UNIVERSIDAD NACIONAL DE RÍO CUARTO

FACULTAD DE INGENIERÍA

INGENIERÍA EN TELECOMUNICACIONES

INFORME DE PRÁCTICA PROFESIONAL

Diseño de etapa de transmisión de Transponder ADS-B utilizando Radio Definida por Software (SDR)

Alumno: Fabrizio Ventura

Tutor: Damián Primo

Director: Martín Escobar

Resumen

La presente práctica profesional se llevó a cabo en el "Grupo de Sistemas de Tiempo Real"; que depende de la Facultad de Ingeniería residente en la Universidad Nacional de Río Cuarto; para la que se diagramó una jornada laboral de 8 hs, las cuales tendrían lugar entre las 9:00 y 17:00 hs de lunes a viernes. El período temporal cubierto abarcó desde el 15/10/2018 hasta el 29/11/2018, cumpliéndose así la totalidad de 310 hs exigidas por la reglamentación de la facultad.

El plan de trabajo diagrama una lista de actividades, las cuales se ordenaron cronológicamente con el fin de permitir al alumno la realización con éxito de las tareas descriptas en cada una de ellas. Parte del desafío de la práctica consistió a aprender a trabajar con plazos, pudiendo ordenar y prever los tiempos necesarios para completar las tareas y así culminar a tiempo con las actividades estipuladas.

Dichas actividades giraron en torno al desarrollo de la etapa de transmisión de un transponder ADS-B utilizando Radio definida por Software (SDR), parte del proyecto de investigación PIDDEF 37/16 [1]. El objetivo del mismo es la construcción de un transponder ADS-B con materiales económicos, cuyo funcionamiento pueda emular al de un equipo homologado como los que se puede encontrar en el mercado; los cuales son muy costosos. Por ejemplo, el Garmin GTX345 [1] posee tanto ADS-B In como ADS-B Out, está aprobado para su uso en vuelos donde apliquen las reglas de vuelo instrumental, y tiene un precio de 4995 dólares.

Por lo tanto, se intentará contribuir a este proyecto con el desarrollo de una etapa que module la información de trama ADS-B en PPM binario y le añada el preámbulo correspondiente según la norma vigente.

Tabla de contenidos

Tabla de figuras

Objetivo

El objetivo de la práctica fue desarrollar una etapa de transcodificación. Dicha etapa debería recibir 112 bits de información, los cuales corresponden a una trama ADS-B Mode S, añadirle un preámbulo (el cual se encuentra especificado en la norma DO-282B), y modular el mensaje resultante en formato PPM binario. La duración de símbolo, del preámbulo y demás especificaciones de interés se encuentran en el texto de la norma citada con anterioridad.

La etapa desarrollada formará parte de un proyecto del Ministerio de Defensa de la Nación (PIDDEF XX), el cual pretende emular la funcionalidad de un transponder ADS-B mode S. Parte del objetivo del proyecto es realizar el prototipo con componentes económicos, motivo por el cual se trabajó en una plataforma de desarrollo de software libre (GNURadio), con un sistema operativo libre (Ubuntu 18.04) y un hardware cuyo costo es de los más bajos del mercado (LimeSdr-Mini).

Es notorio lo ambicioso que resulta dicho objetivo, más aún cuando toda esta tarea será realizada por una sola persona, quien se encuentra realizando sus primeras armas en la profesión. Por lo tanto, es pertinente contemplar el hecho de que el objetivo implícito de la práctica es permitir que el alumno pueda vislumbrar una ínfima porción del abanico de posibilidades de trabajo que ofrece Ingeniería en Telecomunicaciones, con todos los pormenores que dicha porción ofrezca.

Dado que la naturaleza de la tarea puede catalogarse como un trabajo de investigación, pasará a ser mucho más importante la adquisición de nuevos conocimientos que el cumplimiento del objetivo explícito. Como esto significa que la actividad a realizar pasa a ser la investigación de que si es posible cumplir con el objetivo explícito dadas las condiciones planteadas, la atención se verá centrada en la recolección de las experiencias cotidianas relacionadas con la investigación, como por ejemplo:

- El sentimiento de desahucio provocado por horas intentando encontrar la solución a un problema, sin que la misma pueda ser dilucidada.
- Lo enriquecedor que es consultar por ayuda a un par experimentado, quien por más que no pueda brindar un empujón en la dirección correcta siempre será capaz de proveer una visión distinta del problema.
- Los conocimientos extras que son adquiridos durante la búsqueda de aquellos necesarios para el cumplimiento del objetivo
- Los métodos propios de solución de inconvenientes o tareas tediosas que se desarrollan en pos de facilitar el desarrollo de los objetivos planteados.

Respecto al trabajo de investigación, se tratará de detallar al máximo las experiencias realizadas y de dejar bien claros los resultados obtenidos. En caso de no lograr cumplir con el objetivo planteado al comienzo, es menester de que este trabajo sea utilizado como base para

futuros desarrollos que logren cumplir con dicho objetivo o bien presenten un avance en el cumplimiento del mismo.

Es por ello que el segundo objetivo implícito de este proyecto es abrir el debate sobre la plausibilidad del desarrollo bajo las condiciones dadas: Es posible desarrollar el código correspondiente a la etapa de transcodificación en GNURadio? Si es así, es posible ejecutar ese código en un dispositivo LimeSDR-Mini? Si no es posible, que soluciones se proponen? Si es posible ejecutar dicho código en ese dispositivo, como podría montarse el mismo en un sistema embebido que emule a un transponder?

Las respuestas a esas preguntas, tratarán de brindarse durante el desarrollo de este informe.

Capítulo 1

Fundamentos teóricos

ADS-B Mode S

El sistema Automático Dependiente de Vigilancia - Difusión (ADS-B por sus siglas en inglés) es una tecnología de vigilancia cooperativa en la que un avión determina su posición a través de la navegación por satélite y la emite periódicamente, lo que permite realizar su seguimiento. La información puede ser recibida por las estaciones terrestres de control de tráfico aéreo como un reemplazo para el radar secundario ya que no necesita recibir una señal desde tierra para emitir. También puede ser recibida por otras aeronaves para proporcionar conocimiento de la situación y permitir la auto-separación.

Se considera "automático" en cuanto a que no requiere ninguna acción del piloto o entrada externa; y "dependiente", ya que depende de los datos del sistema de navegación de la aeronave.

El ADS-B es un elemento de la próxima generación de sistemas de transporte aéreo de los Estados Unidos (NextGen) y de la Single European Sky ATM Research (SESAR). Los equipos ADS-B actualmente son obligatorios en partes del espacio aéreo de Australia, mientras que en Estados Unidos una parte de los aviones deberá tener estos dispositivos para el año 2020 y el equipo será obligatorio para algunos aviones en Europa a partir de 2020

El ADS-B que consiste en dos servicios diferentes, "ADS-B de emisión" y "ADS-B de recepción" podría ser el reemplazo del radar secundario como método primario de

vigilancia para aeronaves en todo el mundo. En Estados Unidos el ADS-B es un componente integral de la nueva generación en estrategias del espacio aéreo nacional (NextGen) para mejorar o actualizar la infraestructura de la aviación y sus operaciones. El sistema ADS-B también puede proporcionar información sobre el tráfico e información meteorológica gráfica a través de aplicaciones TIS-B y FIS-B. El ADS-B aumenta la seguridad al hacer una aeronave visible en tiempo real para el control de tráfico aéreo y otras aeronaves equipadas debidamente, gracias a los datos de posición y velocidad transmitidos cada segundo. Los datos del ADS-B pueden ser registrados y descargados para un análisis posterior al vuelo y también proporciona la infraestructura de datos para seguimiento de vuelos de bajo costo y su planificación y despacho.

Los sistemas "ADS-B de emisión" difunden periódicamente información acerca de cada aeronave, tales como la identificación, posición actual, altitud y velocidad a través de un transmisor a bordo. También proporcionan a los controladores de tráfico aéreo, información como posición en tiempo real que es en la mayoría de los casos, más precisa que la información disponible de los sistemas basados en radar actualmente. Con una información más precisa, el controlador de tráfico aéreo podrá posicionar y separar aviones con una mayor precisión y oportunidad.

El sistema " ADS-B de recepción" es el que recibe en las aeronaves FIS-B y TIS-B y otros datos ADS-B, como la comunicación directa entre aeronaves cercanas. Los datos de difusión de la estación en tierra normalmente solo se realizan en presencia de un avión emitiendo ADS-B, lo que limita la utilidad de los dispositivos únicamente de recepción.

Se basa en dos componentes aéreos, una fuente GPS de alta integridad y un enlace de datos (unidad ADS-B). Hay varios tipos de enlaces de datos ADS-B certificadas, pero los más comunes operan a 1090 MHz, esencialmente un transpondedor en Modo S modificado o en 978 MHz. La FAA prefiere que los aviones que operen exclusivamente por debajo de los 18000 pies (5500 metros) utilicen el enlace 978 MHz, ya que esto ayudará a aliviar aún más la congestión de la frecuencia de 1090 MHz [2]. La Figura 1 trata de explicar gráficamente el funcionamiento de dicho sistema

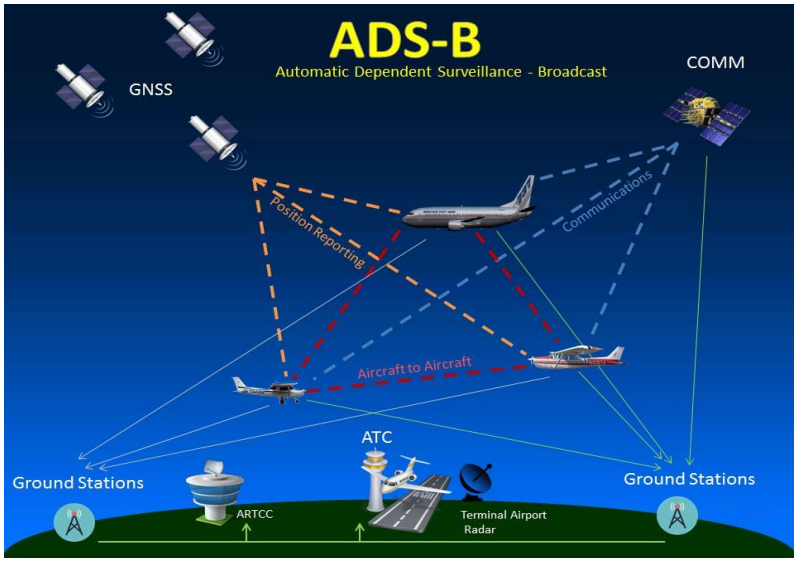

Figura N° 1: Descripción de un sistema ADS-B

Radio definida por software (SDR)

Radio definida por software o SDR (del inglés Software Defined Radio) es un sistema de radiocomunicaciones donde varios de los componentes típicamente implementados en hardware (mezcladores, filtros, moduladores/demoduladores, detectores, etc) son implementados en software, utilizando un ordenador personal u otros dispositivos de computación embebidos. Aunque el concepto de SDR no es nuevo, la reciente evolución de la circuitería digital ha hecho posible desde el punto de vista práctico muchos de los procesos que tiempo atrás eran solamente posibles desde un punto de vista teórico.

Un aparato SDR básico puede estar conformado por una ordenador equipado con una tarjeta de sonido u otro conversor analógico- digital, precedido de algún adaptador de radiofrecuencia (RF). Una gran parte del procesamiento de las señales se realiza en procesadores de propósito general, en lugar de utilizar un hardware de propósito específico. Esta configuración permite cambiar los protocolos y formas de onda simplemente cambiando el software.

La SDR es de gran utilidad tanto en los servicios de telefonía celular como en el ámbito militar, pues en ambos casos se manejan varios protocolos en tiempo real, que cambian casi constantemente según se necesite.

A largo plazo, se prevé que las emisoras definidas por software se conviertan en la tecnología dominante en las radiocomunicaciones, pues es la vía que permite llegar a la radio cognitiva. La idea de la SDR es que un mismo dispositivo programable pueda hacer lo que hace un transceptor de radio, un aparato de bluetooth o cualquier otro aparato que funcione

con ondas radioeléctricas.[3] La Figura N° 2 es una muestra de lo que puede realizarse con un sistema SDR.

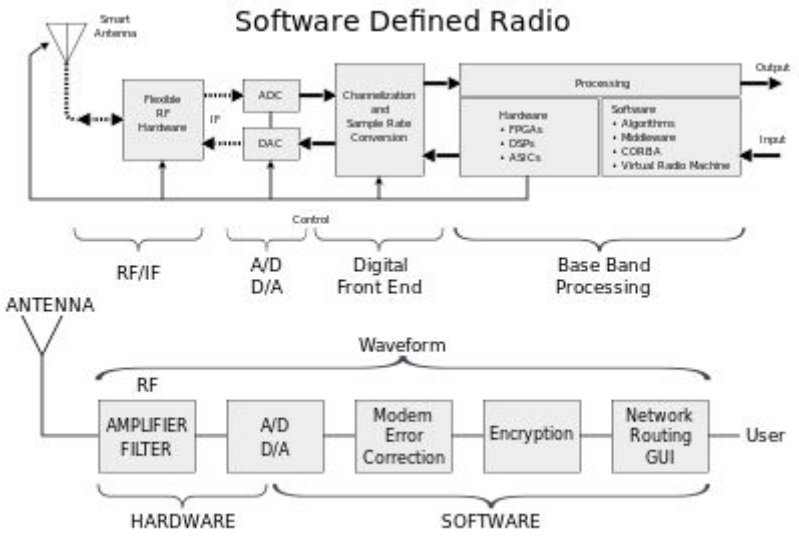

Figura N° 2: Diagrama a bloques de un sistema SDR convencional

GNURadio

GNU Radio es una herramienta de desarrollo libre y abierta que provee bloques de procesamiento de señal para implementar sistemas de radio definida por software. Puede utilizarse con hardware de RF de bajo costo para crear radios definidas por software, o sin hardware en un ambiente de simulación. Es utilizada extensivamente por ambientes académicos, aficionados y comerciales para dar soporte a la investigación en comunicaciones inalámbricas y en sistemas de radio en el mundo real.[4]

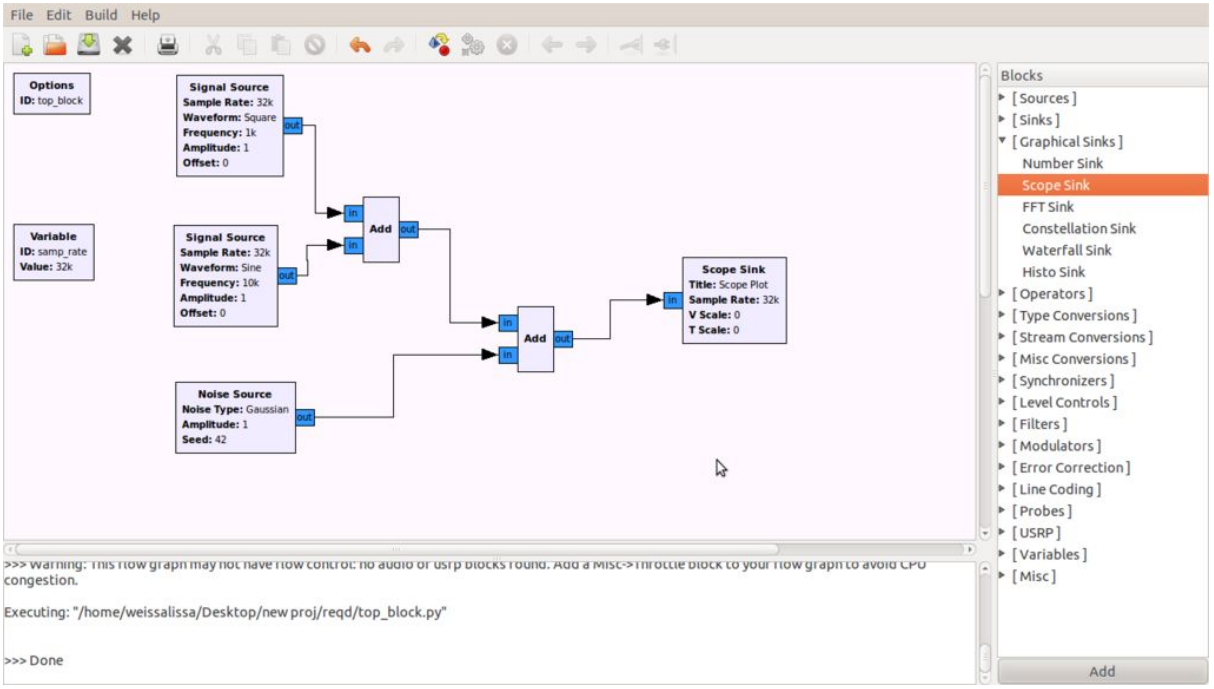

Figura N° 3: Esquema en GNURadio, que ejemplifica la suma de dos señales y la adición de ruido blanco gaussiano, ejemplificando el tránsito por un canal gaussiano

Gitlab

Gitlab es un servicio web de control de versiones y desarrollo de software colaborativo basado en Git. Además de gestor de repositorios, el servicio ofrece también alojamiento de wikis y un sistema de seguimiento de errores, todo ello publicado bajo una Licencia de código abierto.

