ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA INFORMÁTICA
GRADO EN INGENIERÍA INFORMÁTICA

VULNERABILIDADES USANDO SDR
VULNERABILITIES USING SDR

Realizado por
Carlos Ledesma Peña
Tutorizado por
Isaac Agudo Ruiz
Departamento
Lenguajes y Ciencias de la Computación

UNIVERSIDAD DE MÁLAGA
MÁLAGA, DICIEMBRE DE 2014

Fecha de defensa:
El Secretario del Tribunal

Resumen: Un sistema de *SDR* (*Software Defined Radio*) es un sistema de radio programable que delega gran parte del procesamiento hecho clásicamente en hardware, en software corriendo en un ordenador. Dos ventajas inmediatas de un dispositivo de *SDR* frente a un dispositivo de radio tradicional son el abaratamiento del coste del hardware (menos y menos complejo) y la facilidad de modificación de la funcionalidad específica de la radio (implementaciones software, tan simple como programar cualquier protocolo deseado). Debido al abaratamiento de estos productos y su facilidad de programación e interconexión con un ordenador personal, el mundo de la radiocomunicación es bastante más accesible. Cuando un dominio es poco conocido o accesible, es típico que los sistemas no sean seguros por diseño, sino por oscuridad. Si en un corto periodo de tiempo la accesibilidad a ese dominio aumenta considerablemente, los sistemas seguros por oscuridad se encuentran en peligro. Este trabajo pretende estudiar si efectivamente, al ser más accesible el dominio de la radiocomunicación debido a la accesibilidad de los dispositivos de *SDR*, ciertos sistemas se encuentran expuestos. La investigación del estado del arte y el estudio práctico de sistemas públicos en el ámbito local, nos permitirá entender hasta qué punto existen riesgos reales. Si se encuentra en el ámbito local una vulnerabilidad en algún sistema, se documentará y se propondrá una posible forma de aprovecharla y solucionarla.

Palabras claves: radio definida por software, radiofrecuencia, radiocomunicación, seguridad, radio programable, vulnerabilidades, prueba de penetración, pen testing, pager, pocsag, gnu radio, usrp

Abstract: A *SDR* (*Software-defined radio*) system is a programmable radio system which carries most of the processing via software running on a computer, instead of running it on hardware, as usual. Two immediate benefits of a *SDR* device when compared to classic radios are the hardware reduced costs (less and less complex) and how easy is to modify the specific features of the radio (software implementations, as simple as programming any desired protocol). Due to this products becoming cheaper and cheaper and the ease of programming and connecting with a personal computer, the field of radiocommunications is more accesible. When a field is not well-known or accesible, it's usual for systems to not be secure by design, but by obscurity. If in a short period of time the accesibility to that field grows significantly, secure by obscurity systems are in danger. This thesis pursues research to find if this phenomena occurs and several systems are exposed, because of the radiocommunication field being more accesible. A state-of-the-art overview, along with practical research of public systems in the local environment, will allow us to understand how real are these risks. If a vulnerability in a system in the local enviroment is found, it would be documented and a way to exploit and solve the vulnerability would be proposed.

Keywords: software defined radio, radiofrequency, radiocommunication, security, programmable radio, vulnerabilities, penetration test, pen testing, pager, pocsag, gnu radio, usrp

Índice

1. Introducción

1.1. Motivación

Según pasa el tiempo, la seguridad informática adquiere una importancia mayor. Los sistemas informáticos son cada vez más numerosos, y se integran en otros sistemas con el objetivo de gestionar de manera eficiente la información de estos sistemas. Pero allá donde la automatización sea posible, el papel de un sistema informático ya no es únicamente de gestor de información, sino que usa para controlar el sistema donde está integrado.

Como ejemplos de sistemas informáticos, tenemos el control industrial en centrales nucleares y la gestión de la hacienda pública. Estos dos casos extremos nos ayudan a entender por qué la seguridad de los sistemas informáticos integrados en estos casos es crucial. Las consecuencias de la manipulación de estos sistemas con fines destructivos son nefastas. Como ejemplo relativamente reciente, tenemos el gusano informático *Stuxnet*, instrumento principal de un ataque dirigido cuya consecuencia fue el sabotaje de sistemas *SCADA* (control industrial) usados en el programa nuclear iraní [1, 2]

Proteger cualquier sistema informático de una manipulación indebida es el objetivo de la seguridad informática, y genera sin duda motivación suficiente, debido a la importancia de asegurar la integridad, la confidencialidad y la disponibilidad de cualquier tipo de sistema. ¿Acaso no es interesante asegurar que no se pueda manipular la información de una transferencia bancaria (integridad), que no sean legibles por terceros tus correos electrónicos personales (confidencialidad) o asegurar el funcionamiento de un sistema de gestión de intervenciones en ambulancia (disponibilidad)?

Específicamente, y ya sobre la materia que trata este trabajo, la telecomunicación a través de radio (a partir de ahora, radiocomunicación) es un área donde la seguridad cobra especial importancia. Debido a que un ataque a la seguridad de sistemas informáticos a través de sus radiocomunicaciones puede ser llevado a distancia y de forma anónima, y el uso creciente de radiocomunicación en sistemas informáticos y unas herramientas de ataque cada vez más accesibles, es necesario investigar si debemos preocuparnos de la seguridad de los sistemas que usen radiocomunicación.

Finalmente, por mi inclinación personal al mundo de la seguridad informática, y la ausencia de formación sobre telecomunicaciones en mi itinerario educativo, la realización de un trabajo sobre este tema me parece la excusa ideal para aprender los rudimentos de las telecomunicaciones, y así añadir conocimientos y herramientas nuevas a mi perfil profesional, que estoy orientando a la seguridad informática.

1.2. Objetivos

Este trabajo tiene cuatro objetivos. El primero es hacer una investigación sobre el estado del arte, intentando contar el desarrollo de la tecnología *SDR* a lo largo de los años, junto con las vulnerabilidades más destacadas que se atacan en cada momento. No pretende ser exhaustivo, simplemente dar una visión rápida de la evolución de este ámbito

El segundo objetivo es hacer una introducción a los conceptos básicos de radiocomunicación y procesamiento digital de señales (*DSP*), a un nivel bastante básico, lo suficiente para entender la base teórica del caso práctico que se presenta como cuarto objetivo.

El tercer objetivo es introducir el uso del kit de desarrollo *GNU Radio*, más concretamente la interfaz gráfica que envuelve muchas de sus funcionalidades, *GNU Radio Companion* (a partir de ahora *GRC*). De nuevo, a un nivel suficiente para poder seguir y reproducir el caso práctico.

Y finalmente, el cuarto objetivo es aplicar los conocimientos teóricos y prácticos anteriormente en realizar el estudio de una vulnerabilidad existente en algún sistema del mundo real. En este caso, elegimos un *pager* que usa el protocolo *POCSAG*. Un *pager* es una radio muy simple que está sintonizada a una estación fija, y está a la escucha de mensajes (alertas simples, texto, voz...) que le puedan llegar (algunos también pueden emitir). Este *pager* en concreto simplemente suena y vibra si la estación emisora envía su identificador (cada *pager* tiene uno), y se suele usar en restaurantes para avisar de que la comida está lista.

1.3. Estructura de la memoria

Cada objetivo mencionado anteriormente se corresponde con una parte de la memoria. En la sección 2, ”Estado del arte”, tratamos el primer objetivo, dar una visión rápida del estado del arte y la evolución de la tecnología *SDR* en el tiempo. A continuación, en la sección 3, ”Conceptos básicos”, tratamos el segundo objetivo, introducir conceptos básicos de radiocomunicación y *DSP* a un nivel básico.

En la sección 4, ”Introducción a GNU Radio”, tratamos el tercer objetivo, introducir el kit de desarrollo *GNU Radio*, más concretamente su interfaz gráfica principal, *GNU Radio Companion* (*GRC*). Y finalmente, en la sección sección 5, ”Caso práctico: pager POCSAG”, tratamos el cuarto objetivo, relatar de forma secuencial en el tiempo la realización del caso práctico consistente en estudiar una vulnerabilidad en un *pager POCSAG* usado en un restaurante.

2. Estado del arte

2.1. Preliminares

Antes de empezar la descripción del estado del arte y la evolución de la tecnología, es necesario dar una definición de *SDR* y por qué supone una mejora frente al concepto de radio tradicional:

¿Qué significa Software Defined Radio*?*

SDR Forum, en colaboración con el grupo de trabajo *P1900.1* de *IEEE*, trabajó en unas definiciones que traduzco y adapto para este trabajo [4]:

Software Defined Radio
Radio en la cual algunas o todas las funciones de las capas físicas están implementadas en software.

Radio
a) Tecnología que transmite o recibe sin cables radiación electromagnética para transferir información
b) Sistema o dispositivo incorporando tecnología como la definida en (a)

¿Qué diferencias hay entre la radio tradicional y SDR*?*

En los dispositivos de radio tradicional (en un receptor, por ejemplo), todo el proceso de la parte de radiofrecuencia, desde la recepción de la señal hasta la demodulación, es realizado en hardware. Tanto la arquitectura del hardware como la mayoría de los parámetros asociados al procesamiento de la señal son fijos, optimizados para el tipo de señal que están diseñados para recibir.

En los dispositivos de *SDR*, el objetivo es delegar la mayor parte parte posible del procesamiento de la señal de radiofrecuencia, en implementaciones software. De este modo, recibir (o emitir) cualquier tipo de señal es posible simplemente cambiando el software. Básicamente, disponemos de un dispositivo de radio universal y programable. Además de la cantidad de ventajas que nos da el poder trabajar con información digitalizada, como por ejemplo el almacenamiento (permite grabar la señal en bruto para después reproducir experimentos) o la escalabilidad (si se necesita más capacidad de procesamiento, basta con usar más máquinas).

2.2. Comienzos de la tecnología

El nacimiento de la tecnología *SDR* viene, como les pasa a otras muchas tecnologías, de mano del sector de defensa[1]. En la década de los 70 ya había interés en este sector por desarrollar un sistema de radio que fuese flexible [6], y en el año 1984 un equipo de la compañía entonces llamada *E-Systems* (y ahora llamada *Raytheon*) desarrolló un prototipo primitivo de dispositivo *SDR* (únicamente con capacidad de recepción), y acuñó el término *Software Radio* para describirlo [8].

Ya en 1991 nació el programa *SPEAKeasy*, por parte de *DARPA*[2], uno de los primeros proyectos públicos de *SDR*. El primer objetivo era que un único dispositivo de radio (ya con capacidad de transmisión y recepción) pudiese abarcar un rango de frecuencias muy amplio (de 2 MHz a 2 GHz) y fuese compatible con más de 10 dispositivos militares de radio ya existentes [7]. Y el segundo objetivo era que fuese fácil incorporar nuevos estándares de codificación y modulación, que ha terminado siendo la característica más interesante del concepto de *SDR*.

Con el objetivo de fomentar el avance de las tecnologías relacionadas con *SDR*, en 1996 se funda la primera asociación de la industria en este ámbito, con el nombre de *The Modular Multifunction Information Transfer System* (*MMITS*) *Forum*. Cambió de nombre en 1998 (*SDR Forum*) y otra vez en 2010 (*Wireless Innovation Forum*). Este foro estaba y está formado por gente y organizaciones del gobierno, la industria y el ámbito académico, organizados en grupos de trabajo y comités para estimular y dirigir la innovación y crear estándares.

