System On Chip para la Reconstrucción de Señales con Sensado Compresivo

Autor: Mariano Leonel Acosta

Director: Dra. Ing. Luciana De Micco

Trabajo final de la carrera *Ingeniería Electrónica*

Julio de 2018

Resumen

Se llevó a cabo el diseño y puesta en marcha de un sistema reconstructor de señales con Sensado Compresivo (CS) [1] adquiridas a una tasa 16 veces por debajo de la frecuencia de Nyquist. El procesamiento es realizado en el Sistema en Chip (SoC) ZYNQ-7000 [2] y desde una computadora personal (PC) se envían comandos, configuraciones, diccionarios (FFT, DCT, Haar, Gabor, Daubechies, etc.), sub-muestras y secuencias binarias necesarias para el proceso de recuperación. La investigación comenzó con un estudio de la literatura actual sobre el CS [3–20], con el fin de comprender sus fundamentos teóricos y aplicaciones. En primer lugar, se analizaron las técnicas de optimización que permiten la recuperación de la señal submuestreada. El parámetro de decisión en la selección de la técnica adecuada fue la complejidad computacional, optando por aquél que fuese conveniente realizar en el sistema embebido. Se adoptó como sensor al *Demodulador Aleatoreo* (RD) [19] y posteriormente el *Modulador Aleatorio pre-Integrativo* (RMPI) siendo este un arreglo de RD en paralelo con mejores propiedades para CS [17]. Luego, se armó un modelo de simulación del mismo en Matlab, permitiendo el ajuste de sus parámetros (frecuencia de muestreo, factor de reducción, numero de canales, optimizadores, secuencias de mezclado, etc). Fueron evaluados distintos mapas caóticos (Logístico, TWBM, TWBM2, TWBM4 y FWTSM) para la generación de secuencias que el adquisidor requiere en el proceso de mezclado; con el fin de reemplazar el uso de generadores pseudoaleatorios. Las herramientas [21,22] permitieron evaluar distintos algoritmos de optimización (CoSamp, OMP, Gradiente Conjugado, etc).

Una vez que el modelo de simulación mostró resultados favorables para una implementación del sistema, se procedió a estudiar la tecnología del SoC, junto con las herramientas de diseño que ofrece la empresa Xilinx. Con el fin de minimizar el tiempo total en la reconstrucción de la señal, en la lógica programable se diseñaron diversos IP cores, especialmente dos aceleradores de Hardware (multiplicador de matrices y red de ordenamiento *Merge-Sort* [23]). En el microprocesador, se utilizó un *FreeRTOS* [24] en coperación con una pila TCP/IP *LWIP* [25] para poder brindar al SoC con funcionalidades en red. Se diseño un protocolo sencillo de mensajes a nivel de aplicacion para la comunicación entre una PC y la placa de desarrollo Zedboard. Para tal motivo fue programada una *Interfaz Grafica de Usuario* (GUI) con librerias de Python y QT-Designer (PyQT versión 5). Finalmente, se fabricó un conversor Digital/Analógico con el propósito de generar señales de voltajes a partir de las submuestras enviadas desde el GUI.

Agradecimientos

Ante todo quisiera agradecer a mi directora de tesis, la Dra. Ing. Luciana De Micco, por permitirme realizar el tema propuesto, brindar correcciones y sugerencias. Asimismo, por dedicar su tiempo y paciencia. Luciana me dio a conocer sobre del curso Advanced School on Programmable System-on-Chip for Scientific Instrumentation dictado en el Centro Internacional de Física Teórica (ICTP), en Trieste, Italia. Tuve la posibilidad de asistir al mismo en noviembre de 2017. Allí, pude formarme sobre las herramientas que hicieron posible este trabajo. Por este motivo, hago una mención especial a los organizadores del evento Andrés Cicuttin, Maria Liz Crespo y Joseph Niemela; quienes aceptaron mi aplicación en base a mis antecedentes. Particularmente debo agradecer por las clases dictadas al Ing. Cristian Sisterna, de la Universidad Nacional de San Juan, y a los investigadores de la Universidad de Castilla-La Mancha: Fernando Rincón Calle, Julio Dondo y a José Antonio de la Torre. Fernando y Julio fueron fueron quienes me sugirieron la arquitectura de multiplicación matricial utilizando tres puertos HP en simultáneo, mientras que José Antonio me dio la idea de controlar el sistema en red mediante una interfaz gráfica.

En relación a la Universidad Nacional de Mar del Plata, agradezco al Dr.Ing. Maximiliano Antonelli por su colaboración; al profesor encargado de la materia Trabajo Final, el Ing. Gustavo Uicich, por aceptar la propuesta del proyecto y dedicar su tiempo a la evaluación; y al Dr. Ing. Gustavo Meschino, por el apoyo otorgado durante la carrera. A su vez, debo agradecer a las cátedras de Sistemas de Control y Redes de Datos; especialmente a la Mg. Ing. Mónica Liberatori. Ambas cátedras me permitieron viajar a Italia para realizar el curso. Si no hubiese sido así, este proyecto no se hubiera podido llevar a cabo. Finalmente, doy gracias a todos los docentes que me transmitieron sus conocimientos y experiencias durante mis estudios.

Durante un año la investigación fue financiada por el Consejo Interuniversitario Nacional (CIN) dentro del marco de Becas para estudiantes de grado. Por otra parte, agradezco al Ministerio de Educación y la Embajada de Estados Unidos por otorgarme la beca Friends of Fulbright. Esa distinción me dio la posibilidad de participar durante dos meses en un programa de intercambio en Virginia Tech, EEUU. No sólo la experiencia me sirvió para perfeccionar mis habilidades en el idioma Inglés, sino que también pude obtener libros académicos y herramientas que complementaron mi formación. Esto fue de gran ayuda para la comprensión del material bibliográfico del trabajo, debido a que la mayoría se encuentra en Inglés.

Respecto a lo profesional, la empresa INVAP S.E aceptó en enero de 2016 que pudiera realizar mi pasantía supervisada en sus instalaciones de la ciudad de San Carlos de Bariloche. Dicha experiencia fue clave para desarrollar mi entendimiento sobre los sistemas FPGA y sus cuestiones de fabricación. Definitivamente, esos conocimientos generaron un impacto favorable en el resultado final de esta tesis.

Para concluir, le debo un distintivo agradecimiento a toda mi familia por asistirme incondicionalmente en todos los viajes y actividades que realizo siempre. En especial, a mi primo Julian A. Acosta por inspirarme desde muy chico a seguir el camino de la Ingeniería Electrónica. Y a mis padres, Fabio E. Acosta y M. Elena Destefano, porque siempre lo dieron todo por mi bienestar y no habría llegado al lugar donde estoy hoy si no fuera por ellos.

Mariano L. Acosta

Mar del Plata, 14 de julio de 2018

Índice general

Glosario

A/D Analógico/Digital.

ADC Conversor Analógico-Digital, por sus siglas en Inglés.

AIHT Umbralamiento Duro Iterativo Acelerado, por sus siglas en Inglés.

AO Amplificador Operacional.

ASIC Circuito Integrado de Aplicación Específica, por sus siglas en Inglés.

AXI Interfaz Avanzada Extensible, por sus siglas en Inglés.

BRAM Bloque de Memoria de Acceso Aleatorio, por sus siglas en Inglés.

BSP Paquete de Soporte de Placa, por sus siglas en Inglés.

CGP Persecución del Gradiente Conjugada, por sus siglas en Inglés.

CLB Bloques Lógicos Configurables, por sus siglas en Inglés.

CMOS Semiconductor Complementario de Óxido Metálico, por sus siglas en Inglés.

COM Puerto de Comunicación, por sus siglas en Inglés.

CoSaMP Persecución e Igualación de Sensado Compresivo, por sus siglas en Inglés.

CS Sensado Compresivo, por sus siglas en Inglés.

D/A Digital/Analógico.

dB Decibel, unidad de intensidad relativa a un valor base.

DC Componente de Continua, por sus siglas en Inglés.

DCT Transformada Coseno Discreta, por sus siglas en Inglés.

DDR Tasa Doble de Datos, por sus siglas en Inglés.

DFT Transformada Discreta de Fourier, por sus siglas en Inglés.

DHCP Protocolo De Configuración Dinámica De Host , por sus siglas en Inglés.

DMA Acceso Directo a Memoria, por sus siglas en Inglés.

DSP Procesamiento Digital de Señal, por sus siglas en Inglés.

DST Transformada Seno Discreta, por sus siglas en Inglés.

ENOB Cantidad Equivalente de Bits, por sus siglas en Inglés.

FFT Transformada Rápida de Fourier, por sus siglas en Inglés.

FMC Tarjeta Intermedia de FPGA, por sus siglas en Inglés.

FPGA Arreglo de Puertas Programables, por sus siglas en Inglés.

FPU Unidad de Punto Flotante, por sus siglas en Inglés.

FSM Máquina de Estados Finita, por sus siglas en Inglés.

GNU Sistema operativo de tipo UNIX.

GP Persecución del Gradiente, por sus siglas en Inglés.

GSPS Giga Muestras por Segundo, por sus siglas en Inglés.

GUI Interfáz Gráfica de Usuario, por sus siglas en Inglés.

Hz Hertz, unidad de frecuencia.

I/O Entrada/Salida, por sus siglas en Inglés.

IC Circuito Integrado, por sus siglas en Inglés.

IDE Entorno de Desarrollo integrado, por sus siglas en Inglés.

IHT Umbralamiento Duro Iterativo, por sus siglas en Inglés.

IOP Periférico de Entrada/Salida, por sus siglas en Inglés.

IP Protocolo de Internet, por sus siglas en Inglés.

IP Core Bloque de Propiedad Intelectual, por sus siglas en Inglés.

IRQ Pedido de Interrupción, por sus siglas en Inglés.

ISA Arquitectura del Conjunto de Instrucciones, por sus siglas en Inglés.

JTAG Joint Test Action Group en Inglés.

LSB Bit menos Significativo, por sus siglas en Inglés.

LUT Tabla de Búsqueda, por sus siglas en Inglés.

MMU Unidad de Administración de Memoria, por sus siglas en Inglés.

MSB Bit Más Significativo, por sus siglas en Inglés.

MSPS Mega Muestras por Segundo, por sus siglas en Inglés.

N/A No Aplica, omisión deliberada.

NSP Propiedad Del Espacio Nulo, por sus siglas en Inglés.

NUS Muestreador No Uniforme, por sus siglas en Inglés.

OCM Memoria en Chip, por sus siglas en Inglés.

OMP Persecución e Igualación Ortogonal, por sus siglas en Inglés.

PC Computadora Personal, por sus siglas en Inglés.

PCI Interconexión de Componentes Periféricos, por sus siglas en Inglés.

PDM Modulación por Densidad de Pulsos, por sus siglas en Inglés.

PWM Modulación por Amplitud de Pulso, por sus siglas en Inglés.

RAM Memoria de Acceso Aleatorio, por sus siglas en Inglés.

RIP Propiedad Isométrica Restringida, por sus siglas en Inglés.

RMPI Modulador Aleatorio Pre-Integrador, por sus siglas en Inglés.

RTL Nivel de Registro/Transferencia, por sus siglas en Inglés.

SDK Kit de Desarrollo de Software, por sus siglas en Inglés.

SNR Relación Señal a Ruido, por sus siglas en Inglés.

SO Sistema Operativo.

SoC Sistema en Chip, por sus siglas en Inglés.

TCP Protocolo de Control de Transmisión, por sus siglas en Inglés.

uC Microcontrolador.

VPP Voltaje Pico a Pico.

Capítulo 1

Introducción

1.1. Antecedentes y Contexto

Los fundamentos teóricos del procesamiento digital de señales, desarrollados principalmente por Nyquist y Shannon, muestran que señales continuas limitadas en banda pueden ser exactamente reconstruidas muestreando al doble de la frecuencia máxima presente en su espectro [26]. Esto ha producido una revolución digital, en la que sistemas electrónicos de sensado y procesamiento demuestran ser más robustos, versátiles y económicos que sus contrapartes analógicas. En varias aplicaciones críticas, la frecuencia de muestreo requerida resulta ser demasiado elevada. Como consecuencia, la cantidad de datos generados por estos sistemas ha crecido exponencialmente, produciendo inconvenientes en la capacidad de memoria necesaria para almacenar las muestras. Por otro lado, construir un dispositivo de adquisición a altas frecuencias puede ser costoso e incluso físicamente imposible. Un ejemplo de ello son los conversores analógicos/digitales que trabajan en velocidades del GSPS. A pesar de los grandes avances en el poder computacional, en aplicaciones como el procesamiento de imágenes médicas, video, espectroscopía, sensado remoto y astronomía, entre otros, el análisis de datos especialmente el realizado *On-board*[1] resulta un gran desafío. Principalmente, por las dificultades de almacenamiento, procesamiento y transmisión.

De manera típica, para la compresión de información, se procede como muestra la Figura 1.1. Cuando se digitaliza una señal mediante un ADC, la cantidad de datos se puede reducir aplicando un algoritmo *con* ó *sin pérdidas* de información en el dominio digital. Un ejemplo de ello es la codificación MP3 [27] en audio o el formato H.263 en video [28]. Sin embargo, se debe almacenar primero todas las muestras; lo que provoca el inconveniente de cuellos de botella en memoria. Afortunadamente, se han desarrollado técnicas novedosas de compresión de datos, en las cuales se busca encontrar la representación más concisa de la señal a cambio de una cantidad aceptable de distorsión. El Sensado Comprensivo (CS, Compressed Sensing) [3–20] es un nuevo paradigma en la adquisición y reconstrucción eficiente de señales a partir de un número de muestras menor que el requerido por el teorema de

[1]Procesamiento en el sistema de adquisición. Ejemplo: Filtrado de imágenes realizado en un satélite.

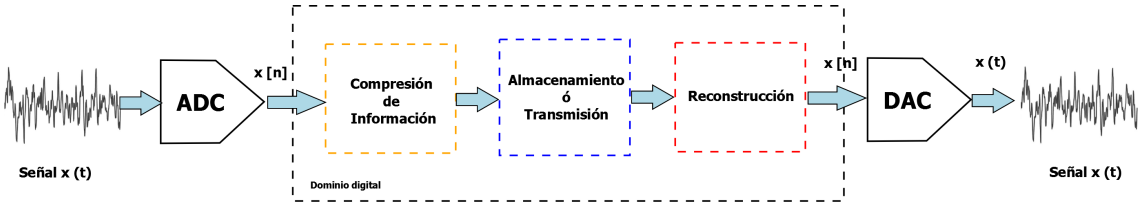


Figura 1.1: Cadena de procesamiento clásica que involucra compresión de datos.

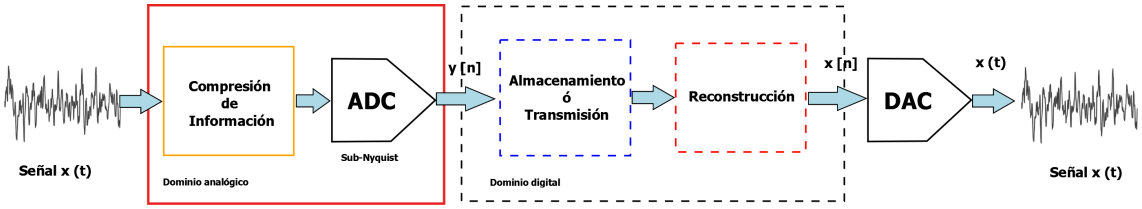


Figura 1.2: Propuesta de Sensado Compresivo.

muestreo de Shannon-Nyquist. En vez de adquirir la señal a la frecuencia de Nyquist y luego aplicar una técnica de compresión, CS propone una manera de comprimir la información directamente en la etapa de adquisición (Figura 1.2). La compresión de datos se realiza en el dominio analógico antes de la adquisición. Se evita almacenar muestras redundantes.

Este novedoso paradigma provee nuevas metodologías para el diseño de sensores y conversores analógicos/digitales. La idea principal de CS es simplificar el diseño en hardware de adquisidores de señales a cambio de aumentar la complejidad computacional en el proceso de reconstrucción digital. Es decir, generar una relación de compromiso o *tradeoff*. Entonces, la reconstrucción se logra encontrando soluciones a un sistema de ecuaciones indeterminado mediante matrices y utilizando restricciones junto con optimización no lineal. Un punto crítico de CS es la elección de la matriz que implementa el muestreo, ésta debe satisfacer la *propiedad de restricción isométrica* (RIP) y *coherencia* [5]. También, es condición necesaria para la señal a reconstruir mediante CS poseer abundante cantidad de muestras nulas en algún dominio, por ejemplo, en el dominio de la transformada wavelet o transformada discreta coseno (DCT). En la literatura reciente se detallan diversos algoritmos de procesamiento para CS, cuyo rendimiento dependen principalmente del tipo de señal y aplicación en concreto. Las principales ventajas que pueden obtenerse con este paradigma son:

- Reducción en el consumo de potencia [29, 30].
- Reducción del tamaños de sensores [31].
- Reducción del ancho de banda en la transmisión de datos [32].
- Muestreo por debajo de la frecuencia de Nyquist [33].
- Restauración de señales contaminadas con ruido o con ausencia de información [34].

Ejemplos de casos exitosos con CS incluyen receptores CMOS en el rango de 100 MHz-2 GHz, fabricación de espectrómetros en astronomía [35], disminución del tiempo de exposición a la radiación en resonancia magnética médica [36], redes de sensores inalámbricos [37], radio cognitiva [38], entre otros.

Además, CS requiere que la matriz de muestreo sea generada de manera aleatoria, con el fin de cumplir con las propiedades matemáticas que aseguran la reconstrucción de la señal [4]. Sin embargo, cuando se realizan implementaciones en hardware se suelen utilizar generadores pseudoaleatorios. Es bien conocido que dichos generadores deben pasar pruebas estadísticas [39] para asemejarse a una fuente verdaderamente aleatoria. En consecuencia, la complejidad circuital de estos generadores es elevada. [40]. Se propone entonces utilizar generadores caóticos como alternativa, ya que si bien son sistemas deterministas simples esta demostrado que originan señales de aspecto estocástico [41]. Por lo tanto, son relativamente sencillos de realizar en hardware. Sólo se necesita una función iterativa[2] del tipo $x_n = f(x_{n-1})$ para lograr la fuente caótica [42]. La sensibilidad a sus condiciones iniciales hace que en estos sistemas la predictibilidad sea a corto plazo, lo que los ubica en una posición intermedia entre deterministas y estocástico. Como consecuencia, se desarrollaron en los últimos años un número creciente de aplicaciones de los sistemas caóticos, empleándolos principalmente como generadores de ruido controlado [43], generadores de números pseudo aleatorios [44], portadoras o señales llave en sistemas de encriptado [45], etc.

Por otra parte, existe una tendencia en mejorar procesos específicos mediante hardware, como es el caso de la *Unidad de Procesamiento Tensorial* (TPU) fabricada por Google (para acelerar su framework *Tensorflow* [46]) ó el *Apple Neural Engine* [47], que realiza reconocimiento de imágenes. A su vez, los dispositivos lógicos programables como los FPGA se han consolidado como la opción estándar en varias aplicaciones de tiempo real, debido a su reducido tamaño, bajo consumo y rendimiento optimizado a nivel hardware [48]. También, existen nuevos *Sistemas en Chip* (SoC) que incorporan núcleos de procesador y FPGA permitiendo que parte de un sistema se ejecute en hardware y parte en software. Estos dispositivos son ideales para implementar la técnica de CS ya que el algoritmo requiere cálculo en paralelo y también desarrollo secuencial. En particular, la empresa Xilinx provee las herramientas *Vivado Design Suite* y *Síntesis de Alto Nivel* (HLS, por sus siglas en Inglés). Las mismas facilitan la creación de bloques IP mediante especificaciones escritas en C, C++ y SystemC para luego ser traducidas a código RTL. Gracias a esto, el tiempo de desarrollo necesario para lanzar una aplicación al mercado se ve considerablemente reducido.

[2]También debe cumplir con las propiedades descriptas en [42].

1.2. Objetivo General

En este proyecto se propone el diseño e implementación de un sistema para la recuperación de señales con Sensado Compresivo en el SoC Zynq 7000 de Xilinx. Se busca reconstruir señales que fueron sub-muestreadas, es decir, que no cumplen con lo enunciado por el teorema de Nyquist-Shannon [26]. Para ello se debe aprovechar las mejoras en rendimiento que ofrece el sistema híbrido microprocesador/FPGA mediante la cooperación hardware/software (HW/SW). Conceptualmente, lo que se espera del diseño se encuentra en la Figura 1.3. Primero, se debe adoptar un modelo de sensor analógico cuyo funcionamiento este basado en CS. Luego, con conocimiento previo sobre la señal de interés $x(t)$, información sobre el sensor y el conjunto de muestras obtenidos por el mismo, ingresar a un reconstructor digital para que genere la versión reconstruida $\hat{x[n]}$. El trabajo consiste en la realización del reconstructor y no del sensor. De este último se pretende armar un modelo de simulación.

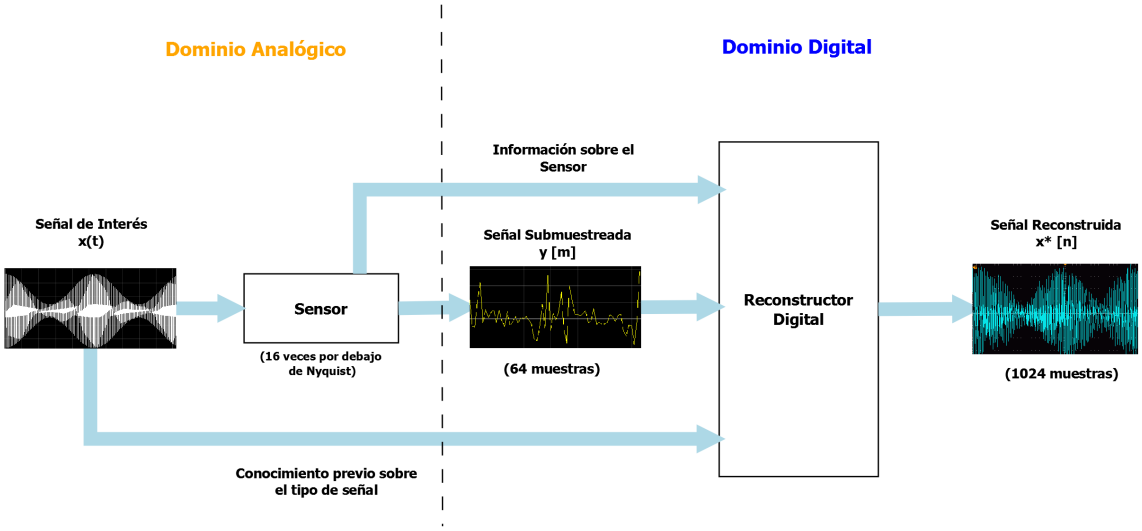


Figura 1.3: Diagrama en bloques conceptual del trabajo final.

Las herramientas principales a utilizar son el SoC Zynq-7000 de Xilinx incluido en la placa de desarrollo Zedboard (internamente presenta un FPGA y dos procesadores ARM Cortex A9), los periféricos presentes en la plataforma (entre ellos una interfaz de red y conectores PMOD de entrada/salida), los programas de síntesis de alto nivel Vivado-HLS de Xilinx (versión 2016), el cual permite desarrollar hardware directamente desde código C/C++ y el entorno Matlab (versión 2013) en conjunto con Simulink para la simulación y generación de señales. Se tiene en cuenta potenciales aplicaciones, como así también se contrasta los resultados obtenidos con una simulación del paradigma clásico.

1.3. Objetivos Particulares

En base a lo discutido en la introducción, se propone evaluar las siguientes hipótesis:

- La técnica CS con secuencias caóticas presenta un buen comportamiento, comparable al sistema clásico.
- Realizar un sistema de reconstrucción con CS mediante co-diseño HW/SW ofrece mejoras de rendimiento. En consecuencia, es posible procesar las señales en tiempo real, es decir, antes de la llegada del siguiente bloque de datos.

Existen varias arquitecturas en hardware para CS [5]. Algunas de ellas, utilizan secuencias generadas estocásticamente. Por ese motivo, se estudia qué ocurre con generadores caóticos. Conjuntamente, se espera demostrar las ventajas que otorga la arquitectura del SoC en CS.

Los pasos que guían el desarrollo del proyecto se enuncian a continuación:

- Estudio de la técnica de Sensado Compresivo y sus aplicaciones.
- Estudio de Sensores Analógicos con CS.
- Creación de un modelo de simulación del sensor en Matlab.
- Simulación, evaluación y selección de un algoritmo de reconstrucción.
- Estudio de las tecnologías SoC, en particular el ZYNQ 7000 de Xilinx.
- Diseño e implementación del sistema en hardware mediante HLS.
- Diseño de *Interfaz Gráfica* (GUI) para control en red del sistema.
- Fabricación de un conversor D/A para generar señales de voltaje.
- Evaluación del rendimiento del sistema y sus aplicaciones.

1.4. Estructura de la Tesis

Se comienza con el Capítulo 2 haciendo una explicación de los fundamentos teóricos del Sensado Compresivo; presentando todas las ecuaciones y conceptos fundamentales mínimos para el entendimiento del trabajo. Queda a cargo del lector ahondar en cualquier tema particular que considere necesario. Para ello se facilitan todas las referencias consultadas. También, se introduce al sensor de adquisición adoptado como modelo. Luego, en el Capítulo 3 se describe todas las simulaciones realizadas previo al diseño definitivo. Se presenta la comparación entre generadores caóticos junto con todos los algoritmos de reconstrucción bajo consideración. A su vez, se definen los cuantificadores de evaluación que son utilizados a lo largo del informe. Los resultados obtenidos, criterios de evaluación y diseño final son también expuestos. Posteriormente, se dedica el Capítulo 4 a introducir al *Sistema en Chip*

ZYNQ 7000 con la placa de desarrollo Zedboard, haciendo énfasis en su arquitectura y prestaciones. Ese capítulo es importante para que el lector pueda comprender el trasfondo del funcionamiento del diseño; en especial las interfaces AXI utilizadas en la comunicación de datos. Además, se muestra las herramientas Vivado y el entorno de creación de software Xilinx SDK. Se concluye con los pasos a seguir para la creación del sistema, desde las especificaciones en lógica programable hasta la capa de aplicación. La implementación y funcionamiento en detalle del proyecto están contemplados en el Capítulo 5. En el mismo se muestra el diagrama en bloques del trabajo final junto con los bancos de mediciones. Asimismo, se explica mediante flujos algorítmicos cada etapa de procesamiento, incluido los bloques del FPGA, el software del procesador, los inconvenientes hallados y las soluciones propuestas. El Capítulo 6 abarca desde las especificaciones técnicas hasta la puesta en marcha de un convertidor Digital/Analógico del tipo $\Delta\Sigma$ en código VHDL y con circuitería externa a la placa. Esto fue realizado para convertir en voltaje a las señales recuperadas con CS. Si bien se podía haber utilizado un conversor D/A comercial, se optó por diseñar y fabricar uno desde cero. La razón fue poner en práctica conocimientos adquiridos durante la carrera relacionados con electrónica analógica y sistemas de control que de otra manera quedaban excluidos del trabajo. Por último, una vez que el reconstructor se encontró finalizado, se hicieron pruebas con distintos tipos de señales para la demostración de su funcionamiento, siendo las mismas presentadas en el Capítulo 7.

Capítulo 2

Sensado Compresivo

2.1. Teoría Matemática

Cuando el proceso de adquisición de muestras es lineal, el problema de la reconstrucción de una señal discreta $x \in \mathbb{C}^N$ a partir de un vector de observaciones $y \in \mathbb{C}^M$ se puede modelar con el siguiente sistema de ecuaciones [3]:

$$y = Ax \tag{2.1}$$

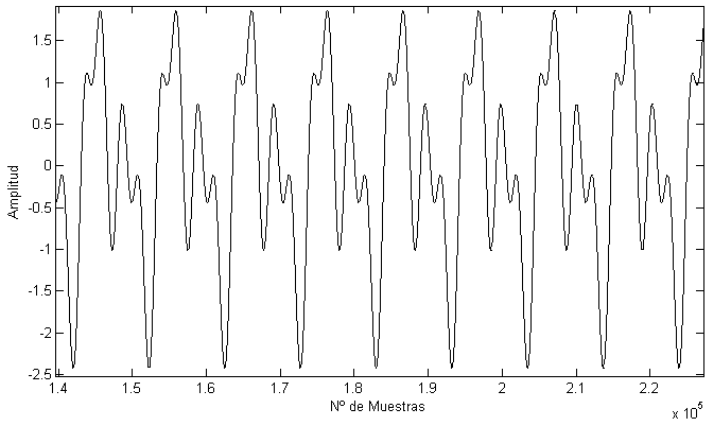
En la expresión (2.1) los vectores x e y están relacionados mediante la matriz $A \in \mathbb{C}^{MxN}$, que modela la operación lineal realizada en el muestreo. Luego, para recuperar la señal x es necesario resolver (2.1). En principio, el sistema de ecuaciones esta determinado si la cantidad M de muestras obtenidas equivalen a la longitud N de la señal. Esto último se cumple por ejemplo en el caso de los conversores A/D convencionales. Sin embargo, como el objetivo en este trabajo es implementar un sistema que sea capaz de reconstruir una señal obtenida con un número menor de muestras que el dictado por el teorema de Nyquist, se requiere analizar el caso $M < N$. En tal situación, la teoría del álgebra lineal establece que el sistema es indeterminado, es decir, existen infinitos vectores x que se pueden recuperar bajo las mismas condiciones de medición. Por lo tanto, se requiere información adicional para resolver el sistema de ecuaciones. La teoría del Sensado Compresivo desarrollada por primera vez en [4] por Terence Tao y Emmanuel Candes demuestra que esto es posible bajo la asunción que el vector x sea disperso.

2.1.1. Señales Dispersas

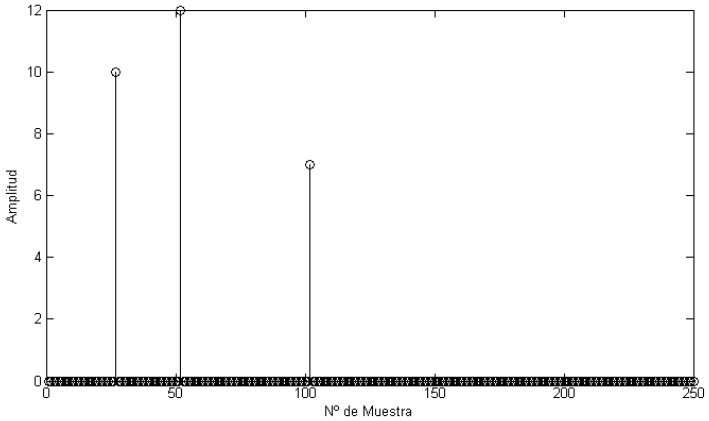
Se define el grado de dispersión o *sparcity* $k \in \mathbb{N}$ de un vector como la cantidad de elementos no nulos [5]. Formalmente:

$$k := \sum_{j=1}^{N} 1_{\{x_j \neq 0\}} \tag{2.2}$$

Basándose en (2.2) decimos que un vector de longitud N es disperso o ralo cuando $k << N$. Existen numerosos casos prácticos en los cuales se obtienen señales con una



(a) Señal en el dominio de tiempo



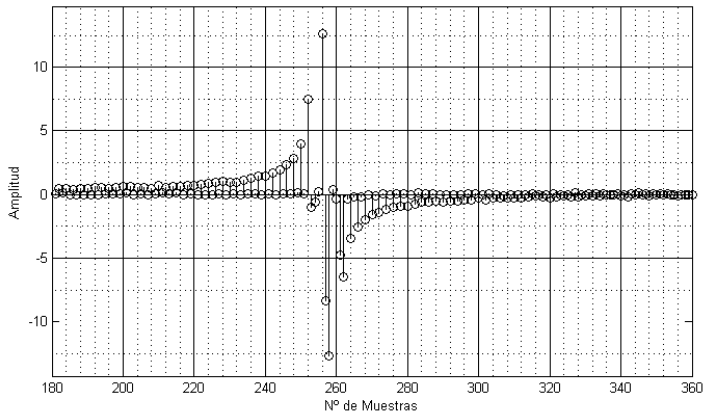
(b) FFT de la señal (Espectro Unilateral)

Figura 2.1: Señal continua en el tiempo y dispersa en frecuencia

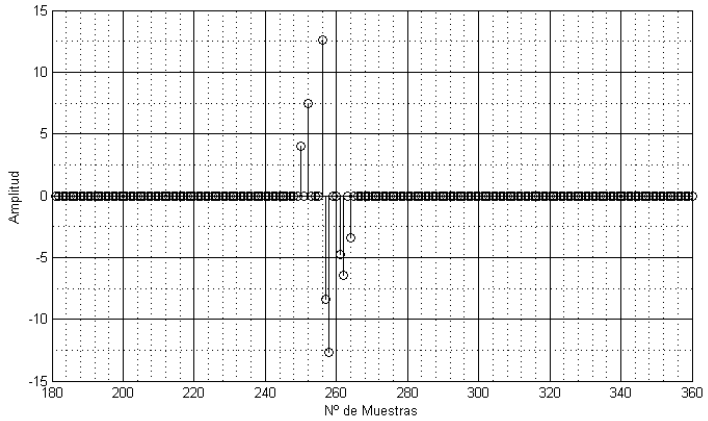
representación dispersa luego de un cambio de base. En otras palabras, no importa que el valor k de una señal no sea pequeño en el dominio que se están obteniendo las muestras, el modelo de CS funciona siempre y cuando x sea rala en alguna base[1]. El siguiente ejemplo presentado en la Figura 2.1 nos muestra cómo una señal constituida por la suma de tres señales senoidales de distintas amplitudes y frecuencias no es dispersa en el dominio del tiempo discreto pero puede ser representada con sólo seis muestras luego de aplicarle el algoritmo FFT. Por supuesto que, en este caso, no se considera el efecto de *dispersión espectral* [11]. Si la longitud de la señal fuese de 512 muestras, 506 de estas serían cero y por lo tanto k resultaría ser seis.

Puede suceder que no sea posible lograr una representación dispersa en el sentido estricto de (2.2). No obstante, en las denominadas señales compresibles se procede a aplicar un umbralamiento previo para lograr el nivel de dispersión deseado. Es decir, los elementos del vector que sean menores a un umbral T en valor absoluto son

[1]Existe un principio de incertidumbre similar al de Heisenberg para representaciones dispersas que indaga esta cuestión [14].



(a) DCT de la señal



(b) DCT de la señal luego de aplicar umbralamiento (T=3))

Figura 2.2: Coeficientes DCT de una señal compuesta por tres tonos

reemplazados por cero. Esta técnica es utilizada frecuentemente en la compresión con pérdidas basada en la transformada coseno discreta (DCT) [7]. Considerando la señal compuesta por tres tonos del ejemplo anterior, se presenta en la Figura 2.2 cómo el grado de dispersión de la señal aumenta en el dominio transformado por la DCT al aplicar umbralamiento. El truncamiento de los coeficientes a cero en este caso resulta en pérdida de información que puede ser tolerable o no dependiendo de la aplicación.

Habiendo enunciado las consideraciones respecto de cómo es posible lograr representaciones dispersas de una señal, es conveniente replantear el sistema de ecuaciones (2.1) de la siguiente manera:

$$y = \Phi\Psi\alpha \tag{2.3}$$

Donde $\Psi \in \mathbb{C}^{NxN}$ es el diccionario o base, α los coeficientes de la señal x en dicha base, $\Phi \in \mathbb{R}^{MxN}$ el kernel de compresión y $A = \Phi\Psi$. Entonces, el sistema modificado

(2.3) recuperará α a partir del vector de submuestras y. Será luego necesario realizar la operación $x = \Psi\alpha$ para obtener finalmente la señal x deseada.

Lo presentado hasta ahora nos muestra que la información intrínseca de una señal compresible es bastante menor que su longitud. El siguiente paso lógico sería responder estas cuestiones sobre el modelo del Sensado Compresivo:

- ¿Cuáles son los algoritmos que existen para recuperar x conociendo A e y?
- ¿Qué propiedades debe cumplir la matriz de sensado $A \in \mathbb{C}^{MxN}$ para asegurar la reconstrucción de x?
- ¿Cómo diseñar un proceso de adquisición lineal de forma tal que pueda ser planteado como un problema de CS?
- ¿Cuál es el grado de dispersión máximo que puede cumplir x en alguna base conocida?
- ¿Qué relación de compresión N/M puede ser alcanzado con este modelo?

2.1.2. Reconstrucción de la Señal

La primera de estas preguntas se resuelve planteando (2.1) como un problema de optimización en el que se busca minimizar la norma l_0 de x.

$$\min \ \|x\|_0 \ \text{ sujeto a } y = Ax \tag{2.4}$$

El problema tendrá como solución la aproximación $\hat{x}$. Como $\|x\|_0$ es equivalente por definición a la expresión (2.2), la norma l_0 se corresponde con la premisa de señal rala. Sin embargo, como se fundamenta en [3] y [5] resolver (2.4) es un problema combinacional NP-Complejo numéricamente intratable. Afortunadamente, es posible aproximar (2.4) con la función convexa $\|.\|_1$ y por ende la optimización se vuelve viable computacionalmente. Además, generalizamos el modelo considerando que las muestras y pueden presentar ruido en la medición. Para ello modificamos la restricción para que la aproximación $\hat{x}$ cumpla con un valor de tolerancia ϵ según se muestra a continuación:

$$\min \ \|x\|_1 \ \text{ sujeto a } \ \|y - Ax\|_2 \leq \epsilon \tag{2.5}$$

Como se verá más adelante, aunque (2.5) puede ser optimizada de manera convencional, existen algoritmos heurísticos que reducen notablemente su resolución en tiempo y complejidad de implementación. En esta etapa la dificultad principal reside en encontrar la ubicaciones no nulas del vector, siendo estas desconocidas en principio.

2.1.3. Garantías en la Reconstrucción

El segundo aspecto a considerar son las propiedades de la matriz de sensado A. Como ya es bien sabido [4, 9] las siguientes propiedades aseguran la reconstrucción de un vector disperso:

- Spark
- Coherencia
- Propiedad Isométrica Restringida (RIP)
- Propiedad del Espacio Nulo (NSP)

Se define el *Spark* de una matriz A como la menor cantidad de columnas que son linealmente dependientes. En contraste, el *Rango* de una matriz es el mayor número de columnas linealmente independientes. En [10] se demuestra un teorema que establece que si $y = Ax$ tiene una solución x tal que $\|x\|_0 < spark(A)/2$ entonces x es la solución más dispersa. A pesar de la conveniencia de este teorema, el *Spark* no es sencillo de calcular ya que requiere una búsqueda combinacional [14]. Es por ello que se utilizan cotas para la estimación de este parámetro.

La siguiente propiedad descripta en [12] es la *Coherencia* de una matriz A, definida como el máximo producto interno normalizado entre diferentes columnas pertenecientes a A. Suponiendo que $A = [a_1\, a_2 \ldots a_n]$, siendo a_i sus columnas constituyentes, la coherencia $\mu(A)$ esta dada por:

$$\mu(A) = \max_{k,j,k\neq j} \frac{|{a_k}^T a_j|}{\|a_k\|_2 \,.\, \|a_j\|_2} \tag{2.6}$$

Luego, de (2.6) se desprende:

- $\mu(A)$ caracteriza la dependencia entre las columnas de A.
- Para matrices con mayor cantidad de columnas que filas se cumple $\mu(A) > 0$
- Si $A \in \mathbb{R}^{MxN}$ con $M < N$ entonces $\mu(A) \geq \frac{1}{\sqrt{M}}$
- Se desea que $\mu(A)$ sea pequeño ya que para matrices unitarias[2] $\mu(A) = 0$

A su vez, en [13] se relaciona los conceptos de *Spark* y *Coherencia* de la siguiente manera:

$$\text{spark}(A) \geq 1 + \frac{1}{\mu(A)} \tag{2.7}$$

Considerando que la ecuación (2.6) es relativamente sencilla de calcular, es posible combinar el primer teorema mencionado en esta sección y (2.7) obtener el corolario:

[2]Una matriz unitaria real es una matriz ortogonal, es decir, su inversa coincide con su traspuesta. A debe ser unitaria para satisfacer con la propiedad RIP que será definida en breve.

- Si $y = Ax$ posee una solución x que cumple $\|x\|_0 < (1 + \mu^{-1}(A)/2)$ entonces x es la solución más dispersa.

Más importante aún, si x cumple con este corolario entonces es el *único* minimizador de las normas l_0 y l_1 y por ende la solución óptima de los problemas de optimización (2.4) y (2.5). Para detalles en la demostración consultar [13]. En relación al análisis de cuánto permite la matriz de sensado A discernir entre dos señales *k-dispersas*, se recurre a la *Propiedad Isométrica Restringida (RIP)* que se define como [5]:

$$(1 - \delta_h)\|x\|_2^2 \leq \|Ax\|_2^2 \leq (1 + \delta_h)\|x\|_2^2 \tag{2.8}$$

La matriz A satisface la propiedad RIP de orden h si existe un valor $\delta_h \in (0, 1)$ tal que cumple con la relación (2.8). Con (2.8) queda establecido un criterio para el estudio de la robustez frente al ruido en el modelo de CS. Esencialmente, se requiere que cada conjunto de columnas de la matriz con cardinalidad [3] igual o menor a k se comporte como un sistema ortonormal. Como es mencionado en [4], se establece que el cálculo del RIP es universal para toda señal con dispersión k. Además, todos los métodos que resuelven (2.1) inclusive mediante (2.4) y (2.5) necesitan $\delta_{2h} \leq 1$ para encontrar una solución x. La condición anterior se satisface con gran probabilidad para determinados tipos de matrices A:

- *Gaussiana*: Los elementos A_{ij} son generados a partir de una distribución normal con media cero y desviación estándar $1/M$. Se verifica $k \leq O(M/log(N/M))$[4].
- *Bernoulli*: Los elementos A_{ij} son generados a partir de valores ± 1 con probabilidad 0.5 de ocurrencia. Se cumple que $k \leq O(M/log(N/M))$.
- Ensamble de *Fourier*: A se obtiene como el resultado de elegir aleatoriamente submatrices de la transformada discreta de Fourier $F \in \mathbb{C}^{NxN}$. En este caso $k \leq O(M/log(N)^4)$.

Cabe aclarar que si se cumple $\delta_{2h} \leq 1$ entonces x es la solución única a (2.1). Con respecto a la relación de compresión N/M, el RIP también nos proporciona una respuesta. Según se muestra en [16], si $\delta_{2h} \leq 1/2$ la cantidad de submuestras M puede ser tan pequeña como se desee siempre que:

$$M \geq 0{,}28 \times k \times log(\frac{N}{k}) \tag{2.9}$$

Otra utilidad de (2.8) es que puede establecer si un método de recuperación de x a partir de A e y es estable [5]. Finalmente, se omite presentar la propiedad del Espacio Nulo (NSP) ya que el análisis del NSP dependerá de cada algoritmo de optimización según se requiera. Aún más, en el libro [5] se demuestra que si una matriz satisface el RIP, entonces también satisface el NSP. Por lo que, se infiere, cumplir con RIP es más fuerte que el NSP.

[3] Número de elementos de un vector.

[4] O significa *cota superior asintótica*. Se lee "es de orden".

Ahora bien, en 2.1.1 se mencionó que es conveniente representar el sistema de muestreo descomponiendo A en una base de representación Ψ, donde x sea dispersa, y un kernel de compresión Φ que genera la reducción dimensional de N muestras a M. Por lo tanto, es necesario analizar la relación entre ambas matrices. Para ello, se utiliza la *Coherencia Mutua* $\mu(\Phi, \Psi)$:

$$\mu(\Phi, \Psi) = \sqrt{N} \underset{1 \leq k,j \leq N}{max} |\langle \Phi_k, \Psi_j \rangle| \tag{2.10}$$

Siendo $\langle ., . \rangle$ el producto interno matricial. La expresión (2.10) se puede interpretar también como la correlación predominante entre dos elementos cualesquiera de Ψ y Φ. Si estas contienen elementos que están correlacionados, entonces $\mu(\Phi, \Psi)$ es un valor grande, en caso contrario, es pequeño. Cuando ambas matrices son idénticas, $\mu = \sqrt{N}$. Por otro lado, el máximo grado de incoherencia se obtiene cuando $\mu = 1$. La teoría de CS requiere que (2.10) sea lo menor posible, ya que con esto se logra que cada fila de la matriz de sensado adquiera la misma cantidad de información sobre x . Si Φ y Ψ son coherentes, entonces cada columna de Φ sólo nos dice cuánto de un elemento de la base Ψ_i esta presente en la señal. En ese caso, necesitaríamos las N muestras para reconstruir x [15].

Existe una gran ventaja en construir la matriz Φ de manera aleatoria, ya que de este modo nos independizamos de la base Ψ que utilicemos para representar la señal. Para comprender esto, si por ejemplo generamos Φ a partir de una distribución Gaussiana y Ψ es una base ortonormal como la DFT, entonces es posible demostrar que $A = \Phi\Psi$ también es Gaussiana y satisface el RIP con alta probabilidad [5] siempre que M cumpla con (2.9). Además, las matrices aleatorias son altamente incoherentes con cualquier base ortonormal normalizada y es bien conocido que en esas situaciones $\mu = \sqrt{2log(N)}$ con alta probabilidad [15]. Por ello, el proceso de adquisición de muestras deberá contar con una etapa de aleatorización que pueda ser representado con Ψ. En la Sección 2.2.2 se verá una arquitectura en hardware que incorpora estos conceptos y en la que se basa el diseño de este trabajo.

2.2. CS en el Dominio Analógico

2.2.1. Conversores Analógico-a-Información (AIC)

Clasicamente, los conversores A/D están basados en el teorema de Nyquist-Shannon, que garantiza la reconstrucción de una señal analógica limitada en banda cuando esta es muestreada uniformemente a una tasa de por lo menos el doble de su frecuencia máxima . Sin embargo, en 2.1.1 se analizó el concepto de *dispersión* y *compresibilidad*, implicando que en esos casos sólo una pequeña parte de la información es relevante. Asimismo, se mostró cómo la teoría de CS permite con un número limitado de muestras M obtener la información necesaria para la recuperación de señales compresibles. Entonces, es posible a partir de CS derivar un nuevo paradigma denominado conversión *analógico a información* [19] (Analog-to-Information en Inglés) como una alternativa a los A/D. La principal consecuencia de los AIC es que

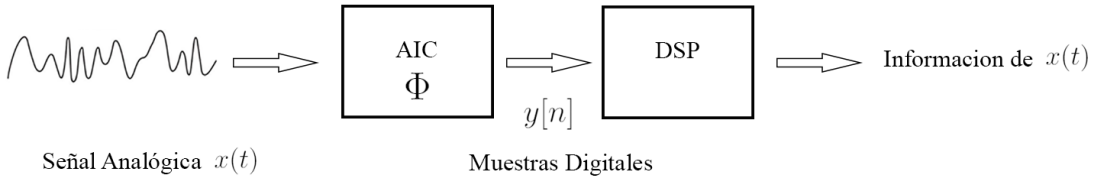


Figura 2.3: Diagrama en bloques general de un convertidor Analógico/Información.

será posible adquirir señales por debajo de la frecuencia de Nyquist. En este modelo, la extracción de las características de la señal a través del operador Φ reemplaza al muestreo convencional. Luego, es factible realizar en el dominio digital mediante técnicas DSP:

- Reconstruir la señal x.
- Aproximar la señal x.
- Obtener sus estadísticas más relevantes.
- Otras aplicaciones.

En la Figura 2.3 se ilustra de manera genérica el proceso de muestreo con un convertidor AIC. El operador Φ obtiene muestras lineales de la señal $x(t)$ continua en el tiempo. Ambas variables se corresponden con el modelo descripto en la ecuación (2.3). El procesamiento digital de la señal produce la salida deseada que, en el caso de este proyecto, será el vector de coeficientes α y/o la aproximación $\tilde{x}$.

Dos ejemplos de AIC implementados [15] son el muestreador no uniforme (NUS) construido por la compañía Northrop Grumman y el Modulador Aleatoreo Pre-Integrativo (RMPI) desarrollado por el Instituto de Tecnología de California en Estados Unidos. El primero es un ADC especializado que adquiere muestras en intervalos irregulares de tiempo siguiendo una grilla temporal que cumple con Nyquist. Esta diseñado para capturar señales de radar en la banda GSM y presenta un ancho de banda igual a 1,2 GHz. Es de particular interés la segunda arquitectura denominada RMPI, ya que permite utilizar ADC convencionales trabajando a frecuencias bajas y constantes. El prototipo fabricado de manera integrada con tecnología 90nm CMOS en [17] logra un ancho de banda igual a 2GHz mientras que adquiere a una tasa de 320 MSPS.

2.2.2. Modulador Pre-Integrador Aleatorio (RMPI)

Los conversores NUS al realizar muestreo aleatorio presentan una series de desventajas [18] de las cuales se pueden mencionar: el incremento en el Jitter [5] debido al uso no uniforme del clock de sincronismo, las situaciones en las que se debe adquirir dos muestras simultaneas en un periodo de tiempo pequeño (conversión de

[5]Desviación en la exactitud temporal en señales digitales.

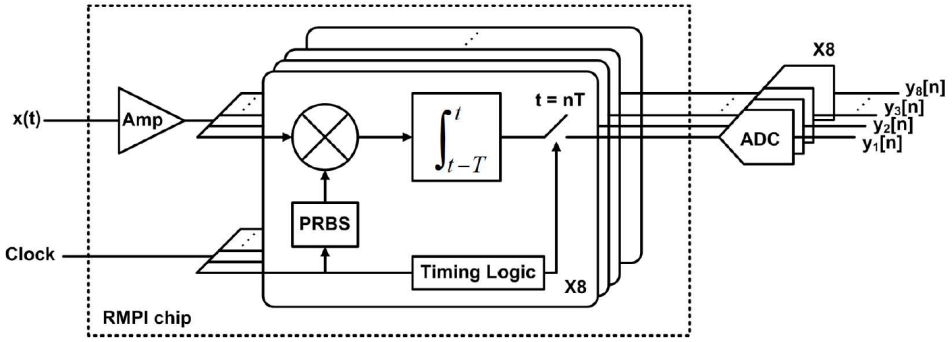


Figura 2.4: Diagrama en bloques del modulador aleatorio pre-integrador.

alta velocidad) y la dificultad en modelar la etapa de conversión con una matrix Φ. Debido a eso, en este trabajo se optó por el sistema RMPI.

El marco teórico para este tipo de sensores requiere un nuevo modelo para señales analógicas dispersas que permite diseñar un sistema que presente propiedades compatibles con las discutidas en la Sección 2.1.3. Dicho de otro modo, el kernel de compresión analógico Φ deberá cumplir con los requerimientos de RIP e Incoherencia Mutua con Ψ, como también generar proyecciones lineales de $x(t)$.

Juhwan Yoo en su tesis doctoral [20] desarrolló la teoría de funcionamiento RMPI junto con sus propiedades que serán presentadas a continuación. Sea $x(t) = \sum_{i=1}^{N} \alpha_i \Psi_i(t)$ la señal de interés que desde ahora asumimos tiene una representación k-dispersa $\alpha \in \mathbb{R}^N$ en el diccionario $\Psi = [\Psi_1(t), ..., \Psi_N(t)]$. Asimismo, definimos $x \in \mathbb{R}^N$ como la versión discreta de $x(t)$ muestreada a la frecuencia de Nyquist, por lo que se desprende $x = \Psi\alpha$, donde Ψ esta compuesta por las columnas $\Psi_i \in \mathbb{R}$ que representan a la correspondiente $\Psi_i(t)$ en instantes de tiempo discretos. El objetivo del conversor RMPI será recuperar una secuencia discreta de duración finita que represente a $x(t)$ durante un intervalo de tiempo de duración T_w. En primer lugar, para lograr la incoherencia requerida por CS se realiza la correlación en hardware de $x(t)$ con secuencias pseudo-aleatorias binarias (PRBS). Esto corresponde a un Φ igual a una matriz de Bernulli y como se enumeró en 2.1.3 satisface el RIP además de ser conveniente para desacoplar la elección de la base Ψ.

El diagrama general del RMPI se aprecia en la Figura 2.4. Este adquisidor está compuesto por un banco de N_{ch} canales en paralelo. Cada canal implementa lo que se denomina una *demodulación aleatoria* [18] y genera muestras incoherentes mediante correlación en el tiempo con la señal de entrada y una secuencia PRBS que llamamos $P_i(t)$ con $i \in \{1, ..., N_{ch}\}$. La correlación es implementada modulando $x(t)$ con $P_i(t)$ seguido de integrar y "descargar"[6] el resultado luego de un período $T_{int} = \frac{1}{f_s}\frac{M}{N}N_{ch}$.
El siguiente paso es modelar matemáticamente el funcionamiento de un RMPI ideal.

[6]La salida del integrador se resetea a un estado predefinido inmediatamente después de muestrear la salida.

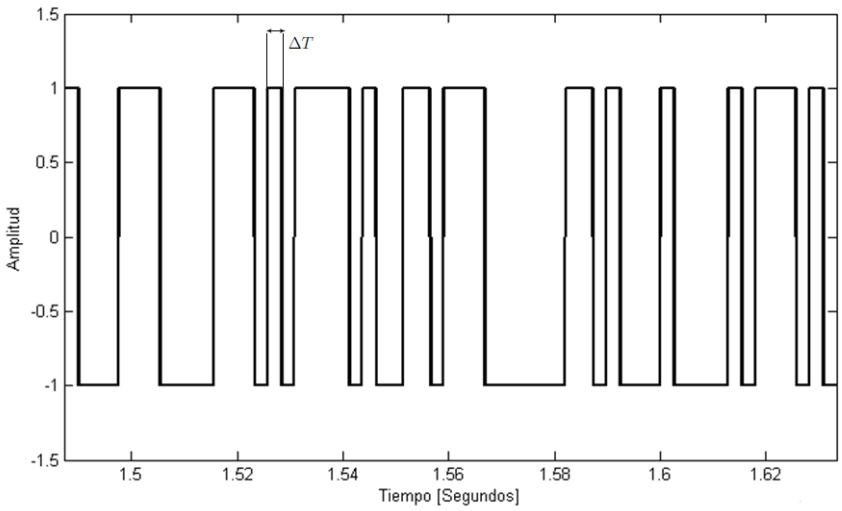


Figura 2.5: Secuencia Binaria Pseudo-Aleatoria con duración de pulso ΔT.

En principio, se analiza el comportamiento de un canal y luego será extrapolado a los demás trabajando en paralelo. Las muestras discretas a la salida del canal i pueden ser representadas con la operación:

$$y_i[m] = \int_{(m-1)T_{int}}^{mT_{int}} x(t)P_i(t)dt \tag{2.11}$$

Aunque en el Capítulo 3 se analizará el uso de secuencias caóticas para $P_i(t)$, a los efectos del estudio del funcionamiento asumimos secuencias PRBS:

$$P_i(t) = p_{i,n} \ , \ t \in [t_{n-1}, t_n) \ , \ t_n = n\Delta T \ , \ n \in \mathbb{Z}^+ \tag{2.12}$$

con $\Delta T = \frac{1}{f_c}$, siendo f_c la frecuencia de conmutación de las secuencias. Los coeficientes $p_{i,n} \in \{\pm 1\}$ generados de manera pseudo-aleatoria imitan el comportamiento dictado por una funcion densidad de probabilidad de Bernoulli discreta. La Figura 2.5 muestra la forma de onda de una PRBS descripta con la ecuación (2.12).

Ahora, es necesario replantear la expresión (2.11) de forma tal que represente un sistema (2.3) y pueda ser resuelto por un proceso de reconstrucción digital. Sustituyendo la ecuación (2.12) en (2.11) obtenemos para el canal i-esimo [20]:

$$y_i[m] = \sum_{n \in \Omega_m} p_{i,n} \int_{t_{n-1}}^{t_n} x(t)dt \tag{2.13}$$

donde Ω_m es un vector de longitud $N_{int} = T_{int}/\Delta T$, $N_{int} \in \mathbb{Z}^+$ que cumple $\Omega_m = \{n : (m-1)T_{int} < (n-1/2)\Delta T < mT_{int}\}$. También, sea el vector $P_i = [p_{i,1}, ..., p_{i,N}]$ el conjunto de valores discretos $\in \{\pm 1\}$. Se elige una discretización de la señal de la forma $x = [x_1, ..., x_N]$ tal que:

$$x_n = \int_{t_{n-1}}^{t_n} x(t)dt,\ n \in \{1, ..., N\} \tag{2.14}$$

Reescribimos (2.13):

$$y_i[m] = \sum_{n \in \Omega_m} p_{i,n} x_n = \langle p_i, x \rangle_{\Omega_m} = \langle \Phi_m, x \rangle \tag{2.15}$$

Siendo Φ_m el vector de la fila m-ésima de la matrix Φ_i (tener en cuenta que $y_i = \Phi_i x$). El operador $\langle , \rangle$ denota el producto escalar entre dos vectores. Con (2.15) podemos modelar matricialmente la acción que realiza el canal i para poder luego recuperar x según (2.5) mediante la composición de dos matrices. La primera de ellas, que representa la modulación del canal, es $D_i = diag\{P_i\} \in \mathbb{R}^{NxN}$:

$$D_i = \begin{bmatrix} p_{i,1} & 0 & 0 & \dots & 0 \\ 0 & p_{i,2} & 0 & \dots & 0 \\ 0 & 0 & p_{i,3} & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & 0 & p_{i,N} \end{bmatrix} \tag{2.16}$$

Después, la segunda matriz $H_i \in \mathbb{R}^{MxN}$ representa la respuesta al impulso $h_i(t)$, correspondiente al integrador por ventana i-ésimo en instantes $t = n\Delta T$, $n \in \{1, 2, ..., N\}$:

$$h_i(t) = u(t) - u(t - T_{int}) \tag{2.17}$$

$u(t)$ es la función escalón. H_i puede ser construida de forma tal que cada fila se corresponda con una ventana de integración. Los elementos que sean instantes dentro de dicha ventana equivalen a 1 y en caso contrario a 0. Con el fin de ilustrar un ejemplo, si tenemos una señal $x \in \mathbb{R}^{Nx1}$ con $N = 3N_{int}$ ($N_{int} = T_{int}/\Delta T$) y $N_{int} = 3$ la matriz que representa la integración descripta en (2.14) puede ser escrita como:

$$H_i = \begin{bmatrix} 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \end{bmatrix} \tag{2.18}$$

En este modelo se cumple $H_i \in \mathbb{R}^{LxN}$ con $L = \frac{M}{N_{ch}}$. Entonces, podemos expresar la función transferencia discreta del canal i de la siguiente manera:

$$y_i = \Phi_i x = H_i D_i x \tag{2.19}$$

$$H_i \in \mathbb{R}^{LxN},\ D_i \in \mathbb{R}^{NxN},\ x \in \mathbb{R}^{Nx1}$$

Siguendo con el ejemplo de la ecuación (2.18) y asumiendo una secuencia pseudo-aleatoria arbitraria que se repite cada 4 elementos $P_i = \{p_{i,1}\,, p_{i,2}\,, p_{i,3}\,, p_{i,4}\}$ la matriz $\Phi_i = H_i D_i$ resulta:

$$\Phi_i = \begin{bmatrix} p_{i,1} & p_{i,2} & p_{i,3} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & p_{i,4} & p_{i,1} & p_{i,2} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & p_{i,3} & p_{i,4} & p_{i,1} \end{bmatrix} \tag{2.20}$$

Por último, para un sistema de N_{ch} canales la matriz $\Phi \in \mathbb{R}^{MxN}$ se construye combinando las filas de cada Φ_i, es decir, se agrupan las que actúan en una misma ventana de integración T_{int} una debajo de la otra. La señal discreta $y \in \mathbb{R}^{Mx1}$ sub-muestreada surge combinando las muestras y_i de todos los canales en el siguiente orden:

$$y = \{y_1[0], ..., y_{N_{ch}}[0],\ , y_1[\frac{M}{N_{ch}} - 1], ..., y_{N_{ch}}[\frac{M}{N_{ch}} - 1]\} \tag{2.21}$$

Hasta aquí fueron presentados los fundamentos teóricos del Sensado Compresivo, junto con una arquitectura de adquisición analógica para basar el diseño del trabajo. Se mostró cómo obtener la matriz Φ del sensor para luego utilizarla en el proceso de reconstrucción digital. Es pertinente aclarar que, en el caso real, el RMPI presenta una serie de imperfecciones [20]. Entre ellas, se puede mencionar la degradación en la performance debido a la introducción de transferencias parásitas, las no linealidades en el mezclador y el Jitter en la señales de sincronismo. Más aún, no es posible implementar un integrador ideal con componentes electrónicos. La adquisición se realiza entonces con un filtro pasabajos. Será necesario estimar experimentalmente su respuesta al impulso y construir la matriz H con muestras de la misma para mejorar la calidad de reconstrucción. Sin embargo, estos aspectos vuelven complejo el diseño del optimizador y escapan al alcance de este trabajo, ya que se realizará la reconstrucción de la señal en un *System On Chip*. Debido a esto, de aquí en más se asume un RMPI ideal.

Ecuaciones de Diseño

A modo de resumen, se presenta los parámetros de diseño del RMPI:

- Número de canales $N_{ch} \in \mathbb{Z}^+$. Relación de compromiso entre performance y realización práctica.
- Ancho de banda AB deseado, en Hertz.
- Frecuencia de muestreo equivalente $f_s = 2AB$.
- Frecuencia de conmutación de las secuencias $f_c = f_s$.
- Cantidad de muestras N. En general, a mayor N mejor calidad en la reconstrucción a expensas de un mayor procesamiento.
- Ventana de observación $T_w = \frac{1}{f_s}N$.
- Dispersión $k \in \mathbb{Z}^+$ esperada de $x(t)$ en la base $\Psi(t)$.
- Relación de compresión $\frac{N}{M}$ con $N, M \in \mathbb{Z}^+$ y $M << N$. Tener en cuenta $M \geq 0{,}28k \times log(\frac{N}{k})$.
- Ventana de integración $T_{int} = \frac{1}{f_s}\frac{M}{N}N_{ch}$.
- Frecuencia de muestreo real $f_{real} = \frac{M}{N}f_s$.
- Frecuencia de muestreo de los ADC $f_{ADC} = \frac{f_{real}}{N_{ch}}$.
- Método de Discretización $x[n] = \int_{(n-1)T_s}^{nT_s} x(t)dt$, $T_s = \frac{1}{f_s}$.
- Transferencia equivalente $H(f) = e^{-j\pi f T_s} Sinc(fT_s)$. Se asume reconstrucción perfecta y RMPI ideal.
- La matriz Φ se construye con las secuencias P_i siguiendo (2.20).

Capítulo 3

Anteproyecto y Simulaciones

3.1. Modelo de Simulación

Previo a realizar el proyecto en el ZYNQ 7000, se creó un script en Matlab (Versión R2013a) con el fin de simular el sistema de la Figura 3.1. En el mismo se puede apreciar al adquisidor RMPI, que convierte la señal x a la versión comprimida y. Luego, le sigue la etapa de optimización cuyas entradas son las secuencias utilizadas en el muestreo, la matriz A de sensado y parámetros para su configuración. Finalmente, se produce el vector de coeficientes α y mediante multiplicación matricial con el diccionario Φ se genera $\hat{x}$. Todas las funciones, con excepción del sensor, serán implementadas en el SoC.

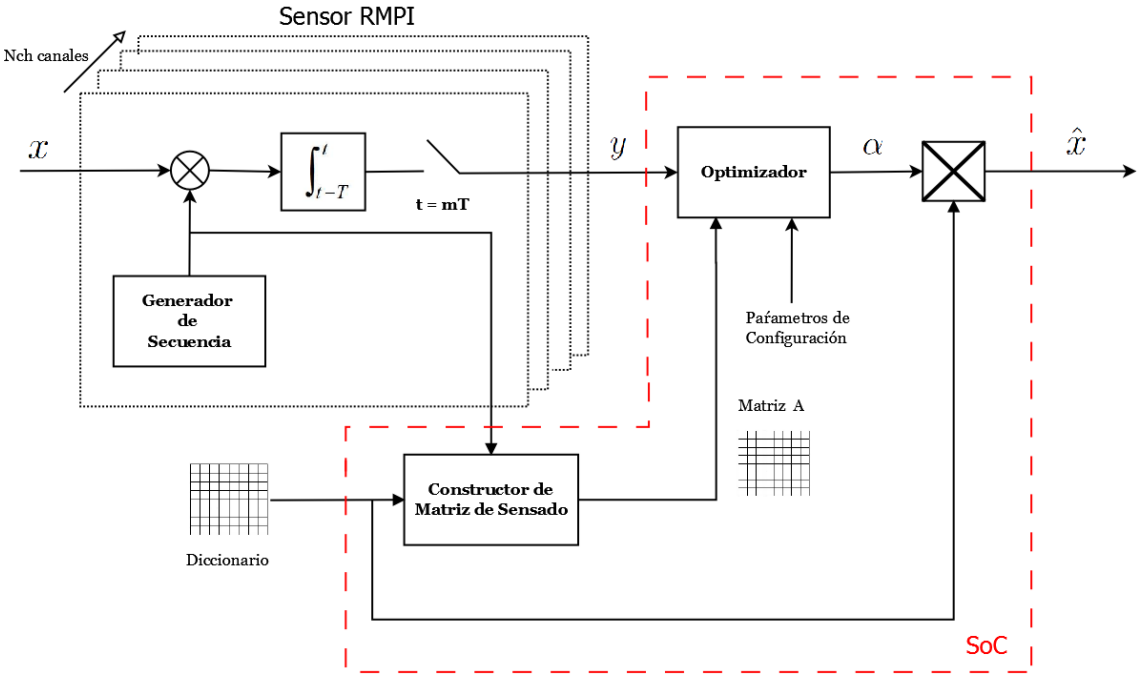


Figura 3.1: Sistema propuesto de Sensado Compresivo realizado en simulación.

Si bien la etapa de adquisición es analógica y la reconstrucción digital, todo el proceso es simulado de manera discreta a una frecuencia de muestreo 256 veces superior al ancho de banda de x. El script, que se encuentra en el Apéndice A.1.1, admite los siguientes parámetros de entrada:

- Señal de prueba x a reconstruir.
- Ancho de banda de la señal (en Hz).
- Longitud de x (N).
- Relación de compresión (N/M).
- Número de canales del RMPI(N_{ch}).
- Tipo de generador para las secuencia de mezclado.
- Matriz del Diccionario.
- Algoritmo de optimización.

Entonces, es posible poner bajo prueba cómo estas variables afectan al proceso de reconstrucción con CS, en términos de error, tiempo de ejecución y complejidad computacional. En la Figura 3.2 se ven más en detalle los pasos secuenciales que realiza el simulador. Primero, se construyen las matrices elementales H y D para generar a posteriori la matriz A con el diccionario ingresado. Con las mismas secuencias que constituyen a D, se multiplica a $x(t)$ con los $p_i(t)$ correspondientes al canal i-esimo y luego se aplica la función *intdump()* de Matlab R2013b para realizar la integración requerida en la modulación. Este proceso genera los vectores y_i y combinándolos entre sí según lo enuncia (2.21) se produce el vector de submuestras y. Antes de que se ejecute la función que realiza la reconstrucción, se invoca el comando *TIC()* que inicia un temporizador. Al finalizar el proceso de optimización, se usa *TOC()* y esto produce una estimación del tiempo requerido para cada algoritmo en recuperar a α. Post-multiplicando al diccionario Φ con α se produce $\hat{x}$. De aquí en más en el simulador se usan versiones sobremuestreadas de x y $\hat{x}$ con el fin de imitar sus formas de onda en el dominio continuo. El resultado obtenido se compara con x utilizando los cuantificadores *Similitud Temporal* y *Similitud Espectral* descriptos en la sección 3.2. Por último, se grafican las señales en los dominios de tiempo y frecuencia.

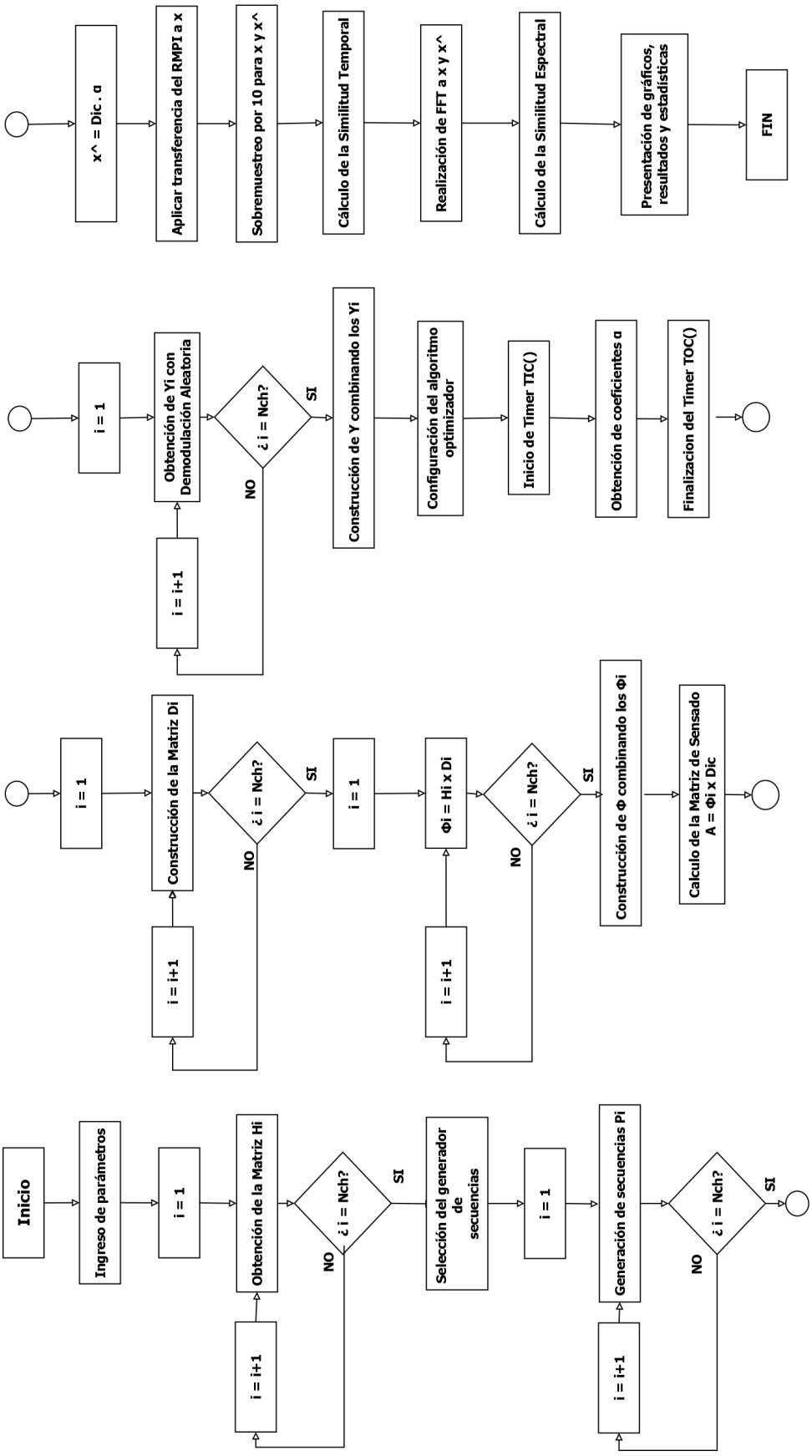


Figura 3.2: Flujo algorítmico en el Script de simulación del sistema CS.

3.1.1. Análisis Espectral del Convertidor RMPI

Un detalle no menor al momento de realizar una evaluación cuantitativa del rendimiento del sistema es analizar qué sucede en el dominio de la frecuencia con la señal de entrada. En el modelo matemático del RMPI se adoptó como método de discretización:

$$x[n] = \int_{T_s(n-1)}^{T_s n} x(t)dt \tag{3.1}$$

Por lo tanto, podemos re-expresar a (3.1) utilizando un tren de impulsos:

$$x_d(t) = \sum_{n=-\infty}^{\infty} \left[\int_{(n-1)T_s}^{nT_S} x(t)dt \right] \delta(t - nT_s) \tag{3.2}$$

La versión discreta de la señal llamada $x_d(t)$ presenta sus funciones $\delta(t)$ pesadas por la integral de retención y descarte (3.1). Siendo así, la expresión (3.2) describe a un sistema lineal $H(s) = \frac{1-e^{sT_s}}{s}$ seguido por un muestreador ideal. La transferencia $H(s)$ se obtiene al aplicar la transformada de Laplace a la respuesta al impulso del integrador, siendo esta $h(t) = u(t) - u(t - T_s)$. Después, si asumimos reconstrucción perfecta de $x(t)$ luego de la fase de optimización y una posterior conversión D/A, $\hat{x}(t) = h(t) * x(t)$. Expuesto de otra manera, $x(t)$ sufre modificaciones en fase y frecuencia conforme a lo representado por el diagrama de Bode de la Figura 3.3. Se puede apreciar que dicha transferencia posee un cero a la frecuencia de muestreo f_s. Por el teorema de Nyquist-Shannnon, se sabe que el ancho de banda de $\hat{x}(t)$ es $fs/2$. A esa frecuencia, la atenuación es -3,92dB (1,52 veces) y el desfasaje -90 grados. Si el espectro de $x(t)$ se encuentra por lo menos una década debajo de $fs/2$, entonces al ingresar al adquisidor sólo presenta un retardo igual a $Ts/2$. Sin embargo, en los casos donde esta atenuación en alta frecuencia impacte negativamente al rendimiento en una aplicación especifica, será necesario utilizar un posterior filtro de ecualización que compense lo producido por $H(s)$. En el caso de la simulación, primero se aplica la transferencia de la Figura 3.3 a $x(t)$ antes de realizar la comparación con $\hat{x}[n]$.

3.1.2. Generadores Caóticos

En el Capítulo 2 se mencionó que la aleatorización en el proceso de muestreo con CS permite satisfacer las propiedades *RIP* e *Incohehencia* necesarias para recuperar la señal x. Para el caso particular del RMPI, se usan las señales de mezclado $P(t)$. Estas señales se construyen con secuencias pseudoaleatorias de longitud finita. Es decir, luego del intervalo de tiempo $T = 2\ N\ 1/f_{Nyq}$ (N = cantidad de muestras de $x[n]$, f_{Nyq} = frecuencia de Nyquist) estas se repiten, exhibiendo un comportamiento periódico. Usualmente los PRNG son implementados en hardware. Por ejemplo, en [20] se utiliza un *Linear Shift-Register* para el RMPI. Sin embargo, los LFSR no son recomendados como PRNG ya que posee propiedades estadísticas [49] pobres. Otros generadores comunes son el *Lagged Fibonacci Generator*, pero en este caso se

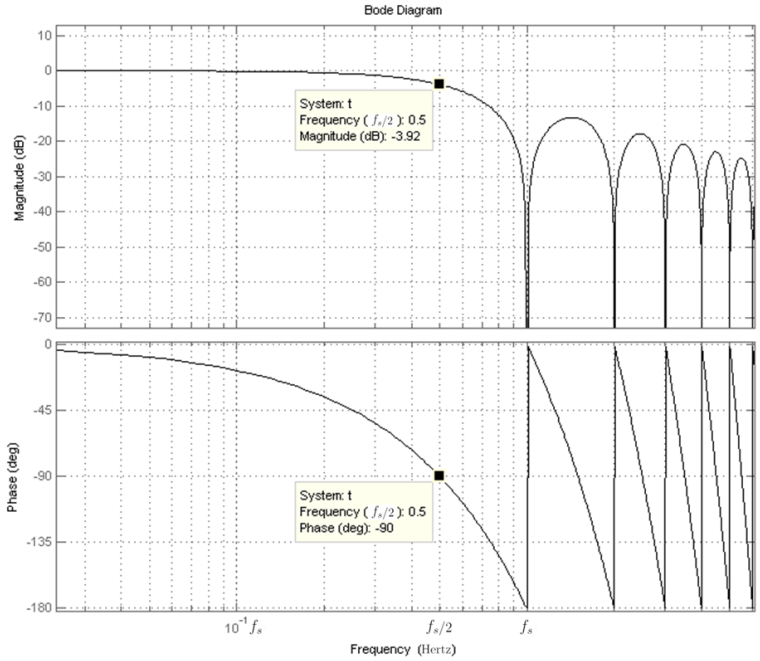


Figura 3.3: Respuesta en frecuencia del convertidor RMPI.

requiere una considerable cantidad de recursos en hardware para conseguir superar los test clásicos de aleatoriedad [50].

Uno de los objetivos de este trabajo es estudiar el reemplazo de las secuencias PRNG por aquellas creadas por mapas caóticos. La ventaja en usar caos es que dicho tipo de sistemas presentan un comportamiento complejo a partir de funciones iterativas sencillas. Por lo que, pueden imitar las propiedades de aleatoriedad bajo ciertas condiciones [51]. Aún más, existe un *principio de equivalencia* entre sistemas caóticos y estocásticos [52].

Sea una función $f : X \to X$ con X un espacio métrico [51] y $x_0 \in X$ el primer elemento de una secuencia y $x_1 = f(x_0)$ el segundo (en forma general $x_{n+1} = f(x_n)$). Dicha función iterativa, también conocida como mapa, es caótica si presenta las siguientes propiedades [51]:

1. Posee una colección *densa* (ver definición A.6 en [51]) de puntos con *órbitas periódicas*(Ver definición A.3 en [51]).

2. Es sensible a las condiciones iniciales. Esto es, puntos x_0 próximos pueden evolucionar rápidamente hacia distintos estados. A tal propiedad se la conoce como *efecto mariposa*.

3. Es topológicamente transitiva (ver definición A.7 en [51]).

Además del clásico generador con números pseudoaleatorios disponible en Matlab r2013b con la función *rand()*, pueden ser seleccionadas los siguientes mapas caóticos implementados para la construcción de secuencias en el simulador:

TWBM	TWBM2 (d=2)	TWBM3 (d=3)	TWBM4 (d=4)
0,010559404			
0,041791613	0,041791613		
0,160180296		0,160180296	
0,538090276	0,538090276		0,538090276
0,994196523			
0,023079185	0,023079185	0,023079185	
0,090186145			
0,328210418	0,328210418		0,328210418
0,881953358		0,881953358	
0,416446528	0,416446528		
0,972075269			
0,10857976	0,10857976	0,10857976	0,10857976
0,387160782			
0,949069244	0,949069244		
0,193347257		0,193347257	
0,62385638	0,62385638		0,62385638

Cuadro 3.1: Illustración del método de **Skipping**.

- LOG: Secuencia generada por el mapa caótico Logístico.
- TWBM: Secuencia generada por el mapa caótico Three-Way Bernoulli.
- FWTSM: Secuencia generada por el mapa caótico Four-Way Tailed Shift.
- TWBMd: Secuencia generada por la d-ésima iteracion del mapa caótico TWBM.
- FWTSMd: Secuencia generada por la d-ésima iteración del mapa caótico FWTSM.

Las secuencias generadas por cada mapa se transforman en un código binario por medio de la regla usual de dinámica simbólica ($[0,0{,}5) \rightarrow -1$, $[0{,}5,1] \rightarrow 1$). Expuesto de otra manera, si el valor generado, perteneciente al intervalo $[0,1]$, es menor a 0,5 entonces se mapea al valor -1. En cambio, si es mayor o igual, se mapea a 1.

Como puede verse, además de los mapas caóticos se consideraron versiones aleatorizadas de los mismos mediante un procedimiento de aleatorización conocido como skipping, [53]. Esto es debido a que las secuencias caóticas, a diferencia de las estocásticas, presentan estructuras internas que degradan su aleatoriedad. Este es el caso de las secuencias TWBMd y FWTSMd. El procedimiento de skipping consiste en saltar $(d-1)$ valores de la secuencia caótica para obtener las secuencias TWBMd y FWTSMd. En otras palabras, se emplea,en lugar del mapa original M, su $d-esima$ iteración (M^d). Esta técnica de aleatorización se utiliza exitosamente con mapas lineales por tramos en muchas aplicaciones [42]. En la Cuadro 3.1, se puede ver un ejemplo para distintos valores de d.

3.1.3. Algoritmos de Optimización

Anteriormente, se había mencionado que la mínimización de la norma l_1 en (2.5) provee la solución al problema del Sensado Compresivo y es computacionalmente viable. No obstante, se han realizados trabajos [54] en donde se investigan algoritmos alternativos para este fin que son más veloces ó presentan un mejor rendimiento en la reconstrucción. En esta sección se revelan los algoritmos de reconstrucción que estuvieron disponibles en la simulación. Todos ellos reconstruyen a x con un cierto error e. Con el propósito de evitar confusiones, en las siguientes técnicas se asume la recuperación de una señal genérica x *k-dispersa*. Se debe tener en cuenta, sin embargo, que en este proyecto se obtiene primero los coeficientes α en vez de x. Para mayor profundidad en los fundamentos matemáticos consultar la bibliografía específica [54–59].

Homotopy (*Soft-Thresholding*)

El primer método considerado por su simplicidad de implementación fue la *optimización por homotopía* [55]. Resuelve el problema de la norma l_1 mediante regulación por mínimos cuadrados usando la disminución de un parámetro regulador. Según sus autores, es un procedimiento eficiente ya que genera la solución en la iteración actual sólo una vez.

Procedure 1 Algoritmo Homotopy

Entradas: Matriz A, vector y, parámetro de reducción γ, dispersión k, cantidad máxima de iteraciones G.

1: Inicializar $x^{(i)} = 0, \lambda_1 = \|Ay\|_\infty$
2: **for** i=1,2,...G **do**
3: $x^{(i+1)} = P_{\lambda^{(i)}}(x^{(i)} - A(A^T x^{(i)} - y))$
4: **if** $\left\|x^{(i+1)}\right\|_0 > 2k$ **then**
5: **return** x_t
6: **end if**
7: $\lambda^{(i+1)} = \gamma \lambda_t$
8: **end for**
9: **return** $x^{(i+1)}$

El operador P_{λ_t} realiza el *Umbralamiento Suave* (*Soft-Thresholding* en Inglés) que se define como:

$$P_{\lambda^{(i)}}(\alpha) = \begin{cases} 0 & si|\alpha| \leq \lambda^{(i)} \\ sgn(\alpha)(|\alpha| - \lambda^{(i)}) & \text{Caso contrario} \end{cases} \tag{3.3}$$

Orthogonal Matching Pursuit

Los algoritmos *Matching Pursuit* son una clase de procedimientos que descomponen una señal en una expansión lineal de funciones que constituyen un diccionario [54]. En cada iteración, selecciona elementos del diccionario Φ que aproximen mejor a x de manera *voraz*. Los *algoritmo voraces* (*Greedy Algorithms* en Inglés) siempre seleccionan iterativamente una solución óptima local con la esperanza que esta decisión lleve a una solución óptima global [60]. Particularmente, *Orthogonal Matching Pursuit* emplea una proyección ortogonal dentro de todos los átomos (elementos constituyentes del diccionario) seleccionados.

Procedure 2 Orthogonal Matching Pursuit

Entradas: Matriz A, vector y, dispersión k.

Salidas: La aproximación $\hat{x}$, un conjunto Λ_k que contiene la posición de los elementos no nulos de $\hat{x}$, una aproximación de y llamada a_k y el residuo $r = y - a_k$.

1: $r^{(0)} \leftarrow y$
2: $\Lambda^{(0)} \leftarrow \varnothing$
3: **for** i=1,...,k **do**
4: $\quad \lambda^{(i)} \leftarrow \arg\max_{j=1,\ldots,n} \left| \left\langle r^{(i-1)}, A_j \right\rangle \right|$
5: $\quad \Lambda^{(i)} \leftarrow \Lambda^{(i-1)} \cup \lambda^i$
6: $\quad A^{(i)} \leftarrow \left[A^{(i-1)} A_{\lambda^{(i)}} \right]$
7: $\quad x^{(i)} \leftarrow \left\| y - A^{(i)} \hat{x} \right\|_2$
8: $\quad a^{(i)} \leftarrow A^{(i)} x^{(i)}$
9: $\quad r \leftarrow y - a^{(i)}$
10: **end for**
11: $\hat{x} \leftarrow x^{(k)}$
12: **return** $\hat{x}, \Lambda^{(k)}, a^{(k)}, r^{(k)}$

Iterative Hard Thresholding

La técnica del *Umbralamiento Duro Iterativo* [56,57] (*Iterative Hard Thresholding* en Inglés) emplea un procedimiento sencillo para resolver el problema de minimización:

$$\min_x \|y - Ax\|_2^2 \text{ sujeto a } \|x\|_0 \leq k \tag{3.4}$$

Se comienza con la primera solución $\hat{x}^{(0)} = 0$. Luego, en la i-ésima iteración, $\hat{x}$ se obtiene como:

$$\hat{x}^{(i+1)} = H_k(\hat{x}^{(i)} + \mu A^T(y - A\hat{x}^{(i)})) \tag{3.5}$$

Donde H_k es un operador no lineal que reemplaza a todos los valores, excepto los k mayores en valor absoluto, por cero. A su vez, $0 \leq \mu \leq 1$ se denomina *tamaño de paso*. La convergencia del algoritmo esta demostrada bajo la hipótesis que $\|A\|_2 < 1$. En este caso, (3.7) converge a un mínimo local de (3.4). Como puede observarse, se requiere realizar operaciones con la matriz A y su transpuesta; junto con dos sumas vectoriales. Por otro lado, H_k involucra un ordenamiento de los valores de $x^{(i)} + \mu A^T(y - A\hat{x}^{(i)})$ en magnitud. En particular, la degradación del rendimiento

es linealmente dependiente con la SNR de x; por lo que admite inmunidad frente al ruido dentro de ciertos márgenes.

Accelerated Iterative Hard Thresholding

La implementación estándar del IHT presenta dos problemas: el paso μ debe ser elegido adecuadamente para evitar la inestabilidad del método y la velocidad de convergencia sólo puede ser lineal [61]. En el *Umbralamiento Duro Iterativo Acelerado* (*Accelerated Iterative Hard Thresholding* en Inglés), la variable μ se obtiene adaptativamente en cada iteración:

$$\mu = \frac{\left\| A^T_{\Gamma^{(i)}}(y - Ax^{(i)}) \right\|_2^2}{\left\| A_{\Gamma^{(i)}} A^T (y - Ax^{(i)}) \right\|_2^2} \tag{3.6}$$

El soporte $\Gamma^{(i)}$ es conjunto de índices para los elementos no nulos de $x^{(i)}$. Además, es posible aumentar la velocidad del algoritmo calculando primero una estimación de $\hat{x}$ llamada $\tilde{x}$:

$$\tilde{x}^{(i+1)} = H_k(\hat{x}^{(i)} + \mu A^T(y - A\hat{x}^{(i)})) \tag{3.7}$$

En vez de seguir el proceso de iteración con $\tilde{x}^{(i+1)}$, se encuentra a $\hat{x}^{(i+1)}$ usando las siguientes premisas:

1. $\hat{x}^{(i+1)}$ tiene k elementos no nulos.
2. $\hat{x}^{(i+1)}$ satisface $\left\| y - A\hat{x}^{(i+1)} \right\|_2 \leq \left\| y - A\tilde{x}^{(i+1)} \right\|_2$

Por consiguiente, considerando que la matriz $A_{\Gamma^{(i)}}$ contiene las columnas que no se encuentran en el conjunto $\Gamma^{(i)}$ removidas y $\tilde{x}_{\Gamma^{(i)}}$ definida de igual forma; entonces es necesario resolver el subproblema de optimización $\min \|y - A_{\Gamma^{(i)}}\tilde{x}\|$. La solución intermedia se encuentra iterando (3.8):

$$\tilde{x}^{(j+1)}_{\Gamma^{(i)}} = \tilde{x}^{(j)}_{\Gamma^{(i)}} + \mu A^T_{\Gamma^{(i)}}(y - A_{\Gamma^{(i)}}\tilde{x}^{(i)}_{\Gamma^{(i)}}) \tag{3.8}$$

Luego, cuando se alcanza un criterio para detener las iteraciones, $\hat{x}^{(i+1)}_{\Gamma^{(i)}} = \tilde{x}^{(j+1)}_{\Gamma^{(i)}}$. Es decir, $\hat{x}^{(i+1)}$ se construye inicializándolo con ceros y después en las posiciones que indique el conjunto $\Gamma^{(i)}$ se colocan los elementos de $\tilde{x}^{(j+1)}_{\Gamma^{(i)}}$.

Gradient Pursuit

Dentro de la categoría *Matching Pursuit* se encuentran los algoritmos de *Persecución de Gradiente* (*GP* en Inglés). En lugar de actualizar al vector $\hat{x}$ como el producto escalar del residuo y un átomo (como en el caso de OMP), el cálculo se realiza en una dirección vectorial particular [58]. De este modo, una iteración de GP es igual a:

$$\hat{x}^{(i)} = \hat{x}^{(i-1)} + \alpha^{(i)}\mathbf{d}^{(i)} \tag{3.9}$$

Siendo $\mathbf{d}^{(i)}$ la dirección de actualización. Una vez que dicha dirección es seleccionada, el paso óptimo $\alpha^{(i)}$ en términos de minimizar el error $\left\| y - A\hat{x}^{(i)} \right\|_2^2$ es:

$$\alpha^{(i)} = \frac{\langle \mathbf{r}^{(i)}, A_{\Gamma^{(i)}}\mathbf{d}^{(i)} \rangle}{\left\| A_{\Gamma^{(i)}} d^{(i)} \right\|_2^2} \tag{3.10}$$

Conjugated Gradient Pursuit

Otra técnica popular para resolución iterativa de CS es la *Persecusión del Gradiente Conjugado* (*CGP* en Inglés). De manera muy resumida, sea $\phi(x) = \frac{1}{2}x^T Gx - b^T$ una función de costo a minimizar, equivalente a resolver $b = Gx$ para x. El método CGP elige una dirección de actualización $\mathbf{d}^{(i)}$ que es *G-conjugada* a todas las direcciones utilizadas previamente:

$$\mathbf{d}^{(i)} G \mathbf{d}^{(j)} = 0, \forall j < i \tag{3.11}$$

En el caso de CS, $G = A^T_{\Gamma^{(i)}} A_{\Gamma^{(i)}}$, implicando que se esta minimizando la expresión:

$$\| y - A_{\Gamma^{(i)}} \tilde{x}_{\Gamma^{(i)}} \|_2^2 \tag{3.12}$$

Sea $\mathbf{D}^{(j)}$ una matriz cuya columnas sean las direcciones de las primeras j direcciones y $\mathbf{g}^{(j)}$ el gradiente de la función de costo en la iteración j. La nueva dirección en la iteración l esta determinada por:

$$\mathbf{d}^{(l)} = \mathbf{g}^{(k)} + \mathbf{D}^{(l-1)} \mathbf{b} \tag{3.13}$$

Con

$$\mathbf{b} = -\left(\left(\mathbf{D}^{(l-1)} \right)^T \mathbf{G} \mathbf{D}^{(l-1)} \right)^{-1} \left(\left(\mathbf{D}^{(l-1)} \right)^T \mathbf{G} \mathbf{g}^{(l-1)} \right) \tag{3.14}$$

A continuación se muestras los pasos más detallados de GP y CGP para ayudar a su compresión (Algoritmo 3).

Compressive Sampling Matched Pursuit (CoSaMP)

Por último, se evaluó la extensión del OMP denominada CoSaMP o (*COmpressive SAmpling Matching Pursuit)* en Inglés) [59]. El Algoritmo 4 muestra sus pasos secuenciales.

