ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA Y SISTEMAS DE TELECOMUNICACIÓN

PROYECTO FIN DE GRADO

TÍTULO: Diseño de una estación remota de comunicaciones basada en radio software

AUTOR: Guillermo Bartolomé Herrador

TITULACIÓN: Grado en Ingeniería de Sistemas de Telecomunicaciones

TUTOR: César Briso Rodríguez

DEPARTAMENTO: Teoría de la Señal y Comunicaciones

V°B°

Miembros del Tribunal Calificador:

PRESIDENTE: Pedro José Lobo

TUTOR: César Briso Rodríguez

SECRETARIO: Juan Moreno García-Loygorri

Fecha de lectura:

Calificación:

El Secretario,

Agradecimientos

Este proyecto fin de grado ha supuesto el último sprint en una carrera de fondo muy larga, pero que muy larga, por consiguiente quiero agradecer a las personas que me han brindado ayuda y ánimos a lo largo de ella. En primer lugar quiero agradecer a mi tutor, César Briso Rodríguez, por su apoyo incondicional así como por darme la oportunidad de hacer un proyecto fin de grado en la tecnología SDR que tanto me apasiona. A mis compañeros de universidad que entre tanto sufrimiento siempre fuimos capaces de sacarnos una sonrisa y una carcajada, en especial a Charlie por ser mi ”fiel esposo” y mi compañero oficial de prácticas y exámenes. A mis amigos del Erasmus, me ofrecieron la posibilidad de ser más humano y tener experiencias que nunca llegaría ni a imaginar, en la ingeniería falta mucha humanidad y estas personas me la han dado. Nadie se creería que haría un Erasmus en Estambul y sobreviviría para contarlo. Gracias a mi gran amigo Luis Buendía, donde el interés por la ciencia hizo que floreciera una amistad tan bonita como una rosa en el desierto y por ser mi fiel profesor de informática y programación. Gracias a mi madre y a mi hermana por ser mis ángeles de la guarda, por secarme las lágrimas de las mejillas con besos y enseñarme la lección que tengo grabada en la piel, ¡lucha por tus sueños!, gracias mamá, gracias Ali ... Os querré por toda la eternidad. Por último agradecerme a mi mismo esta fuerza de voluntad que tengo, el de 9 a 9 en la universidad,”¡hoy cerramos la biblioteca!”. Gracias por hacerme evolucionar, como entré siendo un niño ”bueno” en matemáticas y me convertí en una persona que luchará tanto por resolver una integral como por un mundo más accesible para todos.

Resumen

La realización de este proyecto consiste en el diseño de una estación remota de comunicaciones basada en la tecnología SDR y Raspberry Pi como ordenador de procesado de la señal. Para ello, se analiza la teoría que soporta las radios definidas por software. Se explicarán en detalle los elementos de hardware y el tipo de software que se puede usar para el diseño e implementación de esta clase de radios, basada en software.

En relación con los elementos hardware, se requiere una evaluación previa encaminada a ver su idoneidad. Para ello, se testearán los elementos hardware con dos modalidades de software que permitirán elegir cual es el mejor para un diseño rápido y eficiente.

La implementación se desarrolla en un entorno de programación visual llamado GRC (GNU Radio Companion), cuya flexibilidad permite el diseño por medio de bloques de procesado, lo que permite el diseño por partes independientes. En otras palabras, se realizará la comunicación radio con la SDR y el ordenador, los protocolos de comunicaciones entre ordenadores. Se continuará insertando la SDR y la Raspberry Pi, modificando el diseño de la estación remota de comunicaciones para que pueda trabajar con la Raspberry Pi, la cual actúa como ordenador de procesado, por lo que precisará de algunas adaptaciones para su correcto funcionamiento.

Por último, se hará una evaluación de funcionamiento, consiste en aplicar un conjunto de mediciones para evaluar la estación remota de comunicaciones. Para ello se aplicarán las siguientes medidas: Medidas PMR (Private Movile Radio) 446, que explicará los niveles de potencia y espectro, así como la distancia conseguida. Por otra parte se realizarán medidas de red y protocolos. Por último medidas de buffer, tasas de procesado y análisis de fiabilidad de la estación remota. Se mencionará problemáticas encontradas en la etapa de diseño e implementación, proponiendo posibles soluciones y mejoras como objeto de futuras líneas de trabajo.

Abstract

The accomplishment of this project consists of the design of a remote station of communications based on the technology SDR and Raspberry Pi as signal processing computer. For it, we will study the theory of Software Defined Radio. We explain the hardware elements and software that they could use in the creation of the station of communications.

The hardware elements will be tested by two software that allow to choose which one is the most appropriate to make the design quickly and efficiently.

The implementation is carried out through a visual programming environment called GNU Radio Companion whose flexibility allows the design through processing blocks. The design will be performed by independent parts. In other words, radio communication will be carried out with the SDR and the computer, communication protocols between computers. The SDR and the Raspberry Pi will continue to be inserted, modifying the design of the remote communication station in accordance with working with the Raspberry Pi as a processing computer that will require some adaptations for its operation.

Finally there will be done a set of measurements that evaluate the remote station of communications. The measures to realize they are the following ones. Measures PMR 446 that explains levels of power and spectrum and distance. Measures of networking and protocols .Measures of Buffer, rates and efficiency ofthe remote station of communications. It will be mentioned problematic found in the stage of design and implementation, proposing possible solutions and improvements as object of future lines.

Índice

Lista de acrónimos

ADC Analog to Digital Converter

BB Banda Base

COFDM Coded Orthogonal Frequency Division Multiplexing

CPU Central Processing Unit

CTCSS Continuous Tone-Coded Squelch System

DAC Digital to Analog Converter

DCS Digital Code Squelch

DDC Digital Down Conversion

DNS Domain Name System

DSI Display Serial Interface

DSP Digital Signal Processor

DUC Digital Up Conversion

DVB-T Digital Video Broadcasting – Terrestrial

ERP Effective Radiated Power

FI Frecuencia Intermedia

FM Frecuencia Modulada

FPGA Field-Programmable Gate Array

GPIO General Purpose Input/Output

GPS Global Positioning System

GPU Graphics Processing Unit

GRC GNU Radio Companion

GSM Global System for Mobile communications

GUI Graphical User Interface

HDMI High-Definition Multimedia Interface

IEEE Institute of Electrical and Electronics Engineers

IP Internet Protocol

JTAG Joint Test Action Group

LTE Long Term Evolution

MCX Micro Coaxial Connector

NCS Network-based Call Signaling

NFS Network File System

OS Operating System

PC Personal Computer

PMR Private Movile Radio

PPM: Partes Por Millón

PTT Push To Talk

RAM Random Access Memory

RF Radio Frecuencia

RPC Remote Procedure Call

SD Secure Digital

SDR Software Defined Radio

SDRAM Synchronous Dynamic Random-Access Memory

SNMP Simple Network Management Protocol

SPS Samples Per Second

SoC System on a Chip

TCP Transmission Control Protocol

TFTP Trivial File Transfer Protocol

UDP User Datagram Protocol

USB Universal Serial Bus

Índice de figuras

Índice de tablas

1 Introducción

Este proyecto fin de grado tiene como objetivo diseñar una estación remota de comunicaciones radio basada en radio definida por software y un procesador tipo Raspberry pi como ordenador de procesado de la señal. Supondrá un ahorro económico en hardware y brindará versatilidad a la hora de diseñar la estación de comunicaciones mediante software. A la hora de diseñar de menos a más mediante software en vez de usar hardware puede ser beneficioso antes que ir añadiendo circuitos analógicos extras. Desde otros puntos de vista, como el ambiental por ejemplo, se puede ver que los residuos hardware se verán reducidos.

El emisor transmitiría una señal de audio utilizando un dispositivo PMR446 popularmente conocido como walkie talkie. Su mensaje podría ser uno como: "Hola soy Ana, estoy perdida en la montaña de Abantos", dicha señal se recibiría por el dispositivo SDR (Software Defined Radio) y se procesaría por la Raspberry Pi. Se retransmitiría haciendo uso de un protocolo de la capa de transporte a un portátil personal usado por Guillermo, quien escucharía el audio transmitido por Ana. Tras ser escuchado el audio, Guillermo contestaría a Ana haciendo uso de un fichero . wav de modo reiterado. Su mensaje sería el siguiente: " Soy Guillermo del equipo de emergencias, tranquila Ana, iremos a por ti, a la Montaña Abantos". Dicho audio se transmitiría del portátil personal por medio de un protocolo de la capa de transporte al Raspberry Pi, que haciendo uso de un software, filtrará y modulará la señal a transmitir. La SDR radiará la señal PMR 446 con el audio modulado en FM (Frecuencia Modulada) de banda estrecha. Ana recibirá el mensaje de respuesta con su transceptor PMR 446. Por lo tanto las dos tecnologías a estudiar y desarrollar son las SDR y la Raspberry Pi.

La SDR o radio definida por software supone una revolución en las tecnologías radio debido a la facilidad de poder trabajar con diferentes estándares y comunicaciones radio. Esto es debido a que los componentes hardware pasarán a ser software, permitirá rediseñar de una manera rápida e intuitiva los componentes que forman el sistema de comunicación radio como son: moduladores, filtros, mezcladores etc. En el transmisor la señal se genera en BB (Banda Base), se muestrea y se convierte a FI (Frecuencia Intermedia) transformándose a una señal analógica para posteriormente transformarla hasta RF (Radio Frecuencia). En el receptor se realizará el proceso contrario, donde se convierte la señal de RF a FI, se digitaliza para posteriormente procesarla en BB.

De la misma manera que esta tecnología ha supuesto una revolución en el ámbito de las telecomunicaciones, la Raspberry pi la supuso en el ámbito de la informática. La Raspberry Pi es un microordenador de bajo coste con un precio medio de 30 euros. Surgió de la idea de enseñar informática y computación en los colegios públicos de Inglaterra, gracias a su precio económico cualquier niño podría acceder a este hardware. Por medio de un monitor, teclado y un ratón se tendrá un ordenador completo que brindará la posibilidad tanto de programar códigos básicos en Python, como de un servidor web TCP.

Por todo lo anterior, el proyecto fin de grado se organiza como sigue.

En primer lugar, se estudiarán los aspectos teóricos en la sección 2 que involucran la tecnología SDR, así como sus antecedentes históricos más destacados. Con posterioridad se estudiará los modelos hardware a usar. Se continuará con una explicación de los posible software a usar para el diseño de la estación remota de comunicaciones. Se usará GNU Radio como software a desarrollar gracias al apartado de rendimientos de software. Permitirá mostrar por que GNU Radio es el software más rápido y versátil a la hora de desarrollar sistemas de comunicaciones inalámbricas.

En la secciones 3 y 4 se realizará el diseño de la estación remota de comunicaciones por partes. Primero realizaremos una descripción general de los conceptos teóricos que involucran la estación remota de comunicaciones. Divididos en FM, PMR 446 y UDP/TCP. Se explicará el hardware que la conforma. La sección 4, Desarrollo Software, explicará los bloques de procesado de GNU Radio que permitirán diseñar la estación remota de comunicaciones. Primero se realizarán los diseños PMR 446 en recepción y PMR 446 en transmisión en el portátil personal. Después se realizará la integración electro mecánica de la Raspberry Pi 3 B + como ordenador de procesado de la señal PMR 446. Se deberá entender que aunque la Raspberry pi sea una revolución no llega a las prestaciones de procesado de un portátil personal. Se terminará esta sección explicando como poder trabajar de manera remota con Raspberry Pi 3 B +.

En la sección 5 se terminará realizando mediciones del diseño, las mediciones permitirá evaluar y optimizar la estación remota de comunicaciones. Las medidas a realizar son las siguientes: medidas PMR 446, medidas de protocolos y red y medidas de rendimiento.

Por último se expondrán las principales dificultades encontradas en el proceso de diseño de la estación remota de comunicaciones. Se mencionará otras posibles aplicaciones para la estación remota de comunicaciones así como posibles mejoras que podrían ser implementadas como líneas futuras.

2 Conceptos de Radio definida por Software SDR

En este apartado se explica la definición de las radios definidas por software, su contexto histórico, arquitectura y limitaciones. En subsecciones posteriores se estudiarán los modelos Hardware más comunes así como qué Software se puede usar para desarrollar aplicaciones en RF.

Las SDR o radios definidas por software son sistemas de radiocomunicaciones donde los componentes implementados comúnmente en hardware como pueden ser amplificadores, filtros, moduladores y mezcladores se realizan en software a través de un ordenador. Una definición más específica fue la dada Joseph Mitola quien uso el término software radio como: Una radio de software es una radio cuyas formas de onda moduladas son definidas en software. Es decir las formas de onda son generadas como señales digitales, convertidas de DAC (Digital to Analog Converter) vía de banda ancha digital al convertidor analógico y posiblemente convertidas de FI a RF. El receptor, similarmente, emplea ADC (Analog to Digital Converter) de banda ancha capturando todos los canales de la radio definida por software. El receptor entonces extrae, decrementa en muestras, demodula la forma de onda de canal que usa el software sobre un procesador de propósito general. Las radio de software emplean una combinación de las técnicas que incluyen la multicinta antenas y conversión de RF; ADC y DCA de banda ancha ; y la puesta en práctica de FI, banda de base y bitstream funciones que procesan de procesadores programables. En parte se extiende la evolución de hardware programable debido a su flexibilidad creciente permitiendo su programibilidad.[1]. En conclusión las SDR proporciona un entorno más flexible y eficiente debido a que modificando o sustituyendo su software, o bien añadiendo otros nuevos se consigue cambiar la funcionalidad del sistema radio. Este hecho permite especializar de una manera rápida y sencilla la SDR para cada usuario según sus necesidades. Entendiendo el concepto de SDR se continua explicando su marco histórico.

Durante la evolución de las telecomunicaciones han surgido distintas tecnologías para el intercambio de información entre puntos distantes. Las propias diferencias tecnológicas crearon incompatibilidades entre los estándares y los equipos relacionados. La SDR surge para solucionar estos inconvenientes de compatibilidad definiendo un conjunto de procedimientos y técnicas orientadas a realizar el procesado de señales radio por medio de un dispositivo de propósito general. La primera implementación fue el proyecto militar estadounidense SpeakEasy en 1991. El objetivo del proyecto [2] era implementar en un mismo equipo programable las diez tecnologías inalámbricas más usadas por las Fuerzas Armadas de Estados Unidos. El equipo trabajaría en la banda de frecuencia desde 2 MHz a 2 GHz, permitiría obtener una arquitectura reconfigurable, en software abierto, con varias comunicaciones inalámbricas simultaneas. El proyecto se dio por finalizado cuatro años después cuando se llegó a la producción de un dispositivo que trabajaba en el rango de frecuencia de 4 MHZ a 400 MHz.

A partir de esta tecnología y con la finalidad de desarrollar aplicaciones tanto en el ámbito civil como militar de los sistemas SDR, se funda en 1996 el Wireless Innovation Forum. Este organismo era un foro de colaboración entre empresas dedicado a mediar por el uso del espectro radioeléctrico y el avance de las tecnologías radio que soportan las comunicaciones necesarias en todo el mundo.

Otro hito destacable en la historia de las SDR fue la creación en el año 2001 del Proyecto GNU Radio. GNU Radio comenzó como una bifurcación del código Pspectra que fue desarrollado por el proyecto SpectrumWare en el Massachusetts Institute of Technology (MIT). En 2004 se completó una reescritura de GNU Radio, por lo que hoy en día GNU Radio ya no tiene ningún código original del Pspectra. Matt Ettus se unió al proyecto como uno de los primeros desarrolladores y además creó el Universal Software Radio Peripheral (USRP) para proporcionar una plataforma hardware para usar con el software GNU Radio.

El proyecto fin de grado se realizará en este entorno, software de desarrollo de herramientas de código

libre y abierto, que proporciona bloques de procesado de señales para implementar distintas comunicaciones. GNU Radio puede ser utilizado con hardware de bajo presupuesto para crear comunicaciones radio o sin hardware, utilizándolo como entorno de simulación[3].

En los próximos párrafos se explicará la arquitectura SDR (véase Figura 1) para implementar transmisores y receptores explicando la configuración básica en tres partes diferenciadas. Según [4] la configuración básica, se muestra en la siguiente figura, se compone de tres bloques funcionales: la sección de RF, la sección de FI y la sección de BB.

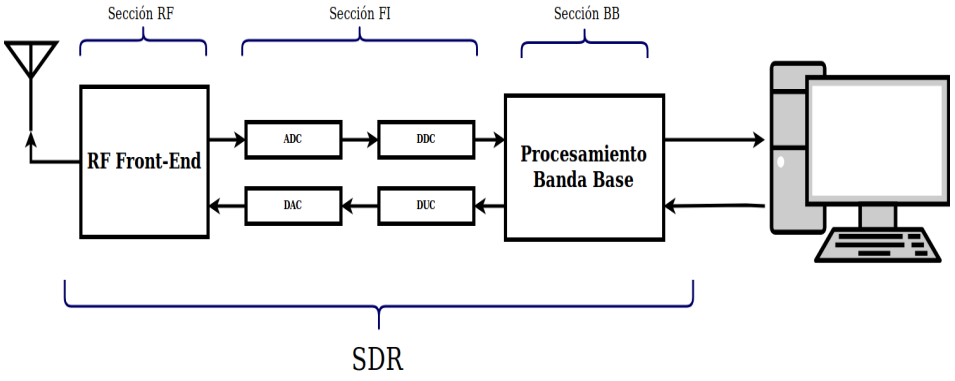

Figura 1: Arquitectura SDR

En primer lugar la sección RF, denominada RF Front-End, es la parte analógica de la arquitectura. Su función es recibir y/o transmitir las señales de radiofrecuencia para adecuarlas y convertirlas a FI en recepción o amplificar o bien modularlas en el caso de transmisión.

La sección de FI se encargará de transformar la señal a BB y digitalizarla en recepción,o bien realizar la función inversa, transformar la señal de BB a FI y hacer la conversión digital-analógica de la señal en el caso de la transmisión. Los encargados de la conversión digital-analógica o analógica-digital de la señal son los módulos DAC/ADC. La idea de este arquitectura es que estas etapas de conversión se encuentren lo más próximo a la antena. Por otra parte los módulos DDC/DUC se insertan para poder bajar o subir, respectivamente, la tasas de muestreo en el sentido de recepción o transmisión. Los módulos comúnmente se implementan en la tecnología FPGA (Field-Programmable Gate Array). La función del DDC (Digital Down Conversion) es convertir una señal digital de FI a una señal digital BB a una tasa de muestreo más baja. Por otro lado la función del DUC (Digital Up Conversion) es convertir las muestras de BB digital en muestras de FI digitales. El proceso conserva toda la información de la señal original menos la que se pierde debido a errores de redondeo en los procesos matemáticos.

Por último la sección BB es la encargada extraer la información deseada por medio del procesado digital de la señal realizando labores de filtrado, interpolación, amplificación, etc. Se destaca que se puede extraer varias señales que estén contenidas dentro del espectro recibido permitiendo la recepción simultánea de distintas señales.

Conociendo su compleja arquitectura se puede destacar las siguientes limitaciones:

•Tasas binarias: Se valorará la limitaciones de las velocidades de transmisión,entre varios Kbps hasta algún Mbps. Se tendrá en cuenta que no todos los bits enviados a través del bus son de datos, afectando directamente al régimen binario total de la SDR.En las futuras secciones se ejemplificará dicho problema y como es necesaria un buen flujo binario para un correcto funcionamiento del sistema a desarrollar.

•Procesamiento: Otra limitación reside en la dificultad que presenta el procesado en tiempo real de los sistemas radio con SDR y ordenadores, tanto para la precisión temporal de la señal, como para el número de operaciones realizadas por el ordenador.

2.1 Hardware

En la actualidad hay varios fabricantes que desarrollan hardware para la tecnología SDR, se destacan los siguientes:

•Ettus Research: Es el creador del Universal Software Radio Peripheral.

•National Instruments: Es el creador del NI-Universal Software Radio Peripheral.

•Analog Devices: Es el creador del Adalm-Pluto SDR.

•Realtek: Creador del RTL2832, un demodulador DVB-T (Digital Video Broadcasting – Terrestrial) DVB-T COFDM (Coded Orthogonal Frequency Division Multiplexing) que puede ser utilizado como un receptor SDR.

A continuación se presentan las dos radios definidas por software usadas en este proyecto: SDR-RTL y ADALM-PLUTO SDR además de explicar la gama de USRP vigentes en la actualidad, siendo los periféricos radio más usado actualmente en el ámbito profesional. Se detallará más exhaustivamente el modelo USRP B200mini-i.

2.1.1 SDR-RTL

SDR-RTL es un periférico radio USB muy económico. Sólo se puede utilizar como receptor radio, basado en software para recibir señales de radio en su área sin necesidad de utilizar internet. Según el modelo se sintonizarán señales radio en el margen de frecuencias de 25 MHz hasta 1,75 GHz. La mayoría del software para RTL-SDR es gratuito y se ha desarrollado en comunidad.

El descubrimiento y desarrollo del SDR-RTL se le atribuye al finlandés Antti Palosaari, quien colaboraba en el proyecto Linux TV escribía código para que el kernel GNU/Linux diera soporte a dispositivos multimedia. En el año 2012, mientras trabajaba con un sintonizador de vídeo digital Ezcap EzTV 668 DVB-T/FM/DAB, descubrió que el chip Realtek RTL2832U de su interior tenía funciones no documentadas. Podía llevarlo a un modo de funcionamiento donde las muestras I/Q en BB se transferirían sin procesar por el puerto USB hacia el PC. El flujo de muestras I/Q procedentes del Realtek RTL2832U era posible procesarlo por medio de un ordenador y algún software. Este modo de funcionamiento del RTL2832U es

el que permitiría brindar la posibilidad de demodular señales como la AM,FM y DAB de manera digital a diferencia de DVB-T que se demodulada por hardware.

El SDR-RTL utilizado en este proyecto fin de grado es el modelo de NooElec NESDR Mini 2 SDR DVB-T USB Stick (RTL2832 + R820T2) además de contar con el periférico radio la compra incluye una antena y un control remoto. La empresa que desarrolló este SDR-RTL es NooElec INC, se especializa en I+D, diseño de circuitos y fabricación de productos electrónicos para fabricantes de equipos originales o usuarios finales. NooElec surgió en 2007, tiene oficinas tanto en Estados Unidos como en Canadá, su característica más destacada es que desarrolla productos a precios extremadamente competitivos. A continuación se muestran sus productos más relevantes y su coste.

•HackRF One Software Defined Radio 329.76 $

•Nooelec NESDR Nano 3: Tiny RTL-SDR USB Set w/ 0.5PPM TCXO, SMA Input, Aluminum Enclosure and Custom Heatsink 32.95 $

Tras conocer los orígenes de la SDR-RTL se pasará a hablar de las las especificaciones técnicas del modelo NESDR Mini 2 SDR DVB-T USB Stick RTL2832 + R820T2 teniendo los siguientes parámetros eléctricos:

•Conversor ADC: 8 bits de resolución de la señal I/Q en BB pero el número efectivo de bits se estima en 7.

•Frecuencia de muestreo : La frecuencia máxima de muestreo es de 3.2 Mbps. Esta frecuencia de muestreo es inestable y puede funcionar de manera errónea. La frecuencia máxima de muestreo que no elimina muestras es de 2.4 Mbps.

• Rendimiento y relojes

  •Cristal interno trabaja a una frecuencia de 28,8 MHz. En la práctica es posible usar hasta un 80% del ancho de banda.

  •Precisión en frecuencia: $\pm$ 1 PPM (Partes Por Millón)

•Nivel de cuantización: 43 dB

•Rendimiento RF

  •Figura de ruido de la antena: 1.5dB

  •Impedancia de entrada es de 75 ohm, las pérdidas por desadaptacion al usar cableado de 50 ohmios en una entrada de 75 ohmios son de 0,177 dB.

  •Cobertura de radiofrecuencia: 25 MHz a 1,75 GHz

La siguiente imagen muestra el diagrama de bloques del SDR-RTL que esta formado por dos componentes hardware bien diferenciados(véase Figura 2):

• Rafael Micro R820T componente RF Front-end.
• Realtek RTL2832U compone el demodulador, interfaz USB y conversores analógico digitales.

A continuación se detallará cada una de las dos partes para terminar explicando sus aspectos técnicos más destacados.

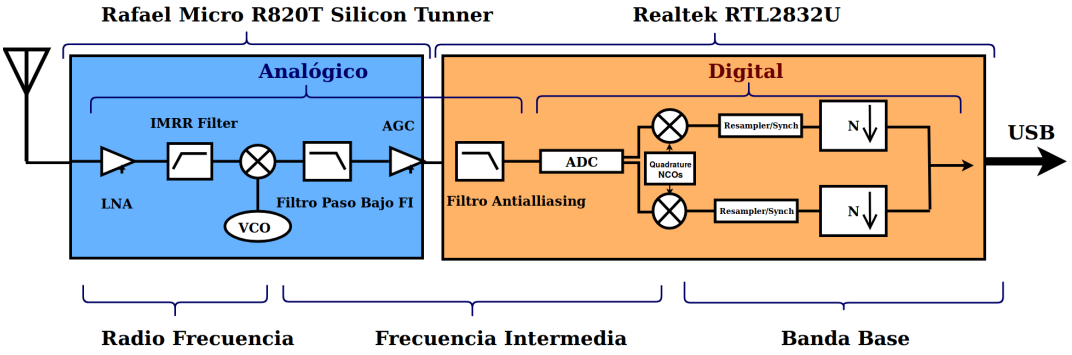

Figura 2: Diagrama de bloques SDR-RTL

El sintonizador digital R820T logra el menor consumo de energía además de ofrecer un correcto rendimiento en RF para todos los estándares de televisión digital, incluyendo DVB-T, ATSC, DMB-T, ISDB-T. R820T proporciona un rendimiento notable en sensibilidad, linealidad, inmunidad al canal adyacente y rechazo de la imagen. El chip incorpora un detector de energía inteligente para optimizar diferentes entradas. El R820T es un sintonizador de silicio altamente integrado que reúne los siguientes componentes: amplificador de bajo ruido (LNA), mezclador, PLL, VGA, regulador de voltaje y filtro de seguimiento, eliminando la necesidad de una interfaz externa, filtros de sierra y un AGC. Gracias a la arquitectura LNA, R820T ofrece el coste más bajo y permite una solución de rendimiento para aplicaciones de TV digital que cumple perfectamente con la tendencia mundial. El tipo de conector en la antena y la placa USB es MCX (Micro Coaxial Connector) - MCX macho en la antena, MCX hembra en el R820T. El diámetro exterior del conector es aproximadamente 3,6 mm y la superficie de contacto esta chapada en oro. El sintonizador digital R820T posee las siguientes características técnicas que se recogen en las dos tabla siguientes[5]:

•En la Tabla 1, parámetros de corriente alterna del R820T, se muestran los siguientes parámetros donde el más destacados de cara a este proyecto es la figura de ruido que es 3,5 dB.

Tabla 1: Parámetros de corriente alterna del R820T

Parámetro	**Condición**	**Unidades**	**Estándar**
Pérdida de retorno de la entrada	S11	dB	-10
Ganancia Tensión		dB	90
Rango AGC		dB	104
Figura de ruido	Máxima ganancia	dB	3,5
IP3 LNA	LNA Max Gain	dBm	35
IP3 LNA	LNA Min Gain	dBm	-7,5

•En la Tabla 2, especificaciones del oscilador, se muestran los siguientes parámetros donde el más importante de cara a este proyecto es el rango de frecuencias del cristal abarcando desde 12 MHz hasta 32 MHz siendo 28 MHz el valor recomendado por el fabricante a usar.

Tabla 2: Especificaciones del oscilador

Parámetro	**Unidades**	**Mínimo**	**Estándar**	**Máximo**
Rango de frecuencias	MHz	12	28.8	32
ESR Equivalent Series Resistance	Ohm		50	
Precisión en frecuencia	ppm		$\pm$ 30	$\pm$ 50
Condensador de carga	pF		16	
Nivel de tensión del Pin XTAL_p	mVpp	120		3300

La función[6] original del Realtek RTL2832U es DVB-T y COFDM Demodulator. La "U" en el nombre significa la Interfaz USB 2.0. Un cable de extensión USB es recomendado con RTL-SDR para mantener la radio alejada del ruido generado por el ordenador. El RTL2832U contiene ocho puertos de entrada/salida de propósito general y un puerto de control remoto por infrarrojos para comunicación con el mando remoto.

El RTL2832U admite sintonizadores en IF, low-IF y zero-IF (conversión directa). El sintonizador más común es el Rafael Micro R820T mostrado anteriormente.

Este chip cuadrado posee una una tarjeta de sonido de gama alta de 32 bits a 192 KHz, un rendimiento de datos de 6.144 Mbps, solo tiene un ADC de 8 bits pero puede funcionar a 3,2 MSps. Esto se calcula con un rendimiento de 25.600 Mbps o cuatro veces el de una buena tarjeta de sonido. El RTL2832U se desarrolló para manejar flujos de transporte comprimidos como son MGEG2 y MPEG4 que contienen vídeo, audio y datos.

El núcleo del RTL2832U es su ADC y su DSP (Digital Signal Processor). Su funcionamiento es el siguiente: Primero se realiza una conversión descendente digital de FI a BB a través de los mezcladores . La fase de los mezcladores es de 90 grados de separación. En segundo lugar las muestras I/Q pasan por un filtrado paso bajo digital. Se termina enviando los datos I/Q de 8 bits a través del puerto USB 2.0 al ordenador.

Para desarrollar aplicaciones usando este hardware se podrá hacer uso de herramienta software como son GNU Radio o Matlab. Para un correcto uso es necesario al menos un procesador de doble núcleo si es requerido utilizar una interfaz gráfica. Si es a través de comandos las prestaciones hardware pueden ser menores. La Raspberry Pi 3 puede ejecutar alguna de las aplicaciones de procesado de la señal.

2.1.2 ADALM-PLUTO SDR

El ADALM-PLUTO SDR es un nuevo SDR que funciona tanto en recepción como en transmisión de Analog Devices que es un gran fabricante de semiconductores. El PlutoSDR da cobertura radio desde 325MHz hasta 3800 MHz, tiene un ADC de 12 bits con una frecuencia máxima de muestreo de 61.44 Mbps y un ancho de banda de 20 MHz. Diseñado para estudiantes se puede utilizar tanto para la investigación como para el aprendizaje de radiofrecuencia y comunicaciones de una manera más empírica[7].

En el pasado las pŕacticas de telecomunicaciones se realizaban con una simulación de Matlab. El Módulo de aprendizaje Pluto SDR es una herramienta que permite la relación entre la teoría de comunicaciones y las actividades prácticas en RF. Proporciona un laboratorio portátil personal que permite aumentar el aprendizaje que tiene lugar en el aula de teoría. Una variedad de software como son MATLAB o Simulink o GNU Radio, permitirá a los estudiantes de telecomunicaciones aprender más rápido y explorar el mundo de las comunicaciones inalámbricas.

Las especificaciones eléctricas de este periférico radio son las siguientes:

- Cobertura RF: 325MHz a 3.8GHz
- Ancho de banda :20 MHz
- Conversor ADC/DCA : 12 bits
- Interfaz USB : USB 2.0
- Frecuencia de muestreo: 61.44 Mbps
- Rendimiento y relojes
  - Velocidad de muestreo de ADC y DAC: 65.2 KSps siendo SPS (Samples Per Second) a 61.44 MSps
  - Precisión de frecuencia: $\pm$ 25ppm
- Rendimiento RF
  - Salida de potencia Tx: 7dBm
  - Figura de Ruido Rx: $<$ 3.5dB
  - Precisión de Modulación de Rx y Tx : -34dB
  - Blindaje de RF: ninguno
- Especificaciones digitales
  - Núcleo: ARM Cortex®-A9 a 667 MHz
  - Células lógicas FPGA: 28000
  - Módulos DSP: 80
  - DDR3L: 4GB (512MB)
  - QSPI Flash: 256MB (32MB)

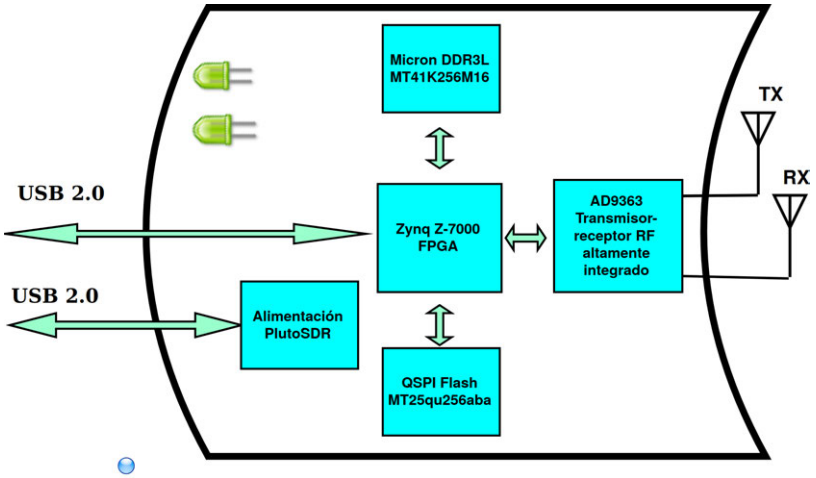

Figura 3: Diagrama de bloques ADALM PLUTO

La Figura 3 muestra el diagrama de bloques que conforman el Pluto SDR, principalmente por dos componentes hardware:

- Analog Devices AD9363 Highly Integrated RF Agile Transceiver.
- Rlinx® Zynq Z-7000 FPGA.

El AD9363 [8] es un transmisor-receptor de RF de alto rendimiento y altamente integrado diseñado para aplicaciones 3G y 4G. Su capacidad de programación y de banda ancha lo hacen ideal para una amplia gama de aplicaciones en radiofrecuencia. El dispositivo combina un parte frontal de radiofrecuencia con una sección de BB de señal mixta flexible y sintetizadores de frecuencia integrados. El AD9363 opera en el rango de frecuencias desde 325 MHz hasta 3.8 GHz, cubriendo la mayoría de las bandas con y sin licencia. Permite anchos de banda instantáneos desde 200 KHz hasta 20 MHz. Posee dos receptores independientes de conversión directa, cada uno de ellos incluye control de ganancia automático independiente (AGC), corrección de desplazamiento de CC, corrección de cuadratura y filtrado digital, eliminando así la necesidad de estas funciones en la BB digital. El AD9363 también tiene modos de ganancia manual flexibles ,dos ADC de alto rango dinámico por canal. Las señales I/Q recibidas pasan por filtros de diezmación y un filtro FIR para producir una señal de salida de 12 bits. Los transmisores utilizan una arquitectura de conversión directa que logra una alta precisión en la modulación. El monitor de potencia de transmisión incorporado se puede utilizar como detector de potencia que permite una potencia de transmisión altamente precisa. EL PLL esta totalmente integrado y proporciona una baja potencia en la síntesis de frecuencia para recibir y transmitir canales de transmisión. Todo el voltaje esta controlado por los osciladores (VCO). El núcleo del AD9363 se puede alimentar directamente desde 1,3V y se controla a través de un puerto serie estándar de 4 hilos y cuatro pines de control de entrada salida en tiempo real.Finalmente el AD9363 está empaquetado en un formato de 10 mm $\times$ 10 mm.

Las características técnicas se recogen en las dos tablas posteriores (véanse las Tablas 3 y 4). En la primera tabla se muestran los parámetros en recepción del AD9363 Highly Integrated RF Agile Transceiver, mientras que en la segunda tabla se muestran los parámetros en transmisión del AD9363 Highly Integrated RF Agile Transceiver.

Por otro parte la familia Zynq-7000 ofrece la flexibilidad y la escalabilidad de una FPGA que propor-

Tabla 3: Parámetros en recepción del AD9363 Highly Integrated RF Agile Transceiver

Parámetro	**Unidades**	**Estándar**
Pérdida de retorno de la entrada	dB	-10
Figura de ruido	dB	2.5
IP3 LNA	dBm	-18
IP2	dbm	40

Tabla 4: Parámetros en transmisión del AD9363 Highly Integrated RF Agile Transceiver

Parámetro	**Unidades**	**Estándar**
Pérdida de retorno de la salida	dB	-10
Potencia de salida máxima	dBm	2.5
OIP3 LNA	dBm	23

ciona potencia, rendimiento y facilidad de uso. La gama de dispositivos de la familia Zynq-7000 permite a los diseñadores apuntar aplicaciones sensibles al coste y de alto rendimiento desde una única plataforma. Los SoC (System on a Chip) Zynq-7000 y Zynq-7000S pueden servir a una amplia gama de aplicaciones que incluyen:

- Asistencia al conductor en automoción.
- IP y cámara inteligente.
- LTE (Long Term Evolution).