Fue escrito por los programadores ucranianos Dmitriy Zaporozhets y Valery Sizov en el lenguaje de programación Ruby. La compañía, GitLab Inc., cuenta con un equipo de 150 miembros y más de 1400 usuarios. Es usado por organizaciones como la NASA, el CERN, IBM o Sony.[5]

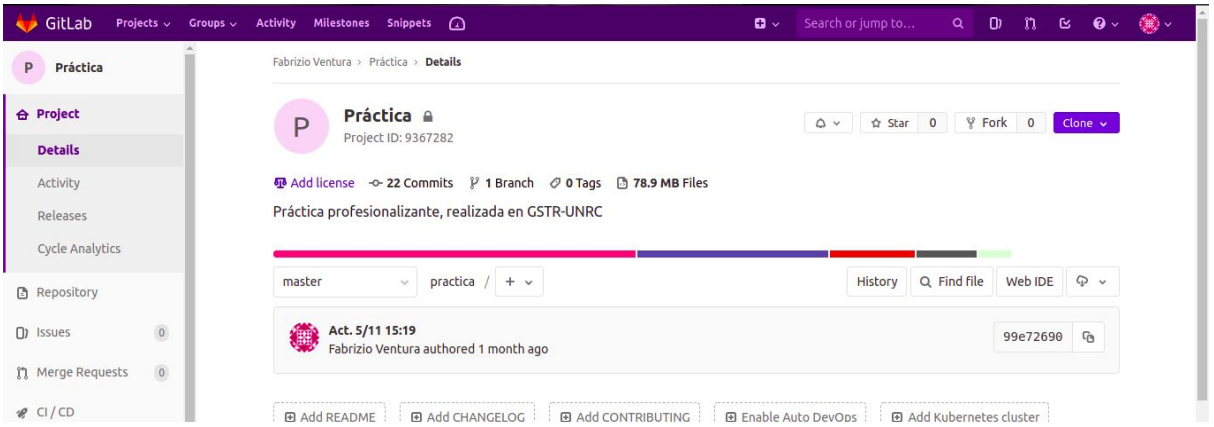

Figura N° 4: Ejemplo de repositorio creado en Gitlab, correspondiente al proyecto de la presente práctica

LimeSDR-Mini

Este dispositivo es una placa de bajo costo orientada a radio definida por software. Provee una plataforma de hardware para el desarrollo y prototipado de diseños RF de alta performance e intensiva lógica, utilizando una FPGA Intel's MAX 10 y un transceptor Lime Microsystems

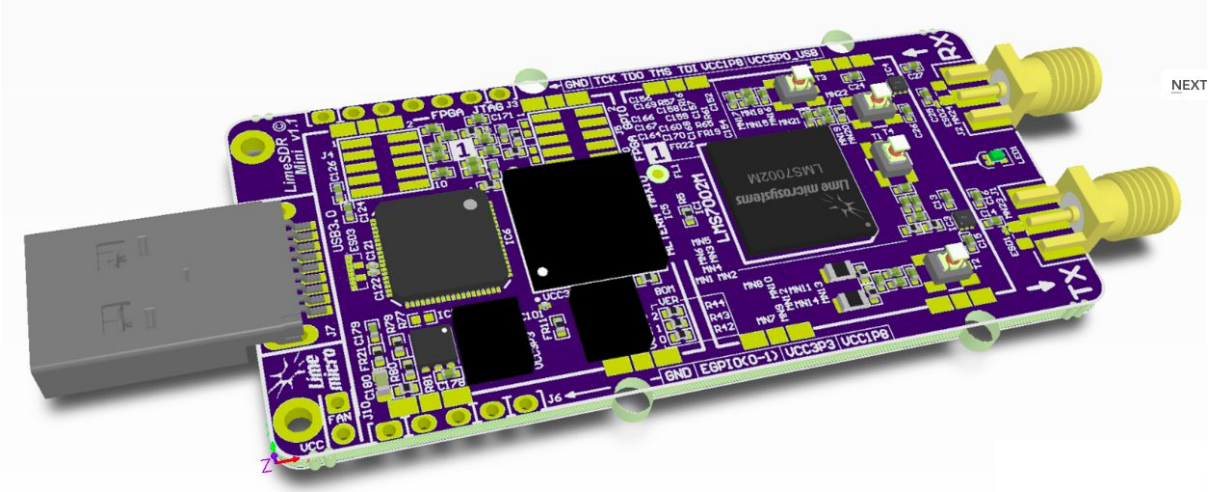

Figura N° 5: Modelo 3D del dispositivo

Permite transmitir y recibir señales en un rango de entre 10 MHz y 3.5GHz, con una frecuencia de muestreo de hasta 36 millones de muestras por segundo. Como se puede notar, posee un hardware potente.

La relación calidad-precio de la LimeSDR-Mini fue el motivo principal por el que fue elegida para llevar a cabo la realización de esta práctica profesional. El hardware descrito con anterioridad cumple teóricamente con las necesidades técnicas de la propuesta.[6]

Capítulo 2

Tareas preliminares

Instalación del sistema operativo

GNURadio no presenta problemas de compatibilidad con sistemas operativos Windows, MacOS o basados en Linux, al igual que tampoco los presentan los drivers necesarios para utilizar el dispositivo elegido; de modo tal que la elección es completamente arbitraria.

Dado que GNURadio es un software de código abierto, se eligió utilizar un sistema operativo que respetase su naturaleza. Esto se sustenta además, con la base de que dicho software fue desarrollado originalmente para sistemas operativos basados en Linux, lo cual garantiza una mayor predisposición para la corrección de bugs que puedan experimentarse; además de poseer una vasta comunidad de usuarios quienes pueden brindar ayuda con los pormenores del uso de GNURadio.

En concreto, el sistema operativo elegido fué Ubuntu 18.04 (LTS). La distribución es una de las más amigables del mundo Linux, con una comunidad activa tanto anglo como hispanoparlante. La versión elegida es la versión más reciente con soporte a largo plazo (Long Term Support), que ofrece 3 años de soporte ininterrumpido a partir de su fecha de lanzamiento (26 de abril de 2018).

Para instalarlo, solo se necesitan seguir estos simples pasos:

1. Descargar la imagen del sistema en formato ISO [7]
2. Crear un DVD o pendrive booteable con dicha imagen
3. Insertar dicho DVD o pendrive en la computadora a utilizar, y reiniciarla
4. Acceder al menú de booteo de dicha computadora, lo cual dependerá del equipo que se esté utilizando
5. Seleccionar el DVD o pendrive booteable
6. Una vez ejecutado el mismo, seleccionar la opción “Install Ubuntu 18”
7. Seguir las instrucciones

Esta distribución de Linux es la más recomendada para usuarios que desean migrar desde sistemas operativos Windows, por lo cual su instalador es muy similar a aquellos con los que los usuarios se encuentran familiarizados. Una vez que se ha concluido la instalación, el siguiente paso es instalar el software requerido para completar los objetivos.

Instalación del software GNURadio

Existen 3 caminos a elegir para la instalación del software elegido para el desarrollo de la etapa de transmisión:

- Instalarlo a través de los repositorios de la distribución linux utilizada (en este caso Ubuntu 18.04 Bionic Beaver)
- Compilar desde los archivos fuente, que se encuentran disponibles en github [8]
- Seguir el tutorial presente en la página de GNURadio [9], que instala una librería de python llamada PyBOMBS. Esta librería brinda la facilidad de añadir bloques externos a GNURadio con sólo ejecutar un comando en la terminal de linux, además de facilitar la instalación a solo eliminar el directorio en el cual se instaló. Por otro lado, presenta la desventaja de ocupar un gran espacio en el disco una vez instalado.

Considerando las ventajas de añadir módulos externos con facilidad (tarea que se supuso necesitar realizar con frecuencia), al comienzo de la práctica se instaló GNURadio siguiendo el tutorial antes descripto. Dicha librería funciona como un repositorio de GNURadio, al igual que el comando pip funciona con python. Esta elección se basó también en el hecho de que fue prevista la facilidad con la que puede estropearse la instalación del software al añadir módulos externos.

Los pasos a seguir para instalar GNURadio con PyBOMBS son (para Linux):

1. Abrir una terminal
2. Ejecutar el siguiente comando: `sudo pip install pybombs`
3. El comando anterior utiliza la herramienta pip de python para instalar el repositorio
4. Una vez instalado, se procede a establecer la "receta" que el mismo utilizará para instalar GNURadio en nuestro sistema con el comando: `pybombs recipes add gr-recipes git+https://github.com/gnuradio/gr-recipes.git`
5. Ya están cubiertos los pasos preliminares, ahora resta crear el directorio "prefix" donde se ubicará la instalación: `mkdir prefix/`
6. Por último, se le dice a PyBOMBS que instale gnuradio y todas las dependencias necesarias en el directorio antes creado: `pybombs prefix init -a default prefix/default/ -R gnuradio-default`

Instalación de drivers y plugins

Una vez instalado GNURadio, resta llevar a cabo un paso muy importante. Se deben instalar los drivers USB de la placa (en el caso de Windows, no es necesario en MacOS y Linux), de modo tal que la misma sea reconocida por el equipo.

Con GNURadio instalado, y los drivers USB también, no se puede utilizar la placa para correr las simulaciones programadas. Resta instalar un plugin para el software, que permitirá la comunicación entre el mismo y la placa. Siguiendo este sencillo tutorial (para distribuciones Ubuntu ó derivadas), se habrá finalizado con éxito la tarea:

1. Instalar el paquete LimeSuite, para lo cual se necesitan ejecutar los siguientes comandos desde una terminal:

```
$sudo add-apt-repository -y ppa:myriadrf/drivers
$sudo apt-get update
$sudo apt-get install limesuite liblimesuite-dev
limesuite-udev limesuite-images
$sudo apt-get install soapysdr-tools
soapysdr-module-lms7
```

2. Instalar las dependencias necesarias, en este caso los paquetes Boost y SWIG:

```
sudo apt-get install libboost-all-dev swig
```

3. Una vez hecho esto, solo resta instalar el plugin en si. Se pueden elegir dos caminos distintos:

3.1. Compilar los archivos fuente, para lo cual deberemos descargar el paquete gr-limesdr ejecutando:

```
$git clone https://github.com/myriadrf/gr-limesdr
```

Y luego instalarlo, corriendo los siguientes comandos:

```
$cd gr-limesdr
$mkdir build
$cd build
$cmake ..
$make
$sudo make install
$sudo ldconfig
```

3.2. En caso de haber realizado una instalación por medio de PyBOMBS, la instalación se reduce a ejecutar este simple comando:

```
$pybombs install gr-limesdr
```

Cabe destacar, como corolario, que se producen errores en la transmisión y/o recepción si la placa no es alimentada con USB 3.0, y que además el plugin de GNURadio necesita del número de serie de la misma para funcionar.

Capítulo N° 3

Diseño de la etapa de transmisión del transponder ADS-B

En esta sección, se detallará con la mayor fidelidad posible el proceso que se llevó a cabo para intentar completar las tareas. Cada subsección contendrá una descripción del objetivo planteado, para introducir al lector/a en el tema.

Actividad N° 1: Inserción

El objetivo de esta actividad es realizar una introducción a los sistemas de radar secundario, Modo S y al funcionamiento y estructura de los sistemas ADS-B. En los subtítulos siguientes, se detallarán los resultados de la investigación en dichos temas.

Radar secundario

Un sistema de RADAR (RAdio Detection And Ranging) es un sistema consistente de un transmisor y un receptor de radio sincronizados; que emite ondas electromagnéticas y procesa las ondas reflejadas para utilizarlas en la detección y localización de objetos tales como aeronaves o barcos, o en la detección de las características de superficies tales como la terrestre, lunar o planetaria.

La Figura N° 6 muestra un ejemplo de aplicación real de este sistema:

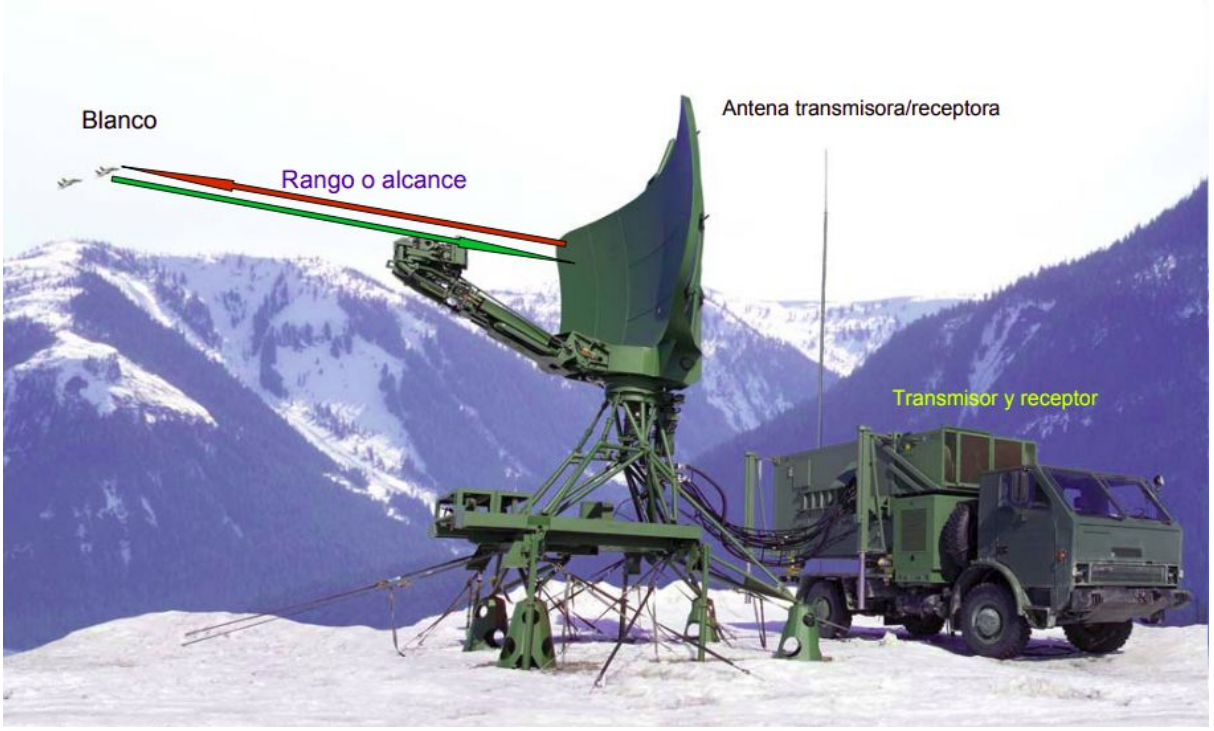

Figura N° 6: Ejemplo de sistema de radar

Existen dos principios básicos para su funcionamiento. El primero, es el llamado radar de onda pulsante. Transmite pulsos de una determinada longitud temporal, y espera una reflexión del pulso enviada. Básicamente, funciona por la detección de ecos.

La Figura N° 7 muestra un diagrama temporal de dicho comportamiento:

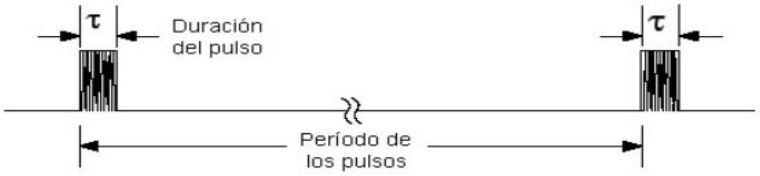

Figura N° 7: Diagrama temporal de las señales de un sistema de onda pulsante

El segundo, es el radar de onda continua. Como su nombre lo indica, transmite una señal senoidal continua, y realiza la detección valiéndose principalmente del efecto doppler.[10]

Habiendo descrito la naturaleza básica de su funcionamiento, toca diferenciar a los radares primarios de los secundarios. Los radares primarios funcionan a través de ecos pasivos, recibiéndose los pulsos reflejados por el objetivo en la misma antena de radar.

