Diseño e implementación de control para el estudio de SFDR en Time-Interleavers Aleatorios con distintos tipos de PRNGs

Universidad Nacional de Mar del plata
Facultad de Ingeniería

Autor: Ezequiel Dario Rodriguez

Director:Lucas Andrés Rabioglio

Co-Director: Matias Medina

11 de julio de 2025

Diseño e implementación de control para el estudio de SFDR en Time-Interleavers Aleatorios con distintos tipos de PRNGs

Universidad Nacional de Mar del plata
Facultad de Ingeniería

Autor: Ezequiel Dario Rodriguez

Director:Lucas Andrés Rabioglio

Co-Director: Matias Medina

11 de julio de 2025

Agradecimientos

A mi director Lucas Rabioglio y a mi co-director Matias Medina, quienes me han guiado durante todo el proceso, aportando la bibliografía necesaria para tener una base sólida, además de su tiempo, consejos, experiencia y conocimientos.

Al personal del LAC, LC Y LSC quienes me han dado el espacio para desarrollar el proyecto y también han estado siempre a disposición para cualquier consulta.

A la Cátedra de Trabajo Final que fue de gran ayuda en el desarrollo de la documentación necesaria, además de estar siempre a disposición para cualquier duda sobre el proyecto.

A mi familia, mi novia y amigos que me han apoyado durante toda la duración, no solo del proyecto, sino de toda la carrera, para que yo pueda lograr completarla.

Contenido

1.Introducción

La demanda de velocidades, cada vez mayores, para la adquisición de señales en diversos ámbitos de la electrónica dio lugar a la aparición de diversas técnicas aplicadas a los esquemas de adquisición y muestreo. Entre las técnicas más destacadas y novedosas, se encuentra la técnica Time-Interleaving.

Esta técnica de muestreo permite disminuir la velocidad de operación del sistema mediante la utilización en paralelo de ADCs idénticos. Estos ADCs reciben señales de reloj desfasadas y se seleccionan de forma alternada, lo que resulta en una adquisición de la señal a una frecuencia mayor a la de cada conversor individual, como se observa en la Fig. 1.

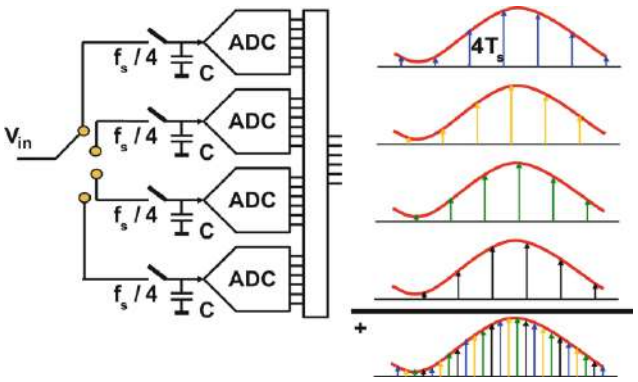


Figura 1.1. Aplicación de la Técnica Time Interleaving.

El problema del método descrito radica en la introducción de distorsión en la señal muestreada debido a que las conversiones realizadas por los ADCs no son exactamente iguales. Es decir, dado que los ADCs no son idénticos, los errores de desajuste de cada ADC (error temporal (jitter), error de offset de amplitud y error de ganancia), provocan variaciones en la toma de muestras. Estas variaciones en la señal muestreada, repetidos periódicamente debido a la selección, generan tonos interferentes en el espectro que disminuyen el rango dinámico libre de espurios (Spurious Free Dynamic Range, SFDR).

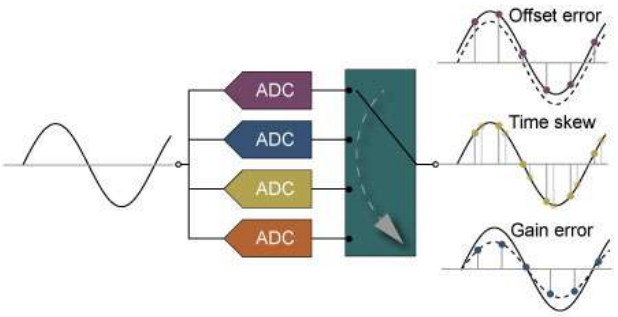


Figura 1.2. Errores asociados a la diferencia entre ADCs reales.

La técnica de Time - Interleaving Aleatorio es una alternativa a la calibración para solucionar la distorsión. En ella, se realiza una selección aleatoria del ADC a muestrear en cada ciclo de reloj, lo que se traduce en un esparcimiento del espectro del tono interfiriente, aumentando el SFDR resultante. Es aquí donde se ponen en juego las diferentes técnicas para la selección aleatoria de los ADC y cobran importancia los generadores de secuencias pseudo-aleatorias (PRNG). En este sentido es importante evaluar el efecto resultante de aplicar diferentes PRNG, tanto en simulación como en entornos de prueba.

Con el fin de evaluar el desempeño real de los generadores PRNG, aplicado a Sistemas de ADC basados en la técnica de Time Interleaving, se propone realizar un banco de prueba. Actualmente, el equipo cuenta con un sistema de adquisición, basado en Time-Interleaving, desarrollado en el proyecto final “Desarrollo de placa de adquisición basada en "Time Interleaving"” del Ing. Medina. Esta placa cuenta con cuatro ADCs, el correspondiente circuito de acondicionamiento de señal y un control a través de una FPGA.

En este informe se desarrollará el proceso para obtener el banco de prueba necesario, que cuenta con una interfaz de PC portable en la que se pueda elegir cualquier secuencia pseudo aleatoria almacenada en un archivo de texto a enviar al sistema de adquisición, la cantidad de muestras a tomar y elegir o no aplicar un factor de diezmado. También, el cambio en el control de la FPGA para poder obtener más cantidad de muestras utilizando una memoria externa al chip pero dentro de la placa de desarrollo DE0-Nano y mejoras posibles sobre la versión inicial del sistema de adquisición.

1.1 Propósito del documento

Este documento corresponde al informe titulado *Diseño e implementación de control para el estudio de SFDR en Time-Interleavers Aleatorios con distintos tipos de PRNGs* y tiene el propósito de resumir el desarrollo realizado para poder cumplir con los objetivos del proyecto, además de proveer información de su funcionamiento y de su construcción. La estructura de este informe se basa en los documentos Especificación de Requerimientos (ER), Especificación Funcional (EF), Especificación Técnica (ET) y Plan de pruebas (PP).

También apunta a mostrar el desarrollo de un proyecto final de la carrera Ingeniería Electrónica desde una perspectiva de gestión de proyectos, mostrando información como un plan de proyecto en donde se estiman los tiempos de cada etapa del trabajo.

Este documento está dirigido a los desarrolladores, solicitantes del proyecto y a la comisión designada para su evaluación.

Alcance del proyecto

Se busca lograr la definición de los requerimientos, implementación y presentación de un banco de prueba para evaluar el desempeño general de los generadores PRNG aplicado a Sistemas de ADCs basados en la técnica de Time Interleaving.

Para el banco de prueba el equipo ya cuenta con un sistema de adquisición basado en Time-Interleaving, desarrollado en el proyecto final “Desarrollo de placa de adquisición basada en "Time Interleaving"” del Ing. Medina, que además, es compatible con la placa de desarrollo DE0- Nano (FPGA).

