro. La captura se hizo remota con el mismo esquema de GNU-Radio presentado y la placa Hack-RF centrada a la frecuencia de downlink de la estación base (ARFCN 51) .

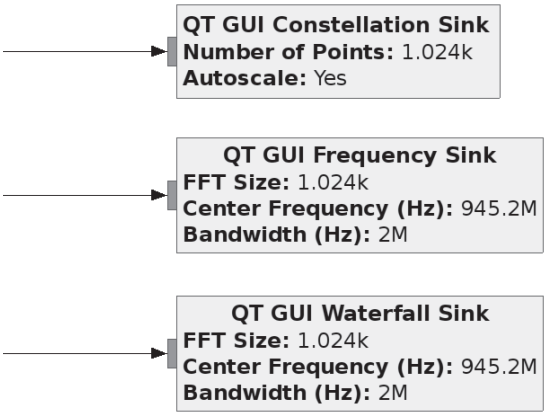


Figura 37: Visualizaciones

Lo primero que se hizo fue setear un mapeador del canal de broadcast de la estación base (BCCH) en el timeslot 0. En la misma vamos a poder ver los mensajes de broadcast que se envían al celular que va a recibir el mensaje para que se sitúe en el canal lógico adecuado para leerlo. Al no estar cifradas las comunicaciones no hace falta un descifrado de la información.

Como vemos en la figura 38 se encontro el "inmmediate assigment" que indica al terminal con el IMSI de destino cual es el canal y el timeslot en donde se le va a enviar la información. En el caso del experimento es el SDDCH/4 (Stand-alone Dedicated Control Channel, que es designado por la estacion base para que el terminal realize acciones, en este caso la recepción de un SMS) en el timeslot 0.

En el caso de las pruebas el mensaje enviado fue "Test"desde el terminal con el numero MSISDN 151111112 que se le asignó previamente en el SipAuthServe.

Cambiamos los mapeadores al canal SDDCH/4 en el timeslot 0 y vemos recibir en la figura 39 un mensaje identificado con el protocolo GSM/SMS el cual indica el IMSI, el MSISDN que le asignamos a ese terminal y el texto del mensaje a recibir con la fecha de envío.

En caso de querer revisar los datos de voz traficada en el canal SDDCH se realiza la indicación de que canal de tráfico (TCH) debe utilizar para traficar los paquetes de VoIP. en el cual se cambian los mapeadores al timeslot y canal adecuados y luego necesitamos de alguna otra herramienta que junte los paquetes de voz y los codifique en GSM para

ser reproducidos. Para esto se puede utilizar el script de python que implementa el código de GNU-Radio grgsm decode [3].

```
329 2.250284700   127.0.0.1           127.0.0.1           GSMTAP      81 (CCCH) (GCC)
330 2.253171541   127.0.0.1           127.0.0.1           GSMTAP      81 (CCCH) (GCC)
331 2.258762691   127.0.0.1           127.0.0.1           GSMTAP      81 (CCCH) (RR) System Information Type 2
332 2.261727193   127.0.0.1           127.0.0.1           GSMTAP      81 (CCCH) (RR) Paging Request Type 1
333 2.266587203   127.0.0.1           127.0.0.1           GSMTAP      81 (CCCH) (RR) Paging Request Type 1
334 2.269399787   127.0.0.1           127.0.0.1           GSMTAP      81 (CCCH) (RR) Immediate Assignment
335 2.273954161   127.0.0.1           127.0.0.1           GSMTAP      81 (CCCH) (CC)
336 2.289619496   127.0.0.1           127.0.0.1           GSMTAP      81 (CCCH) (GCC)
337 2.292578629   127.0.0.1           127.0.0.1           GSMTAP      81 (CCCH) (GCC)
338 2.297740907   127.0.0.1           127.0.0.1           GSMTAP      81 (CCCH) (RR) System Information Type 3
339 2.300985228   127.0.0.1           127.0.0.1           GSMTAP      81 (CCCH) (RR) Paging Request Type 1
340 2.305581072   127.0.0.1           127.0.0.1           GSMTAP      81 (CCCH) (RR) Paging Request Type 1
341 2.308901341   127.0.0.1           127.0.0.1           GSMTAP      81 (CCCH) (RR) Paging Request Type 1
342 2.313247386   127.0.0.1           127.0.0.1           GSMTAP      81 (CCCH) (CC)
343 2.328883064   127.0.0.1           127.0.0.1           GSMTAP      81 (CCCH) (GCC)
344 2.331872244   127.0.0.1           127.0.0.1           GSMTAP      81 (CCCH) (GCC)
345 2.337181349   127.0.0.1           127.0.0.1           GSMTAP      81 (CCCH) (RR) System Information Type 4

▸ .... 0110 = Protocol discriminator: Radio Resources Management messages (0x6)
  Message Type: Immediate Assignment
▸ Page Mode
▾ Dedicated mode or TBF
    0000 .... = Dedicated mode or TBF: This message assigns a dedicated mode resource (0)
▾ Channel Description
    0010 0... = SDCCH/4 + SACCH/C4 or CBCH (SDCCH/4): 4
    Subchannel: 0
    .... .000 = Timeslot: 0
    010. .... = Training Sequence: 2
    ...0 .... = Hopping Channel: No
    ..00 .... = Spare: 0x00
    Single channel ARFCN: 51
```