El entorno de desarrollo de Vivado Design Suite hace posible un producto rápido para ingenieros de software, hardware y sistemas. La inclusión de un procesador de aplicaciones permite el soporte de sistemas operativos de alto nivel por ejemplo GNU. Las características técnicas Zynq Z-7000 FPGA se dividen en las siguientes dos tablas [9].

•En la Tabla 5, parámetros de Processing System del Zynq Z-7000 FPGA,se muestran los siguientes parámetros donde los más relevantes de cara a este proyecto son los 32 Kbytes de la primera memoria cache y la máxima frecuencia del SoC de hasta 866 MHZ.

Tabla 5: Processing System

Nombre del dispositivo	Z-7010
Número de pieza	XC7Z010
Núcleo del procesador	Dual-core ARM Cortex-A9 MPCore con CoreSight
Extensiones del procesador	NEONTM Punto flotante de precisión simple o doble para cada procesador
Frecuencia del SoC	667 MHz 766 MHz 866 MHz
Cache L1	Instrucción 32KB
Cache L2	512 KB
Memoria On-Chip	256 KB
Memoria externa de apoyo	DDR3, DDR3L, DDR2, LPDDR2
Memoria estática externa Apoyo	2x Quad-SPI, NAND, NOR
Canales DMA (Direct Memory Access)	8 (4 dedicados a la lógica programable)
Periféricos	2x UART, 2x CAN 2.0B, 2x I2C, 2x SPI, 4x 32bits GPIO

•La Tabla 6, parámetros de Programmable Logic del Zynq Z-7000 FPGA,se muestran los siguientes características técnicas donde las más importante de cara a este proyecto son las 28000 células lógicas, 17600 LUTs y los 35200 Flip - Flops.

Tabla 6: Programmable Logic

Nombre del dispositivo	Z-7010
Número de pieza	XC7Z010
Celdas lógicas programables	28000
LUTs	17600
Flip-Flops	35200
Bloques de memoria RAM (Bloques de 36 Kb)	2.1 Mb (60)
Bloques DSP	80

2.1.3 USRP

Ettus Research es una marca de National Instruments (NI) desde 2010, es el proveedor líder de radio definidas por software, incluida la familia de productos Universal Software Radio Peripheral USRP. Soporta una gran variedad de entornos de desarrollo en una extensa gama de hardware de radiofrecuencia de alto rendimiento. La plataforma USRP es la plataforma SDR más elegida por ingenieros, científicos y estudiantes de todo el mundo para la creación de prototipos y desarrollo de tecnología inalámbrica de próxima

generación. La familia de productos USRP está diseñada para aplicaciones desde DC hasta 6 GHz, incluidos los sistemas de múltiples antenas (MIMO). Las áreas de aplicación incluyen equipo de pruebas, estaciones base móvil GSM (Global System for Mobile communications), receptor GPS (Global Positioning System), receptor y transmisor de radio FM y radar pasivo. A continuación se presentan los diferentes tipos de USRP actualmente disponibles y sus características más relevantes clasificadas en distintas familias.

1 Familia Networked series:

La familia de USRP Network Series es la que presenta un mayor rendimiento, soporta la tecnología MIMO. La conexión al ordenador tiene que ser obligatoriamente vía Gigabit Ethernet. Los modelos actualmente disponibles son:

•USRP N310 ZYNQ-7100, 4 canales, 10 MHZ a 6 GHZ, 10 GIGE, 10001,00 €

•USRP N300 ZYNQ-7035, 2 canales, 10 MHZ - 6 GHZ, 10 GIGE, 6499,00 €

•USRP N210 Kit USRP N200, 2 Cables RF SMA-Bulkhead , Ethernet cable ethernet, cable de alimentación DC 1947,00 €

2 Familia Embedded Series:

La familia USRP Embedded Series se caracteriza por tener dos procesadores: uno de propósito general y otro específico. El periférico puede operar desde DC hasta 6GHz, este hardware está diseñado para desarrollar aplicaciones en modo autónomo, no se requiere el uso de un ordenador. La plataforma USRP Embedded Series utiliza el marco OpenEmbedded para crear distribuciones Linux personalizadas adaptadas a las necesidades específicas de cada aplicación. Para reducir el esfuerzo de desarrollo, el sistema operativo predeterminado es compatible con la API del software USRP Hardware Driver (UHD) así como con una variedad de herramientas de terceros como GNU Radio. Los usuarios pueden crear prototipos rápidamente y desplegarlos de manera confiable para aplicaciones al aire libre. Los modelos de USRP de esta familia actualmente disponibles son:

• USRP E313 IP67 KIT 2x2 MIMO, 70MHz - 6GHz , 4110,00 €

•USRP E312 Bateria , 2X2 MIMO, 70 MHZ - 6 GHZ, 3286,00 €

• USRP E310 KIT (2x2 MIMO, 70MHz - 6GHz 3059,00 €

3 Familia X series:

• USRP X310 KINTEX7-410T FPGA, 2 canales, 10 GIGE y PCIE BUS 5428,00 €

•USRP X300 (KINTEX7-325TFPGA, 2 canales, 10 GIGE y PCIE BUS 4408,00 €

Ettus Research USRP X310 es una plataforma de radio definida por software escalable de alto rendimiento para diseñar e implementar sistemas de comunicaciones inalámbricas de próxima generación. La arquitectura de hardware combina dos ranuras de placa secundarias que cubren el rango de frecuencia desde DC hasta 6 GHz con un ancho de banda máximo de 120 MHz, múltiples opciones de interfaz de alta velocidad como son PCIe, 10 GigE dual, 1 GigE dual y un gran Kintex programable por el usuario. Además de proporcionar el mejor rendimiento de hardware de su clase, la arquitectura de software de código abierto de X310 ofrece compatibilidad con controladores UHD multiplataforma que lo hacen compatible con una gran cantidad de marcos de desarrollo, arquitecturas de referencia y proyectos de código abierto compatibles.

4 Familia Bus serie: La familia de USRP Bus Series se caracteriza porque la conectividad al ordenador

se realiza a través de un puerto USB, se explicarán brevemente los modelos a excepción del B200MINI-I que se tendrá como referencia comparativa con el SDR-RTL y el Adalm Pluto

•USRP B200MINI-I 1X1, 70 MHZ - 6 GHZ 906,00 €

•USRP B205MINI-I 1X1, 70 MHZ - 6 GHZ) 932,00 €

•USRP1 Kit (USRP1 Motherboard, Chassis, USB Cable, Power, Screw Kit, Fan) descatalogada

El USRP1 es el hardware original de Universal Software Radio Peripheral (USRP), proporciona capacidad de procesado de RF. Está pensado para proporcionar una capacidad de desarrollo de radio definida por software para usuarios y aplicaciones sensibles al coste. La arquitectura incluye un Altera Cyclone FPGA, dos ADC a 64 MSps , dos DAC a 128 MSps y conectividad USB 2.0 para proporcionar datos a los procesadores del ordenador. Un diseño modular permite que el USRP1 funcione desde DC hasta 6 GHz. La plataforma USRP1 puede admitir dos placas secundarias de radiofrecuencia. Esta característica hace que el USRP1 sea ideal para aplicaciones que requieren un alto aislamiento entre los canales de transmisión y recepción .El USRP1 puede transmitir hasta 8 MSps desde y hacia las aplicaciones del ordenador y los usuarios pueden implementar funciones personalizadas en la estructura del FPGA. Este modelo ya no está disponible para su compra.

El USRP B200mini-i ofrece una radio cognitiva con el tamaño de una tarjeta de crédito. Con una amplia gama de frecuencias desde 70 MHz a 6 GHz y un FPGA Xilinx Spartan-6 XC6SLX150 programable por el usuario. Esta plataforma flexible y compacta es ideal para aplicaciones de fábrica y para radioaficionados. El extremo frontal de radiofrecuencia utiliza el transceptor RFIC AD9364 de Analog Devices con un ancho de banda instantáneo de 56 MHz. El USRP B200mini-i está alimentado por bus mediante una conexión USB 3.0 de alta velocidad para transmitir datos al ordenador. El USRP B200mini-i también incluye conectores GPIO (General Purpose Input/Output) y JTAG (Joint Test Action Group) para la sincronización con un de reloj externo de 10 MHz o una señal de entrada de referencia de tiempo PPS.

La Figura 4 muestra el diagrama de bloques del USRP B200mini que esta formado por dos componentes hardware bien diferenciados:

- AD 9364 RFIC
- SPARTAN FPGA.

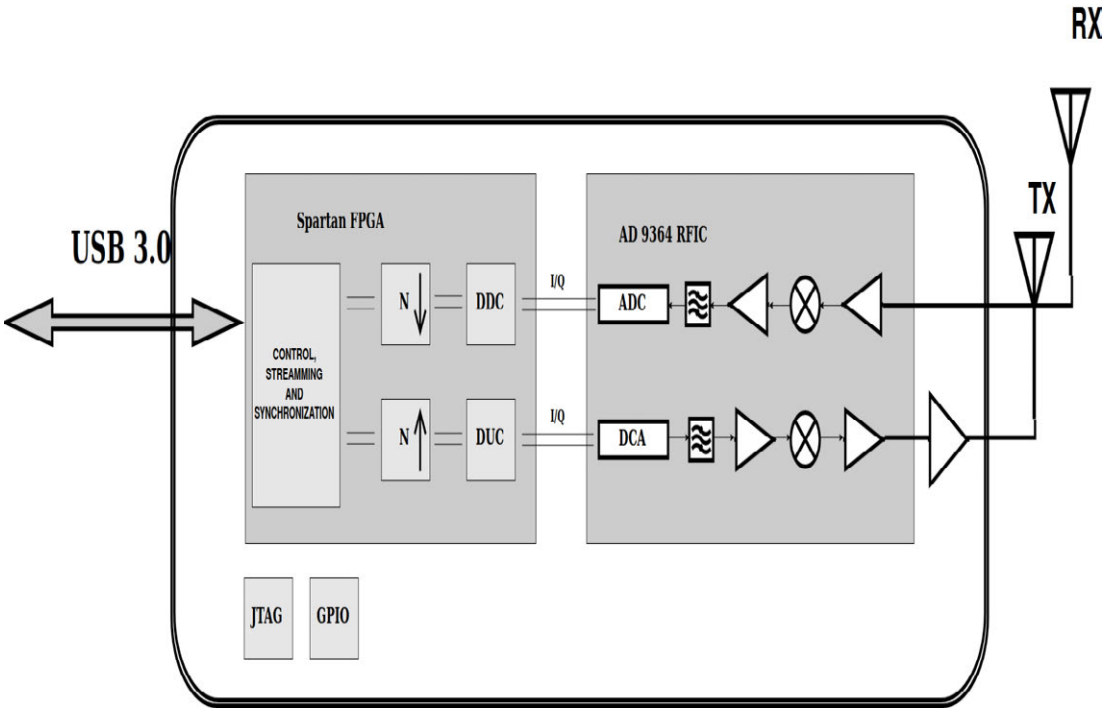

Figura 4: Diagrama de bloques USRP B200mini-i

Sus características técnicas más relevantes son:

•Cobertura RF : 70MHz a 6 GHz

•Ancho de banda : 56 MHz

•Conversor ADC/DCA : 12 bits

•Interfaz USB : USB 3.0

•Frecuencia de muestreo: 61.44 Mbps

•Rendimiento RF

- •Potencia de transmisión: $\geq$ 10dBm
- •Figura de Ruido Rx: $\leq$ 8dB
- •Precisión de frecuencia: $\pm$ 2 ppm

Para terminar este apartado se muestra la comparativa final (véase Tabla 7) de las tres radios definidas por software estudiadas: SDR-RTL, ADALM PLUTO Y USRP B200mini

Tabla 7: Comparativa de módulos SDR

	SDR-RTL	**ADALM-PLUTO**	**USRP B200mini-i**
Funcionamiento	RX	RX/TX	RX/TX
Rango de frecuencias	25 MHz a 1,7GHz	325MHz a 3,8GHz	70MHz a 6 GHz
ADC/DAC	8 bits	12 bits	12 bits
Frecuencia de muestreo	2.4 MSps	61.44 MSps	61.44 Msps
Dimensiones	85mm x 25 mm x 10 mm	117 mm x 79 mm x 24 mm	83.3mm x 50.8mm x 8.4mm
Precio	21,7 €	120,59 €	906.00 €

2.2 Software

Las herramientas software más frecuentes para el procesado de la señal para poder trabajar con la tecnología SDR son:

•GNU Radio: *"GNU Radio es un conjunto de herramientas de desarrollo de software de fuente abierta y gratuita que proporciona bloques de procesado de señales para implementar radios de software. Puede usarse con hardware de RF externo de bajo costo y fácilmente disponible para crear radios definidos por software, o sin hardware en un entorno similar a la simulación. Es ampliamente utilizado en la investigación, la industria, la academia, el gobierno y los entornos de aficionados para apoyar tanto la investigación de comunicaciones inalámbricas como los sistemas de radio del mundo real."*[10]

•Simulink: *"Simulink es un entorno de diagrama de bloques para simulación multidominio y diseño basado en el modelo. Soporta diseño de nivel de sistema, simulación, generación automática de código y verificación de sistemas embebidos. Simulink provee un editor gráfico, bibliotecas de bloques personalizables y solucionadores de problemas para modelar y simular sistemas dinámicos. Está integrado con MATLAB, que permite incorporar modelos de algoritmos MATLAB y exportar los resultados de simulación en MATLAB para su posterior análisis."*.[11]

•LabView: *"LabVIEW es un software de ingeniería de sistemas que requiere pruebas, medidas y control con acceso rápido a hardware e información de datos."*[12]

Conociendo los diferentes software más usado para trabajar con la tecnología se pasará a explicarlos con más detalle GNU Radio al ser usado principalmente en este proyecto fin de grado.

2.2.1 GNU Radio

GNU Radio es un grupo de archivos y aplicaciones agrupadas en diferentes librerías, se implementan en C ++, permiten generar modificar y recibir señales mediante el procesado digital. Las librerías proporcionan la oportunidad de simular o diseñar sistemas radio definidos por software si se conecta al ordenador un SDR. GNU Radio funciona en sistemas operativos Linux como Ubuntu o Debian pudiendo instalarse en Mac y Windows de una manera más compleja. En la Tabla 8 se muestra las diferentes formas de instalar GNU Radio en distintas distribuciones de Linux:

Tabla 8: Tabla de comandos para instalar GNU Radio en distintas distribuciones Linux

Distribución	**Comando**
Debian/Ubuntu	apt install gnuradio
Fedora	dnf install gnuradio
RHEL/CentOS	yum install gnuradio
Archlinux	pacman -S gnuradio
Gentoo Linux	emerge net-wireless/gnuradio

Los diseños realizados en GNU Radio se programan en el lenguaje orientado a objetos Python. Aunque Python sea el lenguaje de programación usado para generar los sistemas de comunicaciones, por norma general se usa la aplicación GNU Radio Companion. GNU Radio Companion es un entono de programación visual que permite a los usuarios diseñar, simular y desplegar sistemas de comunicaciones radio. Es un entorno gráfico altamente modular, orientado al diagrama de flujo a través de la concepción de cajas negras. Esta formado por una biblioteca completa de bloques de procesado "cajas negras" que se pueden combinar fácilmente para realizar aplicaciones de procesado de señales. Para diseñar un sistema de radiocomunicaciones con GNU Radio se debe crear un grafo donde los nodos son los bloques de procesado de la señal y los enlaces entre bloques representan el flujo de datos. GNU Radio genera archivos con extensión .grc[13]. Una vez realizado el diseño en GRC (GNU Radio Companion), GNU Radio Companion genera un código en Python, bien puede ejecutarse desde la propia herramienta o desde la línea de comandos en un terminal como un código python. El siguiente código nos muestra la simple ejecución a través del interprete de comandos.

```
sudo python RX_PMR_446.py
```

Este entorno de trabajo ha permitido el desarrollo de aplicaciones radio, comunicaciones móviles, diseño de radares, redes GSM etc a través de software. No proporciona aplicaciones para el uso de estándares de radiocomunicaciones específicos como el estándar 802.11, LTE, etc pero sí para desarrollar implementaciones esenciales de cualquier sistema de comunicación.

Para iniciarse en la programación de Software Radio con GNU Radio Companion es necesario conocer y saber cómo funcionan las estructuras de grafos, bloques y conexiones. Lo primero a entender es la interfaz gráfica en la cual hay cuatro partes, cada una de las cual se identificará con un color diferente, quedará recogida en la Figura 5.

- Área de trabajo(rojo): contiene los bloques que conforman el diagrama de flujo (grafo).
- Barra de herramientas(amarillo): componente de la interfaz mostrado en pantalla a modo de fila que contiene iconos que activan ciertas funciones como son guardar, copiar, pegar...
- Terminal(verde): muestra el código Python generado por GNU Radio Companion del diseño así como las variables que se están usando.
- Biblioteca(azul): contiene los diferentes bloques de procesado a usar.

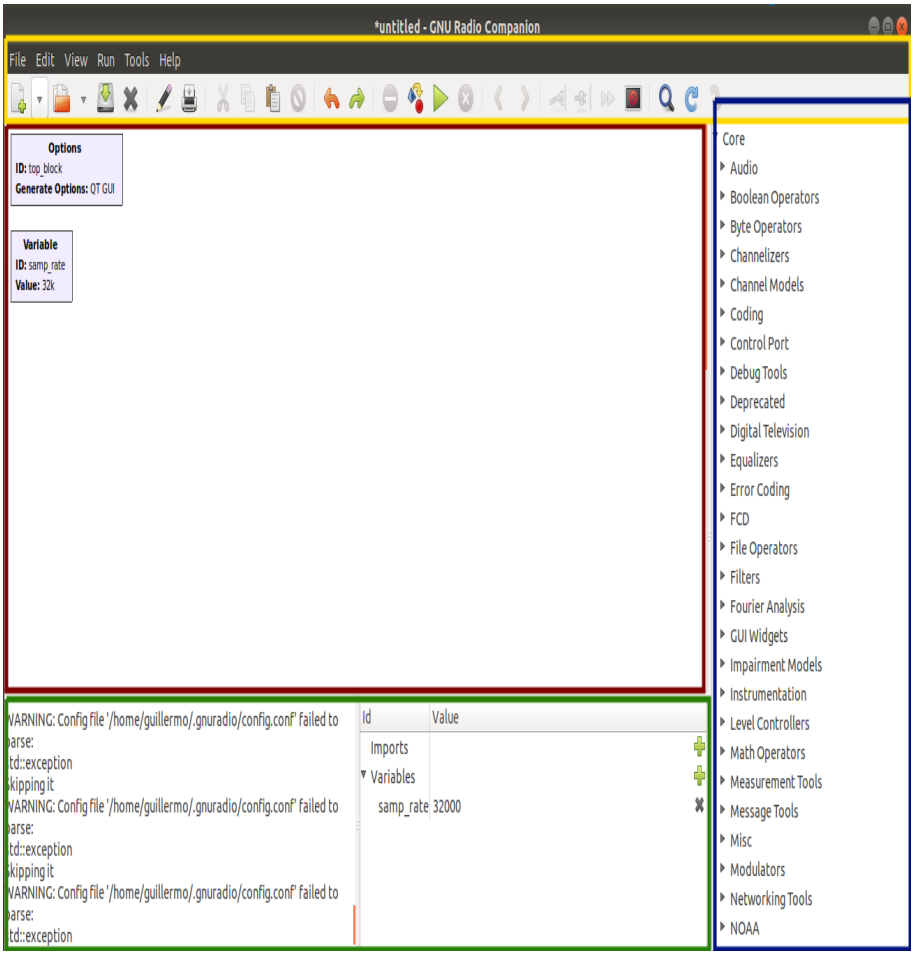

Figura 5: Interfaz gráfico GNU Radio Companion

Para la búsqueda de bloques solo es necesario pulsar Ctrl +f, pinchando, escribir una palabra clave para identificar el bloque y realizar doble clic con el botón izquierdo del ratón, aparecerá directamente en nuestra área de trabajo. Los bloques utilizados se pueden dividir en tres categorías (véase Figura 6):

•Sources : estos bloques especifican cualquier tipo de fuente de información como pueden ser ficheros, constantes numéricas,generador de señal o un micrófono.

•Bloques de procesado de la señal: son aquellos que se encargan de tratar la señal por ejemplo filtros, moduladores, operadores lógicos, diezmadores e interpoladores.

•Sinks: estos bloques son los encargados de almacenar o visualizar la información final del sistema tales como ficheros, visualizadores gráficos o altavoces.
Estos tres tipos de bloques se visualizan en la Figura 6: Bloques Gnu Radio

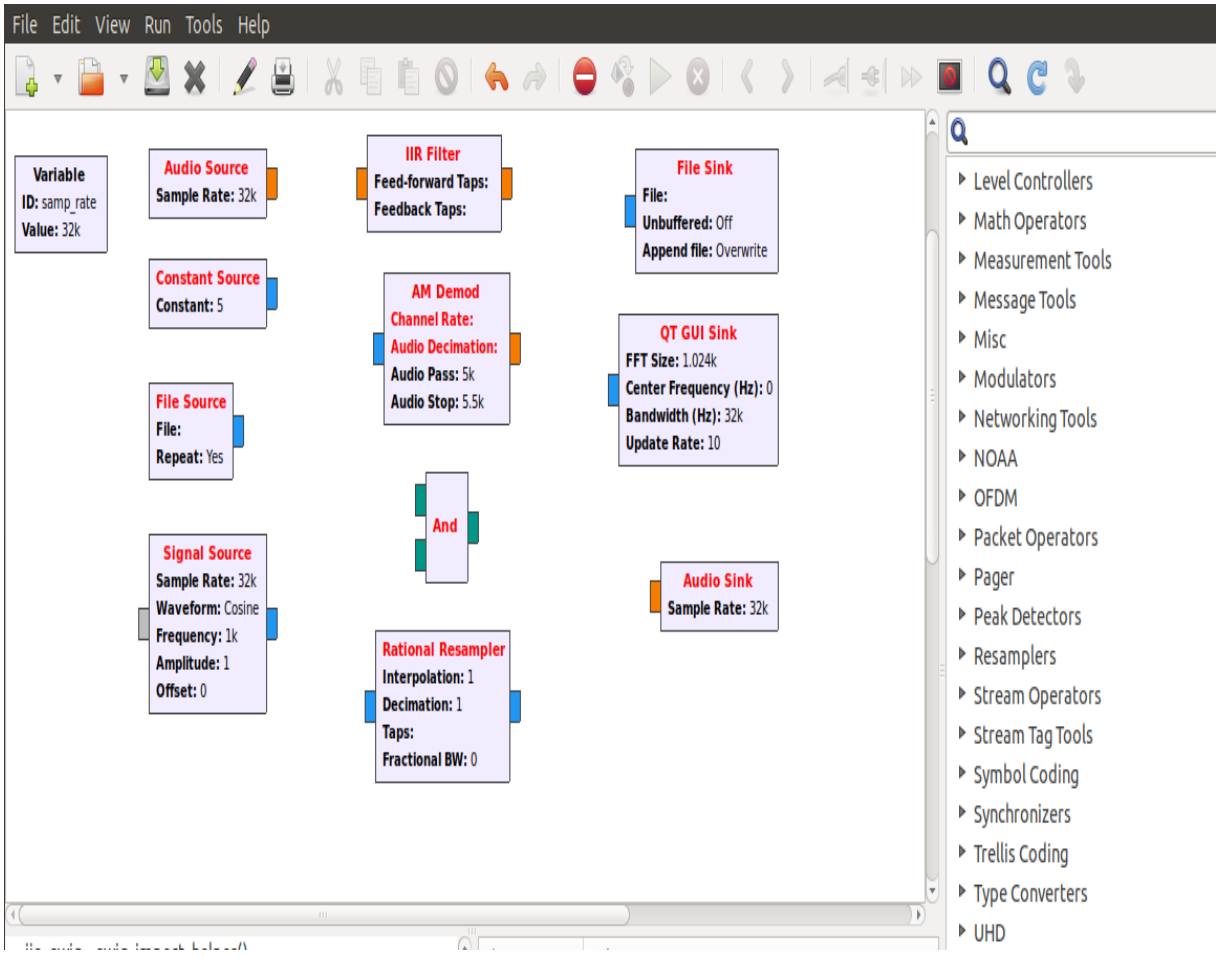

Figura 6: Ejemplos de bloques de procesado de Gnu Radio Companion

Cada uno de los bloque tiene diferentes parámetros de configuración. Se observa que todos los diseños generados especial tienen que tener un bloque llamado Bloque Options ,aparecerá por defecto al generar un nuevo diseño. Además se puede agregar bloques Variable que guardarán un valor numérico a una palabra asociada, más adelante se explicará a modo de ejemplo.

Una vez explicados los tres tipos de bloques, se pasará a realizar el primer diagrama de flujo propuesto por Guided Tutorial GRC a modo de ejemplo. El siguiente diagrama de flujo estará formado por tres bloques :Signal Source (Generador de la señal), un throttle (limita el rendimiento del diagrama a la velocidad de muestreo especificada) y QT GUI Time Sink (un visualizador de la señal en el tiempo)[14]. La finalidad de este flujo es la visualización en el tiempo de un seno de amplitud 1 voltio con una frecuencia de 50 Hz. Para ello se declararán dos variables:ampl (valor 1) y freq (valor 50) situadas en la esquina superior izquierda del área de trabajo. Para modificar las propiedades del bloque Signal Source se pincha doble click. La siguiente imagen muestra las propiedades de Signal Source y como los campos Frecuency y Amplitude se rellenan con las variables anteriores (véase Figura 7).

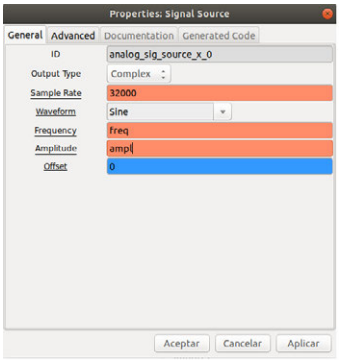

Figura 7: Propiedades del bloque de procesado ”Signal Source”

Tras la inserción de los bloques y ajustar las propiedades se pasara a la ejecución. Para activar el diagrama de flujo basta con pinchar el icono de la flecha verde en el menú, apareciendo automáticamente la visualización del seno. La imagen de la página posterior mostrará el resultado del diagrama de flujo así como el diseño generado. Se ha encuadrado en diferentes colores un resumen de la información más importante para trabajar con GNU Radio Companion. Los dos cuadros rojos indican las variables, en el superior por medio de los bloques y en el inferior la terminal indica las variables usadas. El cuadro azul indica el flujo de diseño formado por los bloques de procesados mencionados anteriormente. El verde muestra la visualización en el tiempo del seno generado. Por último los cuadros amarillo muestra el código Python generado por GNU Radio Companion. En el Anexo I se muestra el código generado además de la ejecución del código Python a través del interprete de comandos.

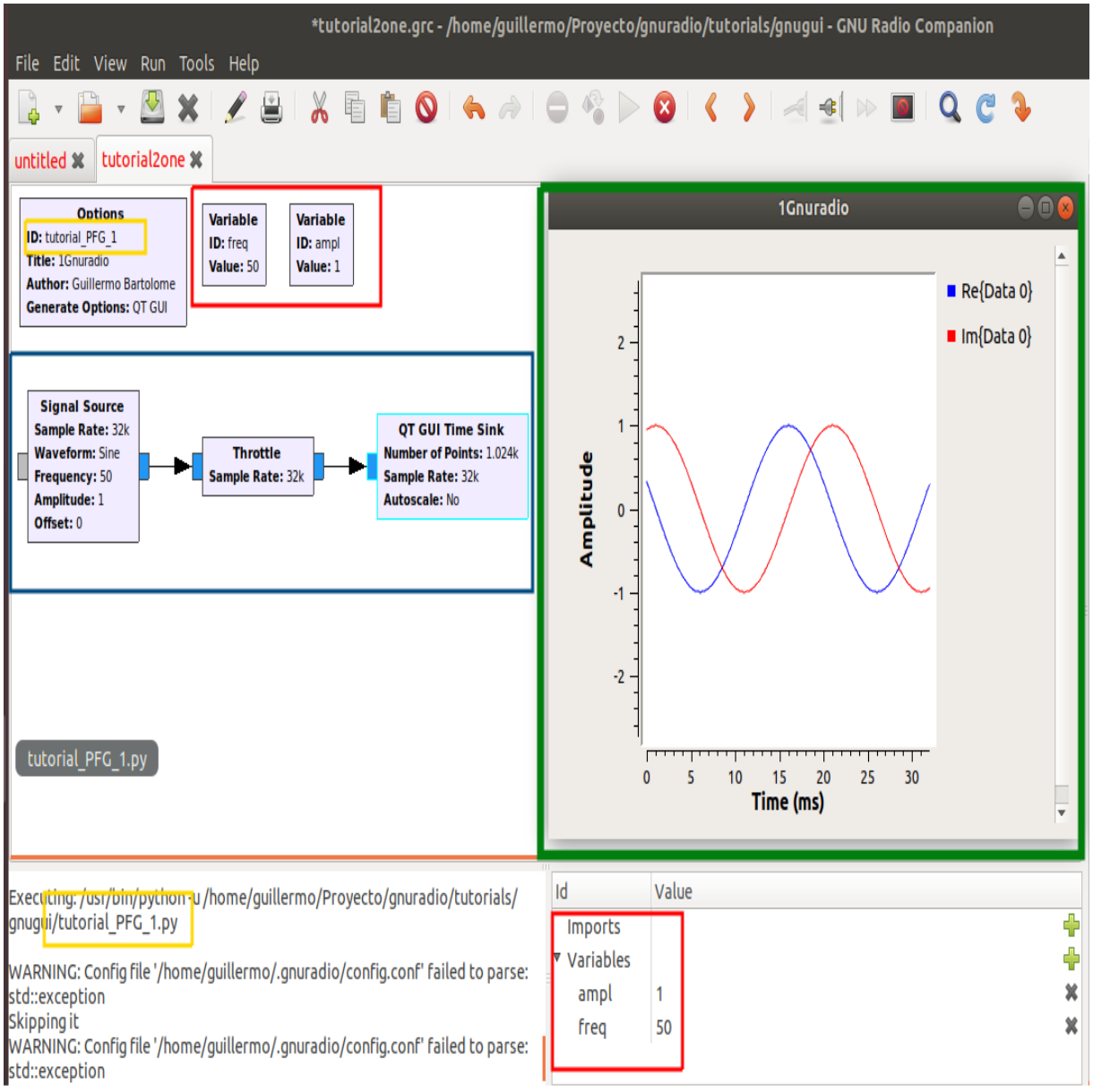

Figura 8: Primer diagrama de flujo y resultado

Para poder trabajar con las SDR y GNU Radio Companion es necesario la creación de las bibliotecas RTL y Adalm-Pluto a través de los repositorios y trabajar con las radios de manera virtual. Los repositorios serán bajados de Github e instalados de manera manual haciendo uso del interprete de comandos de Ubuntu. A continuación se enumerarán los pasos a realizar y se explicará cada acción.

Biblioteca SDR-RTL[15]

1 git clone `git://git.osmocom.org/rtl-sdr.git`

• Clona el repositorio git.osmocom.org en el nuevo directorio rtl-sdr

2 cd rtl-sdr/

• Cambia al directorio rtl-sdr

3 mkdir build

• Crea el directorio build

4 cd build

• Cambia al directorio build

5 cmake ../

6 make

• Permitirá la compilación de los repositorios que componen el código fuente y la construcción de las futuras bibliotecas de la radio.

7 sudo make install

• El comando copiará las bibliotecas creadas y documentación en las ubicaciones correctas.

8 sudo ldconfig

• Creará la memoria caché y los enlaces directos entre el directorio que tiene el repositorio y el GRC para poder ser usado posteriormente por el interfaz gráfico.

La instalación de la biblioteca SDR Adalm Pluto es idéntica a la anterior. Se realizará a través de tres repositorios:

Primer repositorio analogdevicesinc/libiio:

1 git clone `https://github.com/analogdevicesinc/libiio.git`

2 cd libiio

3 mkdir build

4 cd build

5 cmake ../

6 make

7 sudo make install

8 sudo ldconfig

Segundo repositorio analogdevicesinc/libad9361:
`https://github.com/analogdevicesinc/libad9361-iiogit`

Tercer repositorio analogdevicesinc/gr-iio:
`https://github.com/analogdevicesinc/gr-iio.git`

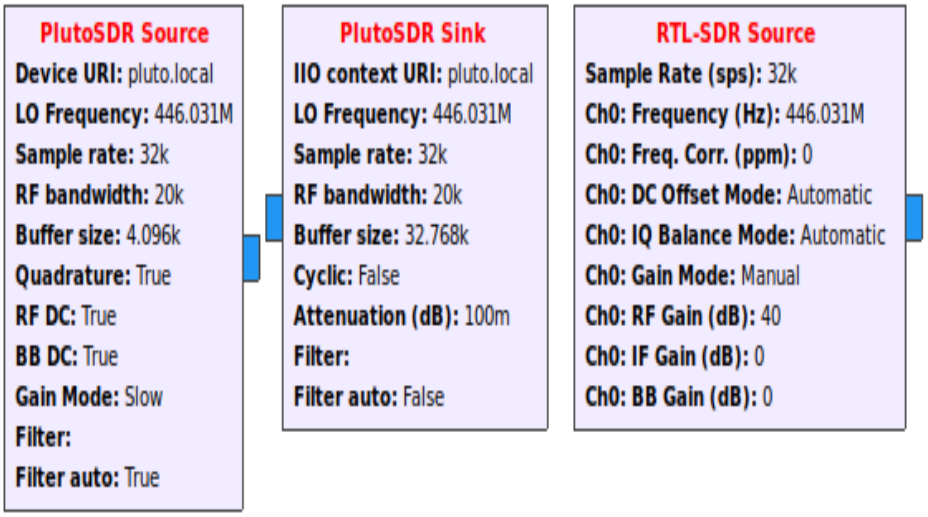

Figura 9: Bloques SDRs en el GRC

De izquierda a derecha el primer bloque es la SDR ADALM-PLUTO receptora, la SDR ADALM-PLUTO transmisora Y la SDR-RTL receptora.

2.2.2 Simulink

Simulink es un entorno de programación gráfica presente dentro del software Matlab. Simulink genera archivos con extensión .mdl de model mientras que Matlab archivos.mat por tanto tiene un nivel de abstracción mayor. Se trata de una herramienta de simulación de sistemas que hace énfasis en el análisis de sucesos a través de bloques. Los bloques que conforman las librerías de Simulink son funciones desarrolladas en Matlab. Ventajas e inconvenientes de Simulink:

•Visualización en tiempo real del flujo de datos e interfaz de programación visual e intuitiva.

•Código de ejecución lento.

Matlab y Simulink para comunicaciones inalámbricas [16]

•Genera formas de onda de los últimos estándares de 5G, LTE y WLAN.

•Cree modelos con elementos de tipo digital, RF ,DSP y antenas para desarrollar sistemas de comunicaciones inalámbricas.