La transición entre la década de los 90 y el presente milenio se caracteriza por ser la etapa en la que empieza a ser accesible a la investigación pública la tecnología *SDR*. En 1998 se puso a la venta una de las primeras plataformas de desarrollo de *SDR*, la primera versión de *SignalMaster*, consistente en una placa uniendo las tecnologías *DSP* (*Texas Instruments*) y *FPGA* (*Xilinx*), y programable a través de modelos *Simulink*. Y en 2001 fue publicado por primera vez el proyecto *GNU Radio*, el entorno de desarrollo *open source* más importante todavía a día de hoy.

Otro hecho destacado es la aprobación por parte de la *FCC*[3] del primer dispositivo *SDR* usado con fines comerciales, una estación base (de telefonía móvil) corriendo simultáneamente los modos *GSM* y *CDMA*. El software de todas las capas de ambos protocolos corría en procesadores *x86*; una implementación *SDR* completa.

[1] Nota: En esta sección se referencian múltiples documentos similares, los cuales fusiono para conseguir un texto más completo y consistente. Cada documento es citado la primera vez que uso parte de su contenido, pero puede aparecer otra parte más adelante en el texto

[2] *Defense Advanced Research Projects Agency*, agencia del Departamento de Defensa de EE.UU.

[3] *Federal Communications Commission*, agencia del gobierno de EE.UU.

2.3. Aumento de la accesibilidad

Pero lo que comenzó a hacer accesible la tecnología *SDR* (siempre y cuando pudieses pagarla, como ahora veremos) fue la comercialización de *Small Form Factor* en 2006 [9], la primera plataforma de desarrollo *SDR* completa y autónoma, ya que tenía todo los componentes necesarios para empezar a desarrollar al momento (contenía un *front-end RF*[4], al contrario que la primera versión de *SignalMaster*)[5].

A pesar del precio (que rondaba los 10,000 dólares [10]), se puede afirmar que hizo más accesible la tecnología por venir listo para usar, como ya hemos dicho antes. Uno se puede imaginar que esto es interesante cuando el objetivo no es estudiar la radiocomunicación en sí, sino usar la tecnología *SDR* como una herramienta para diseñar rápidamente un sistema de radiocomunicación a medida. El no tener que plantearse, por ejemplo, antes de empezar a desarrollar qué características tiene que tener el *front-end SDR* para tu aplicación (y empezar usando el que viene de serie) reduce la curva de nivel, permitiéndote enfrentar los problemas según se desarrolla la aplicación. Esto es especialmente útil cuando no se tiene un conocimiento suficiente de radiocomunicación, pudiendo hacerse un prototipo más o menos funcional, e ir afinándolo según se aprende.

2.4. Primeras preocupaciones por la seguridad

Según se facilita el acceso a la tecnología *SDR*, empieza la preocupación por los posibles impactos sobre la seguridad de los sistemas que usan radiocomunicación. Debido a que la tecnología *SDR* tuvo como uso temprano el desarrollo de sistemas de radiocomunicación proveedores de servicios de uso masivo, como los de telefonía móvil, curiosamente, la preocupación inicial era por la seguridad de los propios sistemas usando tecnología *SDR*, más que por los sistemas que podían ser vulnerados utilizando herramientas usando esta tecnología [11]. Esto queda reflejado en las publicaciones sobre seguridad y *SDR* de principios de este siglo.

Sin negar la importancia de la seguridad en los sistemas de producción usando *SDR*, en este trabajo estudiamos el otro aspecto comentado, si es una amenaza la creciente accesibilidad de la tecnología *SDR* para la seguridad de sistemas que usen radiocomunicación.

[4] Etapa del sistema dedicada al procesamiento de señales de radiofrecuencia; se encuentra entre la antena y la etapa de frecuencia intermedia. Una definición más técnica se dará más adelante

[5] En 2003 salió un transceptor (transmisor-receptor) que utilizaba la tecnología *SDR* y comunicación con un ordenador (sobre los 1,400 dólares además), pero no estaba orientado al desarrollo, sino a los radioaficionados (y por tanto, sus características técnicas y funcionalidades eran distintas) [12, 13]

2.5. Generalización y bajo coste

A partir de 2006, la tecnología *SDR* se abarata significativamente, hasta el punto de usarse implementaciones de esta tecnología en productos de consumo masivo, como veremos más adelante (de hecho, esto permitirá el acceso a la tecnología al aficionado casual, simplemente por el bajo coste).

En un segmento intermedio, por debajo del desarrollo profesional y de la investigación académica, ya habían empezado a surgir varios proyectos que veían en la tecnología *SDR* una forma más flexible y barata de implementar radios para radioaficionados [14, 15]. Algunas de estas radios básicamente consisten en un *front-end RF* (que además en la mayoría de los casos sus parámetros son controlables por ordenador) que transforma la señal de radiofrecuencia en una señal de frecuencia intermedia. Esta última señal es enviada a un ordenador, que la recibe a través de una tarjeta de sonido (al fin y al cabo la entrada de línea de una tarjeta de sonido es principalmente un convertidor analógico-digital).

Lógicamente, que la comunicación con el ordenador sea a través de una tarjeta de sonido (que no está construida para estos propósitos) es una limitación para casi cualquier propósito fuera de la radioafición. Muestreando en cuadratura[6] (con una entrada estéreo muestreando a 48 kHz, típico en tarjetas de sonido integradas), el ancho de banda teórico es de 48 kHz[7].

Ya en 2007 es cuando se empiezan a ver los primeros reportes públicos de ataques a vulnerabilidades usando la tecnología *SDR*. Una publicación con el divertido nombre alternativo de *We know what you typed last summer* (Sabemos lo que tecleaste el último verano) presentaba un ataque sobre teclados inalámbricos (un par de modelos de Microsoft y Logitech) [16]. El ataque consistía en capturar la radiocomunicación a 27 MHz que había entre el teclado y el receptor conectado a un ordenador, y descifrarlo (para lo que tuvieron que hacer ingeniería inversa al protocolo de radiocomunicación), de forma que se podían obtener todas las pulsaciones, sin que la víctima lo supiera, y sin colaboración de ésta. Además de poder inyectar las pulsaciones que se quisieran.

Realmente, el problema no era que se estuviese usando comunicación a través de radio, el problema era que el protocolo usado era inseguro. A pesar de las complicaciones para la seguridad que presenta una comunicación inalámbrica, se puede establecer una comunicación segura si el protocolo está bien diseñado. La tecnología *SDR* simplemente facilita el acceso a la radiocomunicación, y obviamente, no puede hacer nada contra radiocomunicaciones seguras.

[6] Más adelante explicaremos qué significa esto

[7] Para hacerse una idea, el ancho de banda requerido según el estándar moderno de una estación de radio FM comercial es de 200 kHz

Un ejemplo es la posibilidad de fabricarse un lector de etiquetas *RFID*. Ya este hecho en sí mismo puede ser peligroso (se puede leer información de etiquetas RFID que usen protocolos sin encriptación), aunque no es nada a lo que no se pueda acceder con un lector comercial específico (para ese protocolo y frecuencia). La amenaza real viene cuando existe la posibilidad de reprogramar el lector para salirse del comportamiento esperado, lo que te permite atacar un protocolo vulnerable.

A veces, con escuchar es suficiente. Un equipo de seguridad informática de una universidad alemana presentó un ataque contra los pasaportes europeos [22]. Capturando la comunicación entre un lector autorizado y la etiqueta RFID (para lo cual es bastante adecuada una implementación *SDR*), es posible descifrar la parte que contiene los datos personales básicos (nombre, sexo, fecha de nacimiento, nacionalidad, número de pasaporte, fecha de caducidad y una foto de la cara). Esto se debe a que *BAC* (*Basic Access Control*, el protocolo usado para negociar la clave de sesión usada para cifrar los datos personales básicos, usa criptografía simétrica (*Triple-DES*) con una clave derivada de algunos datos impresos en el pasaporte.

Esto reducía el espacio de búsqueda de la clave, ya que la clave se deriva de tres datos , número de pasaporte, fecha de nacimiento y fecha de caducidad, que ocupaban en total 24 bytes (pero no 24 bytes efectivos, a carácter alfanumérico o numérico por byte). Entonces se prueba por fuerza bruta qué datos generaban una clave que al encriptar cierto valor conocido, el resultado coincidía con otro valor conocido (ambos valores estaban en la comunicación). Este tipo de ataque se conoce como *known plaintext attack* (ataque de texto plano conocido) [23], y en esta publicación de hecho se enfocan en este aspecto (en hacer un ataque criptográfico en un tiempo asumible), más que en la adquisición y procesamiento de los datos radiocomunicados.

Si se estaba lo suficientemente cerca (4 metros según el estado del arte del momento) para recibir las respuestas del pasaporte al lector, se podían obtener todos los datos personales, ya que se obtenía la clave y además, los datos personales encriptados con esa clave. Si no se estaba lo suficientemente cerca (y al menos a 25 metros), sólo se obtendría la clave. Lo cual también es útil, pues serviría, además de para identificar de forma inequívoca a un pasaporte a distancia, para comunicarse posteriormente de forma satisfactoria con el pasaporte y obtener los datos personales.

En 2008, el investigador en seguridad de comunicaciones inalambricas Michael Ossmann dio una conferencia en el evento sobre seguridad informática *Black Hat*[8]. En la conferencia, llamada *Software Radio and the Future of Wireless Security* [17], Ossmann hacía un repaso sobre lo que había sido vulnerado hasta ahora que usase radiocomunicación, además de insistir en que la tecnología *SDR* en un estado tan accesible podría ser peligroso.

[8]Los archivos sobre ediciones pasadas de los eventos son fuentes bastante útiles para conocer el estado del arte de una época en particular

Entre otros sistemas usando radiocomunicación que comentaba que habían sido vulnerados, podemos destacar ciertos avances en comprometer la seguridad del estándar de telefonía móvil *GSM* (un año más tarde ya se demostró inseguro), de dispositivos Bluetooth o de marcapasos. Estos últimos, siendo unos dispositivos médicos tan críticos, no tenían seguridad más que por oscuridad del protocolo usado, y se podía obtener información del paciente, además de provocar choques eléctricos a voluntad. Estos tres proyectos comentados se habían llevado a cabo usando la combinación *GNU Radio* [20] y *USRP* [21], bastante usada en este ámbito, sobre la que hablaremos más adelante (y usaremos en el caso práctico de este trabajo).

Ese mismo año, pero en otro evento sobre seguridad informática, *DEFCON*, se iba a presentar un estudio sobre las vulnerabilidades de los sistemas informáticos del sistema de metro de Boston [18], que se canceló por orden judicial [19]. Para estudiar las vulnerabilidades del sistema de pago *RFID*, se usó también la combinación *GNU Radio* y *USRP*. El sistema completo era vulnerable, y concretamente, el cifrado del sistema de pago *RFID* se rompió por un ataque de fuerza bruta con texto plano conocido usando tecnología *FPGA*, lo que permitía clonar tarjetas.