Procedure 3 Gradient Pursuit

Entradas: Matriz A, vector y.
Salidas: La aproximación $\hat{x}$.

$x^{(0)} \leftarrow 0$
$r^{(0)} \leftarrow 0$
$\Gamma \leftarrow 0$
$i \leftarrow 0$
while Condición para detenerse no cumplida **do**
    $i \leftarrow i + 1$
    $g^{(i)} \leftarrow A^T r^{(i-1)}$
    $\lambda^i \leftarrow \arg\max_j |g_j^{(i)}|$
    $\Gamma^i \leftarrow \Gamma^{i-1} \cup \lambda^i$
    if Gradiente Pursuit **then**
        $d^{(i)} \leftarrow A_{\Gamma^{(i)}}(y - A_{\Gamma^{(i)}} x_{\Gamma^{(i)}}^{(i-1)})$
    else if Conjugated Gradiente Pursuit **then**
        $g^{(i)} \leftarrow A_{\Gamma^{(i)}}(y - A_{\Gamma^{(i)}} x_{\Gamma^{(i)}}^{(i-1)})$
        $\mathbf{b} = -\Big(\big(\mathbf{D}^{(l-1)}\big)^T \mathbf{G}\mathbf{D}^{(l-1)}\Big)^{-1}\Big(\big(\mathbf{D}^{(l-1)}\big)^T \mathbf{G}\mathbf{g}^{(l-1)}\Big)$
        $\mathbf{d}^{(l)} = \mathbf{g}^{(k)} + \mathbf{D}^{(l-1)}\mathbf{b}$
    end if

    $\alpha^{(i)} \leftarrow \frac{\langle \mathbf{r}^{(i)}, A_{\Gamma^{(i)}} \mathbf{d}^{(i)} \rangle}{\left\| A_{\Gamma^{(i)}} d^{(i)} \right\|_2^2}$
    $x^{(i)} \leftarrow x^{(i-1)} + \alpha^{(i)} \mathbf{d}^{(i)}$
    $\mathbf{r}^{(i)} \leftarrow \mathbf{r}^{(i-1)} - \alpha^{(i)} A_{\Gamma^{(i)}} \mathbf{d}^{(i)}$
end while
$\hat{x} \leftarrow x^i$
return $\hat{x}$

Procedure 4 CoSaMP

Entradas: La dispersión k de x, el vector y y la matriz A.
Salidas: La aproximación $\hat{x}$.

$x^{(0)} \leftarrow 0$
$v \leftarrow y$
$i \leftarrow 0$
while Condición para detenerse no cumplida **do**
    $i \leftarrow i + 1$
    $z \leftarrow A^T v$
    $\Omega \leftarrow supp_{2k}(z)$ (soporte de los 2k elementos mayores)
    $\Gamma \leftarrow \Omega \cup supp\big(x^{(i-1)}\big)$
    $\tilde{x} \leftarrow \arg\min_{\tilde{x}: supp(\tilde{x}) = \Gamma} \|A\tilde{x} - y\|_2$
    $x^{(i)} \leftarrow H_k \hat{x}$
    $v \leftarrow y - Ax^{(i)}$
end while
$\hat{x} \leftarrow x^i$
return $\hat{x}$

3.2. Resultados

Para evaluar el rendimiento que presentan las distintas configuraciones del sistema simulado se utilizaron tres cuantificadores. Dos de ellos determinan la similitud de la señal recuperada con la original, en el tiempo y en frecuencia.

- Similitud temporal:

$$< S_t >= \frac{\| x - \hat{x}\|_2}{\| x\|_2}$$

- Similitud espectral:

$$< S_e >= \frac{\| \, |fft(x)| - |fft(\hat{x})|\|_2}{\| \, |fft(x)|\|_2} \tag{3.15}$$

También, se obtienen las varianzas $Var(S_e)$ y $Var(S_t)$ de estos parámetros; con el fin de realizar una comparación en términos de precisión. Cuanto menor sea el valor de estos cuantificadores, entonces mejor es el rendimiento del reconstructor. El tercer cuantificador es el tiempo medio requerido por los algoritmos de optimización $< t_m >$.

La señal de prueba utilizada en los siguientes ensayos se muestra en la Figura 3.4. Según se observa, la misma está constituida por la suma de dos señales AM; una con frecuencia de portadora $f_c = 1$GHz y otra con $f_c/2 = 500$MHz:

$$f = \big[sin(2\pi f_c/2.t) + sin(2\pi f_c t)\big](1 + 0{,}7cos(2\pi \frac{3e8}{40} t)) \tag{3.16}$$

Los diccionarios para representar a (3.16) son la DFT y DCT de tipo dos [62]. En ambos casos, el diccionario se construye en Matlab aplicado la transformada que corresponda a la matriz identidad. A su vez, se adoptó como factor de dispersión $k = 12$. Salvo que se especifique de manera contraria, se utiliza sólo un canal ($N_{ch} = 1$). La plataforma donde se realizaron las pruebas fue una computadora personal con sistema operativo Windows 7, 8 GB de RAM y procesador Intel i7 2600k de 3,80 GHz con cuatro núcleos. Por último, las herramientas *l1magic* y *CoSaMP and OMP for sparse recovery* [21,22] permitieron evaluar distintos algoritmos de optimización.

3.2.1. Comparación de Optimizadores

Primero, con una longitud de señal $N = 1024$ y una relación de compresión $N/M = 4$ fija se procedió a evaluar a los ocho algoritmos de reconstrucción. La secuencias utilizadas en cada uno de los casos es pseudoaleatoria. Los resultados de correr 200 veces cada optimizador se presenta en la Cuadro 3.2. El diccionario utilizado fue la DCT-II. De manera complementaria, se muestra en la Figura 3.5 la recuperada producida en una iteración de cada algoritmo.

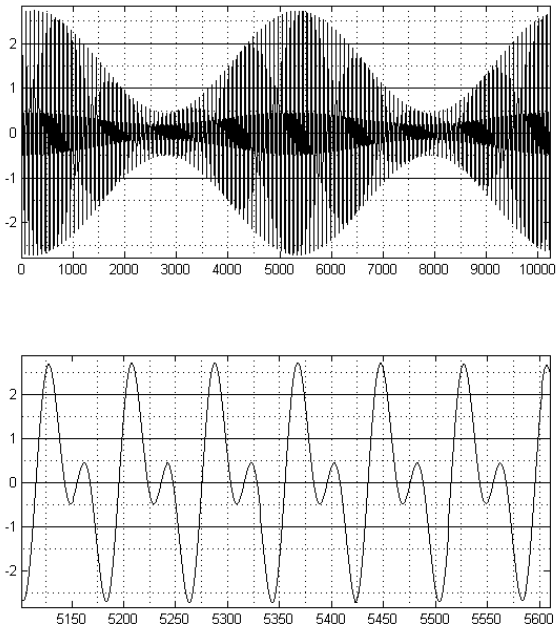
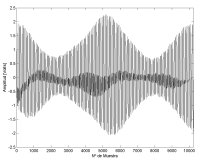


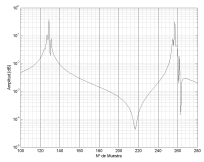
Figura 3.4: Señal de prueba utilizada en las simulaciones (Arriba) y su forma de onda en detalle (Abajo).

Algoritmo	$< S_t >$	$Var(S_t)$	$< S_e >$	$Var(S_e)$	t_m[s]	$Var(t)[s^2]$
Homotopy	0,0957	0,0277	0,0837	0,0213	0,0017	2,110 10^{-5}
OMP	0,0740	0,0171	0,0645	0,0131	0,0033	7,053 10^{-5}
IHT	0,0877	0,0232	0,0710	0,0153	0.0028	4,025 10^{-5}
AIHT	0,0780	0,0211	0,0636	0,0140	0,0006	0,500 10^{-5}
GP	0,0864	0,0225	0,0698	0,0147	0,0011	2,310 10^{-5}
CGP	0,0874	0,0231	0,0709	0,0152	0,0010	2,130 10^{-5}
CoSaMP	0,0852	0,0219	0,0682	0,0140	0,0017	3,727 10^{-5}

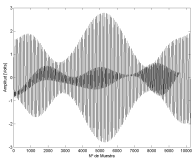
Cuadro 3.2: Resultados de simulación para los distintos algoritmos de Prueba (N = 1024, M = 256).



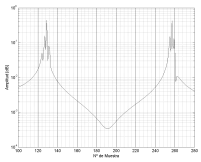
(a) Homotopy (Tiempo)



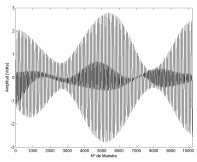
(b) Homotopy (Frecuencia)



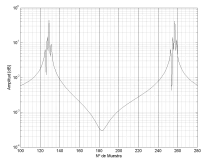
(c) Gradient Pursuit (Tiempo)



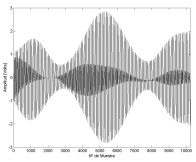
(d) Gradient Pursuit (Frecuencia)



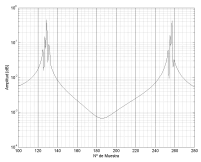
(e) Conjugated GP (Tiempo)



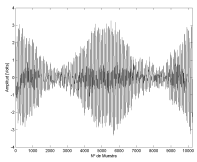
(f) Conjugated GP (Frecuencia)



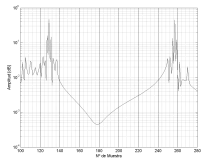
(g) IHT (Tiempo)



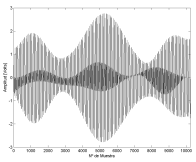
(h) IHT (Frecuencia)



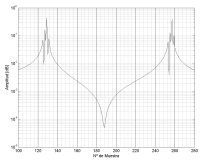
(i) OMP (Tiempo)



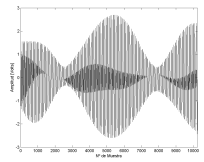
(j) OMP (Frecuencia)



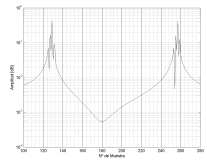
(k) CoSaMP (Tiempo)



(l) CoSaMP (Frecuencia)



(m) AITH (Tiempo)



(n) AITH (Frecuencia)

Figura 3.5: Resultados para la obtención de $\tilde{x}$ con los distintos optimizadores, en el dominio del tiempo y en frecuencia.

3.2.2. Robustez Frente al Ruido

Después, empleando los mismos parámetros que la simulación anterior, analizó el efecto sobre la recuperación de la señal cuando ésta se encuentra contaminada con ruido blanco uniformente distribuido en amplitud (intervalo $[-0{,}5, 0{,}5]$). La señal de prueba ahora es:

$$x_n(t) = x(t) + f.n(t) \tag{3.17}$$

En (3.17) se suma el término de ruido $n(t)$ multiplicado por un factor constante $f \in [0, 1]$. Hay que aclarar que no se informa el tiempo de procesamiento ya que es más importante analizar la modificación en la calidad de la reconstrucción. Los Cuadros 3.3, 3.4 y 3.5 muestra los resultados para $f = 0{,}01$, $f = 0{,}05$ y $f = 0{,}2$ respectivamente. Se encontró que los Algoritmos *Homotopy*, *Gradient Pursuit* y *Conjugated Gradient Pursuit* no convergían cuando se introducía ruido. Es por ello que de aquí en más se descartan del análisis. Además, ahora se utiliza en las simulaciones restantes la FFT como base de representación[1].

Algoritmo	$< S_t >$	$Var(S_t)$	$< S_e >$	$Var(S_e)$
Homotopy	N/A	N/A	N/A	N/A
OMP	1,6610	0,0022	0,0987	$8{,}2\ 10^{-4}$
IHT	0,0613	$3{,}5\ 10^{-7}$	0,0449	$3{,}11\ 10^{-7}$
AIHT	0,0615	$4{,}62\ 10^{-7}$	0,0451	$5{,}6\ 10^{-7}$
GP	N/A	N/A	N/A	N/A
CGP	N/A	N/A	N/A	N/A
CoSaMP	0,0602	$2{,}1\ 10^{-8}$	0,0442	$2{,}5\ 10^{-8}$

Cuadro 3.3: Resultados de simulación para N = 1024, M = 256 y ruido blanco con $f = 0{,}01$.

Algoritmo	$< S_t >$	$Var(S_t)$	$< S_e >$	$Var(S_e)$
Homotopy	N/A	N/A	N/A	N/A
OMP	1,662	0,0023	0,1376	$5\ 10^{-4}$
IHT	0,0665	$2{,}3\ 10^{-4}$	0,0495	$2{,}8\ 10^{-8}$
AIHT	0,0645	$3{,}94\ 10^{-6}$	0,0472	$2{,}16\ 10^{-6}$
GP	N/A	N/A	N/A	N/A
CGP	N/A	N/A	N/A	N/A
CoSaMP	0,0606	$8{,}12\ 10^{-8}$	0,0444	$\$7.94\ 10^{-8}$

Cuadro 3.4: Resultados de simulación para N = 1024, M = 256 y ruido blanco con $f = 0{,}05$.

[1]Esto es así porque el algoritmo Homotopy, debido a sus funciones internas, requería operar con números reales. Cuando se descartó su posibilidad de uso, se emplearon números complejos.

Algoritmo	$< S_t >$	$Var(S_t)$	$< S_e >$	$Var(S_e)$
Homotopy	N/A	N/A	N/A	N/A
OMP	1,672	0,0031	0,2673	0,0025
IHT	0,277	4,3 10^{-5}	0,2724	4,35 10^{-5}
AIHT	0,2755	5,43 10^{-5}	0,2708	5,53 10^{-5}
GP	N/A	N/A	N/A	N/A
CGP	N/A	N/A	N/A	N/A
CoSaMP	,274	5,17 10^{-6}	0,2704	5,5 10^{-6}

Cuadro 3.5: Resultados de simulación para N = 1024, M = 256 y ruido blanco con $f = 0{,}2$.

3.2.3. Capacidad de Compresión

Ejecutando 50 corridas, se repitieron los ensayos realizados en la Sección 3.2.1. No obstante, esta vez se varió la relación de compresión de 4 a 8, 16 y 32. Los resultados se reportan en los Cuadros 3.6, 3.7 y 3.8.

Algoritmo	$< S_t >$	$Var(S_t)$	$< S_e >$	$Var(S_e)$	t_m[s]	$Var(t)[s^2]$
Homotopy	N/A	N/A	N/A	N/A	N/A	N/A
OMP	1,6686	0,0048	0,2673	0,0034	0,0113	1,37 10^{-4}
IHT	0,0627	2,1 10^{-6}	0,0459	2,9 10^{-6}	0,015	1,04 10^{-4}
AIHT	0,0631	3,25 10^{-6}	0,0460	1,9 10^{-6}	0,0028	9,5 10^{-6}
GP	N/A	N/A	N/A	N/A	N/A	N/A
CGP	N/A	N/A	N/A	N/A	N/A	N/A
CoSaMP	0,0608	2,9 10^{-7}	0,0446	2,5 10^{-7}	0,0081	1,58 10^{-4}

Cuadro 3.6: Rendimientos de los optimizadores para N = 1024, M = 128.

Algoritmo	$< S_t >$	$Var(S_t)$	$< S_e >$	$Var(S_e)$	t_m[s]	$Var(t)[s^2]$
Homotopy	N/A	N/A	N/A	N/A	N/A	N/A
OMP	1,6504	0,0062	0,2194	0,0109	0,0085	2,97 10^{-4}
IHT	0,1375	0,0287	0,1198	0,0287	0,0251	4,47 10^{-4}
AIHT	0,1462	0,0271	0,1266	0,0035	0,0028	8,6 10^{-6}
GP	N/A	N/A	N/A	N/A	N/A	N/A
CGP	N/A	N/A	N/A	N/A	N/A	N/A
CoSaMP	0.0897	0.0092	0.0692	0.0078	0.1044	0.0531

Cuadro 3.7: Rendimientos de los optimizadores para N = 1024, M = 64.

Algoritmo	$< S_t >$	$Var(S_t)$	$< S_e >$	$Var(S_e)$	t_m[s]	$Var(t)[s^2]$
Homotopy	N/A	N/A	N/A	N/A	N/A	N/A
OMP	1,4514	0,0501	0,6737	0,088	0,0056	$1\ 10^{-4}$
IHT	0,7492	0,0303	0,7222	0,0336	0,0755	0,0042
AIHT	0,7063	0,07	0,6852	0,0737	0,0192	0,0045
GP	N/A	N/A	N/A	N/A	N/A	N/A
CGP	N/A	N/A	N/A	N/A	N/A	N/A
CoSaMP	0,6817	0,0567	0,6604	0,0589	0,0147	0,0024

Cuadro 3.8: Rendimientos de los optimizadores para N = 1024, M = 32.

3.3. Evaluación de las Alternativas

Analizando el Cuadro 3.2 se encuentra que, en promedio, el reconstructor que genera la mejor *Similitud Temporal* ($< S_t >$) entre la señal original y la recuperada es el *Orthogonal Matching Pursuit*, seguido del *Accelerated Iterative Hard Thresholding* y *CoSaMP*. De igual forma, con la *Similitud Espectral* (S_e) se obtuvieron idénticos resultados. En relación al *Tiempo de Reconstrucción* ($< t_m >$), el algoritmo *AIHT* fue el más rápido en obtener una respuesta, seguidos del *Conjugated Gradient Pursuit* y *Gradient Pursuit*. No existen diferencias significativas en las varianzas de estos tres casos.

Cuando se procedió a contaminar a las muestras de entrada con ruido, se pudo apreciar que en el *OMP* se deteriora notablemente el rendimiento de reconstrucción. Mientras que en *IHT AIHT* y *CoSaMP* se observa una menor degradación frente al agregado de ruido. En estos últimos casos, el rendimiento empeora de manera directamente proporcional con el factor f.

Luego, al variar el tamaño de submuestras M, se exige un mayor procesamiento de los optimizadores. Recordando la expresión (2.9) se establece que el límite inferior teórico para M son 7 muestras. Como puede observarse, la calidad de reconstrucción empeora al aumentar el radio de compresión en el dominio del tiempo y frecuencia. No obstante, el rendimiento en el tiempo requerido no muestra una tendencia clara en función de M. En estas situaciones, *CoSaMP* muestra los mejores resultados, seguido de *AIHT*. Aunque, la diferencia porcentual entre sus resultados no supera el 3 %.

Usando como criterio la calidad de reconstrucción, *CoSaMP* es la mejor alternativa para implementar en el SoC. Pero, realizar en cambio el *AIHT* presenta una serie de beneficios. El primero de ellos, más evidente, es la reducción del tiempo en recuperar a $x[n]$. Es decir, si se comparan estos dos optimizadores observando los Cuadros, puede discernirse que el tiempo se reduce casi en cuatro veces. La segunda ventaja es que *AITH* solo requiere multiplicaciones vectoriales y ordenamiento de elementos, en tanto *CoSAMP* necesita realizar una inversión matricial (siendo una operación computacionalmente intensiva) y una búsqueda lineal para encontrar la solución al término [59]. Por estas razones, se escoge realizar en el Zynq 7000 al

algoritmo AIHT; ya que es razonable sacrificar un poco precisión en la recuperación de la señal a cambio de una reducción considerable en el tiempo de reconstrucción.

Comparando *IHT* con *AIHT*, la única diferencia apreciable es en el tiempo de ejecución, verificándose entonces la eficacia del proceso de aceleración en *AIHT* mediante la sub-optimización. Teniendo en cuenta a los cuantificadores $< S_t >$ y $< S_m >$, no hay mayor discrepancia entre ambos algoritmos.

3.3.1. Efecto de Aumentar el Submuestreo

Con respecto a la relación de compresión, utilizando la señal (3.16), diccionario de representación a la DFT, la secuencia de mezclado generada con un mapa *Logístifco*, agregado de ruido blanco con factor $f = 0{,}1$ y el optimizador con AITH se ejecutó 50 veces la simulación con $N = 1024$ para $M = 256$, $M = 128$, $M = 64$ y $M = 32$. Se definió de manera arbitraria como caso exitoso cuando la *Similitud Espectral* no superaba el valor 0,085 y como fracaso en caso contrario. Los resultados producidos se muestran en el Cuadro 3.9.

Relación de Compresión	**Porcentaje de Casos Exitosos**
4	100 %
8	100 %
16	75 %
32	2 %

Cuadro 3.9: Resultados de simulación para diferente relación de compresión con el algoritmo AIHT.

La taza de éxito es aceptable hasta $M = 64$ ($N/M = 16$). Luego, para $N/M \geq 32$ el sistema no logra recuperar a la señal. Por lo tanto, para maximizar la compresión en la adquisición de $x(t)$ se escoge utilizar $N = 1024$ y $M = 64$. Se verificó también que al aumentar N el porcentaje de casos exitosos es del 100 %, bajo las mismas condiciones, y al ser $N = 512$ o menor el reconstructor vuelve a fallar.

3.3.2. Efecto de Aumentar el Número de Canales

Ahora, repitiendo el ensayo anterior con $N = 1024$ y $M = 64$ pero variando la cantidad de canales (N_{Ch}) en 1, 2, 4 y 8 se construyó la Cuadro 3.10. De esta forma, se verificó que el agregado de canales mejora las propiedades estadísticas del reconstructor (como se justificó teóricamente en el Capítulo 2) y la cantidad de casos exitosos aumenta.

Número de Canales	Porcentaje de Casos Exitosos
1	75 %
2	82 %
4	90 %
8	86 %

Cuadro 3.10: Resultados de simulación para diferente Número de Canales con el algoritmo AIHT.

3.3.3. Rendimiento de los Generadores Caóticos

La instancia final comprendió en evaluar el uso de los generadores caóticos y su comparación con el PRNG. Ahora, utilizando el *AITH* con $N = 1024$ y $M = 64$ se llevaron a cabo 200 corridas para cada generador (Cuadro 3.11). Se encontró que el mapa *Logístico* produjo los mejores resultados, levemente superior al caso PNRG. En tercer lugar, se encuentra el generador TWBM. Los demás mapas caóticos empeoraron notablemente el rendimiento del sistema, por lo que no resultan aptos para este caso de CS. Una característica importante que destacar es que utilizando al generador *Logístico* la varianza en la *Similitud Espectral* y en el *Tiempo Medio de Reconstrucción* resultó ser un orden de magnitud menor que en el caso clásico. De esto último se infiere que ofrece una mayor probabilidad de reconstrucción exitosa.

Algoritmo	$< S_t >$	$Var(S_t)$	$< S_e >$	$Var(S_e)$	$< t_m >$[s]	$Var(t)[s^2]$
PRNG	0,1372	0,0248	0,1172	0,4438	0,00314	$4{,}032\ 10^{-6}$
FWTSM	0,4492	0,0590	0,4279	0,0600	0,1543	0,0231
TWBM	0,1828	0,0389	0,1627	0,0385	0,0298	0,0077
Logístico	0,1336	0,0227	0,1136	0,0217	0,0028	$5{,}958\ 10^{-7}$
TWBM2	0,5754	0,2138	0,5662	0,2221	0,0020	$1{,}064\ 10^{-6}$
TWBM4	0,9337	0,0542	0,9317	0,0571	0,0013	$1{,}653\ 10^{-7}$
TWTSM	0,4600	0,0946	0,4438	0,0972	0,0687	0,0156

Cuadro 3.11: Resultados de simulación para los distintos generadores de secuencias.

3.4. Diseño Final a Realizar

Una vez completado el estudio de las alternativas para el sistema de reconstrucción con CS, se propone implementar el sistema de la Figura 3.6. En primer lugar, se desarrollará una *Interfáz Gráfica de Usuario* que permita configurar, controlar y enviar datos a la placa de desarrollo Zedboard. A su vez, esta debe permitir cargar distintas señales de prueba, secuencias de mezclado y diccionarios. Las estadísticas en la reconstrucción también será informadas.

El reconstructor definitivo para programar en el ZYNQ 7000 será el *AITH*, con

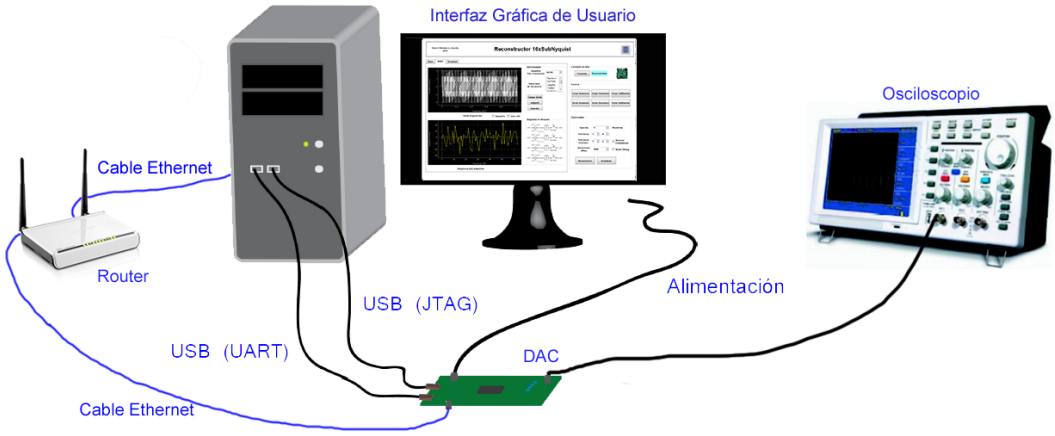


Figura 3.6: Sistema para evaluar el diseño de CS propuesto. Incluye control mediante GUI, comunicación en red y banco de medición (osciloscopio).

cantidad de muestras $N = 1024$, adquisición a una taza 16 veces menor a la de Nyquist ($M = 64$), dispersión $k \in [0, 32]$ y generador de secuencias *Logístico*. Se estableció que utilizar cuatro canales ofrece una confiabilidad razonable (90 %), por lo que se evita usar una cantidad mayor ya que vuelve complejo el sistema.
Con el propósito de aprovechar las FPU y el motor NEON en el SoC (Ver Capítulo 4 para más detalles) el DSP utilizarña punto flotante de simple precisión (32 bits). No obstante, se considerará eventualmente implementar un procesamiento intermedio en punto fijo según sea conveniente, especialmente en la lógica programable.

El envío de las secuencias de mezclado $P(t)$ junto con el diccionario Φ y la señal $y[m]$ implica la transmisión de 4,2 Mbytes de datos. Por ello, se decide implementar la comunicación mediante la pila TCP/IP. Esto permite delegar a los protocolos de red la transmisión, segmentación y recepción de paquetes, simplificando el desarrollo de la aplicación en comparación con la utilización de un puerto serie. Además, un diseño en red genera flexibilidad para distintos equipos funcionando simultáneamente adquiriendo señales y procesando datos.

Por último, se hará uso de una conversión Digital/Analógica con el fin de generar la señal de voltaje $\hat{x}$ y poder visualizarla en un osciloscopio.

Capítulo 4

Xilinx Zynq 7000

4.1. Zynq-7000 All Programmable SoC

4.1.1. Características y Prestaciones

La familia Zynq-7000 de Sistemas en Chip (SoC) integran en un único dispositivo de tecnología de 28 nm un microprocesador *ARM*® Cortex™-A9 de diversas prestaciones, junto con una unidad de lógica programable Xilinx™; resultando en un dispositivo de gran performance y bajo consumo. La Figura 4.1 muestra un diagrama simplificado del SoC, en la que se distingue el sistema de procesamiento (PS) y la lógica programable (PL). El primero, incluye al CPU Cortex A9 que cuenta con dos núcleos junto con memoria integrada (OCM), interfaces externas de memoria y un gran conjunto de periféricos de entrada y salida (I/O). A su vez, el PL en el chip ofrece la flexibilidad y la escalabilidad de una FPGA. En conjunto, el sistema provee el consumo, performance y simplicidad de uso típicamente asociado con ASICs [2].

Esta arquitectura permite la realización de aplicaciones únicas y funciones bien diferenciadas entre Hardware y Software. La integración entre el PL y PS entrega niveles de rendimiento que un sistema de dos chips (un microcontrolador y un FPGA por separado) no pueden lograr debido al ancho de banda limitado en sus I/O, pérdidas de acoplamiento y limitaciones en el consumo de energía. Una presentación más detallada del SoC se aprecia en la Figura 4.2 cuyo bloques constituyentes serán enunciados a continuación.

Por un lado, los componentes del PS incluyen:

- **Unidad de Procesamiento de Aplicación:** Contiene principalmente las CPU, Memorias Internas, el Caché de nivel 2 (L2), la Unidad SCU para el control de Caché y la interconexión de sistema.
- **Periféricos I/O:**
  - Multiplexadas (MIO): Se encuentran conectadas directamente a los periféricos (IOP) del PS. No están disponibles desde el PL.
  - Multiplexadas Extendidas (EMIO): Proveen acceso a la lógica programable junto con pines del encapsulado.

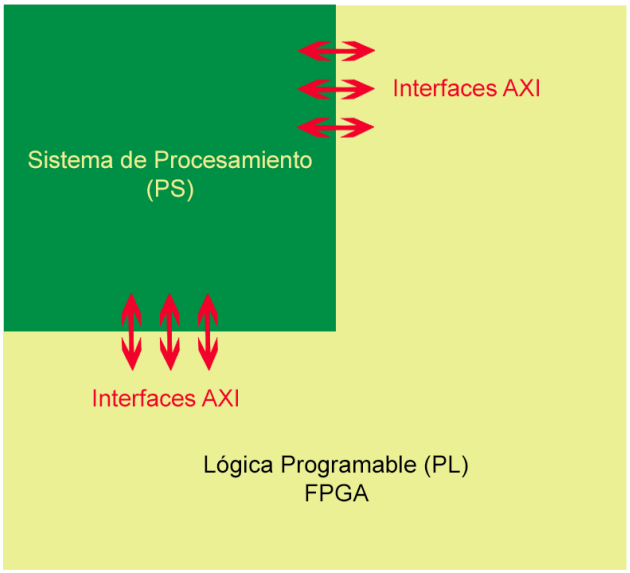


Figura 4.1: Esquema simplificado del SoC Zynq 7000.

- **Interfaces de Memoria**: Conexiones de altas velocidades a memorias DDR2, DDR3, SDRAMs, QDR II+ SRAM, RLDRAM 2/RLDRAM 3 y LPDDR2 [63].
- **Interconexión de Sistema :** Basado en conmutación de pistas de datos AXI[1] de alto rendimiento. Cuenta con los bloques:
  - OCM Interconnect: Provee acceso a la memoria de 256 KB desde el *Central Interconnect* y el PL.
  - Central Interconnect: Sus buses son de 64 bits. Conectan los IOP y el controlador DMA al controlador DDR, a la RAM on-chip y a las interfaces AXI de propósito general con el PL.
- **Controlador DMA:** Utiliza una interfaz AXI maestra de 64 bits funcionando al doble de frecuencia que el clock del CPU para realizar transferencias de acceso directo a memoria desde o hacia memorias del sistema y el PL. Cuenta con ocho canales. El código de programa para instrucciones DMA se escribe por software en una región de la memoria del sistema que es accedida por el controlador usando su propia interfaz AXI.
- **Temporizadores (Timers):** Cada procesador Cortex-A9 posee un timer privado y un *watchdog*, ambos de 32 bits. Los dos núcleos comparten un timer global de 32 bits. A nivel sistema, existe un timer *watchdog* de 24 bits y dos contadores triples de 16 bits (Consultar [2] para más detalles).
- **Controlador General de Interrupciones (GIC):** Maneja interrupciones enviadas al CPU desde el PS y PL. El controlador habilita, deshabilita, enmascara y prioriza las fuentes de interrupciones y lo comunica al núcleo seleccionado (puede ser ambos).

[1]Ver sección 4.1.2 para más detalles

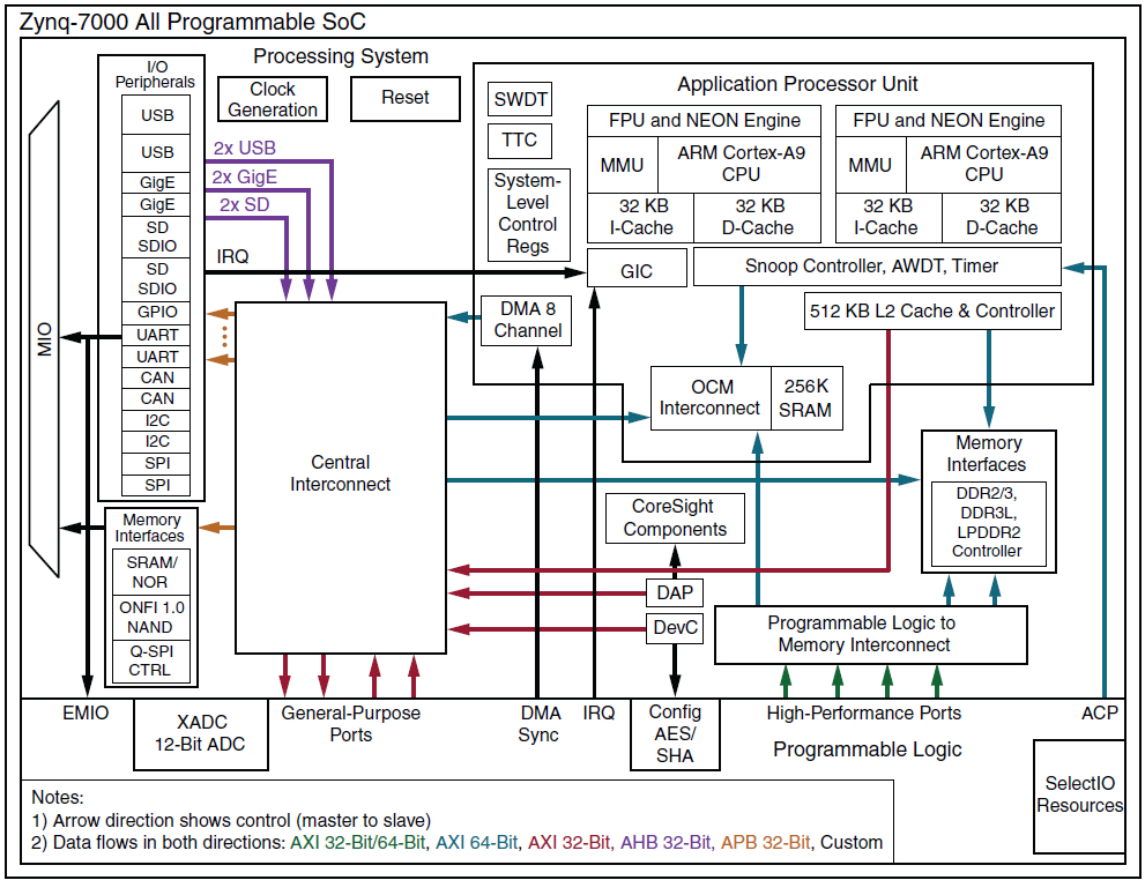


Figura 4.2: Diagrama en Bloques del SoC Zynq 7000 (Fuente [65]).

- **Memoria OCM:** Contiene 256 KB de RAM y 128 KB de ROM (para booteo, no es visible al usuario). Soporta dos puertos AXI esclavos de 64 bits, uno dedicado para el acceso del CPU a travez del SCU y el otro compartido por buses maestros compartidos entre el PS y el PL.
- **CoreSight:** On-Chip Debugger [64].

Asimismo, la parte PL presenta:

- **Bloques Lógicos Configurables (CLB)**: Constituidos por dos segmentos, cada uno compuesto de cuatro Look Up Tables (LUTs) de seis entradas y ocho elementos de almacenamiento. Cada LUT posee funcionalidad de registro y registro desplazamiento.
- **Bloques BRAM de 36kBytes**: Poseen verdadero puerto dual. Permite palabras de hasta 72 bits de longitud. Se puede configurar como un bloque RAM dual de 18KB.
- **Segmentos DSP48E1**: Realizan multiplicación con signo de 18 x 25 bits. Cuentan con una unidad sumador/acumulador de 48 bits con un pre-sumador de 25 bits.

- **Gestión de Reloj**: Además de generar sus propias señales de reloj, el PL también, recibe cuatro señales independientes de reloj desde el PS, asincrónicas entre sí.
- **Transceivers de Alta velocidad**: Soporta transferencia de datos de hasta 12.5 Gbps. Se pueden utilizar 16 transmisores y receptores como máximo.
- **Interfaz integrada para PCI Express**: Es capaz de funcionar hasta con 8 carriles de 500 Mbps.
- **Conversor A/D XADC:** El SoC contiene dos conversores que consiguen sensar tanto el voltaje como la temperatura del chip. Tiene 17 canales externos de entrada diferencial con una tasa máxima de adquisición igual a 1 MSPS.

El procesador Cortex A9 incluido en el Sistema en Chip implementa la arquitectura del conjunto de instrucciones (ISA) ARMv7-A. Soporta instrucciones Thumb de 16 bits como también la teconología Thumb-2 de 32 bits. En particular, la letra A en el nombre del ISA denota "Aplicación" e incluye una unidad de administración de memoria (MMU) que controla el acceso del CPU a los datos [66]. Adicionalmente, dispone de una unidad NEON para procesamiento vectorial de señales compartiendo su conjunto de registros con la unidad de Punto Flotante (FPU). En la hoja de datos XAPP1206 [67] se muestra cómo obtener ventaja del NEON en el procesamiento.

4.1.2. Interfaz AXI™

Existe la necesidad de establecer un único estándar en la comunicación entre los dispositivos incluidos en el sistema embebido, ya que esto facilita la integración de IP cores por parte de terceros. Por otro lado, comprender la manera en que se transmiten los datos es esencial para diseñar sistemas basados en el Zynq 7000. La *Interfaz Extensible Avanzada* (AXI, por sus siglas en inglés) es parte de la familia de buses para microcontroladores ARM AMBA y establece una interfaz[2] de comunicación punto a punto del tipo maestro/esclavo. La versión actual al momento de realización de este trabajo es AXI4, liberada al mercado en 2010.

Como se muestra en la Cuadro 4.1, existen tres protocolos AXI que pueden ser seleccionados para el diseño de una aplicación según se requiera [68]:

- Full AXI4 es para interfaces de memoria mapeadas y permiten ráfagas de hasta 256 datos de transferencia por ciclo utilizando sólo una dirección.
- AXI4-Lite es un diseño que utiliza menos recursos que AXI4 Full por lo que simplifica su uso y empleo en el desarrollo de sistemas.
- AXI4-Stream retira la necesidad de utilizar direcciones para la transmisión de datos. Permite la transferencia continua de información. No son consideradas mapeadas en memoria.

[2]Colección de uno o más conjunto de señales independientes que conectan dos dispositivos.

Interfaz	Características	Ráfaga	Longitud de Datos	Aplicaciones
Full AXI4	Mapeo de memoria tradicional (Dirección única, Datos Múltiples)	Hasta 256	32 a 1024 bits	Memoria Embebida
AXI4-Stream	Sólo ráfaga de datos	Ilimitado	Cualquier tamaño	DSP, Video, Comunicaciones
AXI4-Lite	Mapeo de memoria tradicional (Dirección única, Datos únicos)	1	32 o 64 bits	Lógica de Control Pequeña, FSM

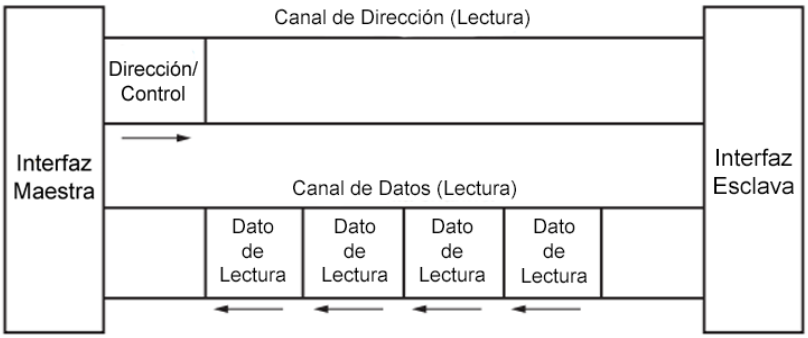
Cuadro 4.1: Interfaces AXI disponibles

En la comunicación dentro del SoC el maestro AXI es siempre quien inicia la transacción, mientras que el esclavo responde a ella. Esto se cumple para procesos de escritura y lectura por igual. La interfaz utiliza la arquitectura de canales como se ve en la Figura 4.3. Para los protocolos que utilizan mapeo de memoria (AXI4-Full y AXI4-Lite), la información puede moverse simultáneamente en ambas direcciones; ya que existen conexiones de datos y direcciones separadas. A nivel de Hardware, AXI4 permite diferente señales de clock para cada par mestro/esclavo. Además, los protocolos admiten la inserción de etapas de pipeline para asistir con requerimientos de timing (Sincronización temporal). Por otra parte, AXI-Stream define un solo canal para datos, ya que puede realizar transmisión sin interrupciones en un único sentido (Figura 4.4). A pesar de ser un protocolo *punto a punto*, cuando existen varios dispositivos AXI pueden ser todos conectados entre sí utilizando una estructura llamada *AXI-Interconnect*. Existe un bloque IP de Xilinx que cumple esta función, incluyendo varias interfaces maestro y esclavo para rutear distintas transacciones de datos [68].

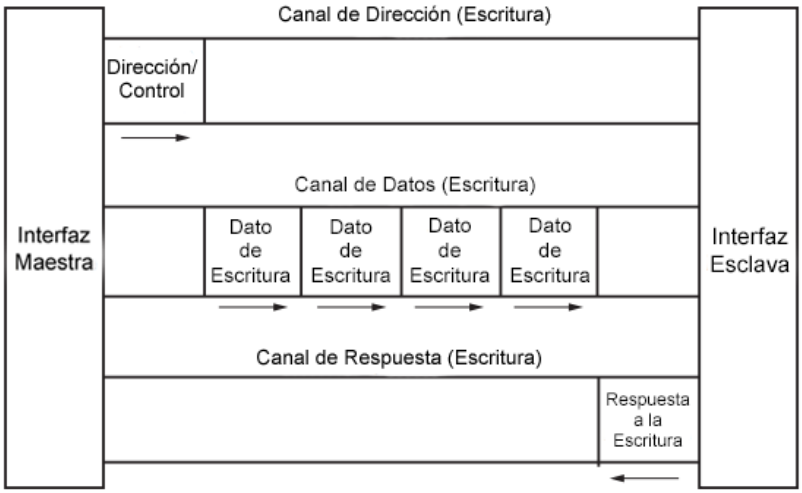
4.1.3. Conexiones PS-PL

Habiendo definido el estándar AXI y sus interfaces, se presenta en más detalle las conexiones entre el PL y PS en el Zynq 7000 (Figura 4.5). Todos los puertos que conectan estas dos secciones del SoC son mapeados en memoria; por lo que, no utilizan AXI-Stream. En primer instancia, se cuenta con cuatro puertos HP de alto rendimiento (*High Perfomance Ports*, en Inglés) de 64 bits. Las interfaces HP son esclavas de la lógica que diseñamos en el PL. Entonces, será necesario que los bloques en el PL presenten una interfaz AXI maestra. De esta manera, es posible acceder al espacio de direcciones del PS. Por ejemplo, un IP core en el PL puede iniciar operaciones de lectura/escritura hacia la memoria DDR3 o el OCM. De igual forma, el CPU también tiene acceso a la DRAM debido a lo cual puede compartir datos con un maestro AXI en la lógica programable.

También, tenemos dos puertos AXI maestros llamados MGP0 y MGP1 (*Master General Purpose Port*). Son las únicas interfaces que pueden iniciar operaciones de transmisión desde el PS hacia el PL. Debido a esto, el ARM se vale de ellos para



(a) Transacción de Lectura



(b) Transacción de Escritura

Figura 4.3: Arquitectura de canales AXI4 para el intercambio de datos.

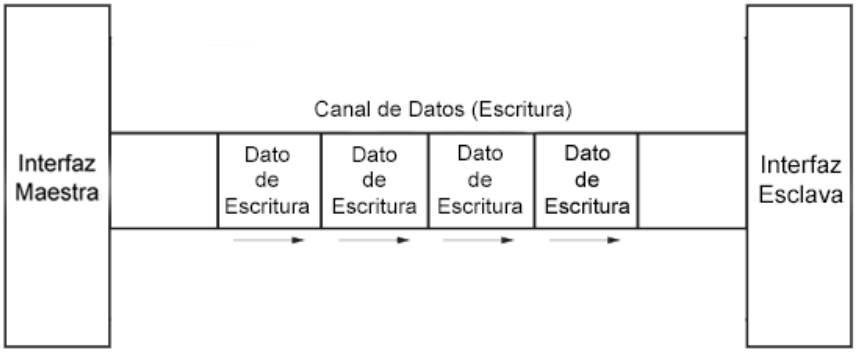


Figura 4.4: Arquitectura de canales para AXI4-Stream.

controlar bloques generalmente usados como *Aceleradores de Hardware*. Es decir, el CPU envía datos al PL para procesar y obtener ventajas de arquitecturas concurrentes. En consecuencia, se reducen tiempos de ejecución por parte del microprocesador. Concretamente, estos puertos son el principal medio para que el controlador DMA en el PS pueda acceder directamente a la lógica programable. Alternativamente, se dispone de los conectores internos SGP0 y SGP1 (*Slave General Purpose Port*) para maestros AXI.

La siguiente interfaz es el puerto ACP (*Accelerator Coherence Port*). Presenta varias similitudes con HP, como ser del tipo AXI esclavo. Sin embargo, el ACP se encuentra conectado directamente al SCU, quien se encarga del control de caché L1 y L2 del CPU. Este hecho trae implicancias que pueden resultar beneficiosas en términos del rendimiento de una aplicación. Cuando una transacción es comenzada por un elemento en el PL utilizando el ACP, entonces el caché es revisado para determinar la coherencia de los datos. Si una instancia de los datos está disponible, entonces la solicitud es respondida con la información almacenada en los cachés. Esto genera transmisiones más rapidas desde el PS al PL. Por el contrario, si no existe una copia de los datos, mediante la comunicación entre la memoria L2 y el controlador DDR3 la solicitud es redirigida a la memoria DRAM. En ese caso, se obtiene una degradación en la performance debido al aumento en la latencia. Es por eso que el ACP debe ser utilizado en aquellas situaciones en las que el acceso directo a L1 y L2 sea ventajoso.

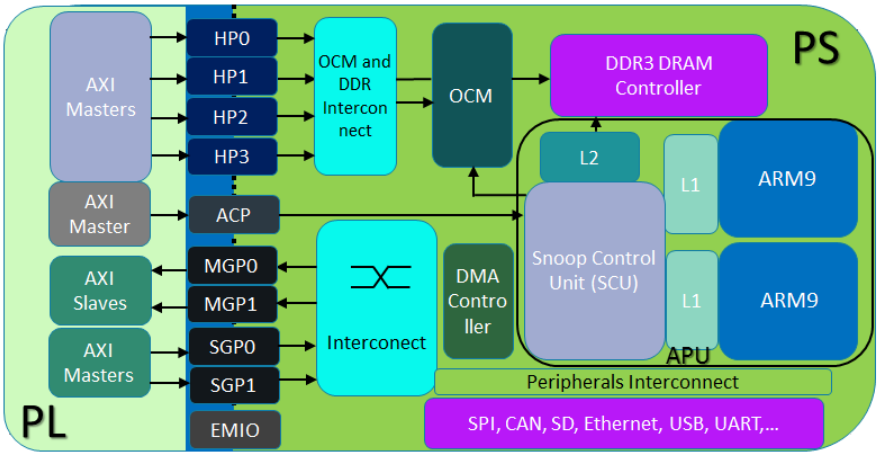


Figura 4.5: Conexiones dentro del Zynq 7000. Se observan los puertos de comunicación HP, ACP, MGP y SGP.

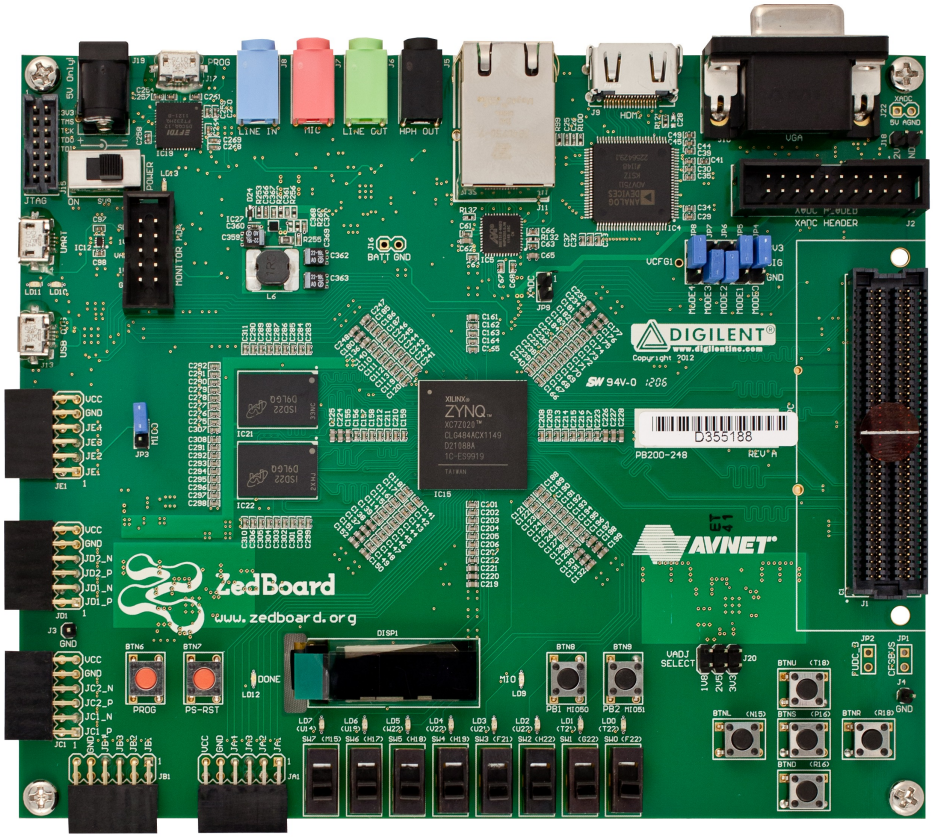


Figura 4.6: Fotografía superior de la placa Zedboard .

4.2. Placa de desarrollo ZedBoard

4.2.1. Prestaciones Generales

La plataforma utilizada en el desarrollo del trabajo fue la *Zedboard*, fabricada por la empresa *Digilent* (Figura 4.6). Sus prestaciones principales son:

- Ethernet 10/100/1000.
- Memoria Flash Quad-SPI de 256 Mb.
- Ranura para tarjeta SD.
- Memoria RAM DDR3 de 512 MB.
- Audio CODEC I2S.
- Presenta Multiples Displays (1080p HDMI, 8-bit VGA y 128 x 32 OLED)
- Programador JTAG incluido.
- Expansiones I/O FMC, Pmod Y XADC.
- SoC Zynq-7000XC7Z020-CLG484-1 (85000 celdas lógicas programables).

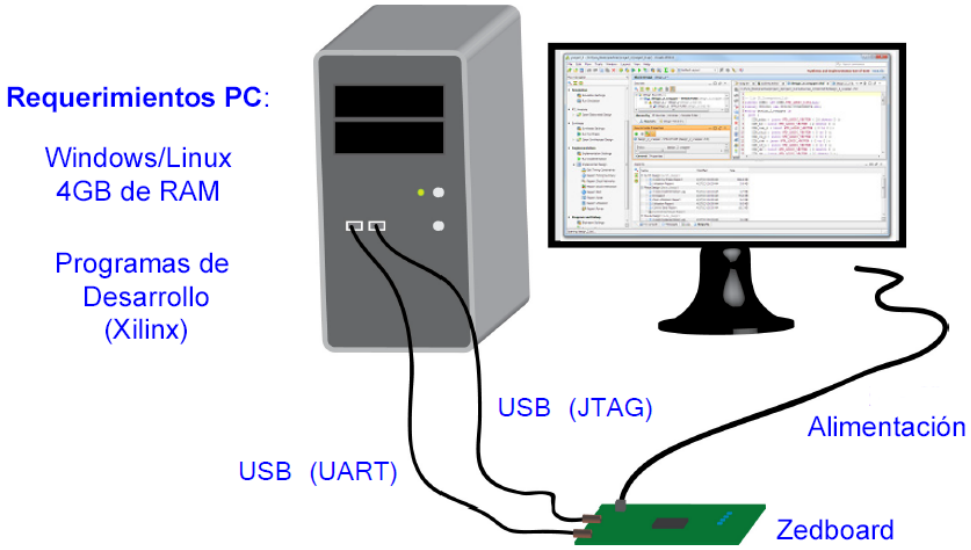


Figura 4.7: Disposición mínima para la utilización de la Zedboard. Requiere dos cables USB tipo C, fuente de alimentación y una (PC).

La configuración primaria del dispositivo se carga en la memoria QSPI Flash. Sin embargo, se puede utilizar de manera auxiliar una tarjeta SD para este fin [69]. A su vez, requiere dos conexiones USB con la PC. La primera de ellas es utilizada para la programación JTAG del SOC, mientras que la otra proporciona comunicación entre la UART 0 (por defecto) y un puerto COM virtual. Los requerimientos se resumen en el esquema de la Figura 4.7.

4.2.2. Conectores PMOD

Las interfaces estándar PMOD (Figura 4.8) son utilizadas en las placas de *Digilent* para conectar periféricos que poseen pocos pines I/O y trabajan a baja frecuencia [70]. Soportan SPI, I2C, UART,I2S, Puentes H y protocolos GPIO. La versión de 12 pines, utilizada en el proyecto, provee ocho I/O para señal junto con dos pines de alimentación (VCC) y dos de masa (GND). El voltage que entrega la placa para conectar periféricos externos es de 3,3 Volts. A su vez, estos no deben consumir más de 100 mA en total. Estos conectores no están pensados para operar en alta frecuencia. Aunque, usando RJ45 y par trenzado, se pueden alcanzar velocidades de 24 MHz hasta 4 metros de distancia. En la placa Zedboard (Figura 4.6) se encuentran en la esquina izquierda inferior. Con respecto a las características de las señales digitales, se espera que utilicen la convención LVCMOS 3.3 Volts o LVTTL 3.3 Volts.

En general, la corriente de los pines está suministrada por la FPGA o uC y el rango esta comprendido entre ± 16 mA y ± 24 mA. Existen tres tipos de conectores: Estandard I/O, MIO (conectado al bus I/O del PS) y de alta velocidad. Los primeros, contienen diodos de protección y resistores series de 200 Ω para prevenir corto circuitos y conflictos de compatibilidad. Como desventaja, su ancho de banda

queda limitado. Luego, los MIO son idénticos a los anteriores pero sólo pueden ser accedidos por el PS. Finalmente, los puertos de alta velocidad están diseñados para transmisión diferencial y no cuentan con protección; pudiendo lograr mayor frecuencia de conmutación. Por ello, no se hará uso de estos últimos para evitar dañar el SoC. [70].

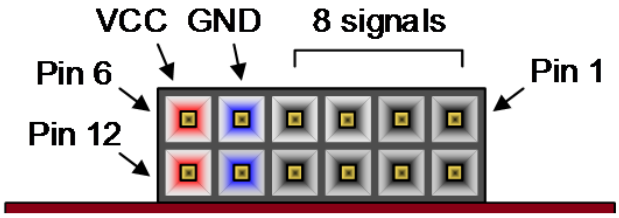


Figura 4.8: Ficha del conector PMOD (vista frontal).

4.3. Paquete Vivado

4.3.1. High Level Synthesis

La empresa Xilinx, dentro de su paquete *Vivado* para desarrollo de aplicaciones en sistemas embebidos, incluye el software de *Síntesis de Alto Nivel* (*High Level Síntesis* en Inglés). Esta herramienta básicamente convierte una especificación escrita en lenguaje C a una implementación a nivel de registros (*RTL* en Inglés) que luego se sintetiza en una FPGA. De modo que, permite obtener beneficios al trabajar a un mayor nivel de abstracción que en VHDL o Verilog en la creación de Hardware. Las ventajas que se pueden enunciar son:

- Reducción de tiempos de desarrollo gracias a la abstracción a los detalles de implementación en RTL.
- Realización de pruebas de verificación en C para validar funcionalmente los algoritmos desarrollados.
- Utilización de directivas para optimizar la síntesis (Pragmas) en el código C, permitiendo la creación de implementaciones en Hardware específicas (sintetizar lazos *for* en paralelo UNROLL, agregar etapas de pipeline, asignar bloques de memoria RAM a arreglos, etc).
- Creación de varias versiones de un IP a partir del código fuente. Gracias a esto se puede evaluar rápidamente distintas opciones de realización y elegir una solución óptima.

El principal elemento que usa Vivado HLS para generar el RTL es una función escrita en C,C++ o SystemC (Figura 4.9). A su vez, esta puede contener una jerarquía de subfunciones. Por otro lado, necesita un archivo que incluya restricciones (período e incerteza de reloj, modelo del SoC) y directivas de síntesis. Estas últimas son opcionales, ya que se utilizan para implementar un comportamiento específico

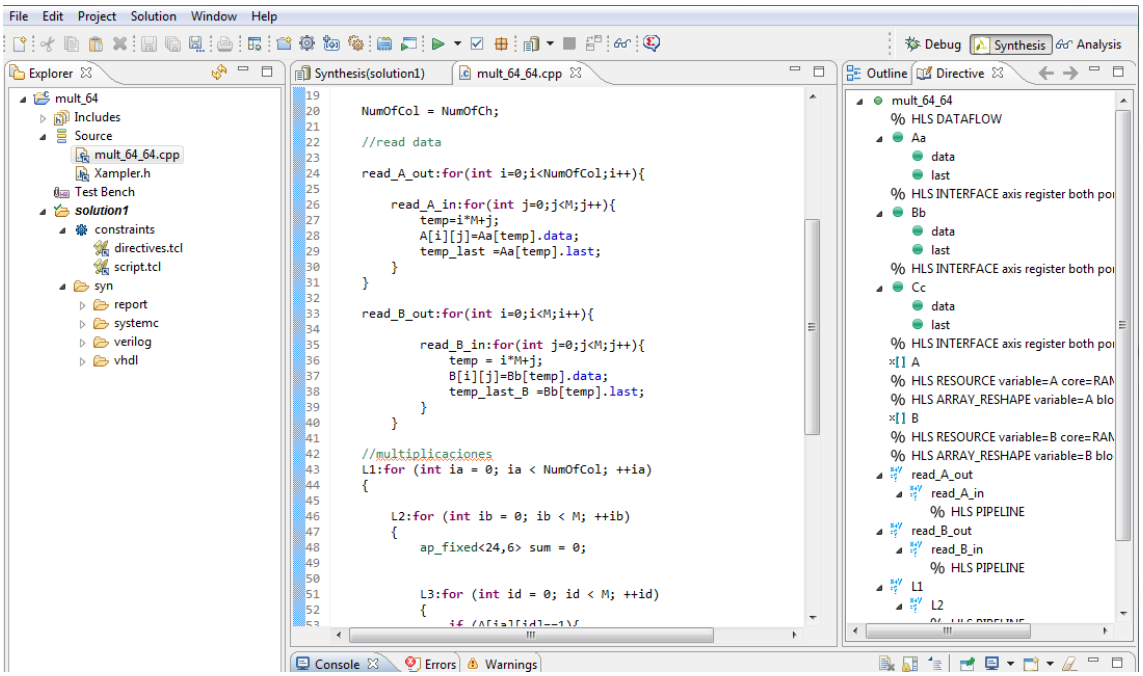


Figura 4.9: Captura de pantalla del entorno Vivado High Level Synthesis (Ver 2016.4 64 bits). En la ventana central se ve el código funcional C++ y en la solapa *Directive* los Pragmas usados.

de la lógica detallada. También, se puede emplear una archivo para pruebas (Test Bench) junto con sus ficheros asociados; con el fin de simular el comportamiento del Hardware antes de la síntesis. Posteriormente, HLS lo vuelve a invocar automáticamente para verificar el diseño en la cosimulación C/RTL [71]. Como salida, HLS genera el código para la síntesis de compuertas lógicas y generación de bitstream. Los formatos estándar que puede producir son VHDL (IEE 1076-2000), Verilog (IEEE 1364-2001) y SystemC (IEEE 1666-2006). Los archivos de implementación son empaquetados como un bloque IP para ser usado dentro de las otras herramientas que provee Xilinx.

En relación al proceso de creación del código HDL, HLS primero obtiene una máquina de estados a partir del comportamiento descripto en C. Luego, determina cuáles operadores elementales debe utilizar con el fin de asignar recursos disponibles en la FPGA. Con toda esa información, crea el denominado *flujo de control*. El término *Scheduling* es usado para mapear este flujo en ciclos de reloj, de manera tal de cumplir con los requerimientos de timing con la tecnología disponible. Finalmente, en el *Binding* se decide con qué elementos de la librería de Hardware realizar las operaciones. La decisión que realiza la herramienta en esta etapa es si es necesario o no compartir recursos.

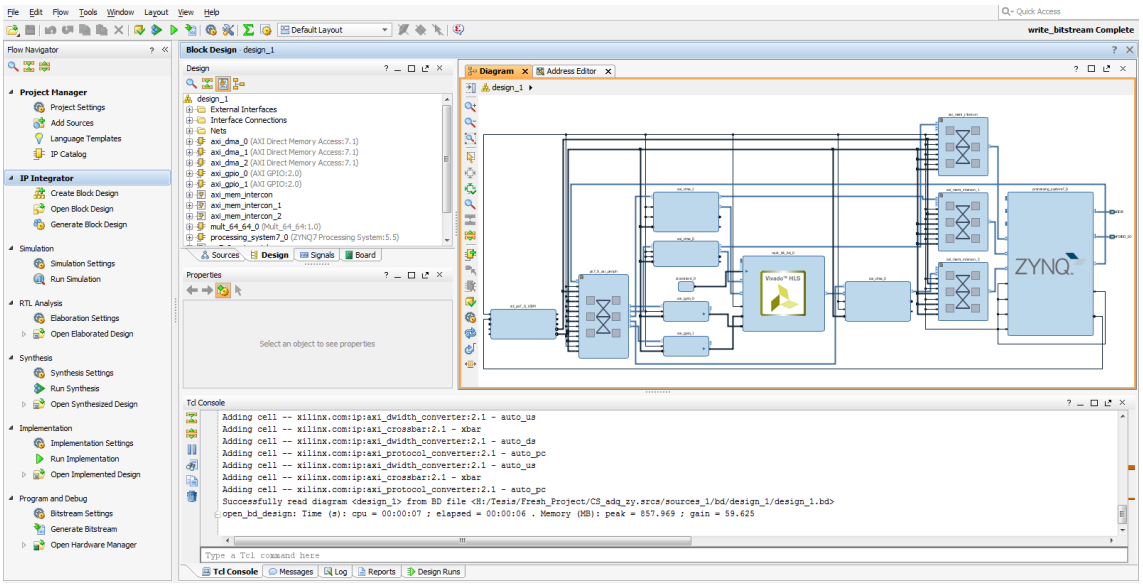


Figura 4.10: Captura de pantalla del entorno Vivado Design Suite (Ver 2016.4 64 bits). Dentro del diseño de bloques, se pueden discernir un IP generado con HLS (contiene el logo de Vivado) y un bloque *ZYNQ* que representa al PS (denominado PS7).

4.3.2. Vivado Design Suite

Habiendo mencionado la posibilidad de crear *IP Cores* con HLS, en esta sección se describe el entorno de programación *Vivado Design Suite* (Figura 4.11). Principalmente, en él se efectúan de manera interactiva el diseño y análisis de sistemas para SoC y FPGA. Permite integrar bloques diseñados con HLS, resolver cuestiones de conectividad, análizar el timing, asignar recursos, generar restricciones (*constraints*), efectuar simulaciones lógicas, asignación de I/O,etc. A su vez, existe un apartado para generar código HDL, incluyendo la posibilidad crear el diseño a partir de esquemáticos jerárquicos.

La metodología de diseño consiste, a partir de especificaciones, en la creación de IPs en HLS ó HLD. Seguidamente, se lleva a cabo el diseño en bloques; en donde se instancia y conectan todas las unidades necesarias (controladores DMA, aceleradores de Hardware, interfaces, microprocesadores, etc). Con las directivas de optimización y el archivo *XDC Constraints* Vivado procede a la síntesis del sistema. En esta etapa es donde se realiza el *Floorplanning*, es decir, las ubicaciones tentativas de los bloques funcionales dentro de un chip. Los reportes disponibles en la síntesis son de Timing, Reloj, utilización de recursos y consumo de potencia [71]. La implementación es la fase siguiente y en ella se genera el diseño definitivo para programar el SoC. Por último, se debe generar el bitstream y opcionalmente el Hardware puede ser exportado hacia el SDK. A modo de resumen, la Figura 4.11 muestra los pasos mencionados anteriormente.

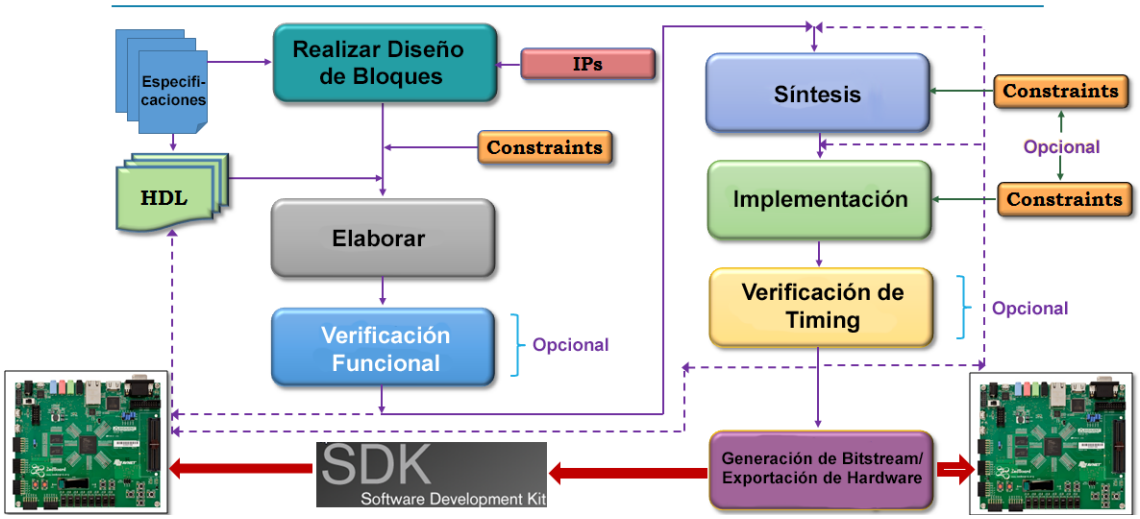


Figura 4.11: Proceso de elaboración de diseños con *Vivado Design Suite*. Una vez generado el bitstream, se puede implementar sólo la lógica en hardware o emplear también el SDK para programar el procesador ARM.

4.3.3. Xilinx Software Development Kit

Para la creación de software embebido se utiliza la herramienta *Xilinx SDK* (Figura 4.12), cuyo IDE está basado en *Eclipse*. Básicamente, posee herramientas de desarrollo GNU, compilador C/C++ para el procesador ARM Cortex-A9, debugger y generador de *Board Support package*. Tambíen es posible programar Softcores como el *Microblaze* [72]. El BSP contiene los drivers básicos y utilidades de Xilinx; sin embargo, no es un RTOS. En primer lugar, se escribe la aplicación en C ó C++ usando los archivos y funciones que correspondan. Después, se puede modificar el *Linker Script* para detallar regiones de memoria que pueden ser asignadas a distintos recursos (Stack, Heap, etc). Como paso posterior se construye el proyecto y si es necesario se programa el PL antes de cargar el software en el PS. La Figura 4.13 muestra el camino típico a seguir a partir de Vivado hacia el SDK.

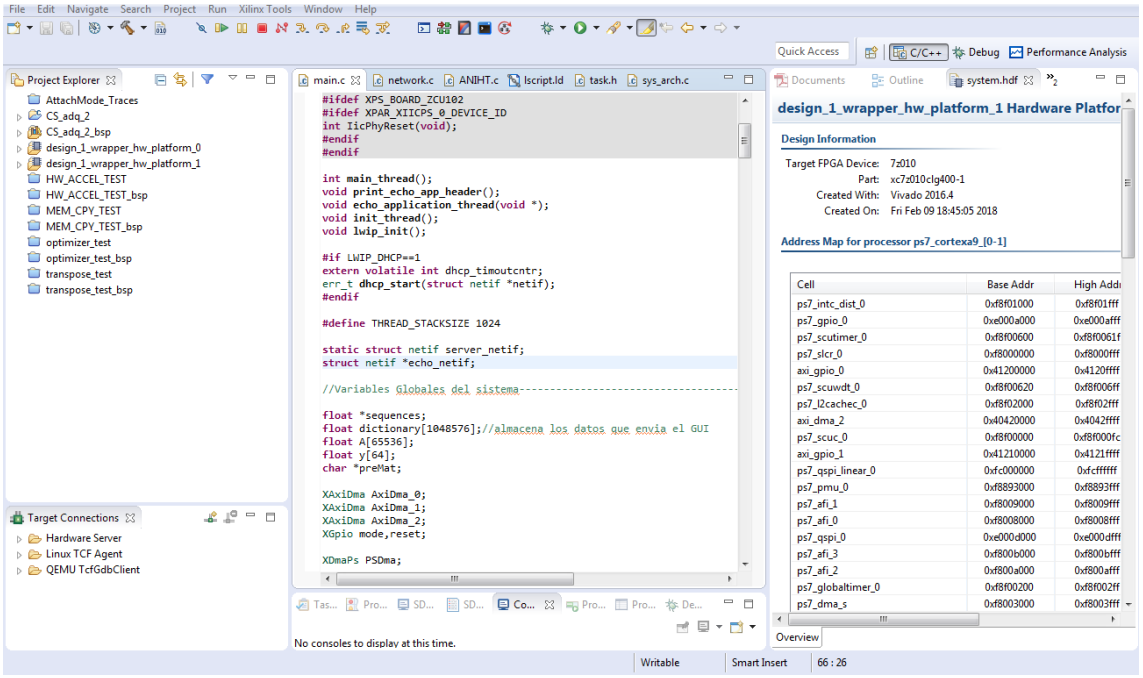


Figura 4.12: Captura de pantalla del entorno Xilinx SDK (Ver 2016.4 64 bits).

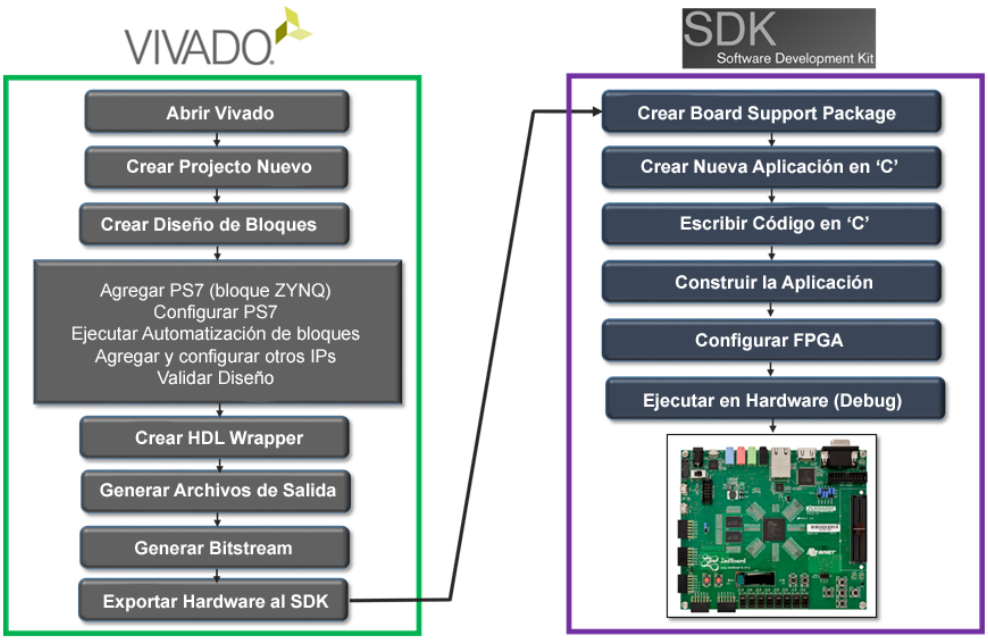


Figura 4.13: Pasos secuenciales para el desarrollo del proyecto con el SoC Zynq 7000.

Capítulo 5

Software Embebido y FPGA

5.1. Banco de Pruebas

La Figura 5.1 presenta el banco de medición empleado para el desarrollo del proyecto. En primer lugar, se puede apreciar que la placa de desarrollo posee tres conexiones con una PC: comunicación serie USB UART (para el envio de mensajes a un terminal), enlace JTAG (programación del Hardware) y conexión en red mediante un cable Ethernet UTP Categoría 5. El sistema embebido se controla con una Interfaz gráfica de Usuario (GUI) desarrollada específicamente para este proyecto (ver Sección 5.3 para más detalles). En el gráfico se puede distinguir principalmente el SoC, una interfaz de red PHY, memorias RAM/ROM externas y un conector PMOD que une un conversor Digital/Analógico diseñado en la lógica programable con un filtro pasabajos Bessel de cuarto orden. Los detalles del DAC son comentados en el Capítulo 6.

Dentro del SoC, el μP recibe comandos, parámetros de configuración y las señales necesarias para la aplicación de CS, a través de la interfaz PHY. El software principal, ejecutado en el núcleo 0 del ARM, procesa las secuencias de mezclado P_i, el vector de submuestras $y[m]$ y el diccionario Ψ; con el fin de obtener la señal aproximada $\hat{x}[n]$. Para reducir el tiempo requerido en la reconstrucción, el procesador utiliza dos IPs realizados en la FPGA: Un multiplicador matricial (4x64 por 64x64) y una red de ordenamiento en paralelo de 128 elementos. Los aceleradores en HW son accedidos mediante controladores DMA implementados en la lógica programable. Gracias a esto, cada IP puede acceder a los contenidos en memoria RAM sin intervención del microprocesador, para lo cual es necesario a su vez utilizar el controlador DRAM. Finalmente, el sistema envía la señal recuperada $\hat{x}[n]$ simultáneamente al GUI (mediante TCP/IP) y por medio del DAC a un osciloscopio para su visualización.

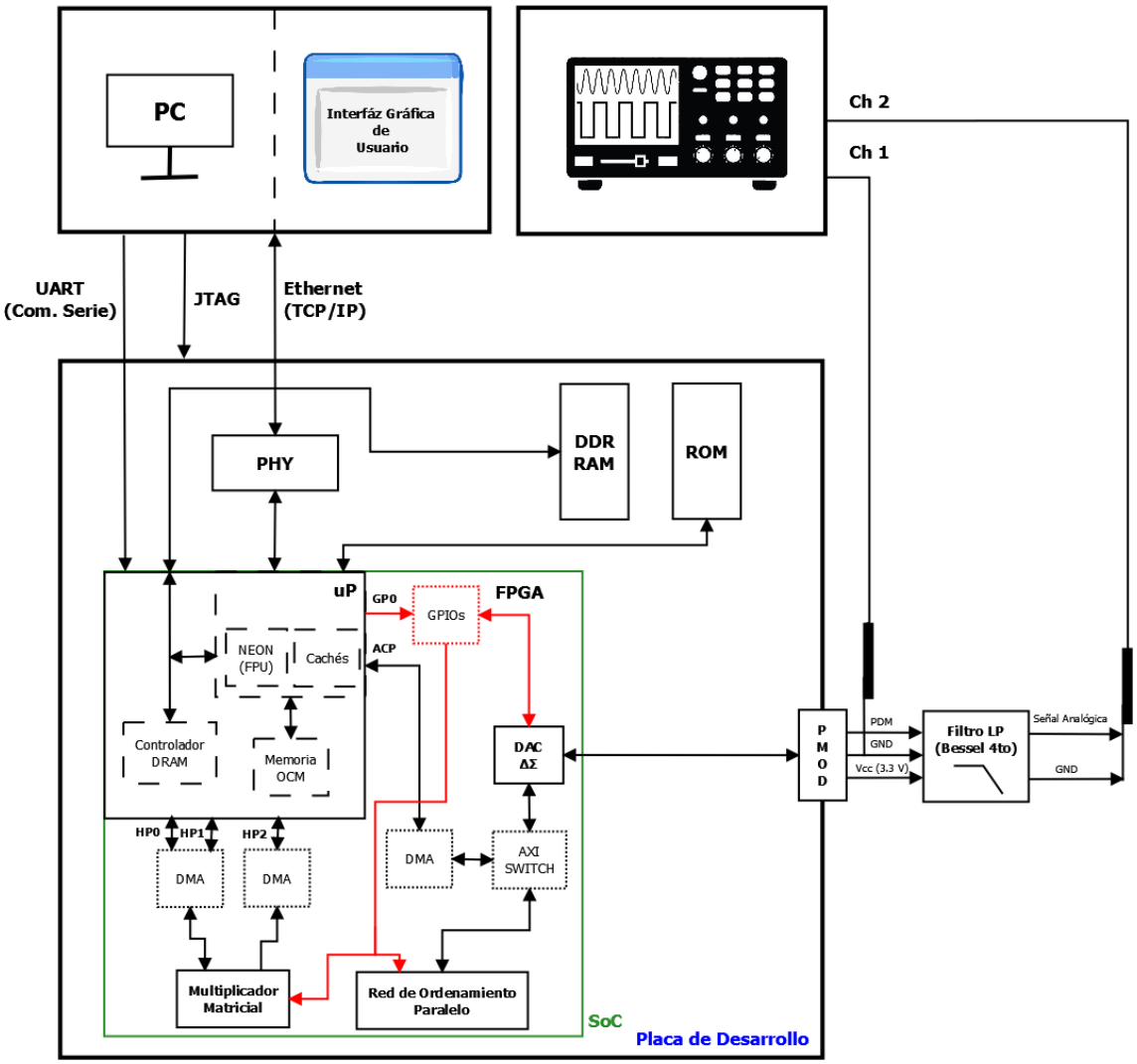


Figura 5.1: prueba utilizado para el Trabajo Final.

5.2. Software Embebido

Antes de detallar cada diseño en Hardware, se explicará el funcionamiento del software que sirve como base principal del sistema. El mismo fue desarrollado en el entorno *Xilinx SDK*. Inicialmente, se introducirá las dos herramientas que permitieron realizar la aplicación del proyecto: *FreeRTOS* para la administración del SoC y *LWIP* empleada en la transmisión de señales y parámetros.

5.2.1. Sistema Operativo en Tiempo Real FreeRTOS

Existen dos enfoques al momento de diseñar el software para un sistema embebido: el *SuperLoop* (superlazo en Inglés) y *Multithreading* (multitarea en Inglés). El primero consiste en el típico lazo *While(1){}* dentro de la función *Main()* {} del lenguaje C. La estructura del *While* itera indefinidamente secuenciando una serie de tareas. Presenta la ventaja de ser simple y sin *Overhead* (exceso en utilización de recursos). Sin embargo, es difícil de escalar (expandir el sistema, agregar más

funcionalidades) y de balancear el tiempo entre tareas y sus prioridades. Por otro lado, la técnica multitarea (empleada en SO) soluciona estos inconvenientes a costa del incremento en requisitos de memoria. En este trabajo se decidió usar el SO *FreeRTOS* [73]. Básicamente, es un *Kernel*[1] de tres archivos en C utilizado para aplicaciones en sistemas embebidos con requerimientos de tiempo real, es decir, las tareas deben ser cumplidas antes de un determinado tiempo o *deadline*. Este SO permite que los programas sean organizados como una colección independiente de Tareas de ejecución. Cuando un procesador contiene un núcleo, sólo una tarea puede ser ejecutada a la vez. El *Scheduler*, que se encuentra contenido en el Kernel, decide cuál de ellas corre en un determinado instante de tiempo examinando la prioridad que cada una tiene asignada [74].

Las principales ventajas que se obtienen al utilizar el RTOS son la modularidad (cada tarea tiene un propósito específico) junto con la abstracción en el manejo de tiempos y eventos; lo cual facilita la implementación de un sistema de comunicación en red basado en TCP/IP.

5.2.2. Pila TCP/IP LWIP

Lightweight IP (LWIP, IP *ligero* en Inglés), es una pila de red TCP/IP sencilla, de código abierto, diseñada específicamente para sistemas embebidos. Soporta varios protocolos comunes, entre ellos IP, ICMP, UDP, TCP, IGMP, ARP, DHCP, DNS, etc [75]. Dentro de FreeRTOS, LWIP emplea una versión reducida (versión *Lite* en Inglés) del socket Berkeley como abstracción para la programación en red. Un socket consiste en una combinación entre una dirección de red IP y un puerto TCP. Desde el punto de vista de la programación, para la transferencia y recepción de paquetes entre dos Hosts Cliente/Servidor, sólo hay que usar las funciones *write()* y *read()*. LWIP junto con FreeRTOS se encarga del encapsulado, segmentación, reconstrucción y control de errores. Por estas razones, se prefirió la comunicación en red en vez de usar la UART.

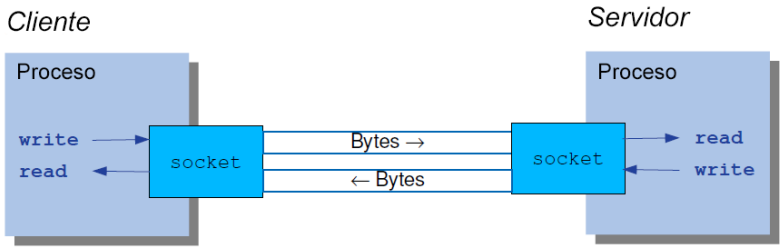


Figura 5.2: Abstracción en la comunicación entre Sockets provista por LWIP.

La arquitectura jerárquica del sistema se muestra en la Figura 5.3. En la mayor capa de abstracción, se encuentra la aplicación en software de Sensado Compresivo. Entre ella y el SO *FreeRTOS* se ubica LWIP. Las flechas bidireccionales denotan

[1]Software importante de un SO. Se encarga de la gestión de recursos del sistema.

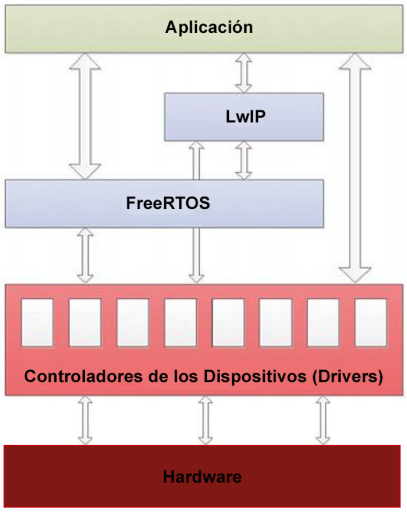


Figura 5.3: Capas de abstracción del sistema embebido. Desde el software de aplicación hacia el hardware.

la comunicación entre capas, principalmente para la trasmisión de datos y control. Más abajo, se encuentra los controladores de los dispositivos de la placa (ej: Interfaz Ethernet, UART, Memorias, GPIOs) en conjunto con los bloques programados en el FPGA. Para el caso de Xilinx, LWIP provee los drivers de la tarjeta Ethernet en la placa y se encuentra dentro del *Board Support Package* (BSP) del SDK. Finalmente, se halla la denominada capa física o hardware.

5.2.3. Flujograma Principal

El sistema programado (Figura 5.4) se ejecuta en el núcleo 0 del ARM y consiste en cuatro tareas administradas por el *FreeRTOS*:

- Tarea 1 (Iniciar del Hardware)
- Tarea 2 (Iniciar de red)
- Tarea 3 (Escuchar por conexiones entrantes)
- Tarea 4 (Ejecutar aplicación de Sensado Compresivo)

Las Tareas 1 y 2 son secuenciales, procede una y luego otra. En cambio, las tareas 3 y 4 se ejecutan de manera simultanea. Al comienzo, el sistema queda a la espera de conexiones entrantes por parte de la PC. Luego, al establecerse la conexión, se crea la aplicación de CS pasando a la número 3 a segundo plano (con menor prioridad). El reconstructor se queda a la espera de recibir cinco comandos posibles enviados desde el GUI:

- **MTXB:** Recepción y carga de memoria del diccionario Ψ.
- **MTXA:** Recepción y carga de memoria de las secuencias P_i.

- **SEQY:** Recepción y carga de memoria de la señal de submuestras $y[n]$.
- **OPCT:** Recepción de los parámetros del reconstructor AIHT.
- **MLTM:** Recepción de la orden para reconstruir a $\hat{x}$ y envío de los resultados al GUI y al DAC.

En función de la orden recibida, el sistema procede a realizar la actividad correspondiente. Los códigos en C del software se encuentran en el Apéndice.

5.2.4. BSP y Parámetros de Compilación

Cabe destacar que fue necesario realizar una serie de ajustes para asegurar el correcto funcionamiento del software. Para evitar los cuelgues del SO debido a fallas en la relocalización de bloques de memoria, se modificó los tamaños asignados para el *Heap* y *Stack* en el archivo *lscript.ld* dentro del proyecto en el SDK:

- Heap: 0xA000000
- Stack: 0xA000000

Después, dentro de la configuración del BSP (accedido desde el archivo *system.mss*) se activó la opción *lwip141* bajo la solapa *Overview*. Ahí mismo, el DHCP (asignación dinámica de direcciones IP) fue desactivado cambiando el valor de todos sus parámetros a FALSE. Esto es así porque el diseño se puso bajo prueba en una red con IP fija. Por otra parte, para aprovechar las prestaciones de la unidad de punto flotante (y por ende acelerar las operaciones vectoriales) se le especificó al compilador que utilice el núcleo NEON mediante la bandera -mpfu=neon. También, se dio la directiva de optimización *-O2*; que permite la realización de *Instruction Scheduling* [76].

5.2.5. Comunicación en Red

Se desarrolló un protocolo de mensajes simple a nivel de aplicación (por encima de TCP/IP) con el fin de transmitir eficazmente los datos entre el GUI y el SoC. Los datos son enviados en formato punto flotante de precisión simple (32-bits, estándar IEEE-754) y en *Big Endian*[2]. Estas características requirieron ser detalladas en la programación de la interfaz con las funciones *setFloatingPointPrecision()* y *setByteOrder()* (ver Sección 5.5 para más detalles). Los parámetros de la red privada donde se realizaron las pruebas se detallan a continuación:

- **Dirección IP** (SoC): 192.168.1.10 (Estática)
- **Puerto de Conexión**: 1491
- **Puerta de Enlace** (Router): 192.168.1.1
- **Máscara de Red**: 255.255.255.0

[2]Los bytes son enviados a partir del más significativo.

- **Dirección IP** (GUI): 192.168.1.30 (Dinámica)

Como se requería conocer la dirección de red de la placa y que no cambie, se desactivó el DHCP (ver Sección 5.2.4). Con esto se logró que la IP fuera estática. En el caso del PC, donde se ejecuta el GUI, la IP es dinámica (DHCP activado). No obstante, no existe peligro de colisión de direcciones de red porque el pool[3] donde obtiene para la PC es mayor a 192.168.1.30. Por otra parte, el puerto 1491 se eligió arbitrariamente, ya que según la *Autoridad de Números Asignados de Internet* (IANA en Inglés) no se encuentra asignado a ninguna aplicación específica al momento de realización del proyecto.

En la Figura 5.5 se muestran las cuatro situaciones posibles en las que, desde la interfaz gráfica se envía la señal a procesar junto con la información requerida. En todos los casos, se comienza con el envío del respectivo comando. Luego, se espera recibir por parte del sistema embebido un mensaje de cuatro caracteres ASCII. Los primeros tres siempre son *SND*, mientras que la cuarta especifica el tipo de información a recibir (A: secuencias P_i, B: diccionario, Y:señal $y[m]$, O: parámetros de configuración para el *AIHT*). Esto significa que el SoC está listo para recibir los datos correspondientes. Posteriormente, se procede a la transmisión en paquetes segmentados; siendo esta actividad delegada a la pila TCP/IP.

Es necesario analizar de manera aislada el comando *MLTM*, que da la orden al sistema para procesar $y[m]$ y obtener $\hat{x}[n]$. Cuando la placa de desarrollo recibe el primer mensaje, procede a realizar las tareas que muestra la Figura 5.6. Básicamente, construye la matriz A con el diccionario y las secuencias recibidas. Luego, ejecuta el optimizador AIHT para obtener la estimación de coeficientes $\hat{\alpha}$. Con ese vector, encuentra $\hat{x}$ post-multiplicando a Ψ por $\hat{\alpha}$. El software utiliza un timer en el PS del chip para medir el tiempo requerido en cada uno de estos tres procesos; almacenando los resultados en memoria. Después, el SoC procede a enviar la señal $\hat{x}$ obtenida por red al GUI. Cuando termina la trasmisión, la interfaz envía un nuevo comando (*SNDT*) solicitando los tiempos medidos con el timer. Entonces, el sistema responde con tal información.

[3]Conjunto de IPs asignables.

