

UNIVERSIDAD NACIONAL DE CÓRDOBA

FACULTAD DE CIENCIAS EXACTAS, FÍSICAS Y NATURALES

CARRERA INGENIERÍA ELECTRÓNICA

PROYECTO INTEGRADOR PARA LA OBTENCIÓN DEL
TÍTULO DE GRADO INGENIERO ELECTRÓNICO

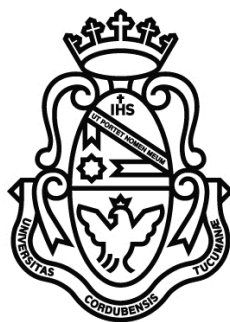
**“DISEÑO E IMPLEMENTACIÓN DE FRONT END
ANALÓGICO PARA SDR”**

Alumno: Banchio, Agustín Enrique

Director: Ing. Rodrigo Bruni

Co-Director: Ing. José Amado

Córdoba, República Argentina – 2017



UNIVERSIDAD NACIONAL DE CÓRDOBA

FACULTAD DE CIENCIAS EXACTAS, FÍSICAS Y NATURALES

CARRERA INGENIERÍA ELECTRÓNICA

**PROYECTO INTEGRADOR PARA LA OBTENCIÓN DEL
TÍTULO DE GRADO INGENIERO ELECTRÓNICO**

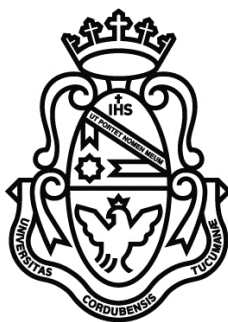
**“DISEÑO E IMPLEMENTACIÓN DE FRONT END
ANALÓGICO PARA SDR”**

Alumno: Banchio, Agustín Enrique

Director: Ing. Rodrigo Bruni

Co-Director: Ing. José Amado

Córdoba, República Argentina – 2017



UNIVERSIDAD NACIONAL DE CÓRDOBA

Facultad de Ciencias Exactas, Físicas y Naturales

Escuela de Ingeniería Electrónica

El Tribunal Evaluador reunido en éste acto y luego de haber aprobado la Solicitud de Aprobación de Tema y efectuado las distintas instancias de correcciones del Informe del Proyecto Integrador para la obtención del Título de Grado “Ingeniero Electrónico” y cumpliendo con el Reglamento correspondiente, declaran el Informe Final de/l los estudiante/s: **Agustín Enrique Banchio** como “aceptado sin correcciones” y la defensa oral Aprobada. Por lo tanto, luego de haber tenido en cuenta los aspectos de evaluación que indica el Reglamento, el Proyecto Integrador se considera Aprobado.

Se firma el Acta de Examen correspondiente y se distribuyen los ejemplares impresos.

Firma y aclaración del Tribunal Evaluador

Fecha:

NOTA:

Agradecimientos

A mi familia por el apoyo que me brindaron durante cursado completo de la carrera.

A mis compañeros y amigos por todos los momentos que compartimos.

A mis profesores por enseñarme y formarme en esta profesión.

Al director y co-director de este proyecto, por su tiempo y dedicación.

Resumen

Los receptores de radio convencionales, como los utilizados generalmente en televisores y receptores de radiofonía, permiten recuperar la información contenida en la señal recibida mediante un circuito específico acorde a la modulación de la señal que se desea demodular.

Una alternativa a los circuitos de recepción tradicionales anteriormente descriptos consiste en el acondicionamiento y digitalización de la señal recibida. La recuperación de la información se realiza por medio de algoritmos de procesamiento de señales. Este sistema es conocido como radio definida por software (SDR) y provee más versatilidad comparado al hardware.

Este trabajo consiste en el diseño e implementación de un front end analógico para SDR (bloque de cadena de recepción encargado de filtrar, amplificar y trasladar en frecuencia la señal).

Adicionalmente se incorpora al diseño la etapa de conversión analógica digital, y como dispositivo de control un microcontrolador encargado tanto del manejo del equipo como de la comunicación con una PC, donde la señal recibida es procesada digitalmente.

El procesamiento es realizado mediante un bloque de programa basado en el software libre GNU Radio. La aplicación además es la encargada de enviar la información para la selección del ancho de banda y frecuencias de trabajo al dispositivo.

Área Temática y Asignaturas

- Área temática: Comunicaciones
- Asignaturas: Electrónica Analógica III, Teoría de las comunicaciones, Electrónica Digital III

Palabras Claves

Front End, Radio Definida por Software, Receptor, Superheterodino, Microcontrolador

Abstract

Conventional radio receivers, such as those commonly used in televisions and audio broadcasting, allow information contained in the received signal to be recovered using a circuit specifically designed according to the modulation of the signal to be demodulated. An alternative to the previously described reception circuits consists of the conditioning and digitalization of the received signal. The recovery of information is done with signal processing algorithms. This system is known as software defined radio (SDR) and provides

If instead of utilizing a specific circuit, the received signal is digitalized and sent to a PC capable of processing it, this information can be recovered with signal processing algorithms. This system is known as software defined radio and provides more versatility compared to hardware solutions.

This project consists of the design and implementation of an analog front end for SDR (block of the reception chain responsible for filtering, amplifying and transferring the signal).

Additionally, the analog to digital conversion stage is incorporated into the design, and as a control device, a microcontroller is added for both the handling of the components configuration and the communication with the PC, where the received signal is digitally processed.

The processing is done through a program blocked based on the free software GNU Radio. The application is also responsible for sending the information for the bandwidth and tuning settings of the device.

Key Words

Front End, Software Defined Radio, Receiver, Superheterodyne, Microcontroller

Índice

Resumen	VII
Área Temática y Asignaturas	VII
Palabras Claves	VII
Abstract	IX
Key Words	IX
Índice	XI
Lista de Tablas	XVII
Lista de Figuras	XIX
Lista de Símbolos y Convenciones	XXIII
Símbolos	XXIII
Abreviaciones	XXIII
Introducción	1
1.1 Objetivos	1
1.1.1 Objetivo General	1
1.1.2 Objetivos específicos	1
1.2 Antecedentes	1
1.3 Motivación	2
1.4 Metodología para lograr los objetivos propuestos	2
1.5 Plan del proyecto	2
Radiocomunicaciones	5
2.1 Radiocomunicación	5
2.2 Espectro Radioeléctrico	5

2.3	Modulación	7
2.4	Receptor de Radio.....	9
	Componentes de un receptor de radio	11
2.4.1	Antena	11
2.4.2	Amplificador de radiofrecuencia.....	11
2.4.2.1	Parámetros de un amplificador de radiofrecuencia	11
2.4.3	Mezclador.....	12
2.4.3.1	Frecuencia Imagen.....	13
2.4.3.2	Parámetros de un mezclador.....	14
2.4.4	Oscilador Local	15
2.4.4.1	Tipos de osciladores locales	15
2.4.5	Filtros	16
2.4.5.1	Tipos de filtros según su tecnología	16
2.4.5.2	Tipos de filtros según su respuesta en frecuencia.....	17
2.4.5.3	Rechazo de frecuencia imagen	17
2.4.6	Demodulador.....	18
2.4.6.1	Ejemplos de técnicas de demodulación	18
2.5	Radio definida por software.....	18
2.5.1	Convertor analógico digital	19
2.5.1.1	Arquitecturas de convertidores analógicos digitales.....	19
2.5.1.2	Parámetros de un convertor analógico digital.....	20
Microcontroladores y comunicación con PC		21
3.1	Microcontrolador	21
3.1.1	Periféricos y Entradas/Salidas.....	21
3.1.2	Interrupciones.....	22
3.2	Comunicación USB	22

3.2.1 Protocolo de comunicación	23
3.3 Ringbuffers	23
Desarrollo del proyecto	25
4.1 Descripción general del proyecto.....	25
4.1.1 Diagrama de Bloques General	26
4.2 Acondicionamiento analógico	27
4.2.1 Arquitectura del receptor.....	27
4.2.2 Osciladores	28
4.2.3 Mezcladores	29
4.2.4 Amplificadores	31
4.2.5 Filtros	35
4.2.5.1 Filtro de la etapa de RF	36
4.2.5.2 Filtro de la etapa de IF	40
4.2.5.3 Filtro de la etapa de banda base.....	41
4.2.5.4 Filtro de salida de osciladores	42
4.2.6 Alimentación	44
4.3 Etapa de microcontrolador y digitalización	45
4.3.1 Microcontrolador.....	45
4.3.2 Conversor Analógico-Digital (ADC):	48
4.3.3 Programación de microcontrolador:	48
4.3.3.1 Diagrama en bloques del código del microcontrolador.....	49
4.3.3.2 Muestreo	49
4.3.3.3 Configuración de osciladores	50
4.3.3.4 Recibiendo la configuración de la PC	52
4.4 Diseño de driver en PC	53
4.4.1 Utilizando el puerto serie (virtual)	53

4.4.2 Bloque de GNU Radio	53
4.5 Circuito Final	57
Pruebas Realizadas	61
5.1 Pruebas Iniciales	61
5.2 Pruebas del circuito final	63
5.2.1 Prueba de modulación analógica.....	64
5.2.2 Prueba de modulación digital.....	67
5.2.3 Resultados	70
5.2.4 Prueba de sensibilidad.....	70
Instrucciones de uso	73
6.1 Consideraciones Preliminares.....	73
6.1.1 Reglas UDEV	73
6.2 Configuración de la placa	74
6.3 Utilización con GNU Radio.....	74
6.3.1 Instalación de paquetes necesarios.....	75
6.3.2 Creación de módulo de GNU Radio	77
6.3.3 Creación del bloque de GNU Radio.....	77
6.3.4 Compilación e Instalación del bloque	78
6.4 Utilización fuera de GNU Radio	80
Conclusiones	81
7.1 Resultados obtenidos en base a los objetivos propuestos	81
7.2 Conclusión General.....	82
7.3 Mejoras a futuro.....	83
Bibliografía y Referencias	85
Anexo 1: Circuito final implementado	87
Anexo 2: Código de microcontrolador	89

Anexo 3: Código de bloque de GNU Radio	95
Solicitud de Aprobación de Tema	99
Nota de aprobación	106

Lista de Tablas

Tabla 1 – Bandas de espectro radioeléctrico según UIT	7
Tabla 2 – Especificaciones de Teensy 3.2.....	47

Lista de Figuras

Figura 1 – Utilización comercial de bandas del espectro radioeléctrico	6
Figura 2 – Representación gráfica de las ondas moduladas en AM y FM, junto a las señales portadora y modulante.	8
Figura 3 – 4 Símbolos de modulación ASK	9
Figura 4 – Diagrama de bloques de receptor superheterodino	9
Figura 5 – Diagramas de bloques de distintos receptores de radios definidas por software	10
Figura 6 – Representación simbólica de un mezclador	12
Figura 7 – Representación de conversión descendente y conversión ascendente	13
Figura 8 – Representación en frecuencia de frecuencia imagen respecto a frecuencia del oscilador y de la señal deseada	14
Figura 9 – Diagrama de bloques de un oscilador PLL.	15
Figura 10 – Diagrama de bloques de oscilador de síntesis digital directa.....	16
Figura 11 – Ringbuffer con los punteros de lectura y escritura.....	24
Figura 12- Diagrama de bloques del proyecto.....	26
Figura 13 – Receptor superheterodino de doble conversión.	27
Figura 14 - Placa de desarrollo del sintetizador AD9850.....	28
Figura 15 – Diagrama de bloques del NE602AN.....	29
Figura 16 – NE602AN empaquetado DIP	30
Figura 17 – Circuito Equivalente del NE602AN.....	30
Figura 18 – Amplificadores MAR-1 (izquierda) y MAR-3 (Derecha)	32
Figura 19 – Circuito recomendado de aplicación de amplificadores MAR según fabricante.	32
Figura 20 – MCP6294 encapsulado DIP	33
Figura 21 – Amplificación de la señal sin sufrir distorsión por corte o saturación.....	34
Figura 22 – Circuito que filtra la señal continua, le suma un cuarto de la tensión de alimentación y amplifica por dos.....	34
Figura 23 – Señal filtrada y desplazada al centro del rango del ADC.....	35

Figura 24 – Circuito que filtra la señal y le suma la mitad de la tensión de referencia, incluyendo un buffer previo al ADC	35
Figura 25 – Diagrama esquemático del filtro de la etapa de RF	36
Figura 26 – Diagrama de Bode del filtro de la etapa de RF	38
Figura 27 – Respuesta del filtro según simulación.....	39
Figura 28 – Respuesta de mediciones del filtro junto a un amplificador MAR-3+.....	39
Figura 29 – Filtro cerámico SFELF10M7HAA0-B0	40
Figura 30 – Respuesta del filtro cerámico	40
Figura 31 – Diagrama esquemático del filtro de la etapa de banda base.....	41
Figura 32 – Diagrama de Bode del filtro de banda base	42
Figura 33 – Diagrama esquemático del filtro sobre la salida de los osciladores.....	42
Figura 34 – Diagrama de Bode del filtro sobre la salida de los osciladores.....	44
Figura 35 – Conversor DC-DC.....	44
Figura 36 –Asignacion de pines de Teensy 3.2.....	46
Figura 37 – Diagrama en bloques del código del microcontrolador	49
Figura 38 – Diagrama de flujo de las funciones que programan los osciladores	51
Figura 39 - Formato de instrucciones que recibe el microcontrolador.....	52
Figura 40 – Diagrama de flujo del bloque de GNU Radio encargado de recibir muestras de la placa.....	55
Figura 41 – Módulo y bloque en la lista de GNU Radio Companion	56
Figura 42 – Bloque Pifsource dentro de GNU Radio Companion	56
Figura 43 – Configuración del bloque dentro de GNU Radio Companion.	57
Figura 44 – Diseño de PCB	58
Figura 45 – Tóner transferido a la placa.....	58
Figura 46 – Placa de cobre sumergida en percloruro férrico a baño maría.	59
Figura 47 – Placa de circuito final con nombre de componentes	60
Figura 48 – Placa de circuito final vista de abajo.....	60
Figura 49 – Circuito para prueba de filtro de etapa de RF	62
Figura 50 – Prueba del circuito en protoboard	63
Figura 51 – Amplificadores desconectados.....	64
Figura 52 – Diagrama de flujo en GNU Radio Companion para la transmisión en FM	64
Figura 53 – Diagrama de flujo en GNU Radio Companion para la recepción de FM utilizando el bloque Pifsource.....	65

Figura 54 – Gráfico en el dominio del tiempo de un fragmento de la señal FM transmitida	65
Figura 55 – Gráfico en el dominio del tiempo de un fragmento de la señal FM recibida (No es del mismo instante de la figura anterior)	66
Figura 56 – Formas de onda de música transmitida antes de ser modulada y enviada (arriba) y luego de ser recibida y demodulada (abajo)	66
Figura 57 - Formas de onda de audio con voz transmitido antes de ser modulado y enviado (arriba) y luego de ser recibido y demodulado (abajo).....	67
Figura 58 – Diagrama de flujo en GNU Radio Companion para la transmisión de FSK	68
Figura 59 – Diagrama de flujo en GNU Radio Companion para la recepción de FSK utilizando el bloque Pifsource.....	68
Figura 60 – Gráfico en el dominio del tiempo de un fragmento de la señal FSK transmitida	69
Figura 61 – Gráfico en el dominio del tiempo de un fragmento de la señal FSK recibida luego de un filtro pasa bajo (No es del mismo instante de la figura anterior).....	69
Figura 62 – Bits enviados [11111111,00000000,00101010,00000000]	69
Figura 63 – Bits recibidos [11111111,00000000,00101010,00000000].....	69
Figura 64 – Configuración para prueba de sensibilidad	71
Figura 65 – Recepción del barrido con -108dBm de entrada	71
Figura 66 – Instalación de paquetes necesarios	76
Figura 67 – Creación de módulo PIFE	77
Figura 68 – Creación de bloque PIFSource	78
Figura 69 – Compilación e instalación del módulo	79
Figura 70 – Módulo y bloque instalados dentro de GNU Radio Companion	79
Figura 71 – Bloque PIFSource en GNU Radio Companion.....	79

Lista de Símbolos y Convenciones

Símbolos

Ω	Ohm – Unidad de resistencia
A	Ampere – Unidad de Medida de corriente
V	Volts – Unidad de medida de tensión o voltaje
Hz	Hertz – Unidad de frecuencia en ciclos por segundo
sps, mps	Samples per second / Muestras por segundo, unidad de medida de la frecuencia de muestreo
m,u,n,p	Prefijos 10^{-3} , 10^{-6} , 10^{-9} , 10^{-12} respectivamente
k, M, G	Prefijos 10^3 , 10^6 , 10^9 respectivamente
Q	Factor de calidad o selectividad, mide relación entre energía reactiva que almacena y energía que disipa en un ciclo de señal, indica que tan aguda es la resonancia.

Abreviaciones

RF	Radiofrecuencia/s
FI, IF	Frecuencia intermedia
ADC	Conversor analógico digital
SDR, RDS	Radio definida por software
USB	Universal Serial Bus
UART	Universal Asynchronous Receiver-Transmitter
SPI	Serial Peripheral Interface Bus

I2C	Inter-Integrated Circuit
ISR	Interrupt Service Routine
CDC	Communications Device Class, clase de dispositivo USB que define su comportamiento.
DC	Tensión Continua
FIFO	First In, First Out, método para organizar elementos en una cola. El elemento más antiguo (primero que entro) es el primero en ser procesado.
IIP3	Punto de intercepción de 3er orden
COM	Puerto de comunicación
GBP	Gain-Bandwidth Product, producto del ancho de banda y la ganancia de un amplificador.
DIP	Dual in-line package, es una forma de empaquetado de circuito integrados que consiste en dos líneas paralelas de pines para insertar en un zócalo o de agujero pasante en un PCB.

CAPÍTULO 1

Introducción

1.1 Objetivos

1.1.1 Objetivo General

El objetivo general del presente proyecto es el diseño y construcción de un prototipo receptor de señales de radiofrecuencia para la banda de HF. El equipo a desarrollar debe poder sintonizar canales dentro de dicha banda, digitalizarlos y transferirlos a una PC para su posterior procesamiento, siguiendo una metodología propia de un desarrollo profesional.

1.1.2 Objetivos específicos

Como objetivos específicos del proyecto integrador se encuentran los siguientes:

- Integrar conocimientos adquiridos durante la carrera
- Obtener experiencia con el manejo de componentes de radiofrecuencia
- Familiarizarse con métodos de programación en el sistema operativo Linux

1.2 Antecedentes

Dentro del Laboratorio de RF Y Microondas (LARFYM) donde se realiza el proyecto no existen antecedentes de proyectos integradores o tesis de grado similares.

Dentro de la Facultad se encuentran algunos trabajos anteriores que realizan módulos específicos de radios definidas por software, generalmente focalizados en el procesamiento de las señales. En este trabajo, a diferencia de los anteriores, se focaliza en la plataforma para el acondicionamiento de la señal con el objetivo de ser digitalizada y el proceso de digitalización, sin realizar ningún desarrollo propio sobre el procesamiento de la información de la señal.

1.3 Motivación

Como motivación personal del proyecto se incluye la persecución del título de grado, y específicamente para este proyecto es valorado por el autor la amplia cobertura de especialidades incluidas, ya que el trabajo incluye desarrollo de electrónica analógica de alta frecuencia, electrónica digital e informática.

Como motivación académica, dentro del LARFYM de la Facultad se deseaba trabajar con radios definidas por software (SDR) debido a su importante flexibilidad y su notable crecimiento como tecnología en comunicaciones a través de radiofrecuencias.

1.4 Metodología para lograr los objetivos propuestos

El método utilizado en el proyecto consistió en modularizar el proyecto en etapas permitiendo trabajar en ellas de forma independiente. Esto dio la posibilidad de trabajar simultáneamente en estas etapas y en caso de demoras u obstáculos en algunas, poder continuar con las demás. También se pudieron realizar pruebas y prototipos en cada etapa para asegurarse su correcta respuesta previo a la conexión entre ellas para su funcionamiento global.

1.5 Plan del proyecto

El proyecto consiste en el desarrollo de un receptor de radiofrecuencia que permita obtener la señal en la PC en tiempo real, siendo este de fácil configuración y que incluya un módulo para hacer de interfaz en el programa de software libre GNU Radio.

En las primeras etapas se realizó una investigación teórica sobre la arquitectura de los receptores de radio y las distintas tecnologías de los componentes. Luego se llevó a cabo un diseño de receptor que será sometido a prueba y corregido experimentalmente.

Se empleó un microcontrolador para que actúe como intermediario entre la PC y el receptor, se encargará de enviar las muestras a la PC y de recibir configuración de esta y aplicarla a los componentes del receptor.

Luego se realizó un módulo dentro del programa GNU Radio en PC para poder utilizar las muestras de la señal dentro de dicho programa.

En cada etapa se efectuaron pruebas para validar el diseño que se fue desarrollando y permitir encontrar soluciones y mejoras de manera temprana. En la última etapa se ejecutaron pruebas del sistema completo para obtener conclusiones en base a los objetivos propuestos.

CAPÍTULO 2

Radiocomunicaciones

El Capítulo 2 presenta un marco teórico general sobre el uso del espectro radioeléctrico, la forma de transportar información en ondas electromagnéticas y como se recibe en los receptores de radio.

2.1 Radiocomunicación

Se define como radiocomunicación a toda transmisión, emisión o recepción de signos, señales, o información de cualquier naturaleza a través de ondas radioeléctricas.

Las ondas radioeléctricas son ondas electromagnéticas que se propagan por el espacio a la velocidad de la luz sin guía artificial fijadas por debajo de los 3000GHz

2.2 Espectro Radioeléctrico

El Espectro Radioeléctrico es un recurso natural, de carácter limitado, que constituye un bien de dominio público, sobre el cual el Estado ejerce su soberanía.¹ Es un medio intangible para servicios de comunicaciones. Se descompone en bandas de frecuencias que se atribuyen a distintos servicios y se asigna a usuarios autorizados para operarlas.

¹ ENACOM: <https://www.enacom.gob.ar/-que-es-el-espectro-radioelectrico- p117> [Última visita: Noviembre 2017]

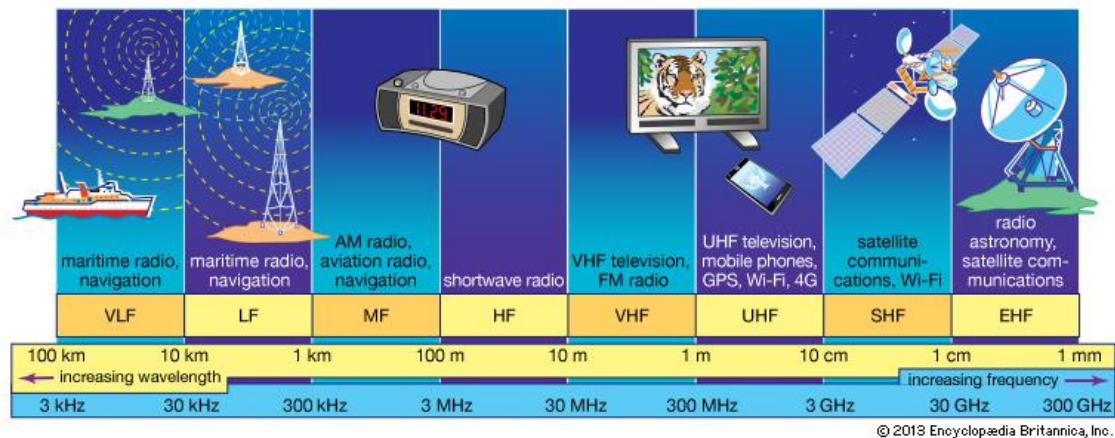


Figura 1 – Utilización comercial de bandas del espectro radioeléctrico²

La autoridad encargada de la atribución del espectro radioeléctrico en Argentina es el ENACOM (Ente Nacional de Comunicaciones), órgano que depende del Ministerio de Modernización. Para la toma de decisiones se tienen en cuenta las recomendaciones de la Unión Internacional de Telecomunicaciones (UIT o ITU en inglés).

Las resoluciones del ENACOM se reflejan en el Cuadro de Asignación de Bandas de la República Argentina.

La UIT subdivide el espectro radioeléctrico en nueve bandas como indica la siguiente tabla:

² Imagen obtenida de: <https://www.britannica.com/science/radio-frequency-spectrum> [Última Visita: Noviembre 2017]

Tabla 1 – Bandas de espectro radioeléctrico según UIT

Número de Banda Según UIT	Símbolos	Banda de Frecuencias	Longitud de Onda	Tipo de Ondas
1	ELF	3 a 30 Hz	100.000 a 10.000 Km	Miriamétricas
2	SLF	30 a 300 Hz	10.000 a 1.000 Km	
3	ULF	300 Hz a 3 KHz	1.000 a 100 Km	
4	VLF	3 a 30 KHz	100 a 10 km	
5	LF	30 a 300 KHz	10 km a 1 Km	Kilométricas
6	MF	300 a 3000 KHz	1.000 a 100 m	Hectométricas
7	HF	3 a 30 MHz	100 a 10 m	Decamétricas
8	VHF	30 a 300 MHz	10 a 1 m	Métricas
9	UHF	300 a 3000 MHz	100 a 10 cm	Decimétricas
10	SHF	3 a 30 GHz	10 a 1 cm	Centimétricas
11	EHF	30 a 300 GHz	10 a 1 mm	Milimétricas
12	THF	300 a 3000 GHz	1 a 0,1 mm	Decimilimétricas

2.3 Modulación

En telecomunicaciones se denomina modulación al proceso de incluir la información a transmitir, sea digital o analógica, dentro de otra señal que puede ser físicamente transmitida.

La señal modulada, que se transmite, es el producto que resulta de modificar la señal portadora con la información de la señal modulante o moduladora.

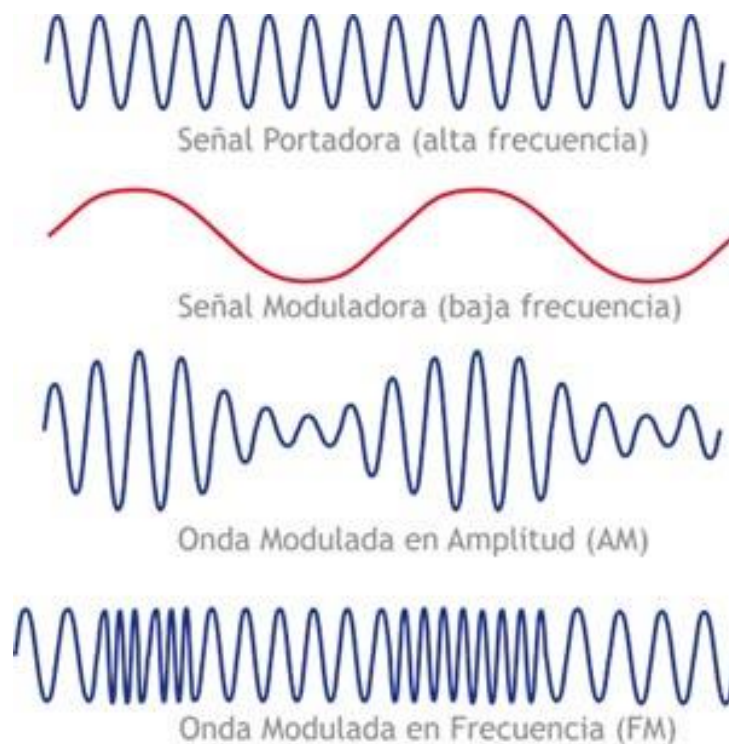


Figura 2 – Representación gráfica de las ondas moduladas en AM y FM, junto a las señales portadora y modulante.³

Se denomina modulador a aquellos dispositivos que permiten incorporar información sobre una señal portadora, siendo los demoduladores los que realizan el proceso inverso. Un modem es un dispositivo que puede realizar ambas operaciones.

Con la modulación se logra transportar la información que se desea transmitir a la frecuencia de la señal portadora. Esto permite ubicar la señal en la banda deseada del espectro radioeléctrico, por ejemplo un canal de radio o un canal de televisión.

En modulación analógica, la modulación se aplica continuamente sobre la señal portadora en respuesta a la modulante. Las modulaciones analógicas más utilizadas son por amplitud (AM) o por frecuencia (FM).

En modulación digital se utiliza un número finito de variaciones de fase, frecuencia o amplitud. Cada una de estas variantes representa un símbolo que se asigna a un patrón único de bits binarios. Si se utilizan $M = 2^N$ símbolos, cada símbolo representa un patrón de N bits. Las modulaciones digitales más utilizadas son por niveles de amplitud (ASK), de frecuencia (FSK), de fase (PSK) o de amplitud y fase (QAM).

³ Imagen obtenida de: <http://www.analfatecnicos.net/pregunta.php?id=15> [Última visita: Octubre 2017]

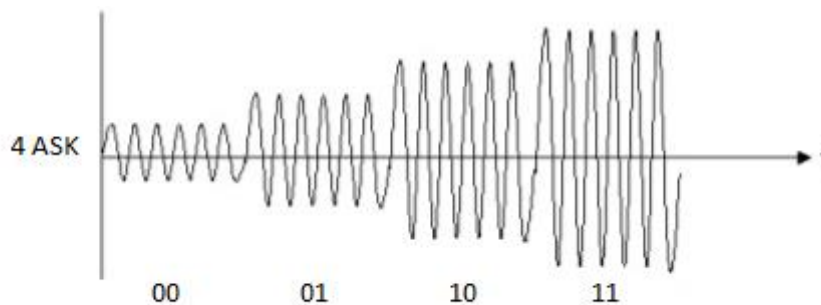


Figura 3 – 4 Símbolos de modulación ASK

2.4 Receptor de Radio

Un receptor de radio es un dispositivo o circuito electrónico capaz de extraer información utilizable de las ondas electromagnéticas que recibe.