Otro ataque más antes de terminar con el año 2008 (que fue bastante productivo en la búsqueda de vulnerabilidades en sistemas de radiocomunicación), esta vez sobre dispositivos de pago automático en peajes (de forma remota, a través de radiocomunicación) [24]. Básicamente se descubrió que el estándar *Fastrak*, usado en California, no tenía seguridad ninguna más allá de la que da el desconocimiento de un sistema. Imitando al dispositivo lector, se puede sacar el identificador asociado al dispositivo del cliente (pudiendo usarse para clonarlos), y además se puede realizar un ataque por repetición, simplemente capturando la respuesta y repitiéndola, sin ser necesario el estudio del estándar. En el trabajo original no se usó tecnología *SDR*, pero sería ideal para realizar los ataques mencionados.

Finalmente, antes de pasar a la siguiente etapa en la evolución de la tecnología *SDR*, merece la pena mencionar otros ataques importantes. En 2009 se consiguió romper de forma práctica el algoritmo de cifrado *A5/1* usado en *GSM* [25]. Ya se había roto hace años de forma teórica, y también alguna aproximación práctica, pero muy costosa computacionalmente. En 2010 se presentó un trabajo bastante interesante sobre la posibilidad de leer ciertas etiquetas *RFID* (como las que incluyen las nuevas licencias de conducir de EE.UU) a una distancia de 150 metros, y además se explicaba los riesgos de seguridad que esto entraña [26]. Y en 2011, en una conferencia en *Black Hat DC* se presenta un ataque *man-in-the-middle*[9] a sistemas usando las tecnologías *GPRS/EDGE y UMTS/HSPA* en el que se comprometía completamente la conexión a Internet. Se llevaba a cabo simulando ser una estación base, y forzando a los terminales a usar siempre *GPRS/EDGE* sin encriptación [27].

[9] Interceptación de una conexión con el objetivo de obtener, modificar o inyectar información en ésta

2.6. Más barato, imposible

Retrocedamos un poco en el tiempo. En marzo de 2010, Eric Fry estaba haciendo ingeniería inversa a un dispositivo *USB* de recepción *FM* y *DAB+*[10] (que contenía el chip *Realtek RTL2382U*, que combina un convertidor analógico-digital y un controlador *USB*)para un ordenador personal [28]. Concretamente, capturaba los paquetes *USB* provenientes del dispositivo destinados a la aplicación para Windows que servía de reproductor, con el objetivo de crear un *driver* para hacerlo compatible con el *kernel Linux*. Pero se dio cuenta de que la información que contenían los paquetes no era a la que él estaba acostumbrado (Eric ya había trabajado con dispositivos de este tipo).

En ese momento, parece ser que tenía otros proyectos con prioridades mayores y lo dejó algo aparcado. Tras varias conversaciones públicas y privadas con gente dedicada a este tema en 2011, presentó a principios de 2012 la información que obtenía capturando los paquetes *USB*, que a él le parecían muestras de la señal en una etapa intermedia entre la señal de radiofrecuencia y la señal de audio. Subió estas muestras a su página web personal, abrió un nuevo tema en una lista de correo de la temática y enlazó el contenido de las muestras, animando a que el resto de personas lo decodificase.

Un tal Alistair Buxton lo consiguió, descubriendo que el dispositivo devolvía muestras *I/Q*, y que el reproductor para Windows se encargaba de efectuar la demodulación en el ordenador. Básicamente, encontraron una versión muy barata de implementación *SDR* con conexión a un ordenador personal. El concepto evoluciona, se desarrolla software usable y finalmente nace el proyecto *rtl-sdr*, encargado de proporcionar una interfaz para usar dispositivos con el chip *RTL2382U* como radios definidas por software (únicamente con capacidad de recepción, por desgracia).

Finalmente, el concepto de *SDR* se hace popular. Nacen múltiples proyectos para crear dispositivos *SDR* de bajo coste (no sólo con el chip *RTL2382U*), incluso con capacidad de transmisión. A día de hoy, tiene su espacio en conferencias de seguridad, en forma de conferencias y talleres. Y parece haber un interés en integrar *SDR* en la comunidad *pentester*[11], como se puede ver en la conferencia *Bringing Software Defined Radio to the Penetration Testing Community* [29], en la que se presenta *Scapy-radio*, una herramienta que combina el acceso a la radiocomunicación que nos da *GNU Radio* con la capacidad de manipular paquetes de distintos protocolos de *Scapy*, permitiendo inyectar paquetes al vuelo. Precisamente lo que temía Michael Ossmann en una de sus conferencias [17]. Desde luego, suena peligroso...

[10] *Digital Audio Broadcasting +*, un estańdar de radiocomunicación de audio digital

[11] Un *pentester* lleva a cabo tests de penetración en sistemas informáticos, para comprobar su seguridad

3. Conceptos básicos

3.1. Radiofrecuencia

En esta sección se pretende dar una base mínima de radiocomunicación, de forma que se pueda entender el caso práctico (ampliando un poco más, por completitud). La intención es que sea una introducción eficiente y sencilla de entender. Michael Ossmann dio una introducción con estas características en su ya mencionada conferencia *Software Radio and the Future of Wireless Security* en *Black Hat 2008 USA* [17], precisamente para introducir lo básico de radiocomunicación para empezar a usar *SDR*. En esta sección adapto buena parte de su contenido, ampliando cuando sea necesario, intentando anticiparme a preguntas que le puedan surgir al lector[12].

¿Qué significa radiofrecuencia?

Es la forma de designar a cada una de las frecuencias de de las ondas electromagnéticas empleadas en la radiocomunicación (en el contexto del espectro electromagnético) [30].

¿Qué características tiene el espectro de radiofrecuencia?

Está compuesto por radiación electromagnética en el rango de frecuencias de 3 Hz a 300 GHz. La mayoría de las aplicaciones prácticas están entre 30 kHz y 30 GHz (ver tabla 1), con longitudes de onda asociadas de entre 10 km y 1 cm, en ese orden. Recordemos la ecuación que relaciona frecuencia (f) con longitud de onda (λ), donde c es la velocidad de la luz en el vacío[13]:

$$\lambda = \frac{c}{f}$$

Una característica importante de las ondas electromagnéticas en este rango de frecuencias es que se pueden generar aplicando corriente alterna a una antena (y una onda al pasar por una antena genera corriente alterna). La longitud de onda determina el tamaño óptimo de la antena, y a veces se hace referencia a una porción del espectro de radiofrecuencia a través de su longitud de onda mínima. Por ejemplo, la banda de 10 metros va desde los 28 MHz hasta los 29.7 MHz.

[12] Otras referencias de esta sección: [31, 32, 33, 34, 35, 36, 37, 38, 39, 40]

[13] $c = 299{,}792{,}458\ \mathrm{m\,s^{-1}}$

Tabla 1: Bandas más comunes en radiofrecuencia

Abreviatura	Frecuencia	Longitud de onda	Nombre completo
LF	30 – 300 kHz	10 – 1 km	Low frequency
MF	300 kHz – 3 MHz	1 km – 100 m	Medium frequency
HF	3 – 30 MHz	100 – 10 m	High frequency
VHF	30 – 300 MHz	10 – 1 m	Very high frequency
UHF	300 MHz – 3 GHz	1 m – 10 cm	Ultra high frequency
SHF	3 – 30 GHz	10 – 1 cm	Super high frequency

Podemos clasificar las bandas en 3 grupos, grosso modo, por sus características:

En primer lugar, las bandas en el rango de los kHz (aproximadamente *LF* y *MF*), debido a sus longitudes de onda, requieren antenas bastante grandes, y tienen un ancho de banda bastante limitado. Por poner un ejemplo, las bandas *LF* y *MF* tienen juntas un ancho de banda de casi 3 MHz, y un único canal de televisión TDT (como se opera en España, con el estándar *DVB-T* en la banda *UHF*) ocupa un ancho de banda de 8 MHz. Tienen una propagación muy buena.

En segundo lugar, las bandas en el rango de los MHz (aproximadamente *HF*, *VHF* y *UHF*) tienen unas longitudes de onda que no requieren antenas demasiado grandes. Tienen un ancho de banda razonable, y una propagación bastante buena. Por sus características tan atractivas es por lo que son las bandas más usadas en la mayoría de las aplicaciones comerciales.

Y finalmente, en el rango de los GHz (aproximadamente *SHF*) el tamaño de las antenas no es un problema (al estar las longitudes de onda en el rango de los centímetros, aunque por cuestiones de eficiencia se construyan más grandes que la longitud de onda asociada). El ancho de banda es bastante grande, pero la propagación es mala. De hecho, apenas tienen capacidad para atravesar materiales, y se suelen propagar en trayectoria óptica directa, por lo que son de corto alcance.

Más sobre antenas

En la práctica, para la mayoría de los trabajos de investigación de vulnerabilidades usando SDR, no es necesario tener una antena óptima para la banda en la que se esté trabajando. Evidentemente, perderá eficiencia, pero es asumible generalmente. En el caso de frecuencias en el rango de los kHz sí que puede ser necesario el uso de antenas especiales (de múltiples espiras). Un ejemplo típico es el de algunas etiquetas *RFID*.

3.2. Radiocomunicación

En su versión más simple, y explicando de forma abstracta, un sistema de radiocomunicación consta de un emisor y un receptor. El emisor genera una señal a cierta frecuencia (señal portadora), que es modulada y emitida. La modulación de una señal puede ser algo tan simple como activar y desactivar la generación de la señal, o cambiar algún parámetro de la señal (de forma instantánea y en cada momento), como la amplitud (*AM*), la frecuencia (*FM*) o la fase (*PM*). Cambiando parámetros de esa señal se introduce información (que será transmitida, que es el objetivo de la comunicación).

Después, el receptor, sabiendo la frecuencia de la onda portadora del emisor, genera su propia copia de la señal portadora. Y entonces compara la señal portadora que viene del receptor (que ha sido modulada y contiene información) con su copia de la portadora. El resultado de esa comparación es la información (ya fuese introducida por la activación y desactivación de la señal o por cambiar parámetros de la señal).