Figura 38: Inmmediate Assigment

```
9131 127.0.0.1           127.0.0.1           GSMTAP      81 (CCCH) (RR) Paging Request Type 1
1974 127.0.0.1           127.0.0.1           LAPDm       81 U P, func=DISC
6779 127.0.0.1           127.0.0.1           GSM SMS     81 I, N(R)=3, N(S)=0(DTAP) (SMS) CP-DATA (RP) RP-DATA (Network to MS)
8945 127.0.0.1           127.0.0.1           LAPDm       81 U, func=UI(DTAP) (RR) System Information Type 6
1582 127.0.0.1           127.0.0.1           LAPDm       81 U, func=UI(DTAP) (RR) System Information Type 6
0754 127.0.0.1           127.0.0.1           GSMTAP      81 (CCCH) (RR) System Information Type 3
9830 127.0.0.1           127.0.0.1           GSMTAP      81 (CCCH) (RR) Paging Request Type 1
9622 127.0.0.1           127.0.0.1           GSMTAP      81 (CCCH) (RR) Paging Request Type 1
1134 127.0.0.1           127.0.0.1           GSMTAP      81 (CCCH) (RR) Paging Request Type 1
5515 127.0.0.1           127.0.0.1           LAPDm       81 U P, func=DISC
7191 127.0.0.1           127.0.0.1           LAPDm       81 S, func=RR, N(R)=4
1428 127.0.0.1           127.0.0.1           LAPDm       81 U, func=UI(DTAP) (RR) System Information Type 6
9563 127.0.0.1           127.0.0.1           LAPDm       81 U, func=UI(DTAP) (RR) System Information Type 5
4433 127.0.0.1           127.0.0.1           GSMTAP      81 (CCCH) (RR) System Information Type 4
6383 127.0.0.1           127.0.0.1           GSMTAP      81 (CCCH) (RR) Paging Request Type 1
7383 127.0.0.1           127.0.0.1           GSMTAP      81 (CCCH) (RR) Paging Request Type 1
5033 127.0.0.1           127.0.0.1           GSMTAP      81 (CCCH) (RR) Paging Request Type 1

4
displayed)
RP-DATA (Network to MS)
SM 03.40) SMS-DELIVER
TP-RP: TP Reply Path parameter is not set in this SMS SUBMIT/DELIVER
TP-UDHI: The TP UD field contains only the short message
TP-SRI: A status report shall not be returned to the SME
TP-LP: The message has not been forwarded and is not a spawned message
TP-MMS: No more messages are waiting for the MS in this SC
TP-MTI: SMS-DELIVER (0)
ig-Address - (411)


entre-Time-Stamp
-Length: (103) depends on Data-Coding-Scheme

2 queued, 1 unlocatable, cell 0.1, IMSI722341261254616, phonenum 1511111112, at Aug  7 22:57:24, 'Test'
```

Figura 39: SMS recibido

3 GRGSM Decode: https://github.com/ptrkrysik/gr-gsm/blob/master/apps/grgsm_decode

3.3 CALCULO DE PERDIDAS EN ESPACIO LIBRE CON MODELADO OKUMURA/HATA

El modelo de propagación de señales de radio y su pérdida modelada con Okumura/Hata nos permite tener una idea del alcance de la cobertura de la antena. Para ello primero calculamos dos modelos, en un ambiente de ciudad pequeña y otro de area suburbana (mas parecida al campus miguelete de la universidad). Definimos Lu como la perdida para el modelo calculado y Ch la correccioón de altura como mas abajo se especifica.