El diseño de receptor más utilizado es el del receptor superheterodino.

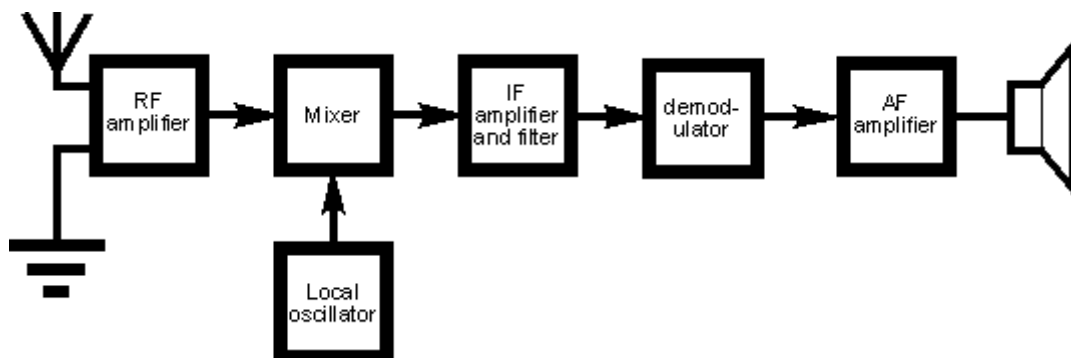


Figura 4 – Diagrama de bloques de receptor superheterodino⁴

En este tipo de receptores, la señal de radiofrecuencia es desplazada a una frecuencia intermedia (IF) a través de un mezclador antes de ser procesada por filtros específicamente seleccionados para esta IF con alta selectividad y finalmente demodulada con algún componente demodulador, si la información es audio, la última etapa consiste en un amplificador de audio y una salida por parlante.

⁴ Imagen obtenida de: <http://www.radio-electronics.com/info/rf-technology-design/superheterodyne-radio-receiver/block-diagram.php> [Última visita: Octubre 2017]

Algunos receptores, llamados de doble conversión, utilizan dos frecuencias intermedias para mayor rechazo de frecuencia imagen y para obtener mayor selectividad. Al utilizar dos frecuencias intermedias requieren de dos mezcladores.

En caso de ser un receptor de radio definida por software, luego del mezclador se coloca un convertor analógico digital que muestrea la señal para luego poder ser procesada por algoritmos de software.

Otra arquitectura de receptores utilizados en radios definidas por software no utiliza mezclador y en cambio ubica el convertor analógico digital luego de una sola etapa de amplificación y filtrado. Al muestrear las señales en su frecuencia de origen sin pasar por un mezclador se necesitan más muestras por segundo, esto a su vez permite tener un ancho de banda más grande. Esta arquitectura precisa de convertidores analógicos digitales de velocidades mucho más altas (incluso llegando a varias Gbps) y se aproxima más a una radio definida por software ideal, que utilizaría un convertor analógico digital conectado directamente en la antena.

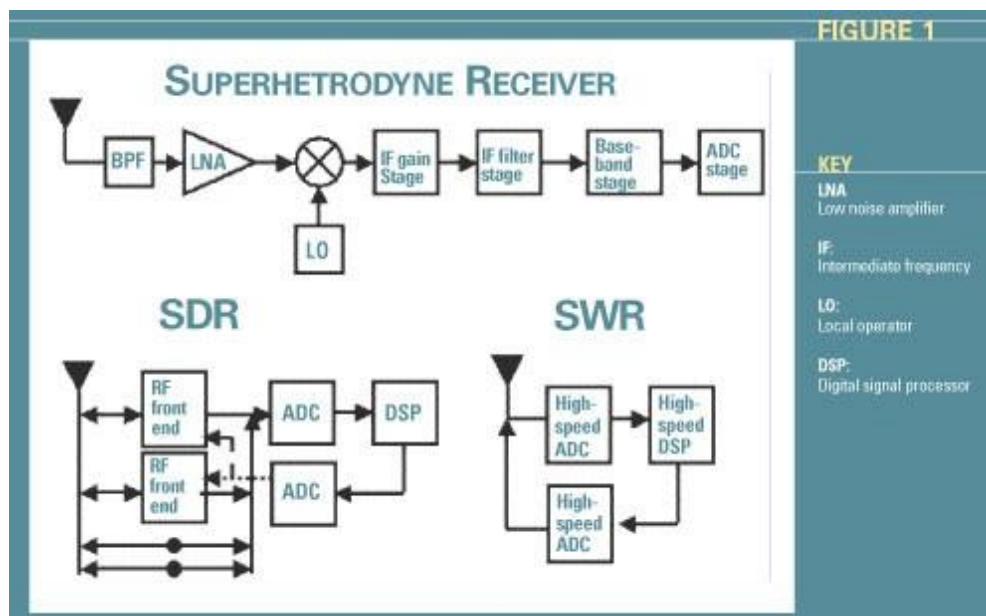


Figura 5 – Diagramas de bloques de distintos receptores de radios definidas por software⁵

⁵ Imagen obtenida de: <http://urgentcomm.com/mag/uwb-brings-radios-future-closer> [Última visita: Noviembre 2017]

Componentes de un receptor de radio

2.4.1 Antena

La antena es el componente que convierte potencia eléctrica en ondas de radio en caso de transmisión, y viceversa para el caso de recepción. Típicamente están construidas con materiales conductores. Los campos eléctricos y magnéticos oscilantes de una onda entrante ejercen una fuerza sobre los electrones de estos conductores, produciendo corrientes oscilantes a lo largo de la antena.

Las antenas pueden ser diseñadas para transmitir o recibir ondas en todas las direcciones horizontales (omnidireccionales) o en alguna dirección en particular (direccional). Por lo general la amplitud de la señal recibida por la antena es demasiado baja (del orden de microvoltios de pico a pico) para ser demodulada directamente y requiere ser amplificada para su utilización.

2.4.2 Amplificador de radiofrecuencia

Un amplificador es un dispositivo electrónico capaz de utilizar energía de una fuente para incrementar la amplitud de una señal. Un amplificador de radiofrecuencia tiene que ser capaz de amplificar señales de frecuencias superiores a las de un amplificador de señales de audio o un controlador para un motor. Además se caracteriza por introducir bajo nivel de ruido.

Un amplificador puede ser una válvula, un transistor, o parte de un circuito integrado con más elementos. Es común en amplificadores de radiofrecuencia que su entrada y salida estén adaptados en impedancia para maximizar la transferencia de energía.

2.4.2.1 Parámetros de un amplificador de radiofrecuencia

- **Ganancia:** Es la medida de la capacidad de incrementar la potencia o amplitud de una señal de la entrada a la salida. Por lo que un amplificador tiene que tener una ganancia superior a 1. La ganancia también puede ser expresada en dB.
- **Frecuencia de trabajo:** Es el rango de frecuencias en el cual el amplificador tiene una respuesta con ganancia superior a la unidad.

- **Figura de ruido:** La figura de ruido es una medida de degradación de la relación señal-ruido causada por componentes en la cadena de la señal de radiofrecuencia. Es la relación entre el ruido en la salida del componente y el ruido que quedaría si éste no hubiera introducido ninguno. Se mide en decibeles, y a valores más chicos mejor desempeño.
- **Estabilidad:** En la práctica, existen caminos de señal desde la salida del amplificador (donde la potencia es mayor debido a la ganancia) hacia la entrada. Es posible que para ciertos valores de carga, el coeficiente de reflexión de entrada exceda la unidad, transformando el circuito en un amplificador de reflexión en la entrada. Algunos valores de coeficiente de reflexión de fuente a la entrada pueden provocar que el coeficiente de reflexión de salida supere la unidad. Si alguno de estos casos o ambos suceden, se dice que el circuito es condicionalmente estable, o potencialmente inestable. Un amplificador es incondicionalmente estable si es estable para todas las frecuencias para la cual su ganancia es mayor a la unidad, y para cualquier impedancia de carga y fuente.

2.4.3 Mezclador

El mezclador es un dispositivo de 3 puertos, activo o pasivo, que puede modular o demodular una señal. Se utiliza para cambiar la frecuencia de una señal mientras se preservan el resto de sus características como fase y amplitud.

La señal de radio frecuencia entra por un puerto, y una señal provista por un oscilador local entra por otro, y por el puerto de salida se obtienen la suma y la diferencia en frecuencia de las entradas.

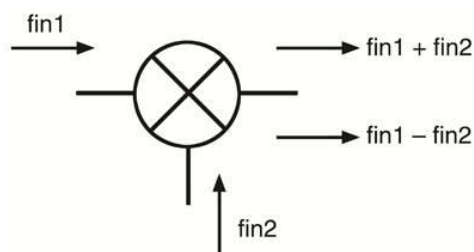


Figura 6 – Representación simbólica de un mezclador⁶

⁶ Imagen obtenida de: <https://www.digikey.com/en/articles/techzone/2011/oct/the-basics-of-mixers> [Última visita: Octubre 2017]

Cuando la frecuencia deseada es menor a la frecuencia de entrada, el proceso se llama conversión descendente. Cuando la frecuencia deseada es mayor, el proceso se llama conversión ascendente.

El mezclador se utiliza en un receptor para desplazar la señal de radiofrecuencia a una de frecuencia intermedia. Los mezcladores activos son configurados para obtener ganancia de conversión, aislación entre puertos y requerir menos potencia del puerto de oscilador local.

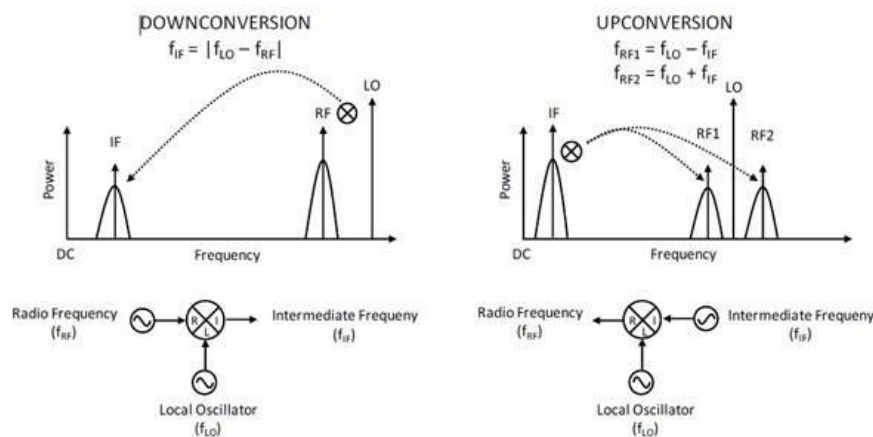


Figura 7 – Representación de conversión descendente y conversión ascendente⁷

2.4.3.1 Frecuencia Imagen

Debido a que en la salida, se encuentran la suma y la diferencia de las entradas, para una frecuencia específica de oscilador local, dos frecuencias de entrada producirán la misma frecuencia intermedia, la frecuencia con la señal deseada y una frecuencia imagen. Para impedir que esto suceda en la entrada del amplificador se debe colocar un filtro para el rango de trabajo del receptor, que filtre todas las frecuencias imagen, así a la salida del mezclador solo está presente la señal deseada.

⁷ Imagen obtenida de: <https://www.digikey.com/en/articles/techzone/2011/oct/the-basics-of-mixers> [Última visita: Octubre 2017]

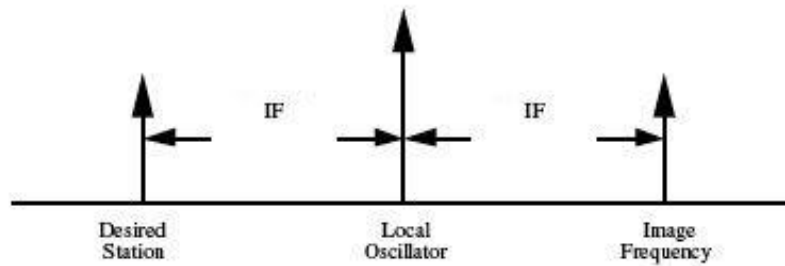


Figura 8 – Representación en frecuencia de frecuencia imagen respecto a frecuencia del oscilador y de la señal deseada⁸

2.4.3.2 Parámetros de un mezclador

- **Pérdida o ganancia de conversión:** Se mide en decibeles y es la ganancia que recibe la señal en un mezclador activo o la pérdida que sufre en un mezclador pasivo.
- **Punto de intercepción de 3er orden (IIP3):** Es el nivel de potencia de la señal de entrada que produce a la salida el mismo nivel de señal deseada que de productos de intermodulación no deseados.
- **Aislación de puertos:** Indica la potencia que se fuga de un puerto a otro. Para no obtener interferencia de la señal de RF y el oscilador local a la salida, se deberá utilizar alta aislación.
- **Figura de ruido:** Al igual que en un amplificador, un mezclador inserta una cantidad de ruido a la salida y determina el valor de figura de ruido.
- **Punto de 1dB de compresión:** Si se aumenta el valor de potencia de RF a la entrada, también aumenta el valor de FI a la salida con la ganancia o pérdida de conversión. Pero si la potencia de RF es demasiado grande el mezclador se saturará (compresión) y la potencia de FI en la salida no aumentará en la misma proporción. Los fabricantes de mezcladores especifican la potencia de entrada de RF que produce 1dB de compresión en la señal.

⁸ Imagen obtenida de: <https://www.quora.com/What-is-the-image-frequency> [Última visita: Octubre 2017]

2.4.4 Oscilador Local

Un oscilador local es un oscilador electrónico que se usa en conjunto con mezcladores. Producen una onda senoidal o cuadrada que es utilizada para cambiar la frecuencia de la señal en un receptor superheterodino. Un requerimiento de los osciladores es que no emitan armónicos ni espurias que puedan interferir en el proceso de conversión.

2.4.4.1 Tipos de osciladores locales

- **Oscilador de cristal:** Utilizando la resonancia de un cristal piezoeléctrico se consigue alta estabilidad por bajo costo, pero a una frecuencia fija. Para cambiar frecuencias se debe cambiar el cristal.
- **Lazo de fase enganchado (PLL):** Utiliza un oscilador controlado por voltaje (VCO) y un sistema de control que relaciona la fase de la señal a la salida con la señal a la entrada para mantener las fases emparejadas. Utilizando divisores de frecuencia dentro del lazo de control se pueden obtener a la salida múltiplos de la frecuencia a la entrada. Y variando estos divisores se obtiene un oscilador de frecuencia variable. Si bien estos osciladores llegan a muy altas frecuencias (GHz), al estar presente un lazo de control estos osciladores tienen limitaciones de estabilidad y velocidad para fijar una frecuencia.

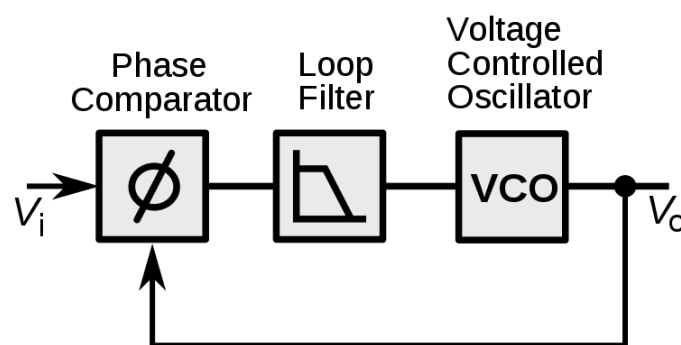


Figura 9 – Diagrama de bloques de un oscilador PLL.⁹

- **Síntesis digital directa (DDS):** Permite crear ondas de forma arbitraria a partir de un clock de frecuencia fija. Contiene almacenados en memoria suficientes puntos

⁹ Imagen obtenida de: https://en.wikipedia.org/wiki/Phase-locked_loop [Última visita: Octubre 2017]

de la función a sintetizar en forma digital. Estos valores digitales son alimentados a un conversor digital analógico a rápida velocidad obteniendo como resultado la onda con la frecuencia deseada con altísima precisión (del orden de miliHertz). Presenta más agilidad para cambiar de frecuencia respecto a un PLL (millones de veces por segundo), no tiene problemas de inestabilidad de lazo de control, pero es costoso para muy altas frecuencias (mayor a 100MHz).

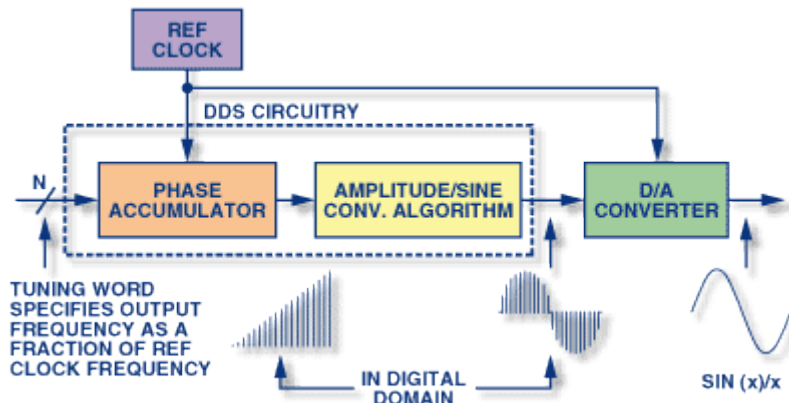


Figura 10 – Diagrama de bloques de oscilador de síntesis digital directa.¹⁰

2.4.5 Filtros

Los filtros electrónicos son circuitos que realizan procesamiento de señales, en especial para disminuir componentes de frecuencia no deseados, amplificar frecuencias deseadas o ambos.

2.4.5.1 Tipos de filtros según su tecnología

- **Pasivos:** Son filtros de combinaciones de Resistencias, Capacitores e Inductores, no dependen de una fuente de alimentación externa.
- **Activos:** Son filtros que además de componentes pasivos incluyen componentes activos, frecuentemente amplificadores operacionales.
- **Filtro de cristal:** Son circuitos que aprovechan la característica piezoeléctrica de los cristales cuarzo. Estos cristales presentan resonancias mecánicas de entre 10.000 a 100.000 veces de factor Q, mucho mayores a los creados con inductores

¹⁰ Imagen obtenida de: <http://www.analog.com/en/analog-dialogue/articles/all-about-direct-digital-synthesis.html> [Última visita: Octubre 2017]

y capacitores. La estabilidad y alto factor Q del cristal permite frecuencias precisas y alta selectividad.

- **Filtro cerámico:** Similar al filtro de cristal pero utilizando resonadores cerámicos, que también tienen características piezoeléctricas. Son más baratos pero presentan menos selectividad.

2.4.5.2 Tipos de filtros según su respuesta en frecuencia

Una frecuencia se considera rechazada cuando su atenuación específica supera los 3dB. La frecuencia a la que se la atenúa exactamente 3dB se la llama frecuencia de corte.

- **Filtro pasa bajo:** Filtro que atenúa las frecuencias superiores a una frecuencia de corte.
- **Filtro pasa alto:** Filtro que atenúa las frecuencias inferiores a una frecuencia de corte.
- **Filtro pasa banda:** Filtro que atenúa frecuencias que se encuentren fuera de un rango o banda.
- **Filtro rechaza banda:** Filtro que atenúa frecuencias dentro de un rango o banda.

También existen filtros de usos muy específicos como filtros peine que presentan varias bandas de paso separadas regularmente, o filtros en los que todas las frecuencias pasan y solo la fase es modificada.

2.4.5.3 Rechazo de frecuencia imagen

Para que no se mezcle la señal deseada con la señal imagen no deseada en el mezclador, se suele seleccionar una FI mayor a la mitad del ancho del rango de trabajo deseado y luego se utiliza un filtro pasa banda que filtre frecuencias fuera de ese rango, de forma que sintonizando cualquier frecuencia del rango de trabajo, la frecuencia imagen quede atenuada por el filtro.

Otra solución es usar distintos filtros y conmutarlos según cual se necesite. De esta forma se puede obtener un rango de trabajo mucho mayor y la FI no tiene que ser mayor a la mitad del ancho.

2.4.6 Demodulador

Un demodulador es un circuito electrónico que se encarga de recuperar la información contenida en la señal. Hay varios métodos de modulación, por lo que hay varios tipos de demoduladores. La salida de estos dispositivos puede ser sonido (si se busca recuperar una señal de audio), imágenes (como en una señal de TV), datos binarios (en una señal digital), etc.

De acuerdo a la técnica de modulación será el modulador que se debe utilizar. Si un receptor de radio debe recuperar información de varias técnicas de modulación deberá tener incluido un circuito específico para cada técnica a demodular.

2.4.6.1 Ejemplos de técnicas de demodulación

La modulación analógica por amplitud puede ser demodulada por un detector de envolvente, que puede ser un simple diodo rectificador. Otro método puede estar basado en el uso de un mezclador, recuperando la banda base.

La modulación analógica por frecuencia puede ser demodulada con un detector de cuadratura que multiplica la señal consigo misma desfasada 90 grados, obteniendo como uno de los términos resultantes la información original que puede ser filtrada. Otro método es alimentar un PLL con la señal, obteniendo la información como la señal de error del lazo de control.

2.5 Radio definida por software

Una radio definida por software (SDR) es un sistema de radiocomunicación donde algunos componentes que normalmente se implementan en hardware, como mezcladores, filtros, demoduladores, amplificadores, etc.) son realizados a través de software por medio de una PC o un sistema embebido.

Las etapas por las que pasa la señal dentro de un receptor o transmisor, pueden describirse matemáticamente, es decir cada componente tiene una respuesta conocida matemáticamente. Esto implica que si la señal se encontrara muestreada de forma digital y se le aplicaran funciones matemáticas con un procesador a esta señal digitalizada, se podrían obtener los mismos resultados que la señal obtendría si pasara por componentes reales que la modifiquen.

Una radio definida por software utiliza un conversor analógico digital para muestrear la señal y una vez que ésta se encuentra en forma digital se le aplican operaciones matemáticas en remplazo de los componentes que realizan la misma función de manera analógica.

Aunque se limite la resolución de la señal a aquella del conversor analógico digital, se pueden implementar operaciones que no son posibles de realizar por hardware o que serían muy costosas e imprecisas. Además el hecho de que se realicen por software brinda un dinamismo muy superior al receptor completamente en hardware. Para cambiar un parámetro del receptor solo se necesita cambiar el software y no algún componente de hardware.

2.5.1 Conversor analógico digital

Un conversor analógico digital (ADC) es un circuito electrónico que convierte una señal analógica en una señal digital. Convierte un voltaje o corriente de entrada a un número proporcional a su magnitud expresado digitalmente. Un conversor digital analógico (DAC) realiza la función inversa.

2.5.1.1 Arquitecturas de conversores analógicos digitales

Existen varias arquitecturas de hardware para implementar un conversor analógico digital. Las 2 más utilizadas son:

- **Por aproximaciones sucesivas:** Utiliza un comparador para comparar el voltaje de entrada con la salida de un conversor digital analógico interno que va aproximándose sucesivamente al valor correcto. En cada paso se obtiene un bit de conversión y el rango de comparación se reduce a la mitad.
- **Conversión directa o Flash:** Utiliza varios comparadores en paralelo, cada uno comparando con escaleras de voltajes construidas generalmente con varios resistores. La salida es luego codificada en binario por un encoder. Como no requieren acercarse al resultado en una serie de etapas son extremadamente rápidos, pero al requerir $2^n - 1$ comparadores para una conversión de n bits se tornan más costosos mientras aumenta la precisión.

2.5.1.2 Parámetros de un conversor analógico digital

Un conversor analógico digital convierte una señal analógica continua en el tiempo y en amplitud en una señal digital discreta en el tiempo y amplitud. El conversor puede trabajar solo hasta cierta velocidad. Esto implica que aparecerán errores, ruidos y limitaciones. Ciertos parámetros caracterizan a los conversores analógicos digitales para saber si se pueden utilizar en una aplicación específica. Entre los más importantes se destacan:

- **Resolución:** Indica la cantidad de valores posibles que puede tener la señal discretizada. Una resolución de 8 bits significa que los infinitos valores de amplitud de la señal se discretizarán en 256 distintos valores según su proximidad. Esto introduce un error de cuantificación por el redondeo de la entrada analógica a la salida digital, que es menor mientras mayor resolución tenga el conversor. Para aprovechar toda la resolución del conversor se debe amplificar la señal a convertir para que se utilice todo el rango de conversión.
- **Velocidad de muestreo:** Es el ritmo en el que la señal es muestreada. Si las muestras se toman cada un cierto tiempo conocido, la señal original puede ser reconstruida a partir de valores discretos de tiempo utilizando fórmulas de interpolación. Esto sólo se logra si la velocidad de muestreo es mayor al doble de la frecuencia máxima presente en la señal, según el teorema de Nyquist-Shannon.

CAPÍTULO 3

Microcontroladores y comunicación con PC

El capítulo 3 brinda un pequeño marco teórico sobre el uso de microcontroladores para configurar los componentes del receptor y establecer comunicación con la PC.

3.1 Microcontrolador

Un microcontrolador es un circuito integrado que contiene al menos un procesador (CPU), memoria y periféricos de entrada y salida programables. Se incluye memoria RAM y memoria de programa EEPROM o flash. Los microcontroladores están diseñados para sistemas embebidos. Su principal fuerte es reducir el tamaño y costo en comparación a diseños que utilizan procesador, memoria y dispositivos de entrada y salida separados. Usualmente los microcontroladores también contienen otros periféricos como conversores analógico digitales (ADC) y conversores digital analógicos (DAC), bloques de PWM, etc.

Los microcontroladores modernos permiten desarrollar prototipos de forma rápida y económica, con interfaces de programación simples y precios cada vez más bajos.

3.1.1 Periféricos y Entradas/Salidas

Los microcontroladores presentan pines que pueden ser utilizados como entrada o como salidas y estos permiten interactuar con otros componentes, pantallas, LEDs, sensores, etc.

Además los microcontroladores pueden incluir circuitos especiales para utilizar protocolos de comunicación a través de ciertos pines sin la necesidad de emular el protocolo a través de software (bit banging). Los protocolos de comunicación más comunes son UART, USB, SPI, I2C y Ethernet. Utilizando hardware dedicado se elimina cierta carga sobre el procesador.

3.1.2 Interrupciones

Las interrupciones le permiten a un sistema responder en tiempo real cuando ciertos eventos ocurren.

El sistema de interrupciones le señala al procesador que suspenda el procesamiento de la instrucción actual y que comience una rutina para “atender” la interrupción (ISR). Esta rutina incluye cualquier procesamiento necesario dependiendo de la interrupción.

Las interrupciones se utilizan para atender de manera rápida a los eventos que necesitan ese nivel de repuesta. En los sistemas de interrupciones avanzados, las interrupciones tienen un número de prioridad que indica qué interrupciones pueden interrumpir a otras, en caso que alguna interrupción interrumpa a otra mientras se la está atendiendo.

Los eventos que pueden causar interrupciones pueden ser externos como el cambio en un pin de entrada o internos como timers, estos últimos son muy utilizados para medir tiempo.

3.2 Comunicación USB

USB es un estándar que define cables, conectores y protocolos de comunicaciones para conectar, comunicar y alimentar computadoras y dispositivos. Fue diseñado para estandarizar la conexión de periféricos de computadora a computadoras personales, tanto para alimentar como para comunicar dichos dispositivos.

En cuanto a especificaciones de potencia todos los puertos USB trabajan a 5V, en el estándar USB 2.0 se permite un máximo de 500mA por puerto, mientras que en USB 3.0 este aumenta a 900mA, y en caso de utilizar solo alimentación el valor de corriente puede superar los 3A y el valor de tensión también puede aumentar a más de 20V.

En cuanto a especificaciones de velocidad de transferencia, en USB 1.0 se definieron ratios de 1.5Mbit/s (Low Speed) y 12Mbit/s (Full Speed). En USB 2.0 se definió 480Mbit/s (High Speed) y en USB 3.0 se definió 5Gbit/s (SuperSpeed).

3.2.1 Protocolo de comunicación

La arquitectura de comunicación USB es asimétrica, consistiendo en un host y múltiples dispositivos (devices). No se puede realizar comunicación entre dos host sin adaptadores que creen dispositivos en el medio. Algunos microcontroladores incluyen puertos que pueden hacer de host o device, pero en su mayoría estos sólo pueden actuar de device.

Las transferencias de datos son dirigidas por el host, quien realiza un polling¹¹ en los dispositivos conectados para revisar si quieren realizar una transferencia.

Las transferencias de datos pueden ser de los siguientes tipos:

- Transferencias isócronas: Garantizan ancho de banda específico.
- Transferencia interrupt: Para dispositivos que necesitan respuestas rápidas garantizadas (Como teclados o ratones.)
- Transferencias bulk: Transferencias grandes esporádicas utilizando todo el ancho de banda disponible, no garantiza latencia ni ancho de banda específicos.

La funcionalidad de un dispositivo USB puede ser definida por su código de clase que envía al host. Por lo general los dispositivos de Communications Device Class (CDC) pueden ser interpretados por la PC como puertos series COM virtuales.

Por lo general el protocolo de comunicación es simplificado mediante abstracción por hardware dedicado y bibliotecas.

3.3 Ringbuffers

Un ringbuffer o buffer circular es una estructura de datos que utiliza un buffer de tamaño fijo como si sus extremos estuvieran conectados. Utiliza un puntero de escritura que introduce los elementos y un puntero de lectura para leerlos, eliminando la necesidad

¹¹ Polling es el proceso por el cual una computadora o dispositivo espera a que un dispositivo externo verifique si se encuentra en determinado estado.

de desplazar todos los elementos cuando uno es leído. Por su funcionamiento el ringbuffer es un buffer adecuado para utilizar como FIFO.