Los radares secundarios, en cambio, trabajan con señales activas de respuesta. Esto significa que no solo envían pulsos de interrogación; si no que además los reciben, procesan y emiten respuestas en función de la situación. Recapitulando, podemos afirmar que un radar primario detecta la presencia de un objeto y un radar secundario nos permite identificarlo.

Radares secundarios Modo S

Modo S refiere al tipo de interrogación que utilizan dichos sistemas. El modo S o selectivo permite una comunicación de aeronave a aeronave, generando como problema principal que los radares secundarios tradicionales sin capacidad de Modo S no puedan seleccionar la aeronave interrogada.

Así pues, cada interrogación realizada recibe las respuestas simultáneas de todos los transpondedores dentro del alcance de la antena y del ángulo de emisión. En los escenarios actuales, donde la densidad del tráfico aéreo es mucho mayor que en el momento de la implantación original del radar secundario, es habitual que las respuestas se solapen entre sí y se confundan, no pudiendo ser recuperadas (fenómeno conocido como "garbling" en terminología radar), y por lo tanto degradando la calidad de la vigilancia aérea.

La implantación del modo S permite que el radar interrogue "aeronave a aeronave" de modo que el radar puede manejar un escenario de tráfico mucho mayor sin complicar dicho escenario.[11]

ADS-B Modo S

Luego de una búsqueda en el material [12], se logró encontrar la información necesaria como para empezar a diagramar un boceto del transcodificador.

A continuación, se presenta un boceto del formato de trama:

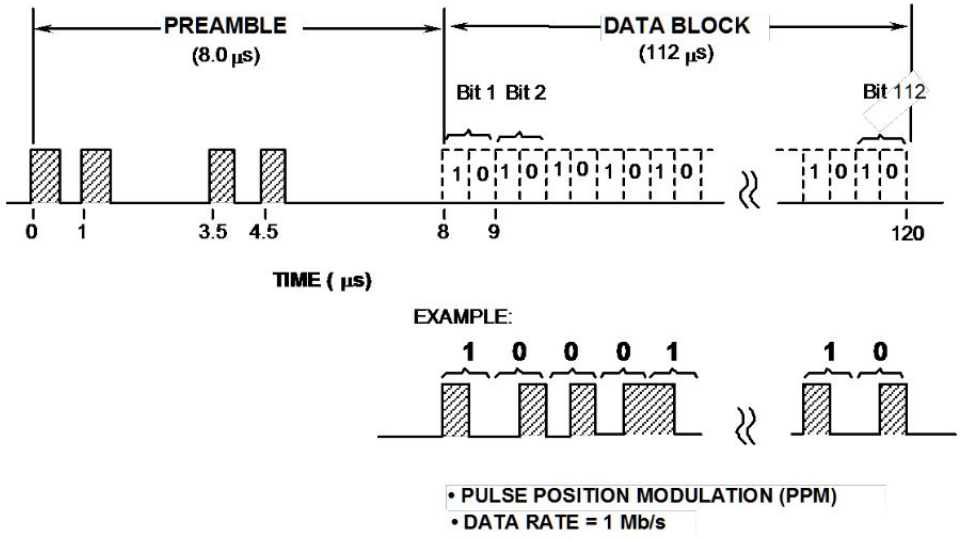

Figura N° 8: Descripción de una trama ADS-B

Pasando en limpio la información obtenida:

- Las tramas poseen una duración fija de 120 us, de los cuales los primeros 8 representan el preámbulo de trama
- Los siguientes 112 representan 112 bits de información (el tiempo de símbolo es de 1 us)

- Los datos se modulan en PPM binario. Si el ciclo de trabajo ocupa la primera mitad del símbolo, dicho símbolo representa a un 1 binario. Caso contrario, estamos hablando de un 0.
- El tiempo de subida no puede ser inferior a 0.05 us, ni mayor a 0.1 us.
- El ancho de banda permitido de la señal transmitida se encuentra entre 2.6 y 14 MHz (correspondiente a frecuencia de media potencia, o caída de 3 dB)
- La tasa de símbolo (que es igual a la tasa de bit, dado el tipo de modulación) es de 1Mbps

Nótese que en ningún momento se habla del proceso de generación de una trama ADS-B. Esto tiene base en que para el transcodificador esta información es transparente, su función es modular los bits de información y añadirles el preámbulo establecido.

Actividad N° 2: Utilización de la herramienta GNURadio y el hardware SDR

Para esta actividad, se requirió el desarrollo de etapas de radio aplicando conceptos aprendidos a lo largo de la carrera. La idea fué adquirir experiencia en el uso de la herramienta, con ejemplos tangibles en los que se aplicaran tanto los conceptos de las materias cursadas como así también los relativos a la naturaleza del funcionamiento de dicha herramienta.

Creación de repositorio en Gitlab

Antes de comenzar a trabajar en los ejemplos, se decidió crear repositorios en Gitlab; plataforma que permite crear repositorios privados de manera gratuita y con una interfaz sencilla. Lo primero es, obviamente, registrarse en la página.

Una vez que se haya completado el registro, hay que desplegar el menú “projects”, y luego hacer un click en el botón verde “New project”. La interfaz para crear un nuevo proyecto es intuitiva, no presenta mayores complicaciones.

De esta manera, se ha creado un proyecto en blanco. El desafío se presenta al vincular dicho proyecto con una carpeta existente en nuestro dispositivo, para lo que se deberán ejecutar los siguientes comandos en una terminal Unix:

```
1. cd /ruta/de/la/carpeta/deseada
2. git init
3. git remote add origin https://gitlab.com/user/proyecto.git
4. git add
```

```
5. git commit -m "Initial commit"
6. git push -u origin master
```

Enhorabuena! La carpeta ha sido vinculada exitosamente al proyecto. Para trabajar con dicho proyecto en otra máquina, se debe ejecutar el siguiente comando:

```
git clone https://gitlab.com/user/proyecto.git
```

En caso de que el proyecto sea privado, git solicitará usuario y contraseña del encargado de mantener el proyecto.

Ejemplo 1: Receptor FM

Teniendo en mente los preceptos discutidos al comienzo, se procedió a desarrollar un receptor de FM, con la convicción de que el ejemplo resultara tan completo como fuese posible. Está basado en una plantilla descargada de un blog [13], la cual luce así:

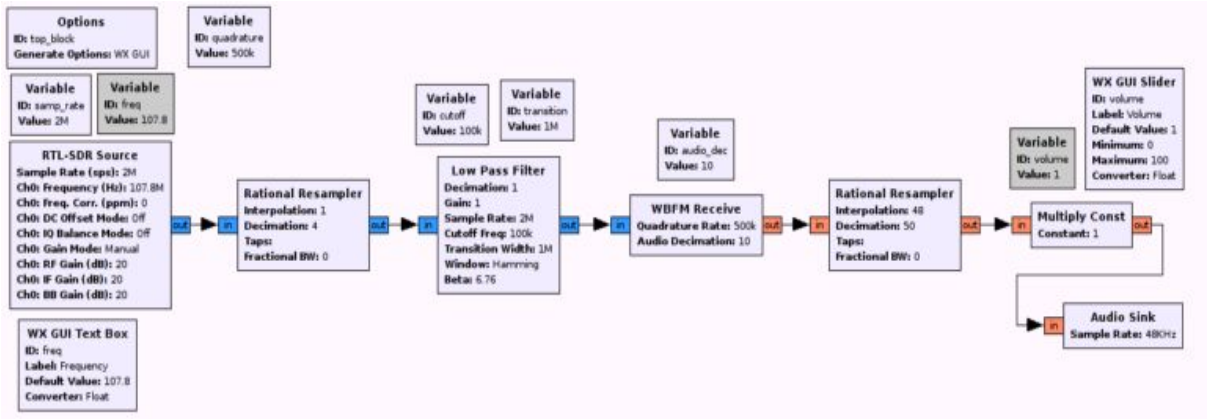

Figura N° 9: Esquema de recepción FM utilizado como base

A continuación, se presenta una captura del ejemplo terminado (Figura N° 10), sobre la cual se explicará el proceso de creación y los detalles respecto a la elección y configuración de los distintos bloques componentes:

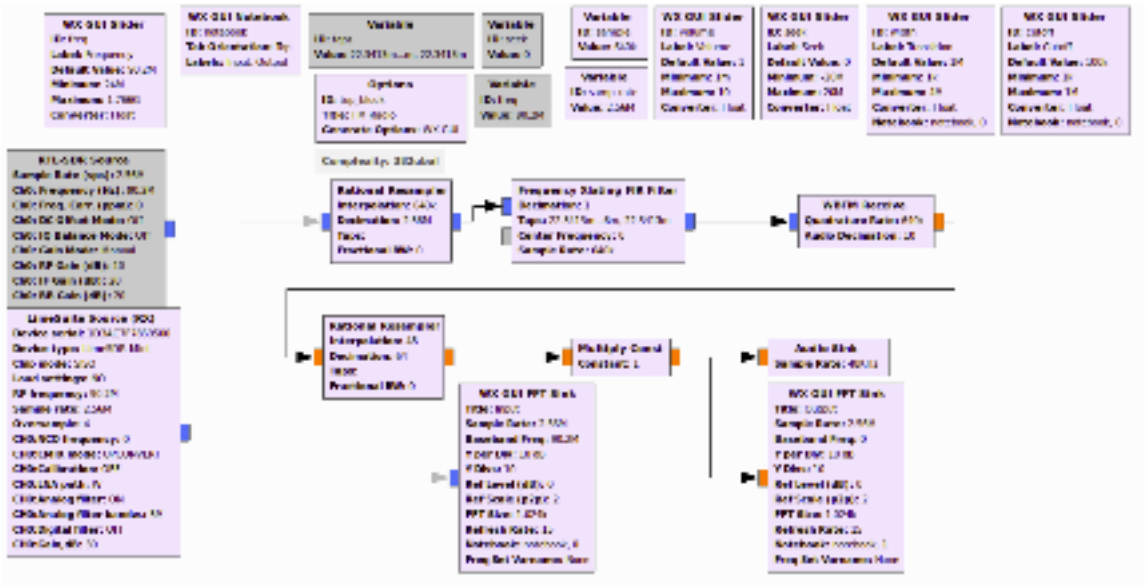

Figura N° 10: Esquema de recepción FM modificado

Como puede observarse, no se han realizado modificaciones vitales en el funcionamiento del receptor. En primera instancia, se modificó la tasa de muestreo del receptor de 2M a 2.56M por un problema de hardware descubierto empíricamente. El mismo fué descubierto al intentar sintonizar el receptor a una frecuencia de RF en el orden de la banda de FM, GNURadio exhibía un mensaje de error puntualizando el hecho de que el oscilador local no podía ser sintonizado a frecuencias mayores de 30MHz; dato irrisorio si uno contempla que en la descripción del hardware que posee LimeSDR-Mini el rango de frecuencias en las que trabaja oscila 10MHz y 3.5GHz.

Probando con una frecuencia mayor a 2MHz se solucionó el inconveniente, por lo cual se eligió la tasa antes mencionada.

El bloque “Frequency Xlating Filter” reemplazó al filtro pasabajos normal, debido a que el primero puede realizar además traslaciones en frecuencia. Esto permite que el receptor sea más funcional, de modo tal que el usuario pueda seleccionar la frecuencia de portadora a través de un deslizador gráfico. Los coeficientes del filtro se calcularon a través de la función firdes(), presente en el paquete de funciones del software.

Se utilizó un bloque demodulador de FM, debido a la imposibilidad de aplicar lazos realimentados en los esquemas a simular. Quizás podría haberse programado un bloque que realizara un tarea similar a la del demodulador, pero pareció una complicación que no era necesaria en estas instancias.

El plugin de recepción es simple de configurar. En Device type, se selecciona la opción LimeSDR-Mini; mientras que en Load settings se selecciona No, ya que es deseable que la placa obtenga los datos de configuración del plugin y no de un archivo externo. La opción de calibración no es necesaria. Por último, se hace uso del filtrado analógico que ofrece el hardware con un ancho de banda de 5 MHz. El filtrado digital se realiza con un bloque en etapas posteriores, pero tranquilamente podría realizarse con el dispositivo.

Por último, está el bloque “WX GUI Notebook”. Cumple la función de presentar la información en formato cuaderno, mostrando un gráfico por hoja. Esto es muy útil a la hora de presentar 2 o más gráficos, pudiendo hacerlo de forma ordenada y agradable a la vista del

usuario. Los bloques “WX GUI Slider” permiten ajustar el valor de una variable de manera “online”, deslizando un botón.

Las pruebas se realizaron originalmente en el laboratorio del GSTR, arrojando resultados negativos. Se repitieron modificando diversos valores del plugin de recepción, sin obtener mejoras significativas.

Utilizando las herramientas de análisis de señal que ofrece GNURadio, se observó que los niveles de señal eran más bajos de lo que esperaba. Se decidió entonces realizar pruebas en otro ambiente fuera del laboratorio. Los resultados fueron exitosos, por lo que se consideró que estaba listo para el segundo ejemplo.

Las Figuras N° 11 y 12 ilustran las afirmaciones realizadas en el párrafo anterior*:*

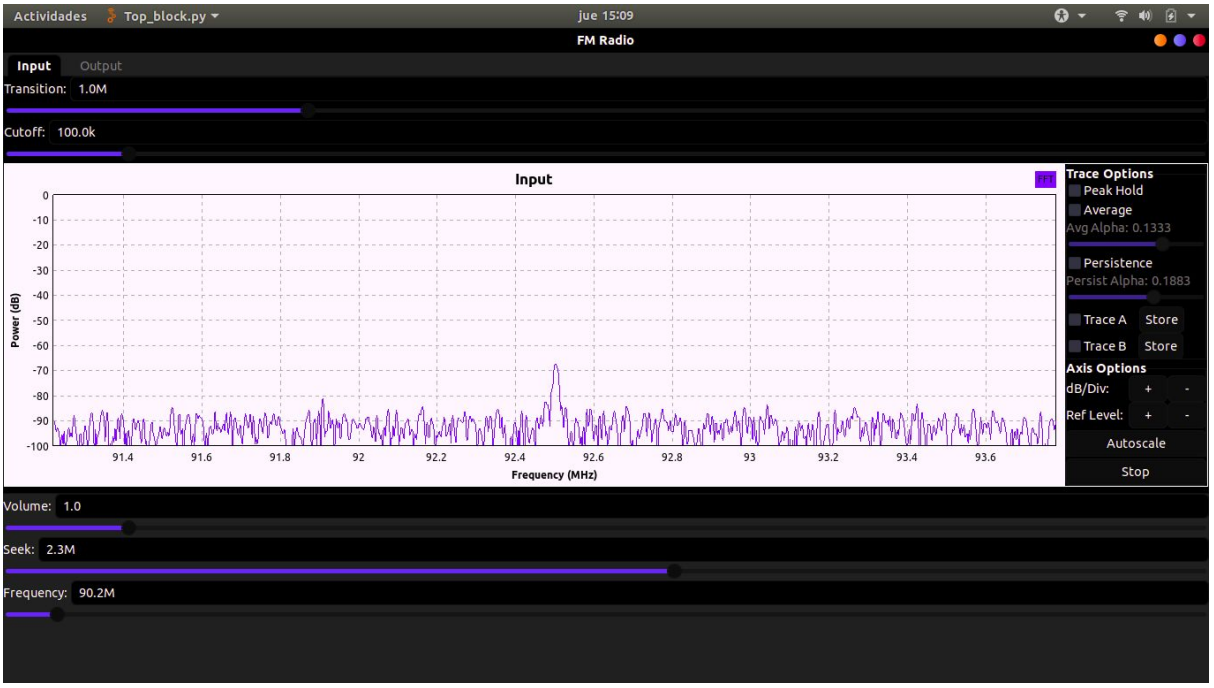

Figura N° 11: Representación en frecuencia de la señal de entrada al modulador

Figura N° 12: Señal de FM demodulada

Ejemplo 2: Loopback FM

Habiendo testeado el correcto funcionamiento de la etapa de recepción de la placa con el ejemplo anterior, era hora de hacer lo propio con la etapa de transmisión. Se pensó entonces que era una buena idea simular ahora un transmisor FM, de modo tal que se pudiese utilizar el ejemplo anterior para probar su correcto funcionamiento.