Se usará Simulink como primer entorno de pruebas con la SDR Adalm Pluto Esta prueba realizada MathWorks muestra cómo utilizar la radio ADALM-PLUTO como analizador de espectro.[17], ejecuta el siguiente comando en Matlab:

```
plutoradioSpectralAnalysisExample
```

Aparecerá el modelo mostrado en la Figura 10

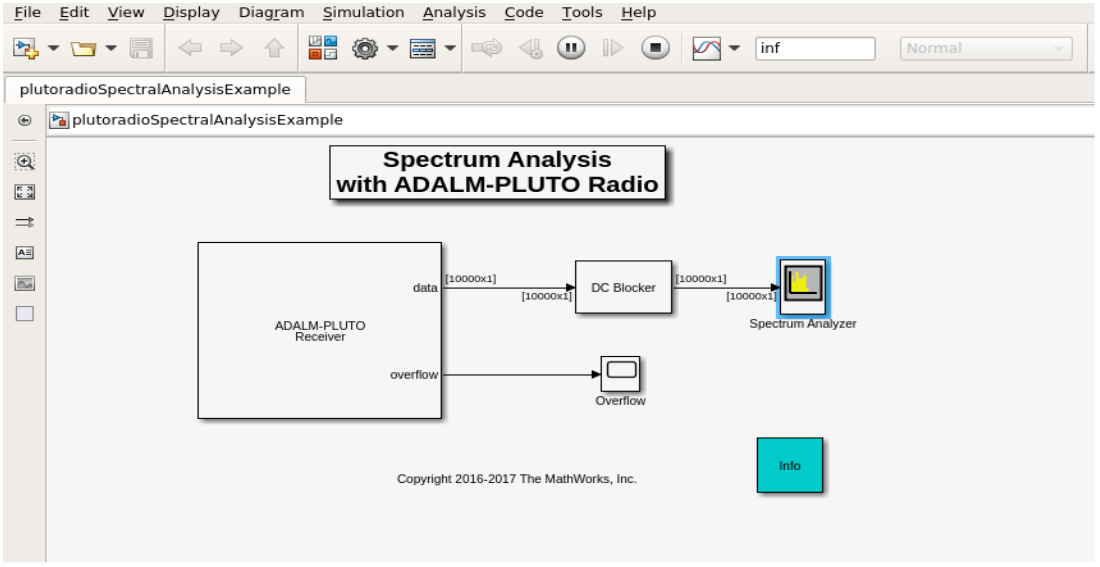

Figura 10: PlutoradioSpectralAnalysisExample

Se cogerá uno de los Twin Talker 4810 sintonizado a la frecuencia 446,032125 MHz transmitiendo un PPT (Push To Talk) para futura visualización del espectro de la señal. Comprobándose un correcto

funcionamiento de la radio basado en software. La imagen posterior mostrará la visualización del espectro de la señal.

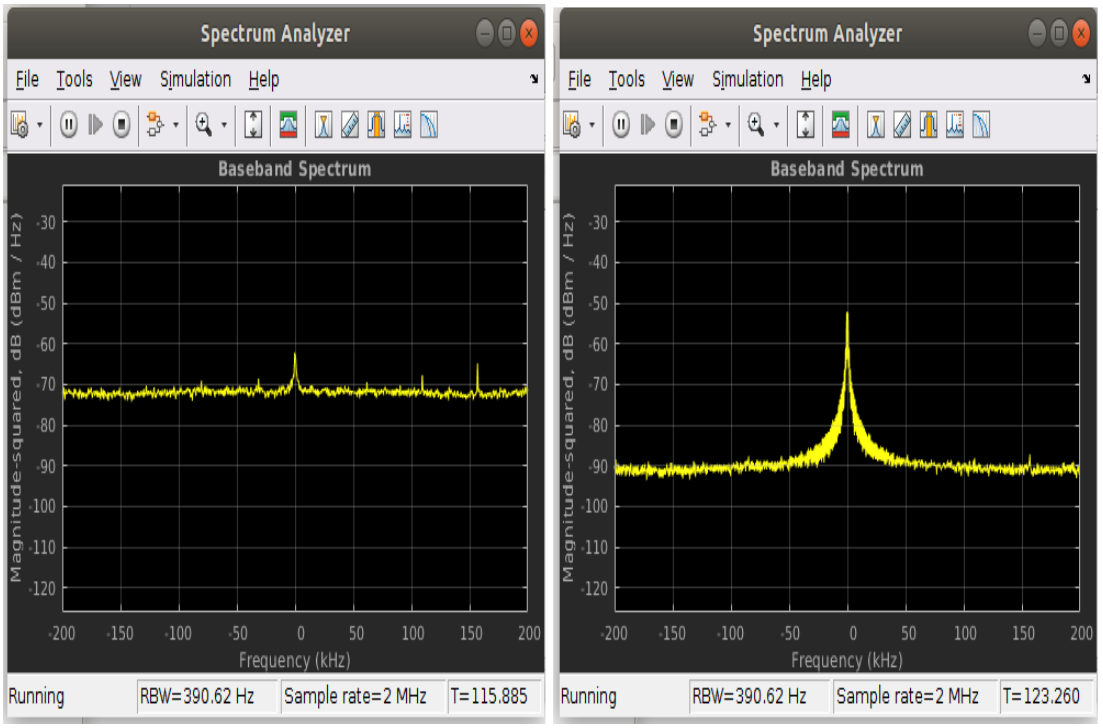

Figura 11: Resultados ”plutoradioSpectralAnalysisExample” : visualización de una señal PMR 446

A la izquierda de la figura se visualiza el espectro sin recepción de la señal PMR mientras que a la derecha se verá el espectro de una modulación en frecuencia de banda estrecha correspondiente a la comunicación del Twin talker 4810.

2.2.3 LabVIEW

LabVIEW es una plataforma y entorno de desarrollo para diseñar sistemas, con un lenguaje de programación visual gráfico pensado para sistemas hardware y software de pruebas, control y diseño.

Este programa fue creado por National Instruments en 1976 para funcionar en máquinas MAC, salió al mercado por primera vez en 1986, teniendo versiones disponibles para los sistemas operativos Windows, UNIX, MAC y Linux. Por otra parte se puede usar para el firmware de un dispositivo RF de última generación. Los programas desarrollados con LabVIEW se llaman Instrumentos Virtuales, o VIs, y su origen provenía del control de instrumentos, aunque hoy en día se ha expandido ampliamente no sólo al control de todo tipo de electrónica sino también a su programación embebida, comunicaciones, matemáticas, etc. Un lema tradicional de LabVIEW es: ”La potencia está en el Software”, que con la aparición de los sistemas multinúcleo se ha desarrollado aún más potente. Entre sus objetivos están el reducir el tiempo de desarrollo de aplicaciones de todo tipo y permitir la entrada a la informática a profesionales de cualquier otro campo de las tecnologías de la información y la comunicación. LabVIEW consigue combinarse con todo tipo de software y hardware, tanto del propio fabricante como de otros fabricantes.

Su principal característica es la facilidad de uso, válido para programadores profesionales como para personas con pocos conocimientos en programación pueden hacer programas relativamente complejos, imposibles para ellos de hacer con lenguajes tradicionales. Para los amantes de lo complejo con LabVIEW se pueden crear programas de miles de VIs que equivaldrían a millones de páginas de código. Incluso existen buenas formas con este software para optimizar el rendimiento y la calidad de la programación del diseño que se quiere desarrolla.

LabView puede desarrollar aplicaciones en los siguientes campos:

- Interfaces de comunicaciones:
  - Puerto serie
  - Puerto paralelo
  - GPIB
  - PXI
  - VXI
  - TCP/IP, UDP, DataSocket
  - Bluetooth
  - USB
- Herramientas gráficas y textuales para el procesado digital de señales.
- Visualización y manejo de gráficas con datos dinámicos.
- Adquisición y tratamiento de imágenes.
- Control de movimiento.
- Programación de FPGAs para control o validación.

2.3 Rendimiento de los softwares

Se procederá a evaluar el rendimiento de los software Simulink y Gnu Radio en el portátil personal por medio del monitor del sistema. Por otro lado se cronometrará el tiempo que tarda en abrirse el software y la ejecución de un diseño de prueba.

El monitor del sistema es una aplicación Ubuntu que mide el rendimiento de las CPU (Central Processing Unit) del ordenador así como la memoria RAM (Random Access Memory) utilizada. En las siguientes tres imágenes se estudiará el rendimiento del portátil personal según el software que se utilice para trabajar con las SDR.

La Figura 12 muestra el rendimiento del ordenador sin ningún software activo, se aprecia que ninguna de las CPU trabaja a más del 5%.

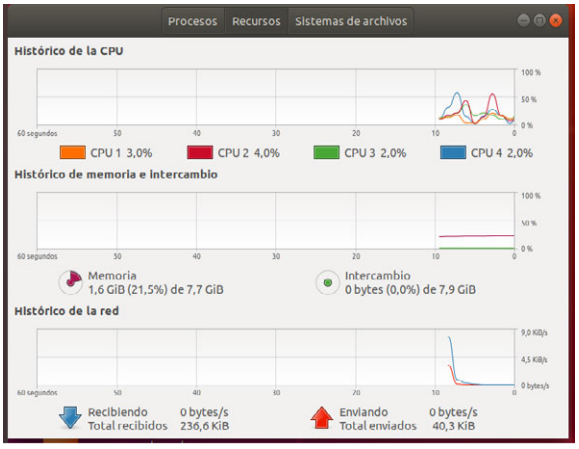

Figura 12: Rendimiento CPU sin software

En la Figura 13 se observa que el rendimiento del ordenador sube al utilizar Simulink, ejecutando el ejemplo plutoradioSpectralAnalysisExample. La tercera CPU alcanza un valor de un 80% por lo que el rendimiento general sube mucho dando lugar a un sobrecalentamiento del portátil personal. El tiempo de inicio de Matlab y después Simulink así como ejecutar el ejemplo plutoradioSpectralAnalysisExample fue de 5 minutos 43 segundos con 60 milésimas.

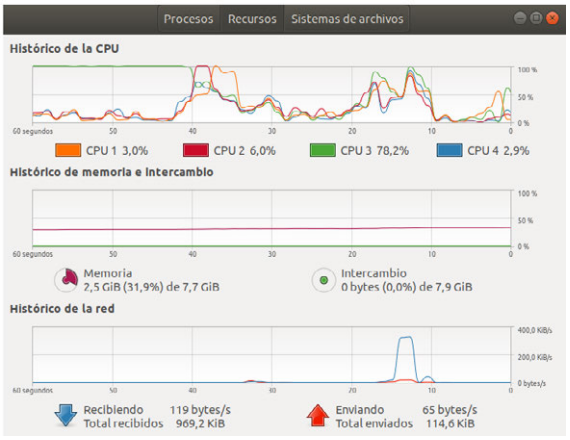

Figura 13: Rendimiento CPU ejecutando Simulink

En la Figura 14 se muestra el rendimiento del portátil personal al hacer uso del Software GNU Radio. Se observa que todas las CPU trabajan por igual, alrededor de un 10% cada una, sin producir sobrecalentamiento. Esta prueba de rendimiento se realizo con la SDR Adalm Pluto como analizador de espectro. El tiempo de inicio de GNU Radio Companion así como ejecutar el diseño fue de 37 segundos con 3 milésimas.

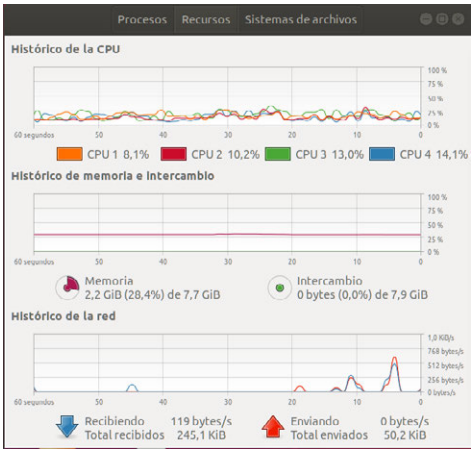

Figura 14: Rendimiento CPU ejecutando GNURadio

Los tiempos de ejecución de GNU Radio Companion son extremadamente inferiores que los de Simulink, aun cuando los dos se ejecutan sobre una distribución Ubuntu que son más rápidas que los Sistemas Windows. Con estas pruebas de rendimiento y medidas de tiempo del software se concluye que el software a utilizar para la sección siguiente, diseño de la estación remota de comunicaciones, es GNU Radio. En futuras pruebas del diseño de la estación remota de comunicaciones es necesario que la ejecución de los diseños sea rápidos, permitirá cambiar valores erróneos o dudosos de manera ágil y dinámica.

3 Diseño de la estación remota de comunicaciones

En este apartado se mostrará el desarrollo del sistema de comunicaciones remota viendo su evolución tanto a nivel esquemático, componentes hardware y desarrollo software, estructurando la información en tres partes. La finalidad principal de la estación remota de comunicaciones es permitir una comunicación PMR 446 distante por medio de un protocolo de comunicación de la capa de transporte. El primer esquemático (véase Figura 15) muestra una visión general de las comunicaciones PMR 446. Por otra parte el segundo esquemático (véase Figura 16) mostrará la idea a realizar de la estación remota de comunicaciones para este proyecto fin de grado.

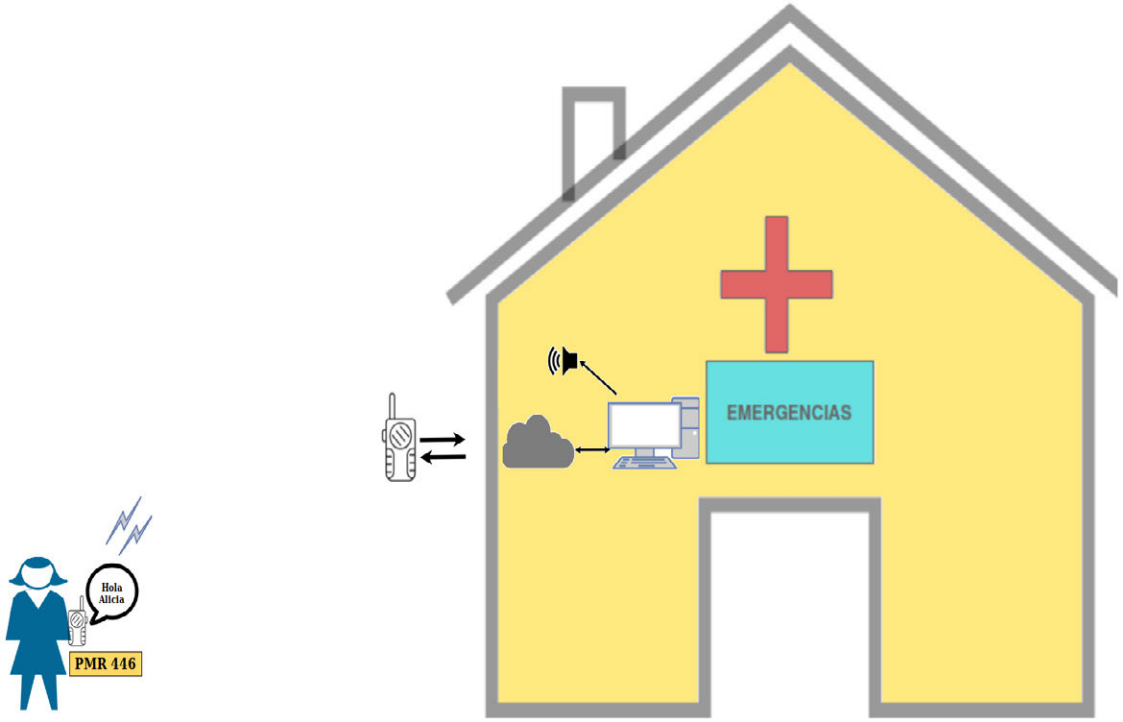

Figura 15: Esquemático comunicación PMR 446 walkie-walkie

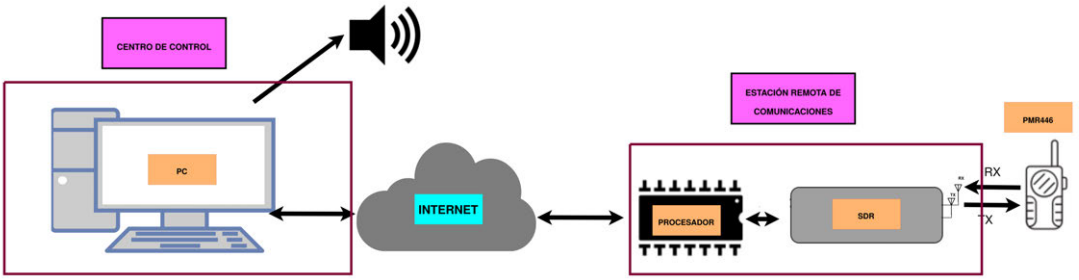

Figura 16: Esquemático estación remota de comunicaciones PMR 446

En primer lugar se realizará una descripción general de la modulación FM, se desarrollarán las comunicaciones PMR 446 con mayor profundidad dado que son las utilizadas en este proyecto. Se completará este apartado explicando los protocolos de comunicación: UDP y TCP.

En una segunda parte realizará una explicación del hardware que conforma la estación remota de comunicaciones. Los componentes hardware son: SDR-RTL, ADALM PLUTO SDR, Twintalker 4810 y Raspberry Pi 3 B+ como ordenador de procesado de la señal de comunicaciones.

El apartado posterior, desarrollo radio software, comprenderá el desarrollo software de la estación remota de comunicaciones. El desarrollo software explicará previamente los bloque de procesado de GNU Radio independientemente que conforman el sistema de comunicaciones remota. Hay que enfatizar la importancia que juega este apartado en el proyecto. El sistema se desarrollará de menor a mayor complejidad. Con objeto de facilitar la compresión de dicho apartado todos los desarrollos software han sido realizados a partir de la interfaz gráfica GNU Radio Companion. Se empezará por una simulación del sistema de comunicaciones inalámbrico y se finalizará por un diseño total de la estación remota de comunicaciones en la Raspberry Pi 3 B+, sera fundamental su entendimiento progresivo. Se dividirá el desarrollo software en las siguientes fases:

- Simulación TX/RX PMR 446
- Desarrollo software PMR446
  - Rx PMR446
  - Tx PMR446
- Estación remota de comunicaciones

3.1 Descripción general

3.1.1 FM

La FM es una modulación que hace posible transmisión de la información a través de una onda portadora variando su frecuencia. La frecuencia de la señal modulada variará en función de los cambios en la amplitud de la señal moduladora.

Aun sabiendo que la FM tiene un ancho de banda infinito será necesario entender cual es el ancho de banda necesario para tratar la FM. Según[18] *" La regla de Carson nos ofrece una estimación del ancho*

de banda necesario para el tratamiento de la señal FM sin que se pierda excesiva información, ya que teóricamente el espectro de señal ocupado por una señal modulada en frecuencia es infinito. Sin embargo, a partir de un cierto valor puede recortarse la señal sin que la pérdida de energía sea importante. La fórmula de la regla de Carson es: "

$$B \approx 2 * (\Delta f + f_m) = 2f_m(\beta + 1) \tag{3.1}$$

Siendo β la máxima desviación de fase:

$$\beta = \Delta f / f_m \tag{3.2}$$

Si tenemos en cuenta [19] se puede diferenciar dos tipos fundamentales de modulaciones en frecuencia en función del ancho de banda usado:

• Si $\beta <$ 0,1 sólo son relevantes 3 deltas. Se está en el caso de banda estrecha, cuyo ancho de banda es igual al de la modulación en amplitud.

• Si $\beta >>$ 0.1 se está en el caso de banda ancha.

3.1.2 PMR 446

¿Qué es PMR446? Los equipos PMR446 (Private Mobile Radio) son radios de uso libre en la banda de UHF (70 cm) y en el rango de frecuencias de 446,00625 MHz - 446,09375 MHz, de gran aceptación por el público en general, siendo usados tanto para actividades profesionales como de ocio[20].

Los dispositivos PMR 446 están diseñado para operar en bandas de frecuencias libres compartidas por muchos usuarios sin ninguna coordinación por lo tanto es necesaria una regulación en potencia. La regulación de esta banda permite una emisión máxima de 0,5 W de potencia PIRE. Los equipos PMR 446 funcionan en distancias cortas y no se pueden utilizar como parte de una red inalámbrica ni como repetidores. Las especificaciones técnicas son las siguientes:

•Numero de canales : 8

•Número de subtonos :CTCSS (Continuous Tone-Coded Squelch System) 38 / DCS (Digital Code Squelch) 83

•Rango de frecuencia :446,00625 MHz - 446,09375 MHz

•Alcance: 8 km en campo abierto y 2 km en ciudades.

•Potencia de transmisión 500 mW ERP (Effective Radiated Power)

•Tipo de modulación FM - F3E

•Separación de canales 12,5 KHz

Según [21] *"La Unión de Telecomunicación Internacional utiliza un sistema internacionalmente acordado para clasificar las señales de frecuencia radiofónica. Cada tipo de emisión radiofónica está clasificado según su ancho de banda, método de modulación, naturaleza de la señal de modular y tipo de la información a transmitir en la señal"*

Por lo tanto F3E significa

•F :FM

•3 :Un canal que contiene información analógica

•E :Telefonía (la voz o la música pretendieron ser escuchadas por humanos)

La modulación usada por los dispositivos PMR 446 es FM de banda estrecha. La Tabla 9 nos mostrará los canales PMR:

Tabla 9: Canales PMR 446

Canal	**Frecuencia(MHz)**
1	446,00625
2	446,01875
3	446,03125
4	446,04375
5	446,05625
6	446,06875
7	446,08125
8	446,09375

En telecomunicaciones, el squelch es una función realizada por un circuito electrónico que actúa para suprimir la salida de audio o vídeo de un receptor ante una señal de entrada. El squelch se usa ampliamente en radios de dos canales y escáneres de radio para suprimir el sonido del ruido del canal cuando la radio no está recibiendo una transmisión. El squelch se puede ”abrir”, lo que permite escuchar todas las señales que ingresan al receptor. Esto puede ser útil cuando se trata de escuchar señales distantes, débiles o de broadcasting de emergencia. Los dos squelch más usados son:

•CTCSS

•DCS

El sistema squelch CTCSS traducido al castellano como Sistema Silenciador Controlado por Tono Continuo es un circuito analógico que esta equipado en los dispositivos PMR 446. La funcionalidad de este circuito analógico es permitir que las señales de audio ,transmitidas por otro dispositivo PMR446, sean revividas en el receptor radio solo con una portadora modulada en un tono de frecuencia específico, evitando escuchar conversaciones ajenas en el mismo canal PMR. Si se elige el subtono 00 el sistema queda desactivado. Si se utiliza el CTCSS, se transmitirá un tono de frecuencia muy baja desde 67 Hz hasta 250 Hz junto con la señal de voz. Como consecuencia del filtrado en recepción, los tonos no suelen ser audibles para que no interfieran en la comunicación punto a punto. La desventaja de usar CTCSS en frecuencias compartidas es que los usuarios no pueden escuchar las transmisiones en grupos. En la Tabla 10 se pueden ver los primeros 20 subtonos CTCSS:

Tabla 10: Subtonos CTCSS

CTCSS	Frecuencia	CTCSS	Frecuencia
0	Nulo	10	94,8
1	67,0	11	97,4
2	71,9	12	100,0
3	74,4	13	103,5
4	77,0	14	107,2
5	79,7	15	110,9
6	82,5	16	114,8
7	85,4	17	118,8
8	88,5	18	123,0
9	91,5	19	127,3

Es necesario la transición hacia la tecnología digital en todos los sectores de las comunicaciones con el fin de satisfacer las necesidades de los usuarios al mismo tiempo que se mejora la eficiencia del espectro. Por este motivo surgieron los subtonos DCS.

DCS (Digital-Coded Squelch) conocido genéricamente como CDCSS traducido al castellano como Sistema Continuo Silenciador con Codificación Digital, fue diseñado como el reemplazo digital de CTCSS. El sistema squelch DCS es un circuito digital que esta equipado en los dispositivos PMR 446. Su funcionalidad es la siguiente, en lugar de transmitir un tono de baja frecuencia constante, el DCS transmite un flujo binario de datos continuo. Este flujo binario se transmite reiteradamente, esta formado por palabras binarias de 23 bits. La tasa de transmisión de bits es de 134.4 bits por segundo teniendo una tolerancia de + o - 0.4 bps. Estos códigos se utilizan como los anteriores para discriminar señales y que no interfieran en nuestras comunicaciones inalámbricas. Si se elige el subtono 0 el sistema queda desactivado. En la Tabla 11 se visualizará los subtonos DCS

Para terminar el uso de sistemas de squelch codificados puede evitar que se reciban señales no deseadas o débiles , por ejemplo cuando la persona que transmite o recibe se encuentra en un área obstruida por edificios o bien a una distancia considerable. Un receptor bien sintonizado que no está configurado para recibir tonos CTCSS o DCS seguirá transmitiendo el mensaje de audio de manera débil junto con el ruido estático de fondo. Por otro lado un receptor que tenga habilitado el squelch no recibirá tonos no deseados y recibirá el audio sin ruido de fondo estático. .

Tabla 11: Subtonos DCS

Número	DCS	Número	DCS	Número	DCS	Número	DCS
0	Disabled	21	134	42	311	63	516
1	23	22	143	43	315	64	532
2	25	23	152	44	331	65	546
3	26	24	155	45	343	66	565
4	31	25	156	46	346	67	606
5	32	26	162	47	351	68	612
6	43	27	165	48	364	69	624
7	47	28	172	49	365	70	627
8	51	29	174	50	371	71	631
9	54	30	205	51	411	72	632
10	65	31	223	52	412	73	654
11	71	32	226	53	413	74	662
12	72	33	243	54	423	75	664
13	73	34	244	55	431	76	703
14	74	35	245	56	432	77	712
15	114	36	251	57	445	78	723
16	115	37	261	58	464	79	731
17	116	38	263	59	465	80	732
18	125	39	265	60	466	81	734
19	131	40	271	61	503	82	743
20	132	41	306	62	506	83	754

3.1.3 UDP/TCP

Al tratarse de una estación remota de comunicaciones es necesario el envió de la información a un sitio distante en el que se procesará la información, para ello se usarán los protocolos de comunicación UDP o TCP. En este apartado se explicará el concepto de protocolo de comunicación y funcionamiento de los protocolos de transporte UDP y TCP.

Se puede diferenciar dos tipos de protocolos en la capa de transporte : orientado a la conexión y no orientado a la conexión. El primero consta de tres partes: establecimiento, transferencia y liberación. En el segundo tipo se envían los paquetes de manera individual sin previo establecimiento ni futura liberación.

UDP

El protocolo UDP, User Datagram Protocol, es un protocolo simple del nivel de transporte basado en la transferencia de datagramas sin previo establecimiento entre el cliente y el servidor. No proporciona confirmación por lo que no se puede asegurar con seguridad si los dato han sido recibidos de manera correcta, ni control de flujo por lo que los datagramas pueden adelantarse unos a otros y ser recibidos de forma desordenada. Para garantizar una transmisión correcta de la información se deberá implementar en capas superiores. Trabaja con datagramas enteros, no con bytes como su homologo orientado a la conexión TCP. Origina poca carga adicional en la red ya que es simples y emplea cabeceras no muy grandes. Un paquete UDP puede ser dividido por el protocolo de la capa de red IP (Internet Protocol) para ser enviado fragmentado en varios paquetes IP si fuera necesario. Hace posible enviar un mismo paquete a varios

destinos debido a que admite direcciones IP de broadcast.

La cabecera UDP tiene el formato que se muestra en la Figura 17:

+	Bits 0 - 15	16 - 31
0	Puerto origen	Puerto destino
32	Longitud del Mensaje	Suma de verificación
64	Datos	

Figura 17: Datagrama UDP

La cabecera de los datagramas UDP consta de cuatro campos tiene una longitud fija de 64 bits[23]:

•Puerto origen : campo opcional de 16 bits, por lo que el rango de valores válidos va de 0 a 65535. Cuando es significativo, indica el proceso de emisión. En caso de no ser usado, el puerto origen debe ser puesto a cero.

•Puerto destino: campo obligatorio de 16 bits,por lo que el rango de valores válidos va de 0 a 65535.

•Longitud del mensaje: campo obligatorio que indica el tamaño en bytes del datagrama UDP incluidos los datos a transmitir. El valor mínimo es de 8 bytes.

•Suma verificación: campo opcional de 16 bits que abarca una pseudo-cabecera IP con la dirección IP origen y destino, el protocolo, la longitud del paquete UDP, la cabecera UDP, los datos y ceros hasta completar un múltiplo de 16. Controla la integridad del datagrama [23]
La pseudo-cabecera IP tiene el siguiente formato (véase Figura 18):

bits	0 – 7	8 – 15	16 – 23	24 – 31
0	Dirección Origen			
32	Dirección Destino			
64	Ceros	Protocolo	Longitud UDP	
96	Puerto Origen		Puerto Destino	
128	Longitud del Mensaje		Suma de verificación	
160	Datos			

Figura 18: Pseudo-Cabecera IP

El protocolo UDP permite la creación de un interfaz por el usuario, dando las siguientes funcionalidades:

•Creación de nuevos puertos de recepción.

•Operaciones en los puertos de recepción que devuelvan los octetos de datos así como una

indicación del puerto de origen y de la dirección IP de origen.

•Operaciones en los puertos de transmisión que permita enviar un datagrama, especificando los datos ,los puertos de origen y de destino y las direcciones IPs las que se quiere enviar la información.

Las aplicaciones principales del protocolo UDP son las siguientes:

•Streamming de vídeo y audio en tiempo real donde no es posible realizar retransmisiones por los requisitos de retardo.

•Su uso principal es para protocolos como TFTP (Trivial File Transfer Protocol) , DNS (DNS Domain Name System), RPC (Remote Procedure Calls), NFS (Network File System), NCS (Network-based Call Signaling) y SNMP (Simple Network Management Protocol) en que el intercambio de datagramas de la conexión y desconexión no son rentables respecto a la información a transmitir.

TCP

TCP (Transmission Control Protocol) es el protocolo orientado a la conexión de la capa de transporte, dedicado a proveer comunicación segura a las capas superiores. Previa a la transferencia de datos es necesario el establecimiento de una conexión. El protocolo TCP consta de tres fases:

•Primera fase: establecimiento de la conexión.

•Segundo fase: transferencia de los datos.

•Tercera fase: liberación de la conexión.

En la primera imagen se ve el establecimiento de la conexión mientras que en la segunda la liberación de la conexión (véase Figura 19).

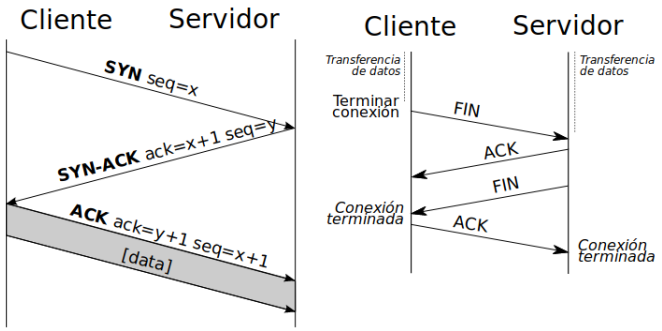

Figura 19: Establecimiento y liberación de una conexión TCP

Se destacarán las siguientes características: ordenación de los datagramas IP, control del flujo de datos de la red y evitando la saturación del buffer en recepción. Asimismo hace posible una comunicación ”agradable” como se veía con anterioridad y permitirá la multiplexación de datos de las capas superiores[24]. El datagrama mostrado en la Figura 20 indica el formato TCP formado por los siguientes campos:

+	Bits 0 - 3	4 - 7	8 - 15	16 - 31
0	Puerto Origen			Puerto Destino
32	Número de Secuencia			
64	Número de Acuse de Recibo (ACK)			
96	longitud cabecera TCP	Reservado	Flags	Ventana
128	Suma de Verificación (Checksum)			Puntero Urgente
160	Opciones + Relleno (opcional)			
224	Datos			

Figura 20: Datagrama TCP

•Puerto origen : campo que identifica el puerto transmisor. Tamaño de 16 bits, rango de valores válidos va desde 0 hasta 65535.

•Puerto destino : campo que identifica el puerto receptor. Tamaño de 16 bits, rango de valores válidos va desde 0 hasta 65535.

•Numero de secuencia : campo de 32 bits, cuya función es comprobar que ningún datagrama se ha perdido y que llegan en orden. Su significado varía según valor de SYN:

•Numero de acuse de recibo (ACK) : campo de 32 bits, si este flag esta activado, contiene el valor de siguiente número de secuencia que se espera recibir.

•Longitud de la cabecera TCP: campo de 4 bits, identifica la longitud del datagrama TCP. La longitud se mide en palabras, cada una tiene un tamaño de 32 bits, oscilando entre un mínimo de 160 bits(5 palabras) y un máximo de 480 bits(15 palabras).

• Reservado: campo de 4 bits, su valor debe ser cero.

•Flags: campo de 8 bits. Cada bit identifica un flag, cada flag indica con 1 si esta activado o con 0 si esta desactivado.Los flags son los siguientes:

- CWR: Congestion Window Reduced Se utiliza para el control de la congestión en la red.
- ECE: ECN-Echo Se utiliza para el establecimiento de la conexión.
- URG: Urgent Pointer field significant
- ACK: Acknowledgment field significant
- PSH: Push Function Se utiliza para que el buffer de recepción transfiera los datos a la aplicación lo antes posible.
- RST: Reset the connection termina la conexión sin esperar respuesta.
- SYN: Synchronize sequence numbers
- FIN: No more data from sender, Si esta activado indica que no hay más datos a enviar por parte del emisor.

•Ventana: campo de 16 bits, indica el tamaño en bytes de la ventana en recepción.

•Checksum(Suma de verificación): campo de 16 bits , complemento A1 que comprueba si hay errores tanto en la cabecera como en los datos.

•Puntero Ugente: campo de 16 bits que indica el valor del tamaño de bytes urgentes.

Las aplicaciones principales del protocolo TCP son las siguientes:

•TELNET : acceso participativo a un terminal remoto.

• FTP : para transferencia de archivos a alta velocidad, entre sistemas conectador por TCP

• SMTP: usado para el envio de correos electrónicos.

3.2 Hardware

3.2.1 Transceptor PMR

Se emplea un PMR estándar tipo Twintalker 4810, se trata de un aparato transmisor-receptor de radiocomunicación portátil PMR446 de baja potencia, normalmente conocido como walkie-talkie. Su alcance máximo es 8 km en campo abierto. No tiene más coste de funcionamiento que la recarga de la batería. El Twintalker funciona en 8 canales de radiofrecuencia con el método PTT (Push To Talk), sirve para hablar en canales half-dúplex de comunicación, presionando un botón para transmitir y soltándolo para recibir. La siguiente tabla recordará las características técnicas necesarias a entender para el desarrollo de la estación remota de comunicaciones.

Tabla 12: Tabla características técnicas Twintalker 4810

Número de canales	8	**Separación de canales**	12,5 KHz
Número de subtonos CTCSS	38	**Número de subtonos DCS**	83
Rango de frecuencia	446,00625 MHz a 446,09375 MHz	**Potencia de transmisión**	=<500 mW ERP
Alcance en campo abierto	8 km	**Alcance en ciudades**	2 km
Tipo de modulación	FM - F3E	**Alimentación**	4 pilas NiMH AAA recargables

3.2.2 Raspberry Pi 3 B+

En general, el ordenador a utilizar en aplicaciones SDR es un ordenador convencional, la apuesta de este proyecto es que el procesado de la señal sea realizado por un Raspberry Pi 3 B+. En este apartado se explicará la definición de Raspberry Pi, su contexto histórico, así como hardware y software y las limitaciones de este miniordenador.

El concepto es el de un ordenador sin accesorios formado por una placa computadora de bajo coste que soporta los componentes necesarios en un ordenador común. El proyecto Raspberry Pi fue ideado en 2006, pero no fue lanzado al mercado hasta febrero de 2012. Se desarrolló en la Universidad de Cambridge y su objetivo era crear un ordenador con la intención de animar a los niños a aprender informática de una manera sencilla y divertida. En 2009 fue creada la fundación Raspberry Pi en Caldecote, South Cambridgeshire, Reino Unido por Eben Upton administrador de la fundación, y David Braven, ideólogo del proyecto. Se mostrarán las fechas más destacadas en lo que se lleva de historia:

•Raspberry Pi 1 Modelo A primer modelo de Raspberry, sus ventas comenzaron en el año 2012.

•Raspberry Pi 1 Modelo B y B+ variante del Modelo A, trajo consigo diversas mejoras: inclusión del doble de memoria RAM, pasando de 256MB a 512MB.

•Raspberry Pi 2 Lanzada en 2014 es el primer modelo que no incluye el mismo procesador que los anteriores, se sustituye por el modelo BCM2836. El procesador pasa a ser de cuatro núcleos y la frecuencia del procesador a 900MHz.

•Raspberry Pi 3 Modelo B Surge en el año 2016, renueva procesador, una vez más un

Quad-Core, pasa de 900MHz a 1.20GHz y mantiene la RAM en 1GB.

•Raspberry Pi 3 Model B+ Último diseño, apareció en marzo del 2018 para actualizar el modelo anterior la Raspberry Pi 3 Model B. Entre sus mejoras cuenta con un nuevo procesador y mejor conectividad, y pasa de tener una frecuencia de procesador de 1.2GHz a 1.4GHz. Incluye conexión Bluetooth Low Cost[25].