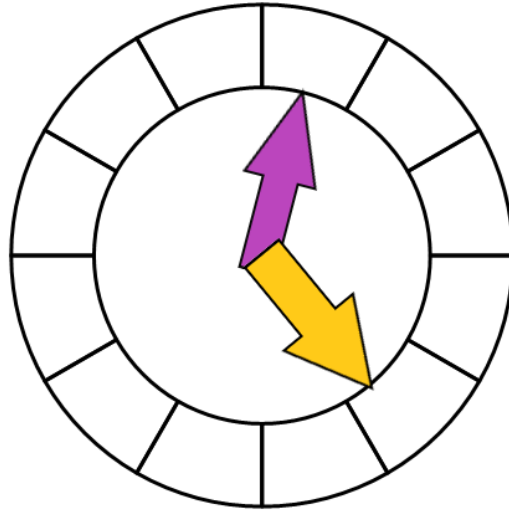


Figura 11 – Ringbuffer con los punteros de lectura y escritura

CAPÍTULO 4

Desarrollo del proyecto

El Capítulo 4 brinda un marco metodológico del desarrollo del prototipo.

4.1 Descripción general del proyecto

El objetivo principal de este trabajo es diseñar y construir un prototipo que sintonice, acondicione señales de RF para su digitalización, las digitalice y luego envíe la señal muestreada por USB a la PC. El proyecto consta de tres partes, la primera es la parte analógica de acondicionamiento de la señal de radiofrecuencia, la segunda es la de digitalización y comunicación con la PC y la tercera es el desarrollo de un driver que permita utilizar el dispositivo con el software de la PC.

El prototipo deberá entregar a la PC las muestras de la señal sintonizada en la frecuencia deseada dentro del rango de trabajo.

Debido a que el microcontrolador se conectará a la PC mediante USB, se aprovechará esa conexión para alimentar todos los componentes. Según las especificaciones USB 1.0 y 2.0, un puerto es capaz de entregar hasta 500mA en la línea de 5V. Por lo que, como criterio de diseño del dispositivo se incluye que su consumo se encuentre por debajo de ese límite.

Otro criterio a tener en cuenta en el diseño es que el sistema sea de fácil testeado, lo que implica que se pueda probar el comportamiento de los distintos componentes, cambiarlos en caso de mal funcionamiento y medir distintos valores de tensión y corriente en varias partes del circuito durante la fase de desarrollo. Para esto se busca que los

componentes sean compatibles con protoboard, de fácil soldado y en lo posible de encapsulado de agujero pasante.

4.1.1 Diagrama de Bloques General

A continuación se muestra el diagrama de bloques general del proyecto:

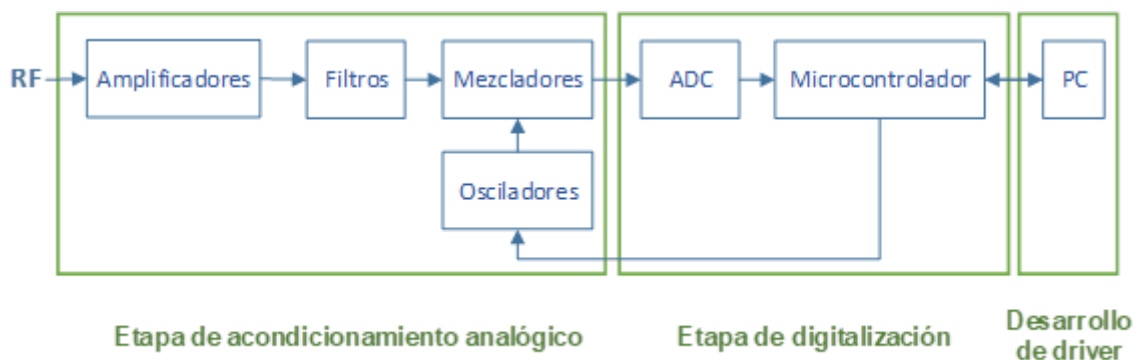


Figura 12- Diagrama de bloques del proyecto

El desarrollo del prototipo se divide en 3 etapas, el acondicionamiento analógico, la digitalización y el driver de PC.

El desarrollo del acondicionamiento analógico comprende:

- Elección de la arquitectura del receptor
- Diseño e implementación de filtros
- Elección de amplificadores, mezcladores y osciladores

El desarrollo de la etapa de digitalización comprende:

- Programación del microcontrolador para que:
 - Reciba la configuración deseada por USB
 - Muestree la señal con su convertor analógico-digital.
 - Configure los osciladores a las frecuencias necesarias.
 - Envíe las muestras por USB

El desarrollo del driver en la PC comprende:

- Desarrollo de un módulo para el software libre GNU Radio en el sistema operativo Linux que :
 - Envíe la configuración deseada al dispositivo
 - Reciba las muestras y las inserte en el flujo de muestras de GNU Radio

4.2 Acondicionamiento analógico

A continuación se desarrollan las etapas que conforman el front end de radiofrecuencia encargado de sintonizar y acondicionar la señal para su digitalización.

4.2.1 Arquitectura del receptor

Se optó por diseñar un receptor superheterodino de doble conversión, con una conversión llevando la señal sintonizada a una frecuencia intermedia para simplificar su filtrado y selectividad, y otra para llevar la señal a banda base donde será digitalizada.

Se prefirió esta arquitectura por sobre la arquitectura que no realiza mezcla, ya que esta última resulta mucho más costosa en cuanto al ADC y además requiere de una velocidad de transferencia muy grande, y por lo tanto también costosa, para transferir todas las muestras a la PC.

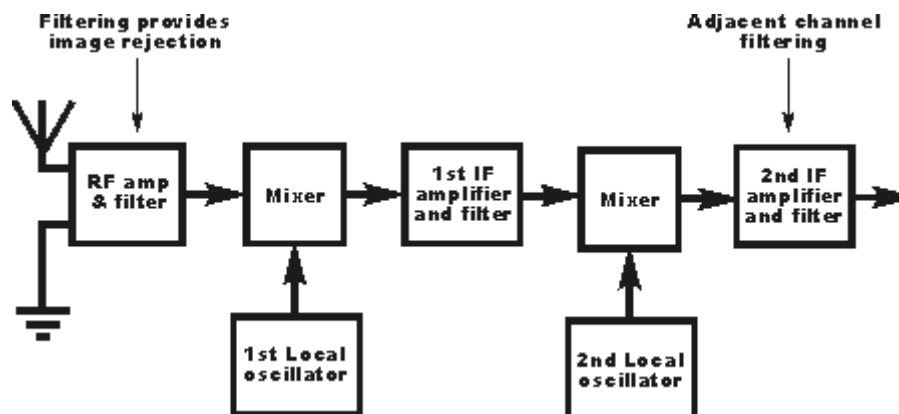


Figura 13 – Receptor superheterodino de doble conversión.¹²

La razón de la doble conversión es la mayor selectividad del canal al utilizar una conversión extra, y al poder utilizar una frecuencia intermedia mayor se aleja la frecuencia imagen que se debe filtrar, la segunda FI es más baja, y permite una etapa de digitalización más simple.

Esta arquitectura precisa de dos osciladores locales, uno de los cuales debe ser variable para poder sintonizar la señal que se quiere recibir, mientras que el otro no

¹² Imagen obtenida de: <http://www.radio-electronics.com/info/rf-technology-design/superheterodyne-radio-receiver/double-superheterodyne-receiver.php> [Última visita: Octubre 2017]

requiere ser variable, ya que la diferencia entre la frecuencia intermedia y la banda base es siempre la misma. También requiere de dos mezcladores.

4.2.2 Osciladores

Ya que la frecuencia a sintonizar debe ser definida digitalmente, los osciladores deben poder variarse a través de una interfaz digital. Entre las distintas opciones, debido a que el DDS no posee los problemas de inestabilidad de un PLL, es un sistema íntegramente digital y posee rapidez para cambiar de frecuencia, se lo consideró superior para esta aplicación ya que las frecuencias utilizadas en este proyecto se encuentran dentro de los rangos en los que comúnmente trabaja este tipo de oscilador.

Los osciladores seleccionados son dos sintetizadores digitales de ondas senoidales AD9850 de Analog Devices. Cada uno ofrece rango dinámico libre de espurias mayor a 50dB en salidas de 40MHz, puede ser programado con una precisión de 0,0291 Hz y permite hasta 23 millones de cambios de frecuencia por segundo. Llegan hasta 50MHz y la configuración puede ser cargada serialmente o en paralelo. Para utilizar la menor cantidad de líneas del microcontrolador, es preferible utilizar el modo serial.

El sintetizador AD9850 fue seleccionado junto a su placa de desarrollo que incluye cristal de clock y polarización necesaria para su funcionamiento.

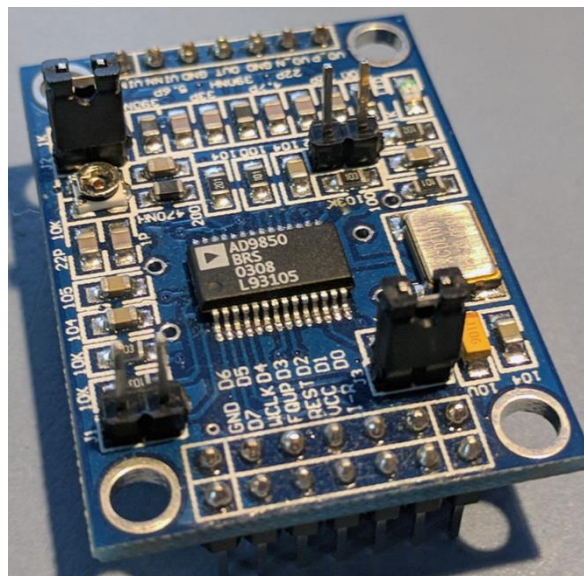


Figura 14 - Placa de desarrollo del sintetizador AD9850

Este sintetizador puede ser alimentado con 5V o 3.3V, y su salida tiene como mínimo 200mVPaP. Consume como máximo 96mA. Se encuentra disponible en pequeñas placas de desarrollo incluyendo cristales y otros componentes que facilitan su uso.

4.2.3 Mezcladores

Los mezcladores deben poder funcionar con señales dentro del rango de trabajo, la frecuencia intermedia y banda base. El mezclador seleccionado NE602AN de Philips Semiconductors es un mezclador activo de baja potencia para frecuencias de VHF.

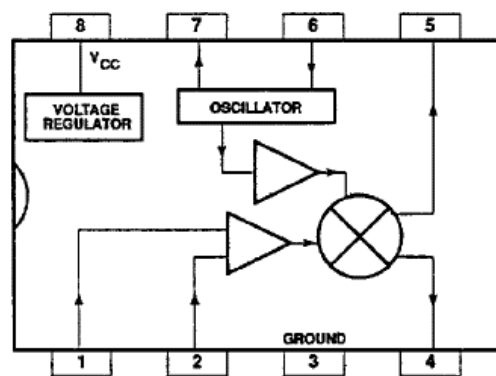


Figura 15 – Diagrama de bloques del NE602AN¹³

El mezclador NE602AN provee una ganancia típica de 18dB, puede configurarse con un cristal, circuito tanque o con oscilador externo. Tiene una baja figura de ruido. Se encuentra disponible en empaquetado DIP8 permitiendo su uso en protoboard, siendo esta una característica decisiva para la elección.

¹³ Imagen obtenida de datasheet del componente. Philips Semiconductors.

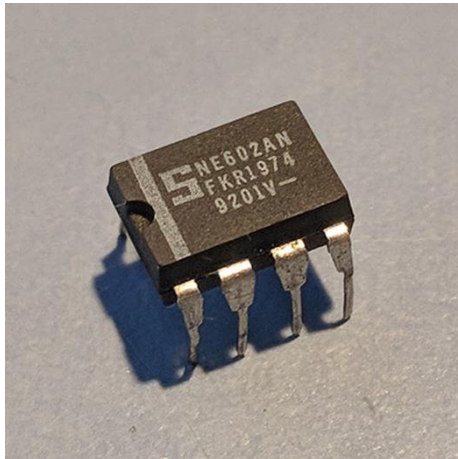


Figura 16 – NE602AN empaquetado DIP

La señal de entrada de este mezclador puede llegar hasta 500MHz y es capaz de recibir señales de hasta -119dBm. Tiene punto de intercepción de tercer orden aproximado de -13dBm. Se alimenta con 5V y consume típicamente 2,4mA. No requiere ningún tipo de polarización especial para sus entradas, de hecho, las señales deben ingresar filtradas por un capacitor para no contener componente continua. Su alimentación también debe ser filtrada para evitar que la señal escape por ahí.

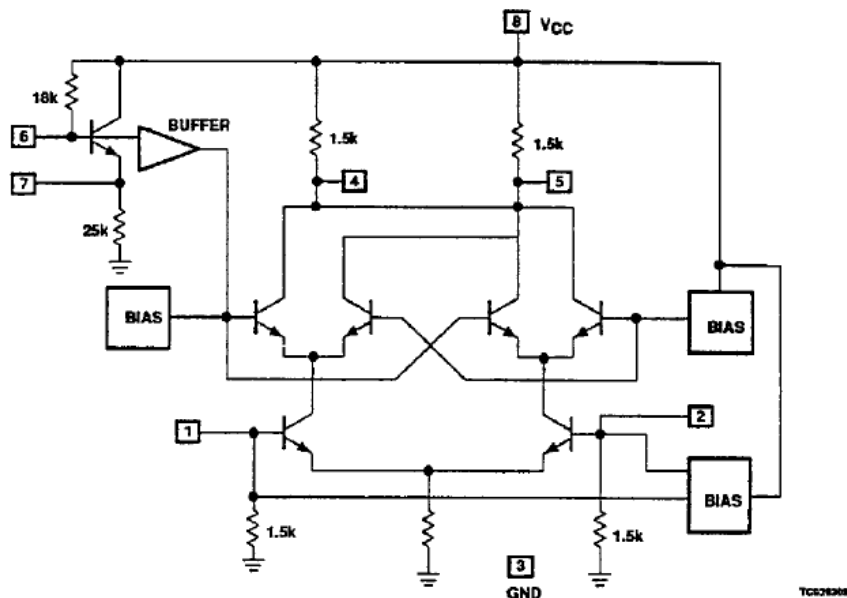


Figura 17 – Circuito Equivalente del NE602AN¹⁴

¹⁴ Imagen obtenida de datasheet del componente. Philips Semiconductors.

Como se observa en el circuito equivalente, los pines 4 y 5 que contienen la salida del mezclador se encuentran polarizados con un resistor de $1,5k\Omega$ conectado a la fuente de alimentación, y de conectar alguno de estos pines a tierra, se produciría un cortocircuito a tierra y el mezclador dejaría de funcionar. Además el pin 6 de la entrada del oscilador no puede ir a tierra, aunque el pin 7 sí, es decir, la entrada del oscilador no es simétrica como lo es la entrada de RF y la salida de IF.

4.2.4 Amplificadores

Existen una amplia gama de amplificadores monolíticos que trabajan en el rango de frecuencias requerido para este proyecto.

Entre ellos se seleccionaron MAR-3+ y, posteriormente, MAR-1+ de MiniCircuits, las características más importantes de cada uno son las siguientes:

- MAR-3+:
 - Rango desde CC a 2GHz
 - Ganancia típica de 12,5dB
 - Compensado internamente para 50 Ohms
 - Figura de ruido de 3,7dB
 - Incondicionalmente estable
 - Tensión de Operación 5V, Corriente 35mA
 - Potencia de salida en punto de 1dB de compresión 10dBm
 - Punto de intercepción de tercer orden (salida) 23dBm
- MAR-1+:
 - Rango desde CC a 1GHz
 - Ganancia típica de 17,8dB
 - Compensado internamente para 50 Ohms
 - Figura de ruido de 3,5dB
 - Incondicionalmente estable
 - Tensión de Operación 5V, Corriente 17mA
 - Potencia de salida en punto de 1dB de compresión 2.5dBm
 - Punto de intercepción de tercer orden (salida) 14dBm



Figura 18 – Amplificadores MAR-1 (izquierda) y MAR-3 (Derecha)

Ambos se encuentran en encapsulado de montaje superficial VV105. Se alimentan por el pin de salida de RF por lo que es necesario un inductor de choque. El fabricante recomienda colocar una resistencia para limitar la corriente y aplicar una tensión mayor de por lo menos 7V.

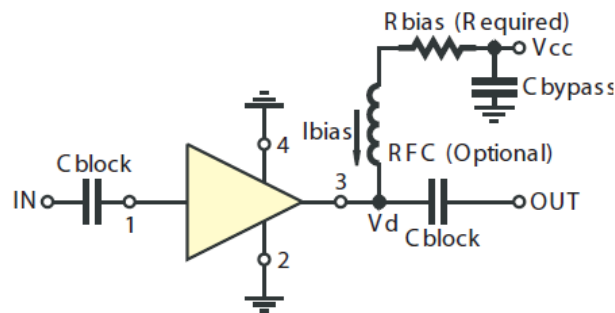


Figura 19 – Circuito recomendado de aplicación de amplificadores MAR según fabricante.¹⁵

Debido a que el amplificador MAR-1 tiene una ganancia típica mayor, una figura de ruido menor y un consumo menor, se lo consideró superior para este caso de uso con respecto al MAR-3. La razón de su elección es su bajo ruido, su amplia ganancia, su frecuencia de trabajo y que es incondicionalmente estable.

Para la amplificación en banda base se seleccionaron amplificadores operacionales los cuales, mediante jumpers, se pueden desactivar o activar variando así la amplificación de banda base.

¹⁵ Imagen obtenida de datasheet del componente. Mini-Circuits.

Los amplificadores operacionales seleccionados fueron MCP6294 de Microchip por sus características rail-to-rail, slew rate, y GBP. Se incluyen 4 amplificadores operacionales por encapsulado.



Figura 20 – MCP6294 encapsulado DIP

Entre las características del MCP6294 se destacan:

- Rail-to-Rail en entrada y salida
- Gain Bandwidth Product de 10MHz
- Consumo de corriente típico de 1mA
- Slew Rate 7V/uS

La señal obtenida a la salida del mezclador tiene una amplitud esperada calculable con las ganancias de los distintos componentes, pero superpuesta a una continua de valor no conocido (la salida máxima no será superior a la alimentación). Con los niveles de amplificación del circuito se espera una salida aproximada menor a 200mVpp.

Para amplificar la señal, se incluye un capacitor para filtrar la tensión continua antes de cada amplificador operacional, y luego se le suma un nivel de tensión conocido equivalente a un cuarto del valor de tensión de alimentación (1,25V para 5V de alimentación), que es suficientemente mayor que la amplitud de la señal. De esta manera los valles de la señal no cruzan por cero y toda la señal se mantiene en valores positivos, y cuando se duplica su amplitud tampoco se supera la tensión de alimentación (5V) y no se pierde información de la señal ni por corte ni por saturación.

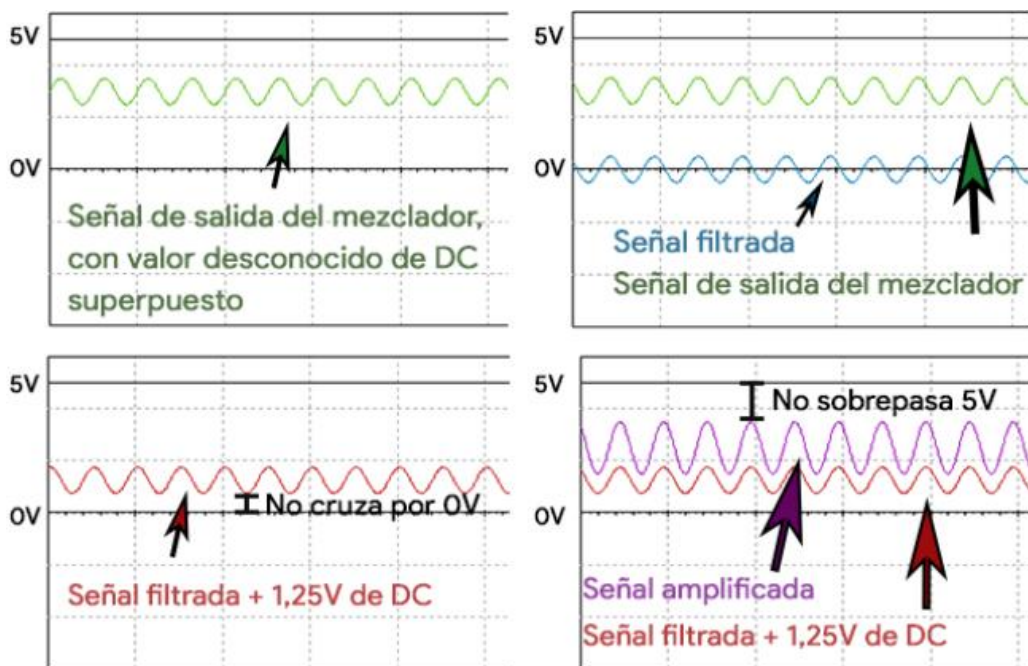


Figura 21 – Amplificación de la señal sin sufrir distorsión por corte o saturación

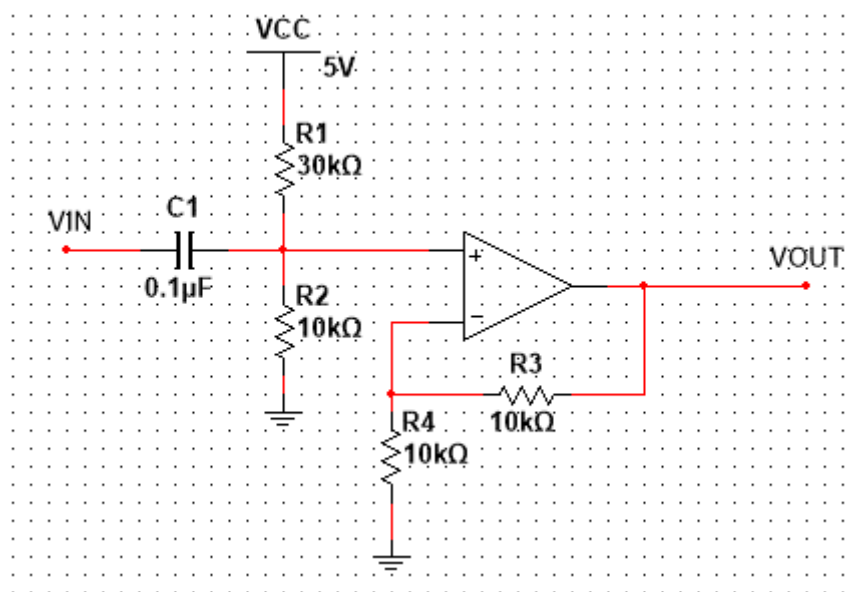


Figura 22 – Circuito que filtra la señal continua, le suma un cuarto de la tensión de alimentación y amplifica por dos.

El mismo procedimiento para amplificar la señal se realiza con amplificadores alimentados con 3,3V y escalando los valores de tensión correspondientemente.

Previo a la entrada del ADC, la componente continua desconocida de la señal es eliminada y al resultado se le suma la mitad de la tensión de referencia del ADC, de manera de colocar la señal en el medio del rango de valores del conversor. El ADC utiliza 3,3V de referencia, por lo que el valor de tensión a sumar es 1,65V.

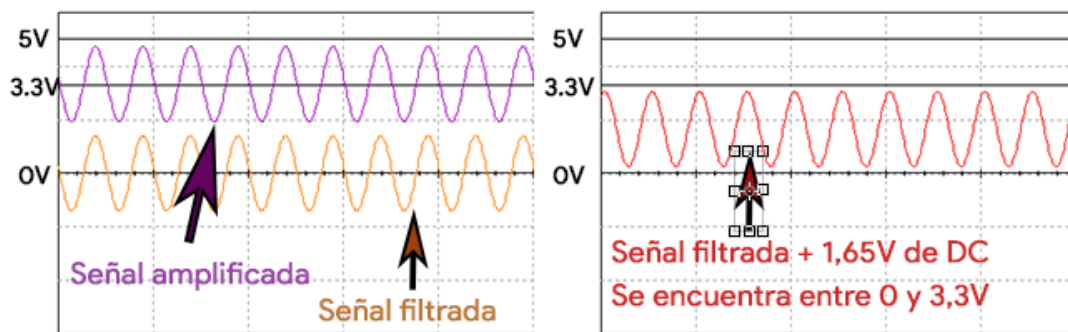


Figura 23 – Señal filtrada y desplazada al centro del rango del ADC

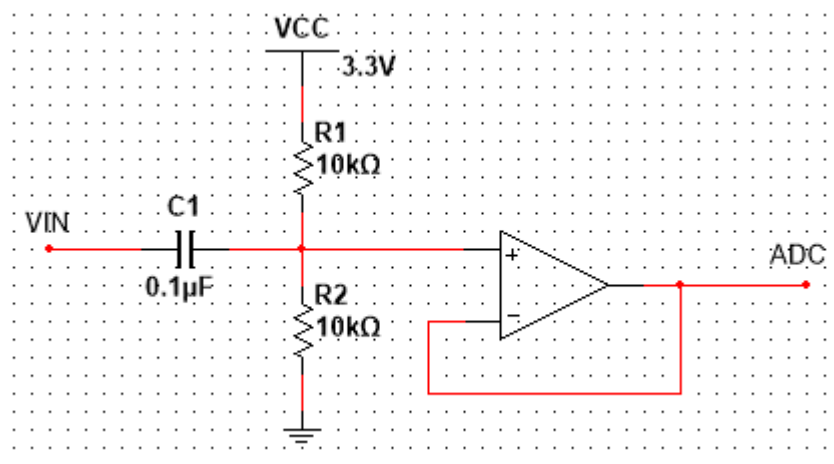


Figura 24 – Circuito que filtra la señal y le suma la mitad de la tensión de referencia, incluyendo un buffer previo al ADC

4.2.5 Filtros

En este trabajo, se implementaron 4 tipos de filtros.

- Un filtro para la etapa de RF, ubicado después del primer amplificador que filtra las frecuencias fuera del rango de trabajo del dispositivo.

- Un filtro para la etapa de IF, de tipo cerámico que filtra un pequeño ancho de banda y brinda mayor selectividad.
- Un filtro para la etapa de Banda Base que elimina altas frecuencias que pueden encontrarse a la salida del mezclador.
- Dos filtros pasa bajos a la salida de los osciladores para eliminar ruidos de mayores frecuencias producidos por los sintetizadores

A continuación se analiza cada tipo de filtro con más detalle.

4.2.5.1 Filtro de la etapa de RF

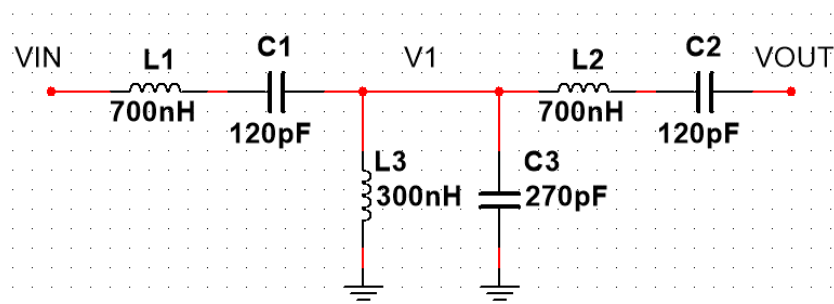


Figura 25 – Diagrama esquemático del filtro de la etapa de RF

El objetivo de este filtro es que las señales de frecuencias inferiores a 10MHz y las superiores a 30MHz sean fuertemente atenuadas para no producir interferencia por frecuencia imagen.