Más formalmente, la señal c que se emite tiene la siguiente forma:

$$c(t) = A(t)\cos(\phi(t))$$

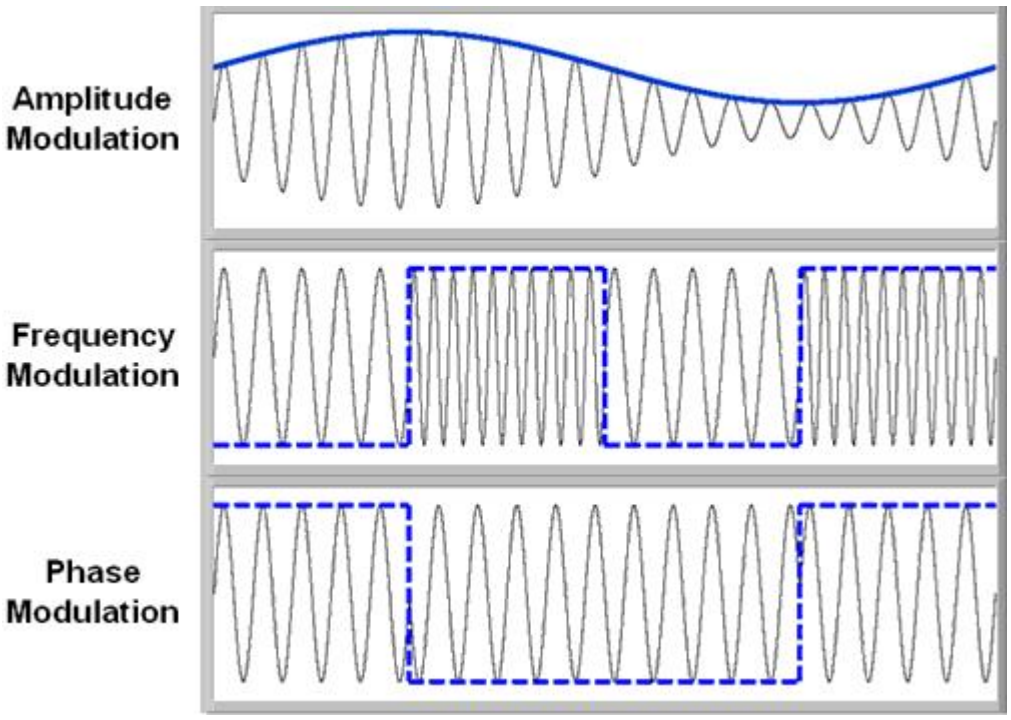
donde t es el tiempo, A la amplitud y ϕ la fase[14]. Si la amplitud, la frecuencia y la fase fuesen constantes en el tiempo, no habría modulación y la señal portadora se emitiría intacta. Si cambiamos uno de los tres parámetros (o varios), la señal producida estaría modulada. Cada tipo de modulación tiene sus características (ancho de banda, inmunidad al ruido, etcétera), además de subtipos de modulaciones según varíe el parámetro. Un ejemplo de cada tipo de modulación se puede ver en la figura 1.

Todo lo contado hasta ahora es teoría, las implementaciones reales de dispositivos de radio son bastante más complejas, ya que muchos componentes y operaciones que se realizan no tienen una implementación física directa, o no son tan ideales como en la teoría. Además de los problemas que presenta trabajar en el dominio analógico (interacciones inesperadas entre componentes, ruido, etcétera).

Por suerte, la tecnología *SDR* permite que estos componentes y operaciones que no tienen una implementación física directa puedan ser emulados en software una vez se digitaliza la señal gracias al procesamiento digital de señales. Muchos componentes tienen una implementación teórica ideal, y los que no, pueden ser mejorados hasta el grado de precisión que se necesite.

[14] Si la fase depende únicamente de la frecuencia, tenemos $\phi(t) = 2\pi t f + \theta$. La modulación por frecuencia es una particularización de la modulación por fase

Figura 1: Los tres tipos de modulación principales



3.3. Procesamiento digital de señales

En esta sección presenta algunos conceptos básicos de señales y su procesamiento. Estos conceptos están respaldados por construcciones matemáticas cuyo aprendizaje está fuera de los objetivos de este trabajo. La intención es dar una visión intuitiva del funcionamiento de la tecnología *SDR*.

Algunos conceptos básicos

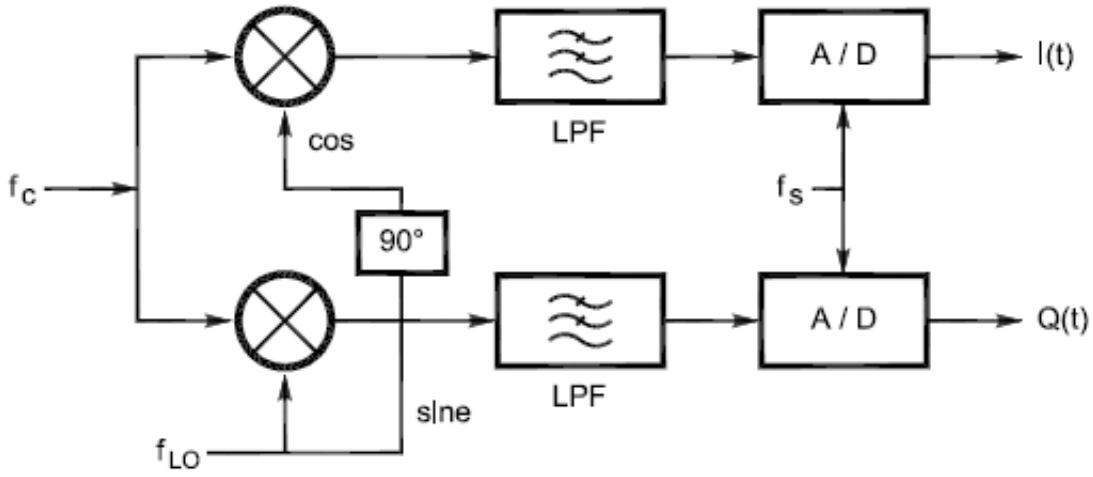
Ancho de banda
Diferencia entre la frecuencia más alta presente en una señal y la más baja.

Frecuencia de muestreo
Número de veces que se toma una muestra (medición de la amplitud de una señal en un momento dado) en un segundo.

Banda base
Banda de frecuencias situada cerca de los 0 Hz. Es la correspondiente a señales sin modular (señales a transmitir).

Por definición, una implementación *SDR* puede tener muchas formas, dependiendo de qué componentes de la radio se implementen en software. Aquí se presenta una versión simplificada, que sin entrar en detalles de cada arquitectura en particular, sirve para explicar a grandes rasgos lo que hace la implementacion *SDR* típica que se conecta a un ordenador personal:

Figura 2: Receptor de conversión directa



De izquierda a derecha:

f_{LO}
Señal proveniente de la antena, en la banda de radiofrecuencia. Idealmente es la señal que ha salido del transmisor correspondiente, por lo que es la señal portadora modulada con la señal que contiene la información.

f_c
Señal proveniente del oscilador local (que produce un coseno a la frecuencia que queremos sintonizar en la banda de radiofrecuencia). Se duplica y una de ellas se desfasa 90º respecto a la otra. Esto es característico del muestreo en cuadratura, que veremos después.

Mezclador de frecuencias
Cada uno de los elementos circulares. Produce una señal que contiene dos señales, cuyas frecuencias son $\mathbf{f_c} + \mathbf{f_{LO}}$ y $\mathbf{f_c} - \mathbf{f_{LO}}$, es decir, la suma y la diferencia de la frecuencia de las señales de entrada. Pero lo importante en este caso es saber que después de los mezcladores, las señales combinadas de ambos caminos forman una señal compleja (perteneciente a $\mathbb{C}$ en vez de a $\mathbb{R}$) situada en banda base. Básicamente, translada la señal de la banda de radiofrecuencia a la banda base.

Filtro de paso bajo
Cada uno de los elementos denominados *LPF*. Idealmente, elimina de la señal entrante toda frecuencia superior a un valor predeterminado. En este caso se usa porque queremos descartar la frecuencia superior de las dos que produce un mezclador.

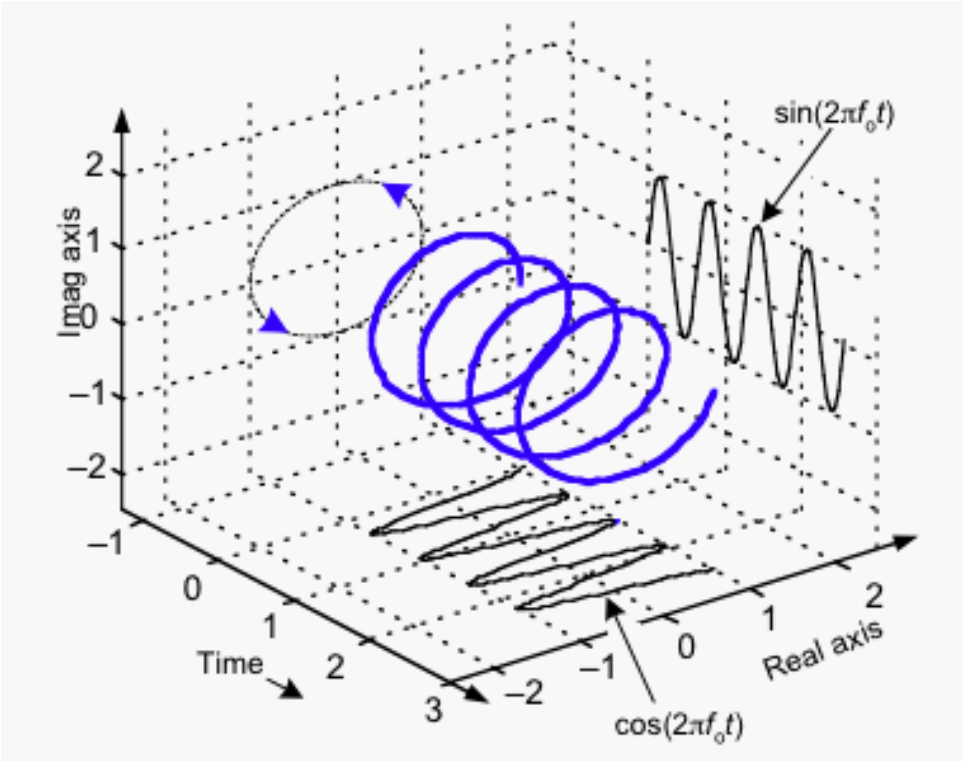
f_s
Frecuencia de muestreo ya definida anteriormente.

Convertidor analógico-digital
Cada uno de los elementos denominados *A/D* . Muestrea a intervalos regulares una señal analógica a la frecuencia **f_s**.

¿Por qué muestras complejas?

En este punto, cabe preguntarse cuál es la necesidad de adquirir muestras complejas, y qué significado tiene que una muestra sea compleja. ¿Acaso no es suficiente con que sea real? La respuesta es que es una construcción matemática que simplifica las operaciones con la señal. Como por ejemplo, detectar la envoltura de una señal *AM* (la parte azul de la señal AM en la figura 1), o la fase de una señal *PM*, como ahora veremos. ¿Pero qué aspecto tiene una señal compleja?

Figura 3: Señal compleja



Cuando transmitimos una señal $c(t) = A\cos(\phi)$ (con $\phi = 2\pi ft + \theta$), lo que transmitimos es c, pero la modulación se hacía a través de A y ϕ. El valor c no es una forma directa de ver la información contenida en la señal, que venía a través de A y ϕ. A grandes rasgos, esta forma de tomar muestras (muestreo en cuadratura), es la forma que tenemos de extraer A y ϕ cuando lo que recibimos es c.

En la figura 3 podemos ver, en el plano horizontal, una función coseno. Ese plano corresponde al eje real y al tiempo. Es lo que estamos acostumbrados a ver cuando representamos una señal. Lo que hay en el plano correspondiente al eje imaginario y al tiempo es una función seno. La función seno no es más que una función coseno con un desfase de 90º. El muestreo en cuadratura toma dos muestras reales, una muestra de la señal y otra muestra de la misma desfasada 90º.