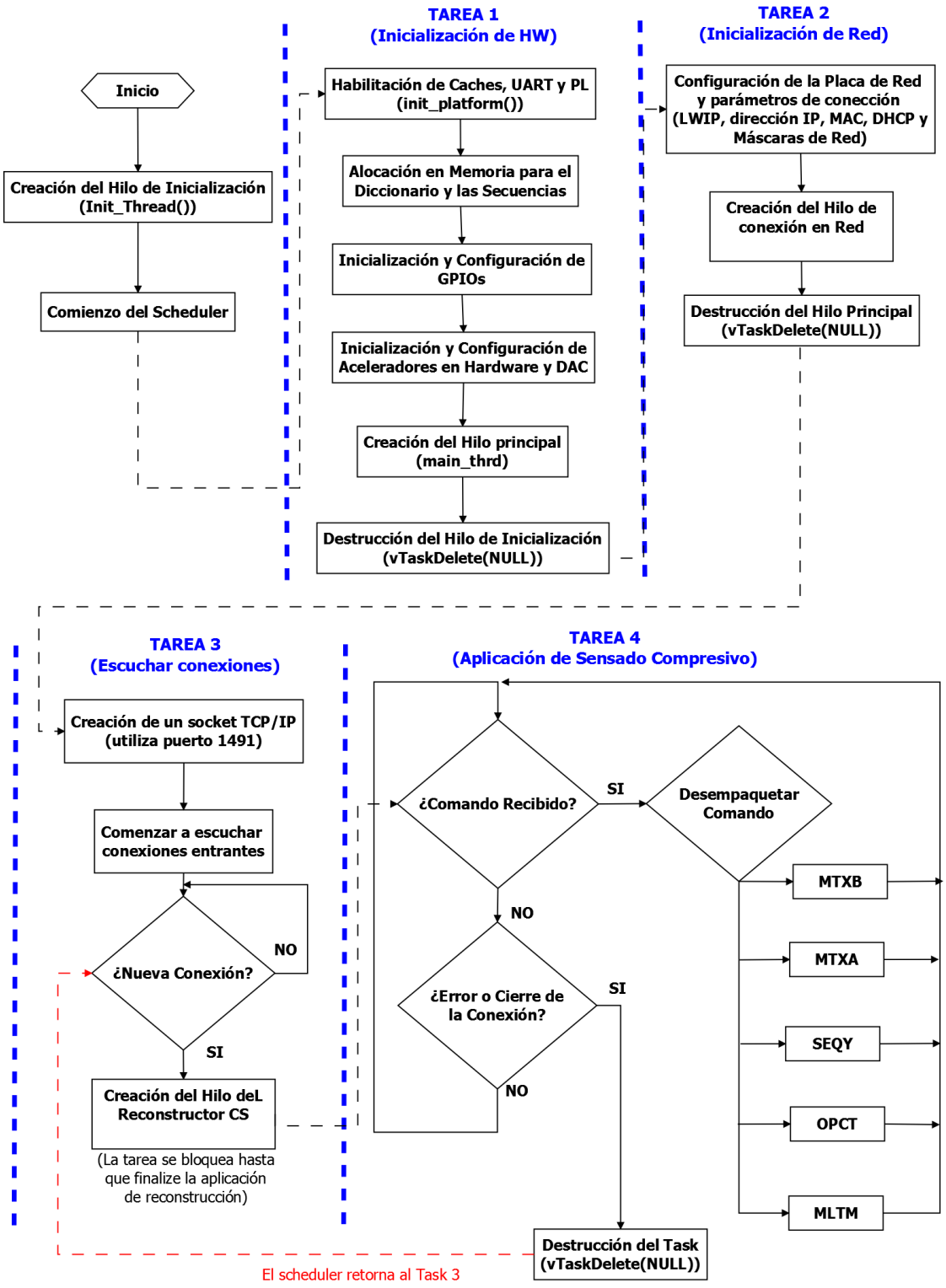
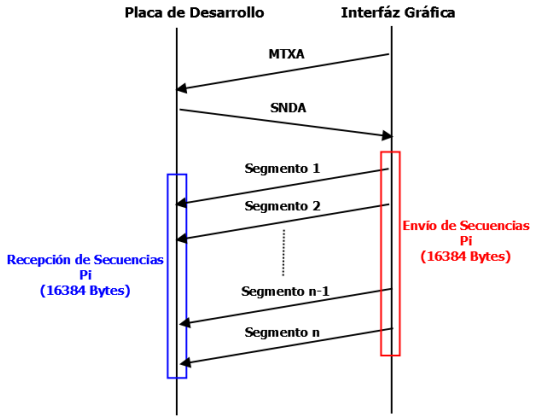
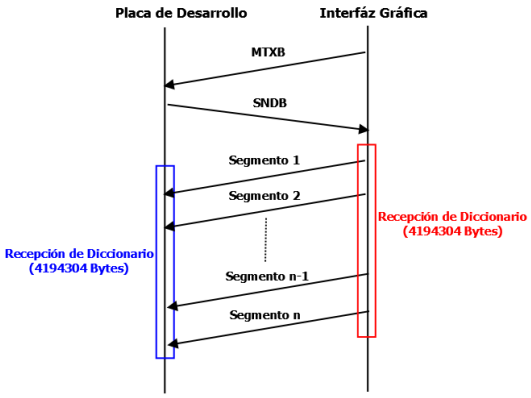


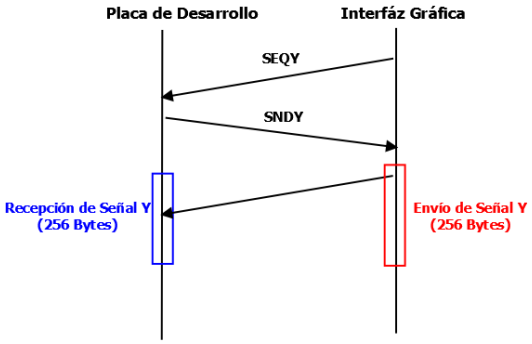
Figura 5.4: Diagrama de Flujo del software principal desarrollado para el SoC.



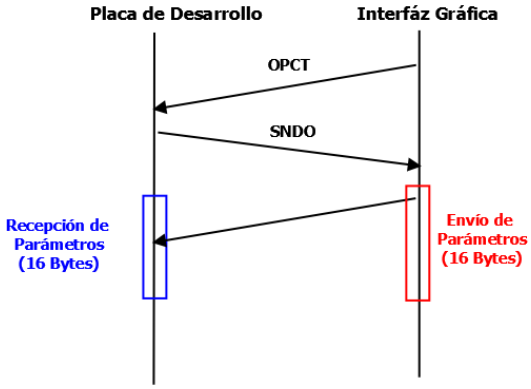
(a) Transmisión de secuencias P_i.



(b) Transmisión de Diccionario Ψ.



(c) Transmisión de señal y.



(d) Transmisión de parámetros para el reconstructor AIHT.

Figura 5.5: Comunicación en red entre el SoC y el GUI.

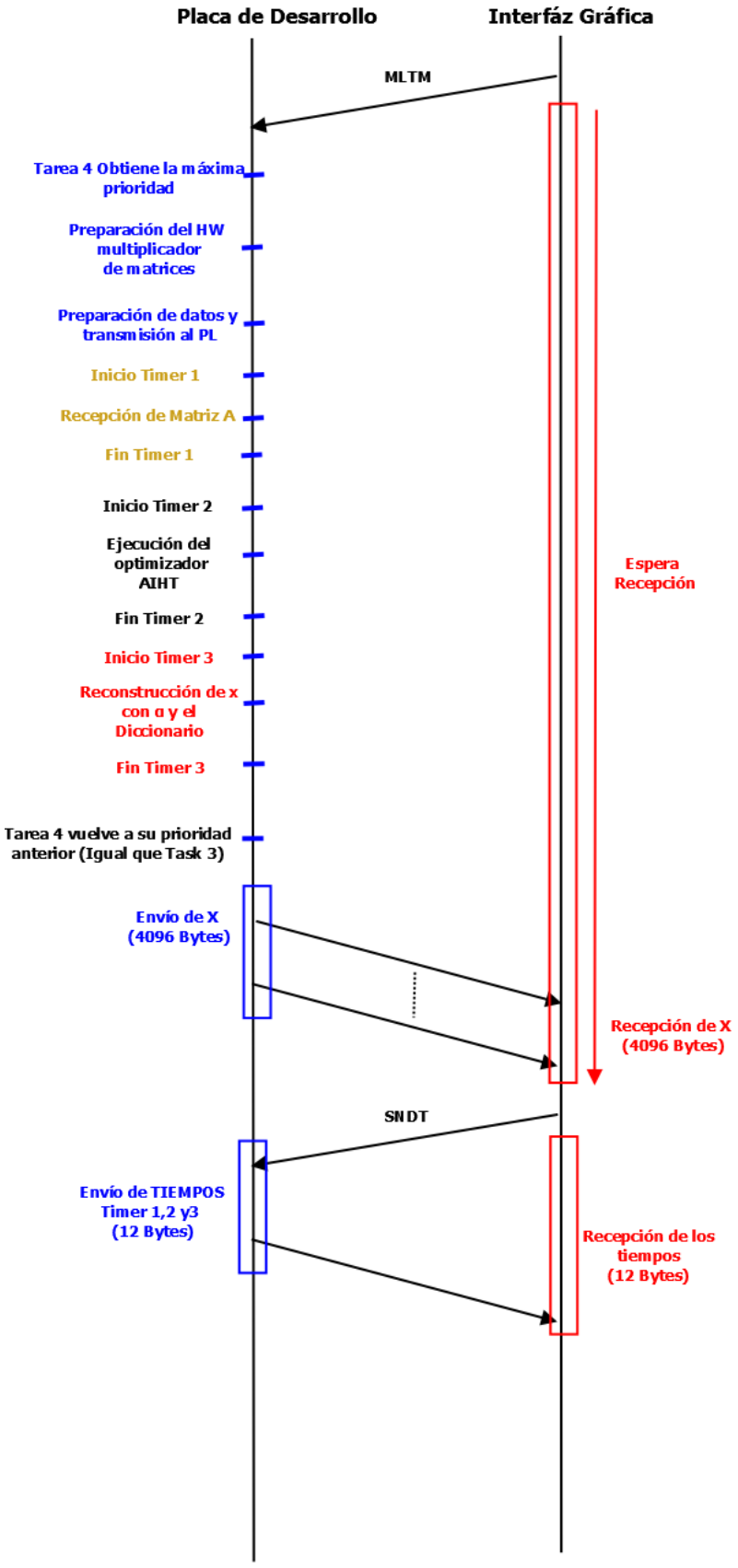


Figura 5.6: Comunicación en red entre el SoC y el GUI cuando se envia el comando de reconstrucción.

5.3. Interfáz Gráfica de Usuario

La aplicación interactiva, desarrollada específicamente para controlar a la Zedboard, se aprecia en las Figuras 5.7 y 5.8. Fue diseñada en el entorno QT Designer (Versión 5.6.0), en donde sólo se establece la ubicación de los botones, pestañas, gráficos, ventanas, mensajes, etc. Para la programación de su funcionamiento, es decir, su *Lógica de Negocios*, primero se debe generar un código en Python (Versión 3.5) a partir del provecto en QT (archivo .py). Luego, la funcionalidad se describe en otro Script; requiriendo importar las librerías *PyQT5* (uso de QT), *math* (para la realización de operaciones matemáticas), *numpy* y *scipy* (operaciones avanzadas, como FFT). Los códigos están disponibles en el Apéndice.

Cuando se ejecuta el GUI, aparece la ventana de la Figura 5.7, mostrando la pestaña RMPI. Dentro de ella se observa una Sección llamada *Demodulador* con los siguientes botones:

- **Cargar Señal**: Abre una ventana emergente en donde se puede seleccionar, un archivo en formato *valores separados por coma* (CSV en Inglés) con extensión .dat que contiene información sobre una señal que se desea procesar con el sistema de CS. La estructura del archivo es la siguiente: los primeros 1024 valores son una señal arbitraria $x[n]$, le siguen 64 valores que son el resultado de submuestrar la señal anterior con el simulador realizado en Matlab del sensor RMPI (señal y[m]) y los últimos 4096 elementos son las secuencias de mezclado que fueron utilizadas en dicho proceso de submuestreo.

- **Adquirir:** Procesa una señal x mediante un RMPI utilizando los valores de *Ganancia Pre-Conversión* y Generador de Secuencia. Esta función se encuentra aún sin implementar[4].

- **Guardar:** Guarda un archivo con los resultados de *Adquirir* siguiendo el formato explicado en el primer ítem.

A la izquierda de la Figura 5.7 se observan dos espacios en negro utilizados para vizualisar señales. En el primero se muestra, sólo para fines comparativos, la señal $x[n]$ que fue leída con el botón *Cargar Señal*. El otro muestra, en amarillo, la señal $y[n]$ a la salida del sensor RMPI. Esta última, junto con las secuencias P_i, son las que serán enviadas al sistema de procesamiento.

En la zona derecha de la interfaz hay tres secciones comunes para todas las pestañas:

1. **Conexión en Red:** Consiste en un botón con la lectura *Conectado/Desconectado* que sirve para iniciar la conexión en red en la dirección y puerto de la placa. En celeste se muestra el estado actual de la comunicación.

[4]El GUI se encuentra en la etapa Alfa de desarrollo, es decir, sólo se realizaron las funcionalidades necesarias para la evaluación del sistema.

2. **Control:** Los tres botones superiores permiten cargar individualmente secuencias Pi, diccionario Ψ y vector de submuestras $y[m]$; todos en formato CSV con extensión *.dat*. Las teclas de abajo envían los datos al SoC.

3. **Optimizador:** Aquí se configuran los parámetros para el optimizador. Permite cambiar los valores escribiendo con el teclado o presionando los cursores deslizantes a la derecha. El botón *Actualizar* procede a enviar el estado actual de la configuración, mientras que *Reconstruct!* envía la orden al sistema para proceder con la reconstrucción. Las opciones *Mostrar Estadísticas* y *Modo Debug* sirven para mostrar información adicional en un terminal serie acerca del sistema embebido. Se activan haciendo click sobre ellas.

Una vez que los datos son enviados al SoC y se procede a la reconstrucción, es necesario moverse a la pestaña *Main* (Figura 5.8). En ellan se muestra ahora las nuevas regiones:

- **Tiempo Requerido:** Presenta los tiempos (en *milisegundos*) que demoró el sistema en calcular la matriz A, realizar la optimización y finalmente obtener la señal reconstruida $\hat{x}$.

- **FPGA Acceleration:** Contiene tres botones seleccionables cuya habilitación significa que el sistema utiliza aceleración en Hardware. El primero habilita el IP multiplicador de matrices, el segundo a la red de ordenamiento y el tercero (no implementado) da lugar para un posible IP para suboptimización (ver Sección 5.4 para más detalles).

- **Comparación:** Calcula los cuantificadores *Similitud Temporal* y *Similitud Espectral* introducidos en el Capítulo 3. Los cálculos los realiza la interfaz a partir de la señal $\hat{x}$ recibida y la señal original x.

- **Señal Original:** Gráfico interactivo con la señal x pre-cargada en el dominio de tiempo. No se utiliza en el procesamiento, sólo sirve para fines comparativos. Permite hacer zoom y mover la señal con el mouse del PC.

- **Señal Reconstruida con CS**: Nuevamente se trata de un gráfico como el anterior, sólo que ahora contiene la señal $\hat{x}$ reconstruida por el SoC a partir de y.

A su vez, se deben destacar cuatro funcionalidades que fueron programadas en esta pestaña:

- **Espectro:** Haciendo click sobre el mismo, la interfaz procede a calcular la FFT y muestra el contenido espectral de ambas señales (Figura 5.9 (a)).

- **Zoom:** Aumenta la base de tiempo en los dos gráficos para apreciar en más detalle las formas de onda (Figura 5.9 (b)).

- **Inter x10:** La interfaz realiza una *interpolación Sinc* (sobremuestreo x10) con el fin de visualizar las señales con mayor suavidad (Figura 5.9 (c)).

- **Superponer:** Desplaza la señal inferior a la ventana superior con el objetivo de comparar punto a punto las dos formas de onda (Figura 5.9 (d)).

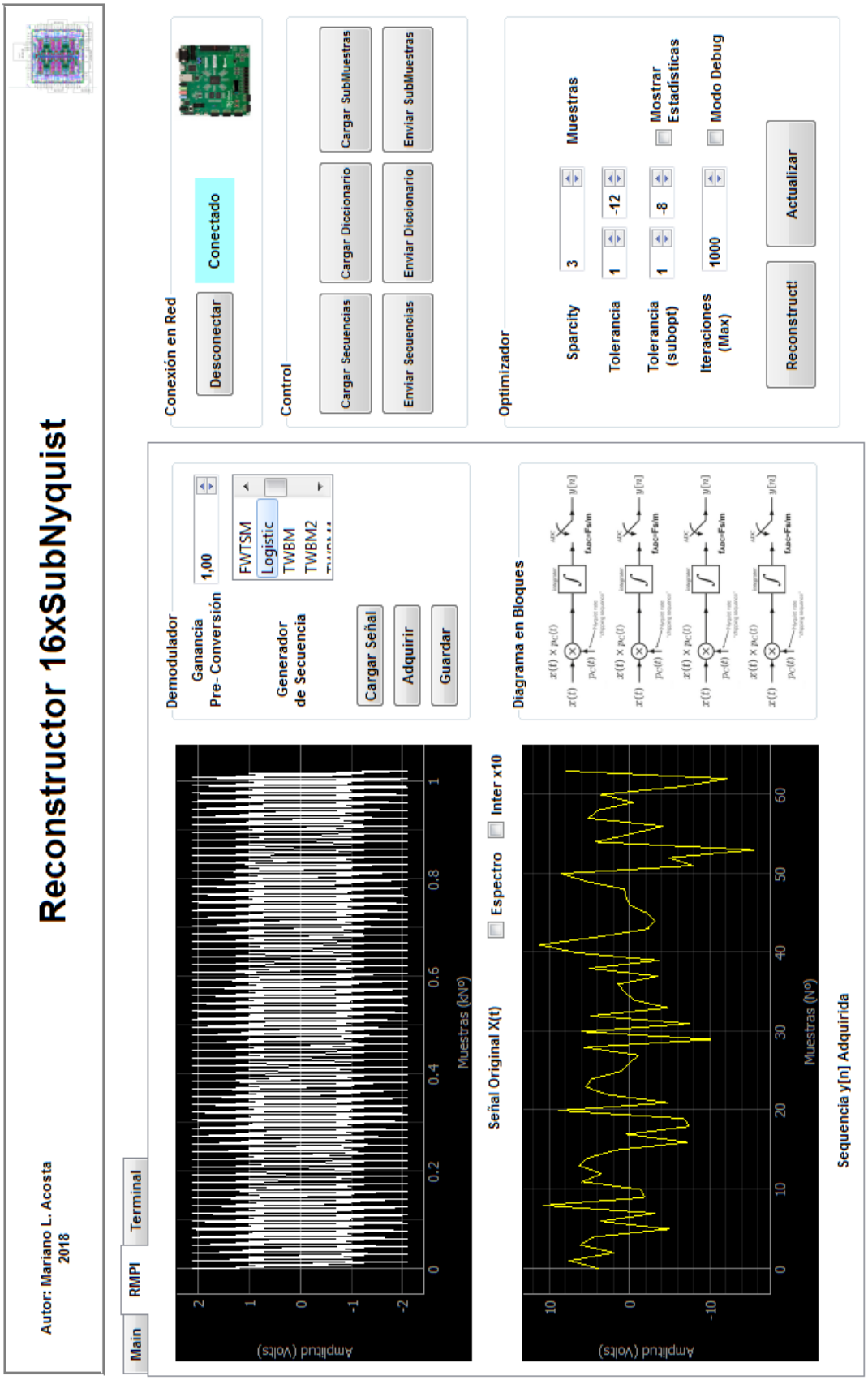


Figura 5.7: GUI Desarrollado (pestaña RMPI).

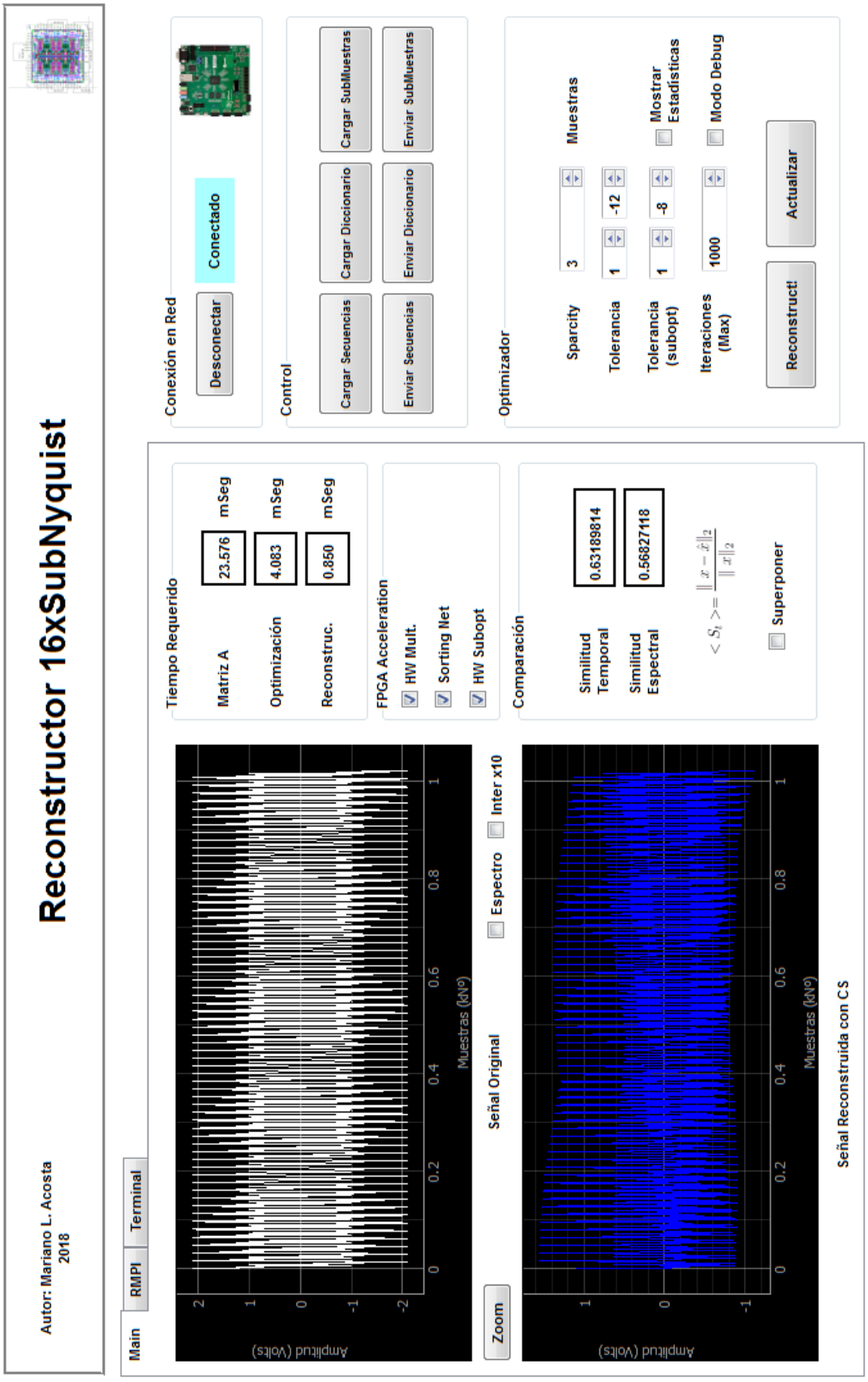
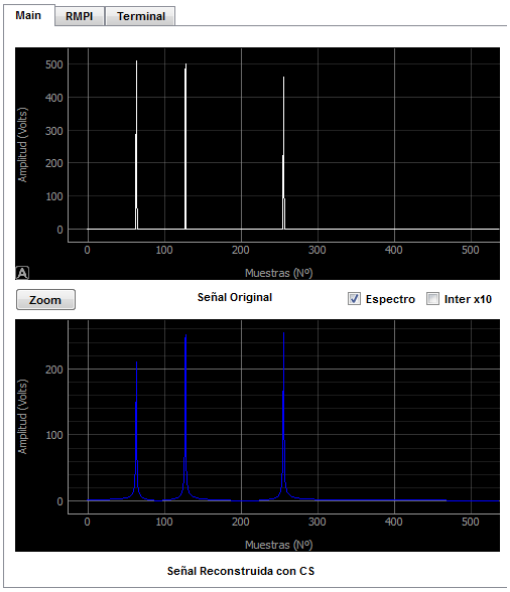
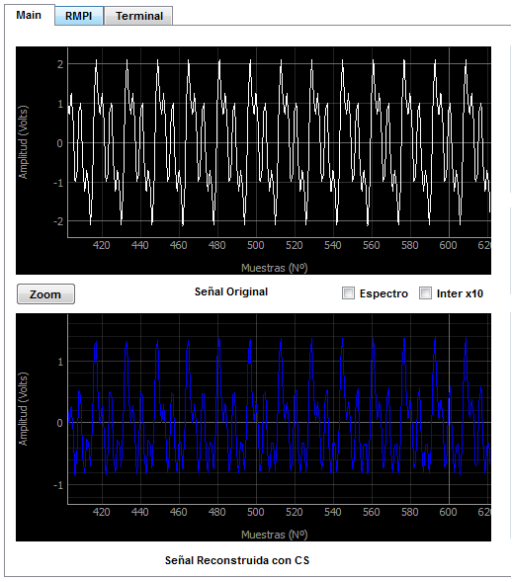


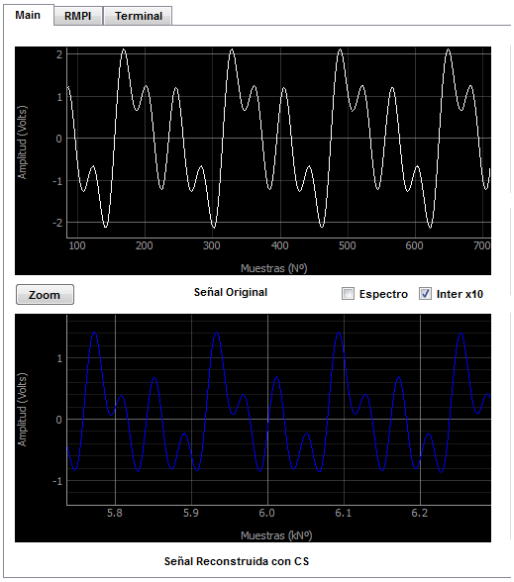
Figura 5.8: GUI Desarrollado (pestaña MAIN).



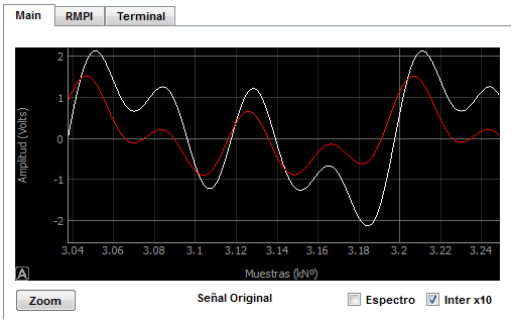
(a) Espectro de ambas señales



(b) Ampliación de la base de tiempo (Zoom)



(c) Interpolación Sinc x10



(d) Superposición de señales.

Figura 5.9: Funcionalidades gráficas para la evaluacion de resultados en el GUI.

5.4. Reconstructor AIHT

Ahora se describirá en detalle el software ejecutado en el SoC que, en conjunción con el hardware personalizado, realiza la reconstrucción de $\hat{x}$ con las señales enviadas por parte del GUI. El código se encuentra en el Apéndice bajo el nombre *Software del Procesador (network.c)*. Se debe recordar que esta parte se ejecuta en el Tarea 4 del sistema.

Con el fin de facilitar la lectura, se vuelven a expresar la operaciones que realizan el algoritmo iterativo de optimización AITH. Sin embargo, tener en cuenta que los fundamentos matemáticos y detalles de operación son explicados en el Capítulo 3 Sección 3.1.3.

$$\Gamma^{(i)} = \text{Soporte}(x^{(i)}) \tag{5.1}$$

$$\mu_i = \frac{\left\| A^T_{\Gamma^{(i)}}(y - A\hat{x}^{(i)}) \right\|_2^2}{\| A_{\Gamma^{(i)}} A^T (y - A\hat{x}^{(i)}) \|_2^2} \tag{5.2}$$

$$\tilde{x}^{(i+1)} = H_k(\hat{x}^{(i)} + \mu_i A^T (y - A\hat{x}^{(i)})) \tag{5.3}$$

$$\mu_j = \frac{\left\| A^T_{\Gamma^{(j)}}(y - A_{\Gamma^{(j)}} \tilde{x}^{(j)}_{\Gamma^{(j)}}) \right\|_2^2}{\left\| A_{\Gamma^{(j)}} A^T_{\Gamma^{(j)}}(y - A_{\Gamma^{(j)}} \tilde{x}^{(i)}_{\Gamma^{(j)}}) \right\|_2^2} \tag{5.4}$$

$$\tilde{x}^{(j+1)}_{\Gamma^{(i)}} = \tilde{x}^{(j)}_{\Gamma^{(i)}} + \mu_j A^T_{\Gamma^{(i)}}(y - A_{\Gamma^{(i)}} \tilde{x}^{(i+1)}_{\Gamma^{(i)}}) \tag{5.5}$$

$$\tilde{x}^{(i+1)}_{\Gamma^{(i)}} = \tilde{x}^{(j+1)}_{\Gamma^{(i)}} \tag{5.6}$$

Siendo $i \in$ el supra-índice que indica el número actual de iteración, $\Gamma^{(i)}$ es el conjunto de índices (llamado soporte) de los $k \in \mathbb{N}$ elementos mayores en modulo del vector $\hat{x}^{(i)}$, A la matriz de sensado, y el vector de submuestras, T indica que la matriz es *transpuesta*,$\|.\|_2^2$ la norma vectorial elevada al cuadrado y H_k el operador de *Hard Thresholding* o *Umbralamiento Duro*. En relación a la notación, $A_{\Gamma^{(i)}}$ significa que la matriz A tiene removidas las columnas cuyo índices no pertenecen a $\Gamma^{(i)}$. De manera análoga, el vector $x_{\Gamma^{(i)}}$ tiene removido los elementos cuyo índices no están en $\Gamma^{(i)}$.

En la iteración *i-ésima*, se realizan en orden las operaciones (5.1), (5.2) y (5.3). Cuando se obtiene $\tilde{x}^{(i+1)}$, se ingresa en otra función optimizadora interna a la principal que ejecuta las operaciones (5.4), (5.5) y (5.6). En ese caso, $j \in \mathbb{N}$ es el número de iteración del subobtimizador. Al finalizar esta rutina interior se reemplaza los elementos no nulos de $\tilde{x}^{(i+1)}$ con $\tilde{x}^{(j+1)}$, obteniéndose finalmente $\hat{x}^{(i+1)}$.

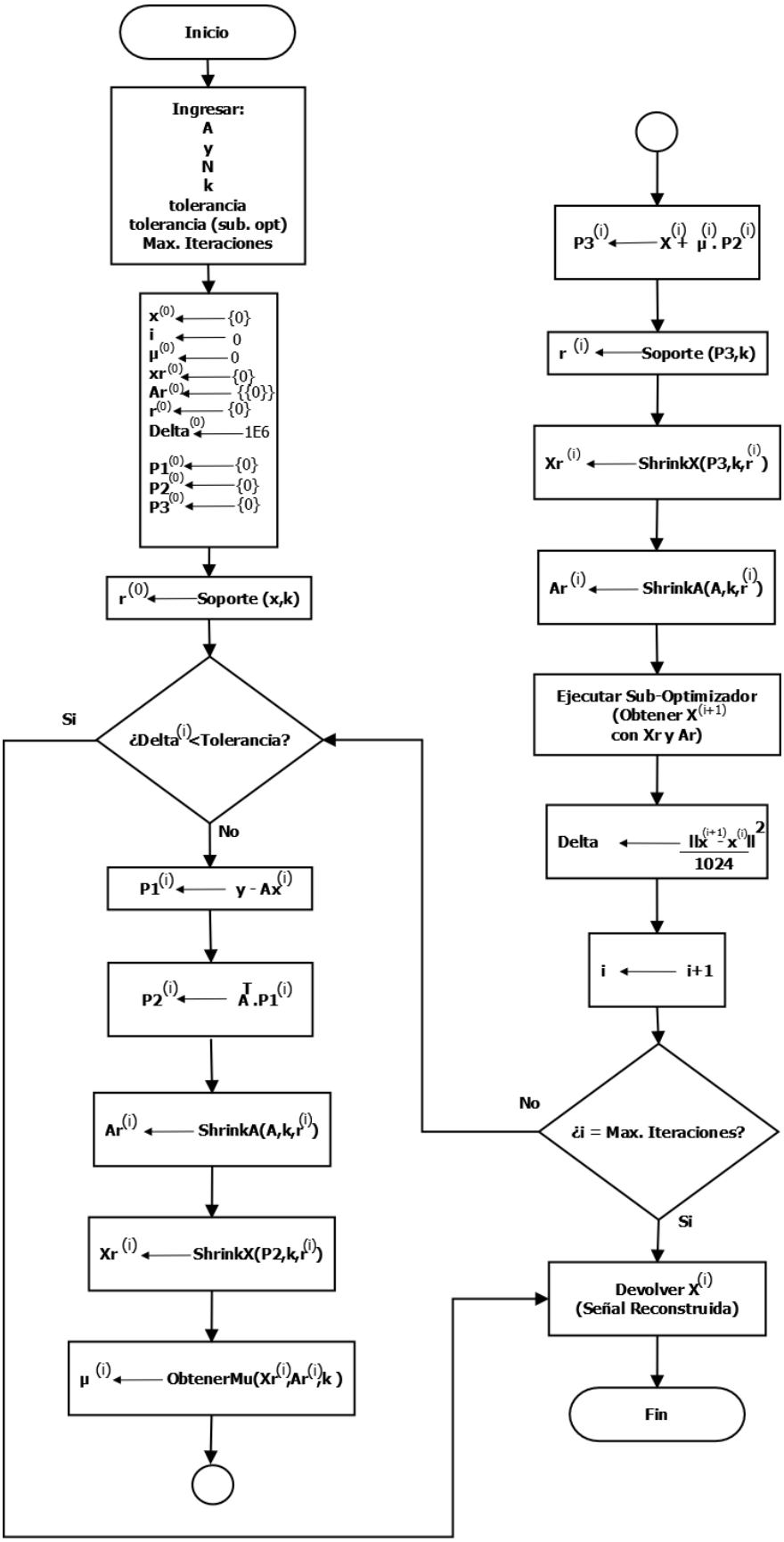


Figura 5.10: Flujo algorítmico del AIHT programado.

5.4.1. Programación en C

El diagrama de flujo del AIHT se encuentra en la Figura 5.10. La función en C *ANIHT()* (archivo *ANIHT.c* del Apéndice) realiza la optimización y tiene las siguientes variables de entrada:

- **float** y[64] (señal de submuestras).
- **float** A[65536] (matriz de sensado A, con sus filas de manera consecutiva).
- **int** N (longitud de x, en este caso N=1024).
- **int** k (cantidad de elementos no nulos de x en la base Ψ, $k \leq 32$).
- **float** tol (tolerancia, mínimo error que se debe obtener entre dos iteraciones consecutivas para detener al algoritmo).
- **float** x (señal x que se desea recuperar, debe estar inicializado con ceros).
- **float** tol_sub (idem con tolerancia, pero ahora con respecto al suboptimizador)
- **int** itMax (Cantidad máxima de iteraciones, para evitar cuelgue del sistema).

Luego, la función define tres arreglos *float* auxiliares P_1 (1x64), P_2 (1x1024) y P_3 (1x1024) para el almacenamiento de resultados intermedios y una variable *Delta*, que guarda la diferencia normalizada entre los resultados de dos iteraciones seguidas. Estos parámetros son inicializados con cero, con excepción de *Delta*, ya que requiere que su valor inicial sea un número grande (1×10^6) para que, por lo menos, la primer iteración se ejecute.

Dentro del lazo *While()* principal se pueden observar las siguientes funciones implementadas:

- **Soporte(x,k):** Devuelve en forma de un vector el conjunto de índices de los k elementos mayores en x. La función se puede configurar para que realice todo sus operaciones en el microprocesador (Software) o utilice la red de ordenamiento (Hardware) diseñada específicamente para reducir el tiempo de cómputo. Si es en software, efectúa estos pasos:
  1. Aplica el valor absoluto al vector x, guardando el resultado en una variable temporal.
  2. Crea una estructura de datos (tipo *strutc* en C) para representar a x con dos campos; uno del tipo **float** para almacenar el dato y otro **int** para almacenar su índice. De esta manera se crea un vector cuyo elementos almacenan el valor numérico y su posición.
  3. Segmenta x en 32 bloques de 32 elementos.
  4. Realiza un ordenamiento por mezcla (*Mergesort* en Inglés) de cada bloque. La razón por la que se escogió este algoritmo de ordenamiento es por su complejidad en el tiempo, siendo *O(nlog(n))* en todos los casos. En la Figura 5.11 se explica en qué consiste.

5. Iterativamente compara dos bloques de 32 elementos guardando los 32 mayores y descartando el resto. La operación se realiza hasta que se haya comparado todos los bloques.
6. Se crea un vector **R** que almacena los índices de los 32 elementos más grandes. Esto es posible gracias a la estructura de datos explicada en 2).
7. Se ordena **R** con *Mergesort* y se devuelven los primeros k índices.

En cambio, si utiliza el acelerador en Hardware:

1. Aplica el valor absoluto al vector x (float), multiplica por 2^{18} (262143), y guarda el resultado en formato entero sin signo (uint32_t) en una variable temporal.
2. Reinicia el acelerador en HW (red de ordenamiento) mediante la escritura en un GPIO del PL.
3. Realiza el envío de los datos (variable temporal) al IP mediante una transferencia DMA de 4096 Bytes (1024 elementos de 4 Bytes cada uno).
4. Espera la recepción de los índices de los 32 elementos mayores del vector por parte del bloque en HW.
5. Ordena los índices por SW mediante *Mergesort*. La primera parte consiste en segmentar al arreglo de entrada en elementos individuales. Seguidamente procede a combinarlos realizando comparaciones.
6. La función devuelve los primeros k índices.

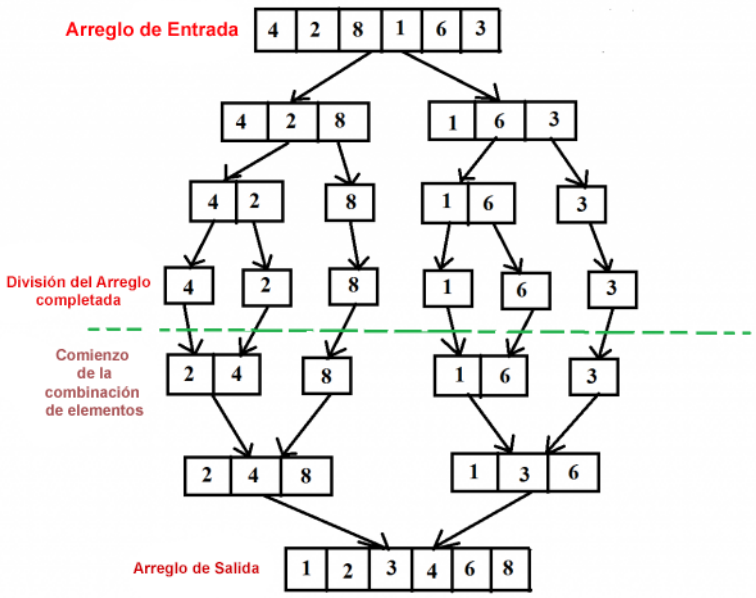


Figura 5.11: Algoritmo *Mergesort* realizado en SW para el ordenamiento de elementos en un vector.

- **ShrinkA(A,k,r)**: Genera la matriz Ar de dimensiones kxN removiendo las columnas de A cuyo índices no se encuentren en r. Sólo tiene en cuenta los primeros k elementos de R.
- **ShrinkX(x,k,r)**: Genera el vector Xr de dimensiones $1xk$ removiendo los elementos de X cuyo índices no se encuentren en r. Sólo tiene en cuenta los primeros k elementos de R.
- **ObtenerMu(Xr,Ar,k)**: Realiza la operación (5.2).

La función *Soporte()* esta definida como *findSupp()* en el archivo *sorting.c*, mientras que el algoritmo *Mergesort* se encuentra en los archivos *mergesort.c* y *mergesort.h*. Todos los programas descriptos hasta aquí se ejecutan en el SoC.

5.4.2. Multiplicaciones

En las partes que el algoritmo requiera realizar multiplicaciones vectoriales cuenta con las funciones programadas *multiplyvecHW()* (uso del IP multiplicador matricial) y *multiplySW()* (uso del FPU Neon). No obstante, en la etapa de desarrollo y pruebas se encontró que, para el caso de multiplicación entre un vector y una matriz, la implementación en software es más eficiente que en Hardware. Esto se debe a que, primero, *multiplySW()* utiliza bloques pequeños de los datos (segmenta los vectores); por lo que toma la ventaja de utilizar el caché del procesador y no una memoria externa[5]. Segundo, las banderas de optimización junto con la directiva de utilizar la unidad de punto flotante NEON reduce considerablemente el tiempo de procesamiento. Por otro lado, si se utiliza la FPGA, se tiene el *Overhead* en preparar, transmitir y recibir los datos por DMA. Para el caso de multiplicaciones vectoriales, una estructura en lógica programable no ofrece ventajas. Aunque, sí se obtienen para multiplicaciones matriciales. Para calcular $\hat{x}$ con $\Psi \times \alpha$, el programa realiza el producto entre filas de Ψ y elementos de α únicamente si este último es no nulo. Con esto se reduce considerablemente el tiempo requerido.

5.4.3. Consideraciones de Programación

Se debe mencionar que fue necesario utilizar dos *palabras claves* [78] en la creación de los códigos.

- *Inline*: Usada en la definición de las funciones que implementan el AIHT (*ANIHT()*) y la multiplicación de $\hat{\alpha}$ con el diccionario Ψ (Recovery()). Fue precisada porque, en caso contrario, el sistema realizaba una copia extra en RAM del diccionario ($\approx 4MBytes$) al momento de llamar a las funciónes. Esto producía el agotamiento del Stack disponible para la tarea del sistema operativo y un consecuente fallo. Esencialmente, esta directiva le ordena al compilador reemplazar la invocación de la función por su definición.

[5]Esto se conoce en Inglés como *Caché Oblivious* [77].

- *Restrict*: Necesaria para indicar al compilador que puede realizar optimizaciones para el procesador NEON. Se usa delante de las definiciones en variables utilizadas para multiplicaciones vectoriales. según la nota de aplicación [67], esta palabra clave avisa que las regiones de memoria delimitadas por los punteros de las variables no se solapan.

5.5. IP Cores Diseñados

5.5.1. Multiplicador Matricial

Para agilizar la construcción de la matriz de sensado A se diseñó un IP en lógica programable llamado *A_GENERATOR*. Antes de proceder con el bloque, se analizará cómo se genera esta matriz. Recordando lo explicado en el Capítulo 2, A se obtiene como el producto entre el kernel de compresión Φ y el Diccionario Ψ. Además, se había expuesto que Φ se construye con las secuencias de mezclado P_i. De forma cualitativa y para su comprensión, la Figura 5.12 muestra cuatro secuencias de mezclado P_1, P_2, P_3 y P_4. segmentándolas en sub-vectores de longitud M para luego agruparlas como muestra la Figura, se haya Φ. La matriz resultante es diagonal por bloques.

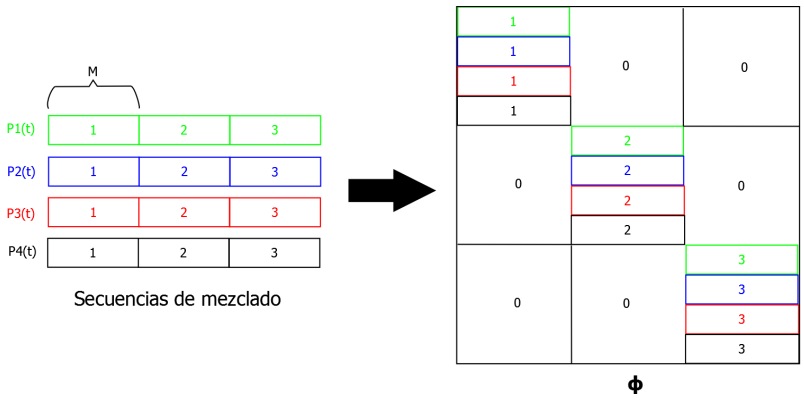


Figura 5.12: Construcción de Φ a partir de las cuatro secuencias de mezclado.

Si se examina ahora el producto $\Phi \times \Psi$ se descubre que existen varias multiplicaciones innecesarias debido a que todos los elementos de Φ fuera de la diagonal son nulos. Siendo así y utilizando álgebra de matrices por bloques, se encuentra que la primer fila de submatrices en A es igual al producto matricial entre el primer bloque en la diagonal de Φ y la primer fila de submatrices en Ψ. La operación es análoga a una multiplicación con una matriz diagonal constituida por elementos escalares. El resultado es representado en la Figura 5.13.

De esta manera, queda en evidencia que la forma óptima de construir A es con un IP como el presentado en la Figura 5.14. Desde el PS es necesario enviar al PL bloques de las secuencias Pi y de A, para luego recibir los resultados parciales y combinarlos obteniendo así a Φ.

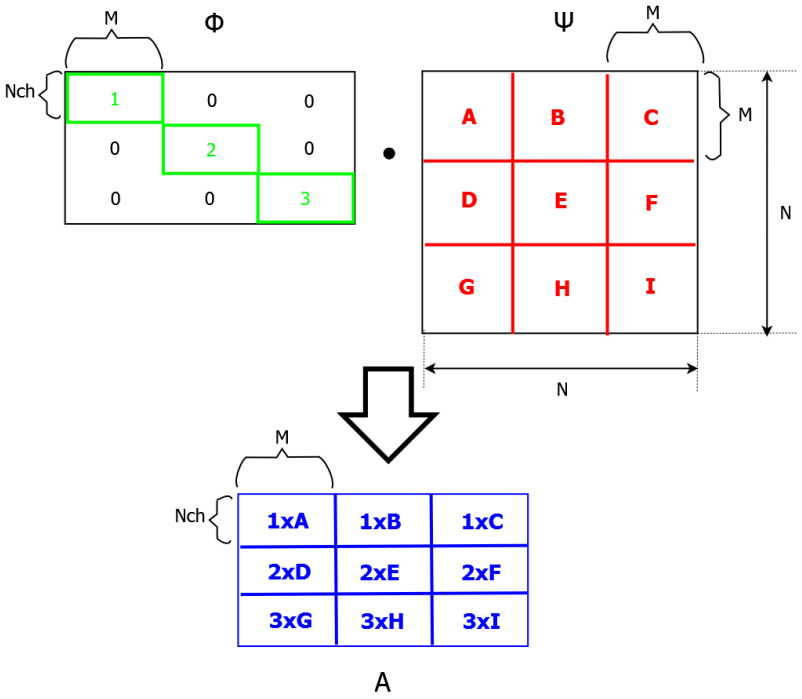


Figura 5.13: Producto entre Φ y Ψ. Se puede realizar mediante submatrices

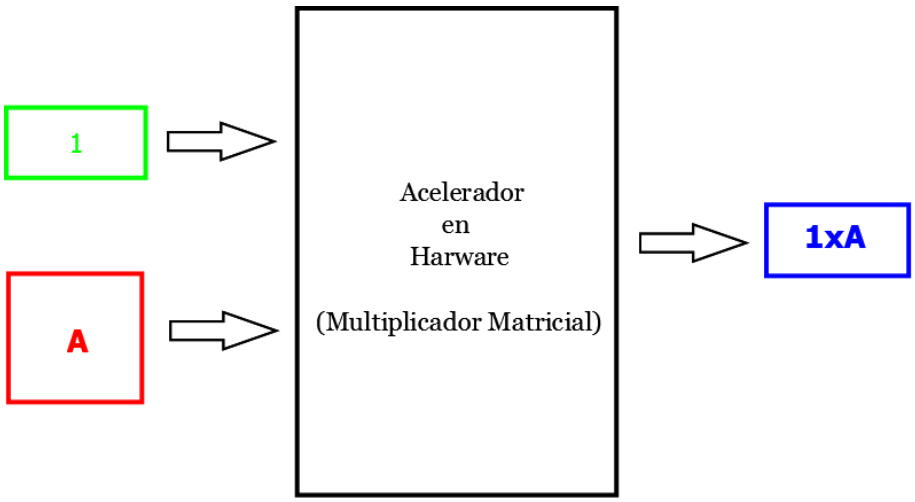


Figura 5.14: IP de multiplicación matricial.

Implementación en Vivado HLS

Como se mencionó en el Capítulo 4, los bloques de lógica programable son diseñados en C++ mediante *Síntesis de Alto Nivel*. El código descriptor de hardware se encuentra en el Apéndice bajo el nombre de *Multiplicador Matricial*. La función principal es la siguiente:

*void A_GENERATOR(stream Aa[NumOfCh*M], stream Bb[M*M], stream Cc[NumOfCh*M], ap_int(1) mode)*

En el archivo *Xampler.h* estan las definiciones:

- **NumOfCh** = 4 (Números de canales)
- **M** = 64 (Longitud de y)

La estructura de datos *stream* contiene dos campos: **float** data (32 bits) y **bool** last. Este último campo es necesario para generar la señal *TLAST* en el protocolo de comunicación [68]. En esencia, las variables *Aa*, *Bb* son mapeadas a puertos de entrada del tipo *AXI Stream* (AXIS) mientras que *Cc* a un puerto de salida, del mismo tipo. Para ello, se deben utilizar los siguientes pragmas para indicarle al sintetizador que se desea generar esta clase de interfaz:

#pragma HLS INTERFACE axis register both port=Cc name=matrixC
#pragma HLS INTERFACE axis register both port=Bb name=matrixB
#pragma HLS INTERFACE axis register both port=Aa name=matrixA

El parámetro *name* otorga un nombre específico a cada puerto. Entonces, el IP recibe por Aa los elementos en forma serie (elemento por elemento) de una matrix 4x64, por Bb otra de 64x64 y devuelve el producto de ambas en Cc. La recepción de datos se encuentra realizada en los lazos *for* con las etiquetas *read_A* y *read_B*. Según se observa en los mismos, las matrices se almacenan en variables llamadas A y B que luego son implementadas como memorias BRAM. También, se puede apreciar una directiva ***#pragma** HLS PIPELINE* para sintetizar el lazo de lectura con Pipeline. Para realizar una determinada acción, como la lectura de información, se requiere cierta cantidad de pasos. Pipeline consiste en llevar a cabo en forma paralela pasos de diferentes acciones con el fin de reducir el tiempo de procesamiento.

Otras directivas presentes en el código son ***#pragma** HLS DATAFLOW*,***#pragma** HLS ARRAY_MAP* y ***#pragma** HLS ARRAY_ RESHAPE*. La primera da la orden al generador de RTL que analice los procesos secuenciales que se realizan en el sistema e implemente concurrencia mediante Pipeline. Un ejemplo de ello sería recibir nuevos datos mientras los anteriores se siguen procesando. Los otros dos pragmas son usadas en las variables A y B. Sirven para dividir la memoria destinada para su almacenamiento en bloques menores y con dimensiones re-ajustadas con el fin de evitar cuellos de botellas en el acceso a los datos. Con esto se mejora la eficiencia total.

Una vez cargadas las matrices, su multiplicación se realiza en los lazos *for* llamados *L1*, *L2* y *L3*:

- **L1:** Lazo externo, recorre a A por filas.
- **L2:** Lazo intermedio, recorre a B por columnas y transmite el cálculo de un elemento de C. Si se trata del último elemento a enviar, pone el campo *last* en alto indicando el fin de la transmisión. A su vez, contiene el pragma *PIPELINE*; implicando que el lazo interno L3 sera realizado totalmente en forma paralela.
- **L3:** Lazo interno, realiza la multiplicación y acumulación.

Síntesis

Los resultados de *timing* y *latencia* obtenidos luego de la generación del código RTL se muestra en la Figura 5.16. El período máximo estimado para un ciclo de reloj resulta 8,43 nseg, con una incerteza de 1,25 nseg. Por lo tanto, el reloj del FPGA con frecuencia $100MHz$ (10 nseg de período) resulta adecuado para señal de sincronismo. A su vez, el tiempo en procesar las dos matrices (*latencia*) es de 6466 ciclos o 64,66 useg. Gracias a la directiva *DATAFLOW*, el intervalo para enviar las matrices es de 4099 ciclos.

El diseño requiere para su implementación la cantidad de recursos que muestra la Figura 5.15. Entre ellos, necesita 32 bloques de memoria, 40 unidades DSP para multiplicación/acumulación y el 28 % de las LUTs disponibles. Vivado HLS en la generación del IP también determinó tres operaciones o *pasos de control* principales (Figura 5.17):

- Lectura de los datos (*Loop_read_A* y *Loop_read_B*). Consiste en dos operaciones. Ambas matrices de entrada se leen de forma paralela, debido a que existe un puerto de entrada dedicado para cada una de ellas.
- Multiplicación Matricial (*Loop_L1_production*).

De manera complementaria, en la Figura 5.18 se puede ver para cada operación definida anteriormente la latencia en ciclos de reloj que demora cada una, junto con los recursos que se le asignaron.

Utilization Estimates

Summary

Name	BRAM_18K	DSP48E	FF	LUT
DSP	-	-	-	-
Expression	-	-	0	15
FIFO	-	-	-	-
Instance	-	40	7316	15322
Memory	32	-	0	0
Multiplexer	-	-	-	6
Register	-	-	6	-
Total	32	40	7322	15343
Available	280	220	106400	53200
Utilization (%)	11	18	6	28

Figura 5.15: Recursos necesarios para implementar el acelerador.

Performance Estimates

Timing (ns)

Summary

Clock	Target	Estimated	Uncertainty
ap_clk	10.00	8.43	1.25

Latency (clock cycles)

Summary

Latency		Interval		
min	max	min	max	Type
6466	6466	4099	4099	dataflow

Figura 5.16: Resultado de timing.

Current Module : A_GENERATOR

	Operation\Control Step	C0	C1	C2	C3	C4	C5
1	Loop_read_A_out_proc...						
2	Loop_read_B_out_proc...						
3	Loop_L1_proc(function)						

Figura 5.17: Operaciones de control en el IP.

	BRAM	DSP	FF	LUT	Latency	Interval	Pipeline type
A_GENERATOR	32	40	7322	15343	6466	4099	dataflow
Loop_L1_proc	0	40	7080	10432	2367	2367	none
Loop_read_B_out_proc	0	0	126	2463	4098	4098	none
Loop_read_A_out_proc	0	0	110	2427	258	258	none

Figura 5.18: Recursos asignados a cada operación de control.

Por último, en el diseño final (Figura 5.19), existe un bus de control (*ap_ctrl*) para iniciar el bloque mediante *ap_start*, los puertos de datos *matrixA*, *matrixB* y *matrixC*, la entrada para la señal de reloj *ap_clk* y el pin *ap_rst_n* activo bajo para resetear el bloque. La entrada *mode_V* pertenece a una funcionalidad no implementada que permitiría seleccionar entre modos de operaciones (multiplicación matricial vs multiplicación vectorial).

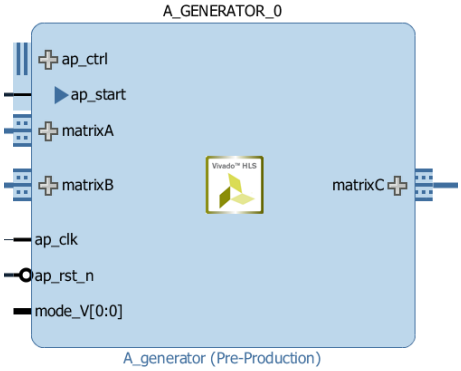


Figura 5.19: Bloque *A_GENERATOR* generado con HLS.

5.5.2. Red de Ordenamiento

El segundo acelerador en hardware desarrollado es un IP que efectúa ordenamiento de números en forma paralela. Se utiliza en el procedimiento del *Umbralamiento Duro*, de ahí deriva nombre del elemento *HARD_THRESH*. Se encarga de recibir un vector de longitud 1024 y devolver el conjunto de índices de los 32 elementos mayores. Este diseño es original y se siguieron las indicaciones de [79].

Implementación en Vivado HLS

Con el fin de describir funcionalmente este sistema, se muestra su diagrama en bloques en la Figura 5.20. Primero, el diseño recibe un vector de 1024 elementos enteros sin signo de 32 bits mediante una interfaz AXI Stream. Luego, se procede a agregar el índice correspondiente (posición del elemento en el vector) como cabecera de 10 bits a cada dato (ver cuadro formato de datos en la Figura 5.20). Seguidamente, el vector ahora de 42 bits por 1024 se almacena en una memoria RAM. La máquina de estado, que controla la secuencia de pasos desde la recepción, procesamiento y transmisión, envía en bloques de 128 los datos a una red de ordenamiento en paralelo (Figura 5.21). En la misma se puede apreciar que los elementos se almacenan en un registro y en cada ciclo de reloj se procede a efectuar comparaciones de manera concurrente. El elemento comparador siempre devuelve el dato de mayor magnitud por su terminal *geq*. Si ambos son iguales, entonces no procede a hacer el intercambio. Como puede observarse, existe dos niveles: 64 comparadores sucedidos por 62. A su vez, el sistema esta realimentado, por lo que luego de cada etapa de ordenamiento el resultado se vuelve a guardar en el registro. Iterando 64 veces, el segmento siempre queda ordenado; de mayor a menor.

Sin embargo, en cada proceso de ordenamiento de un bloque de 128 datos, sólo se almacenan los 32 mayores y se descarta el resto. Es decir, después de utilizar la red 8 veces, se tendrá 256 elementos restantes. Posteriormente, se vuelve a dividir al remanente en dos partes iguales para ordenarlas. Como resultado, se obtiene los 64 elementos mayores divididos en dos grupos ordenados de 32. Gráficamente, este procedimiento descripto se muestra en la Figura 5.22.

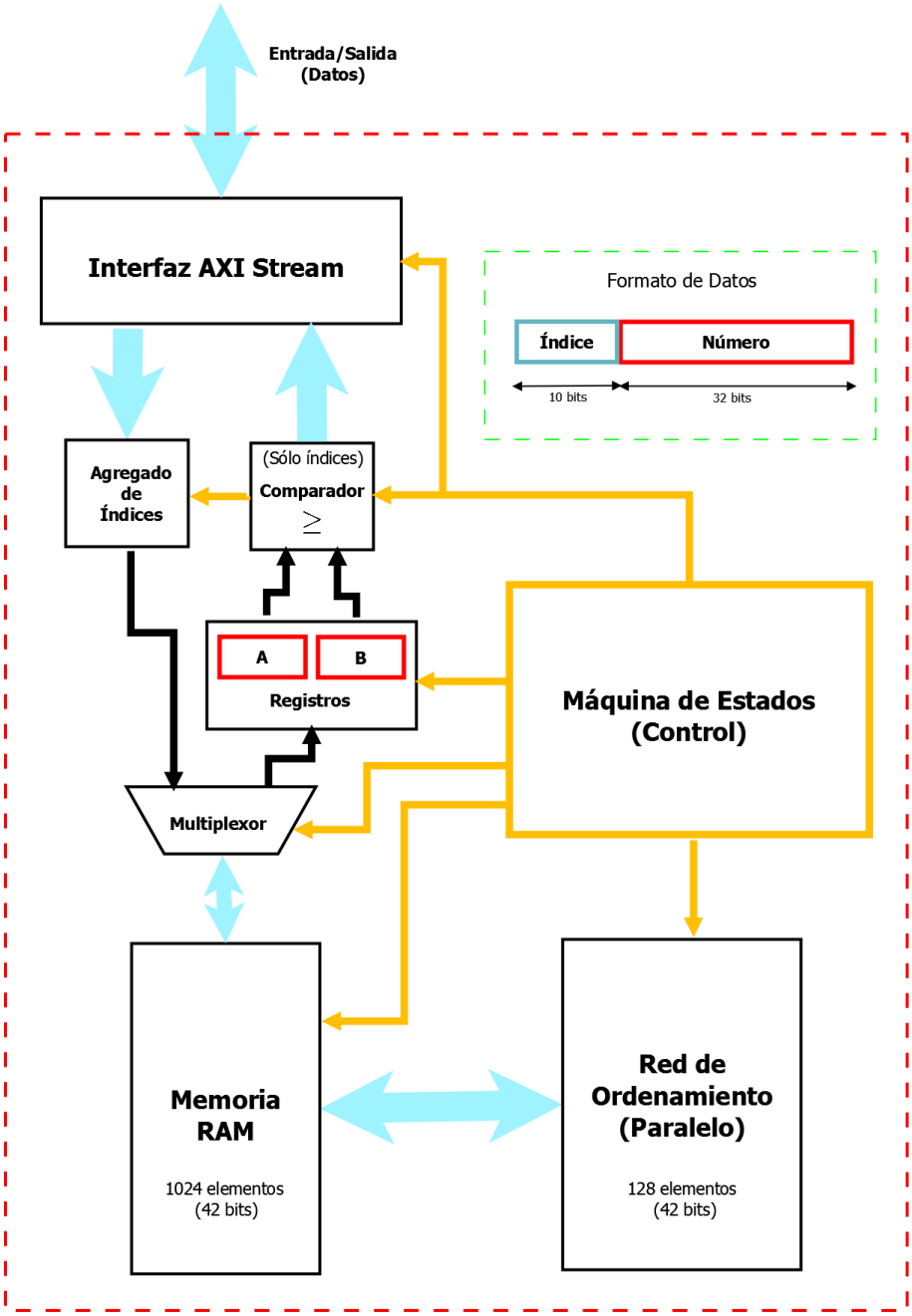


Figura 5.20: Diagrama en Bloques para el IP de ordenamiento.

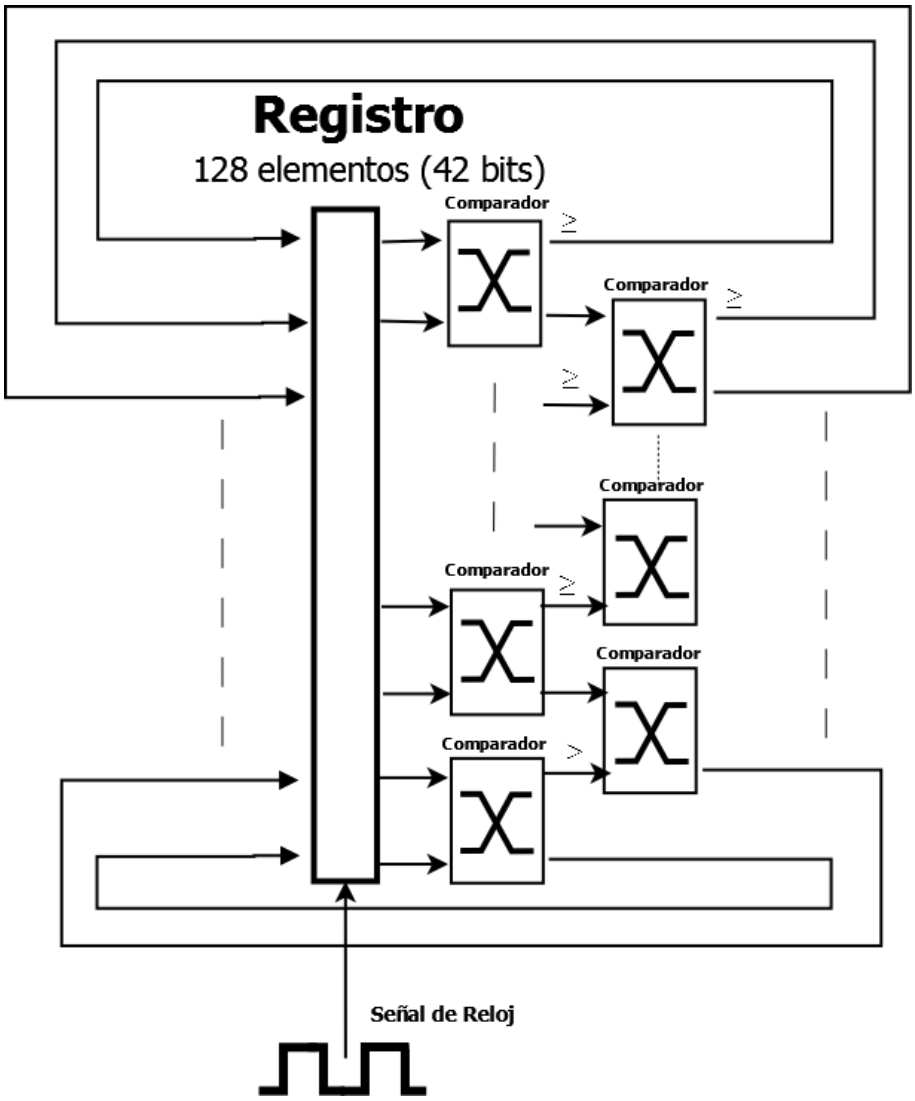


Figura 5.21: Interior del bloque *Red de Ordenamiento (Paralelo).*

Una vez que se obtienen los dos conjuntos de 32 datos, se procede a cargar los registros A y B con el primer dato de cada grupo. Ambos números son comparados y se transmite a la salida el índice (los primeros 10 bits) del elemento mayor o igual. El registro que contenía el valor cuyo índice fue transmitido es reemplazado por uno nuevo perteneciente al mismo conjunto que el anterior. De esta manera, luego de repetir este proceso 32 veces, se envía la ubicación de los 32 números mayores.

Nuevamente se vuelven a utilizar los pragmas *HLS INTERFACE axis register* para indicar que se quiere realizar una comunicación de datos mediante AXI Stream y en particular *HLS ARRAY_PARTITION* para permitir la lectura en paralelo de las memorias. No obstante, se hace uso de la sentencia ***#pragma** HLS INLINE recursive* para que la función *Comp* (que realiza las comparaciones) no se sintetice como un bloque aparte sino como varios elementos dentro de la red. Los lazos *for* llamados *sort_1* y *sort_2* tienen incluidos ***#pragma** HLS UNROLL* lo que indica al sintetizador que implemente todas las comparaciones en forma paralela.

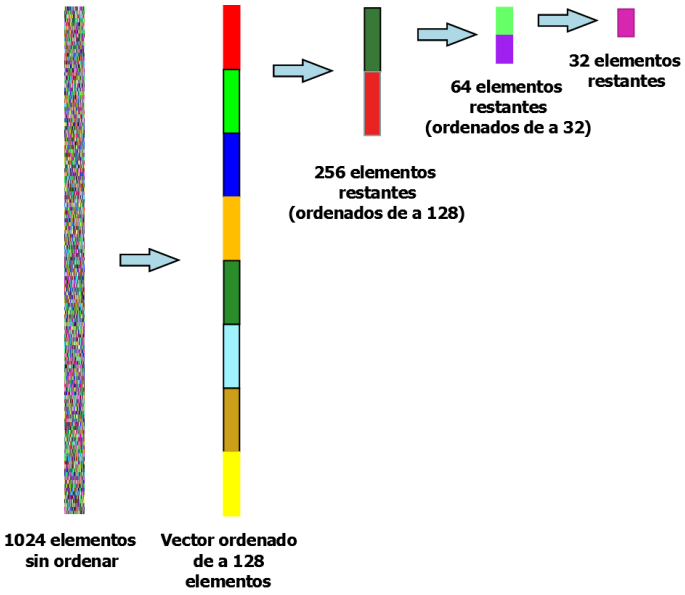


Figura 5.22: Procedimiento de ordenamiento. Se obtienen los 32 elementos mayores.

Síntesis

Los resultados de timing se muestran en la Figura 5.23. Ahora, el período máximo de señal de reloj es 7,78 nseg y el retardo, medido en ciclos, varía entre 3108 (31,08 useg) hasta 5580 (55,80 useg). Si el vector esta completamente ordenado (mejor caso) se tarda el tiempo mínimo. En caso contrario, el máximo. La red de ordenamiento utiliza el 55 % de las LUTs, 3 bloques de memoria BRAM y ningún elemento DSP (Figura 5.24). El bloque generado (Figura 5.25) presenta el mismo bus de control *ap_ctrl* que el multiplicador matricial, la entrada de datos serie *data_in* del tipo AXIS, reset *ap_rst_n* activo bajo, entrada de reloj *ap_clk* y salida de datos *data_out*.

Performance Estimates

Timing (ns)

Summary

Clock	Target	Estimated	Uncertainty
ap_clk	10.00	7.78	1.25

Latency (clock cycles)

Summary

Latency		Interval		
min	max	min	max	Type
3107	5879	3108	5880	none

Figura 5.23: Resultado de timing.

Utilization Estimates

Summary

Name	BRAM_18K	DSP48E	FF	LUT
DSP	-	-	-	-
Expression	-	-	0	12217
FIFO	-	-	-	-
Instance	-	-	0	340
Memory	3	-	0	0
Multiplexer	-	-	-	16707
Register	-	-	16517	-
Total	3	0	16517	29264
Available	280	220	106400	53200
Utilization (%)	1	0	15	55

Figura 5.24: Recursos necesarios para implementar el acelerador.

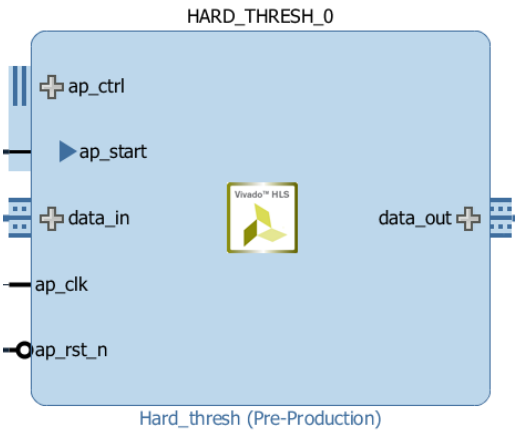


Figura 5.25: Bloque *HARD_THRESH* generado con HLS.

5.6. Diseño en Vivado Design Suite

Una vez que fueron sintetizados los IP de hardware en Vivado HLS se exportó su código RTL al entorno Vivado Design Suite, en donde se armó el diseño en bloques de la Figura 5.26. Principalmente, se pueden contemplar los siguientes elementos:

- **ZYNQ7 Processing System:** Es el PS del SoC. Se habilitaron los puertos de comunicación ACP junto con los HP0, HP1, HP2 y MGP0 (sólo para configurar registros). La opción *TieOff AxUSER* fue activada para el correcto funcionamiento del ACP en el acceso a los caché. La salida *DDR* esta conectada a las memorias RAM externa y *FIXED_IO* a los periféricos de entrada/salida.

- **AXI_Interconnect:** Conjuntamente con *ps7_0_axi_periph*, hace de traductor entre interfaces AXI mapeadas en memoria y AXI Stream (Figura 5.27).

- **Bloques AXI_DMA:** Son controladores de acceso directo a memoria implementados en el FPGA. Los *axi_dma_0* y *axi_dma_1* están conectados a los puertos HP0 y HP1 del PS respectivamente. Son los encargados de enviar dos matrices en paralelo del *uP* al multiplicador matricial. Por otro lado, *axi_dma_2* recibe el resultado de la multiplicación y lo envía al PS por HP2. El *axi_dma_3* cumple tanto la función de transmisión y recepción de datos. Este elemento en particular se comunica con el ACP para un acceso rápido de la información en los caché. Internamente, todos los DMA están configurados de la siguiente manera:

  - **Tamaño del registro del Buffer de longitud**: 23 bits. Necesario para hacer transferencias DMA de gran cantidad de datos.
  - **Longitud de direcciones**: 32 bits.
  - **Longitud de datos del mapa de memori**a: 32 bits.
  - R**esolución del stream de datos**: 32 bits.
  - T**amaño de ráfaga**: 256 (máximo).

  Sus puertos *S_AXI_LITE* están unidos al MGP0 del procesador para configuración mediante escritura en registros por software.

- **AXI Switch:** Realiza la conmutación de datos AXI Stream y es comandada por software. Conecta al *axi_dma_3* con la red de ordenamiento o con el conversor digital/analógico.

- **AXI GPIO:** Piezas generadas en la lógica programable que permiten mediante software reiniciar los IP (Figura 5.28). Todos poseen una línea de datos con un 1 bit de longitud. Se aprecia que *axi_gpio_1* esta conectado al reset de *A_GENERATOR*, en tanto *axi_gpio_3* al reset de *HARD_THRESH* y *axi_gpio_4* al del DAC.

- **Xlconstant:** Cumplen la finalidad de entregar un valor lógico constante (alto) a todas las entradas *ap_start*. Con eso se consigue que los IP se encuentren habilitados para su funcionamiento.

- **rst_ps7_0_100M:** Necesario para reiniciar el sistema.
- **Aceleradores en Hardware:** Consiste en el Multiplicador Matricial (*A_GENERATOR*) y la red de ordenamiento (*HARD_THRESH*). Asisten al software que se ejecuta en el PS a reducir el tiempo total de procesamiento gracias a su capacidad de trabajar en paralelo. Presentan el logo de Vivado en amarillo.
- **Conversor D/A:** IP generado en VHDL que recibe la señal reconstruida $\hat{x}$ enviada por el PS y la almacena en un buffer circular. Envía periódicamente una modulación por densidad de pulso (PDM por sus siglas en Inglés) que contiene la información de esta señal a un puerto PMOD. Exteriormente, se encuentra un filtro pasabajos que elimina las componentes en alta frecuencia del PDM; produciendo así una versión de voltaje (analógica) de $\hat{x}$. Ya que el diseño del DAC es complejo, se dedica todo el Capítulo 6 a su explicación.

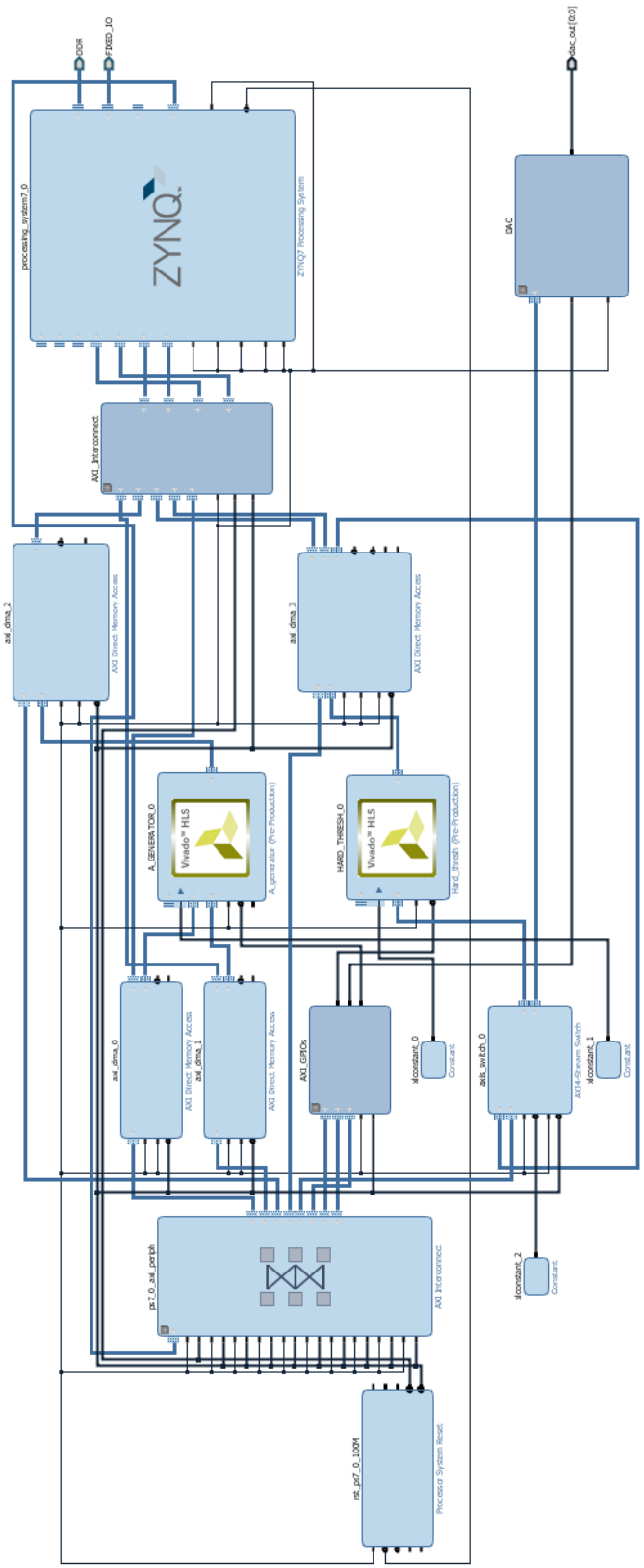


Figura 5.26: Diseño en bloque del Reconstructor CS realizado en Vivado Design Suite.

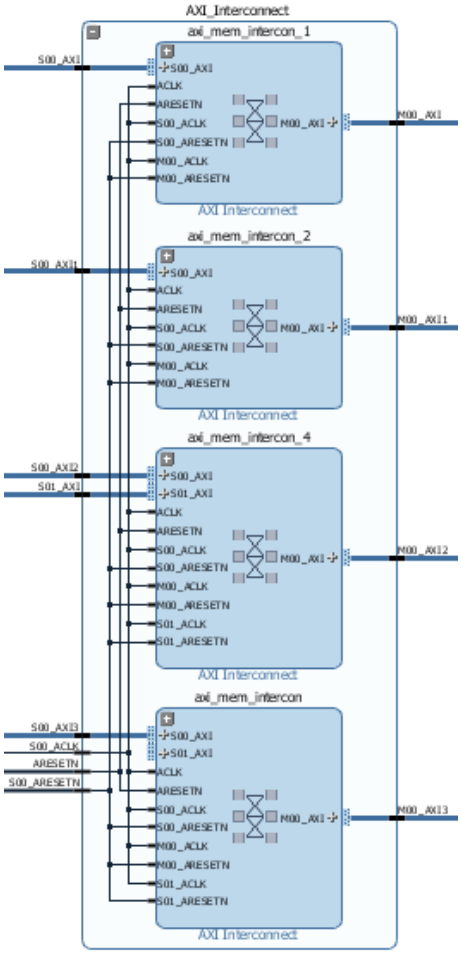


Figura 5.27: Interior del Elemento *AXI_Interconnect*.

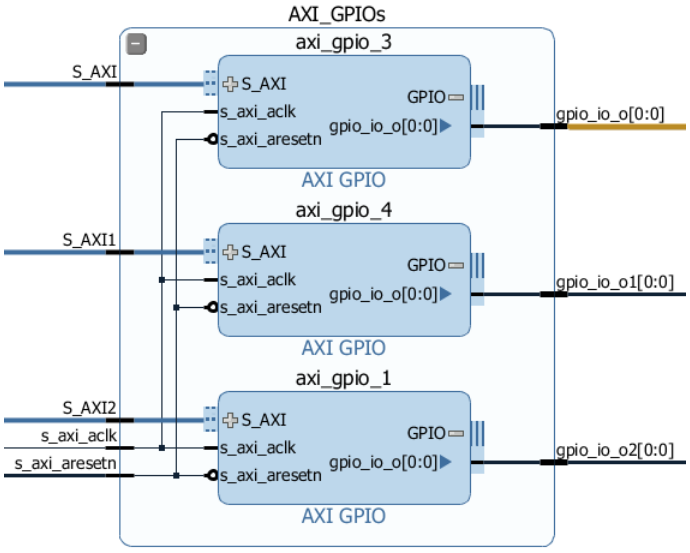


Figura 5.28: Interior del Elemento *AXI_GPIOs*.

5.6.1. Espacio de Direcciones

Vivado asigna un conjunto de direcciones a los DMA y GPIOS para que puedan ser accedidos por el procesador mediante los puertos ACP, HP, y GP. La tabla presentada en la Figura 5.30 muestra que los DMA se encuentra en la región comprendida entre 0x0000_0000 y 0x1FFF_FFFF (512 millones). Se podría pensar que, al estar las direcciones superpuestas, no es posible distinguir los tres dispositivos. Sin embargo, cada uno de los DMA se encuentra asignado a un puerto independiente; por lo que su acceso es en forma paralela. Las interfaces AXI_LITE y full AXIO se encargan de transmitir parámetros de configuración en el primer caso y escritura de GPIO en el segundo. La zona en este caso va de 0x4040_0000 a 0x43C0_FFFF. Existe también regiones excluidas del direccionamiento (*Excluded Address segments*) para el ACP, aunque, no son relevante para el correcto funcionamiento del sistema.

processing_system7_0					
Data (32 address bits : 0x40000000 [1G])					
axi_dma_0	S_AXI_LITE	Reg	0x4040_0000	64K	0x4040_FFFF
axi_dma_1	S_AXI_LITE	Reg	0x4041_0000	64K	0x4041_FFFF
axi_dma_2	S_AXI_LITE	Reg	0x4042_0000	64K	0x4042_FFFF
axi_dma_3	S_AXI_LITE	Reg	0x4043_0000	64K	0x4043_FFFF
AXI_GPIOs/axi_gpio_1	S_AXI	Reg	0x4120_0000	64K	0x4120_FFFF
AXI_GPIOs/axi_gpio_3	S_AXI	Reg	0x4121_0000	64K	0x4121_FFFF
AXI_GPIOs/axi_gpio_4	S_AXI	Reg	0x4122_0000	64K	0x4122_FFFF
axis_switch_0	S_AXI_CTRL	Reg	0x43C0_0000	64K	0x43C0_FFFF
axi_dma_0					
Data_MM2S (32 address bits : 4G)					
processing_system7_0	S_AXI_HP0	HP0_DDR_LOWOCM	0x0000_0000	512M	0x1FFF_FFFF
axi_dma_1					
Data_MM2S (32 address bits : 4G)					
processing_system7_0	S_AXI_HP1	HP1_DDR_LOWOCM	0x0000_0000	512M	0x1FFF_FFFF
axi_dma_2					
Data_S2MM (32 address bits : 4G)					
processing_system7_0	S_AXI_HP2	HP2_DDR_LOWOCM	0x0000_0000	512M	0x1FFF_FFFF
axi_dma_3					
Data_MM2S (32 address bits : 4G)					
processing_system7_0	S_AXI_ACP	ACP_DDR_LOWOCM	0x0000_0000	512M	0x1FFF_FFFF
processing_system7_0	S_AXI_ACP	ACP_QSPI_LINEAR	0xFC00_0000	16M	0xFCFF_FFFF
Excluded Address Segments (2)					
processing_system7_0	S_AXI_ACP	ACP_IOP	0xE000_0000	4M	0xE03F_FFFF
processing_system7_0	S_AXI_ACP	ACP_M_AXI_GP0	0x4000_0000	1G	0x7FFF_FFFF
Data_S2MM (32 address bits : 4G)					
processing_system7_0	S_AXI_ACP	ACP_DDR_LOWOCM	0x0000_0000	512M	0x1FFF_FFFF
processing_system7_0	S_AXI_ACP	ACP_QSPI_LINEAR	0xFC00_0000	16M	0xFCFF_FFFF
Excluded Address Segments (2)					
processing_system7_0	S_AXI_ACP	ACP_IOP	0xE000_0000	4M	0xE03F_FFFF
processing_system7_0	S_AXI_ACP	ACP_M_AXI_GP0	0x4000_0000	1G	0x7FFF_FFFF

Figura 5.29: Direcciones asignadas a los distintos elementos del diseño.

5.6.2. Floorplanning

En diseños preliminares se encontró que el conversor D/A no funcionaba correctamente cuando se lo incluye dentro del sistema. No obstante, sí lo hacía cuando se lo ubica de manera aislada. Más adelante, se halló que el problema se debía a que el sintetizador en Vivado ubicaba los elementos del DAC dispersos en toda la lógica programable. Esto provocaba que no se mantuviera la integridad de la señal, siendo necesaria para respetar el timing del diseño en VHDL. Por otro lado, el conversor depende de un filtrado pasabajos de pulsos a 1,5625MHz. Entonces, en la situación planteada anteriormente, se producía interferencia entre símbolos[6], provocando errores en la señal analógica a la salida del filtro.

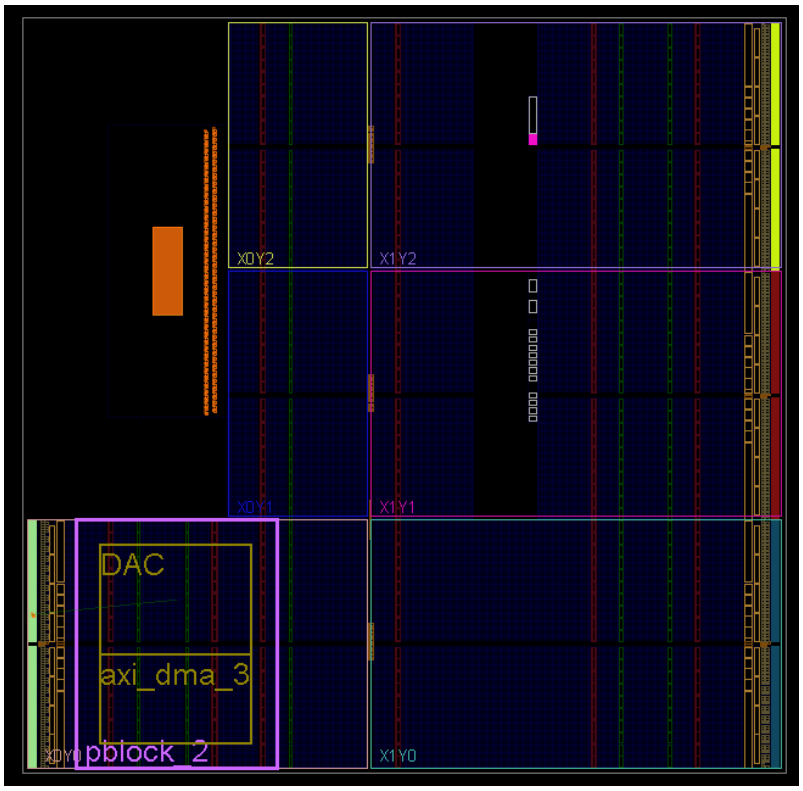


Figura 5.30: Floorplaning del Bloque DAC y el DMA 3 en el PL del SoC.

Debido a esto, se procedió a realizar *floorplanning* previa a la etapa del implementación; para indicarle al entorno de desarrollo que se desea ubicar el DAC en una región específica del PL. Con esto se evita que la señales internas del conversor recorran distancias largas relativa al tamaño del chip y por lo tanto se mantiene su integridad. Además, se indicó que la salida del bloque DAC se conecte al pin *Y11* mapeado a un conector PMOD externo. Se configuró la característica eléctrica para que sea *lvlcmos* de 3,3 volts. Nótese que el bloque esta adyacente al pin de salida, con el fin de minimizar la longitud de la conexión.

[6]Entiéndase como símbolo un pulso de la señal PDM que puede ser de valor lógico *alto* o *bajo*

5.6.3. Resultados de la Implementación

La visualización del dispositivo junto con la utilización de sus elementos se encuentra en la Figura 5.31. Principalmente, se muestra en distintos colores la ubicación del los *DMA* (verde), el *Multiplicador Matricial* (amarillo), la *Red de Ordenamiento* (rojo) y el *DAC* (rosa). Cada bloque con nombre *X0Y1, X0Y2, ..., X1Y0*, etc. delimita un dominio de reloj.

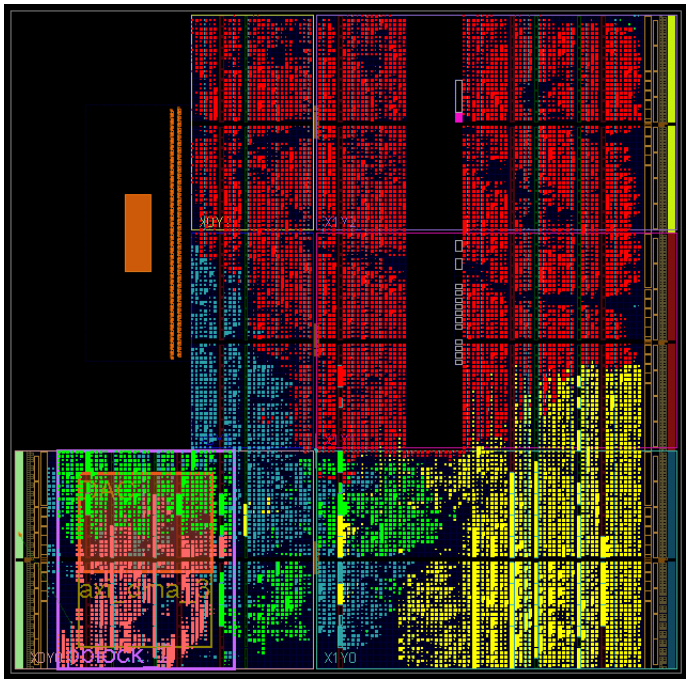


Figura 5.31: Diseño implementado en el SoC.

Se utilizó el 89 % de las LUTs, el 22 % de los bloques de memoria y el 30 % de los elementos DSP. De forma detalla, la Figura 5.32 informa para cada componente del sistema la utilización de recursos en el SoC.

Name	Slice LUTs (53200)	Slice Registers (106400)	F7 Muxes (26600)	F8 Muxes (13300)	Slice (13300)	LUT as Logic (53200)	LUT as Memory (17400)	LUT Flip Flop Pairs (53200)	Block RAM Tile (140)	DSPs (220)
design_1_wrapper	31307	36146	106	32	11810	29057	2250	15154	31	66
design_1_i (design_1)	31307	36146	106	32	11810	29057	2250	15154	31	66
A_GENERATOR_0 (desig...	7258	8874	0	0	2634	5512	1746	2927	16	40
axi_dma_0 (design_1_a...	526	741	0	0	220	491	35	322	2.5	0
axi_dma_1 (design_1_a...	536	777	0	0	242	501	35	302	2.5	0
axi_dma_2 (design_1_a...	851	1243	1	0	375	782	69	550	2.5	0
axi_dma_3 (design_1_a...	1456	1954	1	0	557	1342	114	846	5	0
axi_gpio_1 (design_1_a...	27	26	0	0	10	27	0	18	0	0
axi_gpio_3 (design_1_a...	27	26	0	0	11	27	0	17	0	0
axi_gpio_4 (design_1_a...	27	26	0	0	11	27	0	17	0	0
axi_mem_intercon (desi...	616	832	0	0	298	578	38	287	0	0
axi_mem_intercon_1 (de...	308	461	0	0	125	282	26	175	0	0
axi_mem_intercon_2 (de...	416	492	0	0	162	384	32	250	0	0
axi_mem_intercon_4 (de...	1105	1307	0	0	446	1036	69	568	0	0
axis_switch_0 (design_1...	128	343	0	0	103	128	0	78	0	0
DAC (DAC_imp_1YBUH12)	1738	1924	0	0	622	1714	24	370	1	26
HARD_THRESH_0 (desig...	15593	16391	104	32	6056	15593	0	8104	1.5	0
processing_system7_0 (...	0	0	0	0	0	0	0	0	0	0
ps7_0_axi_periph (desig...	679	700	0	0	301	618	61	288	0	0
rst_ps7_0_100M (design...	17	29	0	0	8	16	1	14	0	0

Figura 5.32: Recursos del chip asignados a cada elemento del sistema.

Para complementar, el entorno de desarrollo informa de manera estimada el perfil energético del reconstructor CS (Figura 5.33). Durante su funcionamiento, se espera que el silicio alcance una temperatura de 51,7°C (*Junction Temperature*) y consuma una potencia total de 2,318 Watts. La mayor parte se distribuye en potencia dinámica (2,141 Watts) y el resto en estática [80]. La primera es el consumo en funcionamiento mientras que la segunda es en reposo (depende del voltaje de alimentación, la temperatura, fabricación, etc.).

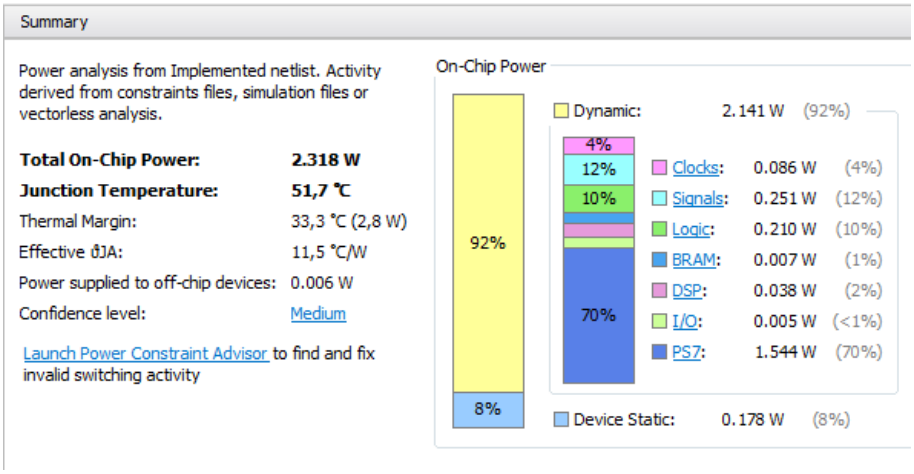


Figura 5.33: Temperatura y distribución de potencia estimada del diseño.