Función de transferencia:

Si $C2 = C1$ y $L2=L1$

$$H(s) = \frac{V_{out}(s)}{V_{in}(s)}$$

$$V_{out}(s) = \frac{V_1(s) \cdot RL}{Z_{L1} + Z_{C1} + RL}$$

$$V_1(s) = \frac{(V_{in}(s) \cdot ((Z_{L3}) \parallel Z_{C3} \parallel (Z_{L1} + Z_{C1} + RL)))}{(Z_{L1} + Z_{C1} + ((Z_{L3}) \parallel Z_{C3} \parallel (Z_{L1} + Z_{C1} + RL)))}$$

$$V_{out}(s) = \frac{(V_{in}(s) \cdot ((Z_{L3}) \parallel Z_{C3} \parallel (Z_{L1} + Z_{C1} + RL)))}{(Z_{L1} + Z_{C1} + ((Z_{L3}) \parallel Z_{C3} \parallel (Z_{L1} + Z_{C1} + RL)))} \cdot RL$$

$$H(S) = \frac{((Z_{L3}) \parallel Z_{C3} \parallel (Z_{L1} + Z_{C1} + RL)) \cdot RL}{(Z_{L1} + Z_{C1} + ((Z_{L3}) \parallel Z_{C3} \parallel (Z_{L1} + Z_{C1} + RL)) \cdot (Z_{L1} + Z_{C1} + RL)}$$

Teniendo en cuenta que:

$$Z_L = L \cdot S$$

$$Z_C = \frac{1}{C \cdot S}$$

$$((Z_{L3}) \parallel Z_{C3} \parallel (Z_{L1} + Z_{C1} + RL)) = \frac{1}{\frac{1}{Z_{L3}} + \frac{1}{Z_{C3}} + \frac{1}{Z_{L1} + Z_{C1} + RL}}$$

$$((Z_{L3}) \parallel Z_{C3} \parallel (Z_{L1} + Z_{C1} + RL)) = \frac{1}{\frac{1}{L3 \cdot S} + C3 \cdot S + \frac{1}{L1 \cdot S + \frac{1}{C1 \cdot S} + RL}}$$

$$H(S) = \frac{\frac{1}{\frac{1}{L3 \cdot S} + C3 \cdot S + \frac{1}{L1 \cdot S + \frac{1}{C1 \cdot S} + RL}} \cdot RL}{\left(L1 \cdot S + \frac{1}{C1 \cdot S} + \frac{1}{\frac{1}{L3 \cdot S} + C3 \cdot S + \frac{1}{L1 \cdot S + \frac{1}{C1 \cdot S} + RL}} \right) \cdot \left(L1 \cdot S + \frac{1}{C1 \cdot S} + RL \right)}$$

$H(S) =$

$$\frac{S^3 \cdot L3 \cdot C1^2 \cdot RL}{S^6 \cdot L1^2 \cdot L3 \cdot C1^2 \cdot C3 + S^5 \cdot L1 \cdot L3 \cdot C1^2 \cdot C3 \cdot RL + S^4 \cdot L1 \cdot C1 \cdot (L1 \cdot C1 + 2 \cdot L3 \cdot (C1 + C3)) + S^3 \cdot C1 \cdot RL \cdot (L1 \cdot C1 + L3 \cdot (C1 + C3)) + S^2 \cdot (2 \cdot L1 \cdot C1 + L3 \cdot (2 \cdot C1 + C3)) + S \cdot C1 \cdot RL + 1}$$

Si $RL = 50 \Omega$, $C1 = 120pF$, $L1 = 700nH$, $L3 = 300nH$ y $C3 = 270pF$

$$H(S) = \frac{2,16 \cdot 10^{-25} \cdot S^3}{5,71536 \cdot 10^{-49} \cdot S^6 + 4,0824 \cdot 10^{-41} \cdot S^5 + 2,6712 \cdot 10^{-32} \cdot S^4 + 1,206 \cdot 10^{-24} \cdot S^3 + 3,21 \cdot 10^{-16} \cdot S^2 + 6 \cdot 10^{-9} \cdot S + 1}$$

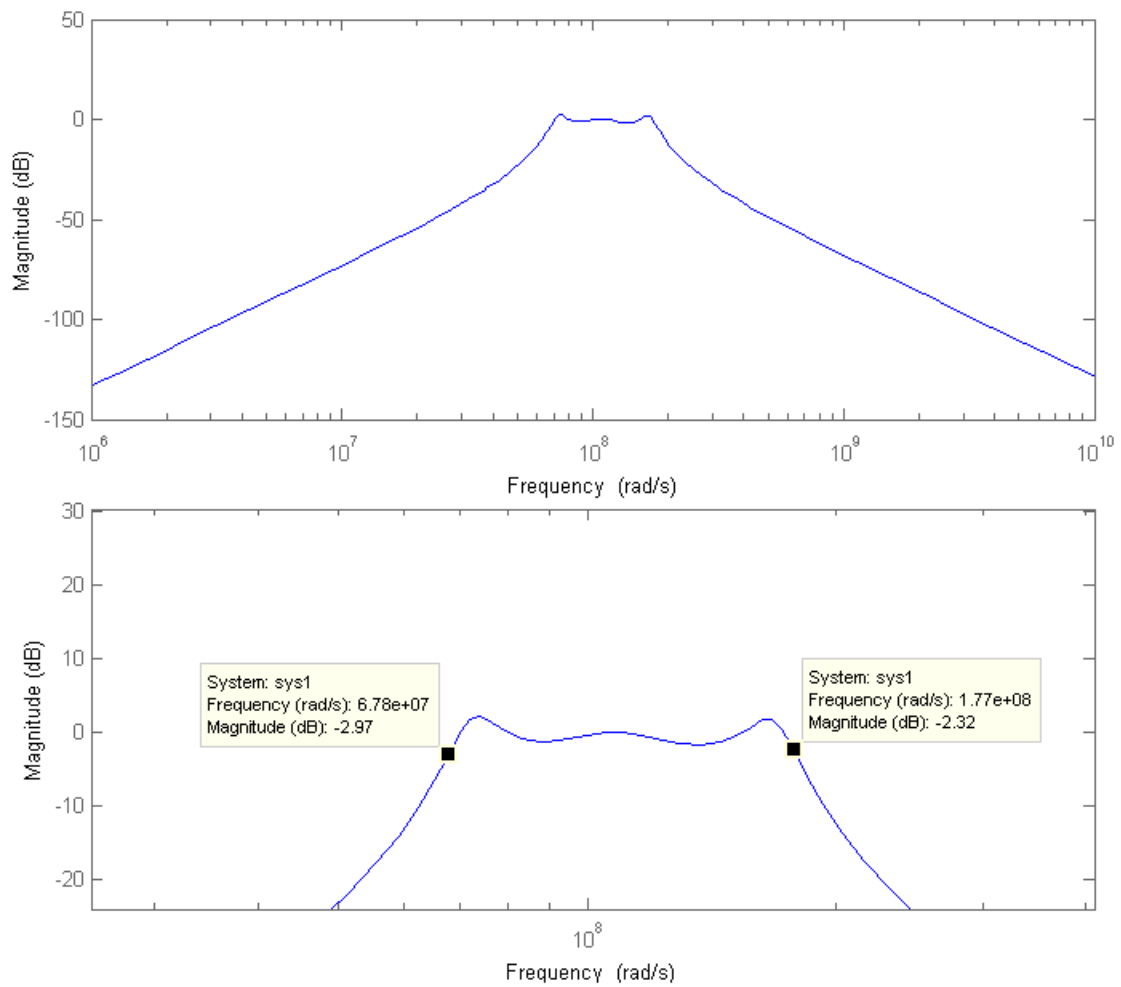


Figura 26 – Diagrama de Bode del filtro de la etapa de RF

Como se observa en el diagrama de Bode del filtro, éste se comporta como pasabanda con frecuencias de corte aproximadamente entre 10,79 MHz y 28,17 MHz, siendo estos valores cercanos a los deseados.

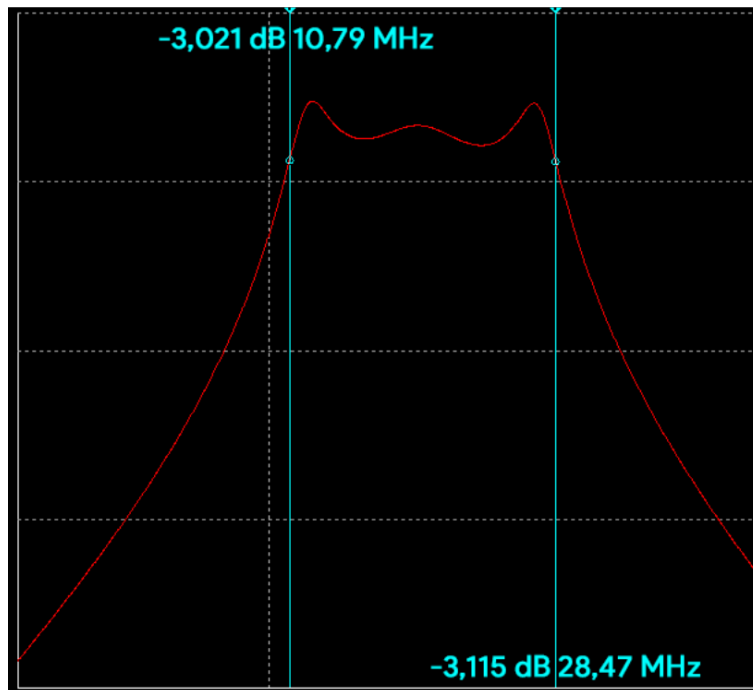


Figura 27 – Respuesta del filtro según simulación

Los resultados de simulación dan una respuesta similar al análisis matemático y al esperado.

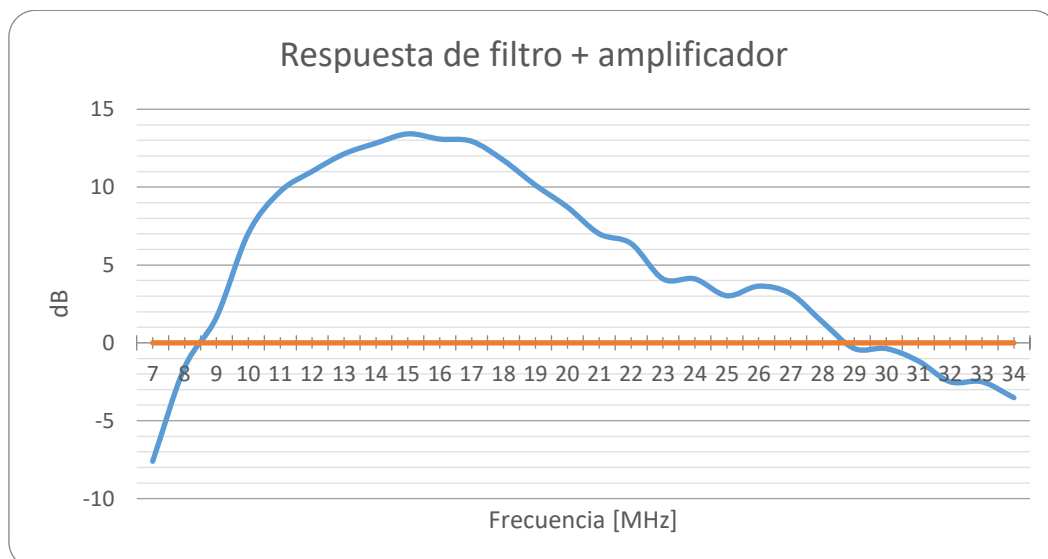


Figura 28 – Respuesta de mediciones del filtro junto a un amplificador MAR-3+

Las mediciones realizadas del filtro junto a un amplificador muestran una respuesta asimétrica dentro del rango de trabajo del equipo, amplificando más frecuencias entre 9MHz y 22MHz, aún así se observa que la banda de paso del filtro es similar a la

esperada, y ya que su principal uso es atenuar señales fuera del rango de trabajo para evitar la frecuencia imagen, se lo consideró aceptable. Las diferencias entre las mediciones y los valores calculados probablemente se deban a las imperfecciones de los componentes utilizados. Se utilizaron capacitores e inductores fijos con distintos valores de tolerancia.

4.2.5.2 Filtro de la etapa de IF

El filtro de la frecuencia intermedia utilizado es un filtro SFELF10M7HAA0-B0 de muRata de encapsulado cerámico.

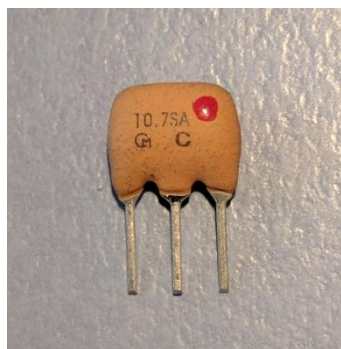


Figura 29 – Filtro cerámico SFELF10M7HAA0-B0

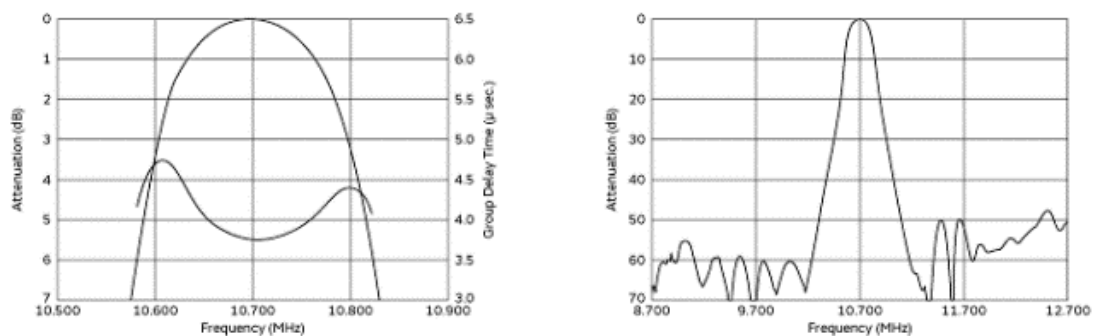


Figura 30 – Respuesta del filtro cerámico¹⁶

El ancho de banda de este filtro es de 180KHz con frecuencia central en 10,7MHz y tiene una pérdida de inserción máxima de 7dB. Se seleccionó este filtro por su formato de agujero pasante y determinó la primera frecuencia intermedia en 10,7 MHz

¹⁶ Imagen obtenida de datasheet del componente. muRata.

4.2.5.3 Filtro de la etapa de banda base

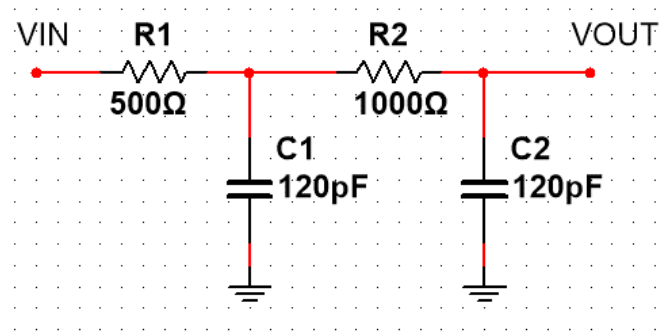


Figura 31 – Diagrama esquemático del filtro de la etapa de banda base

El objetivo del filtro es que las señales de frecuencias superiores a las que se quieren mostrar (180KHz) sean atenuadas, en especial las del oscilador e IF (10,7MHz). Es un filtro RC pasa bajo de segundo orden.

Función de transferencia:

$$H(S) = \frac{V_{out}(S)}{V_{in}(S)}$$

$$H(S) = \frac{(Z_{C1} \parallel (R2 + Z_{C2})) \cdot Z_{C2}}{(R1 + (Z_{C1} \parallel (R2 + Z_{C2}))) \cdot (R2 + Z_{C2})}$$

Teniendo en cuenta que:

$$Z_c = \frac{1}{C \cdot S}$$

$$H(S) = \frac{\frac{1}{C1 \cdot S + \frac{1}{R2 + \frac{1}{C2 \cdot S}}} \cdot \frac{1}{C2 \cdot S}}{\left(R1 + \frac{1}{C1 \cdot S + \frac{1}{R2 + \frac{1}{C2 \cdot S}}} \right) \cdot \left(R2 + \frac{1}{C2 \cdot S} \right)}$$

$$H(S) = \frac{1}{C1 \cdot R1 \cdot S \cdot (C2 \cdot R2 \cdot S + 1) + C2 \cdot S \cdot (R1 + R2) + 1}$$

Si $R1 = 500 \Omega$, $R2 = 1000 \Omega$, $C1 = 120pF$ y $C2 = 120pF$

$$H(S) = \frac{1}{7,2 \cdot 10^{-15} \cdot S^2 + 2,4 \cdot 10^{-7} \cdot S + 1}$$

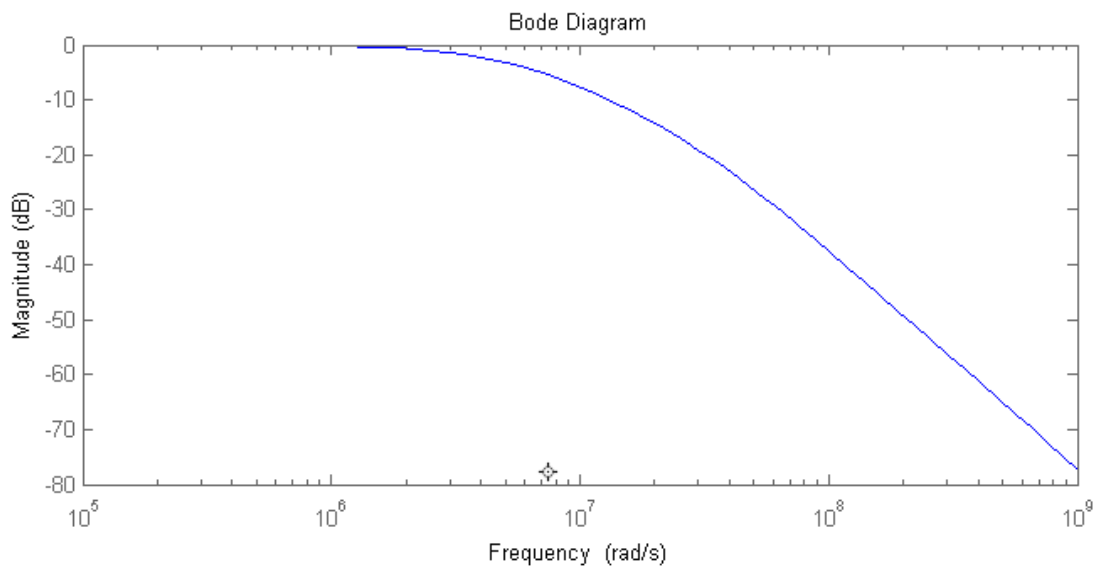


Figura 32 – Diagrama de Bode del filtro de banda base

Se observa que para las frecuencias de banda base (180KHz) la atenuación es mínima. Su frecuencia de corte es aproximadamente 750KHz. Las señales cercanas a 10MHz que pueden provenir del mezclador u oscilador reciben una atenuación de 30dB.

4.2.5.4 Filtro de salida de osciladores

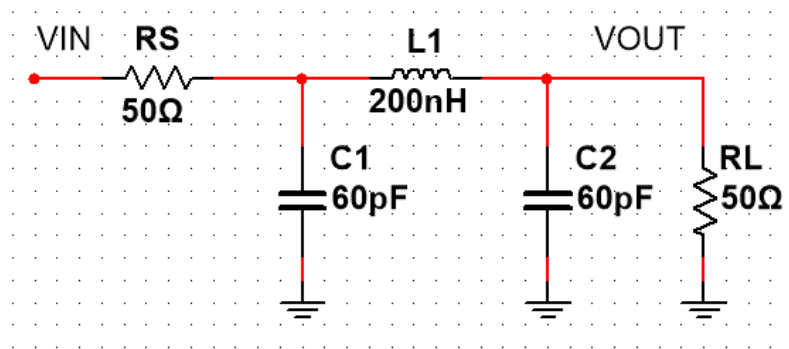


Figura 33 – Diagrama esquemático del filtro sobre la salida de los osciladores

El filtro es un pasa bajo LC diseñado para que las frecuencias superiores a 50MHz sean atenuadas.

Función de transferencia:

$$H(S) = \frac{Vout(S)}{Vin(S)}$$

$$H(S) = \frac{(Z_{C1} \parallel (Z_{L1} + (Z_{C2} \parallel RL))) \cdot (Z_{C2} \parallel RL)}{(RS + (Z_{C1} \parallel (Z_{L1} + (Z_{C2} \parallel RL)))) \cdot (Z_{L1} + (Z_{C2} \parallel RL))}$$

Teniendo en cuenta que:

$$Z_L = L \cdot S$$

$$Z_C = \frac{1}{C \cdot S}$$

$$H(S) = \frac{\left(\frac{1}{C1 \cdot S + \frac{1}{L1 \cdot S + \frac{1}{C2 \cdot S + \frac{1}{RL}}}} \right) \cdot \left(\frac{1}{C2 \cdot S + \frac{1}{RL}} \right)}{\left(RS + \frac{1}{C1 \cdot S + \frac{1}{L1 \cdot S + \frac{1}{C2 \cdot S + \frac{1}{RL}}}} \right) \cdot \left(L1 \cdot S + \frac{1}{C2 \cdot S + \frac{1}{RL}} \right)}$$

$H(S)$

$$= \frac{RL}{C1 \cdot RS \cdot S \cdot (C2 \cdot L1 \cdot S^2 \cdot RL + L1 \cdot S + RL) + C2 \cdot S \cdot RL \cdot (L1 \cdot S + RS) + L1 \cdot S + RS + RL}$$

Si: $C1 = C2 = 60\text{pF}$, $L1 = 200\text{nH}$, $RS = RL = 50 \Omega$

$$H(S) = \frac{0,5}{1,8 \cdot 10^{-26} \cdot S^3 + 1,2 \cdot 10^{-17} \cdot S^2 + 5 \cdot 10^{-9} \cdot S + 1}$$

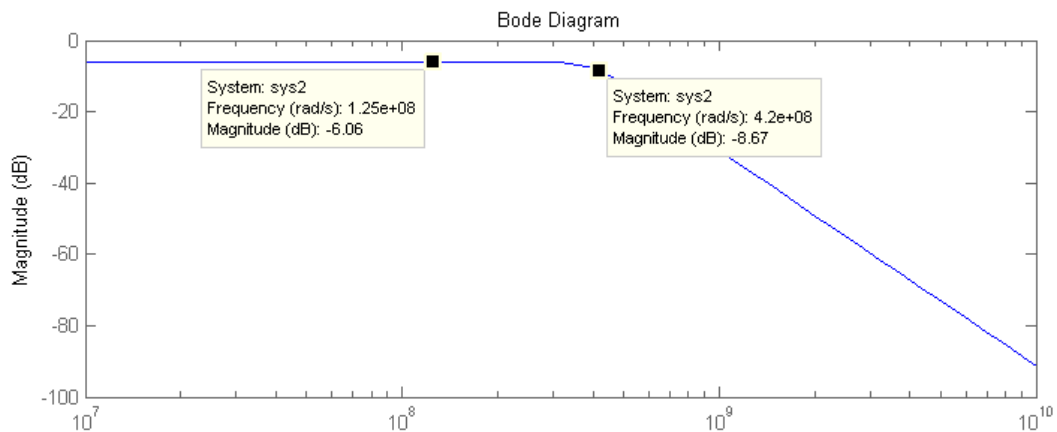


Figura 34 – Diagrama de Bode del filtro sobre la salida de los osciladores

Debido a que se tiene en cuenta la caída sobre la resistencia interna del oscilador, el gráfico muestra una atenuación general de 6dB. La frecuencia de corte del filtro se encuentra aproximadamente a los 67MHz, permitiendo que todos los posibles valores del oscilador se encuentren en la banda de paso, pero que los ruidos producidos por el switching del conversor o de su clock sean atenuados.

4.2.6 Alimentación

Por criterio diseño se buscó que todos los componentes se alimenten a través de la conexión USB. El micro controlador contiene un convertor a 3,3V, por lo que este valor de tensión está disponible junto con 5V. Debido a que los amplificadores de RF (MAR-1 y MAR-3) requieren una tensión mayor, se utilizó un convertor DC-DC (Switching) para elevar 5V a 7V, colocando varios capacitores en sus bornes para eliminar el mayor ruido posible, y en la cercanía de los amplificadores también.

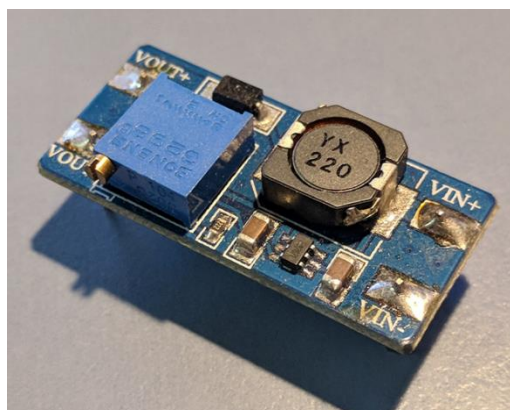


Figura 35 – Conversor DC-DC

4.3 Etapa de microcontrolador y digitalización

En esta etapa se seleccionaron un microcontrolador y un conversor analógico-digital para poder configurar los osciladores y transformar la señal en banda base a digital y enviarla por USB.

En una primera instancia se evaluó utilizar una computadora de placa reducida Raspberry Pi 3, pero como ésta utiliza un sistema operativo que no permite acceder a los periféricos con una latencia de tiempo real, se optó por un microcontrolador con mejor acceso a componentes de bajo nivel.

4.3.1 Microcontrolador

Originalmente se seleccionó una placa de evaluación LPC1769, con un microcontrolador Cortex-M3 con frecuencia hasta 120MHz con 512kB de memoria flash, 64kB de memoria ram y USB 2.0 full-speed. Debido a dificultades con el entorno de programación LPCXpresso para encontrar la placa de evaluación por USB y forzando a utilizar una versión discontinuada en una PC sin las últimas actualizaciones del sistema operativo para poder programarla, y que las bibliotecas provistas por el fabricante para utilizar los distintos periféricos del microcontrolador no contenían documentación, se decidió cambiar de microcontrolador.

Se optó por la placa Teensy 3.2 que es un sistema de desarrollo de microcontrolador basado completamente en USB.

Tabla 2 – Especificaciones de Teensy 3.2

Microcontrolador	MK20DX256VLH7
Núcleo de Procesador	Cortex-M4
Frecuencia de reloj	96 MHz
Memoria Flash	256 KB
RAM	64 KB
Pines I/O	34 (24 con pines pasantes) (12 con salida PWM) (Tolerantes a 5V)
Corriente máxima por pin	10mA
Pines con entrada analógica	21
Conversores ADC	2
Conversores DAC	1
Timers	12
USB	1
UART	3
SPI	1
I2C	2
Tensión de operación	3.3V
Tensión de entrada	3.3V o 5V (USB)

Los conversores analógico digitales pueden ser configurados para velocidades de conversión superiores a 600kSps.

Para programarlos se añade un componente llamado “Teensyduino” al software de programación de “Arduino”. Este componente permite programar la placa con el código en lenguaje “Processing” utilizado comúnmente para programar

microcontroladores de la plataforma mencionada y brinda compatibilidad con muchas de sus bibliotecas. “Teensyduino” incluye además bibliotecas para el uso de todos los periféricos con amplia documentación. El uso de este entorno de programación permite un desarrollo rápido de código y su fácil prueba.

La placa Teensy 3.2 se alimenta directo de USB y tiene su propio conversor a 3,3V capaz de alimentar otros componentes hasta un total de 250mA. También tiene un pin de 5V que se alimenta del USB. A través de estos pines se alimentarán los componentes de la etapa analógica, logrando que el dispositivo se alimente completamente por USB.

4.3.2 Conversor Analógico-Digital (ADC):

Originalmente se seleccionó un conversor externo ADCS7476 de Texas Instruments de 12 bits de resolución y hasta 1MSps con interfaz SPI. Luego se optó por utilizar el conversor interno de la placa Teensy, ya que este era más configurable y eliminaba costo de procesamiento del procesador con el manejo de SPI.

El conversor interno utiliza una referencia de 3,3V y supera los 600kSps con una resolución de 8 bits.

4.3.3 Programación de microcontrolador:

El código completo utilizado para programar el microcontrolador en este proyecto se encuentra en el Anexo y en el DVD provisto junto con este informe. A continuación se explicará su funcionamiento.

4.3.3.1 Diagrama en bloques del código del microcontrolador

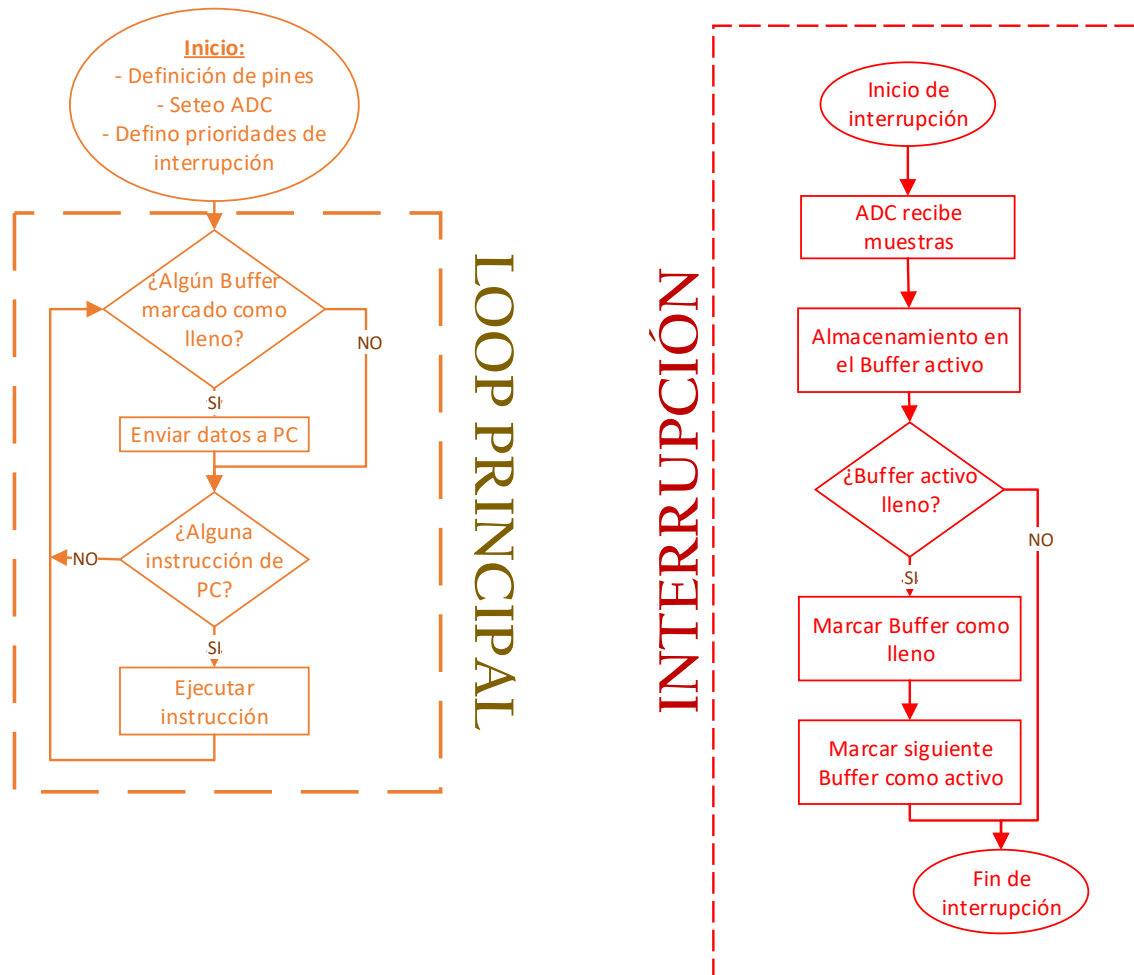


Figura 37 – Diagrama en bloques del código del microcontrolador

4.3.3.2 Muestreo

El muestreo de la señal analógica debe realizarse regularmente (con un intervalo constante) para poder recomponer la señal digitalmente, por esta razón la función que se encarga de tomar una muestra debe llamarse dentro de una interrupción de un timer, asegurando que no se demore por otra operación que el microcontrolador pueda estar realizando.