Por tanto, al muestrear una función coseno de esta forma, lo que obtenemos son muestras de una función coseno y de una función seno. Si representamos cada pareja de muestras en el tiempo, obtenemos la hélice que vemos ahí. Para comprender lo que hace interesante esta forma de muestreo, veamos una señal *AM* muestreada de dos formas:

Figura 4: Muestreo real

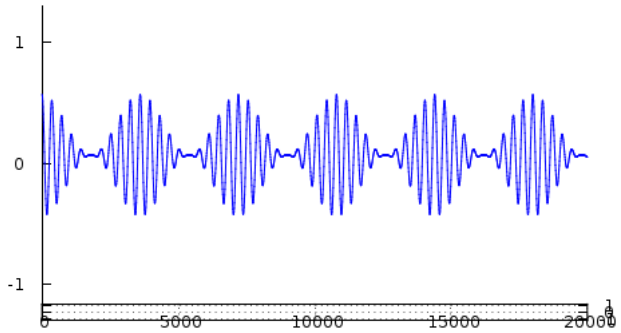
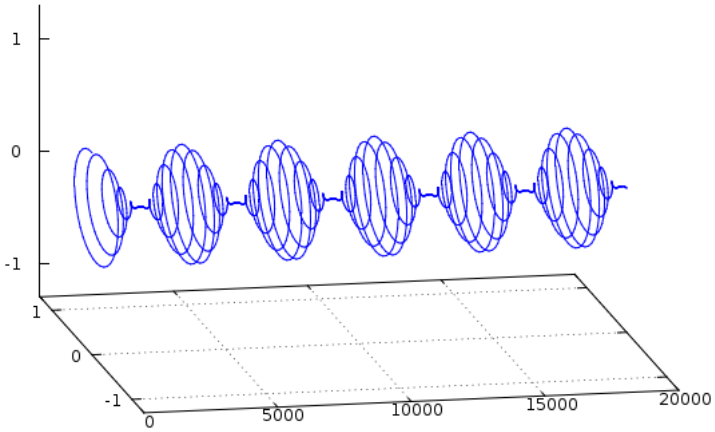


Figura 5: Muestreo en cuadratura

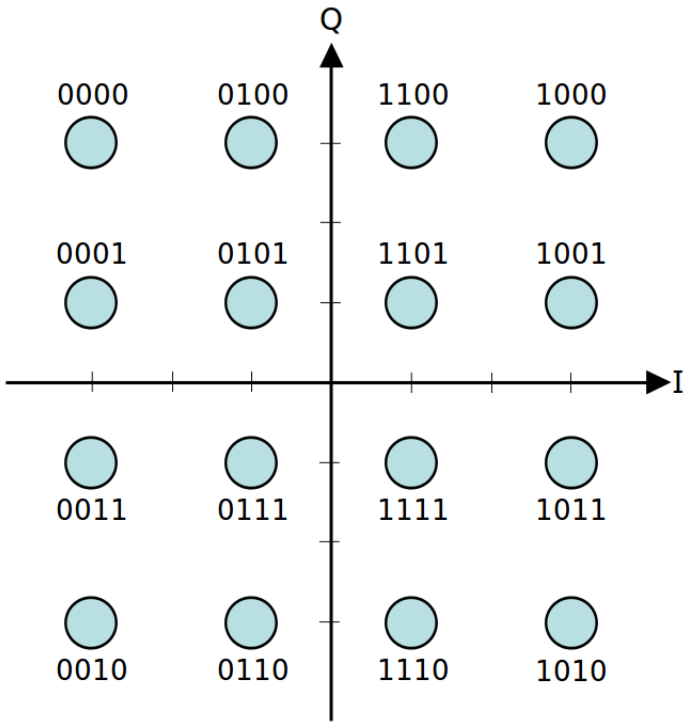


Como se puede apreciar, la figura 4 es la proyección sobre el eje real de la figura 5. Pero lo interesante es que la envoltura de la señal es igual al radio de la hélice en todo momento. Por tanto, teniendo una muestra *I/Q*, con la forma $x + iy$, la amplitud (A, no la amplitud instantánea c) es igual al módulo de $x + iy$, es decir, $\sqrt{x^2 + y^2}$. No sólo eso, también podemos obtener la fase como $\operatorname{arc\,tg}(y/x)$ (teniendo en cuenta los signos para saber el cuadrante y el ángulo correspondiente.

De manera que hemos podido extraer A y ϕ, que eran los dos argumentos para modular la señal portadora, y por tanto podemos demodular cualquier señal. Como antes comentábamos, la frecuencia, siendo la derivada de la fase, también se puede obtener a partir de la fase. En una aproximación ingenua, podríamos estimar la frecuencia instantánea de una forma sencilla mediante la diferencia de fase de dos muestras consecutivas (dada la frecuencia de muestreo, por supuesto).

Lo comentado anteriormente vale para todo tipo de señales, pero algunos tipos de modulaciones digitales son maś simples. Lo que vemos en la figura 6 es *16QAM*, una modulación digital que transmite 4 bits por muestra compleja (se muestran todos los posibles valores de **I** y **J** que puede tomar una muestra). De esta forma, no tenemos que calcular la amplitud o la fase para demodular la señal digital, basta con ver a cuál de los posibles puntos se parece más nuestra muestra.

Figura 6: Muestreo en cuadratura

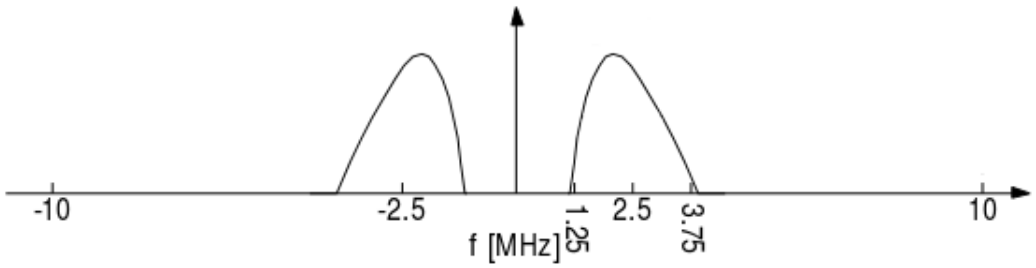


Desde la antena hasta el dominio digital

Finalmente, explicamos de forma breve los cambios que sufre una señal desde que entra por la antena hasta que se convierte en muestras digitales listas para salir de la placa hasta nuestro ordenador. Suponemos nuevamente que estamos ante un receptor de conversión directa.

La señal que entra por la antena tiene este espectro de frecuencias (por ejemplo):

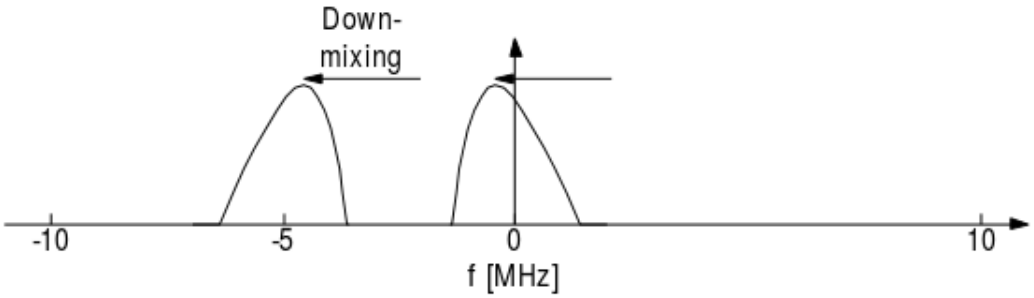
Figura 7: Señal recibida



Puede sorprender que haya frecuencias negativas, pero de hecho, toda señal real tiene frecuencias negativas, y el espectro de frecuencias es simétrico. Por ejemplo, la función $cos(x)$ y la función $cos(-x)$ tienen la misma representación, y la segunda tiene una frecuencia negativa[15].

A continuación, los mezcladores trasladan la señal a banda base:

Figura 8: Traslado a banda base

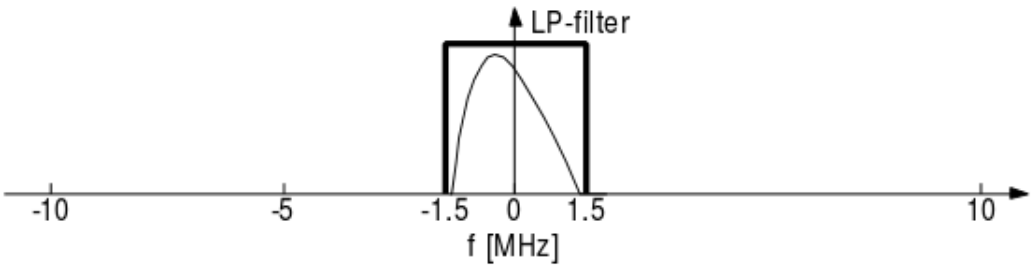


[15]Por desgracia, explicar esta cuestión formalmente no es trivial y se sale del objetivo de este trabajo. Una explicación superficial se da en la respuesta del usuario *endolith* a la pregunta *What is the physical significance of negative frequencies?* en la comunidad *Signal Processing* de *StackExchange* [41]

Teóricamente, la función que implementan los mezcladores es la multiplicación de la señal entrante por una señal sinusoidal compleja con frecuencia negativa, y el resultado es que el espectro entero se desplaza a la izquierda (por ser negativa) según la frecuencia dada (que es $\mathbf{f_{LO}}$, la del oscilador local). Ahora nuestra señal es compleja, y tiene frecuencias negativas no simétricas. Las señales complejas no tienen la restricción de simetría, y modelan individualmente las frecuencias negativas (gráficamente, la hélice gira en el sentido de las agujas del reloj, en vez de en el sentido contrario a las agujas del reloj como en las frecuencias positivas)

Es el momento de eliminar frecuencias no deseadas y ruido:

Figura 9: Aplicación de filtro de paso bajo



Un filtro de paso baja para una señal compleja no es más que dos filtros de paso baja para cada componente real. El resultado es el mismo, eliminar toda frecuencia por encima de un cierto valor, tanto en la parte negativa como en la positiva (en señales reales esto es indiferente, por la simetría).

Finalmente, la señal entra en los convertidores digital-analógico. Una interpretación simplista del teorema de Niquist-Shannon viene a decir que para las señales reales, si tenemos una señal cuya frecuencia máxima es $\mathbf{f_{MAX}}$, la frecuencia de muestreo debe ser algo superior al doble de $\mathbf{f_{MAX}}$ para que la señal se pueda capturar sin pérdida de información. En el caso de las señales complejas, basta con que la frecuencia de muestreo sea algo superior a $\mathbf{f_{MAX}}$. De forma intuitiva, una muestra compleja está compuesta por dos muestras reales independientes, por lo que contiene el doble de información que una muestra real. Por tanto, en el ejemplo de la figura 9, bastaría con una frecuencia de muestreo de 1.5 MHz.

La mayoría de las explicaciones se han orientado al proceso de recepción, pero los conceptos son los mismos y el orden de las transformaciones que sufre la señal se invierte. Por lo que no se estima necesario explicar el proceso de transmisión. Se ha hecho lo posible para, en pocas páginas, dar la introducción de conceptos básicos que permitan empezar a trabajar con implementaciones *SDR*, valorando más la explicación intuitiva de los conceptos que sus formalizaciones matemáticas.