Capítulo 6

Implementación del Conversor Digital/Analógico

6.1. Requerimientos Generales

Si bien la placa de desarrollo no cuenta con un circuito integrado para realizar la conversión digital/analógica, en este capítulo se presenta cómo es posible fabricar un DAC utilizando la lógica programable del Zynq junto con circuitería electrónica externa. Básicamente, esta función es realizada por un sistema electrónico cuya entrada es un valor digital indicando una proporción entre la tensión de salida y un voltaje de referencia K. En principio, se estableció que el conversor D/A a diseñar deberá generar fielmente la señal reconstruida por el algoritmo de Sensado Compresivo. Por ello, se plantearon las siguientes especificaciones:

1. Relación señal a ruido de por lo menos 60 dB (1000 veces). Este parámetro deberá contemplar tanto el error de cuantización[1] como cualquier producto generado a causa de distorsión no lineal.
2. Ancho de banda comprendido en el rango completo del espectro audible humano. Aproximadamente hasta $15 \sim 20$ KHz.
3. Utilizar aritmética de punto fijo en la lógica programable.
4. No exceder los 5 Mhz de frecuencia de salida en los conectores PMOD, para prevenir la utilización de los puertos de alta velocidad.
5. Evitar aumentar la complejidad del circuito analógico externo al SoC siempre que sea posible.

[1] Error producido por aproximar muestras al valor más cercando dentro un conjunto predefinido. Es modelado como ruido agregado a la señal ideal.

Asumiendo que la señal y el ruido de cuantización no están correlacionados, la relación *señal a ruido de cuantización* SQNR se puede estimar como [81]:

$$SQNR = 6{,}02N + 1{,}76\ [dB] \tag{6.1}$$

Esta premisa no se cumple cuando una señal y su frecuencia de muestreo están relacionadas armónicamente. La ecuación (6.1) nos permite encontrar la cantidad de niveles de voltaje discernibles como 2^N, con N la longitud de las muestras digitales en bits (se asume enteros sin signo). La resolución está determinada por $\frac{K}{2^{-N}}$, donde el DAC entrega valores de tensión entre 0 y K volts. Para cumplir la primera especificación, utilizamos la ecuación (6.1) y se obtiene la *cantidad de bits efectivos* o ENOB:

$$ENOB = \left\lceil \frac{60\ [\text{dB}] - 1{,}76\ [\text{dB}]}{6{,}02\ [\text{dB}]} \right\rceil = 10\ \text{bits} \tag{6.2}$$

Una de las primeras opciones consideradas para la generación de la señal de voltaje fue la *Modulación por Ancho de Pulso* (PWM). En esta técnica de conversión, los valores discretos de amplitud de la señal digital $x[n]$ modulan al ciclo de trabajo $\delta = \frac{K\tau}{T}$ a una tasa igual a la frecuencia de muestreo $f_s = \frac{1}{T}$ (ver Figura 6.1). Si $x[n]$ presenta su amplitud A normalizada entre $[-1, 1]$, entonces cada T segundos el tiempo de encendido τ se actualiza como $\tau = T(\frac{A+1}{2})$. Además, realizando la expansión en series de Fourier para la forma de onda en este caso y considerando simetría par, se obtiene [82]:

$$f(t) = A_0 + \sum_{n-1}^{\infty}[A_n cos(\frac{2n\pi t}{T}) + B_n sin(\frac{2n\pi t}{T})] \tag{6.3}$$

con:

$$A_0 = K\delta,\ A_n = K\frac{1}{n\pi}[sin(n\pi\delta) - sin(2n\pi(1 - \frac{\delta}{2}))],\ B_n = 0 \tag{6.4}$$

Por lo tanto, de las ecuaciones (6.3) y (6.4) se infiere que realizando un filtrado pasabajos, es posible conservar la componente de continua A_0 y obtener la versión analógica de la señal $x(t)$; ya que esta contiene la información de x[n] contenida en el valor medio instantáneo. Los términos A_n deberán ser eliminados ya que su presencia degradan la SNR. Una forma de realizar este sistema se muestra en la Figura 6.2, en la cual el modulador PWM se implementa en lógica programable y mediante la utilización de un pin PMOD se obtiene la señal de tensión luego de un filtro analógico. En este caso, la resolución N en bits del DAC esta determinado por la frecuencia de reloj f_{clk} de la siguiente manera:

$$N = log_2\Big(\frac{f_{clk}}{f_s}\Big),\ N \in \mathbb{N} \tag{6.5}$$

Sin embargo, realizar este método presenta una serie de desventajas. En primer lugar, existe una relación de compromiso entre f_{clk} y el numero de bits. Si consideramos que la frecuencia de muestreo f_s es igual a dos veces el ancho de banda AB

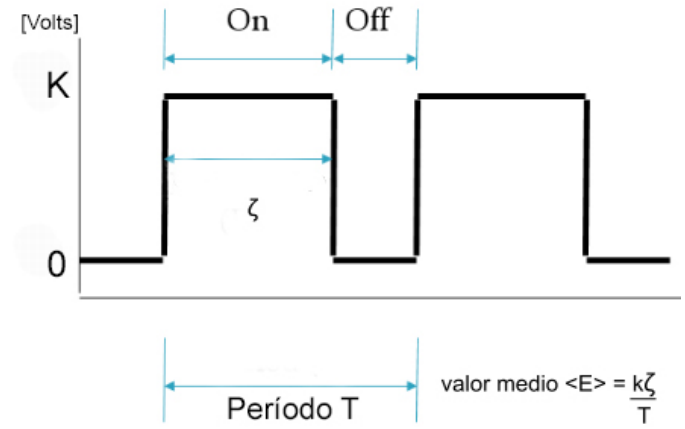


Figura 6.1: Señal PWM

deseado (a la salida del DAC), se cumple que la frecuencia de clock f_{clk} es:

$$f_{clk} = 2^{(N+1)}AB \tag{6.6}$$

Para $N = 10$ y $AB = 20$ KHz se necesita una frecuencia de reloj $f_{clk} = 40{,}96MHz$, por lo que no es posible cumplir con el quinto item de las especificaciones. Si relajamos la condición del ancho de banda a 5 KHz y utilizando 9 bits para la representación, entonces $f_{clk} = 5{,}12MHz$. En segundo lugar, la componente fundamental de la señal PWM estará a una octava de la frecuencia de corte del filtro pasabajos. En ese caso, si se desea por lo menos una atenuación de 40 dB a f_s se necesitará un filtro de séptimo orden (140 dB por década de atenuación) por lo que su diseño y fabricación se vuelve complejo. Una solución a este inconveniente sería disminuir la relación $\frac{AB}{f_s}$ para aliviar los requerimientos del filtro, aunque en ese caso el ancho de banda queda bastante comprometido. A su vez, según se especificó en el Capítulo 4, debido al diodo de protección y resistencia serie de 200 Ω los conectores PMOD presentan una frecuencia máxima de salida igual a 24 MHz. Entonces, no es posible aumentar f_{clk} por encima de este límite. Volviendo al caso de $AB = 5Khz$, el filtro debería tener una pendiente de por lo menos 60dB por octava. Queda entonces fundamentado que las especificaciones establecidas al comienzo de este capítulo no pueden ser cumplidas con la técnica de PWM. Es por ello que se investigaron otras alternativas.

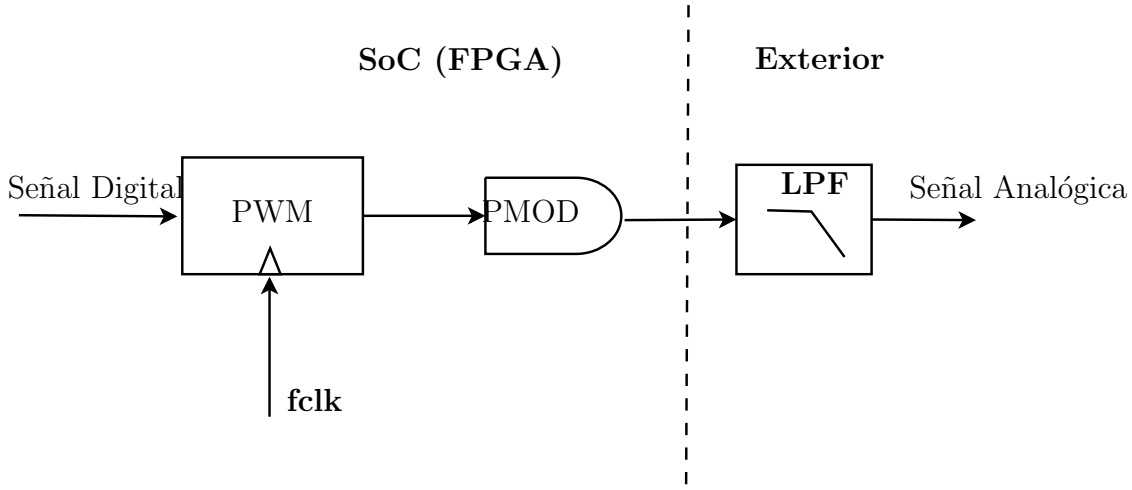


Figura 6.2: Primera propuesta de DAC con PWM.

6.2. Modulador Delta - Sigma

6.2.1. Delta Sigma de Primer Orden

Considerando al DAC como una planta a controlar, en la Figura 6.3 se muestra la esencia del modulador $\Delta\Sigma$. La señal digital $R(z)$ que se desea convertir al dominio analógico es comparada con la salida del conversor D/A generando la señal de Error $E_r(z)$, ingresando al controlador $Gc(z)$. Luego, la señal de control $U(z)$ ingresa a la planta generando la señal en tiempo continuo $Y(s)$. Esta configuración resulta conveniente, ya que la mayor parte del sistema es digital, lo que propicia alta integración en el SoC, no presenta derivas térmicas y es de bajo costo. Además, como se verá en breve, permite obtener excelentes SNR gracias al *Modelado de Ruido*. Inherentemente, los DAC de N bits pueden presentan transferencias alineales como quantización no lineal si no son calibrados [83]. Sin embargo, utilizando sólo un bit en la conversión estos problemas son evitados. Por eso, el diagrama en bloques mostrado en la Figura 6.4 es una realización práctica en donde la realimentación es en el dominio discreto y la salida presenta sólo dos niveles posibles. Linealizando el modelo de control y considerando al error de cuantización como una perturbación que se suma a la señal analógica, el sistema resultante es el de la Figura 6.5. La transferencia respecto a la señal a convertir es:

$$T_R(z) = \frac{G_c(z)}{1 + G_c(z)} \tag{6.7}$$

Mientras que respecto al error $E(z)$ resulta:

$$T_E(z) = \frac{1}{1 + G_c(z)} \tag{6.8}$$

Usualmente, se utiliza como controlador un bloque integrador[2]. Esto permite gene-

[2]De aquí proviene el nombre *sigma*, haciendo referencia al proceso de acumulación de la señal de error. La denominación *delta* surge de la diferencia entre la entrada y la salida.

rar la señal $U(z)$ en función de los valores previos y actuales de $E_r(z)$. La integración discreta se implementa como:

$$G_c(z) = \frac{1}{1 - z^{-1}} = \frac{z}{z - 1} \tag{6.9}$$

Por lo que, reemplazando (6.9) en (6.7) y (6.8) se obtiene:

$$T_R(z) = \frac{Y(z)}{R(z)} = \frac{z^2 - z}{2z^2 - 3z + 1} = \frac{0{,}5z}{z - 0{,}5} \tag{6.10}$$

$$T_E(z) = \frac{Y(z)}{E_r(z)} = \frac{z - 1}{2z - 1} = \frac{0{,}5(z - 1)}{z - 0{,}5} \tag{6.11}$$

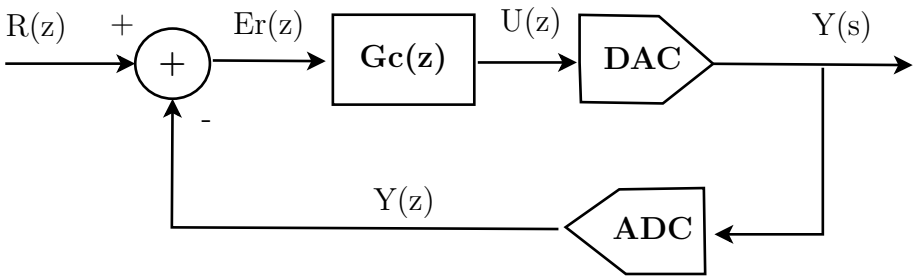


Figura 6.3: Lazo de control para la conversión D/A

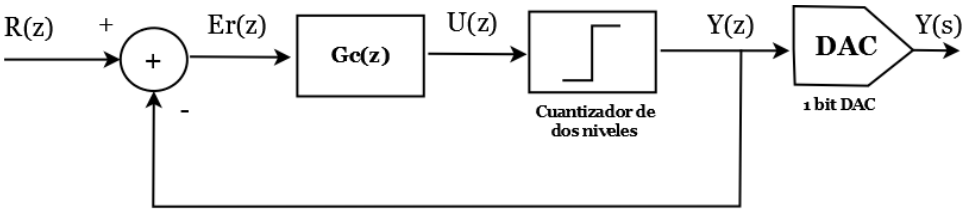


Figura 6.4: Modelo real del modulador Delta Sigma.

Graficando las transferencias (6.10) y (6.11) (Figura 6.6) se puede apreciar que en las bajas frecuencias el lazo no atenúa el contenido espectral de la señal de entrada pero sí rechaza al ruido de cuantización. Esto último es de gran importancia, ya que a pesar que el DAC de un bit presenta demasiado error de conversión (sólo puede generar dos valores discretos) gracias al modulador $\Delta\Sigma$ es posible obtener SRN comparables a conversores multibits. La técnica de aplicar el filtrado pasa-altos de la Figura 6.6b a la cuantización se denomina *Modelado de Ruido*. Aunque, para

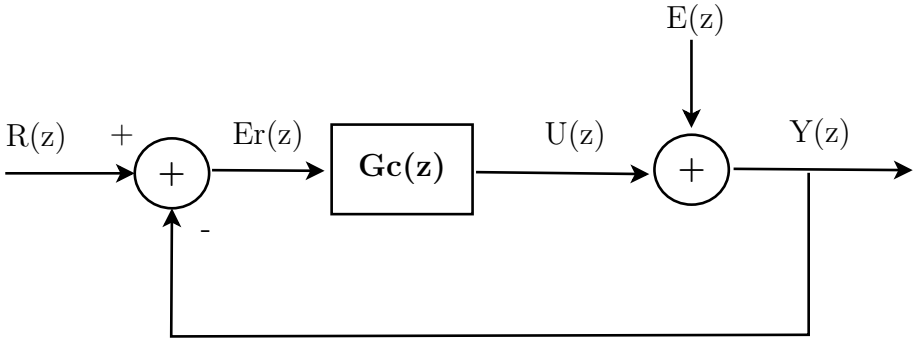


Figura 6.5: Modelo lineal del modulador Delta Sigma.

aprovechar este beneficio, el ancho de banda útil de la señal $R(z)$ debe ser mucho menor que $\frac{F_s}{2}$. Es por ello que se requiere utilizar una etapa de sobremuestreo previo a ingresar al modulador. Se define el factor de sobremuestreo $M \in \mathbb{N}$ de la siguiente manera:

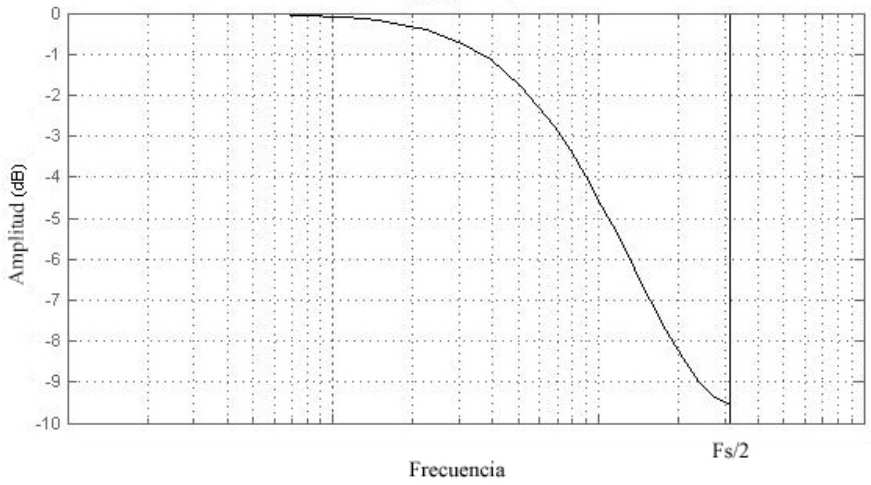
$$M = \frac{f_s}{2AB} \tag{6.12}$$

Entonces, volviendo a las especificaciones requeridas para el DAC, la gráfica de la Figura 6.7 establece que para $M = 32$, la relación SNR es casi 60 dB en un modulador de segundo orden[3]. Implementando al conversor $\Delta\Sigma$ en lógica programable, la frecuencia de reloj requerida sera igual a la frecuencia de muestreo. Es decir, se deberá generar una muestra de salida por cada ciclo de reloj. La siguiente ecuación de diseño resulta entonces:

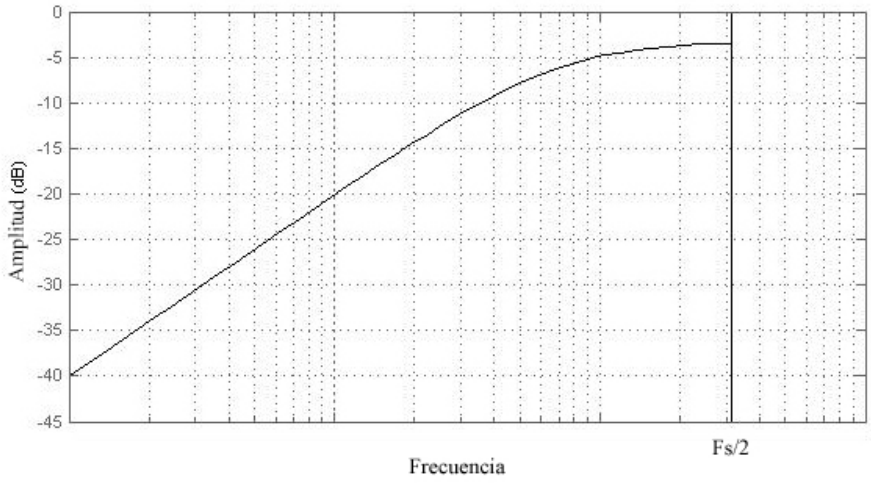
$$f_{clk} = f_s = 2AB.M \tag{6.13}$$

Siendo así, se opta por realizar un conversor de orden dos ya que se requiere un modulador de orden superior para alcanzar mejores SNR sin aumentar demasiado el factor M. A su vez, no se recomienda utilizar un único lazo ya que esto genera oscilaciones de baja frecuencia para entradas DC ([83, 84]).

[3]El orden denota la cantidad de lazos de realimentación.



(a) Transferencia de lazo respecto a la señal de entrada



(b) Transferencia de lazo respecto al error de cuantización

Figura 6.6: Transferencia de lazos a frecuencia normalizada Fs/2 = 1.

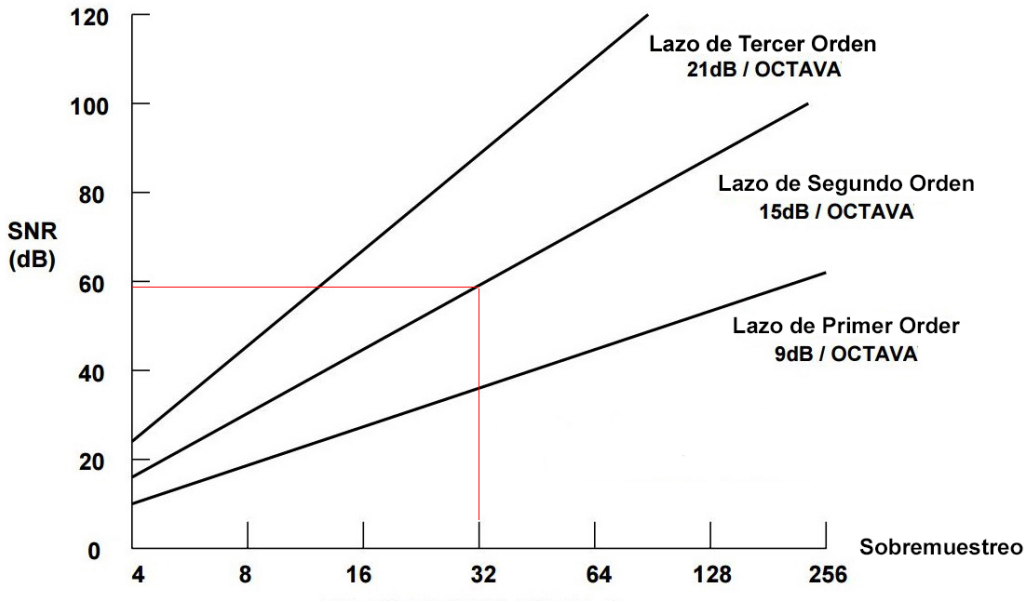


Figura 6.7: Relación entre el orden de un modulador $\Delta\Sigma$ y la cantidad de sobremuestreo necesario para conseguir un SNR determinado [85]. Se muestra también la atenuación del ruido de cuantización en dB por octava.

6.2.2. Delta Sigma de Segundo Orden

La topología del sistema de segundo orden junto con sus filtros de lazo fueron obtenidos de [86], en donde se realiza su diseño para receptores de banda ultra ancha. Se requiere que la amplitud de la señal digital a la entrada del modulador esté acotada en el intervalo [-0.9, 0.9] para evitar problemas de inestabilidad ya que se trata de un sistema no lineal. A la salida, la señal $x_p[m]$ adopta el valor -1 si la entrada al bloque *sgn()* es negativa, en caso contrario vale 1. El diagrama en bloques se presenta en la Figura 6.9. Linealizando como en el caso anterior se obtiene la transferencia total:

$$TLC(z) = \frac{X_p(z)}{X(z)} = \frac{1/4}{(z-p_1)(z-p_2)} \tag{6.14}$$

Con $p_1 = 0{,}75{+}0{,}433j$ y $p_2 = 0{,}75{-}0{,}433j$. El sistema es estable ya que $|p_1| = |p_2| \leq 1$; los polos se encuentran dentro del círculo unitario. Para analizar qué sucede con el margen de fase M_Φ y ganancia M_g primero se debe plantear la transferencia de lazo abierto:

$$GH(z) = \frac{1/4}{(z-1)(z-0{,}5)} \tag{6.15}$$

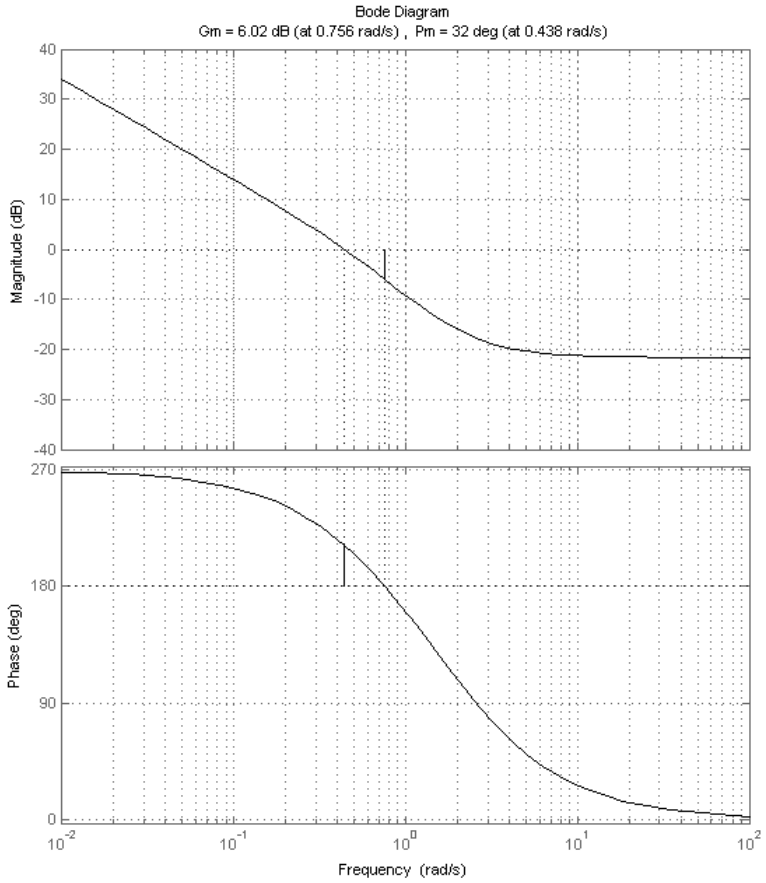


Figura 6.8: Diagrama de Bode para la obtención del margen de fase y ganancia en el modulador de orden dos.

Luego, se aplica la *transformación bilineal* con $f_s = 1$ a la expresión (6.16) a fin de realizar el análisis de estabilidad en el dominio continuo:

$$GH(s) = 0{,}0833\ \frac{(s-2)^2}{s(s+0{,}67)} \tag{6.16}$$

Armando el diagrama de Bode de la Figura 6.8 se halla que $M_\Phi = 32^o$ y $M_g = 6$ dB. Ambos márgenes son positivos, por lo tanto el sistema es estable. En el caso discreto no hay riesgo que se produzca una situación de inestabilidad debido a variaciones en estos valores ya que no existen derivas en los parámetros del sistema.

Re-expresando al modulador en función de bloques integradores y retardos unitarios, con el fin de efectuar el diseño en lógica programable, se obtiene la estructura de la Figura 6.10. Se decidió utilizar representación en punto fijo de 24 bits complemento a dos mediante el formato $Q_{7,17}$, donde 17 bits son fraccionarios, 6 bits representan la

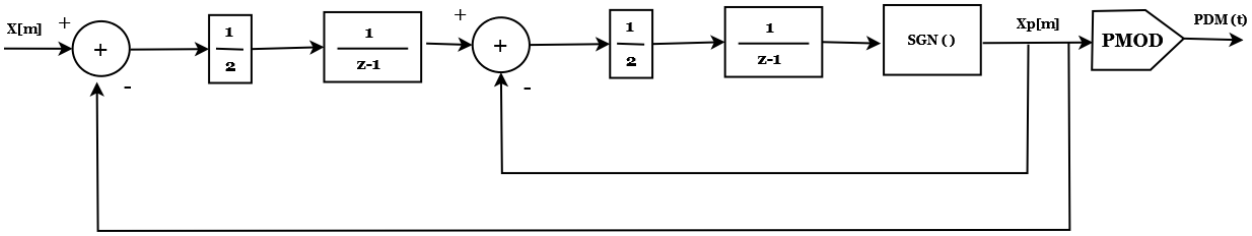


Figura 6.9: Modulador $\Delta\Sigma$ de segundo orden.

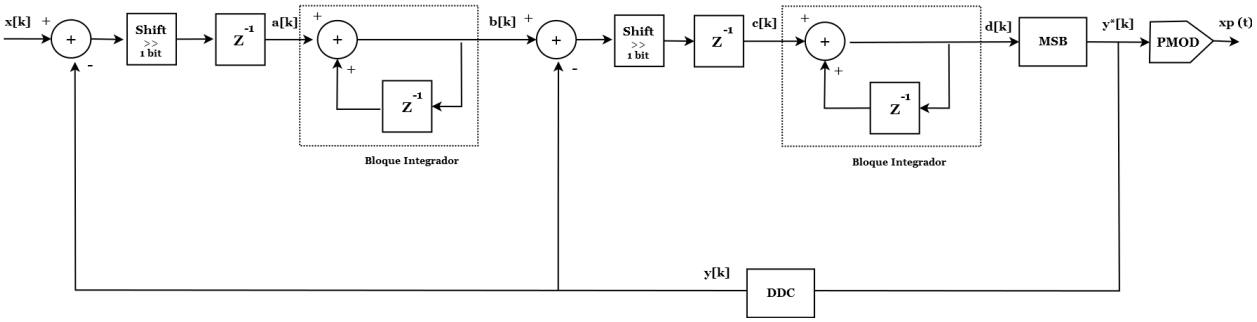


Figura 6.10: Diagrama detallado del modulador realizado en FPGA. Se muestra también la ubicación de las variables de estado a[k], b[k], c[k] y d[k].

parte entera y un bit es utilizado para el signo. La resolución es $2^{-17} = 7{,}6294\ 10^{-6}$ y el rango de valores posibles está comprendido en el intervalo $[-64; 63{,}99999]$. La razón por la que se escogió usar 6 bits de parte entera es para evitar el desborde aritmético en los nodos integrales del modulador. La multiplicación por $\frac{1}{2}$ se realizó mediante un registro desplazamiento con el fin de utilizar LUTs en vez de bloques DSP. La función signo es producida por el bloque MSB, cuya salida es el bit de signo del dato de entrada. Luego, el convertidor DDC genera como respuesta el valor 131071 (0.9999 en formato $Q_{7,17}$) cuando el bit de entrada vale 1 y -131072 (-1 en formato $Q_{7,17}$) si es 0. El siguiente paso fue plantear cuatro variables de estado y la expresión de la salida:

$$\begin{aligned} a[k] &= \frac{1}{2}x[k-1] - \frac{1}{2}y[k-1] \\ b[k] &= a[k] + b[k-1] \\ c[k] &= \frac{1}{2}b[k-1] - \frac{1}{2}y[k-1] \\ d[k] &= c[k] + d[k-1] \\ y[k] &= sgn(d[k]) \end{aligned} \tag{6.17}$$

Por lo tanto, se necesitaron 10 registros para almacenar datos: ocho para las variables de estado (valor actual y anterior), dos para recordar la entrada y salida del instante anterior. En cada flanco positivo de la señal de reloj se genera un bit de salida y se actualizan los estados. El código VHDL del modulador se encuentra en la Sección A.1.3 del Apéndice.

6.3. Diseño completo del DAC

El vector $x[n]$ generado en el sistema reconstructor de Sensado Compresivo es enviado a una etapa de interpolación con un factor $M = 32$. Después, se aplica saturación de manera preventiva para acotar la amplitud entre [-0.9, 0.9]; ya que el filtrado realizado en el sobremuestreo digital puede generar transitorios de amplitud mayores a ese rango. El modulador $\Delta\Sigma$ es el siguiente en la cadena de procesamiento, generando un stream de bits PDM. El pin del conector PMOD actúa como un DAC de 1 bit, generando una tensión de 3.3 Volts para un valor lógico en alto y 0 Volts en bajo. Finalmente, los pulsos de tensión ingresan a un filtro pasabajos Bessel de cuarto orden y frecuencia de corte igual a 20 KHz. Se usó este tipo de filtro ya que presenta respuesta de fase lineal dentro de la banda de paso. Con esto se elude la distorsión de fase lo cual provoca que las envolventes de las señales no conserven su forma original.

El reloj del PL en el Zynq 7000 funciona con una frecuencia global de 100 MHz. Por cuestiones de diseño del interpolador, se optó por dividir esta señal por un factor de 64. Esto genera un f_{clk} para el DAC de 1,56 MHz y utilizando la ecuación (6.13) el ancho de banda para la señal analógica $x(t)$ resulta 24,41 KHz.

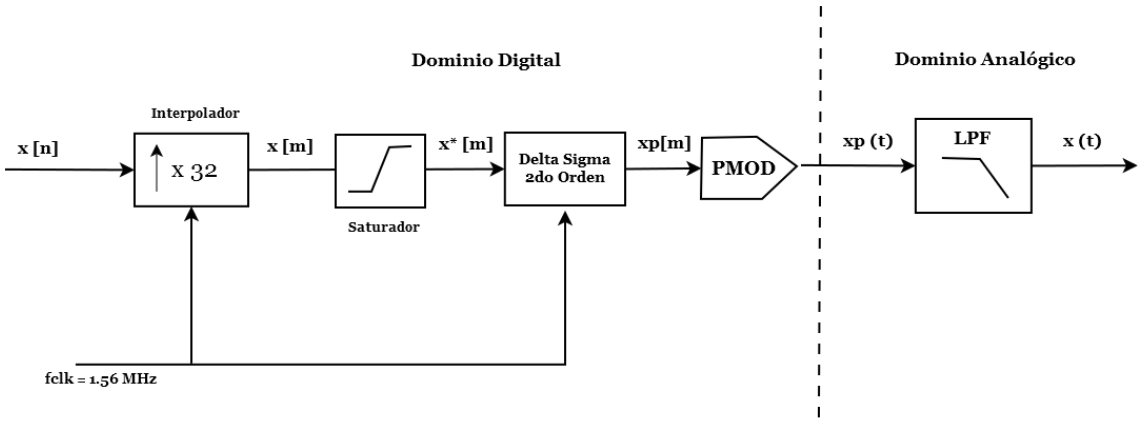


Figura 6.11: Diagrama en bloques del sistema de conversión D/A desarrollado.

En el Cuadro 6.1 se efectúa una comparación entre el DAC PWM propuesto al comienzo junto con el diseño $\Delta\Sigma$ realizado. Se puede apreciar que en el sistema de segundo orden se emplea una frecuencia de reloj tres veces menor, se logra cinco veces mayor ancho de banda y un aumento de la SNR de 20 dB. Sin embargo, la implementación en lógica programable es más compleja ya que necesita de interpolación.

	PWM	**ΔΣ**
f_{clk}	5,12 MHz	1,5625 MHz
AB	5 Khz	24,41 KHz
Filtro	Séptimo Orden	Cuarto Orden
SNR	$\approx 40dB$	$\approx 60dB$
ENOB	6 bits	10 bits
Interpolación	No	Si

Cuadro 6.1: Comparación entre DAC PWM y ΔΣ

6.3.1. Modelo de Simulink

Con el fin de verificar el correcto funcionamiento del modulador y sus especificaciones técnicas, se armó un modelo de simulación (Figura 6.12) utilizando la herramienta Simulink ® incluida en la versión R2013a de Matlab ®. Este paso fue hecho antes de la programación del FPGA en el SoC. Primero, se ingresa al sistema con una sinusoidal discreta (bloque *Señal Digital*) de amplitud 0,7 pico y frecuencia igual a 5 Khz. El período de muestreo empleado es $T_s = 640$ ns. Después, se ubicaron varios cuantizadores con intervalos $q = 2^{-17}$ para simular la precisión del formato $Q_{7,17}$. Los bloques *SGN* Y *DDC* realizan las funciones detalladas en el diseño del ΔΣ de segundo orden. Las señales medidas con el elemento *Scope* que se presentan en la Figura 6.13 fueron la entrada (6.13a), la señal PDM generada por el DAC de 1 bit (6.13b) y la versión analógica a la salida (6.13c). Además, la Figura 6.14 muestra las señales internas a los lazos del sistema. La primera de ellas (6.14a) es la diferencia entre la sinusoidal de entrada y la salida PDM. Luego, se produce la segunda señal de error (6.14b) y finalmente ingresa la forma de onda (6.14c) al DAC.

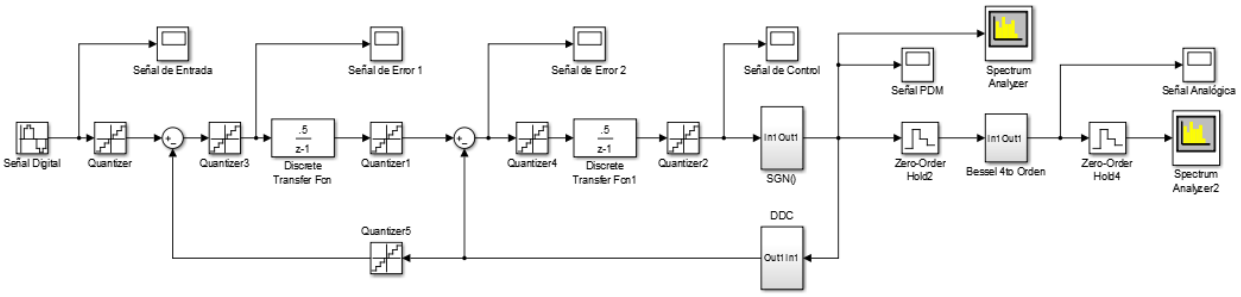
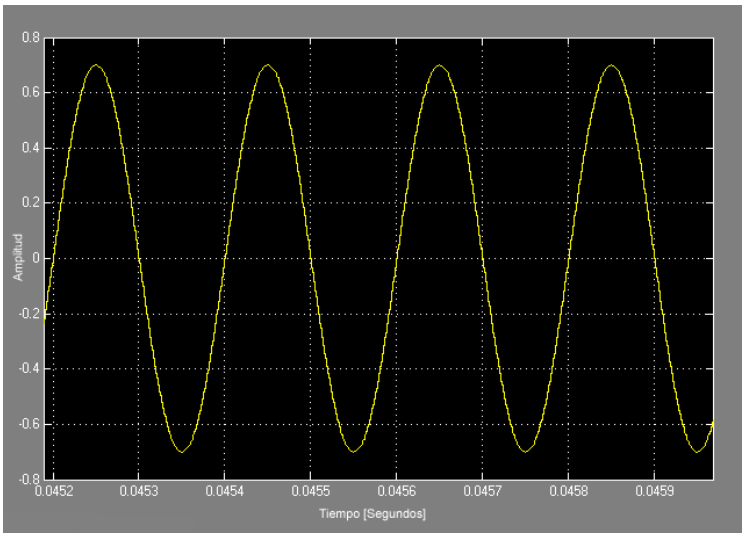
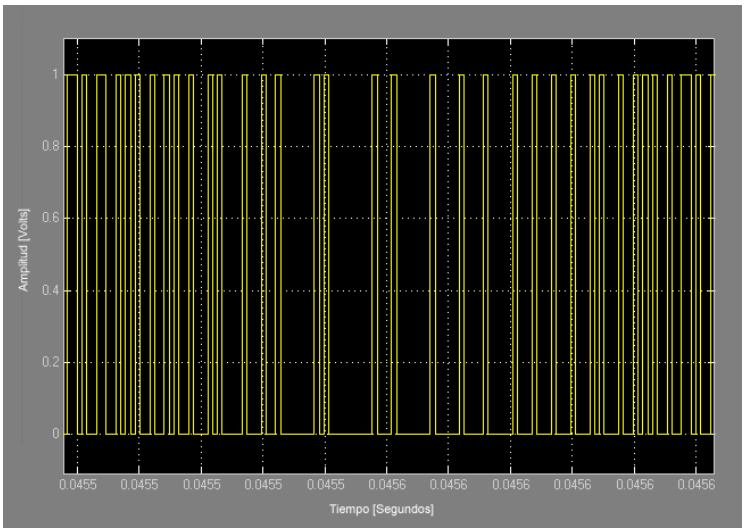


Figura 6.12: Modelo de simulación realizado en Simulink para el conversor Delta-Sigma.

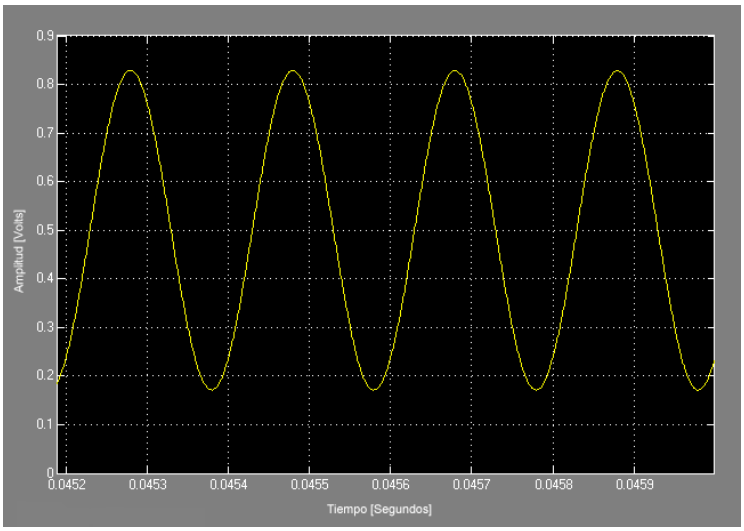
Efectivamente, en la Figura 6.15a se aprecia el efecto de modelado de ruido en el modulador. El ruido, que es tratado independientemente a la señal de entrada por el sistema, presenta la forma de transferencia pasa-altos. En la zona de bajas frecuencias, se encuentra el seno representado como un pico ubicado en 5 Khz extendiendose hasta 0 dBm (carga de referencia de 50Ω). Con el bloque *Bessel 4to Orden* se corroboró que el filtro pasabajos, con frecuencia de corte (3 dB) de 20KHz y atenuación igual a 80 dB por década, mitiga al ruido de cuantización en las altas frecuencias. La sinusoidal analógica finalmente resulta con un SNR ≈ 70 dB (Figura 6.15b).



(a) Señal digital de entrada.

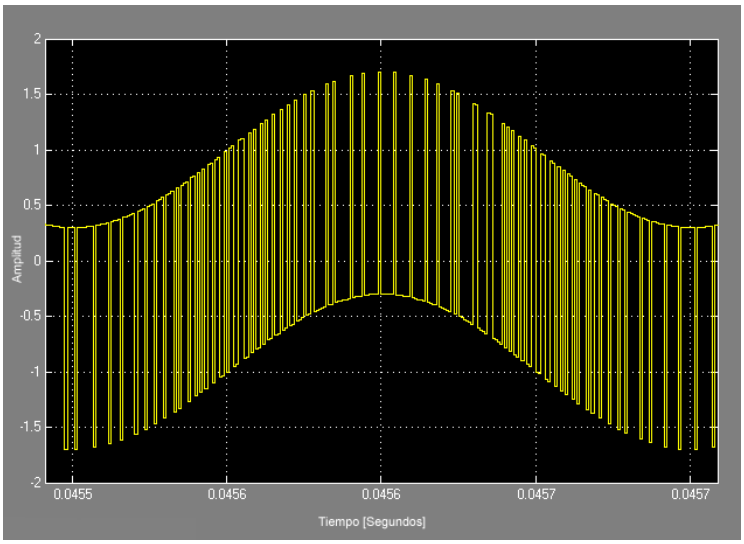


(b) Señal PDM de salida.

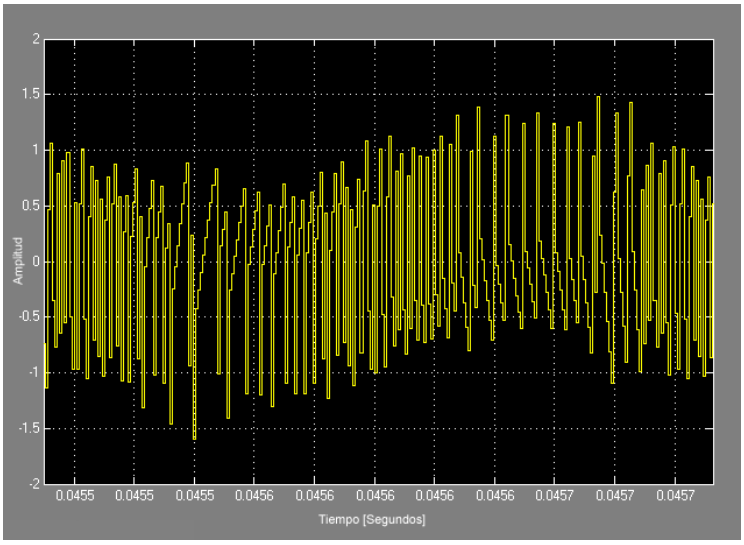


(c) Señal analógica a la salida del filtro.

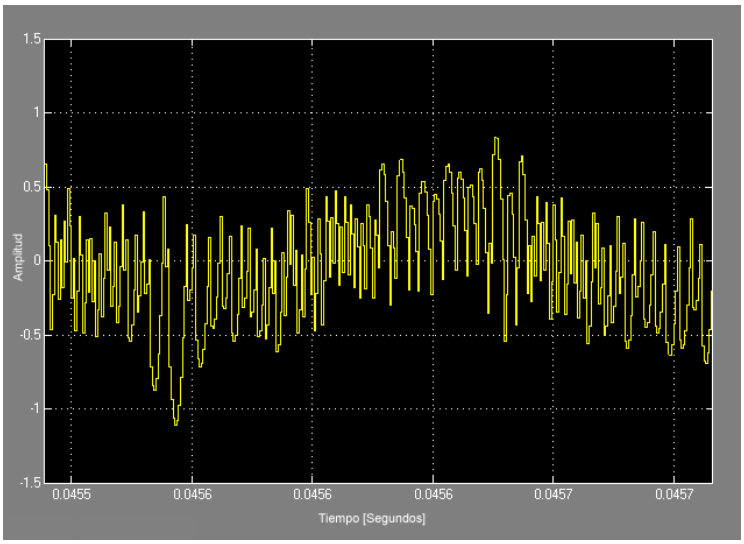
Figura 6.13: Proceso de transformación de la señal digital a analógica.



(a) Diferencia entre señal de entrada y salida PDM (señal de error 1).

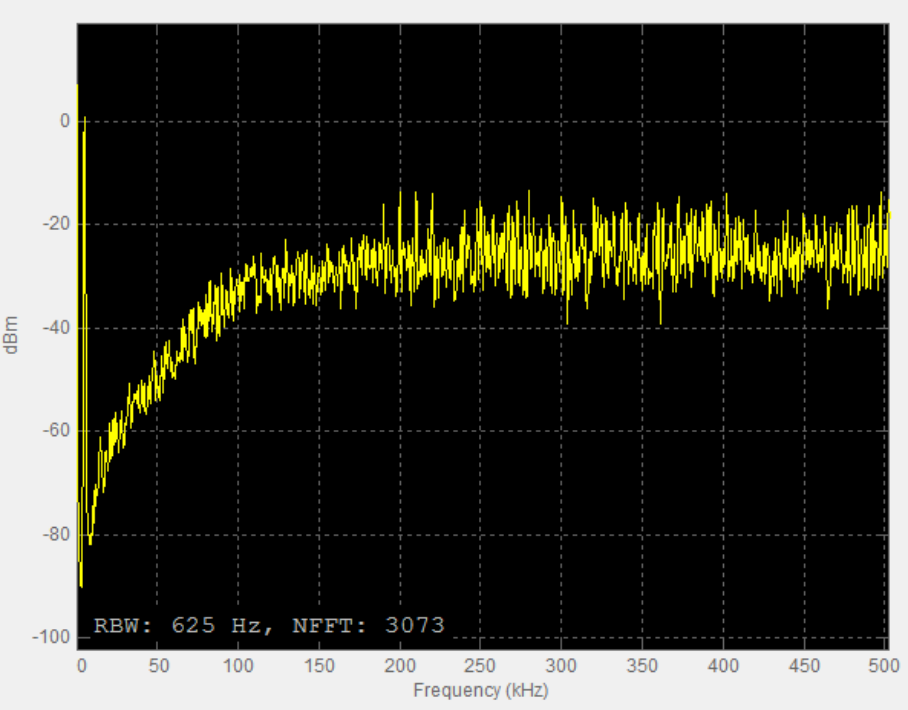


(b) Diferencia entre (señal de error 2).

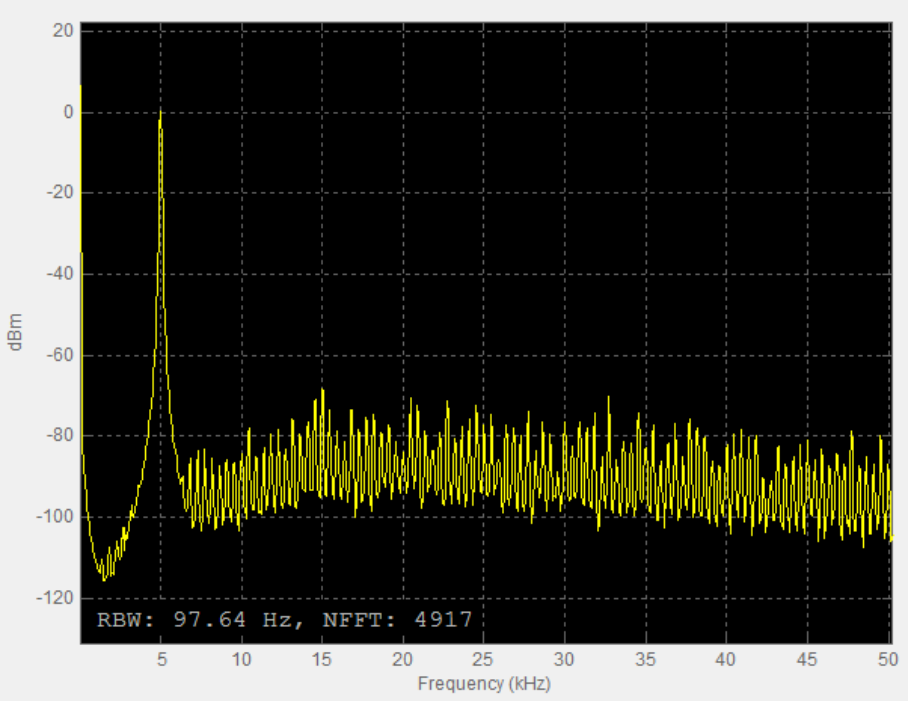


(c) Entrada al DAC de 1 bit cuando el estimulo es sinusoidal (señal de control).

Figura 6.14: Señales internas del modulador $\Delta\Sigma$



(a) Contenido espectral de la señal PDM (pre-filtrado)



(b) Contenido espectral de la señal PDM (post-filtrado)

Figura 6.15: Visualización con la herramienta *Spectrum Analizer* en el dominio frecuencial de las señales analógicas simuladas.

6.4. Filtro Analógico

6.4.1. Diseño y Características

La transferencia normalizada de un filtro pasabajos, con dos polos reales conjugados, es la siguiente:

$$H(s) = \frac{K}{\left(\frac{s}{\omega_o}\right)^2 + \frac{s}{w_o Q} + 1} \tag{6.18}$$

Siendo s la variable compleja en el dominio transformado de Laplace, ω_o la frecuencia de resonancia en radianes por segundos, Q el factor de calidad y K la ganancia arbitraria en veces. Los filtros Bessel están optimizados para retardo de grupo constante. Es decir, que el atraso de fase es lineal con la frecuencia [87]. Como se mencionó anteriormente, esta característica permite que las señales que atraviesan el filtro conserven el aspecto de sus envolventes. Por lo que, dentro del ancho de banda, se espera que el DAC preserve la forma de onda de la señal reconstruida con CS.

Utilizando la Cuadro que se encuentra en la referencia [87] de un polinomio de Bessel de segundo orden se obtienen los ceros: $z_1 = -1{,}103 + j0{,}6368$, $z_1^* = -1{,}103 - j0{,}6368$. Esto genera los coeficientes: $a_0 = 1{,}622$ y $a_1 = 2{,}206$. Entonces, la realización del filtro pasabajos Bessel es generada por un circuito cuya transferencia sea:

$$H_(\omega) = \frac{K}{-\left(\frac{\omega}{2\pi f_c}\right)^2 + 2{,}206\frac{j\omega}{2\pi f_c} + 1{,}622} \tag{6.19}$$

Con f_c la frecuencia de corte (3 dB de atenuación respecto a la ganancia DC) y j la unidad imaginaria. Sin embargo, es necesario normalizar (6.19) para que se corresponda con (6.18). Dividiendo por 1,622 se encuentra:

$$H_(\omega) = \frac{K}{-\left(\frac{\omega}{2\pi \times 1{,}274 f_c}\right)^2 + 1{,}360\frac{j\omega}{2\pi f_c} + 1} \tag{6.20}$$

Por lo que, para $f_c = 20$ KHz, $\omega_o = 2\pi \times 1{,}274 f_c = 160095{,}56$ [rad/seg] (25480 KHz) y $Q = 0{,}577$. El siguiente paso es adoptar una arquitectura circuital para implementar (6.20). La Figura 6.16 muestra la topología Sallen Key pasabajos con ganancia unitaria. Básicamente, consiste en un seguidor de tensión que sensa el voltaje de salida y con realimentación positiva mediante el condensador C_2 logra re-ubicar los polos de su transferencia. De esta manera, se puede conseguir prácticamente cualquier valor de Q [88]. A bajas frecuencias, donde C_1 y C_2 presentan altas impedancias, la señal de salida es simplemente igual a la de entrada. Si el amplificador operacional es ideal, a altas frecuencias los condensadores se comportan como corto circuitos. En este último caso, la entrada es enviada a masa.

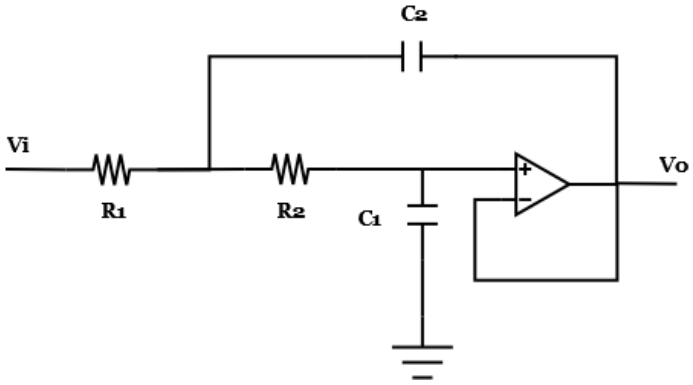


Figura 6.16: Arquitectura Sallen-Key

Desarrollando un modelo de esta arquitectura para encontrar su transferencias y realizar un análisis de su funcionamiento, en la Figura 6.17 se puede discernir los efectos de realimentación positiva y negativa de la salida; junto con el acoplamiento directo no deseado de la entrada. A partir de lo anterior, se desprende el diagrama en bloques de la Figura 6.18. Resolviendo el circuito, se hayan las siguientes transferencias:

$$G_1(s) = \frac{1}{sC_1R_1 + (sC_2R_1 + 1)(sC_1R_2 + 1)} \tag{6.21}$$

$$G_2(s) = \frac{sC_1R_2}{sC_1R_1 + (sC_2R_1 + 1)(sC_1R_2 + 1)} \tag{6.22}$$

$$G_3(s) = \frac{sC_1R_2R_1 + R_1}{z_o\big[sC_1(R_1 + R_2) + 1\big] + sC_1R_2R_1 + R_1} \tag{6.23}$$

$$G_4(s) = \frac{sC_2z_o(sC_1R_2 + 1)}{sC_2R_1(sC_1R_2 + 1) + (sC_2z_o + 1)} \tag{6.24}$$

En la simplificación del AO en la Figura 6.17, z_i y z_o representa su impedancia de entrada y salida respectivamente. Además, la transferencia del operacional es $A(s)$. Considerando que z_o es pequeña en comparación con las demás impedancias del circuito, entonces se pueden hacer las aproximaciones $G_3(s) \approx 1$ y $G_4(s) \approx 0$. Partiendo de esas premisas, se realiza el esquema de la Figura 6.19. Para la fabricación del filtro activo, se dispuso del integrado LM358 de la empresa *Texas Instruments*. Este contiene dos operacionales internamente compensados para realimentación unitaria, por lo que, el primer lazo de realimentación negativa es estable (Consultar hoja de datos [89] para más detalles). En esa situación, el ancho de banda con ganancia

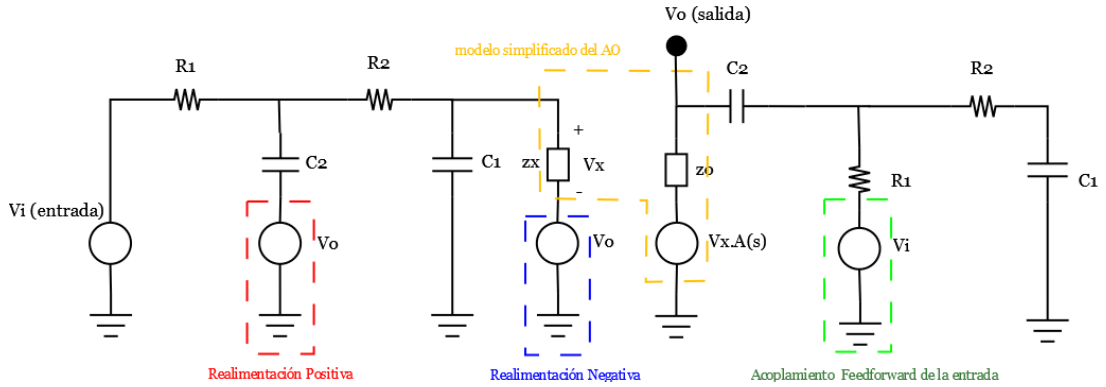


Figura 6.17: Desarrollo del circuito Sallen Key para identificar transferencias.

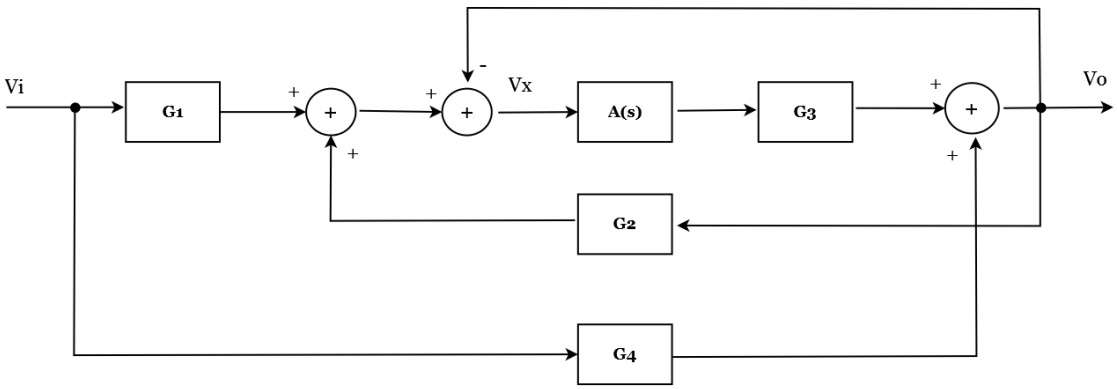


Figura 6.18: Diagrama en bloques de una etapa Sallen-Key

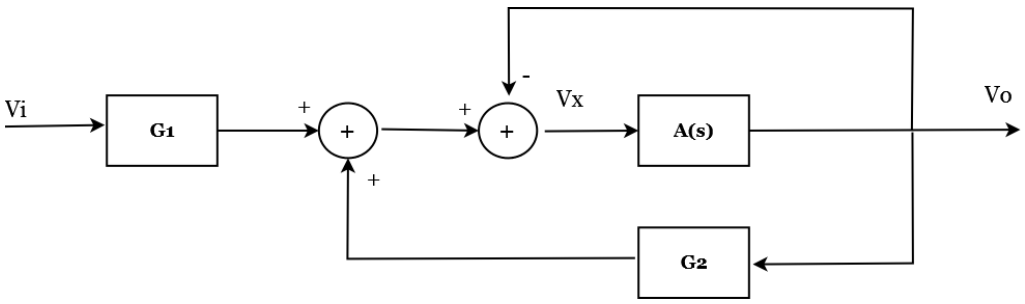


Figura 6.19: Diagrama en bloques simplificado de una etapa Sallen-Key

$0dB$ de los amplificadores se extiende hasta los 0,7 MHz. Es decir, dentro del rango de frecuencias que el operacional se comporta como un seguidor de tensión, la transferencia del filtro se encuentra como:

$$H_f(s) = \frac{G_1(s)}{1 - G_2(s)} = \frac{1}{s^2(C_1C_2R_1R_2) + sC_1(R_1 + R_2) + 1} \tag{6.25}$$

La estabilidad se analiza con la transferencia $G_2(s)$ teniendo en cuenta realimentación positiva. Generando la correspondencia entre (6.25) y (6.20) se encuentra las ecuaciones de diseño:

$$\begin{aligned} C_1C_2R_1R_2 &= \left(\frac{0{,}7849}{2\pi fc}\right)^2 \\ C_1(R_1 + R_2) &= \frac{1{,}360}{2\pi fc} \end{aligned} \tag{6.26}$$

Se debe tener en cuenta que f_c es la frecuencia de corte para una etapa de segundo orden (40 dB/década). Como se necesita que el filtro sea de cuarto orden (80 dB/década), hay que utilizar dos etapas Sallen-Key en cascada. En este caso, la atenuación total a 20 KHz tiene que ser de 3 dB. Para ello se utilizó una herramienta disponible online [90] que obtiene los valores de los componentes[4] ingresando el tipo (Bessel), la frecuencia de corte y el orden del filtro:

Componente	**Etapa 1**	**Etapa 2**
$\boldsymbol{R_1}$	2k5 Ω	3k7 Ω
$\boldsymbol{R_2}$	8k1 Ω	9k4 Ω
$\boldsymbol{C_1}$	1 nf	470 pf
$\boldsymbol{C_2}$	1.5 nf	1 nf

Cuadro 6.2: Componentes necesarios para armar el Filtro Bessel (Figura 6.20) con $f_c = 20$ KHz y 80 dB/dec de atenuación.

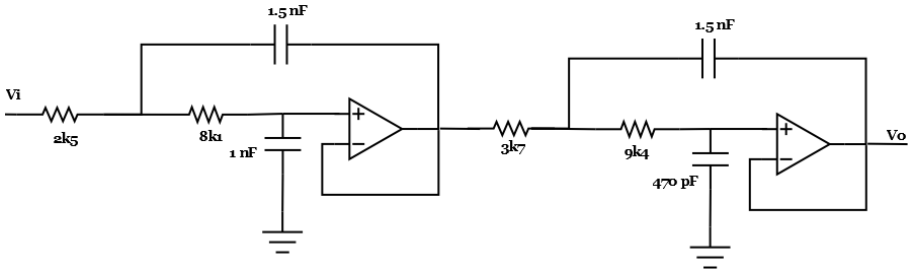


Figura 6.20: Filtro activo con dos etapas Sallen Key en cascada. [90].

[4] Se calcularon las resistencias a partir de valores comerciales de capacitores.

6.4.2. Efectos del Amplificador Operacional no Ideal

Analizando ahora el efecto de la impedancia de entrada y transferencia del AO, se encuentra que a altas frecuencias el filtro en vez de atenuar a 40 dB/década comienza a tener un comportamiento pasa-altos. A frecuencias mucho mayores que la de corte $G_1(s)$ y $G_2(s) \to 0$; pero G_4, que inicialmente era despreciable, comienza a tener preponderancia. Estas características se resumen en el diagrama en bloques de la Figura 6.21. La transferencia de acoplamiento de la entrada resulta:

$$\left.\frac{V_o}{V_1}\right|_{forward} \approx \frac{G_4(s)}{1 + A(s)G_3(s)} \approx \frac{z_o}{R_1(1 + A(s))} \tag{6.27}$$

Donde, a partir de las especificaciones provistas por la hoja de datos [89], se asume la respuesta $A(s)$ del operacional:

$$A(s) = \frac{100000}{(\frac{s}{2\pi 20} + 1)} \tag{6.28}$$

Combinando (6.27) con (6.25) se encuentra la transferencia real de una etapa de segundo orden (Figura 6.22). La atenuación máxima de cada etapa depende de z_o y esta determinada por (6.27) cuando $S \to \infty$. Es decir, aproximadamente es igual a z_o/R_1. Asumiendo $z_o = 25\Omega$, utilizar sólo un bloque Sallen-Key no es suficiente para el filtrado de la señal PDM ya que la atenuación máxima en altas frecuencias es de 40 dB (se requiere por lo menos 60 dB). Por lo tanto, agregando otra instancia del filtro en cascada no solo se consigue mayor pendiente de atenuación o *Roll-Off*, sino que ahora la atenuación máxima es de 80 dB. Como aclaración, a partir de 100 MHz la inductancia parásita comienza a jugar un papel en la respuesta en frecuencia. También, la capacitancia entre pistas aportará a la atenuación [88]. Ambos efectos no son contemplados en el análisis.

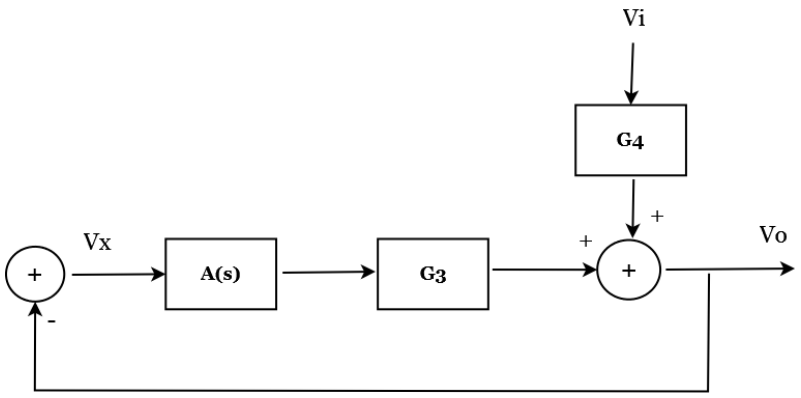


Figura 6.21: Diagrama en bloques para altas frecuencias de la etapa Sallen-Key

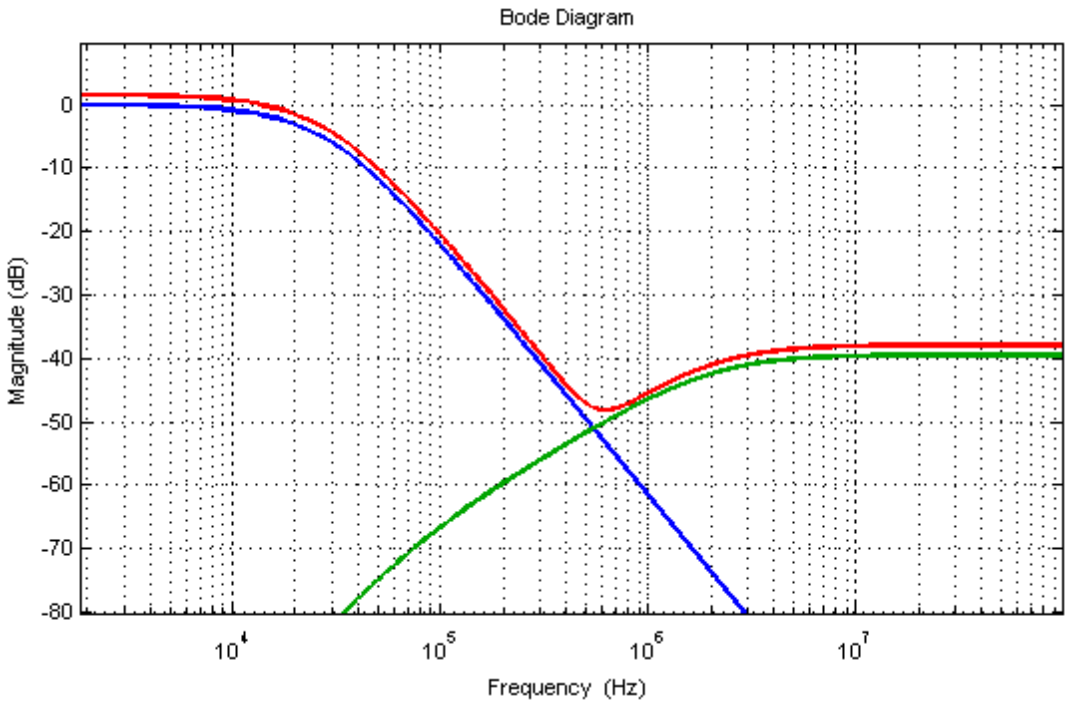
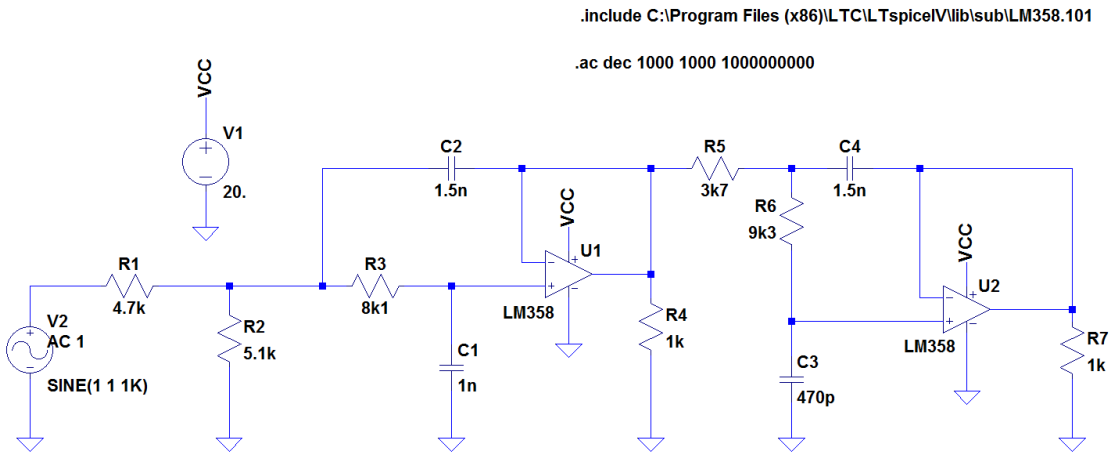


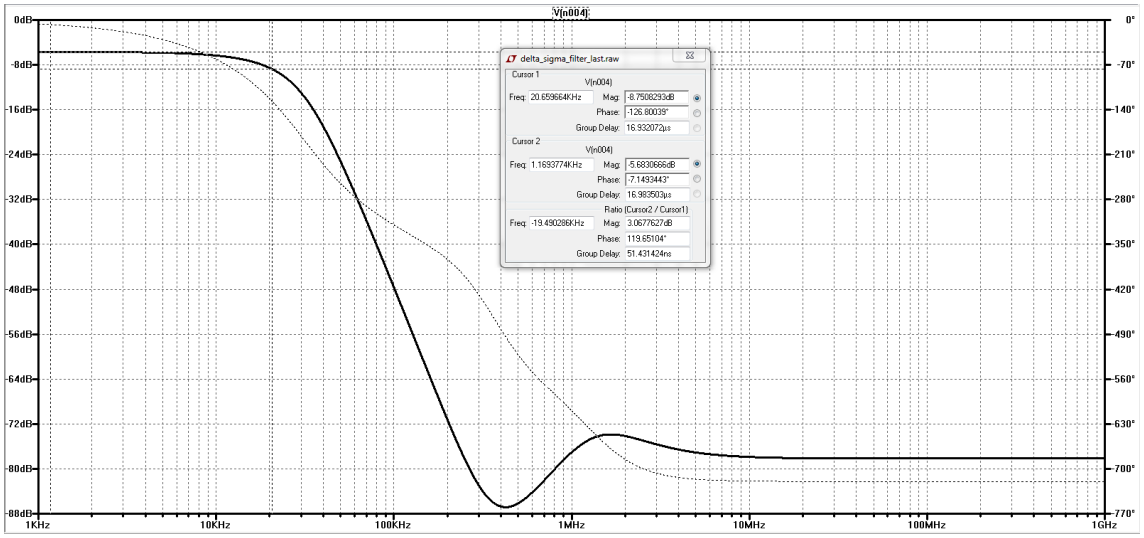
Figura 6.22: Transferencia Real del Filtro Sallen Key considerando z_o y $A(s)$. En azul se aprecia la respuesta en frecuencia ideal, en verde el efecto feed-forward no deseado y en rojo la transferencia total.

6.4.3. Simulación en LTspice

Previo a la construcción del filtro activo, se realizó una simulación circuital en el programa LTspice (Ver 4.23k). El esquemático utilizado es el de la Figura 6.23a, donde el voltage de alimentación (VCC) para los AO es 3,3 Volts con fuente simple. No obstante, se pueden reconocer algunas diferencias con la Figura 6.20. En primera instancia, se reemplazó la resistencia de 2k5 a la entrada por dos resistencias de 4k7 y 5k1. Esto se debe a que el PMOD de la placa ZedBoard sólo puede entregar tensiones de 3,3 o 0 Volts, por lo que se necesita un divisor resistivo a la entrada para evitar saturación. Aplicando el teorema de Thévenin a la tensión de entrada junto con R_1 y R_2 (Figura 6.23a) se obtiene la topología original con la salvedad que V_i se atenúa en 0,52 veces ó 5,7 dB. Por lo tanto, la amplitud máxima de la señal de entrada equivalente es de 1,71 Volts y se evita la saturación del operacional. En la configuración empleada, el AO puede excursionar entre 0 a 2,3 Volts [89].



(a) Esquemático final del filtro activo.



(b) Diagrama de Bode del filtro generado con simulación.

Figura 6.23: Simulación en el programa LTSpice (Versión 4.23k)

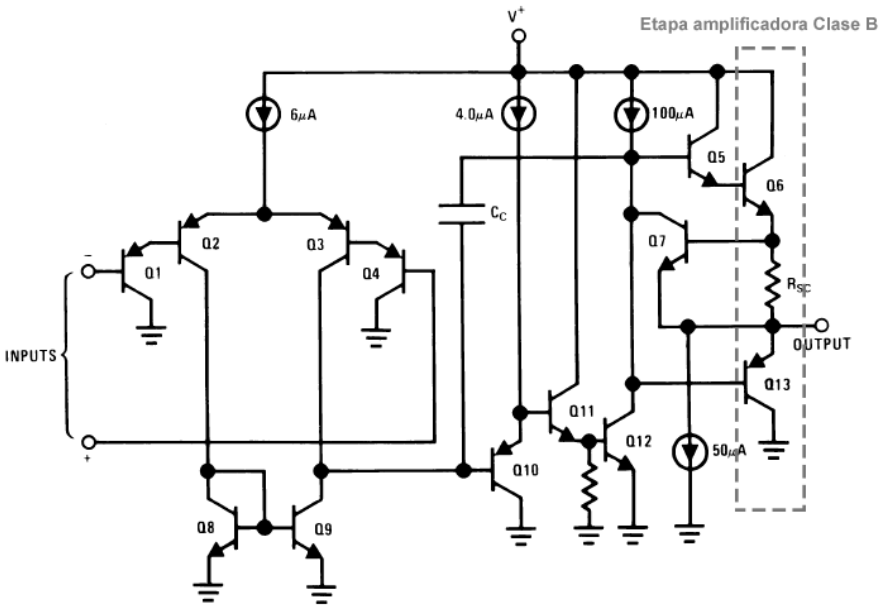


Figura 6.24: Esquemático simplificado de un AO del integrado LM358 [89].

Otra modificación al circuito original es el agregado de resistores a masa de $1k\Omega$ a la salida de cada amplificador. Esto es necesario para generar una corriente de polarización en la etapa final de transistores que funcionan en clase B (Figura 6.24) y prevenir distorsión por cruce. De esta manera, los únicos transistores que conducen en la última etapa son Q_5 y Q_6. Si no se resolvía este inconveniente, la SNR hubiese empeorado bastante debido a la generación de picos de tensión en la conmutación de alta velocidad del PDM.

Para la simulación del diagrama de Bode, se usó el modelo SPICE del LM358 descargado de la página web de Texas Instruments [91]. La fuente que realiza el barrido de frecuencia es *V2 (AC 1)* y la salida de tensión se mide en R_7. La transferencia del circuito se presenta en la Figura 6.23b, donde la respuesta de amplitud se muestra en linea gruesa y la respuesta en fase en línea puntuada. Se puede observar en el cuadro de diálogo agregado en la misma figura que el retardo de grupo es aproximadamente 17 useg dentro de la banda de paso (20 KHz de frecuencia de corte). Asimismo, la atenuación a bajas frecuencias es de 5,68 dB. Finalmente, el efecto de acoplamiento de la entrada descripto anteriormente se comienza a visualizar a partir de los 400 KHz; terminando con una atenuación constante de 78 dB. Como conclusión, la simulación permitió verificar que el diseño cumple con los requerimientos de filtrado.

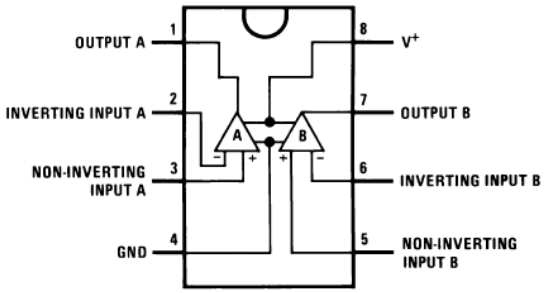


Figura 6.25: Vista superior del encapsulado LMC. Presenta dos operacionales LM358 (A y B) [89].

6.4.4. Fabricación del Periférico

El filtro activo se armó en una placa perforada utilizando resistores de carbón (5 % de tolerancia) y capacitores cerámicos multicapa (10 % de tolerancia). Como algunos componentes no están disponibles comercialmente, se utilizaron combinaciones en serie para lograr los valores de resistencia requeridos (Figura 6.26). El integrado que provee los dos operacionales es el de la Figura 6.25. Cada AO consume 0,5 mA de corriente independientemente de la tensión de alimentación [89]. Conjuntamente, según las especificaciones de los conectores PMOD [70], la placa puede suministrar hasta 100 mA. En consecuencia, no existen problemas en el consumo de corriente. Por otra parte, el rango de tensión de fuente que admite los AO esta comprendido desde 3 hasta 32 Volts. [89].

En la Figura 6.27 se muestra el filtro fabricado. Se puede contemplar sus dimensiones junto con el conector de 12 pines incorporado para enchufar a la placa de desarrollo. Los pines 1, 2, 11 y 12 son utilizados para alimentación (+3.3 V y GND). El pin 6 envía la señal PDM del SoC al filtro. Además, el periférico cuenta con dos cables de salida, uno para la señal analógica (SGN) y otro de referencia (GND).

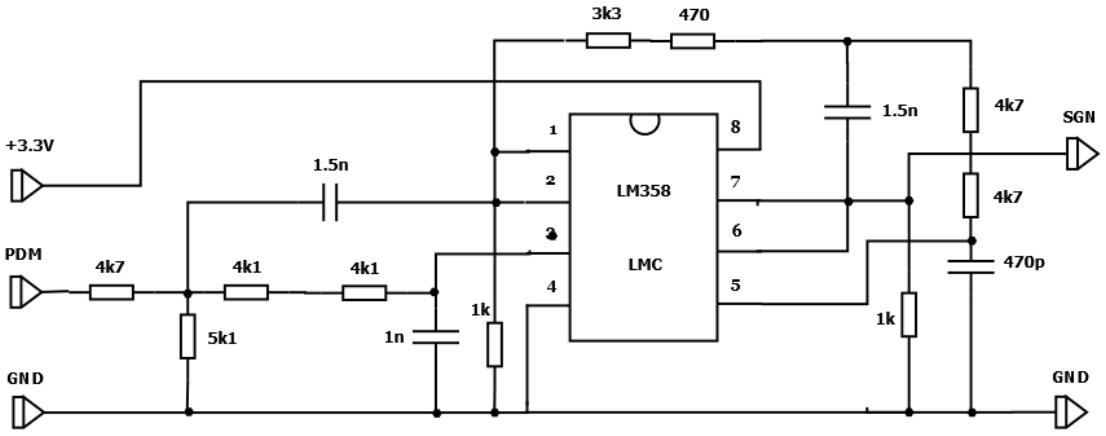


Figura 6.26: Esquemático del periférico desarrollado.

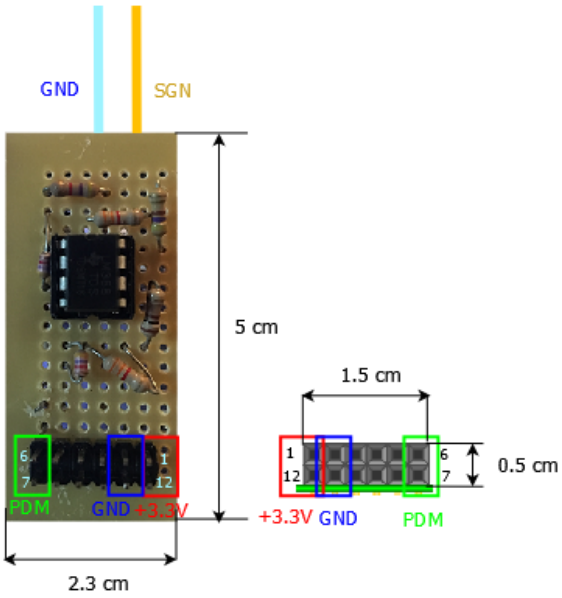


Figura 6.27: Filtro activo en placa perforada (izquierda) y ficha PMOD de la placa de desarrollo (derecha). El periférico presenta conector macho de 12 pines compatibles con la ZedBoard.

6.4.5. Mediciones Experimentales

El banco de medición usado para determinar la respuesta del filtro es el de la Figura 6.28. Se tuvo a disposición un generador de funciones *Hewlett Packard* modelo 33120a para ingresar al circuito con señales de prueba sinusoidales de valor medio 0,6 Volts y amplitud 1 $Volt_{pp}$. Además, con un osciloscopio digital marca *Tektronix* 1052B se visualizaron simultáneamente la tensión de entrada y de salida. La fuente de alimentación fue provista por la placa ZedBoard a travéz de un PMOD.

Variando la frecuencia de la señal de entrada desde 100 Hz hasta 100 KHz, se realizó la Cuadro 6.3 a partir de valores de voltaje y tiempo leídos en la pantalla del osciloscopio.

Frecuencia [KHz]	**Atenuación [dB]**	**Retardo [uSeg]**
.1	-5,72	15,7
.5	-5,7	16,2
1	-5,85	15,8
5	-6,03	16,2
10	-6,72	16
15	-7,55	15,8
20	-8,94	16
40	-19,01	N/A
100	-44,38	N/A

Cuadro 6.3: Valores medidos de atenuación y retardo en el filtro a distintas frecuencias.

Para ejemplificar cómo se efectuaron las mediciones, la Figura 6.29 muestra en el

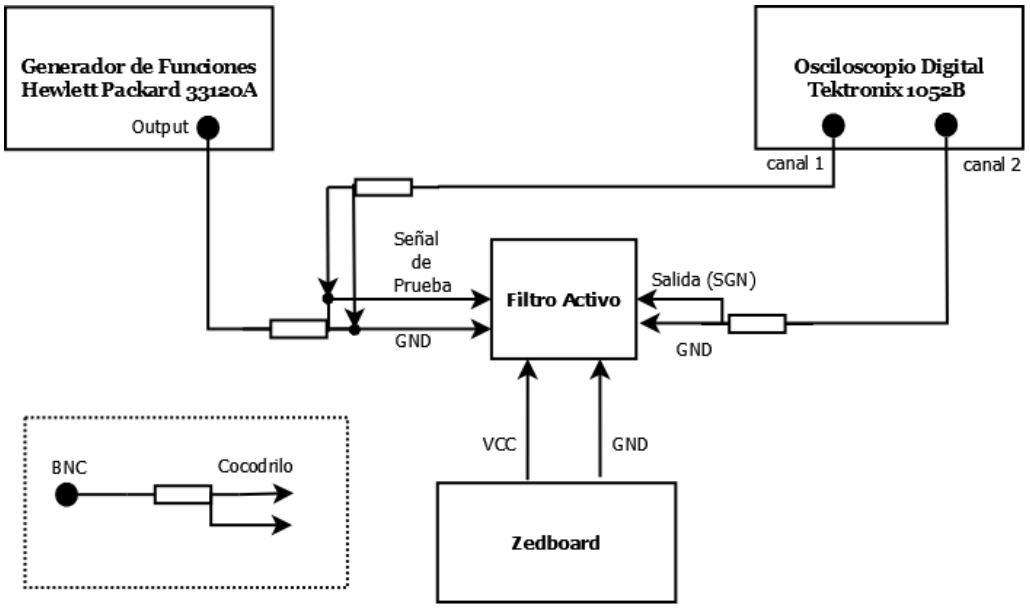
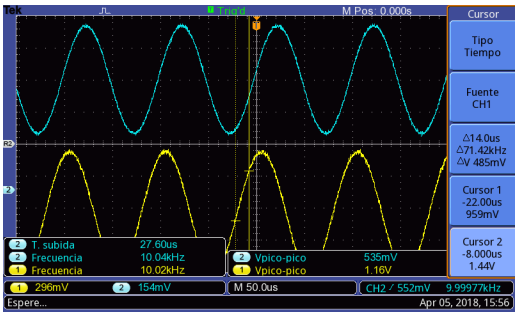


Figura 6.28: Arreglo experimental para evaluar el comportamiento del periférico construido. En el cuadro punteado se encuentra el símbolo que representa al conector BNC/doble cocodrilo que utilizan los instrumentos.

caso de una sinusoidal de 10 KHz la medición automática de Vpp en la entrada (amarillo) es igual a 1,6 Volts mientras que a la salida (azul) vale 0,535 Volts (Figura 6.29a). Después, efectuando la operación $20.log(0{,}535/1{,}16)$ se encuentra que la atenuación a esa frecuencia es -6.72 dB. Adicionalmente, para determinar el retardo se expande la base de tiempo (en este caso a 5 μSeg por división) y con dos cursores de tiempo se mide la diferencia temporal entre los cruces por cero de ambas señales (Figura 6.29b).

Con los datos experimentales, se procedió a realizar el diagrama de Bode de la Figura 6.30. El error adoptado para la amplitud es de $\pm 0{,}15$ dB ya que se corresponde aproximadamente con una incerteza del 1 % en las lecturas de voltaje. Se utilizó un gráfico doble-log para visualizar mejor los rangos. Según puede apreciar-



(a) Medición de amplitud.

(b) Medición de retardo.

Figura 6.29: Captura de pantalla en el osciloscopio digital mientras se medía la respuesta de amplitud y fase del filtro. La entrada se muestra en amarillo y la salida en azul.

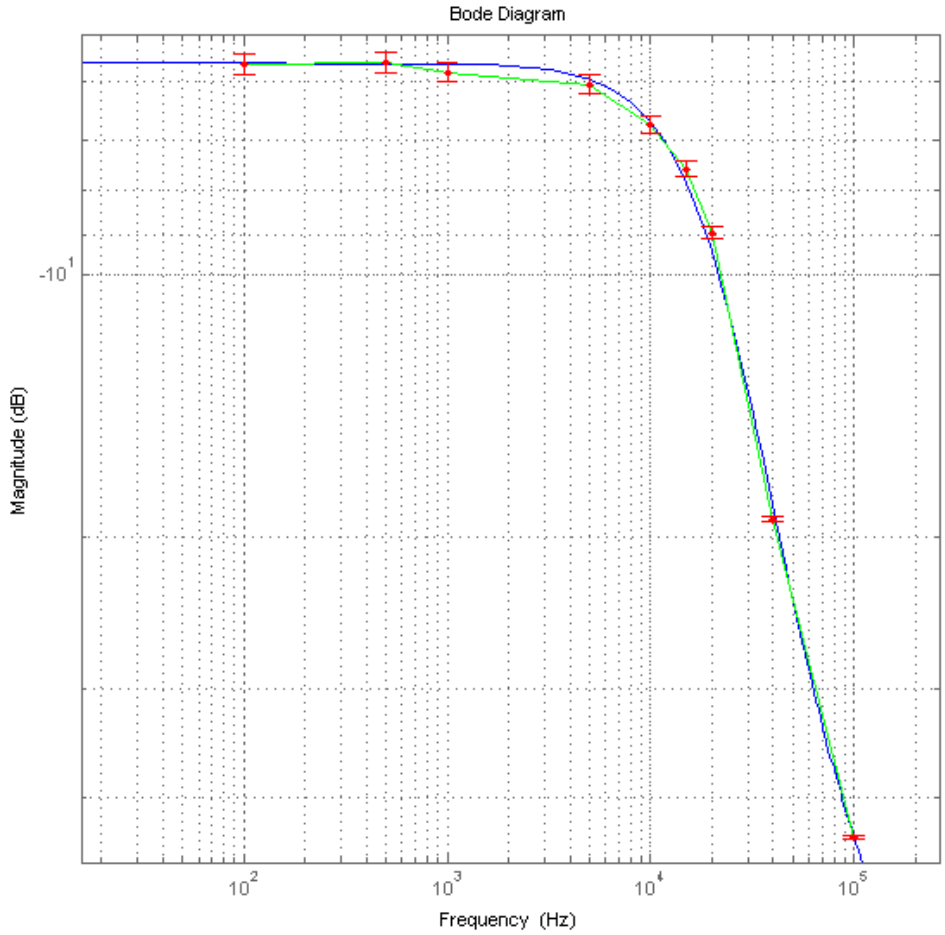


Figura 6.30: Respuesta en amplitud del filtro esperada (línea azul), junto con las mediciones experimentales (puntos rojos) y su interpolación de primer orden (línea verde).

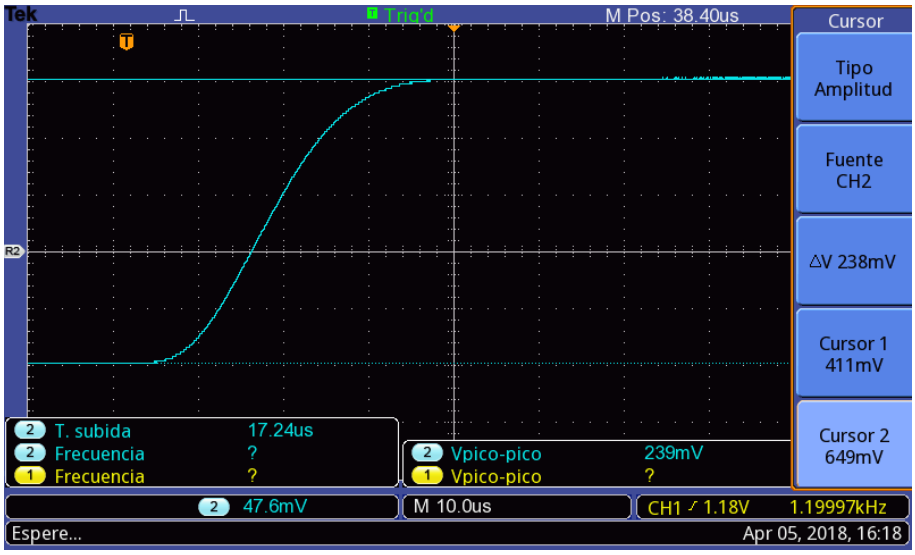


Figura 6.31: Respuesta al escalón experimental del filtro Bessel.

se, la atenuación real del filtro sigue la tendencia establecida por la respuesta teórica. La disminución en amplitud a 20 KHz es de -3,24 dB, por lo que, el error en dB en esa frecuencia es del 6 %; ya que se esperaba que fuese de -3 dB (frecuencia de corte). Como conclusión, se puede enunciar que el filtro fabricado cumple con sus especificaciones de diseño con un error menor al 10 %.

Ahora, sustituyendo la señal de prueba por una onda cuadrada de amplitud 0,5 Volts, valor medio 0,25 Volts y frecuencia 1 KHz se observó la respuesta al escalón del periférico (Figura 6.31). Sus parámetros característicos son presentados a continuación:

- **Tiempo de crecimiento:** 17,24 useg.
- **Tiempo de Demora:** 14 useg.
- **Tiempo de Establecimiento:** 32 useg.
- **Tipo de respuesta:** Sobreamortiguada.

El tiempo de crecimiento está medido desde el 10 % hasta el 90 % del valor final, mientras que el tiempo de demora se establece del 0 % al 50 % y el tiempo de establecimiento del 0 % al 99 %. Se puede reconocer que el filtro activo presenta una respuesta suave y libre de oscilaciones. Recordando que la estabilidad de una etapa Sallen Key se analiza con la transferencia G_2 (Ecuación 6.22) se procede a realizar los gráficos de la Figura 6.32. Usando el criterio de estabilidad de Nyquist se obtiene $N = Z - P = 0$ (nuevamente se verifica la estabilidad). Además, la red de alimentación positiva descripta por G_2 es pasiva; por lo tanto la magnitud de su diagrama de Nyquist nunca alcanza la unidad. Esto se corresponde con la ausencia de *Overshoot*[5] en la respuesta de la Figura 6.31. Es interesante advertir que el tiempo

[5]Desviación máxima del pico en amplitud de la respuesta al escalón con respecto al valor final esperado [92].