Aunque la comunicación USB es a 12Mbps/s, se observó que la operación de agregar bytes a transmitir al buffer USB tiene una demora adicional. Por ello se optó por utilizar tres buffers que almacenen varias muestras para ser enviadas en conjunto. La

función de muestreo toma el valor del conversor analógico digital y lo coloca en uno de tres buffers, cuando un buffer se ha completado se comienza a transmitir por USB mientras los demás buffers se siguen llenando. Los buffers tienen 16364 bytes cada uno, pero se los completa hasta un veintidosavo (1/22) de la velocidad de muestreo seteada, ignorando el resto de su capacidad, de forma que tomen el mismo tiempo en llenarse (aproximadamente 45,46ms) y transmitirse sin importar la velocidad de muestreo. En caso contrario si se eligiera una velocidad de muestreo suficientemente menor, podría demorarse muchísimo tiempo en llenarse un buffer. Una vez colocado el byte de muestra en un buffer se aumenta un contador que cuenta cuantas muestras hay en el buffer, en caso que se haya llenado se marca que el buffer está listo para enviarse y se selecciona otro buffer para llenar luego.

4.3.3.3 Configuración de osciladores

Los sintetizadores AD9850 se configuran de forma serial o paralela, en este caso se optó de forma serial. Se configuran a través de una palabra de 40 bits, los primeros 32 bits indican la frecuencia deseada y los siguientes 8 bits incluyen información de la fase deseada y de control para el fabricante, que no son utilizados en este proyecto. Se envía cada bit por un pin de datos y se realiza un pulso sobre un pin de clock para que el sintetizador lo reciba, al final de la palabra de 40 bits se realiza un pulso sobre el pin de actualización de frecuencia.

Los 32 bits de frecuencia que se tienen que enviar dependen del valor de clock que tenga el sintetizador. Las placas de evaluación tienen un clock de cristal de 125MHz. Los 32 bits de frecuencia a enviar siguen la siguiente fórmula:

$$\text{Valor a enviar} = \text{frecuencia deseada} * \frac{\text{Valor máximo de 32bits}}{\text{Clock de AD9850}}$$

$$\text{Valor a enviar} = \text{frecuencia deseada} * \frac{4294967295}{125000000}$$

El valor del entero resultante, expresado en binario, se envía ordenado desde el bit menos significativo primero.

Una función específica para transmitir un byte al oscilador, escribe el bit menos significativo en el pin de datos del oscilador elegido y realiza un pulso en el pin de clock, luego desplaza el byte un bit hacia la derecha. Esta operación se realiza 8 veces en un loop, dentro de la función, para enviar todo el byte.

La función para fijar la frecuencia recibe como argumentos la frecuencia deseada y el oscilador elegido y llama a la función encargada de transmitir un byte 5 veces (para los 40 bits).

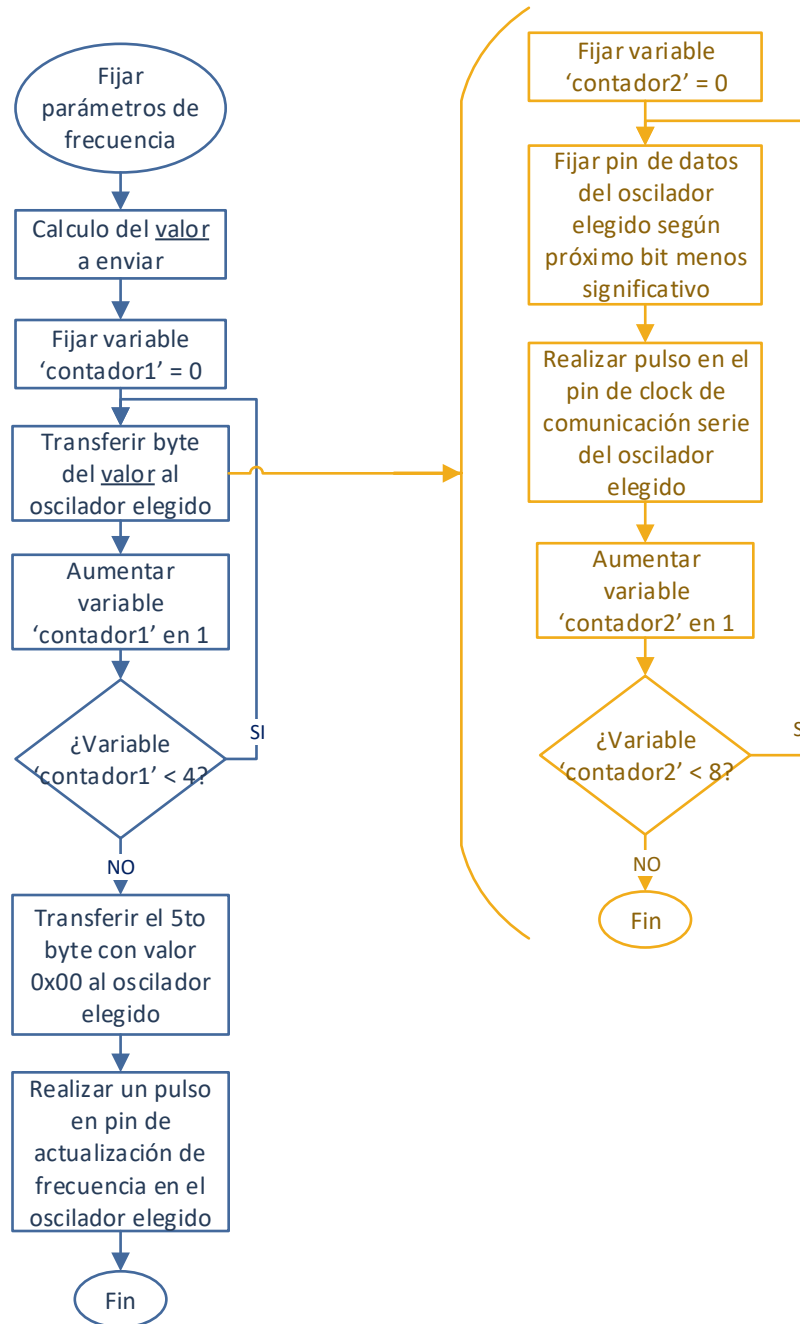


Figura 38 – Diagrama de flujo de las funciones que programan los osciladores

Debido a que el ancho de banda del canal es de 180kHz, la frecuencia central del canal debe ubicarse a los 90kHz. Para esto cuando se requiere sintonizar una frecuencia, el primer oscilador se configura en esa frecuencia sumada a la IF, y el segundo oscilador

a la IF sumada a 90kHz, dando como salida del segundo mezclador el canal de 180kHz centrado en 90kHz.

4.3.3.4 Recibiendo la configuración de la PC

Cuando el microcontrolador no está configurando la frecuencia de un oscilador ni se ocupa de atender la interrupción, se encuentra esperando instrucciones por el puerto USB. Analiza los caracteres recibidos en formato ASCII, uno por uno, para entender que instrucción se le ha enviado. Todas las instrucciones que recibe empiezan con el signo '@', seguido por un número del 1 al 5 que indica la operación a ejecutar y luego 8 caracteres numéricos que indican qué número se usa como argumento para esa instrucción. Si se recibe un carácter que no es '@' ni un número, se ignora tanto el carácter como la instrucción que se estaba construyendo. Todas las instrucciones tienen 10 caracteres de longitud y en caso que el número del argumento tenga menos de 8 dígitos deberá ser precedido por ceros.

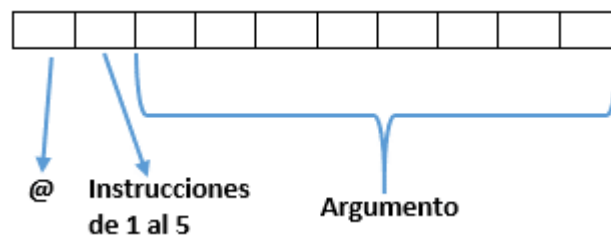


Figura 39 - Formato de instrucciones que recibe el microcontrolador

Las instrucciones posibles son:

1. Setear frecuencia oscilador 1 (@1#####)

Fija la frecuencia a sintetizar por el oscilador 1, devuelve “Valor inválido” si el argumento es mayor a 50.000.000

2. Setear frecuencia oscilador 2 (@2#####)

Fija la frecuencia a sintetizar por el oscilador 2, devuelve “Valor inválido” si el argumento es mayor a 50.000.000

3. Setear cantidad de muestras por segundo (@3#####)

Fija la velocidad de muestreo en muestras por segundo deseadas, modificando el intervalo de interrupción del timer, si el número es mayor a 360.000 devuelve “Valor inválido”, si el número es cero desactiva el muestreo.

4. Setear frecuencia de sintonización (@4#####)

Setea las frecuencias de ambos osciladores para que la frecuencia a sintonizar quede en banda base desplazada 90kHz (centro de ancho de banda que se muestrea). Si el número no está entre 10.000.000 y 30.000.000 devuelve valor inválido.

5. Setear frecuencia de sintonización – rápido (@5#####)

Realiza lo mismo que la instrucción 4 pero no responde ninguna confirmación. Esta instrucción se puede utilizar en programas que necesiten cambiar rápidamente de frecuencia sintonizada.

Todas las instrucciones envían una respuesta por USB con confirmación de su ejecución a excepción de la instrucción número 5.

4.4 Diseño de driver en PC

La comunicación USB se recibe como comunicación por puerto serie virtual, que funciona como un puerto de comunicación serie pero se ignoran los valores de baud rate, paridad y otras especificaciones y se usa el protocolo de USB.

4.4.1 Utilizando el puerto serie (virtual)

El módulo del programa será programado en C++. Para comunicarse con el puerto serie en Linux se puede acceder al dispositivo como si fuera un archivo o utilizar una biblioteca especial como termios. La configuración del puerto se guarda en una estructura “struct termios”. Y utilizando las funciones open, close, read y write se abre una conexión, se cierra, recibe o envían datos.

4.4.2 Bloque de GNU Radio

Los bloques de GNU Radio se insertan en el flujo de muestras y dependiendo del tipo de bloque tienen una entrada y/o una salida. En el caso del módulo para usar de

interface con el dispositivo solo tendrá salida de muestras (las que ha digitalizado), es decir será una fuente de muestras. Para crear un bloque, primero se debe crear un módulo en el cual se incluya dicho bloque.

Los módulos se crean utilizando la herramienta `gr_modtool` incluida en GNU Radio. Esta herramienta crea carpetas y archivos, donde lo único que hay que modificar es el código fuente de los bloques, que pueden ser escritos en C++ o en Python.

El código fuente utilizado para el bloque de GNU Radio se encuentra en el Anexo.

Los bloques pueden ser de varios tipos, en este caso se hizo un bloque tipo “source”, es decir es “fuente” de muestras para GNU Radio.

Luego para añadir el bloque a GNU Radio Companion, se creó un archivo “.xml” que indica que tipo de valores pedir para los argumentos del bloque y que tipo de bloque es. Luego se compiló e instaló el módulo con Cmake.

El nombre del módulo creado en este proyecto es “PIFE” (abreviación de Proyecto Integrador Front End) y el bloque es “PIFESource”.

Este bloque toma como argumento para instanciarse el nombre del puerto, la frecuencia de sintonización y la velocidad de muestreo. En el método de implementación abre una conexión con el puerto especificado y le envía las instrucciones para detener el muestreo (si es que está muestreando), sintonizar la frecuencia deseada y luego comenzar a muestrear a la velocidad indicada, esperando siempre la confirmación del microcontrolador. Luego invoca un hilo que se encarga de leer todos los datos disponibles del puerto. Cada byte es una muestra porque son 8 bits de resolución. A cada muestra se le resta 128 (la mitad de 8 bits de resolución) para eliminarle la tensión continua que fue agregada para que toda la señal se encuentre dentro del rango del conversor. Luego se transforma cada muestra en un “float” entre -1 y 1.

Las muestras ya convertidas a un número real entre -1 y 1 son almacenadas en un ringbuffer, que como fue explicado en el marco teórico utiliza un puntero para escribir y otro para leer. Cuando GNU Radio requiere muestras, el bloque las busca directamente en el ringbuffer y se las entrega para que las puede utilizar el resto del programa.

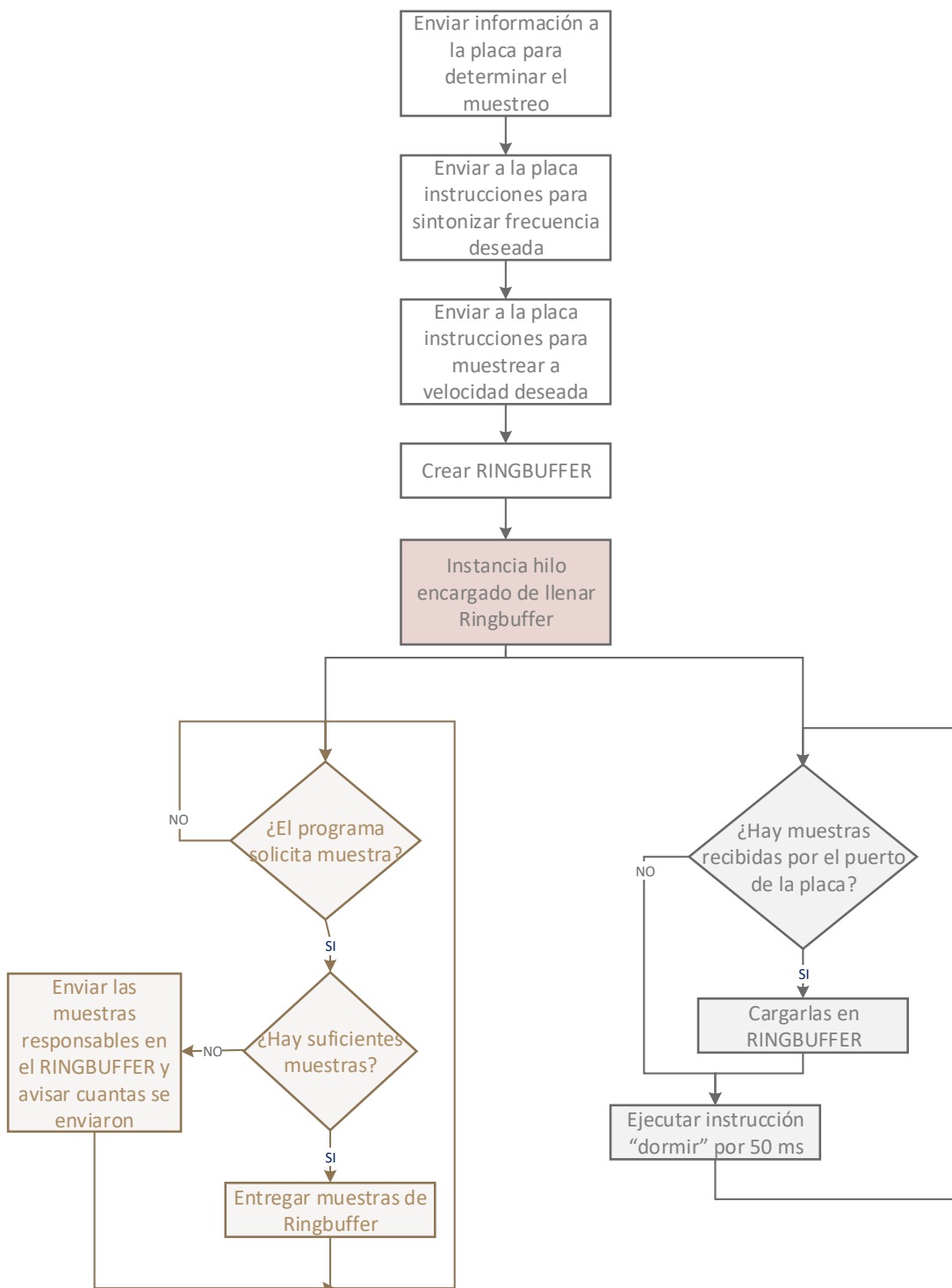


Figura 40 – Diagrama de flujo del bloque de GNU Radio encargado de recibir muestras de la placa

Una vez instalado el módulo, se lo puede utilizar dentro de GNU Radio Companion. Entre la lista de bloques se encuentra el módulo PIFE y dentro el bloque PIFESource.

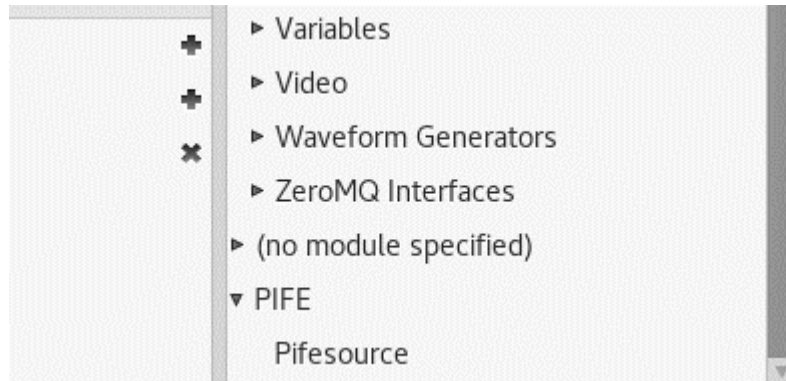


Figura 41 – Módulo y bloque en la lista de GNU Radio Companion

El bloque se puede utilizar como cualquier otro bloque fuente. Las muestras que salen del bloque se encuentran en formato de punto flotante, y si se quisiera utilizar con bloques que utilizan entrada compleja se debe utilizar un bloque Hilbert previamente.

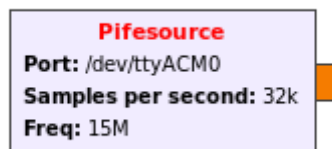


Figura 42 – Bloque Pifesource dentro de GNU Radio Companion

El bloque tiene 3 argumentos:

- El puerto de conexión, que viene completado con el puerto por defecto, sólo en raras ocasiones se necesitará cambiar.
- La cantidad de muestras por segundo, cuyo valor no puede superar los 360kSps.
- La frecuencia a la que se debe sintonizar el receptor, que se debe encontrar entre 10MHz y 30MHz.

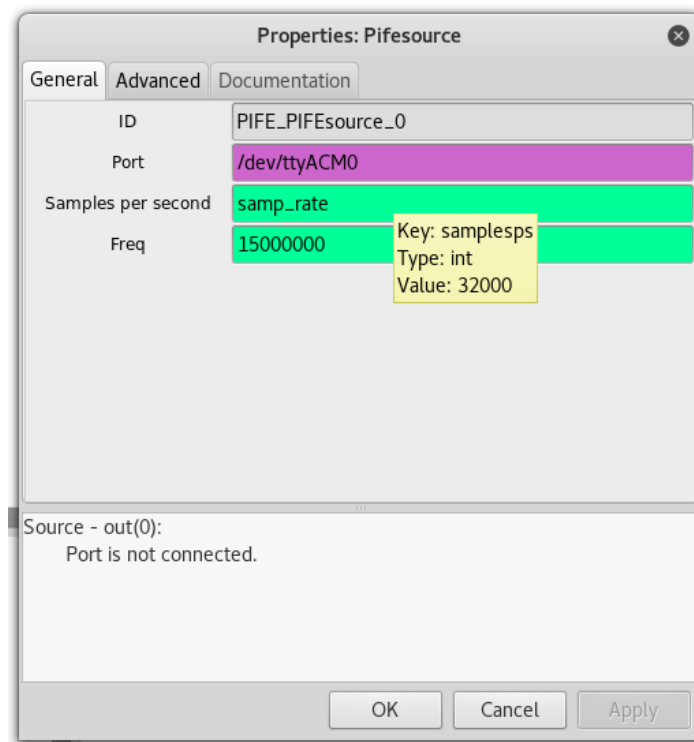


Figura 43 – Configuración del bloque dentro de GNU Radio Companion.

4.5 Circuito Final

El circuito final incluye los 2 mezcladores, los filtros de RF, IF, banda base y de los osciladores, 3 amplificadores en la etapa de RF, 2 amplificadores en la etapa de IF y 5 amplificadores operacionales en banda base. Incluye además pines para conectar los osciladores, el microcontrolador y el convertor DC-DC en forma modular. Incluye pines extra para colocar jumpers y un conector SMA para conectar una antena.

Su construcción comenzó por el diseño del diagrama esquemático del circuito, que se encuentra en el Anexo. Luego se le asignó a cada componente una forma y huella en la placa de circuito impreso y se los ordenó de la manera más conveniente posible teniendo en cuenta las conexiones con componentes externos y el trazado de las pistas de cobre. En el caso de este circuito impreso, se utilizó solo una capa de cobre. Para colocar los pines de conexión para el microcontrolador, los osciladores y el convertor DC-DC, se midieron estos componentes y su separación entre pines.

Luego se realizó el trazado de las pistas de cobre, intentando evitar posibles interferencias entre las mismas.

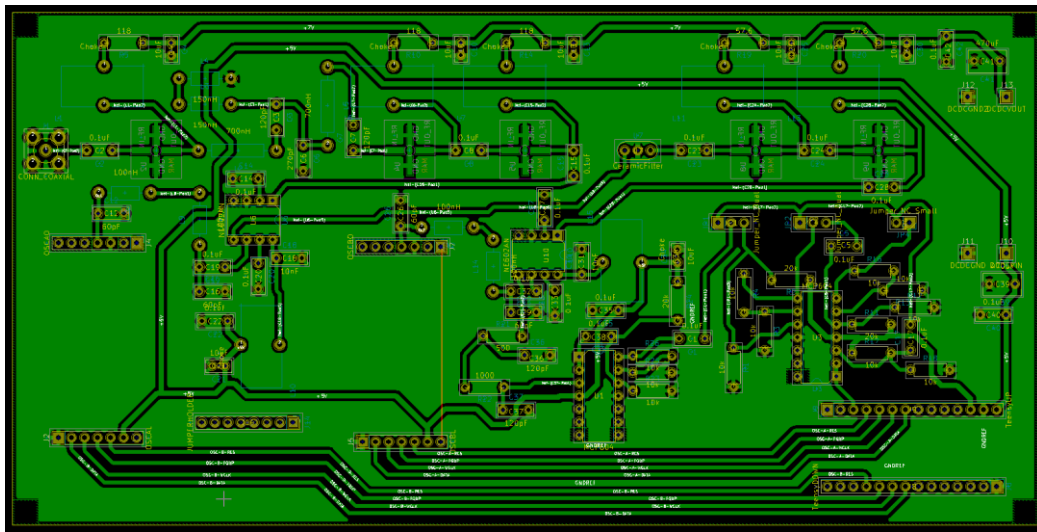


Figura 44 – Diseño de PCB

Una vez que se diseñó el PCB se procedió a desarrollarlo mediante serigrafía. El proceso consiste en imprimir el diseño en papel con una impresora láser y transferir el tóner con una plancha eléctrica.

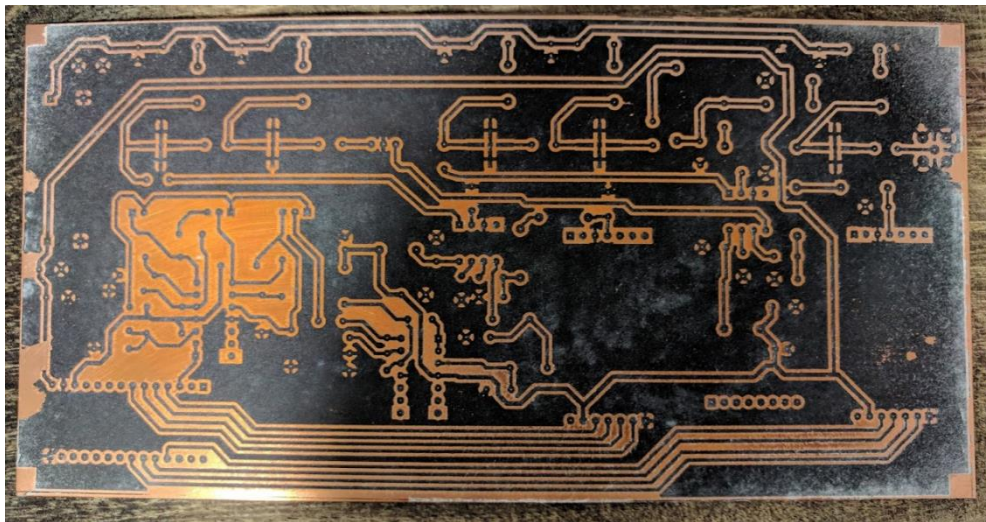


Figura 45 – Tóner transferido a la placa

Una vez transferido el tóner sobre la capa de cobre, en caso de encontrar imperfecciones se pueden corregir con marcador permanente, y finalmente se lo sumerge en percloruro férrico. El ácido disuelve el cobre que no es protegido por el tóner, dejando solamente las pistas de cobre deseadas sobre la placa de fibra de vidrio.

Para un mejor funcionamiento, el ácido se debe encontrar a baño maría entre 45 y 50 grados.



Figura 46 – Placa de cobre sumergida en percloruro férrico a baño maría.

Luego de realizado este procedimiento para el PCB de este trabajo, se procedió a perforar los agujeros para los componentes y a aplicarle una capa de flux en aerosol, esto permite que su soldado sea más fácil.

Una vez seca la capa de flux, se procedió a soldar los componentes, y finalmente a probar su funcionamiento.

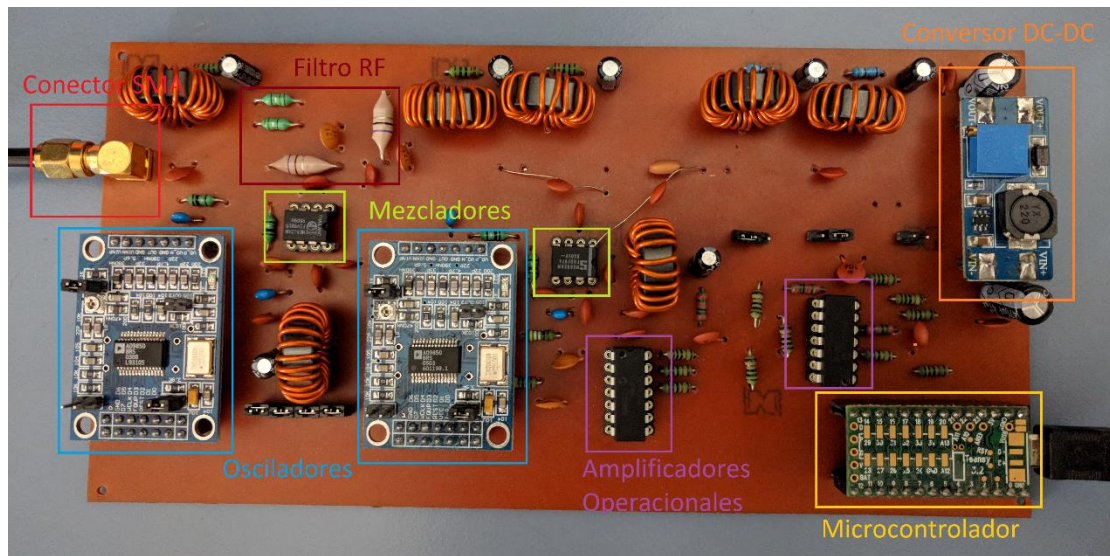


Figura 47 – Placa de circuito final con nombre de componentes

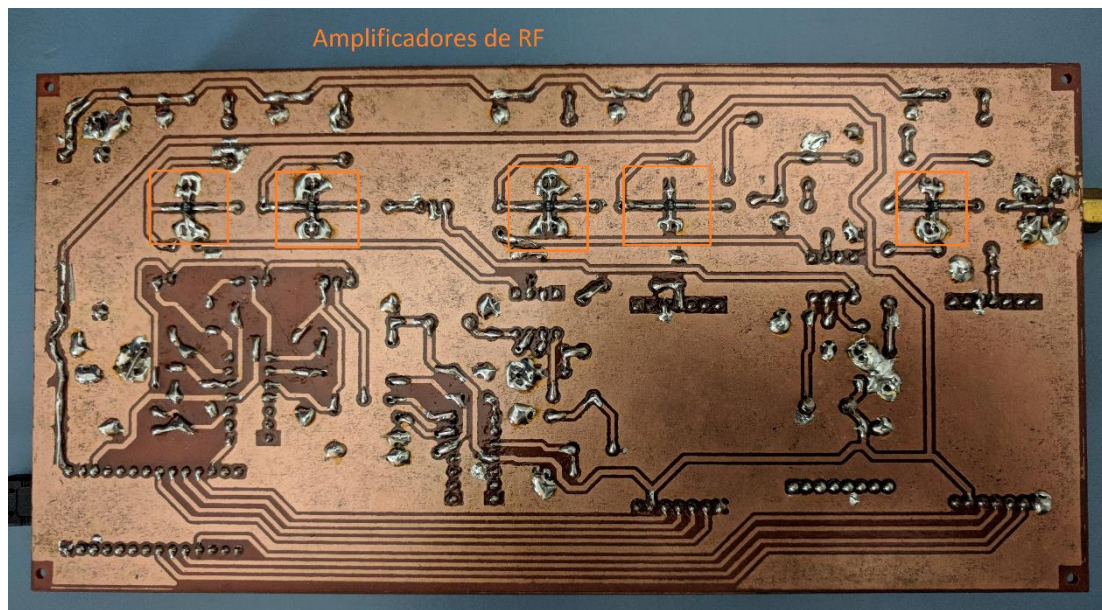


Figura 48 – Placa de circuito final vista de abajo

CAPÍTULO 5

Pruebas Realizadas

Se realizaron pruebas para verificar la respuesta de las distintas etapas del proyecto, y para corroborar el funcionamiento de los componentes que las integran. Una vez desarrollada la placa del circuito final se hicieron pruebas del sistema completo.