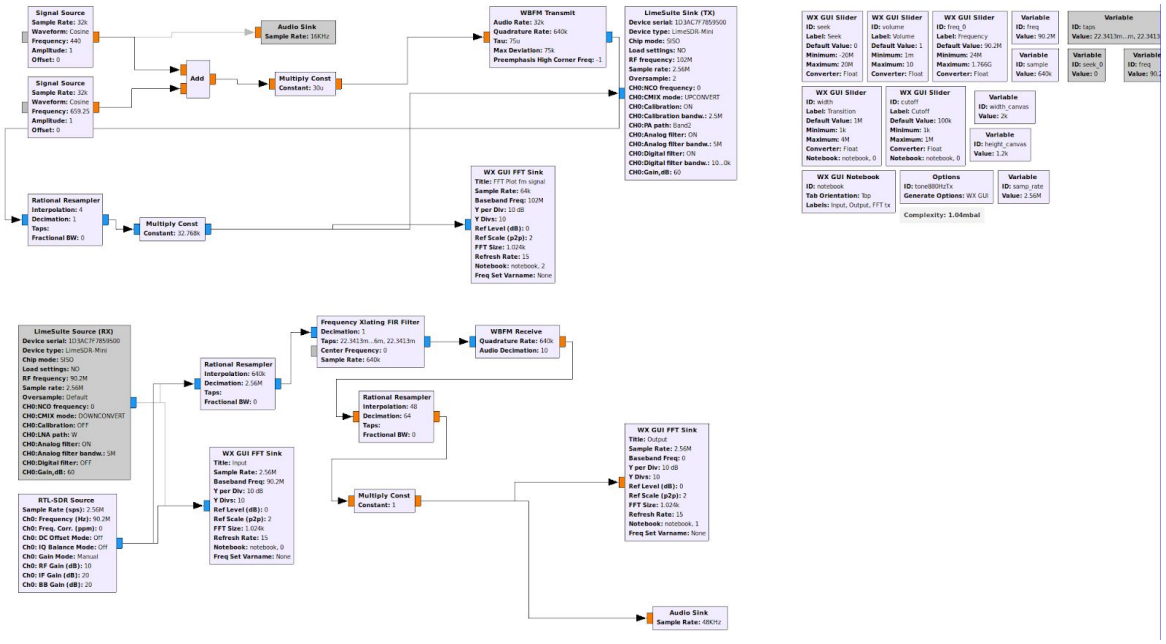

Figura N° 13: Esquema de transmisión y recepción FM

Para no generar complicaciones extras en esta etapa de la práctica, se decidió que la señal a transmitir fuese generada por 2 fuentes de señal provistas por la herramienta, quienes corresponden a un La en 3 octava y un Mi en 4 octava (Formando un acorde La5, muy utilizado en las canciones de rock). De esta forma, se transmite una señal musical agradable al oído y de fácil generación.

Esta señal ingresa al modulador de banda amplia, siendo multiplicada con anterioridad por una constante para disminuir su amplitud y evitar que por los productos cruzados se generen valores muy grandes. La máxima desviación de frecuencia se setea en 75 KHz, valor que representa al 100% de modulación en Argentina según ENACOM (Ente NAcional de COMunicaciones), la autoridad competente en el país.[14]

Una vez modulada, la señal ingresa a un bloque interpolador, de modo que la cantidad de valores por segundo que emita el modulador sea igual a la tasa de muestreo del transmisor. Con todos estos pasos realizados, sólo resta amplificarla, lo que se realiza multiplicándola por una constante que fué elegida empíricamente.

Se presentan a continuación capturas del sistema funcionando:

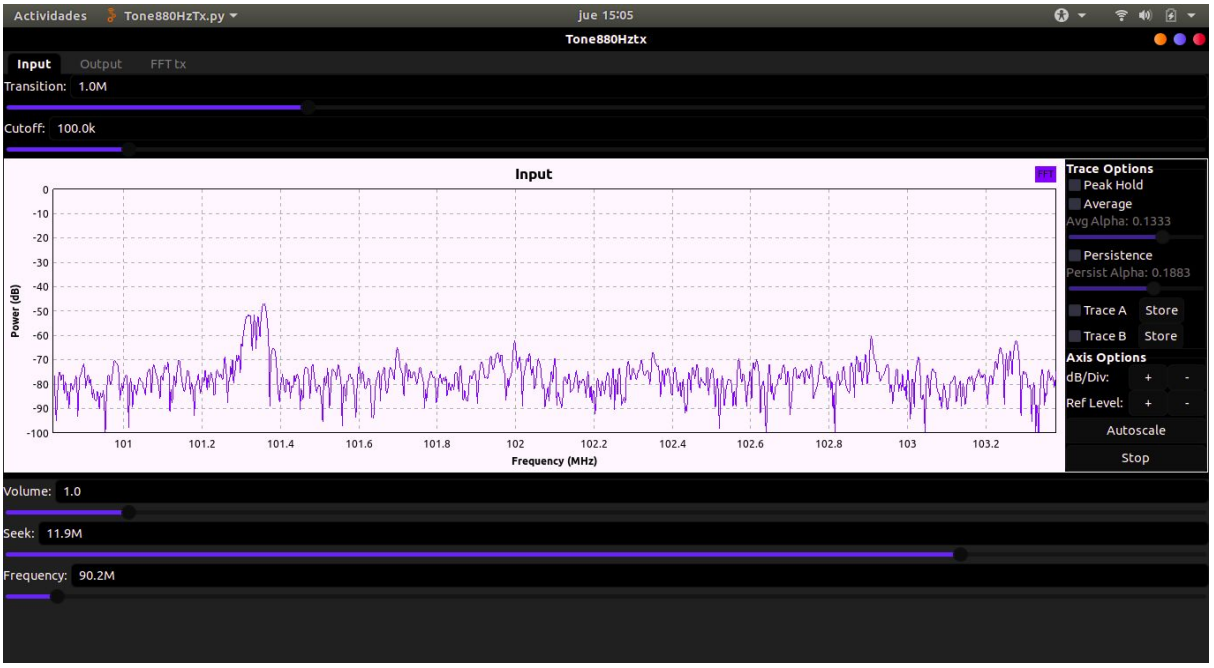

Figura N° 14: Espectro de la señal recibida. La portadora de la señal generada es 102 MHz

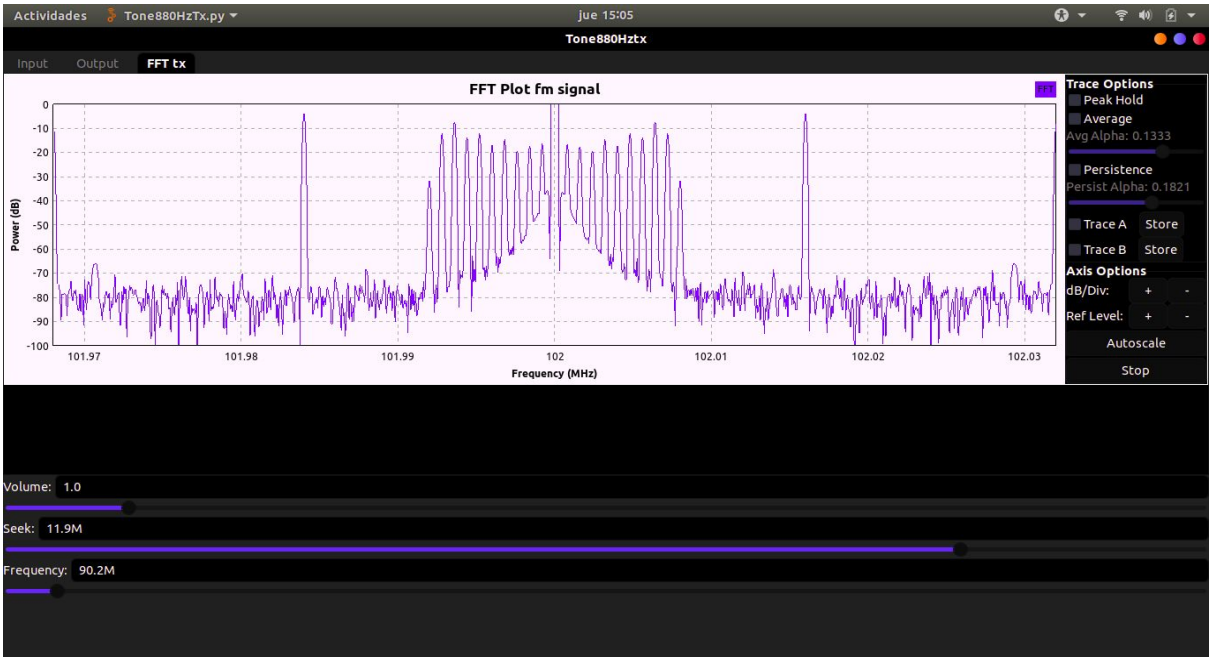

Figura N° 15: Representación espectral de la señal de audio modulada en FM

Actividad N° 3: Desarrollo y diseño de un sistema de radio

El objetivo planteado aquí fue diseñar la etapa de radiofrecuencia con la información brindada de los equipos de adquisición siguiendo el estándar ADS-B..

Teniendo en cuenta lo averiguado sobre el funcionamiento de un transponder ADS-B, se llegó a la conclusión que el primer paso que se debía tomar era el diseño de un modulador PPM, lo que se transformó en la piedra angular de todo el desarrollo. En primera instancia, se realizó una extensa búsqueda en la Web, cuyo objetivo fué un desarrollo similar al cual se propone como primer paso en esta tarea.

Desarrollo de un modulador PPM

Los resultados obtenidos no fueron satisfactorios, ya que lo más se acercaba al desarrollo deseado era un modulador ppm pensado para un avión a radiocontrol, sin brindar información de funcionamiento ni manipulación. Lejos de perder las esperanzas, se dispuso a instalarlo para hacer pruebas y tratar de sacar algo positivo de la experiencia.

Dado que dicho módulo no se encontraba entre los módulos disponibles para instalar con PyBOMBS, la instalación trató de llevarse a cabo por medio de la compilación de los archivos fuente. Esto sirvió como experiencia para descubrir que no es posible compilar archivos fuente en instalaciones de GNURadio hechas mediante el repositorio mencionado con anterioridad.

La solución lógica fué instalar GNURadio desde los repositorios de Ubuntu, lo cual solo trajo más complicaciones, ya que surgió un bug que impide ejecutar la interfaz gráfica de dicha herramienta. Se procedió a desinstalar GNURadio, e intentar reinstalar con PyBOMBS, sin éxito.

A fin de solucionar el problema de raíz, se realizó una copia de seguridad de todos los archivos importantes de la máquina y se instalaron copias nuevas tanto de Ubuntu como de GNURadio, pero esta vez se instaló desde los repositorios de Ubuntu. Esta vez se obtuvieron resultados exitosos.

Del siguiente enlace [15] se obtuvo el desarrollo nombrado unos párrafos atrás. Siendo compilado desde sus archivos fuente, pudo ser instalado correctamente, pero las sucesivas pruebas que se le realizaron no brindaron ninguna información útil. Lo que llevó a la conclusión que se iba a tener que desarrollar el modulador desde 0.

Este tutorial [16] brinda una guía muy completa de cómo desarrollar un módulo personalizado, permitiendo realizar la programación del mismo en lenguaje C++ o Python. Se eligió Python, dado que se posee mucha más experiencia en su uso.

El problema radicaba entonces en cuál camino seguir para desarrollar el modulador. Navegando por la web, se encontró este hilo de discusión [17], en el que un usuario solicitaba ayuda para realizar la misma tarea que se había propuesto. Dado que le recomendaron investigar los ejemplos de moduladores digitales presentes en el código fuente del software, se procedió a seguir dichos consejos.

Lo primero que uno advierte de dichos ejemplos es que se basan en la instanciación de clases para la generación de bloques, los cuales heredan ciertos atributos y generan los propios; demostrando que la programación orientada a objetos es el paradigma a utilizar.

Puede advertirse también que todos respetan el mismo formato de entrada y salida de datos; la entrada siempre consta de datos en formato de bytes y la salida en datos con formato de números complejos. De esta forma, pueden ser conectados directamente a una fuente transmisora, como la LimeSDR-Mini en nuestro caso.

Con respecto a la instanciación de clases, se advierte además que todos los moduladores digitales son instanciaciones de una clase padre llamada "***modulator***". Por lo tanto, el primer intento fué desarrollar el modulador ppm en base a dicha clase, ya que los métodos y atributos que posee resultaron muy interesantes.

Después de un análisis a conciencia de la situación, se descubrió que la única manera en la que se iba a poder realizar la acción antes descrita era creando una constelación personalizada. Se le dedicó un tiempo considerable a realizar esta acción, dado que si se lograba hacerlo se podría disponer de herramientas tan interesantes como el mapeo de los valores recibidos al punto de la constelación más probable.

Desgraciadamente, tras mucho reflexionar sobre el tema se concluyó que era inviable este camino. Fundamentalmente, por la diferencia de naturaleza entre las modulaciones digitales presentes en los ejemplos (QPSK, BPSK, GMSK, QAM) y la modulación PPM correspondiente al sistema ADS-B; dado que las anteriores poseen una relación 1 a 1 entre bit

y símbolo (por ejemplo, BPSK asigna 1+0j y -1+0j para 1 y 0, respectivamente) y para recrear un pulso PPM se necesitan al menos 2 muestras.

Por lo tanto, se basó el diseño en las clases genéricas de bloques (***basic_block***, ***decim_block*** e ***interp_block***). Lo positivo fué que la investigación anterior sirvió para definir los tipos de datos de entrada y salida.

En la actividad N° 1, se aprendió que la tasa de bit de un sistema es de 1 Mbps y que el tiempo de bajada y subida no debe exceder a 0.1 μs. Basado en ese conocimiento, se utilizó un bloque interpolador como cimiento del demodulador; dado que por cada bit de información se necesitarán un número a determinar de muestras que formen el pulso PPM. Si cada símbolo durará 1 μs, 10 muestras por símbolo significarán un tiempo de subida igual a 0.1 μs; de modo tal que la norma se respeta.

La primer versión constó de dos vectores de 10 muestras cada uno (representando el símbolo PPM para 1 y 0 respectivamente); se analizaba bit a bit el dato de entrada y se transmitía la información contenida en el vector correspondiente. Se fueron introduciendo mejoras, en pos de una mayor versatilidad del desarrollo, hasta llegar a la versión final; en la cual se tienen 2 argumentos modificables en el bloque.

Ellos son la cantidad de bits por dato de entrada, y la cantidad de muestras por semiciclo de símbolo. La razón del primer argumento es favorecer a la adaptabilidad del modulador a distintas fuentes de información, permitiendo distintos enfoques en las aplicaciones o desarrollos con los que se alimente al modulador. Para el segundo, se presentan motivos

similares, ya que es crucial que la resolución temporal del pulso de salida pueda ser modificada a voluntad.

El principio de funcionamiento es simple. En función de la cantidad de bits de información que posea la entrada, se realiza una comprobación con motivo de averiguar qué símbolo corresponde a cada bit. Una vez determinado el símbolo, se procede a escribirlo en la salida con una estructura de loop, valiéndose como referencia de la cantidad de muestras por semiciclo.

Diseño y desarrollo del transcodificador completo

Una vez que el modulador PPM estuvo funcionando correctamente, se encontró en condiciones de comenzar con el desarrollo del sistema completo. Comenzando con una versión prototipo que no recibe información de una fuente externa, en cambio, espera un valor específico (159, para ser específico) que actúa como flag de inicio.

La idea de este funcionamiento es conectar el sistema a un generador de números aleatorios, que simule el arribo de tramas al sistema; dado que dicho arribo no tiene por qué ser bajo un intervalo específico de tiempo.