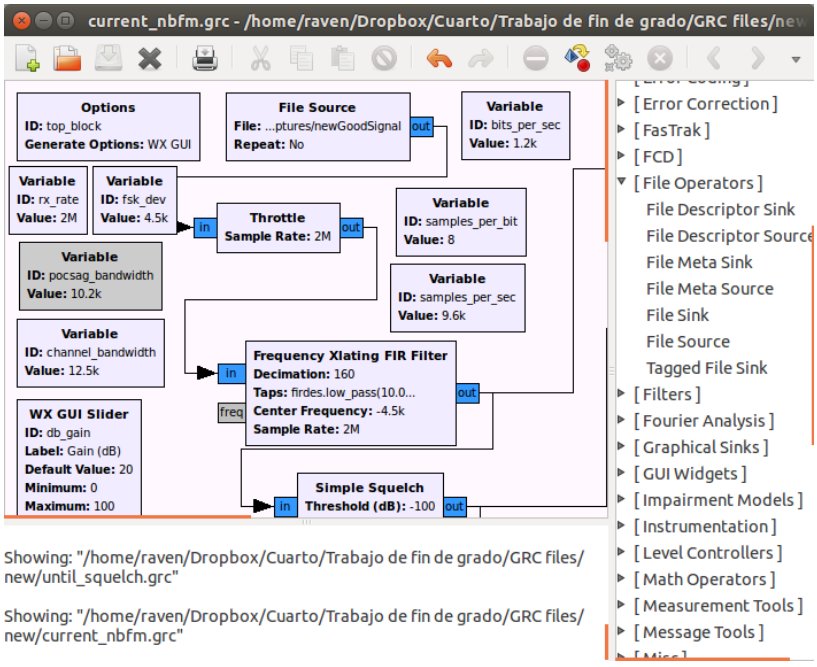
4. Introducción a GNU Radio

La página principal del proyecto *GNU Radio* lo define como un kit de desarrollo que provee bloques de procesamiento de señales para implementar radios definidas por software. Sin duda, la implementación de radios definidas por software es el principal cometido, pero en general es una herramienta de procesamiento de señales, que además tiene módulos de entrada y salida que permiten comunicarse con la parte hardware de radios definidas por software. De hecho, también hay módulos de entrada y salida para leer y escribir en disco duro, grabar y reproducir con una tarjeta de sonido, enviar y recibir a través de sockets...

Por eso quizás una mejor descripción de lo que es *GNU Radio*, es que es un entorno programable de propósito general de procesamiento de señales digitales. Nada impide usar GNU Radio para transmitir información entre dos ordenadores a través de ultrasonidos usando sus tarjetas de sonido, micrófonos y altavoces [42], por poner un ejemplo extremo. Es programable (oficialmente en C++ y en Python) y por ello se puede extender con relativa facilidad.

Hay dos formas de utilizar GNU Radio, usando directamente las librerías o a través de su interfaz gráfica *GNU Radio Companion* (cuyas siglas son *GRC*), que no es más que una abstracción gráfica de las librerías, como ahora se verá. En esta introducción nos vamos a enfocar en *GRC*. Este es el aspecto que tiene (el tamaño está reducido a lo imprescindible):

Figura 10: GNU Radio Companion



En la figura 10, además de la barra de menús (que no aparece en la figura), podemos diferenciar cuatro partes principales:

- **Diagrama de flujo:** En la parte central, está lo principal del programa. Es un área de trabajo donde se sitúan los bloques de procesamiento de señales y se efectúan las conexiones entre ellos. Básicamente es un editor de diagramas de flujo.
- **Bloques disponibles:** En la parte de la derecha, hay una lista de bloques agrupados según sus funciones. Para poner un bloque en el diagrama de flujo, basta con hacer click y arrastrar al diagrama o hacer doble click.
- **Consola:** En la parte inferior, hay una consola donde se reciben mensajes sobre la actividad del programa, ya sean actividades normales, errores, mensajes que algún bloque pueda emitir...
- **Barra de herramientas:** En la parte superior, está la barra de herramientas, que es una forma de acceder rápidamente a las funciones principales de la barra de menús, que ahora comentaremos.

En la barra de menús tenemos las opciones típicas de cualquier programa (abrir y guardar archivos, ayuda, menú de vistas para mostrar u ocultar partes del programa...). Del resto de opciones (y quitando cada opción cuya funcionalidad sea evidente), a continuación se explican las más importantes:

- **File** $\rightarrow$ **Rotate Counterclockwise** / **Rotate Clockwise:** Sirve para girar un bloque que está seleccionado, es sólo una opción visual.
- **File** $\rightarrow$ **Enable** / **Disable:** Activa o desactiva un bloque seleccionado. A efectos prácticos es como si el bloque no existiese, ni las conexiones que tiene con otros bloques.
- **Properties:** Permite configurar un bloque seleccionado. Muestra una ventana con parámetros para ese bloque.
- **Build** $\rightarrow$ **Generate** / **Execute** *Generate* compila el diagrama de flujo, y *Execute* hace lo que *Generate* y además lo ejecuta.
- **Tools** $\rightarrow$ **Filter design tools** Es una herramienta que permite diseñar filtros digitales de forma gráfica, para su uso en el programa.
- **Help** $\rightarrow$ **Types** Es una ventana donde consultar la correspondencia entre colores y tipos de datos (los tipos de datos de entrada y salida de los bloques, como *Complex Float 32*, *Integer 8*...)

Veamos qué proceso sigue *GRC* para ejecutar un diagrama de flujo. Los diagramas de flujo se guardan con la extensión *.grc*, y no son más que ficheros *XML* que especifican qué bloques hay, dónde están situados gráficamente, qué parámetros tienen y cómo se interconectan entre ellos. Cuando se le da a *Generate*, se analiza el fichero y se crea un programa *Python* que se puede ejecutar independientemente de *GRC*.

No es una traducción a un nivel semántico muy profundo, ya que los bloques vienen implementados como clases *Python* en las librerías (muchas envolviendo librerías nativas, para que el procesado de señales sea más rápido). De hecho, un programa típico (que tiene como nombre el asignado por el parámetro *ID* del bloque *Options* y extensión *.py*) se compone de una clase que implementa el diagrama de flujo (se instancian los bloques y se registran sus conexiones) y del programa principal, que instancia la clase del diagrama de flujo definida anteriormente, lo ejecuta (con un método heredado, ya que la clase del diagrama de flujo hereda de otra) y espera a que termine.

Realmente, *GRC* es una capa relativamente fina, apenas gráfica, del software *GNU Radio* en su conjunto. Dicho esto sin desmerecer su importancia, pues hace la curva de aprendizaje bastante menos pronunciada y agiliza el desarrollo. Los diagramas de flujos pueden ser jerárquicos, creando bloques que contienen a otros bloques. Esto ayuda a abstraer un proceso complejo en un bloque, que además puede ser reutilizado.

Los diagramas de flujo

Los diagramas de flujo se componen, esencialmente, de bloques, conexiones entre bloques y variables:

Las variables son globales en el diagrama, y pueden ser usadas por su nombre en lugar de constantes en los parámetros de los bloques. Su sintaxis es la misma que la de las variables *Python* (de hecho, en su implementación son variables Python), y se puede operar con ellas. Así, una variable (o un parámetro de un bloque) puede venir definida por la multiplicación de dos variables, por ejemplo, o de una variable con una constante. De nuevo, las operaciones son *Python*.

Las conexiones no tienen mucho misterio, representan de qué bloque a qué bloque van las muestras. Son inválidas si conectan bloques con tipos incompatibles, y también si conectan con una entrada ya ocupada por otra conexión (pero de una misma salida sí pueden salir varias conexiones).

Los bloques son los elementos básicos de los diagramas de flujo. Los hay de tres tipos tipos principalmente:

- **Sin entradas ni salidas:** Suelen ser bloques con parámetros de configuración (por ejemplo, el bloque *Options* los tiene, y definen ciertas opciones de la ejecución de un diagrama de flujo), o de elementos para la interfaz gráfica del diagrama de flujo (por ejemplo, un elemento deslizable ligado a una variable, para cambiar algún parámetro de bloque al vuelo).
- **Sólo con entradas o sólo con salidas:** Tambień denominados *sinks* o *sources*, respectivamente. Sirven para recibir o enviar muestras a elementos exteriores al diagrama de flujo, ya sea la parte hardware de una implementación *SDR*, una tarjeta de sonido, un archivo en disco duro, un elemento de la interfaz gráfica para representar el espectro de frecuencias...
- **Con entradas y salidas:** Generalmente, estos son los bloques encargados del procesamiento de señales en sí. Moduladores, demoduladores, operaciones aritméticas básicas, filtros, convertidores de tipos, recuperación de señales digitales, correción de errores...

Finalmente, veamos casos de uso típicos y bloques importantes que se utilizan. Si el objetivo es simplemente grabar en un archivo una banda específica del espectro (para procesar posteriormente), se usa el *source* correspondiente de un dispositivo *SDR* (en placas USRP, *UHD: USRP Source*), especificando la frecuencia central y el ancho de banda deseado, y se conecta a un *File Sink*, especificando ruta y nombre del fichero (para la transmisión es análogo).

Si en una parte del diagrama de flujo necesitas centrarte en una parte determinada del espectro de frecuencias, puedes usar un *Frequency Xlating FIR Filter*, que traslada una señal en frecuencia (según *Center Frequency*) y reduce el ancho de banda (según *Decimation*, el concepto complementario a la interpolación). También permite usar un filtro cualquiera (según *Taps*, y en los parámetros del filtro se puede especificar ganancia). Este bloque es muy potente.

Si se quiere ver el espectro de frecuencias, se puede usar *WX GUI FFT Sink*, y si se quiere ver la señal, *WX GUI Scope Sink*. También hay muchos bloques para modular y demodular, y para extraer bits de ondas cuadradas (*Clock Recovery MM*).

De cara a un uso avanzado, sería interesante mirar las funcionalidades que tiene *GNU Radio* para el procesado de paquetes, una vez se ha desmodulado y recuperado la señal digital, además del paso de mensajes entre bloques y otras características avanzadas.

5. Caso práctico: pager POCSAG

Dentro de los muchos dispositivos que usan radio para comunicarse, los *pagers* usados en restaurantes son unos dispositivos ideales para su estudio a este nivel. En un principio me parecieron lo suficientemente simples como para ser abordado su estudio satisfactoriamente según la carga estimada en un trabajo de fin de grado, teniendo en cuenta que empecé el trabajo sin tener conocimiento más que a nivel divulgativo de radiocomunicación.

El relato de la investigación y el desarrollo de este caso práctico pretende reflejar la evolución natural en el tiempo del trabajo. Si relatásemos toda la resolución del caso práctico de una vez, exponiendo los resultados sin explicar cómo llegamos a ellos o qué conocimientos se necesitaron para abordar cada pequeño problema, podemos cometer el error de trivializar el caso práctico. Así que considero necesario exponer de forma temporal cada problema y cómo se abordó su resolución. Pero antes, paso a describir el material de trabajo.