5.1 Pruebas Iniciales

Se efectuaron pruebas con osciladores y osciloscopio para probar los mezcladores y los sintetizadores utilizando protoboards. Se midieron en osciloscopio las ondas generadas por los osciladores para medir la exactitud de la frecuencia y su amplitud. También se midió la salida de los mezcladores utilizando dos osciladores de entrada. Se observó que los osciladores no tenían ningún problema para emitir cualquier frecuencia deseada, y que la amplitud no disminuía de los 200mV_{pap}. En cuanto a los mezcladores, se corroboró que la salida de los mismos correspondía a la mezcla de las frecuencias entrantes.

Se elaboró una prueba del primer filtro y mezclador del front end, junto con un amplificador en una placa de circuito impreso.

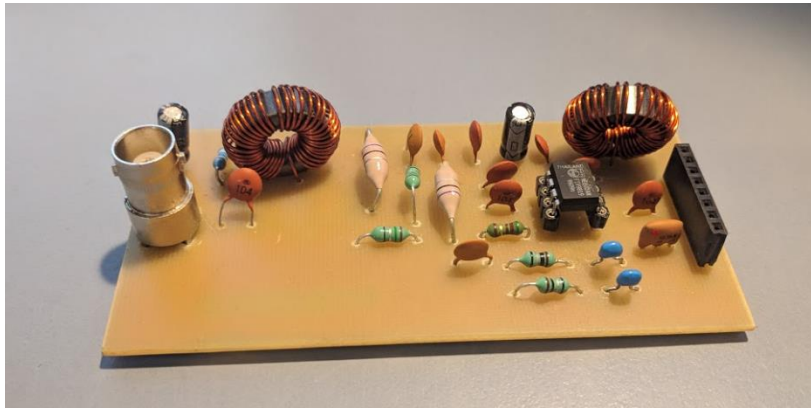


Figura 49 – Circuito para prueba de filtro de etapa de RF

Hubo un error en el diseño del circuito y la entrada del osciloscopio en el mezclador se encontraba invertida (pin 6 fue conectado a tierra), y la salida del mezclador se encontraba conectada a tierra (como se expresó en la descripción del mezclador, sus salidas no pueden ser conectadas a tierra). La prueba fue clave para reconocer estos errores, que fueron corregidos en el diseño de las siguientes placas. Se logró obtener datos sobre la respuesta del filtro de front end, y éste funcionaba de forma aceptable. El gráfico de su respuesta se encuentra en la sección del filtro.

Se llevaron a cabo diferentes pruebas de software con el microcontrolador. Primero para probar la comunicación USB, se enviaron series de números y se observó si se recibían en la PC sin perder información. Se probó el muestreo del ADC, tomando valores directamente de los osciladores para asegurarse que se muestreaba la señal correctamente. En estas pruebas se visualizó que la toma de muestras del ADC y su envío por USB limitaba la velocidad máxima a apenas más de 360Ksps, por lo que se definió utilizar un ancho de banda de 180KHz.

Antes de realizar una placa con el circuito final se lo probó completo con protoboard. Debido al tamaño del circuito se dificultaba su prueba, y los cables de conexión de protoboard no soportaban la corriente suficiente para la alimentación completa del circuito, y eran fuente de ruido.

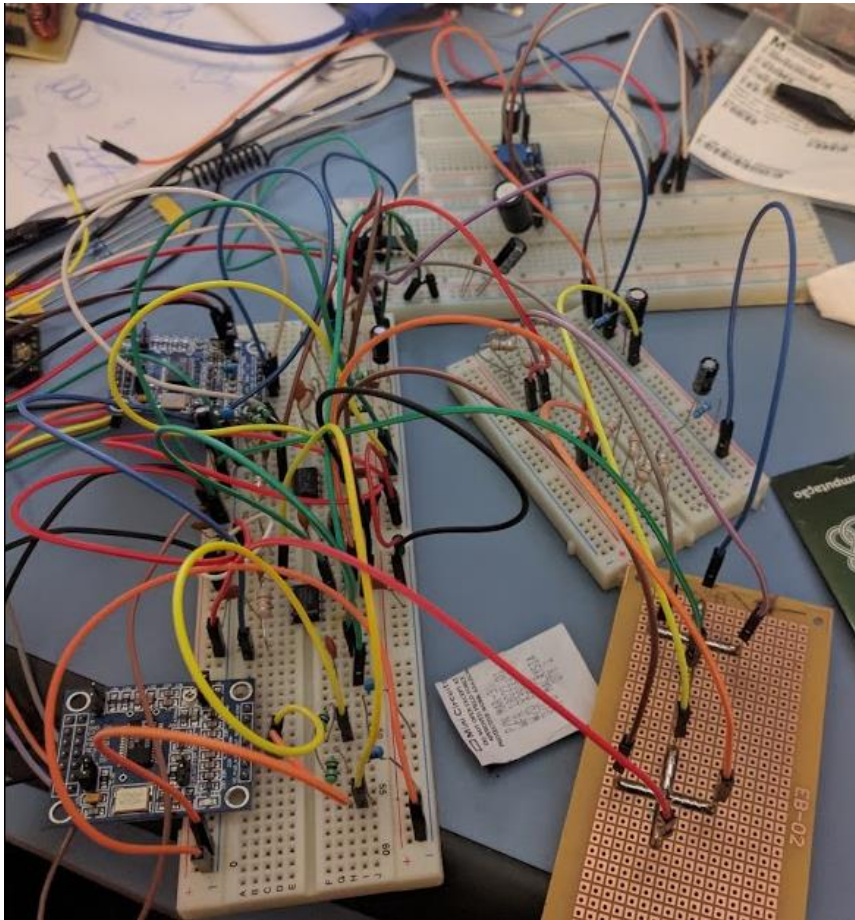


Figura 50 – Prueba del circuito en protoboard

5.2 Pruebas del circuito final

Se hizo una prueba del circuito final con el software y se encontró que la señal recibida contenía un alto piso de ruido. Inspeccionando se encontró que la señal y el ruido superaban los valores de IP3 y compresión de 1dB de los amplificadores y mezcladores, generando ruidos e intermodulaciones. Para intentar solucionarlo se deshabilitaron los dos amplificadores de la etapa de IF y uno de la etapa de RF, de forma que la señal se mantuviera en valores seguros y no se produjera distorsión.

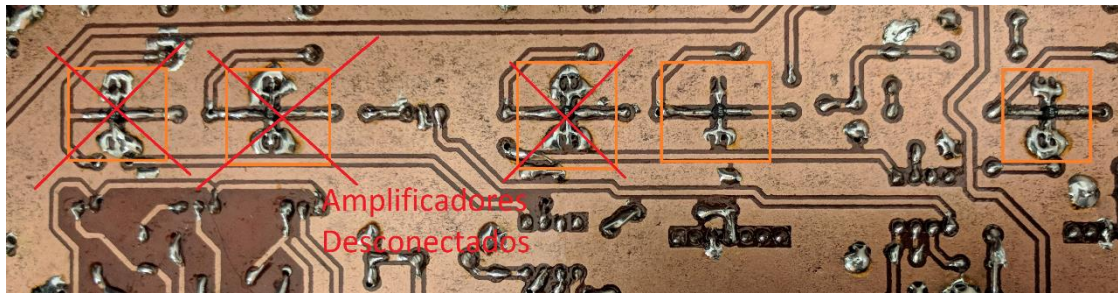


Figura 51 – Amplificadores desconectados

Una vez que este ruido disminuyó notablemente se procedió a probar la recepción de la señal.

Se utilizó una radio definida por software comercial con capacidad de transmisión en ese rango de frecuencias para realizar las pruebas de recepción con el receptor desarrollado en este proyecto integrador. El transmisor se utilizó con GNU Radio en una PC distinta a la del receptor.

5.2.1 Prueba de modulación analógica

Se probó modulación analógica a través de modulación FM. Se moduló una señal de audio de 48KHz de frecuencia de muestreo en un canal FM de 75KHz de ancho de banda, que fue ubicado con frecuencia central en 16MHz.

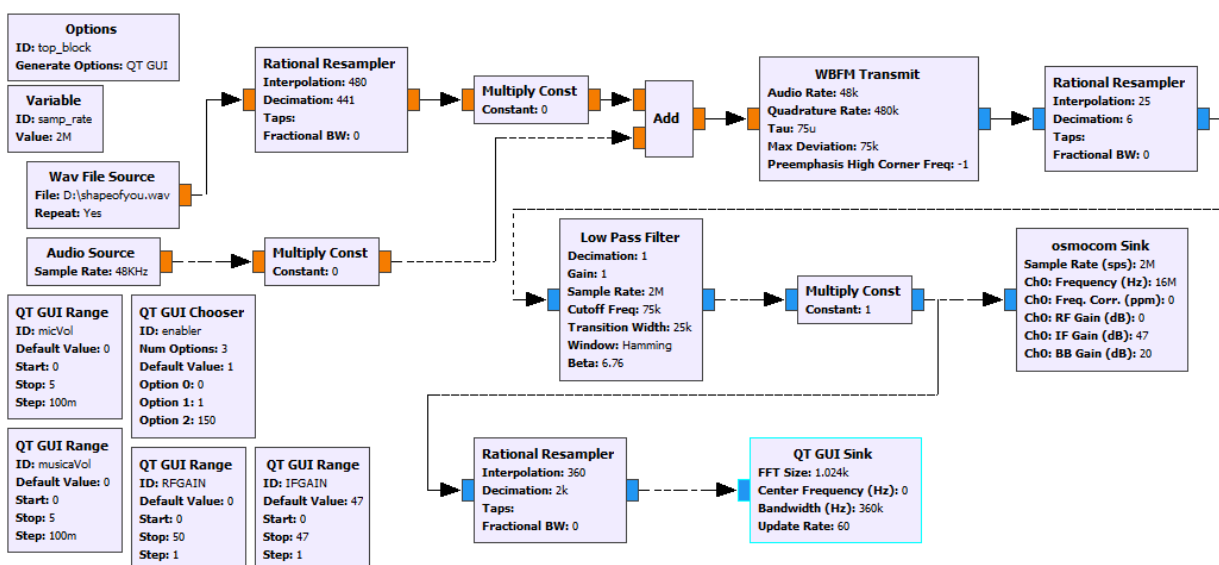


Figura 52 – Diagrama de flujo en GNU Radio Companion para la transmisión en FM

La señal fue recibida y demodulada utilizando las muestras obtenidas por el front end diseñado y otros bloques que procesan dichas muestras.

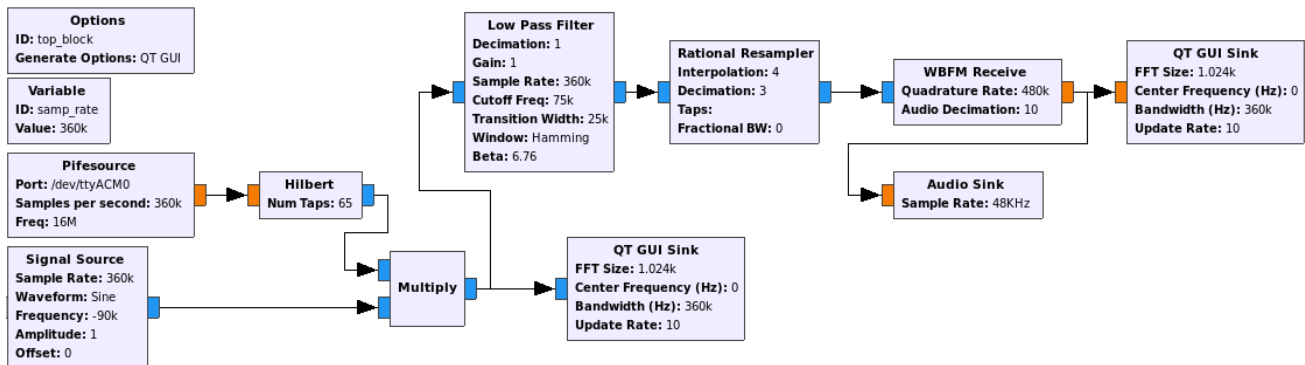


Figura 53 – Diagrama de flujo en GNU Radio Companion para la recepción de FM utilizando el bloque Pifessource

La señal recibida se asemejó a la transmitida sin mostrar distorsión aparente.

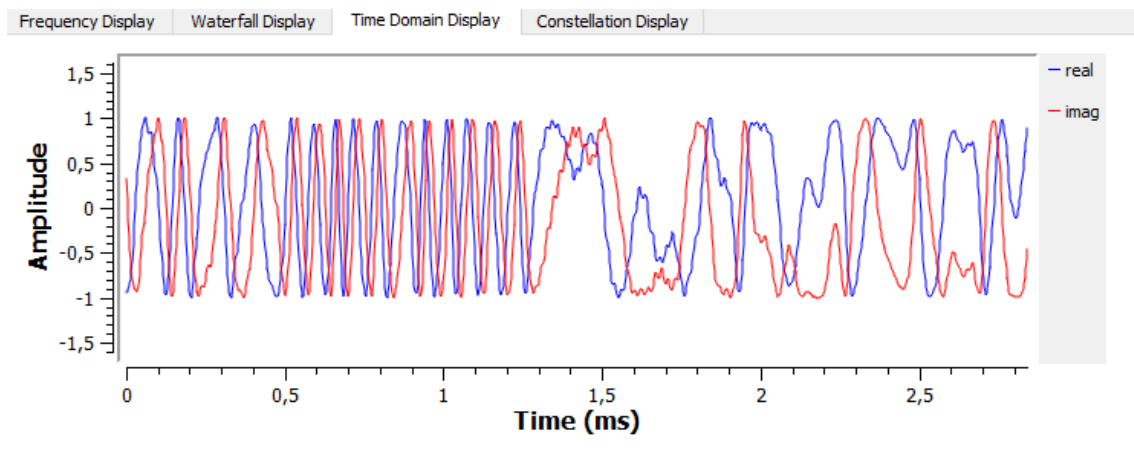


Figura 54 – Gráfico en el dominio del tiempo de un fragmento de la señal FM transmitida

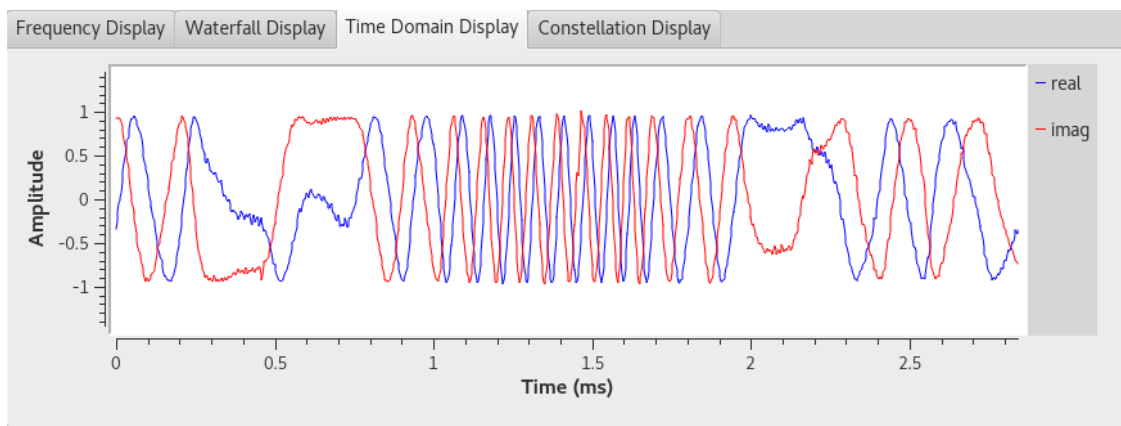


Figura 55 – Gráfico en el dominio del tiempo de un fragmento de la señal FM recibida (No es del mismo instante de la figura anterior)

Se observa que en la señal recibida no se encontró ningún ruido de valor considerable respecto a la señal deseada.

El audio demodulado fue reproducido por los parlantes y se escuchaba sin ruido. Para mejorar su análisis se decidió utilizar bloques que permiten grabar la forma de onda, antes de modularlo y luego de demodularlo. Las grabaciones fueron analizadas en software de edición de audio.

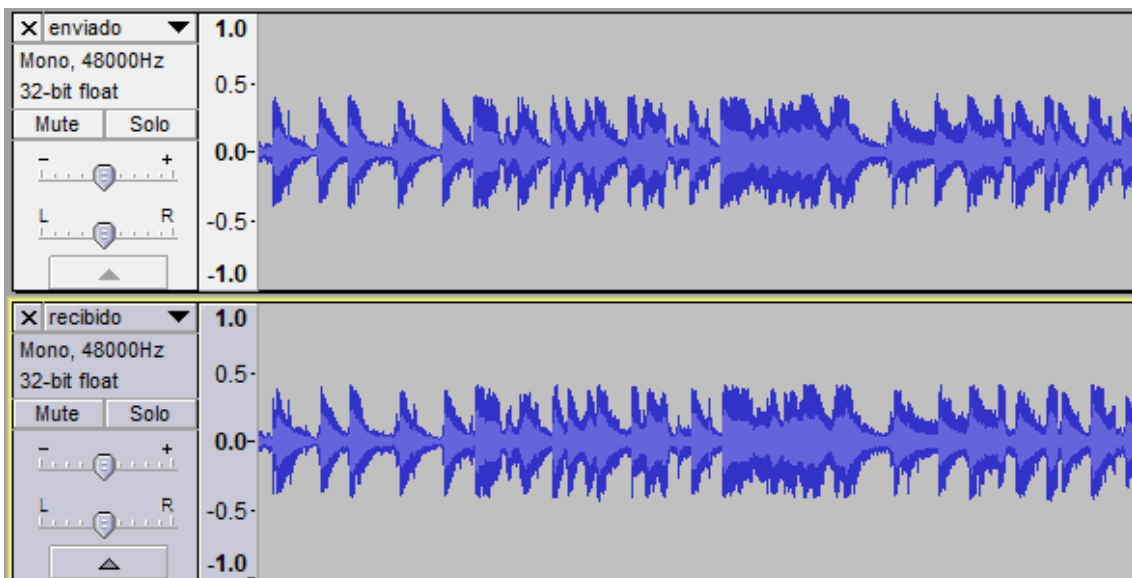


Figura 56 – Formas de onda de música transmitida antes de ser modulada y enviada (arriba) y luego de ser recibida y demodulada (abajo)

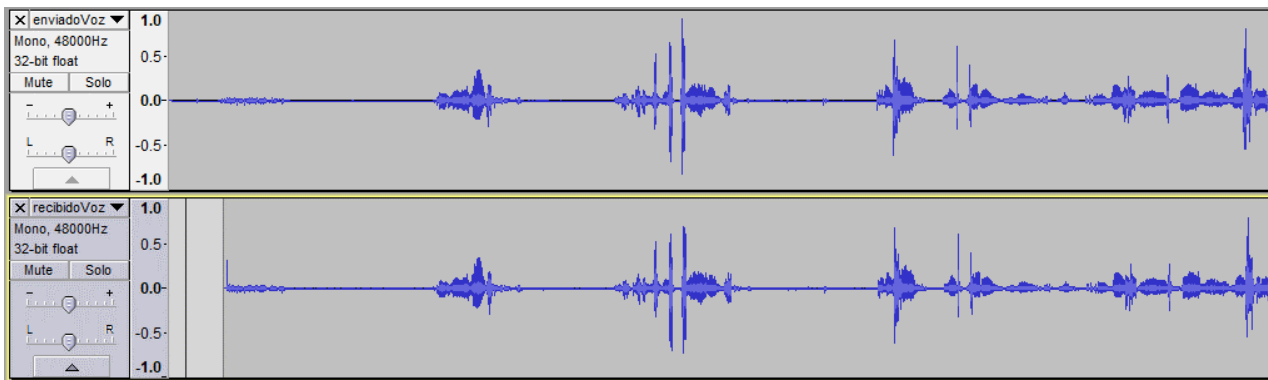


Figura 57 - Formas de onda de audio con voz transmitido antes de ser modulado y enviado (arriba) y luego de ser recibido y demodulado (abajo)

Se observa que el audio enviado y el audio recibido presentaron mínimas diferencias, imperceptibles al oído, tanto cuando se transmite música tomada de un archivo como cuando se transmite voz en vivo de un micrófono.

5.2.2 Prueba de modulación digital

Se probó modulación digital a través de la modulación FSK. Utilizando 4 muestras por símbolo y una ratio de 3200 símbolos por segundo, un total de 12800 muestras por segundo, ocupando un ancho de banda de 6,4KHz ubicados con frecuencia central en 16MHz. La información transmitida era la serie de números [255,0,42,0] en repetición, que en binario es equivalente a [11111111,00000000,00101010,000000].

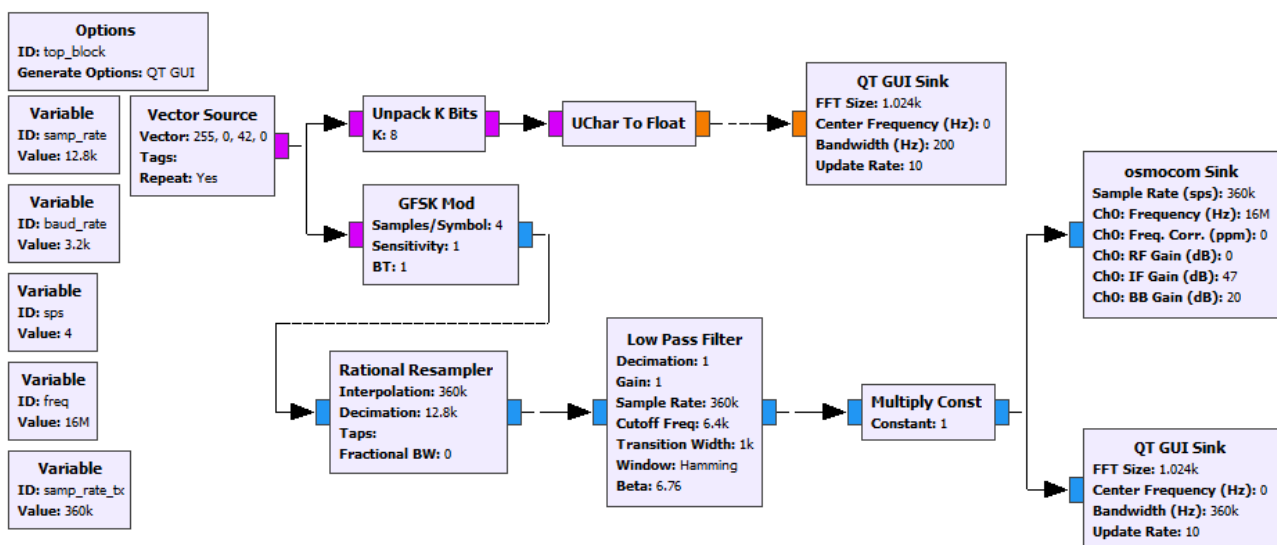


Figura 58 – Diagrama de flujo en GNU Radio Companion para la transmisión de FSK

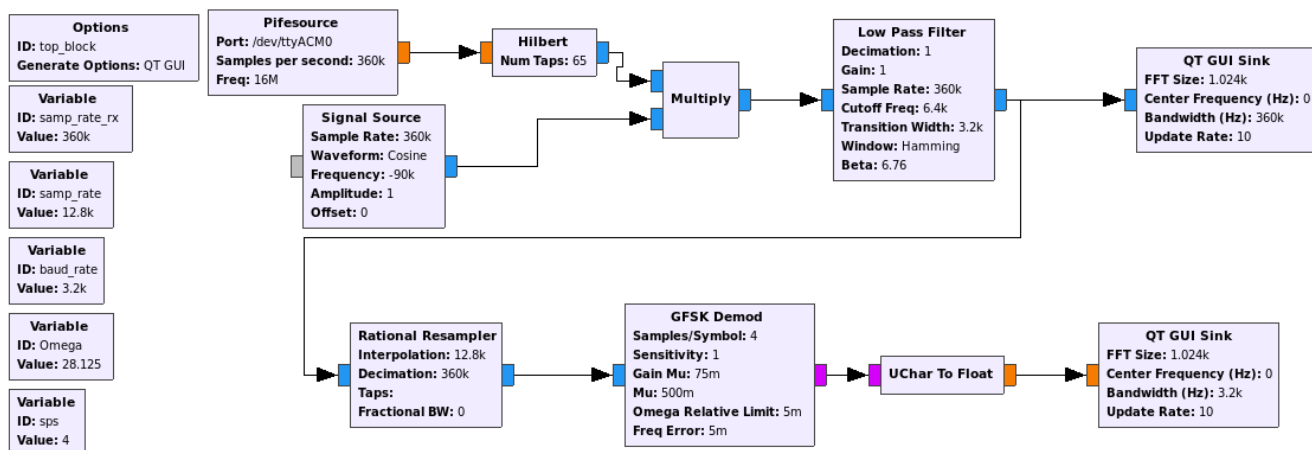


Figura 59 – Diagrama de flujo en GNU Radio Companion para la recepción de FSK utilizando el bloque Pifessource

Al igual que con modulación analógica, la señal recibida se asemejó a la transmitida sin mostrar distorsión aparente.

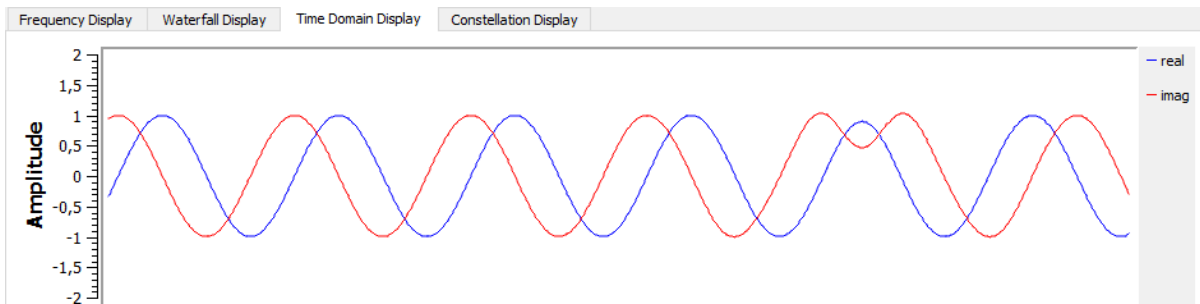


Figura 60 – Gráfico en el dominio del tiempo de un fragmento de la señal FSK transmitida

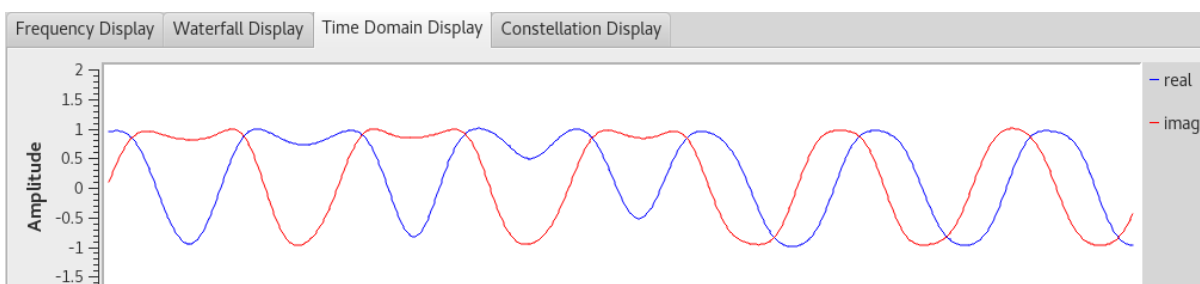


Figura 61 – Gráfico en el dominio del tiempo de un fragmento de la señal FSK recibida luego de un filtro pasa bajo (No es del mismo instante de la figura anterior)

Se observa que la señal recibida en modulaciones digitales tampoco presentó ruido.

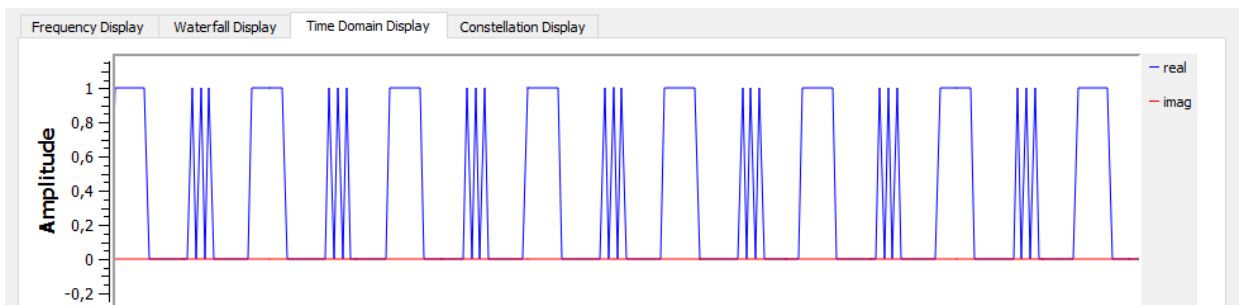


Figura 62 – Bits enviados [11111111,00000000,00101010,00000000]

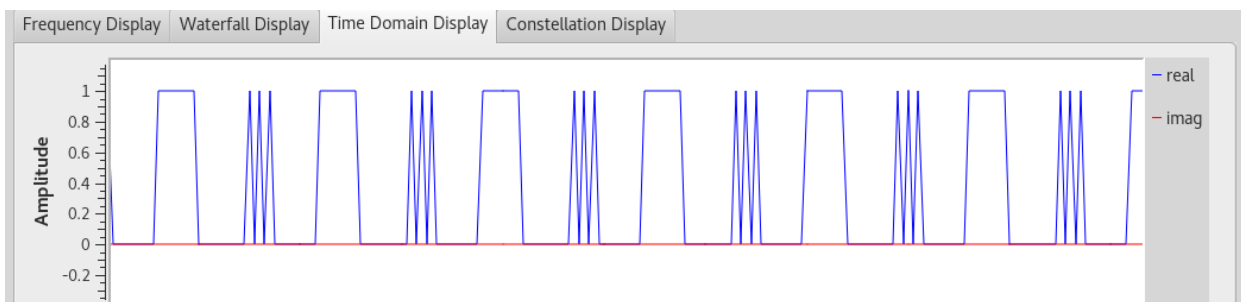


Figura 63 – Bits recibidos [11111111,00000000,00101010,00000000]

No se observa pérdida de datos, los mismos bits enviados fueron los recibidos. Aunque al no haber ningún mecanismo de sincronización el receptor no supo cómo armar los bytes.