Para un modelo urbano:

$$L_U = 69{,}55 + 26{,}16 \log_{10} f - 13{,}82 \log_{10} h_B - C_H + [44{,}9 - 6{,}55 \log_{10} h_B] \log_{10} d$$

Con

$$C_H = 0{,}8 + (1{,}1 \log_{10} f - 0{,}7) h_M - 1{,}56 \log_{10} f$$

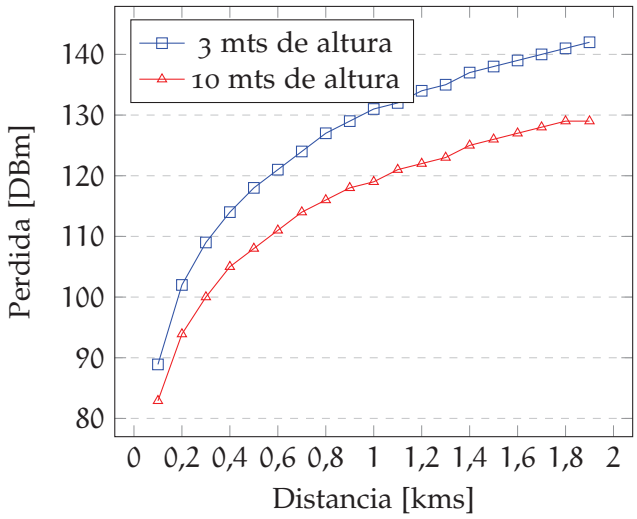
Y para el modelo suburbano

$$L_{SU} = L_U - 2\left(\log_{10} \frac{f}{28}\right)^2 - 5{,}4$$

Siendo, h_B la altura de la estación Base, la cual se fijó a 10 mts como ideal a instalar y a 3mts como se pudo instalar para las mediciones, h_M la altura de la antena del terminal, la cual se fijó a 1.5 mts, f la frecuencia que es para el ARFCN seteado a 945.2 Mhz, C_H el factor de corrección de altura, y d la distancia en kms.

Se calculo con la ayuda de un script en python con librerias de numpy como se puede observar en el anexo 5.4, los resultados se graficaron en el siguiente esquema:

Modelado de Media Okumura/Hata para fq=945.2 Mhz



La potencia entregada en la antena de la estación base está seteada a 10 mW, con lo cual podemos calcular que en el terminal se reciba una potencia mayor a la recomendada de -110 DBm.

$$RSL = -110DBm = 10\log_{10}(10mW) - L_{SU}$$

Nos da un aproximado de alcance de 1.1 km con los calculos teoricos para 10mts de altura y aproximadamente 600mts para 3 mts de altura.

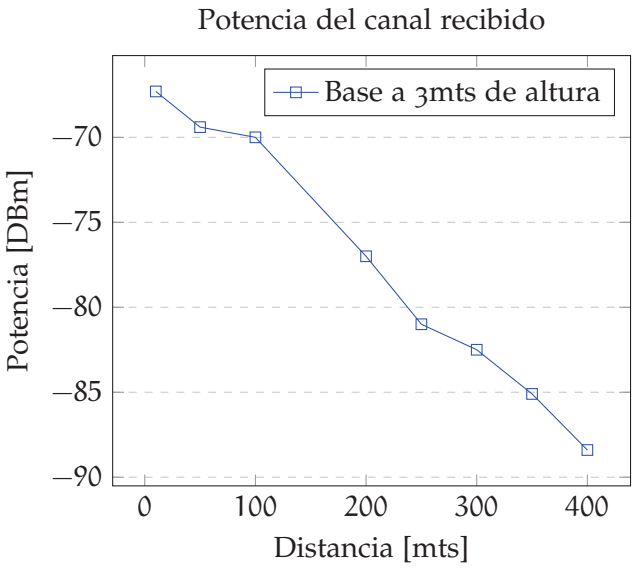
3.4 MEDICIÓN DE COBERTURA CON ANALIZADOR DE FRECUENCIA

Se utilizo un analizador de espectro Anritsu MS2721 [4] para las mediciones con una antena de iguales características que la de la estación base.