5.1. Material de trabajo

La herramienta principal es la placa *USRP B200* (*USRP* significa *Universal Software Radio Peripheral*, lo cual es bastante descriptivo). Básicamente, es un dispositivo *SDR* con las siguientes características [43] :

- Rango de frecuencias: 70 MHz - 6 GHz
- Ancho de banda máximo: 56 MHz
- Canales: 1 *RX* y 1 *TX*, *half* y *full-duplex*[16]
- Conexión al ordenador: *USB* 3.0
- Interfaz de programación: *GNU Radio* (en *C++* y *Python*)

La otra herramienta es un ordenador, un portátil con el sistema operativo *Ubuntu 14.04* y el siguiente software instalado: *GNU Radio* (versión *v3.7.4git-395-gc98699ee*) y los drivers *UHD* para la placa *USRP B200* (versión *UHD_003.007.001-78-gdc48e5ea*).

[16] *RX* significa recepción, *TX* significa transmisión, *half duplex* significa comunicación bidireccional pero no simultánea y *full duplex* significa comunicación bidireccional y simultánea

5.2. Manos a la obra

En esta sección relato de forma secuencial mi experiencia aprendiendo a usar la placa *USRP B200* y *GNU Radio* para estudiar la vulnerabilidad de los *pagers* para restaurantes que usan el protocolo *POCSAG*. Y tras estudiar la vulnerabilidad, proponer posibles ataques y defensas ante éstos.

Primera aproximación

La primera vez que trabajé con la placa fue en el laboratorio de la universidad. Cuando me dieron acceso, me presenté con mi portátil personal, en el que previamente había instalado *GNU Radio* y el *driver* para la placa *USRP B200*. En principio, iba con el objetivo de ver si apenas conseguía hacer funcionar todo. Sorprendemente, lo conseguí, así que empecé a buscar algún ejemplo hecho para empezar a experimentar. Recordé que había un vídeo en la página de *Ettus Research* en el que prometía enseñarte a construir un receptor *FM* en menos de 10 minutos [44], con lo que podría escuchar estaciones de radio comerciales. Si bien tardé algo más de los 10 que prometía, finalmente oí música salir de los altavoces.

A estas alturas, apenas tenía conocimiento teórico de radiocomunicación. Las últimas semanas había estado leyendo varios tutoriales de *National Instruments* sobre el tema [45], pero además de no enterarme muy bien, no terminaba de verle la parte práctica a lo que estaba aprendiendo. En aquel momento, intenté entender qué hacía cada parte del diagrama de flujo, pero la verdad es que no entendía muy bien por qué aquello funcionaba. Entre otras cosas, no entendía por qué se sintonizaba en el centro de la emisión, en vez de en la frecuencia más baja, y no entendía cómo podía haber frecuencias negativas. Así que pensé que lo mejor sería buscar recursos de aprendizaje más prácticos y sencillos de entender. Todavía no tenía ni idea de qué vulnerabilidad iba a estudiar.

La inspiración

Curiosamente, la inspiración me vino descansando. Un día unos amigos y yo decidimos ir a comer algo a un restaurante nuevo que habían abierto. Cuando llegamos, nos sentamos, elegimos la comida y alguien fue a pedir (era un restaurante de los que se pide en la barra). A la vuelta, ese alguien vino con un objeto circular bastante extraño. Le pregunté que qué era aquello, y me dijo que aquello iba a sonar cuando nuestra comida estuviese lista. Por curiosidad, cogí el objeto y comencé a estudiarlo.

Tenía esta pinta:

Figura 11: Objeto extraño

Pero lo más interesante vino cuando le di la vuelta:

Figura 12: Objeto extraño por detrás

Bastante información. Fabricante, modelo, un campo llamado *BASE ID* igual a 17 y otro campo llamado *FR* (probablemente significando *Frequency Range* igual a 433 - 440 MHz. Empecé a pensar en cómo funcionaría este dispositivo. La opción más simple era que cuando recibía una señal determinada, este dispositivo se activase y empezase a sonar y a vibrar. Esa señal tendría que ser emitida por otro dispositivo. Me fui a la barra, y efectivamente, estaban apilados encima de una base con una antena y un teclado incorporados (la base parecía hacer las veces de estación de carga, a través de los conectores de metal que tiene este dispositivo). No llegué a observar si el aviso estaba integrado con el resto del sistema informático o se activaba manualmente, pero tampoco parecía interesante.

La inspiración

Pensé que aquello podía ser interesante para mi trabajo de fin de grado, a ver si era un esquema vulnerable. Si conseguía capturar la señal que recibía este dispositivo, podría estudiarla. Pero antes tenía que saber más información del dispositivo. Así que cuando llegué a casa introduje en la barra del navegador la dirección que aparecía en la parte de atrás. Era la página oficial, que como suponía, no aportaba información técnica detallada, sino información para clientes potenciales. Pero al menos supe que el dispositivo se llamaba *pager*, y que no era simplemente una palabra sin sentido impresa detrás del dispositivo.

Así que busqué *pager* en *Google* y empecé a ver los distintos modelos que había. Haber había bastantes, pero la mayoría eran sospechosamente parecidos entre sí, incluido el del restaurante en el que había estado (probablemente la base fuese siempre la misma, y el distribuidor le pusiese una carcasa). Elegí uno bastante parecido al mío, y me metí en su página web. En esta página ya aparecían más especificaciones técnicas, como que usaba el protocolo *POCSAG*. Entonces pensé que probablemente ya se hubiera estudiado antes este sistema usando *SDR*, así que volví a *Google* y busqué *pager pogsac sdr*.

En la primera página, encontré el blog de Oona Räisänen, que había estudiado un *pager POCSAG* [46] muy parecido al mío. Básicamente decía que había buscado el modelo en *Google* el modelo había visto que usaba el protocolo *POCSAG* y por tanto modulación *BFSK*. Como detrás del dispositivo venía la frecuencia, usando un dispositivo con un chip *RTL2382U*, sintonizó y demodulando *FM* sacó la señal.*BFSK* usa dos frecuencias, para representar dos estados posibles, 0 y 1, por tanto, si demodulamos *FM* lo que obtenemos es una onda cuadrada. Por tanto, descifró la señal entera y vio que lo único que se enviaba la dirección del dispositivo (el protocolo *POCSAG* usa la palabra *address* para referirse a un dispositivo concreto). La pregunta es, ¿sería el pager que quería estudiar igual que éste?

La captura

Gracias a que usando *SDR* se puede grabar para procesar más tarde y sabiendo que tanto la placa *USRP B200* como la conexión *USB 2.0* que la unía con mi ordenador soportaba teóricamente ese ancho de banda, me propuse grabar el rango de 7 MHz que había entre 433 y 440 MHz. Monté un diagrama de flujo en *GNU Radio* bastante simple, que recibía muestras de la placa y a la vez que las guardaba en el disco duro, mostraba por pantalla el espectro de frecuencias. Por desgracia, mi ordenador no soportaba tanta carga, así que me limité a grabar la parte inferior de los 7 MHz, 2 MHz concretamente. Y entonces empecé a ver impulsos en una frecuencia determinada cada vez que sonaba un *pager* por ahí. Con unos cuantos impulsos grabados, me fui de allí.

La construcción

Todavía por aquel entonces seguía sin saber mucho de radiocomunicación. Así que para intentar extraer la señal, pensé que era una buena idea empezar por el receptor FM que había construido, modificándolo para mi propósito. Así que empecé a sustituir bloques y parámetros, según yo pensaba que iba a funcionar. Por supuesto no lo hizo. Y ahí es cuando empecé a estudiar en serio la teoría de la radiocomunicación. Bueno, realmente fue tras la frustración de repetidos intentos con resultados fallidos, no inmediatamente. Parece ser que no iba a poder esquivar fácilmente la teoría. A estos niveles no hay herramienta que venga configurada para que funcione directamente. Aquí me di cuenta de que iba a pasar más tiempo peleándome con la herramienta y los conceptos teóricos que enfocándome en el aspecto de la seguridad.

Finalmente, tras varias tardes repasando conceptos y probando diagramas, completé un diagrama que parecía correcto. Pero por alguna razón, el demodulador *FM* no funcionaba y sólo sacaba ruido. Cansado de reinventar la rueda, busqué si ya había una implementación para *POCSAG* en *GNU Radio*, y encontré una bastante antigua, tanto que los bloques ni siquiera existían en mi versión de *GNU Radio*. Me fijé en el bloque que usaba para demodular, se lo puse al mío, y tras retocar varios parámetros, conseguí demodular la señal. La guardé en un archivo de audio, para poder verla gráficamente en *Audacity*[17]. Efectivamente, era una onda cuadrada, y seguía el protocolo *POCSAG* [48] de la misma forma que el *pager* de Oona.

Ahora vendría una etapa en la que me tiré otras tantas tardes intentando mejorar la calidad de la onda desmodulada, antes de intentar recuperar la señal de forma automática en vez de verla en *Audacity*. Pero fue tiempo perdido, ya que realmente el problema no estaba en el proceso, sino en la señal original. De esto, claro, me di cuenta más tarde.

[17] Editor de audio *open source*

Mejor digital

Me pareció interesante, por completitud, convertir la señal digital en bits de forma automática y después detectar cuándo era una trama *POCSAG*. Para ello, utilicé un bloque que extraía los bits de la señal especificándole el número de símbolos por segundo y el número de muestras por cada símbolo. Me sorprendió que funcionase tan bien. Después añadí un bloque que si en el flujo de bits detectada la palabra clave que determinaba el comienzo de una trama *POCSAG*, marcase el primer bit que iba justo después de esa palabra clave. Finalmente, programé un bloque personalizado en *Python* que detectaba ese bit marcado, extraía el código del dispositivo y lo mostraba por pantalla. Y con este resultado, aquí di por terminado el estudio de la vulnerabilidad.

Vulnerabilidades y cómo solventarlas

Si mi estudio es correcto, este esquema es bastante vulnerable. Debido a que para el mismo *pager*, la trama que se envía siempre es la misma, es vulnerable a un ataque por repetición (es decir, capturar la trama y usarla para activar el dispositivo tantas veces quisiéramos emitiéndola repetidas veces). Es más, ni siquiera sería necesario procesar la señal, bastaría con grabar la señal y reproducirla tal cual. Una forma de evitar esto es, suponiendo que el *pager* tiene memoria (que puede ser suponer mucho) y que sólo recibe y no transmite, a través de generadores de números aleatorios sincronizados (partiendo de la misma semilla) en emisor y receptor, que se enviaría en cada trama, además del código del *pager*.

Pero no es una solución teóricamente segura, ya que capturando un número aleatorio de los emitidos y conociendo el esquema de generación, podríamos reproducir la secuencia futura de números aleatorios y nos saltaríamos por completo la medida. En cualquier caso, si programamos el dispositivo para que se bloquee cuando recibe un número aleatorio que no coincide con el esperado, se podría detectar este tipo de ataque en cuanto la estación auténtica enviase un número aleatorio atrasado (por estar desincronizados por el ataque). Finalmente, se podría provocar una denegación de servicio simplemente emitiendo ruido en esa frecuencia con una potencia superior a la de la base.