El banco de prueba debe automatizar las mediciones y calcular el SFDR mediante la realización de una interfaz de PC portable. Además, en el proyecto final se debe desarrollar el código VHDL correspondiente para almacenar las muestras en la memoria SDRAM IS42S16160G de la FPGA y controlar la identificación de los ADCs para el ordenamiento de las muestras (que se realiza en la interfaz) de la placa de adquisición a partir de una PRNG que será recibida desde la interfaz de usuario.

Por último, se busca realizar una revisión crítica de la placa de adquisición para poder diseñar mejoras.

1.2 Personal Involucrado

Nombre	Ezequiel Dario Rodriguez
Rol	Desarrollador
Categoría Profesional	Estudiante
Responsabilidad	Desarrollo y diseño

Información de contacto	ezequiel.rodriguez18@gmail.com

Nombre	Lucas Rabioglio
Rol	Director
Categoría Profesional	Ingeniero
Responsabilidad	Tutelar y orientar en el diseño y seguimiento del desarrollo del proyecto
Información de contacto	lucas.rabioglio@fi.mdp.edu.ar

Nombre	Matías Medina
Rol	Co-director
Categoría Profesional	Ingeniero
Responsabilidad	Tutelar y orientar en el diseño y seguimiento del desarrollo del proyecto
Información de contacto	matias.medina1302@gmail.com

1.3 Definiciones, siglas y abreviaturas

- **ADC:** Circuito integrado, Conversor Analógico a Digital.
- ***Time Interleaving:*** Técnica mediante la cual se aumenta la frecuencia de muestreo efectiva de un sistema de adquisición.
- **PRNG**: (*Pseudo Random Number Generator*) Generador de Números Pseudo Aleatorio.
- **FPGA**: (*Field Programmable Gate Array*) Dispositivo lógico programable.
- ***VHDL:*** Lenguaje de descripción de hardware.
- **DE0-Nano:** Placa de desarrollo FPGA.

▪ **SFDR:** (Spurious Free Dynamic Range) Rango dinámico libre de espurios.

▪ **RAM:** Memoria de acceso aleatorio.

▪ **SDRAM:** Memoria de acceso aleatorio dinámica y síncrona.

▪**Widget:** Pequeño componente de software de una interfaz que proporciona acceso rápido a una cierta función.

▪ **Bandera o flag:** Palabra de 8 bits que le indica al control de la FPGA que decisión tomar, como por ejemplo : leer la memoria o escribirla.

▪ **ID:** Identificador de 4 bits del ADC que realizó el muestreo. Puede ser 0001, 0010, 0100,1000.

2.Anteproyecto

En la figura 2.1 se puede observar un esquema general inicial propuesto para la resolución del proyecto. El primer recuadro de izquierda a derecha representa la placa de adquisición que se comunica con la placa DE0-nano. En ella se encuentran las funciones y requerimientos que debe cumplir en esta etapa tales como el almacenamiento de muestras, control de ADC, etc. En el proyecto final del Ing.Medina, la generación de la secuencia pseudoaleatoria se obtenía en el mismo código VHDL dentro de la FPGA y se almacenaban las muestras en la propia memoria RAM (74 Kbyte de capacidad) de la FPGA.

En este caso, se busca recibir la PRNG generada desde la PC, almacenar esa secuencia en la memoria RAM de la FPGA y por otro lado, controlar la memoria SDRAM (32 Mbyte) externa que contiene más capacidad para almacenar las muestras. Además esto brinda la facilidad de poder utilizar cualquier PRNG, ya que será generada en la PC con cualquier tipo de software. La DE0-nano se comunica con la interfaz gráfica por medio del módulo UART/USB que integra la placa de adquisición. El tercer recuadro representa la interfaz y las funcionalidades que debe cumplir. La interfaz debe permitir elegir la cantidad de muestras a tomar, aplicar o no un factor de diezmado, elegir la secuencia PRNG almacenada en un archivo de la PC, comunicarse eficazmente con el usuario y la FPGA, para finalmente calcular y graficar la SFDR obtenida.

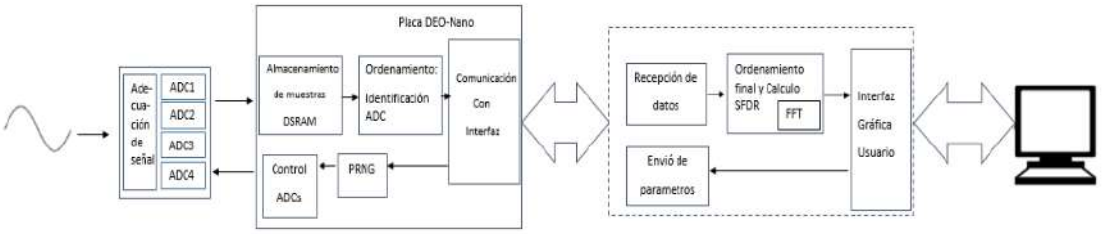


Figura 2.1. Esquema general del banco de prueba.

Finalmente, el proyecto concluye con propuestas de mejoras para un nuevo diseño de la placa de adquisición. Dentro de esas mejoras se encuentra la disminución del área PCB, la mejora en la integridad de la señal de clock de cada ADC y una mejora en el bloque de adaptación y filtrado de la señal de entrada.

En las siguientes dos secciones se pondrá énfasis en los requerimientos que tiene el proyecto y el cronograma diagramado para cumplir con su finalización.

2.1 Requerimientos

El proyecto tiene 12 requerimientos funcionales. Los primeros 5 requerimientos son destinados al código VHDL que ejecuta la placa DE0-nano mientras que el resto están dirigidos a la Interfaz gráfica. Ellos son: Controlar la memoria SDRAM para el almacenamiento y lectura de las muestras (RF01), controlar la memoria RAM del propio chip para almacenar la PRNG y lograr controlar el muestreo de los ADCs(RF02 y RF03), almacenar y utilizar los parámetros de ajuste como la cantidad de muestras (RF04), notificar a la interfaz cuando se termine el almacenamiento de las muestras en la SDRAM (RF05), por otro lado, la interfaz debe notificar cuando la placa de adquisición esté conectada (RF06), bloquear el resto de las funciones de la Interfaz si la cantidad de muestras elegidas no coincide con la longitud de la PRNG (RF07), notificar al usuario cuando finalice el proceso de muestreo (RF08), notificar cuando se estén transmitiendo datos hacia la PC (RF09), guardar las muestras en un archivo de texto en una ruta especificada dentro de la PC (RF10), aplicar el diezmado seleccionado, calcular la SFDR y graficarla (RF11 y RF012).