Se ubicó la estación base en una ubicación de altura aproximada a 3 mts y se procedió a tomar mediciones seteando un marcador en el canal de downlink de frecuencia central 945.2 mhz con un ancho de banda de 200 Khz. Luego de las mediciones se paró hasta llegar a los -90 dbm o un nivel de ruido sin pre amplificador que no se pudiera distinguir la señal.

4 Anritsu ms2721:https://www.anritsu.com/en-US/test-measurement/products/ms2721b

Figura 40: Analizador de espectro centrado en el canal ARFCN 51



Como se observa en el gráfico los valores medidos se ajustan bastante correctamente con el modelado teórico Okumura-Hata para un ambiente suburbano en donde podemos llegar a 500 mts con señal aún utilizable para una comunicación fluida.

El uso de algunos amplificadores para aumentar la potencia de transmisión, separando físicamente las antenas para que no se saturen y aumentando la altura algunos metros más, o colocándola sobre un techo de los edificios del campus puede fácilmente alcanzar el kilómetro y cubrir toda la extensión del campus.

Parte IV

CONCLUSIONES

4

CONCLUSIONES

4.1 CONCLUSIÓN

En este proyecto se pudo realizar una completa implementación de una red GSM desde su parte de radiofrecuencia como todos los servicios básicos para su funcionamiento ya sea de SMS, telefonía y Conexión de datos a Internet.

Se pudo implementar una red de las denominadas 2G/2.5G, que cuenta con establecimiento de llamadas de voz, SMS y salida de datos por medio de GPRS (2.5G).

Se estudió también la implementación de una central VoIP para brindar el servicio de voz a la red y la conexión al mundo exterior por un gateway hacia la PSTN. En cuanto a los datos se implementó un ruteo interno en el servidor y una implementación de una subred privada con NAT al exterior.

Se pudo con éxito hacer mediciones y tomar estadísticas de la red, hacer capturas de tráfico y ver el funcionamiento de los canales lógicos en GSM para las distintas etapas de la comunicación. Se utilizaron elementos de medición de campo para poder ajustar y optimizar los valores de configuración en el transmisor.

Todo el armado del proyecto se realizó con software de código abierto, y se crearon algunas programas para ayudar a la medición. Se utilizaron otras placas SDR para con la ayuda del software GNU Radio poder sensar la información transmitida por la estación base y el tráfico de los terminales.

Gracias al desarrollo del proyecto se pudieron implementar muchos de los temas teóricos que se vieron en distintas materias de la carrera, ya sea en Comunicaciones Digitales, Programacion, Redes protocolos y convergencia, Electrónica Digital, Tecnología de las telecomunicaciones y Teoría de la información y la codificación en agregado a todas las materias que crearon una base teórica para estas últimas mencionadas.

Se deja por ultimo armado el laboratorio en funcionamiento con todas las configuraciones y programas para la medición en el laboratorio de electrónica del Campus Miguelete de la Universidad Nacional de San Martin.

Parte V

ANEXOS

5

ANEXOS

5.1 SCRIPT DE INICIO DE LA INTERFAZ

```
#!/bin/bash

RANGO_IP="170.51.255.240/24"
INTD="eth10"

function start {
        #inserto el kernel module dummy
        sudo modprobe dummy

        # seteo la interfaz con un nombre
        sudo ip link set name $INTD dev dummy0

        # configuro el rango de ip
        sudo ip addr add $RANGO_IP brd + dev $INTD

        #levanto la interfaz
        sudo ifconfig $INTD up
}

function stop {
        sudo ifconfig $INTD down
        sudo ip addr del $RANGO_IP brd + dev $INTD
}

function check {
        #saco la ip de la interfaz
        TEST=‘ifconfig |grep -A3 $INTD | awk -F : ’/inet / {print
             $2}’ | awk ’{print $1}’‘
        if [[ $TEST == "" ]]
        then
                echo "Ocurrio un error o la interfaz no fue
                    iniciada"

        elif [[ $RANGO_IP =~  $TEST.* ]]
        then
```

```
                echo "Configuracion OK IP: $TEST, INTERFAZ: $INTD
                    "
        else
                echo "Ocurrio un error"
        fi
}

#entra en la funcion llamada
$1
###END###
```

