

# **UNIVERSIDAD NACIONAL DE INGENIERÍA**

**FACULTAD DE INGENIERÍA ELÉCTRICA Y ELECTRÓNICA**



## **TESIS**

**IMPLEMENTACIÓN DE UN PROTOTIPO DE RED GSM/GPRS  
UTILIZANDO DISPOSITIVOS BASADOS EN LA TECNOLOGÍA DE  
RADIO DEFINIDA POR SOFTWARE PARA PROVISIONAMIENTO  
DE SERVICIO DE TRANSFERENCIA DE DATOS APLICADOS A IOT.**

**PARA OBTENER EL TÍTULO PROFESIONAL DE  
INGENIERO DE TELECOMUNICACIONES**

**ELABORADO POR:  
OSCAR ENRIQUE LLERENA CASTRO**

**ASESOR  
DANIEL DIAZ ATAUCURI**

**LIMA – PERÚ**

**2018**



**IMPLEMENTACIÓN DE UN PROTOTIPO DE RED GSM/GPRS UTILIZANDO  
DISPOSITIVOS BASADOS EN LA TECNOLOGÍA DE RADIO DEFINIDA POR  
SOFTWARE PARA PROVISIONAMIENTO DE SERVICIO DE TRANSFERENCIA DE  
DATOS APLICADOS A IOT.**

to my

## **AGRADECIMIENTOS**

## **SUMARIO**

## **ABSTRACT**

## ÍNDICE

<b>AGRADECIMIENTOS .....</b>	<b>IV</b>
<b>SUMARIO .....</b>	<b>v</b>
<b>ABSTRACT .....</b>	<b>VI</b>
<b>PRÓLOGO .....</b>	<b>1</b>
<b>CAPÍTULO 1</b>	
<b>INTRODUCCIÓN .....</b>	<b>2</b>
1.1    Descripción del problema. ....	2
1.2    Descripción del prototipo de red propuesto. ....	3
1.3    Estado del Arte. ....	4
1.4    Justificativa. ....	4
1.5    Objetivos. ....	5
1.5.1    Objetivo General. ....	5
1.5.2    Objetivos Específicos. ....	5
1.6    Hipóteis. ....	5
1.7    Contenido. ....	6
<b>CAPÍTULO 2</b>	
<b>FUNDAMENTOS TEÓRICOS Y PRÁCTICOS PARA LA IMPLEMENTACIÓN DE LA RED PROTOTIPO GSM/GPRS.....</b>	<b>7</b>
2.1    Arquitectura de red GSM/GPRS tradicional. ....	8
2.2    Establecimiento del servicio de transferencia de datos entre un MS y una PDN externa sobre una red GSM/GPRS. ....	9
2.2.1    Modulación de señales en una red GPRS. ....	11
2.2.2    Esquemas de acceso TDMA y FDMA. ....	13
2.2.3    Mapeo de canales lógicos en canales físicos. ....	14
2.2.4    Inicio de conexión del dispositivo final. ....	15
2.2.5    Acceso por radio a la red GSM/GPRS. ....	17

2.2.6	Establecimiento de conexión a nivel GPRS. . . . .	19
2.2.7	Establecimiento del contexto PDP. . . . .	22
2.3	Descripción del dispositivo final. . . . .	24
2.3.1	Implementación de una conexión serial para control del dispositivo final. . . . .	27
2.3.2	Estados GPRS en el dispositivo final. . . . .	28
2.3.3	Conexión GPRS y envío de datos a una PDN externa. . . . .	30
2.4	Plataforma de Software OpenBTS. . . . .	34
2.4.1	Comparación entre una red GPRS tradicional y una red GPRS implementada con OpenBTS. . . . .	35
2.4.2	Descripción de los procesos computacionales de OpenBTS relacionados con el establecimiento del servicio GPRS. . . . .	36
2.5	Tecnología Radio Definido por Software (SDR). . . . .	38
2.5.1	Transceiver Superheterodino. . . . .	38
2.5.2	Transceiver SDR Ideal. . . . .	40
2.5.3	Transceiver SDR basado en el mixer digital. . . . .	41
2.5.4	Módulo USRP B210. . . . .	44
<b>CAPÍTULO 3</b>		
	<b>IMPLEMENTACIÓN DEL PROTOTIPO DE RED GPRS. . . . .</b>	<b>48</b>
3.1	Requerimientos de Hardware y Software. . . . .	48
3.2	Instalación e inicialización de OpenBTS. . . . .	49
3.2.1	Instalación de librerías dependencias. . . . .	49
3.2.2	Configuración para la habilitación del servicio GPRS. . . . .	52
3.2.3	Configuración para la habilitación del registro en la red. . . . .	52
3.2.4	Configuración del canal ARFCN. . . . .	53
3.2.5	Configuración de la identidad de la red y de la estación base. . . . .	54
3.2.6	Configuración del número de timeslots asignados para el servicio GPRS. . . . .	54
3.2.7	Configuración del número máximo de canales GPRS asignados a un MS. . . . .	55
3.2.8	Configuración de condiciones mínimas para el RSSI y SNR en el uplink. . . . .	56
3.2.9	Configuración de la potencia de transmisión. . . . .	57
3.2.10	Configuración de parámetros relacionados al GGSN. . . . .	58
3.2.11	Configuración del redireccionamiento de paquetes. . . . .	59
3.2.12	Configuración de la habilitación para la captura de tráfico. . . . .	59

3.2.13 Configuración de visualización de logs. . . . .	60
3.2.14 Visualización de logs. . . . .	61
3.2.15 Visualización del RSSI de los MS conectados vía GPRS. . . . .	61
3.3 Transferencia de Paquetes de Datos sobre la red GPRS. . . . .	62
<b>CAPÍTULO 4</b>	
<b>PRUEBAS Y RESULTADOS. . . . .</b>	<b>66</b>
4.1 Análisis de la señal física del prototipo de red GPRS. . . . .	66
4.2 Verificación de los mensajes de señalización. . . . .	69
4.3 Pruebas de Acceso y Capacidad de Red. . . . .	72
4.4 Implementación de una aplicación IoT usando la red GPRS propuesta. . . . .	80
<b>CONCLUSIONES Y RECOMENDACIONES. . . . .</b>	<b>84</b>
<b>BIBLIOGRAFÍA . . . . .</b>	<b>87</b>
<b>ANEXO A</b>	
<b>INSTALACIÓN DE MÁQUINA VIRTUAL UBUNTU 16.04. . . . .</b>	<b>91</b>
<b>ANEXO B</b>	
<b>CÓDIGO ARDUINO PARA CONEXIÓN GPRS Y ENVÍO DE PINGS. . . . .</b>	<b>100</b>
<b>ANEXO C</b>	
<b>CÓDIGO ARDUINO PARA CONEXIÓN GPRS Y ENVÍO DE DATOS AL SERVIDOR DE THE THINGSPEAK. . . . .</b>	<b>103</b>

## **PRÓLOGO**

este es el pr

## CAPÍTULO 1

### INTRODUCCIÓN

La proliferación del uso de dispositivos y aplicaciones orientadas a internet de las cosas está iniciando su crecimiento exponencial. Investigaciones de importantes consultoras en el mundo predicen que para el año 2020 existirán 20 billones de dispositivos inteligentes desplegados para IoT [1], [2]. Es por ello que desde ya se avisa un importante desarrollo en la línea de investigación en torno al desarrollo de redes de acceso inalámbrico para estos dispositivos, entre otros. La presente tesis es justamente el desarrollo de un prototipo de red de bajo costo para provisionar acceso inalámbrico a dispositivos para aplicaciones de internet de las cosas.

#### 1.1 Descripción del problema.

A puertas de lo que los expertos denominan como la revolución industrial 4.0, la necesidad por implementar soluciones para aplicaciones IoT va en aumento. En respuesta, las operadoras de telecomunicaciones han visto apropiado el desarrollo y despliegue de las tecnologías LPWAN basadas en las redes de telefonía celular móvil tales como LTE-M, NB-IoT o E-GSM. Estas tecnologías son orientadas a operar con un bajo consumo de energía en el dispositivo y bajas tasas de transmisión, lo cual es una característica intrínseca de las aplicaciones IoT.

El problema con estas soluciones es que sólo están disponibles en lugares donde las operadoras de telecomunicaciones tiene algún tipo de red desplegada (ya sea 2G, 3G, o 4G), y además de lo expuesto anteriormente, el esquema de servicio que las operadoras de telecomunicaciones ofrecen consiste en una línea de datos con un plan de suscripción por cada dispositivo final, es decir una especie de simcard postpago por dispositivo. Para el despliegue de una solución IoT en el cual se necesite una importante cantidad de dispositivos nodos sensores que envíen periódicamente data a la nube para post-análisis, esto representa un costo operativo relativamente alto.

## 1.2 Descripción del prototipo de red propuesto.

El sistema de comunicaciones propuesto consiste en una arquitectura de componentes computacionales y electrónicos que emulan una red celular la cual soporta la transferencia de datos vía protocolo *General Packet Radio Service* (GPRS).

Las funcionalidades de la red GPRS propuesta pueden ser aplicadas como plataforma para aplicaciones en el área de sensores e Internet de las Cosas (IoT). En adelante, este sistema de comunicaciones será referido como *red prototipo GPRS*, o simplemente *red GPRS*.

La Figura 1.1 describe el sistema propuesto. Los dos principales componentes de la red GPRS propuesta son:

- la Plataforma de Hardware basado en un dispositivo de Radio Definida por Software, y
- la Plataforma de Software basado en un computador genérico con OpenBTS instalado.

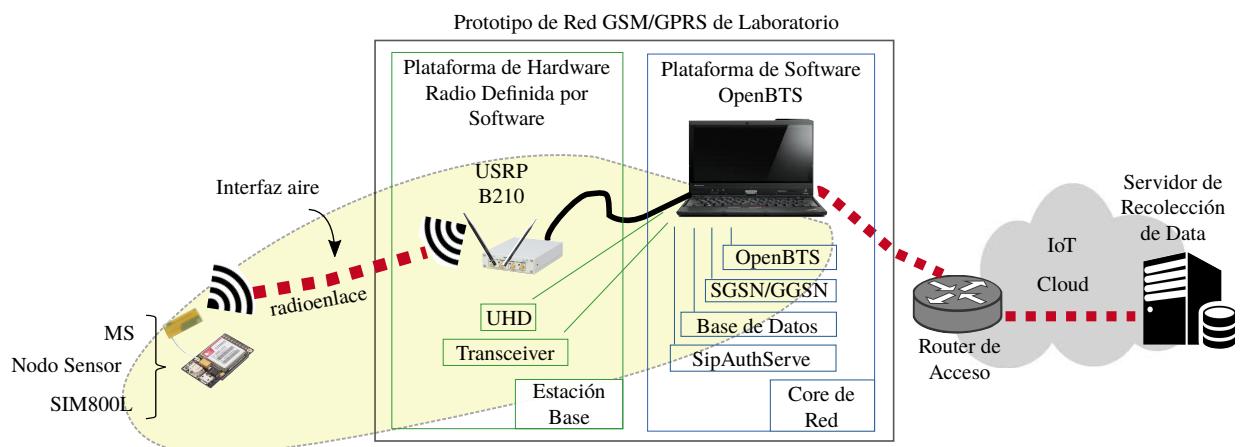


Fig. 1.1: Esquema de una red GPRS sobre una arquitectura de red GSM.

La plataforma de software OpenBTS se implementa en un computador genérico y emula una serie de componentes que en su conjunto se denominan el Núcleo o Core de Red.

La plataforma de hardware se implementa con el dispositivo de radio definida por software USRP B210, cuyo control de funcionalidades está bajo el mando de procesos computacionales implementados en el computador genérico. Este segmento se denomina como la Estación Base e implementa la interfaz aire para el acceso del nodo sensor a la red GPRS.

La red GPRS propuesta permite transportar de manera inalámbrica los radio-paquetes desde el nodo sensor hacia la Estación Base, convertir los radio-paquetes en paquetes IP y re-encaminarlos hacia redes de datos externas a través del Core de Red GPRS. La misma

secuencia en sentido inverso se aplica para paquetes de datos IP provenientes de redes de datos externas dirigiéndose hacia el MS.

### **1.3 Estado del Arte.**

Con la evolución de la tecnología, los sistemas modernos basados en el concepto de Radio Definida por Software (SDR) implementan funciones de radio a nivel de software. Esto quiere decir que ya no existe un hardware específico para cada tipo de modulación o demodulación que se requiera en un sistema. Por el contrario, con el concepto SDR, el modulador/demodulador pasa a ser implementado en software.

En lo que respecta a comunicaciones móviles, el hardware basado en SDR es muy utilizado en la implementación de estaciones base (BTS<sup>1</sup>) [4]. En [5] y [6] se propone la implementación de una red GSM usando dispositivos SDR en conjunto a plataformas de software de código abierto (OpenBTS y Asterisk) para proveer comunicaciones de bajo costo en regiones muy remotas en el África. En [7] y [8] se describe la implementación de un prototipo de Estación Base usando aplicaciones de entorno Unix/Linux en conjunto con el módulo Universal Software Radio Peripheral (USRP) para implementar la interfaz física GSM.

Dadas las características de baja tasa de transmisión y el bajo volumen de los datos generados en aplicaciones IoT, la presente tesis considera la implementación de un prototipo red GPRS que pueda proveer servicio de transferencia de datos a nodos sensores con redes de datos externas. Este trabajo se basa en el proyecto OpenBTS [9], en conjunto con el uso de la plataforma USRP [10].

### **1.4 Justificativa.**

En la mayoría de aplicaciones IoT, un nodo sensor realiza las mediciones propias de su diseño y obtiene data que debe transferir a algún servidor en la nube. Esta transferencia de data la realiza por medio de su módulo de comunicaciones, el cual puede ser un módulo GPRS.

El módulo GPRS necesita una simcard para conectarse a una red comercial y mediante una conexión TCP<sup>2</sup> con el servidor, periódicamente se envía la data obtenida.

Esta solución implica que cada nodo sensor desplegado cuente con un plan de datos

---

<sup>1</sup>BTS: Base Transceiver Station o Estación Base Transceiver.

<sup>2</sup>TCP: Transmission Control Protocol o Protocolo de Control de Transmisión.

por cada simcard, lo cual implica un costo que aumenta con el número de nodos sensores desplegados. Además, esta solución limita el despliegue de los nodos sensores únicamente en áreas donde exista cobertura celular GPRS por parte de algún operador.

La presente investigación busca crear un prototipo de red GPRS de bajo costo que implemente y provea el servicio de transferencia de datos a dispositivos IoT, y de esta manera prescindir del uso de una red comercial.

El prototipo de red propuesto se justifica ya que permite minimizar costos de implementación utilizando una plataforma de hardware basada en Radio Definida por Software y también una plataforma de software de código abierto. Por último, en este trabajo de investigación se expone una metodología de implementación que puede ser reproducida en otras líneas de investigación en las que se necesite implementar soluciones de comunicación utilizando Radio Definida por Software.

## **1.5 Objetivos.**

### **1.5.1 Objetivo General.**

Implementar un prototipo de red GSM/GPRS de laboratorio utilizando SDR para dar servicio de transferencia de datos en aplicaciones IoT.

### **1.5.2 Objetivos Específicos.**

- Implementar un prototipo de red GSM/GPRS de laboratorio de bajo costo utilizando plataformas de hardware basadas en Radio Definida por Software.
- Establecer una conexión de datos entre nodos sensores IoT e Internet a través del prototipo de red GSM/GPRS basado en Radio Definida por Software.
- Compartir el conocimiento adquirido mediante la redacción de un artículo técnico a ser presentado en un congreso nacional y/o internacional.

## **1.6 Hipóteis.**

- Es posible implementar un prototipo de red GSM/GPRS de laboratorio de bajo costo utilizando plataformas de hardware basadas en Radio Definida por Software.
- Es posible establecer conexión de datos entre nodos sensores IoT e Internet a través del prototipo de red GSM/GPRS basado en Radio Definida por Software.

- Es posible compartir el conocimiento adquirido mediante la redacción de un artículo a ser presentado en un congreso nacional y/o internacional.

### **1.7 Contenido.**

El capítulo 2 de la presente tesis contiene los aspectos relacionados a los fundamentos teóricos y prácticos que sirven de base para la implementación y la puesta en funcionamiento de la red prototipo GPRS. El capítulo 3 presenta las consideraciones de diseño e implementación de la red GPRS y las configuraciones necesarias para realizar determinados cambios en el funcionamiento de la red. El capítulo 4 describe las pruebas realizadas sobre el funcionamiento de la red y los resultados obtenidos. Finalmente, el capítulo presenta las conclusiones a partir de los resultados obtenidos y la investigación realizada, así como también los siguientes pasos a seguir para continuar con la mejora del sistema.

## CAPÍTULO 2

### FUNDAMENTOS TEÓRICOS Y PRÁCTICOS PARA LA IMPLEMENTACIÓN DE LA RED PROTOTIPO GSM/GPRS.

El presente capítulo aborda los conceptos generales a tener en cuenta antes de la implementación del prototipo de red propuesto. La sección 2.1 describe la arquitectura de red GSM/GPRS tradicional y sus componentes más importantes. La sección 2.2 explica el funcionamiento de esta red abordando principalmente los procedimientos necesarios para el establecimiento del servicio de transferencia de datos entre un dispositivo final y una PDN<sup>1</sup> externa.

La sección 2.3 describe el hardware y software utilizado para implementar un modulo de comunicaciones típico con soporte GPRS de un nodo sensor, o dispositivo final, y se describe la programación realizada en el módulo para establecer una conexión GPRS y transferir paquetes de datos a través del prototipo de red GPRS implementado.

La sección 2.4 describe a la plataforma de software OpenBTS así como también diversas configuraciones relacionadas a su funcionamiento y el monitoreo de eventos.

Finalmente, la sección 2.5 describe la tecnología SDR y la plataforma de hardware que se utiliza junto con OpenBTS para implementar el prototipo de red GPRS propuesto.

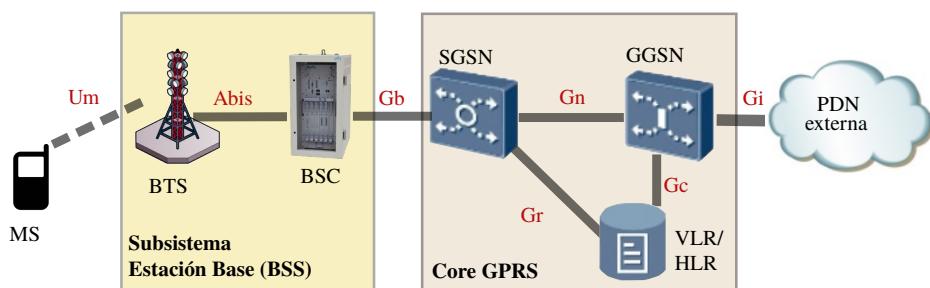


Fig. 2.1: Esquema de una red GPRS sobre una arquitectura de red GSM.

<sup>1</sup>PDN: *Packet data network*, o red de paquete de datos

## 2.1 Arquitectura de red GSM/GPRS tradicional.

La tecnología GSM fue desarrollada originalmente para servicios de voz y sms. Posteriormente, se implementó como funcionalidad adicional la tecnología GPRS, lo que permitía la transferencia de datos IP sobre GSM.

La Figura 2.1 muestra las entidades de una red GSM con soporte GPRS [11]; no se consideran las entidades relacionadas con los servicios de voz y sms, tales como el MSC<sup>2</sup>. Las entidades que componen el segmento del acceso a la red GPRS son:

1. **MS**: de *mobile station* o estación móvil, es el dispositivo final el cual posee un módulo de comunicaciones con soporte GPRS. Existen 03 clases de dispositivos MS según el modo de operación [12]:

- Clase A, es capaz de soportar servicios de voz y datos simultáneamente.
- Clase B, soporta ambos servicios de voz y datos, pero no simultáneamente.
- Clase C, exclusivamente opera servicios de datos GPRS.

**Nota:** En adelante, toda vez que se cite el término MS se referirá al dispositivo final o nodo sensor, y en particular a su módulo de comunicaciones GPRS, el cual soporta modo de funcionamiento Clase B.

2. **BTS**: de *base transceiver station* o estación base transceiver<sup>3</sup>, es el componente de radio que tiene como función principal crear la interfaz de radio con el MS. El BSC, de *base station controller* o estación base controladora, es la entidad que controla a las **BTS** y en conjunto conforman el subsistema de estaciones base (**BSS**), la cual es la entidad de la Figura 2.1 que provee el acceso a red a los MS a través de la interfaz aire Um y los conecta con el SGSN a través de la interfaz Gb [13].

Una vez que se establece el enlace de radio, el BSS y el MS intercambian radio-paquetes. Estos radio-paquetes provienen o van hacia el core GPRS, en donde se administra el encaminamiento de paquetes de datos a nivel de protocolo IP. El core GPRS está compuesto por las siguientes entidades:

3. **SGSN**: de *service gateway support node* o nodo servidor GPRS, es la principal entidad del core GPRS; gestiona la ubicación de cada MS a nivel de BSS y participa en el

---

<sup>2</sup>MSC: *Mobile switching center*, o centro de commutación móvil.

<sup>3</sup>Transceiver es el término en inglés para el componente capaz de transmitir y recibir señales de radio.

control del acceso a la red. El SGSN es el encargado de establecer los contextos de información que sirven para identificar la conexión de datos de un determinado MS con el GGSN [13].

4. **GGSN:** de *gateway GPRS support node* o nodo gateway GPRS, es la entidad que hace de puerta de enlace con las PDNs externas y usa el contexto creado por el SGSN para proveer a los MS el respectivo encaminamiento de paquetes de datos hacia PDN externas [13].
5. **HLR/VLR:** de *home/visitor location register* o registro de ubicación de usuarios locales/visitantes, es la entidad que hace de base de datos dinámica la cual contiene los registros de conexión de un MS a nivel de BSS [13]. También contiene información sobre los servicios disponibles que tiene permitido un determinado MS en la red.

## **2.2 Establecimiento del servicio de transferencia de datos entre un MS y una PDN externa sobre una red GSM/GPRS.**

En esta sección se procede a describir los procedimientos que el MS y la red (BSS, SGSN, y GGSN) realizan a fin de establecer el servicio de transferencia de datos GPRS entre el MS y una PDN externa. Estos conceptos son importantes pues, al momento de estudiar el funcionamiento del prototipo de red GPRS propuesto, se hace referencia a los procedimientos expuestos en esta sección. En la Figura 2.2, se presentan las principales entidades de la red GPRS de la Figura 2.1 que están involucradas directamente con el establecimiento de una conexión de datos entre un MS y una PDN externa.

En resumen, ya sea que el MS precise enviar paquetes de datos a alguna PDN o viceversa, el MS inicialmente debe establecer un enlace de radio con el BSS para intercambiar señalización. Luego, el MS solicita al SGSN activar los mecanismos necesarios entre las entidades de red para establecer una especie de túnel *end-to-end* desde el MS hasta el GGSN más apropiado, de modo que los paquetes de datos del MS encuentren la PDN destino.

Entre estos mecanismos se encuentran el establecimientos de contextos de información entre el SGSN y el GGSN para crear un túnel dedicado en el segmento SGSN-GGSN para el MS solicitante. Otro mecanismo que el SGSN activa son las instrucciones para que el MS y el BSS reconfiguren el radioenlace con nuevos recursos de radio que provisionen la suficiente capacidad de red que el tráfico a intercambiarse necesite.

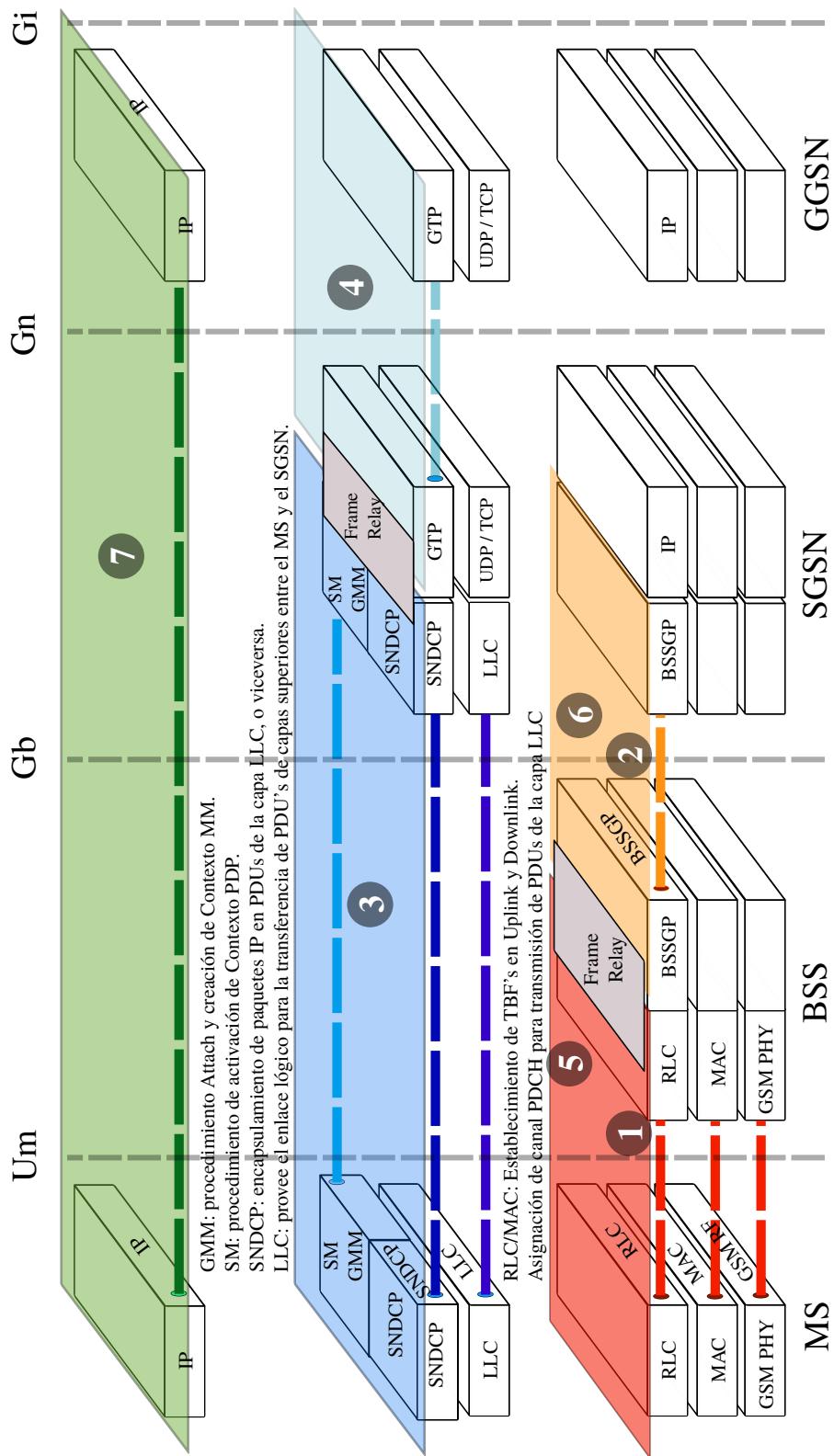


Fig. 2.2: Estructura de capas e interfaces del *stack* de protocolos GSM/GPRS.

Las siguientes secciones presentan los conceptos para entender la secuencia de conexiones y procedimientos que se presentan en la Figura 2.2.

### 2.2.1 Modulación de señales en una red GPRS.

En la capa física de la Figura 2.2, la tecnología de modulación de señales es GMSK<sup>4</sup>, la cual es un esquema de mínimos desplazamientos gaussianos de la frecuencia sin discontinuidades en la fase. El período (T) de cada símbolo es de  $3.692 \mu s$  y la tasa de modulación es de 270.833 ksps lo que corresponde a una tasa de datos de 270.833 kbps. Se utiliza una codificación diferencial para transformar los bits a símbolos [14].

La portadora RF de la señal modulada en GMSK se expresa como una señal *coseno* de la forma [14]:

$$x(t') = \sqrt{\frac{2E_c}{T}} \cos(2\pi f_0 t' + \varphi(t') + \varphi_0) \quad (2.1)$$

Donde  $E_c$  es la energía por bit modulado,  $f_0$  es la frecuencia central o portadora y  $\varphi_0$  es la fase aleatoria y que es constante durante un *burst* o ráfaga de símbolos.

El parámetro que contiene la información codificada es la fase  $\varphi(t')$ , la cual está definida en función a la señal banda base  $g(t)$ , de acuerdo a la siguiente definición [14]:

$$\varphi(t') = \sum_i \alpha_i \pi h \int_{-\infty}^{t'-iT} g(u) du \quad (2.2)$$

Donde el índice de modulación  $h$  es 1/2 (dado que el máximo cambio por fase en radianes es  $\pi/2$  por cada intervalo de dato). La referencia de tiempo  $t' = 0$  corresponde al inicio de la transmisión [15].

La función  $g(t)$  es la señal banda base  $s(t)$  luego de haber pasado a través de un filtro  $h(t)$  de acuerdo a la siguiente relación [14]:

$$g(t) = h(t) * s(t) \quad (2.3)$$

Donde

$$s(t) = \begin{cases} \frac{1}{T}, & |t| < \frac{T}{2} \\ 0, & \text{otro valor} \end{cases}$$

$$h(t) = \frac{\exp(\frac{-t^2}{2\delta^2 T^2})}{\sqrt{(2\pi).\delta T}}$$

---

<sup>4</sup>GMSK: *Gaussian minimum shift keying*, o desplazamiento mínimo gaussiano.

$$\gamma = \frac{\sqrt{\ln(2)}}{2\pi BT}$$

Donde:

$B$  : es el ancho de banda 3dB del pulso

$T$  : es el período del símbolo, de acuerdo a [16]

$BT : 0.3$

La Figura 2.3 (izquierda) muestra el espectro de frecuencias de una señal GMSK correspondiente a la ecuación 2.1. También se muestra (derecha) los símbolos que llegan a un receptor GMSK. La información está codificada en la diferencia de fase que existe entre símbolos consecutivos. Un símbolo es una porción de señal transmitida en un determinado *tiempo de símbolo* y se representan por los puntos azules los cuales se muestran en un diagrama I/Q.

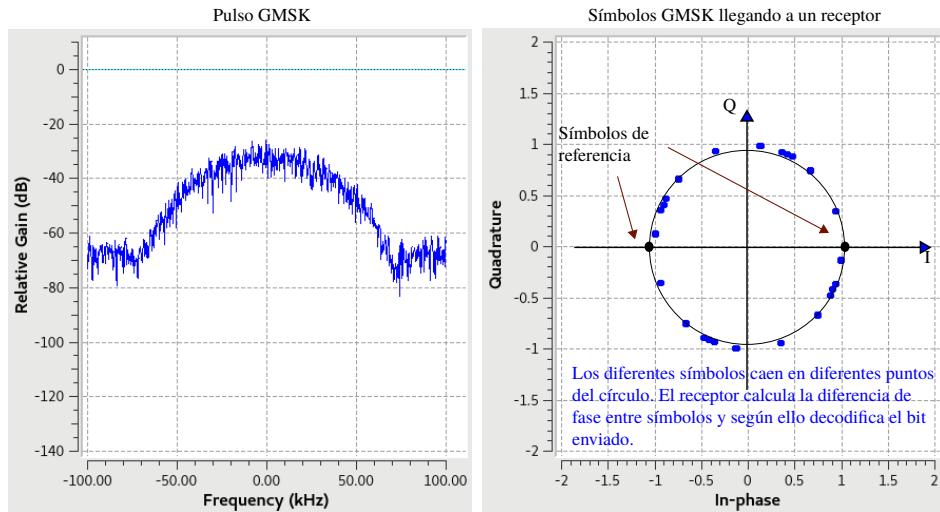


Fig. 2.3: Modulación GMSK y su respectiva constalación de símbolos.

Una ráfaga o *burst* es la transmisión de una determinada cantidad de símbolos durante un lapso de tiempo denominado *timeslot*. Por lo tanto, un *burst* representa el contenido físico de una señal en un *timeslot* [15].

La Figura 2.4 muestra las diferentes configuraciones de ráfagas, dependiendo del tipo de señal que se requiera transmitir. Por ejemplo, la ráfaga del tipo normal se usa para transporte de data y contiene 116 símbolos para transportar *payload* de capa superior; la ráfaga para corrección del desplazamiento en frecuencia contiene 142 símbolos; la ráfaga para corrección de sincronización en el tiempo de trama contiene 78 símbolos; y finalmente, la ráfaga para el

acceso aleatorio, la cual el MS emplea para anunciararse a la red y subsecuentemente acceder al servicio GPRS [15], [17].

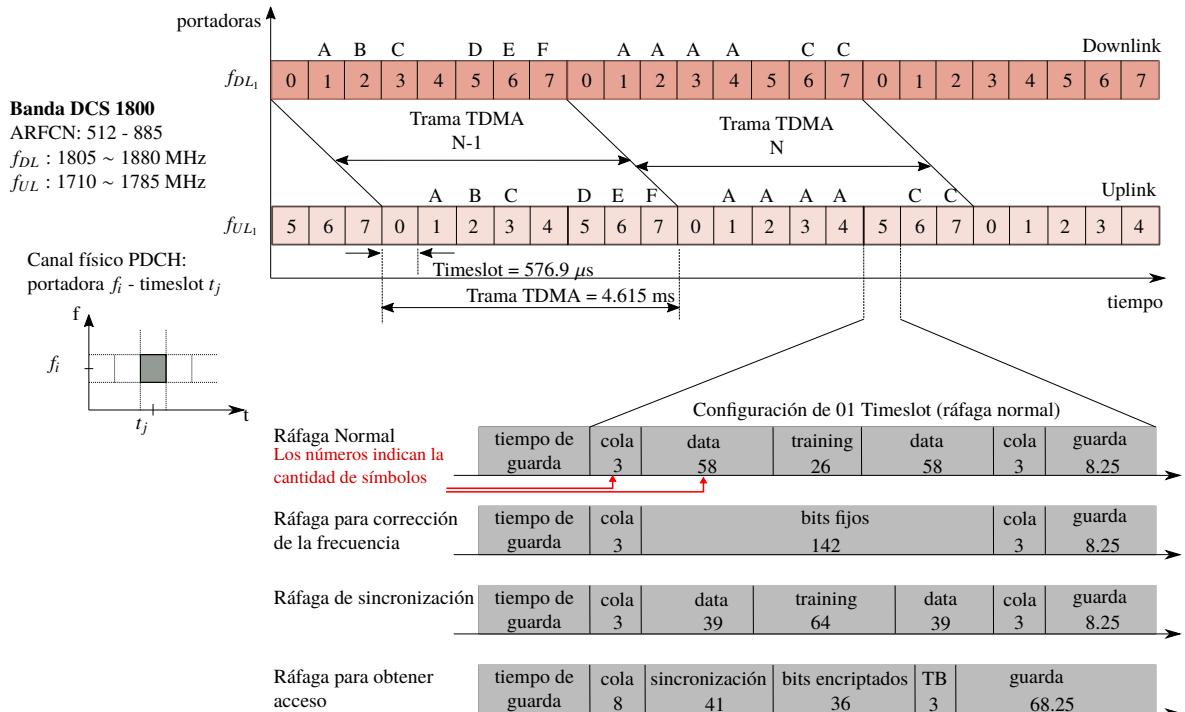


Fig. 2.4: Estructura de la trama TDMA.

### 2.2.2 Esquemas de acceso TDMA y FDMA.

Cada una de estas ráfagas puede ser transmitida en un *timeslot* cuya duración es de aproximadamente 576,9  $\mu$ s, o lo que es equivalente a 156,25 símbolos GMSK. La sucesión de 08 *timeslots* conforman una trama TDMA.

Con respecto a la frecuencia, los canales se distribuyen en esquema FDMA<sup>5</sup>, que consiste en utilizar un canal ARFCN<sup>6</sup> el cual está compuesto de 02 canales de radiofrecuencia, o también denominados portadoras, de 200kHz de ancho de banda, destinados a ser los canales *downlink* y el *uplink* de acuerdo con la Tabla 2.1 [16].

El esquema TDMA<sup>7</sup> consiste en compartir un determinado canal ARFCN en diferentes *timeslots*. Esta repartición de los recursos frecuencia y tiempo permite que puedan ser multiplexados en una misma portadora ARFCN hasta 08 MS.

El canal físico (PCH<sup>8</sup>) se refiere a la dualidad *portadora-timeslot* y cuando el canal

<sup>5</sup>FDMA: Frequency division multiple access en lo que se denomina como canales ARFCN.

<sup>6</sup>ARFCN: Absolute radio frequency channel number, o número de canal absoluto de radiofrecuencia.

<sup>7</sup>TDMA: Time division multiple access, o acceso múltiple por división del tiempo.