A continuación, se presenta un boceto del sistema:

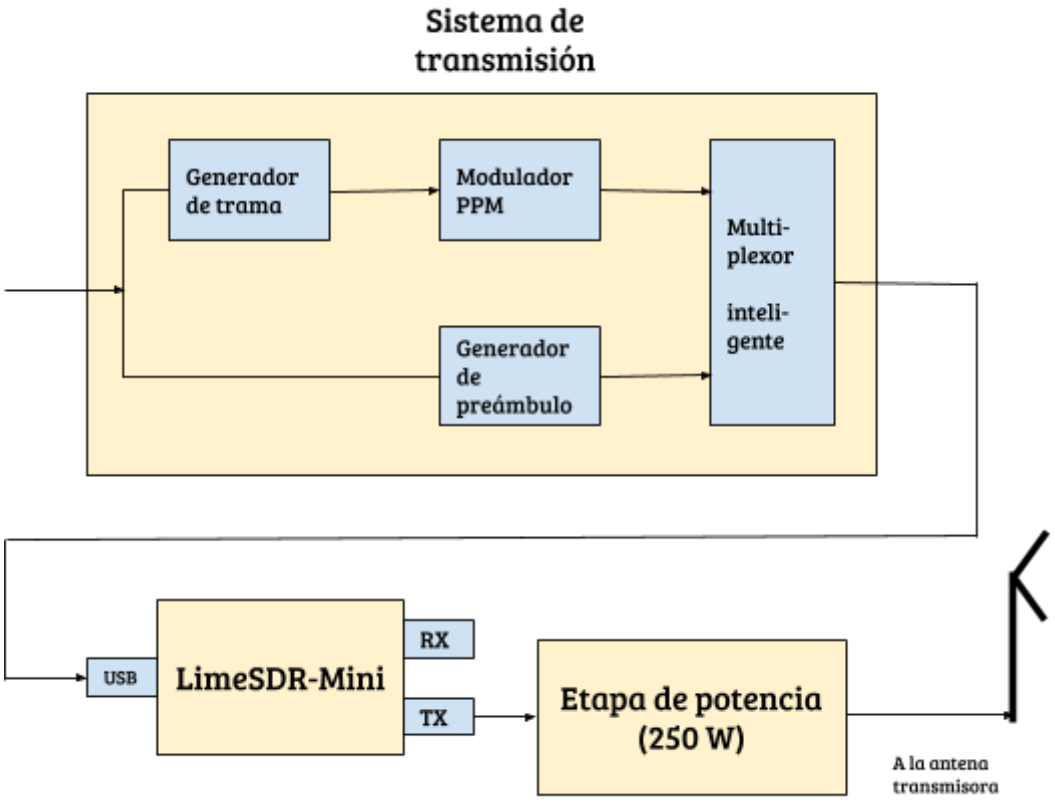

Figura N° 16: Prototipo de sistema completo, incluyendo la etapa de potencia

La primer reflexión que surge del análisis de este prototipo es que el corazón del sistema es el multiplexor inteligente, que se encargará de decidir qué información se envía y en qué momento hacerlo. Por ello fue que primero se trabajó en los generadores de trama y preámbulo, quienes no presentaron mayores dificultades.

El generador de trama está basado en un bloque interpolador, que realiza un escaneo elemento por elemento de los datos de entrada; en búsqueda del valor de inicio del cual se

habló unos párrafos atrás. Si el elemento analizado es dicho valor, se escribe la trama preprogramada en la salida, la cual es **[A000029CFFBAA11E2004727281F1]**. De lo contrario, se completará la cantidad de dígitos hexadecimales que componen a la trama con ceros.

El generador de preámbulo es muy similar en cuanto a funcionamiento. La principal diferencia es que este toma como argumento el número de puntos por semiciclo, además del valor de inicio. Basado en la cantidad de puntos por semiciclo, generará el preámbulo de trama ADS-B con una estructura de loop, el cual es independiente de la información que posee el mensaje.

Dado que el generador de trama discrimina los elementos de salida por dígito hexadecimal, se eligió una cantidad de 4 bits por muestra en el modulador PPM. La cantidad de muestras por semiciclo tanto de modulador como del generador de preámbulo se eligieron iguales, para que haya una equivalencia entre sistemas y la tarea del multiplexor inteligente sea un poco más sencilla.

Tanto el generador de preámbulo como el de trama dependen de la misma entrada, por lo que cuando ingrese el valor de inicio ambos producirán sus correspondientes salidas en simultáneo. Este hecho obliga que al menos una de las salidas sea almacenada en la memoria del sistema mientras se transmite la otra.

Con los conocimientos que se habían adquirido para ese entonces, significaba una tarea más simple almacenar la salida de ambos generadores para luego combinarlas y armar la trama ADS-B completa. Esto, sumado al hecho que tanto el preámbulo como la

información poseen longitudes fijas que están en función de la cantidad de muestras por semiciclo, dato que es conocido.

Una vez decidido el camino a tomar, se debía analizar la plausibilidad de seguirlo. La contra más significativa que este método ofrecía era que el sistema no fuese capaz de respetar las tasas de refresco de información, ya que un retardo mayor al que está exigido por norma en un sistema ADS-B puede provocar un desastre de magnitudes catastróficas.

Por lo tanto, se dispuso a analizar el material provisto por la cátedra en búsqueda de las distintas tasas de refresco de información. La tabla A-1 (Figura 17), presente en el apéndice A del draft proporciona dicha información:

Register number	Assignment	Maximum update interval
05_{16}	Extended Squitter Airborne Position	0.2 s
06_{16}	Extended Squitter Surface Position	0.2 s
07_{16}	Extended Squitter Status	1.0 s
08_{16}	Extended Squitter Identification and Category	15.0 s
09_{16}	Extended Squitter Airborne Velocity	1.3 s
$0A_{16}$	Extended Squitter Event-Driven Information	variable
10_{16}	Data Link Capability Report	≤4.0 s (see Note 2)
17_{16}	Common usage Capability Report	5.0 s
$18_{16} – 1C_{16}$	Mode S Specific Services Capability Report	See Note 5
$1D_{16}$-$1F_{16}$	Mode S Specific Services Capability Report	5.0 s
20_{16}	Aircraft Identification	5.0 s
30_{16}	TCAS Active Resolution Advisory	Annex 10, Vol IV §4.3.8.4.2.2
61_{16}	Emergency/Priority Status	1.0 s
62_{16}	Target State and Status Information	0.5 s
63_{16}-64_{16}	Reserved for Extended Squitter	
65_{16}	Aircraft Operational Status	2.5 s
66_{16}-$6F_{16}$	Reserved for Extended Squitter	

Figura N° 17: Intervalos máximos de actualización de información

La trama completa ADS-B dura 120 µs. Suponiendo un escenario desfavorable, se necesitará ese tiempo para almacenar la trama, que será el mismo que se utilizará para transmitirla una vez se haya realizado el procesamiento correspondiente; y supongamos que se utilice el mismo tiempo para realizar dicho procesamiento, lo cual es irrisorio dadas las velocidades de los procesadores modernos. Bajo estas condiciones, arbitrariamente desfavorables, son requeridos un total de 360 µs para realizar la transmisión de una trama.

Como puede notarse en la tabla anterior, de todos los intervalos de actualización el menor es de 0,2 s. El tiempo de procesamiento expuesto en el párrafo anterior representa un 0,06% del intervalo de actualización más exigente, por lo cual puede concluirse que el retardo por almacenamiento, procesamiento y transmisión de una trama que introduce este método al sistema es imperceptible. Es por ello, que el análisis dictamina que es plausible almacenar las salidas de los generadores de trama y preámbulo.

GNURadio permite analizar las distintas entradas de un bloque de manera individual, almacenando las muestras de cada señal en un vector. Valiéndose de esta característica, y de la simultaneidad con la que los generadores nombrados con anterioridad emiten sus salidas, se diseñó el algoritmo bajo el que funciona el multiplexor inteligente.

Con una estructura de loop, se analiza elemento a elemento la entrada correspondiente al generador de preámbulo. En caso de encontrar un valor distinto de cero, se almacena el índice correspondiente a la componente del vector de entrada cuyo valor es distinto de cero y se comienza un conteo de elementos que cumplan con esta condición. Si se cuentan 3

elementos, se considera el dato como un preámbulo válido y se setea la variable ***frame_tx_ready*** como **True** (dicha variable es un booleano).

La estructura de loop anterior recorre los datos almacenados de ambas entradas en su totalidad. Si la variable ***frame_tx_ready*** es **False**, simplemente se ejecuta el proceso nuevamente, con los nuevos datos que ingresan al bloque.

En cambio, si es **True**, se procede a armar la trama ADS-B. La variable ***new_root_index*** almacena el índice en el cual apareció el primer valor distinto de cero, el cual será utilizado como referencia. Es necesario también conocer cuál es la cantidad de muestras por semiciclo, puesto que dicho dato es vital para saber cuántas de ellas conformarán tanto el preámbulo como la información modulada en PPM.

Con la información nombrada en el párrafo pasado, están dadas las condiciones para armar la trama completa; ya que las longitudes de información y preámbulo son fijas. Un dato que queda para destacar del multiplexor, es que se basó en la clase ***basic_block.*** Esto significa que no existe una proporción entera entre la cantidad de muestras que salen y entran al bloque.

El motivo de ello es que las primeras versiones del desarrollo utilizaron la clase de bloques síncronos, pero se presentaron errores en tiempo de ejecución que acusaban un tamaño de buffer más pequeño del que la aplicación requiere. Dichos errores se le atribuyen a la complejidad en el procesamiento de muestras, característica que impide el establecimiento de una relación fija entre el número de muestras que entran y que salen del bloque.

El desarrollo y testeo de los bloques que conforman el sistema se llevó a cabo utilizando un tipo especial de bloques pre-instalados en GNURadio, que son los "Python block". Dichos bloques permiten ejecutar códigos simples, de modo tal que que se agilice la tarea de prototipado para el procesamiento de señales. Una vez que el desarrollo estuvo lo suficientemente avanzado, se comenzó con la tarea de investigación para el desarrollo de un módulo instalable que contuviese los bloques antes descritos, para darle un aspecto más formal al trabajo y facilitar la portabilidad del mismo.

La documentación provista por los desarrolladores de la herramienta posee un artículo dedicado específicamente a dicha tarea [18]. Si bien hace referencia en gran parte a código escrito en C++, es fácilmente aplicable a código escrito en Python.

Para llevarla a cabo, es preciso invocar por terminal al comando ***gr_modtool***, que es considerado por los desarrolladores como una "navaja suiza para la creación de módulos". Se muestra a continuación un resumen del tutorial disponible en el artículo citado en el párrafo anterior:

1. Abrir una nueva terminal, y ejecutar:

   ```
   gr_modtool newmod tutorial
   ```

2. El comando crea una nueva carpeta en el directorio en el que se encuentra al ejecutar el comando anterior, cuyo nombre será gr-"**nombredelnuevomodulo**". Se cambia a dicha carpeta, con el comando:

   ```
   cd gr-tutorial
   ```

3. Una vez en la carpeta creada con el comando “newmod”, se deben agregar uno por uno los distintos bloques que componen al módulo con el comando “add”:

```
gr_modtool add -t basic -l python
```

4. El programa responderá diciendo que identificó un módulo con el nombre que se le ha designado, y que efectivamente el lenguaje que se utilizará es Python. Luego, requerirá que se le brinde un nombre al bloque se añadirá, sin utilizar como prefijo el nombre del módulo contenedor:

```
Enter name of block/code (without module name prefix):
test_py_bc
```

5. El sufijo “py” denota que se trata de un bloque programado en Python, y “bc” significa que los datos de entrada está en formato de bytes; mientras que la salida está en formato complejo. Por último, solicita que se le indiquen la lista de argumentos que utilizará el bloque:

```
Enter valid argument list, including default arguments: 1 2 3
```

6. El próximo diálogo permite añadir un script de python para realizar un test QA (Quality Assurance). Se puede saltear, dado que el código fué probado con anterioridad. La herramienta mostrará una salida similar a esta:

```
Adding file 'Python/test_py_bc.py'...
Editing Python/CMakeLists.txt...
Adding file 'grc/tutorial_test_py_bc.xml'...
Editing grc/CMakeLists.txt...
```

Con los pasos anteriores, bloque de prueba será añadido al módulo. Solo falta añadir los demás bloques que componen al sistema.

Una vez añadidos, resta trabajo por realizar. Si se ejecuta el comando `ls` en el directorio creado para el tutorial, se tendrá un resultado del siguiente estilo:

```
gr-tutorial$ ls
apps cmake CMakeLists.txt docs examples grc include lib Python swig
```

Las subcarpetas “Python” y “grc” serán de especial interés. En “Python”, se encuentran los archivos .py que usará GNURadio para realizar el procesamiento de señales en tiempo de ejecución. La herramienta gr-modtool crea dichos archivos con una plantilla, por lo cual nada del código que el usuario ha escrito se ve reflejado en ellos; razón que obliga a que el usuario los modifique manualmente. No es una tarea complicada, basta con copiar y pegar el desarrollo realizado, teniendo en cuenta las ligeras diferencias que existen entre las clases y métodos que se utilizan en los bloques creados por la interfaz gráfica de GNURadio y los que se utilizan en los bloques que se compilan con los módulos instalables.

Por último, en la carpeta grc se encuentran los archivos .xml que corresponden a cada bloque. Dichos archivos serán utilizados por la interfaz gráfica para mostrar el bloque instalado por pantalla, con sus entradas, salidas y argumentos. Esta tarea es poco intuitiva, pero por medio de la prueba y el error se completó exitosamente.

El módulo se encuentra ahora listo para ser instalado, compilando los archivos fuente que se han creado; siguiendo el proceso de instalación del plugin de la placa LimeSDR-Mini.

La Figura N° 18 muestra una implementación prototipo del sistema, en la cual se han utilizado los bloques contenidos en el módulo instalado:

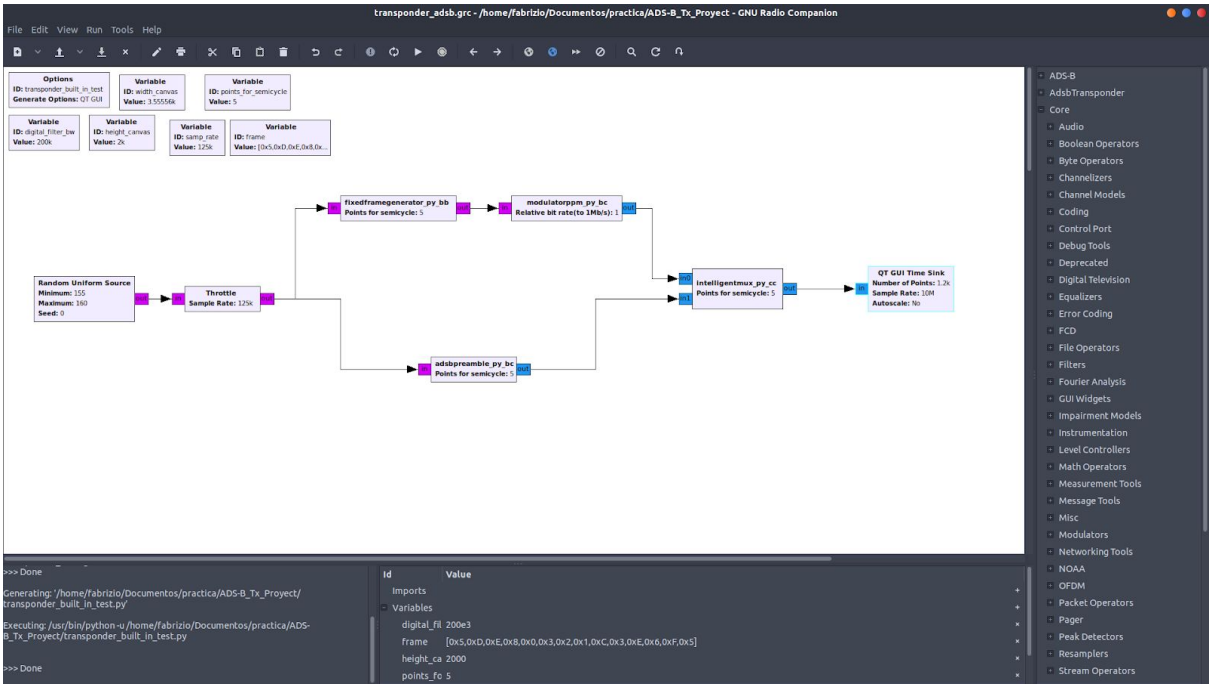

Figura N° 18: Diagrama prototipo implementado en la interfaz gráfica de GNURadio

La Figura N° 19 muestra la señal de salida del sistema, modulada en PPM binario según especificaciones normativas y con el preámbulo añadido:

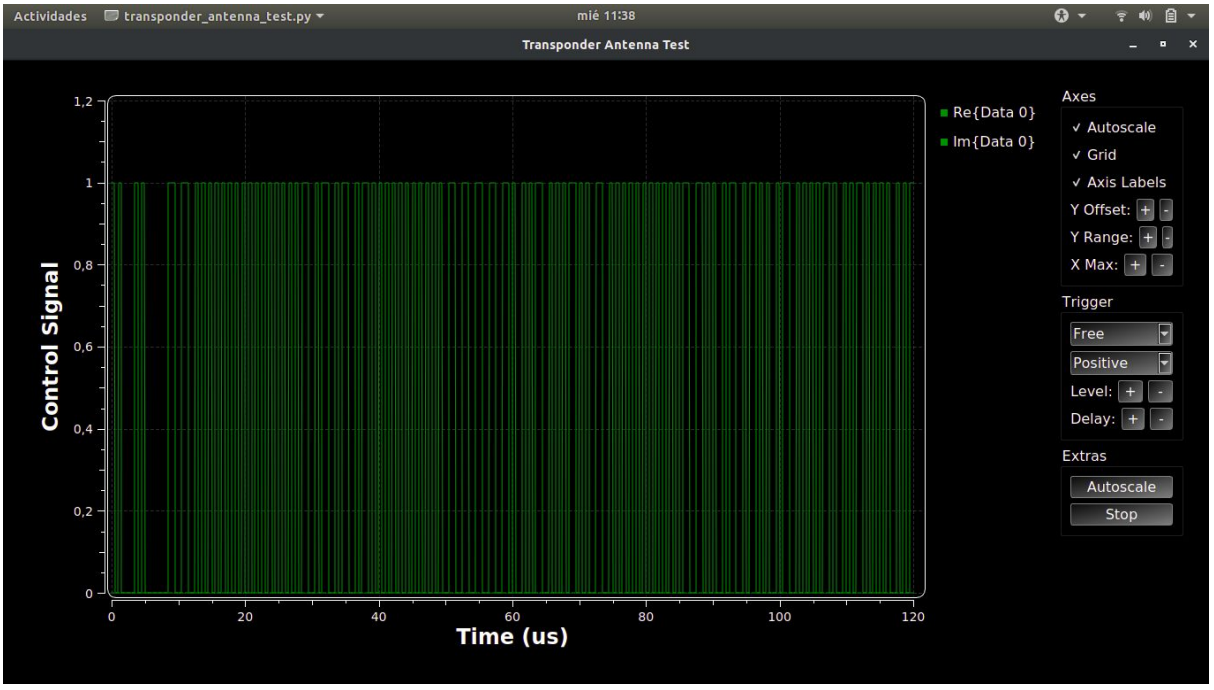

Figura N° 19: Trama ADS-B completa, cuya información es [A000029CFFBAA11E2004727281F1]

La Figura N° 20 es un acercamiento de la Figura N° 19, en la que se permite apreciar que la forma del preámbulo respeta los tiempos establecidos por norma

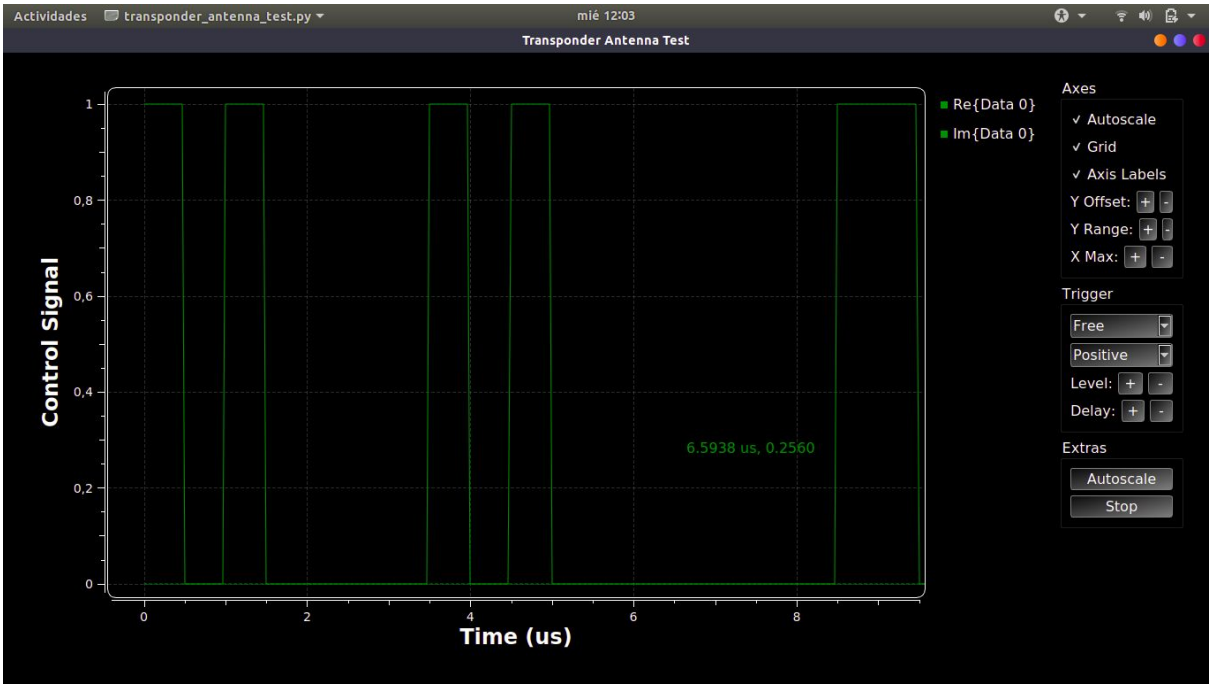

Figura N° 20: Preámbulo de una trama ADS-B

En este punto, se dió por concluída la 3er actividad y se procedió a comenzar con las tareas relativas a la N° 4

Actividad N° 4: Prueba de funcionamiento a nivel inicial

El objetivo de dicha actividad fue realizar medidas con equipos disponibles en los laboratorios de la facultad, de modo tal que se pudiera validar el diseño llevado a cabo en la actividad anterior y comprobar que efectivamente cumple con los requisitos exigidos.

Dado que se encontraba en posesión de una sola placa, los primeros ensayos se realizaron utilizando a la misma LimeSDR como estación transmisora y receptora.

Montando etapas de TX y RX en un mismo dispositivo

El sistema montado en la actividad N° 3 fue pensado para realizar una prueba a nivel de software de su funcionamiento. Para poder cumplir con el objetivo de esta actividad, fueron necesarios ciertos cambios. Se muestra a continuación los resultados de dichos cambios, y se utilizará la Figura 21 para explicarlos:

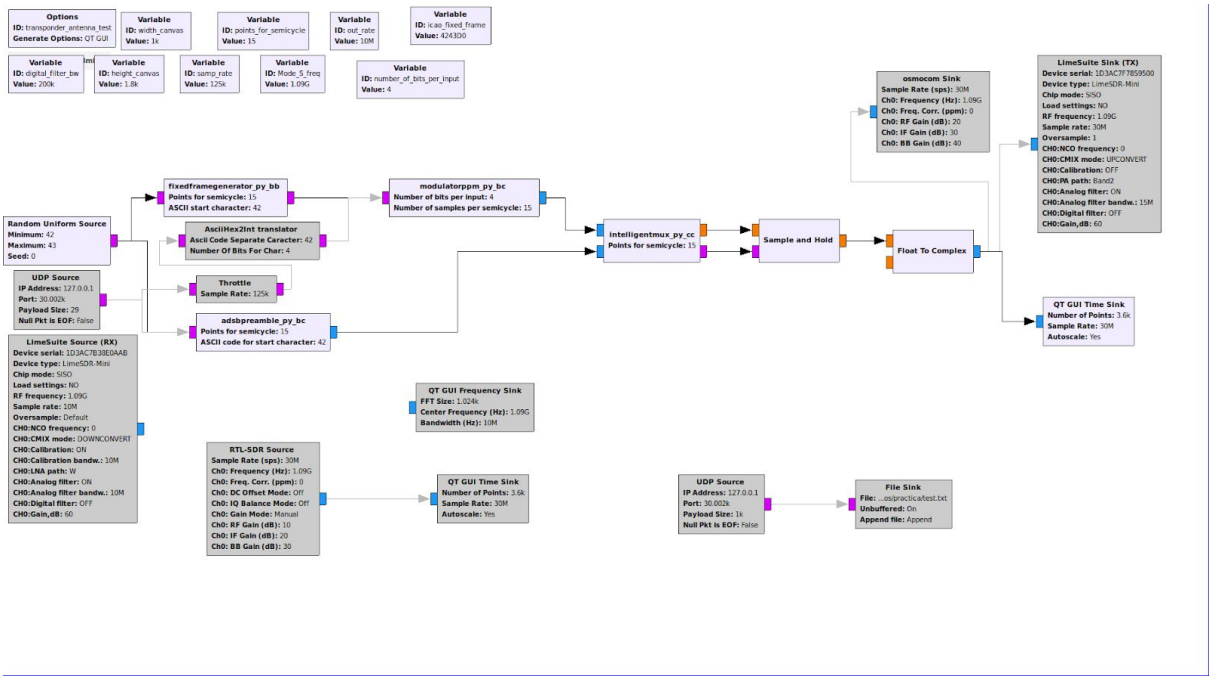

Figura N° 21: Sistema de transmisión modificado

Primeramente, cabe destacar que el valor de inicio cambió de 159 a 42. Esto se dió porque el código ASCII correspondiente a un asterisco (*) es 42, y se pretendía recibir información a través de un puerto UDP por medio de una aplicación que separe las distintas tramas ADS-B con asteriscos; motivo por el cual también se agregó una fuente de datos UDP y un bloque que traduce caracteres ASCII a su valor hexadecimal (A será traducido a 0x10, B a 0x11 y sucesivamente). Dicho desarrollo no llegó a buen puerto, dadas las complicaciones que surgieron luego.

El próximo detalle que se considera de interés es el añadido de un retenedor de orden 0, y la consiguiente modificación del multiplexor inteligente.

Dicho retenedor posee dos entradas, una de señal de información y otra de control. Cuando la señal de control posee un valor distinto de cero, el dispositivo toma una muestra de la señal de información en el próximo ciclo de reloj; y retiene la última muestra tomada en caso de que la señal de control sea cero.

La modificación realizada al multiplexor no fué sustancial, se añadió una variable que almacena la última muestra transmitida; la cual se compara con la muestra a transmitir. Si son iguales, se emite un cero por la señal de control; caso contrario se emite un uno.

Uno podría preguntarse, cuál es el motivo que lleva a añadir otra etapa más a un sistema que virtualmente funciona de manera correcta? La respuesta es que en las primeras etapas de prueba de emisión de señal, se recibieron señales extremadamente caóticas, que no se parecían en nada a la información modulada en PPM que se vió en los ensayos de la actividad N° 3. Se consideró que una de las razones posibles era un problema de sincronismo

entre la emisión del valor de señal discreto del multiplexor y el muestreo de la placa, por lo cual un retenedor de orden cero prometía ser la solución al problema.

A continuación, se muestran los resultados obtenidos de las pruebas:

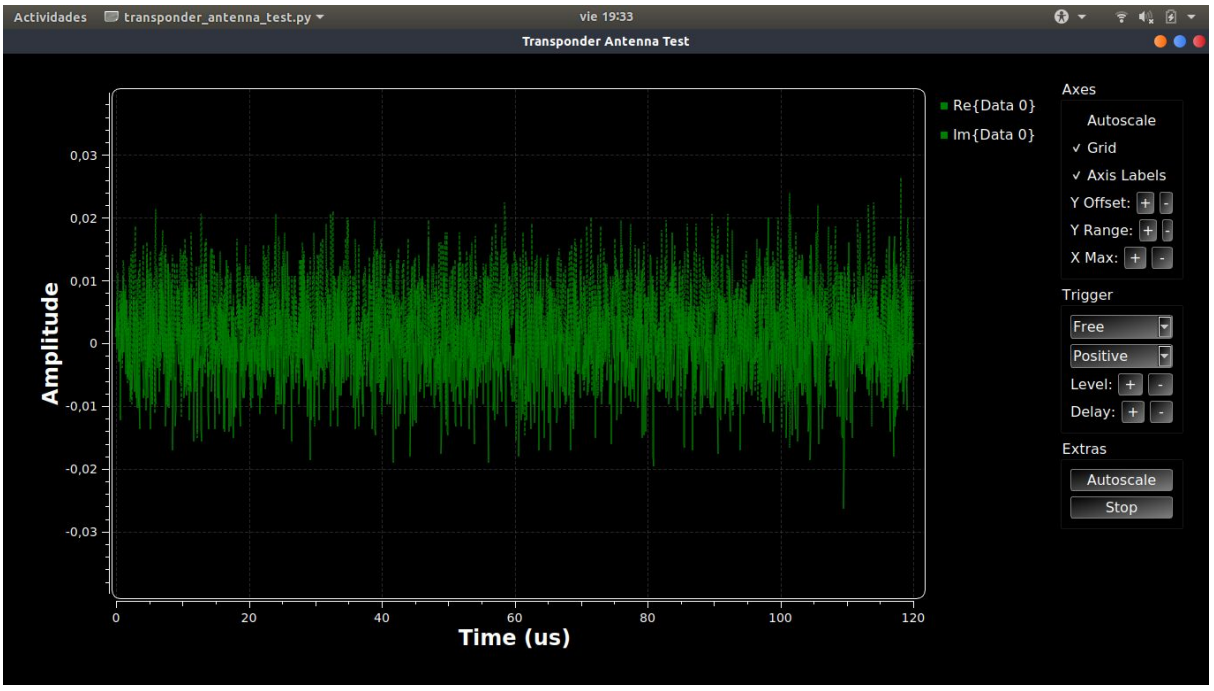

Figura N° 22: Representación temporal de la señal recibida

En la Figura N° 23, se muestra una representación en frecuencia de la señal recibida:

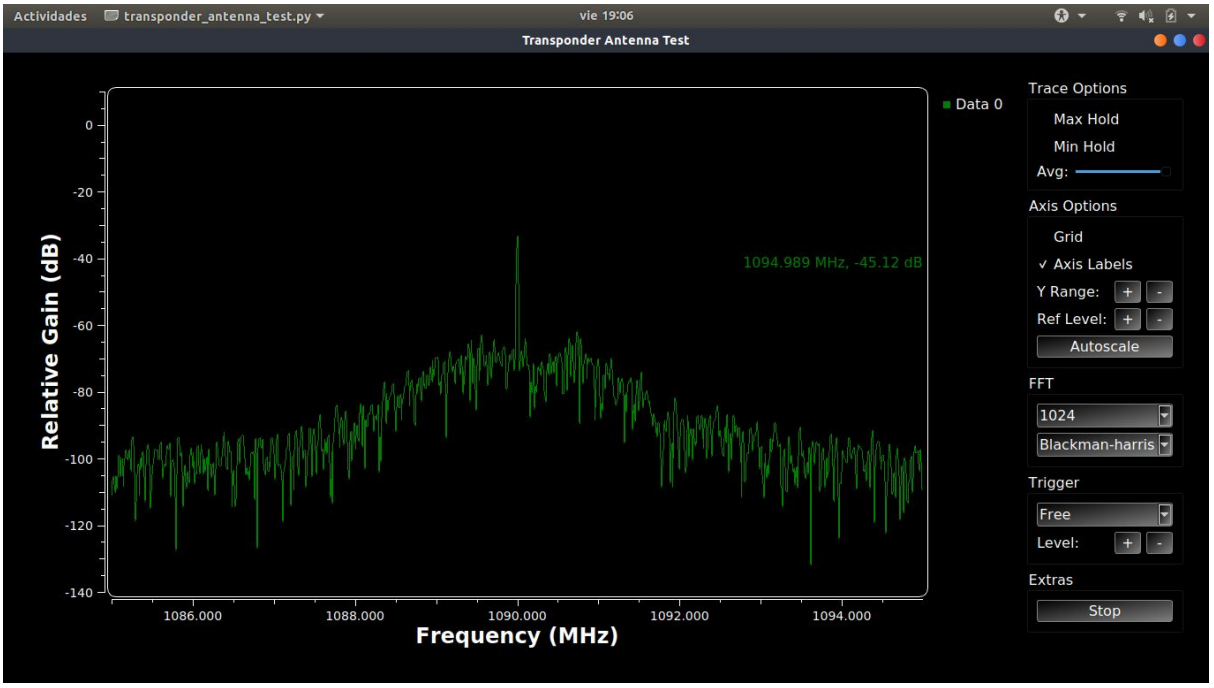

Figura N° 23: Representación espectral de la señal recibida

Es notable en la figura anterior, que la LimeSDR-Mini no tiene la suficiente potencia como para transmitir y recibir a la vez, si se utiliza la ganancia que se le ha configurado. Las pruebas se realizaron con la placa enchufada en un puerto USB 3.0, lo que significa que la potencia con la que se alimentó fué la máxima posible.