5.2 REGLAS IPTABLES

```
*nat
:PREROUTING ACCEPT [0:0]
:POSTROUTING ACCEPT [0:0]
:OUTPUT ACCEPT [0:0]
-A POSTROUTING -o wlp3s0 -j MASQUERADE
COMMIT

*filter
:INPUT ACCEPT [0:0]
:FORWARD ACCEPT [0:0]
:OUTPUT ACCEPT [0:0]
COMMIT
```

5.3 SCRIPT PARA TOMAR DATOS DE MEDICIÓN

```
#!/bin/env python

import subprocess
import time
from datetime import date

today = date.today().strftime("%d-%m-%y-%H%M")
archivo = open("./mediciones-"+today+".log","w")
while True:
        p = subprocess.Popen(["openbtscli", "-c",'"chans"'],
            stdout=subprocess.PIPE)
        distance = raw_input("Ingrese distancia en metros o '
            salir' para salir: ")
        if 'salir' in distance or 'quit' in distance:
                break
        #espera a que se actulizen los datos en openbts
        time.sleep(3)
        stdout,stderr = p.communicate()
        archivo.write("\n---------- Distancia {} metros
            ----------\n".format(distance))
        print ("---------- Distancia {} metros
            ----------".format(distance))
        line = stdout.split('\n')
        #seteado en 2 para saltear las cabezeras de la respuesta
        j=2
        #iteracion por linea y filtrado por columna de los
            valores a sensar
```

```
        while len(line)-3 > j:
                entry=line[j].split()
                if len(entry) >= 12:
                        imsi = entry[12]
                        archivo.write("IMSI {} -> snr:{} txpower
                            :{} rxlevel:{}\n".format(imsi,entry
                            [5],entry[8],entry[9]))
                        print("IMSI {} -> snr:{} txpower:{}
                            rxlevel:{}".format(imsi,entry[5],
                            entry[8],entry[9]))
                j=j+1

archivo.close()
```

5.4 CALCULO DE PUNTOS PARA MODELADO OKUMURA/HATA

```
from __future__ import print_function
import math
import numpy as np


d=np.array(map(lambda x: x/10.0, range(1, 20, 1)));  #array en km
     con divisiones cada 100 metros
hb=20; #Altura de la Base station en metros
hm=1.5;#Altura del terminal en metros
fc=945.2;#Frecuencia en Mhz


#Okumura/Hata
ahm=(1.1*math.log10(fc)-0.7)*hm-(1.56*math.log10(fc)-0.8);
L_50=69.55+26.16*np.log10(fc)+(44.9-6.55*np.log10(hb))*np.log10(d
    )-13.82*np.log10(hb)-ahm;
Ls_50=L_50-2*(math.log10(fc/28))**(2)-5.4
L_50 = np.array(L_50)
Ls_50 = np.array(Ls_50)

print("Valores L50 para modelado Okumura/Hata ")
i=0
print('Ciudad')
for resultado in L_50:
    print('({},{:.3g})'.format(d[i],resultado), end="")
    i=i+1
print('\n\nSuburbano')
i=0
for resultado in Ls_50:
    print('({},{:.3g})'.format(d[i],resultado), end="")
    i=i+1
```