Una diferencia muy importante entre la Raspberry Pi y los ordenadores convencionales, aparte de su tamaño y su coste, es el sistema operativo que utiliza. La mayoría de los ordenadores de sobremesa y portátiles disponibles hoy en día funcionan con alguno de estos dos sistemas operativos: Microsoft Windows o Apple IOS. El Raspberry Pi , por el contrario, usa principalmente sistemas operativos GNU: Raspbian. Este sistema operativo es una distribución derivada de Debian que está optimizada para el hardware de Raspberry Pi, se lanzó en julio de 2012.

La Raspberry Pi 3 Modelo B + es el último producto de la gama Raspberry Pi 3, con un procesador de cuatro núcleos de 64 bits que funciona a una frecuencia de 1.4 GHz, doble banda IEEE 802.11.b/g/n/ac LAN inalámbrica a 2,4 GHz y 5 GHz, Bluetooth 4.2 / BLE y conexión Ethernet más rápida. El Raspberry Pi 3 Modelo B + mantiene el mismo diseño físico que los modelos Raspberry Pi 2 B y la Raspberry Pi 3 B. En los siguientes párrafos se describirá con detalle las especificaciones técnicas del modelo Raspberry Pi 3 B+. Las dimensiones físicas de este microordenador son las que se muestran en la Figura 21:

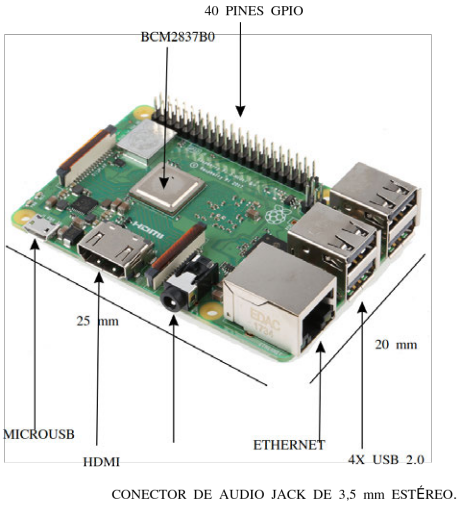

Figura 21: Raspberry Pi 3 B+

•Dimensiones: 25 mm x 20 mm x 9 mm.
•Peso: 59 g
•Temperatura: 0 ^{o}C a 50 ^{o}C

Componentes [26]:
• SoC: BCM2837B0

Este es el chip Broadcom utilizado en los modelos Raspberry Pi 3B + y 3A +. La arquitectura del BCM2837B0 es idéntica al chip BCM2837A0 utilizado en versiones anteriores del Pi. El hardware del núcleo ARM es el mismo aunque la frecuencia del procesador es ligeramente mayor. Los 4 núcleos ARM pueden funcionar hasta 1.4 GHz, lo que hace que el 3B + y 3A + sea un 17% más rápido que el Raspberry Pi 3. La GPU (Graphics Processing Unit) utilizada por el SoC es el VideoCore IV y funciona a 400MHz, decodificará a 1080p30 y traslada contenidos con calidad Blu-Ray. El núcleo ARM es de 64 bits mientras que el VideoCore IV es de 32 bits. El chip BCM2837B0 está empaquetado de manera diferente al BCM2837A0, lo más destacado es que incluye un disipador de calor para mejores térmicas que permite frecuencias de reloj más altas o un funcionamiento a voltajes más bajos para reducir el consumo de energía. Por otra parte el disipador de calor permite una monitorización y control más precisos de la temperatura del SoC.

• Memoria: 1GB LPDDR2 SDRAM (Synchronous Dynamic Random-Access Memory)

La RAM es de 1GB LPDDR2 de SDRAM , en un único módulo, situada encima del SoC. La SDRAM tiene unas dimensiones físicas de 32 mm x 32 mm. Se necesita alimentar con una tensión de 1,2 V.

• Almacenamiento: ranura micro SD para cargar el sistema operativo y almacenar datos.

La Raspberry Pi dispone de una ranura para tarjetas micro SD, no dispone de un disco duro. El arranque del sistema operativo se realiza desde la propia tarjeta SD (Secure Digital), debido a que tiene que albergar todo el sistema operativo, es necesario que la tarjeta tenga capacidad de al menos 2 GB para almacenar todos los archivos requeridos. En el Anexo II, se explicara el proceso de flasheo de la tarjeta SD e instalación del sistema operativo en la Raspberry Pi. Es necesario instalar un sistema operativo para poder empezar a utilizar este ordenador. En este proyecto se ha decidido por la instalación del sistema operativo Raspbian. Siendo el más adecuado para nuevos usuarios de este micro-ordenador.

•Fuente de alimentación: entrada de corriente continua de 5 V y 2.5A

La energía le llega mediante un conector microUSB estándar de 5V. Mucho de los cargadores actuales servirán para alimentar a la Raspberry pi pero la corriente dada por ellos aveces es insuficiente para un correcto funcionamiento . Dependiendo de los periféricos que se quiera conectar puede ser insuficiente. El modelo B+ usa entre 700 mA a 1000 mA dependiendo de qué periféricos sean conectaos. Una incorrecta alimentación, podría suponer aumentos de temperatura y por lo siguiente un bajo rendimiento. Los requisitos de potencia de la Raspberry Pi dependen del uso de las distintas interfaces de comunicación de la Raspberry Pi. Los pines GPIO pueden usar hasta 50 mA de forma segura, la corriente se distribuye en los 40 pines, un pin GPIO individual solo puede extraer hasta 16 mA. El puerto HDMI (High-Definition Multimedia Interface) utiliza 50 mA de alimentación, el módulo de la cámara requiere 250 mA y los teclados y ratones pueden tomar desde 100 mA hasta 1000 mA.

Con la elaboración de los primeros diseños de la estación remota de comunicaciones en la Raspberry Pi 3 B+ aparece en la esquina superior derecha un símbolo de un rayo amarillo. Los motivos son los siguientes[27]:

•Potencia insuficiente de alimentación.

•Temperatura muy alta de la CPU entre 80 y 85ºC.

•Temperatura extrema en la CPU o GPU.

La solución tomada fue la compra de un cargador de 3A de alimentación.

•Multimedia:

- HDMI
- Puerto DSI (Display Serial Interface) para conectar una pantalla táctil al Raspberry Pi
- Puerto de cámara CSI para conectar una cámara al Raspberry Pi
- Conector de audio Jack de 3,5 mm estéreo.

•Conectividad

- 2,4 GHz y 5 GHz IEEE (Institute of Electrical and Electronics Engineers) 802.11.b/g/n/ac inalámbrico LAN
- Bluetooth 4.2
- Gigabit Ethernet (máximo 300 Mbps)
- 4 puertos USB 2.0
- Entrada GPIO de 40 pines

Posee un conector de GPIO de 40 pines a 3,3 voltios de alimentación. Los pines GPIO se pueden configurar como entrada de propósito general, salida de propósito general o como una de hasta seis configuraciones alternativas, cuyas funciones dependen del pin , se puede programar por el usuario en tiempo de ejecución. Todos los pines GPIO vuelven a la configuración general cuando se reinicia la raspberry pi. Se puede configurar como una fuente de interrupción para el SoC.

Por otra parte posee 4 puertos USB 2.0 que permiten la conexión de periféricos como son teclados, ratones, cámaras web que proporcionan al Pi una funcionalidad adicional. Con carácter general todos los dispositivos compatibles con Linux son posibles de usar con el Pi. Los puertos USB de una Raspberry Pi tienen una intensidad de carga de 100 mA cada uno, suficiente para manejar dispositivos de ”bajo consumo” como son ratones y teclados. La Raspberry Pi no puede comunicarse correctamente con dispositivos que

tengan un conector USB 3.0. Las SDR RTL y Adalm Pluto SDR se conectarán a la Raspberry Pi 3 Modelo B + haciendo uso de los puertos USB 2.0. Se recuerda que la SDR RTL funciona hasta 2.4 MSps y la PlutoSDR hasta 61.44Msps, ambos valores son soportados por la Raspberry Pi 3 Modelo B +.

Se finalizará este apartado realizando una comparación del modelo Raspberry Pi 3 B+ con Raspberry Pi B y el ordenador personal a modo de tabla explicativa y se entenderá que limitaciones puede conllevar usar una Raspberry Pi (véase Tabla 13):

Tabla 13: Tabla comparativa de los ordenadores usados en este proyecto

	Raspberry Pi B	**Raspberry Pi 3 B+**	**Toshiba Sattelite**
CPU	ARM 1176JZFS 32 bits	Cortex-A53 (ARMv8) 64 bits	i5-3230M 64 bits
Frecuencia de reloj	700 MHz	1.2 GHz	2.6GHz
Memoria	256MB	1GB	8GB
Ethernet	10 Mbps	300 Mbps	1000 Mbps
Dimensiones	85,6 x 53,98 x 17 mm	25 x 20 x 9 mm.	377 x 244 x 30 mm

La tabla muestra como el modelo actual supera al inicial notablemente. Las prestaciones dadas por la Raspberry Pi 3 B+ son inferiores a las que se obtienen con el ordenador personal. Esto implica que los diseños realizados en el portátil personal pueden no funcionar de manera correcta en la Raspberry pi y se destacan las siguientes limitaciones:

•Menor procesador y memoria RAM que un ordenador convencional.

•Se produce calentamiento al no tener sistema de refrigeración, dando lugar a menor procesado por parte de la CPU.

Según [28] *" La Fundación Raspberry Pi dice que el rendimiento general del Raspberry Pi es comparable con un PC (Personal Computer) con un procesador de 300 MHz Pentium 2, que es posible que haya comprado a mediados y finales de los noventa, con la excepción de que el Raspberry Pi tiene mucho mejores gráficos."*El rendimiento de la estación remota de comunicaciones sera uno de los puntos críticos a estudiar.

3.2.3 SDR

A continuación se presentan las dos radios definidas por software usadas en este proyecto fin de carrera: SDR-RTL y ADALM-PLUTO SDR.

Se recuerda que el SDR-RTL es un periférico radio USB muy económico. Sólo se puede utilizar como receptor radio, basado en software para recibir señales de radio en su área sin necesidad de utilizar internet. Según el modelo se sintonizarán señales radio en el margen de frecuencias de 25 Mhz hasta 1,75 GHz. En el capitulo anterior se mostró en detalle las especificaciones técnicas. A modo de recordatorio se mostrarán las especificaciones técnicas más relevantes de cara al proyecto en los párrafos posteriores

•Modo de funcionamiento: Receptor

•Conversor ADC: 8 bits de resolución de la señal IQ en BB pero el número efectivo de bits se estima en 7

•Frecuencia de muestreo : La frecuencia máxima de muestreo es de 3.2 Mbps. Aunque es inestable a este frecuencia de muestreo y puede funcionar de manera errónea. La frecuencia máxima de muestreo que no elimina muestras es de 2.4 Mbps.

• Rendimiento y relojes

  •Cristal interno trabaja a una frecuencia de 28,8 MHz. En la práctica es posible usar hasta un 80% del ancho de banda.

  •Precisión en frecuencia: 1ppm

•Nivel de cuantización : 43 dB

•Rendimiento RF

  •Figura de ruido de la antena 1.5dB

  •Cobertura de radiofrecuencia: 25 Mhz a 1,7GHz

Por otra parte también se recuerda que el ADALM-PLUTO SDR es un nuevo SDR que funciona tanto en recepción como en transmisión de Analog Devices que es un gran fabricante de semiconductores. El PlutoSDR da cobertura radio desde 325MHz hasta 3800 MHz, tiene un ADC de 12 bits con una frecuencia de muestreo de 61.44 Mbps y un ancho de banda de 20 MHz.Diseñado para estudiantes se puede utilizar tanto para la investigación como para el aprendizaje de Radiofrecuencia y comunicaciones de una manera más empírica. En el capitulo anterior se mostró en detalle las especificaciones técnicas. A modo de recordatorio se mostrarán las más relevantes en los párrafos posteriores

•Modo de funcionamiento: Receptor y transmisor en half or full duplex.

•Cobertura RF de 325MHz a 3.8GHz

•Ancho de banda :20MHz

•Conversor ADC/DCA : 12 bits

•Frecuencia de muestreo: 61.44 Mbps

•Rendimiento RF

  •Rango de sintonización: 325MHz a 3800MHz

  •Salida de potencia Tx: 7dBm

  •Figura de Ruido Rx: 3.5dB

  •Precisión de frecuencia: 25ppm

4 Desarrollo Radio Software

El primer paso antes de desarrollar el sistema de comunicaciones por partes es una explicación de los bloques GNU Radio disponibles y la realización de una simulación en GNU Radio Companion de la transmisión recepción PMR 446 (véase Figura 22). El fin es plantear un modelo teórico sin las SDR. El único hardware utilizado es el ordenador personal, no obstante se tendrá en mente características técnicas de las SDR y los Twin Talker. Para empezar se muestra un diagrama de bloques para entender las etapas que conforman la simulación TX/RX PMR 446. Se empezará con una explicación de todos los bloques Gnu Radio a usar en este proyecto fin de grado.

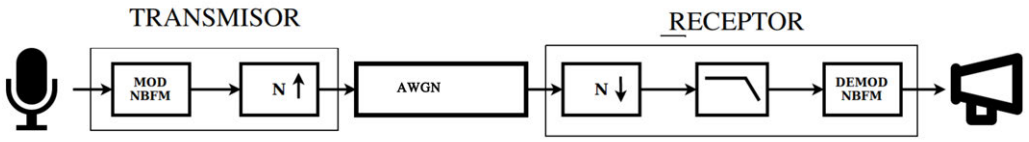

Figura 22: Diagrama de bloques TX/RX PMR 446

4.1 Bloques GNU Radio Companion

En las próximas páginas se explicará en detalle los bloques a usar en el desarrollo software de la estación remota de comunicaciones así como sus propiedades. Todos los bloques utilizados son ”Synchronous” es decir existe el mismo número de puertos de entrada que de salida. Se recuerda que los bloques de GNU Radio se pueden dividir en tres familias: Bloques de procesado de la señal ,Sources y Sinks.

El primer grupo de bloques son los bloques de procesado de la señal, se dividen en las siguientes tres categorías: Modulaciones, Filtrado y Funciones

- Bloques de Modulaciones
  - Bloque NBFM Receive: bloque que demodula la señal modulada en frecuencia de banda estrecha el flujo de datos de la entrada.
  - Bloque NBFM Transmit: bloque que modula en frecuencia de banda estrecha el flujo de datos de entrada.
  - Bloque WBFM Receive: bloque que demodula la señal modulada en frecuencia de banda ancha el flujo de datos de entrada.
- Bloques de Filtrado
  - Bloque Low Pass Filter: define un filtro paso bajo basado principalmente en los parámetros de un ancho de banda de paso y un ancho de banda de transición.
  - Bloque Rational Resampler: este bloque se utiliza para convertir una frecuencia de muestreo de entrada a otra de salida siempre que puedan relacionarse mediante la siguiente

ecuación matemática:

$$Fs_{out} = Fs_{in}\ x\ (Interpolation/Decimation) \tag{4.1}$$

.

Se debe tener en cuenta los bloques que siguen a este bloque en el diagrama de flujo deben tener la frecuencia de muestreo de salida del bloque.

• Bloques de Funciones

• Multiply Const: implementa la siguiente función, siendo K una constante numérica :

$$y = K * x \tag{4.2}$$

• Bloque Channel Model. Este bloque implementa una simulación de modelo de canal básico ruidoso que permitirá evaluar, probar señales y formas de onda. Permite al usuario configurar el voltaje de una fuente de ruido AWGN (ruido gaussiano blanco aditivo) y distintos parámetros que se explicarán posteriormente.

BLOQUES MODULACIONES

La primera categoría es Modulaciones .Gnu Radio da la posible de usar los distintos bloques de FM: Fm Deemphasis, Fm Demod, Fm Preemphasis NBFM Receive, NBFM Transmit, WBFM Receive, WBFM Receive PLL, WBFM Transmit como bloques FM.

La Figura 23 muestra los bloques de procesado NBFM Receive y WBFM Receive

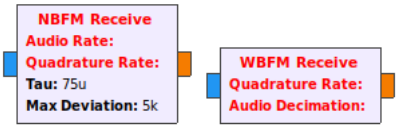

Figura 23: Bloques de procesado NBFM y WBFM Receive

Las propiedades del bloque Narrow Band Fm Tramsmit y Narrow Band FM Receiver son las siguientes:

•Audio Rate: régimen binario que entra al bloque NBFM.

•Quadrature Rate: régimen binario que sale del bloque NBFM

•Tau : constante de deemphasis en el tiempo.

•Max Desviation: desviación máxima en frecuencia.

•Preemphasis High Corner Freq : párametro de preemhasis de frecuencias.

Las propiedades del bloque Wide Band FM Receive son las siguientes:

•Quadrature Rate: frecuencia de muestreo a la salida del bloque.

•Audio Decimation: reduce N veces la frecuencia de muestreo de la tasa de muestras de salida.

La segunda categoría en el grupo de bloques de procesado de la señal son los bloques de filtrado. Se trata de unos bloques con mayor número de parámetros y que guardan una mayor relevancia en el proyecto.

BLOQUE LOW PASS FILTER

La Figura 24 muestra el bloque Low Pass Filter para continuar con sus propiedades:

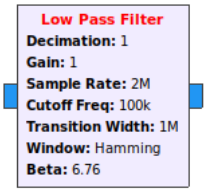

Figura 24: Bloques de procesado Low pass Filter

•FIR Type: especifica el tipo de datos de los flujos de entrada y salida.

•Decimation: parámetro que permite interpola o diezma.

•Gain: parámetro de amplificación del filtro

•Sample Rate: régimen binario del filtro.

•Cutt off freq: frecuencia de corte del filtro.

•Transition Width: banda de paso del filtro.

•Window: especifica el tipo de ventana que se aplicará al filtro . En la Tabla 14 se puede verse las diferentes ventanas.

•Beta: parámetro para la ventana kaiser.

Tabla 14: Ventanas filtros FIR

Window
Hamming
Hann
Blackman
Rectangular
Kaiser

BLOQUE RATIONAL RESAMPLER

La siguiente imagen muestra el bloque Rational Resampler, interpola o diezma la tasa de muestras que recibe en la entrada, sus propiedades son las siguientes:

•Type: especifica el tipo de datos de los flujos de entrada y salida.

•Interpolation: especifica el valor de interpolación.

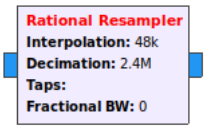

Figura 25: Bloque Rational Resampler

•Decimation: especifica el valor de diezmado.

La ultima categoría usada es funciones.

El bloque Multiply Const tiene las siguientes dos propiedades:

Figura 26: Bloque Multiply Const

•IO Type: especifica el tipo de datos de los flujos de entrada y salida,
son los siguientes: Complex, Float, Int y Short.

• Constant: especifica el valor de la constante.

El Bloque Channel Model tiene las siguientes propiedades:

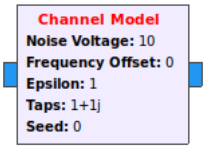

Figura 27: Bloque Channel Model

•Noise Voltage: el nivel de ruido AWGN como voltaje.

•Frequency Offset: desplazamiento de frecuencia normalizado. Un Frequency Offset con valor 0 es sin desplazamiento en frecuencia.

•Epsilon : desplazamiento en tiempo (Timming Offset) de las muestras para emular las diferentes velocidades entre los relojes internos del transmisor y el receptor. Un Epsilon con valor 1.0 implica un desplazamiento nulo en tiempo.

•Taps: emula un perfil de retardo de múltiples canales.

• Noise Seed: un generador de números aleatorios para la fuente de ruido.

El segundo grupo de bloques son los bloques sink, se dividen en las siguientes dos categorías: Audio, GUI (Graphical User Interface)

- Bloques de Audio
  - Audio Sink: representa el hardware de salida de audio dentro del diagrama de flujo. La señal de audio se reproducirá por los altavoces.
  - Wav File Sink. almacena y transforma el audio transmitido a un fichero . wav, para futura escucha si fuera necesario.
- Bloques GUI
  - Bloque FFT Sink: El FFT Sink actúa como un analizador de espectro al hacer una transformada de Fourier (STFT).

BLOQUES DE PROCESADO DE AUDIO

La Figura 28 muestra los bloques Sink Audio Sink y Wav File Sink.

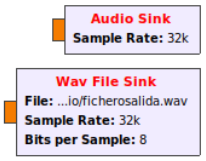

Figura 28: Bloques Audio Sink y Wav File Sink

Audio sink tiene las siguientes propiedades:

•Sample rate: tasa de muestreo a la que se transmite el audio a la tarjeta de sonido, tiene las siguientes tasas de muestreo posibles : valor que se le quiera asignar, 16 KHz, 22,05 KHz, 24 KHz, 32 KHz , 44,1 KHz y 48 KHz.

•Num imputs: el receptor de audio puede tener múltiples entradas dependiendo de su hardware. Establezca las entradas en 2 para tener audio estéreo.

Wav File Sink tiene las siguientes propiedades:

•File: establece la ruta de directorios donde se va guardar el archivo. La ultima palabra es el nombre del archivo seguida de una extensión .wav.

•Num imputs: especifica el número de canales para leer desde el archivo. Los archivos WAV suelen ser mono (un canal) o estéreo (dos canales) que representan dos flujos de datos independientes

•Sample rate: tasa de muestreo a la que se transmite el audio al fichero WAV. La tasa de muestreo del flujo de entrada debe coincidir con este valor.

Por otro lado tenemos los bloques que brindan diferentes posibilidades gráficas como puede ser ver en el tiempo una señal. El bloque usado es el FFT Sink mencionado anteriormente, las propiedades principales son las siguientes:

•Sample rate: tasa de muestreo a la que funciona el analizador de espectros

•BaseBand Freq: frecuencia a la que se centra el analizador de espectros

•Y Per Div: número de decibelios por división

•Y Div: número de divisiones verticales que va a tener el analizador de espectros.

El último grupo de bloques son los bloques Sources. El único bloque de fuente de información usado es Wav File Source. Crea una fuente de datos a partir de un archivo .wav de audio. La siguiente imagen nos mostrará el bloque Wav File Source (véase Figura 29). Las propiedades se explicarán a continuación.

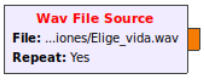

Figura 29: Bloque Wav File Source

•File: establece la ruta de directorios donde se encuentra el archivo a leer.

•Repeat: especifica si se reproduce el archivo en bucle.

•Num imputs: especifica el número de canales para transmitir el flujo de datos desde el archivo. Los archivos WAV suelen ser mono (un canal) o estéreo (dos canales) que representan dos flujos de datos independientes

Entendiendo los diferentes bloques generales que se van a usar en GNU Radio para el diseño de la estación remota de comunicaciones. Se pasará a explicar los bloques particulares y más específicos en el desarrollo de la estación remota de comunicaciones. Estos bloques se pueden dividir en dos grupos :

•SDR
- RTL-SDR Source
- PlutoSDR Source
- PlutoSDR Sink

•Network Tools
- UDP Source
- UDP Sink
- TCP Source
- TCP Sink

BLOQUE RTL-SDR

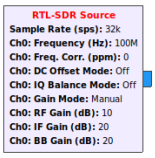

Figura 30: Bloque GNU Radio RTL-SDR Source

La Figura 30 muestra el bloque RTL-SDR Source. Las propiedades del RTL-SDR Source son las siguientes:

•Sample Rate(sps) : frecuencia a la cual el hardware recibirá las muestras.

•Cho : Frequency (Hz): sintoniza la SDR a la frecuencia elegida.

•Cho : Freq.Corr(ppm): corrección en frecuencia en partes por millon.

•Cho : Dc Offset Mode : Automatic Propiedad que realiza la corrección de la desviación de continua en la radio definida por software. Detecta una desviación de DC de la señal y resta la desviación de DC de la señal analógica recibida.

El desequilibrio IQ es un problema que limita el rendimiento en el diseño de receptores de conversión directa, también conocidos como receptores de FI. Esto se produce cuando existe desajustes entre la ganancia y la fase de las dos señales a lo largo de las dos ramas. Las ramas como en la Figura 2: Diagrama de bloques SDR-RTL que poseen mezcladores y diezmadores. Como resultado las señales de banda de base se corrompe. La propiedad IQ Balance se encargará de ajustar la ganancia y fase de las señales IQ •Cho : IQ Balance Mode :Automatic

•Cho: Gain Mode : Automatic o Manual

•Cho: RF Gain(dB) : controla la ganancia en la parte de RF asignando un valor en dB

•Cho: If Gain(dB) : controla la ganancia en la parte de FI asignando un valor en dB

•Cho: BB Gain (dB) : Controla la ganancia en la parte de BB asignando un valor en dB

BLOQUES PLUTOSDR

La Figura 31 muestra el bloque Pluto SDR Sink seguido de sus propiedades[29]:.

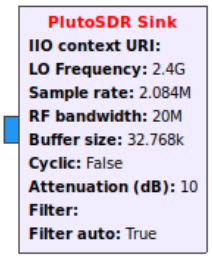

Figura 31: Bloque GNU Radio PlutoSDR Sink

•IIO contex URI:pluto.local. Establézcalo en ”local:” si usa GNU Radio en su ordenador. Si utiliza el control remoto de GNU Radio en una PC, configure la dirección IP de destino usando ip: XXX.XXX.XXX.XXX.

•LO Frequency: sintoniza la SDR a la frecuencia elegida

•Sample rate: frecuencia a la cual el hardware transmitirá las muestras.

•RF bandwith: configura el ancho de banda de la señal a transmitir.

•Buffer size: tamaño del buffer de salida de la SDR .

•Cyclic False: ajústelo a ”true” si se desea el modo ”cíclico”. En este caso, el primer buffer de muestras se repetirá en los canales habilitados hasta que se detenga el programa.

•Attenuation(dB): control de la atenuanción para la transmisión rango de valores desde 0 hasta -89,75dB en saltos de 0.25dB.

•Filter: permite cargar un filtro FIR desde un archivo.

•FIlter auto: False. Si esta en True habilita el filtro anterior.

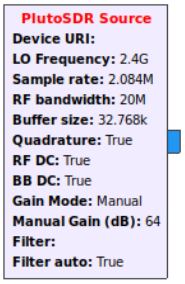

Figura 32: Bloque GNU Radio PlutoSDR Source

Las propiedades del PlutoSDR Source coinciden en su gran mayoría con las del bloque PlutoSDR Sink a excepción de las siguientes:

•Gain Mode: selecciona uno de los modos disponibles de ganancia: Manual slow, hybrid y fast attack. Se explicará cada uno de los modos posibles en detalles[30].

•Manual Gain(dB): controla la ganancia de recepción asignando un valor en dB.

•Slow Attack Mode: diseñado para señales que cambian lentamente como las que se encuentran en algunas aplicaciones FDD, WCDMA y FDD LTE.

•Hybrid Mode: el modo híbrido para el AGC del AD9631 es el mismo que el modo lento AGC, con la excepción de que no se usa el contador de actualización de ganancia.

•Fast Attack Mode: el modo de ataque rápido está diseñado para formas de onda que ”estallan” de forma intermitente, como las que se encuentran en aplicaciones TDD o aplicaciones GSM EDGE FDD. El AGC responde muy rápidamente a las sobrecargas al inicio de una ráfaga, de modo que el AGC puede establecerse en un índice de ganancia óptimo en el momento en que llega la parte de datos de la señal

BLOQUES NETWORKING TOOLS

Por último se explicará los bloques usados como herramientas de red. Permitirán la creación de una red usando la capa de transporte. El bloque UDP Sink escribirá la secuencia en un socket UDP mientras que bloque UDP Source leerá la secuencia de un socket UDP. Por otro parte el bloque TCP Sink escribirá la secuencia en un socket TCP mientras que el bloque TCP Source leerá la secuencia de un socket TCP.La siguiente imagen mostrará los bloques ofrecidos por GNU Radio para el networking (véase Figura 33).

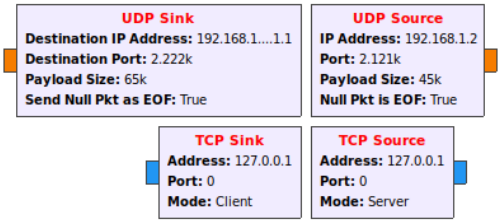

Figura 33: Bloques Networking Tools

Las propiedades de los bloques UDP son:

•Destination IP Address: el nombre o la dirección IP del host receptor.

•Destination Port: puerto destino para conectarse al host receptor.

•Payload Size: el tamaño del datagrama UDP se establece de forma predeterminada en 1472 = (1500 de la trama - (8 bytes del encabezado UDP) - (20 bytes del encabezado IP)).

•Send Null Pkt as EOF: True Parámetro de desconexión cuando se envía el último paquete.

Gnu ofrece los siguientes bloques para crear un servidor TCP: TCP Source y TCP Sink. A continuación se explicará las características de los bloques TCP:

•Address: el nombre o la dirección IP del host receptor.

•Port: puerto destino para conectarse al host receptor.

•Mode:

•Cliente: host transmisor

•Servidor: host receptor

4.2 Desarrollo software Diseños Gnu Radio Companion

4.2.1 Simulación TX/RX PMR 446

Se empezará con el diagrama de flujo TX/RX PMR 446 generado por GNU Radio Companion. El diseño constará de dos partes a explicar. La primera parte explicará las variables usadas y la segunda los bloques GNU Radio explicados en el apartado anterior y su interconexión. La Figura 34 es el diagrama de bloques en una simulación transmisión-recepción PMR 446

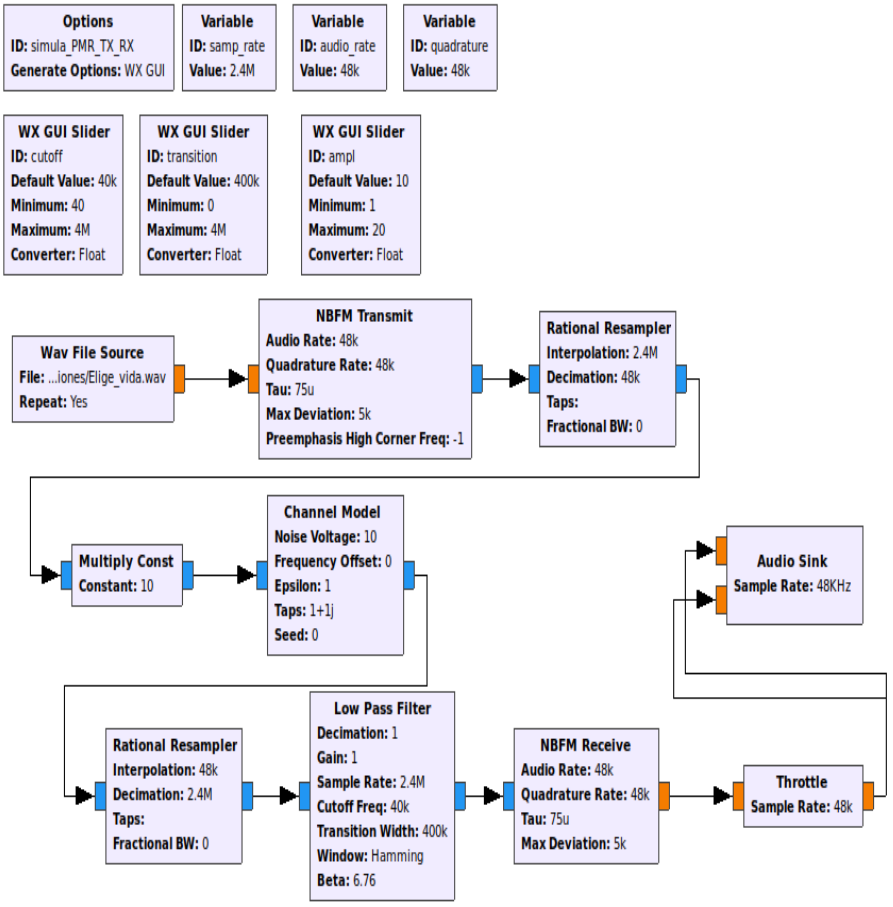

Figura 34: Diseño GRC TX/RX PMR 446

Una idea para realizar este diseño se obtuvo de [31]. Se pueden apreciar numerosos bloques así como numerosas variables y parámetros de configuración. Se explicarán las variables externas al diagrama que aparecen. Éstas están situadas en el margen superior izquierdo de la figura. Se va a trabajar con dos tipos de variables: variables normales y variables Slidder. Las tres primeras tienen un valor inmutable mientras que las tres ultimas pueden cambiar su valor a lo largo de la ejecución. Se recuerda que las variables declaradas aparecerán en la zona derecha del terminal en GNU Radio Companion.

Variables:

• samp_rate: especifica el valor a la cual el hardware recibirá las muestras; 2,4 Msps.

• audio_rate: tasa de muestreo óptima para procesar un audio sin producirse pérdidas; 48 Khz

•quadrature: tasa de muestreo que sale el flujo de datos del bloque NBFM; 48 KHz

•cutoff: variable Slidder, se usará para asignar la frecuencia de corte del filtro paso bajo. El valor por defecto sera 40 KHz , permitirá tomar 100000 valores diferentes en el ancho de banda de 400 Hz a 4 MHz.

•transition: variable Slidder para la banda de paso del filtro. Tomará por defecto el valor de 400 KHz aun cuando se permitirá modificar su valor de ejecución de 0 a 4 MHz.

•ampl: variable Slidder para el valor de amplificación para la señal transmitida al canal de comunicación. Tendrá valores de 0 a 20.

Tras entender todas las variables diferentes del diseño se pasará a explicar el diagrama de flujo. Primero se utilizará como fuente de información un fichero de audio con extensión .wav leído desde el bloque $WavFileSource$ en un bucle continuo. Segundo se utiliza como sumideros del diagrama de flujo dos bloques diferentes: $AudioSink$ y $WavFileSink$. El primero permitirá escuchar el audio transmitido mientras que el segundo transformará y almacenará el audio transmitido a un fichero .wav para futura escucha si fuera necesario.

El bloque $WavFileSource$ se encargará de producir muestras de audio a una tasa de 48 KHz. Son conducidas al bloque $NarrowBandFMTransmit$ que se encargará de modular la señal en frecuencia. Se recuerda que las comunicaciones PMR son comunicaciones punto a punto, por tanto la modulación usada será FM de banda estrecha. Las dos primeras propiedades del bloque tendrán el valor de las variables $audio\_rate$ y $quadrature$. Según la tabla de [32], usaremos el valor de 75us para la Tau. Por último el valor de la máxima desviación en frecuencia es de 5000Hz. Se dejará por defecto la ultima propiedad del bloque. Continuando con el diagrama de flujo, el siguiente bloque, es $RationalResampler$, permitirá interpolar el flujo de muestras a 2.4 Mega muestras por segundo siendo este el valor de muestreo de la radio receptora. Las muestras que salen del Bloque $RationalResamplers$ son conducidas a un bloque $multiplicador$ el cuál se encargará de incrementar el nivel de la señal a transmitir, haciendo uso de la variable $ampl$.

Para que la simulación sea lo más cercana posible a la realidad se inserta un bloque $ChannelModel$, debido a que toda comunicación en la realidad esta influenciada por el ruido.

Tras pasar por el canal se llega a la parte de recepción PMR 446. En primer lugar se ajustará el régimen binario a 48000 Hz haciendo uso de un diezmado en 50 por el bloque $RationalResampler$. Las muestras se conducirán al bloque $LowPassFilter$ En la imagen siguiente se podrá apreciar los valores para el filtro paso bajo. Se han usadolas dos variables slidder $cutoff$ y $transition$ y la escucha del audio para determinar su valores exactos. Un correcto valor para el filtrado es de 40000 Hz de frecuencia de corte y 400KHz de la banda de transición.