<sup>8</sup>PCH: Physical channel, o canal físico.

físico se usa para transportar datos<sup>9</sup> se denomina canal de paquete de datos (PDCH<sup>10</sup>). La red puede asignar uno o más canales físicos PDCH a un MS (en *downlink*, en *uplink*, o en ambos) dependiendo de los requerimientos de capacidad de red y de la configuración *multi-slot* definidas en la red.

TABLA N° 2.1: ARFCNs (n) de cada banda GSM.

Banda	Uplink	ARFCN (n)	Downlink
P-GSM 900	$F_{dl} = 890 + 0.2n$	$1 \leq n \leq 124$	$F_{ul} = F_{dl} + 45$
E-GSM 900	$F_{dl} = 890 + 0.2n$ $F_{dl} = 890 + 0.2(n - 1024)$	$0 \leq n \leq 124$ $975 \leq n \leq 1023$	$F_{ul} = F_{dl} + 45$
R-GSM 900	$F_{dl} = 890 + 0.2n$ $F_{dl} = 890 + 0.2(n - 1024)$	$0 \leq n \leq 124$ $955 \leq n \leq 1023$	$F_{ul} = F_{dl} + 45$
DCS 1800	$F_{dl} = 1710.2 + 0.2(n - 512)$	$512 \leq n \leq 885$	$F_{ul} = F_{dl} + 95$
PCS 1900	$F_{dl} = 1850.2 + 0.2(n - 512)$	$512 \leq n \leq 810$	$F_{ul} = F_{dl} + 80$
GSM 450	$F_{dl} = 450.6 + 0.2(n - 259)$	$259 \leq n \leq 293$	$F_{ul} = F_{dl} + 10$
GSM 480	$F_{dl} = 479 + 0.2(n - 306)$	$306 \leq n \leq 340$	$F_{ul} = F_{dl} + 10$
GSM 850	$F_{dl} = 824.2 + 0.2(n - 128)$	$128 \leq n \leq 251$	$F_{ul} = F_{dl} + 45$

La configuración *multi-slot* consiste en agrupar múltiples canales PDCH para un mismo MS y su especificación se encuentra en la sección 6.4.2.2 de [15]. Por ejemplo, en la Figura 2.4, en la trama TDMA (N-1) se asigna 01 timeslot por cada usuario (A, B, C, D, E, y F), mientras que en la trama (N), una configuración *multi-slot* de 04 timeslots (*ts*: 1, 2, 3 y 4) es asignado al usuario A.

Se debe precisar que las tramas TDMA correspondientes al canal *uplink* y *downlink* de un determinado canal ARFCN están retrazadas temporalmente una con respecto de la otra, a fin de dar suficiente tiempo al MS para que pueda procesar la información recibida en el *downlink* y responder en la misma trama *uplink* si fuera necesario. Según [15], las tramas *uplink* están retrasadas 03 *timeslots* respecto de las tramas *downlink* como se muestra en la Figura 2.4.

### 2.2.3 Mapeo de canales lógicos en canales físicos.

En la Figura 2.5 se muestran todos los canales lógicos relacionados con el servicio de datos, los cuales son mapeados en diferentes canales físicos PDCH dependiendo de la portadora y la posición en la trama que le corresponda al canal lógico.

Los canales lógicos PCCCH<sup>11</sup>, PBCCH<sup>12</sup>, y PDCCH<sup>13</sup> son considerados canales de

<sup>9</sup>En GSM, el canal físico originalmente fue diseñado para transportar tráfico de voz o sms.

<sup>10</sup>PDCH: *Packet data channel*, o canal de paquetes de datos.

<sup>11</sup>PCCCH: *Packet common control channel*, o canal de control común para paquetes.

<sup>12</sup>PBCCH: *Packet broadcast control channel*, o canal *broadcast* de control para paquetes.

<sup>13</sup>PDCCH: *Packet downlink control channel*, o canal de control *downlink* para paquetes.

señalización ya que están involucrados en el intercambio de información de control entre el MS y la red. El canal de tráfico de paquetes PTCH<sup>14</sup> es dedicado exclusivamente al transporte de PDUs<sup>15</sup> de capas superiores.

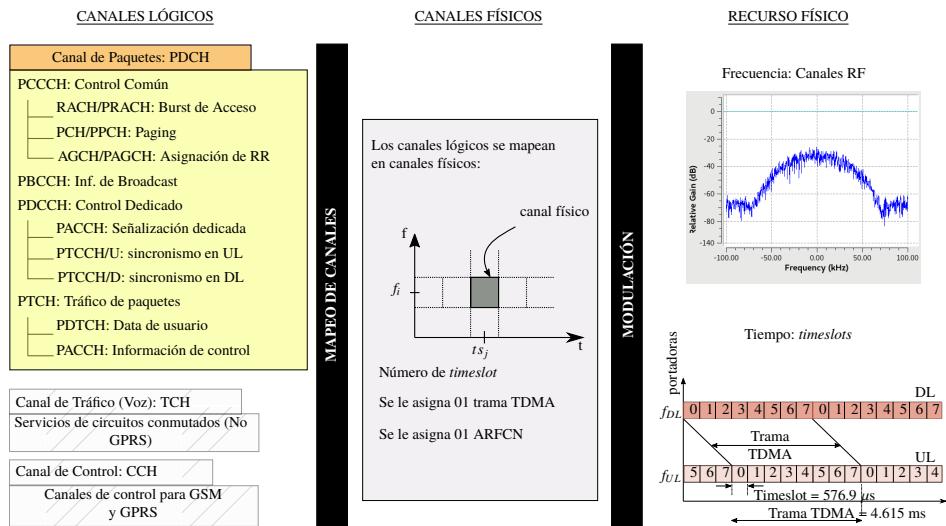


Fig. 2.5: Concepto de mapeo de canales lógicos en canales físicos.

La implementación más común es el uso de un canal ARFCN para servicios de voz y otro ARFCN para servicios de datos. El canal PCCCH contiene información sobre la configuración de servicios por cada canal ARFCN además de la estructura de mapeo de canales lógicos en canales físicos para cada trama TDMA. Las diferentes configuraciones de mapeo de canales lógicos en canales físicos de una red GSM/GPRS se encuentran en la sección 6.4 de [15].

#### 2.2.4 Inicio de conexión del dispositivo final.

Suponiendo que el MS acaba de ser energizado, tal como se muestra en la Figura 2.6, este no cuenta con ningún canal PDCH asignado, de modo que no puede realizar ninguna transmisión o recepción de datos.

El MS debe primero conectarse a la red a nivel de GSM y a partir de ello, realizar la conexión a nivel GPRS. Como procedimiento inicial, el MS realiza una búsqueda de *beacons*<sup>16</sup> sobre cada uno de los canales ARFCN de todas las bandas GSM que soporta.

El MS asume que el *beacon* de mayor potencia pertenece a una estación base cercana y busca en dicha señal el canal de sincronización SCH contenido en el canal BCCH. Si no

<sup>14</sup>PTCH: Packet traffic channel, o canal de tráfico de paquetes.

<sup>15</sup>PDU: Protocol data unit, o unidad de protocolo de datos. Es la unidad de información correspondiente a cada protocolo de cada capa.

<sup>16</sup>Señal de broadcast de una estación base.

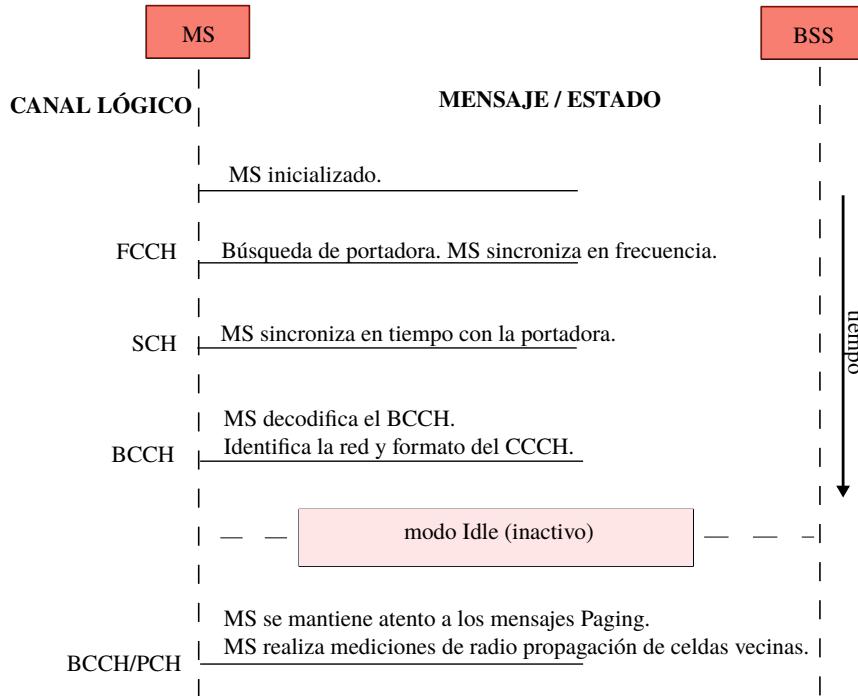


Fig. 2.6: Inicialización del MS, sincronismo en frecuencia/tiempo, y decodificación de canales BCCH y CCCH.

consigue decodificar el SCH, asume que dicha señal no es GSM y continua la búsqueda sobre el *beacon* de siguiente mayor potencia.

Una vez que el MS detecta un *beacon* válido, el MS decodifica el SCH para obtener el inicio de la trama TDMA en ambos canales *downlink* y *uplink*. Luego de la sincronización, el MS decodifica el canal GSM de *broadcast* BCCH y obtiene información de la red, los servicios que soporta, el formato de multiplexación del canal de control común (CCCH), etc. [18], [15], [19].

Una vez que conoce el formato del canal CCCH, el MS sabe cuál es el canal físico para enviar a la red las solicitudes *Packet Channel Request* con el fin de solicitar canales físicos dedicados exclusivamente a la transmisión o recepción de paquetes de datos.

De igual forma, el MS conoce cuál es el canal físico por el cual la red envía los mensajes tipo *paging*<sup>17</sup>, los cuales sirven para informar al MS que debe establecer una conexión de radio con el BSS porque la red precisa entregarle paquetes de datos provenientes de una PDN externa [20].

<sup>17</sup>Paging: Procedimiento que permite la localización de un MS dentro de una área de enrutamiento (RA). El RA es un conjunto de estaciones base que por lo general tienen un área geográfica en común.

### 2.2.5 Acceso por radio a la red GSM/GPRS.

A nivel de radio, la conexión entre el MS y el BSS puede describirse en 03 modos de operación. Estos modos están relacionados con los recursos de radio que un MS tiene asignado en un determinado instante. Los modos de operación son: *Idle*, *Dedicated*, y *Packet Transfer* y se muestran en la Figura 2.7.

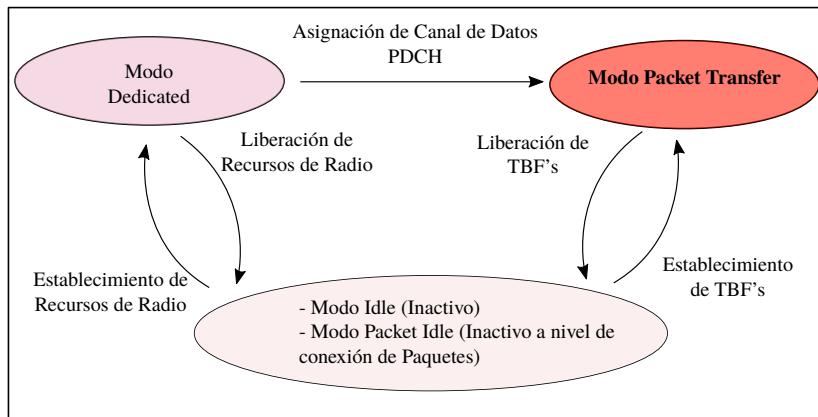


Fig. 2.7: Modos de Operación a nivel de Asignación de Recursos de Radio.

Luego de sincronizarse con la red, si el MS no precisa enviar datos y no recibe ningún *paging* por parte de la red, entonces entra un modo de operación denominado *Idle* o inactivo [21]. En este modo, el MS desactiva sus circuitos del transmisor y receptor y no cuenta con ningún recurso de radio.

Si por el contrario, el MS debe enviar datos o recibe un *paging*, este debe crear TBF<sup>18</sup> con el BSS, en ambos canales *uplink* y *downlink*. Un TBF es un flujo temporal de radio-bloques que se establece tanto en el canal *uplink*, cuando el MS envía paquetes a la red, como en el canal *downlink*, cuando la red envía paquetes al MS.

Para establecer un TBF, el MS debe pasar del modo *Idle* al modo *Packet Transfer* [21] mediante el intercambio de mensajes mostrado en la Figura 2.8. El MS envía un mensaje *Channel Request* o *Packet Channel Request* al BSS, para solicitar que se le asigne un canal PDCH para intercambiar señalización. El BSS responde al MS con un mensaje *Immediate Assignment* o *Packet Uplink Assignment* informándole al MS los recursos de radio asignados.

Con los recursos de radio asignados, el MS envía el mensaje *Packet Resource Request* por el cual solicita recursos de radio orientados a establecer un TBF en el sentido uplink con el BSS [21], [20]. La red responde con el mensaje *Packet Uplink Assignment* en el cual indica

<sup>18</sup>TBF: *Temporal block flow*, o flujo temporal de bloques.

al MS los canales PDCH asignados. En este punto el MS está en el modo *Packet Transfer* de la Figura 2.7.

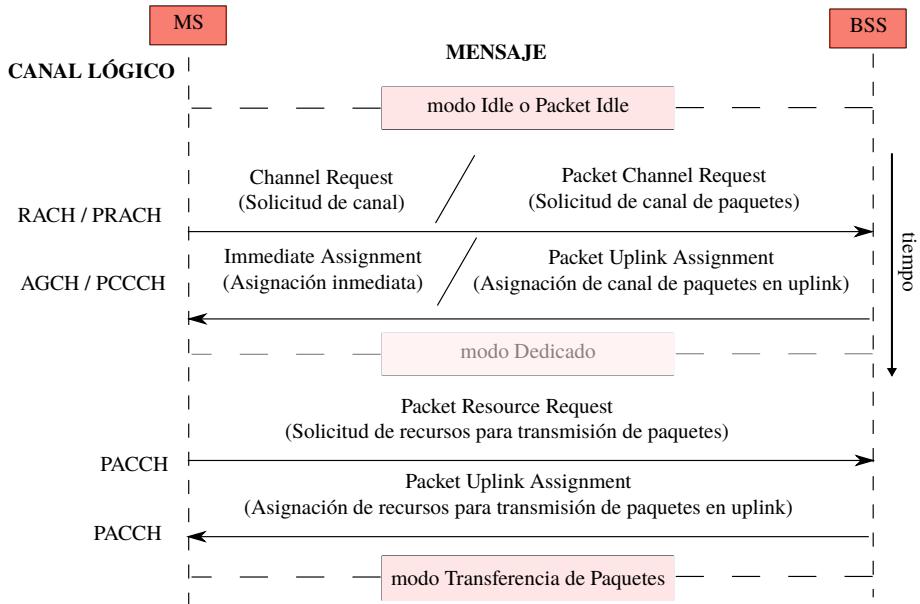


Fig. 2.8: Señalización para el establecimiento de un TBF en el canal uplink.

En el modo *Packet Transfer*, el MS ya puede transmitir los PDU de capas superiores en forma de bloques de ráfagas de símbolos, o radio-bloques. Una vez que el MS no necesite enviar más data, se procede a la liberación de los recursos que constituyen el TBF y el MS pasa nuevamente al estado *Idle*; no obstante, siempre verificando periódicamente el canal *paging* [20].

Cuando el MS necesite nuevamente enviar data entonces volverá a establecer un TBF con la red. Si el MS inicialmente se encuentra en modo *Dedicated* (Figura 2.7), la red sólo reconfigura los recursos de radio asignando nuevos PDCHs al MS [21].

La Figura 2.9 muestra el proceso por el cual el MS pasa al modo *Packet Transfer*, activado por un *Packet Paging Request* proveniente de la red, es decir, el establecimiento del TBF es inicialmente en el *downlink* debido a que es la red quien necesita entregar paquetes de datos al MS.

Luego del *Paging Request*, la red envía un *Immediate Assignment* o *Packet Downlink Assignment* donde básicamente se le especifica al MS los recursos de radio que asignando para establecer el TBF en el *downlink*. Los siguientes mensajes son respuestas ACK<sup>19</sup>, mensajes de configuración de potencia de transmisión, y de ajuste del *timing advance*<sup>20</sup> [21].

<sup>19</sup>Respuesta del tipo *acknowledgment*, también denominado mensaje de acuse de recibo.

<sup>20</sup>*Timing advance*: es el ajuste temporal que debe realizar el MS para estar sincronizado con la red.

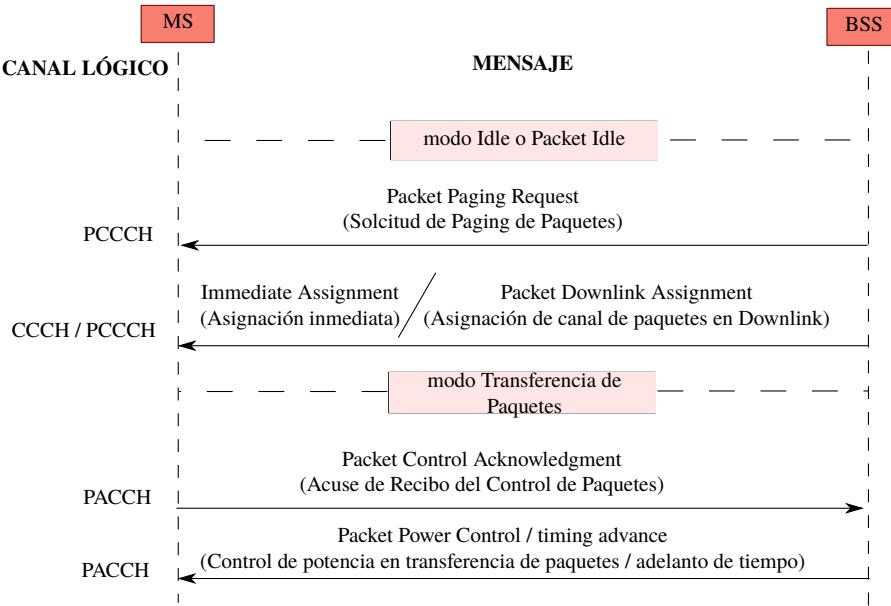


Fig. 2.9: Señalización para el establecimiento de un TBF en el canal downlink.

De la Figura 2.2, la conexión 1 se realiza estableciendo los TBF tanto en *downlink* y *uplink* entre el MS y el BSS. Esta conexión de radio más la conexión troncal a nivel de protocolo BSSGP entre el BSS y el SGSN (conexión 2) proporcionan el soporte para establecer el enlace lógico a nivel de protocolo LLC entre el MS y el SGSN [22].

Sin embargo, la conexión lógica LLC entre el MS y el SGSN aún no está en posibilidades de ofrecer servicios de transporte a los PDUs de capas superiores. Para ello, las subcapas GMM [23], SM [20], y SNDCP [24] deben realizar procedimientos que en conjunto hacen posible que el MS establezca un túnel lógico hasta el GGSN.

## 2.2.6 Establecimiento de conexión a nivel GPRS.

De la Figura 2.2, el protocolo GMM<sup>21</sup> es el que se encarga de la gestión de movilidad a nivel GPRS de los MS, es decir, conoce la localización de los MS a fin de poder ubicarlos cuando la red tenga que entregar datos provenientes de alguna red externa.

Para que el MS acceda al servicio de transferencia de datos vía GPRS, primero es necesario que se establezca el contexto de información GMM, que básicamente es información a nivel de capa GMM que permite asociar a un MS con un BSS [11].

A nivel de capa GMM, los estados del MS se muestran en la Figura 2.10. Luego de pasar al modo *Packet Transfer* a nivel de conexión por radio, el MS automáticamente inicia en estado GMM–NULL a nivel de protocolo GMM. En estado GMM–NULL, el MS no puede

<sup>21</sup>GMM: *GPRS Mobility Management*, o administración de la movilidad a nivel de conexión GPRS.

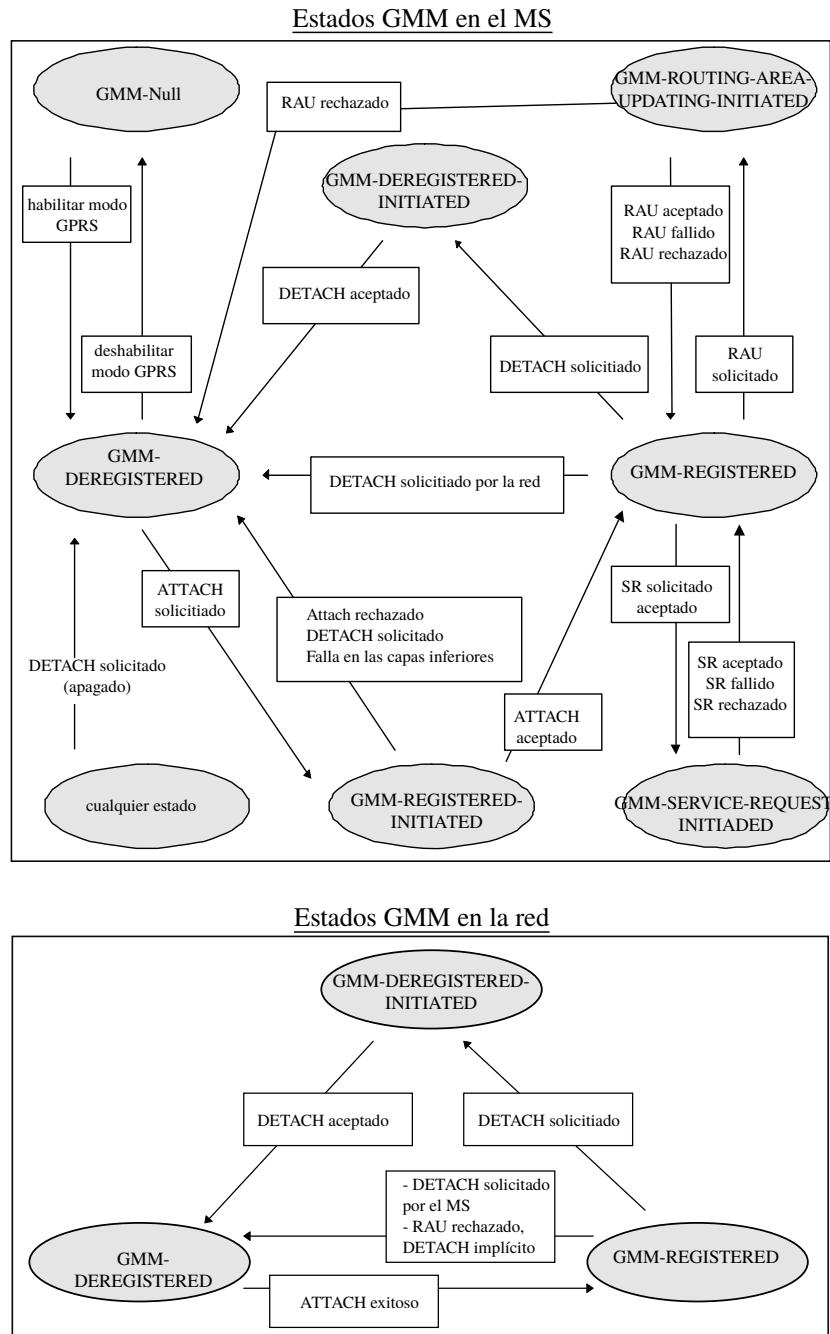


Fig. 2.10: Estados y transiciones propias de la gestión de la movilidad GPRS (GMM) en el MS y en la red.

realizar ninguna tarea relacionada a la administración de la movilidad GMM, por lo tanto, no puede aún establecer un túnel con el SGSN.

Cuando el MS habilita el modo GPRS en su sistema, este pasa al estado GMM-DEREGISTERED en el cual las capacidades GPRS del MS están habilitadas pero no existe aún ningún contexto GMM establecido [23]. Desde el estado GMM-DEREGISTERED, el MS inicia el procedimiento *GPRS Attach* mostrado en la Figura 2.11 y pasa al estado GMM-REGISTERED-INITIATED

mientras espera una respuesta por parte de la red.

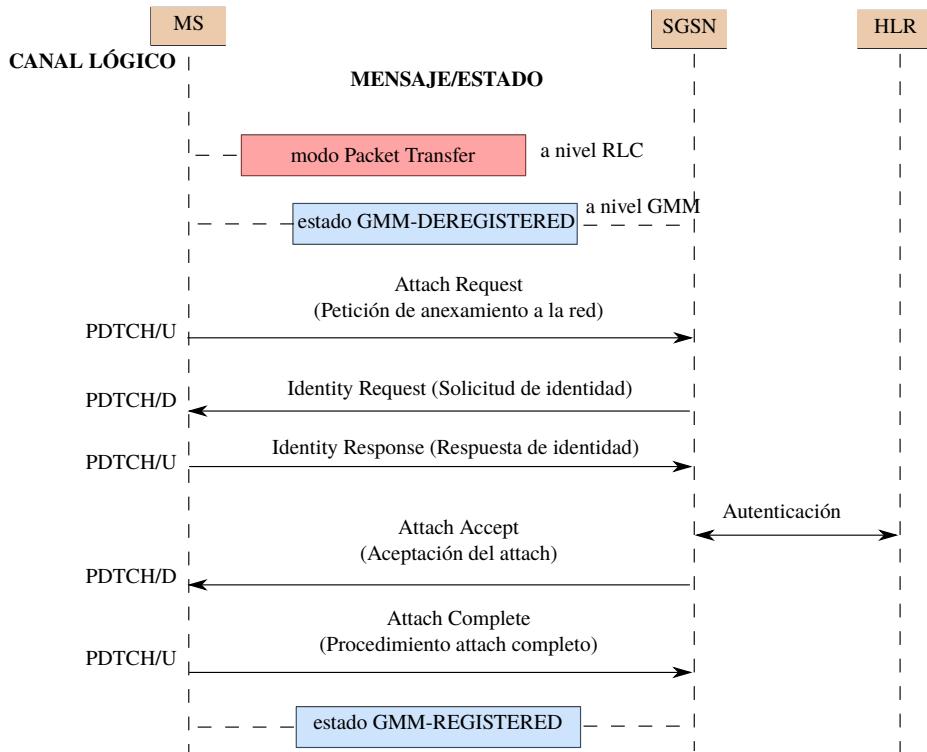


Fig. 2.11: Intercambio de mensajes propio del procedimiento *GPRS Attach*.

Una vez que la red acepta el *Attach Request*, el MS pasa al estado GMM-REGISTERED, con lo cual se establece un contexto de información a nivel de capa GMM entre el MS y el SGSN. Este punto es importante en toda la secuencia de conexión, pues una vez que el MS esté en estado GMM-REGISTERED, este ya puede iniciar el establecimiento de lo que se denomina como contexto PDP<sup>22</sup> con la PDN externa a través del SGSN y el GGSN. Un contexto PDP es un conjunto de información que vinculan un MS con una PDN externa, de modo que es posible el enrutamiento de paquetes de datos entre ambos.

El procedimiento *GPRS Attach* inicia con el MS en modo *Packet Transfer* a nivel de radio y GMM-DEREGISTERED a nivel de capa GMM. El MS envía un mensaje *Attach Request* al SGSN con información como la causa de la solicitud de *attach*<sup>23</sup> a la red GPRS. Luego, el SGSN responde con un mensaje *Identity Request* consultando la identidad del MS. En respuesta, el MS envía su código IMSI<sup>24</sup> en el mensaje *Identity Response* [20]. Con el IMSI, el SGSN realiza consultas al HLR sobre la autenticidad de la información proporcionada por

<sup>22</sup>PDP: *Packet data protocol*, o protocolo de paquetes de datos.

<sup>23</sup>El término *attach* se refiere a establecer una conexión a nivel de capa GMM entre el MS y el SGSN.

<sup>24</sup>IMSI: International Mobile Subscriber Identity, o identidad internacional de subscriptor móvil. Es el identificador único de la línea o servicio.

el MS, e implícitamente el HLR queda informado sobre la ubicación del MS.

Si el MS cuenta con los permisos y accesos necesarios en el HLR, el SGSN envía un mensaje *Attach Accept* para informar al MS que el procedimiento ha sido exitoso y que el SGSN ha pasado al estado GMM-REGISTERED para dicho MS. Finalmente, el MS responde a la red con un mensaje *Attach Complete* con el cual notifica haber pasado al estado GMM-REGISTERED [23].

El procedimiento *GPRS Attach* ha creado un contexto de movilidad GMM entre el MS y SGSN, con el cual el SGSN sabe cómo encaminar los paquetes de datos hasta el MS. En este punto, se ha establecido la conexión 3 de la Figura 2.2. El siguiente paso se lleva a cabo en subcapa SM (*Session Management* o gestión de la sesión) la cual se encarga de la activación, modificación, y desactivación del contexto PDP [20].

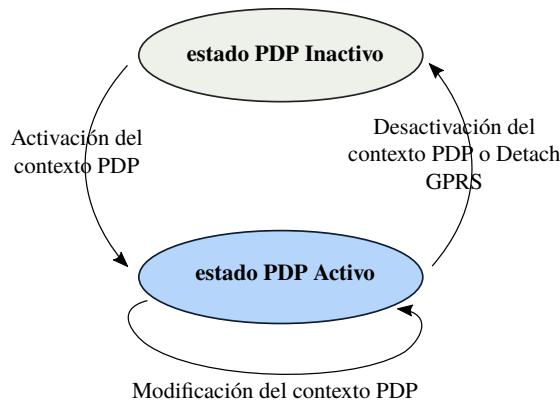


Fig. 2.12: Estados relacionados al establecimiento del Contexto PDP a nivel de capa SM.

### 2.2.7 Establecimiento del contexto PDP.

La siglas PDP provienen de *packet data protocol* y se refieren al protocolo que gestiona el flujo de paquetes de datos entre el MS, SGSN, GGSN y la PDN externa. El contexto PDP es el conjunto de información necesaria para hacer posible el enrutamiento de los paquetes entre el MS y alguna PDN externa. La principal información del contexto PDP es la dirección IP del MS; de esta manera el GGSN conoce cuál es el SGSN adecuado para ubicar a un determinado MS [23].

Entre el SGSN y el MS, los procedimientos relacionados con la creación, modificación, y eliminación de contextos PDP son realizados por la capa SM en la conexión MS - SGSN, y por la capa GTP en la conexión SGSN - GGSN. La Figura 2.12 muestra los estados a nivel de contexto PDP y las respectivas transiciones.

Un contexto PDP en estado PDP INACTIVO significa que el contexto PDP no se ha establecido y por lo tanto no existe información que permita el enrutamiento de paquetes entre el MS y la PDN externa, por lo tanto, la transferencia de datos no es posible.

Un estado PDP ACTIVO significa que el contexto PDP se ha activado en el MS, el SGSN, y el GGSN y todos ellos conocen la dirección IP del MS para el correcto enrutamiento de paquetes entre el MS y el GGSN. En este estado, la transferencia de datos ya es posible.

El procedimiento para crear un contexto PDP puede ser iniciado por el MS, cuando este precisa transferir datos hacia una PDN externa; o por la red, cuando esta precisa entregar al MS data proveniente de una PDN externa.

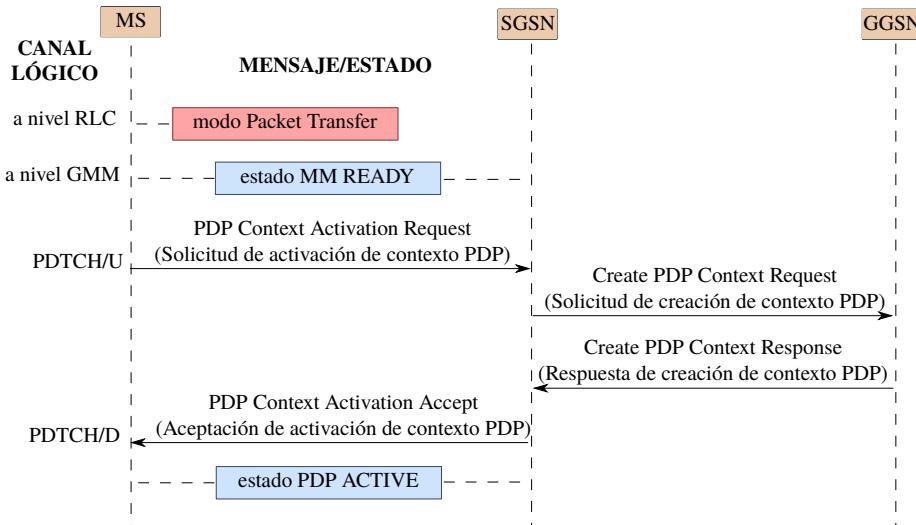


Fig. 2.13: Intercambio de mensajes entre MS, SGSN, y GGSN para el establecimiento de un contexto PDP, iniciado por el MS.

La Figura 2.13 muestra la creación de un contexto PDP iniciada en el MS. El MS envía un mensaje *PDP Context Activation Request* o solicitud de activación de contexto PDP al SGSN con información relevante como lo es la dirección IP del MS y la dirección IP destino.

El SGSN y el GGSN crean nuevas entradas en sus respectivas tablas de contextos PDP para encaminar los paquetes IP entre sí. El GGSN responde al SGSN el resultado de la creación del contexto PDP. Lo siguiente es que el SGSN envía un mensaje *PDP Context Activation Accept* al MS informándole de la activación del contexto PDP tanto en el SGSN como en el GGSN.

La Figura 2.14 muestra la señalización de mensajes cuando la red recibe un paquete IP proveniente de una red externa dirigido a un MS de la red. El GGSN verifica si existe un contexto PDP establecido para la dirección PDP a la que apunta el mensaje entrante. Si no

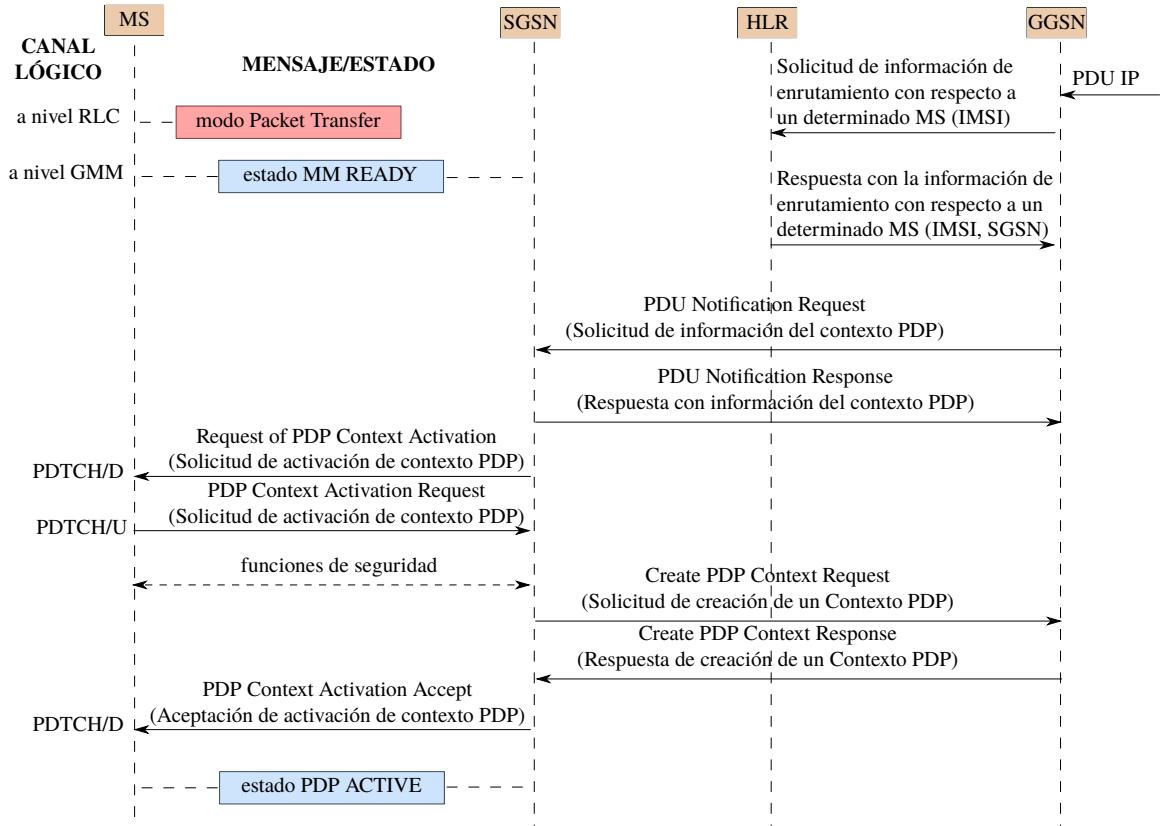


Fig. 2.14: Intercambio de mensajes entre MS, SGSN, y GGSN para el establecimiento de un contexto PDP, iniciado por la red.

existe, el GGSN envía un mensaje *PDU Notificacition Request* al SGSN.