5.5 ESQUEMA GNURADIO PARA RECEPCIÓN

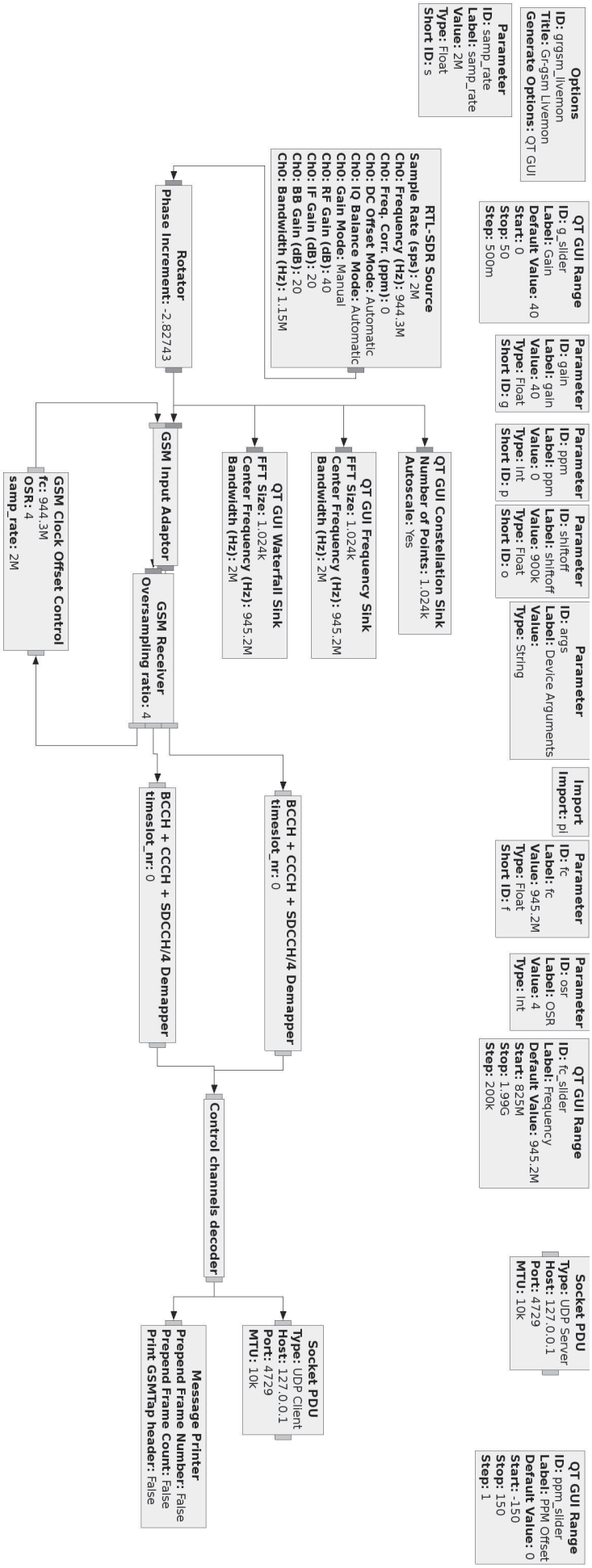


Figura 41: Diagrama completo de recepción

BIBLIOGRAFÍA

[1] Leslie Lamport, *LaTeX: a document preparation system*, Addison Wesley, Massachusetts, 2nd edition, 1994.

[2] I. Penttinen, Jyrki T. J., editor, *The telecommunications handbook: engineering guidelines for fixed, mobile, and satellite systems*, John Wiley & Sons Ltd, The Atrium, Southern Gate, Chichester, West Sussex, PO19 8SQ, United Kingdom, 1st edition, 2015.

[3] H. Schulzrinne, *RFC 2833: RTP Payload for DTMF Digits, Telephony Tones and Telephony Signals*, Columbia University May 2000.

[4] DARPA Internet Program, *RFC 791: INTERNET PROTOCOL - PROTOCOL SPECIFICATION*, Defense Advanced Research Projects Agency Sep 1981.

[5] J. Rosenberg, H. Schulzrinne, G. Camarillo, A. Johnston, J. Peterson, R. Sparks, M. Handley, E. Schooler, *RFC 3261: SIP: Session Initiation Protocol*, dynamicsoft, Columbia U., Ericsson, WorldCom, Neustar, ICIR, AT&T Jun 2002.

[6] Y. Rekhter, B. Moskowitz, D. Karrenberg, G. J. de Groot, E. Lear *RFC 1918: Address Allocation for Private Internets*, Cisco Systems, Chrysler Corp., RIPE NCC, Silicon Graphics, Inc., Feb 1996.

[7] S. Donovan, *RFC 2976: The SIP INFO Method*, dynamicsoft Oct 2000.

[8] Range Networks, Inc., *OpenBTS Application Suite 4.0 - User Manual*, Range Networks, Inc. Apr 2014.

[9] Natalizio, E., V. Loscri, G. Aloi, N. Paoli and N. Barbaro. , *The practical experience of implementing a GSM BTS through open software/hardware.*, ISABEL. 2010.

[10] Kostanic R. *RF Propagation - Okumura and Hata Macroscopic Propagation Models*, Florida Insitute of Technology. 2011.

[11] Mameri Z. *Wireless and Mobile Networking*, IRIT 2008

[12] Rappaport, T. S *Wireless communications: Principles and practice*, Englewood Cliffs: Prentice Hall International. 2002