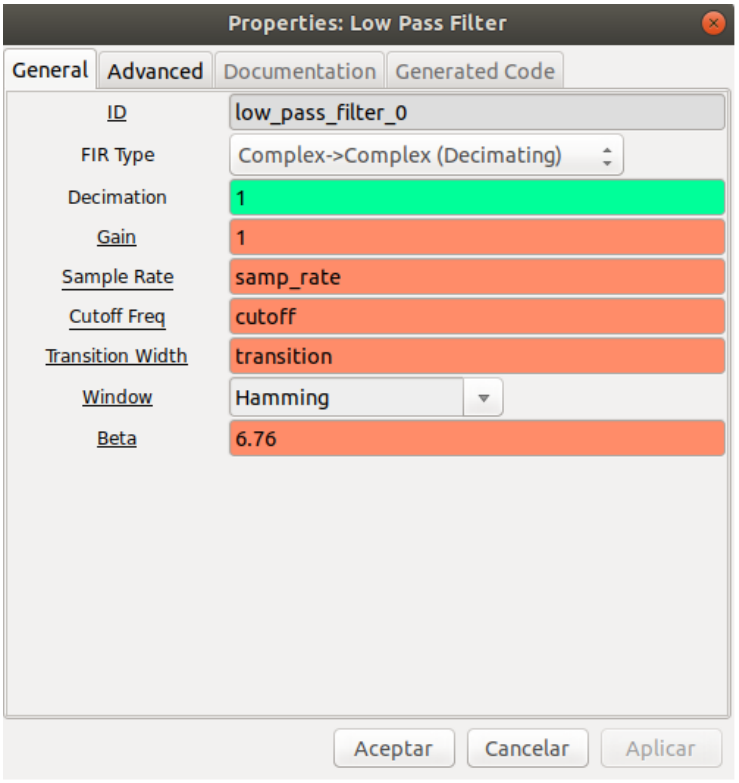

Figura 35: Bloque Filtro paso bajo GNU Radio

El último bloque antes de las Sinks, es el receptor NBFM Receive, con las mismas características que el bloque Narrow Band Fm Tramsmit.Por último se entregan los datos demodulados a las dos sinks.

4.2.2 PMR 446

Una vez analizado en detalle la simulación se procederá al desarrollo de los diseños Recepción PMR 446 y Transmisión PMR 446 (véanse Figuras 37 y 40).

El hardware a utilizar es :

- •Portátil Personal (Toshiba Sattelite)
- • Pluto SDR
- • Twin Talker PMR 446 (Walkie Talkie)

El software a utilizar:

- •GNU Radio

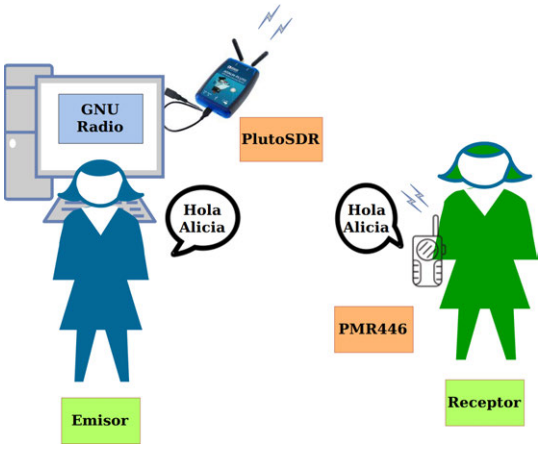

Figura 36: Esquemático TX PMR 446 con SDR en PC

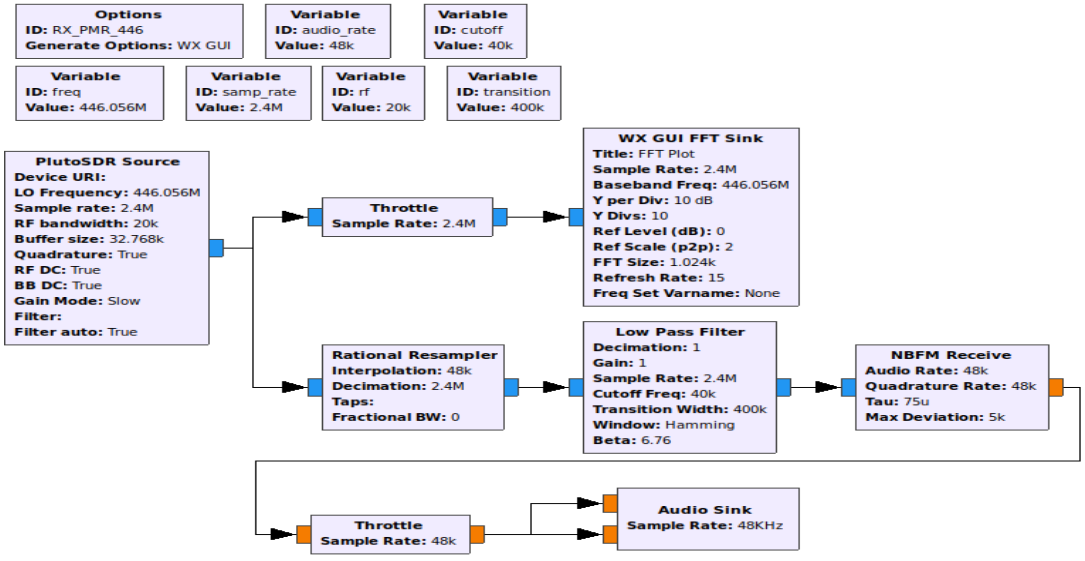

Figura 37: Diagrama de flujo PMR446 RX

La Figura 37 muestra el diseño RX PMR 446. Se pueden apreciar numerosos bloques así como numerosas variables y parámetros de configuración. Primero se explicarán las variables externas al diagrama que aparecen. Éstas están situadas en el margen superior izquierdo de la figura. Las variables Slidder pasarán a ser variables normales tras ajustar su valor gracias al diseño anterior. Las variables utilizadas en el esquemático son las mismas que en el diseño de la simulación a excepción de las nuevas variable freq y rf.

•freq: frecuencia sintonizada en la Pluto SDR. La variable tomará el valor es 446,05625 MHz corresponde con el 5 canal PMR 446.

•rf: Ancho de banda de la Pluto SDR de la señal recibida con un valor de 20 KHz.
Los canales PMR 446 tienen una separación de 12,5 KHz.

En segundo lugar se explicará el diagrama de flujo así como las propiedades de los bloques que lo conforman. Para comenzar esta parte se utilizará como fuente de información el bloque PlutoSDR Source, la información que suministra la SDR Pluto es:

•Lo Frequency: con valor de la variable $freq$ 446056250.

•Sample Rate: con valor 2,4 M por la variable $samp_rate$. La SDR procesará muestras en recepción a dicho valor.

•RF Bandwith: se le asignará un ancho de banda en recepción de 20000 Hz, variable rf.

•Buffer size: 0X8000. Tamaño del buffer de entrada. Se dejá por defecto.

Por lo tanto la función de la SDR será recibir la señal PMR 446 a 2,4 millones de muestras por segundo a la frecuencia de 446.05625e6 con un ancho de banda de 20 KHz, sin aplicar ninguna corrección en frecuencia.

Seguido al bloque Source el diagrama de flujo queda dividió en dos caminos. El primer camino, el superior, lleva las muestras al analizado de espectros pasando por un bloque throttle, Limita el rendimiento de datos a la tasa de muestreo especificada, esto evita que GNURadio consuma todos los recursos de las CPU.

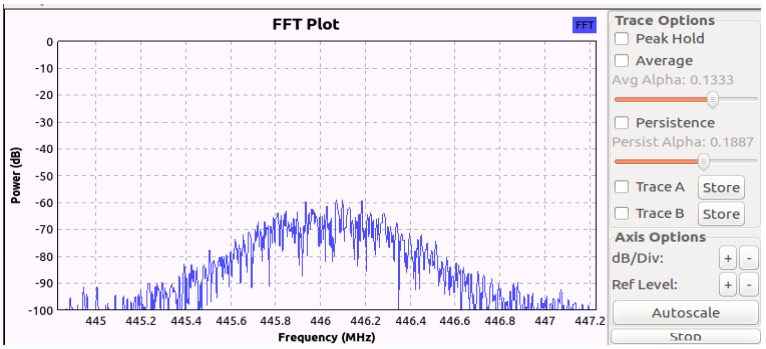

Figura 38: FFT PMRRX 1

La FFT sink centrará el espectro a nuestra frecuencia, tendrá 10 divisiones es decir irá de 0 dB a -100 dB. La primera de las dos imágenes muestra el espectro cuando aun no se ha recibido la señal PMR 446.

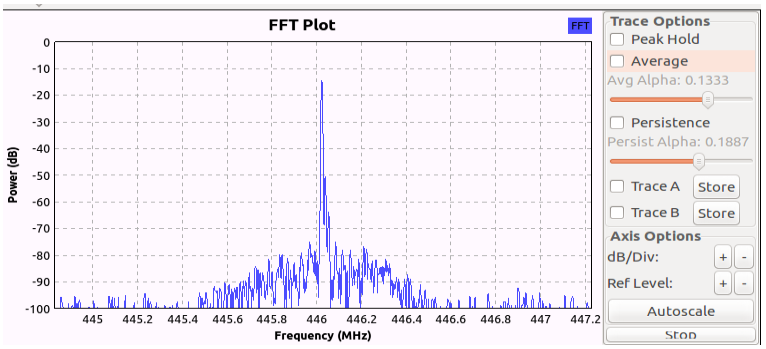

Figura 39: FFT PMRRX 2

La segunda muestra el espectro en frecuencia de la señal PMR 446. Por sus pocas deltas, se saca de nuevo la conclusión,que se trata nuevamente de la modulación en frecuencia de banda estrecha.

En la parte inferior del diagrama se continua el flujo con el bloque $RationalResampler$ utilizado para diezmar el numero de muestras de 2.4M a 48000. A continuación aparece el filtro paso bajo cuya propiedades se mostraron en el apartado anterior. Se usarán las variables $cutoff$ y $transition$ en los parámetros Cuttoff Freq y Transition With. Después de sintonizar el receptor, eliminar parte del ruido y filtrar la señal fm de banda estrecha, se está en disposición de demodular la señal con el bloque NBFM. Los parámetros de este elemento son los mismos que para la simulación realizada en el apartado anterior. Una vez finalizado el flujo de señal se conecta a la sink final que será la tarjeta de sonido, cuyo único parámetro es la tasa de audio, 48 KHz. Permitirá escuchar el audio transmitido por el TwinTalker.

Se destacan las siguientes observaciones:

•Es necesario que los regímenes binarios del flujo estén adaptados correctamente, es decir se recomienda que en procesos de diezmación o interpolación sea por números enteros. En el flujo se pasa de 2.4 millones de muestras a 48000, haciéndose una diezmación en 50. Se recuerda que los flujos binarios son inmutables durante la ejecución del diseño.

•Pérdida del canal de comunicación por los relojes internos de la SDR Adalm Pluto y el portátil personal.

Al ejecutar el diseño de recepción PMR 446 varias veces en el tiempo o incluso a la primera recepción surge en la ventana del terminal el siguiente mensaje auauauaua segun[33] *"au : se llama'audio underruns'. Si su diagrama de flujo tiene la siguiente forma: Radio Hardware − > Demodulator − > Audio Sink ... y ves estos errores, es muy probable que sea un "problema de 2 relojes interno". El problema es que tienes 2 piezas de hardware con un reloj de hardware y no hay forma de alinearlas perfectamente. La imagina del reloj de la radio es un poco más lento de lo anunciado, y el reloj de audio es más rápido. Entonces, invariablemente, en algún momento, no habrá datos suficientes para alimentar el receptor de audio y se agotará. Además, recuerde que GNU Radio baraja los datos en ambas partes, por lo que puede haber suficientes datos, pero no en el momento preciso en que el receptor de audio lo necesita. No hay una solución trivial para esto, pero una solución simple es alimentar los datos del receptor de audio con una tasa ligeramente más alta de lo que está configurado."*

La solución tomada fue reiniciar el sistema receptor de manera automática cada ciertos segundos. Per-

mitirá no tener problemas en recepciones no continuas o pérdida del canal de comunicación por los relojes internos. Para ello tenemos que generar el código Python del diseño GNU Radio. El código generado se incluye en el Anexo III. El nombre del código python es:

```
RX_PMR_446.py
```

La ejecución del código se realiza a través del interprete de comandos en el directorio donde este guardado el fichero. Poniendo Python y el nombre del código. Para reiniciarlo de manera automática se hará uso de un fichero .sh Los archivos sh permite almacenar en un fichero un conjunto de comandos y ejecutarlos de manera automatica. Es decir sin tener que escribirlos manualmente .El siguiente código nos permitirá lanzar nuestro receptor PMR446 de manera automático y reiniciarlo cada 30 segundos. A continuación se explicará cada linea de código:

```
#!/bin/bash
while :
do

sudo python  RX_PMR_446.py &

sudo sleep 30

sudo killall python RX_PMR_446.py

echo "RELANZAMOS LA RECEPCIÖN PMR 446"

done
```

- •While do nos permite estar en un bucle infinito.
- •La siguiente linea ejecuta el receptor PMR 466.
- •$Sleep$ 30, suspende el receptor a los 30 segundos de ejecutarlo.
- •El mandato $Killall$ elimina el proceso del receptor PMR 466.
- •Como estamos en un bucle infinito, se vuelve a ejecutar el receptor PMR446.

TX PMR 446

Tras la realización del diseño receptor se pasara a ejecutar una Transmisión PMR 446. Para ello se usará el hardware y software mencionado con anterioridad. La transmisión PMR 446 consistirá en modular en frecuencia un flujo de audio provisto por un fichero .WAV . La modulación sera una FM de banda estrecha. Seguido de la modulación se realizará una interpolación del numero de muestras para radiar la señal FM a los walkie talkie. La interpolación se realiza por que la SDR Pluto no es capaz de radiar a 48000 muestras por segundo. A continuación se muestra el diseño generado gracias a GNU Radio y las variables usadas.

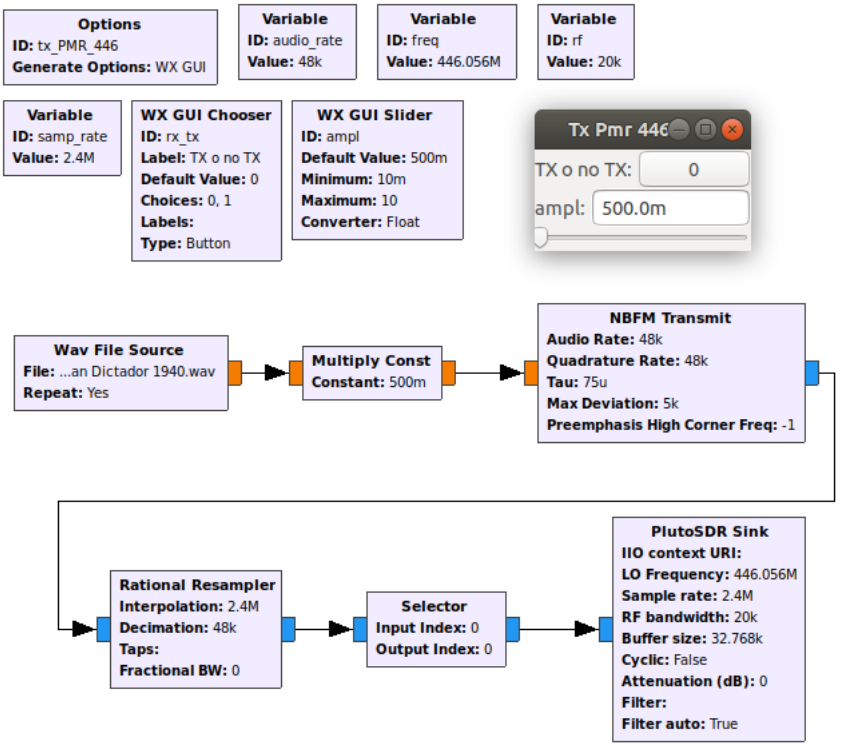

Figura 40: Diagrama de flujo PMR446 TX

En la Figura 40 se pueden apreciar numerosa variables y parámetros de configuración. La W x GUI slider $ampl$, es una variable slider que nos permitirá ajustar un valor de amplificación del volumen del audio a transmitir a la SDR. Tomará valores desde 0.5 hasta 10. La variable inferior siguiente es una variable Chooser. Actúa como multiplexor o desmultiplexor para distintas rutas de la señal. En este diseño al tener solo un flujo su función es activar o desactivar la radiación de la señal PMR 446. Las siguientes variables son:

•audio_rate:

•samp_rate:

•freq:

•rf:

En la figura superior se muestra el diseño para el transmisor PMR446. Tras la fuente de información, Bloque Wav File Source, se encuentra el bloque de procesado Multiply Const. Como se vio en el apartado de bloques GNU Radio se encargará de incrementar el nivel de la señal a transmitir haciendo uso de la variable $ampl$. En este caso se atenuará la señal por 0.5 antes de llegar al bloque NBFM Transmit. Las propiedades son las mencionadas previamente en la simulación. Las muestras son conducidas al bloque Rational Resampler para una interpolación a 2,4 millones de muestras, tasa binaria de transmisión de la SDR. Previo al bloque PlutoSDR Sink se encuentra el Selector que con la variable Chooser permitirá detener o activar el flujo de muestras. Para finalizar el bloque Pluto SDR Sink radiará la señal PMR 446 según las siguientes propiedades con sus correspondientes valores:

•LO Frequency: Selecciona la frecuencia sintonizada en este caso 446,05625 MHz.

•Sample rate: 2400000. Frecuencia a la cual el hardware transmitirá las muestras.

•RF bandwith: Configura el ancho de banda de la señal a transmitir, en este caso 20 KHz.

•Buffer size: 0X8000. Tamaño del buffer de salida . En futuros apartados se modificará esta propiedad para la transmisión con la Raspberry Pi. Por el momento se deja por defecto.

Encendiendo uno de los Twin talker, situándonos en el canal 5 de comunicación y desactivando los canales CTCSS se escuchará el audio transmitido con calidad y precisión. De esta forma podremos ajustar aun más detalladamente los parámetros mencionados. Por norma general es mucho más fácil implementar diseños de transmisión que recepción. En el diseño final de la estación remota se verá esta dificultad.

4.2.3 ESTACIÓN REMOTA DE COMUNICACIONES

Los apartados hasta ahora presentados se basan en la interacción de la herramienta software GNU Radio con las radios definidas por software. La herramienta GNU Radio ha brindado la posibilidad de realizar los diseños PMR 446 por separado. Se había trabajado únicamente con el portátil personal como plataforma de procesado de la señal. El último paso en la elaboración del diseño es la inserción de la Raspberry Pi 3 B+. Se comentaba en el apartado anterior que los diseños pueden no funcionar correctamente en este hardware. Por consiguiente los diseños TX PMR446 y Rx PMR446 se verán modificados para un correcto funcionamiento en la estación remota de comunicaciones. El diagrama inferior dará una visión esquemática del futuro diseño y los componentes hardware que lo conforman.

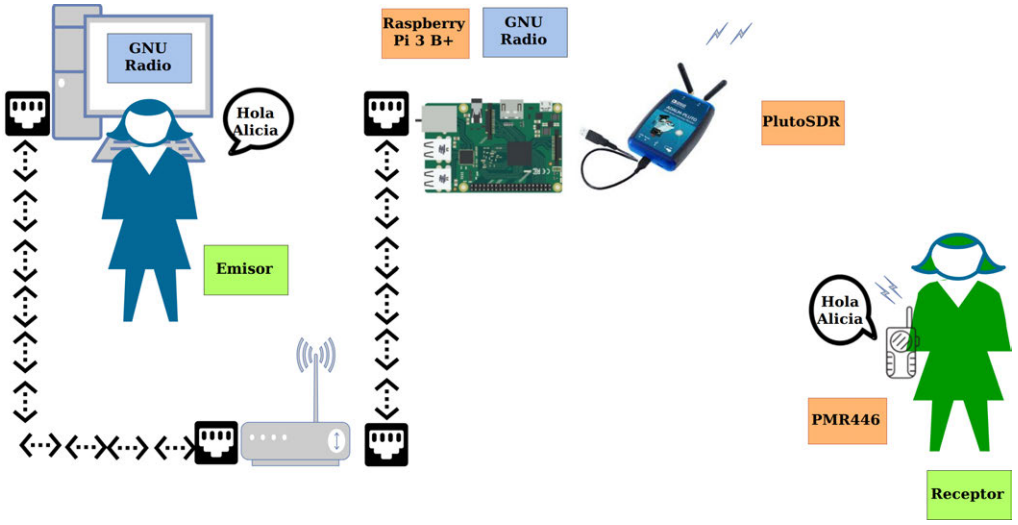

Figura 41: Diagrama estación remota de comunicaciones

El Hardware a utilizar es el siguiente:

•Portátil Personal

•SDR ADALM PLUTO

•TWINTALKER 4810

•Raspberry Pi 3 B+

•Cables ethernet x 3

•Switch

El Software a utilizar es:

•GNU Radio

Tras citar los componentes hardware (véase Figura 41) que forman la estación remota y el software a usar en el diseño de la estación remota de comunicaciones se explicará su modo de funcionamiento brevemente:

•Transmisión:

El portátil personal mandará un flujo audio realizando una lectura de un fichero .wav. Un socket transmisor UDP o TCP recibirá el flujo de audio y lo enviará al servidor receptor UDP o TCP destino. El servidor se encuentra en la Raspberry Pi 3 B+. La comunicación entre Sockets se realizará vía IP. Por otra parte la SDR Adalm Pluto estará conectada a la a Raspberry Pi 3 B+ electro mecánicamente. El audio recibido por la Raspberry Pi 3 B+ será modulado en FM de banda estrecha para radiarlo a los walkie talkie por la SDR Adalm Pluto como una señal PMR 446. El Walkie talkie situado en el canal N y sin subtonos CTCSS o DCS escuchará el audio mandado por el portátil personal.

•Recepción:

Se transmitirá un audio por medio de un walkie talkie con el método PPT a la Pluto SDR que se encuentra conectada electro mecánicamente a la Raspberry Pi 3 B+. La Raspberry Pi 3 B+ se encargará de procesarla señal PMR 446 sintonizada por la Pluto SDR. En apartados anteriores era un ordenador convencional el encargado de esta función. La Raspberry Pi 3 B+ por medio del software GNU Radio diezmará, filtrada y demoludará en NBFM la señal recibida. El audio filtrado y demodulado será enviado a un socket receptor UDP o TCP situado en en el portátil personal. El audio será escuchado en el portátil personal, realizando así una recepción PMR 446 remota.

Los diseños GNU Radio realizados a continuación suponen un ahorro económico debido a que la estación remota es la Raspberry Pi 3 B+ y no un ordenador convencional. Esta estación remota de comunicaciones supone un gasto menor en equipos hardware además de en espacio. El sistema operativo Raspbian permite trabajar de manera similar como en UBUNTU. Solo es necesario introducir en el interprete de mandatos del Raspbia el siguiente comando para ejecutar el GNU Radio Campanion.

```
sudo gnuradio-companion
```

La instalación de la librería SDR Adalm Pluto en GNU Radio es idéntica al apartado 2.2.1 Ubuntu y Raspbian son similares a la hora de instalar paquetes, repositorios y librerías además de contar con un interfaz gráfico semejante. No supuso ningún problema a la hora de instalar los requerimientos y hacer pruebas con el GNU Radio Companion. Una vez configurado el Raspbian se paso a la ejecución del diseño TX PMR446 como en la Figura 40: Diagrama de flujo PMR446 TX. La ejecución fue errónea, el único audio recibido por el TwinTalker fue un pitido agudo y molesto.

La manera de resolver el problema fue con el método prueba y error, se trata de un método heurístico. Este procedimiento consiste en probar una alternativa y verificar si funciona. Se volvió a hacer uso de las variables Slidder para modificar los parámetros del diseño de manera rápida. Variables como $samp\_rate$, rf, no modificaban el funcionamiento erróneo anteriormente mencionado,se seguía escuchando el pitido agudo. La Raspberry pi no era capaz de procesar la señal PMR 446 para su futura transmisión. Al final se consiguió cambiar en el sonido transmitido al aumentar el parámetro $Buffer$ del bloque PlutoSDR Sink. El proceso desarrollado consistió en ir duplicando en cada prueba el valor del buffer. Se empezó con un valor de 0x2000 hasta 0x970000. Se observó que el pitido agudo y constante iba cambiando hacia un sonido intermitente y más grabe en cada prueba. Cuando el tamaño del buffer era 0x500000 el audio se empezó a escuchar de forma intermitente. Con el valor de 0x970000 se conseguía una escucha del audio

clara aunque cada seis segundos se interrumpida la comunicación para reanudarse tres segundos más tarde. En ningún momento se perdía información dado que el restablecimiento de la comunicación continuaba a partir del punto interrumpido.Valores superiores al mencionado sacan error de compilación del GNU Radio Companion.La transmisión de muestras de RF usando la Raspberry Pi requiere bastante recurso de procesado, se tendrá que configurar buffer E/S más grandes en la comunicación USB radios basadas en software con Portátil Personal. Si fuera un ordenador convencional con USB3.0 no sería necesaria esta reconfiguración.

Consiguiendo una solución incompleta en la Transmisión PMR 446. Se planteaba como se podría solucionar las interrupciones mencionadas en el párrafo anterior. La señal se interrumpe por que la Raspberry Pi 3 B+ conectada electro mecánicamente a la Pluto SDR no puede manejar tasa binarias tan elevadas como 2.4 Mbps. La solución fue sencilla, decrementar el valor en 4 veces la tasa binaria por lo tanto la tasa de transmisión de la Pluto SDR seria 600 Kbps.

Para terminar este apartado se muestra en la Figura 42 el diseño final GNU de la estación remota de comunicaciones en transmisión. El funcionamiento se detallará en los párrafos posteriores a la imagen del diseño software. En este caso se ha realizado usando el protocolo UDP. En el capitulo posterior de medidas y resultados se hará un estudio más detallado de ambos protocolos viendo sus ventajas e inconvenientes de manera práctica.

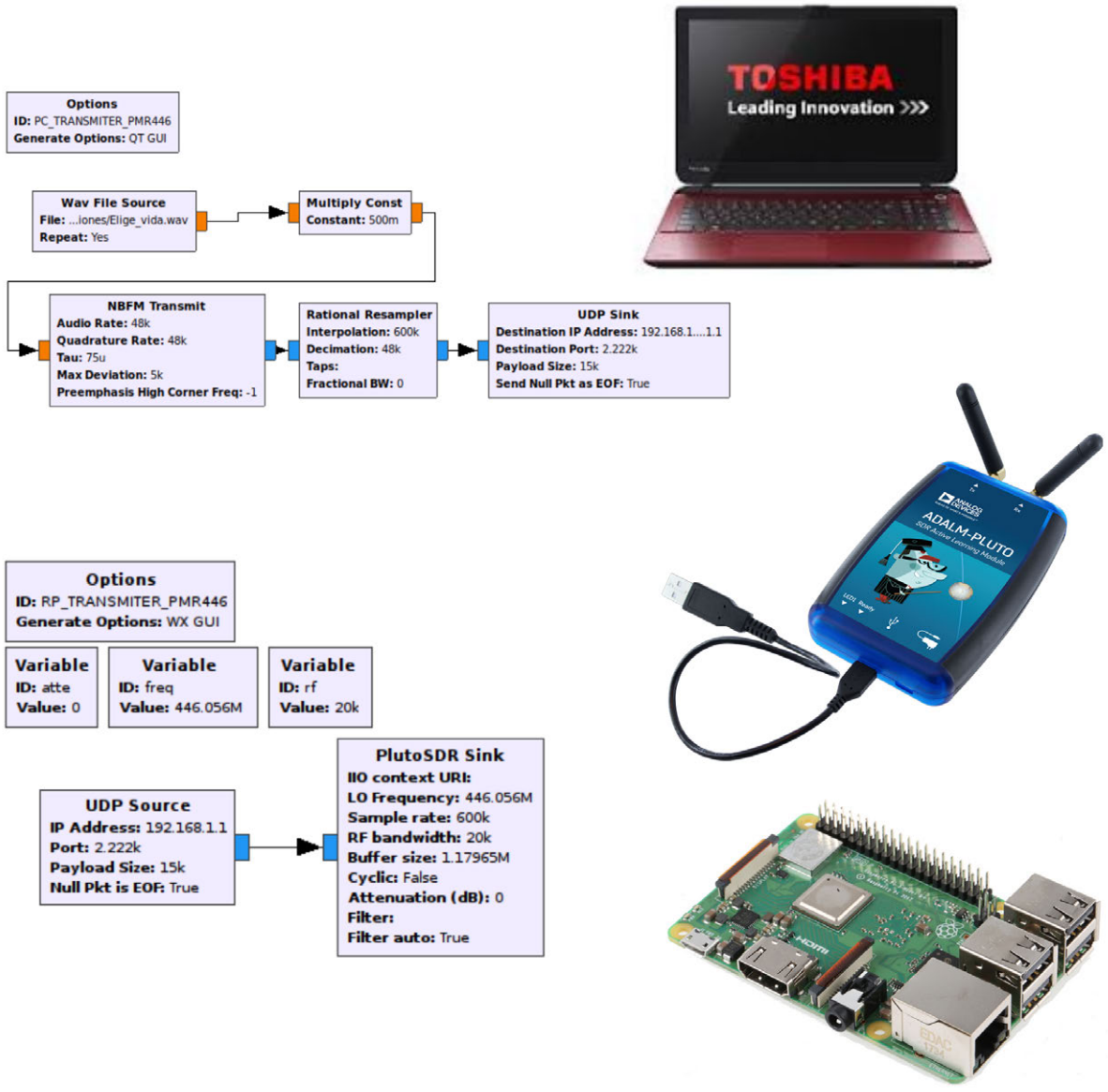

Figura 42: Diagrama de flujo GNU Radio estación remota TX PMR 446

El funcionamiento de la estación remota de comunicaciones en transmisión se divide en dos diagramas de flujo:

•Por una parte el portátil personal tendrá la función de procesar el audio: modulando e interpolando el audio hasta convertirlo en una señal PMR446 y hará de socket transmisor UDP. Tras el procesado de la señal se realizará streamming de muestras PMR 446 por el socket UDP. En el capitulo siguiente se estudiará el tamaño óptimo de las tramas UDP.

•Por otro parte la labor de la Raspberry Pi 3 b+ sera recibir las señales PMR 446 y radiarlas haciendo uso de la SDR Adalm Pluto. Su labor consistirá en hacer streamming del audio ya modulado a la frecuencia sintonizada.

Con este diseño final limitamos la función de la Raspberry Pi 3 b+ como estación remota a trasmitir la señal y que la preocupación de procesado la señal sea realizada por el PC convencional que tiene una mayor cantidad de recursos. En secciones previas se especifico que el audio se mandaba de manera reiterada. El fichero . Wav debería ser de unos 30 segundos claro y conciso que permita una escucha correcta gracias a los walkie talkie. Si se quiere trabajar sin el interfaz gráfica basta con recordar que se puede generar el código Python y ejecutarlo a través del interprete de comandos.

Estación remota de comunicaciones PMR446 en recepción

Al insertar el diseño Rx PMR446 en la Raspberry Pi 3 B+ como micro ordenador de procesado se volvió a repetir como en el apartado anterior un mal comportamiento. Aun cuando el sistema desarrollado funcionaba en el portátil personal con sistema operativo Ubuntu ,en la Raspberry Pi no se producía una correcta recepción de la señal PMR 446 como en la Figura 37: Diagrama de flujo PMR446 RX. Para ver que parte del sistema fallaba se paso a realizar un debugeo del diseño. Según [34] se puede realizar debugeo de tres formas:

•Utilice los códigos de control de calidad.

•Usando el GRC y los sumideros gráficos.

•Volcado de datos en archivos entre bloques.

La forma seleccionada es la tercera, en nuestro diseño RX PMR 446 vamos a introducir bloques File Sinks para guardar el flujo de datos entre bloque como se muestra en la figura posterior.

Se realizará tanto en el portátil personal como en la Raspberry Pi 3 b+ pudiendo ver en que parte del sistema el flujo de datos es diferente (véase Figura 43). El siguiente comando de Ubuntu nos permitirá ver si los ficheros son distintos, comparándolos linea a linea:

```
diff dataA1.dat raspiidata1.dat
```

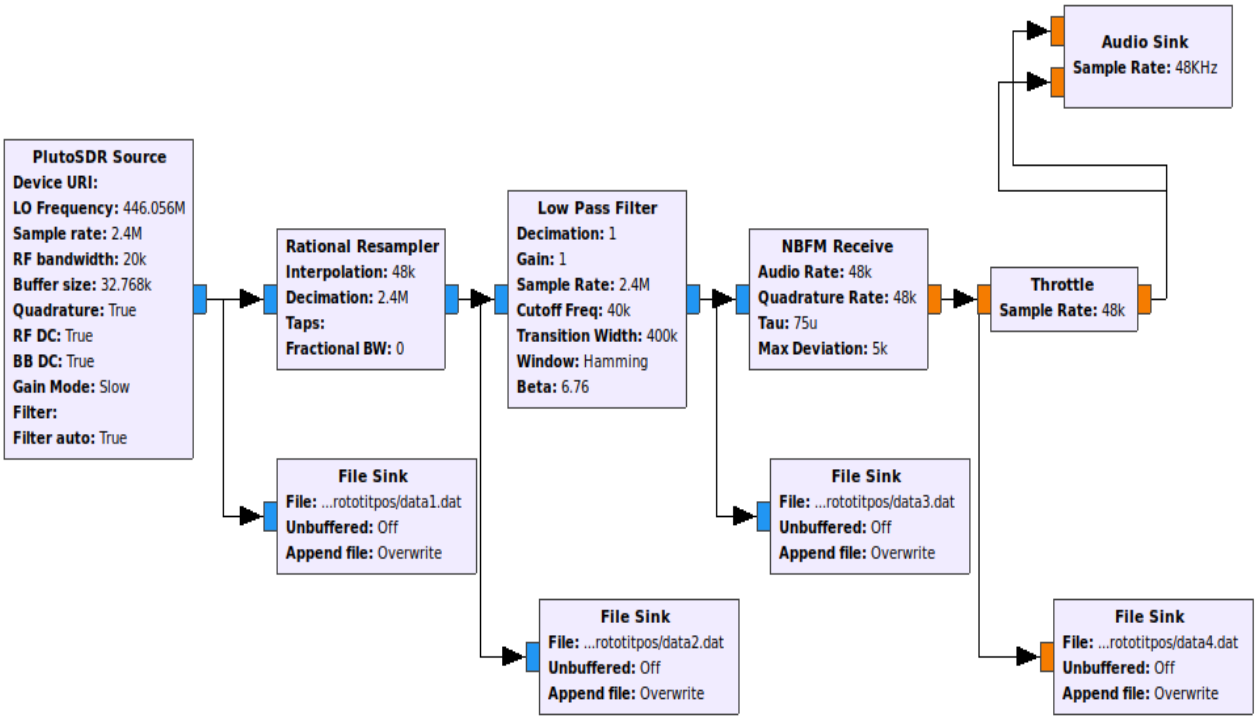

Figura 43: Debugeo RX PMR446

La siguiente tabla recoge los datos correspondiente a la salida del bloque de procesado.

Tabla 15: Tabla debugeo

	Toshiba Sattelite	**Raspberry Pi 3 B+**
Pluto SDR Source	dataA1.dat	raspiidata1.dat
Rational Resampler	dataA2.dat	raspiidata2.dat
Low Pass Filter	dataA3.dat	raspiidata3.dat
NBFM Receive	dataA4.dat	raspiidata4.dat

La Figura 44 nos mostrará por medio del interprete de comandos cual de los 4 ficheros es diferente en la Raspberry pi y en el ordenador.

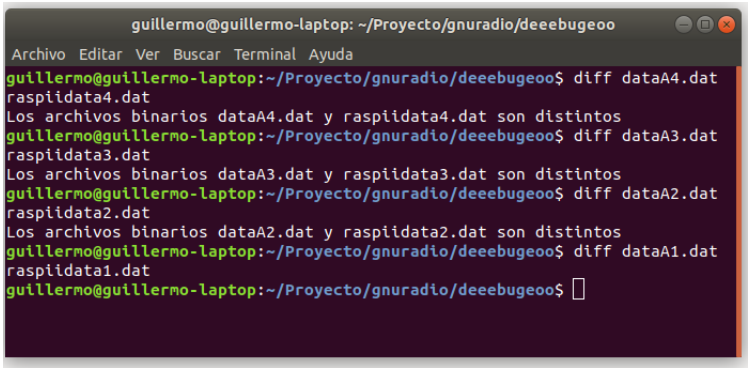

Figura 44: Diff comparación de los ficheros de debugeo

La conclusión sacada es que el flujo de datos recibido por la radio basada en software es idéntico en la Raspberry pi y en el portátil personal. Por otro lado el procesado de la señal se realiza de forma diferente como indican el resto de fichero. Esta información permitía entender que el procesado debía realizarse en el portátil personal. Por consiguiente el diseño de la estación remota receptora tomará la forma mostrada en la Figura 45.