Además, el proyecto tiene un requerimiento no funcional y 4 requerimientos de rendimiento:

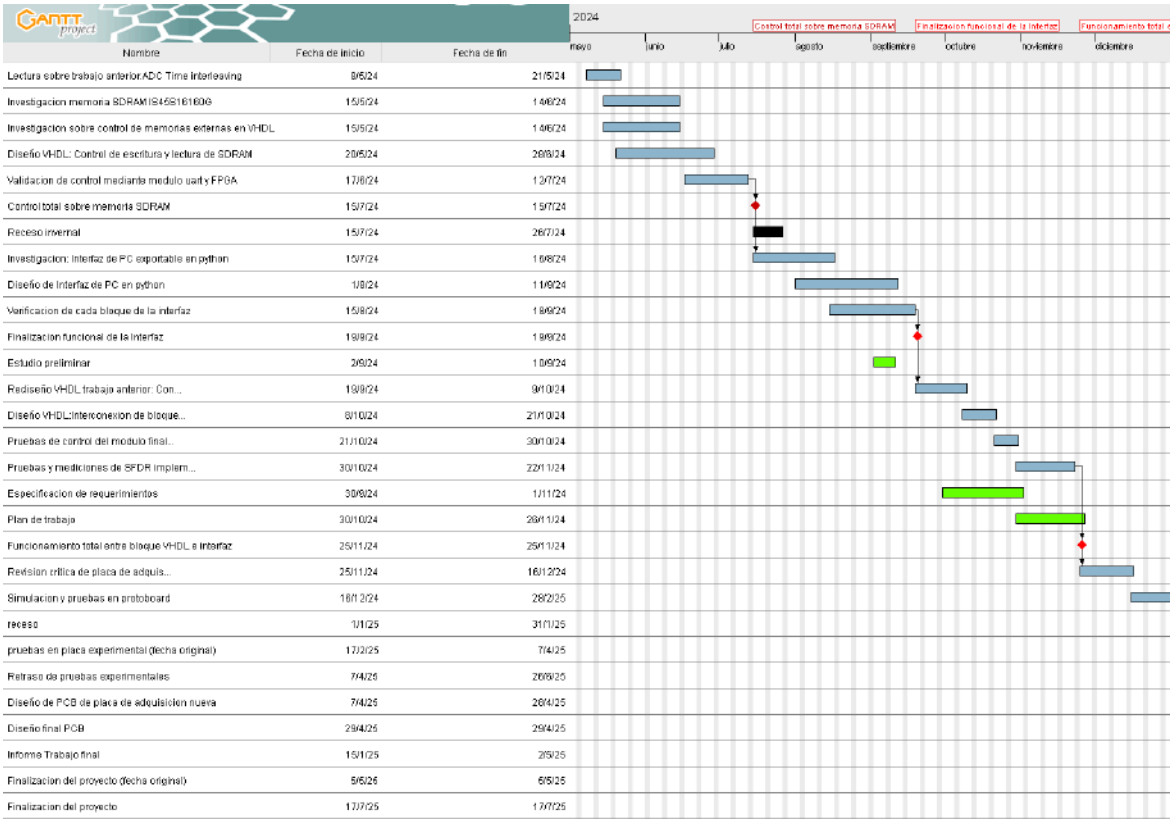
La disminución del tamaño del diseño para la futura placa de adquisición (RNF1), la transferencia de datos entre la placa y PC debe ser de 115200 baudios (RR1), la mejora en

la integridad de la señal de clock (RR2), y la mejora en el bloque de adecuación de la señal (RR3). Para más detalle de los requerimientos funcionales, en la sección *Apéndice*, se encuentra el documento completo.

2.2 Plan de proyecto

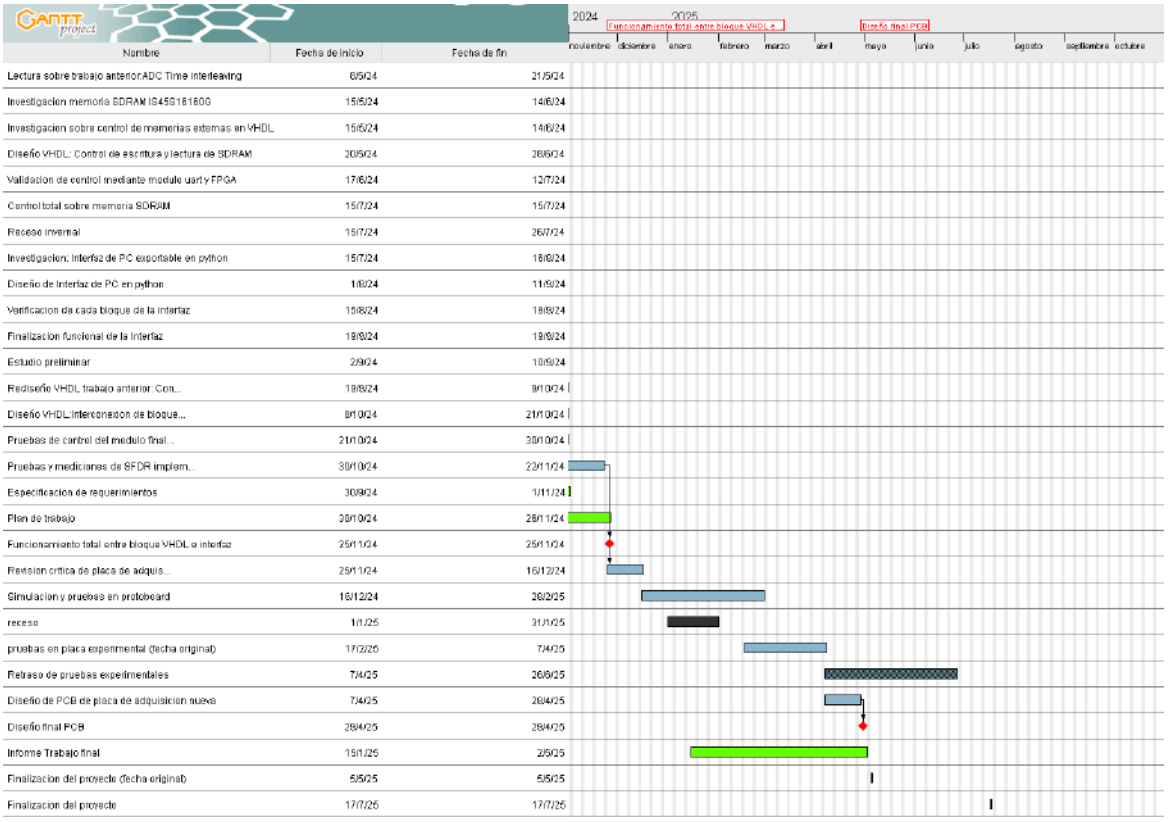
En el diagrama de gantt que se ve en la figura 2.3 se puede observar el desarrollo del proyecto. Se puede dividir el mismo en 4 etapas. La primera etapa consta de la investigación para la utilización de la memoria SDRAM y el control sobre ella. En la segunda etapa se diseña la interfaz gráfica. La tercera etapa consta de la integración del diseño VHDL completo, sumado a la utilización de la interfaz gráfica. La cuarta etapa consta del análisis y prueba de las posibles mejoras enunciadas para un nuevo diseño de la placa de adquisición.

El proyecto tuvo una duración final de 14 meses. El plan consta de algunas modificaciones con respecto al plan original que son aclaradas en las tareas finales del diagrama.