El SGSN notifica al MS que hay paquetes de datos para serle entregados y el MS inicia el procedimiento para subsecuentemente activar el contexto PDP que le permita recibir los paquetes de datos. El SGSN vuelve a establecer un contexto PDP con el GGSN y luego envía al MS la confirmación de la activación del contexto PDP.

### 2.3 Descripción del dispositivo final.

En la presente tesis, el término MS se refiere al dispositivo final o nodo sensor, el cual cuenta con un módulo de comunicaciones con soporte GPRS. El módulo de comunicaciones se compone principalmente del chip SIM800L [25] el cual se muestra en la Figura 2.15 y cuyos pines de interés se indican en la Tabla 2.2.

El SIM800L es un chip de procesamiento de señales con soporte de conectividad GSM, GPRS, y GPS del fabricante SIMCom. El SIM800L soporta conexiones con redes GS-M/GPRS en las 04 bandas de telefonía celular GSM a nivel mundial (GSM800, EGSM900, DCS1800, PCS1900) [26].

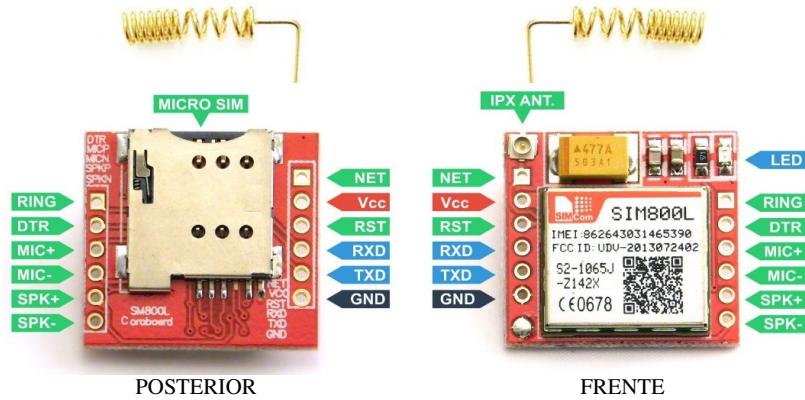


Fig. 2.15: Módulo de comunicaciones GPRS - SIM800L.

TABLA N° 2.2: Esquemático de Pin's.

Pin	I/O	Descripción
NET	-	Pin para soldar la antena
Vcc	I	Pin para alimentación eléctrica
RST	I	Pin para hacer <i>reset</i> del módulo
RXD	I	Recepción de data serial
TXD	O	Transmisión de data serial
GND	I	Pin para el aterramiento o <i>ground</i>

El diagrama de bloques funcionales del módulo SIM800L se muestra en la Figura 2.16.

El frente RF contiene los componentes para filtrar señales GSM en las bandas celulares mencionadas anteriormente y está conectado con el módulo de procesamiento digital en banda base. En el *downlink*, el frente RF realiza la captura de la señal GSM, el respectivo filtrado, el proceso de *downconversion* y finalmente envía la señal banda base al módulo de procesamiento digital donde se decodifica el paquete recibido.

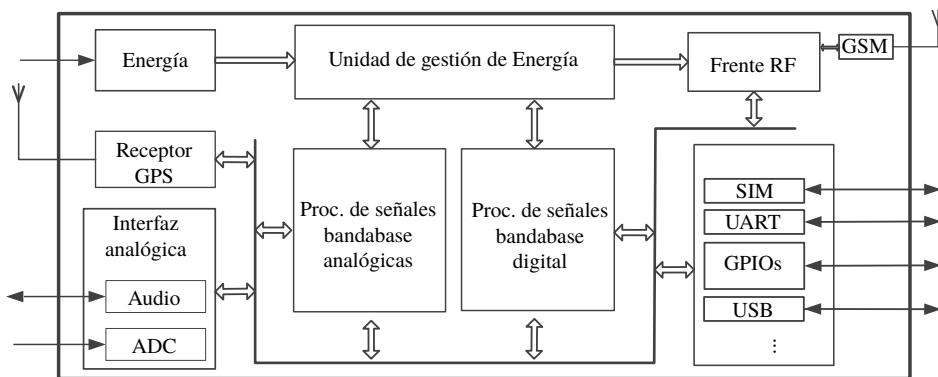


Fig. 2.16: Diagrama funcional del chip SIM800L.

En el caso de paquetes de voz, estos son procesados en el bloque vecino de procesamiento

analógico en banda base. La voz decodificada pasa a la interfaz analógica en forma de señal de audio. En el caso de paquetes de datos, estos pueden ser transferidos hacia otros módulos externos mediante conexión serial a través de una interfaz UART [25].

El manual que describe el hardware SIM800L [25] proporciona las tablas 2.3, 2.4, 2.5, las cuales indican los niveles de potencia de transmisión, niveles de sensibilidad en la recepción, y las bandas de frecuencia en las que opera el SIM800L.

TABLA N° 2.3: Potencia de salida del módulo SIM800L en las bandas DCS1800 y PCS1900.

Cód.	Potencia de Salida Nominal (dBm)	Tolerancia (dB)	
		Normal	Extremo
29	36	±2	±2.5
30	34	±3	±4
31	32	±3	±4
0	30	±3	±4
1	28	±3	±4
2	26	±3	±4
3	24	±3	±4
4	22	±3	±4
5	20	±3	±4
6	18	±3	±4
7	16	±3	±4
8	14	±3	±4
9	12	±4	±5
10	10	±4	±5
11	8	±4	±5
12	6	±4	±5
13	4	±4	±5
14	2	±5	±6
15-28	0	±5	±6

TABLA N° 2.4: Niveles de sensibilidad RF en el SIM800L.

Frecuencia	Sensibilidad de recepción (Típica)	Sensibilidad de recepción (Máxima)
GSM850	-109 dBm	-107 dBm
EGSM900	-109 dBm	-107 dBm
DCS1800	-109 dBm	-107 dBm
PCS1900	-109 dBm	-107 dBm

TABLA N° 2.5: Bandas de funcionamiento del SIM800L para los canales *Downlink* y *Uplink*.

Frecuencia	Recepción ( <i>Downlink</i> )	Transmisión ( <i>Uplink</i> )
GSM850	869 ~ 894 MHz	824 ~ 849 MHz
EGSM900	925 ~ 960 MHz	880 ~ 915 MHz
DCS1800	1805 ~ 1880 MHz	1710 ~ 1785 MHz
PCS1900	1930 ~ 1990 MHz	1850 ~ 1910 MHz

La banda de interés en la presente tesis es la DCS1800, debido a que actualmente en Perú esta banda no está asignada a ningún operador y por lo tanto no existen servicios que puedan perjudicarse por las pruebas realizadas.

Para el caso de la potencia de transmisión, el SIM800L llega emitir un máximo de 36 dBm en la banda DCS1800 y en la recepción, alcanza una sensibilidad típica de  $-109\text{dBm}$ , según las tablas 2.3, 2.4.

### 2.3.1 Implementación de una conexión serial para control del dispositivo final.

El SIM800L soporta conexión serial, para ello se hacen uso de los puertos Tx y Rx (Figura 2.15). Mediante la implementación de una conexión serial con algún HOST, es posible enviar comandos AT<sup>25</sup> al SIM800L para ejercer control sobre sus funciones [28].

La Figura 2.17 muestra el esquema de conexiones de los pins del SIM800L con un Arduino Mega 2560 para implementar la comunicación serial entre el SIM800L y un HOST con sistema operativo Linux o Windows.

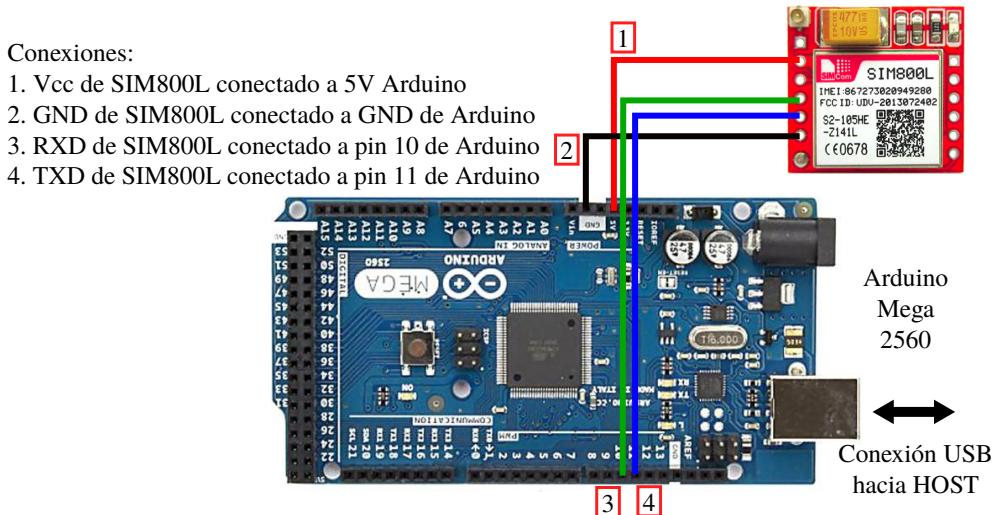


Fig. 2.17: Implementación de una conexión serial con el SIM800L haciendo uso de un microcontrolador Arduino Mega 2560.

En lo que respecta a la alimentación eléctrica del SIM800L, los pins Vcc y GND van conectados a los pins de salida de 5V y GND del Arduino, respectivamente. Respecto a la comunicación serial, los puertos RXD y TXD del SIM800L van conectados a los pins 11 y 10 del Arduino, respectivamente.

En el HOST, la aplicación Arduino [29] permite ejecutar el *script* de código presentado a continuación, el cual implementa una conexión serial con el SIM800L:

---

<sup>25</sup>AT: Attention commands [27].

```

//Conectar VIO to +5V
//Conectar GND to Ground
//Conectar RX (data into SIM808) to Digital 11
//Conectar TX (data out from SIM808) to Digital 10

#include <SoftwareSerial.h>
SoftwareSerial mySerial(10,11); // Pin RX y TX en el Arduino, para la conexión serial

void setup(){
// Abre una comunicación serial y espera a que se aperture el puerto:
  Serial.begin(9600);
  mySerial.begin(9600);
}

void loop() {
// código que hace posible el envío de comandos AT y la recepción de respuesta
  if (mySerial.available()) // Si existe data disponible para enviarse al Lonet
    Serial.write(mySerial.read()); // entonces enviar esa data al Serial del Lonet
  if (Serial.available()){ // Luego de ello, si el Serial del Lonet tiene data para
    enviar
    while(Serial.available()){ // Mientras que exista aquella data
      mySerial.write(Serial.read()); // Presentar en la consola la data enviada por el
        Lonet.
    }
    mySerial.println(); // Salto de línea
  }
}
}

```

La ejecución del anterior código en el IDE<sup>26</sup> de Arduino implementa una conexión serial mediante el terminal de comandos que se muestra en la Figura 2.18. Por medio de este terminal se pueden enviar comandos AT al SIM800L y recibir información relacionada con el comando AT enviado.

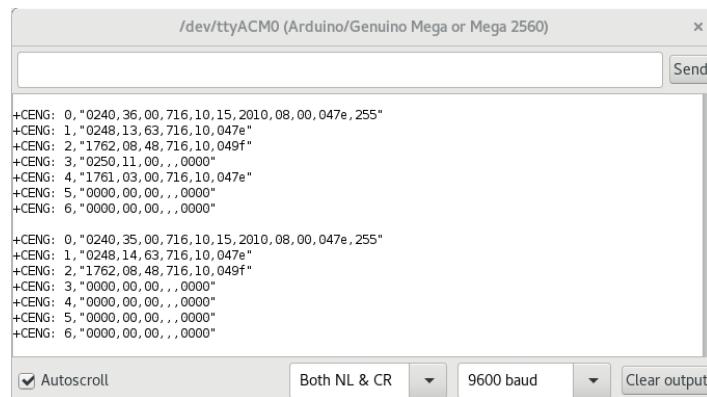


Fig. 2.18: Terminal de conexión serial con el SIM800L.

### 2.3.2 Estados GPRS en el dispositivo final.

Para establecer una conexión GPRS, el SIM800L debe pasar por diferentes estados GPRS, los cuales se muestran en la Figura 2.19 y se detallan a continuación [30]:

- IP INITIAL: Estado inicial de todo establecimiento de conexión GPRS. En este

<sup>26</sup>IDE: *Integrated Development Environment* o entorno de desarrollo integrado. Es un software de interfaz gráfica que permite editar y compilar código de un determinado lenguaje de programación.

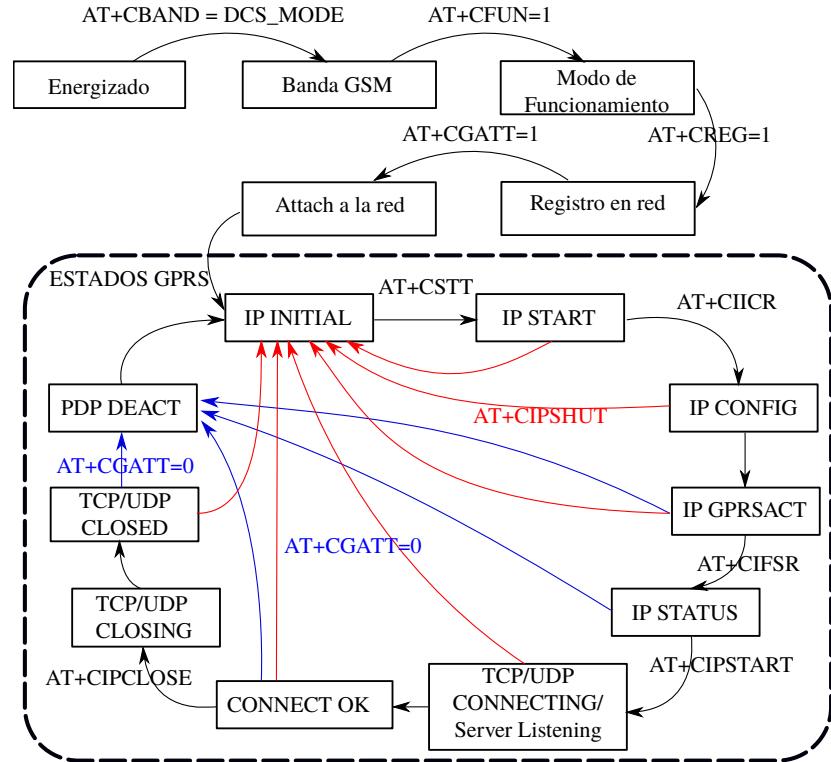


Fig. 2.19: Diagrama de transiciones entre estados GPRS y los comandos AT respectivos.

punto se asume que no existe ninguna conexión GPRS establecida.

- **IP START:** Es el estado siguiente a IP INITIAL y se activa mediante el comando `AT+CSTT=<APN>, <USER>, <PASSW>`. En este estado se inicia una tarea TCP/UDP.
- **IP CONFIG:** Es el estado siguiente a IP START y se activa con el comando `AT+CIICR`. En este estado se configura el contexto PDP.
- **IP GPRSACT:** Este estado es inmediato a IP CONFIG. En este punto el contexto ya está activo.
- **IP STATUS:** Es el estado siguiente a IP GPRSACT y se activa mediante el comando `AT+CIFSR`. En este punto el SIM800L obtiene una dirección IP local de la red GPRS.
- **TCP / UDP CONNECTING:** Es el estado siguiente a IP STATUS y se activa mediante el comando `AT+CIPSTART`. En este punto, el SIM800L está intentando conectarse al servidor destino indicado en el comando `AT+CIPSTART` [28], mediante protocolo TCP/UDP.
- **CONNECT OK:** Es el estado siguiente a TCP / UDP CONNECTING y se activa cuando el SIM800L consigue conectarse vía protocolo TCP/UDP al servidor y puerto indicado.

- TCP/UDP CLOSING: Es el estado que se activa cuando se ejecuta el comando AT+CIPCLOSE. El SIM800L está cerrando la conexión con el servidor.
- TCP/UDP CLOSED: El SIM800L ha conseguido cerrar la conexión TCP/UDP con el servidor. El Contexto PDP activado en el estado IP GPRSACT aún sigue activo.
- PDP DEACT: En este estado, el contexto PDP es desactivado. Esto se puede conseguir también desde los estados IP GPRSACT, IP STATUS, CONNECT OK, mediante el comando AT+CGATT=0.

### 2.3.3 Conexión GPRS y envío de datos a una PDN externa.

De la Figura 2.19, una vez que el SIM800L es energizado, se selecciona la banda de operación mediante el comando AT+CBAND=DCS\_MODE. Posteriormente, se hace la consulta mediante el comando AT+CBAND? para verificar que el SIM800L esté operando en la banda DCS1800.

```
AT+CBAND = DCS_MODE
OK

AT+CBAND?
+CBAND: DCS_MODE

OK
```

El paso a seguir es seleccionar el modo de funcionamiento del SIM800L mediante el comando AT+CFUN=1. Este paso es necesario, caso contrario el SIM800L permanecerá en el modo de funcionamiento AT+CFUN=0 cuyas funcionalidades son mínimas y no es posible establecer ninguna conexión GPRS.

```
AT+CFUN = 1
OK

AT+CFUN?
+CFUN: 1

OK
```

Posteriormente, es necesario realizar el registro en la red sirviente mediante el comando AT+CREG=1 y la consulta correspondiente para verificar que el registro fue exitoso.

```
AT+CREG = 1
OK

AT+CREG?
+CREG: 1,1

OK
```

Una vez que el SIM800L se registra en la red, se activa el procedimiento de "GPRS Attach" (sección 2.2.6) mediante el comando AT+CGATT=1. Con ello, se activa la conexión a nivel GPRS y se establecen los TBFs con la red.

```
AT+CGATT = 1
OK

AT+CGATT?
+CGATT: 1

OK
```

Una vez que el SIM800L realiza el *attach* con la red, este pasa al estado IP INITIAL según la Figura 2.19. En este momento, el SIM800L tiene asignado recursos de radio para iniciar una conexión de datos. Para ello, es necesario que se configure el parámetro APN<sup>27</sup>, el cual es un nombre o identificador de la red externa con la cual el GGSN se conecta para dar salida a los paquetes IP. No necesariamente se trata de la PDN externa destino.

```
AT+CSTT="CMNET"
OK

AT+CSTT?
+CSTT: "CMNET", "", ""

OK
```

En este punto el SGSN establece los túneles con el GGSN y activa el contexto PDP tal como se explicó en la sección 2.2.7. Para especificar el APN se puede usar la recomendación de los manuales del SIM800L [28].

En las redes comerciales, por temas de seguridad el APN debe ir acompañado por datos de autenticación; sin embargo, en el prototipo de red propuesto, este requerimiento no es tomado en consideración y por ello no se especifican los parámetros usuario y contraseña al momento de activar el APN.

Una vez definido el APN, el SIM800L pasa al estado GPRS denominado IP START. El dispositivo ha iniciado una tarea TCP y es necesario ejecutar el comando AT+CIICR para activar la conexión inalámbrica GPRS del SIM800L con la red.

```
AT+CIPSTATUS
OK

STATE: IP START

AT+CIICR
OK

AT+CIPSTATUS
OK

STATE: IP GPRSACT
```

En el log anterior se puede apreciar como, luego de la ejecución del comando AT+CIICR, el estado CIPSTATUS cambia de IP START a IP GPRSACT.

---

<sup>27</sup>APN: *Access point name*, o nombre del punto de acceso. El APN es un punto de entrada hacia una red IP para un dispositivo móvil.

El siguiente paso es obtener una dirección IP mediante el comando AT+CIFSR. El siguiente log muestra que luego de ejecutar el comando, se obtiene la IP: 192.168.99.1 para el SIM800L y el estado GPRS cambia a IP STATUS.

```
AT+CIFSR
192.168.99.1

AT+CIPSTATUS
OK

STATE: IP STATUS
```

Una vez que la dirección IP ha sido asignada al SIM800L, se puede proceder a enviar pings hacia alguna IP externa. En este caso, la dirección IP destino es la 8.8.8.8 correspondiente al portal [www.google.com](http://www.google.com).

```
AT+CIPPING="8.8.8.8",5
+CIPPING: 1,"8.8.8.8",27,60
+CIPPING: 2,"8.8.8.8",4,60
+CIPPING: 3,"8.8.8.8",4,60
+CIPPING: 4,"8.8.8.8",4,60
+CIPPING: 5,"8.8.8.8",4,60

OK
```

La Figura 2.20 corresponde al consumo de corriente de un SIM800L cuando realiza un intento de conexión a la red GPRS de OpenBTS. Inicialmente, el MS es energizado y permanece en un período de establecimiento del sistema similar a lo descrito en la sección 2.2.4.

Luego, aproximadamente a los 100ms comienza una alternancia entre ventanas de transmisión y recepción. Este período de tiempo corresponde al procedimiento de acceso aleatorio y al intercambio de mensajes entre el MS y la red para obtener recursos de radio descritos en la sección 2.2.5.

Posteriormente, antes de llegar a los 300 ms, la red finalmente concede los recursos de radio al MS. Este pasa brevemente a estado *idle* para luego activar el procedimiento *GPRS Attach* descrito en la sección 2.2.6, a fin de establecer la conexión a nivel GPRS con la red

Posteriormente, en un determinado instante alrededor de los 400ms, el MS realiza la transmisión de un *ping* de datos IP hacia una red externa y una vez que finaliza el flujo de radio bloques, el MS pasa a modo *idle* nuevamente. La Figura 2.20 muestra que el consumo de corriente es de aproximadamente 95mA cuando transmite señales y de 65mA cuando recibe señales.

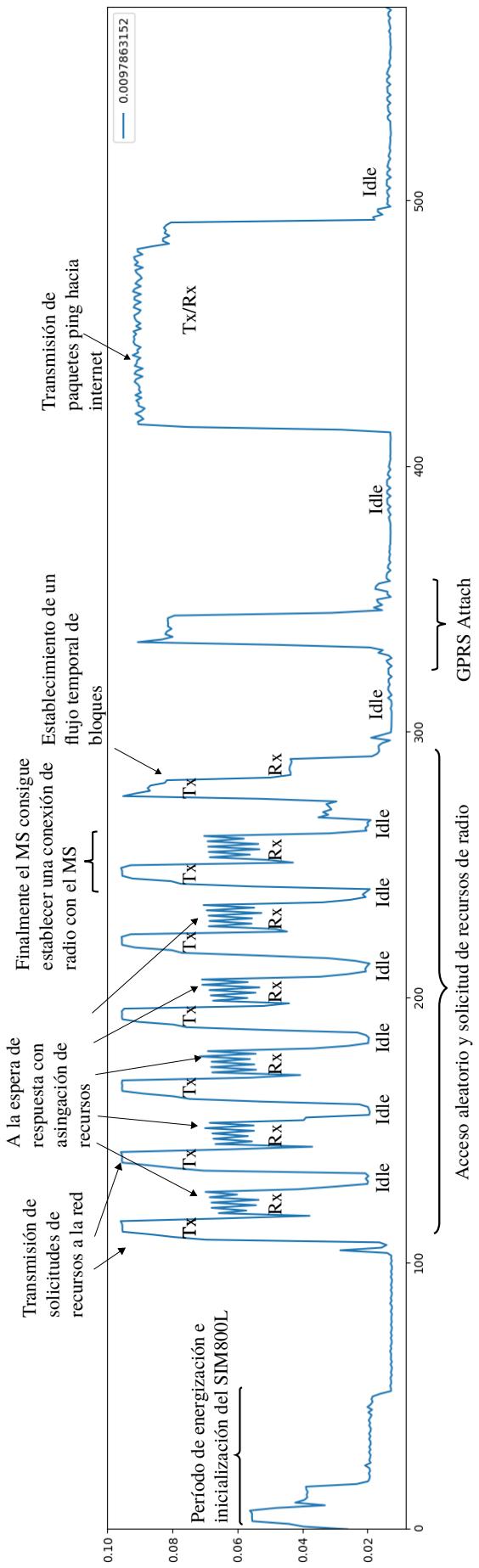
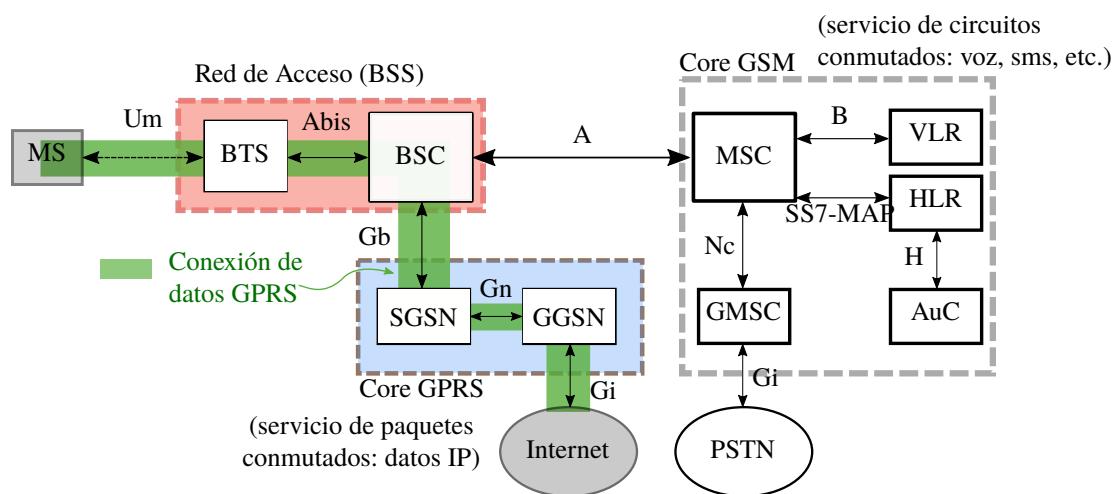


Fig. 2.20: Consumo de corriente de los diferentes procedimientos que el MS desempeña desde su energización hasta el establecimiento de transferencia de paquetes a la red.

## 2.4 Plataforma de Software OpenBTS.

Según [31], OpenBTS es un conjunto de componentes de software de código abierto, que implementados en un computador genérico conectado a un periférico universal basado en Radio Definido por Software (SDR), crea la interfaz de radio Um de una red GSM para interconectar terminales que soporten dicho estándar y proveerles servicios de telefonía (voz, sms, y datos) sobre Internet.

La característica más conocida de OpenBTS es la factibilidad de usar el servidor VoIP Asterisk [32] para virtualizar una centralita de llamadas y de esa manera emular servicios de voz propios de una red de telefonía móvil 2G; sin embargo, a partir de la versión v3.1.3 (año 2015), OpenBTS incluye servicios y procesos computacionales que en conjunto emulan una red de datos GPRS, abriendo la posibilidad de proveer acceso a terminales que soportan el estándar GPRS. La presente tesis usa la versión 5.0 de OpenBTS [33].



A = interfaz A

Abis = interfaz Abis

AuC = Centro de Autenticación

B = interfaz B

BSC = Estación Base Controladora

BTS = Estación Base Transceiver

BSS = Subsistema Estación Base

Gb = Interfaz Gb

GGSN = Nodo Gateway GPRS

GMSC = Nodo Gateway GSM

H = Interfaz H

Gi = Interfaz Gi

HLR = Registro local de ubicación

MSC = Centro de conmutación móvil

Nc = Interfaz Nc

SGSN = Nodo Servidor GPRS

SS7-MAP = Sistema señalización 7

Um = Interfaz aire GSM

VLR = Registro de ubicación para visitantes

Gn = Interfaz Gn

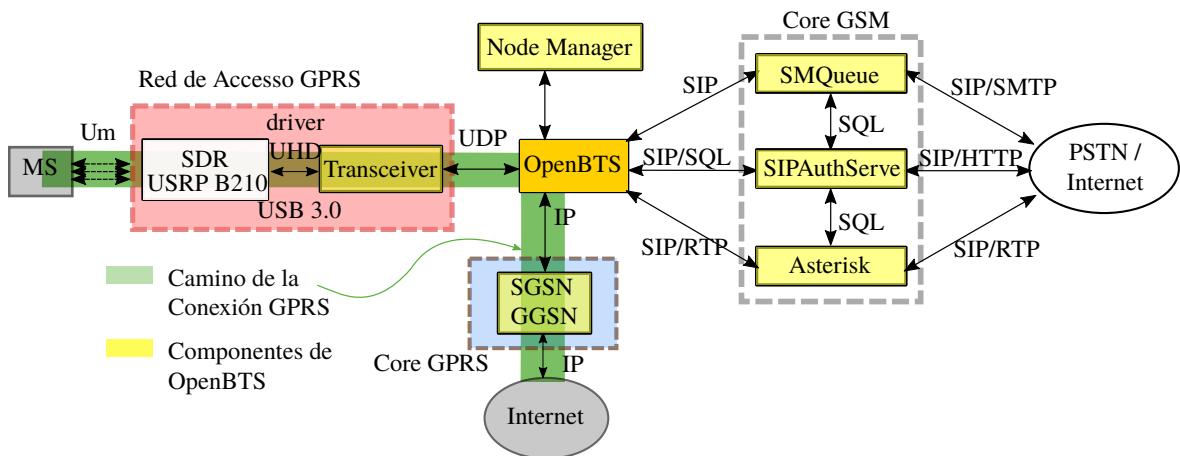
Fig. 2.21: Red GSM/GPRS Tradicional. El camino que recorren los datos IP tanto en downlink y uplink a través de la arquitectura de red.

#### 2.4.1 Comparación entre una red GPRS tradicional y una red GPRS implementada con OpenBTS.

La Figura 2.21 muestra los componentes de una red GSM/GPRS de un operador de telefonía móvil tradicional. Esta red está compuesta principalmente por la red de acceso, el core de red para servicios GPRS, y el core de red para servicios GSM.

La red de acceso está compuesta por la estación base (BTS) y su controladora (BSC), los cuales conforman el subsistema estación base (BSS) tal como se explicó en la sección 2.1, cuya principal función es de proveer el acceso inalámbrico a los MS.

La red de acceso se comunica con el core GPRS mediante la conexión BSC-SGSN, a través de la interfaz Gb. El SGSN mantiene una comunicación con el GGSN estableciendo un túnel con la PDN externa. Los paquetes IP generados o destinados al MS siguen el camino de color verde de la Figura 2.21.



Um = Interfaz aire GSM

USB 3.0 = Conexión USB 3.0

SDR = Software Defined Radio

Transceiver = Modulo controlador del SDR

UDP = Protocolo de Datagramas de usuario

IP = Protocolo de Internet

OpenBTS = master software

SGSN = Nodo Servidor GPRS

GGSN = Nodo Gateway GPRS

SIP = Protocolo de Inicio de Sesión

SQL = Protocolo SQL, comunicación con BBDD

RTP = Protocolo de transporte en tiempo real

SMQueue = Protocolo de mensajes de texto

SIPAuthServe = Servido de autenticación

Asterisk = centralita/servidor de llamadas SIP

SMTP = Protocolo simple de transferencia de email

HTTP = Protocolo de transferencia de datos vía web

PSTN = red de telefonía pública

Fig. 2.22: Red GSM/GPRS de la plataforma OpenBTS y camino que recorre los datos IP a través de la arquitectura.

Por otro lado, la Figura 2.22 describe la red OpenBTS como un conjunto de componentes en forma de procesos computacionales (bloques amarillos). De acuerdo a la versión 5.0 de OpenBTS [33], estos componentes son:

- **OpenBTS.** El componente de software que gobierna a toda la plataforma OpenBTS.

Este módulo no sólo controla a los demás componentes sino que también es el componente que implementa el *stack* de protocolos de capa GSM PHY, MAC, RLC, y LLC.

- **Transceiver.** este módulo administra el controlador o *driver* UHD<sup>28</sup> para ejercer control sobre las funcionalidades del hardware USRP B210. Las funciones del módulo *transceiver* son de transmisión o recepción de las señales de radio, o la intercambio de las muestras correspondientes a señales de radio entre el USRP B210 y el computador.
- **SIPAuthServe.** Es el componente computacional que implementa los procesos relacionados con el acceso y registro de los MS cada vez que estos se conectan a la red GPRS. El SIPAuthServe realiza funciones similares al de un HLR de una red GSM convencional (sección 2.1).
- **SGSN/GGSN.** Los componentes SGSN y GGSN emulan a las respectivas entidades en una red GPRS convencional. El SGSN/GGSN se encarga principalmente de establecer los contextos MM y contextos PDP necesarios para establecer la transferencia de datos GPRS sobre GSM (sección 2.2).
- **NodeManager.** Permite que otras aplicaciones gestionen los diferentes componentes de OpenBTS tales como el propio OpenBTS, SIPAuthServe, SGSN/GGSN, etc. a través de la manipulación APIs<sup>29</sup> de gestión JSON.

En conclusión, en una red tradicional GPRS, los paquetes de datos recorren las entidades MS - BTS - BSC - SGSN - GGSN hasta llegar a internet. En el caso de la red GPRS propuesta, la dualidad USRP B210 - Transceiver reemplaza al par BTS - BSC, y los componentes OpenBTS y SGSN/GGSN se encargan de procesar los paquetes para luego enviarlos hacia la PDN destino, o viceversa.

#### **2.4.2 Descripción de los procesos computacionales de OpenBTS relacionados con el establecimiento del servicio GPRS.**

OpenBTS es el nombre de la plataforma y también del componente de software responsable de implementar los protocolos de las capas GSM PHY (Capa L1), MAC/RLC (Capa L2), y LLC (Capa L3) descritas en la sección 2.2. El subcomponente *Transceiver*, mediante

---

<sup>28</sup>UHD: USRP Hardware Driver o *driver* para dispositivos USRP.

<sup>29</sup>API es el acrónimo de *Application Programming Interface* o interfaz de programación para aplicaciones, la cual sirve para poder interactuar con los componentes de software que soporten el API.

el uso del driver UHD, ejerce control sobre el USRP B210 y de esta manera recrea la interfaz aire para el acceso de los MS.

En el sentido *uplink*, el USRP B210 recibe los radio-paquetes provenientes del MS, los encapsula en paquetes UDP y los entrega al subcomponente *Transceiver* vía conexión USB 3.0. El subcomponente *Transceiver* redirige los paquetes UDP hacia el motor GSM de OpenBTS, el cual aplica el *stack* de protocolos para remover las cabeceras de los protocolos de radio (GSM PHY/MAC/RLC) y obtener el PDU de capa LLC (Figura 2.2).

El motor GSM envía el PDU de capa LLC hacia el motor de comando y control, el cual deduce que la trama LLC debe ir dirigida al subcomponente SGSN/GGSN ya que se trata de un paquete IP.

OpenBTS utiliza los servicios del protocolo TCP/IP implementado por su propio *stack* para encaminar los paquetes IP hacia el proceso computacional del subcomponente SGSN/GGSN, donde se realizará el respectivo procesamiento para dirigir el paquete IP hacia la PDN destino.

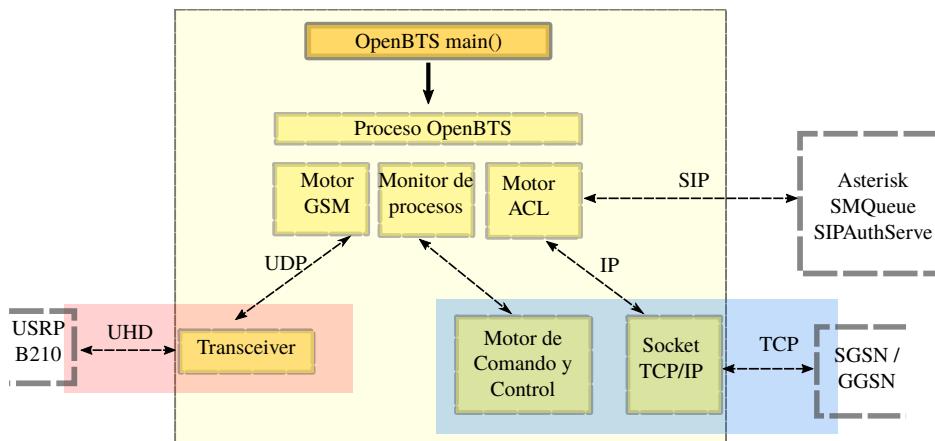


Fig. 2.23: OpenBTS desde el punto de vista de procesos computacionales.