El funcionamiento de la estación remota de comunicaciones en recepción se divide en dos diagramas de flujo:

•Por un lado la Raspberry Pi 3 b+ y la SDR Pluto harán labor de receptor ”tonto” PMR 446. Es decir la Raspberry Pi 3 b+ no procesará la señal PMR 446 dicho de otro modo ni filtrará ni diezmará ni demodulará la señal. Por contrario el flujo de muestras rf será transmitidas a un servidor UDP que transmitirá las muestras a un cliente UDP por medio de su dirección IP. En el capitulo posterior se estudiará el tamaño óptimo de las tramas UDP a transmitir.

•Por otro lado la labor del Portátil personal sera recibir el streamming PMR 446 por medio de un servidor UDP. EL Portátil Personal diezmará filtrará y demodulará para finalmente sacar el audio por la tarjeta de sonido.

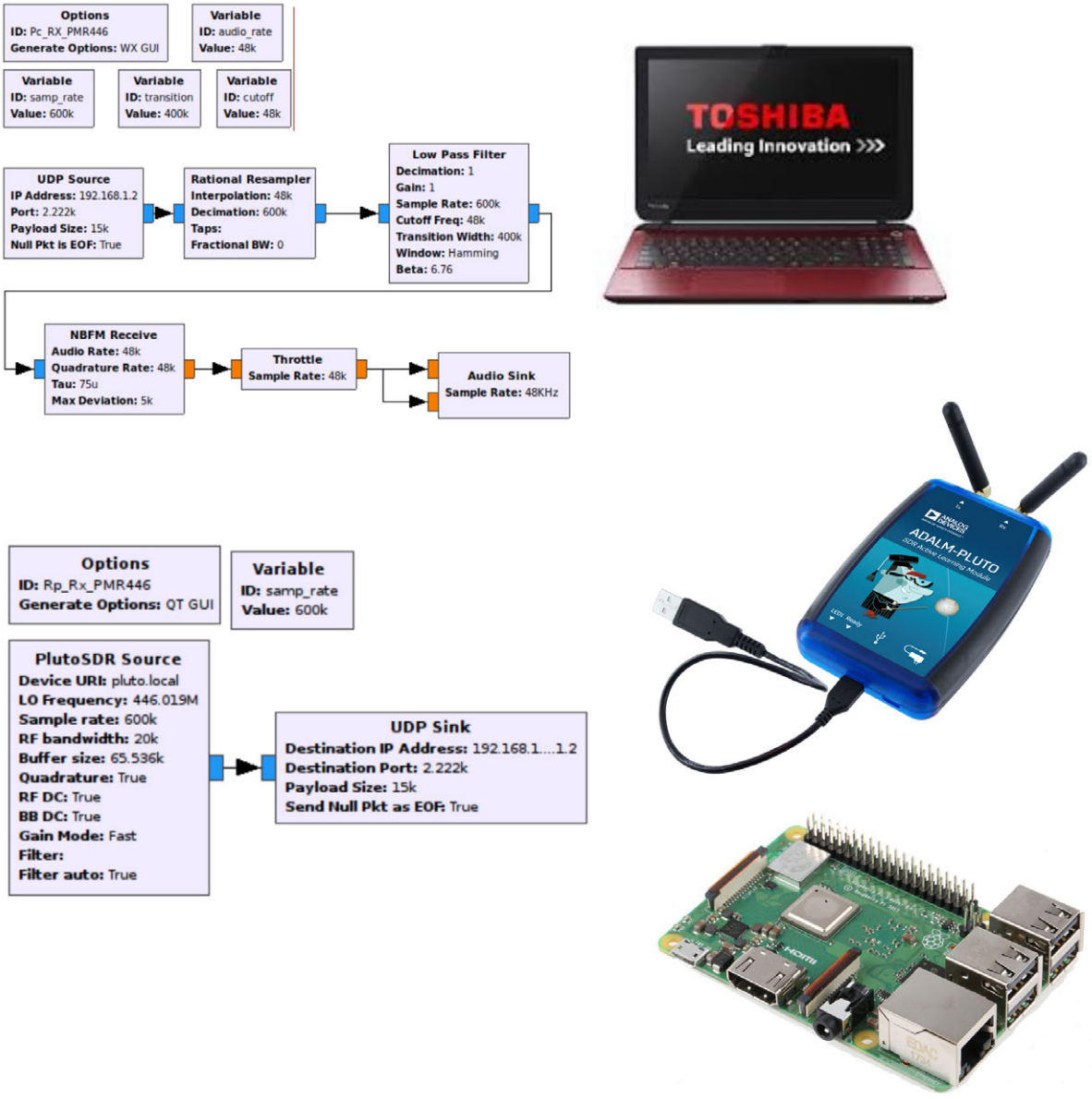

Figura 45: Diagrama de flujo GNU Radio estación remota RX PMR 446

4.3 Raspberry Pi Remota

Conectarse a la Raspberry pi de manera distante significa dar otro nivel más a la implementación de la estación remota de comunicaciones. Ello permitirá poder trabajar con el portátil personal y correr todos los software en un único ordenador, supondrá también un ahorro en hardware al no tener que utilizar teclado, ratón y monitor para la Raspberry Pi. Para el desarrollo se ha realizado la conexión punto a punto entre el portátil personal y la Raspberry pi como muestra la figura posterior, se utilizará un cable de red ethernet RJ45 (véase Figura 46).

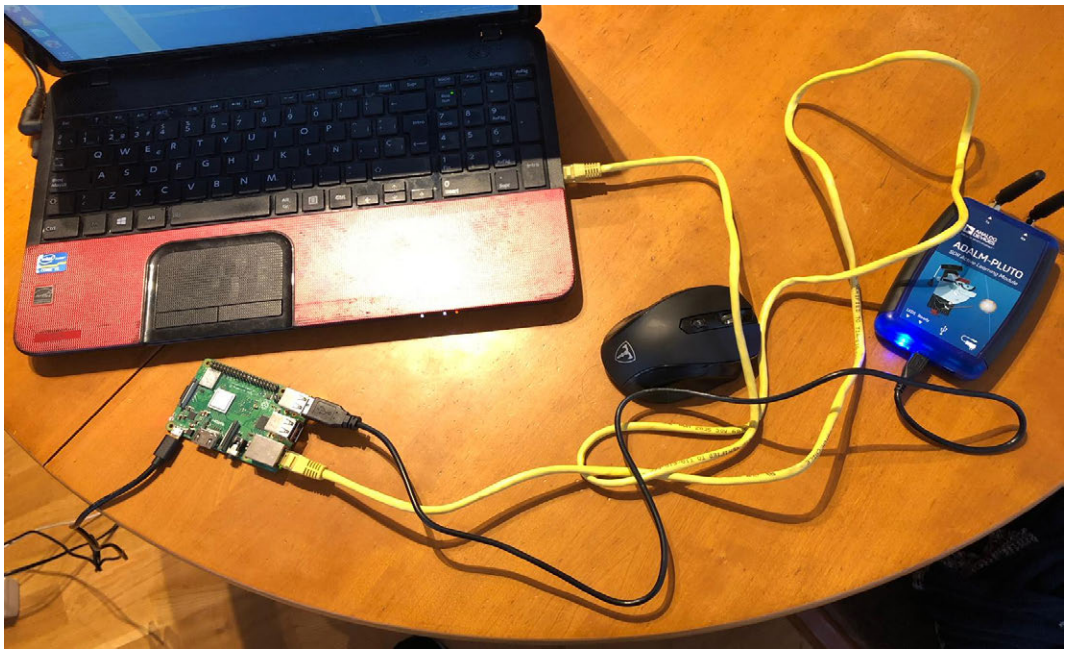

Figura 46: Sistema punto a punto portátil personal Raspberry Pi con cable ethernet RJ45

Para realizar la conexión punto a punto solo es necesario establecer una IP al portátil personal (192.168.1.2) y otra a la Raspberry Pi(192.168.1.1) como se muestra a continuación, (véase Figura 47), entre dos sistemas Linux es muy simple.

Figura 47: Configuracion punto a punto entre dos sistemas linux

Se comprueba la conexión entre los dos terminales con un simple ping a la IP 192.168.1.1 si se recibe es que estan conectados.

```
From 192.168.1.2 icmp_seq=9 Destination Host Unreachable
From 192.168.1.2 icmp_seq=10 Destination Host Unreachable
64 bytes from 192.168.1.1: icmp_seq=11 ttl=64 time=67.4 ms
64 bytes from 192.168.1.1: icmp_seq=12 ttl=64 time=0.286 ms
```

Para conectarnos remotamente es necesario hacer uso del protocolo SSH, segun el MIT [35] *"SSH (o Secure SHell) es un protocolo que facilita las comunicaciones seguras entre dos sistemas usando una arquitectura cliente/servidor y que permite a los usuarios conectarse a un host remotamente."*

La conexión ssh entre dos sistemas operativos Linux es muy sencilla, basta con introducir el siguiente comando con el usuario y la IP del host al que quiere conectarse.

```
guillermo@guillermo-laptop:~$ ssh pi@192.168.1.1
pi@192.168.1.1's password:
Linux raspberrypi 4.14.71-v7+ #1145 SMP Fri Sep 21 15:38:35 BST 2018 armv7l

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Fri Nov  2 16:45:07 2018 from 192.168.1.2
pi@raspberrypi:~
```

Se comprobará si se encuentra conectada la plutoSDR al a Raspberry pi, al trabajar aun sin interfaz gráfica, se usará el siguiente comando.

```
pi@raspberrypi:~ $ iio_info -n pluto.local
Library version: 0.15 (git tag: 6ecff5d)
Compiled with backends: local xml ip usb serial
```

El siguiente paso es ejecutar la transmisión PMR446 a modo ejemplo en código python, para comprobar la ejecución remota de la Raspberry pi. Apareció el siguiente error mostrado a continuación:

```
Unable to access the X Display, is $DISPLAY set properly?
```

Indica que no tenemos interfaz gráfica para la ejecución del código Python, se proponen las siguientes soluciones:

- No trabajar con interfaz gráfica
- SSH con display X, permite activar aplicaciones gráficas
- RPD, saca el escritorio remotamente.

Para realizar la primera alternativa, no trabajar con interfaz gráfica, es necesario desactivar la GUI de los diseños GRC como se muestra en la Figura 48. El inconveniente es que al ejecutar el diseño, a posterior, no se puede cambiar los valores de manera sencilla, teniendo que realizarse a través del código Python generado.

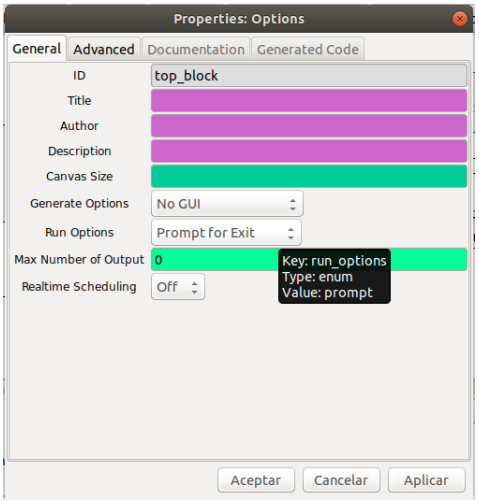

Figura 48: Desactivar interfaz grafica en GNU Radio

La segunda opción que se valora, SSH con display X, permite habilitar el display gráfico, así como ejecutar en remoto aplicaciones gráficas. Se explica como activar esta característica a través de una serie de comandos en ficheros del sistema:

Primer comando nano : permite habilitar las siguientes configuraciones X del servicio SSH en el fichero. Las configuraciones que se habilitan son las mostradas a continuación.

```
sudo nano /etc/ssh/sshd_config .
```

AllowAgentForwarding yes X11Forwarding yes X11DisplayOffset 10 X11UseLocalhost yes

Segundo reinicie el servicio SSH por medio del siguiente comando

```
sudo systemctl restart ssh
```

Para ejecutar el SSH con interfaz gráfica basta con ejeturar el siguiente mandato en el interprete de comandos

```
ssh pi@192.168.1.1 -Y -X
```

Surgirá nuevamente el interprete de comandos de la Raspberry pi, pero esta vez permitirá ejecutar el GNU Radio. La Figura 49 mostrará el interfaz de la aplicación GNU Radio ,es mucho más simple pero idéntica a la trabajada con anterioridad.

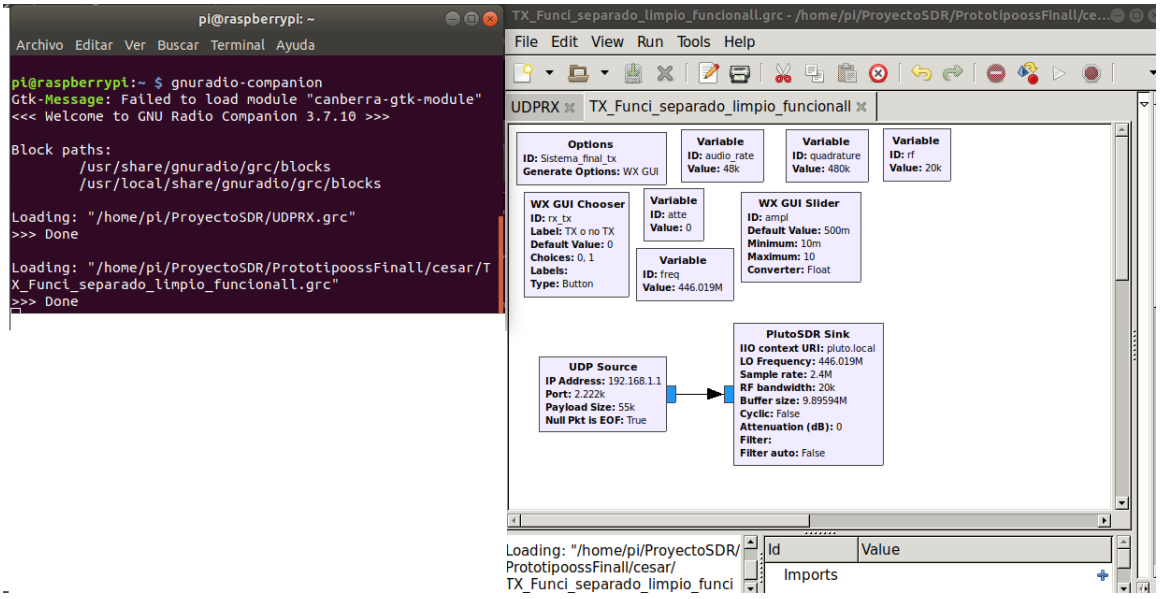

Figura 49: SSH X ,aplicaciones graficas

La ultima solución es sacar el escritorio remoto de la Raspberry pi por medio de el protocolo RPD. Según [36] *"Protocolo de escritorio remoto se basa y es una extensión de la familia de T-120 de los estándares de protocolo. Un protocolo compatible con multicanal permite canales virtuales independientes para transportar datos de presentación, comunicación de dispositivos serie, información de licencia, datos cifrados altamente (teclado, actividad del mouse) y así sucesivamente. Como una extensión del núcleo del protocolo T.Share RDP, varias otras capacidades se conservan como parte de RDP, como los elementos arquitectónicos necesarios para soportar multipunto (sesiones con varios participantes). Entrega de datos multipunto permite que los datos de una aplicación se entreguen en "tiempo real" en varias partes sin tener que enviar los mismos datos para cada sesión de forma individual (por ejemplo, pizarras virtuales)".*

Para el uso del protocolo RPD es necesario la instalación de la aplicación xrdp tanto en el portátil personal como en la raspberry pi utilizando los siguientes comandos:

sudo apt install xrdp

sudo systemctl enable xrdp

Para su ejecución basta con introducir el siguiente mandato rdesktop con la IP de la raspberry p,i surgiendo la siguiente ventana de logeo para posteriormente aparecer la GUI de la raspberri pi como muestran las imagenes posteriores. Se considera que es la mejor solución por que sacar el interfaz gráfico de la raspbery pi, pudiendo trbajar con los dos escritorios de manera sencilla. Si es necesario cambiar algún valor del diseño se podra realizar con facilidad.

Una consideración a tener en cuenta, en el apartado de diseño de la estación remota , el puerto asignado para el protocolo UDP era el 626. Al utilizar el escritorio remoto y utilizar este puerto saca error, por que los 1000 primeros puertos estan reservados para el sistema. La solución es usar un puerto del 1001 en adelante si se quiere trabajar con el escritorio remoto y ejecutar a la par el protocolo UDP. La Figura 50 mostrará la ventana de logeo usando el RPD y el interfaz gráfico tras logear del Raspbian.

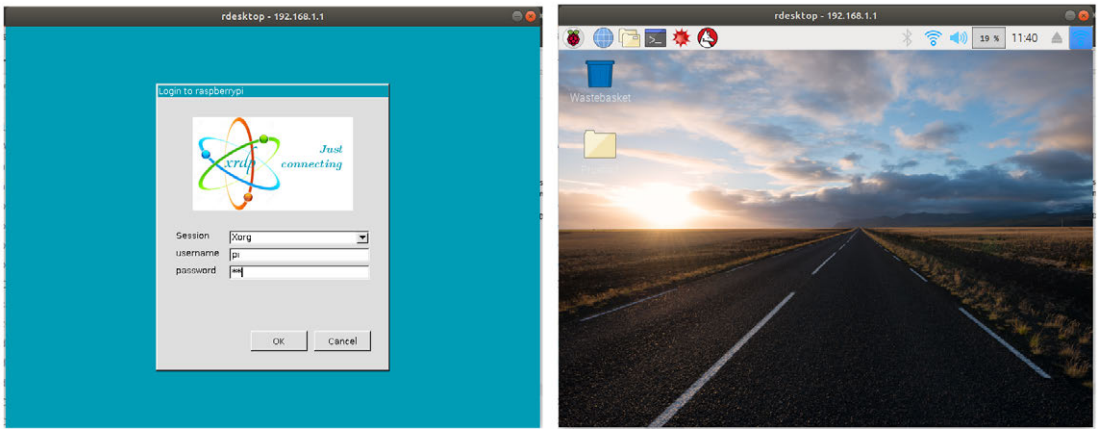

Figura 50: Ventana y escritorio remoto Raspbian

5 Medidas y resultados

En este apartado, se procede a analizar de forma cualitativa la estación remota de comunicaciones implementada en la Raspberry Pi. La estación remota de comunicaciones esta formado por distintas partes hardware y software por lo tanto se requiere distintas técnicas de medición para evaluar su funcionalidad total. Las medidas a realizar son:

- Medidas PMR 446.
- Medidas de red y protocolos.
- Medidas de Buffer, sample rate y análisis de fiabilidad.

5.1 Medidas PMR 446

Para evaluar el funcionamiento de la estación remota de comunicaciones a nivel de radiofrecuencia se va a realizar las siguientes medidas PMR 446:

- Distancia PMR 446
- Espectro y niveles de potencia PMR 446

La primera medida a realizar es la distancia de transmisión y recepción de la señal PMR446 por la estación remota de comunicaciones. Para hacer el calculo de las medidas se trabajará con la aplicación Google Maps. Segun [37] *"Google Maps es un servidor de aplicaciones de mapas en la web que pertenece a Alphabet Inc. Ofrece imágenes de mapas desplazables, así como fotografías por satélite del mundo e incluso la ruta entre diferentes ubicaciones o imágenes a pie de calle con Google Street View, condiciones de tráfico en tiempo real (Google Traffic) y un calculador de rutas a pie, en coche, bicicleta (beta) y transporte público y un navegador GPS."* La medida de la distancia se va realizar en el entorno:

- Urbano

El entorno de pruebas de distancias se ha realizado en la residencia familiar a continuación se explicara la localización así como factores destacados de la zona. La localidad familiar se sitúa en la Calle Santa Mónica con código postal 28043, perteneciente al barrio de La Colina y al distrito de Ciudad Lineal.

Se tienen en cuenta las siguientes dos consideraciones de la localización, estar situada al lado de la M-30 y de la calle López de Hoyos, implica que el transito de taxis es alto. Los taxis utilizan comunicaciones PMR446 para comunicarse entre ellos, el número de taxis en Madrid es alrededor 15000, por consiguiente puede llegar a interferir con nuestras transmisiones y recepciones, incluso pueden ser "víctimas" de nuestras transmisiones de pruebas. Se debe entender que en zonas urbanas el espectro radioeléctrico esta mucho más usado que en zonas rurales, por tanto las interferencias entre comunicaciones y el nivel de ruido es mucho mayor. La segunda consideración es la localización como se menciono en el párrafo anterior la prueba a realizar es en el barrio La Colina. Como bien indica el nombre es una colina y la vivienda familiar se encuentra en la base de la colina al lado de la M30.

En las siguientes imágenes se captura las distancias urbanas en las pruebas realizadas. Primera imagen comunicación Walkie - Walkie, segunda imagen transmisión PMR 446 por la estación remota de comunicaciones a un walkie talkie y recepción PMR 446 por la estación remota de comunicaciones de un walkie talkie (véanse Figuras 51 y 52).

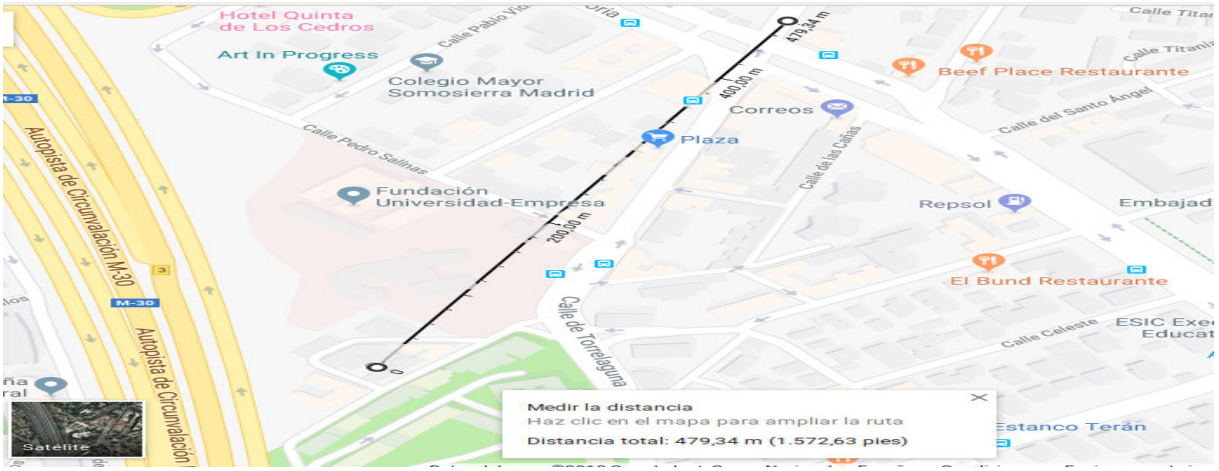

Figura 51: Medida distancia en metros PMR 446 walkie walkie

Figura 52: Medida distancia en metros PMR 446 walkie estacion remota de comunicaciones

La segunda medida de radiofrecuencia a realizar es el espectro y niveles de potencia PMR 446. Para ello se usará como hardware adicional el Portátil Personal con la SDR-RTL y realizarán la labor de analizador de espectros. Para el software del analizador de espectro se hará uso de el siguiente esquemático de GNU Radio (véase Figura 53).

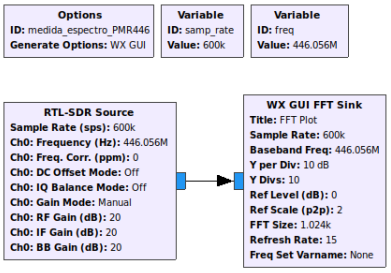

Figura 53: Diseño GNU Radio Analizador del Espectro PMR446

Permitirá realizar las mediciones de potencia de la Adalm PLuto y los Walkie talkie. Estudiar la potencia recibida por la SDR RTL y sacar conclusiones.

Primero se ejecuta el diseño mostrado con anterioridad y se visualiza el espectro desde 445.8MHZ a 446.3MHz . Se observa en la imagen inferior, Figura 54, que se tiene un ancho de banda de 600 KHZ. Esto permitirá visualizar las señales PMR 446. El analizado de espectro se mide en dbW. 0dbW correspondera 1 W de potencia recibida. 10 dbW a 10 W de potencia recibida.-10 dbW corresponde a 0.1 W y -40 a 0.1 mW.

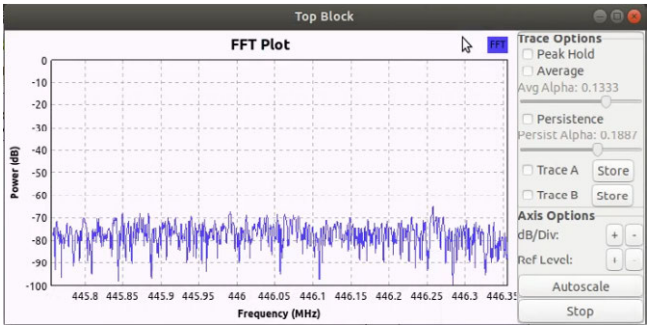

Figura 54: Visualización del analizador de espectro PMR446

La primera medida consistirá en situar la estación remota de comunicacones a un metro de distancia del analizador de espectro. Las antenas se encuentra obstaculizadas en visión por el portátil personal. Resultado la SDR RTL recibe una potencia de -40 dbW como se muestra en la imagen inferior.

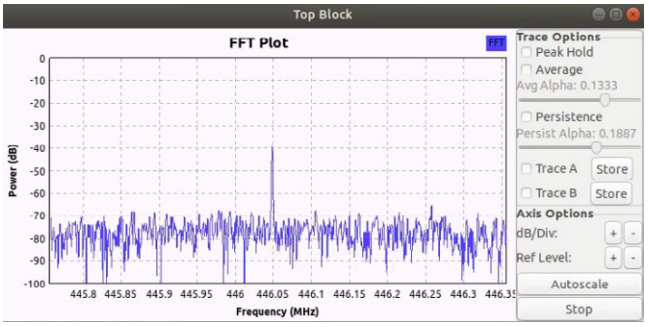

Figura 55: Primera medida del analizador de espectro PMR446

La segunda medida consistirá en situar las antenas, de la estación remota y el analizador de espectros, apuntandose a la distancia de un metro. La SDR RTL recibe una potencia de -11 dbW como se muestra en la imagen posterior.

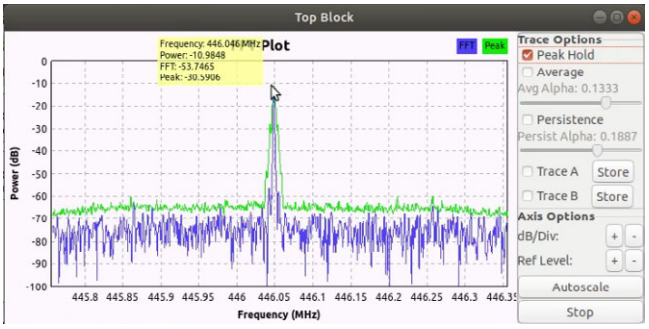

Figura 56: Segunda medida del analizador de espectro PMR446

La máxima potencia que se ha conseguido, situando la antena receptora entre medias de las antenas de la PLuto es de -2.34 dbW unos 0.6 W de potencia recibida. Por tanto la SDR Pluto tranmsite una potencia un poco mayor. En el datasheet estudiado se decia que la potencia máxima de salida eran 7dbm. Pero esto es en condiciones muy extrañas que la antenas esten en contacto es una situación irreal .

Se vuelve a situar la estación remota a la distancia de 1 metro como anteriormente. La tercera medida a realizar sera la comparación de niveles de potencia transmitidos por el walkie talkie y la estación remota de comunicaciones. La imagen posterior muestra los resultados. En color verde se visualiza la señal transmitida por la estación remota y en azul por el walkie talkie.

La potencia recibida en el analizador llega a -11 dbW transmitida por la estación remota de comunicaciones. Incluso situandonos a una distancia aproximadamente de 2 a 3 metros de la estación remota si transmitimos un pulso con uno de los walkie talkie la potencia media recibida es de -3,5dbW unos 0.45 W. Dicho valor tiene sentido a que las especificaciones de los walkies indica que tienen una potencia PIRE menos o igual que 0.5W. Situando la Pluto y el walkie talkie paralelos y apuntando a la antena receptora. si e comparan ambos espectro,azul walkie talke y verde estación remota, la diferencia de potencia es de 22dB.

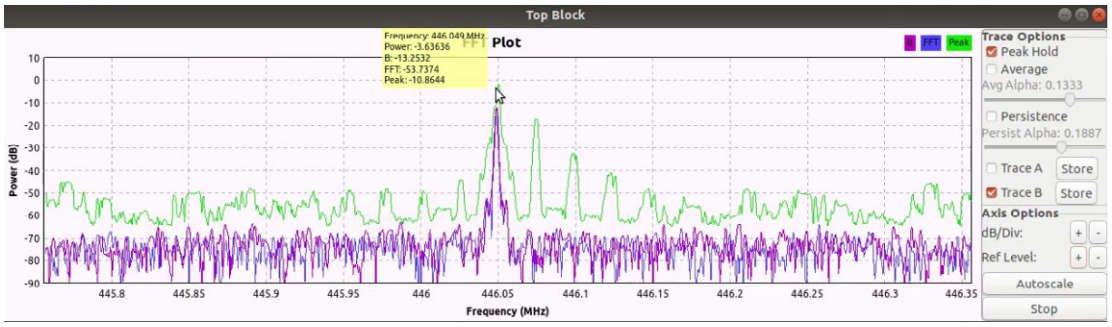

Figura 57: Última medida del analizador de espectro PMR446

Tras la prueba anterior de distancia y esta de potencia. La capacidad eficiente de la estación remota en comunicacion PMR446 es inferior a la dada por un walkie talkie convencional. El motivo es que la antena de la Pluto SDR por defecto no es tan idonea para las PMR 446 como la que tiene un walkie talkie. En futuras líneas se propondrán soluciones no muy dificiles para mejorar aun más la estación remota de comunicaciones.

5.2 Medidas de red y protocolos

Para evaluar el funcionamiento de la estación remota de comunicaciones a nivel protocolos y red se va a realizar las siguientes medidas:

5.2.1 Medidas de red

La primera medida a realizar será el tipo de red IP se necesita para transmitir el streamming PMR 446. El factor a estudiar será la latencia. En redes de datos la latencia es la suma de retardos temporales dentro de una red. Un retardo es producido por la demora en la propagación y transmisión de paquetes dentro de la propia red. Por lo tanto la latencia es el tiempo que tarda en transmitirse un paquete dentro de la red siendo un factor clave en las conexiones a Internet.

Los factores que influyen en la latencia de una red son:

•El tamaño de los paquetes transmitidos.

•El tamaño de los búferes dentro de los equipos de conectividad.

Para el estudio de que tipo de red IP se tiene que montar entre el Portátil personal y Raspberry Pi 3 b+. Se hará el estudio de latencias (retardos) en los siguientes tres casos.

1 Wifi

2 Punto a Punto

3 Red cableada

Para ello se usará el mandato tan usualmente conocido como ping, que mide en milisegundo el tiempo que tardan en comunicarse una conexión local con un equipo remoto en la red IP. En este caso el Portátil personal con la Raspberry Pi 3 b+. Una latencia de 3 ms se considerará imprescindible. El comando ping envía paquetes ICMP ECHO REQUEST a servidores de red. La opcion -s especifica el número de bytes de

datos que se van a enviar. La cantidad por defecto es 56, que pasan a ser 64 bytes de datos ICMP cuando se combinan con los 8 bytes de los datos de la cabecera ICMP. Anexo IV

Además de la prueba de ping el valor asociado al ping se probará en la propiedad ITEM SIZE del bloque UDP. Comprobando si el audio transmitido se escucha con precisión y sin pérdidas. El diseño usado de la estación remota es RXPMR446 de la página 70. Se usará archvio wav tiene este tañañooooo 13,2 MB es decir 13.234.254 bytes. Los valores a probar en bytes son:

- 65000
- 32500
- 15000
- 5000
- 1000
- 100
- 56

1 Prueba : Red WIFI

La siguiente prueba se realizará con el ordenador personal y la estación remota de comunicaciones conectados a un router domestico Wifi, como se muestra en la Figura 58, siendo las IP las siguientes:

•IP Raspberry Pi 3 b+: 192.168.1.100

•IP Portátil Personal: 192.168.1.23

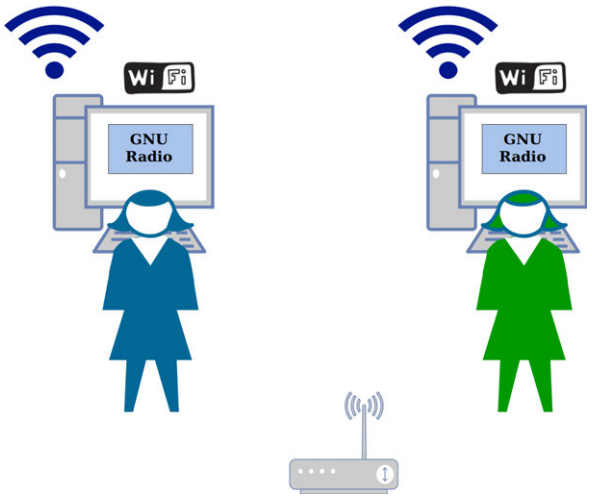

Figura 58: Esquemático Wifi

Tabla 16: Latencias wifi

Payload Size (Bytes)	AVG (ms)	MIN (ms)
65000	59.398	51.807
32500	27.615	24.413
15000	16.754	13.085
5000	7.922	7.263
1000	8.807	3.052
100	3.135	2.175
56	14.232	2.099

2 Prueba: Red Punto a Punto.

La segunda prueba se realizará con el ordenador personal y la estación remota de comunicaciones conectados por cable punto a punto, como se muestra en la Figura 59, siendo las
IP las siguientes:

•IP Raspberry Pi 3 b+: 192.168.1.1

•IP Portátil Personal: 192.168.1.2

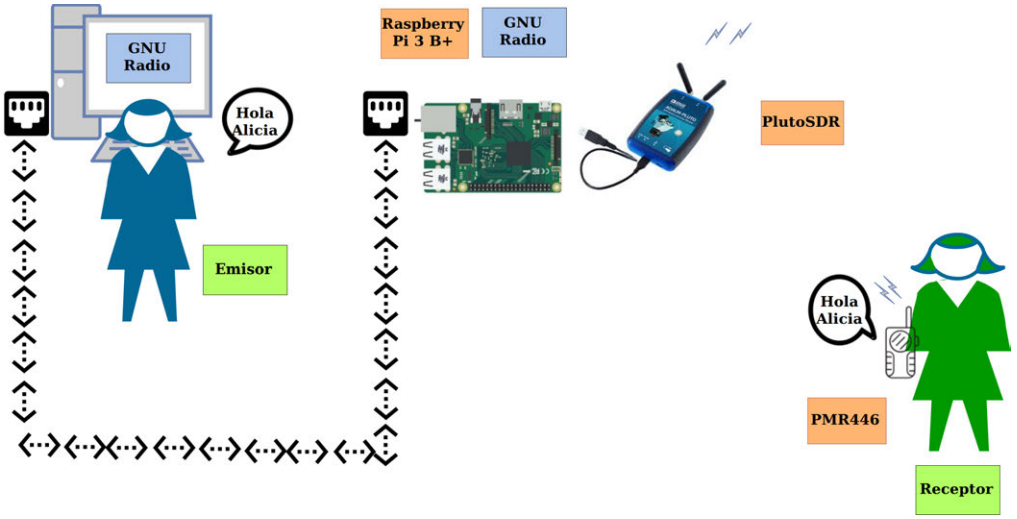

Figura 59: Esquematico punto a punto

Tabla 17: Latencias punto a punto cable

Payload Size (Bytes)	AVG (ms)	MIN (ms)
65000	11.469	11.428
32500	5.898	5.857
15000	2.911	2.873
5000	1.231	1.201
1000	0.549	0.476
100	0.355	0.283
56	1.824	1.071

3 Prueba: Red Cableada

La ultima prueba se realizará con el ordenador personal y la estación remota de comunicaciones conectados a la red domestica cada una a una entrada Ethernet del router domestico, como se muestra en la Figura 60, por cable siendo las IP las siguientes:

•IP Raspberry Pi 3 b+: 192.168.1.36

•IP Portátil Personal: 192.168.1.68

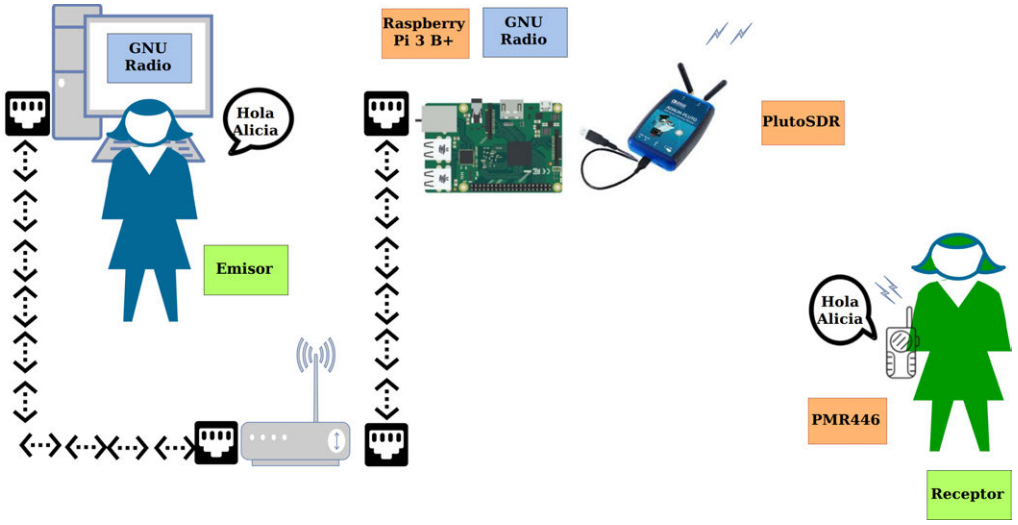

Figura 60: Esquemático cable

Tabla 18: Latencias red lan cable

Payload Size (Bytes)	AVG (ms)	MIN (ms)
65000	11.669	11.609
32500	6.167	6.144
15000	2.929	2.892
5000	1.228	1.202
1000	0.580	0.528
100	0.397	0.321
56	0.332	0.285

Se concluirá este apartado que realizando el streamming PMR 446 por medio de UDP se usará la conexión punto a punto. Con una conexión buena Wifi se podría ahorrar el cable pero no se asegura la llegada de paquetes. Mientras que la mejor red es la ultima pero supondria un aumento en gasto hardware. Se va a usar la red Punto a Punto entre el Portátil Personal y la Raspberry Pi 3 B +. Usando tamaños de hasta 15000 Bytes se consiguen latencias inferiores a 3 ms por tanto una latencia imprescindible. Si se trabaja con tamaños de tramas 100 bytes podría surgir.

```
     gr::log :WARN: udp_source0 - Too much data; dropping packet.
gr::log :WARN: udp_source0 - Too much data; dropping packet.
gr::log :WARN: udp_source0 - Too much data; dropping packet.
gr::log :WARN: udp_source0 - Too much data; dropping packet.
```

5.2.2 Medidas de protocolos

Por otra parte se va a comparar la estación remota con UDP sustituyendo los bloques de procesado por bloques TCP. La primera variación del diseño de la estación remota es el uso de un servidor TCP en vez de UDP, tomará el flujo de audio y lo envía a través de socket. Se puede usar para transmitir fácilmente un audio vía Internet o una red de manera segura, al tratarse de un protocolo orientado a la conexión. Los servidores remotos son útiles, ya que es posible que desee configurar una SDR en una ubicación distante. Para la realización del cliente-servidor TCP se volverá a hacer uso de la herramienta GNU Radio. Permitirá una creación sencilla y una integración al diseño anterior sustituyendo los bloques de procesado UDP. Se recuerda que trabajar con GNU Radio permitirá una integración de los diseños sin tener que preocuparse pos compatibilidades de software (véase Figura 61).