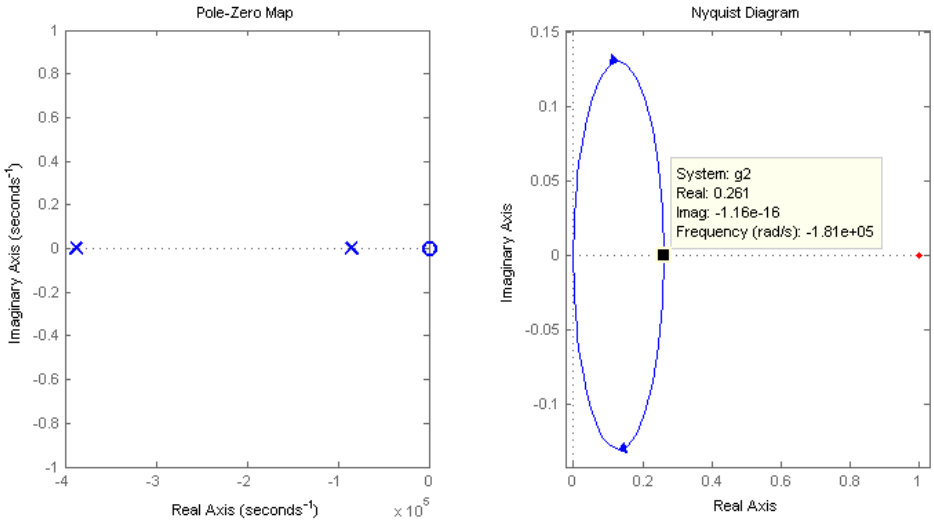


Figura 6.32: Polos y ceros de la transferencia de realimentación $G_2(s)$ (izquierda) junto con su diagrama de Nyquist (derecha) para el caso del filtro Bessel. Como se trata de realimentación positiva, el valor de referencia es $S = 1$ (punto rojo).

de crecimiento es similar al retardo constante teórico dentro de la banda de paso (17 useg). También, si consideramos al tiempo de establecimiento como medio período de una onda sinusoidal se obtiene una frecuencia máxima de 15625 Hz (levemente inferior al ancho de banda de 20000 Hz).

6.5. Interpolador

6.5.1. Principios básicos

La interpolación consiste en dos partes: el aumento de la tasa de muestreo agregando ceros entre muestras (*zero-stuffing* en Inglés) y un filtrado digital pasabajos. El proceso es ilustrado en la Figura 6.33 en el dominio temporal y frecuencial con un ejemplo de interpolación por dos. La primer secuencia $x[n]$ de longitud $N \in \mathbb{N}$ contiene la información de su espectro acotado en el intervalo $[-\pi, \pi]$ de frecuencia, luego se repite periódicamente cada 2π ya que la señal es discreta. Al agregar un cero entre muestras, se obtiene la secuencia $x[m]$. El efecto de *zero-stuffing* en el dominio transformado es el escalamiento del eje de frecuencia por el factor de sobremuestreo M. Es decir, para el caso del ejemplo ($M = 2$), ahora la información de la señal está contenida en $[\frac{-\pi}{2}, \frac{\pi}{2}]$. Después, se aplica un filtro digital con ganancia M en $\omega \in [0, \frac{\pi}{2}]$ y 0 para $\omega \in [\frac{\pi}{2}, \pi]$. El espectro se ve disminuido M veces cuando se realiza la interpolación y en consecuencia el filtro debe compensar esta atenuación. Consultar [62] para la demostración matemática. Como resultado, se consigue la secuencia $x^*[m]$ de longitud $2N$, siendo esta la versión sobremuestreada de $x[n]$.

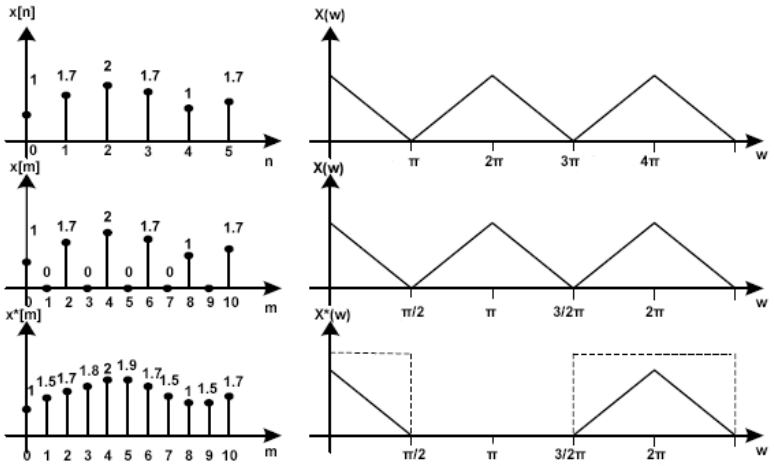


Figura 6.33: Proceso de interpolación digital en el dominio del tiempo discreto (izquierda) y en frecuencia (derecha). Secuencia original (arriba), con *zero-stuffing* (medio) y luego del filtrado (abajo).

La respuesta al impulso del filtro pasabajos ideal es:

$$H_M[m] = M.\text{Sinc}\Big[m\frac{1}{M}\Big] \ , \ m \in [-\infty, \infty] \tag{6.29}$$

Por supuesto que $H_M[m]$ con longitud infinita no es realizable, siendo así que los filtros reales son diseñados en base a tolerancias respecto a la desviación esta característica deseada. Los parámetros a considerar son el ripple en la banda de paso, la atenuación en la banda de rechazo y el ancho de la banda de transición. También, como en el caso analógico, se desea retardo de grupo constante; debido a lo cual se utilizan filtros FIR de respuesta simétrica. Una primera aproximación para la interpolación con $M = 32$ requerida previo al modulador $\Delta\Sigma$ es utilizando un filtro con las características que se muestran en la Figura 6.34. Las especificaciones (atenuación 0,001 dB en la banda de paso y 80 dB en la banda de rechazo) son realistas para la aplicación del DAC. Como puede verse, debido a la elevada tasa de sobremuestreo, la banda de paso junto con la de transición son bastantes angostas en comparación con la banda de Nyquist. Esto produce que el orden necesario sea de 5482. Cuando se tienen demasiados coeficientes como en este caso, la memoria requerida junto a las multiplicaciones por muestras de entrada vuelven ineficiente e impráctica la implementación del interpolador. Afortunadamente, existen estrategias para simplificar la complejidad considerablemente [93].

Una forma de reducir el procesamiento es utilizando varios interpoladores en cascada (Figura 6.35). Para comprender la ventaja de esta implementación, la Figura 6.36 presenta la forma de los filtros en las primeras tres etapas. A medida que avanza la cadena de sobremuestreo, se precisa una banda de paso menos estrecha. Esto se debe a que el espectro periódico de la señal discreta se aleja luego de atravesar cada bloque. La Figura 6.37 muestra más detalladamente cómo es el proceso para aumentar la frecuencia de muestreo.

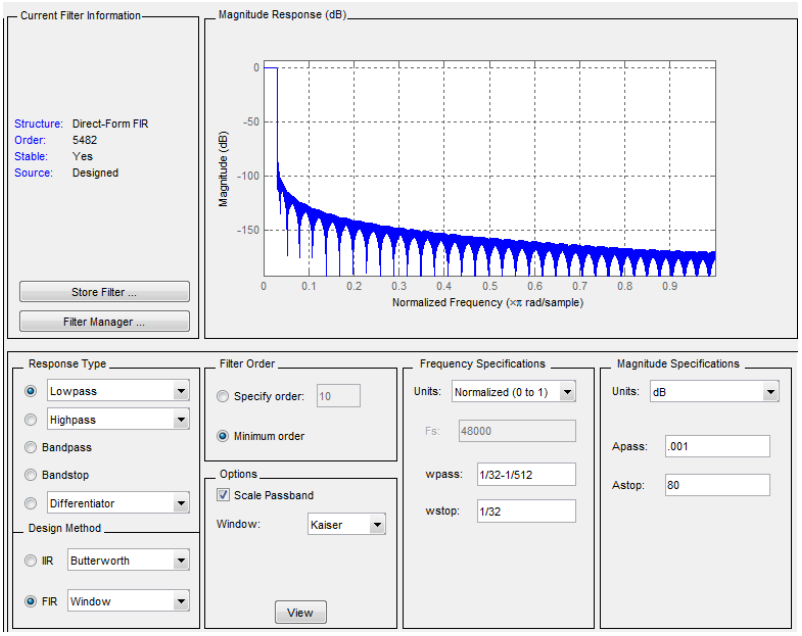


Figura 6.34: Filtro para interpolación con $M = 32$ obtenido con la herramienta *Filter Design & Analysis* del entorno Matlab R2013a.

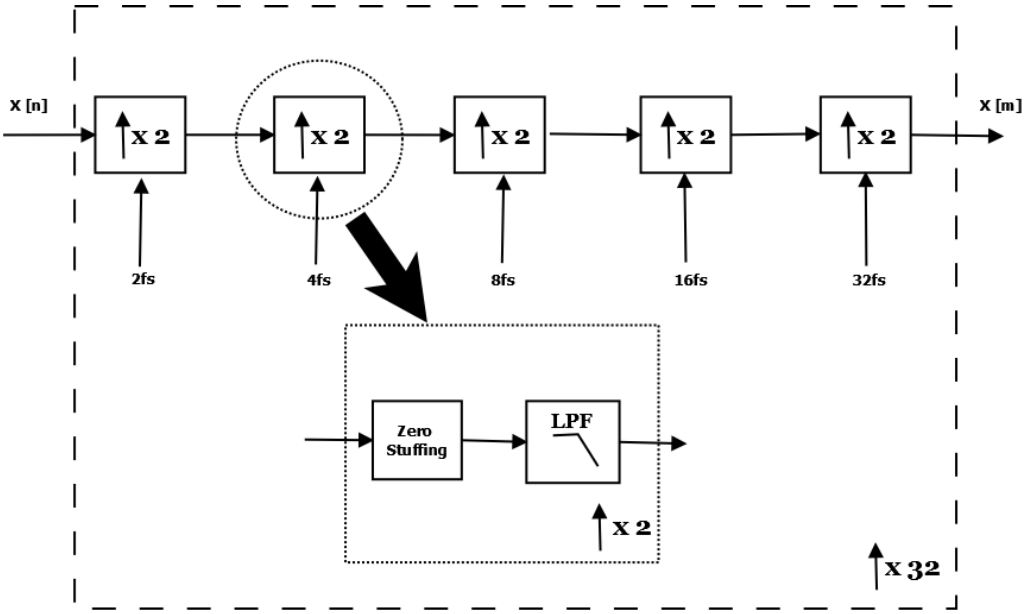


Figura 6.35: Interpolación x32 usando cinco bloques x2 en cascada.

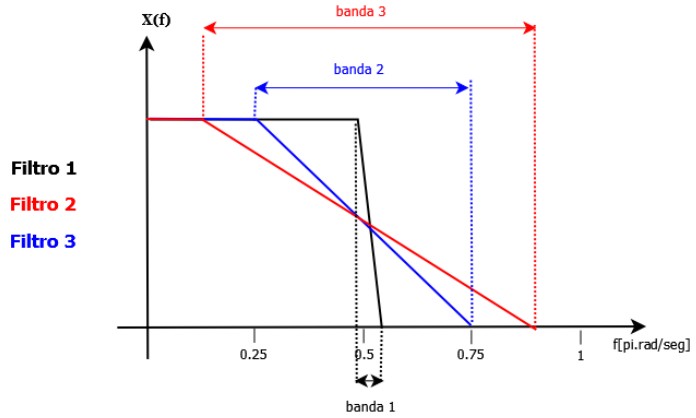


Figura 6.36: Envolventes de los filtros digitales en la interpolación sucesiva.

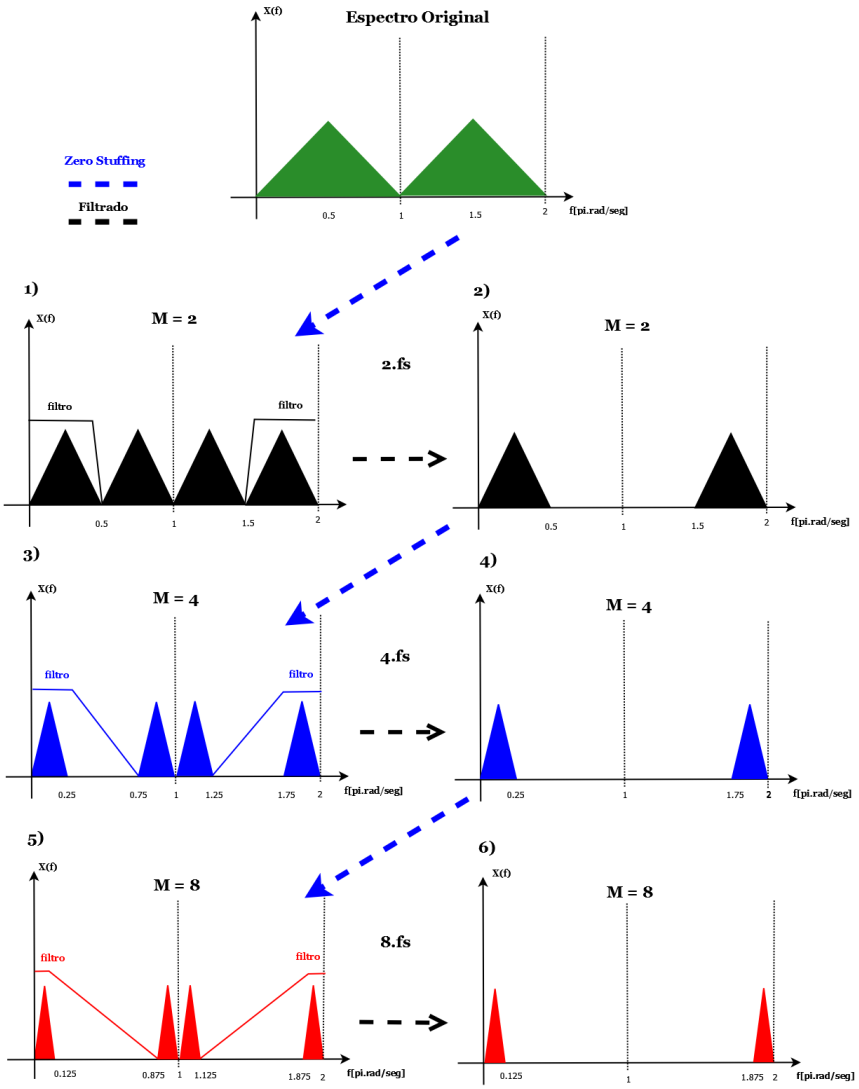


Figura 6.37: Transformación de la señal de entrada en la cadena de sobremuestreo. Cada vez que se realiza *zero-stuffing*, el eje de frecuencia sufre un escalamiento por dos; generando un espectro imagen en las altas frecuencias. Como la distancia entre picos aumenta, los requerimientos de filtrado pasabajos se vuelven menos estrictos.

6.5.2. Filtros FIR de Media Banda

La respuesta al impulso de los filtros FIR de longitud impar y simétricos logran que, mediante convolución discreta con una secuencia $x[n]$, presenten la propiedad de fase lineal [62]. Debido a lo cual, la cantidad de coeficientes a almacenar pueden ser reducidos a la mitad más uno aprovechando la propiedad de simetría. En este caso, si bien se disminuyen los requerimientos de memoria, la cantidad de bloques de retardos y sumadores permanece intacta. La estructura a implementar en VHDL para realizar la convolución se muestra en la Figura 6.38. En ella se puede observar cómo se comparten sumadores y multiplicadores con coeficientes idénticos.

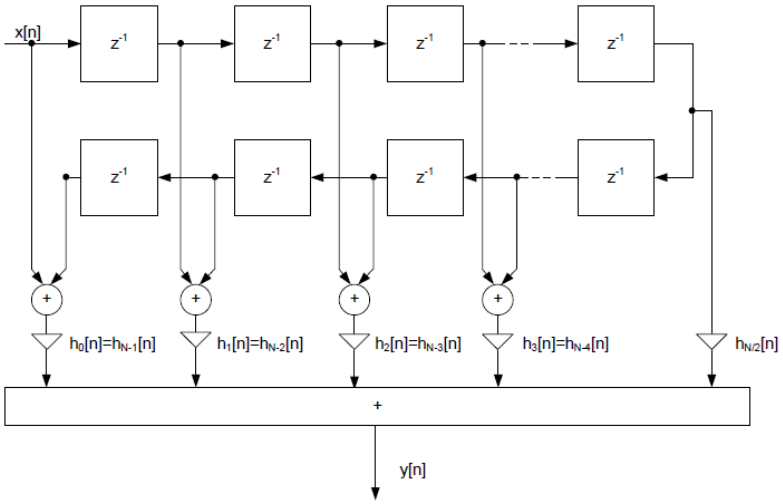


Figura 6.38: Arquitectura para la realización de filtros FIR simétricos

Por otra parte, cuando la respuesta *en frecuencia* es simétrica, o sea, la banda de transición se encuentra exactamente a la mitad de la frecuencia de Nyquist, las muestras impares resultan ser cero (Figura 6.39). Estos filtros son considerados una sub-clases de FIR y se denominan de *Media Banda*. En consecuencia, la cantidad de valores de la respuesta al impulso a utilizar se reducen a aproximadamente un cuarto del total. A su vez, es condición necesaria que el ripple de la banda de atenuación y de paso sean iguales para lograr esta propiedad.

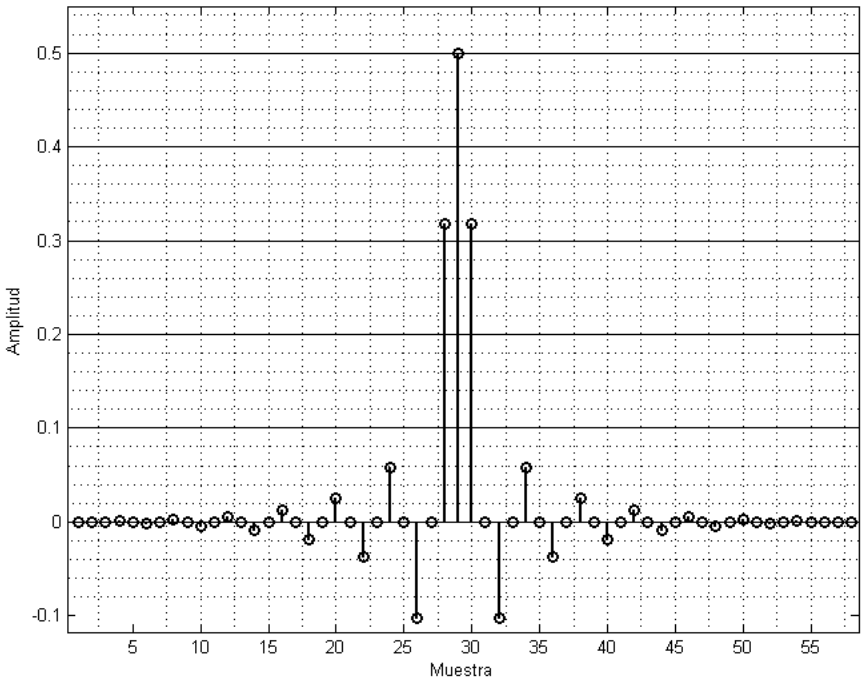


Figura 6.39: Respuesta al impulso de un filtro FIR simétrico de media banda con longitud L=59. Todas las muestras impares (excepto $\frac{L-1}{2} = 29$) son cero. Se requiere almacenar sólo 15 coeficientes en este caso.

6.5.3. Realización Polifásica

El hecho que la señal requiera el agregado de ceros al interpolar mediante el *zero-stuffing* puede ser aprovechado usando la *descomposición polifásica* [93]. Para ilustrar este principio, se muestra a continuación una respuesta al impulso $H(z)$ de longitud $L = 7$:

$$H(z) = h[0] + h[1]z^{-1} + h[2]z^{-2} + h[3]z^{-3} + h[4]z^{-4} + h[5]z^{-5} + h[6]z^{-6} \quad (6.30)$$

En el caso general, conviene reagrupar en potencias de z^{-m}; donde m es el factor de interpolación. Reagrupando (6.30) convenientemente en potencias de z^{-2} se logra:

$$H(z) = (h[0] + h[2]z^{-2} + h[4]z^{-4} + h[6]z^{-6}) + z^{-1}(h[1] + h[3]z^{-2} + h[5]z^{-4}) \quad (6.31)$$

Considerando un filtro de media banda se tiene que $h[1] = h[3] = 0$, entonces:

$$H(z) = (h[0] + h[2]z^{-2} + h[4]z^{-4} + h[6]z^{-6}) + z^{-1}(h[5]z^{-4}) \tag{6.32}$$

Llamamos al primer paréntesis de (6.32) E_0 y al segundo E_1. Ergo, la descomposición polifásica de la respuesta al impulso es $H(z) = E_0 + z^{-1}E_1$. Volviendo al caso de la implementación en cascada con bloques de sobremuestreo x2, la arquitectura que realiza la interpolación con el filtrado polifásico es el de la Figura 6.40. Se sabe que, en este caso, una de cada dos muestras que ingresan al filtro es cero. Por lo tanto, podemos reemplazar las potencias z^{-2} en cada término E_i por z^{-1}. Esto permite utilizar cada rama de la Figura 6.40 de manera concurrente y con la misma tasa de muestreo que la señal $x[n]$ de entrada. Más aún, el retardo y el sumador pueden ser sustituidos por una llave que conmuta entre E_0 y E_1 al doble de frecuencia y obtener así dos muestras por cada entrada; logrando el efecto de sobremuestreo.

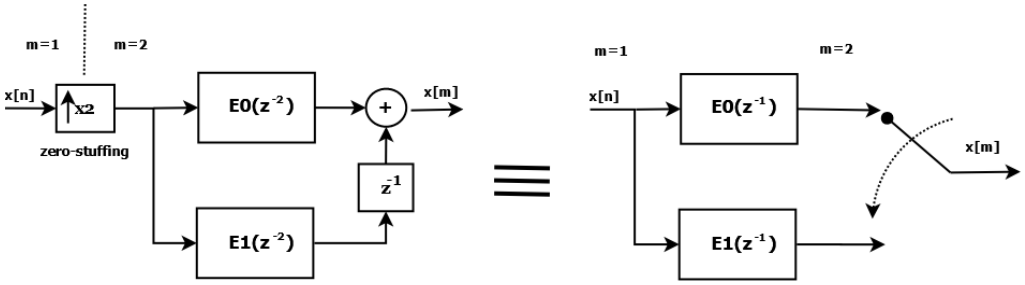


Figura 6.40: Interpolador x2 con filtro de estructura polifásica (izquierda), junto con su realización equivalente más eficiente (derecha).

6.5.4. Estructura de conmutación

Habiendo presentado cómo disminuir la cantidad necesaria de coeficientes en los filtros, junto con la forma polifásica que evita el agregado de ceros, se muestra ahora el interpolador x32 realizado para el DAC. Utilizando cinco estructuras de conmutación en cascada, como exhibe la Figura 6.41, se logra obtener una réplica 32 veces sobremuestreada de la señal $x[n]$. En cada etapa, la frecuencia de muestreo f_s aumenta el doble, ya que cada llave conmuta dos veces más rápido que la anterior. Luego de un transitorio inicial, se genera la señal de salida $x[m]$ manifestando un retardo temporal en relación a $x[n]$.

Las especificaciones utilizadas en los filtros fueron 80 dB de atenuación en la banda de rechazo y ripple de 0,001 dB en la banda de paso. La función *firhalfband()* disponible en Matlab R2013a permitió generar los coeficientes FIR simétricos y de media banda. Las frecuencias digitales de corte resultaron $0,5\pi$ (filtro 1), $0,25\pi$ (filtro 2), $0,125\pi$ (filtro 3) y $0,0625\pi$ (filtro 4 y 5). En la Cuadro 6.4 se encuentran el orden N resultante de cada filtro y los valores de su respuesta al impulso $h[n]$ relevantes. Los bloques $E_0(z^{-1})$ se realizaron con la estructura de la Figura 6.38 empleando sólo las

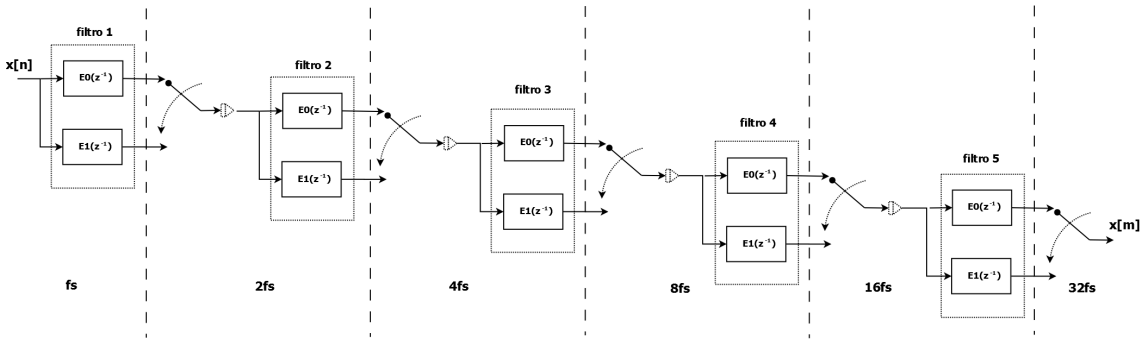


Figura 6.41: Diagrama en bloques del interpolador realizado en FPGA.

muestras de h[n] pares (n = 0, 2, 4..., $\frac{N}{2}$-1), ya que h[n] = h[n-N]:

$$E_0(z) = \sum_{i=0}^{\frac{N}{2}-1} z^{-i} h[2i] \tag{6.33}$$

Observando los $h[N/2]$, se encuentra que en todos los filtros valen 0,5. Entonces, $E_1(0)$ se construye de la siguiente manera:

$$E_1(z) = \frac{1}{2} z^{-\frac{N/2-1}{2}} \tag{6.34}$$

Por lo tanto, cada bloque interpolador x2 consiste en una estructura para FIR simétricos y una cadena de retardo ambas trabajando en paralelo. A su vez, es necesario aplicar una ganancia por dos a la salida de las etapas para compensar la disminución que provoca el sobremuestreo en este caso [62]. Con esto, se omite realizar la multiplicación por $\frac{1}{2}$ en (6.34). Más adelante se verá que la ganancia en (6.33) se efectúa con un registro desplazamiento.

A modo de realizar una comparación, al principio de esta sección se enunció que un solo filtro requería 5483 coeficientes (N+1). Sin ningún tipo de optimización, se efectúan 5483.32 = 175456 multiplicaciones por muestra de entrada; ya que el sobremuestreo es x32. En el sistema de la Figura 6.41 se utilizan sólo 15 + 4 + 3 + 2 + 2 = 26 multiplicadores. Puede apreciarse que la reducción en costos computacionales son bastante considerables.

En resumen, se logró realizar una implementación eficiente del interpolador utilizando menos recursos gracias a:

- La realización en cascada con bloques x2.
- La respuesta al impulso simétrica de los filtros.
- Utilizando FIRs de media banda.
- Estructura polifásica de conmutación.

	Filtro 1	**Filtro 2**	**Filtro 3**	**Filtro 4**	**Filtro 5**
Orden	58	14	10	6	6
$h[0]$	1.45×10^{-4}	2.881×10^{-3}	6.8×10^{-3}	-0.032	-0.032
$h[2]$	-3.44×10^{-4}	0.0182	-0.0515	0.282	0.282
$h[3]$	-	-	-	0.5	0.5
$h[4]$	7.25×10^{-4}	-0.0693	0.2947	-	-
$h[5]$	-	-	0.5	-	-
$h[6]$	1.35×10^{-3}	0.304	-	-	-
$h[7]$	-	0.5	-	-	-
$h[8]$	2.33×10^{-3}	-	-	-	-
$h[10]$	3.77×10^{-3}	-	-	-	-
$h[12]$	5.83×10^{-3}	-	-	-	-
$h[14]$	-8.7×10^{-3}	-	-	-	-
$h[16]$	0.0127	-	-	-	-
$h[18]$	-0.0182	-	-	-	-
$h[20]$	0.026	-	-	-	-
$h[22]$	-0.037	-	-	-	-
$h[24]$	0.058	-	-	-	-
$h[26]$	-0.1026	-	-	-	-
$h[28]$	-0.3171	-	-	-	-
$h[29]$	0.5	-	-	-	-

Cuadro 6.4: Cuadro de coeficientes para los filtros en el interpolador. Se omitieron las muestras que valen cero y la segunda mitad de h[n], ya que la respuesta es simétrica.

6.6. Proyecto en Vivado

El diseño en bloques creado en Vivado para poner bajo prueba al DAC es el de la Figura 6.42. Se puede divisar el ZYNQ PS, donde se ejecuta el software, un controlador AXI DMA para transferencia directa desde la memoria del PS hacia el PL, el *AXI Interconnect* que funciona como traductor entre protocolos AXI y un bloque llamado *DAC* que efectúa el sistema conversión D/A desarrollado. Este último presenta un bus de entrada de datos *AXI-STREAM* llamado *data in* (longitud de 32 bits), señal de reloj *clk*, señal de reset *ap_rst* y la salida *dac out* (longitud 1 bit) que genera el PDM enviado al filtro a través del pin V12 (ver [69] para más detalles del *Pin-Out*).

El programa en C creado en Xilinx SDK se encuentra en el Apéndice A.1.4. Básicamente, la función *DMA_INIT()* configura e inicializa el controlador AXI DMA; luego se resetea al DAC con la función *XGpio_DiscreteWrite ()* a través un bloque AXI_GPIO en el PL. La señal de prueba en punto flotante se genera en el vector llamado *datosFloat[ ]* con longitud 1024, seguidamente se formatea a punto fijo con la función *convert to_dac_format()* y se envía al PL con *DMA_TRANSFER()*.

La Figura 6.44 muestra el interior del bloque *DAC*. El *DAC_CONTROLLER* es un IP desarrollado en *High Level Synthesis* que recibe la señal $x[n]$ enviada desde el PS a través de AXI stream, la almacena en una memoria RAM que funciona como buffer circular y la transmite periódicamente hacia al interpolador; con una frecuencia de muestreo $f_s = 48828{,}125$ Hz (Ver Apéndice A.1.2). Después, sucede la secuencia de interpoladores constituida por los IP *INTER_X*. Cada uno de estos núcleos cuenta con una entrada y salida de datos de 24 bits (*data_in* y *data_out* respectivamente), *RESET* activo alto y dos entradas de reloj: una llamada *clk* de frecuencia igual a la de muestreo en la *i-ésima* etapa ($i.f_s$) y otra *OSRclk* (reloj de sobremuestro) funcionando a $2 \times i.f_s$.

Todos los componentes en el conversor están sincronizados gracias al distribuidor de reloj llamado *CLOCK_DIST*. Este elemento recibe la señal *clk* global del PL (100 MHz) e implementa un contador módulo 32, produciendo así una señal interna de 3,125MHz llamada *TEMPORAL*. En cada flanco positivo de *TEMPORAL* se incrementa un registro de 6 bits en uno. La salida ***x1*** está conectada al MSB del registro, ***x2*** al MSB-1 y así sucesivamente hasta llegar a ***x32*** unido al LSB. La Cuadro 6.5 muestra la frecuencia de señal en cada pin del distribuidor. A su vez, se presenta sus formas de onda obtenidas en simulación (Figura 6.43). El *DAC_CONTROLLER* envía una muestra de $x[n]$ al interpolador en cada flanco positivo de ***a***. Seguidamente ***x1***, que está en contrafase de ***a***, comanda la lectura de datos a la entrada de *INTER_X2* que genera a su vez una muestra a la salida cada vez que ***x2*** esta en alto. De esta manera, los bloques *INTER* subsiguientes operan al doble de frecuencia respecto a la anterior gracias a las restantes señales del distribuidor. En la última fase, el modulador $\Delta\Sigma$ usa a ***X32*** como señal de reloj.

Para detallar la operación de los bloques interpoladores, en la Figura 6.45 se aprecia

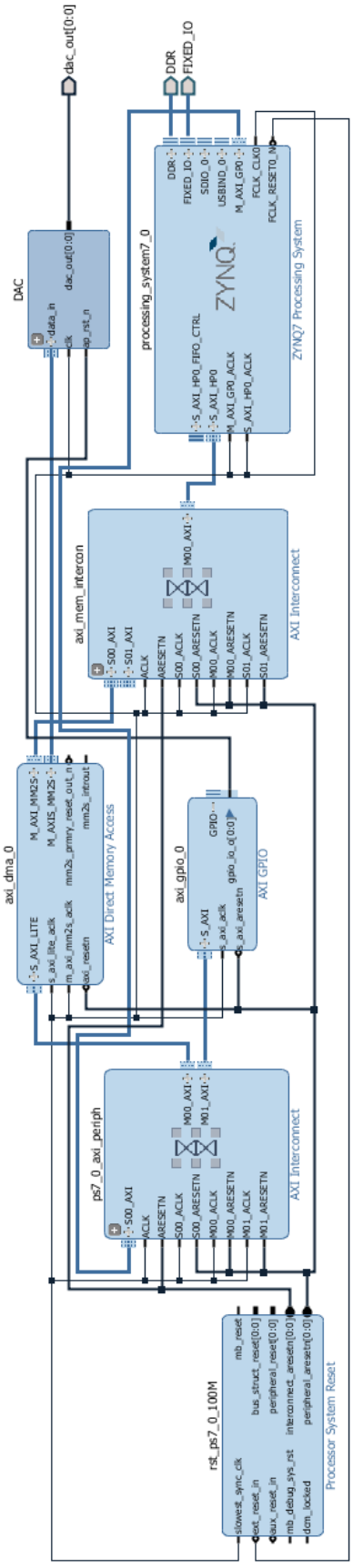


Figura 6.42: Diseño de Bloques en *Vivado Design Suite* para realizar pruebas en el DAC

Pin	Frecuencia [KHz]
a	48,828
x1	48,828
x2	97,656
x4	195,312
x8	390,625
x16	781,125
x32	1562,5

Cuadro 6.5: Frecuencias para la señales de reloj a la salida de los pines en el bloque *CLOCK_DIST*.

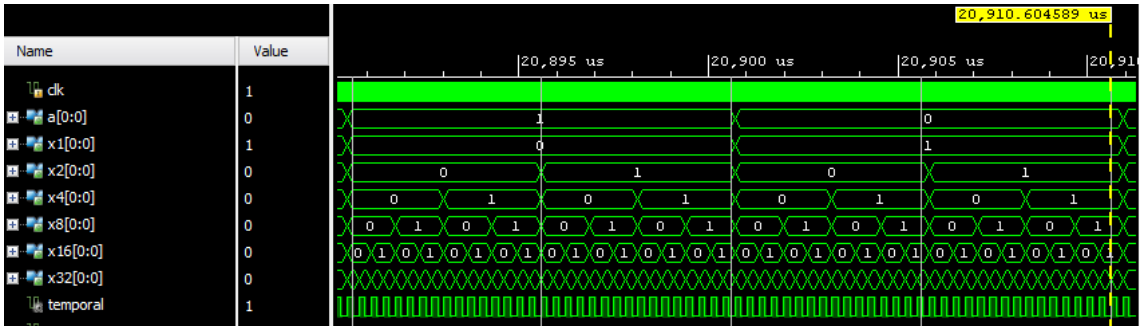


Figura 6.43: Simulación lógica realizada en Vivado del distribuidor de reloj. La frecuencia de *clk* es 100 MHz.

los elementos constituyentes de *INTER_X2*. En primer lugar, se encuentran los bloques *fir_X2* y *DELAY_14* que se corresponden respectivamente con E_0 y E_1 de la representación polifásica. Luego, el multiplexor *MUX* junto con el contador módulo dos *COUNTER* hacen de llave de conmutación. Cuando la entrada *SELECTOR* vale '0', se selecciona la salida el filtro FIR y en caso contrario la del retardo. Por último, se genera un latch de datos (*LATCH_0*) por flanco positivo de reloj para dejar un resultado estable a la parte siguiente. Todo los códigos VHDL del sistema están el la Sección A.1.3 del Apéndice.

En relación a los filtros FIR, los coeficientes fueron representado en formato $Q_{1,17}$ mientras que las muestras de la señal en $Q_{7,17}$. Cuando se produce la multiplicación, el resultado de 42 bits se almacena en acumuladores/sumadores. Utilizar estas longitudes de datos resulta ventajoso, ya que los bloques *DSP* en el FPGA contienen internamente multiplicadores de 18×25 bits y acumuladores de 48 bits [94]. Por lo que, se necesita sólo 26 de estos elementos para realizar el sobremuestreo. Además, se obtiene el beneficio del paralelismo gracias a la estructura de la Figura 6.38. Para volver a convertir el resultado del producto a formato $Q_{7,17}$ es necesario realizar un desplazamiento binario hacia la derecha de 17 lugares. Sin embargo, como también se debe aplicar una ganancia por dos según lo dicho en la sección del interpolador, el desplazamiento finalmente se realiza con 16 lugares.

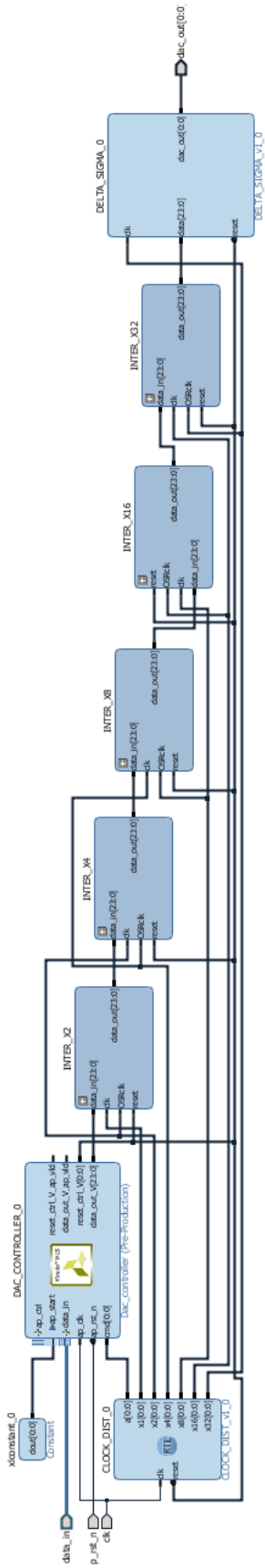


Figura 6.44: Sistema de conversión D/A con Interpolador x32 y modulador $\Delta\Sigma$ de segundo orden (Bloque DAC).

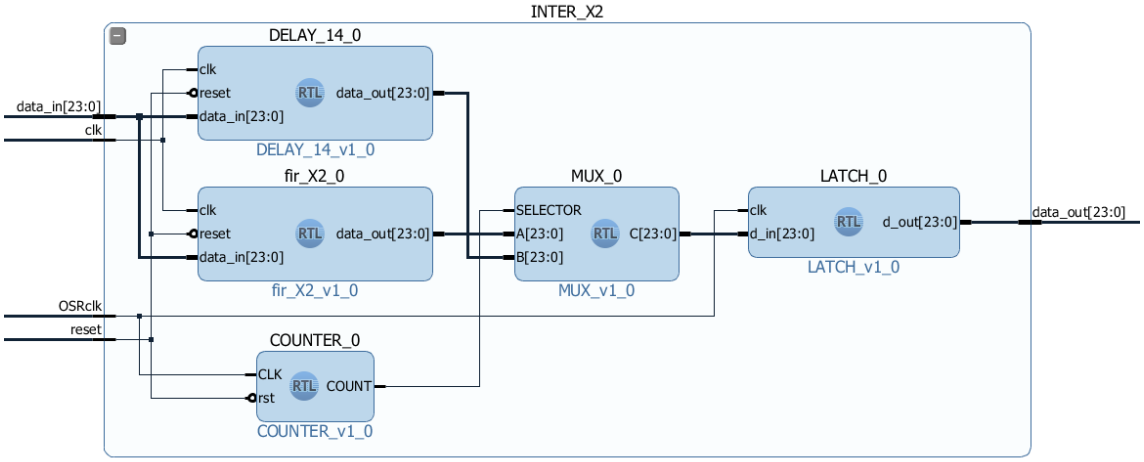


Figura 6.45: Interior de un bloque interpolador x2.

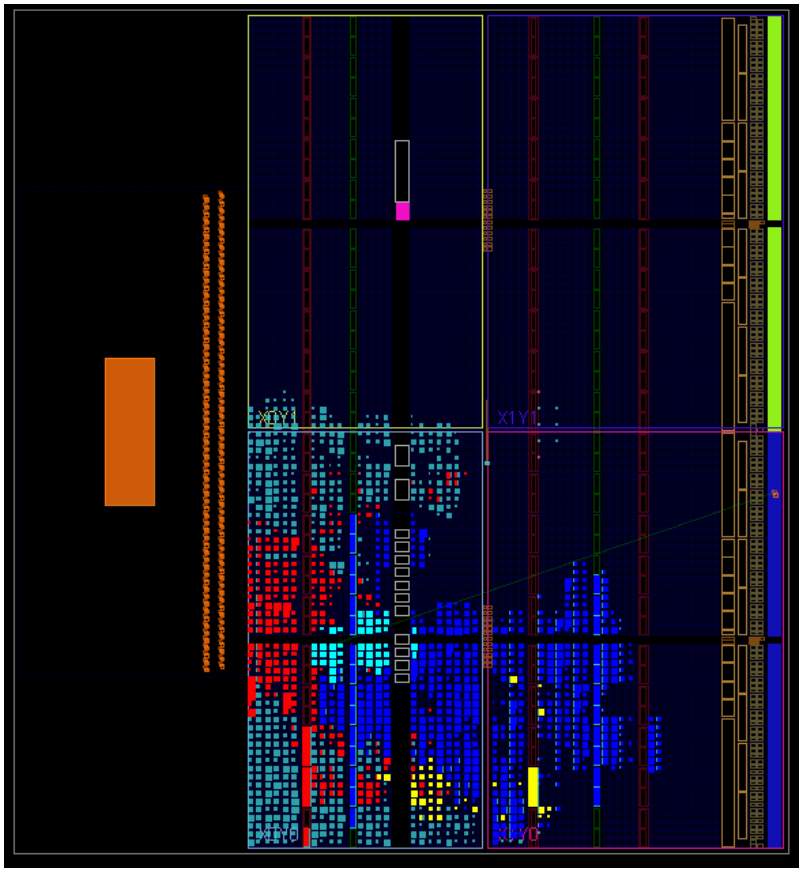


Figura 6.46: Ubicación de los bloques del DAC en el SoC. En rojo: AXI DMA, en azul: INTERPOLADOR, en Amarillo: DAC_CONTROLLER y en Celeste: MODULADOR $\Delta\Sigma$.

Luego de la síntesis e implementación del diseño en *Vivado Design Suite*, se obtuvo la utilización del área en el ZYNQ 7000 (Figura 6.46). Para la generación del bistream no fue utilizada ninguna directiva de *síntesis* ni de *placement*. Finalmente, los recursos necesarios para el conversor D/A diseñado en este capítulo se ven la Figura 6.47.

Hierarchy

Name	Slice LUTs (17600)	Slice Registers (35200)	Slice (4400)	LUT as Logic (17600)	LUT as Memory (6000)	LUT Flip Flop Pairs (17600)	Block RAM Tile (60)	DSPs (80)	Bonded IOB (100)	Bonded IOPADs (130)	BUFGCTRL (32)
design_1_wrapper	3406	4312	1306	3247	159	1300	3.5	26	1	130	1
design_1_i (design_1)	3406	4312	1306	3247	159	1300	3.5	26	0	0	1
axi_dma_0 (design_1_a…	525	761	246	490	35	290	2.5	0	0	0	0
axi_gpio_0 (design_1_a…	27	26	10	27	0	17	0	0	0	0	0
axi_mem_intercon (desi…	604	832	238	566	38	282	0	0	0	0	0
DAC (DAC_imp_1YBUH12)	1687	1972	611	1663	24	359	1	26	0	0	0
CLOCK_DIST_0 (desi…	10	13	4	10	0	9	0	0	0	0	0
DAC_CONTROLLER…	67	157	52	67	0	16	1	0	0	0	0
DELTA_SIGMA_0 (de…	180	75	48	180	0	49	0	0	0	0	0
INTER_X2 (INTER_X…	508	831	231	484	24	77	0	15	0	0	0
INTER_X4 (INTER_X…	220	314	98	220	0	39	0	4	0	0	0
INTER_X8 (INTER_X…	184	242	78	184	0	42	0	3	0	0	0
INTER_X16 (INTER_…	259	170	78	259	0	59	0	2	0	0	0
INTER_X32 (INTER_…	259	170	87	259	0	50	0	2	0	0	0
processing_system7_0 (…	0	0	0	0	0	0	0	0	0	0	1
ps7_0_axi_periph (desig…	547	692	243	486	61	331	0	0	0	0	0

Figura 6.47: Utilización de recursos totales (LUTs, BRAM, DSPs, etc.) del DAC desarrollado.

6.7. Resultados

Una vez construido el conversor D/A y verificado su diseño mediante simulaciones, se armó el banco de medición de la Figura 6.48; con el fin de evaluar su funcionamiento real. El canal uno del osciloscopio digital se usa para visualizar la señal PDM producida por el modulador $\Delta\Sigma$ mientras que en el canal dos se mide la señal analógica resultante después del filtrado. Primero, se enviaron las secuencias constantes $x[n] = 0$, $x[n] = 0{,}9$ y $x[n] = -0{,}9$ obteniéndose las formas de onda de la Figura 6.49. Posteriormente, se generó una señal sinusoidal de amplitud 0,9 y frecuencia 1 KHz(Figura 6.50). También, se genero una señal diente de sierra (Figura 6.51) de manera tal que su período abarque exactamente la longitud de 1024 muestras de $x[n]$.

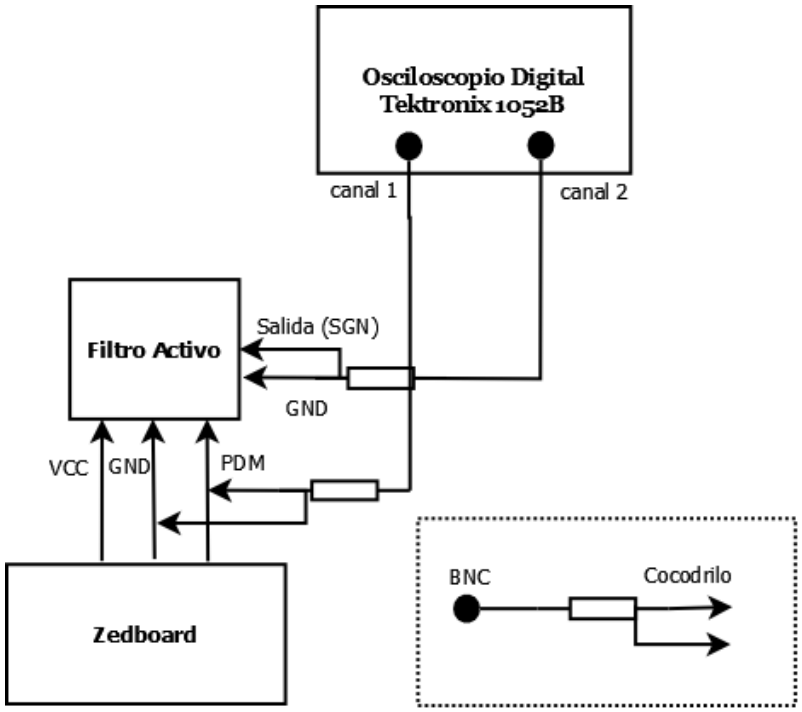
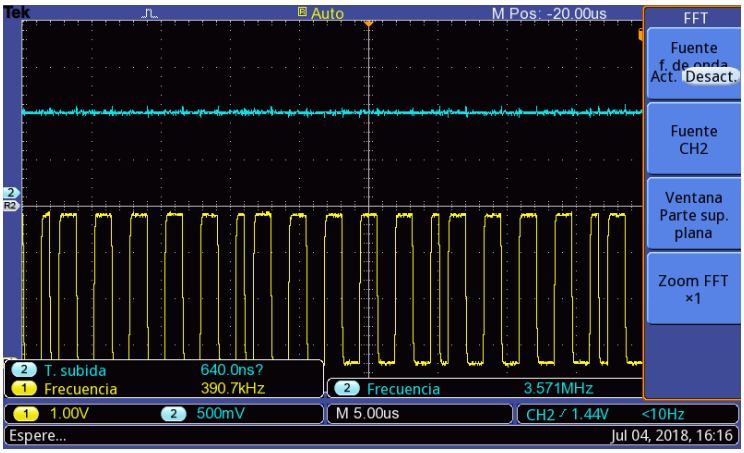
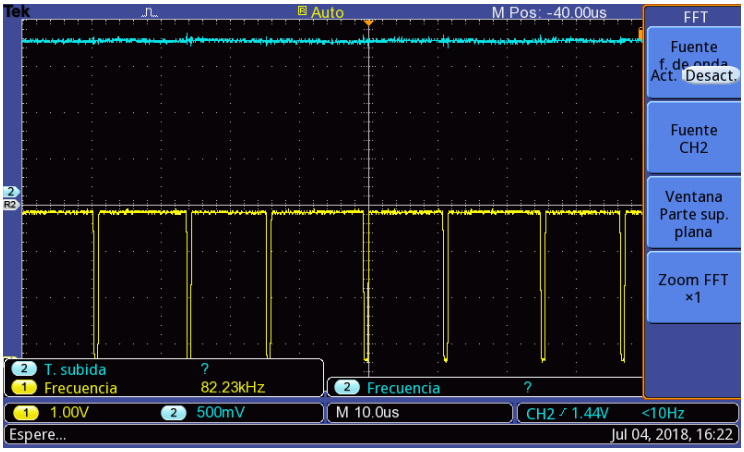


Figura 6.48: Banco de medición para el DAC.

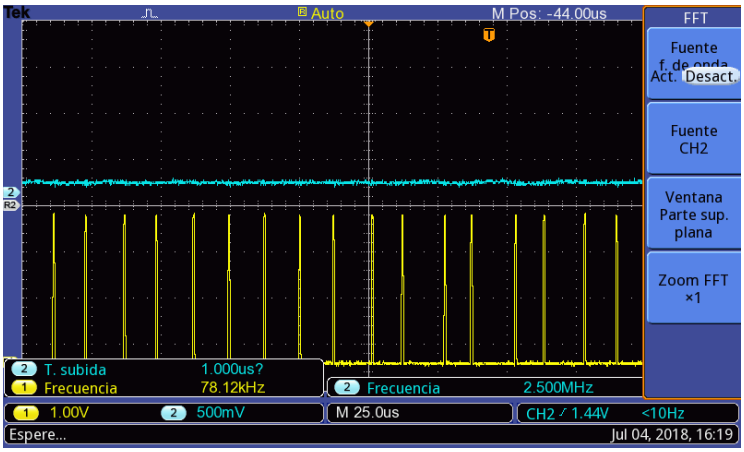
El osciloscopio presenta la función de mostrar el contenido espectral mediante FFT de la señal que está bajo prueba. Contemplando la medición (Figura 6.52), se aprecia algunas componentes espurias generadas por la no linealidad del modulador. Por otro lado, la relación señal a ruido es de 52 dB. Este valor es inferior al los 70 dB alcanzados en simulación. No obstante, si se tiene en cuenta que el osciloscopio adquiere con una resolución de 8 bits, la SQNR resulta empleando la ecuación (6.1) de 50 dB. Esto se correspondiéndose con la relación medida. En definitiva, este instrumento no es adecuado para determinar la SNR del conversor.



(a) Conversión D/A para x = 0.

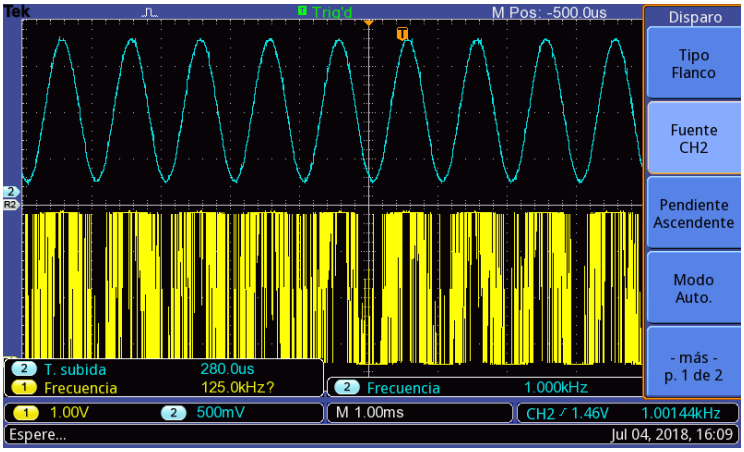


(b) Conversión D/A para x = 0.9

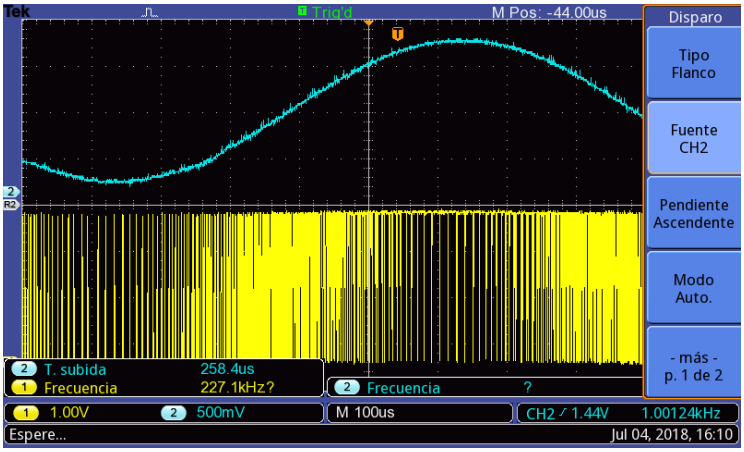


(c) Conversión D/A para x = -0.9

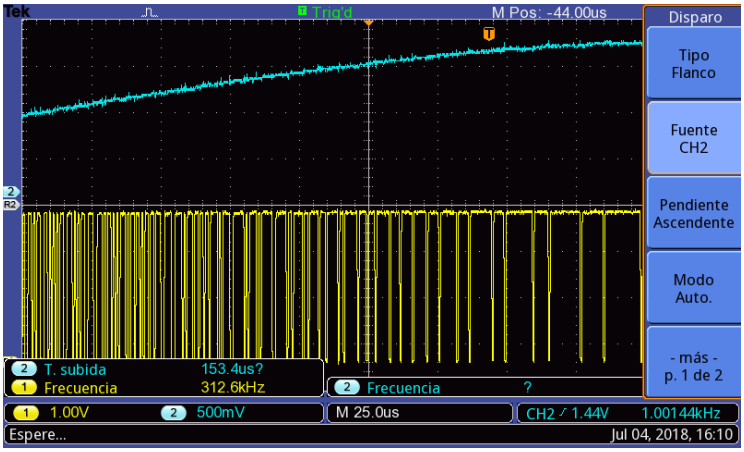
Figura 6.49: Medición de las respuestas del DAC ante entradas constantes. En amarillo: señal PDM. En celeste: señal analógica.



(a) Conversión D/A para x sinusoidal (base de tiempo = $1mseg.$)

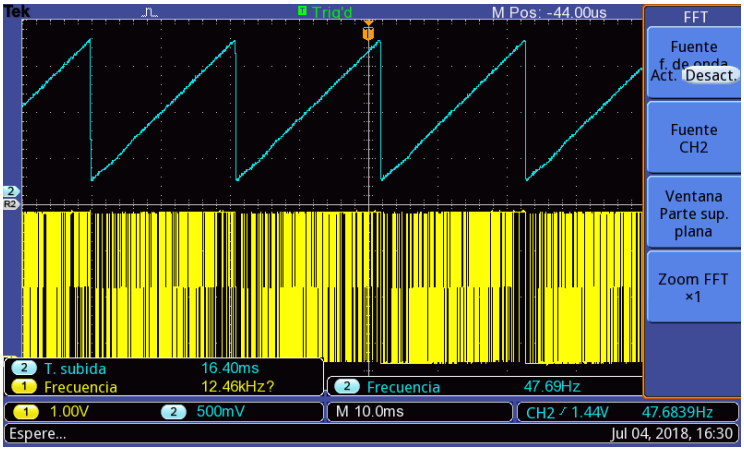


(b) Conversión D/A para x sinusoidal (base de tiempo = $0{,}25mseg.$)

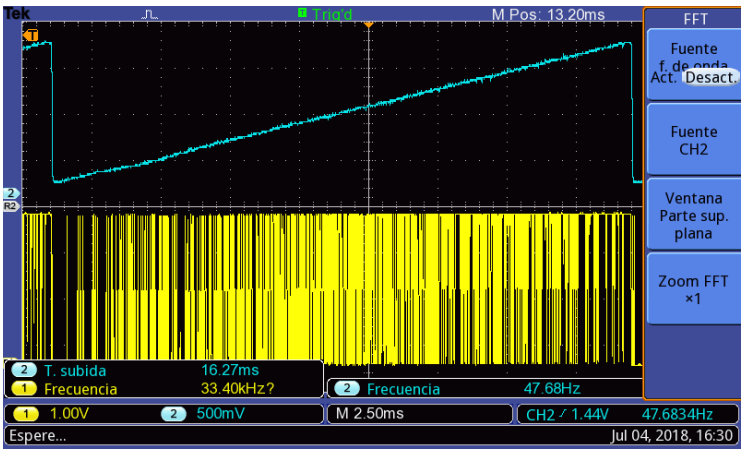


(c) Conversión D/A para x sinusoidal (base de tiempo = $5\mu seg.$)

Figura 6.50: Medición de las respuesta del DAC ante una entrada sinusoidal (1.000 Hz). En amarillo: señal PDM. En celeste: señal analógica.



(a) Conversión D/A para x diente de sierra (base de tiempo = $1mseg.$)



(b) Conversión D/A para x diente de sierra (base de tiempo = $5mseg.$)

Figura 6.51: Medición de las respuesta del DAC ante una entrada diente de sierra (47,68 Hz). En amarillo: señal PDM. En celeste: señal analógica.

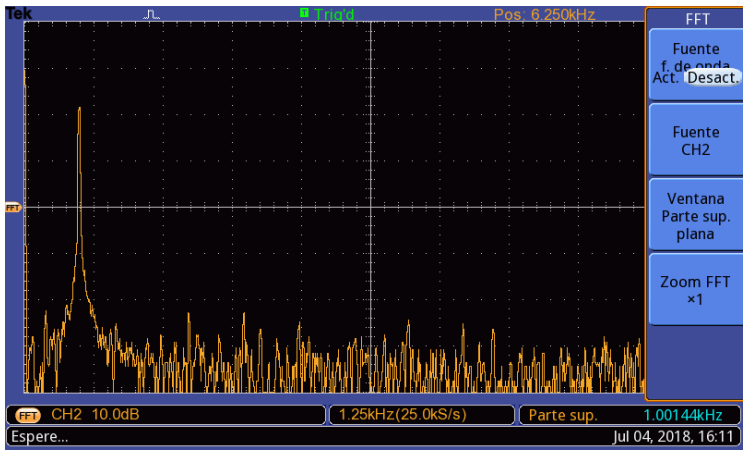


Figura 6.52: FFT a la salida del filtro para una señal sinusoidal de 1000 Hz y máxima amplitud posible.

Capítulo 7

Resultados Experimentales

En este capítulo se muestra los resultados obtenidos utilizando el sistema de Sensado Compresivo desarrollado e implementado en esta tesis. Se hace uso de las herramientas que proporciona la interfaz creada para realizar un análisis cuantitativo con tres señales de prueba: radar de pulso, señales de voz y señales moduladas en amplitud. El diccionario utilizado es la *transformada discreta coseno* o DCT de tipo II. La tasa de muestreo es 16 veces menor que la dictada por el teorema de Nyquist- Shannon. Se debe remarcar que las secuencias de mezclado utilizadas fueron las generadas por el mapa *Logístico*, debido a que ofrecen la mejor calidad de reconstrucción según lo expuesto en 3.3.3.

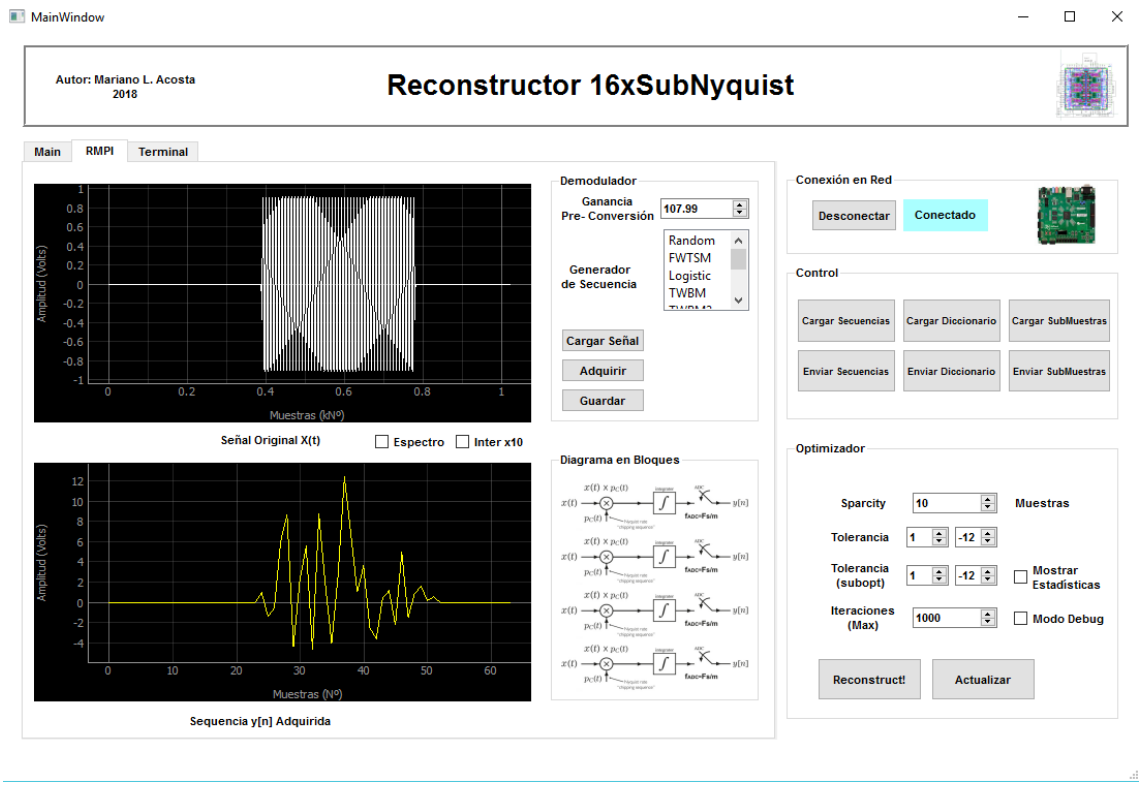
7.1. Radar de Pulso

Un sistema de radar consiste en un transmisor y un receptor de ondas electromagnéticas. Se basa en la emisión de un pulso de portadora y en la recepción de su versión reflejada por un obstáculo. Por lo tanto, sirve para la detección y localización de objetos, como para la determinación de superficies. Sus bandas de operación van desde los 3 MHz hasta los 110 GHz [95].

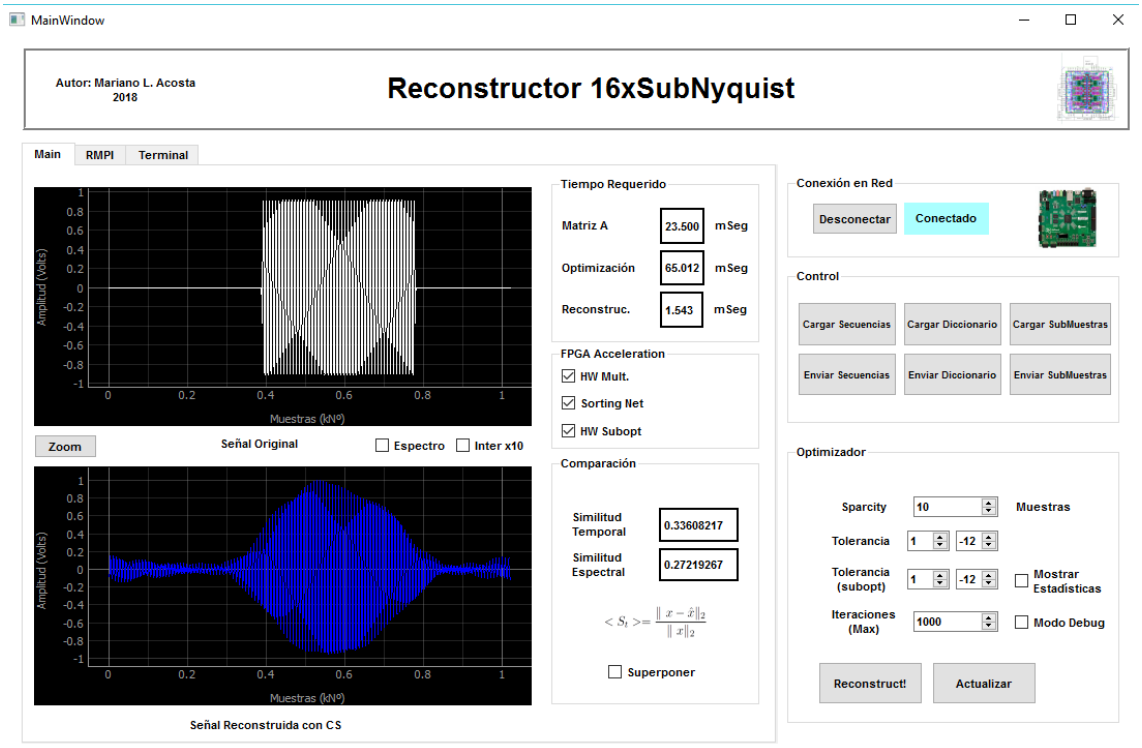
Se cargó un pulso generado digitalmente como muestra la Figura 7.1. En amarillo se encuentra el resultado después del proceso de mezcla e integración que realiza el sensor RMPI. Luego, la ventana MAIN presenta la señal reconstruida por el SoC (en azul) junto con los tiempos requeridos y errores:

- **Tiempo para obtener la Matriz A**: 23,5 mseg.
- **Tiempo para la Optimización**: 65,01 mseg.
- **Tiempo para la Reconstrucción**: 1,543 mseg.
- **Similitud Temporal**: 0,336.
- **Similitud Espectral**: 0,2721.

A su vez, el pulso reconstruido fue visualizado en el osciloscopio a través del DAC fabricado (Figura 7.2). Analizando en el dominio de la frecuencia (Figura 7.3), se



(a) GUI en la ventana RMPI para la señal de radar.



(b) GUI en la ventana MAIN (con resultados) para la señal de radar.

Figura 7.1

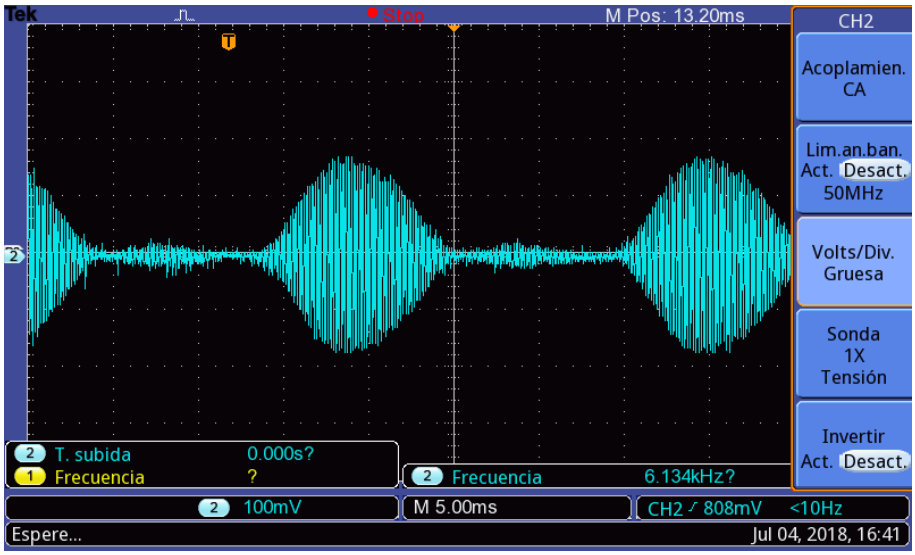
puede observar que el reconstructor acertó con la frecuencia y amplitud de la portadora. Sin embargo, como se evidencia en la forma de onda en el tiempo, la modulante de la señal recuperada no preserva las altas frecuencias. Es decir, la envolvente es más suave. Con respecto a la fase, se puede decir que el sistema es capaz de sincronizarse con la portadora (Figura 7.5).

Para determinar el ancho de banda máximo AB_{max} en que el sistema puede operar en tiempo real, se recurre a la siguiente ecuación:

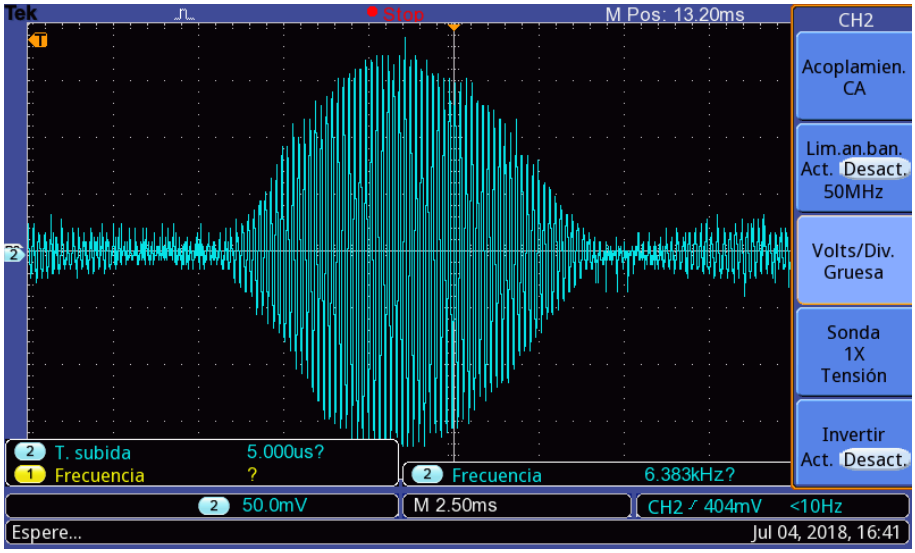
$$AB_{max} = \frac{1024}{2.t_{req}} \tag{7.1}$$

Donde t_{req} es el tiempo requerido en el proceso de optimización más el de reconstrucción. La forma en que se encontró esta relación fue considerando que llega un segmento nuevo de la señal cada 1024 muestras. El sistema en consecuencia debe estar listo para procesar el siguiente bloque. Por supuesto, se debe realizar un análisis exhaustivo, con el fin de conseguir un tiempo promedio. Así, se obtendrá una mejor estimación del ancho de banda máximo. A fines de realizar sólo una estimación, se considera un único caso.

Empleando (7.1) con los tiempos obtenidos, AB_{max} para el caso del radar resulta 7,756Hz. El tiempo para construir A no se tiene en cuenta debido a que, para una aplicación específica, sólo de debe computar una vez. Las secuencias de mezclados a utilizar deben ser siempre las mismas para este caso. Evidentemente, no es posible una reconstrucción en tiempo real si se consideran las frecuencias de operación (3 MHz $\sim$ 110 GHz). No obstante, los pulsos de radar son enviados en promedio cada 2,5mseg. [95]. Por lo que, se dispone del tiempo anterior para la recuperación del pulso. Aun así, el sistema desarrollado resulta inapropiado en esa situación. Se deberá analizar si otros diccionarios ofrecen mejoras al respecto.

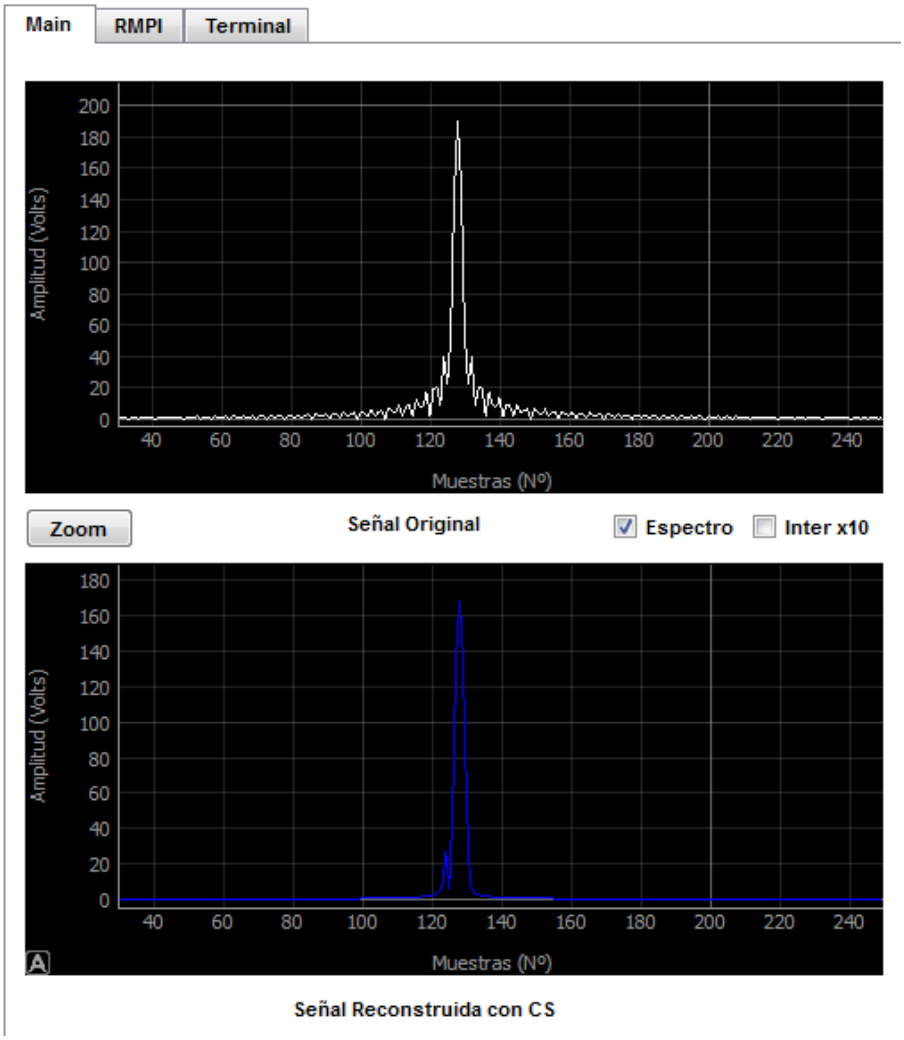


(a) Base de tiempo: 5 mseg por división.

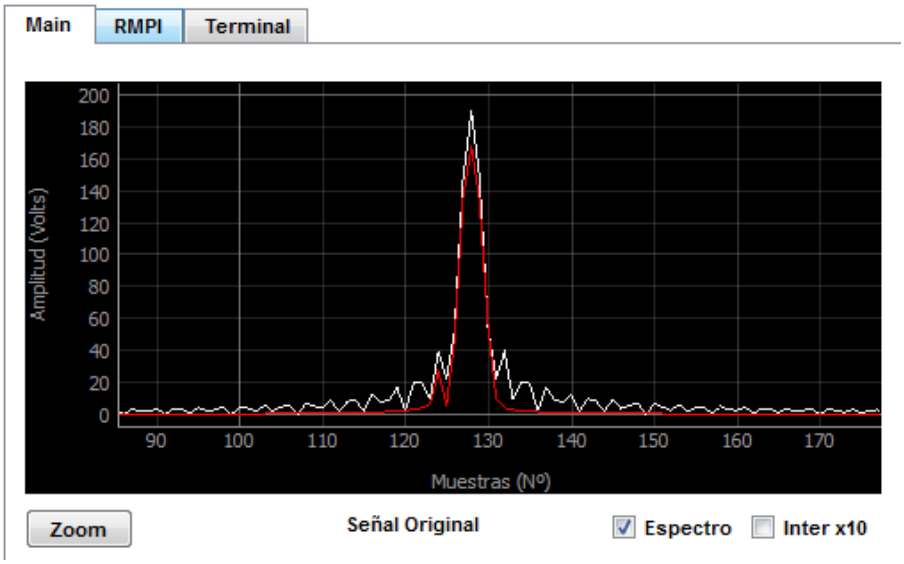


(b) Base de tiempo: 2,5 mseg por división.

Figura 7.2: Señal de radar reconstruida a la salida del conversor D/A.



(a) Función *superponer* desactivada.



(b) Función *superponer* activada.

Figura 7.3: Espectro de la señal de radar vista desde el GUI.

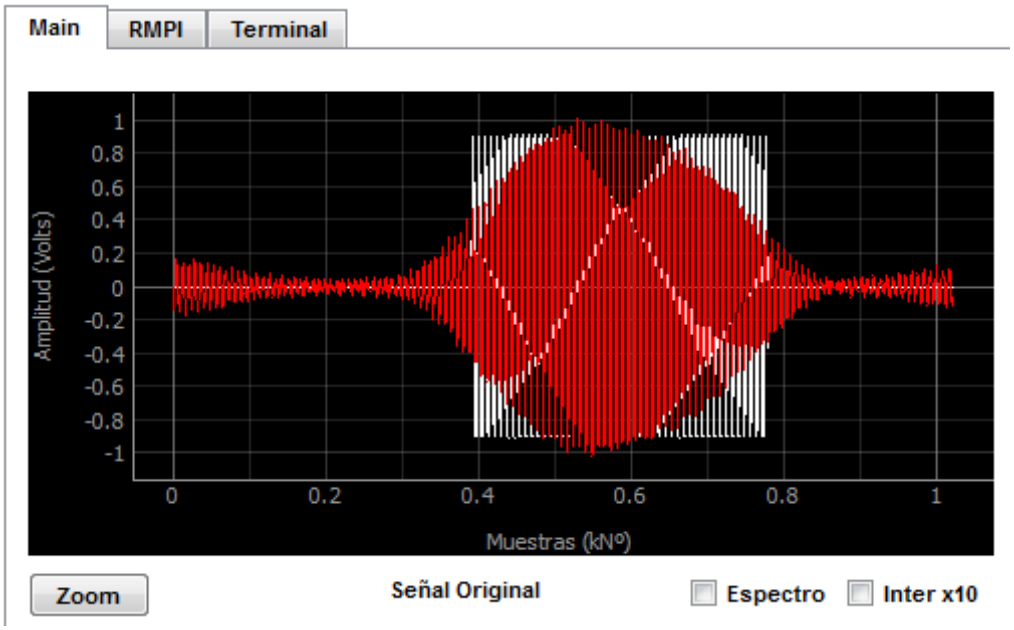


Figura 7.4: Pulso de radar (blanco) superpuesto con su reconstrucción (rojo).

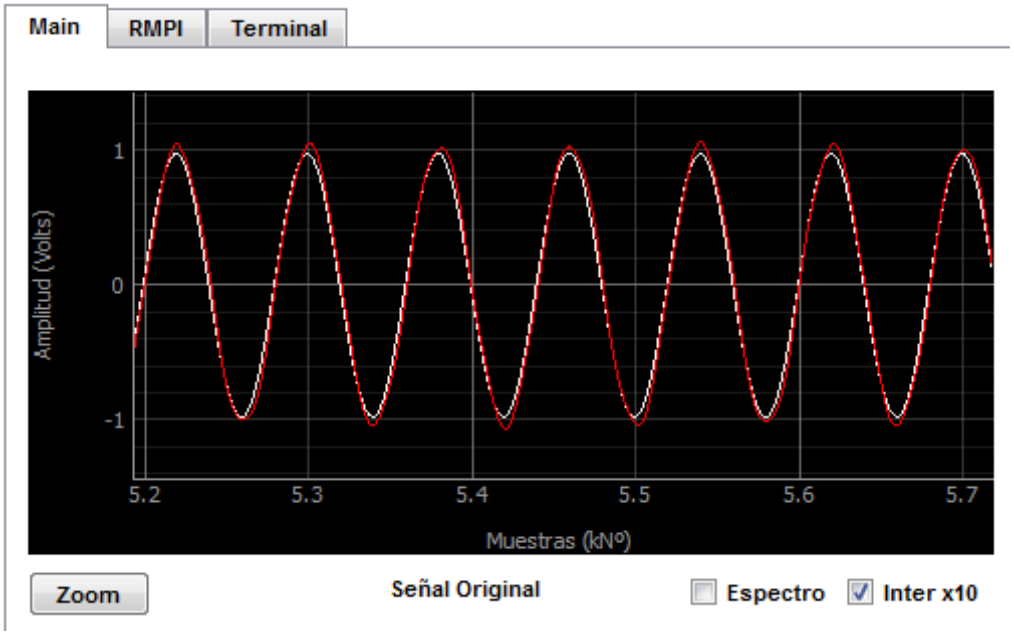


Figura 7.5: Comparación de fase y frecuencia entre señal original (blanco) y recuperada (rojo).

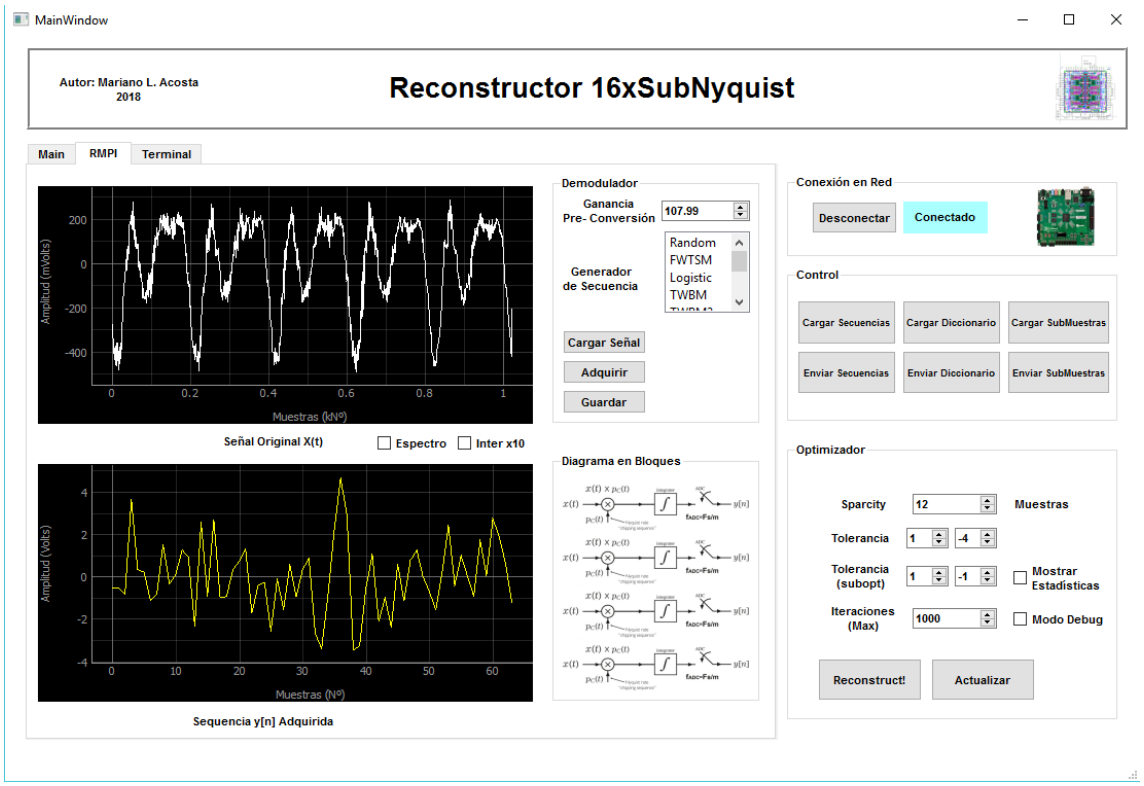
7.2. Señales de Voz

La voz humana consiste en sonido producido por una persona usando el tracto vocal en conjunción con las cuerdas vocales [96]. Es utilizada en la comunicación mediante el lenguaje hablado, en la transmisión de emociones o como instrumento musical; entre otros. Como en el caso anterior, se presenta la captura de pantalla del GUI con un fragmento de voz (Figura 7.6). Se envió dicha señal submuestreada (en amarillo) al sistema y se obtuvieron los siguientes resultados:

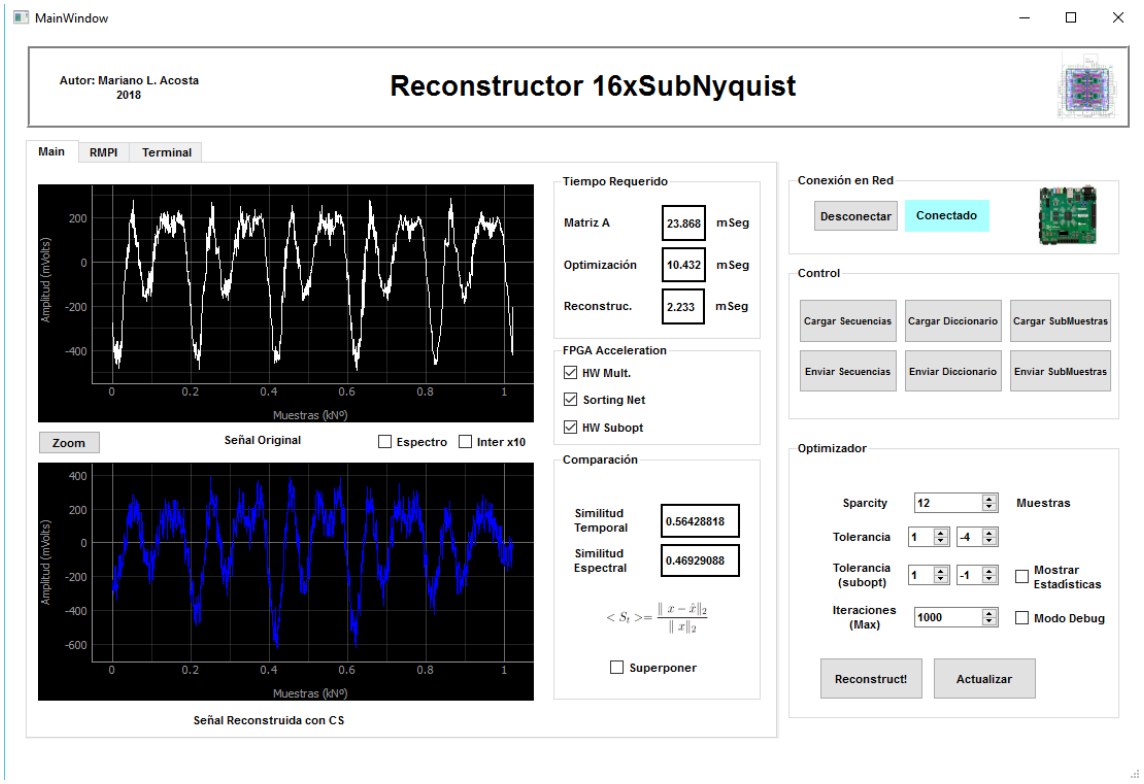
- **Tiempo para obtener la Matriz A**: 23,86 mseg.
- **Tiempo para la Optimización**: 10,432 mseg.
- **Tiempo para la Reconstrucción**: 2,233 mseg.
- **Similitud Temporal**: 0,564.
- **Similitud Espectral**: 0,469.

La voz recuperada se visualizó también en el osciloscopio (Figura 7.7). Además, en la Figura 7.8 se puede apreciar en detalle la forma de onda original (en blanco) con la recuperada (en rojo). Claramente, la versión reconstruida sigue la tendencia de la señal en el tiempo. Si se realiza una comparación en el espectro (Figura 7.9), se ve que fue posible representar fielmente las tres componentes en baja frecuencia. Aunque, en la reconstrucción, aparecieron frecuencias espurias.

Para aplicaciones de tiempo real, el ancho de banda máximo permitido es aproximadamente 40 KHz si nos basamos en (7.1). Típicamente, el contenido espectral relevante del habla se encuentra comprendido entre 300 Hz hasta 4000 Hz. Por lo que, con el sistema implementado, sí es posible realizar adquisición sub-Nyquist en tiempo real. Particularmente, el reconstructor se podría utilizar en detección de formantes, reconocimiento de voz y/o en su transmisión eficiente [97].



(a) GUI en la ventana RMPI para la señal de voz.



(b) GUI en la ventana MAIN para la señal de voz.

Figura 7.6

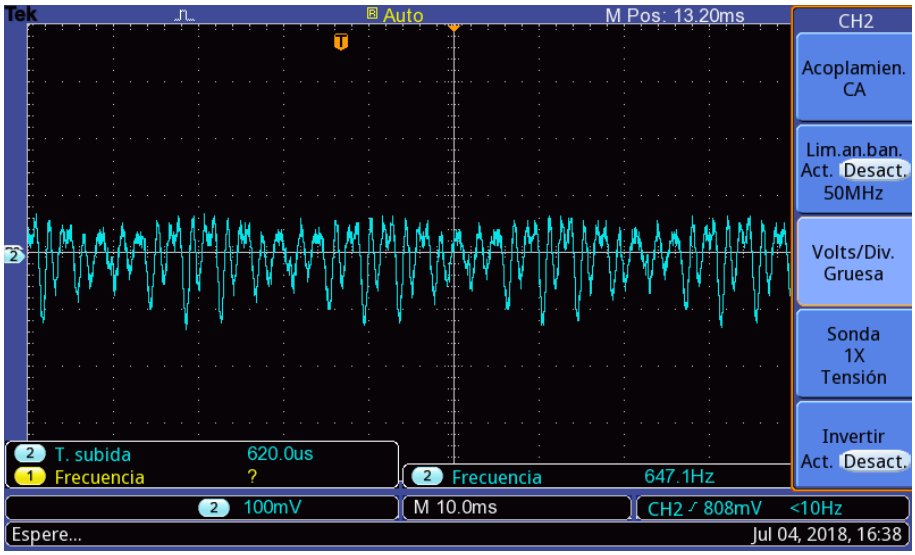


Figura 7.7: Señal de voz reconstruida a la salida del conversor D/A.