La Figura 2.24 muestra el *stack* de protocolos en una conexión entre un MS y la red GPRS de OpenBTS. Las flechas bidireccionales rojas representan las conexiones a nivel de radio para establecer el flujo temporal de bloques entre el MS y OpenBTS.

La capa LLC provee el soporte para el transporte de los PDUs de las capas GMM, SM, y SNDCP entre el MS y el SGSN. De la sección 2.2, GMM es la capa que hace posible que OpenBTS conozca la ubicación de los MS, de modo que puede encaminar correctamente los paquetes de datos que tengan como destino un determinado MS.

Por otro lado, SM es el protocolo encargado de establecer el contexto PDP entre el MS,

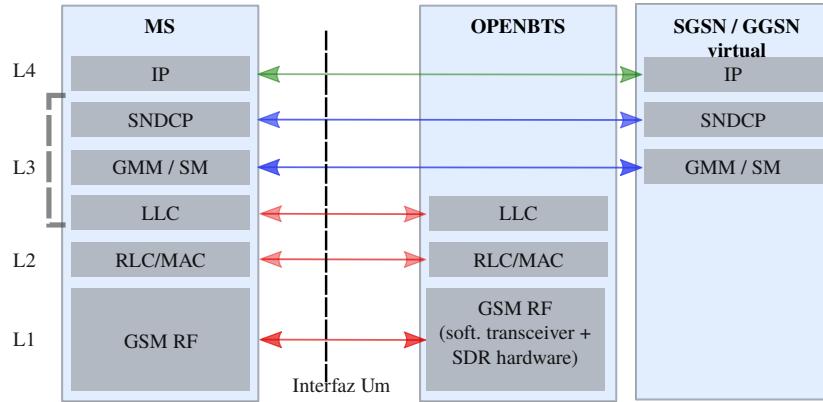


Fig. 2.24: OpenBTS desde el punto de vista de procesos computacionales.

SGSN, y GGSN, de modo que se pueda establecer un túnel *end-to-end* entre el MS y la PDN externa pasando a través del SGSN y GGSN.

Una vez implementado el contexto MM y el contexto PDP, las capas SNDCP del MS y de la red están comunicadas entre sí haciendo posible el intercambio de paquetes IP entre ambas entidades. Finalmente, el GGSN establece la puerta de enlace con las redes de datos externas a OpenBTS.

## 2.5 Tecnología Radio Definido por Software (SDR).

En telecomunicaciones, el término *radio* se usa para denominar a cualquier dispositivo usado para intercambiar información digital por medio de ondas de radio. Al inicio del avance tecnológico, los dispositivos de radio fueron diseñados para operar con cierta tecnología de modulación de señales lo que hacía de ellos unos dispositivos poco flexibles. Con el pasar de los años, la tecnología fue evolucionando hasta dar paso al concepto de radio definido por software, el cual engloba la idea de implementar a nivel software operaciones que tradicionalmente eran implementadas a nivel de hardware, tales como el filtrado de señales, la modulación/demodulación, la canalización, ecualización, etc. Con el objetivo de describir la tecnología SDR, a continuación se describen el *transceiver* superheterodino, el *transceiver* SDR ideal y el *transceiver* SDR basado en el mixer digital.

### 2.5.1 Transceiver Superheterodino.

El diagrama de bloques de un radio *transceiver* convencional se muestra en la Figura 2.25. Siguiendo el *path*<sup>30</sup> de transmisión (Tx), la etapa inicial es el bloque DSP, el cual es un procesador digital de señales y se encarga de convertir secuencias binaria en símbolos.

<sup>30</sup>*Path* es el término que se utiliza para la secuencia de dispositivos que recorre la señal tanto en la transmisión como en la recepción.

Luego, los símbolos pasan al modulador para su respectiva conversión a señales en el dominio de banda base. Los antiguos *transceivers* superheterodinos incorporaban un mixer para realizar una primera operación de *upconversion*<sup>31</sup> en la etapa IF y posteriormente, la señal pasaba por un segundo *upconversion* en la etapa RF. Los superheterodinos modernos removieron el *upconversion* en la etapa IF debido a que se mejoró la tecnología de fabricación de mixers para señales RF.

Por otro lado, en el *path* de recepción (Rx), la señal es recibida por la antena y pasa a la etapa RF donde se realizan procesos de limpieza y *boost*<sup>32</sup> de la señal (filtro pasabanda, amplificador de bajo ruido, etc.) y el respectivo *downconversion*<sup>33</sup> (mixer). Opcionalmente puede existir una segunda etapa de *downconversion* en el dominio IF. Luego del *downconversion*, la señal en banda base pasa a un demodulador donde se obtienen los símbolos los cuales pasan al bloque de procesamiento digital de señales en el cual se extrae la secuencia de bits recibida.

En el *transceiver* superheterodino, los filtros, los moduladores/demoduladores, ecualizadores, mixers, y otros, se implementan en hardware; es decir, existe un dispositivo físico que desempeña una determinada tarea. Esto significa que el *transceiver* sólo podía ser diseñado para operar en cierto rango de frecuencias y con anchos de banda limitados, además de que sólo podía funcionar con señales que sus moduladores o demoduladores podían entender. Todas estas características hacían que la tecnología carezca de flexibilidad suficiente y sólo permitía el diseño de sistemas de configuración fija en cuanto al ancho de banda, frecuencia de operación, técnica de modulación/demodulación, etc.

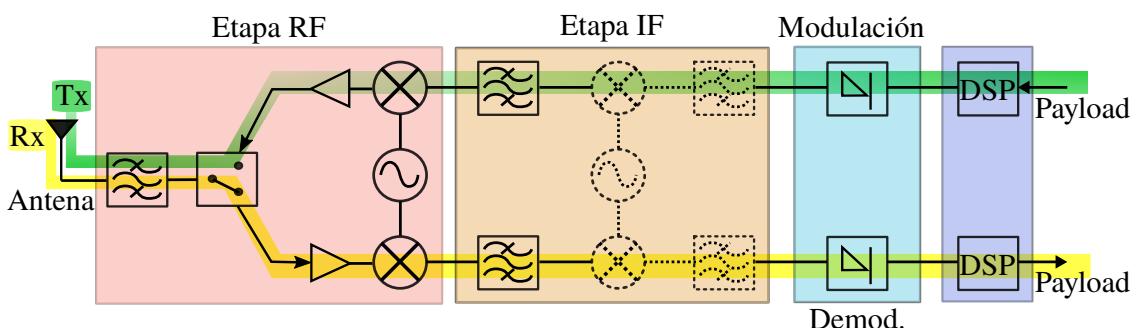


Fig. 2.25: Arquitectura de un transceiver superheterodino.

<sup>31</sup>En procesamiento de señales, *upconversion* es el traslado del espectro de una señal en banda base hacia el dominio de frecuencias IF o RF.

<sup>32</sup>El término *boost* se refiere al aumento de la energía en una señal, generalmente realizable con la ayuda de amplificadores de potencia.

<sup>33</sup>En procesamiento de señales analógicas, *downconversion* es el traslado de una señal en el dominio de frecuencias RF o IF hacia el dominio de banda base.

### 2.5.2 Transceiver SDR Ideal.

La necesidad por un radio transceiver que sea capaz de manejar más canales, con mayores anchos de banda, y rangos de frecuencia de operación más dinámicos, dio lugar a la búsqueda de formas más eficientes para pasar del dominio analógico al digital, y viceversa. La mejora en cuanto a la velocidad de los conversores Analógico-Digital y Digital-Analógico (ADC/DAC) dio lugar a nuevas arquitecturas y conceptos en radio-*transceivers*.

Uno de esos conceptos es el de radio definido por software (SDR). La Figura 2.26 muestra un *transceiver* SDR ideal en el cual la única etapa analógica es la antena y el amplificador de potencia.

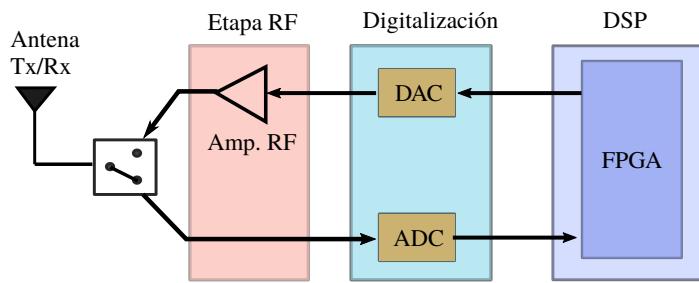


Fig. 2.26: Arquitectura de un transceiver SDR Ideal.

Dado que no se visualizan bloques de *downconversion*, esta arquitectura sugiere que los conversores DAC/ADC deben ser lo suficiente veloces para realizar el muestreo de todo el dominio RF, desde 0 Hz hasta la máxima frecuencia RF de la señal. En otras palabras, digitalizar todo el espectro de manera directa en un sólo bloque, sin etapas intermedias.

Esto conlleva a que operaciones como el filtrado del canal, la ecualización de la banda de interés, el submuestreo de la señal, etc., se realicen en el dominio digital, es decir a nivel de software y en un bloque de procesamiento de banda base como puede ser un FPGA<sup>34</sup>.

De esta manera, la señal RF tiene una transición directa a la etapa de procesamiento digital lo que resulta en una arquitectura más reducida y eficiente. La velocidad de los conversores ADC/DAC ha mejorado considerablemente en los últimos años y, por consiguiente, cada vez se logra muestrear mayores anchos de banda.

Sin embargo, aún existen limitaciones en cuanto a la tasa de muestreo de los ADC/DAC y por lo tanto, no se puede aún realizar el muestreo de todo el dominio RF hasta la frecuencia

---

<sup>34</sup>FPGA: Field Programmable Gate Array. Una FPGA o matriz de puertas programables (del inglés field-programmable gate array) es un dispositivo programable que contiene bloques de lógica cuya interconexión y funcionalidad puede ser configurada en el momento mediante un lenguaje de descripción especializado. La lógica programable puede reproducir desde funciones tan sencillas como las llevadas a cabo por una puerta lógica o un sistema combinacional hasta complejos sistemas en un chip.

máxima de una determinada señal. Por ello, es necesario implementar una etapa adicional dedicada a las operaciones de *downconversion* y *upconversion*.

### 2.5.3 Transceiver SDR basado en el mixer digital.

De todas las arquitecturas SDR que existen en la actualidad, la Figura 2.27 muestra la que está basada en el mixer digital, o también denominado modulador en cuadratura. En el *path Rx*, la señal RF es recibida por una antena de banda ancha. La señal pasa a un amplificador de bajo ruido para elevar su potencia y luego pasa a la etapa del mixer digital.

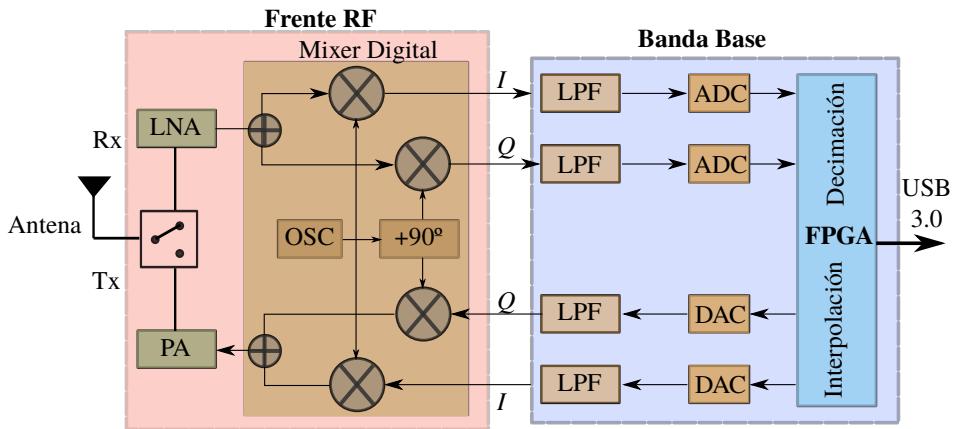


Fig. 2.27: Arquitectura SDR basada en el mixer digital.

El mixer digital consiste de dos mixers, cada uno con una señal de oscilador local con frecuencia igual a la frecuencia portadora de la señal RF pero desfasadas  $90^\circ$  entre sí. Estos mixers convierten la señal RF en dos señales banda base denominadas señales I/Q. Las señales I/Q, aún analógicas, pasan por un filtro pasabajas (LPF) para eliminar las frecuencias imágenes resultantes en cualquier mezclado de señal.

Finalmente, las señales I/Q pasan hacia sus respectivos conversores ADC donde son digitalizadas, siendo las muestras enviadas al FPGA para el respectivo procesamiento en banda base. Partiendo de principios de análisis matemático, se puede calcular la fase y la amplitud de la señal RF original conociendo los espectros de frecuencia de las señales I/Q.

En conjunto, las señales I/Q son una representación de la señal RF en banda base. Este tipo de modulador/demodulador en cuadratura también se denomina de modulador/demodulador de conversión directa o conversión zero-IF debido a que no se necesita una etapa intermedia en la cual la señal pase del dominio RF a IF, y luego de IF a banda base.

A continuación se proporciona una breve explicación de cómo es que las señales discretas I/Q en conjunto son una fiel representación de la señal RF analógica en la antena. La

explicación se realiza sobre el *path* de recepción pero el mismo principio se puede aplicar para explicar cómo las señales I/Q pueden ser moduladas para generar la señal RF en Tx.

La Figura 2.28 muestra el espectro  $S_{RF}(f)$  de una señal continua  $s_{RF}(t)$ , de ancho de banda B y con una frecuencia portadora  $f_c$ , que ingresa por la antena Rx de la Figura 2.27. En el primer mixer, la señal  $s_{RF}(t)$  es multiplicada por la señal del oscilador, la cual asumimos es una función  $\cos(2\pi f_c t)$  con frecuencia portadora igual a  $f_c$  para realizar la conversión directa de la señal RF a banda base<sup>35</sup>.

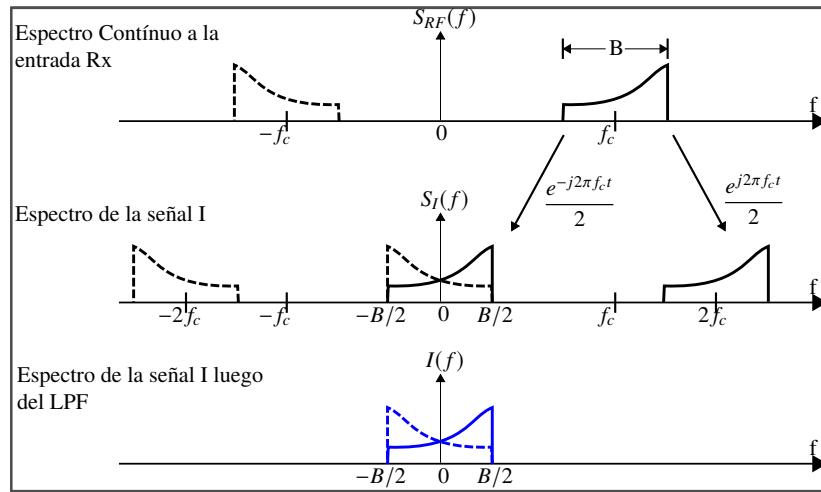


Fig. 2.28: Secuencia de cambios en el espectro de la señal  $s_{RF}(t)$  luego de pasar por el mixer superior del demodulador y el filtro pasabajas (LPF).

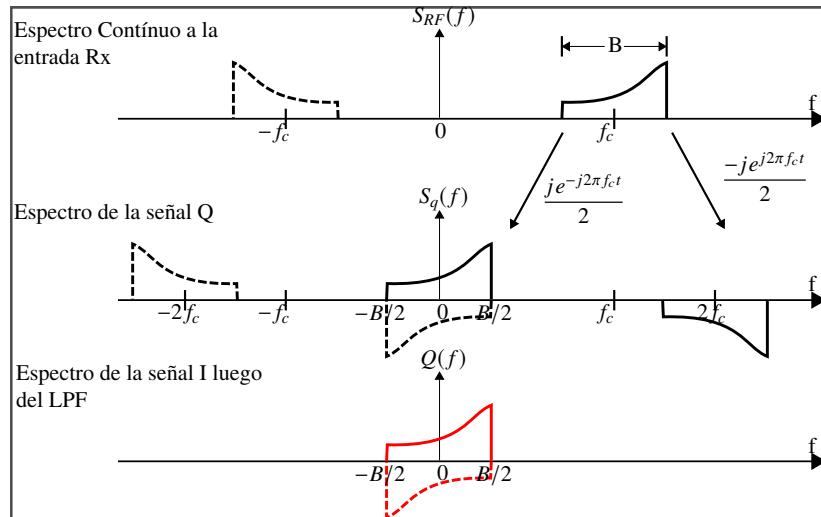


Fig. 2.29: Secuencia de cambios en el espectro de la señal  $s_{RF}(t)$  luego de pasar por el mixer inferior del demodulador y el filtro pasabajas (LPF).

<sup>35</sup>La explicación es la misma si la función del primer oscilador fuese  $\sin(2\pi f_c t)$  debido a que el otro oscilador estaría de igual manera desfasado  $\pi/2$  rad.

En otras palabras, el espectro  $S_{RF}(f)$  convoluciona con el espectro de la función  $\cos(2\pi f_c t)$  del oscilador y esto se refleja en la Figura 2.28 cuando la señal  $S_{RF}(f)$  se traslada a las frecuencias 0 y  $2f_c$  Hz como señalan las flechas. La imagen del espectro de  $S_{RF}(f)$  (líneas punteadas) sufre el mismo traslado hacia 0 y  $-2f_c$  Hz. La señal resultante del mixer pasa luego a través de un filtro pasabajas (LPF) el cual elimina los espectros de las frecuencias  $-2f_c$  y  $2f_c$  Hz quedando el espectro de la señal I (líneas de color azul).

La Figura 2.29 describe lo que sucede en el otro ramal, es decir la convolución de la señal  $S_{RF}(f)$  con la señal del oscilador en cuadratura pero esta vez desfasada  $90^\circ$ , es decir una señal  $\sin(2\pi f_c t)$ . En este caso, el espectro  $S_{RF}(f)$  se traslada hacia 0 Hz con el mismo signo y hacia  $2f_c$  con el signo invertido. El mismo efecto sucede con el espectro imagen de  $S_{RF}(f)$  (líneas punteadas). Luego, la señal resultante pasa a través de un filtro pasabajas con lo cual se eliminan los espectros centrados en  $-2f_c$  y  $2f_c$  Hz quedando el espectro de la señal Q (líneas de color rojo).

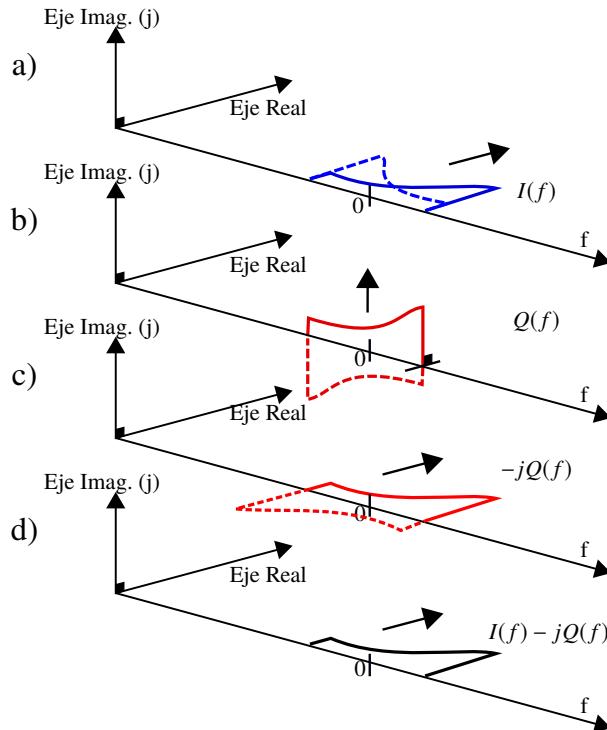


Fig. 2.30: a) Espectro de  $i(t)$ . b) Espectro de  $q(t)$ . c) Espectro de  $-jq(t)$ . d) Una versión de  $s_{RF}(t)$  en banda base en la forma de  $i(t) - jq(t)$ .

La Figura 2.30 muestra la idea esencial del modulador en cuadratura. En a) se presenta el espectro de la señal I recostado sobre el plano del eje real para ir con la idea de que es el resultado del mixing de la señal RF con la señal coseno, y en b) el espectro de la señal Q,

de color rojo, recostado sobre el plano del eje imaginario denotando que es el resultado del mixing de la señal RF con una señal seno. Es decir, la señal coseno es la señal *In-phase* y la señal seno es la señal en *Quadrature*.

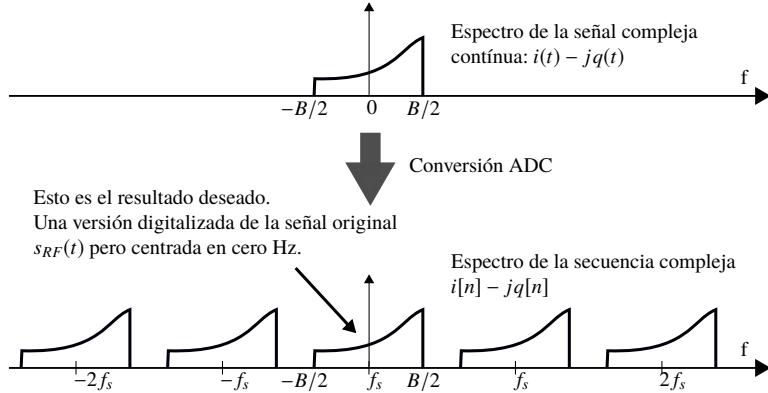


Fig. 2.31: Representación espectral de la conversión Analógico a Digital de la señal banda base compleja I-jQ.

Dado que las señales I y Q son el resultado de un modulador en cuadratura, es lógico que estén representados sobre ejes perpendiculares entre sí. Si a la señal Q se le multiplica un factor  $-j$ , el espectro realiza una rotación de  $90^\circ$  debido al factor  $j$  y luego realiza una inversión de signo quedando el espectro como se muestra en c).

Finalmente, si se realiza la suma algebraica del espectro de frecuencias de a) y c), es decir  $I(f) - jQ(f)$ , el resultado obtenido sería el espectro la señal  $s_{RF}(t)$  centrado en 0 Hz, en el eje real. De este modo  $i(t) - jq(t)$  es una versión en banda base de la señal  $s_{RF}(t)$ .

La Figura 2.31 muestra en la parte superior el espectro de la señal continua  $i(t) - jq(t)$  y en la parte inferior las réplicas de la señal en cada zona de Nyquist, como resultado del muestreo realizado por el conversor ADC. De esta manera, a la salida de los conversores ADC, se obtiene las señales discretas I/Q que en su conjunto, en la forma de  $i[n] - jq[n]$ , representan una versión digitalizada de la señal RF trasladada a banda base.

#### 2.5.4 Módulo USRP B210.

Una vez que se entiende el concepto de radio definido por software y la conversión directa de RF a banda base, es momento de presentar el dispositivo SDR con el que se realiza la implementación de la interfaz de radio. La Figura 2.32 muestra la tarjeta USRP B210 del fabricante Ettus Research y su respectivo diagrama de bloques funcionales.

La arquitectura de un USRP B210 está compuesta por una etapa RF, la cual consiste básicamente de una red de switches para la conmutación de las antenas. El USRP B210 posee

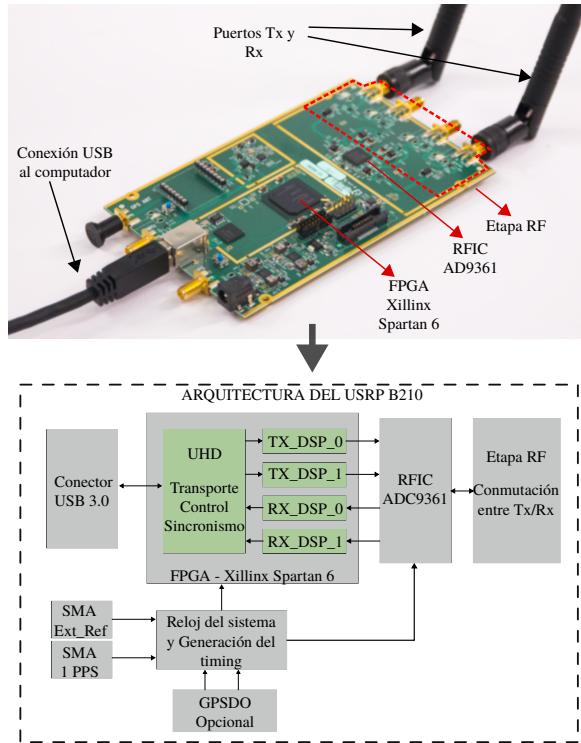


Fig. 2.32: Tarjeta USRP B210. Fabricante Ettus Research - National Instruments.

dos módulos Tx/Rx por lo cual es necesario un circuito para elegir que módulo usar en un determinado instante o para usar ambos módulos Tx/Rx en configuración MIMO<sup>36</sup> 2x2.

La siguiente etapa es el dispositivo RFIC<sup>37</sup> modelo AD9631 del fabricante *Analog Devices*, el cual se muestra en la Figura 2.33. El RFIC AD9631 es un transceiver de alta performance diseñado para aplicaciones de estación base 3G y 4G [3].

La sección del receptor (Rx) contiene hasta 03 entradas de RF las cuales pasan a través de un demodulador digital similar al presentado en la sección 2.5.3. La salida del mixer digital va hacia una etapa de filtros pasabajas y finalmente llega a una etapa de conversores ADC. Los ADC obtienen muestras de la señal compleja en banda base para ser utilizadas por una etapa procesamiento digital de señales (DSP).

La sección del transmisor (Tx) inicia con los datos digitales provenientes del procesador de banda base que pasan a una etapa de filtro FIR para interpolación de muestras. Las muestras en banda base de las señales complejas I/Q pasan a un conversor Digital a Analógico (DAC) y luego a una etapa de filtros pasabajas y amplificación. Luego sigue la etapa del mixer digital basado en el modulador en cuadratura. Finalmente, son 02 las salidas de los mixers que van

<sup>36</sup>MIMO: Multiple Input, Multiple Output. MIMO 2x2 es el modo de operación en el cual el sistema usa 2 antenas para Tx y 2 antenas para Rx.

<sup>37</sup>RFIC: *Radio frequency integrated circuit*, o circuito integrado de radio frecuencia.

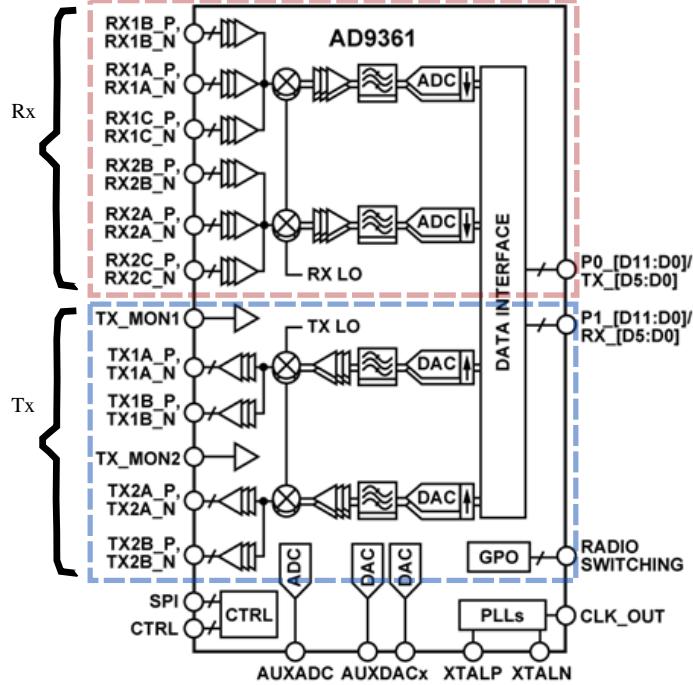


Fig. 2.33: Arquitectura RFIC AD9361. Extraído de [3].

a una etapa final de amplificación previo al puerto RF de salida.

En este punto, las señales I y Q se recombinan y modulan en la frecuencia portadora para su transmisión a la etapa RF. La señal combinada también pasa a través de filtros analógicos que también proporcionan filtrado de banda adicional, y luego la señal se transmite al amplificador de salida. Cada canal de transmisión proporciona un amplio rango de ajuste de atenuación con granularidad fina para ayudar a los diseñadores a optimizar la relación señal-ruido (SNR).

El FPGA es un circuito integrado diseñado para ser programable por el usuario. En relación a SDR, es un circuito digital cuyo funcionamiento puede ser definida por el usuario y se localiza entre el ADC y el software de procesamiento digital de datos.

Los cientos de miles de celdas lógicas de un FPGA pueden configurarse para que el dispositivo se comporte como un circuito digital y, además, en configuraciones que permiten el procesamiento de varios circuitos digitales en paralelo. El FPGA del USRP B210 es un Xilinx Spartan 6.

El driver UHD provee el control necesario para transportar las muestras de onda hacia y desde el hardware USRP así como también el control de varios parámetros del radio (ejemplo: tasa de muestras, frecuencia central, ganancia, etc.).

El driver UHD GPP y el código de firmware está escrito en C/C++ mientras que el código desarrollado para el FPGA está escrito en el lenguaje Verilog. Existe una interfaz

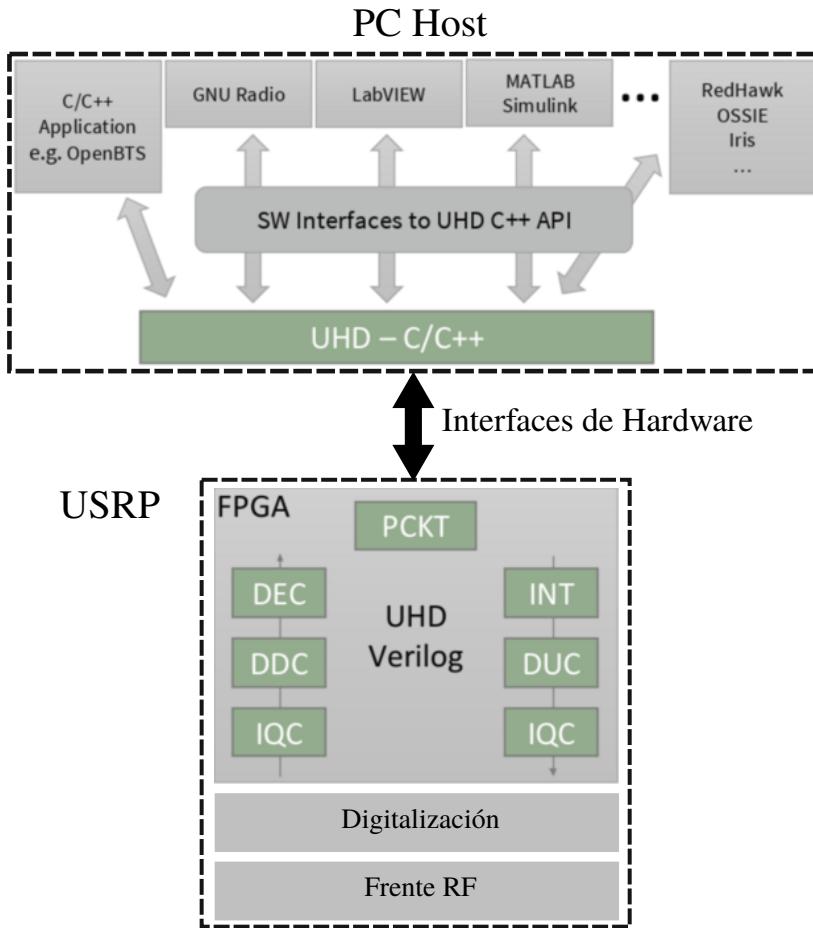


Fig. 2.34: Diagrama de bloques del funcionamiento del *driver* UHD, controlador del USRP B210.

de aplicación programable basado en C/C++ que puede hacer de interfaz con otro entorno de software, tal como es el caso de GNU Radio, o un usuario puede simplemente construir aplicaciones personalizadas de procesamiento de señal directamente en el API C/C++ de UHD. La Figura 2.34 muestra el concepto descrito anteriormente.

## CAPÍTULO 3

### IMPLEMENTACIÓN DEL PROTOTIPO DE RED GPRS.

El presente capítulo detalla los requerimientos de hardware y software, así como los pasos, necesarios para la implementación de una red de acceso GPRS para conectar dispositivos.

#### 3.1 Requerimientos de Hardware y Software.

La implementación de OpenBTS se realiza sobre un computador con procesador AMD de 64 bits (x86-64), con sistema operativo basado en Linux. Este servidor puede ser una máquina física o una máquina virtual (Virtualbox / VMWare).

Entre los requerimientos mínimos más importantes a nivel de hardware están el computador genérico con procesador Intel o AMD i5 de cuarta generación o mejor, memoria RAM mínima de 2GB, y contar con una interfaz USB 3.0 para la conexión del periférico de radio tal como se muestra en la Figura 3.1. La interfaz USB 3.0 es necesaria debido a la cantidad y tamaño de las muestras que se transfieren del computador hacia el USRP B210, o viceversa.

En general, la capacidad de procesamiento del servidor debe soportar los procesos de modulación y demodulación de señales de modo que no se produzcan retardos por algún *delay* de índole computacional [31].



Fig. 3.1: Setup de Hardware para la implementación de OpenBTS.

El periférico de radio es un USRP B210, dispositivo SDR cuya función de generación y recepción de señales de radiofrecuencia es utilizado para crear la interfaz de radio del prototipo de red GPRS. OpenBTS puede ser implementado con diferentes dispositivos SDR de diversos fabricantes tales como *Ettus Research, Fairwaves, Nuand* y *Range Networks*. El

dispositivo SDR elegido es un USRP B210 del fabricante *Ettus Research* [34], [10]. Las antenas que van conectadas en los puertos Rx y Tx del USRP B210 son omnidireccionales con una ganancia en el rango de 1dBi a 3dBi.

### **3.2 Instalación e inicialización de OpenBTS.**

La implementación de OpenBTS se realiza en una máquina virtual utilizando VirtualBox como software de virtualización [35]. La referencia [36] explica como habilitar, en la BIOS del HOST<sup>1</sup>, la opción de virtualización de hardware.

El anexo A explica como instalar el sistema operativo Ubuntu 16.04 versión 64 bits en una máquina virtual de VirtualBox. Los mismos pasos pueden usarse para instalar Ubuntu 16.04 directamente en el HOST.

Luego de haber creado la máquina virtual, en un terminal de comandos se procede a ingresar la siguiente secuencia de comandos. Es importante indicar que, a menos que explícitamente se indique algo diferente, los siguientes comandos se deben ingresar a la consola, presionar ENTER y esperar que el computador nos muestre el *prompt* nuevamente para poder ingresar el siguiente comando.

#### **3.2.1 Instalación de librerías dependencias.**

Para implementar OpenBTS, primero se deben instalar los siguientes programas o librerías denominados *dependencias*, pues OpenBTS depende de ellos para poder ser implementado:

```
gprs@ubuntu:~$ sudo apt-get install software-properties-common python-software-properties
```

Lo siguiente es darle a conocer al sistema la dirección de los repositorios donde se aloja el código fuente de la aplicación Git y de la librería ZeroMQ, para luego proceder a actualizar las librerías del sistema:

```
gprs@ubuntu:~$ sudo add-apt-repository ppa:git-core/ppa
gprs@ubuntu:~$ sudo add-apt-repository ppa:chris-lea/zeromq
gprs@ubuntu:~$ sudo apt-get update
```

Luego, se procede a instalar la aplicación *git*, el cual es un gestor de descargas para obtener software de los repositorios de *GitHub*[37]:

```
gprs@ubuntu:~$ sudo apt-get install git
```

Una vez que se cuenta con la aplicación *git*, se procede a descargar el *script* de instalación de OpenBTS [33]:

---

<sup>1</sup>Por lo general, una máquina virtual denominada GUEST, utiliza parte de los recursos físicos del computador físico donde está instalado VirtualBOX, denominado HOST.

```
gprs@ubuntu:~$ git clone https://github.com/RangeNetworks/dev.git
```

Este comando procederá a descargar una carpeta de nombre `dev` a la cual se debe acceder y ejecutar el *script* `clone.sh`, el cual sirve para descargar los diferentes componentes de OpenBTS.

```
gprs@ubuntu:~$ cd dev
gprs@ubuntu:~/dev$ ./clone.sh
```

El *script* `clone.sh` copia cada archivo del código de OpenBTS en el servidor local. Luego del clonado de archivos, se procede a seleccionar la versión de OpenBTS que se desea instalar. En este caso, se desea trabajar con la última versión de código (v5.0):

```
gprs@ubuntu:~/dev$ ./switchto.sh 5.0
```

Una vez que se tiene el código actualizado en todos los componentes de OpenBTS, se procede a compilar el código. Se debe especificar el tipo de dispositivo SDR que se utilizará en dicha instalación, de modo que se optimizan la configuración del driver de control del periférico de radio (UHD) [31]:

```
gprs@ubuntu:~/dev$ ./build.sh B210
```

Luego de que el *script* anterior ejecute todas sus instrucciones, se puede apreciar que en el carpeta de instalación se ha creado una carpeta de nombre `BUILDS` y dentro de ella un subdirectorio cuyo nombre contiene la fecha y hora de creación. Se debe proceder a entrar a esa carpeta:

```
gprs@ubuntu:~/dev$ cd BUILDS/2014-07-29--20-44-51/
```

e instalar todos los binarios que contenga:

```
gprs@ubuntu:~/dev/BUILDS/2014-07-29--20-44-51$ sudo dpkg -i *.deb
```

Si la terminal muestra un error en la instalación de los paquetes anteriores, esto se debe a un problema de dependencias faltantes. Para resolverlo, simplemente se debe ejecutar:

```
gprs@ubuntu:~/dev/BUILDS/2014-07-29--20-44-51$ sudo apt-get install -f
```

Es posible que, al momento de realizar la instalación de los componentes de OpenBTS con los archivos `*.deb`, se solicite permisos para sobre-escribir ciertos archivos de configuración de los componentes que se están instalando. En todos los casos se debe responder el *prompt* con "Y" y un "ENTER".