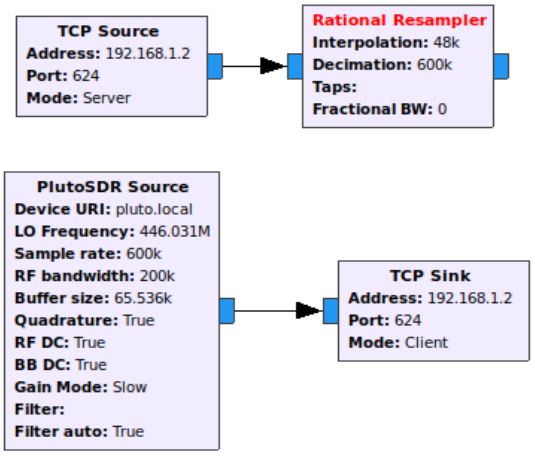

Figura 61: TCP Gnu Radio

La integración del diseño TCP se realiza sustituyendo los bloques UDP por los TCP. A continuación se realizarán pruebas de rendimiento de la CPU y se explicaran las ventajas e inconvenientes de usar este protocolo orientado a la conexión, en contraposición a UDP que no es orientado al a conexión.

Ventajas:

•Al tratarse de un protocolo orientado a la conexión, no se producirá pérdida del audio en ningún momento de la recepción PMR446 independientemente de la red a usar. La calidad del audio es identicá al usar el protocolo UDP.

Desventajas:

•La primera de ellas es la posibilidad de no realizar la conexión entre el servidor y el cliente TCP, las siguientes líneas muestran este problema. Se aprecia que se ha producido una rechazo de la conexión.

```
File "/usr/lib/python2.7/dist-packages/grc_gnuradio/blks2/tcp.py", line 69, in __ini
  fd = _get_sock_fd(addr, port, server)
File "/usr/lib/python2.7/dist-packages/grc_gnuradio/blks2/tcp.py", line 47, in _get_
  sock.connect((addr, port))
File "/usr/lib/python2.7/socket.py", line 228, in meth
  return getattr(self._sock,name)(*args)

  socket.error: [Errno 111] Connection refused
```

•El siguiente inconveniente es el tiempo que tarda en conectarse el servidor y el cliente, el tiempo de conexión puede llegar a ser de 8 segundos. Solo se puede dejar las parte del servidor activa. Se quedará esperando la trama de establecimiento de la conexión.

•Por otro lado el protocolo UDP cualquiera de las dos partes puede estar activada sin consumir recursos hasta que se active la otra, mientras que con este protocolo no se pude realizar. Es un inconveniente muy grande tener que estar ejecutando manualmente ambas partes con el protocolo TCP haciendo que el diseño de la estación remota no sea tan práctico.

•Otro inconveniente es que al producirse el establecimiento de conexión con el servidor si se cancela el cliente ocurre lo siguiente. Si se quiere intentar conectar segundos después al mismo servidor se produce error de conexión como el anterior. La solución es cancelar el servidor. Se volvería volver a ejecutar ambos diseño primero el servidor y luego el cliente.

Otro objetivo de este diseño era comparar el consumo de recursos de la CPU y memoria de ambos protocolos, para ver cual seria más óptimo a la hora de ejecutarlo en la estación. Por un lado se hará uso del monitor del sistema como herramienta para evaluar los recursos de la CPU del portátil personal. Por otra parte se hará uso de un código sacado de [38] que permitirá ver las temperaturas y tiempos de ejecución de la CPU y memoria ejecutando los diseños TCP Y UDP. La prueba consistía en realizar la recepción PMR 446 con el protocolo TCP y UDP durante 20 segundos. Las imágenes posteriores mostrarán el uso de las CPU, primero por TCP y segundo por UDP en el portátil personal (véanse Figuras 62 y 63).

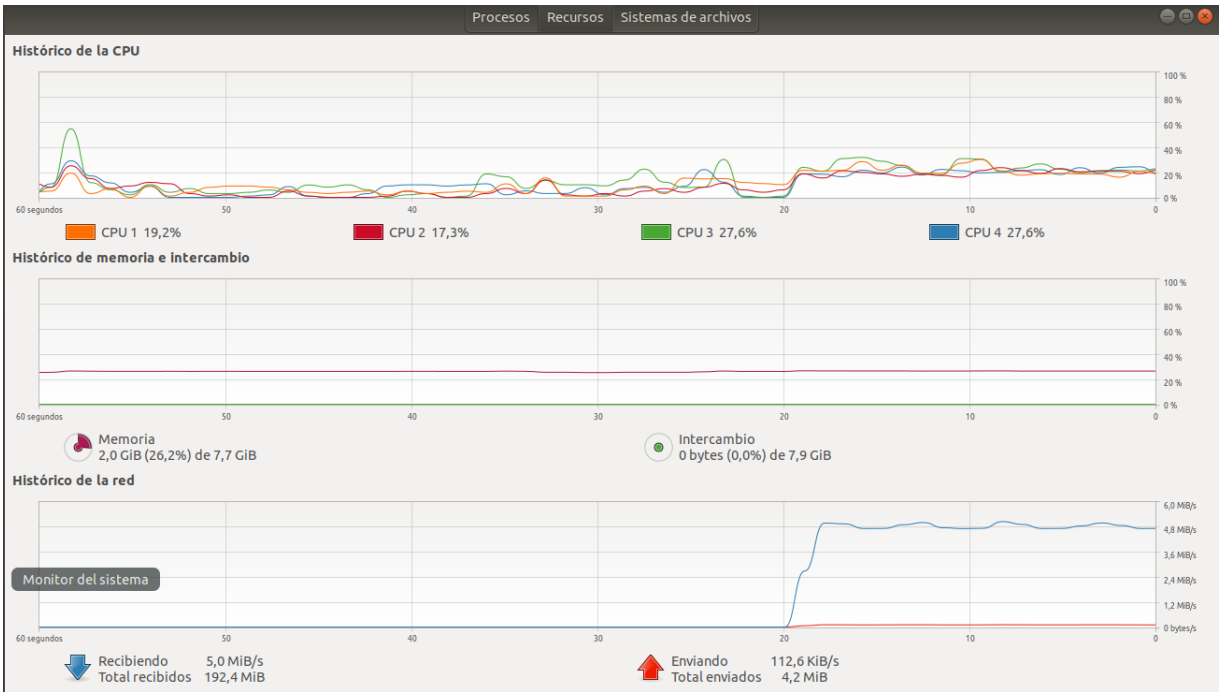

Figura 62: Rendimiento TCP CPU portátil personal

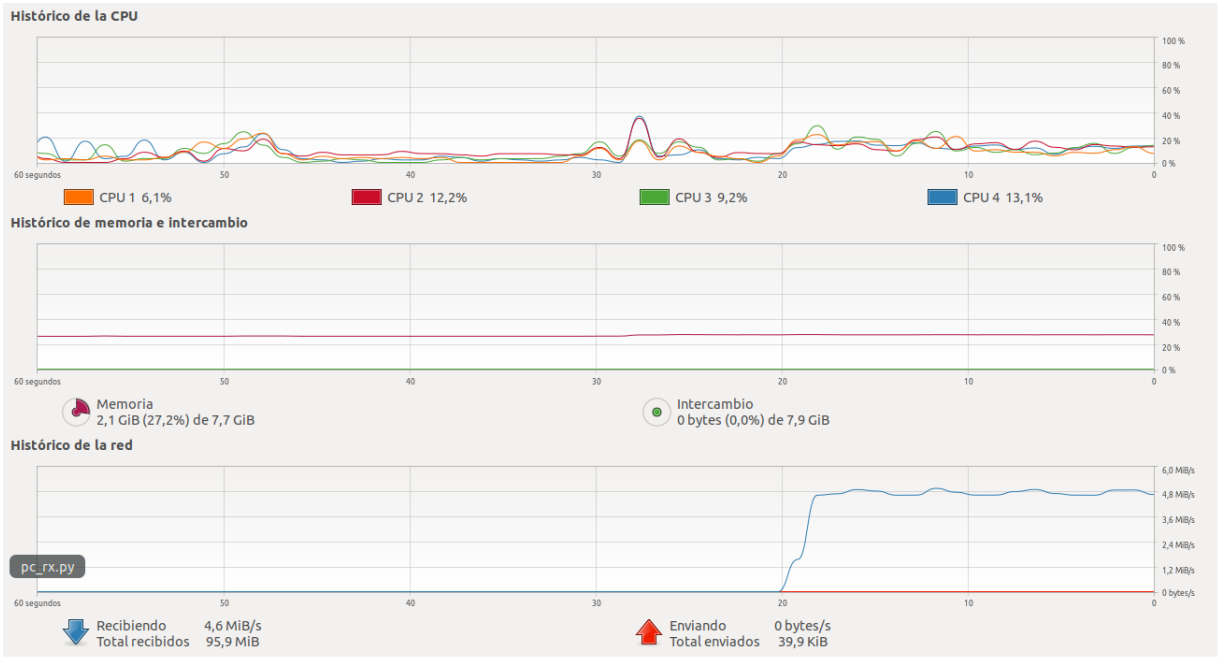

Figura 63: Rendimiento UDP CPU portátil personal

Se observa que el rendimiento de las CPU usando TCP es algo superior. Usando el protocolo orientado al a conexión dos de las CPU están cerca del 30% mientras que con UDP ninguna de las CPU supera el 13%. Se ha comprobado el consumo de recursos en el portátil personal ahora se pasará a ver el rendimiento de la Raspberry Pi como servidor"tonto" por medio del scrip del anexo V, se mostrarán las temperaturas del micro ordenador, que es un elemento muy visualizador de su uso. A mayor temperatura mayor consumo de recursos.

Tabla 19: Temperaturas Raspberry Pi

	Temp Sistema	**Temp test CPU**	**Temp test Memoria**
Raspberry Pi inactiva	40,8ºC	51,5ºC	55,3ºC
Raspberry Pi TCP	49,4ºC	58,5ºC	60,1ºC
Raspberry Pi UDP	49,4ºC	59,1ºC	60,7ºC

En la Tabla 19 se puede apreciar las diferentes temperaturas de la Raspberry Pi. Primero se ha ejecutado una prueba para ver la temperatura cuando se encuentra en reposo, se observa que el sistema se encuentra a 40,8 ºC. Las temperaturas de los test de CPU y memoria durante la ejecución del Receptor PMR446 TCP y PMR446 UDP, no son muy distantes siendo un poco superior en el protocolo UDP. La Raspberry Pi no puede alcanzar los 100ºC, una temperatura de esa magnitud dañaría el microordenador. En el caso de uso de ambos protocolos el sistema no supera los 49,4ºC, según el siguiente estudio de temperatura [39] que la temperatura no supere los 60 ºC indica que estamos dentro del rango de temperaturas razonables.

Concluiremos esta comparación de diseño que el principal inconveniente de usar TCP es tener que estar ambas partes activadas para realizar la conexión. Por otro lado podemos tener el receptor SDR UDP de manerá activa sin un consumo muy grande y cuando sea necesario la escucha del receptor, activar la parte del portátil personal.

5.3 Medidas de Buffer, sample rate y análisis de fiabilidad

El último conjunto de pruebas puede ser el final de mi proyecto fin de grado. Trabajar con el buffer y sample rate ha supuesto un papel muy importante a la hora de virtualizar las comunicaciones hasta ahora realizadas con hardware. El funcionamiento de un portátil personal es claramente superior al micro ordenador Raspberry Pi 3 b+. Pero este micro ordenador puede situarse a la misma altura. En las tablas posteriores se podrán comparar las siguientes medidas.

- Tasas de muestreo
- Medidas buffer

5.3.1 Medidas de tasas de muestreo

La Tabla 20 muestra las diferentes tasas de muestreo con las que se ha probado la SDR Pluto. La medida de tasa de muestreo permitirá entender que el procesado de la Raspberry Pi 3 B+ debe ser menor que si se realiza en el portátil personal.

Tabla 20: Medidas de tasas de muestreo

Intervalos de muestreo	**RX PMR 446 Portátil Personal**	**RX PMR 446 Raspberry Pi 3 B+**	**TX PMR 446 Portátil Personal**	**TX PMR 446 Raspberry Pi 3 B+**
300 ksps - 600 ksps	*Error de compilación 1	*Error de compilación 1	*Error de compilación 2	*Error de compilación 2
600 ksps - 900 ksps	1	1	6	6
900 ksps - 1200 ksps	1	1	6	7
1200 ksps - 1500 ksps	1	2	6	8
1500 ksps - 1800 ksps	1	3	6	9
1800 ksps - 2400 ksps	1	4	6	9
2400 ksps - 3000 ksps	1	4	6	10
3000 ksps - 3600 ksps	1	5	8	10
3600 ksps - 4200 ksps	2	5	9	10
4200 ksps - 4800 ksps	3	5	10	10
4800 ksps - 5400 ksps	4	5	10	10
>5400 ksps	5	5	10	10

1 Se escucha sin ruido ni distorsión y con claridad el audio.

2 Se escucha el audio y se empieza a escuchar un ruido de fondo.

3 Se escucha mal el audio y continuadamente el sonido agudo.

4 No se escucha el audio recivido, unicamente un sonido agudo e intermietente.

5 No se escucha nada.

6 Se transmite sin ruido ni distorsión y con claridad el audio.

7 eventualmente se produce un fallo de comunicación.

8 Se transmite el audio con distorsion y ruido agudo de fondo.

9 Transmite ruido ilegible e intermitentemente.

10 No se transmite nada.

*Error de compilación 1: Error de compilación sacado por el Gnu Radio Companion en recepción.

*Error de compilación 2: Error de compliación sacado por el Gnu Radio Companion en transmisión.

5.3.2 Medidas buffer

Cuando se realizo la inserción electro mecánicamente de la Raspberry Pi 3 b+ con la SDR Pluto se recuerda que no funciono correctamente. La primera solución encontrada fue el aumento del parámetro Buffer del bloque SDR Pluto en Gnu Radio que permitió la transmisión PMR 446 con interrupciones. Aunque a posteriori el decremento de la tasa de muestreo fue la que supuso un correcto funcionamiento de la estación remota de comunicaciones. Se ha realizado una prueba de buffer para ver cual es el valor óptimo de este parámetro y el resultado de aumentar o decrementar este valor. A continuación a modo de tabla se muestran las conclusiones sacadas (véase Tabla 21).

Tabla 21: Medidas del parámetro buffer de los bloques GNU Radio PlutoSDR Source y PlutoSDR Sink

Buffer Hexadecimal	**TX PMR 446**	**RX PMR 446**
0x8	NO Tranmiste	NO Recibe
0x100	Transmite ruido agudo y de fondo el audio a tranmitir	auaua*
0x150	Se transmite con distorsión	Se recibe con ruido y distorsión
0x200	Se transmite claro y sin distorsión ni ruido	Se recibe claro y sin distorsión ni ruido
0x1944	Se transmite claro y sin distorsión ni ruido	Se recibe claro y sin distorsión ni ruido
0x8000	Se transmite claro y sin distorsión ni ruido	Se recibe claro y sin distorsión ni ruido
0x10000	Transmite	Retardo de 1 s
0x250000	Error en ejecución 1*	Retardo de 2,87 s
0x550000	Error en ejecución 1*	Retardo de 4.701 s
0x1000000	NO Tranmiste	NO Recibe
0x1417176	Error de compilación 1*	Error de compilación 2*

*auauaua: Se produce audio underruns

*Error en ejecución 1: El diseño TX PMR 446 no transmite la señal PMR 446 debido a un buffer grande.

* Error de compilación 1 y 2: GNU Radio Companion compila el diagrama de flujo mostrando un error Python debido a un buffer muy grande.

Se concluye que si el buffer es muy elevado se pueden producir retardos en la recepción PMR 446 pero si es muy pequeño la SDR y el ordenador usado no serán capaces de procesar señal correctamente.

5.3.3 Análisis de Fiabilidad de la Estación Remota de Comunicaciones PMR 446

El último conjunto de pruebas a realizar es el análisis de fiabilidad de la estación remota de comunicaciones.n Para poder evaluar el análisis de fiabilidad en una prueba larga y continuada se hará uso del grabador de pantalla SimpleScreenRecorder. Se hará una grabación de la pantalla del portátil personal que visualizará el rendemiento del Portátil Personal y la Raspberry Pi. Constará de dos pruebas individuales:

La primera prueba de análisis de fiabilidad estudiará la RX PMR 446 estación remota de comunicaciones.

La segunda prueba de análisis de fiabilidad estudiará la TX PMR 446 estación remota de comunicaciones.

Primera prueba.

•Duración de 5 horas y media.

•Se probarán diferentes recepciones PMR 446 a lo largo de este intervalo.

La siguiente imagen mostrará las conclusiones sacadas y posteriormente se comentarán en detalle.

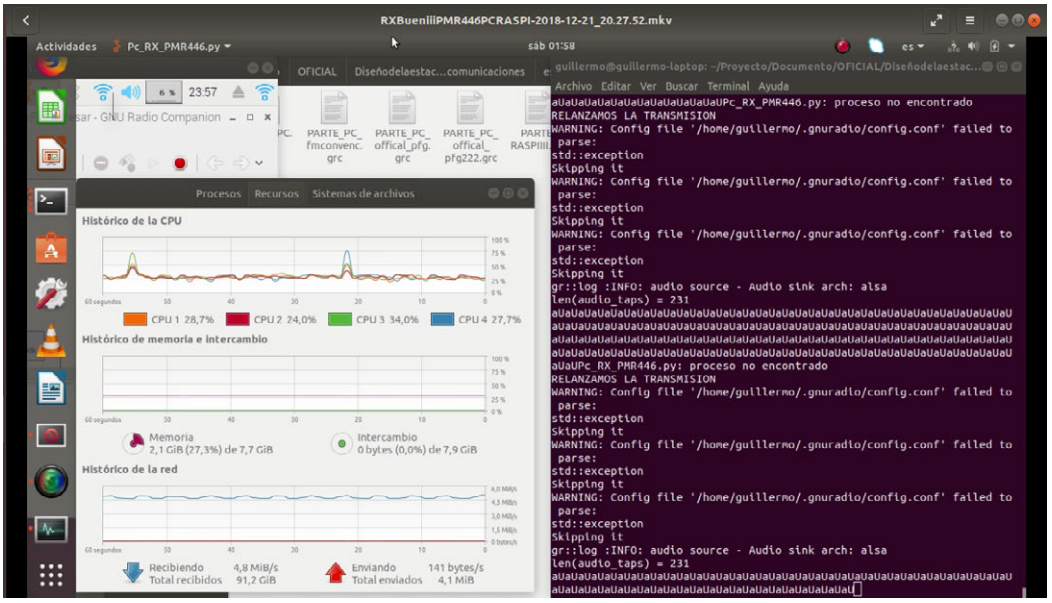

Figura 64: Análisis de fiabilidad hardware de la estación remota de comunicaciones funcionando en recepción PMR 446

Conclusiones:

•El histórico de la CPU indica dos conclusiones:

•Las Cpus del portátil personal trabajan a 20% a un 30% de forma constante.

•Cuando se relanzan la recepción PMR se produce un pico en el uso de las CPUS.
El pico no superara el 75% de una CPUs y durá menos de 3 segundos.

•El histórico de la red indica que se produce una tasa de recepción de datos entre 4.5 y 4,9MiB/s durante la medida. En la prueba de análisis de fiabilidad se han recibido 92,1GiB de datos.

•El histórico de memoria e intercambio indica que se usa 2 GiB de memoria es decir un 27,1% de la memoria total del ordenador.

•Por otra parte la Raspberry Pi 3 b+. trabaja de un 6% a un 14% durante 5 horas y media.
Se ha realizado una recepción PMR446 tanto al inicio como al final y la recepción es correcta.

La segunda prueba de análisis de fiabilidad:

•Duración de 3 horas y media.
•Se probarán diferentes transmisiones consecutivas PMR 446 a lo largo de este intervalo.

La siguiente imagen muestra las conclusiones sacadas y posteriormente se comentarán en detalle.

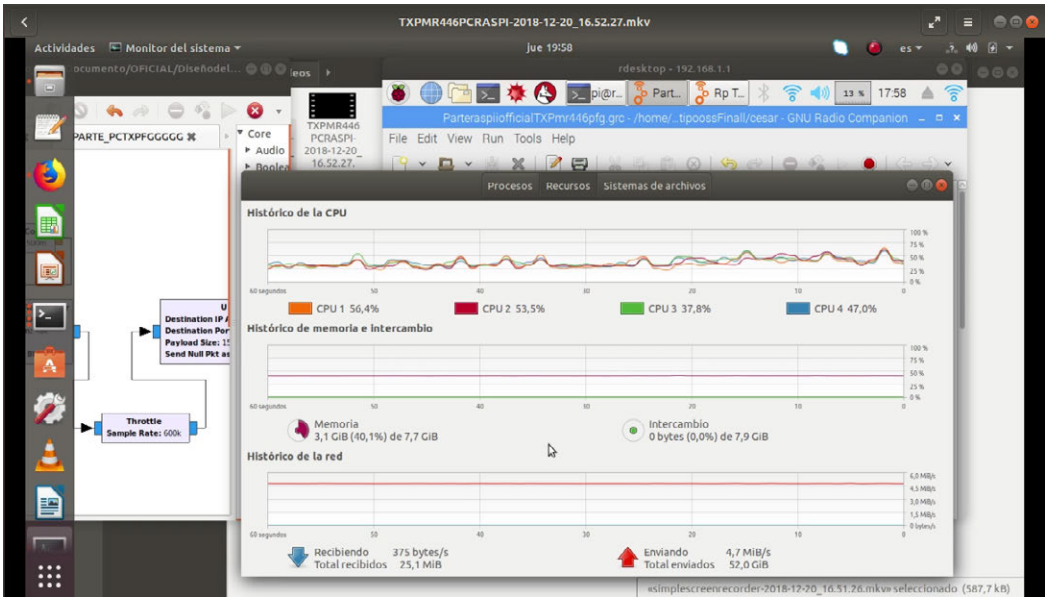

Figura 65: Análisis de fiabilidad hardware de la estación remota de comunicaciones funcionando en transmisión PMR 446

Conclusiones:

•El histórico de la CPU indica que las Cpus del ordenador trabajan de un 30% a un 60%.

•Cuando se relanzan la transmisión PMR se produce un pico en el uso de las CPU. El pico no superara el 80/% de una CPU y durá menos de 3 segundos.

•El histórico de la red indica que se produce una recepción de datos 4.5 y 4,9 MiB/s. En la prueba de análisis de fiabilidad se han transmitido 52,1 GiB de datos por la red.

•El histórico de memoria e intercambio indica que se usa 3.1 GiB de memoria es decir un 40,1% de la memoria total del ordenador.

•Por otra parte la Raspberry Pi 3 b+ trabaja de un 6% a un 14% durante 3 horas y media. Se ha realizado una transmisión PMR446 tanto al inicio como al final y la recepción es correcta.

En la prueba de análisis de fiabilidad de transmisión de la estación remota se ha trabajado con el interfaz gráfica de GNU Radio. Puede que si se trabajase sin el GUI tanto el histórico de CPUs como la histórico de memoria e intercambiando tuviesen valores inferiores. Se decidido realizar una prueba de fiabilidad mayor en numero de horas de ejecución de la estación remota de comunicaciones. Se saca una breve conclusión. Al estar trabajando la estación remota de comunicaciones PMR 446 durante días realizando streamming de señales PMR 446 de manera constante se produjo un efecto secundario en la red local. Se obtuvo una saturación del router personal produciendo un funcionamiento erróneo del Wifi para el resto de usuarios de la red, no podían acceder a internet.

6 Presupuesto

El costo de este proyecto son los gastos hardware y el tiempo invertido en desarrollarlo. En la Tabla X se muestran los precios aproximados del equipamiento. La sección A, se corresponde al equipamiento que se disponía antes de la realización del proyecto. La sección B, se corresponde al equipamiento fundamental para llevar a cabo este proyecto, exceptuando el monitor HDMI y cable HDMI. Finalmente la sección C estipula el tiempo neto aproximado para el desarrollo proyecto y el coste que puede suponer la mano de obra.

Tabla 22: Presupuesto

Equipamiento	**Ud. / Tiempo**	**Precio**
Sección A		
Toshiba Satellite	1	600
SDR-RTL	1	30
Teclado	1	10
Ratón	1	10
Cable de Red Ethernet RJ45	2	1,8
Cable HDMI	1	3
Monitor HDMI	1	50
Sección B		
Raspberry Pi B	1	20
Raspberry Pi 3 B+	1	36,65
Tarjeta microSD	1	30
TWINTALKER 4810	1	30
Switch	1	10
Cargador MicroUSB 3000 mA	1	10
ADALM-PLUTO SDR	1	84,65
Sección C		
Mano de obra (15 euros / hora)	300	4500
Total		5427,9

7 Conclusiones y líneas futuras

7.1 Conclusiones

La estación remota de comunicaciones, es el resultado de muchas pruebas que se han ido realizando a lo largo del proyecto fin de grado, desarrollo del mismo, en función de los problemas encontrados y otras mejoras que se han dejado de cara a futuras líneas como el último mencionado. Por tanto las pruebas del apartado anterior verifican un correcto funcionamiento de la estación remota de comunicaciones PMR 446. A continuación se van a sacar las conclusiones:

Las comunicaciones PMR 446 se empezaron a desarrollar en, el portátil personal Toshiba, un ordenador con prestaciones superiores a las proporcionadas por la Raspberry Py 3b +. Por consiguiente afectando directamente las tasas de muestreo de las radios definidas por software para una correcta recepción o transmisión PMR 446 tienen que ser iguales o inferiores a 600 kbps; como se puede apreciar en la Tabla 20. En relación al parámetro buffer de los bloques PlutoSDR, 0x8000, el valor por defecto es correcto. Si se aumenta demasiado podría suponer retardos de hasta 4 segundos. En caso contrario si se reduce significativamente la señal PMR se puede recibir erróneamente o producirse audio underruns.

A efectos de trabajar con Raspberry Py 3B + como ordenador de procesado de la señal PMR 446 se destaca que no resulta necesario un hardware adicional como es la pantalla o el teclado, siendo unicamente necesario el hardware extra de una alimtentacion microUSB de 3A como medida necesaria para trabajar. Las prestaciones para que la Raspberry Py 3b + trabaje con señales PMR u otras son suficientes de cara al avance de la tecnología radio definida por software.

El software ha supuesto una herramienta más que necesaria para el desarrollo de la estación remota de comunicaciones. Un problema que se ha encontrado es la instalación de los repositorios para formar las bibliotecas de las SDR en GNU Radio. Se recomienda seguir los pasos de páginas oficiales como [29]. Se advierte que se debe tener conocimiento fluido de sistemas operativos Linux. Entendiendo que si un mandato falla o se requiere ciertas modificaciones manuales. Por ejemplo requerir que se copie manualmente los bloques GNU Radio Companion entre / usr / lib y / usr / local / lib si recibe un error de importación. Otra conclusión sacada de realizar la estación remota en software GNU/Linux es que los usuarios Linux tienen las cuatro libertades digitales: ejecutar el código, estudiar y modificar el código fuente del programa, redistribuir copias exactas y distribuir versiones modificadas. Concluyendo que en el campo de la digitalización de señales se cumple excelentemente.

A pesar de las dificultades sufridas, se han logrado el objetivo principal que es crear una estación remota operativa para desplegar. Las aplicaciones para la estación remota de comunicaciones son infinitas en la era del 5G y el Internet of Things. Una comparativa de funcionamiento entre la estación remota de comunicaciones y la USRP E310 KIT, plataforma de radio definida por software independiente y portátil diseñada para la implementación al aire libre, para un propósito especifico como las comunicaciones PMR 446 seria muy recomendado. La estación remota de comunicaciones pudría ser un competidor extremadamente competitivo con las USRP Familia Embedded Series.

7.2 Líneas futuras

El diseño de la estación remota de comunicaciones con las tecnologías SDR y Raspberry Pi podría abrir las siguientes líneas de investigación futuras:

•Linea 1: se podría desarrollar la estación remota de comunicaciones en otro software.

Se podría implementar en Simulink y Matlab, se compararía con el diseño original y se estudiaría cual de ellos es más práctico a la hora de desarrollar el sistema así como el consumo de recursos por la Raspberry pi. Se podría desarrollar en base a la página de mathworks.

•Linea 2: trabajar en el proyecto OpenBTS.

Es un proyecto de código abierto dedicado a revolucionar las redes móviles mediante la sustitución de los protocolos de telecomunicaciones tradicionales y los sistemas propios de hardware por el Protocolo de Internet y una arquitectura de software flexible. Esta arquitectura es abierta a la innovación por cualquier persona, permitiendo el desarrollo de nuevos usos y servicios y simplificando el establecimiento y el funcionamiento de una red móvil. El proyecto consistiría en estudiar la plataforma hardware Raspberry Pi como base de procesado de la señal de telefonía GSM así como utilizar SDR de bajo coste como la ADALM Pluto, actualmente se utiliza SDR de gama alta para el proyecto OpenBTS-GSM como es la ETTUS. La estación remota de comunicaciones pasaria a ser una BTS de telefonía supondria un ahorro en equipo hardware y brindar flexivilidad a la hora de instalación.

•Linea 3: introducir hardware a la estación remota de comunicaciones.

Es decir se podría realizar un estudio de directividad, ganancia y efectividad de antenas. Permitirá elegir una antena más óptima para las comunicaciones PMR 446 que la antenna JCG401 que tiene la SDR Pluto por defecto.

•Linea 4: estación remota de comunicaciones 2.0

Como ultima línea futura se podría realizar una versión 2.0 de la estación. Para realizar esta segunda versión se utilizaría nuevo hardware para el diseño de la misma y se mantendría el software GNU Radio. El hardware nuevo a utilizar sería la Raspberry Pi 4 como microordenador y la SDR USRP B200mini-i como SDR. Estas dos piezas hardware cuentan con conexión USB 3.0 para realizar la conexión electro mecánica. Gracias a las mejora de características del modelo 4 de la Raspberry Pi se podría realizar el procesado de las señales digitales directamente en el microordenador. Permitiendo que la estación remota 2.0 gane en funcionalidad respecto a su antecesora.

Referencias

[1] J Mitola, “The Software Radio”, IEEE National Telesystems Conference, 1992 - Digital Object Identifier 10.1109/ NTC.1992.267870

[2] R. I. Lackey y D. W. Upmal. “SpeakEasy: The military SoftwareRadio”. IEEE Communications Magazine, Vol.33 No.5. Nueva York,Mayo de 1995.

[3] G. N. U. Radio, ”Home page”, GNU Radio. [En línea].Disponible en:`https://www.gnuradio.org/`. [Accedido: 18-sep-2018].

[4] A. Galvis Quintero, C. A. Ceballos Betancour, L. De Sanctis Gil, ”SDR: La alternativa para la evolución inalámbrica a nivel físico,” Universidad Pontificia Bolivariana - in Medellín.”, 2014.

[5] V. Huang, ”R820T High Performance Low Power Advanced Digital TV Silicon Tuner Datasheet”,Chung Hsing Road, Chutung, HsinChu 310, Taiwan. 2011.

[6] Dr. Phil, ”Realtek RTL2832U The mystery chip at the heart of RTL-SDR”, 2015.

[7] ”ADALM-PLUTO Active Learning Module” ADI — Mouser España”. [En línea]. Disponible en: `https://www.mouser.es/new/Analog-Devices/adi-adalm-pluto/`. [Accedido: 26-sep-2018]

[8] Analog Devices, ” Data Sheet RF Agile Transceiver AD9363”,One Technology Way, P.O. Box 9106, Norwood, MA 02062-9106, U.S.A.2016.

[9] Xilinx, ” Zynq-7000 SoC Data Sheet: Overview Product Specification”. 2018.

[10] G. N. U. Radio, ”Home page”, GNU Radio. [En línea]. Disponible en: https://www.gnuradio.org/. [Accedido: 27-nov-2018]

[11] ”Getting Started with Simulink - MathWorks España”. [En línea]. Disponible en: `https://es.mathworks.com/help/simulink/getting-started-with-simulink.html`. [Accedido: 27-nov-2018].

[12] ”¿Qué es LabVIEW? - National Instruments”. [En línea]. Disponible en: `http://www.ni.com/es-es/shop/labview.html`. [Accedido: 27-nov-2018].

[13] ”Guided Tutorial Introduction - GNU Radio”. [En línea]. Disponible en: `https://wiki.gnuradio.org/index.php/Guided_Tutorial_Introduction#Introduction_to_GNU_Radio_and_Software_Radio` . [Accedido: 27-sep-2018].

[14] ”Guided Tutorial GRC - GNU Radio”. [En línea]. Disponible en: `https://wiki.gnuradio.org/index.php/Guided_Tutorial_GRC#My_First_Flowgraph`[Accedido: 27-sep-2018].

[15] ”Rtl-sdr - rtl-sdr - Open Source Mobile Communications”. [En línea]. Disponible en: `https://osmocom.org/projects/rtl-sdr/wiki/Rtl-sdr`. [Accedido: 26-sep-2018].

[16] ”Comunicaciones inalámbricas - MATLAB Simulink Solutions”. [En línea]. Disponible en : `https://es.mathworks.com/solutions/wireless-communications.html`. [Accedido: 18-oct-2018].