Nombre	Fecha de inicio	Fecha de fin
Lectura sobre trabajo anterior:ADC Time Interleaving	8/5/24	21/5/24
Investigacion memoria SDRAM IS45S16160G	15/5/24	14/6/24
Investigacion sobre control de memorias externas en VHDL	15/5/24	14/6/24
Diseño VHDL: Control de escritura y lectura de SDRAM	20/5/24	28/6/24
Validacion de control mediante modulo uart y FPGA	17/6/24	12/7/24
Control total sobre memoria SDRAM	15/7/24	15/7/24
Receso invernal	15/7/24	26/7/24
Investigacion: Interfaz de PC exportable en python	15/7/24	18/8/24
Diseño de interfaz de PC en python	1/8/24	11/9/24
Verificacion de cada bloque de la interfaz	15/8/24	18/8/24
Finalizacion funcional de la interfaz	19/8/24	19/8/24
Estudio preliminar	2/9/24	10/9/24
Rediseño VHDL trabajo anterior: Con...	19/8/24	9/10/24
Diseño VHDL:Interconexion de bloque...	9/10/24	21/10/24
Pruebas de control del modulo final...	21/10/24	30/10/24
Pruebas y mediciones de SFDR implem...	30/10/24	22/11/24
Especificacion de requerimientos	30/8/24	1/11/24
Plan de trabajo	30/10/24	28/11/24
Funcionamiento total entre bloque VHDL e interfaz	25/11/24	25/11/24
Revision critica de placa de adquis...	25/11/24	16/12/24
Simulacion y pruebas en protoboard	16/12/24	28/2/25
receso	1/1/25	31/1/25
pruebas en placa experimental (fecha original)	17/2/25	7/4/25
Retraso de pruebas experimentales	7/4/25	26/6/25
Diseño de PCB de placa de adquisicion nueva	7/4/25	28/4/25
Diseño final PCB	28/4/25	29/4/25
Informe Trabajo final	15/1/25	2/5/25
Finalizacion del proyecto (fecha original)	5/5/25	5/5/25
Finalizacion del proyecto	17/7/25	17/7/25

Figura 2.2. Diagrama de gantt parte 1.



Nombre	Fecha de inicio	Fecha de fin
Lectura sobre trabajo anterior:ADC Time Interleaving	8/5/24	21/5/24
Investigacion memoria SDRAM IS45S16160G	15/5/24	14/6/24
Investigacion sobre control de memorias externas en VHDL	15/5/24	14/6/24
Diseño VHDL: Control de escritura y lectura de SDRAM	20/5/24	28/6/24
Validacion de control mediante modulo uart y FPGA	17/6/24	12/7/24
Control total sobre memoria SDRAM	15/7/24	15/7/24
Receso invernal	15/7/24	26/7/24
Investigacion: Interfaz de PC exportable en python	15/7/24	18/8/24
Diseño de interfaz de PC en python	1/8/24	11/9/24
Verificacion de cada bloque de la interfaz	15/8/24	18/8/24
Finalizacion funcional de la interfaz	19/8/24	19/8/24
Estudio preliminar	2/9/24	10/9/24
Rediseño VHDL trabajo anterior: Con...	19/8/24	9/10/24
Diseño VHDL:Interconexion de bloque...	9/10/24	21/10/24
Pruebas de control del modulo final...	21/10/24	30/10/24
Pruebas y mediciones de SFDR implem...	30/10/24	22/11/24
Especificacion de requerimientos	30/8/24	1/11/24
Plan de trabajo	30/10/24	28/11/24
Funcionamiento total entre bloque VHDL e interfaz	25/11/24	25/11/24
Revision critica de placa de adquis...	25/11/24	16/12/24
Simulacion y pruebas en protoboard	16/12/24	28/2/25
receso	1/1/25	31/1/25
pruebas en placa experimental (fecha original)	17/2/25	7/4/25
Retraso de pruebas experimentales	7/4/25	26/6/25
Diseño de PCB de placa de adquisicion nueva	7/4/25	28/4/25
Diseño final PCB	28/4/25	28/4/25
Informe Trabajo final	15/1/25	2/5/25
Finalizacion del proyecto (fecha original)	5/5/25	5/5/25
Finalizacion del proyecto	17/7/25	17/7/25

Figura 2.3. Diagrama de gantt parte 2.

3.Proyecto

De acuerdo a la implementación del banco de prueba, se dividirá el desarrollo del proyecto en tres bloques : Código VHDL, Interfaz gráfica y mejoras en la placa de adquisición. El primer bloque consiste en la investigación realizada para el control de la memoria SDRAM y en cómo se implementó la totalidad del código para cumplir con los requerimientos previamente descritos. En el segundo bloque se explicará el desarrollo para la interfaz gráfica y como se logró cada una de las funcionalidades necesarias. Finalmente se explicarán las propuestas y las pruebas realizadas para contar con las mejoras de una futura placa de adquisición.

3.1 Código VHDL

Como se explicó en la sección de ANTEPROYECTO, se busca controlar la memoria SDRAM externa para tener más capacidad a la hora de almacenar las muestras. Además, con el uso de una interfaz gráfica en la PC, se facilita la generación de cualquier tipo de PRNG que provenga desde la PC.

Para comenzar, lo primero que se realizó fue una investigación sobre cómo controlar la memoria **SDRAM IS45S16160G-7TLA1** que contiene embebida la placa DE0-nano por fuera del chip CYCLONE IV E.

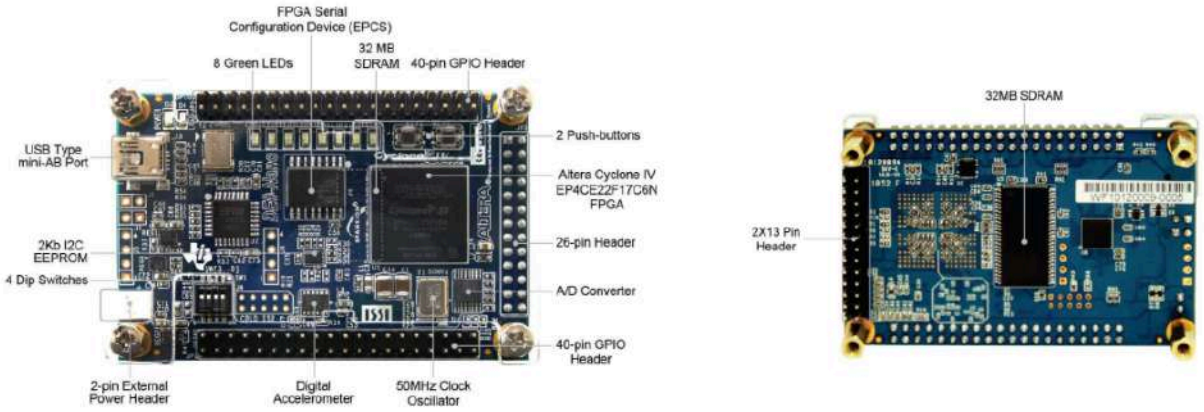


Figura 3.1. Placa de desarrollo DE0-nano.

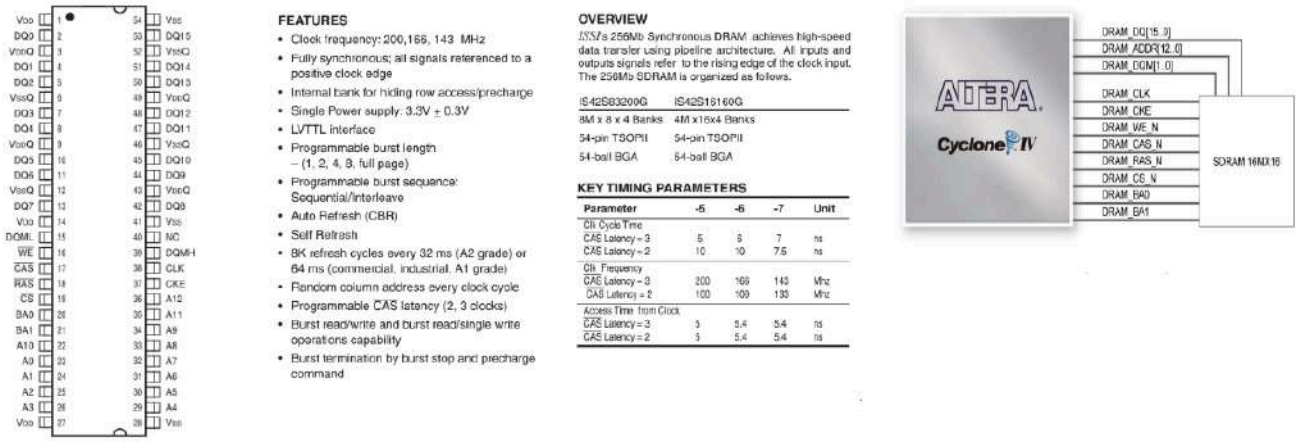


Figura 3.2. Características de la memoria SDRAM IS45S16160G.