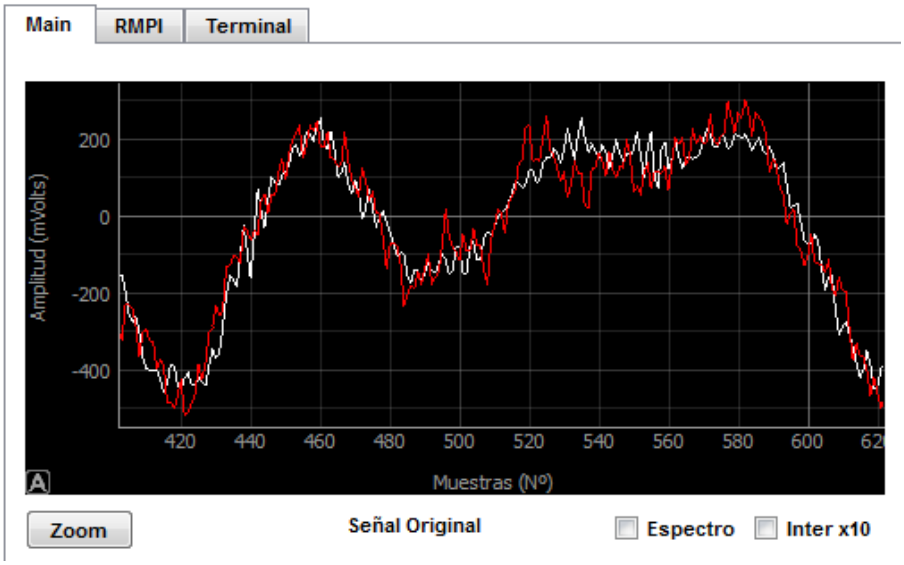
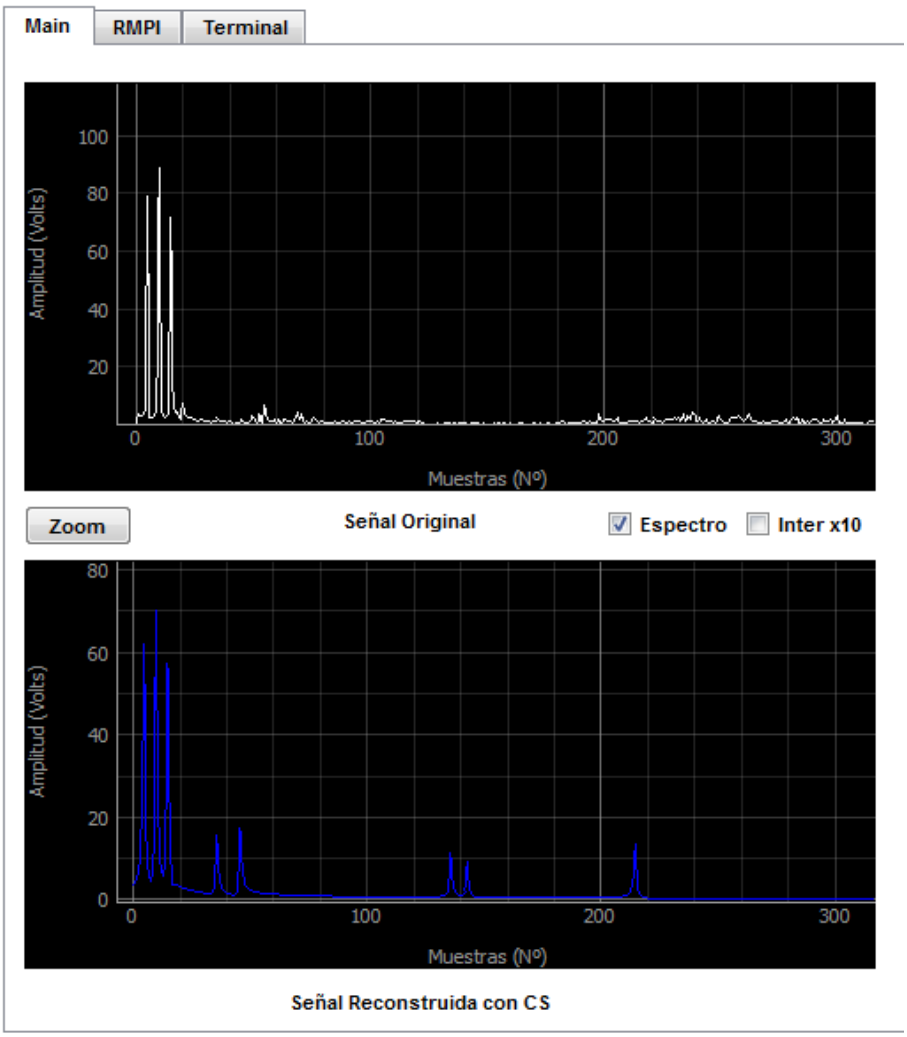
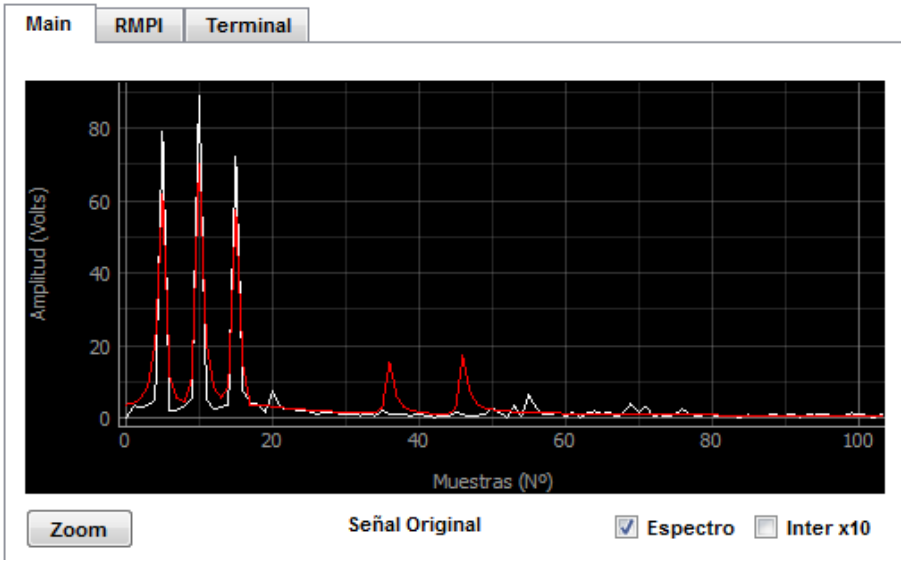


Figura 7.8: Comparación en el tiempo de la señal de voz original (blanco) y recuperada (rojo).



(a) Función *superponer* desactivada.



(b) Función *superponer* activada.

Figura 7.9: Espectro de la señal de voz vista desde el GUI.

7.3. Señales Moduladas en Amplitud

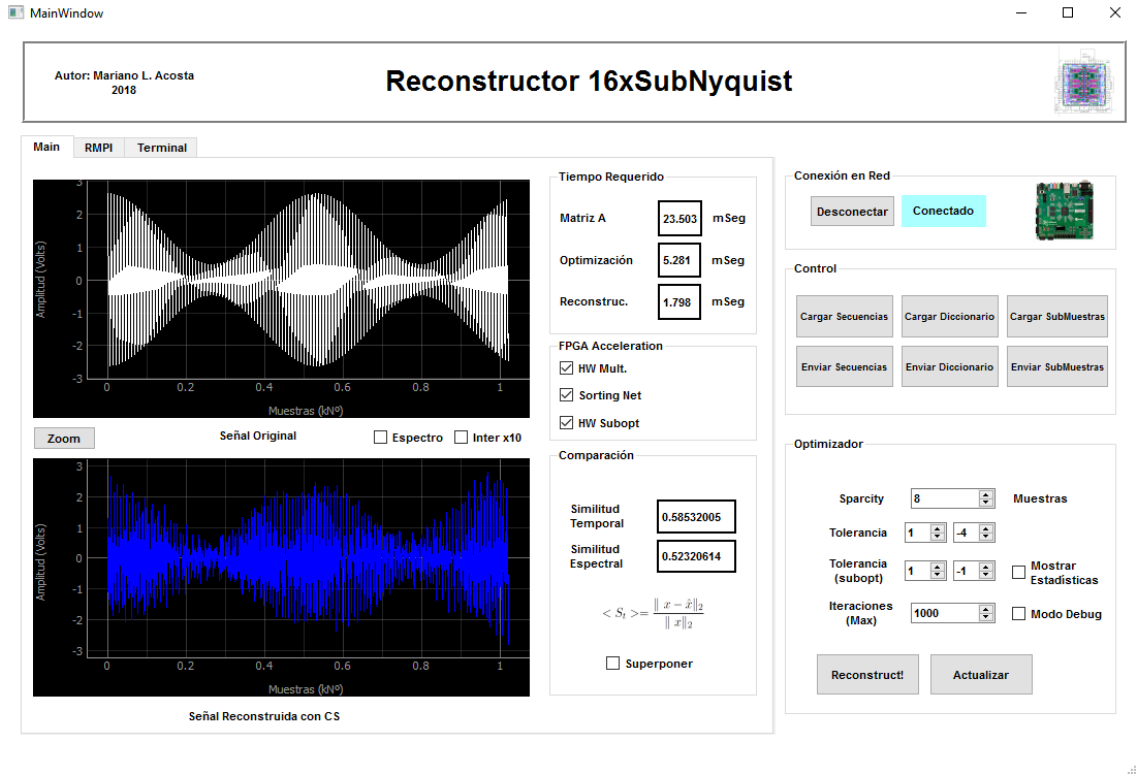
La *modulación por amplitud* (AM) es un método de modulación analógica que utiliza variaciones de amplitud en una portadora de frecuencia fija [98]. Habitualmente, se utiliza en comunicaciones de radio en la cual la señal AM es demodulada por un receptor. La envolvente contiene la información útil transmitida. Se evaluó al reconstructor con la misma señal utilizada en la simulación del Capítulo 3:

$$x(t) = \big[sin(2\pi f_a.t) + sin(2\pi f_b.t)\big](1 + 0{,}7cos(2\pi f_c.t)) \tag{7.2}$$

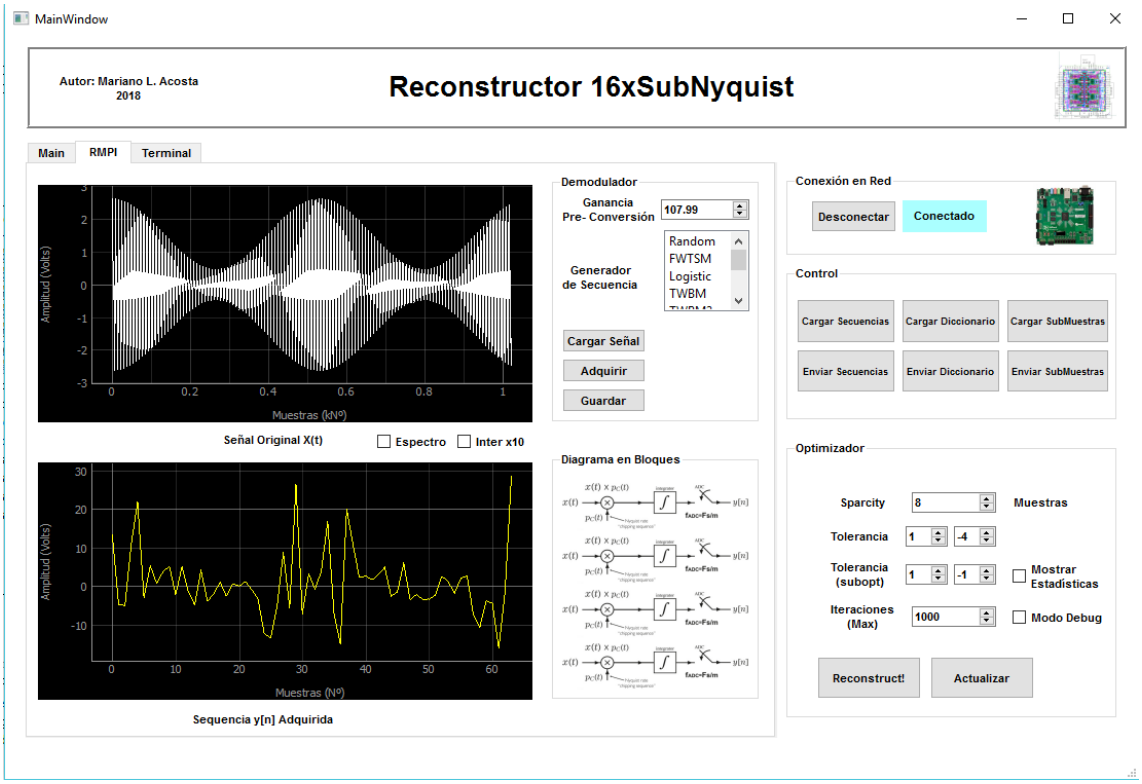
Según se contempla en (7.2), $x(t)$ esta constituida por la suma de dos señales AM; cada una con portadora de frecuencias diferentes: f_a y f_b respectivamente. A su vez, la modulante es una onda sinusoidal con frecuencia f_c, mucho menor que de las portadoras. Con el objetivo de evaluar si el sistema es capás de recuperar una señal de estas características, se procedió a cargar la forma de onda en la interfaz y enviarla al sistema embebido (Figura 7.10). Nótese que la señal en amarillo, de baja frecuencia, es la versión submuestreada de 7.2. El sistema CS respondió de la siguiente manera:

- **Tiempo para obtener la Matriz A**: 23,503 mseg.
- **Tiempo para la Optimización**: 5,281 mseg.
- **Tiempo para la Reconstrucción**: 1,798 mseg.
- **Similitud Temporal**: 0,583.
- **Similitud Espectral**: 0,523.

Usando la herramienta de comparación incluida en el GUI (Figura 7.11), se ve que la forma de onda luego de la reconstrucción (en rojo) se asemeja bastante a la original (en blanco). Aunque, la envolvente recuperada presenta irregulares. Para ver más en detalle por qué ocurría esto último, se analizó el contenido en frecuencia (Figura 7.12). Si bien las dos portadoras fueron identificadas correctamente, el sistema falló en recuperar la componente lateral derecha en ambos casos. Por último, en el osciloscopio se visualizó como muestra la Figura 7.13.

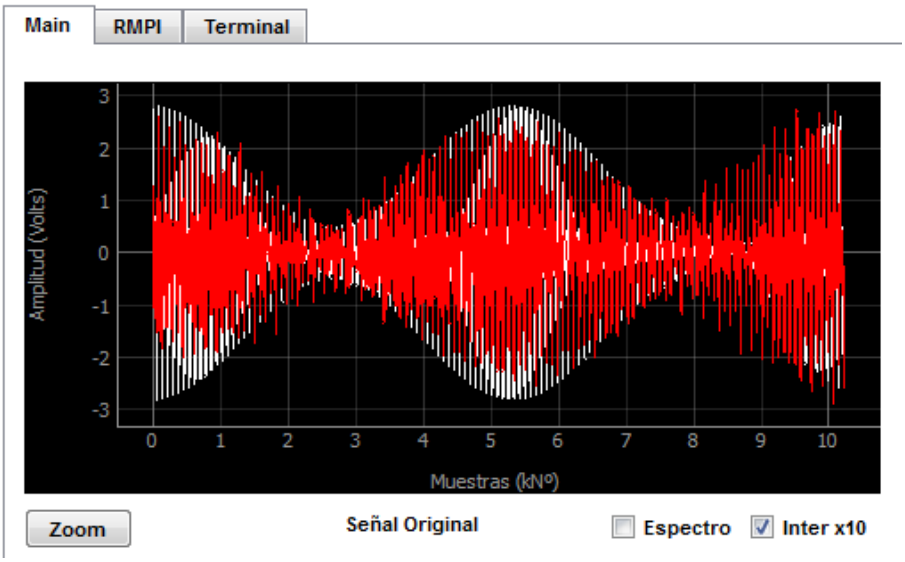


(a) GUI en la ventana RMPI para las señales AM.

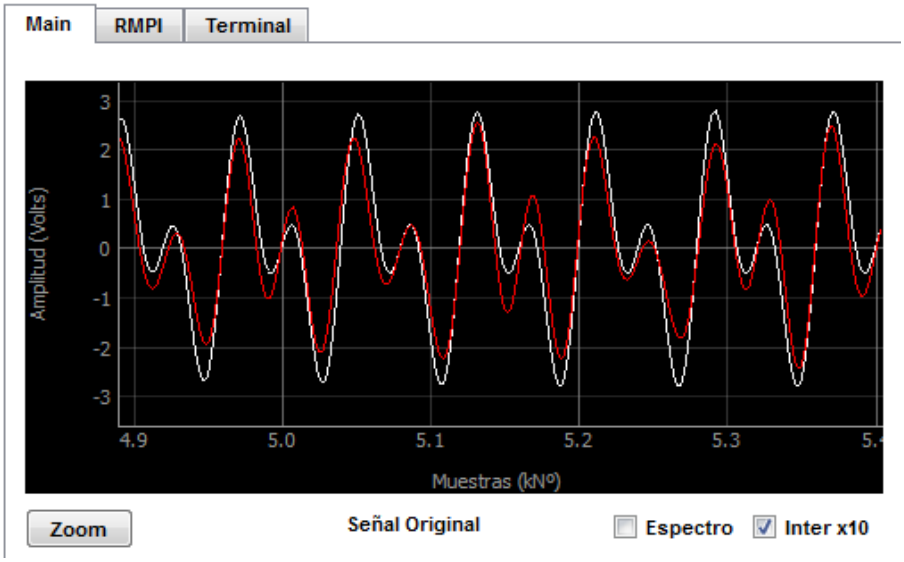


(b) GUI en la ventana MAIN para las señales AM.

Figura 7.10

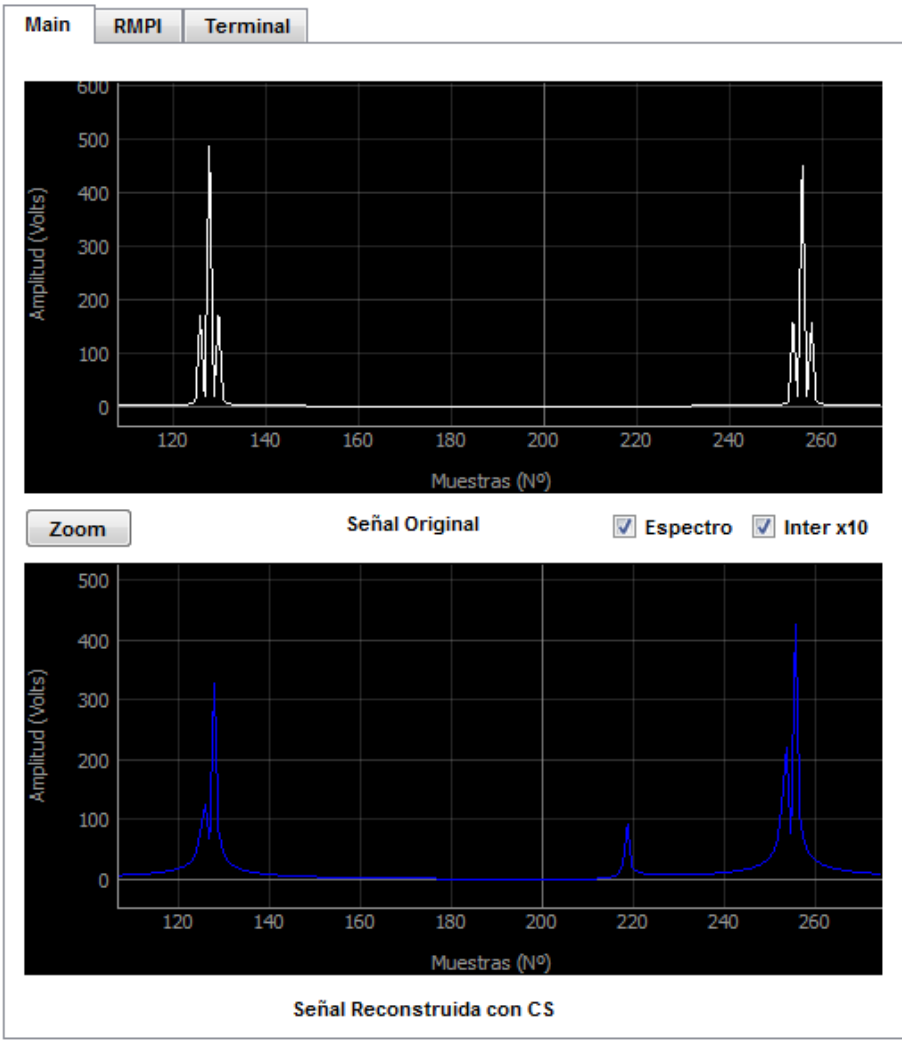


(a) Envolvente (sin Zoom).

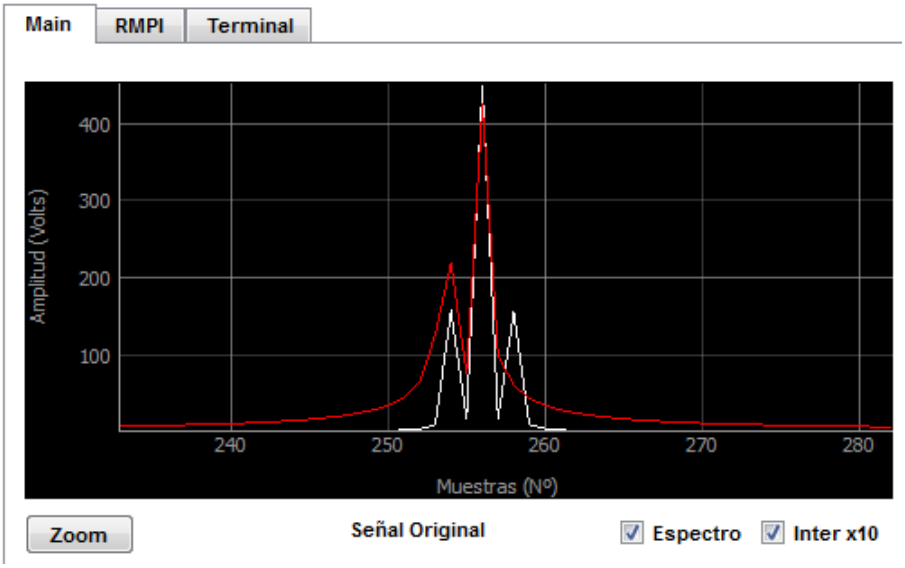


(b) Forma de onda (con Zoom).

Figura 7.11: Comparación en el tiempo de las señales AM originales (blanco) y recuperadas (rojo).

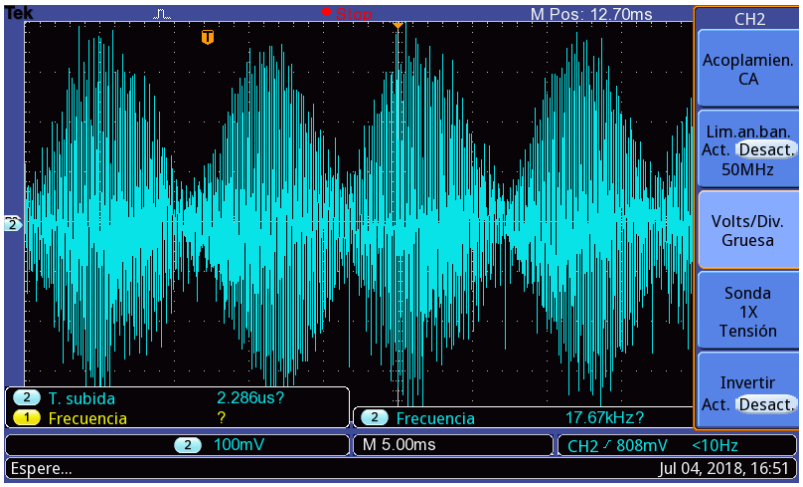


(a) Función *superponer* desactivada.

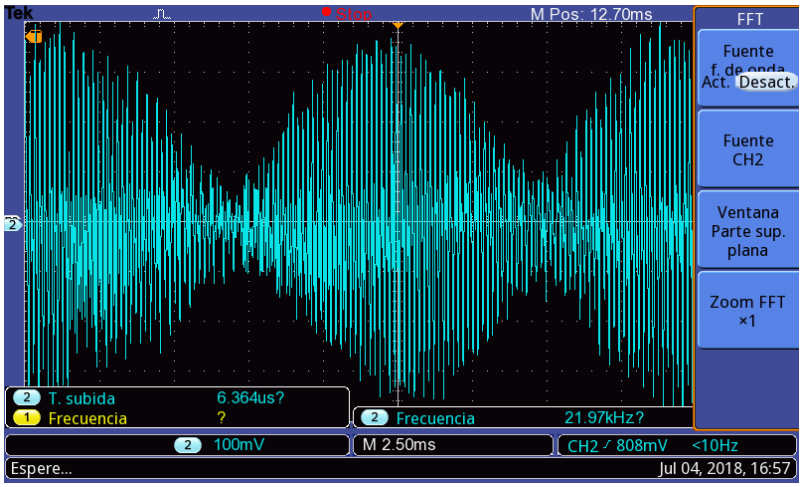


(b) Función *superponer* activada.

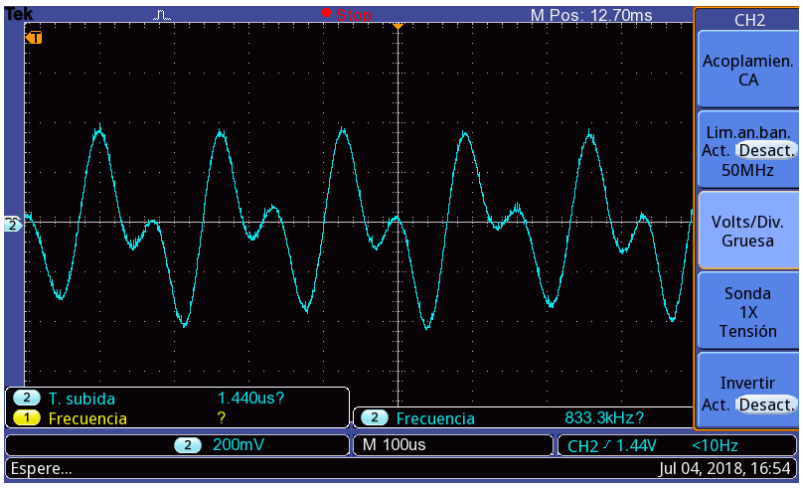
Figura 7.12: Espectro de la señales AM vista desde el GUI



(a) Base de tiempo: 5 mseg por división.



(b) Base de tiempo: 2,5 mseg por división.



(c) Base de tiempo: 0,1 mseg por división.

Figura 7.13: Señales AM reconstruidas a la salida del conversor D/A.

7.4. Comparación entre Hardware y Software

Para finalizar la evaluación, se comparó los tiempos medidos en las aplicaciones para tres casos:

- Ejecución del reconstructor CS sólo en software (ARM A9).
- Ejecución del reconstructor CS en software con los aceleradores en hardware (ARM A9 + FPGA).
- Ejecución del reconstructor CS en una PC (Matlab 2013b).

Se conservaron en todas las instancias las banderas de compilación detalladas en el Capítulo 5 Sección 5.4.3. Las Tablas 7.1, 7.2 y 7.3 muestran los resultados para el pulso de radar, la señal de voz y las señales AM respectivamente. El software utilizado en Matlab es el simulador del Capítulo 3 y se encuentra en el Apéndice. La computadora cuenta con procesador Intel i7 2600k de cuatro núcleos, 8 GB de RAM y disco duro de estado sólido de 256 GB.

	ARM A9	**ARM A9 + FPGA**	**Matlab 2013b**
MatrizA	270,21 mseg	23,5 mseg	2,52 mseg
Optimización	2450 mseg	65,01 mseg	19,83 mseg
Reconstrucción	4,7 mseg	1,543 mseg	0,55 mseg

Cuadro 7.1: Tiempos de ejecución (Radar de Pulso).

	ARM A9	**ARM A9 + FPGA**	**Matlab 2013b**
MatrizA	270,47 mseg	23,86 mseg	2,7 mseg
Optimización	43,89 mseg	10,42 mseg	13,76 mseg
Reconstrucción	4,381 mseg	2,233 mseg	0,57 mseg

Cuadro 7.2: Tiempos de ejecución (Señal de Voz).

	ARM A9	**ARM A9 + FPGA**	**Matlab 2013b**
MatrizA	270,47 mseg	23,50 mseg	2,56 mseg
Optimización	1888,7 mseg	5,28 mseg	11,57 mseg
Reconstrucción	3,24 mseg	1,798 mseg	0,1 mseg

Cuadro 7.3: Tiempos de ejecución (Señales AM).

En principio, la mejora en rendimiento es evidente si se equipara los tiempos del sistema haciendo uso del FPGA que utilizando únicamente el microprocesador. A continuación, se muestra la reducción en veces del tiempo necesario para cada operación.

- **Obtener A**: 11,5x
- **Optimización (Radar)**: 37,68x
- **Optimización (Voz)**: 4,21x
- **Optimización (AM)**: 357,7x
- **Reconstrucción**: 1,96x

Comparando ahora el reconstructor con la misma aplicación en Matlab:

- **Obtener A**: 0,11x
- **Optimización (Radar)**: 0,30x
- **Optimización (Voz)**: 1,32x
- **Optimización (AM)**: 2,19x
- **Reconstrucción**: 0,05x

Inicialmente se puede divisar que el multiplicador matricial ofrece siempre una mejora de 11,5 veces. No obstante, la ventaja en usar la red de ordenamiento depende del tipo de aplicación. Para la modulación de amplitud es 357,7 veces, en tanto para la voz 4,21 veces. La diferencia se debe en que el optimizador necesita ordenar vectores una mayor cantidad de veces en un caso que en el otro. En constraste, Matlab realiza más eficientemente las multiplicaciones que el sistema embebido. A pesar de ello, en la optimización el tiempo es comparable e incluso en ocasiones superior en la placa de desarrollo.

7.5. Discusión sobre los Resultados Obtenidos

De las tres aplicaciones bajo prueba, la recuperación de la señal de voz fue la que presentó el mejor rendimiento en términos de calidad de reconstrucción y potencial aplicación en tiempo real. A pesar de ello, hay que considerar que el diccionario utilizado fue la DCT. Es bien conocido la compresibilidad de la voz en ese dominio [99]. En las instancias de simulación, se encontró que la FFT en el caso de las señales AM recupera correctamente las frecuencias de portadora junto con sus lóbulos laterales. Por ello, una mejora futura sería incorporar en el SoC bloques que operen con números complejos. También, se recomiendan evaluar más aplicaciones con los siguientes diccionarios:

- Diccionarios Wavelets (números reales): entre ellos Haar y Daubechies. La forma de construirlos se encuentra en [100]. Presenta aplicabilidad señales de electroencefalograma (EEG) [101] y del corazón (EGC) [29].
- Diccionario en el dominio del tiempo (números reales): Consiste en la matriz identidad. Particularmente útil si la señal es rala en el tiempo.

- Diccionario de Fourier (números complejos): Conservan con exactitud las componentes frecuenciales y su relación de fase.
- Diccionario de Gauss (números reales): Cada columna esta compuesta por pulsos gaussianos [102].

El sistema creado en este proyecto es universal, es decir, admite la posibilidad de sensar toda señal unidimensional con cualquier base de representación. Por ese motivo, es posible que uno de los diccionarios mencionados anteriormente resulte más adecuado para el pulso de radar, por ejemplo. En esa situación, el tiempo para la reconstrucción podría verse reducido y entonces sería posible el procesamiento continuo en tiempo real.

Bajo el entorno de Matlab, las multiplicaciones fueron más eficiente debido a que hace uso de la librería optimizada MKL (*Math Kernel Library* en Inglés) [103]. Aunque, quedó justificado mediante las tablas mostradas en la Sección 7.4 que existe una reducción de tiempo en el proceso de reconstrucción en el SoC. Si se omite la construcción de la matriz A y se la carga previamente en la memoria del chip, se lograría una aplicación embebida superior en comparación con una computadora de propósito general.

Capítulo 8

Conclusiones

El proyecto de grado presentado consistió en el estudio de la teoría del Sensado Compresivo para la realización de un sistema de reconstrucción en el SoC ZYNQ 7000. Fue posible recuperar de manera exitosa señales que fueron submuestreadas hasta 16 veces por debajo de su frecuencia de muestreo convencional. Para ello, se hicieron uso de las herramientas que provee Xilinx para el desarrollo de sistemas embebidos. A su vez, se propuso realizar un co-diseño entre hardware y software para lograr la reducción del tiempo total de procesamiento. Una de las hipótesis planteadas fue que las secuencias caóticas resultan adecuadas para este tipo de aplicación y pueden ser usadas en lugar de generadores aleatorios. La otra, fue analizar la factibilidad de realizar la reconstrucción en tiempo real. Por otra parte, fue necesario diseñar e implementar un conversor digital/analógico para que la placa de desarrollo pudiese generar señales de voltajes con las muestras generadas con CS.

Las pruebas realizadas en el Capítulo 7 arrojaron que el tiempo de procesamiento se reduce considerablemente usando lógica programable. Si bien el tiempo requerido en el SoC fue similar al obtenido con Matlab, se debe tener en cuenta que la capacidad de procesamiento de la PC usada en el proyecto es superior a la placa de desarrollo. Se refutó la posibilidad de realizar reconstrucciones en tiempo real en radiofrecuencia con el sistema realizado, aunque sí es factible en aplicaciones de voz. En relación a la segunda hipótesis, fue validado el uso de secuencias caóticas en CS, en especial con la señal binaria de mezclado en arquitectura RMPI. Esto implica que es factible reemplazar los generadores pseudo-aleatorios por mapas caóticos iterativos y en consecuencia lograr una reducción en la utilización de recursos en HW. En particular, el mapa logístico fue el único que mostró resultados similares, e incluso superiores, al caso pseudo-aleatorio; en términos de minimización del error en la señal reconstruida. Con los demás generadores, se obtuvo una peor performance.

Conocer en detalle la plataforma de desarrollo permitió aprovechar completamente los recursos del sistema embebido, desde el acceso eficiente a memoria hasta la configuración de la unidad de punto flotante para acelerar las operaciones matemáticas. Respecto a la lógica programable, como se mostró en el desarrollo, los bloques DSP están diseñados para operar con acumuladores y sumadores de longitud de bits predeterminada. Con este conocimiento en mente, se logró reducir la utilización de

área en el chip evitando el uso de elementos innecesarios. Asimismo, la *Síntesis de Alto Nivel* (HLS) de Xilinx facilitó la creación de IP complejos de manera sencilla gracias al uso de programación en C. Particularmente, se redujo el tiempo de diseño en aquellos IP que requerían integrar una interfaz AXI. También, resulta sustancial conocer el hardware para aprovechar eficazmente las prestaciones que otorga HLS mediante el uso de pragmas como *Pipeline*, *Dataflow*, *Unroll*, *Reshape*, etc. Sin embargo, cuando se precisó un uso más eficiente del *timing* y/o probar una arquitectura en hardware de manera detallada, se hizo uso de la programación en VHDL. Por lo que, si bien HLS es útil en algunas situaciones, no es un reemplazo de VHDL.

A continuación se discuten las futuras líneas de desarrollo y sugerencias que pueden derivarse de este trabajo:

Evaluar otros algoritmos de reconstrucción: Se recomienda implementar CoSaMP ya que en la etapa de simulación mostró resultados similares al AIHT. Especialmente, mejorar el paralelismo y aprovechar aún más los recursos del FPGA. Para ello, se debe primero realizar el algoritmo únicamente en software. Luego, mediante *profiling* [104] determinar que operaciones demandan más tiempo y por lo tanto serían beneficiadas en con una implementación en lógica programable.

Fabricación del Sensor RMPI: Comandar la adquisición mediante el FPGA, enviando las señales de sincronismo y mezclado a través de los pines de entrada/-salida. También, se precisa calibrar el RMPI. Puesto que esta tarea es compleja, abarcaría la totalidad de un proyecto de grado.

Desarrollo para una aplicación específica: En este trabajo se demostró la potencialidad de uso de CS en diversas situaciones. Si se comienza el diseño con una aplicación en mente (ej: señales biomédicas, de voz, radiofrecuencia, etc.) es posible mejorar aún más la eficiencia, utilizando menos recursos de la placa y disminuyendo el tiempo de procesamiento. Inclusive, se podría prescindir de la construcción de la matriz A; porque el diccionario junto con las secuencias de mezclado estarían predeterminadas. Es necesario previamente efectuar un análisis de costos, factibilidad y procurar que esté justificado el uso del SoC, en especial si la aplicación es embebida y se generan beneficios en términos de consumo de potencia.

Utilizar un mejor modelo del reconstructor: Según lo expuesto en la teoría del sensor RMPI, se asumió un integrador ideal. A pesar de ello, se sabe que en la práctica no es posible lograr un circuito electrónico que realice perfectamente esta función. Por eso, se debe adoptar un filtro pasabajos con respuesta al impulso $h(t)$ en lugar del integrador. En consecuencia, la matriz de sensado se construirá de manera diferente a lo presentado en este proyecto. Aunque, al tener un modelo más exacto del filtro, la reconstrucción será mejor. Los trabajos [20] y [15] indagan en esta cuestión.

Automatizar los parámetros del algoritmo de reconstrucción: Como se apreció a lo largo del desarrollo, cada vez que se procede a reconstruir una señal

se ingresan parámetros de configuración en la interfaz gráfica. Generalmente, este procedimiento se hace con conocimiento previo o por simple prueba y error. Una forma de solucionar este inconveniente sería, por ejemplo, mediante un algoritmo genético u otro algoritmo evolutivo encontrar los parámetros óptimos del reconstructor para un caso particular. Conjuntamente, cuando se desconoce a priori el grado de dispersión k, se podría usar la técnica presentada en [105] para su estimación.

Utilizar una relación de compresión menor: Los resultados arrojados en la simulaciones muestran que a medida que disminuye la cantidad de muestras adquiridas en el proceso de submuestreo, la confiabilidad de la aplicación disminuye. Se recomienda utilizar una relación de compresión igual a ocho para mejorar tanto la probabilidad de éxito como la calidad en la reconstrucción. Además, como se justificó en el Capítulo 3, con una relación menor se podría usar una ventana de observación (longitud de $x[n]$) igual a 512. En consecuencia, la exigencia de memoria para el sistema se vería disminuida.

Implementar operaciones con números complejos: Existe una relación de compromiso entre utilizar números complejos y la complejidad resultante del sistema. Pero, se debe evaluar esta alternativa cuando se obtenga una ventaja en usar diccionarios como la FFT; en especial si se requiere representar con exactitud la frecuencia y fase de las componentes de una señal.

Considerar otras formas de onda para las secuencias de mezclado: Un detalle a tener en cuenta es que las secuencias de mezclado deben conmutar a por lo menos a la frecuencia de Nyquist. En todo el proyecto se usó una forma de onda cuadrada para las mismas. Si en fabricación del RMPI se está limitado en banda, entonces los armónicos de las secuencias serán atenuados. Se deberá analizar que sucede en esos casos.

Construcción del Generador Logístico en Hardware: Aprovechar la lógica programable para generar función iterativa del mapa caótico. Luego, enviar las señales producidas al sensor físico.

Apéndice A

Apéndice

A.1. Códigos

A.1.1. Matlab

Simulación RPMI y CS (Comparación de Optimizadores)

```
clear all
clc
Resultados = zeros(7,200,3) %algoritmo/iteracion/cuantificadores
for opp =7:7

    %%Par?metros de la Simulaci?n

    n=1024;
    factor_reduccion = 32;
    numCh = 1;
    Af=12;
    %%

    W = 4e9; %frecuencia de muestreo normal
    fs = W*256; %frecuencia de muestreo de la simulacion
    R = W/factor_reduccion; %frecuencia de submuestreo
    fc= 1e9;
    t=[0:1/fs:1/fs*(n*256-1)];

    %%funciones
    f = sin(2*pi*fc/2*t).*(1+.7*cos(2*pi*3e8/40*t))
    +sin(2*pi*fc*t).*(1+.7*cos(2*pi*3e8/40*t));

    %%
    f_int = zeros(size(f));

    f_int=intdump(f,256);

    f=(f+.0*rand(size(f)));

    ff=f(1:fs/W:fs/W*n);

    %%Diccionario
    Dic = fft(eye(n));

    %% Par metros del Optimizador

    sparcity=2;
    alfa=0.9;

    %%Respuesta al Impulso
    wr = ones(1,W/R*numCh);
    Hr = kron(eye(n*R/W/numCh),wr);

for jjk=1:50
    %%Generacion de las secuencias aleatoreas
    p = zeros(n,numCh);
    pr = zeros(n*fs/W,numCh);
    P = zeros(n,n,numCh);

    for i=1:numCh

        p(:,i) = ((rand(1,n) > 0.5)*2 - 1);
        P(:,:,i) =diag(p(:,i));
        pr(:,i) = reshape(repmat(p(:,i)',fs/W,1),[1,fs/W*n])';
        %pr(:,i) = pr(:,i).*abs(sin(pi*W*t))';
    end
```

```
    pr=pr';

    %%Matriz A

    HPr = zeros(R/W*n/numCh,n,numCh);

    for i=1:numCh

        HPr(:,:,i)=Hr*P(:,:,i);
    end

    HP =[];

    for i=1:R/W*n/numCh
        for j=1:numCh
            HP=[HP;HPr(i,:,j)];
        end
    end

    A =HP*Dic;

    %%Random Demodulation

    yy=[];

    for i=1:numCh
        b=pr(i,:).*f;
        yy = [yy ;intdump(b,fs/R*numCh)*factor_reduccion*numCh];
    end

    y=[];
    for i=1:R/W*n/numCh
        for j=1:numCh
            y=[y yy(j,i)];
        end
    end

    y=y';

    %%Reconstruction

    switch opp
        case 1
            tic
            xp = Homotopy(A,y,10e9,Af,0.95);
            ttime=toc;
        case 2
             tic
            xp=greed_gp(y,A,n,'stopCrit','M','stopTol',Af);
            ttime=toc;
        case 3
             tic
            xp=greed_pcgp(y,A,n,'stopCrit','M','stopTol',Af);
            ttime=toc;
        case 4
             tic
            xp=hard_l0_Mterm(y,A,n,Af);
            ttime=toc;
        case 5
             tic
            xp=greed_omp(y,A,n);
            ttime=toc;
        case 6
             tic
            xp=CoSaMP( A, y, Af );
            ttime=toc;
        case 7
             tic
            xp = optimizer(y,A,n,Af,10e-8); %AIHT
            ttime=toc;
    end

    x = real(fft(xp));
    ff=f_int;

    ff=resampleSINC(ff, 10);
    x=resampleSINC(x', 10);


    %%Similarity

    Similarity = norm(ff-x,2)/norm(ff,2);
    Espectral_Similarity = norm(abs(fft(ff))
                           -abs(fft(x)),2)/norm(abs(fft(ff)),2);
    Resultados(opp,jjk,1) = Similarity;
    Resultados(opp,jjk,2) = Espectral_Similarity;
    Resultados(opp,jjk,3) = ttime;
    end
end
```

Simulación RPMI y CS (Generadores Caóticos)

```
clear all
clc
Resultados_SECUENCIAS = zeros(7,50,3);
%algoritmo/iteracion/cuantificadores
noise=0;
for opp =1:7

    %%Par?metros de la Simulaci?n

    n=1024;
    factor_reduccion = 16;
    numCh = 4;
    Af=12;
    %%

    W = 4e9; %frecuencia de muestreo normal
    fs = W*256; %frecuencia de muestreo de la simulacion
    R = W/factor_reduccion; %frecuencia de submuestreo
    fc= 1e9;
    t=[0:1/fs:1/fs*(n*256-1)];

    %%funciones
    f = sin(2*pi*fc/2*t).*(1+.7*cos(2*pi*3e8/40*t))
      +sin(2*pi*fc*t).*(1+.7*cos(2*pi*3e8/40*t));

    %%
    f_int = zeros(size(f));

    f_int=intdump(f,256);

    f=(f+noise*rand(size(f)));

    ff=f(1:fs/W:fs/W*n);

    %%Diccionario
    Dic = fft(eye(n));

    %% Par?metros del Optimizador

    sparcity=2;
    alfa=0.9;

    %%Respuesta al Impulso
    wr = ones(1,W/R*numCh);
    Hr = kron(eye(n*R/W/numCh),wr);

for jjk=1:50
    %%Generacion de las secuencias aleatoreas
    p = zeros(n,numCh);
    pr = zeros(n*fs/W,numCh);
    P = zeros(n,n,numCh);

    for i=1:numCh
        switch opp
            case 1
                p(:,i) = ((rand(1,n) > 0.5)*2 - 1);
            case 2
                p(:,i) = ((FWTSM(1,n) > 0.5)*2 - 1);
            case 3
                p(:,i) = ((TWBM1(1,n) > 0.5)*2 - 1);
            case 4
                p(:,i) = ((logistico(1,n) > 0.5)*2 - 1);
            case 5
                p(:,i) = ((TWBM21(1,n) > 0.5)*2 - 1);
            case 6
                p(:,i) = ((TWBM41(1,n) > 0.5)*2 - 1);
            case 7
                p(:,i) = ((TWTSM1(1,n) > 0.5)*2 - 1);
        end
        P(:,:,i) =diag(p(:,i));
        pr(:,i) = reshape(repmat(p(:,i)',fs/W,1),[1,fs/W*n])';
        %pr(:,i) = pr(:,i).*abs(sin(pi*W*t))';
    end

    pr=pr';


    %%Matriz A

    HPr = zeros(R/W*n/numCh,n,numCh);


    for i=1:numCh

        HPr(:,:,i)=Hr*P(:,:,i);
    end

    HP =[];

    for i=1:R/W*n/numCh
        for j=1:numCh
            HP=[HP;HPr(i,:,j)];
```

```
        end
    end

    A =HP*Dic;

    %%Random Demodulation

    yy=[];

    for i=1:numCh
        b=pr(i,:).*f;
        yy = [yy ;intdump(b,fs/R*numCh)*factor_reduccion*numCh];
    end

    y=[];
    for i=1:R/W*n/numCh
        for j=1:numCh
            y=[y yy(j,i)];
        end
    end


    y=y';

    %%Reconstruction

    tic
    xp = optimizer(y,A,n,Af,10e-8);
    ttime=toc;
    x = real(fft(xp));
    ff=f_int;
    ff=resampleSINC(ff, 10);
    x=resampleSINC(x', 10);

    %%Similarity

    Similarity = norm(ff-x,2)/norm(ff,2);
    Espectral_Similarity = norm(abs(fft(ff))
                           -abs(fft(x)),2)/norm(abs(fft(ff)),2);
    Resultados_SECUENCIAS(opp,jjk,1) = Similarity;
    Resultados_SECUENCIAS(opp,jjk,2) = Espectral_Similarity;
    Resultados_SECUENCIAS(opp,jjk,3) = ttime;

    end
end
```

Filtro Sallen-Key de Segundo Orden (Bessel)

```
clc,clear all,close all
%%parametros del circuito
s=tf('s')
ra1=2.5e3
rb1=8.1e3
ca1=1e-9
cb1=1.5e-9
s= tf(s)

%%op amp-------------------------
zo=27;
fc=20;
gain_dc=10^(100/20);

%%

g1=1/(s*ca1*ra1+(s*cb1*ra1+1)*(s*ca1*rb1+1));
g2=s*cb1*ra1/(s*ca1*ra1+(s*cb1*ra1+1)*(s*ca1*rb1+1));
g3=(s*ca1*rb1*ra1+ra1)/(zo*(s*ca1*(ra1+rb1)+1)+s*ca1*rb1*ra1+ra1);

A=gain_dc/(s/2/pi/fc+1);

zt= 1/(1/ra1+s*ca1/(s*ca1*rb1+1))
g4=((s*ca1*rb1+1)*s*cb1*zo)/(s*cb1*ra1*(s*ca1*rb1+1)
  +(s*cb1*zo+1)*s*ca1*(rb1+ra1)+1);

gh1=A*g3;

tlc1=A*g3/(1+gh1);

gh2=tlc1*g2;

tlc2=g1*tlc1/(1-tlc1*g2)

bode(tlc2),grid minor,

%%ahora agregando el efecto feedforward

tlc1=A*g3/(1+gh1);

gh2=tlc1*g2;

tlc2=g1*tlc1/(1-tlc1*g2)+g4/(1+A*g3);

w=linspace(1000,10^9,100000);
```

```
options = bodeoptions;
options.FreqUnits = 'Hz'; %
hold on
bode(tlc2,w,options),grid minor,
```

A.1.2. C++/High Level Synthesis

Multiplicador Matricial (A_GENERATOR)

```
#include "H:\Tesis\ProgramasHLS\Includes\Xampler.h"


/*Descripci n:
 *
 *
 * Cuando mode es 1 trabaja para la construccion de la matriz A.
 *
 * Se ales de Control:
 *
 ** mode: switchea entre modos de operacion (Construcci n A / Optimizaci n)
 *
 */

void A_GENERATOR(stream Aa[NumOfCh*M],stream Bb[M*M], stream Cc[NumOfCh*M],ap_int<1> mode){
        #pragma HLS INTERFACE axis register both port=Cc name=matrixC
        #pragma HLS INTERFACE axis register both port=Bb name=matrixB
        #pragma HLS INTERFACE axis register both port=Aa name=matrixA
        #pragma HLS DATAFLOW
//      #pragma HLS ARRAY_RESHAPE variable=C block factor=4 dim=2
//      #pragma HLS ARRAY_MAP variable=C horizontal

        int NumOfCol;
        float A[NumOfCh][M];
        #pragma HLS ARRAY_MAP variable=A horizontal
        #pragma HLS ARRAY_RESHAPE variable=A block factor=4 dim=2
        float B[M][M];
        #pragma HLS ARRAY_MAP variable=B horizontal
        #pragma HLS ARRAY_RESHAPE variable=B block factor=4 dim=1

        int temp;
        bool temp_last;
        bool temp_last_B;
        bool temp_last_C=0;



        NumOfCol = NumOfCh;

        //read data

        read_A_out:for(int i=0;i<NumOfCol;i++){

                read_A_in:for(int j=0;j<M;j++){
                #pragma HLS PIPELINE
                        temp=i*M+j;
                        A[i][j]=Aa[temp].data;
                        temp_last =Aa[temp].last;
                }
        }

        read_B_out:for(int i=0;i<M;i++){

                        read_B_in:for(int j=0;j<M;j++){
                        #pragma HLS PIPELINE
                                temp = i*M+j;
                                B[i][j]=Bb[temp].data;
                                temp_last_B =Bb[temp].last;
                        }
                }

        //multiplicaciones
        L1:for (int ia = 0; ia < NumOfCol; ++ia)
        {

                L2:for (int ib = 0; ib < M; ++ib)
                {
                        #pragma HLS PIPELINE
                        float sum = 0;

                        L3:for (int id = 0; id < M; ++id)
                        {
                                sum += A[ia][id] * B[id][ib];
                        }

                        if((ia*M+ib) == (NumOfCol*M-1)){
                                temp_last_C=1;
                        }
                        Cc[ia*M+ib].data=sum;
                        Cc[ia*M+ib].last=temp_last_C;
                }
```

```
        }

}
```

Xampler.h

```
#ifndef XAMPLER_H_
#define XAMPLER_H_

#define N 1024
#define M 64
#define NumOfCh 4
#define CONSTRUCTOR 0b1
#define OPTIMIZER 0b0

#include "ap_int.h"
#include "ap_axi_sdata.h"

struct stream{
  float data;
  bool last;
};

void mult_64_64(stream Aa[NumOfCh*M],stream Bb[M*M], stream Cc[NumOfCh*M],ap_int<1> mode);
void read_A(stream Aa[NumOfCh*M],float A[NumOfCh][M],int NumOfCol);
void read_B(stream Bb[NumOfCh*M],float B[NumOfCh][M]);
#endif
```

Red de Ordenamiento (HARD_THRESH)

```
#include "ap_int.h"
#define NUM_ELEMENT 128
struct  celda{
        ap_uint<32> value;
        ap_uint<10> ind;
};

struct dataOut{
        ap_uint<32> data;
        bool last;
};

struct stream{
  ap_uint<32> data;
  bool last;
};


void Comp(celda A,celda B,celda* min,celda* max);

void Comp(celda A,celda B,celda* min,celda* max){

        celda max_d,min_d;

        iff:if (A.value>B.value){
                max_d=A;
                min_d=B;
        }else{
                max_d=B;
                min_d=A;
        }

        *max=max_d;
        *min=min_d;
}
//envio 32 numeros uint32, recibo 32 indices

void HARD_THRESH (stream data_in[1024],stream data_out[32]){
#pragma HLS INTERFACE axis register both port=data_out
#pragma HLS INTERFACE axis register both port=data_in
#pragma HLS INLINE recursive

        celda data[NUM_ELEMENT];
        #pragma HLS ARRAY_PARTITION variable=data complete dim=1

        celda buffer[NUM_ELEMENT];
        #pragma HLS ARRAY_PARTITION variable=buffer complete dim=1

        celda buff_in[1024];
        celda buff_out[1024];

        bool temp=0;
        bool flag_1=0;


        READ:for (int i=0;i<1024;i++){
                buff_in[i].value = data_in[i].data;
                buff_in[i].ind = i;
                temp=data_in[i].last;
        }
```

```
//-----------------------------------------------------------------------
        short int num_iter;

        flag_1=0;

        for(int y=0;y<2;y++){

                if (flag_1==0){
                        num_iter=8;
                }else{
                        num_iter=2;
                }

                for (int m=0;m<num_iter;m++){

                        for(int h=0;h<128;h++){
                                #pragma HLS PIPELINE
                                data[h]=buff_in[h+m*128];
                        }

                        for (int m=0;m<NUM_ELEMENT/2;m++){

                                sort_1:for (int i=0;i<NUM_ELEMENT/2;i++){
                                        #pragma HLS UNROLL
                                        Comp(data[i*2],data[i*2+1],&buffer[i*2+1],&buffer[i*2]);
                                }


                                sort_2:for (int i=1;i<NUM_ELEMENT-2;i=i+2){
                                        #pragma HLS UNROLL
                                        Comp(buffer[i],buffer[i+1],&data[i+1],&data[i]);
                                }

                                data[0] = buffer[0];
                                data[NUM_ELEMENT-1] = buffer[NUM_ELEMENT-1];

                        }

                        for(int h=0;h<32;h++){
                                #pragma HLS PIPELINE
                                buff_in[h+m*32]=data[h];
                        }
                }

                if (flag_1==0){
                        flag_1=1;
                }
        }
//-----------------------------------------------------------------------
        temp = 0;
        short int pointer_A,pointer_B;

        pointer_A = 0;
        pointer_B = 0;

        for (int i=0;i<32;i++){
                if(i==(32-1)){
                        temp = 1;
                }

                if (buff_in[pointer_A].value>=buff_in[pointer_B+32].value){
                        data_out[i].data=buff_in[pointer_A].ind;
                        data_out[i].last=temp;
                        pointer_A++;
                }else{
                        data_out[i].data=buff_in[pointer_B+32].ind;
                        data_out[i].last=temp;
                        pointer_B++;
                }
        }
}
```

Controlador del DAC

(*DAC_CONTROLLER*)

```
#define SAMPLE_SIZE 1024
#define SAMPLE_RESOLUTION 24
#define UPSAMPLING_FACTOR 32

#include "ap_int.h"

struct stream{
  ap_int<32> data;
  bool last;
};

bool edge2pulse(bool signal){

                static ap_int<3> reg=0;
                reg=reg<<1;
                reg.bit(0)=signal;
```

```
                if (!reg.bit(2) & reg.bit(1)) return true;
                else return false;
}

void DAC_CONTROLLER(stream data_in[SAMPLE_SIZE],
                ap_uint<1>* reset_ctrl,
                volatile ap_int<SAMPLE_RESOLUTION> *data_out,
                volatile bool *cmd){

#pragma HLS INTERFACE axis register both port=data_in


        ap_int<SAMPLE_RESOLUTION> data[SAMPLE_SIZE];
        bool temp=0;

        *reset_ctrl=1;
        for (int i=0;i<SAMPLE_SIZE;i++){
                data[i]=(ap_int<SAMPLE_RESOLUTION>)data_in[i].data;
                temp=data_in[i].last;
        }

        *reset_ctrl=1;
        int index=0;
        int counter=0;
        *reset_ctrl=0;
        ap_int<SAMPLE_RESOLUTION> buffer;
        static ap_int<3> reg = 0;

        while(1){

                if (edge2pulse(*cmd)){

                        index++;
                }

                *data_out=data[index];

                if (index==1024){
                        index=0;
                }

        }

}
```

A.1.3. VHDL

Modulador Delta-Sigma

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.numeric_std.ALL;

entity DELTA_SIGMA is
    generic (N : integer:=24;
             M : integer:=18 );

    Port ( clk : in STD_LOGIC;
               data : in signed ((N-1) downto 0);
           dac_out : out signed (0 downto 0);
           reset : in STD_LOGIC);
end DELTA_SIGMA;

architecture Behavioral of DELTA_SIGMA is

begin

    process(clk,reset)

        variable reg_a: signed ((N-1) downto 0)
                        :=to_signed(0,N);
        variable reg_b: signed ((N-1) downto 0)
                        :=to_signed(0,N);
        variable reg_c: signed ((N-1) downto 0)
                        :=to_signed(0,N);
        variable reg_d: signed ((N-1) downto 0)
                        :=to_signed(0,N);
        variable reg_y: signed ((N-1) downto 0)
                        :=to_signed(0,N);
        variable reg_a_1: signed ((N-1) downto 0)
                        :=to_signed(0,N);
        variable reg_b_1: signed ((N-1) downto 0)
                        :=to_signed(0,N);
        variable reg_c_1: signed ((N-1) downto 0)
                        :=to_signed(0,N);
        variable reg_d_1: signed ((N-1) downto 0)
                        :=to_signed(0,N);
        variable reg_y_1: signed ((N-1) downto 0)
                        :=to_signed(0,N);
```

```
        variable reg_x: signed ((N-1) downto 0)
                    :=to_signed(0,N);
        variable reg_x_1: signed ((N-1) downto 0)
                    :=to_signed(0,N);
        variable temp1,temp2: signed ((N-1) downto 0)
                    :=to_signed(0,N);
        begin

            if (reset = '1') then

                reg_a:=to_signed(0,N);
                reg_c:=to_signed(0,N);
                reg_b:=to_signed(0,N);
                reg_d:=to_signed(0,N);
                reg_y:=to_signed(0,N);
                reg_a_1:=to_signed(0,N);
                reg_c_1:=to_signed(0,N);
                reg_b_1:=to_signed(0,N);
                reg_d_1:=to_signed(0,N);
                reg_y_1:=to_signed(0,N);
                reg_x_1:=to_signed(0,N);
                reg_x:=to_signed(0,N);

            end if;

            if rising_edge(clk) then

                reg_x:=data;
                temp1:=reg_x_1-reg_y_1;
                temp2:=reg_b_1-reg_y_1;

                if(temp1>=0) then
                    reg_a:=shift_right(temp1,1);
                else
                    reg_a:=-shift_right(-temp1,1);
                end if;

                if(temp2>=0) then
                    reg_c:=shift_right(temp2,1);
                else
                    reg_c:=-shift_right(-temp2,1);
                end if;

                reg_b:=reg_a+reg_b_1;

                reg_d:=reg_c+reg_d_1;

               if (reg_d>=to_signed(0,N)) then
                    reg_y:=to_signed(2 **(M-1)-1,N);

                else

                    reg_y:=to_signed(-2 **(M-1),N);

                end if;

                ------------------------------------------------------------

                dac_out<=not(reg_y((N-1) downto (N-1)));

                --actualizar estados
                reg_a_1:=reg_a;
                reg_c_1:=reg_c;
                reg_b_1:=reg_b;
                reg_d_1:=reg_d;
                reg_y_1:=reg_y;
                reg_x_1:=reg_x;

            end if;
        end process;

end Behavioral;
```

Distribuidor de Reloj (*CLOCK_DIST*)

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.numeric_std.ALL;
entity CLOCK_DIST is
    Port ( clk : in STD_LOGIC;
           reset : in STD_LOGIC;
           a : out SIGNED (0 downto 0);
           x1 : out SIGNED (0 downto 0);
           x2 : out  SIGNED (0 downto 0);
           x4 : out  SIGNED (0 downto 0);
           x8 : out  SIGNED (0 downto 0);
           x16 : out  SIGNED (0 downto 0);
           x32 : out  SIGNED (0 downto 0)
           );
end CLOCK_DIST;

architecture Behavioral of CLOCK_DIST is
```

```
    signal temporal: STD_LOGIC;


    signal counter : integer range 0 to 32 := 0;

begin

   process (clk,reset)

    begin

        if (reset = '1') then
            temporal <= '0';
            counter<= 0;
        end if;

       if rising_edge(clk) then
            if (counter = 15) then --mitad menos uno
                temporal<= NOT(temporal);
                counter <= 0;
            else
                counter <= counter + 1;
            end if;
        end if;
    end process;

    process(temporal,reset)

        variable registro: SIGNED (6 downto 0) := to_signed(0,7);

        begin
        if (reset = '1') then
            registro:=to_signed(0,7);

        end if;
        if rising_edge(temporal) then
            if (registro = to_signed(63,7)) then
                registro:=to_signed(0,7);

            else
                registro:= registro + to_signed(1,7);

            end if;
        end if;

        a <= not registro(5 downto 5);
        x1 <=  registro(5 downto 5);
        x2 <=  registro(4 downto 4);
        x4 <=  registro(3 downto 3);
        x8 <=  registro(2 downto 2);
        x16 <=  registro(1 downto 1);
        x32 <=  registro(0 downto 0);

    end process;


end Behavioral;
```

Contador (*COUNTER*)

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
entity COUNTER is
    Port ( CLK : in STD_LOGIC;
           rst: in STD_LOGIC;
           COUNT : out STD_LOGIC);
end COUNTER;

architecture Behavioral of COUNTER is

    signal counter : integer range 0 to 1 := 0;
    signal temporal: STD_LOGIC;

begin

   process (clk,rst)


    begin

        if (rst = '1') then
            counter<= 0;
        end if;

       if rising_edge(clk) then
            if (counter = 1) then --mitad menos uno
                temporal<='1';
                counter <= 0;
            else
                temporal<='0';
                counter <= counter + 1;
            end if;
        end if;
```

```
        COUNT<=temporal;
    end process;

end Behavioral;
```

Multiplexor (*MUX*)

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.numeric_std.ALL;

entity MUX is
    Port ( SELECTOR : in STD_LOGIC;
           A : in SIGNED (23 downto 0);
           B : in SIGNED (23 downto 0);
           C : out SIGNED (23 downto 0));
end MUX;

architecture Behavioral of MUX is

begin

    process(SELECTOR)

    begin

        if SELECTOR = '0' then
            C<=A;
        elsif SELECTOR = '1' then
            C<=B;
        end if;

    end process;

end Behavioral;
```

Cadena de retardo $E_1(z^{-1})$ (*DELAY_14*)

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.numeric_std.ALL;

entity DELAY_14 is

    Port (
          clk : in STD_LOGIC;
          reset : in STD_LOGIC;
           data_in : in SIGNED (23 downto 0);
           data_out : out SIGNED (23 downto 0));

end DELAY_14;

architecture Behavioral of DELAY_14 is

begin

    process(clk,reset)

        type mem is array (0 to 14) of SIGNED (23 downto 0);


        variable TAPS: mem;
        variable xn:  SIGNED (23 downto 0);
        variable acumulator: SIGNED (41 downto 0):=to_signed(0,42);


        begin

            if (reset = '1') then

                for I in 0 to 14 loop
                    TAPS(I):=to_signed(0,24);
                end loop;

            end if;

            if rising_edge(clk) then

                TAPS(0):=data_in;
                data_out<=TAPS(14);

                for I in 14 downto 1 loop
                    TAPS(I):=TAPS(I-1);
                end loop;

            end if;

        end process;

end Behavioral;
```

FIltro FIR 1 (*fir_X2*)

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.numeric_std.ALL;

entity fir_X2 is

    Port (
          clk : in STD_LOGIC;
          reset : in STD_LOGIC;
           data_in : in SIGNED (23 downto 0);
           data_out : out SIGNED (23 downto 0));

end fir_X2;

architecture Behavioral of fir_X2 is

begin

    process(clk,reset)

        type mem is array (0 to 29) of SIGNED (23 downto 0);
        type coeficiente is array (0 to 14) of SIGNED (17 downto 0);

        variable TAPS: mem;
        variable xn:  SIGNED (23 downto 0);
        variable acumulator: SIGNED (41 downto 0):=to_signed(0,42);

        variable fir_c:coeficiente:=(0 =>to_signed(19,18),1=>
        to_signed(-45,18),2=>to_signed(95,18),3=>to_signed(-178,18),
        4=>to_signed(306,18),5=>to_signed(-495,18),6=>to_signed(765,18),
        7=>to_signed(-1142,18),8=>to_signed(1661,18),
        9=>to_signed(-2381,18),10=>to_signed(3406,18),11=>to_signed(-4953,18),
        12=>to_signed(7597,18),13=>to_signed(-13447,18),
        14=>to_signed(41566,18));

        begin

            if (reset = '1') then

                for I in 0 to 29 loop
                    TAPS(I):=to_signed(0,24);
                end loop;

            end if;

            if rising_edge(clk) then

                TAPS(0):=data_in;
                acumulator:=to_signed(0,42);

                for I in 0 to 14 loop
                      acumulator
                      :=acumulator +fir_c(I)*(TAPS(I)+TAPS(29-I));
                  end loop;

                  --luego de acumular, shift y resize
                 if (acumulator >= 0 ) then
                    data_out<=resize(shift_right(acumulator,16),24);
                 else
                     data_out<=resize(-shift_right(-acumulator,16),24);
                 end if;

                -- actualizar registros

                for I in 29 downto 1 loop
                    TAPS(I):=TAPS(I-1);
                end loop;

            end if;

        end process;

end Behavioral;
```

LATCH

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.numeric_std.ALL;


entity LATCH is
    Port ( clk : in STD_LOGIC;
           d_in : in SIGNED (23 downto 0);
           d_out : out SIGNED(23 downto 0));
end LATCH;

architecture Behavioral of LATCH is

begin
```

```
    process(clk)
        begin
            if rising_edge(clk) then
                d_out<=d_in;
            end if;
        end process;

end Behavioral;
```

A.1.4. C Embebido

Software del Procesador (main.c)

```
#define XAXIS_SWITCH_DEVICE_ID          XPAR_AXIS_SWITCH_0_DEVICE_ID

#include <stdio.h>
#include "xparameters.h"
#include "netif/xadapter.h"
#include "platform_config.h"
#include "xil_printf.h"
#include "xaxidma.h"
#include "xgpio.h"
#include "xdmaps.h"
#include "xaxis_switch.h"

#if LWIP_DHCP==1
#include "lwip/dhcp.h"
#endif

#ifdef XPS_BOARD_ZCU102
#ifdef XPAR_XIICPS_0_DEVICE_ID
int IicPhyReset(void);
#endif
#endif

int main_thread();
void print_echo_app_header();
void echo_application_thread(void *);
void init_thread();
void lwip_init();
void DAC_INIT();

#if LWIP_DHCP==1
extern volatile int dhcp_timoutcntr;
err_t dhcp_start(struct netif *netif);
#endif

#define THREAD_STACKSIZE 1024

static struct netif server_netif;
struct netif *echo_netif;

//Variables Globales del sistema-----------------------------------------------------------------------

float *sequences;
float dictionary[1048576];//almacena los datos que envia el GUI
float A[65536];
float y[64];
char *preMat;

u8 MiIndex;
u8 SiIndex;

XAxiDma AxiDma_0;
XAxiDma AxiDma_1;
XAxiDma AxiDma_2;
XAxiDma AxiDma_3;
XAxiDma AxiDma_4;

XGpio mode,reset,reset_sorter,reset_dac;

XDmaPs PSDma;



XAxis_Switch AxisSwitch;

extern void INIT_HARD_THRESH();
void multiplyVecHW(float A[],float B[],float C[]);
//-----------------------------------------------------------------------------------------------------

void
print_ip(char *msg, struct ip_addr *ip)
{
        xil_printf(msg);
        xil_printf("%d.%d.%d.%d\n\r", ip4_addr1(ip), ip4_addr2(ip),
                        ip4_addr3(ip), ip4_addr4(ip));
}

void
print_ip_settings(struct ip_addr *ip, struct ip_addr *mask, struct ip_addr *gw)
```

```
{

        print_ip("Board IP: ", ip);
        print_ip("Netmask : ", mask);
        print_ip("Gateway : ", gw);
}

int main()
{
        sys_thread_new("init_thrd", (void(*)(void*))init_thread, 0,
                        THREAD_STACKSIZE,
                        DEFAULT_THREAD_PRIO);
        vTaskStartScheduler();
        while(1);
        return 0;
}

void init_thread(){

        /*Aca se ejecuta la inicializacion del sistema , incluidos
         * sus perifericos y variables globales
        */
        int Status;


        init_platform(); //habilito PL

        //Variables Globales-----------------------------------------------------------------------


        sequences = (float*)malloc(16384);

        if (sequences == NULL){
                xil_printf("error al alocar memoria para Sequencias\n");

        }

        printf("pointer SEQ: %p\n",(void *)sequences);

        //dictionary = (float*)malloc(4194304);

        if (dictionary == NULL){
                xil_printf("error al alocar memoria para Dictionary\n");

        }

        printf("pointer DIC: %p\n",(void *)dictionary);


        printf("pointer RES: %p\n",(void *)A);

        //GPIOs-----------------------------------------------------------------------------------
        XGpio_Config *modeCfg;

        XGpio_Config *resetCfg;

        Status = XGpio_Initialize(&reset, XPAR_AXI_GPIO_1_DEVICE_ID);

        if(Status != XST_SUCCESS){
                xil_printf("Error al inicializar GPIO reset");
        }

        resetCfg = XGpio_LookupConfig(XPAR_AXI_GPIO_1_DEVICE_ID);

        if(!modeCfg ){
                xil_printf("Cf not found GPIO (reset)\n");
        }

        Status = XGpio_CfgInitialize(&reset, resetCfg,resetCfg->BaseAddress);

        if(Status != XST_SUCCESS){
                xil_printf("GPIO (reset) Init Failed.\n");
        }

        //DMA CONFIG HW Accelerator----------------------------------------------------------------

        XAxiDma_Config *CfgPtr0;
        XAxiDma_Config *CfgPtr1;
        XAxiDma_Config *CfgPtr2;

        CfgPtr0 = XAxiDma_LookupConfig(XPAR_AXI_DMA_0_DEVICE_ID);

        if(!CfgPtr0 ){
                xil_printf("Cf not found DMA 0\n");
        }

        CfgPtr1 = XAxiDma_LookupConfig(XPAR_AXI_DMA_1_DEVICE_ID);

        if(!CfgPtr1 ){
                xil_printf("Cf not found DMA 1\n");
        }

        CfgPtr2 = XAxiDma_LookupConfig(XPAR_AXI_DMA_2_DEVICE_ID);

        if(!CfgPtr2 ){
```

```
                xil_printf("Cf not found DMA 2\n");
        }

        Status = XAxiDma_CfgInitialize(&AxiDma_0,CfgPtr0);

        if(Status != XST_SUCCESS){
                xil_printf("DMA Init Failed.\n");
        }

        Status = XAxiDma_CfgInitialize(&AxiDma_1,CfgPtr1);

        if(Status != XST_SUCCESS){
                xil_printf("DMA Init Failed.\n");
        }

        Status = XAxiDma_CfgInitialize(&AxiDma_2,CfgPtr2);

        if(Status != XST_SUCCESS){
                xil_printf("DMA Init Failed.\n");
        }

        XAxiDma_IntrDisable(&AxiDma_0,XAXIDMA_IRQ_ALL_MASK,XAXIDMA_DEVICE_TO_DMA);
        XAxiDma_IntrDisable(&AxiDma_0,XAXIDMA_IRQ_ALL_MASK,XAXIDMA_DMA_TO_DEVICE);

        XAxiDma_IntrDisable(&AxiDma_1,XAXIDMA_IRQ_ALL_MASK,XAXIDMA_DEVICE_TO_DMA);
        XAxiDma_IntrDisable(&AxiDma_1,XAXIDMA_IRQ_ALL_MASK,XAXIDMA_DMA_TO_DEVICE);

        XAxiDma_IntrDisable(&AxiDma_2,XAXIDMA_IRQ_ALL_MASK,XAXIDMA_DEVICE_TO_DMA);
        XAxiDma_IntrDisable(&AxiDma_2,XAXIDMA_IRQ_ALL_MASK,XAXIDMA_DMA_TO_DEVICE);



        //------------------------------------------------------------------------
        INIT_HARD_THRESH();
        DAC_INIT();
        //------------------------------------------------------------------------
        //Configuracion DMA PS

        XDmaPs_Config *DmaCfg;
        XDmaPs_Cmd DmaCmd;

        memset(&DmaCmd, 0, sizeof(XDmaPs_Cmd));

    /*
        DmaCmd.ChanCtrl.SrcBurstSize = 4;
        DmaCmd.ChanCtrl.SrcBurstLen = 4;
        DmaCmd.ChanCtrl.SrcInc = 1;
        DmaCmd.ChanCtrl.DstBurstSize = 4;
        DmaCmd.ChanCtrl.DstBurstLen = 4;
        DmaCmd.ChanCtrl.DstInc = 1;
        DmaCmd.BD.SrcAddr = (u32) 0;
        DmaCmd.BD.DstAddr = (u32) 0;
        DmaCmd.BD.Length = 64 * sizeof(float);*/


        DmaCfg = XDmaPs_LookupConfig(XPAR_PS7_DMA_NS_DEVICE_ID);

        if (DmaCfg == NULL) {
                xil_printf("DMA PS Init Failed.\n");
        }

        Status = XDmaPs_CfgInitialize(&PSDma,DmaCfg,DmaCfg->BaseAddress);

        if (Status != XST_SUCCESS) {
                xil_printf("DMA PS Init Failed.\n");
        }
        //------------------------------------------------------------------------
        //AXI SWITCH
        XAxis_Switch_Config *Config;
        Config = XAxisScr_LookupConfig(XAXIS_SWITCH_DEVICE_ID);
        if (NULL == Config) {
                xil_printf("Error al Iniciar AXI SWITCH");
        }

        Status = XAxisScr_CfgInitialize(&AxisSwitch, Config,
                                                Config->BaseAddress);
        if (Status != XST_SUCCESS) {
                xil_printf("AXI4-Stream initialization failed.\r\n");

        }

        MiIndex = 0;
        SiIndex = 0;
        XAxisScr_MiPortEnable(&AxisSwitch, MiIndex, SiIndex);
        XAxisScr_RegUpdateEnable(&AxisSwitch);

        //------------------------------------------------------------------------

        sys_thread_new("main_thrd", (void(*)(void*))main_thread, 0,
                        THREAD_STACKSIZE,
                        DEFAULT_THREAD_PRIO);
        vTaskDelete(NULL);
}

void network_thread(void *p)
```

```
{
    struct netif *netif;
    struct ip_addr ipaddr, netmask, gw;

    /* the mac address of the board. this should be unique per board */
    unsigned char mac_ethernet_address[] = { 0x00, 0x0a, 0x35, 0x00, 0x01, 0x02 };

    netif = &server_netif;

#if LWIP_DHCP==0
    /* initliaze IP addresses to be used */
    IP4_ADDR(&ipaddr,  192, 168, 1, 10);
    IP4_ADDR(&netmask, 255, 255, 255,  0);
    IP4_ADDR(&gw,      192, 168, 1, 1);
#endif

    /* print out IP settings of the board */
    xil_printf("\r\n\r\n");
    xil_printf("-----lwIP Socket Aplicaci n Sensado Compresivo------\r\n");
    xil_printf("            Autor: Mariano Leonel Acosta\n\n");

#if LWIP_DHCP==0
    print_ip_settings(&ipaddr, &netmask, &gw);
    /* print all application headers */
#endif


    /* Add network interface to the netif_list, and set it as default */
    if (!xemac_add(netif, &ipaddr, &netmask, &gw, mac_ethernet_address, PLATFORM_EMAC_BASEADDR))
    {
        xil_printf("Error adding N/W interface\r\n");
        return;
    }
    netif_set_default(netif);

    /* specify that the network if is up */
    netif_set_up(netif);

    /* start packet receive thread - required for lwIP operation */
    sys_thread_new("xemacif_input_thread", (void(*)(void*))xemacif_input_thread, netif,
            THREAD_STACKSIZE,
            DEFAULT_THREAD_PRIO);


    xil_printf("\r\n");
    xil_printf("%20s %6s %s\r\n", "Server", "Port", "Connect With..");
    xil_printf("%20s %6s %s\r\n", "--------------------", "------", "--------------------");

    print_echo_app_header();
    xil_printf("\r\n");
    sys_thread_new("echod", echo_application_thread, 0,
                THREAD_STACKSIZE,
                DEFAULT_THREAD_PRIO);
    vTaskDelete(NULL);

    return;
}

int main_thread()
{


#ifdef XPS_BOARD_ZCU102
        IicPhyReset();
#endif

        /* initialize lwIP before calling sys_thread_new */
    lwip_init();

    /* any thread using lwIP should be created using sys_thread_new */
    sys_thread_new("NW_THRD", network_thread, NULL,
                THREAD_STACKSIZE,
            DEFAULT_THREAD_PRIO);
#if LWIP_DHCP==1
    while (1) {
        vTaskDelay(DHCP_FINE_TIMER_MSECS / portTICK_RATE_MS);
                if (server_netif.ip_addr.addr) {
                        xil_printf("DHCP request success\r\n");
                        print_ip_settings(&(server_netif.ip_addr), &(server_netif.netmask),
                        &(server_netif.gw));
                        print_echo_app_header();
                        xil_printf("\r\n");
                        sys_thread_new("echod", echo_application_thread, 0,
                                        THREAD_STACKSIZE,
                                        DEFAULT_THREAD_PRIO);
                        break;
                }
                mscnt += DHCP_FINE_TIMER_MSECS;
                if (mscnt >= 10000) {
                        xil_printf("ERROR: DHCP request timed out\r\n");
                        xil_printf("Configuring default IP of 192.168.1.10\r\n");
                        IP4_ADDR(&(server_netif.ip_addr),  192, 168, 1, 10);
                        IP4_ADDR(&(server_netif.netmask), 255, 255, 255,  0);
                        IP4_ADDR(&(server_netif.gw),  192, 168, 1, 1);
                        print_ip_settings(&(server_netif.ip_addr), &(server_netif.netmask),
```

```
                        &(server_netif.gw));
                        /* print all application headers */
                        xil_printf("\r\n");
                        xil_printf("%20s %6s %s\r\n", "Server", "Port", "Connect With..");
                        xil_printf("%20s %6s %s\r\n", "--------------------", "------",
                        "--------------------");

                        print_echo_app_header();
                        xil_printf("\r\n");
                        sys_thread_new("echod", echo_application_thread, 0,
                                        THREAD_STACKSIZE,
                                        DEFAULT_THREAD_PRIO);
                        break;
                }
        }
#endif
    vTaskDelete(NULL);
    return 0;
}

void multiplyVecHW(float A[],float B[],float C[]){

        int opt = 1;
        int status ;
                switch (opt){
                        case 0:
                                Xil_DCacheFlushRange((u32)C,256); //PASO IMPORTANTE: Limpiar Cache (sino no anda)
                                Xil_DCacheFlushRange((u32)A,256);
                                Xil_DCacheFlushRange((u32)B,16384);

                                //XTime_GetTime(&tStart);
                                status = XAxiDma_SimpleTransfer(&AxiDma_0,(u32)A,256,XAXIDMA_DMA_TO_DEVICE);
                                //ENVIAR A

                                if (status != XST_SUCCESS){
                                        xil_printf("error al enviar matriz A\n");
                                        break;

                                }


                                status = XAxiDma_SimpleTransfer(&AxiDma_1,(u32)B,16384,XAXIDMA_DMA_TO_DEVICE);
                                //ENVIAR B
                                if (status != XST_SUCCESS){
                                        xil_printf("error al enviar  matriz B\n");
                                        break;

                                }

                                while(XAxiDma_Busy(&AxiDma_0,XAXIDMA_DMA_TO_DEVICE)||XAxiDma_Busy(&AxiDma_1,
                                XAXIDMA_DMA_TO_DEVICE));


                                //Recibir Multiplicacion


                                status = XAxiDma_SimpleTransfer(&AxiDma_2,(u32)C,256,XAXIDMA_DEVICE_TO_DMA);

                                if (status != XST_SUCCESS){
                                        xil_printf("error al recibir matriz C\n");
                                        break;

                                }

                                while(XAxiDma_Busy(&AxiDma_2,XAXIDMA_DEVICE_TO_DMA));


                                while((XAxiDma_Busy(&AxiDma_2,XAXIDMA_DEVICE_TO_DMA))||(
                                (XAxiDma_Busy(&AxiDma_1,XAXIDMA_DMA_TO_DEVICE))||(XAxiDma_Busy(&AxiDma_0,
                                XAXIDMA_DMA_TO_DEVICE)) ));

                                Xil_DCacheFlushRange((u32)C,256);
                                //XTime_GetTime(&tEnd);

                                break;

                        case 1:
                                //XTime_GetTime(&tStart);


                                for (int z=0;z<256;z++){
                                        C[z]=0;
                                }
                                for(int j=0;j<1;j++){
                                        for(int i=0;i<64;i++){
                                                for(int k=0;k<64;k++){

                                                        C[i+j*64]+= A[k+j*64]* B[64*k+i];
                                                }
                                        }
                                }

                                //XTime_GetTime(&tEnd);
                                break;
```

```
                }


}
void DAC_INIT(){

        //GPIO reset_dac

        XGpio_Config *resetCfg;

         int Status = XGpio_Initialize(&reset_dac, XPAR_AXI_GPIO_4_DEVICE_ID);

        if(Status != XST_SUCCESS){
                xil_printf("Error al inicializar GPIO reset_dac");
        }

        resetCfg = XGpio_LookupConfig(XPAR_AXI_GPIO_4_DEVICE_ID);

        if(!resetCfg ){
                xil_printf("Cf not found GPIO (reset_dac)\n");
        }

         Status = XGpio_CfgInitialize(&reset_dac, resetCfg,resetCfg->BaseAddress);

        if(Status != XST_SUCCESS){
                xil_printf("GPIO (reset_dac) Init Failed.\n");
        }else{
                xil_printf("Init GPIO  ok\n");
        }
}
```

Software del Procesador (network.c)

```
#include <stdio.h>
#include <string.h>

#include "lwip/sockets.h"
#include "netif/xadapter.h"
#include "lwipopts.h"
#include "xil_printf.h"
#include "FreeRTOS.h"
#include "task.h"
#include "xtime_l.h"
#include "xaxidma.h"
#include "xgpio.h"
#include "xil_cache.h"
#include "xdmaps.h"
#include "xaxis_switch.h"

#define THREAD_STACKSIZE 1024

u16_t port_num = 1491; //user port no asignado por el IANA

extern float *sequences;
extern float dictionary[1048576];
extern float A[65536];
extern float y[64];
float x[1024] = {[0 ... 1023]=0};
float zeros[1024] = {[0 ... 1023]=0};
extern char *preMat;
float opt[4];
float r[32];
float signal[1024];

extern XAxiDma AxiDma_0;
extern XAxiDma AxiDma_1;
extern XAxiDma AxiDma_2;
extern XAxiDma AxiDma_3;

extern XGpio mode;
extern XGpio reset;
extern XGpio reset_dac;

extern XDmaPs PSDma;

extern XAxis_Switch AxisSwitch;
extern u8 MiIndex;
extern u8 SiIndex;

float buffA[256];
//float buffB[4096];
float *buffB;
float buffC[256];

XTime tStart,tEnd;
float times[3] = {[0 ... 2]=0};

int32_t datosDAC[1024] = {[0 ... 1023]=0};

//void multiplyAB(void* p);
void multiplyAB();
extern void ANIHT(float y[64],float A[65536],int N,
```

```
int k, float tol,float x[1024],float tol_sub,int itMax,float rr[32]);
extern void recovery(float Dic[1024*1024],float r[],
int k,float x[1024],float a[1024]);
void DMA_TRANSFER_DAC(int32_t datos[1024]);
extern void convert_to_dac_format(float datosIn[1024],
int32_t datosOut[1024],float maxValue,int res);

xTaskHandle netHandler;
xTaskHandle optHandler;
xTaskHandle multiHandler;

void print_echo_app_header()
{
    xil_printf("%20s %6d %s\r\n", "echo server",
                        port_num,
                        "$ telnet <board_ip> 7");

}

/* thread spawned for each connection */
void process_echo_request(void *p)
{
        int sd = (int)p;
        int RECV_BUF_SIZE = 2048;
        char recv_buf[RECV_BUF_SIZE];
        char Cmd [4];

        int n, nwrote;



        while (1) {
                /* read a max of RECV_BUF_SIZE bytes from socket */
                if ((n = read(sd, recv_buf, RECV_BUF_SIZE)) < 0) {
                        xil_printf("%s: error reading from socket %d,
                        closing socket\r\n", __FUNCTION__, sd);
                        break;
                }else{
                        if (n<=0){
                                break; //desconectarse si se cierra la conexion
                        }
                        xil_printf("\nComando recibido.\n");
                        xil_printf("Tama o (bytes):%d.\n",n);

                        for (int i=0;i<4;i++){
                                Cmd[i]=recv_buf[i+4];
                                printf("%c",*(Cmd+i));

                        }
                        printf("\n");
                }


                if (!strncmp(Cmd, "MTXB", 4)){

                        char send_buf[4]={'S','N','D','B'};

                        if ((nwrote = write(sd, send_buf, 4)) < 0) { // comando SNDA
                                xil_printf("%s: ERROR responding to client echo
                                request. received = %d, written = %d\r\n",
                                        __FUNCTION__, n, nwrote);
                                xil_printf("Closing socket %d\r\n", sd);
                                break;
                        }else{ // Si se envio correctamente, recibir matriz


                                preMat = (char*)dictionary;
                                int pointer = 0;

                                while(pointer<4194304){

                                        if ((n = read(sd, recv_buf, RECV_BUF_SIZE)) < 0) {
                                                xil_printf("%s: error reading from socket %d,
                                                closing socket\r\n", __FUNCTION__, sd);
                                                break;
                                        }else{

                                                        memcpy((preMat+pointer),recv_buf,n);


                                                        pointer+=n;
                                                }
                                        };



                                xil_printf("\nDictionary Loaded!\n");
                                /*for (int i=0;i<1048576;i++){
                                        printf("%d: %.10f ,\n",i,dictionary[i]);
                                }*/

                        }

                }else if(!strncmp(Cmd, "MTXA", 4)){
```

```
                char send_buf[4]={'S','N','D','A'};
                if ((nwrote = write(sd, send_buf, 4)) < 0) { // comando SNDB
                                xil_printf("%s: ERROR responding to client
                                echo request. received = %d, written = %d\r\n",
                                                __FUNCTION__, n, nwrote);
                                xil_printf("Closing socket %d\r\n", sd);
                                break;
                        }else{ // Si se envio correctamente, recibir matriz


                                preMat = (char*)malloc(16384);
                                int pointer = 0;

                                while(pointer<16384){

                                        if ((n = read(sd, recv_buf, RECV_BUF_SIZE)) < 0) {
                                                xil_printf("%s: error reading from
                                                socket %d, closing socket\r\n", __FUNCTION__, sd);
                                                break;
                                        }else{

                                                        memcpy((preMat+pointer),recv_buf,n);


                                                        pointer+=n;
                                                }
                                        };



                                memcpy(sequences,preMat,16384);
                                free(preMat);

                                xil_printf("\nSequences Loaded!\n");

                                /*for (int i=0;i<4096;i++){
                                        printf("%d: %.10f ,\n",i,sequences[i]);
                                }*/

                                }
        }else if(!strncmp(Cmd, "MLTM", 4)){
                int p =0;

                vTaskPrioritySet(NULL,6);


                XGpio_DiscreteWrite(&reset, 1,0);//setear gpio_1 a 0; //reset

                XGpio_DiscreteWrite(&reset, 1,1);//setear gpio_1 a 1; //enable

                buffB = (float*)malloc(16384);

                Xil_DCacheFlushRange((u32)A,262144);
                XTime_GetTime(&tStart);

                for(int k=0;k<16;k++){

                        for (int f=0;f<4;f++){

                                memcpy((void*)(buffA + f*64),(void*)(sequences +64*k + 1024*f),256);

                        }

                        for (int j=0;j<16;j++){

                                /*for (int i=0;i<64;i++){

                                        memcpy((void*)(buffB + i*64),(void*)(dictionary
                                        + j*64 + 1024*i + 65536*k),256);

                                }*/

                                buffB = (dictionary +j*4096+65536*k);

                                //multiHandler = sys_thread_new("multiply",
                                multiplyAB,(void*)p,THREAD_STACKSIZE*4 ,2);

                                multiplyAB();

                                //vTaskSuspend(NULL);
                                //vTaskDelete(multiHandler);

                                //memcpy((void*)(A + j*256 + 4096*k ),(void*)buffC,1024);
                                //copiar submatrix a Aado


                                for (int f=0;f<4;f++){

                                        memcpy((void*)(A + 1024*f + j*64 + k*4096),
                                        (void*)(buffC+f*64),256);

                                }
```

```
            }
        }

        XTime_GetTime(&tEnd);

        vTaskPrioritySet(NULL,DEFAULT_THREAD_PRIO);
        xil_printf("\nMatriz A Obtenida\n");
        times[0] = 1.0*(tEnd-tStart)/(COUNTS_PER_SECOND/1000000)/1000;
        printf("Time required: %.5f mSeg\n",times[0]);
        Xil_DCacheFlushRange((u32)A,262144);


        vTaskPrioritySet(NULL,6);

    XTime_GetTime(&tStart);
    memcpy(x,zeros,4096);


        ANIHT(y,A,1024,(int)opt[0],opt[1],x,opt[2],(int)opt[3],r);

        XTime_GetTime(&tEnd);
        vTaskPrioritySet(NULL,DEFAULT_THREAD_PRIO);
        xil_printf("\nOptimizaci n Completada\n");
        times[1] = 1.0*(tEnd-tStart)/(COUNTS_PER_SECOND/1000000)/1000;
        printf("Time required: %.5f mSeg\n",times[1]);

        XTime_GetTime(&tStart);
        recovery(dictionary,r,(int)opt[0],signal,x);
        XTime_GetTime(&tEnd);

        xil_printf("\nSe al Reconstruida\n");
        times[2] = 1.0*(tEnd-tStart)/(COUNTS_PER_SECOND/1000000)/1000;
        printf("Time required: %.5f mSeg\n\n",times[2]);

        //envio de la se al reconstruida-----------------------------------------------------------
        if ((nwrote = write(sd, signal, 1024)) < 0) { // comando SNDA
                xil_printf("%s: ERROR responding to client echo request. received = %d,
                written = %d\r\n",
                                __FUNCTION__, n, nwrote);
                xil_printf("Closing socket %d\r\n", sd);
                break;}

        for (int g=0;g<3;g++){
                if ((n = read(sd, recv_buf, RECV_BUF_SIZE)) < 0) {
                        xil_printf("%s: error reading from socket %d,
                        closing socket\r\n", __FUNCTION__, sd);
                        break;
                }

                for (int i=0;i<4;i++){
                        Cmd[i]=recv_buf[i+4];
                        printf("%c",*(Cmd+i));

                }

                if ((nwrote = write(sd, (signal+256*(g+1)), 1024)) < 0) {// comando SNDA
                        xil_printf("%s: ERROR responding to client echo request.
                        received = %d, written = %d\r\n",
                                        __FUNCTION__, n, nwrote);
                        xil_printf("Closing socket %d\r\n", sd);
                        break;}
        }

        if ((n = read(sd, recv_buf, RECV_BUF_SIZE)) < 0) {
                xil_printf("%s: error reading from socket %d, closing socket\r\n",
                __FUNCTION__, sd);
                break;
        }

        //envio de tiempos
        for (int i=0;i<4;i++){
                Cmd[i]=recv_buf[i+4];
                printf("%c",*(Cmd+i));

        }

        if ((nwrote = write(sd, times, 12)) < 0) { // comando SNDA
                xil_printf("%s: ERROR responding to client echo request. received = %d,
                written = %d\r\n",
                                __FUNCTION__, n, nwrote);
                xil_printf("Closing socket %d\r\n", sd);
                break;}

        //-----------------------------------------------------------------------------------------
        /* Disable register update */
        XAxisScr_RegUpdateDisable(&AxisSwitch);

        /* Disable all MI ports */
        XAxisScr_MiPortDisableAll(&AxisSwitch);

        /* Source SI[1] to MI[0] */
        MiIndex = 1;
        SiIndex = 0;
        XAxisScr_MiPortEnable(&AxisSwitch, MiIndex, SiIndex);
```

```
            /* Enable register update */
            XAxisScr_RegUpdateEnable(&AxisSwitch);