Finalizada la instalación de los drivers y OpenBTS, es necesario conectar el USRP B210 al puerto USB 3.0. En [38] se muestra como conectar un periférico a una máquina virtual.

Lo siguiente es verificar el correcto funcionamiento del driver UHD y para ello se deben ejecutar los siguientes comandos [39]:

Comando para realizar una búsqueda de todos los USRPs conectados:

```
gprs@ubuntu:~$ uhd_find_devices
linux; GNU C++ version 5.3.1 20151219; Boost_105800;
UHD003.009.002-0-unknown
-----
-- UHD Device 0
-----
Device Address:
type: b200
name: KUDOS
serial: KUDOS
product: B210
```

Comando para verificar el funcionamiento del USRP:

```
gprs@ubuntu:~$ uhd_usrp_probe
root@researcher:/usr/local/src/openbts_systemd_scripts# uhd_usrp_probe
linux; GNU C++ version 5.3.1 20151219; Boost_105800;
UHD_003.009.002-0-unknown
-- Detected Device: B210
-- Loading FPGA image: /usr/share/uhd/images/usrp_b210_fpga.bin... 25%
```

Con ello, se confirma que el USRP B210 está siendo reconocido e inicializa sus tarjetas RF sin problemas. A continuación se procede a descargar unos *scripts* muy útiles para facilitar la inicialización de OpenBTS y el respectivo uso de su línea de comandos [40].

```
gprs@ubuntu:~$ git clone https://github.com/nadiia-kotelnikova/openbts_systemd_scripts.git
```

Entrar a la carpeta recién descargada y acceder a la carpeta `systemd`:

```
gprs@ubuntu:~$ cd /openbts_systemd_scripts/systemd
gprs@ubuntu:~/openbts_systemd_scripts/systemd$ ls
asterisk.service    sipauthserve.service
openbts.service     smqueue.service
```

Proceder a copiar todos los archivos de esa carpeta a la carpeta `/etc/systemd/system/`:

```
gprs@ubuntu:~/openbts_systemd_scripts/systemd$ sudo cp * /etc/systemd/system/
gprs@ubuntu:~/openbts_systemd_scripts/systemd$ cd ..
```

Reiniciar el servidor Linux para que los cambios y configuraciones se reflejen en el nuevo inicio de sesión del sistema.

```
gprs@ubuntu:~/openbts_systemd_scripts/systemd$ sudo reboot
```

Una vez reiniciado el sistema, se procede a obtener una terminal para inicializar OpenBTS. Se debe acceder nuevamente a la carpeta `openbts_systemd_scripts` y ejecutar el *script* `./openbts-start.sh` con privilegios de superusuario y si no existe ningún problema, seinizian los procesos *Asterisk*, *Sipauthserve*, *Smqueue* y *OpenBTS*.

```
gprs@ubuntu:~/openbts_systemd_scripts$ sudo ./openbts-start.sh
Asterisk PID: 4143
Sipauthserve PID: 4146
Smqueue PID: 4185
OpenBTS PID: 4146
```

Si por algún motivo el proceso `Sipauthserve` no llega a inicializarse (no tiene PID), entonces es necesario detener todos los procesos (`sudo ./openbts-stop`), y luego nuevamente iniciar OpenBTS (`sudo ./openbts-start`). Finalmente, se puede acceder a la interfaz de línea de comandos (CLI<sup>2</sup>) para configurar OpenBTS mediante la ejecución del script `./openbts-cli.sh` (sin necesidad de privilegios de superusuario):

```
gprs@ubuntu:/openbts_systemd_scripts/$ ./openbts-cli.sh
OpenBTS Command Line Interface (CLI) utility
Copyright 2012, 2013, 2014 Range Networks, Inc.
Licensed under GPLv2.
Includes libreadline, GPLv2.
Connecting to 127.0.0.1:49300...
Remote Interface Ready.
Type:
"help" to see commands,
"version" for version information,
"notices" for licensing information,
"quit" to exit console interface.
OpenBTS>
```

El CLI está listo para recibir comandos de configuración o consultas de información. A continuación se describen múltiples configuraciones necesarias en el *GUEST* y en OpenBTS para un correcto funcionamiento de la plataforma.

### **3.2.2 Configuración para la habilitación del servicio GPRS.**

Por defecto, OpenBTS no tiene el servicio GPRS habilitado. En ese sentido, se debe modificar el parámetro `GPRS.Enable` a 1 para habilitar el servicio. Es necesario reiniciar el sistema para visualizar los cambios:

```
OpenBTS> devconfig GPRS.Enable
GPRS.Enable 1
- description: If enabled, GPRS service is advertised in the C0T0 beacon, and GPRS
  service may be started on demand. See also GPRS.Channels.*.
- type: boolean
- default value: 0
- visibility level: customer - can be freely changed by the customer without any detriment
  to their system
- static: 1
- valid values: 0 = disabled
- valid values: 1 = enabled

GPRS.Enable is static; change takes effect on restart

OpenBTS> devconfig GPRS.Enable 1
```

Con la configuración anterior y luego de haber reiniciado OpenBTS, la interfaz de radio comienza a reservar un determinado número de *timeslots* (determinado por el parámetro `GPRS.Channels.Min.C0`) para dedicarlos estrictamente a transportar señalización o data GPRS.

### **3.2.3 Configuración para la habilitación del registro en la red.**

OpenBTS no tiene el acceso habilitado por defecto. Esta medida es preventiva, pues si la red usa un ARFCN de uso comercial habría la posibilidad de que cualquier dispositivo móvil pueda conectarse. Este escenario no es deseable pues puede entorpecer las pruebas, además de los temas de seguridad.

---

<sup>2</sup>Command Line Interface o interfaz para ingresar comandos en línea.

Sin embargo, el hecho de configurar OpenBTS para operar en una banda "limpia" como la DCS1800 es una medida preventiva para evitar que otros dispositivos se conecten a la red pues en Perú ningún operador tiene asignada esa banda. Debido a ello, es posible configurar el registro en red de manera abierta (sin filtros a nivel de IMSI) de modo que cualquiera de los MS pueda registrarse en la red sin tener que pasar por filtros de acceso.

```
OpenBTS> devconfig Control.LUR.OpenRegistration
Control.LUR.OpenRegistration 1
- description: This value is a regular expression. Any handset with an IMSI matching
  the regular expression is allowed to register, even if it is not provisioned. By
  default, this feature is disabled. To enable open registration, specify a regular
  expression to match. E.g. ^{}001, which matches any IMSI starting with 001, the MCC for
  test networks. To disable open registration again, execute "unconfig Control.LUR.
  OpenRegistration".
- type: regular expression (optional)
- visibility level: customer warn - a warning will be presented and confirmation required
  before changing this sensitive setting
- static: 0

OpenBTS> devconfig Control.LUR.OpenRegistration .*
Control.LUR.OpenRegistration changed from "disable" to ".+"
```

### 3.2.4 Configuración del canal ARFCN.

Es necesario configurar el canal ARFCN en la banda DCS1800 que utilizará el sistema. Para ello, se configuran los siguientes parámetros de radio en el CLI de OpenBTS:

El parámetro GSM.Radio.Band permite especificar la banda de operación. Se selecciona la opción DCS1800 en mención a la banda DCS en 1800 MHz:

```
OpenBTS> config GSM.Radio.Band
GSM.Radio.Band 1800
- description: The GSM operating band. Valid values are 850 for GSM850, 900 for
  PGSM900, 1800 for DCS1800 and 1900 for PCS1900. For non-multiband units, this value is
  dictated by the hardware and should not be changed.
- type: multiple choice
- default value: 900
- visibility level: customer warn - a warning will be presented and confirmation required
  before changing this sensitive setting
- static: 1
- valid values: 850 = GSM850
- valid values: 900 = PGSM900
- valid values: 1800 = DCS1800
- valid values: 1900 = PCS1900

GSM.Radio.Band is static; change takes effect on restart

OpenBTS> config GSM.Radio.Band 1800
```

Finalmente, una vez seleccionada la banda DCS1800, se puede proceder a seleccionar el canal ARFCN específico. En este caso, se elige el ARFCN 512 que consiste, según la Tabla 2.1, en un canal downlink en la frecuencia 1805.2MHz y un canal uplink en 1710.2MHz:

```

OpenBTS> config GSM.Radio.C0
GSM.Radio.C0 512
- description:      The C0 ARFCN. Also the base ARFCN for a multi-ARFCN configuration.
- type:             multiple choice
- default value:   51
- visibility level: customer site - these values are different for each BTS and should not
  be left default
- static:           1
- valid values:    512 = DCS1800 #512 : 1805.2 MHz downlink / 1710.2 MHz uplink
- valid values:    513 = DCS1800 #513 : 1805.4 MHz downlink / 1710.4 MHz uplink
...
- valid values:    885 = DCS1800 #885 : 1879.78 MHz downlink / 1784.78 MHz uplink
- scope:            value must be unique across all neighbors
GSM.Radio.C0 is static; change takes effect on restart

OpenBTS> config GSM.Radio.C0 512

```

### 3.2.5 Configuración de la identidad de la red y de la estación base.

La identidad de la red se determinada por los parámetros MCC<sup>3</sup> y MNC<sup>4</sup>, los cuales se anuncian en los mensajes de *broadcast* (BCH). Las simcards que van inseridas en los SIM800L poseen un identificador IMSI de la forma 90170XXXXXXXXXX, lo cual indica que el MCC es 901 y el MNC es 70.

```

OpenBTS> config GSM.Identity.MCC
GSM.Identity.MCC 901
- description:      Mobile country code; must be three digits. Defined in ITU-T E.212.
  Value of 001 for test networks.
- type:             string
- default value:   001
- visibility level: customer site - these values are different for each BTS and should not
  be left default
- static:           0
- valid val regex: ^[0-9]{3}$
- scope:            value must be the same across all nodes

OpenBTS> config GSM.Identity.MNC
GSM.Identity.MNC 70
- description:      Mobile network code, two or three digits. Assigned by your national
  regulator. 01 for test networks.
- type:             string
- default value:   01
- visibility level: customer site - these values are different for each BTS and should not
  be left default
- static:           0
- valid val regex: ^[0-9]{2,3}$
- scope:            value must be the same across all nodes

```

Se configuran los valores para el MCC y MNC:

```

OpenBTS> config GSM.Identity.MCC 901
GSM.Identity.MCC is already set to "901", nothing changed

OpenBTS> config GSM.Identity.MNC 70
GSM.Identity.MNC is already set to "70", nothing changed

```

### 3.2.6 Configuración del número de timeslots asignados para el servicio GPRS.

OpenBTS es una plataforma que emula una red GSM lo cual permite establecer llamadas de voz y mensajes SMS entre terminales que soporten la tecnología GSM. Sin embargo, OpenBTS también

<sup>3</sup>MCC: Mobile Country Code o Código Móvil de País.

<sup>4</sup>MNC: Mobile Network Code o Código Móvil de Red

ofrece el servicio de transferencia de datos vía GPRS. Para ello, es necesario que una vez habilitado el servicio GPRS (sub-sección 3.2.2), se especifique en el sistema cuántos *timeslots* de la trama TDMA serán dedicados a GPRS.

Tal como se describe en la sección 2.2, la trama TDMA está constituida por 8 *timeslots* de los cuales se pueden asignar desde 0 hasta 7 *timeslots* para el servicio GPRS.

La configuración por defecto de OpenBTS asigna al servicio GPRS un único *timeslot*. Este valor se incrementa a su máximo (7) debido a que el uso de OpenBTS es estrictamente para transportar data GPRS en la presente implementación. Posteriormente en la sección 4.3, este parámetro se modifica para encontrar la mejor configuración para soportar conexiones de múltiples MS de manera más eficiente.

```
OpenBTS> devconfig GPRS.Channels.Min.C0
GPRS.Channels.Min.C0 2      [default]
- description: Minimum number of channels allocated for GPRS service on ARFCN C0.
- units: channels
- type: value range
- default value: 2
- visibility level: customer tune - should only be changed to tune an installation to
                     better suit the physical environment or MS usage pattern
- static: 1
- valid values: from 0 to 7

GPRS.Channels.Min.C0 is static; change takes effect on restart

OpenBTS> devconfig GPRS.Channels.Min.C0 7
```

### 3.2.7 Configuración del número máximo de canales GPRS asignados a un MS.

OpenBTS permite configurar el número máximo de *timeslots* GPRS que se puede asignar a cada MS por trama TDMA. La asignación por defecto es de 1 *timeslot* por cada MS; sin embargo, usando el modo de configuración *multislot* es posible asignar desde 2 hasta 10 *timeslots* a cada usuario, tanto en *downlink* como *uplink*.

A continuación se muestra la descripción de ambos parámetros según OpenBTS:

```
OpenBTS> devconfig GPRS.Multislot.Max.Downlink
GPRS.Multislot.Max.Downlink 1
- description: Maximum number of channels used for a single MS in downlink.
- units: channels
- type: value range
- default value: 3
- visibility level: customer tune - should only be changed to tune an installation to
                     better suit the physical environment or MS usage pattern
- static: 0
- valid values: from 0 to 10

OpenBTS> devconfig GPRS.Multislot.Max.Uplink
GPRS.Multislot.Max.Uplink 1
- description: Maximum number of channels used for a single MS in uplink.
- units: channels
- type: value range
- default value: 2
- visibility level: customer tune - should only be changed to tune an installation to
                     better suit the physical environment or MS usage pattern
- static: 0
- valid values: from 0 to 10
```

y la configuración respectiva para utilizar todos los *timeslots* disponibles para GPRS, tanto en *downlink* como *uplink*.

```
OpenBTS> devconfig GPRS.Multislot.Max.Downlink 7
GPRS.Multislot.Max.Downlink is already set to "1", nothing changed

OpenBTS> devconfig GPRS.Multislot.Max.Uplink 7
GPRS.Multislot.Max.Uplink is already set to "1", nothing changed
```

La Tabla 3.1 muestra las configuraciones para la funcionalidad *Multislot* [15] que soporta OpenBTS:

TABLA N° 3.1: Configuraciones *Multislot* soportadas por OpenBTS.

Configuración Multislot	Máximo Número de Timeslots		
	Rx	Tx	Suma
1	1	1	2
2	2	1	3
3	2	2	3
4	3	1	4
5	2	2	4
6	3	2	4
7	3	3	4
8	4	1	5
9	3	2	5
10	4	2	5
11	4	3	5
12	4	4	5

La columna Rx describe el máximo número de *timeslots* en *downlink* que el MS puede usar por trama TDMA. De igual forma la columna Tx describe el máximo número de *timeslots* que se pueden asignar al MS para que transmita en *uplink*. En ciertas configuraciones *multislot*, la suma de los valores Rx y Tx no coincide con el valor de la columna Suma puesto que existe la regla que indica que un MS no puede tener asignado más *timeslots* que el valor indicado en la columna Suma [15].

### 3.2.8 Configuración de condiciones mínimas para el RSSI y SNR en el uplink.

OpenBTS permite configurar los requerimientos mínimos de las condiciones de radio para la señal *uplink*. El parámetro `GSM.Radio.RSSITarget` controla la potencia de transmisión del MS, de modo que la señal *uplink* llegue al USRP con un nivel mayor o igual al especificado en el parámetro `GSM.Radio.RSSITarget` y satisfaciendo la condición de la relación señal a ruido (SNR) impuesta por el parámetro `GSM.Radio.SNRTarget`.

El ruido que se detecta en la antena Rx se puede visualizar con el comando `noise`. Del siguiente comando se visualiza que el ruido del canal de recepción está en -80dB.

```
OpenBTS> noise
noise RSSI is -80 dB wrt full scale
MS RSSI target is -50 dB wrt full scale
INFO: the current noise level is acceptable.
```

Es ideal que el parámetro `GSM.Radio.RSSITarget` se configure con valores entre 6 y 10 dB por encima del piso de ruido. Por otro lado, el nivel de SNR<sup>5</sup> de la señal recibida se debe configurar con el parámetro `GSM.Radio.SNRTarget` y debe estar entre 6 a 20 dB.

```
OpenBTS> devconfig GSM.Radio.RSSITarget
GSM.Radio.RSSITarget -50      [default]
- description:      Target uplink RSSI (Received Signal Strength Indication) for MS power
control loop, in dB wrt to A/D full scale. The MS power control loop adjusts MS TXPWR
(transmit power) to try to keep RSSI at this level, or to satisfy GSM.Radio.SNRTarget,
whichever requires more power. Should be 6-10 dB above the noise floor.
- units:          dB
- type:           value range
- default value: -50
- visibility level: customer tune - should only be changed to tune an installation to
better suit the physical environment or MS usage pattern
- static:         0
- valid values:   from -75 to -25
```

```
OpenBTS> devconfig GSM.Radio.SNRTarget
GSM.Radio.SNRTarget 10      [default]
- description:      The MS power control loop adjusts MS TXPWR (transmit power) to try to
keep SNR (Signal to Noise Ratio) above this level.
- units:          ratio
- type:           value range
- default value: 10
- visibility level: customer tune - should only be changed to tune an installation to
better suit the physical environment or MS usage pattern
- static:         0
- valid values:   from 6 to 20
Configurar:
```

Se configura:

```
OpenBTS> devconfig GSM.Radio.RSSITarget -50
OpenBTS> devconfig GSM.Radio.SNRTarget 10
```

### 3.2.9 Configuración de la potencia de transmisión.

Al momento de inicializar OpenBTS, el módulo *transceiver* configura, entre muchos otros parámetros, la ganancia del amplificador RF ( $G_{usrp}$ ) en el circuito de transmisión del USRP B210.

La ganancia del amplificador y el parámetro *power* están relacionados por la siguiente relación:

$$G_{usrp} = 89.75 - power$$

Para visualizar esta configuración es necesario habilitar el LOG de eventos de OpenBTS. Para ello en el CLI se debe ejecutar el comando:

```
OpenBTS> config Log.Level INFO
Log.Level is already set to "INFO", nothing changed
```

y en otra terminal ejecutar el comando:

```
root@researcher:~# tail -f /var/log/OpenBTS.log
```

Al momento de reiniciar OpenBTS, el terminal que registra el *log* de eventos mostrará un mensaje como el siguiente:

---

<sup>5</sup>Relación señal a ruido.

```
May 27 14:35:04 researcher transceiver: INFO 3426:3446 2018-05-27T14:35:04.9 UHDDevice.cpp
:465:setTxGain: Set TX gain to 89.75dB
```

donde se indica que la ganancia del amplificador del USRP B210 ha sido configurada a un valor de 89.75 dB. Si, por ejemplo, en el CLI de OpenBTS se ejecuta el comando:

```
OpenBTS> power 10
current downlink power -10 dB wrt full scale
```

el mensaje que aparece en el *log* de eventos de OpenBTS es:

```
May 27 15:01:02 researcher transceiver: INFO 3426:3446 2018-05-27T15:01:02.3 UHDDevice.cpp
:465:setTxGain: Set TX gain to 79.75dB
May 27 15:01:02 researcher openbts: INFO 3094:3094 2018-05-27T15:01:02.3
PowerManager.cpp:36:pmSetAttenDirect: setting power to -10 dB at uptime=1679
```

lo que indica que la ganancia en el amplificador ahora tiene el valor de 79.75dB, es decir 10dB menos que el valor inicial. La sección 4.1 proporciona valores de la potencia de salida del USRP B210 con diferentes valores del parámetro *power*.

### 3.2.10 Configuración de parámetros relacionados al GGSN.

Para activar el servicio GPRS es preciso especificar un servidor DNS con el parámetro *GGSN.DNS* para que OpenBTS resuelva correctamente las direcciones IP de los servidores a los cuales los MS precisen enviar data. También es necesario deshabilitar el *firewall* configurado por defecto en OpenBTS (*GGSN.Firewall.Enable*) con el objetivo de hacer las pruebas lo más transparentes posibles (es decir que no haya ningún obstáculo para el envío de la data).

```
OpenBTS> devconfig GGSN.DNS
GGSN.DNS 8.8.8.8
- description:      The list of DNS servers to be used by downstream clients. By default,
                  DNS servers of the host system are used. To override, specify a space-separated list
                  of DNS servers, in IP dotted notation, eg: 1.2.3.4 5.6.7.8. To use the host system DNS
                  servers again, execute "unconfig GGSN.DNS".
- type:            space-separated list of IP addresses (optional)
- visibility level: customer warn - a warning will be presented and confirmation required
                     before changing this sensitive setting
- static:          1

GGSN.DNS is static; change takes effect on restart

OpenBTS> devconfig GGSN.Firewall.Enable
GGSN.Firewall.Enable 0
- description:      0=no firewall; 1=block MS attempted access to OpenBTS or other MS; 2=
                  block all private IP addresses.
- type:             multiple choice
- default value:   1
- visibility level: customer warn - a warning will be presented and confirmation required
                     before changing this sensitive setting
- static:          1
- valid values:    0 = Disable Firewall
- valid values:    1 = Block MS Access to OpenBTS and Other MS
- valid values:    2 = Block All Private IP Addresses

GGSN.Firewall.Enable is static; change takes effect on restart
```

Configurar:

```
OpenBTS> devconfig GGSN.DNS 8.8.8.8
OpenBTS> devconfig GGSN.Firewall.Enable 0
```

### 3.2.11 Configuración del redireccionamiento de paquetes.

Una configuración muy importante para el correcto redireccionamiento de paquetes IP hacia redes externas es la especificación de una regla de "nateo" (proveniente de NAT<sup>6</sup>), la cual especifica lo que se debe hacer con un paquete de datos originado en un MS y dirigido hacia una red ya sea local o externa o un paquete de datos que proviene de una red local o externa y que está dirigido a un MS.

La configuración se logra mediante la inclusión de una regla de "nateo" en la tabla IPTABLES de la siguiente manera:

```
root@researcher:~# iptables -t nat -A POSTROUTING -o enp0s3 -j MASQUERADE
```

La regla POSTROUTING indica la política de enrutamiento, el parámetro enp0s3 corresponde a la interfaz física de red por la cual se recibe o se envía el paquete y finalmente el parámetro MASQUERADE que indica la regla de "nateo" de dirección. Para verificar si la regla se ha aplicado, se realiza con el comando:

```
root@researcher:~# iptables -t nat -L -v -n
Chain PREROUTING (policy ACCEPT 1 packets, 182 bytes)
pkts bytes target     prot opt in     out     source          destination
Chain INPUT (policy ACCEPT 1 packets, 182 bytes)
pkts bytes target     prot opt in     out     source          destination
Chain OUTPUT (policy ACCEPT 4 packets, 376 bytes)
pkts bytes target     prot opt in     out     source          destination
Chain POSTROUTING (policy ACCEPT 3 packets, 300 bytes)
pkts bytes target     prot opt in     out     source          destination
1    76 MASQUERADE  all   -- *      enp0s3  0.0.0.0/0           0.0.0.0/0      <<<<<
```

### 3.2.12 Configuración de la habilitación para la captura de tráfico.

OpenBTS permite habilitar la captura de tráfico mediante la habilitación de los parámetros Control.GSMTAP.GSM y Control.GSMTAP.GPRS. Una que se habilitan ambos parámetros, es posible capturar el tráfico de señalización que intercambian los nodos sensores con la red mediante el uso de aplicaciones externas tales como Wireshark o Tcpdump.

```
OpenBTS> devconfig Control.GSMTAP.GSM
Control.GSMTAP.GSM 1
- description: Capture GSM signaling at L1/L2 interface via GSMTAP.
- type: boolean
- default value: 0
- visibility level: customer warn - a warning will be presented and confirmation required before changing this sensitive setting
- valid values: 0 = disabled
- valid values: 1 = enabled

OpenBTS> devconfig Control.GSMTAP.GPRS
Control.GSMTAP.GPRS 1
- description: Capture GPRS signaling and traffic at L1/L2 interface via GSMTAP.
- type: boolean
- default value: 0
- visibility level: customer warn - a warning will be presented and confirmation required before changing this sensitive setting
- valid values: 0 = disabled
- valid values: 1 = enabled
```

<sup>6</sup>NAT: Network Address Translation o Tabla de traducción de direcciones de red.

Configurar así:

```
OpenBTS> devconfig Control.GSMTAP.GSM 1
OpenBTS> devconfig Control.GSMTAP.GPRS 1
```

### 3.2.13 Configuración de visualización de logs.

OpenBTS permite la visualización de sus eventos en la forma de *logs*. Para ello es necesario configurar los archivos en los cuales se imprimen los logs:

1) Habilitar el DEBUG a nivel de GPRS.

```
OpenBTS> devconfig GPRS.Debug
GPRS.Debug 1
- description: Toggle GPRS debugging.
- type: boolean
- default value: 0
- visibility level: developer - should only be changed by developers to debug/optimize the implementation
- static: 0
- valid values: 0 = disabled
- valid values: 1 = enabled

OpenBTS> devconfig GPRS.Debug 1
GPRS.Debug is already set to "1", nothing changed
```

2) Especificar el archivo de *logs*:

```
OpenBTS> devconfig GGSN.Logfile.Name
GGSN.Logfile.Name /tmp/GGSN.log
- description: If specified, internet traffic is logged to this file. E.g. ggsn.log.
- type: file path (optional)
- visibility level: factory - set once at the factory, should never be changed
- static: 1

GGSN.Logfile.Name is static; change takes effect on restart

OpenBTS> devconfig GGSN.Logfile.Name /tmp/GGSN.log
```

3) Definir el alcance de los *logs* ya que estos pueden ser de diferente nivel: eventos notificativos, informativos, de emergencia, de alertas, etc.)

```
OpenBTS> devconfig Log.Level
Log.Level INFO
- description: Default logging level when no other level is defined for a file.
- type: multiple choice
- default value: NOTICE
- visibility level: customer - can be freely changed by the customer without any detriment to their system
- static: 0
- valid values: EMERG = EMERGENCY - report serious faults associated with service failure or hardware damage
- valid values: ALERT = ALERT - report likely service disruption caused by misconfiguration or poor connectivity
- valid values: CRIT = CRITICAL - report anomalous events that are likely to degrade service
- valid values: ERR = ERROR - report internal errors of the software that may result in degradation of service in unusual circumstances
- valid values: WARNING = WARNING - report anomalous events that may indicate a degradation of normal service
- valid values: NOTICE = NOTICE - report anomalous events that probably do not affect service but may be of interest to network operators
- valid values: INFO = INFORMATION - report normal events
- valid values: DEBUG = DEBUG - only for use by developers and will degrade system performance
```

Configurar el Log.Level a nivel INFO:

```
OpenBTS> devconfig Log.Level INFO
Log.Level is already set to "INFO", nothing changed
```

### 3.2.14 Visualización de logs.

El sistema operativo Linux permite visualizar los logs de cualquier sistema (incluido OpenBTS) de la siguiente manera

#### 1) Visualización de eventos de OpenBTS:

```
root@researcher:~# tail -f /var/log/OpenBTS.log
Aug  6 10:36:32 researcher openbts: INFO 11105:11550 2018-08-06T10:36:32.8 GSMCCCH.cpp:518:
    newPageAll: paging 0 mobile(s)
Aug  6 10:36:33 researcher transceiver: INFO 11414:11445 2018-08-06T10:36:33.8 Transceiver.
    cpp:827:writeClockInterface: ClockInterface: sending IND CLOCK 146845
Aug  6 10:36:33 researcher openbts: INFO 11105:11440 2018-08-06T10:36:33.8 TRXManager.cpp
    :126:clockHandler: CLOCK indication, current clock = 0:146845 new clock =146845
Aug  6 10:36:34 researcher transceiver: INFO 11414:11445 2018-08-06T10:36:34.8 Transceiver.
    cpp:827:writeClockInterface: ClockInterface: sending IND CLOCK 147064
Aug  6 10:36:34 researcher openbts: INFO 11105:11440 2018-08-06T10:36:34.8 TRXManager.cpp
    :126:clockHandler: CLOCK indication, current clock = 0:147064 new clock =147064
```

#### 2) Visualización de eventos del GGSN:

```
root@researcher:~# tail -f /tmp/GGSN.log
08:02:11.9:  GGSN.IP.TossDuplicatePackets=0
08:02:11.9:GGSN: DNS servers: 8.8.8.8 0.0.0.0
08:02:11.9:ip link set sgsntun up
08:02:12.2:ip route add to 192.168.99.0/24 dev sgsntun
08:02:12.4:SGSN:service loop policy=0 priority=0
08:02:12.4:SGSN:service loop policy=0 priority=0
08:30:52.1:ggsn: received proto=128 48 byte packet from 0.0.0.0 to 229.179.71.96 at
    08:30:52.1
08:30:52.1:ggsn: error: cannot find PDP context for incoming packet for IP dstaddr
    =229.179.71.96
09:34:13.2:ggsn: received proto=128 48 byte packet from 0.0.0.0 to 229.179.71.96 at
    09:34:13.2
09:34:13.2:ggsn: error: cannot find PDP context for incoming packet for IP dstaddr
    =229.179.71.96
```

### 3.2.15 Visualización del RSSI de los MS conectados vía GPRS.

El comando para poder visualizar el nivel de potencia con el que se recibe la señal (RSSI) en la estación base es `gprs list`. Este comando básicamente muestra diversos parámetros relacionados con la conexión GPRS de todos los MS. Uno de ellos es el nivel de señal RSSI. En el siguiente ejemplo, los MS #1, #2, y #3 presentan niveles RSSI promedio de -59dBm, -56dBm, y -53dBm respectivamente:

```

OpenBTS> gprs list
MS#1,TLLI=c0041001,78182a2b rrmode=PacketIdle Bytes:989108up/976766down Utilization=0 %
GMM Context: imsi=901700000023614 ptmsi=0x41001 tlli=0xc0041001 state=GmmRegisteredNormal
    age=12381 idle=9 IPs=192.168.99.1
TimingError=(-1.12 min=-1.54 max=-0.70 avg=-1.09 N=369961) RSSI=(-59 min=-75 max=-50 avg
    =-58.07 N=369961) CV=(59 min=0 max=63 avg=60.10 N=3296) ILev=(0) RXQual=(0 min=0 max=4
    avg=0.00 N=2533) SigVar=(0 min=0 max=63 avg=0.04 N=2533) ChCoding=(0)
dataER:.17% (128944) recent:.16% (142) low:.4% (57) tbfER:.03% (1584)
rrbpER:.02% (19155) recent:0% (24) low:.14% (7) ccchER:.0% (36) recent:0% (0)

MS#2,TLLI=c0041002,78873ded rrmode=PacketIdle Bytes:977618up/976586down Utilization=0 %
GMM Context: imsi=901700000023610 ptmsi=0x41002 tlli=0xc0041002 state=GmmRegisteredNormal
    age=12790 idle=13 IPs=192.168.99.2
TimingError=(-0.96 min=-1.53 max=-0.62 avg=-1.10 N=346741) RSSI=(-56 min=-75 max=-50 avg
    =-57.86 N=346741) CV=(57 min=0 max=60 avg=57.38 N=3175) ILev=(0) RXQual=(0) SigVar=(0
    min=0 max=19 avg=0.02 N=2460) ChCoding=(0)
dataER:.16% (125803) recent:.17% (143) low:1.0% (23) tbfER:.04% (1494)
rrbpER:.01% (18051) recent:0% (23) low:.2% (4) ccchER:1.0% (100) recent:0% (0)

MS#3,TLLI=c0041003 rrmode=PacketIdle Bytes:893734up/892080down Utilization=0 %
GMM Context: imsi=901700000023615 ptmsi=0x41003 tlli=0xc0041003 state=GmmRegisteredNormal
    age=12825 idle=15 IPs=192.168.99.3
TimingError=(-0.90 min=-1.52 max=-0.73 avg=-1.09 N=327365) RSSI=(-53 min=-70 max=-45 avg
    =-54.19 N=327365) CV=(55 min=50 max=61 avg=57.61 N=2966) ILev=(0) RXQual=(0) SigVar=(0
    min=0 max=63 avg=0.03 N=2310) ChCoding=(0)
dataER:.16% (115460) recent:.15% (142) low:.5% (41) tbfER:.03% (1403)
rrbpER:.01% (16938) recent:0% (23) low:.5% (2) ccchER:1.0% (112) recent:0% (0)

PDCH ARFCN=512 TN=1 FER=.5 %
PDCH ARFCN=512 TN=2 FER=.5 %
PDCH ARFCN=512 TN=3 FER=0 %
PDCH ARFCN=512 TN=4 FER=.01 %
PDCH ARFCN=512 TN=5 FER=.5 %
PDCH ARFCN=512 TN=6 FER=0 %
PDCH ARFCN=512 TN=7 FER=36 %

```

### 3.3 Transferencia de Paquetes de Datos sobre la red GPRS.

A continuación se describe la secuencia de pasos para establecer en el MS (SIM800L) una conexión de datos GPRS con alguna red y el envío de *pings* para verificar conectividad con la red externa. De la Figura 2.19, una vez que el MS es energizado, se debe seleccionar la banda de operación mediante el comando AT+CBAND=DCS\_MODE. Luego se puede hacer la consulta mediante el comando AT+CBAND? para verificar que el SIM800L está operando en dicha banda

```

AT+CBAND = DCS_MODE
OK

AT+CBAND?
+CBAND: DCS_MODE

OK

```

y posteriormente se selecciona el modo de funcionamiento AT+CFUN=1. Este paso es necesario, caso contrario el SIM800L permanecerá en el modo de funcionamiento AT+CFUN=0 en el cual no es posible la conexión GPRS. De igual forma se hace la verificación correspondiente con el comando

AT+CFUN?.

```

AT+CFUN = 1
OK

AT+CFUN?

```

```
+CFUN: 1
OK
```

Posteriormente es necesario realizar el registro en la red mediante el comando AT+CREG=1 y la consulta correspondiente para verificar que el registro fue exitoso con AT+CREG?.

```
AT+CREG = 1
OK

AT+CREG?
+CREG: 1,1
OK
```

Una vez que el SIM800L está registrado en la red se activa el procedimiento de "Attach" a la red GPRS mediante el comando AT+CGATT=1. Con el procedimiento "Attach" completo se consigue establecer un flujo temporal de bloques (TBFs) tanto en *uplink* y *downlink*. Con este paso, el MS realiza el procedimiento "GPRS Attach" de la Figura 2.11.

```
AT+CGATT = 1
OK

AT+CGATT?
+CGATT: 1
OK
```

Una vez que se realiza el "attach" con la red, el MS pasa al estado GPRS denominado como IP INITIAL. En este momento el dispositivo tiene asignados recursos de radio para iniciar una conexión de datos. Para ello es necesario que se configuren parámetros como el APN<sup>7</sup> el cual puede ser cualquier texto no mayor a 50 bytes y que hace referencia al GGSN.

Al momento de activar el contexto PDP, el SGSN obtiene la dirección del GGSN a partir del APN. En el caso de OpenBTS, el APN puede ser el texto que se recomienda en los manuales de los chips SIMCOM, APN="CMNET". En las redes comerciales por temas de seguridad el APN debe ir acompañado por datos de autenticación (usuario y contraseña), sin embargo en OpenBTS no es necesario.

```
AT+CSTT="CMNET"
OK

AT+CSTT?
+CSTT: "CMNET", "", ""
OK
```

Una vez definido el APN, el SIM800L pasa al estado GPRS denominado IP START. El dispositivo ha iniciado el stack de protocolos TCP/IP y es necesario ejecutar el comando AT+CIICR para activar la conexión inalámbrica vía GPRS. En el siguiente log se puede apreciar como, luego del comando AT+CIICR, el CIPSTATUS cambia de IP START a IP GPRSACT (pasando por IP CONFIG).