La memoria tiene una capacidad de 256Mb o 32 MB y en particular la integrada en la placa DE0-nano, una frecuencia máxima de clock de 143 MHz. Tiene la capacidad de refrescar manualmente o automáticamente los datos almacenados. La memoria está subdividida en 4 bancos de 8192 filas y 512 columnas de 16 bits. Estas características son importantes para entender cómo debe ser el bloque que se encargue de administrar la escritura y lectura de la memoria.

La memoria tiene un tiempo de inicialización y luego se pone en modo registro para configurar su modo de operación. Como se ve en la figura 3.3 colocando los bits correctos en el bus de adress se configuran parámetros como la longitud de la ráfaga de lectura o

escritura, el tipo de ráfaga, la latencia, el modo de operación o el modo de ráfaga de escritura.

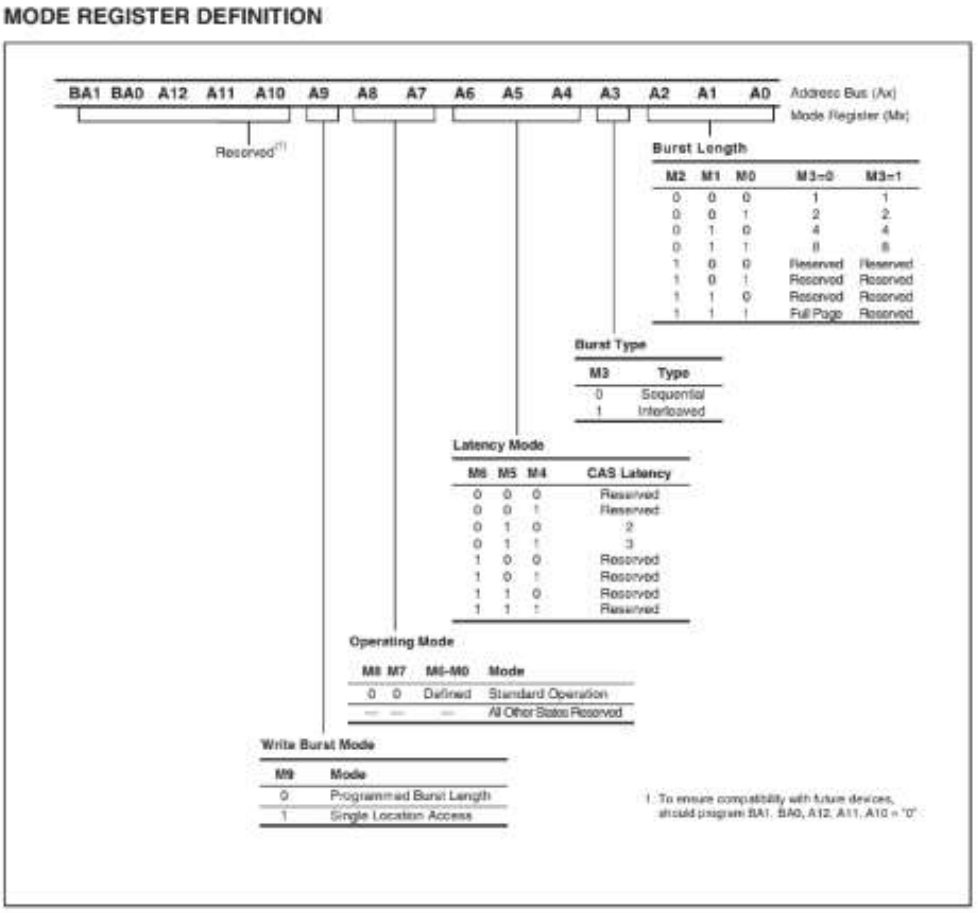


Figura 3.3. Modo registro de la memoria SDRAM.

Por otro lado, cada acción que efectúe la memoria, es controlada por las entradas de señalización y se necesitan tiempos específicos para realizar cada comando que dependen de la señal de clock. En la tabla siguiente se observa la tabla de verdad de comandos dependiendo de los valores encontrados en las entradas de señalización $\overline{CS}$, $\overline{CAS}$, $\overline{RAS}$, $\overline{WE}$.

COMMAND TRUTH TABLE

Function	CKE n – 1	CKE n	$\overline{CS}$	$\overline{RAS}$	$\overline{CAS}$	$\overline{WE}$	BA1	BA0	A10	A12, A11 A9 - A0
Device deselect (DESL)	H	×	H	×	×	×	×	×	×	×
No operation (NOP)	H	×	L	H	H	H	×	×	×	×
Burst stop (BST)	H	×	L	H	H	L	×	×	×	×
Read	H	×	L	H	L	H	V	V	L	V
Read with auto precharge	H	×	L	H	L	H	V	V	H	V
Write	H	×	L	H	L	L	V	V	L	V
Write with auto precharge	H	×	L	H	L	L	V	V	H	V
Bank activate (ACT)	H	×	L	L	H	H	V	V	V	V
Precharge select bank (PRE)	H	×	L	L	H	L	V	V	L	×
Precharge all banks (PALL)	H	×	L	L	H	L	×	×	H	×
CBR Auto-Refresh (REF)	H	H	L	L	L	H	×	×	×	×
Self-Refresh (SELF)	H	L	L	L	L	H	×	×	×	×
Mode register set (MRS)	H	×	L	L	L	L	L	L	L	V

Note: H=V_{IH}, L=V_{IL} x= V_{IH} or V_{IL}, V = Valid Data.

Figura 3.4. Tabla de verdad de comandos de la memoria SDRAM.

El dispositivo funciona como una gran máquina de estados como se ve en el esquema de la figura 3.5. El bloque VHDL que controla la memoria debe tener las mismas salidas que tiene la memoria como entrada y debe ser capaz de controlar los tiempos de señalización correctos para que la memoria funcione correctamente. Debe funcionar como interfaz entre el diseño VHDL y la memoria física.