Montando etapa de TX en LimeSDR y RX en foxwey

Se le solicitó al tutor un segundo equipo, para resolver el problema de la potencia y poder ubicar cuál era el verdadero problema que producía que la señal se volviese así de

caótica. Mientras se tramitaba dicho préstamo, el tutor facilitó otro equipo para que se continuase con los ensayos, el FoxWey fsdr02:

FIgura N° 24: Receptor SDR Foxwey FSDR02

Dicho equipo tiene una banda de recepción de 100KHz-1.7GHz [19], la cual es más que suficiente para recibir y demodular una señal de ADS-B. El transmisor y receptor que se creó para estos ensayos, se modificó de forma tal que se utilice el anterior dispositivo para realizar tareas de recepción y la LimeSDR-Mini para tareas de transmisión. Los resultados obtenidos muestran que si bien se soluciona el problema de nivel de señal, la misma continúa adquiriendo formas caóticas una vez transmitida; como muestran las capturas siguientes:

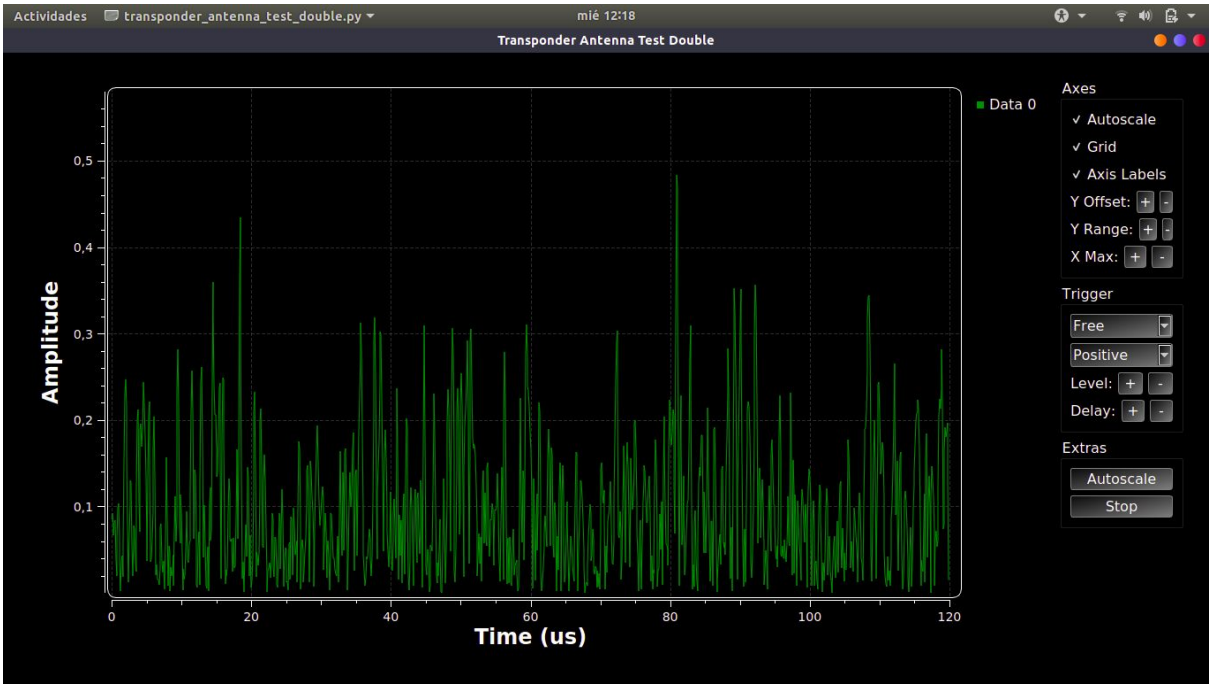

Figura N° 25: Señal recibida con Foxwey FSDR02, representada en tiempo

Se muestra en la Figura N° 26, una representación espectral de la señal recibida:

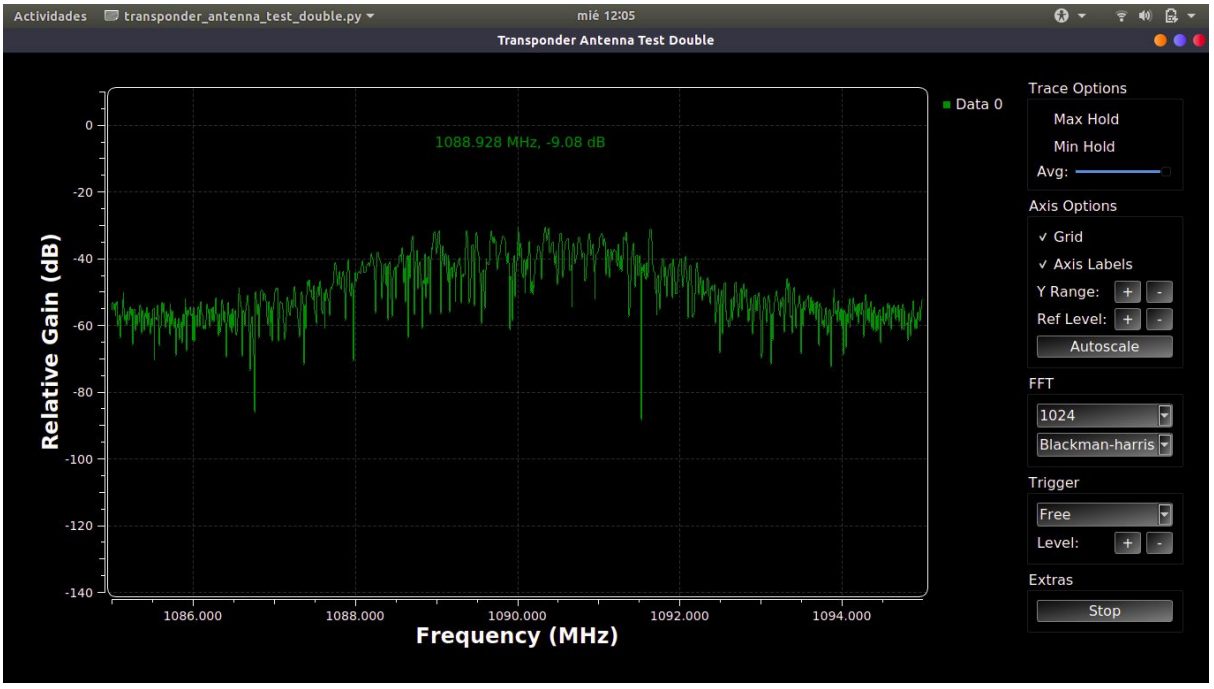

Figura N° 26: Espectro de la señal recibida

El espectro de la señal recibida brindó un dato esperanzador: la frecuencia de portadora es la correcta. Las pruebas a nivel de software mostraron que la señal adquiere la forma de onda que se intenta que obtenga, lo que significa que la falla se encuentra en algún punto entre que la señal ingresa al transmisor situado en la LimeSDR-Mini y es recibida por el otro dispositivo.

Dado que no estaba completamente eliminada la opción de que el fallo proviniera de la etapa de transmisión o recepción, se comenzó con el desarrollo de una etapa que adquiriese

información a través de un puerto UDP y la convirtiese en datos legibles por el modulador PPM.

Allí fué cuando se diseñó el bloque que se encuentra deshabilitado en el diagrama principal, ASCIIHex2IntConverter. La idea de un nombre tan largo es que describa la función a cumplir, que es transformar la información (que son dígitos hexadecimales en formato de caracteres ASCII) en enteros.

La versión preliminar constó de una comparación del dato de entrada con los elementos de un vector (hardcodeado). Este vector contiene cada carácter ASCII en la posición correspondiente a su valor entero, y en la última posición contiene el carácter de inicio de trama. Ya se había dicho que este carácter es el asterisco.

La razón de esa elección es debido a que se pensó alimentar al sistema con una aplicación que tome datos de un simulador de vuelo y los convierta a tramas ADS-B, adoptando el formato “raw beast”. Dicho formato consiste en convertir la trama a dígitos hexadecimales, siendo que dichos dígitos se representarán con el carácter ASCII correspondiente, y colocar un asterisco al inicio de cada una de ellas. Por ejemplo: *A000029CFFBAA11E2004727281F1

Habiendo cumplido con este requisito, el próximo paso fue añadirle cierta inteligencia al bloque, para que pudiese reconocer que el asterisco es el carácter que denota el inicio de trama. Esta función es necesaria porque la conversión de datos solo debe realizarse en caso de que arriben tramas válidas, y sólo de la cantidad correcta de datos. Con el carácter de inicio,

se le brinda la referencia necesaria al bloque para que cumpla con la tarea asignada de manera satisfactoria.

Esta tarea quedó inconclusa, ya que surgieron problemas de dudoso origen durante la conversión de los datos. No se pudo identificar la fuente de dicho problema durante el tiempo que se le dedicó, y cuando el tutor proveyó el segundo dispositivo que se le había solicitado esto pasó a segundo plano. Se brinda el código junto con los diagramas principales, para que futuros practicantes logren descubrir cuál es el error y puedan transmitir información al sistema por medio de red.

Montando las etapas de TX y RX en distintas LimeSDR-Mini

Como se puntualizó en la sección en la que se describió el plugin para GNURadio, cada bloque que interactúe con una placa LimeSDR-Mini deberá identificarse con el número de serie correspondiente. Gracias a esta característica, es posible montar las etapas en distintos dispositivos.

Con esto en mente, se modificaron los bloques encargados de comunicarse con los dispositivos para que cada placa realice una de las tareas; pero utilizando el multiplexor inteligente de salida única. Ya se había dicho con anterioridad que las pruebas no mostraron los resultados esperados.

A continuación, se muestra una representación temporal de la señal recibida:

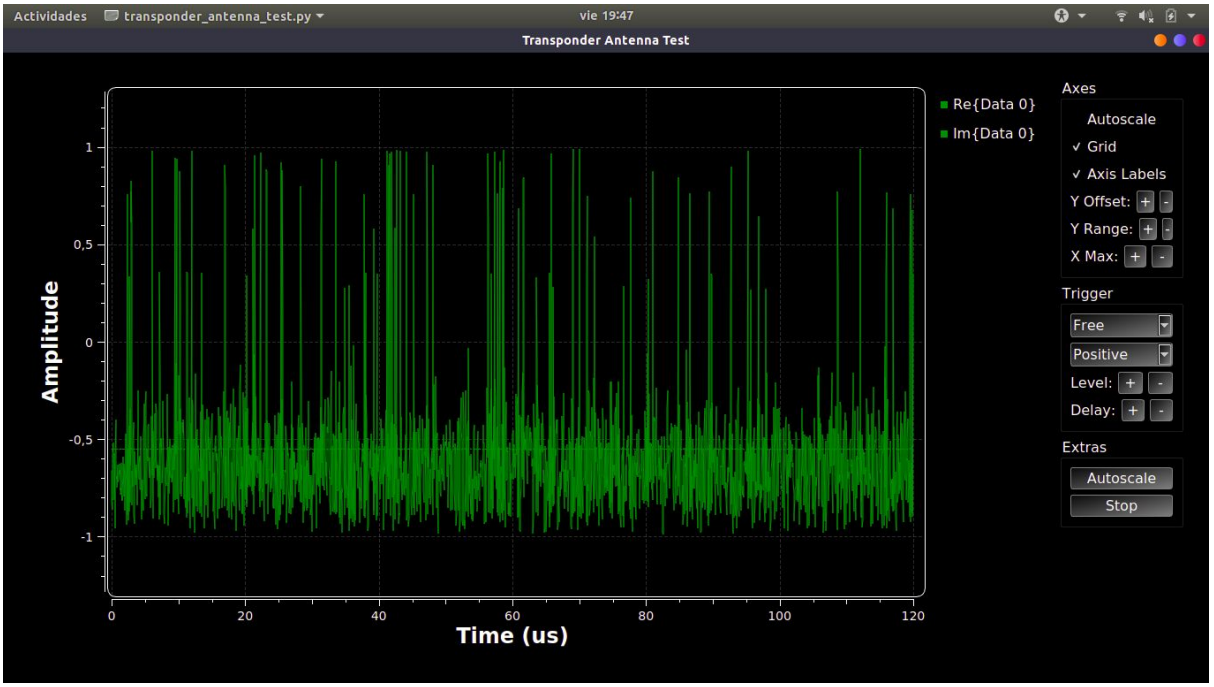

Figura N° 27: Representación temporal de la señal recibida, ahora con los niveles de señal esperados

Las imágenes hablan por sí solas, es notorio que los niveles de señal son los esperados mas la forma de onda parece completamente aleatoria.

Lo positivo fue que se resolvió el problema de nivel de señal. Por lo tanto, se comprobó empíricamente que el error de diseño está en algún punto entre que se genera la señal a enviar con el modulador inteligente y se envía a través del dispositivo elegido para llevar a cabo las prácticas.

Surgieron muchos interrogantes. Como primer hipótesis, se planteó el hecho de que quizás había que multiplicar la señal PPM por una portadora con la frecuencia de trabajo del sistema ADS-B (1090 MHz) ya que la placa no tenía por qué realizar esa operación de manera automática. Se comprobó que estaba equivocada dicha hipótesis cuando se investigaron los ejemplos preliminares, la transmisión loopback de FM no multiplica a la señal que sale del modulador por una portadora y sin embargo los resultados son los esperados.

En un afán de proponer soluciones novedosas, se aplicaron cambios al multiplexor inteligente, con el afán de eliminar un supuesto problema de sincronismo entre modulador y placa SDR (se le añadió una señal de salida, de forma tal que se pudiese comandar el retenedor de orden cero mencionado con anterioridad). Desgraciadamente, la forma de onda se mantuvo aleatoria.

La Figura N° 28 muestra una captura de la señal recibida, en función del tiempo:

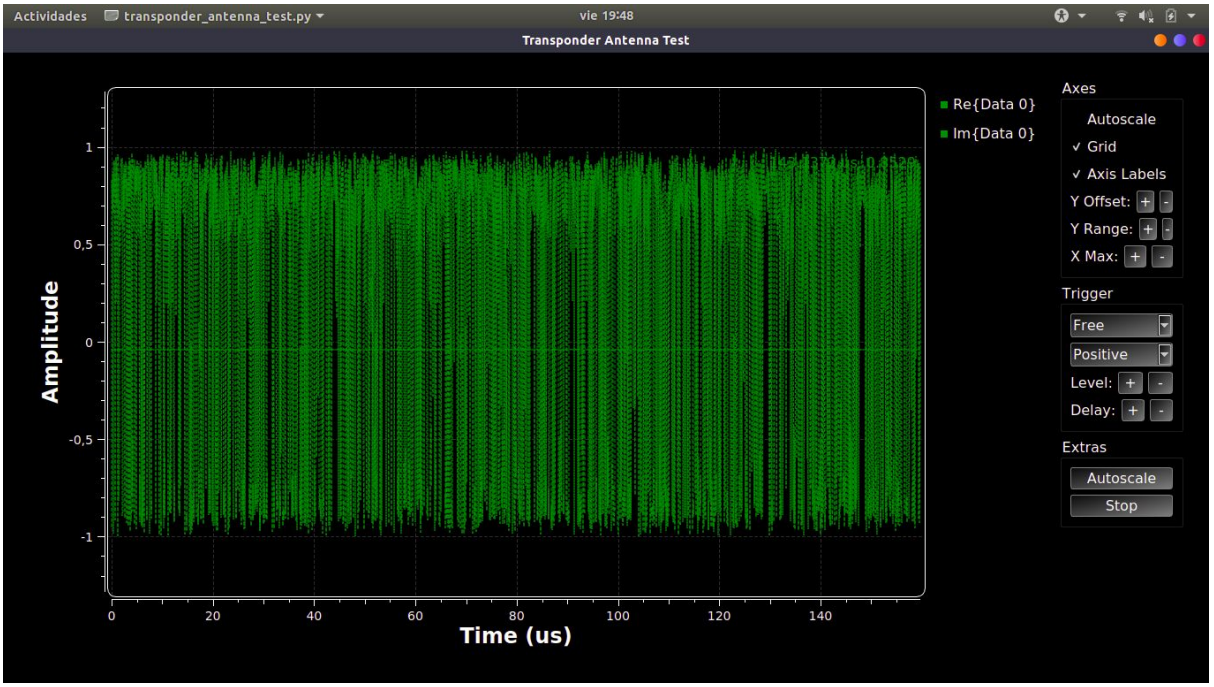

Figura N° 28: Representación temporal de la señal recibida

Si bien la Figura N° 28 parece mostrar que la forma de onda de la señal recibida es la esperada, al realizársele un acercamiento se comprueba que sigue siendo aleatoria, como bien lo muestra la Figura N° 29:

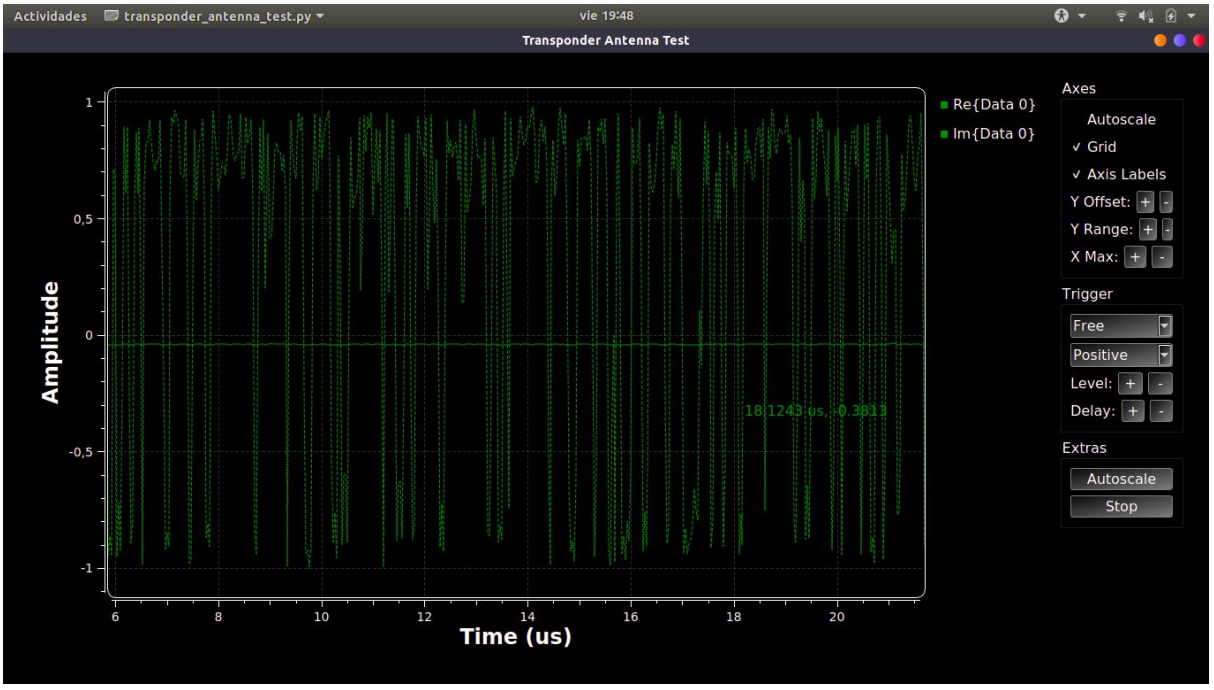

Figura N° 29: Acercamiento de la Figura 28. La forma de onda sigue sin ser la esperada

En la Figura N° 30 se muestra la representación espectral de la señal recibida, demostrando que la energía de dicha señal se encuentra en las frecuencias correctas:

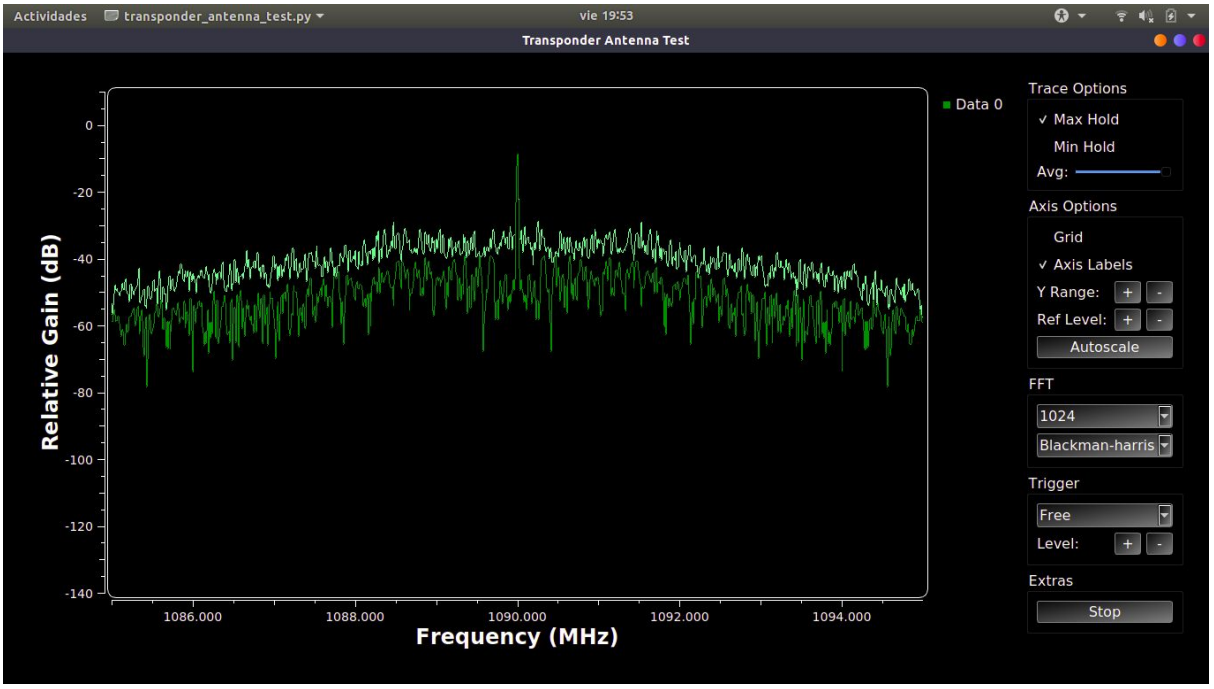

Figura N° 30: Representación espectral de dicha señal. En un tono de verde más claro se muestran los valores pico

La presencia de energía espectral alrededor de la frecuencia de portadora se encargó de proveer una luz de esperanza, trayendo la idea de que quizás la señal si tenía la forma de onda prevista; solo que el método de demodulación era el incorrecto.

Para aceptar o descartar dicha hipótesis, el tutor facilitó un dispositivo que forma parte del proyecto de investigación al que se anexará este trabajo. El mismo emula un receptor ADS-B.

Dicho dispositivo tiene un puerto de comunicación RJ45, con el que se conecta a la PC y permite visualizar en un mapa interactivo la información de los aviones que se encuentran en las cercanías.

Tristemente, se repitieron los ensayos descritos en párrafos anteriores pero no fue posible visualizar la información que se emitía a través del sistema diseñado.

Se continuaron las pruebas, se introdujeron pequeños cambios en el sistema; pero la fecha final llegó y el error de transmisión no pudo resolverse.

Conclusiones

Conclusión del trabajo

Al comentar los objetivos de la práctica profesional, se hizo un hincapié especial en que uno de los objetivos implícitos primordiales era que el alumno conociese una minúscula parte del gran abanico laboral que ofrece Ingeniería en Telecomunicaciones. Este objetivo puede considerarse cumplido, ya que el transcurso de los días que se correspondieron con la ejecución de dicha práctica me enfrentó con una buena parte de los pormenores que ocurren en un grupo de investigación.

Asimismo, la cordial relación con los demás profesionales brinda una experiencia en relaciones humanas que no siempre recibe la importancia que realmente se merece. En el ámbito ingenieril, es inconcebible que un profesional encare en solitario un proyecto; siempre se forman equipos de trabajo en los cuales no solo hay que conocer el procedimiento para realizar una tarea, sino que también se debe poder explicar y defender dicho procedimiento.

Siguiendo con ese tema, es considerable también adquirir la capacidad de ser objetivo. Porque no solo debe defender sus ideas el ingeniero, debe saber que los debates en los cuales defiende sus ideas son en pos de encontrar la verdad y no de tener la razón. Un buen ingeniero necesita reconocer cuando está equivocado, asumir dicho error y aceptar una solución proveniente de un colega si esta es acertada.

Respecto al uso de la herramienta GNURadio, es notorio que posee un potencial que va más allá de lo que uno puede imaginarse. Su interfaz gráfica intuitiva, su gran comunidad

online, su filosofía de software libre y el extenso material que proveen los desarrolladores son características que la convierten en la herramienta por excelencia cuando se desea prototipar un sistema de radiofrecuencias de media o alta complejidad.

El sistema que se diseña con la interfaz gráfica, es cargado en un script de python. Esto significa que no es necesario ejecutar dicha interfaz y luego el proyecto para poner a funcionar un sistema, solo basta con ejecutar dicho script.

Esta característica es la que postula a dicha herramienta como candidata estrella a ser la utilizada para completar el proyecto que dió origen a esta práctica. Una vez corregido el error de transmisión, el sistema estaría funcionando. Luego, se instala GNURadio en una computadora pequeña del estilo de una Raspberry Pi u Orange Pi y se configura el sistema operativo para que ejecute el script de python cada vez que se inicie. La placa se conecta a una etapa de amplificación que le brinde la potencia de señal necesaria y listo, se estaría emulando un componente altamente costoso con materiales ultra económicos; demostrando la importancia que tienen las ingenierías para nuestro país.

Esto lleva a otro de los grandes objetivos implícitos, que fue determinar la plausibilidad de realizar el proyecto bajo las condiciones que se impusieron.

La respuesta es que no. Y ese no se basa en que si bien la herramienta de software funcionó de manera correcta, la herramienta de hardware no brindó motivos suficientes como para que se la considere apta para la función asignada.

En primera instancia, el dispositivo que se adquiere no está equipado con una carcasa protectora. Dado que la función principal es realizar diseños de prototipos, es el usuario quien

debe improvisar una carcasa o arriesgarse a que los complejos circuitos reciban una descarga de electricidad estática y se vean afectados para siempre. Y considerando que el precio no es económico en lo absoluto, dicha falencia es muy grave.

Por otro lado, es reprochable que ofrezca ganancias en el orden de los 60 dB (tanto para la transmisión como para la recepción) y no posea ningún conector para alimentarla con una fuente externa. En caso de realizar ensayos en una pc no provoca mayores molestias, pero si fuese montada en una mini computadora (por ej, una raspberry pi) podría provocar fallas en su funcionamiento debido a los grandes consumos de corriente que demanda.

Eso deriva en otra falencia grave; las temperaturas que alcanza. Si se eligen ganancias altas en el software, basta con acercar la mano al dispositivo para notar el gran calor que desprende. Es por ello que deben realizarse pruebas en tiempos limitados, para evitar daños por sobrecalentamientos.

Es vital destacar que amén de todos los puntos negativos comentados de la placa LimeSDR-Mini, no se puede constatar que los errores de transmisión que surgieron durante la actividad N° 4 hayan correspondido a las falencias del aparato, dado que los ejemplos simples funcionaron como se esperó.

Como soluciones al error nombrado con anterioridad, se sugiere realizar pruebas a campo abierto, en lugares alejados de las grandes urbes; de modo tal que se eliminen la mayor cantidad de interferencias posibles. En segundo lugar, es necesario realizar una investigación a fondo del funcionamiento del complemento que comunica GNURadio con el anterior dispositivo, de modo que si fuese un error de software pudiese parcharse. En caso de

que la falla fuera de hardware, se sugiere utilizar LimeSDR-Mini para tareas de recepción; pero para realizar la transmisión se utilice HackRF one, un dispositivo altamente recomendado por los usuarios de la comunidad de GNURadio, o en su defecto un USRP.

Por último, el objetivo explícito de esta práctica no se completó. Esto sirve como enseñanza en varios niveles: en cómo administrar los tiempos cuando se trabaja con plazos pre pactados, en la importancia de solicitar ayuda cuando uno se encuentra completamente estancado en lo que está haciendo (ámbito en el cuál se debe mejorar, y en gran medida), en que los proyectos de investigación priorizan el conocimiento adquirido al realizar un proyecto por sobre los resultados obtenidos, y que en la vida las cosas no siempre salen como se las planea. Este último punto refiere a la importancia de aceptar que hay variables que uno no puede manejar, y poder ver el panorama de las situaciones; en vez de enfocarse en un punto concreto.

Pero, si bien es cierto que no se logró realizar con éxito la transmisión, ha de reconocerse que se logró un avance muy importante en la tarea. Los próximos alumnos que se dediquen a este proyecto tendrán toda la etapa de generación de señal por software completa, con lo que solo bastará con solucionar el problema de transmisión de datos y de adquisición de datos por medio de red para que la transcodificación de los datos de sensado se encuentre finalizada.

Conclusiones personales

En lo personal, la experiencia deja un saldo más que positivo. Se comienza la carrera con esperanzas y temores, se adquiere experiencia y conocimientos tanto de aciertos como de

errores. Se pasa años acumulando muchos conocimientos teóricos y en menor medida prácticos, razón por la que es alimentado el fantasma del fracaso a nivel laboral. A veces, no es posible resolver cuestiones prácticas simples que surgen en la cotidianeidad, generando cuestionamientos propios y de personas que pertenecen al círculo íntimo. Todas las personas que eligen esta carrera se topan alguna vez con la típica frase: "Pero, vos no estudiás ingeniería?".

Todos estos pensamientos han aparecido tanto en el inicio como en el transcurso de la experiencia. El desafío de adentrarse en un ámbito en el que poco se había hecho (la transmisión de datos ADS-B con GNURadio) fue la razón más atractiva para entrar en el proyecto en un principio, y a su vez la más desalentadora cada vez que se produjeron estancamientos en el desarrollo.

En esto jugaron un papel muy importante las personas que brindaron su compañía. Tanto el grupo de investigación, que abrió sus puertas y aceptó como uno más desde el principio, como aquellos que pertenecen al círculo íntimo; siempre brindaron el apoyo necesario como para confiar en que se podía continuar.

Las charlas con el tutor, Damián Primo, brindaron una perspectiva más amplia de la situación. Se logró comprender que era probable que no se pudiese completar con éxito la transmisión. Pero eso no era lo importante. Lo importante era poder justificar ese hecho, y brindar una explicación con un criterio que se asemeje al de un profesional. Era adquirir tantos conocimientos como fuera posible, y avanzar tanto en la tarea como las distintas situaciones que surgieran lo fuesen permitiendo.

Por ello es que se considera que a nivel personal el objetivo está cumplido. Se encontró una situación problemática, y se aplicaron todos los conocimientos adquiridos a lo largo de estos años para brindar una solución; utilizando las herramientas que fueron provistas. Y si bien no se proporcionó una solución completa, se pudo realizar un gran avance en la materia, facilitando la tarea de quienes vengan en un futuro. Situaciones que se presentarán con frecuencia durante la carrera profesional.

Bibliografía

[1] https://www.argentina.gob.ar/defensa/piddef/piddef-convocatoria-2016

[2] https://buy.garmin.com/en-US/US/p/140949

[3] https://es.wikipedia.org/wiki/Sistema_de_Vigilancia_Dependiente_Autom%C3%A1tica

[4] https://es.wikipedia.org/wiki/Radio_definida_por_software

[5] https://es.wikipedia.org/wiki/GNU_Radio

[6] https://es.wikipedia.org/wiki/GitLab

[7] https://wiki.myriadrf.org/LimeSDR-Mini_v1.1_hardware_description

[8] http://releases.ubuntu.com/18.04.1.0/

[9] https://github.com/gnuradio/gnuradio

[10] https://www.gnuradio.org/blog/pybombs-the-what-the-how-and-the-why

[11] http://personales.unican.es/perezvr/pdf/Introduccion%20al%20Radar.pdf

[12] https://es.wikipedia.org/wiki/Radar_secundario#Modo_S

[13] https://mafiadoc.com/download/1090-wp30-18-draftdo-260b-v42_59f96fcd1723dd7676261005.html

[14] https://re-ws.pl/2017/09/importusing-gnu-radio-companion-simple-fm-radio-tutorial/

[15] https://www.enacom.gob.ar/fm_p565

[16] https://github.com/lscardoso/gr-ppm-rc

[17] https://wiki.gnuradio.org/index.php/Guided_Tutorial_GNU_Radio_in_Python

[18] https://lists.gnu.org/archive/html/discuss-gnuradio/2011-12/msg00243.html

[19]https://wiki.gnuradio.org/index.php/Guided_Tutorial_GNU_Radio_in_Python#3.2.1._Using_gr_modtool

[20]https://es.aliexpress.com/item/Best-RTL-SDR-radio-receiver-with-free-SDR-radio-software-with-Chip-RTL2832-SDR-and-R820T/32720852511.html