---

<sup>7</sup>APN: Access Point Name o Nombre del Punto de Acceso

```
AT+CIPSTATUS
OK

STATE: IP START

AT+CIICR
OK

AT+CIPSTATUS
OK

STATE: IP GPRSACT
```

El siguiente paso es obtener una dirección IP por parte de la red GPRS mediante el comando AT+CIFSR. El siguiente log muestra que luego de ejecutar el comando se obtiene la IP 192.168.99.1 para el SIM800L y el estado GPRS cambia a IP STATUS.

```
AT+CIPSTATUS
OK

STATE: IP GPRSACT

AT+CIFSR
192.168.99.1

AT+CIPSTATUS
OK

STATE: IP STATUS
```

Una vez que el SIM800L cuenta con dirección IP, se puede proceder a enviar *pings* hacia algún HOST destino. En el siguiente ejemplo se puede apreciar el comando para enviar 10 *pings* hacia la IP destino 8.8.8.8 (*google*).

```
AT+CIPPING="8.8.8.8",10

+CIPPING: 1,"8.8.8.8",27,60
+CIPPING: 2,"8.8.8.8",4,60
+CIPPING: 3,"8.8.8.8",4,60
+CIPPING: 4,"8.8.8.8",4,60
+CIPPING: 5,"8.8.8.8",4,60
+CIPPING: 6,"8.8.8.8",5,60
+CIPPING: 7,"8.8.8.8",4,60
+CIPPING: 8,"8.8.8.8",4,60
+CIPPING: 9,"8.8.8.8",4,60
+CIPPING: 10,"8.8.8.8",4,60
OK
```

La Figura 3.2 muestra la secuencia de entidades IP que recorren los paquetes *ping* generados en el MS. Los paquetes IP luego de llegar al SGSN/GGSN pasan hacia la *wildcard* o tarjeta de red de la máquina virtual sobre el cual corre el sistema operativo Ubuntu, sobre el cual está instalado OpenBTS.

La configuración del `iptables` de la sección 3.2.11 permite el direccionamiento de los paquetes provenientes de la red 192.168.99.0/24 hacia redes externas pasando a través de la configuración de red interna propia de la aplicación de virtualización que se esté usando (en este caso VirtualBox), para finalmente obtener la salida hacia internet.

Con esta secuencia de pasos se ha conseguido establecer una conexión GPRS y comprobar la conectividad con el envío de datos en forma de *pings*. En el Anexo 2 se usan muchos de estos pasos

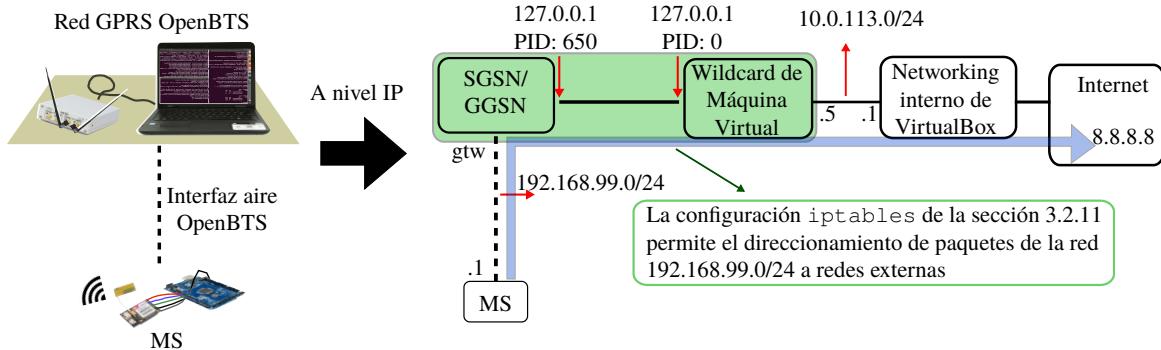


Fig. 3.2: Diagrama de bloques para explicar el *networking* que recorren los paquetes IP hasta internet.

para confeccionar el *script* que se usa para implementar la conexión GPRS y el envío datos de las pruebas del capítulo 4.

## CAPÍTULO 4

### PRUEBAS Y RESULTADOS.

En el presente capítulo se describen escenarios de pruebas realizadas a la red GPRS propuesta. Se presentan los resultados correspondientes y se realiza una análisis de los mismos. Los aspectos de la tecnología en los cuales se enfocan las pruebas son:

- Análisis de la señal física generada computacionalmente por OpenBTS y transmitida por el USRP B210.
- Verificación de los mensajes de señalización de una conexión típica de un MS con la red GPRS propuesta.
- Pruebas sobre la disponibilidad de acceso y capacidad de red con diferentes configuraciones en la asignación de canal.
- Implementación de una aplicación típica para IoT sobre la red GPRS propuesta.

#### 4.1 Análisis de la señal física del prototipo de red GPRS.

El método para analizar la señal transmitida por la red GPRS consiste en conectar el puerto Tx del USRP B210 al puerto RF Input del equipo de medición Anritsu MS2721B, tal como se muestra en la Figura 4.1.

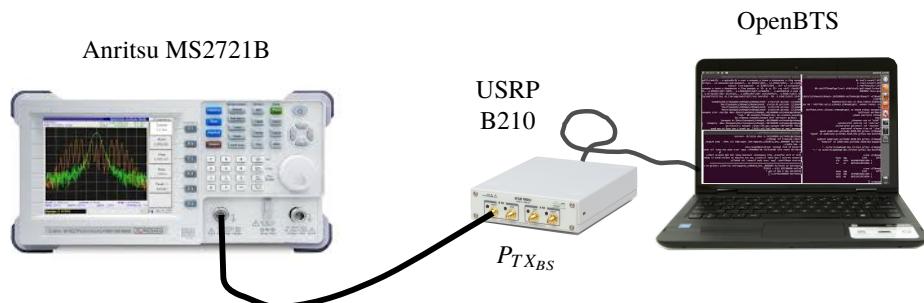


Fig. 4.1: Setup para la captura de la señal generada por la red GPRS propuesta.

La Figura 4.2 muestra el espectro de frecuencias de la señal capturada por el analizador de espectro. En ella se puede apreciar que el piso de ruido (marcadores 1 y 3) está alrededor

de -57dBm y la intensidad máxima de la señal en la frecuencia central 1805.2MHz (marcador 2) llega a los 6.4dBm, cuando el parámetro `power` de OpenBTS está configurado a 0.

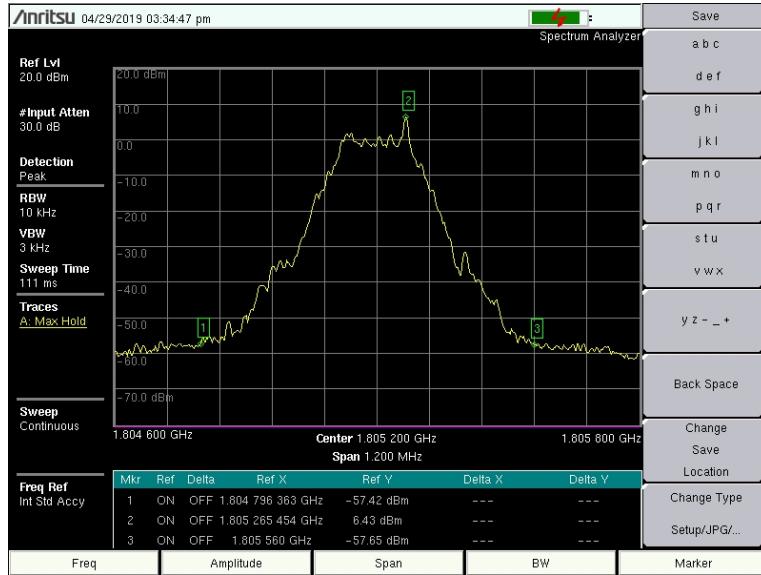


Fig. 4.2: Señal capturada con el Analizador de espectro Anritsu MS2721B.

Realizando una inspección rápida a la Figura 4.2, se puede comprender que cada división horizontal equivale a 120kHz, por lo cual la señal debería comprender las 2 divisiones centrales. Sin embargo, se puede apreciar que la señal transmitida termina ocupando hasta 4 divisiones horizontales, es decir 480 kHz.

La fuga de energía hacia las frecuencias laterales de la señal puede ser producto de un filtrado poco eficiente en la etapa de generación de los pulsos (*pulse shaping*). Un posible trabajo futuro puede ser el elaborar un mejor filtrado de la señal.

Otro efecto que se percibe es el alto nivel del piso de ruido, el cual llega a niveles de alrededor de -57dBm. Este valor es poco conveniente, pues un piso de ruido tan alto como -57dBm causa limitaciones en el radio de cobertura dado que la relación señal a ruido decrece más rápido cuando existe demasiado ruido en el canal.

La medición de la intensidad de potencia de la señal se realiza para diferentes canales ARFCNs de la banda DCS1800, utilizando diferentes valores del parámetro `power` (sección 3.2.9). Los valores obtenidos se presentan en la Tabla 4.1.

El objetivo de esta medición es conocer la máxima potencia de transmisión que se obtiene del puerto Tx del USRP B210, así como también su relación con el parámetro `power`. `Power` controla la ganancia del amplificador de potencia del USRP B210 y por ende la potencia que este transmite. A  $P_{TX_{BS}}$  se le adiciona entre 1dBi a 2dBi correspondientes a la ganancia

real de una antena omnidireccional para obtener la potencia equivalente isotrópica radiada, o también denominada PIRE.

TABLA N° 4.1: Valores de la Potencia de Transmisión del USRP B210 en diferentes valores del parámetro `power` de OpenBTS, y diferentes canales ARFCN.

power	$P_{TX_{BS}}$ , Potencia de Tx (dBm) según el ARFCN									
	512	549	586	623	660	697	734	771	808	845
0	6.4	6.1	6.2	6.1	6.3	6.2	6.1	6.4	5.9	5.8
5	2.5	2.4	2.2	2.3	2.5	2.2	1.9	2.1	2.3	2.7
10	-4.1	-3.7	-4.3	-4.4	-4.9	-3.9	-4.5	-5.1	-5.0	-4.9
15	-10.7	-10.7	-9.8	-9.8	-10.1	-10.1	-10.2	-9.9	-10.1	-10.0
20	-12.6	-12.4	-12.9	-12.8	-13.8	-13.6	-12.6	-12.8	-13.6	-14.8
25	-17.8	-17.9	-18.6	-17.8	-17.6	-18.6	-17.8	-18.6	-19.2	-19.8
30	-22.8	-23.2	-22.9	-23.5	-22.2	-22.8	-23.0	-24.8	-23.6	-24.2

De las mediciones de la Tabla 4.1, se concluye que la máxima potencia de transmisión es 6.4 dBm cuando el parámetro `power` es 0 y usando el canal ARFCN 512. En un escenario de producción, es preferible no utilizar toda la ganancia del amplificador de potencia del USRP B210, y por lo tanto es importante conocer los valores de potencia con otros valores de `power`.

La Figura 4.3 muestra curvas de potencia de transmisión del USRP B210 correspondientes a diferentes canales ARFCN en la banda DCS1800. Se puede apreciar que la potencia de transmisión disminuye conforme se aumente el parámetro `power` y, en general, la potencia de transmisión es lineal en toda la banda.

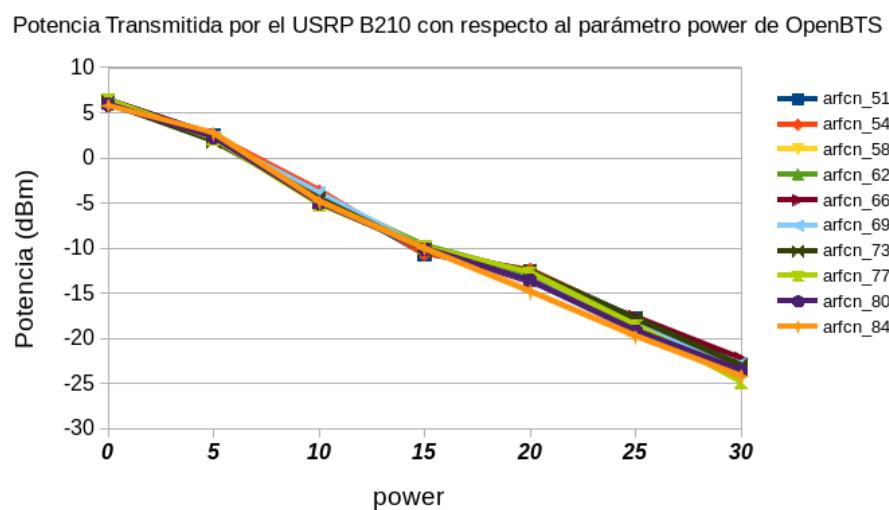


Fig. 4.3: Señal capturada con el Analizador de espectro Anritsu MS2721B.

## 4.2 Verificación de los mensajes de señalización.

En la presente prueba, se realiza la conexión de un MS con la red GPRS propuesta de igual manera que se realizó en la sección 3.3. El objetivo es capturar el intercambio de señalización entre el MS y la red, desde la red como referencia. Esta prueba servirá para verificar que se realizan los procedimientos de conexión de acuerdo al estándar GPRS descrito en la sección 2.2.

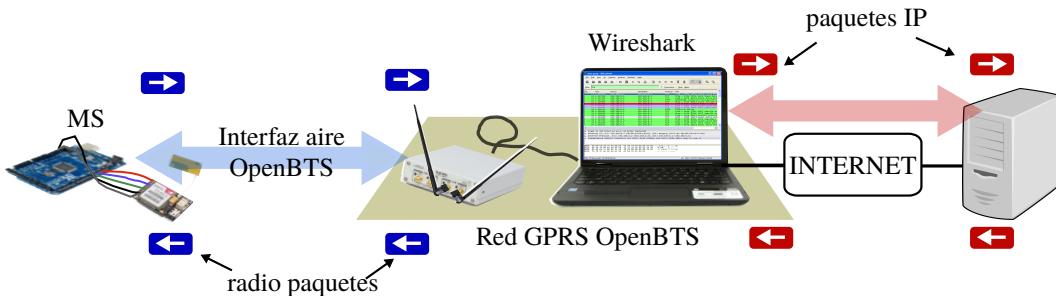


Fig. 4.4: Esquema de pruebas para captura de paquetes IP en la red GPRS.

La Figura 4.4 muestra el esquema de pruebas para esta sección. Se instala la aplicación Wireshark[41] en el computador donde está instalada la plataforma OpenBTS. Se realiza una conexión GPRS entre el MS y la red y se interceptan los paquetes de datos que transitan por el puerto loopback (`10: 127.0.0.1`) del computador.

Una vez interceptados los paquetes en el evento de la conexión GPRS, se procede a configurar los filtros mostrados en la tabla 4.2 en la barra de filtrado de paquetes de Wireshark [42]:

TABLA N° 4.2: Filtros aplicados al archivo `pcapng` capturado en Wireshark para poder visualizar la señalización de mensajes vista desde la red GPRS.

Filtro	Detalle
<code>(gsmtap or sip) and !icmp</code>	Filtrar paquetes con cabeceras GSM y SIP, protocolos de OpenBTS
<code>!(gsm_a.dtap.msg_rr_type == 0x21)</code>	Remueve mensajes <i>Paging Request Type 1</i> del canal CCCH, capa GSM
<code>!(gsm_a.dtap.msg_rr_type == 0x19)</code>	Remueve mensajes <i>System Information Type 1</i> del canal CCCH, capa GSM
<code>!(lapdm.control.f == 1)</code>	Remueve mensajes <i>System Information Type 5/6 y Measurement Reports</i> , capa LAPDm
<code>!(gsmtap.chan_type == 1)</code>	Remueve diversos paquetes canal BCCH, capa GSM
<code>!(gsmtap.chan_type == 135)</code>	
<code>!(lapdm.control_field == 0x03)</code>	
<code>!(lapdm.control_field == 0x21)</code>	
<code>!(lapdm.control_field == 0x50)</code>	
<code>!(lapdm.control_field == 0x30)</code>	
<code>!(lapdm.control_field == 0x29)</code>	
<code>!(lapdm.control_field == 0x22)</code>	
<code>!(lapdm.control_field == 0x61)</code>	
<code>!(lapdm.control_field == 0x69)</code>	
<code>!(lapdm.control_field == 0x81)</code>	
<code>!(lapdm.control_field == 0x53)</code>	

La Tabla 4.3 muestra los mensajes capturados y filtrados en *Wireshark*. El intercambio de mensajes inicia con un *Immediate Assignment* el cual seguramente es la respuesta de la red al mensaje *Random Access Request* del MS. A continuación, el MS envía su código IMSI y otros parámetros en el mensaje *Location Updating Request*.

La red recibe el IMSI del MS y verifica si dicho IMSI está registrado en la red de OpenBTS. Si es la primera conexión, es muy probable que el MS no esté registrado. Por ello, la red procede a solicitar al MS su código IMEI<sup>1</sup> a través del mensaje *Identity Request*. El MS responde con el mensaje *Identity Response* el cual contiene el IMEI del MS.

Una vez que la red recibe el IMEI y registra al MS en la red, envía el mensaje *Location Updating Accept* indicando al MS que su registro en la red ha sido exitoso y posteriormente, dado que el MS no solicita más servicios de la red, procede a enviar el mensaje *Channel Release* para liberar los recursos de radio por donde se estableció la señalización hasta ahora mencionada.

Luego, se pueden apreciar hasta cuatro mensajes consecutivos *Immediate Assignment* originados en la red hacia el MS. Estos mensajes son respuestas a los mensajes *Random Access Request* que el MS vuelve a enviar a la red para establecer conexión. En los tres primeros mensajes, es posible que el MS no consiga decodificar correctamente los mensajes *Immediate Assignment* y recién al cuarto mensaje es cuando el MS contesta a la red con un mensaje *GPRS UL: Packet Resource Request*.

Con el *GPRS UL: Packet Resource Request*, el MS solicita recursos de radio para establecer un nuevo TBF en *uplink*. La red responde con el mensaje *GPRS DL: Packet Uplink Assignment* en el cual concede los recursos necesarios al MS.

Posteriormente, la red y el MS intercambian mensajes *Packet Control Acknowledgement* y *Packet Downlink Dummy Control Block*, los cuales sirven para el control y verificación del TBF establecido. A continuación, sigue el establecimiento del TBF en el sentido *Downlink* que la red gestiona a través del mensaje *GPRS DL: Packet Downlink Assignment*. En este punto, el MS y la red han establecido TBFs bidireccionales para el transporte de datos de capas superiores.

Del intercambio de mensajes entre el MS y la red GPRS propuesta mostrado en la Tabla 4.3, se puede inferir que el MS obtiene una conexión de datos GPRS mediante la activación de los procedimientos descritos en la sección 2.2.

---

<sup>1</sup>IMEI: International Mobile Station Equipment Identity, o identidad internacional del equipo móvil.

TABLA N° 4.3: Señalización de la conexión de un MS capturada en el Host que contiene a OpenBTS.

No.	Tiempo	Origen	Destino	Protocolo	Tamaño	Información	Sentido
1891	14.166577626	127.0.0.1	127.0.0.1	GSMTAP	89	(CCCH) (RR) Immediate Assignment	Downlink
1918	14.335664491	127.0.0.1	127.0.0.1	LAPDm	83	U_P, func=SABM(DTAP) (MM) Location Updating Request	Uplink
1926	14.3.36779137	127.0.0.1	127.0.0.1	LAPDm	89	I, N(R)=0, N(S)=0DTAP (MM) Identity Request	Downlink
2112	15.2.71456479	127.0.0.1	127.0.0.1	LAPDm	83	I, N(R)=1, N(S)=0DTAP (MM) Identity Response	Uplink
2345	16.355418564	127.0.0.1	127.0.0.1	LAPDm	89	I, N(R)=1, N(S)=2DTAP (MM) Location Updating Accept	Downlink
2920	19.389607015	127.0.0.1	127.0.0.1	LAPDm	89	I, N(R)=1, N(S)=3DTAP (RR) Channel Release	Downlink
3003	19.837607263	127.0.0.1	127.0.0.1	GSMTAP	89	(CCCH) (RR) Immediate Assignment	Downlink
3471	23.342097726	127.0.0.1	127.0.0.1	GSMTAP	89	(CCCH) (RR) Immediate Assignment	Downlink
3906	26.646022852	127.0.0.1	127.0.0.1	GSMTAP	89	(CCCH) (RR) Immediate Assignment	Downlink
4338	29.9.38496314	127.0.0.1	127.0.0.1	GSMTAP	89	(CCCH) (RR) Immediate Assignment	Downlink
4361	30.0638865793	127.0.0.1	127.0.0.1	GSM RLC/MAC	83	GPRS UL:PACKET_RESOURCE_REQUEST	Uplink
4363	30.065186387	127.0.0.1	127.0.0.1	GSM RLC/MAC	83	GPRS DL:GPRS DL:PACKET_UPLINK_ASSIGNMENT	Downlink
4396	30.184270412	127.0.0.1	127.0.0.1	GSM RLC/MAC	83	GPRS DL:PACKET_CONTROL_ACKNOWLEDGEMENT	Uplink
4398	30.18904774	127.0.0.1	127.0.0.1	GSM RLC/MAC	83	GPRS DL:GPRS DL:PACKET_DOWNLINK_DUMMY_CONTROL_BLOCK	Downlink
4412	30.189157683	127.0.0.1	127.0.0.1	GSM RLC/MAC	83	GPRS DL:GPRS DL:PACKET_DOWNLINK_DUMMY_CONTROL_BLOCK	Downlink
4432	30.203884017	127.0.0.1	127.0.0.1	GSM RLC/MAC	83	GPRS DL:GPRS DL:PACKET_DOWNLINK_DUMMY_CONTROL_BLOCK	Downlink
4562	30.267342891	127.0.0.1	127.0.0.1	GSM RLC/MAC	83	GPRS DL:GPRS DL:PACKET_DOWNLINK_DUMMY_CONTROL_BLOCK	Downlink
4564	30.267498362	127.0.0.1	127.0.0.1	GSM RLC/MAC	83	GPRS DL:GPRS DL:PACKET_DOWNLINK_DUMMY_CONTROL_BLOCK	Downlink
4578	30.28602489	127.0.0.1	127.0.0.1	GSM RLC/MAC	83	GPRS DL:GPRS DL:PACKET_DOWNLINK_DUMMY_CONTROL_BLOCK	Downlink
4580	30.28687304	127.0.0.1	127.0.0.1	GSM RLC/MAC	83	GPRS DL:GPRS DL:PACKET_DOWNLINK_DUMMY_CONTROL_BLOCK	Downlink
4586	30.290237639	127.0.0.1	127.0.0.1	GSM RLC/MAC	83	GPRS DL:GPRS DL:PACKET_DOWNLINK_DUMMY_CONTROL_BLOCK	Downlink
4600	30.290367348	127.0.0.1	127.0.0.1	GSM RLC/MAC	83	GPRS DL:GPRS DL:PACKET_DOWNLINK_DUMMY_CONTROL_BLOCK	Downlink
4618	30.305101426	127.0.0.1	127.0.0.1	GSM RLC/MAC	83	GPRS DL:GPRS DL:PACKET_DOWNLINK_DUMMY_CONTROL_BLOCK	Downlink
4632	30.305338021	127.0.0.1	127.0.0.1	GSM RLC/MAC	83	GPRS DL:GPRS DL:PACKET_DOWNLINK_DUMMY_CONTROL_BLOCK	Downlink
4634	30.309395305	127.0.0.1	127.0.0.1	GSM RLC/MAC	83	GPRS DL:GPRS DL:PACKET_DOWNLINK_DUMMY_CONTROL_BLOCK	Downlink
4648	30.305533898	127.0.0.1	127.0.0.1	GSM RLC/MAC	83	GPRS DL:GPRS DL:PACKET_DOWNLINK_DUMMY_CONTROL_BLOCK	Downlink
4657	30.324946664	127.0.0.1	127.0.0.1	GSM RLC/MAC	83	GPRS DL:GPRS DL:PACKET_DOWNLINK_DUMMY_CONTROL_BLOCK	Downlink
4671	30.325064119	127.0.0.1	127.0.0.1	GSM RLC/MAC	83	GPRS DL:GPRS DL:PACKET_DOWNLINK_DUMMY_CONTROL_BLOCK	Downlink
4689	30.327774284	127.0.0.1	127.0.0.1	GSM RLC/MAC	83	GPRS DL:GPRS DL:PACKET_DOWNLINK_DUMMY_CONTROL_BLOCK	Downlink
4691	30.32786353	127.0.0.1	127.0.0.1	GSM RLC/MAC	83	GPRS DL:GPRS DL:PACKET_DOWNLINK_DUMMY_CONTROL_BLOCK	Downlink
4698	30.34290743	127.0.0.1	127.0.0.1	GSM RLC/MAC	83	GPRS DL:GPRS DL:PACKET_DOWNLINK_DUMMY_CONTROL_BLOCK	Downlink
4700	30.342992135	127.0.0.1	127.0.0.1	GSM RLC/MAC	83	GPRS DL:GPRS DL:PACKET_DOWNLINK_DUMMY_CONTROL_BLOCK	Downlink
4706	30.350285725	127.0.0.1	127.0.0.1	GSM RLC/MAC	83	GPRS DL:GPRS DL:PACKET_DOWNLINK_DUMMY_CONTROL_BLOCK	Downlink
4766	30.365567293	127.0.0.1	127.0.0.1	GSM RLC/MAC	83	GPRS DL:GPRS DL:PACKET_DOWNLINK_DUMMY_CONTROL_BLOCK	Downlink
4768	30.365652656	127.0.0.1	127.0.0.1	GSM RLC/MAC	83	GPRS DL:GPRS DL:PACKET_DOWNLINK_DUMMY_CONTROL_BLOCK	Downlink
4777	30.3840448553	127.0.0.1	127.0.0.1	GSM RLC/MAC	83	GPRS DL:GPRS DL:PACKET_DOWNLINK_DUMMY_CONTROL_BLOCK	Downlink
4795	30.3883364896	127.0.0.1	127.0.0.1	GSM RLC/MAC	83	GPRS DL:GPRS DL:PACKET_DOWNLINK_DUMMY_CONTROL_BLOCK	Downlink
4797	30.388485345	127.0.0.1	127.0.0.1	GSM RLC/MAC	83	GPRS DL:GPRS DL:PACKET_DOWNLINK_DUMMY_CONTROL_BLOCK	Downlink
4810	30.403359352	127.0.0.1	127.0.0.1	GSM RLC/MAC	83	GPRS DL:GPRS DL:PACKET_DOWNLINK_DUMMY_CONTROL_BLOCK	Downlink
4812	30.4043462305	127.0.0.1	127.0.0.1	GSM RLC/MAC	83	GPRS DL:GPRS DL:PACKET_DOWNLINK_DUMMY_CONTROL_BLOCK	Downlink
4820	30.407696974	127.0.0.1	127.0.0.1	GSM RLC/MAC	83	GPRS DL:GPRS DL:PACKET_DOWNLINK_DUMMY_CONTROL_BLOCK	Downlink
4834	30.407837333	127.0.0.1	127.0.0.1	GSM RLC/MAC	83	GPRS DL:GPRS DL:PACKET_DOWNLINK_DUMMY_CONTROL_BLOCK	Downlink

### 4.3 Pruebas de Acceso y Capacidad de Red.

Los objetivos más importantes de la red propuesta son: proveer acceso a servicio de datos y garantizar la conectividad de varios usuarios compartiendo los recursos comunes de la interfaz inalámbrica.

OpenBTS es una plataforma que inicialmente fue diseñada para emular el funcionamiento de una red GSM, y posteriormente se implementó el servicio de datos GPRS. Debido a ello, los canales físicos se reparten en canales físicos para servicios de voz y sms, y canales físicos para servicios de datos GPRS.

De acuerdo con [43], la red GPRS propuesta permite configurar cuántos de los canales físicos de la trama TDMA son asignados como canales PDCH. Esta configuración se realiza con el parámetro `GPRS.Channels.Min.C0` como se describe en la sección 3.2.6.

Los canales PDCH especificados en el parámetro `GPRS.Channels.Min.C0` pueden ser agrupados en canales consecutivos y ser asignados a un usuario que necesite esa cantidad de canales PDCH para satisfacer los requerimientos de capacidad de canal para transmitir o recibir cierto tráfico. Esta configuración se denomina *multislot allocation*.

OpenBTS sigue la especificación de la sección 6.4.2 de [15] para crear las diferentes configuraciones de *multislot allocation*. El máximo número de canales PDCH consecutivos que se pueden agrupar y asignar a un MS en *uplink* y *downlink* se especifican con los parámetros `GPRS.Multislot.Max.Uplink` y `GPRS.Multislot.Max.Downlink`, respectivamente.

Un razonamiento lógico indicaría que a mayor número de canales PDCH asignados al servicio GPRS, mayor sería la capacidad de la red; sin embargo, en el anexo A.4.3 de [43], se explica que la capacidad de la red depende también de la configuración del *multislot allocation*. Debido a ello, con el objetivo de obtener cuál es la configuración más óptima para el multi-acceso de varios usuarios GPRS, es necesario experimentar el envío de paquetes IP con las diferentes configuraciones de la Tabla 4.4.

TABLA N° 4.4: Valores de la Potencia de Transmisión del USRP B210 en diferentes valores del parámetro `power` de OpenBTS, y diferentes canales ARFCN.

Configuración	<code>GPRS.Channels.Min.C0</code>	<code>GPRS.Multislot.Max.Uplink</code>	<code>GPRS.Multislot.Max.Downlink</code>
1	7	1	1
2	7	2	2
3	6	1	1
4	6	2	2

## Metodología del experimento.

El esquema de pruebas consiste en escribir un código Arduino (Anexo B) que permita que el chip SIM800L se active y, mediante la secuencia de pasos descritos en la sección 3.3, consiga establecer una conexión GPRS con el prototipo de red, y finalmente enviar datos hacia una red externa en forma de *pings*.

El código del Anexo B contiene una lógica que permite contabilizar los *pings* que obtuvieron respuesta por parte de la IP destino, y los *pings* que no. Esta prueba permite cuantificar el porcentaje de éxito de un MS cuando precisa acceder y hacer uso de la capacidad del prototipo de red en el servicio GPRS.

El código está programado para que un MS en un determinado instante envíe un *ping* a una IP destino y al terminar la transmisión espera la respuesta de la IP externa durante un tiempo aleatorio entre 10 y 15 segundos. Al cabo de dicho tiempo aleatorio e independientemente de si se obtuvo respuesta o no al *ping* anterior, el MS procede a enviar un nuevo *ping* tal como se muestra en la Figura 4.5.

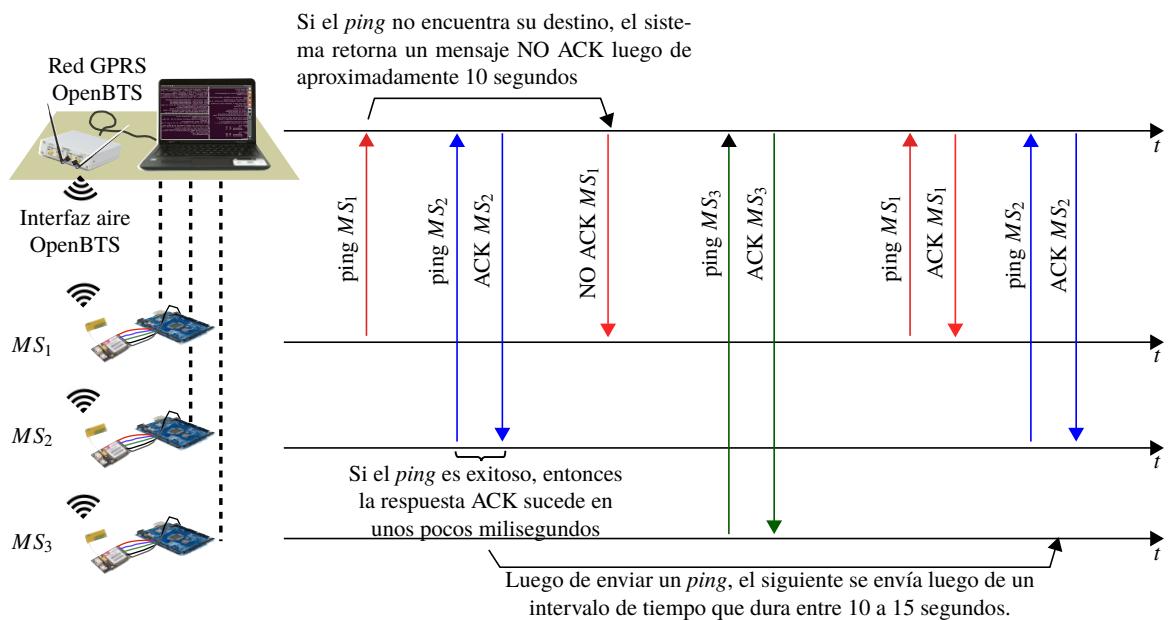


Fig. 4.5: Esquema de pruebas de acceso y capacidad de red con 1 MS, 2 MS, y 3 MS enviando *pings* a través de la interfaz aire de OpenBTS.

El escenario típico consiste en que un MS arbitrario envía *pings* de 1024 Bytes hacia la dirección IP 10.0.113.5. Dicha dirección IP corresponde a la interfaz virtual Ethernet de la misma máquina virtual Ubuntu que contiene a OpenBTS como se muestra en la Figura 4.6. En el sentido *uplink*, luego de que los paquetes IP llegan al SGSN/GGSN, estos son reencaminados hacia la *Wildcard* de la máquina virtual para ser enviados hacia la red IP

externa. Sin embargo, en este experimento la IP destino corresponde a la wildcard de la máquina virtual.

Dado que el SGSN/GGSN y la interfaz Eth son procesos propios de la máquina virtual, es prácticamente imposible que el paquete IP se pierda en el segmento SGSN/GGSN - Wildcard, con lo cual, el único segmento de red que se estaría verificando sería el acceso inalámbrico entre el MS y el USRP B210. Si ocurriese algún *ping* errado, el problema con certeza ocurre en el acceso inalámbrico.

De este modo, es posible contabilizar la probabilidad de éxito en el acceso a la red por parte de 01 MS arbitrario calculando el porcentaje de *pings* exitosos durante el experimento. Esta prueba se realiza con diferentes combinaciones valores de GPRS.Channels.Min.C0, GPRS.Multislot.Max.Uplink y GPRS.Multislot.Max.Downlink, y en escenarios de 01 MS, 02 MS, y 03 MS conectados en simultáneo al prototipo de red para verificar la eficiencia en la asignación de recursos por parte de OpenBTS.

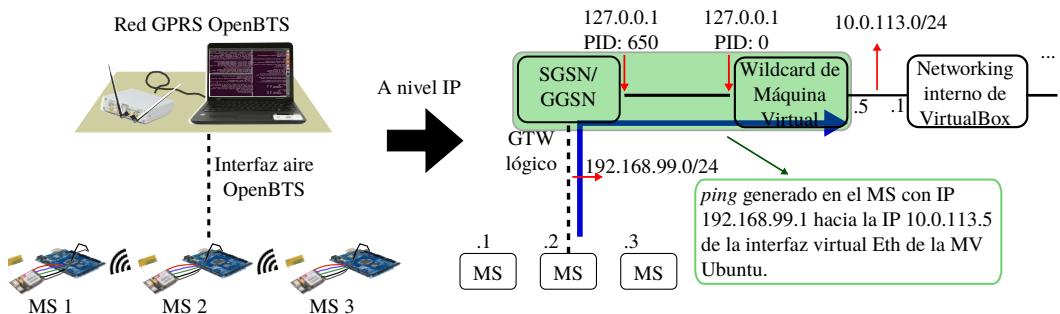


Fig. 4.6: Diagrama de bloques para explicar el *networking* que recorren los paquetes IP hasta internet.

La duración de esta prueba es de varias horas y es importante acotar que en la medida de lo posible, cada configuración de parámetros se experimenta hasta 3 veces para obtener cierto grado de correlación de la data obtenida.

Además, tal como se muestra en la Figura 4.7, se verifican los niveles de potencia de recepción y transmisión en cada elemento del radioenlace (MS y BS) para determinar las pérdidas existentes en los aproximadamente 2m de distancia que separan ambos elementos.

Las ecuaciones 4.1, 4.2 describen las pérdidas del canal tanto en *uplink* y *downlink*, respectivamente. La ecuación 4.3 es resultado de una aproximación entre las pérdidas de ambos caminos. Dada la configuración de frecuencias del canal ARFCN, el canal *uplink* se transmite en una frecuencia que es 50MHz menor que el canal *downlink*, sin embargo las pérdidas en ambos canales es aproximadamente igual.