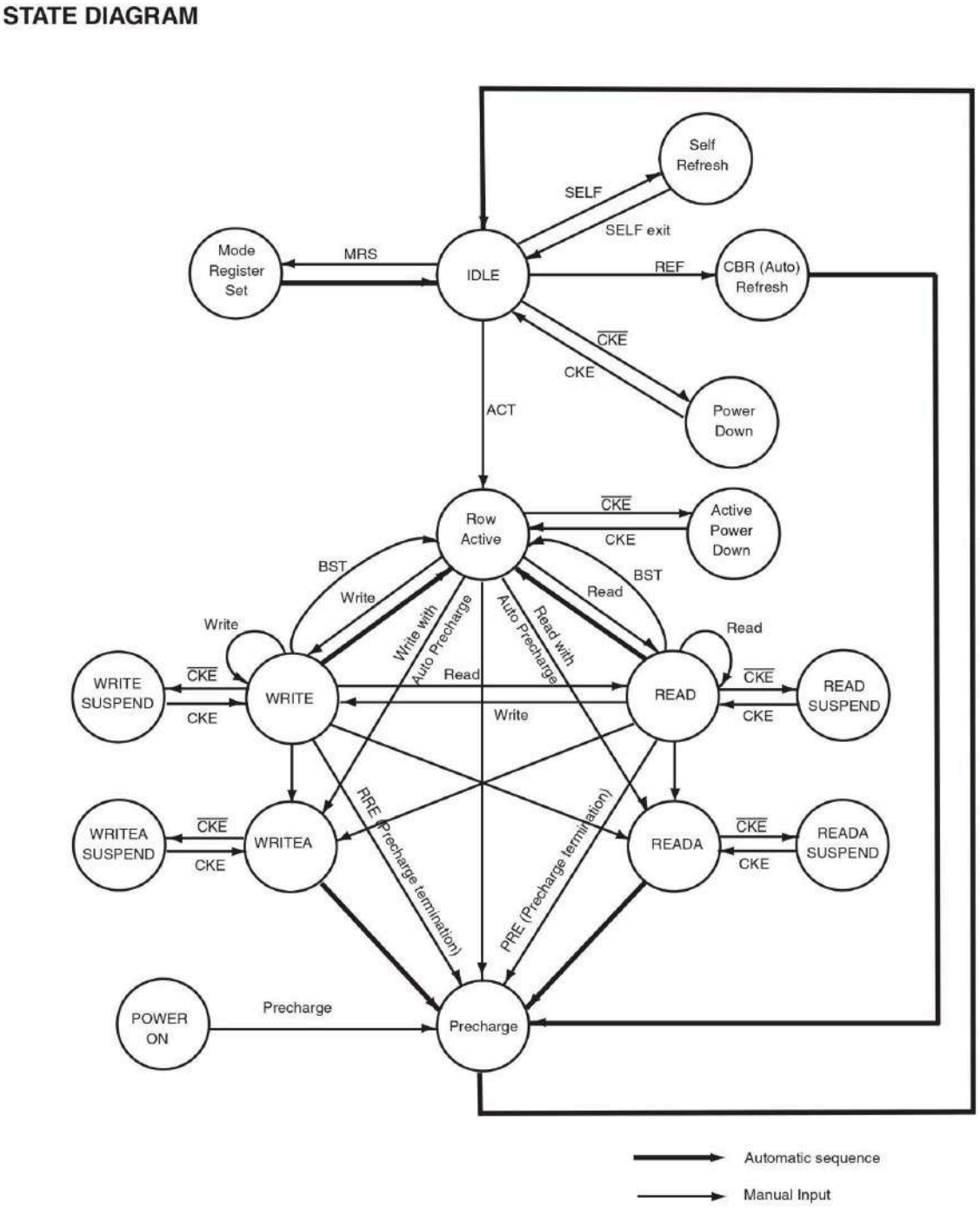


Figura 3.5. Diagrama de estado de la memoria SDRAM.

Antes de realizar un código VHDL con toda la complejidad descrita para controlar la memoria desde cero, se investigó si existía algún bloque IP utilizable que simplifique el control de la memoria.

El software Quartus II (donde se describe el código VHDL) cuenta con una función llamada *Megawizard Plug in Manager* que genera bloques de memoria de las FPGA disponibles dentro de su librería. Aunque, dentro de las opciones se encontraba la FPGA

utilizada en este proyecto, el bloque de memoria no era compatible con la SDRAM que se encuentra en la placa de0-nano. Por lo que se descartó esta opción.

Se decide entonces basarse en un controlador provisto por el LSC, el cual estaba destinado a manejar la SDRAM de similares características en una placa de desarrollo antigua que contenía una FPGA CYCLONE II. Rediseñando el bus de address, de datos y configurando los tiempos para cada comando, se ajustó el controlador a la SDRAM de la DE0-Nano.

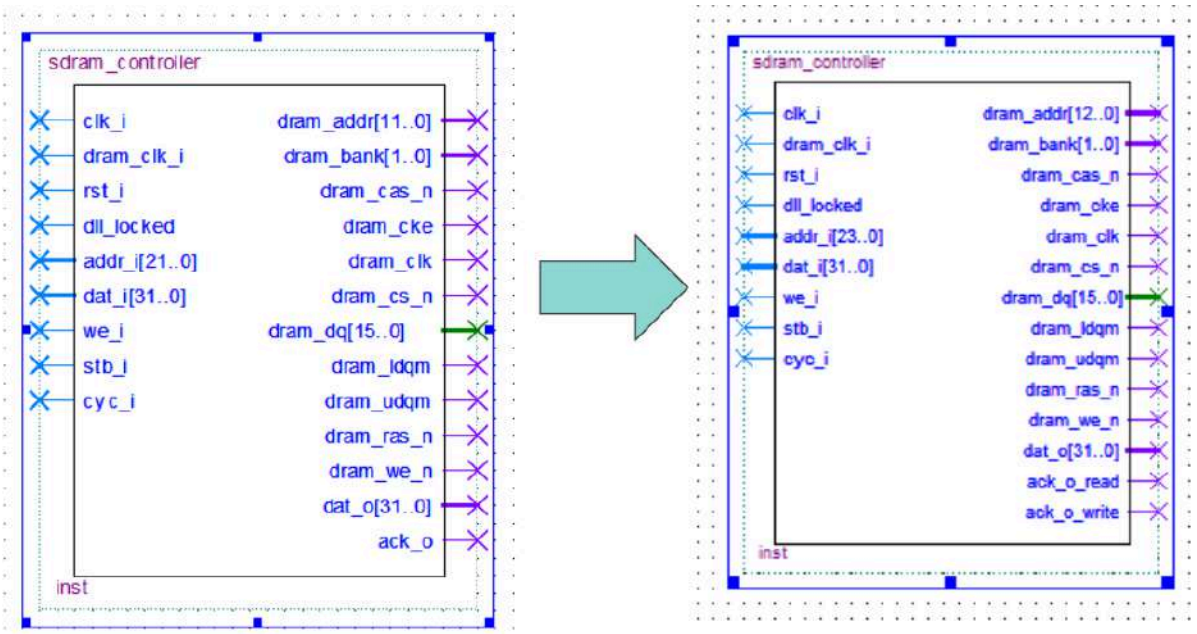


Figura 3.6. Bloque de interfaz a la memoria SDRAM.

Se agregaron dos salidas *ack_o_read* y *ack_o_write* para que el bloque avise cuando termine el proceso de escritura y o de lectura. La entrada de datos *dat_i[31..0]* está diseñada para escribir y leer de a dos datos de 16 bits, ya que en su configuración normal, la memoria lee y escribe de a dos datos. La entrada de *address* es de 24 bits, pero la memoria tiene un bus de 13 entradas para el address. Esto se debe a que la memoria lee en dos periodos de reloj la dirección. En un primer periodo lee en el bus de adress la columna y el banco, mientras que en un segundo periodo lee la fila. Tener un bus de entrada más grande permite colocar la dirección que se quiere utilizar solo una vez y no tener que cambiar la dirección por cada estado del clock para indicar fila, columna y banco. Mediante las entradas *we_i, stb_i y cyc_i* se controlaran las salidas *dram_cs_n, dram_ras_n, dram_cas_n y dram_we_n* que corresponden a las entradas de señalización de la memoria para efectuar cada comando.