5.2.3 Resultados

Se pueden ver como resultados de estas pruebas que el sistema funciona correctamente. El receptor es capaz de sintonizar la frecuencia deseada y enviar las muestras a la PC, y estas se pueden procesar mediante software para demodular la información. Se observa también que, si bien no se cuenta con instrumental específico para medir, el receptor no produce demasiado ruido ni distorsión.

También se observa que la desconexión de los amplificadores solucionó el problema de ruido, por lo que en una futura revisión del circuito deberán ser eliminados.

5.2.4 Prueba de sensibilidad

La sensibilidad del receptor se puede especificar de varios métodos, y su uso depende de la aplicación. Todos los métodos tienen en cuenta que el factor limitante no es el nivel de amplificación del receptor, sino el nivel de ruido que está presente, ya sea propio del receptor o externo.

Debido a que no se tiene el instrumental para medir la figura de ruido, el método que se utilizó para probar la sensibilidad en este receptor es el de la señal mínima discernible. Esta es la señal más pequeña que puede ser detectada por un receptor, procesada y demodulada proveyendo información utilizable en la salida.

Se utilizó un generador de un analizador de espectro que realiza un barrido con una potencia de salida de -50dBm conectado a la entrada del receptor. La caída del cable se midió en aproximadamente -2dBm.

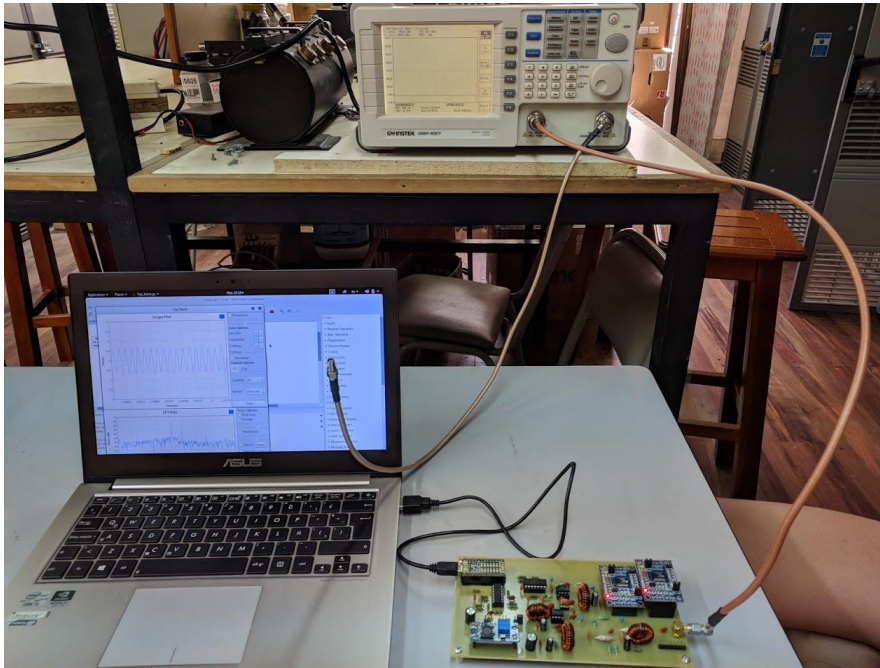


Figura 64 – Configuración para prueba de sensibilidad

Se corroboró que la señal digitalizada que se recibía en la PC muestre el barrido y este se encuentre sobre el nivel de ruido. Luego se comenzaron a añadir atenuadores en la entrada del receptor. Hasta que con una potencia total de -108dBm la señal era apenas discernible.

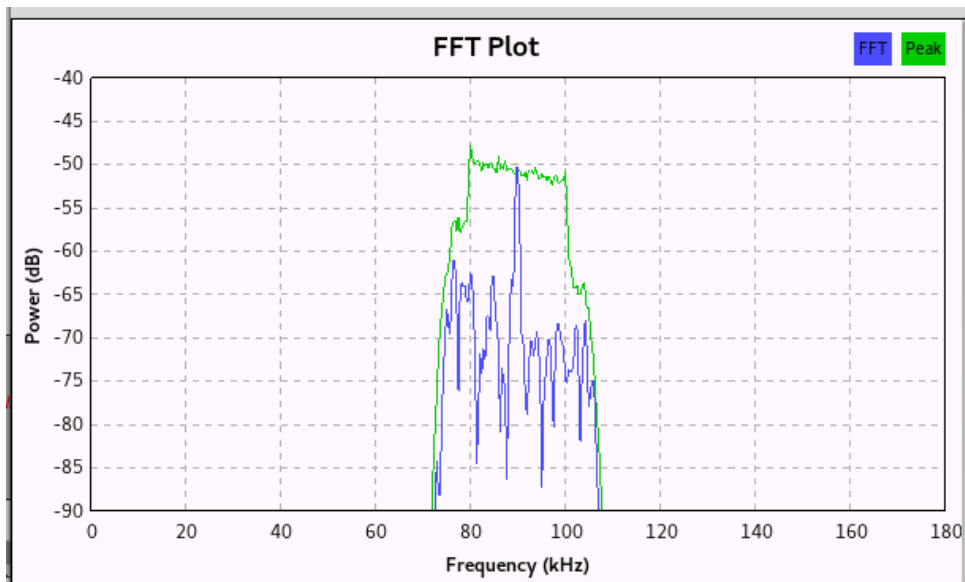


Figura 65 – Recepción del barrido con -108dBm de entrada

Con esta potencia de entrada se encontraban aproximadamente 5dB de diferencia entre la señal y el ruido.

Esto permite discernir señales tan bajas como 1,25uVp. Aunque por el método de prueba se obtiene un valor mejor al real, y este valor de sensibilidad no es el mismo para todos los tipos de modulación, se brinda un valor aproximado del comportamiento esperado.

A comparación, un receptor comercial de AM tiene una sensibilidad típica de 50uVp.¹⁸

¹⁸ Sensibilidad típica de receptor AM según:
http://www2.el.vgtu.lt/~sarunas/english/students/rysiu_tech/am_reception.pdf [Última visita: Noviembre 2017]

CAPÍTULO 6

Instrucciones de uso

En este Capítulo se describe el procedimiento para utilizar este receptor correctamente.

6.1 Consideraciones Preliminares

Este dispositivo se alimenta exclusivamente por USB, por lo que se requiere una computadora con un puerto USB libre para utilizarse. El dispositivo puede utilizarse con los sistemas operativos Windows, Mac OS y Linux, pero el bloque de GNU Radio sólo fue desarrollado para el sistema operativo Linux. Para utilizar el receptor con GNU Radio se deberá instalar por única vez el módulo y, en caso de ser necesario, las herramientas que permiten compilar el módulo.

6.1.1 Reglas UDEV

En el sistema operativo Linux, hay que instalar las reglas UDEV, copiando un archivo con extensión “.rules” en la ubicación de estas reglas, por lo general: “/etc/udev/rules.d”.

Con estas reglas se cambian los permisos para que a nivel usuario se pueda enviar y recibir con el dispositivo y el nombre que se le otorga al dispositivo. Si la comunicación se configuró en modo serial, el dispositivo aparecerá como “/dev/ttyACM#”, siendo # un número comenzando por 0 y dependiendo de cuántos dispositivos utilizando USB serial

haya conectados. El fabricante de la placa con el microcontrolador provee el archivo “49-tenesy.rules”, que se encuentra también en el DVD que se adjunta con este informe.

Una vez que el archivo con las reglas se encuentra en la ubicación correcta ya se puede comunicar con el dispositivo como si fuera un puerto serie, solo hay que utilizar el puerto llamado: “/dev/ttyACM0” (cambiando el cero por el número correcto en caso que haya más dispositivos).

6.2 Configuración de la placa

Para el uso correcto de la placa, antes de conectarla por USB a la computadora, es recomendable asegurarse que todos los componentes modulares (Convertor DC-DC, Osciladores y Microcontrolador) se encuentren correctamente conectados en los pines correspondientes.

Los jumpers que configuran la amplificación en la etapa de banda base deben posicionarse en la amplificación deseada. Su posición puede ser modificada mientras se utiliza.

Posición Jumpers	Salida
○ ○ ○ ○ ○ ○ ○ ○	Sin salida
■ ○ ○ ○ ○ ○ ○ ○	Salida normal
○ ■ ■ ■ ○ ○ ○ ○	Salida amplificada x2
○ ■ ■ ○ ■ ■ ■ ○ ○	Salida amplificada x4

Figura – Configuración de jumpers para ajustar amplificación

6.3 Utilización con GNU Radio

Para utilizar el dispositivo con GNU Radio es necesario instalar el módulo que contiene el bloque desarrollado para comunicarse con la placa. Para compilarlo e instalarlo se requieren los siguientes paquetes de Linux:

- gnuradio
- swig
- cmake

6.3.1 Instalación de paquetes necesarios

Estos paquetes pueden ser instalados en la mayoría de distribuciones de Linux con los siguientes comandos en la consola, siempre que haya conexión a internet:

“sudo apt-get install gnuradio”

“sudo apt-get install swig”

“sudo apt-get install cmake”

Y se debe responder con Y (yes) o S (sí) en caso que pregunte si desea continuar.

```
ubuntu@ubuntu: ~  
ubuntu@ubuntu:~$ sudo apt-get install gnuradio  
  
ubuntu@ubuntu: ~  
libgnuradio-pager3.7.9 libgnuradio-pmt3.7.9 libgnuradio-qtgui3.7.9  
libgnuradio-runtime3.7.9 libgnuradio-trellis3.7.9 libgnuradio-uhd3.7.9  
libgnuradio-video-sdl3.7.9 libgnuradio-vocoder3.7.9 libgnuradio-wavelet3.7.9  
libgnuradio-wxgui3.7.9 libgnuradio-zeromq3.7.9 libgps22 libgsm1 librtlsdr0  
libuhd003 libvolk1-bin libvolk1-dev libvolk1.1 rtl-sdr uhd-host  
0 upgraded, 36 newly installed, 0 to remove and 5 not upgraded.  
Need to get 16.4 MB of archives.  
After this operation, 137 MB of additional disk space will be used.  
Do you want to continue? [Y/n] Y  
  
ubuntu@ubuntu: ~  
Setting up libvolk1-dev (1.2.1-1) ...  
Setting up gnuradio-dev (3.7.9.1-2ubuntu1) ...  
Setting up uhd-host (3.9.2-1) ...  
Adding group `usrp' (GID 128) ...  
Done.  
Setting up librtlsdr0:amd64 (0.5.3-5) ...  
Setting up rtl-sdr (0.5.3-5) ...  
Processing triggers for libc-bin (2.23-0ubuntu5) ...  
ubuntu@ubuntu:~$ sudo apt-get install swig  
  
ubuntu@ubuntu: ~  
swig-doc swig-examples swig3.0-examples swig3.0-doc  
The following packages will be REMOVED:  
swig2.0  
The following NEW packages will be installed:  
swig swig3.0  
0 upgraded, 2 newly installed, 1 to remove and 5 not upgraded.  
Need to get 1,001 kB of archives.  
After this operation, 1,035 kB of additional disk space will be used.  
Do you want to continue? [Y/n] Y  
  
ubuntu@ubuntu: ~  
swig-doc swig-examples swig3.0-examples swig3.0-doc  
The following packages will be REMOVED:  
swig2.0  
The following NEW packages will be installed:  
swig swig3.0  
0 upgraded, 2 newly installed, 1 to remove and 5 not upgraded.  
Need to get 1,001 kB of archives.  
After this operation, 1,035 kB of additional disk space will be used.  
Do you want to continue? [Y/n] Y  
  
ubuntu@ubuntu: ~  
Setting up swig3.0 (3.0.8-0ubuntu3) ...  
Setting up swig (3.0.8-0ubuntu3) ...  
ubuntu@ubuntu:~$ sudo apt-get install cmake  
Reading package lists... Done  
Building dependency tree  
Reading state information... Done  
cmake is already the newest version (3.5.1-1ubuntu3).  
0 upgraded, 0 newly installed, 0 to remove and 5 not upgraded.  
ubuntu@ubuntu:~$
```

Figura 66 – Instalación de paquetes necesarios

Estos comandos no sólo instalan los paquetes en caso que no se encuentren ya instalados, sino que también los actualizan a su última versión. Si alguno ya se encuentra en su última versión no realiza nada.

6.3.2 Creación de módulo de GNU Radio

Los módulos se crean utilizando la herramienta `gr_modtool` incluida en GNU Radio.

Para crear un módulo llamado PIFE se realiza el siguiente comando en una terminal:

```
“gr_modtool newmod PIFE”
```

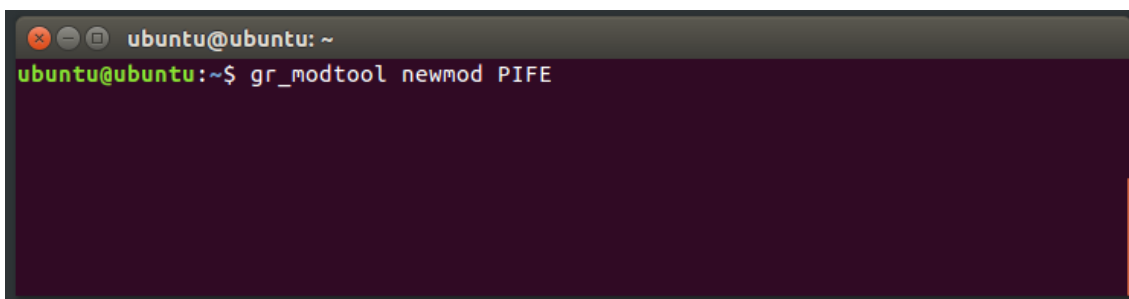


Figura 67 – Creación de módulo PIFE

La herramienta `gr_modtool` creará una carpeta llamada “gr-PIFE” con varias carpetas dentro ya configuradas, incluyendo carpetas donde se incluirá el código fuente de los bloques, que pueden ser escritos en C++ o en Python.

6.3.3 Creación del bloque de GNU Radio

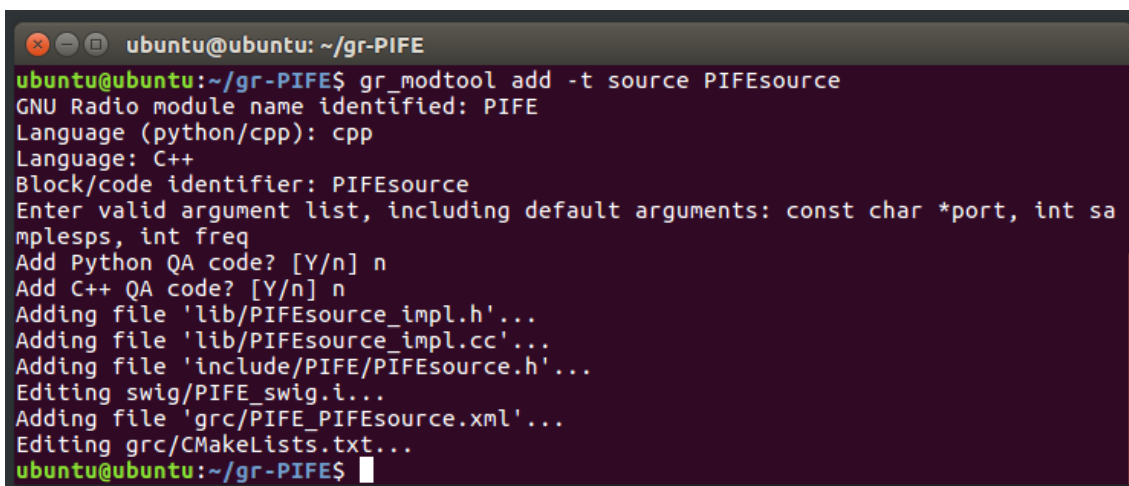
Para crear un bloque se debe utilizar, dentro de la carpeta del módulo:

```
“gr_modtool add -t source PIFESource”
```

Cuando pregunte el lenguaje del bloque contestar con `cpp`, en la lista de argumentos escribir:

```
“const char *port, int samplesps, int freq”
```

y contestar con `n` (no) para añadir código de prueba.



```
ubuntu@ubuntu: ~/gr-PIFE
ubuntu@ubuntu:~/gr-PIFE$ gr_modtool add -t source PIFESource
GNU Radio module name identified: PIFE
Language (python/cpp): cpp
Language: C++
Block/code identifier: PIFESource
Enter valid argument list, including default arguments: const char *port, int samples, int freq
Add Python QA code? [Y/n] n
Add C++ QA code? [Y/n] n
Adding file 'lib/PIFESource_impl.h'...
Adding file 'lib/PIFESource_impl.cc'...
Adding file 'include/PIFE/PIFESource.h'...
Editing swig/PIFE_swig.i...
Adding file 'grc/PIFE_PIFESource.xml'...
Editing grc/CMakeLists.txt...
ubuntu@ubuntu:~/gr-PIFE$
```

Figura 68 – Creación de bloque PIFESource

gr_modtool creará archivos del bloque y solo se necesitará editar el código fuente del bloque en la carpeta “lib”, donde habrá un archivo de header “.h” y un archivo de código fuente de “.cc”

Reemplazar los archivos “PIFESource_impl.cc” y “PIFESource_impl.h” dentro de la carpeta “lib” con los que se encuentran en el DVD adjunto. Estos archivos tienen el código fuente del bloque que se conecta con el receptor.

Luego para añadir el bloque a GNU Radio Companion, la interfaz gráfica de GNU Radio, se debe crear un archivo “.xml” con el nombre del bloque en la carpeta “grc” del módulo, este archivo debe indicar qué tipo de valores pedir para los argumentos del bloque y que tipo de bloque es.

Copiar el archivo “PIFE_PIFESource.xml” que se encuentra en el DVD adjunto dentro de la carpeta “grc”.

6.3.4 Compilación e Instalación del bloque

Una vez que el bloque fue creado, este se debe compilar e instalar. Para eso se requiere “CMake” y “swig”.

Se crea una carpeta “build” dentro de la carpeta del módulo y se indica que las instrucciones se encuentran una carpeta más arriba con la instrucción “cmake ..”, después se compila e instala con “sudo make install”.


```

ubuntu@ubuntu: ~/gr-PIFE/build
ubuntu@ubuntu:~/gr-PIFE$ mkdir build
ubuntu@ubuntu:~/gr-PIFE$ cd build
ubuntu@ubuntu:~/gr-PIFE/build$ cmake ..

ubuntu@ubuntu: ~/gr-PIFE/build
ubuntu@ubuntu:~/gr-PIFE/build$ sudo make install
Scanning dependencies of target gnuradio-PIFE
[ 4%] Building CXX object lib/CMakeFiles/gnuradio-PIFE.dir/PIFESource_impl.cc.o
[ 9%] Linking CXX shared library libgnuradio-PIFE-1.0.0git.so
[ 9%] Built target gnuradio-PIFE
Scanning dependencies of target test-PIFE
[ 14%] Building CXX object lib/CMakeFiles/test-PIFE.dir/test_PIFE.cc.o
[ 19%] Building CXX object lib/CMakeFiles/test-PIFE.dir/qa_PIFE.cc.o
[ 23%] Linking CXX executable test-PIFE
[ 23%] Built target test-PIFE
Scanning dependencies of target _PIFE_swig_doc_tag
[ 28%] Building CXX object swig/CMakeFiles/_PIFE_swig_doc_tag.dir/_PIFE_swig_doc_tag.cpp.o
[ 33%] Linking CXX executable _PIFE_swig_doc_tag
[ 33%] Built target _PIFE_swig_doc_tag
Scanning dependencies of target PIFE_swig_swig_doc

```

Figura 69 – Compilación e instalación del módulo

Una vez finalizada la instalación el bloque debería aparecer en GNU Radio Companion dentro del módulo PIFE.

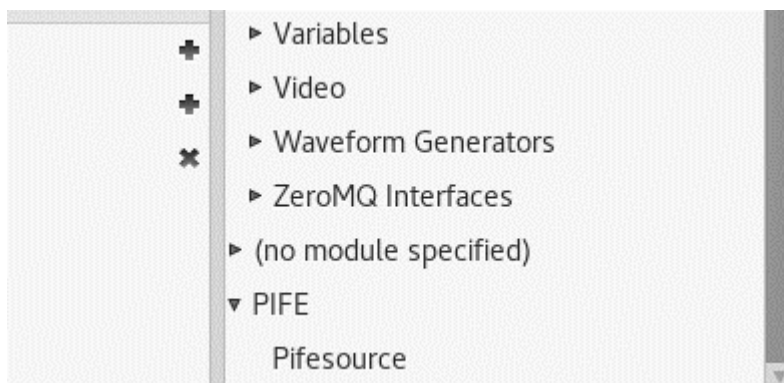


Figura 70 – Módulo y bloque instalados dentro de GNU Radio Companion

Este bloque toma como argumento para instanciarse el nombre del puerto, la frecuencia de sintonización y la velocidad de muestreo.

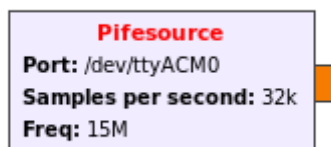


Figura 71 – Bloque PIFESource en GNU Radio Companion

El bloque pasa las muestras en su salida como float. Si se requieren en formato complejo para utilizar con otros bloques se puede agregar un bloque Hilbert.

6.4 Utilización fuera de GNU Radio

El receptor puede utilizarse fuera de GNU Radio. Una vez conectado por USB, se puede acceder como puerto COM virtual en Windows, Linux o Mac OS.

Para desarrollar un programa que utilice las muestras hay que recordar que se le debe enviar las instrucciones para sintonizar y cambiar frecuencia de muestreo. Se recomienda utilizar bibliotecas para uso de puertos serie para facilitar el proceso.

Por ejemplo, enviando “@416150000” sintoniza la frecuencia 16,15MHz, y enviando “@300360000” se comienza a muestrear con 360.000 muestras por segundo. Se debe tener en cuenta que estas instrucciones responden con un texto de confirmación que no debe ser interpretado como muestras.

Una vez que se comienza a muestrear, cada byte corresponderá a una muestra.

CAPÍTULO 7

Conclusiones

7.1 Resultados obtenidos en base a los objetivos propuestos

Estableciendo una comparación entre las características del dispositivo desarrollado y los objetivos propuestos al inicio se elaboran las siguientes conclusiones:

- Se pudo desarrollar un receptor de radio con un rango de trabajo (10MHz a 30MHz) que se encuentra en la banda de alta frecuencia y que es capaz de sintonizar un canal de 180kHz en cualquier punto de ese rango de trabajo, digitalizarlo y enviar las muestras a la PC mediante una conexión USB en tiempo real. Las muestras recibidas en la PC pueden ser procesadas y se ha comprobado que a través de éstas se puede recuperar la información embebida con modulaciones analógicas o digitales.
- El desarrollo del prototipo se realizó luego de realizar una investigación acerca de las distintas alternativas para cada componente del sistema. La selección de cada una se basó en los criterios de diseño de consumo menor al máximo permitido por USB y la facilidad para su montaje y prueba de circuitos.
- El funcionamiento del prototipo se verificó con pruebas realizadas que involucran todos los bloques de desarrollo de este proyecto y su resultado fue satisfactorio.
- En la arquitectura del sistema desarrollado un microcontrolador hace de nexo para recibir instrucciones y configurar los componentes de bajo nivel mientras mantiene el ritmo de muestreo. Esto no puede ser logrado por un procesador con

un sistema operativo de alto nivel porque sus interrupciones tendrían una latencia mucho mayor.

- Los conocimientos adquiridos durante el cursado de la carrera han permitido no sólo la toma de decisiones durante el diseño y la implementación del mismo, sino que también han aportado a la detección de problemas y a la resolución de dichos obstáculos. Se lograron identificar las posibles causas y mediante pruebas confirmar estas suposiciones y corregir el diseño.
- En las situaciones en las que los conocimientos adquiridos durante el cursado de la carrera no fueron suficientes, estos de igual manera han servido como base para la adquisición de otros nuevos que permitieron la continuación del proyecto.
- Al haber trabajado con componentes de radiofrecuencia en forma modular y separada se ha podido experimentar y familiarizarse con el funcionamiento de cada uno de ellos.
- Al haber trabajado de forma modular los componentes se pudo lograr una implementación de la arquitectura general del sistema en bloques, lo que permite la modificación y/o mejora de cada uno de estos por separado.
- Un objetivo personal del autor mencionado al inicio del proyecto consistía en expandir su conocimiento con la programación en el sistema operativo Linux. Aunque el desarrollo no abarca demasiadas formas de alcanzarlo, se ha logrado conseguir un entendimiento mucho más amplio de este sistema operativo luego y se ha logrado programar, compilar y utilizar programas en este sistema operativo, del cual el autor no poseía experiencia previa.

7.2 Conclusión General

En una época donde los componentes DIP o de inserción están comenzando a desaparecer para dar lugar a componentes de montaje superficial, muchos de los cuales no pueden ser soldados sin herramientas y hornos caros y especializados, se destaca la importancia de estos primeros para el armado de prototipos y la prueba de los circuitos. Aunque estos elementos no sean adecuados para un producto final, la facilidad con la que permiten diseñar y verificar circuitos, que sin duda fue esencial para el desarrollo de este

proyecto, es una característica que no puede desaparecer. Por suerte, esta facilidad está reapareciendo en placas de desarrollo que permiten utilizar los componentes difíciles de montar sin esa dificultad.

Los microcontroladores son cada vez más capaces y se vuelven cada vez más baratos, esto sumado a que una gran cantidad de fabricantes busca facilitar su programación lo que permite que se integren en cualquier sistema en poco tiempo, sin requerir capacitación especializada. Y esto no sólo incluye el campo de las comunicaciones, la electrónica digital está incrementando su presencia para resolver todo tipo de problemas de formas más eficientes y simplificando su modificación, mejora o actualización.

7.3 Mejoras a futuro

Como se mencionó previamente, la implementación modular del sistema permite su modificación y este trabajo como todo diseño de ingeniería está sujeto a posibles mejoras. En la siguiente sección se mencionan mejoras a este proyecto integrador que pueden ser aplicadas por un equipo de trabajo interesado.

- El desarrollo se enfocó en la recepción de señales de radio y su digitalización. Un posible trabajo a futuro consistiría en realizar el proceso inverso y a partir de muestras digitales, obtener una señal analógica que luego es amplificada para transmitir. Constituyendo el transmisor equivalente al receptor desarrollado.
- El microcontrolador utilizado permite capturar y enviar apenas más de 360kSps, si se escogiera un microcontrolador más potente o si se lo acompañara de algún hardware dedicado (FPGA), se podría aumentar considerablemente el ancho de banda del canal digitalizado.
- Los sintetizadores digitales elegidos no superan los 50 MHz, si se modifica por lo menos uno de ellos por uno de mayor frecuencia se puede utilizar un rango de trabajo distinto, en VHF o UHF.
- El control de ganancia en la banda base se realiza mediante la posición de jumpers que saltan o no amplificadores operacionales. Esto se podría cambiar por algún sistema de resistencias variables, o utilizando un ADC con ganancia ajustable, permitiendo ajustar la ganancia mediante software.

- La ganancia en RF e IF en el trabajo desarrollado se encuentra fija y no se puede modificar sin cambiar los componentes, se podría desarrollar un sistema de ganancia ajustable por software que el microcontrolador se encargue de modificar.
- Se podría realizar un receptor de radio definido por software utilizando una arquitectura que se asemeje lo más posible a una radio de software ideal. Utilizando un ADC de mayor velocidad, no realizando mezcla en el front end y buscando una solución de mayor velocidad de transferencia para conectar con la PC.