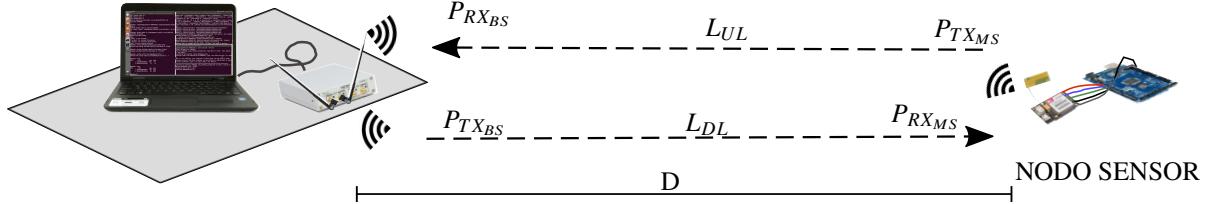


Fig. 4.7: Potencias de transmisión y recepción de un radioenlace entre el MS y el BS .

$$P_{TX_{MS}} - P_{RX_{BS}} = L_{UL} \quad (4.1)$$

$$P_{TX_{BS}} - P_{RX_{MS}} = L_{DL} \quad (4.2)$$

$$L_{UL} \approx L_{DL} \quad (4.3)$$

En el MS, los valores de la potencia de transmisión ( $P_{TX_{MS}}$ ) y recepción ( $P_{RX_{MS}}$ ) se pueden obtener con el comando AT+CENG? descrito en [28] mientras que en el BS, la potencia de recepción  $P_{RX_{BS}}$  se obtiene consultando el estado de la conexión GPRS mediante el comando gprs list tal como se describió en la sección 3.2.15.

Para el caso del parámetro  $P_{TX_{BS}}$ , las pruebas de la sección 4.1 determinan que la potencia de transmisión del USRP B210 ( $P_{TX_{USRP}}$ ) es de aproximadamente 0 dBm cuando el parámetro power es 5 en la configuración de OpenBTS. A este valor se le debe añadir 1dB que corresponde a la ganancia de la antena omnidireccional con lo cual se deduce la ecuación 4.4:

$$P_{TX_{BS}} = P_{TX_{USRP}|power=5} + 1dB \quad (4.4)$$

## Resultados.

La Tabla 4.5 muestra los parámetros medidos en 03 diferentes escenarios: 01 MS conectado, 02 MS conectados en simultáneo, y 03 MS conectados en simultáneo. En cada escenario, las configuraciones descritas en la Tabla 4.4 se repiten tres veces.

Del experimento 1 al 12 corresponden a un sólo MS conectado al prototipo de red. Del 13 al 28 corresponden a dos MS conectados en simultáneo y del 29 al 44 corresponden a tres MS conectados. Los experimentos 25 al 28 y 41 al 44 son para las configuraciones 1 y 2, de la Tabla 4.4, pero con un tiempo de duración más extendido.

**TABLA N° 4.5:** Resultados de pruebas de acceso y capacidad de red. Las columnas  $MS_1$ ,  $MS_2$ ,  $MS_3$  muestran los porcentajes de éxito de los pings enviados por cada MS a lo largo de toda la prueba.

Exp.	Durac.	GPRS.		GPRS.Mutislot.Max.		$MS_i$		$MS_{\Sigma}$		$MS_j$		$MS_k$		Prob. de Éxito (%)
		Channels.	Min.C0	DL	UL	$P_{TX_{MS}}$ (dBm)	$P_{RX_{MS}}$ (dBm)	$P_{TX_{KS}}$ (dBm)	$P_{RX_{KS}}$ (dBm)	$P_{TX_{MS}}$ (dBm)	$P_{RX_{MS}}$ (dBm)	$P_{TX_{KS}}$ (dBm)	$P_{RX_{KS}}$ (dBm)	
1	05h42m	7	1	1	0	-54	5	-70	-	-	-	-	-	98.5
2	05h50m	7	1	1	0	-53	5	-66	-	-	-	-	-	96.1
3	05h25m	7	1	1	0	-54	5	-68	-	-	-	-	-	97.5
4	06h30m	7	2	2	0	-52	5	-65	-	-	-	-	-	98.9
5	05h47m	7	2	2	0	-54	5	-70	-	-	-	-	-	98.1
6	05h55m	7	2	2	0	-52	5	-64	-	-	-	-	-	99.1
7	05h30m	6	1	1	0	-54	5	-62	-	-	-	-	-	96.1
8	05h38m	6	1	1	0	-51	5	-61	-	-	-	-	-	94.7
9	05h32m	6	1	1	0	-53	5	-62	-	-	-	-	-	95.8
10	05h40m	6	2	2	0	-50	5	-60	-	-	-	-	-	91.4
11	05h41m	6	2	2	0	-55	5	-65	-	-	-	-	-	89.7
12	06h02m	6	2	2	0	-53	5	-68	-	-	-	-	-	92.5
13	05h22m	7	1	1	0	-54	5	-66	0	-56	5	-60	-	92.5
14	05h45m	7	1	1	0	-50	5	-70	0	-58	5	-64	-	89.1
15	05h45m	7	1	1	0	-51	5	-68	0	-54	5	-66	-	93.1
16	05h52m	7	2	2	0	-52	5	-64	0	-55	5	-70	-	98.9
17	05h38m	7	2	2	0	-51	5	-65	0	-58	5	-64	-	97.9
18	06h22m	7	2	2	0	-52	5	-64	0	-57	5	-65	-	98.5
19	05h47m	6	1	1	0	-53	5	-65	0	-52	5	-62	-	96.2
20	05h30m	6	1	1	0	-50	5	-66	0	-58	5	-64	-	97.1
21	05h32m	6	1	1	0	-50	5	-65	0	-58	5	-65	-	94.1
22	05h45m	6	2	2	0	-52	5	-62	0	-56	5	-65	-	98.5
23	05h30m	6	2	2	0	-52	5	-70	0	-54	5	-68	-	94.7
24	05h38m	6	2	2	0	-51	5	-62	0	-55	5	-68	-	95.3
25	12h05m	7	1	1	0	-52	5	-64	0	-58	5	-70	-	92.4
26	12h22m	7	1	1	0	-54	5	-60	0	-60	5	-62	-	96.1
27	13h34m	7	2	2	0	-55	5	-68	0	-62	5	-68	-	98.8
28	12h36m	7	2	2	0	-55	5	-70	0	-58	5	-68	-	97.2
29	05h32m	7	1	1	0	-53	5	-68	0	-56	5	-66	-	96.1
30	05h45m	7	1	1	0	-50	5	-66	0	-56	5	-64	-	94.7
31	05h44m	7	1	1	0	-51	5	-68	0	-54	5	-62	-	93.7
32	05h52m	7	2	2	0	-53	5	-70	0	-50	5	-62	-	94.7
33	05h58m	7	2	2	0	-52	5	-67	0	-56	5	-64	-	99.2
34	05h25m	7	2	2	0	-52	5	-68	0	-52	5	-62	-	99.3
35	05h33m	6	1	1	0	-55	5	-62	0	-58	5	-60	-	98.9
36	05h34m	6	1	1	0	-54	5	-64	0	-58	5	-60	-	92.3
37	05h52m	6	1	1	0	-51	5	-64	0	-56	5	-62	-	90.4
38	05h27m	6	2	2	0	-56	5	-65	0	-58	5	-66	-	94.4
39	05h39m	6	2	2	0	-52	5	-65	0	-56	5	-68	-	99.2
40	06h41m	6	2	2	0	-54	5	-65	0	-54	5	-57	-	98.1
41	16h30m	7	1	1	0	-52	5	-63	0	-55	5	-70	-	97.2
42	22h30m	7	1	1	0	-50	5	-62	0	-57	5	-68	-	96.4
43	12h30m	7	2	2	0	-50	5	-60	0	-52	5	-66	-	98.2
44	15h30m	7	2	2	0	-56	5	-62	0	-58	5	-66	-	97.5

Para el caso de un MS conectado (experimentos 1 al 12), la Figura 4.8 muestra la comparación de los porcentajes de éxito obtenidos en tres pruebas diferentes para cada configuración de la Tabla 4.4. En ella se puede apreciar que la configuración 2 es la que obtiene mejores números seguido de la configuración 1, 3 y finalmente 4.

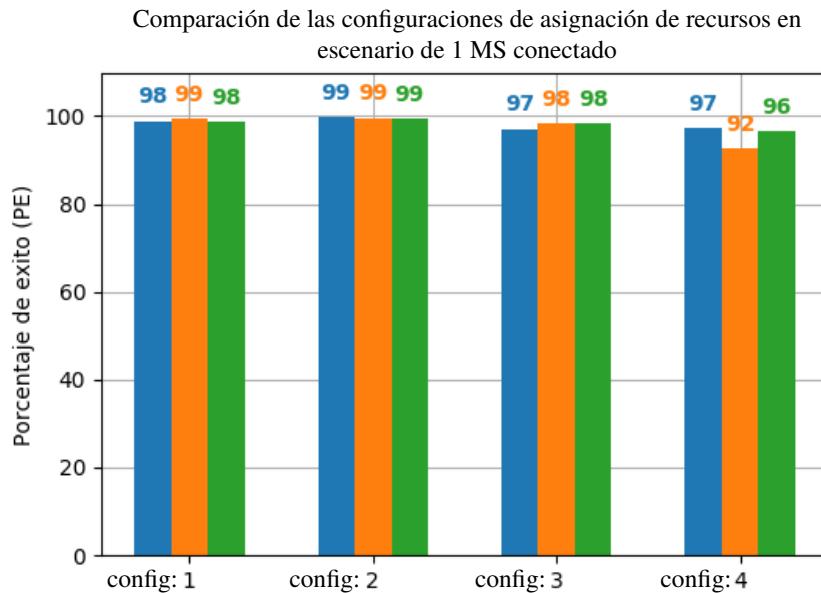


Fig. 4.8: Comparación de los porcentajes de éxito de diferentes configuraciones (Experimentos del 1 al 12 de la Tabla 4.5) en el envío de *pings* en el escenario en que un sólo usuario está conectado a la red.

Las Figuras 4.9, 4.10, 4.11, 4.12 muestran la comparación de los porcentajes de éxito para cada configuración de la Tabla 4.4 en los tres escenarios: 01 usuario conectado, 02 usuarios conectados, y 03 usuarios conectados. Para facilitar la visualización de los gráficos, es importante recordar que cada configuración de la Tabla 4.4 se experimenta tres veces, debido a ello es que se muestran tres barras por cada MS.

En todas las Figuras 4.9, 4.10, 4.11, 4.12 se nota una disminución en el porcentaje de éxito de envío de *pings* conforme el número de MS aumenta. La Figura 4.10, correspondiente a la Configuración 2, es la única en la que los porcentajes de éxito no disminuyen considerablemente. Esto significa que en dicha configuración, los MS consiguen acceder a la red con mayor éxito que en las otras configuraciones.

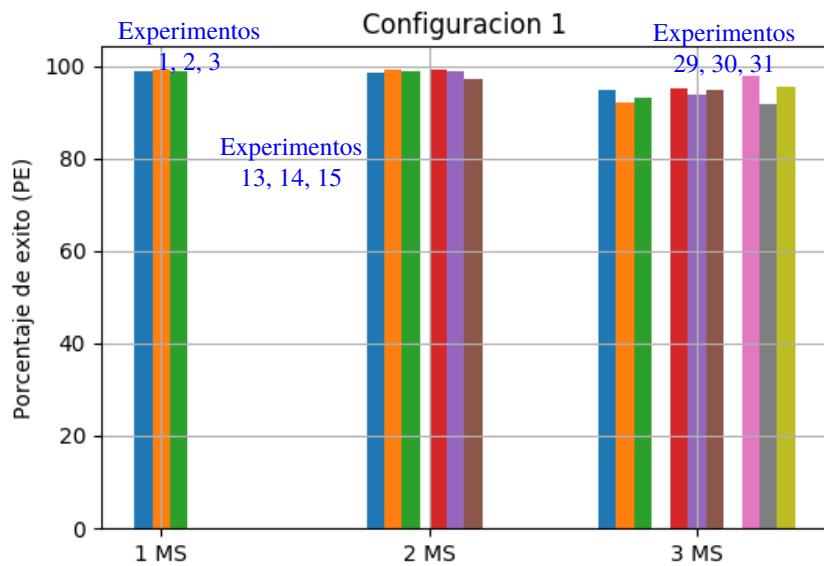


Fig. 4.9: Comparación de las probabilidades de éxito usando la configuración 1 en escenarios de 1 MS, 2 MS, y 3 MS conectados en simultáneo.

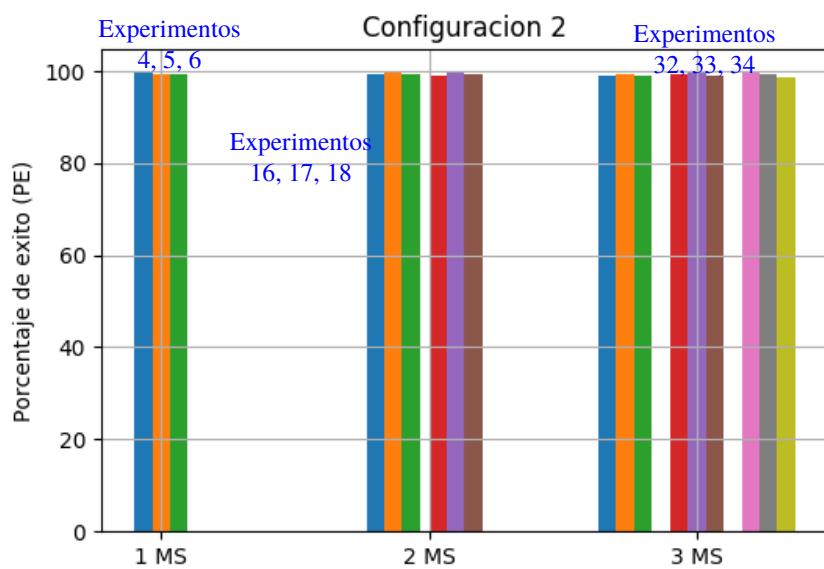


Fig. 4.10: Comparación de las probabilidades de éxito usando la configuración 2 en escenarios de 1 MS, 2 MS, y 3 MS conectados en simultáneo.

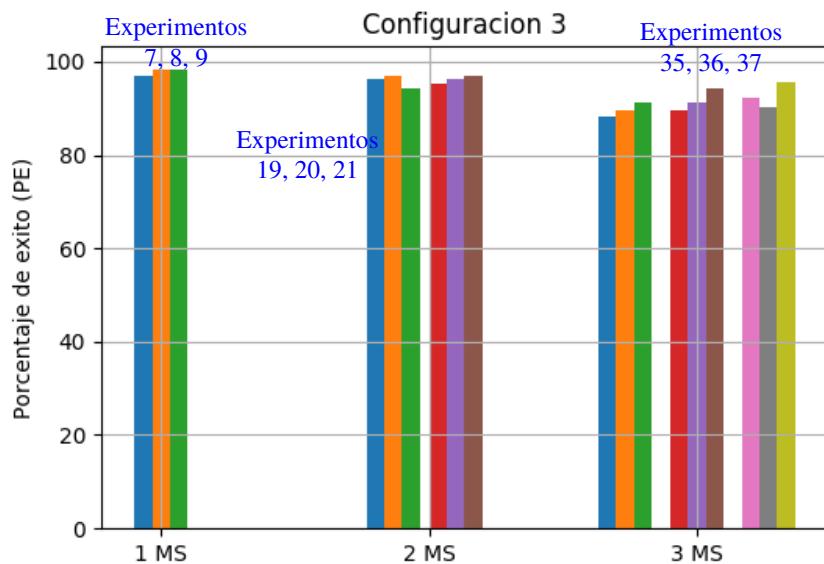


Fig. 4.11: Comparación de las probabilidades de éxito usando la configuración 3 en escenarios de 1 MS, 2 MS, y 3 MS conectados en simultáneo.

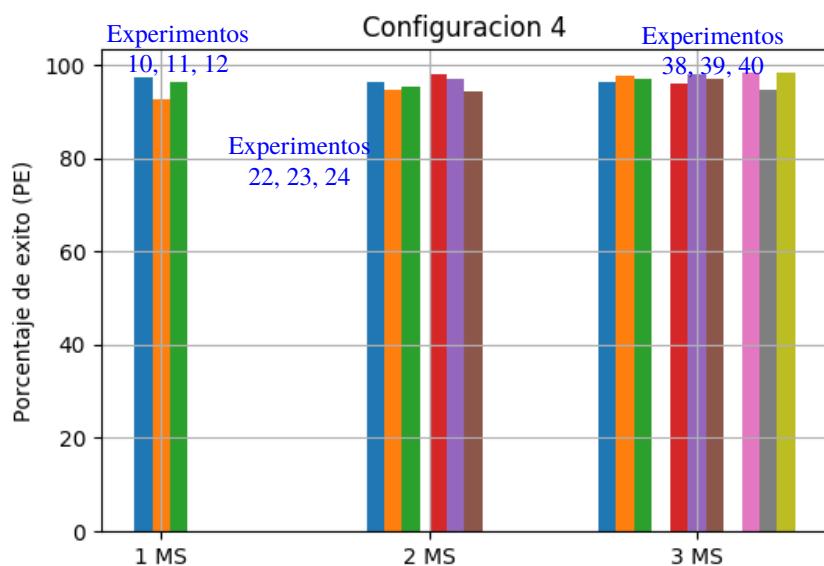


Fig. 4.12: Comparación de las probabilidades de éxito usando la configuración 4 en escenarios de 1 MS, 2 MS, y 3 MS conectados en simultáneo.

#### 4.4 Implementación de una aplicación IoT usando la red GPRS propuesta.

La Figura 4.13 muestra el esquema de pruebas para esta sección. El SIM800L, o dispositivo final, establece una conexión con la red GPRS propuesta y envía datos a través de algún protocolo de capa aplicación hacia un servidor en la nube, el cual recolecta estos datos y permite realizar análisis sobre ellos.

El SIM800L sigue las instrucciones del código provisto en el Anexo 3 para primero establecer una conexión con la red GPRS y, sobre ella, establecer una conexión TCP/IP con el servidor de aplicaciones analíticas *The ThingSpeak* [44]. *The ThingSpeak* es una plataforma de servidores online que proveen aplicaciones de recolección de datos para su post-análisis en la misma nube. Un usuario puede obtener una cuenta gratuita en *The ThingSpeak* y seguir los pasos de [45] para enviar o recolectar datos hacia o desde un dispositivo o aplicación.

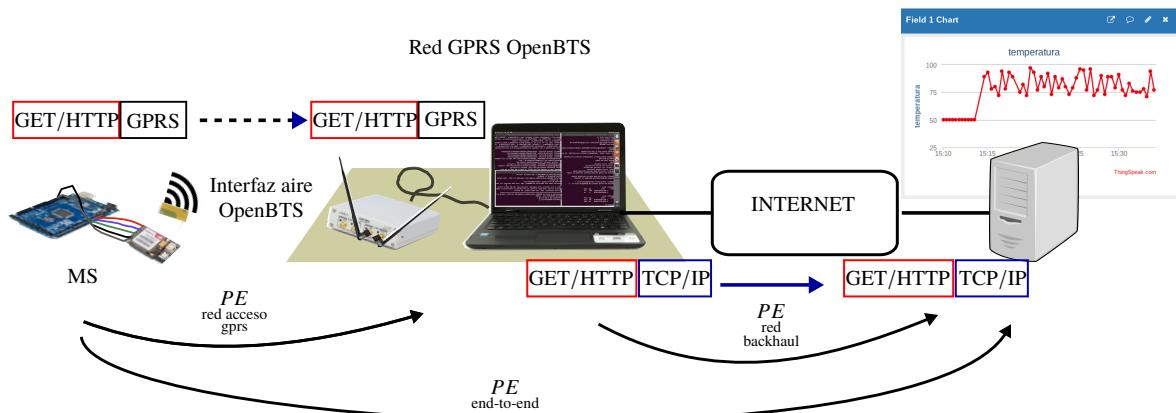


Fig. 4.13: Esquema de pruebas de envío de paquetes HTTP hasta un servidor en internet para su visualización en la aplicación web del servidor.

En general, *The ThingSpeak* provee canales de recolección en los cuales se puede almacenar datos con su respectiva marca de tiempo. Los datos pueden ser números enteros, números reales, cadenas de *strings*, entre otros. *The ThingSpeak* provee una interfaz REST API de la cual es posible invocar solicitudes HTTP tales como GET, POST, PUT, y DELETE para enviar o recibir data[46].

El SIM800L se conecta a la red GPRS propuesta y establece una conexión TCP con el servidor de *The ThingSpeak* en el puerto 80 y utiliza solicitudes HTTP del tipo GET[28] para escribir datos en el canal de recolección. El detalle de la configuración de los parámetros `api_key` y `field` se encuentra en [47].

La Figura 4.14 muestra una tabla online típica que la plataforma *The ThingSpeak* provee, en la cual se muestra vía online la data que está siendo enviada por el SIM800L. En este caso

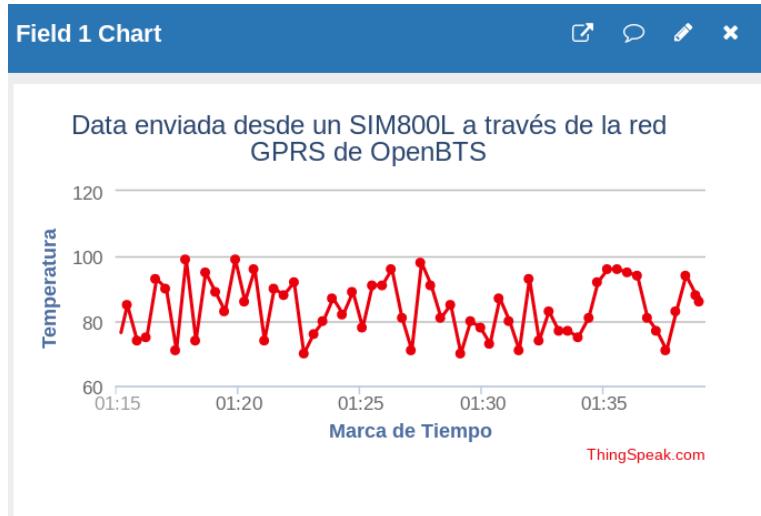


Fig. 4.14: Gráfico de la data acumulada en el servidor *The ThingSpeak*. El SIM800L alimenta esta data mediante el envío de valores de un determinado parámetro encapsulados en paquetes HTTP tipo GET.

el canal ha sido personalizado para un parámetro *Temperatura*. Cada punto de medición es un valor del parámetro *temperatura* enviado dentro de solicitudes GET hacia la respectiva función del REST API del canal en *The ThingSpeak*.

El servidor *The ThingSpeak* permite exportar los datos recolectados en archivos .csv, con lo cual la idea del experimento es utilizar el *script* del Anexo 3 para que un SIM800L envíe paquetes HTTP con valores aleatorios del parámetro *Temperatura* por un determinado período de tiempo y, posteriormente, exportar la data para obtener cuántos paquetes HTTP fueron recibidos en el servidor.

El *script* en el SIM800L permite conocer cuántos paquetes HTTP fueron enviados desde el SIM800L y con la data exportada de *The ThingSpeak* se puede conocer de manera directa cuántos paquetes llegaron al canal de recolección y, de manera indirecta, cuántos paquetes se perdieron en alguno de los segmentos de red, ya sea acceso o *backhaul*.

Con lo anterior, es posible hallar el porcentaje de éxito (PE) en el servicio *end-to-end* PE, desde el MS hasta el servidor *The ThingSpeak*. Además, el *script* implementa una función para contabilizar los paquetes HTTP que se pierden en la interfaz aire con la red GPRS, y con ello se obtiene el porcentaje de éxito en la red de acceso, PE<sub>red acceso gprs</sub>.

De la Figura 4.13 se obtiene la siguiente relación:

$$\begin{pmatrix} PE_{\text{red acceso}} \\ PE_{\text{gprs}} \end{pmatrix} \cdot \begin{pmatrix} PE_{\text{red IP}} \\ PE_{\text{backhaul}} \end{pmatrix} = \begin{pmatrix} PE_{\text{end-to-end}} \\ PE_{\text{gprs}} \end{pmatrix} \quad (4.5)$$

Dado que se conocen los valores de la porcentaje de éxito *end-to-end* y del acceso (columnas F y G de la Tabla 4.6), la ecuación 4.5 se modifica para dar lugar a la ecuación 4.6 con la que se obtiene la probabilidad de éxito de la red *backhaul*,  $PE_{\text{red IP backhaul}}$ :

$$\left( \frac{PE_{\text{red IP backhaul}}}{PE_{\text{end-to-end}}} \right) = \left( \frac{PE_{\text{red acceso gprs}}}{PE_{\text{end-to-end}}} \right) \quad (4.6)$$

La Tabla 4.6 presenta las estadísticas de 10 pruebas realizadas utilizando el esquema de la Figura 4.13. De izquierda a derecha, las columnas C y D presentan la cantidad de paquetes originados en el MS y recibidos en el servidor de *The ThingSpeak* (TTS), respectivamente. La diferencia entre ambas columnas corresponde al número de paquetes que se perdieron en la red de acceso o en la red *backhaul*. La columna E presenta la cantidad de paquetes perdidos en la red de acceso y las columnas F, G, y H presentan los porcentaje de éxito en cada segmento parcial y total del esquema de la Figura 4.13.

TABLA N° 4.6: Resultados de prueba de envío de paquetes HTTP hacia los servidores de *The ThingSpeak* a través de la red GPRS de OpenBTS.

A	B	C	D	E	F	G	H
test	Duración (hh:mm)	# de paq. originados en el MS	# de paq. recibidos en TTS	# de paq. perdidos en acceso	PE (%) servicio total	PE (%) acceso gprs	PE (%) red backhaul
1	02:26	366	359	5	98.1	98.6	99.4
2	02:50	425	411	9	96.7	97.9	98.8
3	03:18	495	479	11	96.8	97.8	99.0
4	03:18	496	485	7	97.8	98.6	99.2
5	03:22	505	498	5	98.6	99.0	99.6
6	03:24	510	505	4	99.0	99.2	99.8
7	03:24	510	502	6	98.4	98.8	99.6
8	03:28	520	510	4	98.1	99.2	98.8
9	03:34	535	521	9	97.4	98.3	99.0
10	04:13	633	625	4	98.7	99.4	99.4

A partir de los datos de la Tabla 4.6 se construyen las curvas presentadas en la Figura 4.15. Las tres curvas corresponden a los valores de las columnas 5, 6, y 7. El eje y es el porcentaje de éxito y el eje x indica a que test corresponde.

Los tests se colocan en valor ascendente respecto del número de paquetes enviados. Según la Tabla 4.6, el test 10 contiene 633 paquetes HTTP originados en el MS mientras que el test 1 contiene 366 paquetes.

La curva de color azul corresponde al porcentaje de éxito de que el paquete HTTP llegue hasta el servidor *The ThingSpeak*. Esta curva es la que presenta menores valores debido a

que justamente es el producto de los porcentajes de éxito de las otras dos curvas, los cuales son menores que 1.

La curva de color rojo corresponde al porcentaje de éxito de que el paquete acceda correctamente a la red GPRS. La calidad en la interfaz de radio es el principal factor que se visualiza en este parámetro. La curva de color amarillo corresponde a la probabilidad de éxito de que el paquete deje la red GPRS y llegue con éxito al servidor *The ThingSpeak*.

En una red inalámbrica de datos, por lo general el acceso es más problemático que el *backhaul*. Esto se visualiza en la gráfica dado que la curva de la probabilidad de éxito en la red *backhaul* se sitúa por encima de la curva de la probabilidad de éxito en la red GPRS, atenuándose la diferencia conforme aumenta el número de paquetes HTTP enviados.

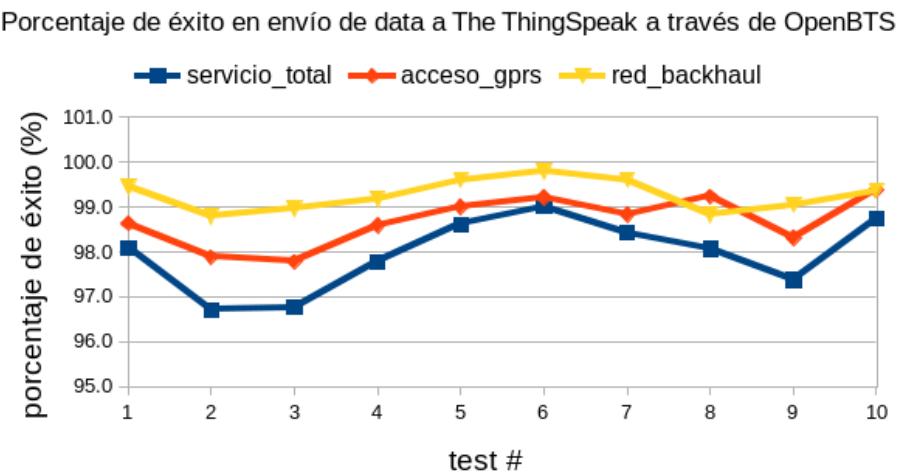


Fig. 4.15: Gráfico comparativo de los porcentajes de éxito en la red de acceso GPRS y la red backhaul hasta el servidor de *The ThingSpeak*.

## CONCLUSIONES Y RECOMENDACIONES

### Conclusiones

Este capítulo comprende las conclusiones obtenidas en esta tesis:

1. Se consigue implementar un prototipo de red GPRS. El prototipo de red emula tanto las entidades que conforman la red de acceso como las entidades que conforman el *core* de red, de una red GSM/GPRS.
2. El prototipo de red GPRS se consigue implementar sobre un computador genérico conectado a un periférico de radio basado en la tecnología de radio definido por software. El computador genérico tiene requerimientos mínimos los cuales podrían considerarse de bajo costo comparado a las actuales soluciones de hoy en día (sección 3.1). El periférico de radio también se puede considerar de bajo costo teniendo en cuenta el precio de equipos similares.
3. Se consigue realizar la conexión GPRS entre un MS (SIM800L) y el prototipo de red GPRS. Sobre dicha conexión GPRS, se realiza el envío de paquetes de datos en forma de *pings* hacia una IP externa. Esto demuestra la prueba de concepto sobre el acceso a la red, la conectividad con redes de datos externas y el intercambio de paquetes con dichas redes externas.
4. Como aporte de conocimiento adquirido a lo largo del desarrollo de esta tesis, se puede indicar que la implementación del prototipo de red demandó desarrollar habilidades en diferentes tópicos técnicos entre los cuales se puede mencionar el uso sistemas operativos basados en Linux, el aprendizaje del funcionamiento de una red GPRS a nivel de cada capa del *stack* de protocolos, programación en C++, programación en Python, entre otros.

5. Se consigue entender el funcionamiento de OpenBTS, en especial, los procesos computacionales involucrados en el establecimiento de la conexión GPRS (sección 2.4). Además, se obtiene conocimiento sobre la interacción de OpenBTS con el periférico de radio USRP B210 (sección 2.5).
6. Las pruebas de la sección 4.1 permitieron obtener que la máxima potencia en el puerto Tx del USRP B210 es de 6.4dBm. Si a esto se le suma 2dBi de ganancia aproximada de una antena omnidireccional tipo *duck*, se puede estimar que el PIRE del prototipo de red implementado es de 8.4dBm.
7. De la Figura 4.3, se concluye que la potencia de transmisión del USRP B210 es aproximadamente igual en todos los ARFCN cuando el parámetro `power` está entre 0 y 15, es decir, cuando el amplificador de potencia del USRP B210 está trabajando con la máxima ganancia configurada. Cuando el parámetro `power` aumenta, o lo que es lo mismo, cuando la ganancia del USRP B210 disminuye, se nota una ligera divergencia entre las potencias de transmisión del USRP B210 diferentes ARFCNs.
8. De las pruebas realizadas en la sección 4.2, se comprueba que el MS (SIM800L) realiza el procedimiento de acceso aleatorio a la red, el prototipo de red GPRS le asigna recursos de radio, y ambos establecen los respectivos TBF para el envío de los radio-paquetes en *uplink* y *downlink*. El *stack* de protocolos de OpenBTS extrae las cabeceras de radio de los paquetes y los ensambla dentro de paquetes IP para su envío hacia las redes externas mediante el protocolo IP.
9. Las pruebas de la sección 4.3 comparan la probabilidad de éxito del acceso a la red con diferentes configuraciones de los parámetros relacionados con la asignación de canales PDCHs para cada dispositivo final. Se hayó que la configuración más óptima para soportar conexiones de varios usuarios a la vez corresponde a los valores:
 

```
GPRS.Channels.Min.C0 = 7
GPRS.Multislot.Max.Uplink = 2
GPRS.Multislot.Max.Downlink = 2
```
10. En las pruebas de la sección 4.4, se implementa una transferencia de datos IP desde el MS hacia un canal de recolección de data proporcionado por el servidor de *The*

*ThingSpeak*. Esta prueba implementa lo que sería un típica aplicación IoT en la cual el nodo sensor se conecta a la red mediante su dispositivo GPRS, establece una conexión de datos y transfiere la data recolectada hacia un servidor en la nube donde el usuario puede realizar tareas de procesamiento y análisis de datos.

### **Recomendaciones y Trabajos Futuros**

Como recomendaciones a realizarse a partir del trabajo presentado, o trabajos futuros para mejorar la solución propuesta tenemos:

1. Las pruebas sobre la capacidad del prototipo de red propuesto, expuestas en la 4.4, deben replicarse con un mayor número de usuarios conectados de modo que se representa un escenario de conexión típico para este tipo de aplicaciones.
2. Dado que el prototipo de red propuesto tiene un radio de cobertura limitado, es necesario el desarrollo de las etapas de amplificación de potencia, para la Tx, y la etapa de amplificación de bajo ruido (en el RX).
3. El prototipo de red propuesto precisa de un estudio sobre su desempeño en torno al procesamiento de los paquetes de datos bajo el punto de vista de la calidad de servicio (QoS). Debido a que OpenBTS es una distribución de software libre, es posible adicionar código al código ya existe para implementar nuevas funcionalidades sobre el QoS.
4. Un desarrollo interesante podría ser, la elaboración de una plataforma para visualización de eventos del prototipo de red propuesto, de preferencia en un entorno web de modo que un usuario pueda ingresar en remoto a la plataforma y visualizar, por ejemplo, los dispositivos conectados, estados de batería útil, etc.
5. Teniendo como base el prototipo de red propuesto, otro desarrollo interesante sería una plataforma para análisis de datos en un entorno de red local, es decir, que no necesite de una conexión a internet con alguna plataforma de post-análisis de datos en la nube. Dicha solución debe ser adecuadamente instalada en el mismo computador genérico que contiene al OpenBTS.

## BIBLIOGRAFÍA

- [1] M. Hung, «Leading the IoT», 2017.
- [2] IHS.com, «IoT platforms: enabling the Internet of Things», 2016.
- [3] Analog Devices, «RF Agile Transceiver AD9361», 2016.
- [4] N. Ryu, Y. Yun, S. Choi, R. C. Palat y J. H. Reed, «Smart antenna base station open architecture for SDR networks», *IEEE Wireless Communications*, volumen 13, número 3, páginas 58-69, junio de 2006.
- [5] A. Anand, V. Pejovic, E. Belding y D. L. Johnson, «VillageCell: cost effective cellular connectivity in rural areas», marzo de 2012.
- [6] J. Mpala y G. Van Stam, «Open BTS, a GSM experiment in rural Zambia», noviembre de 2012.
- [7] A. Azad, «Open BTS Implementation With Universal Software Radio Peripheral», abril de 2019.
- [8] T. Zhao, P. Yang, H. Pan, R. Deng, S. Zhou y Z. Niu, «Software Defined Radio Implementation of Signaling Splitting in Hyper-Cellular Network», *CoRR*, volumen abs/1312.0720, 2013. arXiv: 1312.0720.
- [9] D. A. Burgess, H. S. Samra y col., «The openbts project», *Report available at http://openbts.sourceforge.net, http://openBTS.org*, 2008.
- [10] Ettus Research. (2019). USRP Hardware Driver and USRP Manual, dirección: [ht tp : / / files . ettus . com / manual / page \\_ usrp \\_ b200 . html](http://files.ettus.com/manual/page_usrp_b200.html) (visitado 06-02-2019).
- [11] *Digital cellular telecommunications system (Phase 2+); General Packet Radio Service (GPRS) Service description; Stage 2*, 3GPP TS 03.60, V7.9.0, 3GPP - ETSI, 2002.
- [12] *Digital cellular telecommunications system (Phase 2+) (GSM); Universal Mobile Telecommunications System (UMTS); General Packet Radio Service (GPRS); Service description; Stage 1*, 3GPP TS 22.060, V15.0.0, 3GPP - ETSI, 2018.