Una vez obtenido el bloque para controlar la memoria, había que probar su funcionamiento, para ello, se diseñó en primera instancia el sistema que se ve en la figura 3.7.

Un bloque **pll** para crear la señal de clock necesaria. Dos bloques para controlar la escritura y lectura de la memoria. El bloque **enable_flag** recibe las palabras y selecciona si la memoria va a ser escrita o leída. El bloque **wr_and_r** controla la señalización para la escritura y lectura del bloque **sdram_controller**. El sistema será controlado enviando los datos mediante módulo UART/USB a un pin del GPIO de la FPGA. La entrada, se dirige al módulo **uart_rx**, que convierte la palabra que llega en formato serie, a paralelo . Del otro lado, el bloque **shif_tregister** convertiría los 32 bits de datos leídos a palabras de 8 bits. El bloque **uart_tx** convierte las palabras de 8 bit paralelo a bits en serie para ser transmitidas a través de un módulo UART/USB. Este módulo, se conectaría a la PC, donde se podría leer los datos en el software Docklight. Los FF 's agregados entre el bloque **wr_and_r** y **sdram_controller** funcionan para mantener el estado del pulsado hasta que se finalice la escritura o lectura. La compuerta OR que realimenta el circuito resetea los FF 's. Este sistema primero se simuló en *Quartus II* y luego se comprobó físicamente probando distintas direcciones y datos al conectar la FPGA al módulo UART/USB y este a la PC. Mediante el software Docklight, que permite enviar y recibir palabras hexadecimales en formato serie por el puerto USB, se puede observar la lectura de los datos guardados en memoria.

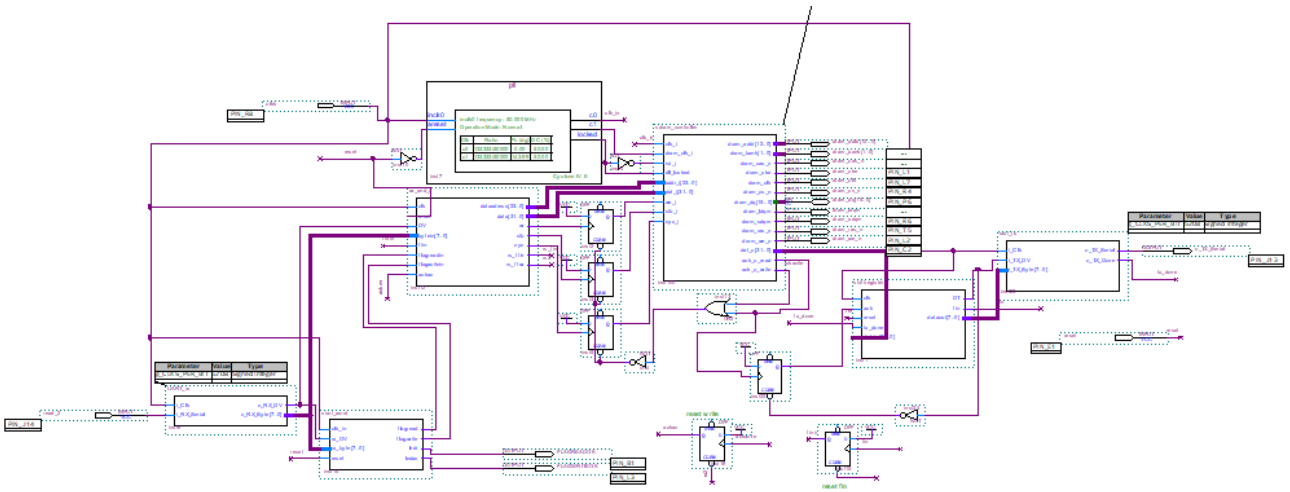


Figura 3.7. Sistema para prueba de control de memoria SDRAM.

En la figura 3.8 se puede observar la interfaz del software Docklight. Mediante el envío del comando 53 representado en hexadecimal, se comunica de que, a continuación

se envían 6 datos de 16 bits para guardar en la memoria. Luego con el envío del comando A3, se avisa al sistema que se quiere leer la memoria. En rojo, se observan los datos recibidos que se han escrito correctamente. En la esquina superior derecha se puede observar la velocidad de transmisión ajustada en 9600 baudios. Para ajustar esa velocidad en el sistema VHDL, tanto el bloque uart_tx como uart_rx tiene un contador interno para enviar y recibir los bits de acuerdo a la relación que existe entre el clock interno del código VHDL y la velocidad a la que se quiere transmitir y recibir datos. Esta variable de ajuste permitirá más adelante, transmitir y recibir los datos a 115200 baudios.

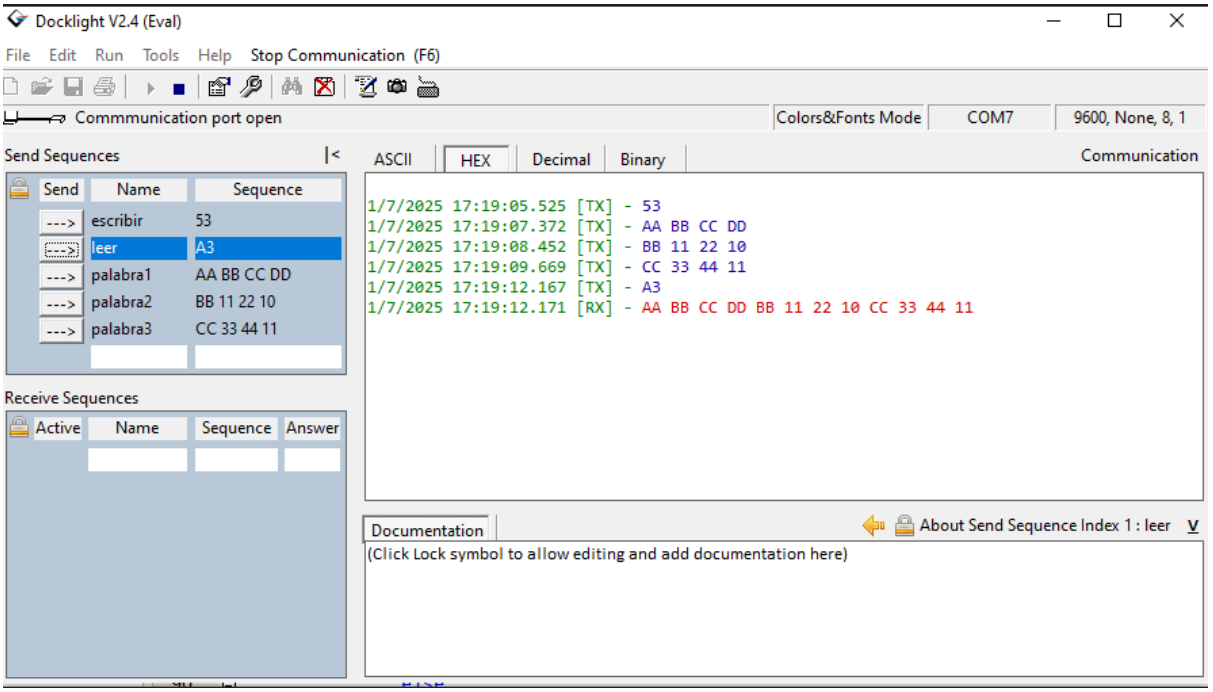


Figura 3.8. Prueba de control de memoria SDRAM.