            XGpio_DiscreteWrite(&reset_dac, 1,0);//setear gpio_1 a 0; //reset_dac
        convert_to_dac_format(signal,datosDAC,4,18);
        XGpio_DiscreteWrite(&reset_dac, 1,1);//setear gpio_1 a 1; //enable


        DMA_TRANSFER_DAC(datosDAC);


    }else if(!strncmp(Cmd, "SEQY", 4)){

            char send_buf[4]={'S','N','D','Y'};

            if ((nwrote = write(sd, send_buf, 4)) < 0) { // comando SNDA
                    xil_printf("%s: ERROR responding to client echo request.
                    received = %d, written = %d\r\n",
                                    __FUNCTION__, n, nwrote);
                    xil_printf("Closing socket %d\r\n", sd);
                    break;
            }else{ // Si se envio correctamente, recibir matriz


                    preMat = (char*)y;
                    int pointer = 0;

                    while(pointer<256){

                            if ((n = read(sd, recv_buf, RECV_BUF_SIZE)) < 0) {
                                    xil_printf("%s: error reading from socket %d,
                                    closing socket\r\n", __FUNCTION__, sd);
                                    break;
                            }else{

                                            memcpy((preMat+pointer),recv_buf,n);


                                            pointer+=n;

                                    }
                            };



                    xil_printf("\nSequence Y Loaded!\n");
                    /*for (int i=0;i<64;i++){
                            printf("%d: %.10f ,\n",i,y[i]);
                    }*/

            }
    }else if (!strncmp(Cmd, "OPCT", 4)){

            char send_buf[4]={'S','N','D','O'};

            if ((nwrote = write(sd, send_buf, 4)) < 0) { // comando SNDA
                    xil_printf("%s: ERROR responding to client echo request.
                    received = %d, written = %d\r\n",
                                    __FUNCTION__, n, nwrote);
                    xil_printf("Closing socket %d\r\n", sd);
                    break;
            }else{ // Si se envio correctamente, recibir matriz


                    preMat = (char*)opt;
                    int pointer = 0;

                    while(pointer<1){

                            if ((n = read(sd, recv_buf, RECV_BUF_SIZE)) < 0) {
                                    xil_printf("%s: error reading from socket %d,
                                    closing socket\r\n", __FUNCTION__, sd);
                                    break;
                            }else{

                                            memcpy((preMat+pointer),recv_buf,n);


                                            pointer+=n;
                                    }
                            };



                    xil_printf("\nParameters Loaded!\n");

                    printf("Sparcity: %2.0f ,\n",opt[0]);
                    printf("Tolerancia: %1.2e ,\n",opt[1]);
                    printf("Tol (SubOptimizer): %1.2e ,\n",opt[2]);
                    printf("Max. Iteraciones: %5.0f ,\n",opt[3]);
```

```
                        }
                }


        }

        /* close connection */
        close(sd);
        xil_printf("Connection closed\n");
        vTaskDelete(NULL);
}

void echo_application_thread()
{
        int sock, new_sd;
        struct sockaddr_in address, remote;
        int size;

        if ((sock = lwip_socket(AF_INET, SOCK_STREAM, 0)) < 0)
                return;

        address.sin_family = AF_INET;
        address.sin_port = htons(port_num);//puerto de escucha
        address.sin_addr.s_addr = INADDR_ANY;

        if (lwip_bind(sock, (struct sockaddr *)&address, sizeof (address)) < 0)
                return;

        lwip_listen(sock, 0);

        size = sizeof(remote);

        while (1) {
                if ((new_sd = lwip_accept(sock, (struct sockaddr *)&remote,
                (socklen_t *)&size)) > 0) {
                        netHandler = sys_thread_new("echos", process_echo_request,
                                (void*)new_sd,
                                THREAD_STACKSIZE*8,
                                DEFAULT_THREAD_PRIO);
                }
        }
}

//void multiplyAB(void* p){

void multiplyAB(){
        int opt = 0; // 0=Hw, 1=Sw
        int status;

        switch (opt){
                case 0:

                        Xil_DCacheFlushRange((u32)buffC,1024);
                        Xil_DCacheFlushRange((u32)buffA,1024);
                        Xil_DCacheFlushRange((u32)buffB,16384);

                        //XTime_GetTime(&tStart);
                        status = XAxiDma_SimpleTransfer(&AxiDma_0,
                        (u32)buffA,1024,XAXIDMA_DMA_TO_DEVICE);

                        if (status != XST_SUCCESS){
                                xil_printf("error al enviar matriz A\n");
                                break;

                        }


                        status = XAxiDma_SimpleTransfer(&AxiDma_1,(u32)buffB,
                        16384,XAXIDMA_DMA_TO_DEVICE);//ENVIAR B
                        if (status != XST_SUCCESS){
                                xil_printf("error al enviar  matriz B\n");
                                break;

                        }

                        while(XAxiDma_Busy(&AxiDma_0,XAXIDMA_DMA_TO_DEVICE)||
                        XAxiDma_Busy(&AxiDma_1,XAXIDMA_DMA_TO_DEVICE));


                        //Recibir Multiplicacion


                        status = XAxiDma_SimpleTransfer(&AxiDma_2,(u32)buffC,
                        1024,XAXIDMA_DEVICE_TO_DMA);

                        if (status != XST_SUCCESS){
                                xil_printf("error al recibir matriz C\n");
                                break;

                        }

                        while(XAxiDma_Busy(&AxiDma_2,XAXIDMA_DEVICE_TO_DMA));
```

```
                        while((XAxiDma_Busy(&AxiDma_2,XAXIDMA_DEVICE_TO_DMA))
                        ||( (XAxiDma_Busy(&AxiDma_1,XAXIDMA_DMA_TO_DEVICE))||
                        (XAxiDma_Busy(&AxiDma_0,XAXIDMA_DMA_TO_DEVICE)) ));

                        Xil_DCacheFlushRange((u32)buffC,1024);

                        break;

                case 1:


                        for (int z=0;z<256;z++){
                                buffC[z]=0;
                        }
                        for(int j=0;j<4;j++){
                                for(int i=0;i<64;i++){
                                        for(int k=0;k<64;k++){

                                                buffC[i+j*64]+=buffA[k+j*64]*buffB[64*k+i];
                                        }
                                }
                        }

                        break;

        }



        //printf("Time required: %.5f uS\n",1.0*(tEnd-tStart)/(COUNTS_PER_SECOND/1000000));

        //vTaskPrioritySet(NULL,DEFAULT_THREAD_PRIO);
        //vTaskResume(netHandler);

}

void DMA_TRANSFER_DAC(int32_t datos[1024]){
        Xil_DCacheFlushRange((u32)datos,1024*4);

        //XTime_GetTime(&tStart);

        int status = XAxiDma_SimpleTransfer(&AxiDma_3,(u32)datos,1024*4,
        XAXIDMA_DMA_TO_DEVICE);//ENVIAR A

        if (status != XST_SUCCESS){
                xil_printf("\n error al enviar senal, causa:%d\n",status);

        }else{
                xil_printf("\nEntering while \n");
                while(XAxiDma_Busy(&AxiDma_3,XAXIDMA_DMA_TO_DEVICE));
                xil_printf("Transferencia exitosa");

        }

        /* Disable register update */
        XAxisScr_RegUpdateDisable(&AxisSwitch);

        /* Disable all MI ports */
        XAxisScr_MiPortDisableAll(&AxisSwitch);

        /* Source SI[1] to MI[0] */
        MiIndex = 0;
        SiIndex = 0;
        XAxisScr_MiPortEnable(&AxisSwitch, MiIndex, SiIndex);

        /* Enable register update */
        XAxisScr_RegUpdateEnable(&AxisSwitch);
}
```

Software del Procesador (ANIHT.c)

```
/*
 * ANIHT.c
 *
 *  Created on: 22 de ene. de 2018
 *      Author: Mariano
 */

#define MAX_SPARCE 32
#include <stdlib.h>
#include <stdio.h>
#include <math.h>

#include "xil_printf.h"
#include "xgpio.h"
/*variables de testeo, reemplazar por externas
float x[1024] ={[0 ... 1023] = 0};
float A[1024*64];
*/
void obtain_p1(float y[64],float A[65536],float x[1024],float p1[64]);
void obtain_p1_fast(float y[64],float A[65536],float x[1024],
float p1[64],float r[MAX_SPARCE],int k);
```

```
void obtain_p2(float A[65536],float p1[64],float p2[1024]);
float obtain_u(float p3[],float Ar[],int k);
void obtain_p3(float x[1024],float Mu,float p2[1024],float p3[1024]);
void non_zeros(float x[1024],int k,float r[MAX_SPARCE]);
void shrinkA(float A[65536],float Ar[],int k,float r[MAX_SPARCE]);
void subOptimizer(float tol,float Ar[],float Xr[],float X[],float Y[],int k);
void shrinkX(float X[1024],float Xr[],int k,float r[MAX_SPARCE]);
float norm_dif(float x[1024],float x_1[1024]);
extern void multiplyVecHW(float A[],float B[],float C[]);
extern void findSupp(float X[1024],float r[32],int f);
void prepVec(void);
float norm(float A[],int l);
void multiplySW(float * restrict A,int af,int ac,float* restrict B,
float* restrict C,int bc);

void Transpose(float A[],int f,int c,float At[]);
void scatter(float X[],int k,float r[]);
void ANIHT(float y[64],float A[65536],int N,int k, float tol,
float x[1024],float tol_sub,int itMax,float rr[32]);

void recovery(float * restrict Dic,float * restrict r,int k,
float * restrict x,float * restrict a);

float p1[64] = {[0 ... 63] = 0};
float p2[1024] = {[0 ... 1023] = 0};
float p3[1024] = {[0 ... 1023] = 0};

extern XGpio reset_sorter;
extern void DMA_TRANSFER(uint32_t datos[1024],uint32_t support[32]);

inline void ANIHT(float y[64],float A[65536],int N,int k,
float tol,float x[1024],float tol_sub,int itMax,float rr[32]){

        float x_1 [1024] = {[0 ... 1023] = 0};
        float r[MAX_SPARCE] = {0};
        float crit = 10E6;
        float Mu = 0;
        //float tol_sub = 10E-3;
        float Ar[64*32];
        float Xr[MAX_SPARCE] = {0};
        float N2;
        int num = 0;
        int debug = 0;


        non_zeros(x,k,r); //ok

        if (debug ==1) {printf("r: ");
        for (int z=0;z<k;z++){printf("%f ",r[z]);}printf("\n\n");}

        while (crit>tol){


                if (debug ==1) {printf("\nOPTIMIZER it %d.:\n\n",num);}

                if (num!=0) obtain_p1_fast(y,A,x,p1,r,k);
                else memcpy(p1,y,256);

                if (debug ==1) {printf("p1: ")
                ;for (int z=0;z<64;z++){printf("%f ",p1[z]);}printf("\n");}

            obtain_p2(A,p1,p2);

            if (debug ==1) {printf("p2: ");for (int z=0;z<1024;z++){printf("%f
            ",p2[z]);}printf("\n\n");}

            shrinkA(A,Ar,k,r);
            shrinkX(p2,Xr,k,r);

                Mu = obtain_u(Xr,Ar,k);


                obtain_p3(x,Mu,p2,p3);
                non_zeros(p3,k,r);

                if (debug ==1) {printf("r: ");
                for (int z=0;z<k;z++){printf("%f ",r[z]);}printf("\n\n");}

                shrinkX(p3,Xr,k,r);

                if (debug ==1) {printf("Xr: ");
                for (int z=0;z<k;z++){printf("%f ",Xr[z]);}printf("\n");}

                shrinkA(A,Ar,k,r);
                if (debug ==1) {printf("Ar: ");
                for (int z=0;z<64*k;z++){printf("%f ",Ar[z]);}printf("\n\n");}
                subOptimizer(tol_sub,Ar,Xr,x_1,y,k);

                if (debug == 1){
                        printf("\nx_1: \n");
                        for(int z=0;z<k;z++){
                                printf("%f, ",x_1[z]);
                        }
                        printf("\n");
                }
```

```
                scatter(x_1,k,r);

                N2 = norm_dif(x,x_1);

                crit = N2*N2/1024;

                memcpy(x,x_1,4096);

                if (debug == 1){
                        printf("\nMu: %f\n",Mu);
                        printf("N2 = %f\n",N2 );
                        printf("crit = %3.10f\n",crit );

                }
                num++;
                if (num == itMax) break;
                //printf("crit = %f\n",crit );


        }
        memcpy(rr,r,k*4);
}

float norm_dif(float x[1024],float x_1[1024]){

        float N = 0;

        for (int i=0;i<1024;i++){
                        N += powf((x[i]-x_1[i]),2);
        }

        return sqrtf(N);

}

void obtain_p1(float y[64],float A[65536],float x[1024],float p1[64]){

        int i,j;
        int p_A = 0;
        int p_X = 0;
        float temp[64];
        float buffA[4096];

        memcpy(p1,y,256);

        for (i=0;i<16;i++){


                Transpose((A+p_A),64,64,buffA);
                multiplyVecHW((x+p_X),buffA,temp);
                //Transpose(buffT,64,64,temp); no hace falta transponer, es un vector!!

                for(j=0;j<64;j++){ // se podria mejorar por HW??
                        p1[j]-=temp[j];
                }

                p_X+=64;
                p_A+=4096;
        }

};


void obtain_p2(float A[65536],float p1[64],float p2[1024]){

        float buffer[4096];
        int i,j,k;
        int p_P2 = 0;

        for (k=0;k<16;k++){

                //no hace falta transponer entrada (ver cuaderno)

                for(j=0;j<64;j++){
                        for(i=0;i<64;i++){
                                buffer[i+64*j] = A[i+1024*j+k*64];
                        }
                }

                multiplyVecHW(p1,buffer,(p2+p_P2));

                p_P2+=64;
        }

};

float obtain_u(float p3[],float Ar[],int k){

        float mu;
        float temp[64];
        float N1,N2;

        multiplySW(Ar,64,k,p3,temp,1);// realizar por HW?
```

```
        N1 = norm(p3,k);
        N2 = norm(temp,64);
        mu = N1/N2;
        mu = powf(mu,2);

        return mu;

}

void obtain_p3(float x[1024],float Mu,float p2[1024],float p3[1024]){

        for (int i=0;i<1024;i++){

                p3[i]= x[i] + Mu*p2[i];

        }
};

void non_zeros(float x[1024],int k,float r[MAX_SPARCE]){

/*
        float temp[1024]={0};

        for(int i=0;i<1024;i++){
                temp[i]=fabsf ((x[i]));
        }
        findSupp(temp,r,k);
        */
        uint32_t temp[1024]={0};
        uint32_t supp[MAX_SPARCE]={0};

        for(int i=0;i<1024;i++){
                temp[i]=(uint32_t)(262143*fabsf ((x[i])));
        }
        XGpio_DiscreteWrite(&reset_sorter, 1,0);
        //setear gpio_1 a 0; //reset_sorter

        XGpio_DiscreteWrite(&reset_sorter, 1,1);
        //setear gpio_1 a 1; //enable
        DMA_TRANSFER(temp,supp);

        for(int i=0;i<k;i++){
                r[i]=(float)(supp[i]);
        }

        mergeSort(r,0,k);
}

void shrinkA(float A[65536],float Ar[],int k,float r[MAX_SPARCE]){

        int i,j;

        for (i=0;i<64;i++){

                for (j=0;j<k;j++){

                        Ar[j+k*i]=A[(int)(r[k-j-1])+i*1024];
                }

        }
}

void shrinkX(float X[1024],float Xr[],int k,float r[MAX_SPARCE]){

        int j;


        for (j=0;j<k;j++){

                Xr[j]=X[(int)(r[k-j-1])];
        }


}

void subOptimizer(float tol,float Ar[],float Xr[],
float X[],float Y[],int k){

        float crit = 10E6;
        float p1[64];
        float p2[k];
        float temp1[64];
        float temp2[64];
        float Art[64*k];
        float mu;
        float N1;
        float N2;
        float N3=0;
        int i,j,z;
        int num = 0;
        int debug = 0;

        while(crit>tol){

                if (debug ==1) xil_printf("\nsuboptimizer:\n");
```

```
                N3=0;
                multiplySW(Ar,64,k,Xr,temp1,1);

                for(i=0;i<64;i++){
                        p1[i]=Y[i]-temp1[i];
                }

                if (debug ==1) {printf("p1: ");
                for (int z=0;z<64;z++){printf("%f ",p1[z]);}printf("\n");}

                Transpose(Ar,64,k,Art);
                multiplySW(Art,k,64,p1,p2,1);

                if (debug ==1) {printf("p2: ");
                for (int z=0;z<k;z++){printf("%f ",p2[z]);}printf("\n");}

                multiplySW(Ar,64,k,p2,temp2,1);

                N1 = norm(p2,k);
                N2 = norm(temp2,64);
                mu = N1/N2;
                mu = powf(mu,2);

                if (debug ==1) {printf("mu: %f \n",mu);}

                for(z=0;z<k;z++){
                        X[z] = Xr[z]+mu*p2[z];
                }

                for (j=0;j<k;j++){
                                N3 += powf((Xr[j]-X[j]),2);
                }

                crit = N3/k;

                if (debug ==1) {printf("crit: %f \n",crit);}

                memcpy(Xr,X,k*4);
                num++;
                //printf("sub it N %d. crit = %f\n",num,crit);


        }

}


float norm(float A[],int l){

        float N = 0;

        N=0;
        for (int i=0;i<l;i++){
                N+=powf(A[i],2);
        }

        N = sqrtf(N);
        return N;
};

//el restric ayud al compilador gcc a optimizar la multiplicacion usando el neon
//ver app note de neon para m s detalles
void multiplySW(float * restrict A,int af,
int ac,float* restrict B,float* restrict C,int bc){//C=AxB


        for (int z=0;z<(af*bc);z++){
                C[z]=0;
        }
        for(int j=0;j<bc;j++){
                for(int i=0;i<af;i++){
                        for(int k=0;k<ac;k++){

                                C[j+i*bc]+=A[k+i*ac]*B[k*bc+j];

                        }
                }
        }

}

void Transpose(float A[],int f,int c,float At[]){

        int i,j;

        for(i=0;i<c;i++){

                for(j=0;j<f;j++){

                        At[j+i*f] = A[j*c+i];

                }
```

```
        }

}

void scatter(float X[],int k,float r[]){
        float temp[k];

        memcpy(temp,X,k*4);

        for(int i=0;i<1024;i++){
                X[i]=0;
        }

        for(int j=0;j<k;j++){
                X[(int)(r[k-j-1])]=temp[j];
        }
}

void obtain_p1_fast(float y[64],float A[65536],
float x[1024],float p1[64],float r[MAX_SPARCE],int k){
        int i,j;
        int index;

        for (i=0;i<64;i++){
                p1[i]=0;
                for (j=0;j<k;j++){
                        index = (int)r[-j-1+k];
                        p1[i]+=A[index+i*1024]*x[index];
                }
                p1[i] = y[i]-p1[i];
        }
}

inline void recovery(float * restrict Dic,
float * restrict r,int k,float * restrict x,float * restrict a){
        int i,j,l;
        int index;
        int subMc;
        int subMf;


                for(j=0;j<1024;j++){
                        x[j]=0;
                        for(l=0;l<k;l++){
                                index=(int)r[k-l-1];
                                subMc = (int)floorf((float)index/64);
                                subMf = (int)floorf((float)j/64);
                                x[j]+=Dic[(j%64)*64+65536*subMf
                                +4096*subMc+index-64*subMc]*a[index];
                        }

        }


}
```

Software del Procesador (sorting.c)

```
#include <stdio.h>
#include <stdlib.h>
#include "xil_printf.h"
#include "xtime_l.h"
//#include "BST.h"
#include "mergesort.h"

arx Ax[1024];
arx Bx[1024];
arx bufferx[1024];



void findSupp(float X[1024],float r[32],int f){

        int i;

    for (i=0;i<1024;i++){
        Ax[i].data = X[i];
        Ax[i].inx = i;
    };

    for (i =0;i<32;i++){ //ordeno los primeros 32
        mergeSortx(Ax, i*32, (i+1)*32-1);//esto se
        //puede reemplazar por HW
    }

    for (int k=0;k<5;k++){// me quedo con los 32 mas
    //grande en cada iteracion
        for (i =0;i<(16>>k);i++){
                biggest32x((Ax+i*64),bufferx);
                memcpy((Bx+i*32),bufferx,sizeof(arx)*32);
        }

        memcpy(Ax,Bx,(sizeof(arx)*(512>>k)));
```

```
    }

    //mergeSortx(Ax, 0, 1024);
    for (i = 0;i<f;i++){
        r[i]=(float)Ax[i].inx;
    }

    mergeSort(r,0,f);
}
```

Software del Procesador (mergesort.h)

```
/*
 * mergesort.h
 *
 *  Created on: 2 de feb. de 2018
 *      Author: Mariano
 */

#ifndef SRC_MERGESORT_H_
#define SRC_MERGESORT_H_

struct arreglo{
        float data;
        int inx;
};

typedef struct arreglo arx;

void merge(float arr[], int l, int m, int r);

void biggest32(float A[64],float B[32]);

void mergeSort(float arr[], int l, int r);
void printArray(float A[], int size);

void mergex(arx arr[], int l, int m, int r);

void biggest32x(arx A[64],arx B[32]);

void mergeSortx(arx arr[], int l, int r);
void findSupp(float X[1024],float r[32],int f);

#endif /* SRC_MERGESORT_H_ */
```

Software del Procesador (mergesort.h)

```
/* C program for Merge Sort */
#include<stdlib.h>
#include<stdio.h>
#include "mergesort.h"

// Merges two subarrays of arr[].
// First subarray is arr[l..m]
// Second subarray is arr[m+1..r]
void merge(float arr[], int l, int m, int r)
{
    int i, j, k;
    int n1 = m - l + 1;
    int n2 =  r - m;

    /* create temp arrays */
    float L[n1], R[n2];

    /* Copy data to temp arrays L[] and R[] */
    for (i = 0; i < n1; i++)
        L[i] = arr[l + i];
    for (j = 0; j < n2; j++)
        R[j] = arr[m + 1+ j];

    /* Merge the temp arrays back into arr[l..r]*/
    i = 0; // Initial index of first subarray
    j = 0; // Initial index of second subarray
    k = l; // Initial index of merged subarray
    while (i < n1 && j < n2)
    {
        if (L[i] >= R[j])
        {
            arr[k] = L[i];
            i++;
        }
        else
        {
            arr[k] = R[j];
            j++;
        }
        k++;
    }

    /* Copy the remaining elements of L[], if there
       are any */
```

```
    while (i < n1)
    {
        arr[k] = L[i];
        i++;
        k++;
    }

    /* Copy the remaining elements of R[], if there
       are any */
    while (j < n2)
    {
        arr[k] = R[j];
        j++;
        k++;
    }
}

/* l is for left index and r is right index of the
   sub-array of arr to be sorted */

void biggest32(float A[64],float B[32]){


        int i=0;
        int j=0;
        int k=0;

    while (k<32)
    {
        if (A[i] >= A[j+32])
        {
            B[k] = A[i];
            i++;
        }
        else
        {
            B[k] = A[j+32];
            j++;
        }
        k++;
    }
}

void mergeSort(float arr[], int l, int r)
{
    if (l < r)
    {
        // Same as (l+r)/2, but avoids overflow for
        // large l and h
        int m = l+(r-l)/2;

        // Sort first and second halves
        mergeSort(arr, l, m);
        mergeSort(arr, m+1, r);

        merge(arr, l, m, r);
    }
}

/* UTILITY FUNCTIONS */
/* Function to print an array */
void printArray(float A[], int size)
{
    int i;
    for (i=0; i < size; i++)
        printf("%d. %f\n ",i, A[i]);
    printf("\n");
}


void mergex(arx arr[], int l, int m, int r)
{
        int i, j, k;
            int n1 = m - l + 1;
            int n2 =  r - m;

            /* create temp arrays */
            arx L[n1], R[n2];

            /* Copy data to temp arrays L[] and R[] */
            for (i = 0; i < n1; i++)
                L[i] = arr[l + i];
            for (j = 0; j < n2; j++)
                R[j] = arr[m + 1+ j];

            /* Merge the temp arrays back into arr[l..r]*/
            i = 0; // Initial index of first subarray
            j = 0; // Initial index of second subarray
            k = l; // Initial index of merged subarray
            while (i < n1 && j < n2)
            {
                if (L[i].data >= R[j].data)
                {
                    arr[k] = L[i];
                    i++;
```

```
            }
            else
            {
                arr[k] = R[j];
                j++;
            }
            k++;
        }

        /* Copy the remaining elements of L[], if there
           are any */
        while (i < n1)
        {
            arr[k] = L[i];
            i++;
            k++;
        }

        /* Copy the remaining elements of R[], if there
           are any */
        while (j < n2)
        {
            arr[k] = R[j];
            j++;
            k++;
        }
}

/* l is for left index and r is right index of the
   sub-array of arr to be sorted */

void biggest32x(arx A[64],arx B[32]){

        int i=0;
        int j=0;
        int k=0;

    while (k<32)
    {
        if (A[i].data >= A[j+32].data)
        {
            B[k] = A[i];
            i++;
        }
        else
        {
            B[k] = A[j+32];
            j++;
        }
        k++;
    }
}

void mergeSortx(arx arr[], int l, int r)
{
    if (l < r)
    {
        // Same as (l+r)/2, but avoids overflow for
        // large l and h
        int m = l+(r-l)/2;

        // Sort first and second halves
        mergeSortx(arr, l, m);
        mergeSortx(arr, m+1, r);

        mergex(arr, l, m, r);
    }
}
```

Programa para probar el DAC

```
#include <stdio.h>
#include "platform.h"
#include "xil_printf.h"
#include "xaxidma.h"
#include "xparameters.h"
#include "xgpio.h"
#define PI 3.141592654
XAxiDma AxiDma_0;
void DMA_INIT();
void convert_to_dac_format(float datosIn[1024],int32_t datosOut[1024],
                           float maxValue,int res);
void DMA_TRANSFER(int32_t datos[1024]);

XGpio mode,reset;
float datosFloat [1024];
float datosFloat5 [1024]={[0 ... 1023]= 0.9};
int32_t datos[1024]= {[0 ... 1023]= 0};

int main()
{
    init_platform();
```

```
    DMA_INIT();
        XGpio_DiscreteWrite(&reset, 1,0);//setear gpio_1 a 0; //reset



        float f=17000; //frecuencia del seno de prueba
        float fs=48828.125; //frecuencia de muestreo del DAC

    for (int i=0;i<1024;i++){
        float t =1/fs;
        datosFloat[i]=.5*cos(2*PI*f*t*i);

    }
    convert_to_dac_format(datosFloat,datos,1,18);
    XGpio_DiscreteWrite(&reset, 1,1);//setear gpio_1 a 1; //enable

    for(int i=0;i<1024;i++){
        xil_printf("%d, ",datos[i]);
    }

    DMA_TRANSFER(datos);
    for(;;);

    cleanup_platform();
    return 0;
}


void DMA_INIT(){
        XAxiDma_Config *CfgPtr0;


        CfgPtr0 = XAxiDma_LookupConfig(XPAR_AXI_DMA_0_DEVICE_ID);

        if(!CfgPtr0 ){
                xil_printf("Cf not found DMA 0\n");
        }

        int Status = XAxiDma_CfgInitialize(&AxiDma_0,CfgPtr0);

        if(Status != XST_SUCCESS){
                xil_printf("DMA Init Failed.\n");
        }


        XAxiDma_IntrDisable(&AxiDma_0,XAXIDMA_IRQ_ALL_MASK,
                                XAXIDMA_DEVICE_TO_DMA);
        XAxiDma_IntrDisable(&AxiDma_0,XAXIDMA_IRQ_ALL_MASK,
                        XAXIDMA_DMA_TO_DEVICE);

        xil_printf("DMA Initialized OK\n");


        //GPIO RESET

        XGpio_Config *resetCfg;

         Status = XGpio_Initialize(&reset, XPAR_AXI_GPIO_0_DEVICE_ID);

        if(Status != XST_SUCCESS){
                xil_printf("Error al inicializar GPIO reset");
        }

        resetCfg = XGpio_LookupConfig(XPAR_AXI_GPIO_0_DEVICE_ID);

        if(!resetCfg ){
                xil_printf("Cf not found GPIO (reset)\n");
        }

         Status = XGpio_CfgInitialize(&reset, resetCfg
                ,resetCfg->BaseAddress);

        if(Status != XST_SUCCESS){
                xil_printf("GPIO (reset) Init Failed.\n");
        }

}

void DMA_TRANSFER(int32_t datos[1024]){
        Xil_DCacheFlushRange((u32)datos,1024*4); /

        //XTime_GetTime(&tStart);
        int status = XAxiDma_SimpleTransfer(&AxiDma_0,(u32)datos,
                        1024*4,XAXIDMA_DMA_TO_DEVICE);//ENVIAR A

        if (status != XST_SUCCESS){
                xil_printf("error al enviar senal\n");

        }else{


        while(XAxiDma_Busy(&AxiDma_0,XAXIDMA_DMA_TO_DEVICE));
                xil_printf("Transferencia exitosa");
        }
}
```

```
//maxValue = valor maximo de la senal,
la funcion mapea a +-0.9maxValue
//res = cantidad de bits para representar
a la senal, 18 en el caso que se use 24.

void convert_to_dac_format(float datosIn[1024],int32_t datosOut[1024],
float maxValue,int res){

        int max;

        float buffer[1024];

        memcpy(buffer,datosIn,1024*4);
        //max = (powf(2,res-1)-1)*.9;
        max = (powf(2,res-1));

        for (int i=0;i<1024;i++){
                buffer[i]/=maxValue;
                buffer[i]*=max;
                datosOut[i]=(int32_t)buffer[i];
        }
}
```

A.1.5. Python

Interfaz Gráfica

```
# -*- coding: utf-8 -*-

# Form implementation generated from reading ui file 'CS_GUI_3.ui'
#
# Created by: PyQt5 UI code generator 5.6
#
# WARNING! All changes made in this file will be lost!

from PyQt5 import QtCore, QtGui, QtWidgets

class Ui_MainWindow(object):
    def setupUi(self, MainWindow):
        MainWindow.setObjectName("MainWindow")
        MainWindow.resize(1103, 724)
        palette = QtGui.QPalette()
        brush = QtGui.QBrush(QtGui.QColor(0, 0, 0))
        brush.setStyle(QtCore.Qt.SolidPattern)
        palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.WindowText, brush)
        brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
        brush.setStyle(QtCore.Qt.SolidPattern)
        palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.Button, brush)
        brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
        brush.setStyle(QtCore.Qt.SolidPattern)
        palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.Light, brush)
        brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
        brush.setStyle(QtCore.Qt.SolidPattern)
        palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.Midlight, brush)
        brush = QtGui.QBrush(QtGui.QColor(127, 127, 127))
        brush.setStyle(QtCore.Qt.SolidPattern)
        palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.Dark, brush)
        brush = QtGui.QBrush(QtGui.QColor(170, 170, 170))
        brush.setStyle(QtCore.Qt.SolidPattern)
        palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.Mid, brush)
        brush = QtGui.QBrush(QtGui.QColor(0, 0, 0))
        brush.setStyle(QtCore.Qt.SolidPattern)
        palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.Text, brush)
        brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
        brush.setStyle(QtCore.Qt.SolidPattern)
        palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.BrightText, brush)
        brush = QtGui.QBrush(QtGui.QColor(0, 0, 0))
        brush.setStyle(QtCore.Qt.SolidPattern)
        palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.ButtonText, brush)
        brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
        brush.setStyle(QtCore.Qt.SolidPattern)
        palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.Base, brush)
        brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
        brush.setStyle(QtCore.Qt.SolidPattern)
        palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.Window, brush)
        brush = QtGui.QBrush(QtGui.QColor(0, 0, 0))
        brush.setStyle(QtCore.Qt.SolidPattern)
        palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.Shadow, brush)
        brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
        brush.setStyle(QtCore.Qt.SolidPattern)
        palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.AlternateBase, brush)
        brush = QtGui.QBrush(QtGui.QColor(255, 255, 220))
        brush.setStyle(QtCore.Qt.SolidPattern)
        palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.ToolTipBase, brush)
        brush = QtGui.QBrush(QtGui.QColor(0, 0, 0))
        brush.setStyle(QtCore.Qt.SolidPattern)
        palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.ToolTipText, brush)
        brush = QtGui.QBrush(QtGui.QColor(0, 0, 0))
        brush.setStyle(QtCore.Qt.SolidPattern)
        palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.WindowText, brush)
        brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
```

```
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.Button, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.Light, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.Midlight, brush)
brush = QtGui.QBrush(QtGui.QColor(127, 127, 127))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.Dark, brush)
brush = QtGui.QBrush(QtGui.QColor(170, 170, 170))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.Mid, brush)
brush = QtGui.QBrush(QtGui.QColor(0, 0, 0))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.Text, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.BrightText, brush)
brush = QtGui.QBrush(QtGui.QColor(0, 0, 0))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.ButtonText, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.Base, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.Window, brush)
brush = QtGui.QBrush(QtGui.QColor(0, 0, 0))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.Shadow, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.AlternateBase, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 220))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.ToolTipBase, brush)
brush = QtGui.QBrush(QtGui.QColor(0, 0, 0))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.ToolTipText, brush)
brush = QtGui.QBrush(QtGui.QColor(127, 127, 127))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.WindowText, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.Button, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.Light, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.Midlight, brush)
brush = QtGui.QBrush(QtGui.QColor(127, 127, 127))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.Dark, brush)
brush = QtGui.QBrush(QtGui.QColor(170, 170, 170))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.Mid, brush)
brush = QtGui.QBrush(QtGui.QColor(127, 127, 127))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.Text, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.BrightText, brush)
brush = QtGui.QBrush(QtGui.QColor(127, 127, 127))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.ButtonText, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.Base, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.Window, brush)
brush = QtGui.QBrush(QtGui.QColor(0, 0, 0))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.Shadow, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.AlternateBase, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 220))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.ToolTipBase, brush)
brush = QtGui.QBrush(QtGui.QColor(0, 0, 0))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.ToolTipText, brush)
MainWindow.setPalette(palette)
font = QtGui.QFont()
font.setFamily("Arial")
font.setBold(True)
font.setWeight(75)
MainWindow.setFont(font)
MainWindow.setAutoFillBackground(True)
MainWindow.setInputMethodHints(QtCore.Qt.ImhDigitsOnly)
self.centralwidget = QtWidgets.QWidget(MainWindow)
```

```
self.centralwidget.setObjectName("centralwidget")
self.frame = QtWidgets.QFrame(self.centralwidget)
self.frame.setGeometry(QtCore.QRect(20, 10, 1071, 81))
self.frame.setFrameShape(QtWidgets.QFrame.Box)
self.frame.setFrameShadow(QtWidgets.QFrame.Raised)
self.frame.setLineWidth(2)
self.frame.setObjectName("frame")
self.Titulo = QtWidgets.QLabel(self.frame)
self.Titulo.setGeometry(QtCore.QRect(130, 6, 841, 71))
palette = QtGui.QPalette()
brush = QtGui.QBrush(QtGui.QColor(0, 0, 0))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.WindowText, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.Button, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.Light, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.Midlight, brush)
brush = QtGui.QBrush(QtGui.QColor(127, 127, 127))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.Dark, brush)
brush = QtGui.QBrush(QtGui.QColor(170, 170, 170))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.Mid, brush)
brush = QtGui.QBrush(QtGui.QColor(0, 0, 0))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.Text, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.BrightText, brush)
brush = QtGui.QBrush(QtGui.QColor(0, 0, 0))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.ButtonText, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.Base, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.Window, brush)
brush = QtGui.QBrush(QtGui.QColor(0, 0, 0))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.Shadow, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.AlternateBase, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 220))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.ToolTipBase, brush)
brush = QtGui.QBrush(QtGui.QColor(0, 0, 0))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.ToolTipText, brush)
brush = QtGui.QBrush(QtGui.QColor(0, 0, 0))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.WindowText, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.Button, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.Light, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.Midlight, brush)
brush = QtGui.QBrush(QtGui.QColor(127, 127, 127))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.Dark, brush)
brush = QtGui.QBrush(QtGui.QColor(170, 170, 170))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.Mid, brush)
brush = QtGui.QBrush(QtGui.QColor(0, 0, 0))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.Text, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.BrightText, brush)
brush = QtGui.QBrush(QtGui.QColor(0, 0, 0))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.ButtonText, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.Base, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.Window, brush)
brush = QtGui.QBrush(QtGui.QColor(0, 0, 0))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.Shadow, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.AlternateBase, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 220))
```

```
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.ToolTipBase, brush)
brush = QtGui.QBrush(QtGui.QColor(0, 0, 0))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.ToolTipText, brush)
brush = QtGui.QBrush(QtGui.QColor(127, 127, 127))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.WindowText, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.Button, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.Light, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.Midlight, brush)
brush = QtGui.QBrush(QtGui.QColor(127, 127, 127))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.Dark, brush)
brush = QtGui.QBrush(QtGui.QColor(170, 170, 170))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.Mid, brush)
brush = QtGui.QBrush(QtGui.QColor(127, 127, 127))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.Text, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.BrightText, brush)
brush = QtGui.QBrush(QtGui.QColor(127, 127, 127))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.ButtonText, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.Base, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.Window, brush)
brush = QtGui.QBrush(QtGui.QColor(0, 0, 0))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.Shadow, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.AlternateBase, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 220))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.ToolTipBase, brush)
brush = QtGui.QBrush(QtGui.QColor(0, 0, 0))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.ToolTipText, brush)
self.Titulo.setPalette(palette)
font = QtGui.QFont()
font.setPointSize(14)
font.setItalic(False)
font.setUnderline(False)
font.setStrikeOut(False)
self.Titulo.setFont(font)
self.Titulo.setAutoFillBackground(True)
self.Titulo.setFrameShadow(QtWidgets.QFrame.Sunken)
self.Titulo.setLineWidth(9)
self.Titulo.setTextFormat(QtCore.Qt.AutoText)
self.Titulo.setObjectName("Titulo")
self.EstadoConexion_3 = QtWidgets.QLabel(self.frame)
self.EstadoConexion_3.setGeometry(QtCore.QRect(30, 22, 141, 41))
palette = QtGui.QPalette()
brush = QtGui.QBrush(QtGui.QColor(0, 0, 0))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.WindowText, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.Button, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.Light, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.Midlight, brush)
brush = QtGui.QBrush(QtGui.QColor(127, 127, 127))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.Dark, brush)
brush = QtGui.QBrush(QtGui.QColor(170, 170, 170))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.Mid, brush)
brush = QtGui.QBrush(QtGui.QColor(0, 0, 0))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.Text, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.BrightText, brush)
brush = QtGui.QBrush(QtGui.QColor(0, 0, 0))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.ButtonText, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.Base, brush)
```

```
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.Window, brush)
brush = QtGui.QBrush(QtGui.QColor(0, 0, 0))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.Shadow, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.AlternateBase, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 220))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.ToolTipBase, brush)
brush = QtGui.QBrush(QtGui.QColor(0, 0, 0))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.ToolTipText, brush)
brush = QtGui.QBrush(QtGui.QColor(0, 0, 0))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.WindowText, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.Button, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.Light, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.Midlight, brush)
brush = QtGui.QBrush(QtGui.QColor(127, 127, 127))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.Dark, brush)
brush = QtGui.QBrush(QtGui.QColor(170, 170, 170))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.Mid, brush)
brush = QtGui.QBrush(QtGui.QColor(0, 0, 0))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.Text, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.BrightText, brush)
brush = QtGui.QBrush(QtGui.QColor(0, 0, 0))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.ButtonText, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.Base, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.Window, brush)
brush = QtGui.QBrush(QtGui.QColor(0, 0, 0))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.Shadow, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.AlternateBase, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 220))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.ToolTipBase, brush)
brush = QtGui.QBrush(QtGui.QColor(0, 0, 0))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.ToolTipText, brush)
brush = QtGui.QBrush(QtGui.QColor(127, 127, 127))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.WindowText, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.Button, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.Light, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.Midlight, brush)
brush = QtGui.QBrush(QtGui.QColor(127, 127, 127))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.Dark, brush)
brush = QtGui.QBrush(QtGui.QColor(170, 170, 170))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.Mid, brush)
brush = QtGui.QBrush(QtGui.QColor(127, 127, 127))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.Text, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.BrightText, brush)
brush = QtGui.QBrush(QtGui.QColor(127, 127, 127))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.ButtonText, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.Base, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.Window, brush)
brush = QtGui.QBrush(QtGui.QColor(0, 0, 0))
brush.setStyle(QtCore.Qt.SolidPattern)
```

```
palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.Shadow, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.AlternateBase, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 220))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.ToolTipBase, brush)
brush = QtGui.QBrush(QtGui.QColor(0, 0, 0))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.ToolTipText, brush)
self.EstadoConexion_3.setPalette(palette)
self.EstadoConexion_3.setAutoFillBackground(True)
self.EstadoConexion_3.setLineWidth(9)
self.EstadoConexion_3.setObjectName("EstadoConexion_3")
self.label = QtWidgets.QLabel(self.frame)
self.label.setGeometry(QtCore.QRect(1000, 6, 61, 71))
self.label.setFrameShape(QtWidgets.QFrame.NoFrame)
self.label.setText("")
self.label.setPixmap(QtGui.QPixmap(":/newPrefix/RMPI_chip_small.png"))
self.label.setScaledContents(True)
self.label.setWordWrap(False)
self.label.setObjectName("label")
self.layoutWidget = QtWidgets.QWidget(self.centralwidget)
self.layoutWidget.setGeometry(QtCore.QRect(0, 0, 2, 2))
self.layoutWidget.setObjectName("layoutWidget")
self.verticalLayout_2 = QtWidgets.QVBoxLayout(self.layoutWidget)
self.verticalLayout_2.setContentsMargins(0, 0, 0, 0)
self.verticalLayout_2.setObjectName("verticalLayout_2")
self.layoutWidget1 = QtWidgets.QWidget(self.centralwidget)
self.layoutWidget1.setGeometry(QtCore.QRect(0, 0, 2, 2))
self.layoutWidget1.setObjectName("layoutWidget1")
self.verticalLayout_4 = QtWidgets.QVBoxLayout(self.layoutWidget1)
self.verticalLayout_4.setContentsMargins(0, 0, 0, 0)
self.verticalLayout_4.setObjectName("verticalLayout_4")
self.RMPI = QtWidgets.QTabWidget(self.centralwidget)
self.RMPI.setGeometry(QtCore.QRect(20, 103, 731, 581))
self.RMPI.setTabShape(QtWidgets.QTabWidget.Rounded)
self.RMPI.setObjectName("RMPI")
self.tab = QtWidgets.QWidget()
self.tab.setObjectName("tab")
self.graphicsView = PlotWidget(self.tab)
self.graphicsView.setGeometry(QtCore.QRect(10, 20, 481, 231))
self.graphicsView.setObjectName("graphicsView")
self.graphicsView_2 = PlotWidget(self.tab)
self.graphicsView_2.setGeometry(QtCore.QRect(10, 290, 481, 231))
self.graphicsView_2.setObjectName("graphicsView_2")
self.groupBox_4 = QtWidgets.QGroupBox(self.tab)
self.groupBox_4.setEnabled(True)
self.groupBox_4.setGeometry(QtCore.QRect(510, 10, 201, 161))
self.groupBox_4.setAutoFillBackground(False)
self.groupBox_4.setObjectName("groupBox_4")
self.label_7 = QtWidgets.QLabel(self.groupBox_4)
self.label_7.setGeometry(QtCore.QRect(240, 10, 61, 61))
self.label_7.setText("")
self.label_7.setPixmap(QtGui.QPixmap(":/newPrefix/ZedBoard_RevA_sideA_0_0 (1)_0.jpg"))
self.label_7.setScaledContents(True)
self.label_7.setObjectName("label_7")
self.layoutWidget_5 = QtWidgets.QWidget(self.groupBox_4)
self.layoutWidget_5.setGeometry(QtCore.QRect(10, 30, 184, 116))
self.layoutWidget_5.setObjectName("layoutWidget_5")
self.verticalLayout_6 = QtWidgets.QVBoxLayout(self.layoutWidget_5)
self.verticalLayout_6.setContentsMargins(0, 0, 0, 0)
self.verticalLayout_6.setObjectName("verticalLayout_6")
self.horizontalLayout_8 = QtWidgets.QHBoxLayout()
self.horizontalLayout_8.setObjectName("horizontalLayout_8")
self.EstadoConexion_9 = QtWidgets.QLabel(self.layoutWidget_5)
palette = QtGui.QPalette()
brush = QtGui.QBrush(QtGui.QColor(0, 0, 0))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.WindowText, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.Button, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.Light, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.Midlight, brush)
brush = QtGui.QBrush(QtGui.QColor(127, 127, 127))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.Dark, brush)
brush = QtGui.QBrush(QtGui.QColor(170, 170, 170))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.Mid, brush)
brush = QtGui.QBrush(QtGui.QColor(0, 0, 0))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.Text, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.BrightText, brush)
brush = QtGui.QBrush(QtGui.QColor(0, 0, 0))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.ButtonText, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
```

```
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.Base, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.Window, brush)
brush = QtGui.QBrush(QtGui.QColor(0, 0, 0))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.Shadow, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.AlternateBase, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 220))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.ToolTipBase, brush)
brush = QtGui.QBrush(QtGui.QColor(0, 0, 0))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.ToolTipText, brush)
brush = QtGui.QBrush(QtGui.QColor(0, 0, 0))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.WindowText, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.Button, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.Light, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.Midlight, brush)
brush = QtGui.QBrush(QtGui.QColor(127, 127, 127))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.Dark, brush)
brush = QtGui.QBrush(QtGui.QColor(170, 170, 170))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.Mid, brush)
brush = QtGui.QBrush(QtGui.QColor(0, 0, 0))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.Text, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.BrightText, brush)
brush = QtGui.QBrush(QtGui.QColor(0, 0, 0))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.ButtonText, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.Base, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.Window, brush)
brush = QtGui.QBrush(QtGui.QColor(0, 0, 0))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.Shadow, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.AlternateBase, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 220))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.ToolTipBase, brush)
brush = QtGui.QBrush(QtGui.QColor(0, 0, 0))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.ToolTipText, brush)
brush = QtGui.QBrush(QtGui.QColor(127, 127, 127))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.WindowText, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.Button, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.Light, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.Midlight, brush)
brush = QtGui.QBrush(QtGui.QColor(127, 127, 127))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.Dark, brush)
brush = QtGui.QBrush(QtGui.QColor(170, 170, 170))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.Mid, brush)
brush = QtGui.QBrush(QtGui.QColor(127, 127, 127))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.Text, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.BrightText, brush)
brush = QtGui.QBrush(QtGui.QColor(127, 127, 127))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.ButtonText, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.Base, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.Window, brush)
```

```
brush = QtGui.QBrush(QtGui.QColor(0, 0, 0))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.Shadow, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.AlternateBase, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 220))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.ToolTipBase, brush)
brush = QtGui.QBrush(QtGui.QColor(0, 0, 0))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.ToolTipText, brush)
self.EstadoConexion_9.setPalette(palette)
self.EstadoConexion_9.setAutoFillBackground(True)
self.EstadoConexion_9.setFrameShadow(QtWidgets.QFrame.Plain)
self.EstadoConexion_9.setLineWidth(9)
self.EstadoConexion_9.setObjectName("EstadoConexion_9")
self.horizontalLayout_8.addWidget(self.EstadoConexion_9)
self.horizontalLayout_21 = QtWidgets.QHBoxLayout()
self.horizontalLayout_21.setObjectName("horizontalLayout_21")
self.matrizAt = QtWidgets.QLabel(self.layoutWidget_5)
palette = QtGui.QPalette()
brush = QtGui.QBrush(QtGui.QColor(0, 0, 0))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.WindowText, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.Button, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.Light, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.Midlight, brush)
brush = QtGui.QBrush(QtGui.QColor(127, 127, 127))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.Dark, brush)
brush = QtGui.QBrush(QtGui.QColor(170, 170, 170))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.Mid, brush)
brush = QtGui.QBrush(QtGui.QColor(0, 0, 0))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.Text, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.BrightText, brush)
brush = QtGui.QBrush(QtGui.QColor(0, 0, 0))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.ButtonText, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.Base, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.Window, brush)
brush = QtGui.QBrush(QtGui.QColor(0, 0, 0))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.Shadow, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.AlternateBase, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 220))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.ToolTipBase, brush)
brush = QtGui.QBrush(QtGui.QColor(0, 0, 0))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.ToolTipText, brush)
brush = QtGui.QBrush(QtGui.QColor(0, 0, 0))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.WindowText, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.Button, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.Light, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.Midlight, brush)
brush = QtGui.QBrush(QtGui.QColor(127, 127, 127))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.Dark, brush)
brush = QtGui.QBrush(QtGui.QColor(170, 170, 170))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.Mid, brush)
brush = QtGui.QBrush(QtGui.QColor(0, 0, 0))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.Text, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.BrightText, brush)
brush = QtGui.QBrush(QtGui.QColor(0, 0, 0))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.ButtonText, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
```

```
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.Base, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.Window, brush)
brush = QtGui.QBrush(QtGui.QColor(0, 0, 0))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.Shadow, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.AlternateBase, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 220))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.ToolTipBase, brush)
brush = QtGui.QBrush(QtGui.QColor(0, 0, 0))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.ToolTipText, brush)
brush = QtGui.QBrush(QtGui.QColor(127, 127, 127))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.WindowText, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.Button, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.Light, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.Midlight, brush)
brush = QtGui.QBrush(QtGui.QColor(127, 127, 127))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.Dark, brush)
brush = QtGui.QBrush(QtGui.QColor(170, 170, 170))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.Mid, brush)
brush = QtGui.QBrush(QtGui.QColor(127, 127, 127))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.Text, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.BrightText, brush)
brush = QtGui.QBrush(QtGui.QColor(127, 127, 127))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.ButtonText, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.Base, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.Window, brush)
brush = QtGui.QBrush(QtGui.QColor(0, 0, 0))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.Shadow, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.AlternateBase, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 220))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.ToolTipBase, brush)
brush = QtGui.QBrush(QtGui.QColor(0, 0, 0))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.ToolTipText, brush)
self.matrizAt.setPalette(palette)
self.matrizAt.setAutoFillBackground(True)
self.matrizAt.setFrameShape(QtWidgets.QFrame.WinPanel)
self.matrizAt.setLineWidth(9)
self.matrizAt.setObjectName("matrizAt")
self.horizontalLayout_21.addWidget(self.matrizAt)
self.EstadoConexion_22 = QtWidgets.QLabel(self.layoutWidget_5)
palette = QtGui.QPalette()
brush = QtGui.QBrush(QtGui.QColor(0, 0, 0))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.WindowText, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.Button, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.Light, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.Midlight, brush)
brush = QtGui.QBrush(QtGui.QColor(127, 127, 127))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.Dark, brush)
brush = QtGui.QBrush(QtGui.QColor(170, 170, 170))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.Mid, brush)
brush = QtGui.QBrush(QtGui.QColor(0, 0, 0))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.Text, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.BrightText, brush)
brush = QtGui.QBrush(QtGui.QColor(0, 0, 0))
```

```
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.ButtonText, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.Base, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.Window, brush)
brush = QtGui.QBrush(QtGui.QColor(0, 0, 0))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.Shadow, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.AlternateBase, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 220))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.ToolTipBase, brush)
brush = QtGui.QBrush(QtGui.QColor(0, 0, 0))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.ToolTipText, brush)
brush = QtGui.QBrush(QtGui.QColor(0, 0, 0))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.WindowText, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.Button, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.Light, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.Midlight, brush)
brush = QtGui.QBrush(QtGui.QColor(127, 127, 127))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.Dark, brush)
brush = QtGui.QBrush(QtGui.QColor(170, 170, 170))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.Mid, brush)
brush = QtGui.QBrush(QtGui.QColor(0, 0, 0))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.Text, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.BrightText, brush)
brush = QtGui.QBrush(QtGui.QColor(0, 0, 0))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.ButtonText, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.Base, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.Window, brush)
brush = QtGui.QBrush(QtGui.QColor(0, 0, 0))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.Shadow, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.AlternateBase, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 220))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.ToolTipBase, brush)
brush = QtGui.QBrush(QtGui.QColor(0, 0, 0))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.ToolTipText, brush)
brush = QtGui.QBrush(QtGui.QColor(127, 127, 127))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.WindowText, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.Button, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.Light, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.Midlight, brush)
brush = QtGui.QBrush(QtGui.QColor(127, 127, 127))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.Dark, brush)
brush = QtGui.QBrush(QtGui.QColor(170, 170, 170))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.Mid, brush)
brush = QtGui.QBrush(QtGui.QColor(127, 127, 127))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.Text, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.BrightText, brush)
brush = QtGui.QBrush(QtGui.QColor(127, 127, 127))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.ButtonText, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.Base, brush)
```

```
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.Window, brush)
brush = QtGui.QBrush(QtGui.QColor(0, 0, 0))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.Shadow, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.AlternateBase, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 220))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.ToolTipBase, brush)
brush = QtGui.QBrush(QtGui.QColor(0, 0, 0))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.ToolTipText, brush)
self.EstadoConexion_22.setPalette(palette)
self.EstadoConexion_22.setAutoFillBackground(True)
self.EstadoConexion_22.setLineWidth(9)
self.EstadoConexion_22.setObjectName("EstadoConexion_22")
self.horizontalLayout_21.addWidget(self.EstadoConexion_22)
self.horizontalLayout_8.addLayout(self.horizontalLayout_21)
self.verticalLayout_6.addLayout(self.horizontalLayout_8)
self.horizontalLayout_9 = QtWidgets.QHBoxLayout()
self.horizontalLayout_9.setObjectName("horizontalLayout_9")
self.EstadoConexion_10 = QtWidgets.QLabel(self.layoutWidget_5)
palette = QtGui.QPalette()
brush = QtGui.QBrush(QtGui.QColor(0, 0, 0))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.WindowText, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.Button, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.Light, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.Midlight, brush)
brush = QtGui.QBrush(QtGui.QColor(127, 127, 127))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.Dark, brush)
brush = QtGui.QBrush(QtGui.QColor(170, 170, 170))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.Mid, brush)
brush = QtGui.QBrush(QtGui.QColor(0, 0, 0))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.Text, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.BrightText, brush)
brush = QtGui.QBrush(QtGui.QColor(0, 0, 0))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.ButtonText, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.Base, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.Window, brush)
brush = QtGui.QBrush(QtGui.QColor(0, 0, 0))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.Shadow, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.AlternateBase, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 220))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.ToolTipBase, brush)
brush = QtGui.QBrush(QtGui.QColor(0, 0, 0))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.ToolTipText, brush)
brush = QtGui.QBrush(QtGui.QColor(0, 0, 0))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.WindowText, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.Button, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.Light, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.Midlight, brush)
brush = QtGui.QBrush(QtGui.QColor(127, 127, 127))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.Dark, brush)
brush = QtGui.QBrush(QtGui.QColor(170, 170, 170))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.Mid, brush)
brush = QtGui.QBrush(QtGui.QColor(0, 0, 0))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.Text, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.BrightText, brush)
```

```
brush = QtGui.QBrush(QtGui.QColor(0, 0, 0))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.ButtonText, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.Base, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.Window, brush)
brush = QtGui.QBrush(QtGui.QColor(0, 0, 0))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.Shadow, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.AlternateBase, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 220))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.ToolTipBase, brush)
brush = QtGui.QBrush(QtGui.QColor(0, 0, 0))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.ToolTipText, brush)
brush = QtGui.QBrush(QtGui.QColor(127, 127, 127))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.WindowText, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.Button, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.Light, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.Midlight, brush)
brush = QtGui.QBrush(QtGui.QColor(127, 127, 127))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.Dark, brush)
brush = QtGui.QBrush(QtGui.QColor(170, 170, 170))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.Mid, brush)
brush = QtGui.QBrush(QtGui.QColor(127, 127, 127))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.Text, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.BrightText, brush)
brush = QtGui.QBrush(QtGui.QColor(127, 127, 127))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.ButtonText, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.Base, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.Window, brush)
brush = QtGui.QBrush(QtGui.QColor(0, 0, 0))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.Shadow, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.AlternateBase, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 220))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.ToolTipBase, brush)
brush = QtGui.QBrush(QtGui.QColor(0, 0, 0))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.ToolTipText, brush)
self.EstadoConexion_10.setPalette(palette)
self.EstadoConexion_10.setAutoFillBackground(True)
self.EstadoConexion_10.setFrameShadow(QtWidgets.QFrame.Plain)
self.EstadoConexion_10.setLineWidth(9)
self.EstadoConexion_10.setObjectName("EstadoConexion_10")
self.horizontalLayout_9.addWidget(self.EstadoConexion_10)
self.horizontalLayout_10 = QtWidgets.QHBoxLayout()
self.horizontalLayout_10.setObjectName("horizontalLayout_10")
self.optT = QtWidgets.QLabel(self.layoutWidget_5)
palette = QtGui.QPalette()
brush = QtGui.QBrush(QtGui.QColor(0, 0, 0))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.WindowText, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.Button, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.Light, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.Midlight, brush)
brush = QtGui.QBrush(QtGui.QColor(127, 127, 127))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.Dark, brush)
brush = QtGui.QBrush(QtGui.QColor(170, 170, 170))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.Mid, brush)
brush = QtGui.QBrush(QtGui.QColor(0, 0, 0))
```

```
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.Text, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.BrightText, brush)
brush = QtGui.QBrush(QtGui.QColor(0, 0, 0))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.ButtonText, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.Base, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.Window, brush)
brush = QtGui.QBrush(QtGui.QColor(0, 0, 0))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.Shadow, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.AlternateBase, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 220))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.ToolTipBase, brush)
brush = QtGui.QBrush(QtGui.QColor(0, 0, 0))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.ToolTipText, brush)
brush = QtGui.QBrush(QtGui.QColor(0, 0, 0))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.WindowText, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.Button, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.Light, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.Midlight, brush)
brush = QtGui.QBrush(QtGui.QColor(127, 127, 127))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.Dark, brush)
brush = QtGui.QBrush(QtGui.QColor(170, 170, 170))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.Mid, brush)
brush = QtGui.QBrush(QtGui.QColor(0, 0, 0))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.Text, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.BrightText, brush)
brush = QtGui.QBrush(QtGui.QColor(0, 0, 0))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.ButtonText, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.Base, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.Window, brush)
brush = QtGui.QBrush(QtGui.QColor(0, 0, 0))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.Shadow, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.AlternateBase, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 220))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.ToolTipBase, brush)
brush = QtGui.QBrush(QtGui.QColor(0, 0, 0))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.ToolTipText, brush)
brush = QtGui.QBrush(QtGui.QColor(127, 127, 127))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.WindowText, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.Button, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.Light, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.Midlight, brush)
brush = QtGui.QBrush(QtGui.QColor(127, 127, 127))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.Dark, brush)
brush = QtGui.QBrush(QtGui.QColor(170, 170, 170))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.Mid, brush)
brush = QtGui.QBrush(QtGui.QColor(127, 127, 127))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.Text, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.BrightText, brush)
```

```
brush = QtGui.QBrush(QtGui.QColor(127, 127, 127))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.ButtonText, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.Base, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.Window, brush)
brush = QtGui.QBrush(QtGui.QColor(0, 0, 0))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.Shadow, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.AlternateBase, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 220))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.ToolTipBase, brush)
brush = QtGui.QBrush(QtGui.QColor(0, 0, 0))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.ToolTipText, brush)
self.optT.setPalette(palette)
self.optT.setAutoFillBackground(True)
self.optT.setFrameShape(QtWidgets.QFrame.WinPanel)
self.optT.setLineWidth(9)
self.optT.setObjectName("optT")
self.horizontalLayout_10.addWidget(self.optT)
self.EstadoConexion_20 = QtWidgets.QLabel(self.layoutWidget_5)
palette = QtGui.QPalette()
brush = QtGui.QBrush(QtGui.QColor(0, 0, 0))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.WindowText, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.Button, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.Light, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.Midlight, brush)
brush = QtGui.QBrush(QtGui.QColor(127, 127, 127))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.Dark, brush)
brush = QtGui.QBrush(QtGui.QColor(170, 170, 170))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.Mid, brush)
brush = QtGui.QBrush(QtGui.QColor(0, 0, 0))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.Text, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.BrightText, brush)
brush = QtGui.QBrush(QtGui.QColor(0, 0, 0))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.ButtonText, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.Base, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.Window, brush)
brush = QtGui.QBrush(QtGui.QColor(0, 0, 0))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.Shadow, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.AlternateBase, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 220))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.ToolTipBase, brush)
brush = QtGui.QBrush(QtGui.QColor(0, 0, 0))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.ToolTipText, brush)
brush = QtGui.QBrush(QtGui.QColor(0, 0, 0))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.WindowText, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.Button, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.Light, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.Midlight, brush)
brush = QtGui.QBrush(QtGui.QColor(127, 127, 127))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.Dark, brush)
brush = QtGui.QBrush(QtGui.QColor(170, 170, 170))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.Mid, brush)
brush = QtGui.QBrush(QtGui.QColor(0, 0, 0))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.Text, brush)
```

```
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.BrightText, brush)
brush = QtGui.QBrush(QtGui.QColor(0, 0, 0))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.ButtonText, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.Base, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.Window, brush)
brush = QtGui.QBrush(QtGui.QColor(0, 0, 0))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.Shadow, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.AlternateBase, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 220))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.ToolTipBase, brush)
brush = QtGui.QBrush(QtGui.QColor(0, 0, 0))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.ToolTipText, brush)
brush = QtGui.QBrush(QtGui.QColor(127, 127, 127))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.WindowText, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.Button, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.Light, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.Midlight, brush)
brush = QtGui.QBrush(QtGui.QColor(127, 127, 127))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.Dark, brush)
brush = QtGui.QBrush(QtGui.QColor(170, 170, 170))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.Mid, brush)
brush = QtGui.QBrush(QtGui.QColor(127, 127, 127))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.Text, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.BrightText, brush)
brush = QtGui.QBrush(QtGui.QColor(127, 127, 127))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.ButtonText, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.Base, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.Window, brush)
brush = QtGui.QBrush(QtGui.QColor(0, 0, 0))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.Shadow, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.AlternateBase, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 220))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.ToolTipBase, brush)
brush = QtGui.QBrush(QtGui.QColor(0, 0, 0))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.ToolTipText, brush)
self.EstadoConexion_20.setPalette(palette)
self.EstadoConexion_20.setAutoFillBackground(True)
self.EstadoConexion_20.setLineWidth(9)
self.EstadoConexion_20.setObjectName("EstadoConexion_20")
self.horizontalLayout_10.addWidget(self.EstadoConexion_20)
self.horizontalLayout_9.addLayout(self.horizontalLayout_10)
self.verticalLayout_6.addLayout(self.horizontalLayout_9)
self.horizontalLayout_11 = QtWidgets.QHBoxLayout()
self.horizontalLayout_11.setObjectName("horizontalLayout_11")
self.EstadoConexion_11 = QtWidgets.QLabel(self.layoutWidget_5)
palette = QtGui.QPalette()
brush = QtGui.QBrush(QtGui.QColor(0, 0, 0))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.WindowText, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.Button, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.Light, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.Midlight, brush)
brush = QtGui.QBrush(QtGui.QColor(127, 127, 127))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.Dark, brush)
```

```
brush = QtGui.QBrush(QtGui.QColor(170, 170, 170))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.Mid, brush)
brush = QtGui.QBrush(QtGui.QColor(0, 0, 0))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.Text, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.BrightText, brush)
brush = QtGui.QBrush(QtGui.QColor(0, 0, 0))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.ButtonText, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.Base, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.Window, brush)
brush = QtGui.QBrush(QtGui.QColor(0, 0, 0))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.Shadow, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.AlternateBase, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 220))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.ToolTipBase, brush)
brush = QtGui.QBrush(QtGui.QColor(0, 0, 0))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.ToolTipText, brush)
brush = QtGui.QBrush(QtGui.QColor(0, 0, 0))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.WindowText, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.Button, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.Light, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.Midlight, brush)
brush = QtGui.QBrush(QtGui.QColor(127, 127, 127))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.Dark, brush)
brush = QtGui.QBrush(QtGui.QColor(170, 170, 170))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.Mid, brush)
brush = QtGui.QBrush(QtGui.QColor(0, 0, 0))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.Text, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.BrightText, brush)
brush = QtGui.QBrush(QtGui.QColor(0, 0, 0))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.ButtonText, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.Base, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.Window, brush)
brush = QtGui.QBrush(QtGui.QColor(0, 0, 0))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.Shadow, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.AlternateBase, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 220))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.ToolTipBase, brush)
brush = QtGui.QBrush(QtGui.QColor(0, 0, 0))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.ToolTipText, brush)
brush = QtGui.QBrush(QtGui.QColor(127, 127, 127))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.WindowText, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.Button, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.Light, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.Midlight, brush)
brush = QtGui.QBrush(QtGui.QColor(127, 127, 127))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.Dark, brush)
brush = QtGui.QBrush(QtGui.QColor(170, 170, 170))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.Mid, brush)
brush = QtGui.QBrush(QtGui.QColor(127, 127, 127))
brush.setStyle(QtCore.Qt.SolidPattern)
```

```
palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.Text, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.BrightText, brush)
brush = QtGui.QBrush(QtGui.QColor(127, 127, 127))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.ButtonText, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.Base, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.Window, brush)
brush = QtGui.QBrush(QtGui.QColor(0, 0, 0))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.Shadow, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.AlternateBase, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 220))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.ToolTipBase, brush)
brush = QtGui.QBrush(QtGui.QColor(0, 0, 0))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.ToolTipText, brush)
self.EstadoConexion_11.setPalette(palette)
self.EstadoConexion_11.setAutoFillBackground(True)
self.EstadoConexion_11.setFrameShadow(QtWidgets.QFrame.Plain)
self.EstadoConexion_11.setLineWidth(9)
self.EstadoConexion_11.setObjectName("EstadoConexion_11")
self.horizontalLayout_11.addWidget(self.EstadoConexion_11)
self.horizontalLayout_12 = QtWidgets.QHBoxLayout()
self.horizontalLayout_12.setObjectName("horizontalLayout_12")
self.recT = QtWidgets.QLabel(self.layoutWidget_5)
palette = QtGui.QPalette()
brush = QtGui.QBrush(QtGui.QColor(0, 0, 0))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.WindowText, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.Button, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.Light, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.Midlight, brush)
brush = QtGui.QBrush(QtGui.QColor(127, 127, 127))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.Dark, brush)
brush = QtGui.QBrush(QtGui.QColor(170, 170, 170))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.Mid, brush)
brush = QtGui.QBrush(QtGui.QColor(0, 0, 0))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.Text, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.BrightText, brush)
brush = QtGui.QBrush(QtGui.QColor(0, 0, 0))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.ButtonText, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.Base, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.Window, brush)
brush = QtGui.QBrush(QtGui.QColor(0, 0, 0))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.Shadow, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.AlternateBase, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 220))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.ToolTipBase, brush)
brush = QtGui.QBrush(QtGui.QColor(0, 0, 0))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.ToolTipText, brush)
brush = QtGui.QBrush(QtGui.QColor(0, 0, 0))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.WindowText, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.Button, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.Light, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.Midlight, brush)
brush = QtGui.QBrush(QtGui.QColor(127, 127, 127))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.Dark, brush)
```

```
brush = QtGui.QBrush(QtGui.QColor(170, 170, 170))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.Mid, brush)
brush = QtGui.QBrush(QtGui.QColor(0, 0, 0))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.Text, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.BrightText, brush)
brush = QtGui.QBrush(QtGui.QColor(0, 0, 0))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.ButtonText, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.Base, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.Window, brush)
brush = QtGui.QBrush(QtGui.QColor(0, 0, 0))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.Shadow, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.AlternateBase, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 220))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.ToolTipBase, brush)
brush = QtGui.QBrush(QtGui.QColor(0, 0, 0))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.ToolTipText, brush)
brush = QtGui.QBrush(QtGui.QColor(127, 127, 127))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.WindowText, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.Button, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.Light, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.Midlight, brush)
brush = QtGui.QBrush(QtGui.QColor(127, 127, 127))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.Dark, brush)
brush = QtGui.QBrush(QtGui.QColor(170, 170, 170))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.Mid, brush)
brush = QtGui.QBrush(QtGui.QColor(127, 127, 127))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.Text, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.BrightText, brush)
brush = QtGui.QBrush(QtGui.QColor(127, 127, 127))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.ButtonText, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.Base, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.Window, brush)
brush = QtGui.QBrush(QtGui.QColor(0, 0, 0))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.Shadow, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.AlternateBase, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 220))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.ToolTipBase, brush)
brush = QtGui.QBrush(QtGui.QColor(0, 0, 0))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.ToolTipText, brush)
self.recT.setPalette(palette)
self.recT.setAutoFillBackground(True)
self.recT.setFrameShape(QtWidgets.QFrame.WinPanel)
self.recT.setLineWidth(9)
self.recT.setObjectName("recT")
self.horizontalLayout_12.addWidget(self.recT)
self.EstadoConexion_21 = QtWidgets.QLabel(self.layoutWidget_5)
palette = QtGui.QPalette()
brush = QtGui.QBrush(QtGui.QColor(0, 0, 0))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.WindowText, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.Button, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.Light, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.Midlight, brush)
```

```
brush = QtGui.QBrush(QtGui.QColor(127, 127, 127))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.Dark, brush)
brush = QtGui.QBrush(QtGui.QColor(170, 170, 170))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.Mid, brush)
brush = QtGui.QBrush(QtGui.QColor(0, 0, 0))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.Text, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.BrightText, brush)
brush = QtGui.QBrush(QtGui.QColor(0, 0, 0))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.ButtonText, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.Base, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.Window, brush)
brush = QtGui.QBrush(QtGui.QColor(0, 0, 0))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.Shadow, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.AlternateBase, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 220))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.ToolTipBase, brush)
brush = QtGui.QBrush(QtGui.QColor(0, 0, 0))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.ToolTipText, brush)
brush = QtGui.QBrush(QtGui.QColor(0, 0, 0))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.WindowText, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.Button, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.Light, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.Midlight, brush)
brush = QtGui.QBrush(QtGui.QColor(127, 127, 127))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.Dark, brush)
brush = QtGui.QBrush(QtGui.QColor(170, 170, 170))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.Mid, brush)
brush = QtGui.QBrush(QtGui.QColor(0, 0, 0))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.Text, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.BrightText, brush)
brush = QtGui.QBrush(QtGui.QColor(0, 0, 0))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.ButtonText, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.Base, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.Window, brush)
brush = QtGui.QBrush(QtGui.QColor(0, 0, 0))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.Shadow, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.AlternateBase, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 220))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.ToolTipBase, brush)
brush = QtGui.QBrush(QtGui.QColor(0, 0, 0))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.ToolTipText, brush)
brush = QtGui.QBrush(QtGui.QColor(127, 127, 127))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.WindowText, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.Button, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.Light, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.Midlight, brush)
brush = QtGui.QBrush(QtGui.QColor(127, 127, 127))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.Dark, brush)
brush = QtGui.QBrush(QtGui.QColor(170, 170, 170))
brush.setStyle(QtCore.Qt.SolidPattern)
```

```
palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.Mid, brush)
brush = QtGui.QBrush(QtGui.QColor(127, 127, 127))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.Text, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.BrightText, brush)
brush = QtGui.QBrush(QtGui.QColor(127, 127, 127))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.ButtonText, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.Base, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.Window, brush)
brush = QtGui.QBrush(QtGui.QColor(0, 0, 0))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.Shadow, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.AlternateBase, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 220))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.ToolTipBase, brush)
brush = QtGui.QBrush(QtGui.QColor(0, 0, 0))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.ToolTipText, brush)
self.EstadoConexion_21.setPalette(palette)
self.EstadoConexion_21.setAutoFillBackground(True)
self.EstadoConexion_21.setLineWidth(9)
self.EstadoConexion_21.setObjectName("EstadoConexion_21")
self.horizontalLayout_12.addWidget(self.EstadoConexion_21)
self.horizontalLayout_11.addLayout(self.horizontalLayout_12)
self.verticalLayout_6.addLayout(self.horizontalLayout_11)
self.groupBox_5 = QtWidgets.QGroupBox(self.tab)
self.groupBox_5.setGeometry(QtCore.QRect(510, 280, 201, 241))
self.groupBox_5.setObjectName("groupBox_5")
self.label_8 = QtWidgets.QLabel(self.groupBox_5)
self.label_8.setGeometry(QtCore.QRect(240, 10, 61, 61))
self.label_8.setText("")
self.label_8.setPixmap(QtGui.QPixmap(":/newPrefix/ZedBoard_RevA_sideA_0_0 (1)_0.jpg"))
self.label_8.setScaledContents(True)
self.label_8.setObjectName("label_8")
self.superponer = QtWidgets.QCheckBox(self.groupBox_5)
self.superponer.setGeometry(QtCore.QRect(55, 200, 91, 18))
self.superponer.setObjectName("superponer")
self.label_13 = QtWidgets.QLabel(self.groupBox_5)
self.label_13.setGeometry(QtCore.QRect(50, 140, 101, 41))
self.label_13.setText("")
self.label_13.setPixmap(QtGui.QPixmap(":/newPrefix/similitud.PNG"))
self.label_13.setScaledContents(True)
self.label_13.setObjectName("label_13")
self.layoutWidget2 = QtWidgets.QWidget(self.groupBox_5)
self.layoutWidget2.setGeometry(QtCore.QRect(20, 50, 161, 71))
self.layoutWidget2.setObjectName("layoutWidget2")
self.horizontalLayout_22 = QtWidgets.QHBoxLayout(self.layoutWidget2)
self.horizontalLayout_22.setContentsMargins(0, 0, 0, 0)
self.horizontalLayout_22.setObjectName("horizontalLayout_22")
self.verticalLayout_7 = QtWidgets.QVBoxLayout()
self.verticalLayout_7.setObjectName("verticalLayout_7")
self.EstadoConexion_13 = QtWidgets.QLabel(self.layoutWidget2)
palette = QtGui.QPalette()
brush = QtGui.QBrush(QtGui.QColor(0, 0, 0))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.WindowText, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.Button, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.Light, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.Midlight, brush)
brush = QtGui.QBrush(QtGui.QColor(127, 127, 127))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.Dark, brush)
brush = QtGui.QBrush(QtGui.QColor(170, 170, 170))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.Mid, brush)
brush = QtGui.QBrush(QtGui.QColor(0, 0, 0))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.Text, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.BrightText, brush)
brush = QtGui.QBrush(QtGui.QColor(0, 0, 0))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.ButtonText, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.Base, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
```

```
palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.Window, brush)
brush = QtGui.QBrush(QtGui.QColor(0, 0, 0))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.Shadow, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.AlternateBase, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 220))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.ToolTipBase, brush)
brush = QtGui.QBrush(QtGui.QColor(0, 0, 0))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.ToolTipText, brush)
brush = QtGui.QBrush(QtGui.QColor(0, 0, 0))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.WindowText, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.Button, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.Light, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.Midlight, brush)
brush = QtGui.QBrush(QtGui.QColor(127, 127, 127))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.Dark, brush)
brush = QtGui.QBrush(QtGui.QColor(170, 170, 170))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.Mid, brush)
brush = QtGui.QBrush(QtGui.QColor(0, 0, 0))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.Text, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.BrightText, brush)
brush = QtGui.QBrush(QtGui.QColor(0, 0, 0))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.ButtonText, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.Base, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.Window, brush)
brush = QtGui.QBrush(QtGui.QColor(0, 0, 0))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.Shadow, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.AlternateBase, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 220))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.ToolTipBase, brush)
brush = QtGui.QBrush(QtGui.QColor(0, 0, 0))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.ToolTipText, brush)
brush = QtGui.QBrush(QtGui.QColor(127, 127, 127))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.WindowText, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.Button, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.Light, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.Midlight, brush)
brush = QtGui.QBrush(QtGui.QColor(127, 127, 127))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.Dark, brush)
brush = QtGui.QBrush(QtGui.QColor(170, 170, 170))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.Mid, brush)
brush = QtGui.QBrush(QtGui.QColor(127, 127, 127))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.Text, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.BrightText, brush)
brush = QtGui.QBrush(QtGui.QColor(127, 127, 127))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.ButtonText, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.Base, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.Window, brush)
brush = QtGui.QBrush(QtGui.QColor(0, 0, 0))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.Shadow, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
```

```
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.AlternateBase, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 220))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.ToolTipBase, brush)
brush = QtGui.QBrush(QtGui.QColor(0, 0, 0))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.ToolTipText, brush)
self.EstadoConexion_13.setPalette(palette)
self.EstadoConexion_13.setAutoFillBackground(True)
self.EstadoConexion_13.setLineWidth(9)
self.EstadoConexion_13.setObjectName("EstadoConexion_13")
self.verticalLayout_7.addWidget(self.EstadoConexion_13)
self.EstadoConexion_15 = QtWidgets.QLabel(self.layoutWidget2)
palette = QtGui.QPalette()
brush = QtGui.QBrush(QtGui.QColor(0, 0, 0))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.WindowText, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.Button, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.Light, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.Midlight, brush)
brush = QtGui.QBrush(QtGui.QColor(127, 127, 127))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.Dark, brush)
brush = QtGui.QBrush(QtGui.QColor(170, 170, 170))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.Mid, brush)
brush = QtGui.QBrush(QtGui.QColor(0, 0, 0))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.Text, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.BrightText, brush)
brush = QtGui.QBrush(QtGui.QColor(0, 0, 0))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.ButtonText, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.Base, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.Window, brush)
brush = QtGui.QBrush(QtGui.QColor(0, 0, 0))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.Shadow, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.AlternateBase, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 220))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.ToolTipBase, brush)
brush = QtGui.QBrush(QtGui.QColor(0, 0, 0))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.ToolTipText, brush)
brush = QtGui.QBrush(QtGui.QColor(0, 0, 0))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.WindowText, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.Button, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.Light, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.Midlight, brush)
brush = QtGui.QBrush(QtGui.QColor(127, 127, 127))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.Dark, brush)
brush = QtGui.QBrush(QtGui.QColor(170, 170, 170))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.Mid, brush)
brush = QtGui.QBrush(QtGui.QColor(0, 0, 0))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.Text, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.BrightText, brush)
brush = QtGui.QBrush(QtGui.QColor(0, 0, 0))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.ButtonText, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.Base, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.Window, brush)
brush = QtGui.QBrush(QtGui.QColor(0, 0, 0))
brush.setStyle(QtCore.Qt.SolidPattern)
```

```
palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.Shadow, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.AlternateBase, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 220))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.ToolTipBase, brush)
brush = QtGui.QBrush(QtGui.QColor(0, 0, 0))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.ToolTipText, brush)
brush = QtGui.QBrush(QtGui.QColor(127, 127, 127))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.WindowText, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.Button, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.Light, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.Midlight, brush)
brush = QtGui.QBrush(QtGui.QColor(127, 127, 127))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.Dark, brush)
brush = QtGui.QBrush(QtGui.QColor(170, 170, 170))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.Mid, brush)
brush = QtGui.QBrush(QtGui.QColor(127, 127, 127))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.Text, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.BrightText, brush)
brush = QtGui.QBrush(QtGui.QColor(127, 127, 127))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.ButtonText, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.Base, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.Window, brush)
brush = QtGui.QBrush(QtGui.QColor(0, 0, 0))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.Shadow, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.AlternateBase, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 220))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.ToolTipBase, brush)
brush = QtGui.QBrush(QtGui.QColor(0, 0, 0))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.ToolTipText, brush)
self.EstadoConexion_15.setPalette(palette)
self.EstadoConexion_15.setAutoFillBackground(True)
self.EstadoConexion_15.setLineWidth(9)
self.EstadoConexion_15.setObjectName("EstadoConexion_15")
self.verticalLayout_7.addWidget(self.EstadoConexion_15)
self.horizontalLayout_22.addLayout(self.verticalLayout_7)
self.verticalLayout_8 = QtWidgets.QVBoxLayout()
self.verticalLayout_8.setObjectName("verticalLayout_8")
self.simTemp = QtWidgets.QLabel(self.layoutWidget2)
palette = QtGui.QPalette()
brush = QtGui.QBrush(QtGui.QColor(0, 0, 0))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.WindowText, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.Button, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.Light, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.Midlight, brush)
brush = QtGui.QBrush(QtGui.QColor(127, 127, 127))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.Dark, brush)
brush = QtGui.QBrush(QtGui.QColor(170, 170, 170))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.Mid, brush)
brush = QtGui.QBrush(QtGui.QColor(0, 0, 0))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.Text, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.BrightText, brush)
brush = QtGui.QBrush(QtGui.QColor(0, 0, 0))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.ButtonText, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.Base, brush)
```

```
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.Window, brush)
brush = QtGui.QBrush(QtGui.QColor(0, 0, 0))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.Shadow, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.AlternateBase, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 220))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.ToolTipBase, brush)
brush = QtGui.QBrush(QtGui.QColor(0, 0, 0))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.ToolTipText, brush)
brush = QtGui.QBrush(QtGui.QColor(0, 0, 0))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.WindowText, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.Button, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.Light, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.Midlight, brush)
brush = QtGui.QBrush(QtGui.QColor(127, 127, 127))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.Dark, brush)
brush = QtGui.QBrush(QtGui.QColor(170, 170, 170))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.Mid, brush)
brush = QtGui.QBrush(QtGui.QColor(0, 0, 0))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.Text, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.BrightText, brush)
brush = QtGui.QBrush(QtGui.QColor(0, 0, 0))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.ButtonText, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.Base, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.Window, brush)
brush = QtGui.QBrush(QtGui.QColor(0, 0, 0))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.Shadow, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.AlternateBase, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 220))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.ToolTipBase, brush)
brush = QtGui.QBrush(QtGui.QColor(0, 0, 0))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.ToolTipText, brush)
brush = QtGui.QBrush(QtGui.QColor(127, 127, 127))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.WindowText, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.Button, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.Light, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.Midlight, brush)
brush = QtGui.QBrush(QtGui.QColor(127, 127, 127))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.Dark, brush)
brush = QtGui.QBrush(QtGui.QColor(170, 170, 170))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.Mid, brush)
brush = QtGui.QBrush(QtGui.QColor(127, 127, 127))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.Text, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.BrightText, brush)
brush = QtGui.QBrush(QtGui.QColor(127, 127, 127))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.ButtonText, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.Base, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.Window, brush)
brush = QtGui.QBrush(QtGui.QColor(0, 0, 0))
brush.setStyle(QtCore.Qt.SolidPattern)
```

```
palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.Shadow, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.AlternateBase, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 220))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.ToolTipBase, brush)
brush = QtGui.QBrush(QtGui.QColor(0, 0, 0))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.ToolTipText, brush)
self.simTemp.setPalette(palette)
self.simTemp.setAutoFillBackground(True)
self.simTemp.setFrameShape(QtWidgets.QFrame.WinPanel)
self.simTemp.setLineWidth(9)
self.simTemp.setObjectName("simTemp")
self.verticalLayout_8.addWidget(self.simTemp)
self.simEspec = QtWidgets.QLabel(self.layoutWidget2)
palette = QtGui.QPalette()
brush = QtGui.QBrush(QtGui.QColor(0, 0, 0))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.WindowText, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.Button, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.Light, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.Midlight, brush)
brush = QtGui.QBrush(QtGui.QColor(127, 127, 127))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.Dark, brush)
brush = QtGui.QBrush(QtGui.QColor(170, 170, 170))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.Mid, brush)
brush = QtGui.QBrush(QtGui.QColor(0, 0, 0))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.Text, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.BrightText, brush)
brush = QtGui.QBrush(QtGui.QColor(0, 0, 0))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.ButtonText, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.Base, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.Window, brush)
brush = QtGui.QBrush(QtGui.QColor(0, 0, 0))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.Shadow, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.AlternateBase, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 220))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.ToolTipBase, brush)
brush = QtGui.QBrush(QtGui.QColor(0, 0, 0))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.ToolTipText, brush)
brush = QtGui.QBrush(QtGui.QColor(0, 0, 0))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.WindowText, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.Button, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.Light, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.Midlight, brush)
brush = QtGui.QBrush(QtGui.QColor(127, 127, 127))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.Dark, brush)
brush = QtGui.QBrush(QtGui.QColor(170, 170, 170))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.Mid, brush)
brush = QtGui.QBrush(QtGui.QColor(0, 0, 0))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.Text, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.BrightText, brush)
brush = QtGui.QBrush(QtGui.QColor(0, 0, 0))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.ButtonText, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.Base, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
```

```
palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.Window, brush)
brush = QtGui.QBrush(QtGui.QColor(0, 0, 0))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.Shadow, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.AlternateBase, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 220))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.ToolTipBase, brush)
brush = QtGui.QBrush(QtGui.QColor(0, 0, 0))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.ToolTipText, brush)
brush = QtGui.QBrush(QtGui.QColor(127, 127, 127))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.WindowText, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.Button, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.Light, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.Midlight, brush)
brush = QtGui.QBrush(QtGui.QColor(127, 127, 127))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.Dark, brush)
brush = QtGui.QBrush(QtGui.QColor(170, 170, 170))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.Mid, brush)
brush = QtGui.QBrush(QtGui.QColor(127, 127, 127))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.Text, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.BrightText, brush)
brush = QtGui.QBrush(QtGui.QColor(127, 127, 127))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.ButtonText, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.Base, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.Window, brush)
brush = QtGui.QBrush(QtGui.QColor(0, 0, 0))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.Shadow, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.AlternateBase, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 220))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.ToolTipBase, brush)
brush = QtGui.QBrush(QtGui.QColor(0, 0, 0))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.ToolTipText, brush)
self.simEspec.setPalette(palette)
self.simEspec.setAutoFillBackground(True)
self.simEspec.setFrameShape(QtWidgets.QFrame.WinPanel)
self.simEspec.setLineWidth(9)
self.simEspec.setObjectName("simEspec")
self.verticalLayout_8.addWidget(self.simEspec)
self.horizontalLayout_22.addLayout(self.verticalLayout_8)
self.layoutWidget.raise_()
self.label_8.raise_()
self.superponer.raise_()
self.label_13.raise_()
self.label_9 = QtWidgets.QLabel(self.tab)
self.label_9.setGeometry(QtCore.QRect(190, 260, 81, 16))
self.label_9.setObjectName("label_9")
self.label_10 = QtWidgets.QLabel(self.tab)
self.label_10.setGeometry(QtCore.QRect(160, 530, 151, 20))
self.label_10.setObjectName("label_10")
self.layoutWidget3 = QtWidgets.QWidget(self.tab)
self.layoutWidget3.setGeometry(QtCore.QRect(340, 260, 150, 20))
self.layoutWidget3.setObjectName("layoutWidget3")
self.horizontalLayout_14 = QtWidgets.QHBoxLayout(self.layoutWidget3)
self.horizontalLayout_14.setContentsMargins(0, 0, 0, 0)
self.horizontalLayout_14.setObjectName("horizontalLayout_14")
self.espectroCheck = QtWidgets.QCheckBox(self.layoutWidget3)
self.espectroCheck.setObjectName("espectroCheck")
self.horizontalLayout_14.addWidget(self.espectroCheck)
self.interCheck = QtWidgets.QCheckBox(self.layoutWidget3)
self.interCheck.setObjectName("interCheck")
self.horizontalLayout_14.addWidget(self.interCheck)
self.groupBox_6 = QtWidgets.QGroupBox(self.tab)
self.groupBox_6.setGeometry(QtCore.QRect(510, 174, 201, 100))
self.groupBox_6.setObjectName("groupBox_6")
self.checkBox_3 = QtWidgets.QCheckBox(self.groupBox_6)
self.checkBox_3.setGeometry(QtCore.QRect(10, 40, 91, 30))
self.checkBox_3.setChecked(True)
self.checkBox_3.setObjectName("checkBox_3")
self.checkBox_4 = QtWidgets.QCheckBox(self.groupBox_6)
```

```
self.checkBox_4.setGeometry(QtCore.QRect(10, 74, 91, 16))
self.checkBox_4.setChecked(True)
self.checkBox_4.setObjectName("checkBox_4")
self.checkBox_5 = QtWidgets.QCheckBox(self.groupBox_6)
self.checkBox_5.setGeometry(QtCore.QRect(10, 14, 91, 30))
self.checkBox_5.setChecked(True)
self.checkBox_5.setObjectName("checkBox_5")
self.zoom = QtWidgets.QPushButton(self.tab)
self.zoom.setGeometry(QtCore.QRect(10, 259, 61, 23))
self.zoom.setObjectName("zoom")
self.layoutWidget.raise_()
self.graphicsView.raise_()
self.graphicsView_2.raise_()
self.groupBox_4.raise_()
self.groupBox_5.raise_()
self.label_9.raise_()
self.label_10.raise_()
self.groupBox_6.raise_()
self.zoom.raise_()
self.RMPI.addTab(self.tab, "")
self.tab_2 = QtWidgets.QWidget()
self.tab_2.setObjectName("tab_2")
self.graphicsView_3 = PlotWidget(self.tab_2)
self.graphicsView_3.setGeometry(QtCore.QRect(10, 290, 481, 231))
self.graphicsView_3.setObjectName("graphicsView_3")
self.label_11 = QtWidgets.QLabel(self.tab_2)
self.label_11.setGeometry(QtCore.QRect(190, 260, 101, 16))
self.label_11.setObjectName("label_11")
self.layoutWidget_6 = QtWidgets.QWidget(self.tab_2)
self.layoutWidget_6.setGeometry(QtCore.QRect(340, 260, 150, 20))
self.layoutWidget_6.setObjectName("layoutWidget_6")
self.horizontalLayout_17 = QtWidgets.QHBoxLayout(self.layoutWidget_6)
self.horizontalLayout_17.setContentsMargins(0, 0, 0, 0)
self.horizontalLayout_17.setObjectName("horizontalLayout_17")
self.checkBox_9 = QtWidgets.QCheckBox(self.layoutWidget_6)
self.checkBox_9.setObjectName("checkBox_9")
self.horizontalLayout_17.addWidget(self.checkBox_9)
self.interpolar = QtWidgets.QCheckBox(self.layoutWidget_6)
self.interpolar.setObjectName("interpolar")
self.horizontalLayout_17.addWidget(self.interpolar)
self.label_12 = QtWidgets.QLabel(self.tab_2)
self.label_12.setGeometry(QtCore.QRect(160, 530, 151, 20))
self.label_12.setObjectName("label_12")
self.graphicsView_4 = PlotWidget(self.tab_2)
self.graphicsView_4.setGeometry(QtCore.QRect(10, 20, 481, 231))
self.graphicsView_4.setObjectName("graphicsView_4")
self.groupBox_7 = QtWidgets.QGroupBox(self.tab_2)
self.groupBox_7.setGeometry(QtCore.QRect(510, 10, 201, 241))
self.groupBox_7.setObjectName("groupBox_7")
self.layoutWidget_8 = QtWidgets.QWidget(self.groupBox_7)
self.layoutWidget_8.setGeometry(QtCore.QRect(10, 20, 182, 113))
self.layoutWidget_8.setObjectName("layoutWidget_8")
self.verticalLayout_9 = QtWidgets.QVBoxLayout(self.layoutWidget_8)
self.verticalLayout_9.setContentsMargins(0, 0, 0, 0)
self.verticalLayout_9.setObjectName("verticalLayout_9")
self.horizontalLayout_23 = QtWidgets.QHBoxLayout()
self.horizontalLayout_23.setObjectName("horizontalLayout_23")
self.EstadoConexion_14 = QtWidgets.QLabel(self.layoutWidget_8)
self.EstadoConexion_14.setEnabled(True)
sizePolicy = QtWidgets.QSizePolicy(QtWidgets.QSizePolicy.Fixed,
QtWidgets.QSizePolicy.Fixed)
sizePolicy.setHorizontalStretch(0)
sizePolicy.setVerticalStretch(0)
sizePolicy.setHeightForWidth(self.EstadoConexion_14.sizePolicy().hasHeightForWidth())
self.EstadoConexion_14.setSizePolicy(sizePolicy)
palette = QtGui.QPalette()
brush = QtGui.QBrush(QtGui.QColor(0, 0, 0))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.WindowText, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.Button, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.Light, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.Midlight, brush)
brush = QtGui.QBrush(QtGui.QColor(127, 127, 127))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.Dark, brush)
brush = QtGui.QBrush(QtGui.QColor(170, 170, 170))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.Mid, brush)
brush = QtGui.QBrush(QtGui.QColor(0, 0, 0))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.Text, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.BrightText, brush)
brush = QtGui.QBrush(QtGui.QColor(0, 0, 0))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.ButtonText, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
```