- [13] *Digital cellular telecommunications system (Phase 2+) (GSM); Universal Mobile Telecommunications System (UMTS); LTE; Network architecture*, 3GPP TS 23.002, V15.0.0, 3GPP - ETSI, 2018.
- [14] *Digital cellular telecommunications system (Phase 2+); Modulation*, 3GPP TS 05.04, V8.4.0, 3GPP - ETSI, 2001.
- [15] *Digital cellular telecommunications system (Phase 2+); Multiplexing and Multiple Access on the Radio Path*, 3GPP TS 05.02, V8.11.0, 3GPP - ETSI, 2003.
- [16] *Digital cellular telecommunications system (Phase 2+); Radio Transmission and Reception*, 3GPP TS 05.05, V8.20.0, 3GPP - ETSI, 2005.
- [17] *Digital cellular telecommunications system (Phase 2+); Physical Layer on the Radio Path (General Description)*, 3GPP TS 05.01, V8.9.0, 3GPP - ETSI, 2004.
- [18] *Digital cellular telecommunications system (Phase 2+); Radio subsystem synchronization*, 3GPP TS 05.10, V8.12.0, 3GPP - ETSI, 2001.
- [19] *Digital cellular telecommunications system (Phase 2+); Radio subsystem link control*, 3GPP TS 05.08, V8.23.0, 3GPP - ETSI, 2005.
- [20] *Digital cellular telecommunications system (Phase 2+); Mobile radio interface layer 3 specification*, 3GPP TS 04.08, V7.21.0, 3GPP - ETSI, 2003.
- [21] *Digital cellular telecommunications system (Phase 2+); General Packet Radio Service (GPRS); Overall description of the GPRS radio interface; Stage 2*, 3GPP TS 03.64, V8.12.0, 3GPP - ETSI, 2004.
- [22] *Digital cellular telecommunications system (Phase 2+); General Packet Radio Service (GPRS); Mobile Station - Serving GPRS Support Node (MS-SGSN) Logical Link Control (LLC) layer specification*, 3GPP TS 04.64, V8.7.0, 3GPP - ETSI, 2001.
- [23] *Digital cellular telecommunications system (Phase 2+) (GSM); Universal Mobile Telecommunications System (UMTS); LTE; Mobile radio interface Layer 3 specification; Core network protocols; Stage 3*, 3GPP TS 24.008, V15.4.0, 3GPP - ETSI, 2018.
- [24] *Digital cellular telecommunications system (Phase 2+); General Packet Radio Service (GPRS); Mobile Station (MS) - Serving GPRS Support Node (SGSN); Subnetwork Dependent Convergence Protocol (SNDCP)*, 3GPP TS 04.65, V8.2.0, 3GPP - ETSI, 2001.
- [25] SIMCom, «SIM800 Hardware Design, V1.09», 2016.
- [26] Wikipedia. (2018). SIM808 GSM/GPRS/GPS Module, dirección: [https://www.itead.cc/wiki/SIM808\\_GSM/GPRS/GPS\\_Module](https://www.itead.cc/wiki/SIM808_GSM/GPRS/GPS_Module) (visitado 06-02-2019).

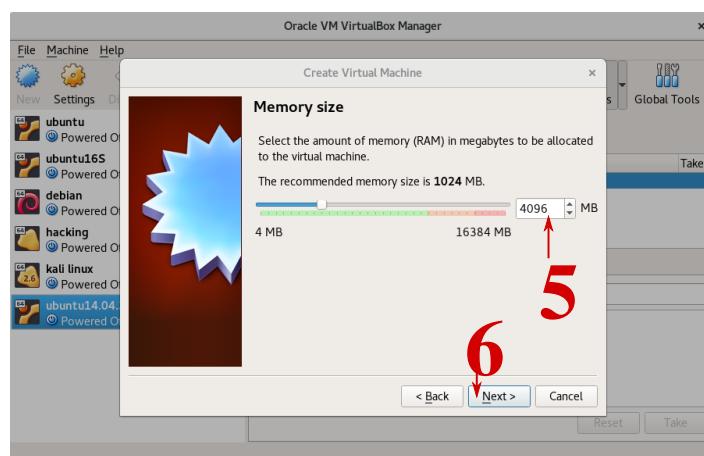
- [27] Nokia. (2009). AT Commands, dirección: <https://doc.qt.io/archives/qtextended4.4/atcommands.html> (visitado 05-05-2019).
- [28] SIMCom, «SIM800 Series AT Command, Manual V1.10», 2016.
- [29] Arduino. (2019). Download the Arduino IDE, dirección: <https://www.arduino.cc/en/Main/Software> (visitado 05-05-2019).
- [30] SIMCom, «SIM800 Series TCPIP Application Note V1.02», 2016.
- [31] M. Iedema, *Getting Started with OpenBTS: Build Open Source Mobile Networks.* "O'Reilly Media, Inc.", 2014.
- [32] Digium. (2019). Asterisk, dirección: <https://www.asterisk.org/> (visitado 05-05-2019).
- [33] Range Networks. (2019). Range Networks OpenBTS, dirección: <https://github.com/RangeNetworks/dev.git> (visitado 06-02-2019).
- [34] Ettus Research. (2019). USRP B210, dirección: <https://www.ettus.com/product/details/UB210-KIT> (visitado 06-02-2019).
- [35] Virtualbox. (2019). Download VirtualBox, dirección: <https://www.virtualbox.org/wiki/Downloads> (visitado 05-04-2019).
- [36] Berkeley Common Environment. (2019). Enabling Virtualization in your PC BIOS, dirección: <https://bce.berkeley.edu/enabling-virtualization-in-your-pc-bios.html> (visitado 06-02-2019).
- [37] GitHub. (2019). GitHub, Inc (US), dirección: <https://github.com/> (visitado 06-02-2019).
- [38] Code and Make. (2019). How to connect a USB device to a VirtualBox VM, dirección: <https://www.youtube.com/watch?v=saR-nVRV5rk> (visitado 06-02-2019).
- [39] Ettus Research. (2019). UHD, dirección: <https://kb.ettus.com/UHD> (visitado 06-02-2019).
- [40] N. Kotelnikova. (2019). OpenBTS scripts for systemd, dirección: [https://github.com/nadiia-kotelnikova/openbts\\_systemd\\_scripts](https://github.com/nadiia-kotelnikova/openbts_systemd_scripts) (visitado 06-02-2019).
- [41] Wireshark. (2019). Wireshark Home Page, dirección: <https://www.wireshark.org/> (visitado 05-05-2019).
- [42] Wireshark. (2019). Wireshark Display Filters, dirección: <https://wiki.wireshark.org/DisplayFilters> (visitado 05-05-2019).

- [43] Range Networks, *OpenBTS Application Suite: User Manual*. 2014.
- [44] The ThingSpeak. (2019). The ThingSpeak, dirección: <https://thingspeak.com> (visitado 05-05-2019).
- [45] MathWorks. (2019). Collect Data in a New Channel, dirección: <https://www.mathworks.com/help/thingspeak/collect-data-in-a-new-channel.html> (visitado 05-05-2019).
- [46] MathWorks. (2019). REST API, dirección: <https://www.mathworks.com/help/thingspeak/rest-api.html> (visitado 05-05-2019).
- [47] MathWorks. (2019). Write Data, dirección: <https://www.mathworks.com/help/thingspeak/writedata.html> (visitado 05-05-2019).

**ANEXO A**  
**INSTALACIÓN DE MÁQUINA VIRTUAL UBUNTU 16.04.**

## Configuración de Virtual Box.

1. Abrir el Virtual Box y presionar en el botón “New”.
2. Se abre la ventana para la creación de una nueva máquina virtual.
3. Colocar el nombre “Ubuntu 16.04” a la máquina virtual y seleccionar la opción **Linux** en “Type” y la opción **Ubuntu (64-bit)** en “Version”.
4. Click en “Next”.
5. Configurar la memoria RAM de la máquina virtual en 4096 MB. Lo mínimo que OpenBTS acepta es 2048 MB.
6. Click en “Next”.



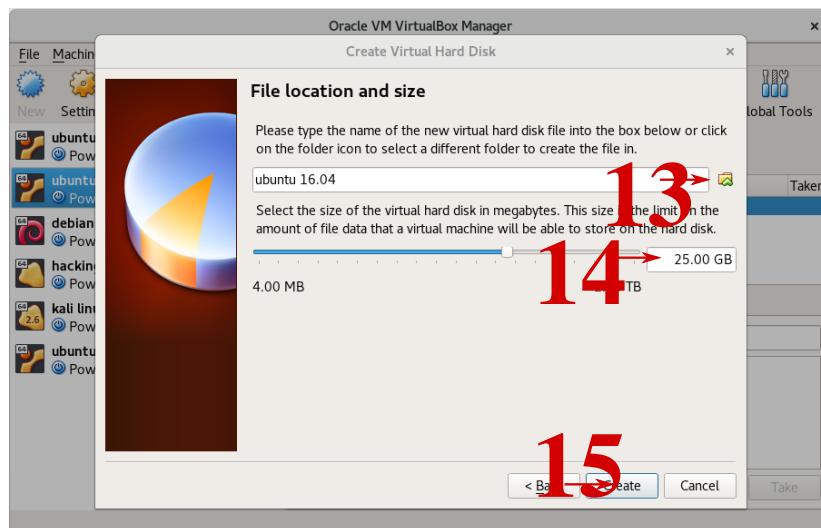
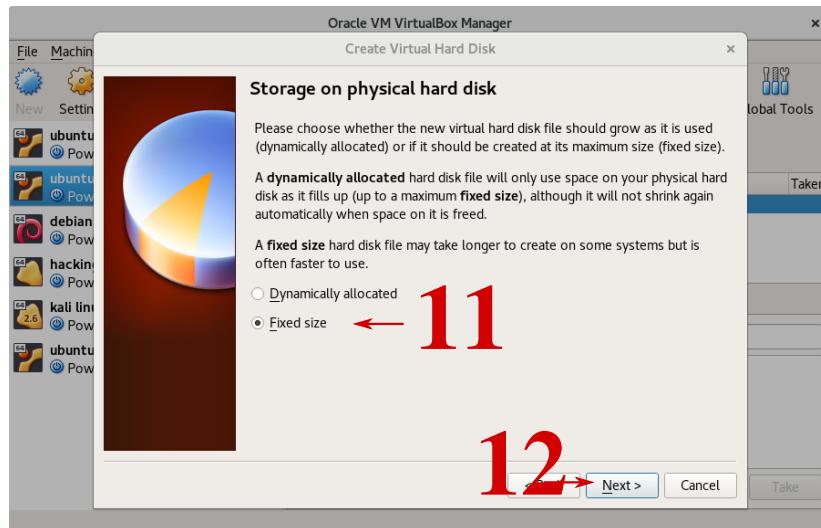
7. En la configuración del disco duro, seleccionar la opción “**Create a virtual hard disk now**”.

8. Seleccionar “Create” para crear la máquina virtual.
9. Seleccionar el tipo de archivo que almacenará el disco duro de la máquina virtual.  
Escoger la opción **VDI (VirtualBox Disk Image)**.
10. Click en “Next”.



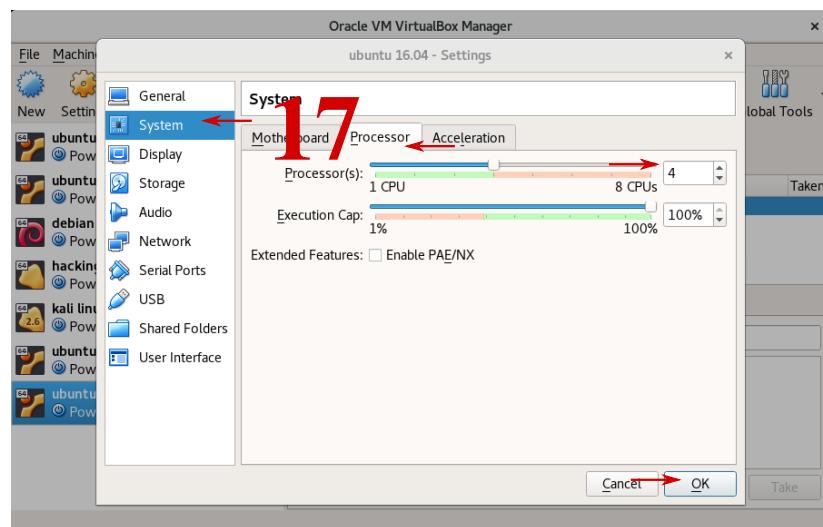
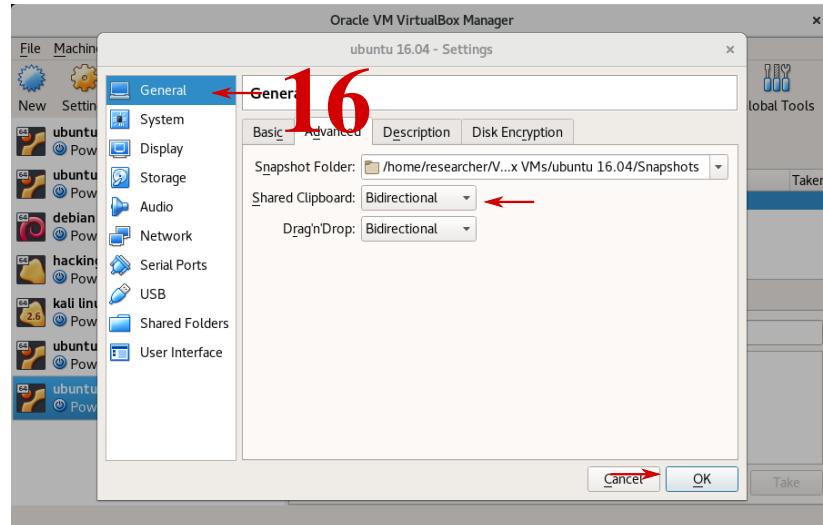
11. El tipo de almacenamiento es **“Fixed size”**.
12. Click en “Next”.
13. Escoger la ruta donde va a ir el disco duro virtual.
14. Seleccionar el espacio del disco duro virtual. Para OpenBTS, 25.00 GB es espacio más que suficiente.

15. Click en “Create”.

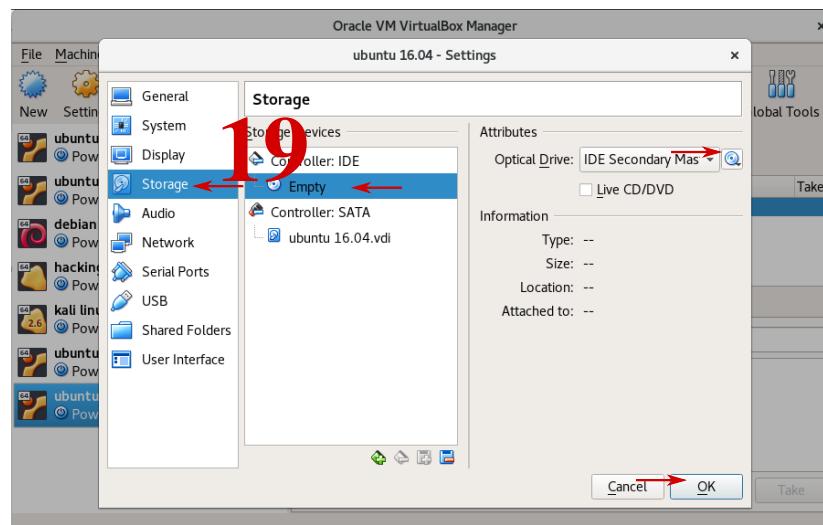
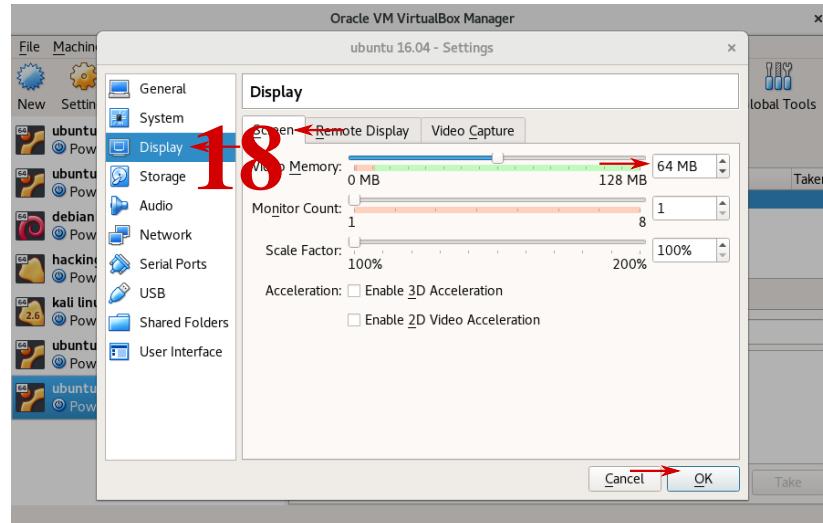


Luego hay que entrar al menú de configuraciones de la máquina virtual y realizar estas modificaciones:

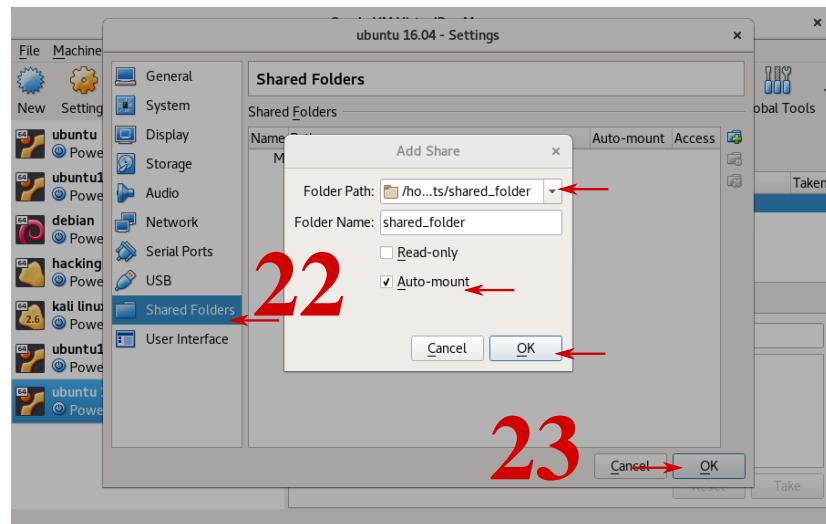
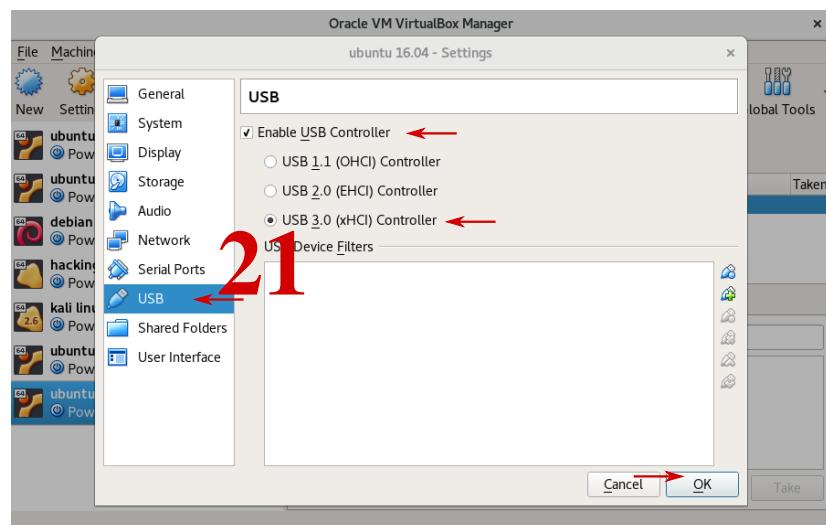
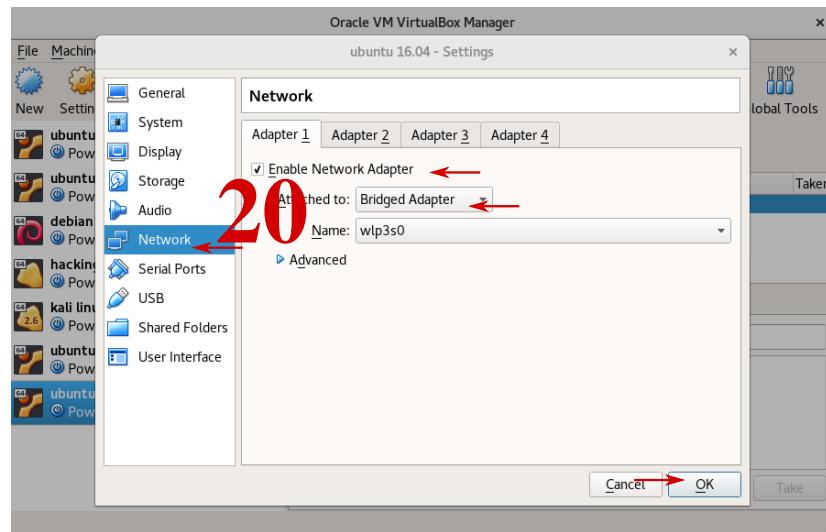
16. En las opciones de la columna izquierda ir a General y en la pestaña “Advanced”, colocar las opciones **Shared Clipboard** y **Drag and drop**, ambas en “Bidirectional”.
17. En la opción System, en la pestaña de “Processor”, colocar el número de procesadores. Para esta instalación se usa 4 CPUs.
18. En la opción Display, en la pestaña “Screen”, colocar el Video Memory en 64 MB.

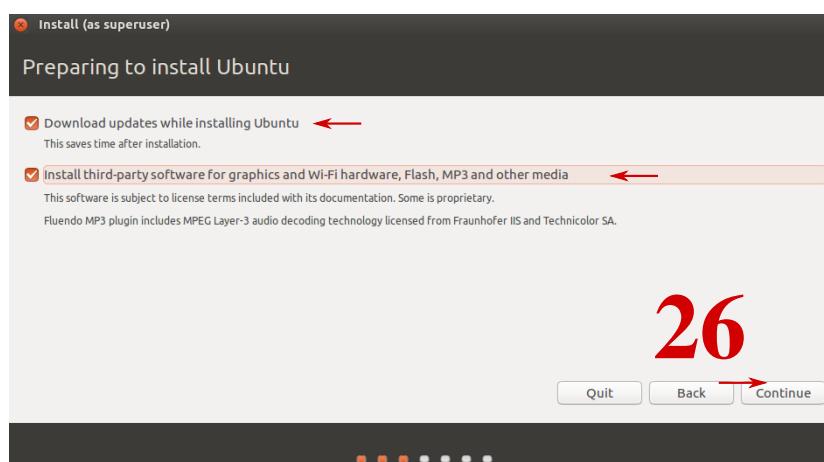
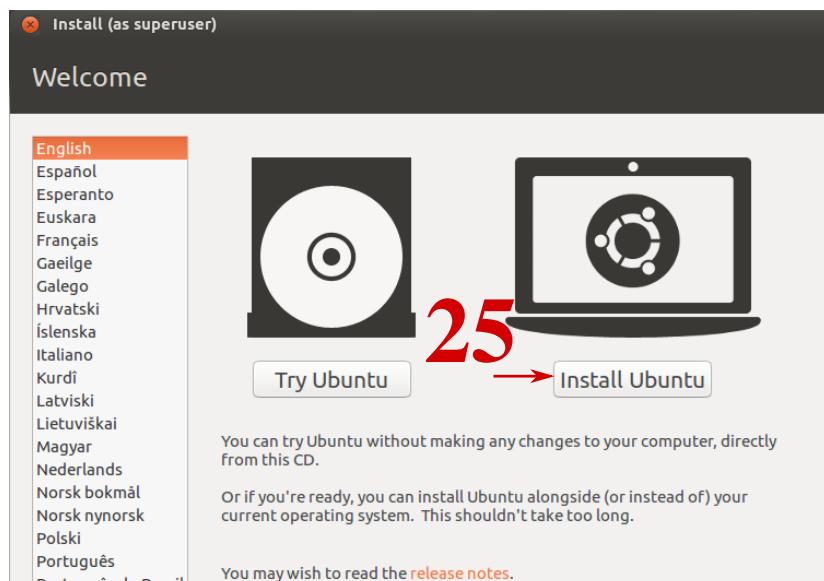
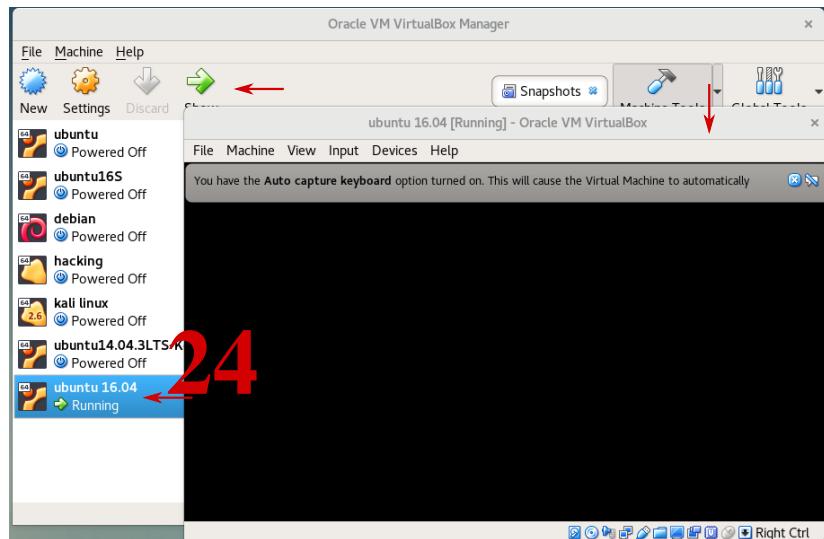


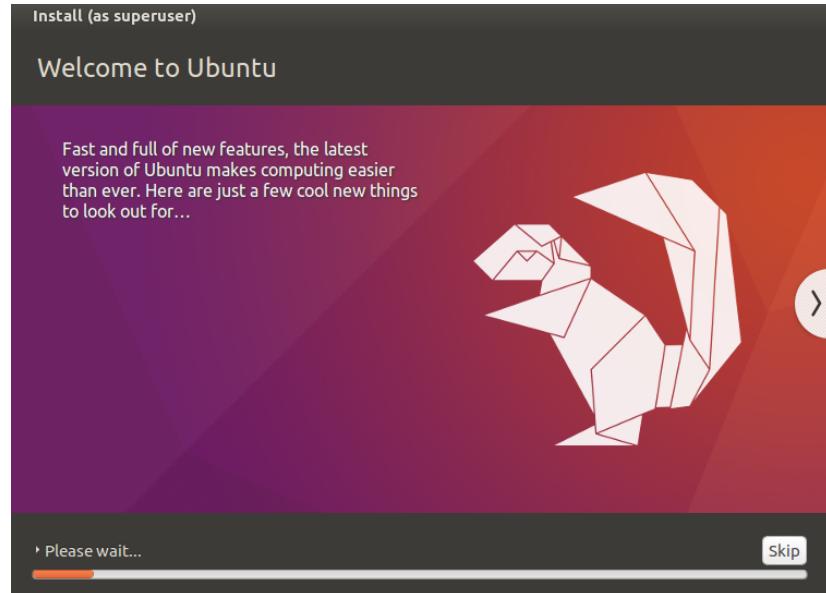
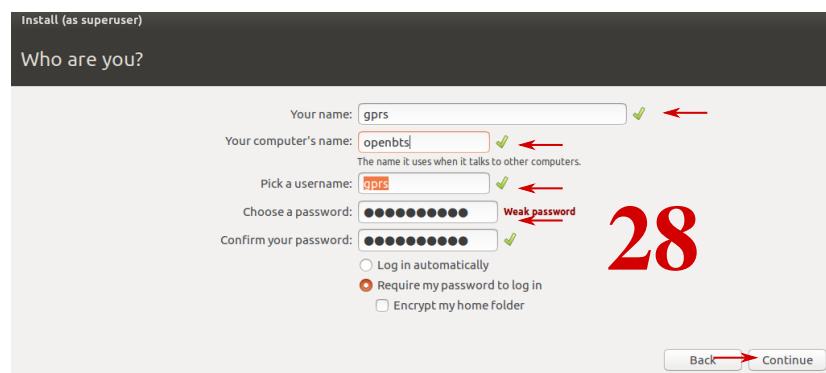
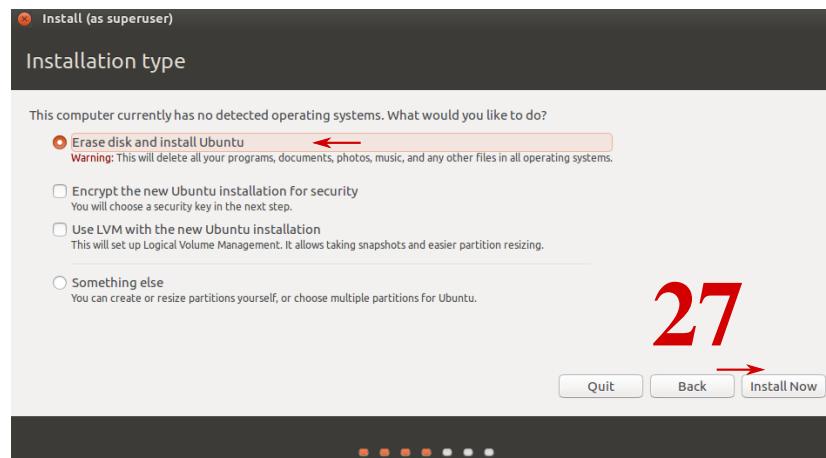
19. En la opción Storage, en la sub-sección “Storage Devices”, seleccionar la opción Controller IDE (cuyo parámetro aparece como “Empty”), y en los Atributos, señalar en “Optical Drive” la ruta del archivo instalador ISO de Ubuntu 16.04. El instalador debe ser descargado de REFERENCIA.
20. En la opción Network, habilitar el **Enable Network Adapter**. En “Attached to”, seleccionar **Bridged Adapter**.
21. En la opción USB, habilitar el **Enable USB Controller** y seleccionar **USB 3.0 (xHCI) Controller**.
22. En la opción Shared Folders, especificar la ruta donde se ubica la carpeta compartida entre el Host y la máquina virtual. La opción Auto-mount es necesaria para evitar montar de manera manual la carpeta compartida. Click en Ok.



23. Finalmente click en Ok para cerrar el menú de configuraciones.
24. Seleccionar la máquina virtual e inicializar.
25. Seleccionar la opción **Install Ubuntu**.
26. Seleccionar las opciones correspondientes a descargar actualizaciones e instalar software de terceros, drivers, etc. Luego clicl en “Continuar”.
27. Seleccionar “Erase disk and Install Ubuntu” y luego en “Install Now”. Luego en la Instalación pide seleccionar la ubicación geográfica y el idioma del sistema.
28. Completar los datos de identidad de la cuenta de usuario del Ubuntu 16.04.
29. La instalación inicia y una vez que finaliza va a pedir reiniciar el sistema.







## **ANEXO B**

**CÓDIGO ARDUINO PARA CONEXIÓN GPRS Y ENVÍO DE PINGS.**

```

#include <SoftwareSerial.h>
#include <TimeLib.h>

SoftwareSerial mySerial(10, 11); // RX, TX
#define arraySize 128
int counter = 0;
int success_counter = 0;
int error_counter = 0;
int i;
int temp;

void setup() { // Open serial communications and wait for port to open:
  Serial.begin(9600);
  mySerial.begin(9600);
  initGPRS(); // inicia el modulo GPRS
}

void initGPRS(){
  connectGPRS("AT+CIPSHUT", "SHUT OK");
  delay(2000);
  connectGPRS("AT+CREG=0", "OK");
  delay(2000);
  connectGPRS("AT+CREG=1", "OK");
  delay(2000);
  connectGPRS("AT+CGATT=1", "OK");
  delay(2000);
  connectGPRS("AT+CIPSHUT", "SHUT OK");
  delay(2000);
  connectGPRS("AT+CSTT=\\"CMNET\\\"", "OK");
  delay(2000);
  connectGPRS_CIICR("AT+CIICR", "OK");
  delay(5000);
  connectGPRS_CIFSR("AT+CIFSR");
  delay(1000);
  connectGPRS("AT+CENG=4", "OK");
  delay(1000);
}

void connectGPRS(String cmd, char *res){
  while(1){
    Serial.println(cmd);
    mySerial.println(cmd);
    delay(500);
    while(mySerial.available()>0){
      if(mySerial.find(res)){
        Serial.println(res);
        delay(2000);
        return;
      }
    }
    delay(2000);
  }
}

void connectGPRS_CIICR(String cmd, char *res){
  while(1){
    Serial.println(cmd);
    mySerial.println(cmd);
    delay(10000);
    while(mySerial.available()>0){
      if(mySerial.find(res)){
        Serial.println(res);
        delay(1000);
        return;
      }
    }
  }
}

void connectGPRS_CIFSR(String cmd){
  static char buffer[256];
  static size_t pos;
}

```

```

mySerial.println(cmd);
delay(500);
while(mySerial.available() && pos < sizeof buffer - 1){
    char c = mySerial.read();
    buffer[pos++] = c;
    if(c == '\n'){
        buffer[pos] = '\0';
        Serial.print(buffer);
        pos = 0;
    }
}
}

bool connectGPRS_PING(){
bool ping_ok = true;
char inData[arraySize];
mySerial.println("AT+CIPPING=\"192.168.216.119\",1,1024\r");
delay(10000);
int c = 0;
while(mySerial.available()){
    inData[c] = mySerial.read();
    Serial.print(inData[c]);
    c++;
}
Serial.println();
if(strstr(inData, "600")){
    ping_ok = false;
}
else{
    ping_ok = true;
}
return ping_ok;
}

void connectCENG(){
char inData2[arraySize];
mySerial.println("AT+CENG?\r");
delay(1000);
int c = 0;
while(mySerial.available()){
    inData2[c] = mySerial.read();
    Serial.print(inData2[c]);
    c++;
}
Serial.println();
}

void loop(){// run over and over
Serial.println( String(hour()) + ":" + String(minute())+":"+String(second()));
connectCENG();
bool value = connectGPRS_PING();
if (value == true){
    success_counter++;
}
else{
    error_counter++;
}
counter = success_counter + error_counter;
Serial.println((String)"Total paquetes enviados: " + counter);
Serial.println((String)"Ping errados: " + error_counter);
Serial.println((String)"Ping correctos: " + success_counter);
temp = random(10,50)*100;
delay(temp);
}
}

```

## **ANEXO C**

**CÓDIGO ARDUINO PARA CONEXIÓN GPRS Y ENVÍO DE DATOS AL SERVIDOR DE *THE THINGSPEAK*.**

```

#include <SoftwareSerial.h>
int midZ=2000;//time delays
int minZ=500;
int maxZ=5000;
int random_value; // parametro a ser transmitido a TheThingSpeak

SoftwareSerial MySerial(10, 11); // creacion de la coneixon serial

void setup(){
  Serial.begin(9600);
  while (!Serial);
  MySerial.begin(9600);
  delay(midZ);
  MySerial.println("at+csq"); //verificacion de parametros de senal
  delay(minZ);
  feedback();
  MySerial.println("at+cipshut");//desactivacion de cualquier conexion GPRS
  delay(minZ);
  feedback();
  MySerial.println("at+cstt=\"cmnet\",,\"\",,\"\"");//configuracion de APN
  feedback();
  MySerial.println("at+ciicr");//iniciar conexion con red celular
  feedback();
  MySerial.println("at+cifsr");//solicitar una IP para transmision de datos
  feedback();
}

// funcion que verifica el buffer de entrada y si encuentra algo lo imprime.
void feedback(){
  delay(midZ);
  while (MySerial.available()) {
    Serial.write(MySerial.read());
  }
}

void loop(){ // funcion loop
  MySerial.println("at+cipstart=\"TCP\",\"api.thingspeak.com\",\"80\"");//conexion TCP
  feedback();
  MySerial.println("at+cipsend=71");//declara el numero de caracteres a enviarse
  feedback();
  MySerial.print("GET /update?"); //Escribe el comando GET para enviar data al...
  feedback();
  MySerial.print("api_key=VBI484F2F3BQ4WNN");//...canal API de TheThingSpeak...
  feedback();
  MySerial.print("&field1="); //...especificando el campo (temperatura)...
  feedback();
  random_value = random(70,100); // ... el cual es un valor aleatorio entre 70 y 100...
  MySerial.println(random_value); //se cierra el comando AT+CIPSEND
  feedback();
  MySerial.println("at+cipack");//sobre la misma conexion TCP, este comando solicita al
  //servidor
  //conocer si el ultimo envio fue exitoso o no
  feedback();
  MySerial.println("at+cipclose");//cerrar la conexion TCP
  feedback();
  delay(8000);
}

```