En el CD adjunto los archivos necesarios para reproducir el experimento (diagrama de flujo, muestras grabadas, módulo personalizado...), además del resultado de éste.

6. Conclusiones

Es evidente que el estudio de la seguridad informática es necesario para proteger los sistemas informatizados y evitar que un sistema vea alterado su funcionamiento normal. Desde el punto de vista económico (que es lo importante en los procesos productivos), merece la pena invertir en seguridad informática, antes que arriesgarse al mal funcionamiento de los sistemas. La seguridad no sólo protege de las malas intenciones, también protege de las buenas intenciones equivocadas, que son potencialmente peligrosas

Respecto a qué supone que la tecnología *SDR* sea más accesible y se haya popularizado, entiendo que no es más que una herramienta, que no es automática y que requiere conocimientos para usarla, y que no supone un desastre inminente el simple hecho de ser más accesible y popular. Pero es una herramienta nueva, para un canal nuevo de ataques, canal por el que no se esperaban ataques. Quizás los fabricantes de los dispositivos que usan radiocomunicación fabricados hace 5 años empezase a preocuparse por la seguridad de sus radiocomunicaciones, pero hay equipos en producción en la industria maś antiguos que muy probablemente no tuvieran en cuenta este canal de ataque, y por tanto son vulnerables.

Finalmente, el objetivo es que las radiocomunicaciones sean teóricamente seguras, y que no se confíe en que no se dispone de herramientas para atacarlas. Porque ahora tenemos las herramientas.

Referencias

[1] Stamatis Karnouskos. ”Stuxnet worm impact on industrial cyber-physical system security”. IECON 2011 - 37th Annual Conference on IEEE Industrial Electronics Society , vol., no., pp.4490,4494, 7-10 Nov. 2011

[2] Jacob Appelbaum and Laura Poitras. ”Edward Snowden Interview: The NSA and Its Willing Helpers”. DER SPIEGEL, issue 28/2013 (July 8, 2013)

[3] Jeff Tyson. "How Restaurant Pagers Work". HowStuffWorks, http://electronics.howstuffworks.com/everyday-tech/restaurant-pager1.htm

[4] Software Defined Radio Forum. ”SDRF Cognitive Radio Definitions”. Software Defined Radio Forum, http://www.sdrforum.org/pages/documentLibrary/documents/SDRF-06-R-0011-V1_0_0.pdf

[5] Maxime Dumas. ”A short history of software-defined radio (SDR) technology”. Nutaq, http://nutaq.com/en/blog/short-history-software-defined-radio-sdr-technology

[6] J. Duffy Beischel. ”Software Defined Radios - SDR”. Amateur Radio WB8NUT, http://wb8nut.com/sdr/

[7] William Lane. ”Topic 4: Software Defined Radio”. Federal Communications Commision, http://transition.fcc.gov/pshs/techtopics/techtopics4.html

[8] Space Systems Technology Group, Garland Division. ”New Research Lab Leads to Unique Radio Receiver”. Team, E-Systems (May 1985, volume 5, no. 4, page 6)

[9] Texas Instruments. "SFF SDR Development Platform". Texas Instruments Product Bulletin, http://www.ti.com/lit/ml/sprt434a/sprt434a.pdf

[10] PR Newswire. ”Lyrtech Launches New Small Form Factor SDR Development Platforms at SDR Forum Technical Conference”. PR Newswire, Nov. 14, http://www.prnewswire.com/news-releases/lyrtech-launches-new-small-form-factor-sdr-development-platforms-at-sdr-forum-technical-conference-56330167.html

[11] Systems Software Research Group. ”Security of Software Defined Radio”. Department of Computer Science, University of Illinois at Urbana-Champaign, http://srg.cs.uiuc.edu/swradio/

[12] RigReference.com. "FlexRadio SDR-1000". RigReference.com, http://rigreference.com/en/rig/4568-FlexRadio_SDR_1000

[13] eHam.net. Reviews Summary for FlexRadio SDR-1000". eHam.net, http://www.eham.net/reviews/detail/4108

[14] Alan Johnson, N4LUS. ”Microtelecom Perseus Software Defined Receiver”. NASWA Journal Columns, Equipment Reviews, May 2008, http://www.naswa.net/journal/2008/05/equip052008

[15] WB5RVZ. ”Softrock Kits - an Overview”. WB5RVZ Software Defined Radio Homepage, http://wb5rvz.com/sdr/

[16] Max Moser and Philipp Schrödel. ”27Mhz Wireless Keyboard Analysis Report”. Dreamlab Technologies AG & Remote-exploit.org , https://www.dreamlab.net/en/files/2012/06/Whitepaper-27_Mhz_keyboard_insecurities.pdf

[17] Michael Ossmann. ”Software Radio and the Future of Wireless Security”. Black Hat 2008 USA, slides (clean version), http://www.ahzf.de/itstuff/conferences/BH_US_08_Ossmann_Software_Radio.pdf

[18] Zack Anderson. ”Boston Subway Security Analysis”. Zack Anderson personal website at MIT, http://web.mit.edu/zacka/www/mbta.html

[19] Wikileaks. ”Anatomy of a Subway Hack 2008”. Wikileaks, https://wikileaks.org/wiki/Anatomy_of_a_Subway_Hack_2008

[20] GNU Radio. ”Overview”. GNU Radio website, https://gnuradio.org/redmine/projects/gnuradio

[21] Ettus Research. ”About Ettus Research”. Ettus Research website, http://www.ettus.com/

[22] Yifei Liu, Timo Kasper, Kerstin Lemke-Rust, and Christof Paar. ”E-Passport: Cracking Basic Access Control Keys”. On the Move to Meaningful Internet Systems 2007: CoopIS, DOA, ODBASE, GADA, and IS Lecture Notes in Computer Science Volume 4804, 2007, pp 1531-1547

[23] Tech-FAQ. ”Known Plaintext Attack”. Tech-FAQ, http://www.tech-faq.com/known-plaintext-attack.html

[24] Nate Lawson. ”Highway to Hell: Hacking Toll Systems”. Black Hat Usa 2008, http://www.root.org/talks/BH2008_HackingTollSystems.pdf

[25] Karsten Nohl. ”Subverting the security base of GSM”. Karsten Nohl personal website at University of Virginia, http://www.cs.virginia.edu/~kn5f/pdf/090815.HAR.A51.Cracking.pdf

[26] Chris Paget. ”Extreme-range RFID tracking”. Black Hat USA 2010, http://www.tombom.co.uk/extreme_rfid.pdf

[27] David Pérez and José Pico. ”A practical attack against GPRS/EDGE/UMTS/HSPA mobile data communications”. Black Hat DC 2011, https://media.blackhat.com/bh-dc-11/Perez-Pico/BlackHat_DC_2011_Perez-Pico_Mobile_Attacks-wp.pdf

[28] rtlsdr.org wiki. ”History and Discovery of RTLSDR”. rtlsdr.org wiki, http://rtlsdr.org/#history_and_discovery_of_rtlsdr

[29] Jean-Michel Picod, Arnaud Lebrun and Jonathan-Christofer Demay. ”Bringing Software Defined Radio to the Penetration Testing Community”. Black Hat USA 2014, https://www.blackhat.com/docs/us-14/materials/us-14-Picod-Bringing-Software-Defined-Radio-To-The-Penetration-Testing-Community-WP.pdf

[30] Real Academia Española. ”Radiofrecuencia”. rae.es, http://lema.rae.es/drae/?val=radiofrecuencia

[31] David, AJ4TF. ”Inexpensive Software Defined Radio”. Greensboro Amateur Radio Association, http://www.w4gso.org/pdffiles/InexpensiveSDR.pdf

[32] Mikael Q Kuisma. ”I/Q Data for Dummies”. Whiteboard Web, http://whiteboard.ping.se/SDR/IQ

[33] National Instruments. ”What is I/Q Data?”. National Instruments website, hrefhttp://www.ni.com/white-paper/4805/en/http://www.ni.com/white-paper/4805/en/

[34] National Instruments. ”Frequency Modulation (FM)”. National Instruments website, hrefhttp://www.ni.com/white-paper/3361/en/http://www.ni.com/white-paper/3361/en/

[35] Johan Kirkhorn. ”Introduction to IQ-demodulation of RF-data”. IFBT, NTNU, http://folk.ntnu.no/htorp/Undervisning/TTK10/IQdemodulation.pdf

[36] Doug Smith, KF6DX/7. ”Signals, Samples and Stuff: A DSP Tutorial - Part 1”. QEX, March 1998, pp. 3-16, http://www.arrl.org/files/file/Technology/tis/info/pdf/98qex003.pdf

[37] Richard Lyons. ”Quadrature Signals: Complex, But Not Complicated”. dspGuru, http://www.dspguru.com/sites/dspguru/files/QuadSignals.pdf

[38] Michael Ossmann. ”Software Defined Radio with HackRF”. Great Scott Gadgets, https://greatscottgadgets.com/sdr/

[39] W2AEW. ”Basics of IQ Signals and IQ modulation & demodulation - A tutorial”. YouTube, https://www.youtube.com/watch?v=h_7d-m1ehoY

[40] W2AEW. ”IQ Signals Part II: AM and FM phasor diagrams, SSB phasing method”. YouTube, https://www.youtube.com/watch?v=5GGD99Qi1PA

[41] endolith. ”What is the physical significance of negative frequencies?”. Signal Processing, in StackExchange, post 449,http://dsp.stackexchange.com/questions/431/what-is-the-physical-significance-of-negative-frequencies

[42] Anfractuosity. ”Ultrasound Networking”. Anfractuosity, https://www.anfractuosity.com/projects/ultrasound-networking/

[43] Ettus Research. ”USRP B200/B210 Specification Sheet”. Ettus Research, http://www.ettus.com/content/files/kb/b200-b210_spec_sheet.pdf

[44] Ettus Research. ”How To Build an FM Receiver with the USRP in Less Than 10 Minutes”. YouTube, https://www.youtube.com/watch?v=KWeY2yqwVA0

[45] National Instruments. ”National Instruments Measurement Fundamentals Series”. National Instruments website, http://www.ni.com/white-paper/4523/en/

[46] Oona Räisänen. ”The burger pager” absorptions, http://www.windytan.com/2013/09/the-burger-pager.html

[47] smunaut. ”osmo-pocsag”. GitHub, https://github.com/smunaut/osmo-pocsag

[48] Adam Hickerson. ”The POCSAG paging protocol”. Raveon Technologies website, http://www.raveon.com/pdfiles/AN142%28POCSAG%29.pdf

Elementos gráficos

Tabla 1: http://en.wikipedia.org/wiki/Radio_frequency, 2014/11/24 05:01 CET

Figura 1: http://www.ni.com/white-paper/4805/en/

Figura 2: http://srv2.flex-radio.com/Products.aspx?topic=faq

Figura 3: http://www.eetimes.com/document.asp?doc_id=1275580

Figuras 4 y 5: http://whiteboard.ping.se/SDR/IQ

Figura 6: http://en.wikipedia.org/wiki/Quadrature_amplitude_modulation#Rectangular_QAM

Figuras 7, 8 y 9: http://folk.ntnu.no/htorp/Undervisning/TTK10/IQdemodulation.pdf

Figuras 10, 11 y 12: De elaboración propia.