```
palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.Base, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.Window, brush)
brush = QtGui.QBrush(QtGui.QColor(0, 0, 0))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.Shadow, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.AlternateBase, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 220))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.ToolTipBase, brush)
brush = QtGui.QBrush(QtGui.QColor(0, 0, 0))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.ToolTipText, brush)
brush = QtGui.QBrush(QtGui.QColor(0, 0, 0))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.WindowText, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.Button, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.Light, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.Midlight, brush)
brush = QtGui.QBrush(QtGui.QColor(127, 127, 127))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.Dark, brush)
brush = QtGui.QBrush(QtGui.QColor(170, 170, 170))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.Mid, brush)
brush = QtGui.QBrush(QtGui.QColor(0, 0, 0))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.Text, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.BrightText, brush)
brush = QtGui.QBrush(QtGui.QColor(0, 0, 0))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.ButtonText, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.Base, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.Window, brush)
brush = QtGui.QBrush(QtGui.QColor(0, 0, 0))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.Shadow, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.AlternateBase, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 220))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.ToolTipBase, brush)
brush = QtGui.QBrush(QtGui.QColor(0, 0, 0))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.ToolTipText, brush)
brush = QtGui.QBrush(QtGui.QColor(127, 127, 127))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.WindowText, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.Button, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.Light, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.Midlight, brush)
brush = QtGui.QBrush(QtGui.QColor(127, 127, 127))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.Dark, brush)
brush = QtGui.QBrush(QtGui.QColor(170, 170, 170))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.Mid, brush)
brush = QtGui.QBrush(QtGui.QColor(127, 127, 127))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.Text, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.BrightText, brush)
brush = QtGui.QBrush(QtGui.QColor(127, 127, 127))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.ButtonText, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.Base, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.Window, brush)
brush = QtGui.QBrush(QtGui.QColor(0, 0, 0))
```

```
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.Shadow, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.AlternateBase, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 220))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.ToolTipBase, brush)
brush = QtGui.QBrush(QtGui.QColor(0, 0, 0))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.ToolTipText, brush)
self.EstadoConexion_14.setPalette(palette)
self.EstadoConexion_14.setAutoFillBackground(True)
self.EstadoConexion_14.setLineWidth(9)
self.EstadoConexion_14.setObjectName("EstadoConexion_14")
self.horizontalLayout_23.addWidget(self.EstadoConexion_14)
self.doubleSpinBox = QtWidgets.QDoubleSpinBox(self.layoutWidget_8)
self.doubleSpinBox.setMaximum(10000.0)
self.doubleSpinBox.setProperty("value", 107.99)
self.doubleSpinBox.setObjectName("doubleSpinBox")
self.horizontalLayout_23.addWidget(self.doubleSpinBox)
self.verticalLayout_9.addLayout(self.horizontalLayout_23)
self.horizontalLayout_24 = QtWidgets.QHBoxLayout()
self.horizontalLayout_24.setObjectName("horizontalLayout_24")
self.EstadoConexion_24 = QtWidgets.QLabel(self.layoutWidget_8)
palette = QtGui.QPalette()
brush = QtGui.QBrush(QtGui.QColor(0, 0, 0))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.WindowText, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.Button, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.Light, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.Midlight, brush)
brush = QtGui.QBrush(QtGui.QColor(127, 127, 127))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.Dark, brush)
brush = QtGui.QBrush(QtGui.QColor(170, 170, 170))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.Mid, brush)
brush = QtGui.QBrush(QtGui.QColor(0, 0, 0))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.Text, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.BrightText, brush)
brush = QtGui.QBrush(QtGui.QColor(0, 0, 0))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.ButtonText, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.Base, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.Window, brush)
brush = QtGui.QBrush(QtGui.QColor(0, 0, 0))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.Shadow, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.AlternateBase, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 220))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.ToolTipBase, brush)
brush = QtGui.QBrush(QtGui.QColor(0, 0, 0))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.ToolTipText, brush)
brush = QtGui.QBrush(QtGui.QColor(0, 0, 0))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.WindowText, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.Button, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.Light, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.Midlight, brush)
brush = QtGui.QBrush(QtGui.QColor(127, 127, 127))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.Dark, brush)
brush = QtGui.QBrush(QtGui.QColor(170, 170, 170))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.Mid, brush)
brush = QtGui.QBrush(QtGui.QColor(0, 0, 0))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.Text, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.BrightText, brush)
```

```
brush = QtGui.QBrush(QtGui.QColor(0, 0, 0))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.ButtonText, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.Base, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.Window, brush)
brush = QtGui.QBrush(QtGui.QColor(0, 0, 0))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.Shadow, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.AlternateBase, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 220))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.ToolTipBase, brush)
brush = QtGui.QBrush(QtGui.QColor(0, 0, 0))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.ToolTipText, brush)
brush = QtGui.QBrush(QtGui.QColor(127, 127, 127))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.WindowText, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.Button, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.Light, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.Midlight, brush)
brush = QtGui.QBrush(QtGui.QColor(127, 127, 127))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.Dark, brush)
brush = QtGui.QBrush(QtGui.QColor(170, 170, 170))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.Mid, brush)
brush = QtGui.QBrush(QtGui.QColor(127, 127, 127))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.Text, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.BrightText, brush)
brush = QtGui.QBrush(QtGui.QColor(127, 127, 127))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.ButtonText, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.Base, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.Window, brush)
brush = QtGui.QBrush(QtGui.QColor(0, 0, 0))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.Shadow, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.AlternateBase, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 220))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.ToolTipBase, brush)
brush = QtGui.QBrush(QtGui.QColor(0, 0, 0))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.ToolTipText, brush)
self.EstadoConexion_24.setPalette(palette)
self.EstadoConexion_24.setAutoFillBackground(True)
self.EstadoConexion_24.setLineWidth(9)
self.EstadoConexion_24.setObjectName("EstadoConexion_24")
self.horizontalLayout_24.addWidget(self.EstadoConexion_24)
spacerItem = QtWidgets.QSpacerItem(17, 20, QtWidgets.QSizePolicy.Fixed,
QtWidgets.QSizePolicy.Minimum)
self.horizontalLayout_24.addItem(spacerItem)
self.listWidget = QtWidgets.QListWidget(self.layoutWidget_8)
sizePolicy = QtWidgets.QSizePolicy(QtWidgets.QSizePolicy.Preferred,
QtWidgets.QSizePolicy.Preferred)
sizePolicy.setHorizontalStretch(0)
sizePolicy.setVerticalStretch(0)
sizePolicy.setHeightForWidth(self.listWidget.sizePolicy().hasHeightForWidth())
self.listWidget.setSizePolicy(sizePolicy)
self.listWidget.setObjectName("listWidget")
item = QtWidgets.QListWidgetItem()
self.listWidget.addItem(item)
item = QtWidgets.QListWidgetItem()
self.listWidget.addItem(item)
item = QtWidgets.QListWidgetItem()
self.listWidget.addItem(item)
item = QtWidgets.QListWidgetItem()
self.listWidget.addItem(item)
item = QtWidgets.QListWidgetItem()
self.listWidget.addItem(item)
item = QtWidgets.QListWidgetItem()
self.listWidget.addItem(item)
item = QtWidgets.QListWidgetItem()
```

```
self.listWidget.addItem(item)
item = QtWidgets.QListWidgetItem()
self.listWidget.addItem(item)
self.horizontalLayout_24.addWidget(self.listWidget)
self.verticalLayout_9.addLayout(self.horizontalLayout_24)
self.layoutWidget4 = QtWidgets.QWidget(self.groupBox_7)
self.layoutWidget4.setGeometry(QtCore.QRect(10, 150, 81, 83))
self.layoutWidget4.setObjectName("layoutWidget4")
self.verticalLayout_10 = QtWidgets.QVBoxLayout(self.layoutWidget4)
self.verticalLayout_10.setContentsMargins(0, 0, 0, 0)
self.verticalLayout_10.setObjectName("verticalLayout_10")
self.cargarSignal = QtWidgets.QPushButton(self.layoutWidget4)
self.cargarSignal.setObjectName("cargarSignal")
self.verticalLayout_10.addWidget(self.cargarSignal)
self.adquirir = QtWidgets.QPushButton(self.layoutWidget4)
self.adquirir.setObjectName("adquirir")
self.verticalLayout_10.addWidget(self.adquirir)
self.guardar = QtWidgets.QPushButton(self.layoutWidget4)
self.guardar.setObjectName("guardar")
self.verticalLayout_10.addWidget(self.guardar)
self.groupBox_8 = QtWidgets.QGroupBox(self.tab_2)
self.groupBox_8.setGeometry(QtCore.QRect(510, 280, 201, 241))
self.groupBox_8.setObjectName("groupBox_8")
self.layoutWidget_7 = QtWidgets.QWidget(self.groupBox_8)
self.layoutWidget_7.setGeometry(QtCore.QRect(10, 30, 181, 201))
self.layoutWidget_7.setObjectName("layoutWidget_7")
self.verticalLayout_5 = QtWidgets.QVBoxLayout(self.layoutWidget_7)
self.verticalLayout_5.setContentsMargins(0, 0, 0, 0)
self.verticalLayout_5.setObjectName("verticalLayout_5")
self.label_2 = QtWidgets.QLabel(self.layoutWidget_7)
self.label_2.setText("")
self.label_2.setPixmap(QtGui.QPixmap(":/newPrefix/FIG_2_2.png"))
self.label_2.setScaledContents(True)
self.label_2.setWordWrap(False)
self.label_2.setObjectName("label_2")
self.verticalLayout_5.addWidget(self.label_2)
self.label_3 = QtWidgets.QLabel(self.layoutWidget_7)
self.label_3.setText("")
self.label_3.setPixmap(QtGui.QPixmap(":/newPrefix/FIG_2_2.png"))
self.label_3.setScaledContents(True)
self.label_3.setWordWrap(False)
self.label_3.setObjectName("label_3")
self.verticalLayout_5.addWidget(self.label_3)
self.label_4 = QtWidgets.QLabel(self.layoutWidget_7)
self.label_4.setText("")
self.label_4.setPixmap(QtGui.QPixmap(":/newPrefix/FIG_2_2.png"))
self.label_4.setScaledContents(True)
self.label_4.setWordWrap(False)
self.label_4.setObjectName("label_4")
self.verticalLayout_5.addWidget(self.label_4)
self.label_5 = QtWidgets.QLabel(self.layoutWidget_7)
self.label_5.setText("")
self.label_5.setPixmap(QtGui.QPixmap(":/newPrefix/FIG_2_2.png"))
self.label_5.setScaledContents(True)
self.label_5.setWordWrap(False)
self.label_5.setObjectName("label_5")
self.verticalLayout_5.addWidget(self.label_5)
self.graphicsView_3.raise_()
self.label_11.raise_()
self.layoutWidget_6.raise_()
self.label_12.raise_()
self.graphicsView_4.raise_()
self.groupBox_7.raise_()
self.groupBox_8.raise_()
self.label_2.raise_()
self.RMPI.addTab(self.tab_2, "")
self.tab_3 = QtWidgets.QWidget()
self.tab_3.setObjectName("tab_3")
self.RMPI.addTab(self.tab_3, "")
self.groupBox = QtWidgets.QGroupBox(self.centralwidget)
self.groupBox.setGeometry(QtCore.QRect(760, 225, 321, 150))
palette = QtGui.QPalette()
brush = QtGui.QBrush(QtGui.QColor(0, 0, 0))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.WindowText, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.Button, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.Light, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.Midlight, brush)
brush = QtGui.QBrush(QtGui.QColor(127, 127, 127))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.Dark, brush)
brush = QtGui.QBrush(QtGui.QColor(170, 170, 170))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.Mid, brush)
brush = QtGui.QBrush(QtGui.QColor(0, 0, 0))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.Text, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
```

```
palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.BrightText, brush)
brush = QtGui.QBrush(QtGui.QColor(0, 0, 0))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.ButtonText, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.Base, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.Window, brush)
brush = QtGui.QBrush(QtGui.QColor(0, 0, 0))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.Shadow, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.AlternateBase, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 220))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.ToolTipBase, brush)
brush = QtGui.QBrush(QtGui.QColor(0, 0, 0))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.ToolTipText, brush)
brush = QtGui.QBrush(QtGui.QColor(0, 0, 0))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.WindowText, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.Button, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.Light, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.Midlight, brush)
brush = QtGui.QBrush(QtGui.QColor(127, 127, 127))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.Dark, brush)
brush = QtGui.QBrush(QtGui.QColor(170, 170, 170))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.Mid, brush)
brush = QtGui.QBrush(QtGui.QColor(0, 0, 0))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.Text, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.BrightText, brush)
brush = QtGui.QBrush(QtGui.QColor(0, 0, 0))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.ButtonText, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.Base, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.Window, brush)
brush = QtGui.QBrush(QtGui.QColor(0, 0, 0))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.Shadow, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.AlternateBase, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 220))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.ToolTipBase, brush)
brush = QtGui.QBrush(QtGui.QColor(0, 0, 0))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.ToolTipText, brush)
brush = QtGui.QBrush(QtGui.QColor(127, 127, 127))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.WindowText, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.Button, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.Light, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.Midlight, brush)
brush = QtGui.QBrush(QtGui.QColor(127, 127, 127))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.Dark, brush)
brush = QtGui.QBrush(QtGui.QColor(170, 170, 170))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.Mid, brush)
brush = QtGui.QBrush(QtGui.QColor(127, 127, 127))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.Text, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.BrightText, brush)
brush = QtGui.QBrush(QtGui.QColor(127, 127, 127))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.ButtonText, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
```

```
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.Base, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.Window, brush)
brush = QtGui.QBrush(QtGui.QColor(0, 0, 0))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.Shadow, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.AlternateBase, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 220))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.ToolTipBase, brush)
brush = QtGui.QBrush(QtGui.QColor(0, 0, 0))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.ToolTipText, brush)
self.groupBox.setPalette(palette)
self.groupBox.setObjectName("groupBox")
self.layoutWidget_3 = QtWidgets.QWidget(self.groupBox)
self.layoutWidget_3.setGeometry(QtCore.QRect(10, 32, 304, 91))
self.layoutWidget_3.setObjectName("layoutWidget_3")
self.verticalLayout = QtWidgets.QVBoxLayout(self.layoutWidget_3)
self.verticalLayout.setContentsMargins(0, 0, 0, 0)
self.verticalLayout.setObjectName("verticalLayout")
self.gridLayout_5 = QtWidgets.QGridLayout()
self.gridLayout_5.setObjectName("gridLayout_5")
self.sendSequences = QtWidgets.QPushButton(self.layoutWidget_3)
self.sendSequences.setEnabled(True)
sizePolicy = QtWidgets.QSizePolicy(QtWidgets.QSizePolicy.Minimum,
QtWidgets.QSizePolicy.Ignored)
sizePolicy.setHorizontalStretch(0)
sizePolicy.setVerticalStretch(0)
sizePolicy.setHeightForWidth(self.sendSequences.sizePolicy().hasHeightForWidth())
self.sendSequences.setSizePolicy(sizePolicy)
font = QtGui.QFont()
font.setPointSize(7)
self.sendSequences.setFont(font)
self.sendSequences.setObjectName("sendSequences")
self.gridLayout_5.addWidget(self.sendSequences, 1, 0, 1, 1)
self.openSequences = QtWidgets.QPushButton(self.layoutWidget_3)
sizePolicy = QtWidgets.QSizePolicy(QtWidgets.QSizePolicy.Minimum,
QtWidgets.QSizePolicy.Ignored)
sizePolicy.setHorizontalStretch(0)
sizePolicy.setVerticalStretch(0)
sizePolicy.setHeightForWidth(self.openSequences.sizePolicy().hasHeightForWidth())
self.openSequences.setSizePolicy(sizePolicy)
font = QtGui.QFont()
font.setPointSize(7)
self.openSequences.setFont(font)
self.openSequences.setObjectName("openSequences")
self.gridLayout_5.addWidget(self.openSequences, 0, 0, 1, 1)
self.sendDictionary = QtWidgets.QPushButton(self.layoutWidget_3)
sizePolicy = QtWidgets.QSizePolicy(QtWidgets.QSizePolicy.Minimum,
QtWidgets.QSizePolicy.Ignored)
sizePolicy.setHorizontalStretch(0)
sizePolicy.setVerticalStretch(0)
sizePolicy.setHeightForWidth(self.sendDictionary.sizePolicy().hasHeightForWidth())
self.sendDictionary.setSizePolicy(sizePolicy)
font = QtGui.QFont()
font.setPointSize(7)
self.sendDictionary.setFont(font)
self.sendDictionary.setObjectName("sendDictionary")
self.gridLayout_5.addWidget(self.sendDictionary, 1, 1, 1, 1)
self.sendSubMuestras = QtWidgets.QPushButton(self.layoutWidget_3)
sizePolicy = QtWidgets.QSizePolicy(QtWidgets.QSizePolicy.Minimum,
QtWidgets.QSizePolicy.Ignored)
sizePolicy.setHorizontalStretch(0)
sizePolicy.setVerticalStretch(0)
sizePolicy.setHeightForWidth(self.sendSubMuestras.sizePolicy().hasHeightForWidth())
self.sendSubMuestras.setSizePolicy(sizePolicy)
font = QtGui.QFont()
font.setPointSize(7)
self.sendSubMuestras.setFont(font)
self.sendSubMuestras.setObjectName("sendSubMuestras")
self.gridLayout_5.addWidget(self.sendSubMuestras, 1, 2, 1, 1)
self.openSubMuestras = QtWidgets.QPushButton(self.layoutWidget_3)
sizePolicy = QtWidgets.QSizePolicy(QtWidgets.QSizePolicy.Minimum,
QtWidgets.QSizePolicy.Ignored)
sizePolicy.setHorizontalStretch(0)
sizePolicy.setVerticalStretch(0)
sizePolicy.setHeightForWidth(self.openSubMuestras.sizePolicy().hasHeightForWidth())
self.openSubMuestras.setSizePolicy(sizePolicy)
font = QtGui.QFont()
font.setPointSize(7)
self.openSubMuestras.setFont(font)
self.openSubMuestras.setObjectName("openSubMuestras")
self.gridLayout_5.addWidget(self.openSubMuestras, 0, 2, 1, 1)
self.openDictionary = QtWidgets.QPushButton(self.layoutWidget_3)
sizePolicy = QtWidgets.QSizePolicy(QtWidgets.QSizePolicy.Minimum, QtWidgets.QSizePolicy.Ignored)
sizePolicy.setHorizontalStretch(0)
sizePolicy.setVerticalStretch(0)
sizePolicy.setHeightForWidth(self.openDictionary.sizePolicy().hasHeightForWidth())
self.openDictionary.setSizePolicy(sizePolicy)
font = QtGui.QFont()
```

```
font.setPointSize(7)
self.openDictionary.setFont(font)
self.openDictionary.setObjectName("openDictionary")
self.gridLayout_5.addWidget(self.openDictionary, 0, 1, 1, 1)
self.verticalLayout.addLayout(self.gridLayout_5)
self.label_14 = QtWidgets.QLabel(self.groupBox)
self.label_14.setGeometry(QtCore.QRect(30, 130, 47, 13))
self.label_14.setText("")
self.label_14.setScaledContents(True)
self.label_14.setObjectName("label_14")
self.groupBox_2 = QtWidgets.QGroupBox(self.centralwidget)
self.groupBox_2.setGeometry(QtCore.QRect(760, 395, 321, 270))
font = QtGui.QFont()
font.setPointSize(8)
self.groupBox_2.setFont(font)
self.groupBox_2.setObjectName("groupBox_2")
self.EstadoConexion_5 = QtWidgets.QLabel(self.groupBox_2)
self.EstadoConexion_5.setGeometry(QtCore.QRect(217, 50, 61, 21))
palette = QtGui.QPalette()
brush = QtGui.QBrush(QtGui.QColor(0, 0, 0))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.WindowText, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.Button, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.Light, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.Midlight, brush)
brush = QtGui.QBrush(QtGui.QColor(127, 127, 127))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.Dark, brush)
brush = QtGui.QBrush(QtGui.QColor(170, 170, 170))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.Mid, brush)
brush = QtGui.QBrush(QtGui.QColor(0, 0, 0))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.Text, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.BrightText, brush)
brush = QtGui.QBrush(QtGui.QColor(0, 0, 0))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.ButtonText, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.Base, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.Window, brush)
brush = QtGui.QBrush(QtGui.QColor(0, 0, 0))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.Shadow, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.AlternateBase, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 220))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.ToolTipBase, brush)
brush = QtGui.QBrush(QtGui.QColor(0, 0, 0))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.ToolTipText, brush)
brush = QtGui.QBrush(QtGui.QColor(0, 0, 0))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.WindowText, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.Button, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.Light, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.Midlight, brush)
brush = QtGui.QBrush(QtGui.QColor(127, 127, 127))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.Dark, brush)
brush = QtGui.QBrush(QtGui.QColor(170, 170, 170))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.Mid, brush)
brush = QtGui.QBrush(QtGui.QColor(0, 0, 0))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.Text, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.BrightText, brush)
brush = QtGui.QBrush(QtGui.QColor(0, 0, 0))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.ButtonText, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.Base, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
```

```
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.Window, brush)
brush = QtGui.QBrush(QtGui.QColor(0, 0, 0))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.Shadow, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.AlternateBase, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 220))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.ToolTipBase, brush)
brush = QtGui.QBrush(QtGui.QColor(0, 0, 0))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.ToolTipText, brush)
brush = QtGui.QBrush(QtGui.QColor(127, 127, 127))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.WindowText, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.Button, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.Light, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.Midlight, brush)
brush = QtGui.QBrush(QtGui.QColor(127, 127, 127))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.Dark, brush)
brush = QtGui.QBrush(QtGui.QColor(170, 170, 170))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.Mid, brush)
brush = QtGui.QBrush(QtGui.QColor(127, 127, 127))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.Text, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.BrightText, brush)
brush = QtGui.QBrush(QtGui.QColor(127, 127, 127))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.ButtonText, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.Base, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.Window, brush)
brush = QtGui.QBrush(QtGui.QColor(0, 0, 0))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.Shadow, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.AlternateBase, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 220))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.ToolTipBase, brush)
brush = QtGui.QBrush(QtGui.QColor(0, 0, 0))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.ToolTipText, brush)
self.EstadoConexion_5.setPalette(palette)
self.EstadoConexion_5.setAutoFillBackground(True)
self.EstadoConexion_5.setLineWidth(9)
self.EstadoConexion_5.setObjectName("EstadoConexion_5")
self.Reconstruct = QtWidgets.QPushButton(self.groupBox_2)
self.Reconstruct.setGeometry(QtCore.QRect(30, 210, 101, 41))
self.Reconstruct.setObjectName("Reconstruct")
self.checkBox = QtWidgets.QCheckBox(self.groupBox_2)
self.checkBox.setGeometry(QtCore.QRect(220, 106, 91, 51))
self.checkBox.setObjectName("checkBox")
self.checkBox_2 = QtWidgets.QCheckBox(self.groupBox_2)
self.checkBox_2.setGeometry(QtCore.QRect(220, 146, 91, 51))
self.checkBox_2.setObjectName("checkBox_2")
self.Actualizar = QtWidgets.QPushButton(self.groupBox_2)
self.Actualizar.setGeometry(QtCore.QRect(140, 210, 101, 41))
self.Actualizar.setObjectName("Actualizar")
self.layoutWidget_4 = QtWidgets.QWidget(self.groupBox_2)
self.layoutWidget_4.setGeometry(QtCore.QRect(33, 43, 171, 151))
self.layoutWidget_4.setObjectName("layoutWidget_4")
self.verticalLayout_3 = QtWidgets.QVBoxLayout(self.layoutWidget_4)
self.verticalLayout_3.setContentsMargins(0, 0, 0, 0)
self.verticalLayout_3.setObjectName("verticalLayout_3")
self.horizontalLayout_7 = QtWidgets.QHBoxLayout()
self.horizontalLayout_7.setObjectName("horizontalLayout_7")
self.EstadoConexion_4 = QtWidgets.QLabel(self.layoutWidget_4)
palette = QtGui.QPalette()
brush = QtGui.QBrush(QtGui.QColor(0, 0, 0))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.WindowText, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.Button, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.Light, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
```

```
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.Midlight, brush)
brush = QtGui.QBrush(QtGui.QColor(127, 127, 127))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.Dark, brush)
brush = QtGui.QBrush(QtGui.QColor(170, 170, 170))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.Mid, brush)
brush = QtGui.QBrush(QtGui.QColor(0, 0, 0))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.Text, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.BrightText, brush)
brush = QtGui.QBrush(QtGui.QColor(0, 0, 0))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.ButtonText, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.Base, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.Window, brush)
brush = QtGui.QBrush(QtGui.QColor(0, 0, 0))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.Shadow, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.AlternateBase, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 220))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.ToolTipBase, brush)
brush = QtGui.QBrush(QtGui.QColor(0, 0, 0))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.ToolTipText, brush)
brush = QtGui.QBrush(QtGui.QColor(0, 0, 0))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.WindowText, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.Button, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.Light, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.Midlight, brush)
brush = QtGui.QBrush(QtGui.QColor(127, 127, 127))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.Dark, brush)
brush = QtGui.QBrush(QtGui.QColor(170, 170, 170))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.Mid, brush)
brush = QtGui.QBrush(QtGui.QColor(0, 0, 0))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.Text, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.BrightText, brush)
brush = QtGui.QBrush(QtGui.QColor(0, 0, 0))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.ButtonText, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.Base, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.Window, brush)
brush = QtGui.QBrush(QtGui.QColor(0, 0, 0))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.Shadow, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.AlternateBase, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 220))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.ToolTipBase, brush)
brush = QtGui.QBrush(QtGui.QColor(0, 0, 0))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.ToolTipText, brush)
brush = QtGui.QBrush(QtGui.QColor(127, 127, 127))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.WindowText, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.Button, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.Light, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.Midlight, brush)
brush = QtGui.QBrush(QtGui.QColor(127, 127, 127))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.Dark, brush)
```

```
brush = QtGui.QBrush(QtGui.QColor(170, 170, 170))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.Mid, brush)
brush = QtGui.QBrush(QtGui.QColor(127, 127, 127))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.Text, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.BrightText, brush)
brush = QtGui.QBrush(QtGui.QColor(127, 127, 127))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.ButtonText, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.Base, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.Window, brush)
brush = QtGui.QBrush(QtGui.QColor(0, 0, 0))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.Shadow, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.AlternateBase, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 220))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.ToolTipBase, brush)
brush = QtGui.QBrush(QtGui.QColor(0, 0, 0))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.ToolTipText, brush)
self.EstadoConexion_4.setPalette(palette)
self.EstadoConexion_4.setAutoFillBackground(True)
self.EstadoConexion_4.setLineWidth(9)
self.EstadoConexion_4.setObjectName("EstadoConexion_4")
self.horizontalLayout_7.addWidget(self.EstadoConexion_4)
self.sparcityBox = QtWidgets.QSpinBox(self.layoutWidget_4)
self.sparcityBox.setMinimum(1)
self.sparcityBox.setMaximum(32)
self.sparcityBox.setProperty("value", 1)
self.sparcityBox.setObjectName("sparcityBox")
self.horizontalLayout_7.addWidget(self.sparcityBox)
self.verticalLayout_3.addLayout(self.horizontalLayout_7)
self.horizontalLayout_6 = QtWidgets.QHBoxLayout()
self.horizontalLayout_6.setObjectName("horizontalLayout_6")
self.EstadoConexion_6 = QtWidgets.QLabel(self.layoutWidget_4)
palette = QtGui.QPalette()
brush = QtGui.QBrush(QtGui.QColor(0, 0, 0))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.WindowText, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.Button, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.Light, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.Midlight, brush)
brush = QtGui.QBrush(QtGui.QColor(127, 127, 127))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.Dark, brush)
brush = QtGui.QBrush(QtGui.QColor(170, 170, 170))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.Mid, brush)
brush = QtGui.QBrush(QtGui.QColor(0, 0, 0))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.Text, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.BrightText, brush)
brush = QtGui.QBrush(QtGui.QColor(0, 0, 0))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.ButtonText, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.Base, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.Window, brush)
brush = QtGui.QBrush(QtGui.QColor(0, 0, 0))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.Shadow, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.AlternateBase, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 220))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.ToolTipBase, brush)
brush = QtGui.QBrush(QtGui.QColor(0, 0, 0))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.ToolTipText, brush)
brush = QtGui.QBrush(QtGui.QColor(0, 0, 0))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.WindowText, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
```

```
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.Button, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.Light, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.Midlight, brush)
brush = QtGui.QBrush(QtGui.QColor(127, 127, 127))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.Dark, brush)
brush = QtGui.QBrush(QtGui.QColor(170, 170, 170))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.Mid, brush)
brush = QtGui.QBrush(QtGui.QColor(0, 0, 0))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.Text, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.BrightText, brush)
brush = QtGui.QBrush(QtGui.QColor(0, 0, 0))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.ButtonText, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.Base, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.Window, brush)
brush = QtGui.QBrush(QtGui.QColor(0, 0, 0))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.Shadow, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.AlternateBase, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 220))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.ToolTipBase, brush)
brush = QtGui.QBrush(QtGui.QColor(0, 0, 0))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.ToolTipText, brush)
brush = QtGui.QBrush(QtGui.QColor(127, 127, 127))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.WindowText, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.Button, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.Light, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.Midlight, brush)
brush = QtGui.QBrush(QtGui.QColor(127, 127, 127))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.Dark, brush)
brush = QtGui.QBrush(QtGui.QColor(170, 170, 170))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.Mid, brush)
brush = QtGui.QBrush(QtGui.QColor(127, 127, 127))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.Text, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.BrightText, brush)
brush = QtGui.QBrush(QtGui.QColor(127, 127, 127))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.ButtonText, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.Base, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.Window, brush)
brush = QtGui.QBrush(QtGui.QColor(0, 0, 0))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.Shadow, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.AlternateBase, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 220))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.ToolTipBase, brush)
brush = QtGui.QBrush(QtGui.QColor(0, 0, 0))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.ToolTipText, brush)
self.EstadoConexion_6.setPalette(palette)
self.EstadoConexion_6.setAutoFillBackground(True)
self.EstadoConexion_6.setLineWidth(9)
self.EstadoConexion_6.setObjectName("EstadoConexion_6")
self.horizontalLayout_6.addWidget(self.EstadoConexion_6)
self.horizontalLayout_2 = QtWidgets.QHBoxLayout()
self.horizontalLayout_2.setObjectName("horizontalLayout_2")
self.tolBaseBox = QtWidgets.QSpinBox(self.layoutWidget_4)
self.tolBaseBox.setInputMethodHints(QtCore.Qt.ImhDigitsOnly)
```

```
self.tolBaseBox.setMinimum(-12)
self.tolBaseBox.setMaximum(9)
self.tolBaseBox.setSingleStep(1)
self.tolBaseBox.setProperty("value", 1)
self.tolBaseBox.setDisplayIntegerBase(10)
self.tolBaseBox.setObjectName("tolBaseBox")
self.horizontalLayout_2.addWidget(self.tolBaseBox)
self.tolExpBox = QtWidgets.QSpinBox(self.layoutWidget_4)
self.tolExpBox.setInputMethodHints(QtCore.Qt.ImhDigitsOnly)
self.tolExpBox.setMinimum(-12)
self.tolExpBox.setMaximum(0)
self.tolExpBox.setSingleStep(1)
self.tolExpBox.setProperty("value", -9)
self.tolExpBox.setDisplayIntegerBase(10)
self.tolExpBox.setObjectName("tolExpBox")
self.horizontalLayout_2.addWidget(self.tolExpBox)
self.horizontalLayout_6.addLayout(self.horizontalLayout_2)
self.verticalLayout_3.addLayout(self.horizontalLayout_6)
self.horizontalLayout_4 = QtWidgets.QHBoxLayout()
self.horizontalLayout_4.setObjectName("horizontalLayout_4")
self.EstadoConexion_7 = QtWidgets.QLabel(self.layoutWidget_4)
palette = QtGui.QPalette()
brush = QtGui.QBrush(QtGui.QColor(0, 0, 0))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.WindowText, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.Button, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.Light, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.Midlight, brush)
brush = QtGui.QBrush(QtGui.QColor(127, 127, 127))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.Dark, brush)
brush = QtGui.QBrush(QtGui.QColor(170, 170, 170))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.Mid, brush)
brush = QtGui.QBrush(QtGui.QColor(0, 0, 0))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.Text, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.BrightText, brush)
brush = QtGui.QBrush(QtGui.QColor(0, 0, 0))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.ButtonText, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.Base, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.Window, brush)
brush = QtGui.QBrush(QtGui.QColor(0, 0, 0))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.Shadow, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.AlternateBase, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 220))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.ToolTipBase, brush)
brush = QtGui.QBrush(QtGui.QColor(0, 0, 0))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.ToolTipText, brush)
brush = QtGui.QBrush(QtGui.QColor(0, 0, 0))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.WindowText, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.Button, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.Light, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.Midlight, brush)
brush = QtGui.QBrush(QtGui.QColor(127, 127, 127))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.Dark, brush)
brush = QtGui.QBrush(QtGui.QColor(170, 170, 170))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.Mid, brush)
brush = QtGui.QBrush(QtGui.QColor(0, 0, 0))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.Text, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.BrightText, brush)
brush = QtGui.QBrush(QtGui.QColor(0, 0, 0))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.ButtonText, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
```

```
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.Base, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.Window, brush)
brush = QtGui.QBrush(QtGui.QColor(0, 0, 0))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.Shadow, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.AlternateBase, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 220))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.ToolTipBase, brush)
brush = QtGui.QBrush(QtGui.QColor(0, 0, 0))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.ToolTipText, brush)
brush = QtGui.QBrush(QtGui.QColor(127, 127, 127))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.WindowText, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.Button, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.Light, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.Midlight, brush)
brush = QtGui.QBrush(QtGui.QColor(127, 127, 127))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.Dark, brush)
brush = QtGui.QBrush(QtGui.QColor(170, 170, 170))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.Mid, brush)
brush = QtGui.QBrush(QtGui.QColor(127, 127, 127))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.Text, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.BrightText, brush)
brush = QtGui.QBrush(QtGui.QColor(127, 127, 127))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.ButtonText, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.Base, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.Window, brush)
brush = QtGui.QBrush(QtGui.QColor(0, 0, 0))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.Shadow, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.AlternateBase, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 220))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.ToolTipBase, brush)
brush = QtGui.QBrush(QtGui.QColor(0, 0, 0))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.ToolTipText, brush)
self.EstadoConexion_7.setPalette(palette)
self.EstadoConexion_7.setAutoFillBackground(True)
self.EstadoConexion_7.setLineWidth(9)
self.EstadoConexion_7.setObjectName("EstadoConexion_7")
self.horizontalLayout_4.addWidget(self.EstadoConexion_7)
self.horizontalLayout = QtWidgets.QHBoxLayout()
self.horizontalLayout.setObjectName("horizontalLayout")
self.subtolBaseBox = QtWidgets.QSpinBox(self.layoutWidget_4)
self.subtolBaseBox.setInputMethodHints(QtCore.Qt.ImhDigitsOnly)
self.subtolBaseBox.setMinimum(-12)
self.subtolBaseBox.setMaximum(1)
self.subtolBaseBox.setSingleStep(1)
self.subtolBaseBox.setProperty("value", 1)
self.subtolBaseBox.setDisplayIntegerBase(10)
self.subtolBaseBox.setObjectName("subtolBaseBox")
self.horizontalLayout.addWidget(self.subtolBaseBox)
self.subtolExpBox = QtWidgets.QSpinBox(self.layoutWidget_4)
self.subtolExpBox.setInputMethodHints(QtCore.Qt.ImhDigitsOnly)
self.subtolExpBox.setMinimum(-12)
self.subtolExpBox.setMaximum(0)
self.subtolExpBox.setSingleStep(1)
self.subtolExpBox.setProperty("value", -2)
self.subtolExpBox.setDisplayIntegerBase(10)
self.subtolExpBox.setObjectName("subtolExpBox")
self.horizontalLayout.addWidget(self.subtolExpBox)
self.horizontalLayout_4.addLayout(self.horizontalLayout)
self.verticalLayout_3.addLayout(self.horizontalLayout_4)
self.horizontalLayout_3 = QtWidgets.QHBoxLayout()
self.horizontalLayout_3.setObjectName("horizontalLayout_3")
self.EstadoConexion_8 = QtWidgets.QLabel(self.layoutWidget_4)
palette = QtGui.QPalette()
brush = QtGui.QBrush(QtGui.QColor(0, 0, 0))
brush.setStyle(QtCore.Qt.SolidPattern)
```

```
palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.WindowText, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.Button, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.Light, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.Midlight, brush)
brush = QtGui.QBrush(QtGui.QColor(127, 127, 127))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.Dark, brush)
brush = QtGui.QBrush(QtGui.QColor(170, 170, 170))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.Mid, brush)
brush = QtGui.QBrush(QtGui.QColor(0, 0, 0))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.Text, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.BrightText, brush)
brush = QtGui.QBrush(QtGui.QColor(0, 0, 0))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.ButtonText, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.Base, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.Window, brush)
brush = QtGui.QBrush(QtGui.QColor(0, 0, 0))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.Shadow, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.AlternateBase, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 220))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.ToolTipBase, brush)
brush = QtGui.QBrush(QtGui.QColor(0, 0, 0))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.ToolTipText, brush)
brush = QtGui.QBrush(QtGui.QColor(0, 0, 0))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.WindowText, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.Button, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.Light, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.Midlight, brush)
brush = QtGui.QBrush(QtGui.QColor(127, 127, 127))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.Dark, brush)
brush = QtGui.QBrush(QtGui.QColor(170, 170, 170))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.Mid, brush)
brush = QtGui.QBrush(QtGui.QColor(0, 0, 0))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.Text, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.BrightText, brush)
brush = QtGui.QBrush(QtGui.QColor(0, 0, 0))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.ButtonText, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.Base, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.Window, brush)
brush = QtGui.QBrush(QtGui.QColor(0, 0, 0))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.Shadow, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.AlternateBase, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 220))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.ToolTipBase, brush)
brush = QtGui.QBrush(QtGui.QColor(0, 0, 0))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.ToolTipText, brush)
brush = QtGui.QBrush(QtGui.QColor(127, 127, 127))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.WindowText, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.Button, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
```

```
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.Light, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.Midlight, brush)
brush = QtGui.QBrush(QtGui.QColor(127, 127, 127))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.Dark, brush)
brush = QtGui.QBrush(QtGui.QColor(170, 170, 170))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.Mid, brush)
brush = QtGui.QBrush(QtGui.QColor(127, 127, 127))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.Text, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.BrightText, brush)
brush = QtGui.QBrush(QtGui.QColor(127, 127, 127))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.ButtonText, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.Base, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.Window, brush)
brush = QtGui.QBrush(QtGui.QColor(0, 0, 0))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.Shadow, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.AlternateBase, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 220))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.ToolTipBase, brush)
brush = QtGui.QBrush(QtGui.QColor(0, 0, 0))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.ToolTipText, brush)
self.EstadoConexion_8.setPalette(palette)
self.EstadoConexion_8.setAutoFillBackground(True)
self.EstadoConexion_8.setLineWidth(9)
self.EstadoConexion_8.setObjectName("EstadoConexion_8")
self.horizontalLayout_3.addWidget(self.EstadoConexion_8)
self.maxiterBox = QtWidgets.QSpinBox(self.layoutWidget_4)
self.maxiterBox.setInputMethodHints(QtCore.Qt.ImhNone)
self.maxiterBox.setMinimum(100)
self.maxiterBox.setMaximum(10000)
self.maxiterBox.setSingleStep(50)
self.maxiterBox.setProperty("value", 1000)
self.maxiterBox.setObjectName("maxiterBox")
self.horizontalLayout_3.addWidget(self.maxiterBox)
self.verticalLayout_3.addLayout(self.horizontalLayout_3)
self.groupBox_3 = QtWidgets.QGroupBox(self.centralwidget)
self.groupBox_3.setGeometry(QtCore.QRect(760, 135, 321, 80))
self.groupBox_3.setObjectName("groupBox_3")
self.layoutWidget_2 = QtWidgets.QWidget(self.groupBox_3)
self.layoutWidget_2.setGeometry(QtCore.QRect(24, 26, 171, 31))
self.layoutWidget_2.setObjectName("layoutWidget_2")
self.horizontalLayout_5 = QtWidgets.QHBoxLayout(self.layoutWidget_2)
self.horizontalLayout_5.setContentsMargins(0, 0, 0, 0)
self.horizontalLayout_5.setObjectName("horizontalLayout_5")
self.conectar = QtWidgets.QPushButton(self.layoutWidget_2)
sizePolicy = QtWidgets.QSizePolicy(QtWidgets.QSizePolicy.Minimum,
QtWidgets.QSizePolicy.Ignored)
sizePolicy.setHorizontalStretch(0)
sizePolicy.setVerticalStretch(0)
sizePolicy.setHeightForWidth(self.conectar.sizePolicy().hasHeightForWidth())
self.conectar.setSizePolicy(sizePolicy)
self.conectar.setObjectName("conectar")
self.horizontalLayout_5.addWidget(self.conectar)
self.EstadoConexion = QtWidgets.QLabel(self.layoutWidget_2)
palette = QtGui.QPalette()
brush = QtGui.QBrush(QtGui.QColor(0, 0, 0))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.WindowText, brush)
brush = QtGui.QBrush(QtGui.QColor(170, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.Button, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.Light, brush)
brush = QtGui.QBrush(QtGui.QColor(212, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.Midlight, brush)
brush = QtGui.QBrush(QtGui.QColor(85, 127, 127))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.Dark, brush)
brush = QtGui.QBrush(QtGui.QColor(113, 170, 170))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.Mid, brush)
brush = QtGui.QBrush(QtGui.QColor(0, 0, 0))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.Text, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
```

```
palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.BrightText, brush)
brush = QtGui.QBrush(QtGui.QColor(0, 0, 0))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.ButtonText, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.Base, brush)
brush = QtGui.QBrush(QtGui.QColor(170, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.Window, brush)
brush = QtGui.QBrush(QtGui.QColor(0, 0, 0))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.Shadow, brush)
brush = QtGui.QBrush(QtGui.QColor(212, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.AlternateBase, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 220))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.ToolTipBase, brush)
brush = QtGui.QBrush(QtGui.QColor(0, 0, 0))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.ToolTipText, brush)
brush = QtGui.QBrush(QtGui.QColor(0, 0, 0))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.WindowText, brush)
brush = QtGui.QBrush(QtGui.QColor(170, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.Button, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.Light, brush)
brush = QtGui.QBrush(QtGui.QColor(212, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.Midlight, brush)
brush = QtGui.QBrush(QtGui.QColor(85, 127, 127))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.Dark, brush)
brush = QtGui.QBrush(QtGui.QColor(113, 170, 170))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.Mid, brush)
brush = QtGui.QBrush(QtGui.QColor(0, 0, 0))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.Text, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.BrightText, brush)
brush = QtGui.QBrush(QtGui.QColor(0, 0, 0))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.ButtonText, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.Base, brush)
brush = QtGui.QBrush(QtGui.QColor(170, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.Window, brush)
brush = QtGui.QBrush(QtGui.QColor(0, 0, 0))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.Shadow, brush)
brush = QtGui.QBrush(QtGui.QColor(212, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.AlternateBase, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 220))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.ToolTipBase, brush)
brush = QtGui.QBrush(QtGui.QColor(0, 0, 0))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.ToolTipText, brush)
brush = QtGui.QBrush(QtGui.QColor(85, 127, 127))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.WindowText, brush)
brush = QtGui.QBrush(QtGui.QColor(170, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.Button, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.Light, brush)
brush = QtGui.QBrush(QtGui.QColor(212, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.Midlight, brush)
brush = QtGui.QBrush(QtGui.QColor(85, 127, 127))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.Dark, brush)
brush = QtGui.QBrush(QtGui.QColor(113, 170, 170))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.Mid, brush)
brush = QtGui.QBrush(QtGui.QColor(85, 127, 127))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.Text, brush)
brush = QtGui.QBrush(QtGui.QColor(255, 255, 255))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.BrightText, brush)
brush = QtGui.QBrush(QtGui.QColor(85, 127, 127))
brush.setStyle(QtCore.Qt.SolidPattern)
palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.ButtonText, brush)
brush = QtGui.QBrush(QtGui.QColor(170, 255, 255))
```

```
        brush.setStyle(QtCore.Qt.SolidPattern)
        palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.Base, brush)
        brush = QtGui.QBrush(QtGui.QColor(170, 255, 255))
        brush.setStyle(QtCore.Qt.SolidPattern)
        palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.Window, brush)
        brush = QtGui.QBrush(QtGui.QColor(0, 0, 0))
        brush.setStyle(QtCore.Qt.SolidPattern)
        palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.Shadow, brush)
        brush = QtGui.QBrush(QtGui.QColor(170, 255, 255))
        brush.setStyle(QtCore.Qt.SolidPattern)
        palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.AlternateBase, brush)
        brush = QtGui.QBrush(QtGui.QColor(255, 255, 220))
        brush.setStyle(QtCore.Qt.SolidPattern)
        palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.ToolTipBase, brush)
        brush = QtGui.QBrush(QtGui.QColor(0, 0, 0))
        brush.setStyle(QtCore.Qt.SolidPattern)
        palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.ToolTipText, brush)
        self.EstadoConexion.setPalette(palette)
        self.EstadoConexion.setAutoFillBackground(True)
        self.EstadoConexion.setLineWidth(9)
        self.EstadoConexion.setObjectName("EstadoConexion")
        self.horizontalLayout_5.addWidget(self.EstadoConexion)
        self.label_6 = QtWidgets.QLabel(self.groupBox_3)
        self.label_6.setGeometry(QtCore.QRect(240, 10, 61, 61))
        self.label_6.setText("")
        self.label_6.setPixmap(QtGui.QPixmap(":/newPrefix/ZedBoard_RevA_sideA_0_0 (1)_0.jpg"))
        self.label_6.setScaledContents(True)
        self.label_6.setObjectName("label_6")
        self.RMPI.raise_()
        self.layoutWidget.raise_()
        self.layoutWidget.raise_()
        self.frame.raise_()
        self.groupBox.raise_()
        self.groupBox_2.raise_()
        self.groupBox_3.raise_()
        MainWindow.setCentralWidget(self.centralwidget)
        self.menubar = QtWidgets.QMenuBar(MainWindow)
        self.menubar.setGeometry(QtCore.QRect(0, 0, 1103, 21))
        self.menubar.setObjectName("menubar")
        MainWindow.setMenuBar(self.menubar)
        self.statusbar = QtWidgets.QStatusBar(MainWindow)
        self.statusbar.setObjectName("statusbar")
        MainWindow.setStatusBar(self.statusbar)

        self.retranslateUi(MainWindow)
        self.RMPI.setCurrentIndex(1)
        QtCore.QMetaObject.connectSlotsByName(MainWindow)

    def retranslateUi(self, MainWindow):
        _translate = QtCore.QCoreApplication.translate
        MainWindow.setWindowTitle(_translate("MainWindow", "MainWindow"))
        self.Titulo.setText(_translate("MainWindow", "<html><head/><body>
        <p align=\"center\"><span style=\" font-size:20pt;\">
        Reconstructor 16xSubNyquist</span></p></body></html>"))
        self.EstadoConexion_3.setText(_translate("MainWindow",
        "<html><head/><body><p align=\"center\">Autor: Mariano L.
        Acosta<br/>2018</p></body></html>"))
        self.groupBox_4.setTitle(_translate("MainWindow",
        "Tiempo Requerido"))
        self.EstadoConexion_9.setText(_translate("MainWindow",
        "<html><head/><body><p>Matriz A</p></body></html>"))
        self.matrizAt.setText(_translate("MainWindow", "<html><head/><body><p
        align=\"center\">0</p></body></html>"))
        self.EstadoConexion_22.setText(_translate("MainWindow", "<html><head/><body><p
        align=\"center\">mSeg</p></body></html>"))
        self.EstadoConexion_10.setText(_translate("MainWindow",
        "<html><head/><body><p>Optimizaci n</p></body></html>"))
        self.optT.setText(_translate("MainWindow", "<html><head/><body><p
        align=\"center\">0</p></body></html>"))
        self.EstadoConexion_20.setText(_translate("MainWindow", "<html><head/><body><p
        align=\"center\">mSeg</p></body></html>"))
        self.EstadoConexion_11.setText(_translate("MainWindow",
        "<html><head/><body><p>Reconstruc.</p></body></html>"))
        self.recT.setText(_translate("MainWindow", "<html><head/><body><p
        align=\"center\">0</p></body></html>"))
        self.EstadoConexion_21.setText(_translate("MainWindow", "<html><head/><body><p
        align=\"center\">mSeg</p></body></html>"))
        self.groupBox_5.setTitle(_translate("MainWindow", "Comparaci n"))
        self.superponer.setText(_translate("MainWindow", "Superponer"))
        self.EstadoConexion_13.setText(_translate("MainWindow",
        "<html><head/><body><p>Similitud <br/>Temporal</p></body></html>"))
        self.EstadoConexion_15.setText(_translate("MainWindow", "
        <html><head/><body><p>Similitud <br/>Espectral</p></body></html>"))
        self.simTemp.setText(_translate("MainWindow", "<html><head/><body><p
        align=\"center\">0</p></body></html>"))
        self.simEspec.setText(_translate("MainWindow", "<html><head/><body><p
        align=\"center\">0</p></body></html>"))
        self.label_9.setText(_translate("MainWindow", "Se al Original"))
        self.label_10.setText(_translate("MainWindow", "Se al Reconstruida con CS"))
        self.espectroCheck.setText(_translate("MainWindow", "Espectro"))
        self.interCheck.setText(_translate("MainWindow", "Inter x10"))
        self.groupBox_6.setTitle(_translate("MainWindow", "FPGA Acceleration"))
        self.checkBox_3.setText(_translate("MainWindow", "Sorting Net"))
        self.checkBox_4.setText(_translate("MainWindow", "HW Subopt"))
        self.checkBox_5.setText(_translate("MainWindow", "HW Mult."))
```

```
        self.zoom.setText(_translate("MainWindow", "Zoom"))
        self.RMPI.setTabText(self.RMPI.indexOf(self.tab), _translate("MainWindow", "Main"))
        self.label_11.setText(_translate("MainWindow", "Se al Original X(t)"))
        self.checkBox_9.setText(_translate("MainWindow", "Espectro"))
        self.interpolar.setText(_translate("MainWindow", "Inter x10"))
        self.label_12.setText(_translate("MainWindow", "Sequencia y[n] Adquirida"))
        self.groupBox_7.setTitle(_translate("MainWindow", "Demodulador"))
        self.EstadoConexion_14.setText(_translate("MainWindow", "<html><head/><body><p
        align=\"center\">Ganancia <br/>Pre- Conversi n</p></body></html>"))
        self.EstadoConexion_24.setText(_translate("MainWindow", "<html><head/><body><p
        align=\"center\"><br/>Generador <br/>de Secuencia </p></body></html>"))
        __sortingEnabled = self.listWidget.isSortingEnabled()
        self.listWidget.setSortingEnabled(False)
        item = self.listWidget.item(0)
        item.setText(_translate("MainWindow", "Random"))
        item = self.listWidget.item(1)
        item.setText(_translate("MainWindow", "FWTSM"))
        item = self.listWidget.item(2)
        item.setText(_translate("MainWindow", "Logistic"))
        item = self.listWidget.item(3)
        item.setText(_translate("MainWindow", "TWBM"))
        item = self.listWidget.item(4)
        item.setText(_translate("MainWindow", "TWBM2"))
        item = self.listWidget.item(5)
        item.setText(_translate("MainWindow", "TWBM4"))
        item = self.listWidget.item(6)
        item.setText(_translate("MainWindow", "TWTS"))
        item = self.listWidget.item(7)
        item.setText(_translate("MainWindow", "TWTSM"))
        self.listWidget.setSortingEnabled(__sortingEnabled)
        self.cargarSignal.setText(_translate("MainWindow", "Cargar Se al"))
        self.adquirir.setText(_translate("MainWindow", "Adquirir"))
        self.guardar.setText(_translate("MainWindow", "Guardar"))
        self.groupBox_8.setTitle(_translate("MainWindow", "Diagrama en Bloques"))
        self.RMPI.setTabText(self.RMPI.indexOf(self.tab_2), _translate("MainWindow", "RMPI"))
        self.RMPI.setTabText(self.RMPI.indexOf(self.tab_3), _translate("MainWindow",
        "Terminal"))
        self.groupBox.setTitle(_translate("MainWindow", "Control"))
        self.sendSequences.setText(_translate("MainWindow", "Enviar Secuencias"))
        self.openSequences.setText(_translate("MainWindow", "Cargar Secuencias"))
        self.sendDictionary.setText(_translate("MainWindow", "Enviar Diccionario"))
        self.sendSubMuestras.setText(_translate("MainWindow", "Enviar SubMuestras"))
        self.openSubMuestras.setText(_translate("MainWindow", "Cargar SubMuestras"))
        self.openDictionary.setText(_translate("MainWindow", "Cargar Diccionario"))
        self.groupBox_2.setTitle(_translate("MainWindow", "Optimizador"))
        self.EstadoConexion_5.setText(_translate("MainWindow", "<html><head/><body><p
        align=\"center\">Muestras</p></body></html>"))
        self.Reconstruct.setText(_translate("MainWindow", "Reconstruct!"))
        self.checkBox.setText(_translate("MainWindow", "Mostrar \n"
"Estad sticas"))
        self.checkBox_2.setText(_translate("MainWindow", "Modo Debug"))
        self.Actualizar.setText(_translate("MainWindow", "Actualizar"))
        self.EstadoConexion_4.setText(_translate("MainWindow", "<html><head/><body><p
        align=\"center\">Sparcity</p></body></html>"))
        self.EstadoConexion_6.setText(_translate("MainWindow", "<html><head/><body><p
        align=\"center\">Tolerancia</p></body></html>"))
        self.EstadoConexion_7.setText(_translate("MainWindow", "<html><head/><body><p
        align=\"center\">Tolerancia<br/>(subopt)</p></body></html>"))
        self.EstadoConexion_8.setText(_translate("MainWindow", "<html><head/><body><p
        align=\"center\">Iteraciones<br/>(Max)</p></body></html>"))
        self.groupBox_3.setTitle(_translate("MainWindow", "Conexi n en Red"))
        self.conectar.setText(_translate("MainWindow", "Conectar"))
        self.EstadoConexion.setText(_translate("MainWindow", "<html><head/><body><p
        align=\"center\">Desconectado</p></body></html>"))

from pyqtgraph import PlotWidget
import j2_rc
import j3_rc
import j4_rc
import j5_rc
import j_rc

if __name__ == "__main__":
    import sys
    app = QtWidgets.QApplication(sys.argv)
    MainWindow = QtWidgets.QMainWindow()
    ui = Ui_MainWindow()
    ui.setupUi(MainWindow)
    MainWindow.show()
    sys.exit(app.exec_())
```

Lógica de Negocios

```
import sys,time
from PyQt5 import QtCore, QtGui, QtWidgets, QtNetwork
from PyQt5.QtCore import QDataStream, QIODevice,QByteArray
import struct
import math
import numpy
import scipy
import pyqtgraph as pg

import CS_GUI_3,csv,sinc
```

```
class Business:

    def __init__(self,interface):
        self.interface=interface
        self.connect_signals_to_methods()
        self.tcp_client = QtNetwork.QTcpSocket()
        self.signal = []
        self.signal_long = []
        self.Y = []
        self.X = []
        self.X_10 = []
        self.signal_10 =[]
        self.Fx = []
        self.Fs = []
        self.matrixA = []
        self.matrixB = []
        self.matrixC = []
        self.Cbuffer = []
        self.Cdata = []
        self.opt = [0, 0, 0, 0]
        self.zbool = 0
        self.sbool = 0
        self.supbool = 0
        self.spec_sim = 0
        self.ibool = 0
        self.temp_sim = 0



    def connect_signals_to_methods(self):
        self.interface.conectar.clicked.connect(self.establecerConexion)
        self.interface.sendSequences.clicked.connect(self.enviarSecuencias)
        self.interface.sendDictionary.clicked.connect(self.enviarDiccionario)
        self.interface.openSequences.clicked.connect(self.abrirArchivoA)
        self.interface.openDictionary.clicked.connect(self.abrirArchivoB)
        self.interface.Reconstruct.clicked.connect(self.multiplicarAB)
        self.interface.openSubMuestras.clicked.connect(self.abrirArchivoY)
        self.interface.sendSubMuestras.clicked.connect(self.enviarY)
        self.interface.Actualizar.clicked.connect(self.actualizar)
        self.interface.zoom.clicked.connect(self.zoom)
        self.interface.espectroCheck.clicked.connect(self.plotSpectrum)
        self.interface.superponer.clicked.connect(self.superponer)
        self.interface.cargarSignal.clicked.connect(self.abrirSignal)
        self.interface.adquirir.clicked.connect(self.adquirir)
        self.interface.interCheck.clicked.connect(self.interpolar)

    def interpolar(self):

        if self.ibool == 0:
            self.X_10 = self.X
            self.signal_10 = self.signal
            self.X = sinc.resample(numpy.array(self.X), 10)
            self.signal = sinc.resample(numpy.array(self.signal), 10)
            self.ibool = 1
        else:
            self.X = self.X_10
            self.signal = self.signal_10
            self.ibool = 0

    def adquirir(self):
        pass


    def abrirSignal(self):
        name, _ = QtWidgets.QFileDialog.getOpenFileName(self.interface.centralwidget,
        'Open File')

        if not (name == ''):
            with open(name,'r') as csv_file:
                self.buffer = csv.reader(csv_file, quoting=csv.QUOTE_NONNUMERIC)
                self.buffer = list(self.buffer)
                self.buffer = self.buffer[0]

                self.signal_long = self.buffer[0:1024]

                self.signal=self.signal_long #no hace nada
                self.interface.graphicsView.clear()
                self.interface.graphicsView_4.clear()
                self.interface.graphicsView.plot(self.signal, pen='w')
                self.interface.graphicsView_4.plot(self.signal, pen='w')

                self.interface.graphicsView_4.showGrid(1, 1, .7)

                self.interface.graphicsView_4.setLabel('left', 'Amplitud', 'Volts')
                self.interface.graphicsView_4.setLabel('bottom', 'Muestras', 'N ')
                self.interface.graphicsView_4.autoRange()

                #submuestras

                self.Y= self.buffer[1024:1088]
                self.interface.graphicsView_3.clear()
                self.interface.graphicsView_3.plot(self.Y, pen='y')
```

```
            self.interface.graphicsView_3.showGrid(1, 1, .7)

            self.interface.graphicsView_3.setLabel('left', 'Amplitud', 'Volts')
            self.interface.graphicsView_3.setLabel('bottom', 'Muestras', 'N ')
            self.interface.graphicsView_3.autoRange()

            #sequencias

            self.matrixA = self.buffer[1088:5184]



def zoom(self):
    if (self.zbool == 0):
        self.interface.graphicsView_2.setXRange(412, 612, padding=None, update=True)
        self.interface.graphicsView.setXRange(412, 612, padding=None, update=True)
        self.zbool = 1
    else:
        self.interface.graphicsView_2.autoRange()
        self.interface.graphicsView.autoRange()
        self.zbool = 0

def plotSpectrum(self):
    if (self.sbool == 0):

        self.interface.graphicsView_2.clear()
        self.interface.graphicsView_2.plot(abs(self.Fx), pen='b')
        self.interface.graphicsView_2.setXRange(0, 512)
        self.interface.graphicsView_2.showGrid(1,1,.7)

        self.interface.graphicsView_2.setLabel('left', 'Amplitud', 'Volts')
        self.interface.graphicsView_2.setLabel('bottom', 'Muestras', 'N ')
        self.interface.graphicsView_2.autoRange()
        self.interface.graphicsView_2.setXRange(0, 512)

        self.interface.graphicsView.clear()
        self.interface.graphicsView.plot(abs(self.Fs), pen='w')
        self.interface.graphicsView.setXRange(0, 512)
        self.interface.graphicsView.showGrid(1,1,.7)

        self.interface.graphicsView.setLabel('left', 'Amplitud', 'Volts')
        self.interface.graphicsView.setLabel('bottom', 'Muestras', 'N ')
        self.interface.graphicsView.autoRange()
        self.interface.graphicsView.setXRange(0, 512)


        self.sbool = 1
    else:
        self.sbool = 0
        self.plotNormal()

def plotNormal(self):
    self.interface.graphicsView.clear()
    self.interface.graphicsView.plot(self.signal, pen='w')
    self.interface.graphicsView.showGrid(1, 1, .7)
    self.interface.graphicsView_2.setLogMode(False, False)

    self.interface.graphicsView.setLabel('left', 'Amplitud', 'Volts')
    self.interface.graphicsView.setLabel('bottom', 'Muestras', 'N ')

    self.interface.graphicsView_2.clear()
    self.interface.graphicsView_2.plot(self.X, pen='b')
    self.interface.graphicsView_2.showGrid(1,1,.7)

    self.interface.graphicsView_2.setLabel('left', 'Amplitud', 'Volts')
    self.interface.graphicsView_2.setLabel('bottom', 'Muestras', 'N ')
    self.interface.graphicsView_2.autoRange()
    self.interface.graphicsView.autoRange()

def superponer(self):

    if self.supbool == 0:
        self.interface.graphicsView_2.clear()

        self.interface.graphicsView.clear()
        if self.sbool == 0:
            self.interface.graphicsView.plot(self.signal, pen='w')
            self.interface.graphicsView.plot(self.X, pen='r')
        else:
            self.interface.graphicsView.plot(abs(self.Fs), pen='w')
            self.interface.graphicsView.plot(abs(self.Fx), pen='r')
        self.supbool = 1
    else:

        if self.sbool == 1:
            self.plotSpectrum()
        else:
            self.plotNormal()

        self.supbool = 0

        if self.zbool == 1:
            self.interface.graphicsView_2.setXRange(412, 612, padding=None, update=True)
```

```
            self.interface.graphicsView.setXRange(412, 612, padding=None, update=True)


def abrirArchivoA(self):

    name, _ = QtWidgets.QFileDialog.getOpenFileName(self.interface.centralwidget,
    'Open File')

    if not (name == ''):
        with open(name,'r') as csv_file:
            self.matrixA = csv.reader(csv_file, quoting=csv.QUOTE_NONNUMERIC)
            self.matrixA = list(self.matrixA)
            self.matrixA = self.matrixA[0]

def abrirArchivoB(self):

    name, _ = QtWidgets.QFileDialog.getOpenFileName(self.interface.centralwidget,
    'Open File')

    if not (name == ''):
        with open(name,'r') as csv_file:
            self.matrixB = csv.reader(csv_file, quoting=csv.QUOTE_NONNUMERIC)
            self.matrixB = list(self.matrixB)
            self.matrixB = self.matrixB[0]

def abrirArchivoY(self):

    name, _ = QtWidgets.QFileDialog.getOpenFileName(self.interface.centralwidget,
    'Open File')

    if not (name == ''):
        with open(name,'r') as csv_file:
            self.Y = csv.reader(csv_file, quoting=csv.QUOTE_NONNUMERIC)
            self.Y = list(self.Y)
            self.Y = self.Y[0]

    self.interface.graphicsView_3.clear()
    self.interface.graphicsView_3.plot(self.Y, pen='y')

    self.interface.graphicsView_3.showGrid(1, 1, .7)

    self.interface.graphicsView_3.setLabel('left', 'Amplitud', 'Volts')
    self.interface.graphicsView_3.setLabel('bottom', 'Muestras', 'N ')
    self.interface.graphicsView_3.autoRange()


def recibirDatos(self):
    comando = 'ECHA'
    comando = bytes(comando, encoding='ascii')

    txBuffer = QByteArray()
    out = QDataStream(txBuffer,QIODevice.ReadWrite)
    out.writeString(comando)

    self.tcp_client.write(txBuffer)
    print('request to receive matrix sent')
    self.tcp_client.waitForReadyRead()
    echo = self.tcp_client.readAll()
    print(type(echo))

def enviarSecuencias(self):

    comando = 'MTXA'
    comando = bytes(comando, encoding='ascii')


    txBuffer = QByteArray()
    out = QDataStream(txBuffer,QIODevice.ReadWrite)
    out.writeString(comando)

    self.tcp_client.write(txBuffer)
    print('request to send matrix A sent')
    self.tcp_client.waitForReadyRead()
    # read incomming data
    instr = self.tcp_client.readAll()
    instr= str(instr, encoding='ascii')

    # in this case we print to the terminal could update text of a widget if we wanted.
    if instr == ('SNDA'):
        print('confirmation to send sequences received')
        txBuffer = QByteArray()
        out = QDataStream(txBuffer, QIODevice.ReadWrite)
        out.setFloatingPointPrecision(0); #single precision
        out.setByteOrder(1);

        for i in self.matrixA:
            out.writeFloat(i)
        print('...')
        time.sleep(.01)
        self.tcp_client.write(txBuffer)
        print('matrix A sent')
    else:
        print('comando no recibido')
        print(instr)
```

```
def enviarDiccionario(self):

    comando = 'MTXB'
    comando = bytes(comando, encoding='ascii')


    txBuffer = QByteArray()
    out = QDataStream(txBuffer,QIODevice.ReadWrite)
    out.writeString(comando)

    self.tcp_client.write(txBuffer)
    print('request to send matrix A sent')
    self.tcp_client.waitForReadyRead()
    # read incomming data
    instr = self.tcp_client.readAll()
    instr= str(instr, encoding='ascii')

    # in this case we print to the terminal could update text of a widget if we wanted.
    if instr == ('SNDB'):
        print('confirmation to send matrix B received')
        txBuffer = QByteArray()
        out = QDataStream(txBuffer, QIODevice.ReadWrite)
        out.setFloatingPointPrecision(0); #single precision
        out.setByteOrder(1);

        for i in self.matrixB:
            out.writeFloat(i)
        print('...')
        time.sleep(.01)
        self.tcp_client.write(txBuffer)
        print('matrix B sent')
    else:
        print('comando no recibido')
        print(instr)


def multiplicarAB(self):
    self.matrixC = []
    comando = 'MLTM'
    comando = bytes(comando, encoding='ascii')

    time.sleep(.1)
    txBuffer = QByteArray()
    out = QDataStream(txBuffer,QIODevice.ReadWrite)
    out.writeString(comando)

    self.tcp_client.write(txBuffer)
    print('request to multipy matrix sent')

    self.tcp_client.waitForReadyRead()
    instr = self.tcp_client.readAll()
    instr = bytes(instr);
    X = struct.unpack('256f',instr)

    for i in [0,1,2]:
        comando = 'SNDM'
        comando = bytes(comando, encoding='ascii')


        txBuffer = QByteArray()
        out = QDataStream(txBuffer,QIODevice.ReadWrite)
        out.writeString(comando)
        self.tcp_client.write(txBuffer)

        self.tcp_client.waitForReadyRead()
        instr = self.tcp_client.readAll()
        instr = bytes(instr);
        X = X +struct.unpack('256f',instr)

    #recepcion de los tiempos

    comando = 'SNDT'
    comando = bytes(comando, encoding='ascii')

    txBuffer = QByteArray()
    out = QDataStream(txBuffer, QIODevice.ReadWrite)
    out.writeString(comando)
    self.tcp_client.write(txBuffer)

    self.tcp_client.waitForReadyRead()
    instr = self.tcp_client.readAll()
    instr = bytes(instr);
    times = struct.unpack('3f', instr)
    self.interface.matrizAt.setText("{0:4.3f}".format(times[0]))
    self.interface.optT.setText("{0:4.3f}".format(times[1]))
    self.interface.recT.setText("{0:4.3f}".format(times[2]))

    print(X)
    print(times)
    self.interface.graphicsView.clear()
    self.interface.graphicsView.plot(self.signal, pen='w')
    self.interface.graphicsView.showGrid(1, 1, .7)

    self.interface.graphicsView.setLabel('left', 'Amplitud', 'Volts')
```

```
        self.interface.graphicsView.setLabel('bottom', 'Muestras', 'N ')

        self.interface.graphicsView_2.clear()
        self.interface.graphicsView_2.plot(X, pen='b')
        self.interface.graphicsView_2.showGrid(1,1,.7)

        self.interface.graphicsView_2.setLabel('left', 'Amplitud', 'Volts')
        self.interface.graphicsView_2.setLabel('bottom', 'Muestras', 'N ')
        self.interface.graphicsView_2.autoRange()
        self.interface.graphicsView.autoRange()
        self.X =X

        self.temp_sim = numpy.linalg.norm(numpy.subtract(numpy.array(self.signal),
        numpy.array(self.X)))/numpy.linalg.norm(numpy.array(self.signal))
        self.interface.simTemp.setText("{0:4.8f}".format(self.temp_sim))

        self.Fx = numpy.fft.fft(numpy.asarray(self.X))
        self.Fs = numpy.fft.fft(numpy.asarray(self.signal))

        self.spec_sim = numpy.linalg.norm(numpy.subtract(numpy.array(abs(self.Fs)),
        numpy.array(abs(self.Fx)))/numpy.linalg.norm(numpy.array(abs(self.Fs))))
        self.interface.simEspec.setText("{0:4.8f}".format(self.spec_sim))



    def actualizar(self):
        comando = 'OPCT'
        comando = bytes(comando, encoding='ascii')


        txBuffer = QByteArray()
        out = QDataStream(txBuffer,QIODevice.ReadWrite)
        out.writeString(comando)

        self.tcp_client.write(txBuffer)
        print('request to send Config. Parameters sent')
        self.tcp_client.waitForReadyRead()
        # read incomming data
        instr = self.tcp_client.readAll()
        instr= str(instr, encoding='ascii')

        # in this case we print to the terminal could update text of a widget if we wanted.
        if instr == ('SNDO'):
            print('confirmation to send Parameters received')
            txBuffer = QByteArray()
            out = QDataStream(txBuffer, QIODevice.ReadWrite)
            out.setFloatingPointPrecision(0); #single precision
            out.setByteOrder(1);

            self.opt[0] = self.interface.sparcityBox.value()
            base = self.interface.tolBaseBox.value()*10
            expo = self.interface.tolExpBox.value()
            self.opt[1] = math.pow(base,expo)
            base = self.interface.subtolBaseBox.value()*10
            expo = self.interface.subtolExpBox.value()
            self.opt[2] = math.pow(base,expo)
            self.opt[3] = self.interface.maxiterBox.value()

            for i in self.opt:
                out.writeFloat(i)
            print('...')
            time.sleep(.01)
            self.tcp_client.write(txBuffer)
            print('parameters sent')
        else:
            print('comando no recibido')
            print(instr)


    def enviarY(self):

        comando = 'SEQY'
        comando = bytes(comando, encoding='ascii')


        txBuffer = QByteArray()
        out = QDataStream(txBuffer,QIODevice.ReadWrite)
        out.writeString(comando)

        self.tcp_client.write(txBuffer)
        print('request to send Y sent')
        self.tcp_client.waitForReadyRead()
        # read incomming data
        instr = self.tcp_client.readAll()
        instr= str(instr, encoding='ascii')

        # in this case we print to the terminal could update text of a widget if we wanted.
        if instr == ('SNDY'):
            print('confirmation to send Y received')
            txBuffer = QByteArray()
            out = QDataStream(txBuffer, QIODevice.ReadWrite)
            out.setFloatingPointPrecision(0); #single precision
            out.setByteOrder(1);
```

```
            for i in self.Y:
                out.writeFloat(i)
            print('...')
            time.sleep(.01)
            self.tcp_client.write(txBuffer)
            print('matrix Y sent')
        else:
            print('comando no recibido')
            print(instr)

    def establecerConexion(self):

        estado = self.tcp_client.state()

        if (estado == 0):

            HOST = '192.168.1.10'
            PORT = 1491
            Timeout = 1000
            self.tcp_client.connectToHost(HOST, PORT,QIODevice.ReadWrite)
            self.tcp_client.connected.connect(self.dealCommunication)
            self.tcp_client.disconnected.connect(self.endCommunication)
            self.tcp_client.error.connect(self.displayError)

        elif(estado == 3):

            self.tcp_client.close()

    def dealCommunication(self):
        self.interface.EstadoConexion.setText(
            "<html><head/><body><p align=\"center\"><span style=\"
            font-size:8pt;\">Conectado</span></p></body></html>")
        self.interface.conectar.setText('Desconectar')

    def endCommunication(self):
        self.interface.EstadoConexion.setText(
            "<html><head/><body><p align=\"center\"><span style=\"
            font-size:8pt;\">Desconectado</span></p></body></html>")
        self.interface.conectar.setText('Conectar')

    def displayError(self):
        #self.interface.consola
        print(self, "The following error occurred: %s." % self.tcp_client.errorString())

def main():

    app = QtWidgets.QApplication(sys.argv)
    MainWindow = QtWidgets.QMainWindow()
    ui = CS_GUI_3.Ui_MainWindow()
    ui.setupUi(MainWindow)
    c = Business(ui)
    MainWindow.show()
    app.exec_()


if __name__ == "__main__":
    # execute only if run as a script
    main()
```

Bibliografía

[1] Massimo Fornasier and Holger Rauhut. Compressive sensing. In *Handbook of mathematical methods in imaging*, pages 187–228. Springer, 2011.

[2] Xilinx. *Zynq-7000 All Programmable SoC Technical Reference Manual*, ug585 (v1.11) edition, sep 2016.

[3] Simon Foucart and Holger Rauhut. *A mathematical introduction to compressive sensing*, volume 1. Birkhäuser Basel, 2013.

[4] Emmanuel J Candes and Terence Tao. Near-optimal signal recovery from random projections: Universal encoding strategies. *IEEE transactions on information theory*, 52(12):5406–5425, 2006.

[5] Yonina C Eldar and Gitta Kutyniok. *Compressed sensing: theory and applications*. Cambridge University Press, 2012.

[6] Tim Roughgarden and Gregory Valiant. Cs168: The modern algorithmic toolbox lecture# 17: Compressive sensing. 2015.

[7] MV Patil, Apoorva Gupta, Ankita Varma, and Shikhar Salil. Audio and speech compression using dct and dwt techniques. *International Journal of Innovative Research in Science, Engineering and Technology*, 2(5):1712–1719, 2013.

[8] Hong Cheng. The fundamentals of compressed sensing. In *Sparse Representation, Modeling and Learning in Visual Recognition*, pages 21–53. Springer, 2015.

[9] Emmanuel Candes and Justin Romberg. Sparsity and incoherence in compressive sampling. *Inverse problems*, 23(3):969, 2007.

[10] Irina F Gorodnitsky and Bhaskar D Rao. Sparse signal reconstruction from limited data using focuss: A re-weighted minimum norm algorithm. *IEEE Transactions on signal processing*, 45(3):600–616, 1997.

[11] Douglas A Lyon. The discrete fourier transform, part 4: spectral leakage. *Journal of object technology*, 8(7), 2009.

[12] Stéphane G Mallat and Zhifeng Zhang. Matching pursuits with time-frequency dictionaries. *IEEE Transactions on signal processing*, 41(12):3397–3415, 1993.

[13] David L Donoho and Michael Elad. Optimally sparse representation in general (nonorthogonal) dictionaries via 1 minimization. *Proceedings of the National Academy of Sciences*, 100(5):2197–2202, 2003.

[14] Alfred M Bruckstein, David L Donoho, and Michael Elad. From sparse solutions of systems of equations to sparse modeling of signals and images. *SIAM review*, 51(1):34–81, 2009.

[15] Stephen R Becker. *Practical compressed sensing: modern data acquisition and signal processing*. California Institute of Technology, 2011.

[16] Mark A Davenport. *Random observations on random observations: Sparse signal acquisition and processing*. Rice University, 2010.

[17] Juhwan Yoo, Stephen Becker, Manuel Monge, Matthew Loh, Emmanuel Candes, and Azita Emami-Neyestanak. Design and implementation of a fully integrated compressed-sensing signal acquisition system. In *Acoustics, Speech and Signal Processing (ICASSP), 2012 IEEE International Conference on*, pages 5325–5328. IEEE, 2012.

[18] Pawel Jerzy Pankiewicz. Implementation aspects of the random demodulator for compressive sensing. 2013.

[19] Jason N Laska, Sami Kirolos, Marco F Duarte, Tamer S Ragheb, Richard G Baraniuk, and Yehia Massoud. Theory and implementation of an analog-to-information converter using random demodulation. In *Circuits and Systems, 2007. ISCAS 2007. IEEE International Symposium on*, pages 1959–1962. IEEE, 2007.

[20] Juhwan Yoo. *Compressed sensing receivers: theory, design, and performance limits*. California Institute of Technology, 2012.

[21] Becker Stephen. Cosamp and omp for sparse recovery. `https://la.mathworks.com/matlabcentral/fileexchange/32402-cosamp-and-omp-for-sparse-recovery`. [Online; accedido el 22-Abril-2018].

[22] L1-magic. `https://statweb.stanford.edu/~candes/l1magic/`. [Online; accedido el 22-Abril-2018].

[23] Rene Mueller, Jens Teubner, and Gustavo Alonso. Sorting networks on fpgas. *The VLDB Journal—The International Journal on Very Large Data Bases*, 21(1):1–23, 2012.

[24] Richard Barry et al. Freertos. *Internet, Oct*, 2008.

[25] Adam Dunkels et al. The lwip tcp/ip stack. *lwIP–A LightWeight TCP/IP Stack*, 2004.

[26] Abdul J Jerri. The shannon sampling theorem—its various extensions and applications: A tutorial review. *Proceedings of the IEEE*, 65(11):1565–1596, 1977.

[27] Karlheinz Brandenburg. Mp3 and aac explained. In *Audio Engineering Society Conference: 17th International Conference: High-Quality Audio Coding*. Audio Engineering Society, 1999.

[28] Guy Cote, Berna Erol, Michael Gallant, and Faouzi Kossentini. H. 263+: Video coding at low bit rates. *IEEE Transactions on circuits and systems for video technology*, 8(7):849–866, 1998.

[29] Hossein Mamaghanian, Nadia Khaled, David Atienza, and Pierre Vandergheynst. Compressed sensing for real-time energy-efficient ecg compression on wireless body sensor nodes. *IEEE Transactions on Biomedical Engineering*, 58(9):2456–2466, 2011.

[30] Pawan K Baheti and Harinath Garudadri. An ultra low power pulse oximeter sensor based on compressed sensing. In *Wearable and Implantable Body Sensor Networks, 2009. BSN 2009. Sixth International Workshop on*, pages 144–148. IEEE, 2009.

[31] Chengbo Li. *An efficient algorithm for total variation regularization with applications to the single pixel camera and compressive sensing*. PhD thesis, Rice University, 2010.

[32] Jarvis Haupt, Waheed U Bajwa, Michael Rabbat, and Robert Nowak. Compressed sensing for networked data. *IEEE Signal Processing Magazine*, 25(2):92–101, 2008.

[33] Yonina C Eldar and Tomer Michaeli. Beyond bandlimited sampling. *IEEE signal processing magazine*, 26(3):48–68, 2009.

[34] A. Burg N. Felber H. Kaeslin D. Bellasi, P. Maechler. Real-time audio restoration using sparse signal recovery. *Rice University, USA;*.

[35] Jérôme Bobin, Jean-Luc Starck, and Roland Ottensamer. Compressed sensing in astronomy. *IEEE Journal of Selected Topics in Signal Processing*, 2(5):718–726, 2008.

[36] Michael Lustig, David L Donoho, Juan M Santos, and John M Pauly. Compressed sensing mri. *IEEE signal processing magazine*, 25(2):72–82, 2008.

[37] Shancang Li, Li Da Xu, and Xinheng Wang. Compressed sensing signal and data acquisition in wireless sensor networks and internet of things. *IEEE Transactions on Industrial Informatics*, 9(4):2177–2186, 2013.

[38] Zhi Tian and Georgios B Giannakis. Compressed sensing for wideband cognitive radios. Technical report, MICHIGAN TECHNOLOGICAL UNIV HOUGHTON, 2007.

[39] Stefan Wegenkittl. *Empirical testing of pseudorandom number generators*. na, 1995.

[40] Matthias Krause and Stefan Lucks. On the minimal hardware complexity of pseudorandom function generators. In *Annual Symposium on Theoretical Aspects of Computer Science*, pages 419–430. Springer, 2001.

[41] Stephen Eubank and Doyne Farmer. An introduction to chaos and randomness. In *1989 lectures in complex systems. Proceedings: Lectures, Volume 2*. 1990.

[42] G. Setti, G. Mazzini, R. Rovatti, and S. Callegari. Statistical modeling of discrete-time chaotic processes: Basic finite-dimensional tools and applications. *Proceedings of the IEEE*, 90(5):662–689, Mayo 2002.

[43] AR Murch and RHT Bates. Colored noise generation through deterministic chaos. *IEEE Transactions on circuits and systems*, 37(5):608–613, 1990.

[44] Sergio Callegari, Riccardo Rovatti, and Gianluca Setti. Embeddable adc-based true random number generator for cryptographic applications exploiting nonlinear signal processing and chaos. *IEEE transactions on signal processing*, 53(2):793–805, 2005.

[45] CHEN Shuai and Xian-Xin Zhong. Confidential communication through chaos encryption in wireless sensor network. *Journal of China University of Mining and Technology*, 17(2):258–261, 2007.

[46] Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, et al. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*, 2016.

[47] Manish Pandey. Machine learning and systems for building the next generation of eda tools. In *Proceedings of the 23rd Asia and South Pacific Design Automation Conference*, pages 411–415. IEEE Press, 2018.

[48] Shuai Che, Jie Li, Jeremy W Sheaffer, Kevin Skadron, and John Lach. Accelerating compute-intensive applications with gpus and fpgas. In *Application Specific Processors, 2008. SASP 2008. Symposium on*, pages 101–107. IEEE, 2008.

[49] Andrew Rukhin, Juan Soto, James Nechvatal, Miles Smid, and Elaine Barker. A statistical test suite for random and pseudorandom number generators for cryptographic applications. Technical report, Booz-Allen and Hamilton Inc Mclean Va, 2001.

[50] Luigi Accardi and Markus Gaebler. Statistical analysis of random number generators. In *Quantum Bio-Informatics IV: From Quantum Information to Bio-Informatics*, pages 117–128. World Scientific, 2011.

[51] Nabarun Mondal and Partha P Ghosh. A pseudo random number generator from chaos. *arXiv preprint arXiv:1203.5731*, 2012.

[52] Charlotte Werndl. Are deterministic descriptions and indeterministic descriptions observationally equivalent? *Studies in history and philosophy of science part B: studies in history and philosophy of modern physics*, 40(3):232–242, 2009.

[53] L. De Micco, C. M. González, H. A. Larrondo, M. T. Martin, A. Plastino, and O. A. Rosso. Randomizing nonlinear maps via symbolic dynamics. *Physica A*, 387:3373–3383, 2008.

[54] Graeme Pope. *Compressive sensing: A summary of reconstruction algorithms*. PhD thesis, ETH, Swiss Federal Institute of Technology Zurich, Department of Computer Science, 2009.

[55] Lijun Zhang, Tianbao Yang, Rong Jin, and Zhi-Hua Zhou. A simple homotopy algorithm for compressive sensing. In *Artificial Intelligence and Statistics*, pages 1116–1124, 2015.

[56] Thomas Blumensath and Mike E Davies. Iterative hard thresholding for compressed sensing. *Applied and computational harmonic analysis*, 27(3):265–274, 2009.

[57] Thomas Blumensath and Mike E Davies. Iterative thresholding for sparse approximations. *Journal of Fourier analysis and Applications*, 14(5-6):629–654, 2008.

[58] Thomas Blumensath and Mike E Davies. Gradient pursuits. *IEEE Transactions on Signal Processing*, 56(6):2370–2382, 2008.

[59] Deanna Needell and Joel A Tropp. Cosamp: Iterative signal recovery from incomplete and inaccurate samples. *Applied and computational harmonic analysis*, 26(3):301–321, 2009.

[60] Thomas H Cormen. *Introduction to algorithms*. MIT press, 2009.

[61] Thomas Blumensath. Accelerated iterative hard thresholding. *Signal Processing*, 92(3):752–756, 2012.

[62] AV Oppenheim, RW Schafer, and J Buck. Tratamiento digital de señales en tiempo discreto. *en español, Prentice Hall*, 2000.

[63] Xilinx. Zynq-7000 ap soc and 7 series devices memory interface solutions (v4.2). Release note, 2017.

[64] LauterBach. Debugging with arm coresight. Publication.

[65] Louise H Crockett, Ross A Elliot, Martin A Enderwitz, and Robert W Stewart. The zynq book. *Strathclyde Academic Media*, 2014.

[66] ARM. *ARM® Architecture Reference Manual*, armv7-a and armv7-r edition, 2014.

[67] Haoliang Qin. Boost software performance on zynq-7000 ap soc with neon. Technical report, Xilinx, jun 2014.

[68] Xilinx. *AXI Reference Guide UG761 (v13.1)*, mar 2011.

[69] ZedBoard Hardware User's Guide. Avnet inc., 2014.

[70] Digilent. Digilent pmod interface specification 1.2.0. Technical report, oct 2017.

[71] Xilinx. *Vivado Design Suite User Guide: High-Level Synthesis - UG902 (v2014.1)*, may 2014.

[72] Xilinx. Microblaze soft processor presets. Technical report, 2018.

[73] Richard Barry. *FreeRTOS reference manual: API functions and configuration options*. Real Time Engineers Limited, 2009.

[74] Richard Barry. Mastering the freertos real time kernel-a hands on tutorial guide. *Real Time Engineers Ltd*, 2016.

[75] Adam Dunkels. lwip-a lightweight tcp/ip stack. *Available from World Wide Web: http://www. sics. se/ adam/lwip/index. html*, 2002.

[76] Philip B Gibbons and Steven S Muchnick. Efficient instruction scheduling for a pipelined architecture. In *Acm sigplan notices*, volume 21, pages 11–16. ACM, 1986.

[77] Matteo Frigo, Charles E Leiserson, Harald Prokop, and Sridhar Ramachandran. Cache-oblivious algorithms. In *Foundations of Computer Science, 1999. 40th Annual Symposium on*, pages 285–297. IEEE, 1999.

[78] Wikipedia. Palabra clave — wikipedia, la enciclopedia libre, 2017. [Internet; descargado 13-julio-2018].

[79] Valery Sklyarov, Iouliia Skliarova, and Alexander Sudnitson. Fpga-based accelerators for parallel data sort. *Applied Computer Systems*, 16(1):53–63, 2014.

[80] Power Methodology Guide Xilinx. Ug786 (v13. 1) march 1, 2011.

[81] Walt Kester. Mt-001: Taking the mystery out of the infamous formula,"snr= 6.02 n+ 1.76 db,.ªnd why you should care. *REV. 0*, pages 10–03, 2005.

[82] David M Alter. Using pwm output as a digital-to-analog converter on a tms320f280x digital signal controller. *Texas Instruments Application Report, SPRAA88A*, 2008.

[83] Richard Schreier. *Understanding Delta-Sigma Data Converters*. JOHN WILEY & SONS. INC.

[84] E Perez Gonzalez and JD Reiss. Idle tone behavior in sigma delta modulation. In *Audio Engineering Society 122nd Convention Papers CD-ROM (2007 May), convention paper*, volume 7108, 2007.

[85] Maxim Intregated. Demystifying delta-sigma adcs. `"https://www.maximintegrated.com/en/app-notes/index.mvp/id/1870"`.

[86] K Lokesh Krishna, T Ramashri, and D Srihari. Design and vlsi implementation of second order sigma-delta modulation adc for i-uwb receiver.

[87] Jim Karki. Active low-pass filter design. *Texas Instruments Application Report*, 2000.

[88] James Karki. Analysis of the sallen-key architecture. *Texas Instruments Application report*, 2002.

[89] Texas Instruments and LM35 Precision Centigrade Temperature Sensors. Lm358 datasheet. *Texas Instruments Incorporated*, 1999.

[90] Active low-pass filter design and dimensioning. `http://www.beis.de/Elektronik/Filter/ActiveLPFilter.html`. [Online; accedido el 22-Abril-2018].

[91] Lm358 dual operational amplifiers. `http://www.ti.com/product/LM358/toolssoftware`. [Online; accedido el 23-Abril-2018].

[92] Katsuhiko Ogata. *Discrete-time control systems*, volume 2. Prentice Hall Englewood Cliffs, NJ, 1995.

[93] Ivar Løkken. Interpolation filters in delta-sigma dacs 1. 2006.

[94] Xilinx. 7 series dsp48e1 slice. User guide, mar 2018. UG479 (v1.10).

[95] Merrill I Skolnik. Introduction to radar. *Radar handbook*, 2, 1962.

[96] Wikipedia contributors. Human voice — Wikipedia, the free encyclopedia, 2018. [Online; accessed 7-July-2018].

[97] Ian McLoughlin. *Applied speech and audio processing: with Matlab examples*. Cambridge University Press, 2009.

[98] Saleh Faruque. *Radio frequency modulation made Easy*. Springer, 2017.

[99] Ilan Sadeh. Methods and means for image and voice compression, November 10 1998. US Patent 5,836,003.

[100] Jie Yan. Wavelet matrix. *Department of Electrical and Computer Engineering, University of Victoria*, nov 2009.

[101] Zhilin Zhang, Tzyy-Ping Jung, Scott Makeig, and Bhaskar D Rao. Compressed sensing of eeg for wireless telemonitoring with low energy consumption and inexpensive hardware. *IEEE Transactions on Biomedical Engineering*, 60(1):221–224, 2013.

[102] Velazquez Alexis, L Paredes José, and Viloria Franscisco. Implementación a nivel de hardware de la teoría de compressive sensing. *Proyecto de Grado. Escuela de Ing. Eléctrica. Universidad de Los Andes. Mérida*, 2008.

[103] MKL Intel. Intel math kernel library. 2007.

[104] Ashish Mishra, Kritika Garg, AR Asati, and Kota Solomon Raju. Hardware software co-design using profiling and clustering. In *Communication, Information & Computing Technology (ICCICT), 2012 International Conference on*, pages 1–4. IEEE, 2012.

[105] Miles Lopes. Estimating unknown sparsity in compressed sensing. In *International Conference on Machine Learning*, pages 217–225, 2013.

[106] Kris A Jamsa and Ken Cope. *Internet programming*. Delmar Thomson Learning, 1995.

[107] Juhwan Yoo, Stephen Becker, Matthew Loh, Manuel Monge, Emmanuel Candes, and Azita Emami-Neyestanak. A 100mhz–2ghz 12.5 x sub-nyquist rate receiver in 90nm cmos. In *Radio Frequency Integrated Circuits Symposium (RFIC), 2012 IEEE*, pages 31–34. IEEE, 2012.