[17] ”Spectral Analysis with ADALM-PLUTO Radio - MATLAB Simulink Example - MathWorks España”. [En línea]. Disponible en: https://es.mathworks.com/help/supportpkg/plutoradio/examples/spectral-analysis-with-adalm-pluto-radio.html. [Accedido: 18-oct-2018].

[18] ”ITE”. [En línea]. Disponible en: http://www.ite.educacion.es/formacion/materiales157cdm73recursoseducativosenlineaite.html. [Accedido: 19-sep-2018].

[19] ”Modulaciones Analogicas”,class notes for 595000022, Departamento de Ingeniería Telemática y Electrónica, Universidad Politécnica de Madrid, Primavera 2018.

[20] ”PMR446”. [En línea]. Disponible en: http://www.canal77pmr.com/index.php/pmr446.[Accedido: 25-sep-2018].

[21] ”Tipos de emisiones de radio”, Wikipedia, la enciclopedia libre. 06-sep-2018.

[22] ”Protocolo de comunicaciones”, Wikipedia, la enciclopedia libre. 20-sep-2018.

[23] J. Postel, ”Internet Procotocol” RFC 760, USC/Information Sciences Institute University of Southern California, Enero 1980.

[24] J. Postel, ”TRANSMISSION CONTROL PROTOCOL” RFC 793, USC/Information Sciences Institute University of Southern California, Septiembre 1981.

[25] ”Raspberry Pi - FdIwiki ELP”. [En línea]. Disponible en: http://wikis.fdi.ucm.es/ELP/Raspberry_Pi. [Accedido: 04-sep-2018].

[26] ” Raspberry Pi hardware - Raspberry Pi Documentation”. [En línea]. Disponible en: https://www.raspberrypi.org/documentation/hardware/raspberrypi/README. [Accedido: 03-dic-2018].

[27] ”Simbolo de un rayo en color amarillo...”, 08-dic-2016. [En línea]. Disponible en: https://www.fororaspberry.es/viewtopic.php?t=957.[Accedido:01-oct-2018].

[28] ”Limitaciones de la frambuesa pi”, Cómo ayudar y Videos - For Dummies. [En línea]. Disponible en: http://maniqui.ru/computadoras-y-software/pi-frambuesa-2/pi-frambuesa/11108-limitaciones-de-la-frambuesa-pi.html. [Accedido: 20-sep-2018].

[29] ”GNU Radio [Analog Devices Wiki]”. [En línea]. Disponible en: https://wiki.analog.com/resources/tools-software/linux-software/gnuradio. [Accedido: 09-oct-2018].

[30] ”AD9361 high performance, highly integrated RF Agile Transceiver™ Linux device driver [Analog Devices Wiki]”. [En línea]. Disponible en: https://wiki.analog.com/resources/tools-software/linux-drivers/iio-transceiver/ad9361#calibration_tracking_controls. [Accedido: 05-dic-2018].

[31] ”Tutorial: Creating an FM Receiver in GNURADIO using an RTL-SDR source”, rtl-sdr.com, 04-nov-2013. [En línea]. Disponible en: https://www.rtl-sdr.com/tutorial-creating-fm-receiver-gnuradio-rtl-sdr/. [Accedido: 02-oct-2018].

[32] ”Time constant and cutoff frequency calculator upper and lower corner frequency RIAA frequency response break RC pad tau and f RIAA NAB CCIR DIN FM conversion cut-off cut off EQ filter emphasis Pre-Emphasis De-Emphasis preemphasis deemphasis - sengpielaudio Sengpiel Berlin”. [En línea]. Disponible en: http://www.sengpielaudio.com/calculator-timeconstant.htm. [Accedido: 03-oct-2018].

[33] ”FAQ - GNU Radio”. [En línea]. Disponible en: https://wiki.gnuradio.org/index.php/FAQ#I_have_a_receiver_with_acoustic_output.2C_and_keep_getting_aUaUaU_errors.3F. [Accedido: 08-oct-2018].

[34] ”TutorialsDebugging - GNU Radio”. [En línea]. Disponible en: https://wiki.gnuradio.org/index.php/TutorialsDebugging. [Accedido: 16-oct-2018].

[35] ”Protocolo SSH”. [En línea]. Disponible en: https://web.mit.edu/rhel-doc/4/RH-DOCS/rhel-rg-es-4/ch-ssh.html. [Accedido: 02-nov-2018].

[36] ”Descripción del protocolo de escritorio remoto (RDP)”. [En línea]. Disponible en: https://support.microsoft.com/es-es/help/186607/understanding-the-remote-desktop-protocol-rdp. [Accedido: 03-nov-2018].

[37] ”Google Maps”, Wikipedia, la enciclopedia libre. 12-nov-2018.

[38] ”Script para medir el rendimiento de la Raspberry”, 10-ago-2017. [En línea]. Disponible en: https://www.fororaspberry.es/viewtopic.php?t=7077. [Accedido: 31-oct-2018].

[39] G. United, ¿Se calienta el ordenador Raspberry Pi? Estudio de sus temperaturas en funcionamiento, Geektopia. [En línea]. Disponible en:https://www.geektopia.es. [Accedido: 02-nov-2018].

References

[1] ”getting started with the raspberrypi v11”, class notes for 595010047, Departamento de Teoría de la Señal y Comunicaciones, Universidad Politécnica de Madrid, Primavera 2018.

Anexo I: Código Primer GRC

```
    #!/usr/bin/env python2
# -*- coding: utf-8 -*-
##################################################
# GNU Radio Python Flow Graph
# Title: 1Gnuradio
# Author: Guillermo Bartolome
# Generated: Wed Nov 28 12:44:22 2018
##################################################

from distutils.version import StrictVersion

if __name__ == '__main__':
    import ctypes
    import sys
    if sys.platform.startswith('linux'):
        try:
            x11 = ctypes.cdll.LoadLibrary('libX11.so')
            x11.XInitThreads()
        except:
            print "Warning: failed to XInitThreads()"

from PyQt5 import Qt
from PyQt5 import Qt, QtCore
from gnuradio import analog
from gnuradio import blocks
from gnuradio import eng_notation
from gnuradio import gr
from gnuradio import qtgui
from gnuradio.eng_option import eng_option
from gnuradio.filter import firdes
from optparse import OptionParser
import sip
import sys
from gnuradio import qtgui


class tutorial_PFG_1(gr.top_block, Qt.QWidget):

    def __init__(self):
        gr.top_block.__init__(self, "1Gnuradio")
```

```
Qt.QWidget.__init__(self)
self.setWindowTitle("1Gnuradio")
qtgui.util.check_set_qss()
try:
    self.setWindowIcon(Qt.QIcon.fromTheme('gnuradio-grc'))
except:
    pass
self.top_scroll_layout = Qt.QVBoxLayout()
self.setLayout(self.top_scroll_layout)
self.top_scroll = Qt.QScrollArea()
self.top_scroll.setFrameStyle(Qt.QFrame.NoFrame)
self.top_scroll_layout.addWidget(self.top_scroll)
self.top_scroll.setWidgetResizable(True)
self.top_widget = Qt.QWidget()
self.top_scroll.setWidget(self.top_widget)
self.top_layout = Qt.QVBoxLayout(self.top_widget)
self.top_grid_layout = Qt.QGridLayout()
self.top_layout.addLayout(self.top_grid_layout)

self.settings = Qt.QSettings("GNU Radio", "tutorial_PFG_1")

if StrictVersion(Qt.qVersion()) < StrictVersion("5.0.0"):
self.restoreGeometry(self.settings.value("geometry").toByteArray())
else:
    self.restoreGeometry(self.settings.value
    ("geometry", type=QtCore.QByteArray))

##################################################
# Variables
##################################################
self.freq = freq = 50
self.ampl = ampl = 1

##################################################
# Blocks
##################################################
self.qtgui_time_sink_x_0 = qtgui.time_sink_c(
 1024, #size
 32000, #samp_rate
 "", #name
 1 #number of inputs
)
self.qtgui_time_sink_x_0.set_update_time(0.10)
self.qtgui_time_sink_x_0.set_y_axis(-2, 2)

self.qtgui_time_sink_x_0.set_y_label('Amplitude', "")
```

```
self.qtgui_time_sink_x_0.enable_tags(-1, True)
self.qtgui_time_sink_x_0.set_trigger_mode
(qtgui.TRIG_MODE_FREE, qtgui.TRIG_SLOPE_POS, 0.0, 0, 0, "")
self.qtgui_time_sink_x_0.enable_autoscale(False)
self.qtgui_time_sink_x_0.enable_grid(False)
self.qtgui_time_sink_x_0.enable_axis_labels(True)
self.qtgui_time_sink_x_0.enable_control_panel(False)
self.qtgui_time_sink_x_0.enable_stem_plot(False)

if not True:
  self.qtgui_time_sink_x_0.disable_legend()

labels = [’’, ’’, ’’, ’’, ’’,
          ’’, ’’, ’’, ’’, ’’]
widths = [1, 1, 1, 1, 1,
          1, 1, 1, 1, 1]
colors = ["blue", "red", "green", "black", "cyan",
          "magenta", "yellow", "dark red", "dark green", "blue"]
styles = [1, 1, 1, 1, 1,
          1, 1, 1, 1, 1]
markers = [-1, -1, -1, -1, -1,
           -1, -1, -1, -1, -1]
alphas = [1.0, 1.0, 1.0, 1.0, 1.0,
          1.0, 1.0, 1.0, 1.0, 1.0]

for i in xrange(2):
    if len(labels[i]) == 0:
        if(i % 2 == 0):
self.qtgui_time_sink_x_0.set_line_label(i, "Re{{Data {0}}}".format(i/2))
        else:
self.qtgui_time_sink_x_0.set_line_label(i, "Im{{Data {0}}}".format(i/2))
    else:
        self.qtgui_time_sink_x_0.set_line_label(i, labels[i])
    self.qtgui_time_sink_x_0.set_line_width(i, widths[i])
    self.qtgui_time_sink_x_0.set_line_color(i, colors[i])
    self.qtgui_time_sink_x_0.set_line_style(i, styles[i])
    self.qtgui_time_sink_x_0.set_line_marker(i, markers[i])
    self.qtgui_time_sink_x_0.set_line_alpha(i, alphas[i])

self._qtgui_time_sink_x_0_win = sip.wrapinstance
(self.qtgui_time_sink_x_0.pyqwidget(), Qt.QWidget)
self.top_layout.addWidget(self._qtgui_time_sink_x_0_win)
self.blocks_throttle_0 = blocks.throttle
(gr.sizeof_gr_complex*1, 32000,True)
self.analog_sig_source_x_0 = analog.sig_source_c
(32000, analog.GR_SIN_WAVE, freq, ampl, 0)
```

```
        ##################################################
        # Connections
        ##################################################
        self.connect((self.analog_sig_source_x_0, 0),
        (self.blocks_throttle_0, 0))
        self.connect((self.blocks_throttle_0, 0),
        (self.qtgui_time_sink_x_0, 0))

    def closeEvent(self, event):
        self.settings = Qt.QSettings("GNU Radio", "tutorial_PFG_1")
        self.settings.setValue("geometry", self.saveGeometry())
        event.accept()

    def get_freq(self):
        return self.freq

    def set_freq(self, freq):
        self.freq = freq
        self.analog_sig_source_x_0.set_frequency(self.freq)

    def get_ampl(self):
        return self.ampl

    def set_ampl(self, ampl):
        self.ampl = ampl
        self.analog_sig_source_x_0.set_amplitude(self.ampl)


def main(top_block_cls=tutorial_PFG_1, options=None):

    if StrictVersion("4.5.0") <= StrictVersion(Qt.qVersion()) <
    StrictVersion("5.0.0"):
        style = gr.prefs().get_string('qtgui', 'style', 'raster')
        Qt.QApplication.setGraphicsSystem(style)
    qapp = Qt.QApplication(sys.argv)

    tb = top_block_cls()
    tb.start()
    tb.show()

    def quitting():
        tb.stop()
        tb.wait()
    qapp.aboutToQuit.connect(quitting)
    qapp.exec_()
if __name__ == '__main__':
    main()
```

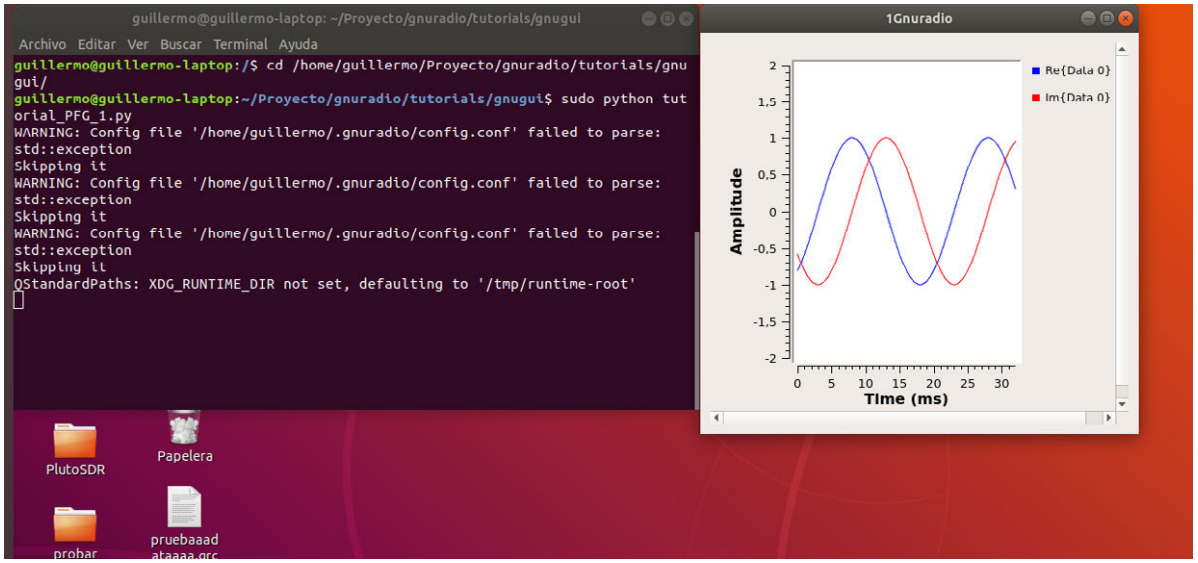

Figure 1: Código Python Primer GRC y resultado

Anexo II: Instalación Raspbian en Raspberry Pi

Este anexo describe los paso básico que se deben seguir para la instalación del SO Raspbian en la Raspberry-Pi [1]

1 Descargue la imagen RASPBIAN Debian Wheezy de: `http://www.raspberrypi.org/downloads/`

2 Grabe la imagen bajada en una tarjeta SD

2.1 Descargue SD Formatter software de:

`https://www.sdcard.org/downloads/formatter_4/eula_windows/`

Formatea la tarjeta SD al sistema de ficheros FAT32 usando el programa mencionado.

2.2 Descarge el programa Win32DiskImager de

`http://win32diskimager.sourceforge.net/`

La copia de la imagen Raspbian se realizará por medio del software WIN32. El proceso tendrá una duración de aproximadamente 10 minutos.

3 Alimenta la Raspberry Pi a través del micro-USB.

4 Conecte el cable GPIO al conector. Se conectará los colores : negro al pin uno, naranja al pin cuarto y amarillo al quinto pin para realizar una conexión en serie.

5 Identifique el número COM donde está conectado el Raspberry Py.

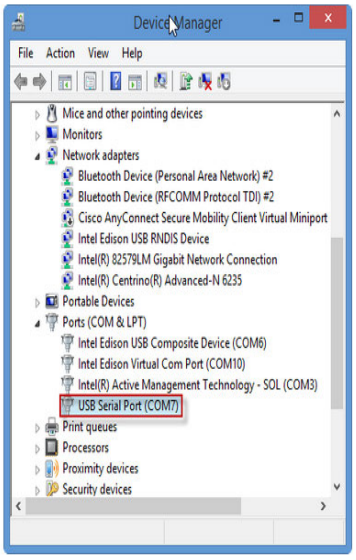

Figure 2: Identifica COM

6 Descargue el software Putty de `www.putty.org`

Abra una sesión serial en Putty como en la siguiente imagen.

7 Tras la realización en Putty, aparecerá una consola de comandas pidiendo inicio de sesión.

Raspberry login: pi

Password: raspberry

8 Finalizada la instalación se debe actualizar el sistema operativo y sus repositorios.

Se usará los siguientes comandos:

•sudo apt-get –y update

•sudo apt-get -y upgrade

Anexo III: Codigo Python RX PMR 446

```
#!/usr/bin/env python2
# -*- coding: utf-8 -*-
##################################################
# GNU Radio Python Flow Graph
# Title: Rx Pmr 446
# Generated: Fri Jan 10 12:41:51 2020
##################################################


if __name__ == '__main__':
    import ctypes
    import sys
    if sys.platform.startswith('linux'):
        try:
            x11 = ctypes.cdll.LoadLibrary('libX11.so')
            x11.XInitThreads()
        except:
            print "Warning: failed to XInitThreads()"

from gnuradio import analog
from gnuradio import audio
from gnuradio import blocks
from gnuradio import eng_notation
from gnuradio import filter
from gnuradio import gr
from gnuradio import iio
from gnuradio import wxgui
from gnuradio.eng_option import eng_option
from gnuradio.fft import window
from gnuradio.filter import firdes
from gnuradio.wxgui import fftsink2
from grc_gnuradio import wxgui as grc_wxgui
from optparse import OptionParser
import wx


class RX_PMR_446(grc_wxgui.top_block_gui):

    def __init__(self):
        grc_wxgui.top_block_gui.__init__(self, title="Rx Pmr 446")
        _icon_path = "/usr/share/icons/hicolor/32x32/apps/gnuradio-grc.png"
        self.SetIcon(wx.Icon(_icon_path, wx.BITMAP_TYPE_ANY))

        ##################################################
        # Variables
```

```
##################################################
self.transition = transition = 400000
self.samp_rate = samp_rate = 2400000
self.rf = rf = 20000
self.freq = freq = 446056250
self.cutoff = cutoff = 40000
self.audio_rate = audio_rate = 48000

##################################################
# Blocks
##################################################
self.wxgui_fftsink2_0 = fftsink2.fft_sink_c(
 self.GetWin(),
 baseband_freq=freq,
 y_per_div=10,
 y_divs=10,
 ref_level=0,
 ref_scale=2.0,
 sample_rate=samp_rate,
 fft_size=1024,
 fft_rate=15,
 average=False,
 avg_alpha=None,
 title='FFT Plot',
 peak_hold=False,
)
self.Add(self.wxgui_fftsink2_0.win)
self.rational_resampler_xxx_0 = filter.rational_resampler_ccc(
        interpolation=audio_rate,
        decimation=samp_rate,
        taps=None,
        fractional_bw=None,
)
self.pluto_source_0 = iio.pluto_source
('', int(446056250), int(samp_rate), int(rf), 0x8000, True, True, True,
"slow_attack", 40.0, '', True)
self.low_pass_filter_0 = filter.fir_filter_ccf(1, firdes.low_pass(
 1, samp_rate, cutoff, transition, firdes.WIN_HAMMING, 6.76))
self.blocks_throttle_0_0_0_0 = blocks.throttle
(gr.sizeof_gr_complex*1, samp_rate,True)
self.blocks_throttle_0_0_0 = blocks.throttle
(gr.sizeof_float*1, audio_rate,True)
self.audio_sink_0 = audio.sink(48000, '', True)
self.analog_nbfm_rx_0 = analog.nbfm_rx(
 audio_rate=audio_rate,
 quad_rate=audio_rate,
 tau=0.75e-4,
```

```
        max_dev=5e3,
         )

    ##################################################
    # Connections
    ##################################################
    self.connect((self.analog_nbfm_rx_0, 0),
    (self.blocks_throttle_0_0_0, 0))
    self.connect((self.blocks_throttle_0_0_0, 0), (self.audio_sink_0, 0))
    self.connect((self.blocks_throttle_0_0_0, 0), (self.audio_sink_0, 1))
    self.connect((self.blocks_throttle_0_0_0_0, 0),
    (self.wxgui_fftsink2_0, 0))
    self.connect((self.low_pass_filter_0, 0), (self.analog_nbfm_rx_0, 0))
    self.connect((self.pluto_source_0, 0),
    (self.blocks_throttle_0_0_0_0, 0))
    self.connect((self.pluto_source_0, 0),
    (self.rational_resampler_xxx_0, 0))
    self.connect((self.rational_resampler_xxx_0, 0),
    (self.low_pass_filter_0, 0))

def get_transition(self):
    return self.transition

def set_transition(self, transition):
    self.transition = transition
    self.low_pass_filter_0.set_taps(firdes.low_pass(1, self.samp_rate,
    self.cutoff, self.transition, firdes.WIN_HAMMING, 6.76))

def get_samp_rate(self):
    return self.samp_rate

def set_samp_rate(self, samp_rate):
    self.samp_rate = samp_rate
    self.wxgui_fftsink2_0.set_sample_rate(self.samp_rate)
    self.pluto_source_0.set_params(int(446056250), int(self.samp_rate),
    int(self.rf), True, True, True, "slow_attack", 40.0, '', True)
    self.low_pass_filter_0.set_taps(firdes.low_pass(1, self.samp_rate,
    self.cutoff, self.transition, firdes.WIN_HAMMING, 6.76))
    self.blocks_throttle_0_0_0_0.set_sample_rate(self.samp_rate)

def get_rf(self):
    return self.rf

def set_rf(self, rf):
    self.rf = rf
    self.pluto_source_0.set_params(int(446056250), int(self.samp_rate),
    int(self.rf), True, True, True, "slow_attack", 40.0, '', True)
```

```
    def get_freq(self):
        return self.freq

    def set_freq(self, freq):
        self.freq = freq
        self.wxgui_fftsink2_0.set_baseband_freq(self.freq)

    def get_cutoff(self):
        return self.cutoff

    def set_cutoff(self, cutoff):
        self.cutoff = cutoff
        self.low_pass_filter_0.set_taps(firdes.low_pass(1, self.samp_rate,
        self.cutoff, self.transition, firdes.WIN_HAMMING, 6.76))

    def get_audio_rate(self):
        return self.audio_rate

    def set_audio_rate(self, audio_rate):
        self.audio_rate = audio_rate
        self.blocks_throttle_0_0_0.set_sample_rate(self.audio_rate)


def main(top_block_cls=RX_PMR_446, options=None):

    tb = top_block_cls()
    tb.Start(True)
    tb.Wait()


if __name__ == '__main__':
    main()
```

Anexo IV: Prueba de latencia de paquetes

Latencias Wifi

```
pi@raspberrypi:~ $ ping -s 65000 192.168.1.23
65008 bytes from 192.168.1.23: icmp_seq=1 ttl=64 time=51.8 ms
65008 bytes from 192.168.1.23: icmp_seq=5 ttl=64 time=70.3 ms
--- 192.168.1.23 ping statistics ---
10 packets transmitted, 10 received, 0% packet loss, time 9013ms
rtt min/avg/max/mdev = 51.807/59.398/76.637/8.056 ms


pi@raspberrypi:~ $ ping -s 32500 192.168.1.23
PING 192.168.1.23 (192.168.1.23) 32500(32528) bytes of data.
32508 bytes from 192.168.1.23: icmp_seq=2 ttl=64 time=24.4 ms
32508 bytes from 192.168.1.23: icmp_seq=4 ttl=64 time=32.5 ms
--- 192.168.1.23 ping statistics ---
10 packets transmitted, 10 received, 0% packet loss, time 9012ms.
rtt min/avg/max/mdev = 24.413/27.615/32.598/2.721 ms


pi@raspberrypi:~ $ ping -s 15000 192.168.1.23
PING 192.168.1.23 (192.168.1.23) 15000(15028) bytes of data.
15008 bytes from 192.168.1.23: icmp_seq=2 ttl=64 time=24.1 ms
15008 bytes from 192.168.1.23: icmp_seq=9 ttl=64 time=13.0 ms
--- 192.168.1.23 ping statistics ---
12 packets transmitted, 12 received, 0% packet loss, time 11017ms
rtt min/avg/max/mdev = 13.085/16.754/24.133/3.140 ms


pi@raspberrypi:~ $ ping -s 5000 192.168.1.23
PING 192.168.1.23 (192.168.1.23) 5000(5028) bytes of data.
5008 bytes from 192.168.1.23: icmp_seq=1 ttl=64 time=8.61 ms
5008 bytes from 192.168.1.23: icmp_seq=10 ttl=64 time=7.26 ms
--- 192.168.1.23 ping statistics ---
10 packets transmitted, 10 received, 0% packet loss, time 9012ms
rtt min/avg/max/mdev = 7.263/7.922/8.611/0.328 ms


pi@raspberrypi:~ ping -s 1000 192.168.1.23
PING 192.168.1.23 (192.168.1.23) 1000(1028) bytes of data.
1008 bytes from 192.168.1.23: icmp_seq=3 ttl=64 time=3.05 ms
1008 bytes from 192.168.1.23: icmp_seq=5 ttl=64 time=53.8 ms
--- 192.168.1.23 ping statistics ---
12 packets transmitted, 12 received, 0% packet loss, time 11016ms
rtt min/avg/max/mdev = 3.052/8.807/53.849/13.708 ms
```

```
pi@raspberrypi:~ $ ping -s 100 192.168.1.23
PING 192.168.1.23 (192.168.1.23) 100(128) bytes of data.
108 bytes from 192.168.1.23: icmp_seq=7 ttl=64 time=4.81 ms
108 bytes from 192.168.1.23: icmp_seq=14 ttl=64 time=2.17 ms
--- 192.168.1.23 ping statistics ---
14 packets transmitted, 14 received, 0% packet loss, time 13022ms
rtt min/avg/max/mdev = 2.175/3.135/4.819/0.970 ms
```

```
    pi@raspberrypi:~ $ ping 192.168.1.23
PING 192.168.1.23 (192.168.1.23) 56(84) bytes of data.
64 bytes from 192.168.1.23: icmp_seq=1 ttl=64 time=2.09 ms
64 bytes from 192.168.1.23: icmp_seq=7 ttl=64 time=126 ms
--- 192.168.1.23 ping statistics ---
11 packets transmitted, 11 received, 0% packet loss, time 10016ms
rtt min/avg/max/mdev = 2.099/14.232/126.746/35.598 ms
```

Latencias Punto a Punto cable

```
pi@raspberrypi:~ $ ping -s 65000 192.168.1.2
PING 192.168.1.2 (192.168.1.2) 65000(65028) bytes of data.
65008 bytes from 192.168.1.2: icmp_seq=7 ttl=64 time=11.5 ms
65008 bytes from 192.168.1.2: icmp_seq=10 ttl=64 time=11.4 ms
--- 192.168.1.2 ping statistics ---
10 packets transmitted, 10 received, 0% packet loss, time 9016ms
rtt min/avg/max/mdev = 11.428/11.469/11.504/0.087 ms
```

```
pi@raspberrypi:~ $ ping -s 32500 192.168.1.2
PING 192.168.1.2 (192.168.1.2) 32500(32528) bytes of data.
32508 bytes from 192.168.1.2: icmp_seq=4 ttl=64 time=5.93 ms
32508 bytes from 192.168.1.2: icmp_seq=5 ttl=64 time=5.86 ms
--- 192.168.1.2 ping statistics ---
10 packets transmitted, 10 received, 0% packet loss, time 9015ms
rtt min/avg/max/mdev = 5.857/5.898/5.935/0.043 ms
```

```
pi@raspberrypi:~ $ ping -s 15000 192.168.1.2
PING 192.168.1.2 (192.168.1.2) 15000(15028) bytes of data.
15008 bytes from 192.168.1.2: icmp_seq=7 ttl=64 time=2.87 ms
15008 bytes from 192.168.1.2: icmp_seq=9 ttl=64 time=2.93 ms
--- 192.168.1.2 ping statistics ---
10 packets transmitted, 10 received, 0% packet loss, time 9012ms
rtt min/avg/max/mdev = 2.873/2.911/2.936/0.019 ms
```

```
pi@raspberrypi:~ $ ping -s 5000 192.168.1.2
PING 192.168.1.2 (192.168.1.2) 5000(5028) bytes of data.
5008 bytes from 192.168.1.2: icmp_seq=3 ttl=64 time=1.25 ms
5008 bytes from 192.168.1.2: icmp_seq=5 ttl=64 time=1.20 ms
--- 192.168.1.2 ping statistics ---
10 packets transmitted, 10 received, 0% packet loss, time 9013ms
rtt min/avg/max/mdev = 1.201/1.231/1.257/0.039 ms
```

```
pi@raspberrypi:~ $ ping -s 1000 192.168.1.2
PING 192.168.1.2 (192.168.1.2) 1000(1028) bytes of data.
1008 bytes from 192.168.1.2: icmp_seq=10 ttl=64 time=0.476 ms
1008 bytes from 192.168.1.2: icmp_seq=12 ttl=64 time=0.613 ms
--- 192.168.1.2 ping statistics ---
12 packets transmitted, 12 received, 0% packet loss, time 11445ms
rtt min/avg/max/mdev = 0.476/0.549/0.613/0.039 ms
```

```
pi@raspberrypi:~ $ ping -s 100 192.168.1.2
PING 192.168.1.2 (192.168.1.2) 100(128) bytes of data.
108 bytes from 192.168.1.2: icmp_seq=4 ttl=64 time=0.283 ms
108 bytes from 192.168.1.2: icmp_seq=9 ttl=64 time=0.405 ms
--- 192.168.1.2 ping statistics ---
10 packets transmitted, 10 received, 0% packet loss, time 9372ms
rtt min/avg/max/mdev = 0.283/0.355/0.405/0.044 ms
```

```
pi@raspberrypi:~ $ ping 192.168.1.2
PING 192.168.1.2 (192.168.1.2) 56(84) bytes of data.
64 bytes from 192.168.1.2: icmp_seq=1 ttl=64 time=3.74 ms
64 bytes from 192.168.1.2: icmp_seq=7 ttl=64 time=1.07 ms
--- 192.168.1.2 ping statistics ---
10 packets transmitted, 10 received, 0% packet loss, time 9012ms
rtt min/avg/max/mdev = 1.071/1.824/3.745/0.961 ms
```

Latencias Red Lan cable

```
pi@raspberrypi:~ $ ping -s 65000 192.168.1.68
PING 192.168.1.68 (192.168.1.68) 65000(65028) bytes of data.
65008 bytes from 192.168.1.68: icmp_seq=1 ttl=64 time=11.7 ms
65008 bytes from 192.168.1.68: icmp_seq=2 ttl=64 time=11.6 ms
--- 192.168.1.68 ping statistics ---
10 packets transmitted, 10 received, 0% packet loss, time 9010ms
rtt min/avg/max/mdev = 11.609/11.669/11.705/0.138 ms


pi@raspberrypi:~ $ ping -s 32500 192.168.1.68
PING 192.168.1.68 (192.168.1.68) 32500(32528) bytes of data.
32508 bytes from 192.168.1.68: icmp_seq=5 ttl=64 time=6.14 ms
32508 bytes from 192.168.1.68: icmp_seq=10 ttl=64 time=6.20 ms
--- 192.168.1.68 ping statistics ---
10 packets transmitted, 10 received, 0% packet loss, time 9016ms
rtt min/avg/max/mdev = 6.144/6.167/6.201/0.093 ms


pi@raspberrypi:~ $ ping -s 15000 192.168.1.68
PING 192.168.1.68 (192.168.1.68) 15000(15028) bytes of data.
15008 bytes from 192.168.1.68: icmp_seq=1 ttl=64 time=2.98 ms
15008 bytes from 192.168.1.68: icmp_seq=9 ttl=64 time=2.89 ms
15008 bytes from 192.168.1.68: icmp_seq=10 ttl=64 time=2.92 ms
^C
--- 192.168.1.68 ping statistics ---
10 packets transmitted, 10 received, 0% packet loss, time 9008ms
rtt min/avg/max/mdev = 2.892/2.929/2.986/0.042 ms


pi@raspberrypi:~ $ ping -s 5000 192.168.1.68
PING 192.168.1.68 (192.168.1.68) 5000(5028) bytes of data.
5008 bytes from 192.168.1.68: icmp_seq=1 ttl=64 time=1.25 ms
5008 bytes from 192.168.1.68: icmp_seq=3 ttl=64 time=1.20 ms
--- 192.168.1.68 ping statistics ---
10 packets transmitted, 10 received, 0% packet loss, time 9013ms
rtt min/avg/max/mdev = 1.202/1.228/1.257/0.042 ms


pi@raspberrypi:~ $ ping -s 1000 192.168.1.68
PING 192.168.1.68 (192.168.1.68) 1000(1028) bytes of data.
1008 bytes from 192.168.1.68: icmp_seq=1 ttl=64 time=0.649 ms
1008 bytes from 192.168.1.68: icmp_seq=6 ttl=64 time=0.528 ms
--- 192.168.1.68 ping statistics ---
10 packets transmitted, 10 received, 0% packet loss, time 9385ms
rtt min/avg/max/mdev = 0.528/0.580/0.649/0.037 ms
```

```
pi@raspberrypi:~ $ ping -s 100 192.168.1.68
PING 192.168.1.68 (192.168.1.68) 100(128) bytes of data.
108 bytes from 192.168.1.68: icmp_seq=3 ttl=64 time=0.321 ms
108 bytes from 192.168.1.68: icmp_seq=6 ttl=64 time=0.462 ms
--- 192.168.1.68 ping statistics ---
10 packets transmitted, 10 received, 0% packet loss, time 9341ms
rtt min/avg/max/mdev = 0.321/0.397/0.462/0.044 ms
```

```
pi@raspberrypi:~ $ ping 192.168.1.68
PING 192.168.1.68 (192.168.1.68) 56(84) bytes of data.
64 bytes from 192.168.1.68: icmp_seq=1 ttl=64 time=0.394 ms
64 bytes from 192.168.1.68: icmp_seq=7 ttl=64 time=0.285 ms
--- 192.168.1.68 ping statistics ---
10 packets transmitted, 10 received, 0% packet loss, time 9366ms
rtt min/avg/max/mdev = 0.285/0.332/0.394/0.029 ms
```

Anexo V: Código Rendimiento Rasperry Pi

```
#!/bin/bash

# Script para realizar un test a la raspberry pi
# Basado en el script de AikonCwd podeis descargarlo de
# https://raw.githubusercontent.com/aikoncwd/rpi-benchmark/master/rpi-benchmark.sh

# Instalar sysbench
  apt-get install -y sysbench

# Preparar numero de pases

if [ -z $1]
then
   pasadas=1
else
   pasadas=$1
fi

# Comienzo del test
clear
sync
echo "Raspberry Pi Test"

# Mostrar información del sistema
echo "Información del sistema:"
vcgencmd measure_temp
vcgencmd get_config int | grep arm_freq
vcgencmd get_config int | grep core_freq
vcgencmd get_config int | grep sdram_freq
vcgencmd get_config int | grep gpu_freq
printf "reloj sd="
grep "actual clock" /sys/kernel/debug/mmc0/ios 2>/dev/null
| awk '{printf("%0.3f MHz", $3/1000000)}'
echo "\n"
echo "Ejecutando test de la CPU..."
for pase in `seq 1 $pasadas`;
do
if [ $pasadas -gt 1 ]
then
   echo "Pase $pase de $pasadas \n"
fi
sysbench --num-threads=4 --validate=on --test=cpu --cpu-max-prime=5000 run
| grep 'total time:\|min:\|avg:\|max:'
| tr -s [:space:]
vcgencmd measure_temp
echo "\n"
```

```
done
echo "\n"

echo "Ejecutando test de hilos..."
for pase in `seq 1 $pasadas`;
do
if [ $pasadas -gt 1 ]
then
   echo "Pase $pase de $pasadas \n"
fi
sysbench --num-threads=4 --validate=on --test=threads --thread-yields=4000
--thread-locks=6 run | grep 'total time:\|min:\|avg:\|max:' | tr -s [:space:]
vcgencmd measure_temp
echo "\n"
done
echo "\n"

echo "Ejecutando test de memoria..."
for pase in `seq 1 $pasadas`;
do
if [ $pasadas -gt 1 ]
then
   echo "Pase $pase de $pasadas \n"
fi
sysbench --num-threads=4 --validate=on --test=memory --memory-block-size=1K
--memory-total-size=3G --memory-access-mode=seq run | grep 'Operations\
|transferred\|total time:\|min:\|avg:\|max:' | tr -s [:space:]
vcgencmd measure_temp
echo "\n"
done

echo "\nFin del test"
exit 0
```