Bibliografía y Referencias

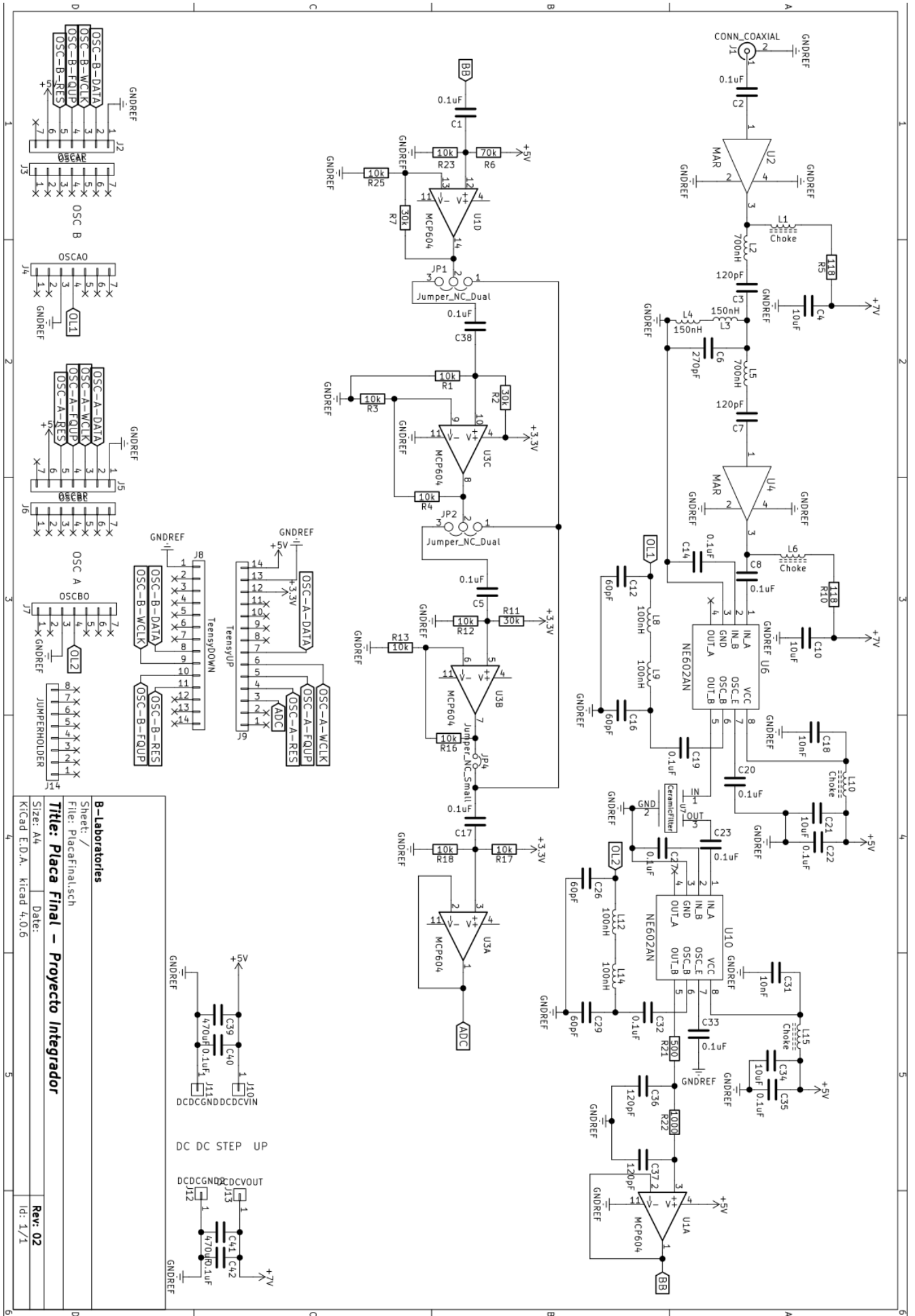
- [1] Liendo C. G. Material de Estudio – Sistemas de radiocomunicaciones
- [2] Dixon, R (1998) - Radio Receiver Design - CRC Press
- [3] Danizio, P. E. (2010). Teoría de las Comunicaciones – Editorial Universitas, Córdoba.
- <https://en.wikipedia.org/wiki/Modulation> [Última visita: Octubre 2017]
 - <http://www.analfatecnicos.net/pregunta.php?id=15> [Última visita: Octubre 2017]
 - https://en.wikipedia.org/wiki/Software-defined_radio [Última visita: Octubre 2017]
 - https://en.wikipedia.org/wiki/Analog-to-digital_converter [Última visita: Octubre 2017]
 - https://en.wikipedia.org/wiki/Flash_ADC [Última visita: Octubre 2017]
 - https://en.wikipedia.org/wiki/Radio_receiver [Última visita: Octubre 2017]
 - https://en.wikipedia.org/wiki/Superheterodyne_receiver [Última visita: Octubre 2017]
 - https://en.wikipedia.org/wiki/Frequency_mixer [Última visita: Octubre 2017]
 - [https://en.wikipedia.org/wiki/Antenna_\(radio\)](https://en.wikipedia.org/wiki/Antenna_(radio)) [Última visita: Octubre 2017]
 - <https://en.wikipedia.org/wiki/Amplifier> [Última visita: Octubre 2017]
 - <http://www.mwrf.com/components/ensure-stability-amplifier-designs> [Última visita: Octubre 2017]
 - https://en.wikipedia.org/wiki/Noise_figure [Última visita: Octubre 2017]
 - <https://www.digikey.com/en/articles/techzone/2011/oct/the-basics-of-mixers> [Última visita: Octubre 2017]
 - <http://www.ittc.ku.edu/~jstiles/622/handouts/Mixer%20Compression%20and%200Intercept%20Points.pdf> [Última visita: Octubre 2017]
 - https://en.wikipedia.org/wiki/Direct_digital_synthesizer [Última visita: Octubre 2017]
 - https://en.wikipedia.org/wiki/Phase-locked_loop [Última visita: Octubre 2017]
 - https://en.wikipedia.org/wiki/Electronic_filter [Última visita: Octubre 2017]

- https://en.wikipedia.org/wiki/Crystal_filter [Última visita: Octubre 2017]
- <https://en.wikipedia.org/wiki/Demodulation> [Última visita: Octubre 2017]
- <https://en.wikipedia.org/wiki/Microcontroller> [Última visita: Octubre 2017]
- <https://en.wikipedia.org/wiki/USB> [Última visita: Octubre 2017]
- <http://www.i-micro.com/pdf/articulos/usb.pdf> [Última visita: Octubre 2017]
- <https://www.britannica.com/science/radio-frequency-spectrum> [Última Visita: Noviembre 2017]
- <http://urgentcomm.com/mag/uwb-brings-radios-future-closer> [Última visita: Noviembre 2017]
- <http://www.radio-electronics.com/info/rf-technology-design/rf-noise-sensitivity/receiver-sensitivity-performance-tutorial.php> [Última visita: Noviembre 2017]

Anexo 1: Circuito final implementado

DISEÑO E IMPLEMENTACIÓN DE FRONT END ANALÓGICO PARA SDR

BANCHIO, Agustín Enrique



Anexo 2: Código de microcontrolador

```
#include <ADC.h>

IntervalTimer timerMuestreo;
ADC *adc = new ADC();

#define ADPin 15

#define OSCB_D 19
#define OSCB_C 18
#define OSCB_F 17
#define OSCB_R 16

#define OSCA_D 6
#define OSCA_C 7
#define OSCA_F 8
#define OSCA_R 9

//El AD9850 tiene un clock de 125MHz
#define AD9850_CLOCK 125000000

//Funcion para hacer un pulso rápido
#define pulseHigh(pin) {digitalWrite(pin, HIGH); digitalWrite(pin, LOW); }

int velMuestreo;          //Variable global de velocidad de muestreo

void setup() {
  Serial.begin(9600);      //El baud rate se ignora

  pinMode(ADPin, INPUT);  //Pin de ADC como entrada
  pinMode(OSCA_D, OUTPUT); //Pines de configuracion de osciladores como salida
  pinMode(OSCA_C, OUTPUT);
  pinMode(OSCA_F, OUTPUT);
  pinMode(OSCA_R, OUTPUT);

  pinMode(OSCB_D, OUTPUT);
  pinMode(OSCB_C, OUTPUT);
  pinMode(OSCB_F, OUTPUT);
  pinMode(OSCB_R, OUTPUT);

  pulseHigh(OSCA_R);      //Configuro entrada de osciladores como serial mediante pulsos
  pulseHigh(OSCA_C);
  pulseHigh(OSCA_F);

  pulseHigh(OSCB_R);
  pulseHigh(OSCB_C);
  pulseHigh(OSCB_F);

  timerMuestreo.priority(5); //Configuro prioridad de interrupcion

  adc->setAveraging(0);     //Configuro ADC (resolucion y velocidad)
  adc->setResolution(8);
  adc->setSamplingSpeed(ADC_SAMPLING_SPEED::VERY_HIGH_SPEED);
  adc->setConversionSpeed(ADC_CONVERSION_SPEED::VERY_HIGH_SPEED);
  adc->startContinuous(15, ADC_0);
}

//Función para setear frecuencia de osciladores
```

```
void setFrequency(double frequency, int osc)
{
//Calculo número a enviar:
int32_t freq = frequency * 4294967295/AD9850_CLOCK;
//Transfiero de a 1 byte:
for (int i=0; i<4; i++)
{
transferByte(freq & 0xFF, osc);
freq >>= 8;
}
//Transfiero ultimo byte 0x00
transferByte(0x00, osc);
//Actualizo frecuencia de osciladores:
if (osc == 1)
{
pulseHigh(OSCA_F);
}
else if (osc == 2)
{
pulseHigh(OSCB_F);
}
}

//Función para transferir un byte al oscilador correcto
void transferByte(byte data, int osc)
{
for (int i = 0; i<8; i++)
{
if (osc == 1)
{
digitalWrite(OSCA_D,data & 0x01);
pulseHigh(OSCA_C);
}
else if (osc==2)
{
digitalWrite(OSCB_D,data & 0x01);
pulseHigh(OSCB_C);
}
data>>=1;
}
}

//Variables globales para el muestreo: buffers y contadores.
byte buf1[16364], buf2[16364], buf3[16364];
volatile int contadorBuf = 0, buf = 1;
volatile uint8_t value;
bool buf1ready = false, buf2ready = false, buf3ready = false;

//Función de muestreo, se ejecuta en la interrupción
void muestrear()
{
noInterrupts();
//Leo valor del ADC
value = adc->analogReadContinuous(ADC_0);
//Guardo valor en buffer correspondiente
if (buf == 1)
{
buf1[contadorBuf] = value;
}
if (buf == 2)
{
buf2[contadorBuf] = value;
}
if (buf == 3)
{
buf3[contadorBuf] = value;
}
//Aumento contador
contadorBuf++;
}

//Si el contador llega a la cantidad prevista cambia de buffer
```

```
if (contadorBuf >= (velMuestreo/22))
{
  contadorBuf = 0;
  if (buf == 1)
  {
    buf = 2;
    buf1ready = true;
  }
  else if (buf == 2)
  {
    buf = 3;
    buf2ready = true;
  }
  else if (buf == 3)
  {
    buf = 1;
    buf3ready = true;
  }
}
interrupts();
}

//Loop principal
void loop() {
  //Instrucciones que puede recibir:
  //@1FFFFFFFF Setear Frec OSC 1
  //@2FFFFFFFF Setear Frec OSC 2
  //@3MMMMMMMM Setear Muestras por segundo
  //@4FFFFFFFF Setear Frecuencia de sintonizacion
  //@5FFFFFFFF Setear FrecuenciaRapido
  int count = 0;
  int operacion = 0;
  int numero = 0;
  while (count<10)
  {
    //Si algun buffer esta lleno enviarlo
    if (buf1ready && !buf3ready)
    {
      Serial.write(buf1, (velMuestreo/22));
      buf1ready = false;
    }
    if (buf2ready && !buf1ready)
    {
      Serial.write(buf2, (velMuestreo/22));
      buf2ready = false;
    }
    if (buf3ready && !buf2ready)
    {
      Serial.write(buf3, (velMuestreo/22));
      buf3ready = false;
    }

    //Revisa si hay caracteres disponibles para leer
    if (Serial.available())
    {
      char c = Serial.read();
      //Forma instruccion
      if (c == '@')
      {
        count = 1;
        numero = 0;
        operacion = 0;
      }
      else if (count==1)
      {
        switch (c)
        {
          case '1':
            operacion = 1;
            numero = 0;
            break;
          case '2':
            operacion = 2;
            numero = 0;

```

```
        break;
        case '3':
            operacion = 3;
            numero = 0;
            break;
        case '4':
            operacion = 4;
            numero = 0;
            break;
        case '5':
            operacion = 5;
            numero = 0;
            break;
        default:
            operacion = 0;
            count = 0;
            numero = 0;
            break;
    }
    count=2;
}
else if (count>1)
{
    switch (c)
    {
        case '0':
            numero += 0;
            count ++;
            break;
        case '1':
            numero += 1 * pow(10,abs(count-9));
            count ++;
            break;
        case '2':
            numero += 2 * pow(10,abs(count-9));
            count ++;
            break;
        case '3':
            numero += 3 * pow(10,abs(count-9));
            count ++;
            break;
        case '4':
            numero += 4 * pow(10,abs(count-9));
            count ++;
            break;
        case '5':
            numero += 5 * pow(10,abs(count-9));
            count ++;
            break;
        case '6':
            numero += 6 * pow(10,abs(count-9));
            count ++;
            break;
        case '7':
            numero += 7 * pow(10,abs(count-9));
            count ++;
            break;
        case '8':
            numero += 8 * pow(10,abs(count-9));
            count ++;
            break;
        case '9':
            numero += 9 * pow(10,abs(count-9));
            count ++;
            break;
        default:
            operacion = 0;
            count = 0;
            break;
    }
}
}
```

```
//Según la instrucción que operación se ejecuta
if (operacion == 1)
{
    noInterrupts();
    if (numero<=50000000)
    {
        setFrequency(numero,1);
        Serial.print("Oscilador 1 seteado a frecuencia: "); Serial.print(numero);
Serial.println(" Hz");
    }
    else
    {
        Serial.println("Valor inválido" );
    }
    interrupts();
}

if (operacion == 2)
{
    noInterrupts();
    if (numero<=50000000)
    {
        setFrequency(numero,2);
        Serial.print("Oscilador 2 seteado a frecuencia: "); Serial.print(numero);
Serial.println(" Hz");
    }
    else
    {
        Serial.println("Valor inválido" );
    }
    interrupts();
}

if (operacion == 3)
{
    setMuestreo(numero);
}

if (operacion == 4)
{
    if (numero >= 10000000 && numero <= 30000000)
    {
        //Sintonizo frecuencia, centro del canal en 90kHz
        setFrequency(numero+10700000,1);
        setFrequency(10700000+90000,2);
        Serial.print("Sintonizando frecuencia: "); Serial.print(numero); Serial.println("
Hz");
    }
    else
    {
        Serial.println("Valor inválido" );
    }
}

if (operacion == 5)
{
    if (numero >= 10000000 && numero <= 30000000)
    {
        setFrequency(numero+10700000,1);
        setFrequency(10700000+90000,2);
    }
}

operacion = 0;
count = 0;
numero = 0;
}
```

```
//Funcion para setear frecuencia de muestreo
```

```
void setMuestreo(int numero)
{
    //Si es 0, deshabilitar muestreo
    if (numero == 0)
    {
        noInterrupts();
        timerMuestreo.end();
        Serial.println("Muestreo Deshabilitado");
        velMuestreo = 0;
        interrupts();
    }
    else
    {
        if (numero<=360000)
        {
            noInterrupts();
            timerMuestreo.end();
            velMuestreo = numero;
            Serial.print("Muestreando a "); Serial.print(numero); Serial.println(" veces por
segundo");
            timerMuestreo.begin(muestrear, (double)1000000/(double)numero);
            interrupts();
        }
        else
        {
            Serial.println("Valor inválido" );
        }
    }
}
```

Anexo 3: Código de bloque de GNU Radio

```
#ifndef HAVE_CONFIG_H
#include "config.h"
#endif

#include <string>
#include <fcntl.h>
#include <string.h>
#include <termios.h>
#include <unistd.h>
#include <stdio.h>
#include <stdlib.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <gnuradio/io_signature.h>
#include "PIFESource_impl.h"

namespace gr {
  namespace PIFE {

    PIFESource::sptr
    PIFESource::make(const char *port, int samplesps, int freq)
    {
      return gnuradio::get_initial_sptr
        (new PIFESource_impl(port,samplesps,freq));
    }

    //Constructor privado
    PIFESource_impl::PIFESource_impl(const char *port, int samplesps, int freq)
      : gr::sync_block("PIFESource",
        gr::io_signature::make(0,0,0),
        gr::io_signature::make(1,1, sizeof(float)),
        d_port(port), d_samp_rate(samplesps), d_freq(freq)
      )
    {
      gr::thread::scoped_lock (d_ringbuffer_mutex);

      d_USB = open(port,O_RDWR|O_NOCTTY);

      //Si no puedo abrir el puerto, dar error.
      if (d_USB < 0)
        {
          throw std::runtime_error("can't open port");
          //error
        }

      //Si no esta dentro del rango permitido, dar error.
      if (d_freq < 10000000 && d_freq > 30000000)
        {
          throw std::runtime_error("Frecuencia fuera de rango");
          //error
        }

      //Configurar puerto
      tcgetattr(d_USB, &d_settings);
      cfmakeraw(&d_settings);
      tcsetattr(d_USB, TCSANOW, &d_settings);
    }
  }
}

```

```
//Enviar instruccion para detener muestreo
char cmd[] = "@300000000";
int n_written = write(d_USB,cmd,sizeof(cmd));
std::cout << cmd << std::endl;
int n = 0;
char response[1];
do
{
    n = read(d_USB, response, 1);
    std::cout<<response[0];
}
while(response[0] != '\n' && n>0);
std::cout<<std::endl;

//Enviar instruccion para sintonizar frecuencia
std::cout << "Frequency " << d_freq << std::endl;
snprintf(cmd,sizeof(cmd),"@4%08d",d_freq);
n=0;
n_written = write(d_USB,cmd,sizeof(cmd));
std::cout << cmd;
do
{
    n = read(d_USB,response, 1);
    std::cout<<response[0];
}
while(response[0] != '\n' && n>0);
std::cout<<std::endl;

//Enviar instruccion para setear frecuencia de muestreo
n=0;
std::cout << "Sample Rate: " << d_samp_rate << std::endl;
snprintf(cmd,sizeof(cmd),"@3%08d",d_samp_rate);
n_written = write(d_USB,cmd,sizeof(cmd));
std::cout << cmd;
do
{
    n = read(d_USB,response, 1);
    std::cout<<response[0];
}
while(response[0] != '\n' && n>0);
std::cout<<std::endl;

//Crear Ringbuffer
create_ringbuffer();
std::cout << "Ring Buffer Creado, Iniciando Thread" << std::endl;;

//Crear thread
d_thread = boost::shared_ptr<gr::thread::thread>
    (new gr::thread::thread(boost::bind(&PIFESource_impl::callback, this)));
}
```

```
/*
 * Destructor
 */
PIFESource_impl::~PIFESource_impl()
{

gr::thread::scoped_lock(d_ringbuffer_mutex);
//Detiene muestreo y cierra puerto USB
if (d_USB > 0)
{
char cmd[] = "@3000000000";
int n_written = write(d_USB,cmd,sizeof(cmd));
std::cout << cmd << std::endl;
int n = 0;
char response[1];
do
{
n = read(d_USB, response, 1);
std::cout<<response[0];
}
while(response[0] != '\n' && n>0);
std::cout<<std::endl;
close(d_USB);

}

//Termina thread
d_thread->interrupt();
d_thread->join();
}

//Funcion para crear ringbuffer
void PIFESource_impl::create_ringbuffer(void)
{
d_writer = gr::make_buffer(60000,sizeof(float));
d_reader = gr::buffer_add_reader(d_writer,0);
}

//Funcion que lee del puerto y carga en ringbuffer
void PIFESource_impl::callback()
{
unsigned char recbuf[60000];
float samplesbuf[60000];
int cantRecib = 0;
int nframes_room ;
while(1)
{
nframes_room = d_writer->space_available();
gr::thread::scoped_lock(d_ringbuffer_mutex);
cantRecib = read(d_USB,recbuf, nframes_room);

for (int i=0; i< cantRecib; i++)
{
samplesbuf[i] = (float)((int)(recbuf[i])- 128)/128;
}

memcpy(d_writer->write_pointer(),
samplesbuf,
cantRecib*sizeof(float));

d_writer->update_write_pointer(cantRecib);
boost::this_thread::sleep(boost::posix_time::milliseconds(40));
}
}
}
```

```
//Funcion que envia muestras a GNU Radio
int PIFEsource_impl::work(int noutput_items,
    gr_vector_const_void_star &input_items,
    gr_vector_void_star &output_items)
{
    float *out = (float*)output_items[0];
    int k;
    for (k=0; k< noutput_items;)
    {
        int nframes = d_reader->items_available();
        if (nframes == 0) {
            return k;
        }

        gr::thread::scoped_lock guard(d_ringbuffer_mutex);
        int nf = std::min(noutput_items - k, nframes);
        const float *p = (const float*) d_reader->read_pointer();
        for (int i = 0; i< nf; i++)
        {
            out[k+i] = *p++;
        }
        d_reader->update_read_pointer(nf);
        k += nf;
    }
    return k;
}

} /* namespace PIFE */
} /* namespace gr */
```

Solicitud de Aprobación de Tema

DISEÑO E IMPLEMENTACIÓN DE FRONT END ANALÓGICO PARA SDR

BANCHIO, Agustín Enrique



Facultad de Ciencias Exactas, Físicas y Naturales

AREA INGENIERÍA

ESCUELA DE ELECTRONICA

C.C. 755 - Correo Central - 5000 - CÓRDOBA

Tel. Directo (0351) 33-4147 int 110

Commutador: 433-4141 y 33-4152 - Interno 10

Sr. Director de la Escuela de Ingeniería Electrónica

Ing.: Rodrigo Bruni

Me dirijo a Ud. a fin de solicitar la **aprobación del tema del Proyecto Integrador (PI)** que propongo a continuación:

TEMA

Nombre del Proyecto: DISEÑO E IMPLEMENTACION DE FRONT END ANALÓGICO PARA SDR

Descripción: **Ver Anexo I**

Desarrollo del prototipo: Sí

Director de PI

Nombre: Ing. Prof. Rodrigo Bruni

Cargo: Profesor Asistente en Electrónica Analógica 3

Dirección Personal o Laboral: Av. Vélez Sarsfield 1611- Facultad de Ciencias Exactas, Físicas y Naturales - LIADE

TE: 4334147

Email: rodrigo.gabriel.bruni@unc.edu.ar

Firma del Director:

Co-Director de PI

Nombre: Ing. Prof. José Amado

Cargo: Profesor Asistente en Electrónica Analógica 3

Dirección Personal o Laboral: Av. Vélez Sarsfield 1561 – Instituto Nacional de Tecnología Industrial

TE: 4684835

Email: jamado@inti.gov.ar

Firma del Co-Director:

Datos del Estudiante

Nombre y Apellido: Agustín Enrique Banchio

Matrícula: 37852695

Materias que faltan aprobar: Ninguna

Dirección: Av. Valparaíso 4250 Manzana 8 Lote 26

Localidad: Córdoba Provincia: Córdoba

e-mail: agustin.banchio@gmail.com

Teléfono: 4621184

Firma:.....

Objetivo:

Se propone desarrollar una etapa analógica de recepción de radiofrecuencia incluyendo un convertor analógico digital para permitir el análisis de la señal digitalmente en una PC con software libre para radio definida por software.

Se buscara:

- Relevar las distintas tecnologías para cada etapa, y seleccionar las adecuadas.
- Comparar las alternativas de componentes para el desarrollo de la parte analógica y de digitalización.
- Diseñar el firmware de microcontrolador para la configuración de componentes como osciladores y la comunicación con la PC.
- Adaptar, en caso de ser necesario, software específico de radio definida por software ya existente para compatibilidad con este proyecto.
- Realizar pruebas para verificar el correcto funcionamiento del prototipo.

Antecedentes de Proyectos similares:

Se han realizado proyectos integradores que involucran radios definidas por software. Sin embargo, estos no consistieron en el diseño del Front End. Las principales diferencias entre otros proyectos son:

- El proyecto que se propone incluye el diseño y desarrollo de la etapa analógica para la recepción de señales de radiofrecuencia.
- El proyecto que se propone no utiliza FPGA para implementar ninguna etapa, en cambio, utiliza un microcontrolador.
- En proyectos anteriores se trabaja casi exclusivamente sobre FPGA en diseño de algún módulo de procesamiento digital de señales.

Duración y Fases de las tareas previstas: **Ver Anexo II**

Metodología

Lugar previsto de realización: Laboratorio de radiofrecuencias y microondas, Facultad de Ciencias Exactas, Físicas y Naturales, UNC.

Requerimiento de Instrumental y equipos: Osciloscopio, Analizador de espectro.

Inversión estimativa prevista por el alumno: \$3500

Apoyo Económico externo a la Facultad: Ninguna

Recibido Cátedra PI

.....

Firma

Córdoba, / / .

ANEXO I

1. Descripción Detallada del Proyecto

El proyecto consiste en el diseño e implementación de un prototipo de una radio receptora que permite analizar las señales por software.

Una radio definida por software permite implementar componentes que generalmente eran implementados en hardware (ej. demoduladores, mezcladores, filtros, amplificadores, etc.) a través de software.

El proyecto propone desarrollar una configuración como la siguiente:

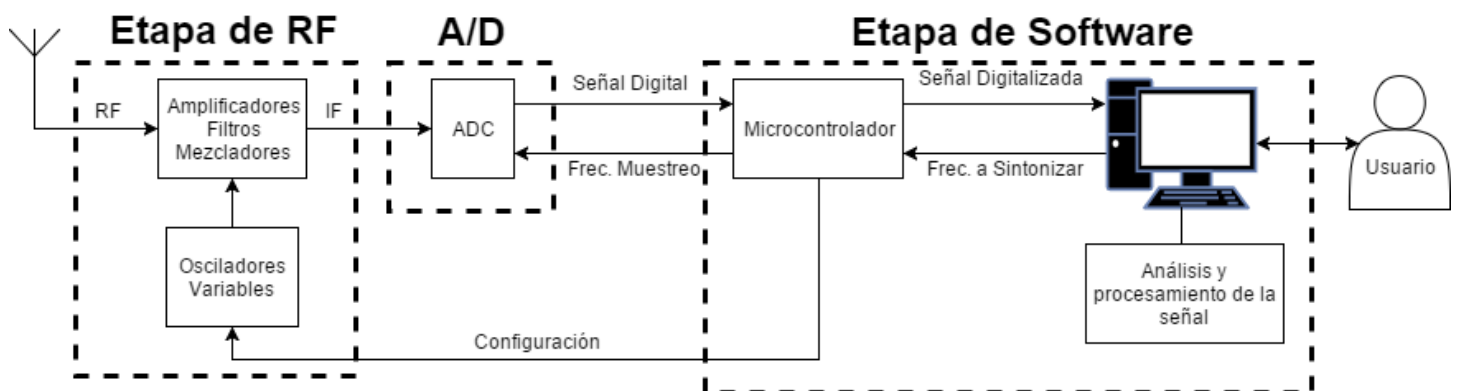


Diagrama en bloques

Como se puede observar en el diagrama, en este proyecto se desarrollará una etapa de RF para acondicionar la señal para ser digitalizada, se digitalizará la señal y en una etapa de software se la enviará a la PC y en ella se aplicaran algoritmos para procesarla digitalmente.

En la etapa de RF se trabajara con los siguientes componentes:

- Amplificadores, ya que la señal no se recibe con amplitud suficiente para ser utilizada directamente de la antena.
- Filtros para impedir que las señales que estén fuera del rango de trabajo interfieran con las señales que lo estén.
- Mezcladores que combinen la señal recibida por la antena con señales sintetizadas por osciladores y permiten obtener la diferencia de las frecuencias de ambas señales y transportar la información de la señal de frecuencia deseada a una frecuencia intermedia (IF) que el conversor analógico digital es capaz de digitalizar.
- Osciladores variables que permitan sintonizar la frecuencia deseada.

Se deberán seleccionar todos estos componentes de entre las alternativas del mercado e implementar sus circuitos de aplicación.

El conversor analógico digital debe ser capaz de tomar muestras de la señal por lo menos al doble de la frecuencia mayor de la señal entrante, acorde al teorema de Nyquist. Se pretende recibir un ancho de banda aproximado de 280kHz

Se deberá diseñar un software permanente para el microcontrolador (firmware) para que realice las siguientes funciones:

- Recibir configuración deseada de la PC
- Cambiar la frecuencia sintetizada por los osciladores según lo ordenado por la PC
- Determinar la frecuencia de muestreo del conversor AD según lo ordenado por la PC
- Recibir la señal ya digitalizada del ADC y enviarla a la PC

En la PC se utilizará software dedicado al procesamiento de señales de radiofrecuencia, como GNU Radio, el cuál de ser necesario será adaptado para comunicarse con el microcontrolador directamente. El conversor muestreará la señal en amplitud (no muestreo complejo).

Luego de diseñada la etapa analógica y el software se desarrollará una placa prototipo con estos diseños.

El rango de trabajo previsto para el receptor es entre 10Mhz y 30Mhz. Y se pretende demodular a través de software dedicado ya existente modulaciones analógicas y digitales.

El prototipo final deberá poder sintonizar una frecuencia dentro de su rango de trabajo y en la PC poder procesar en tiempo real la señal recibida.

Durante el desarrollo del proyecto, se probaran las diferentes etapas con instrumentos de medición de señales (osciloscopios, analizadores de espectro).

El prototipo terminado se probará con la ayuda de otros equipos transmisores que puedan enviar una señal modulada, en el mismo rango de trabajo, permitiendo al prototipo recibir la señal y demodularla por software.

2. Referencias Bibliográficas y de Software

https://es.wikipedia.org/wiki/Radio_definida_por_software

https://en.wikipedia.org/wiki/Software-defined_radio

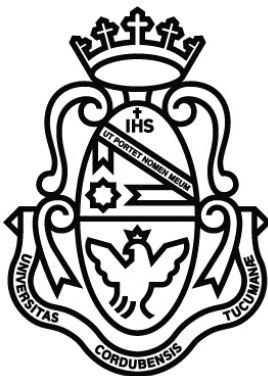
https://en.wikipedia.org/wiki/Nyquist%E2%80%93Shannon_sampling_theorem

<https://www.gnuradio.org/>

ANEXO II

Actividad	Mes					
	Primero	Segundo	Tercero	Cuarto	Quinto	Sexto
Selección de componentes y diseño de circuitos de aplicación	■					
Diseño de Software	■		■			
Implementación de placa receptora				■		
Pruebas y puesta a punto						■
Desarrollo de Informe	■					

Nota de aprobación



UNIVERSIDAD NACIONAL DE CÓRDOBA

Facultad de Ciencias Exactas, Físicas y Naturales

Escuela de Ingeniería Electrónica

Quien suscribe, el Profesor Rodrigo Bruni en su carácter de Director del Proyecto Integrador del Estudiante Agustín Enrique Banchio, denominado: “DISEÑO E IMPLEMENTACIÓN DE FRONT END ANALÓGICO PARA SDR” considera que el desarrollo del trabajo se ha completado según lo especificado en la Solicitud de Aprobación de Tema y se encuentra en condiciones de tramitar su defensa.

A los efectos de quién corresponda, en fecha/...../.....

Firma y aclaración del Director