Una vez comprobado el funcionamiento del control de memoria, en la siguiente etapa se diseñaron y comprobaron en otro archivo los bloques necesarios para recibir la secuencia PRNG, almacenarla en la memoria RAM de la propia FPGA y enviar la secuencia al siguiente bloque que controla la selección de los ADC 's.

Se diseñó el sistema que se ve en la figura 3.9.

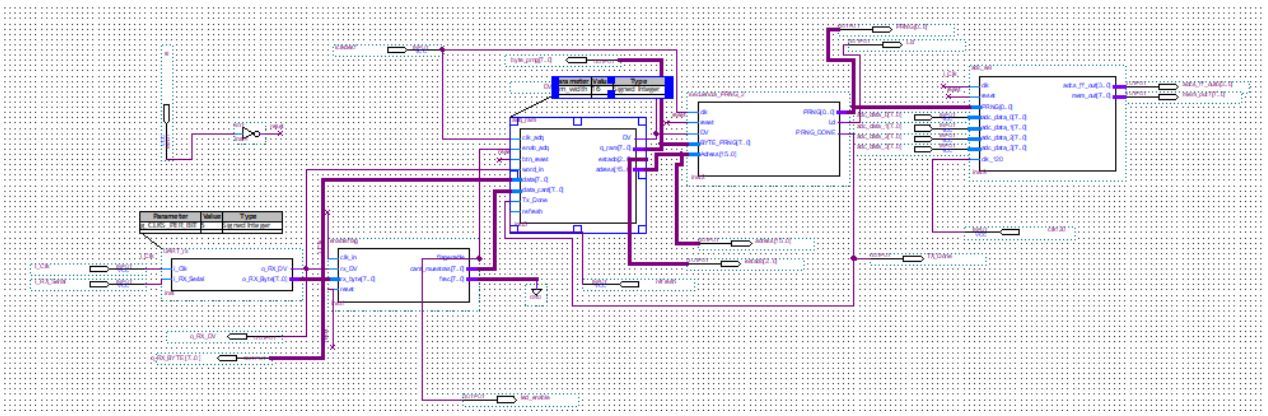


Figura 3.9. Prueba de control de memoria RAM y selección de ADCs.

El bloque **uart_rx** se encargará de recibir los bits en serie transmitidos por el módulo UART/USB. El bloque **enable_flag** ahora, tendrá una función más importante en el diseño final, y se encarga de recibir las banderas que se envían desde la PC para decidir qué acción debe realizar el sistema. En este caso, activa al bloque siguiente para que comience a almacenar los datos siguientes que corresponden a la secuencia PRNG. El bloque **adq_ram**, es el bloque encargado de instanciar la propia memoria RAM de la FPGA. Luego de recibir la PRNG, continuará a un estado donde se leerá la memoria y se enviará la secuencia en orden al bloque siguiente. El bloque **secuencia_PRNG_2** tiene la función de convertir la palabra de 8 bits paralelo que corresponde a la secuencia, en bits en serie. Finalmente, el bloque **adc_sel**, selecciona el ADC correspondiente de acuerdo al valor de la secuencia que tiene en su entrada. Además, recibe las muestras, y entrega en su salida al doble de velocidad la muestra tomada y el ID que indica que ADC muestreó en ese momento, aprovechando el semiciclo positivo y negativo de la señal de clock.La selección de los ADC sigue el comportamiento del esquema que se ve en la figura 3.10.

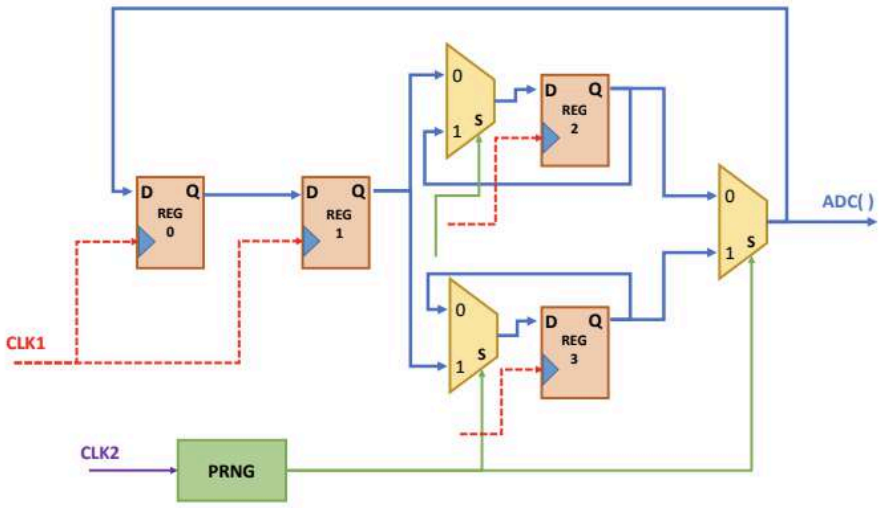


Figura 3.10. Esquema de selección de los ADC.

Este diseño fue probado por simulación como se ven en las siguientes figuras utilizando el software *Modelsim* de Altera que trabaja en conjunto con el *Quartus II*. Las primeras dos señales corresponden a la frecuencia del sistema de 60 MHz y la misma desfasada. La señal desfasada la utilizará el bloque **secuencia_PRNG_2** para entregar en el momento correcto al valor de la PRNG para su adquisición en el siguiente bloque. Luego se ve una señal de 120 MHz para simular el clock de la memoria SDRAM. La señal *rs* representa el reset, la señal *RX_Serial* representa la entrada del bloque **uart_rx** y las salidas *RX_Byte y RX_DV* son las salidas que representan la palabra de 8 bits recibidas desde la PC y el aviso de que esa palabra es válida para leer. A continuación siguen las señales *flag_enable* correspondiente a la salida del bloque **enable_flag** el cual le avisa al bloque **adq_ram** que a continuación llegan las secuencia PRNG a almacenar.

Correspondiente al bloque **adq_ram** tenemos la entrada *refresh* que representa el si la memoria SDRAM se encuentra o no en ese estado y las salidas *byte_prng, DV_TX* que representan cuando se encuentra la palabra prng de 8 bits lista en su salida y el aviso de que se puede adquirir esta palabra por el siguiente bloque. Además, las salidas *estado y adress* que representan en qué estado se encuentra el bloque **adq_ram** (escribiendo o almacenando) y el *adress* que representa las direcciones de la memoria. Estas dos son salidas que se utilizan solamente para la simulación.

Del bloque **secuencia_PRNG_2** tenemos la salida PRNG que representan cada blit de la palabra de 8 bits de la PRNG y la salida *TX_Done* que le avisa al bloque **adq_ram** para que envíe otra palabra. Finalmente del bloque **adc_sel** se observan las entradas de adquisición la salida *adc_ff_out y mem_out* que representan la selección de cada ADC y los datos que se entregarán a la memoria sdram, respectivamente.

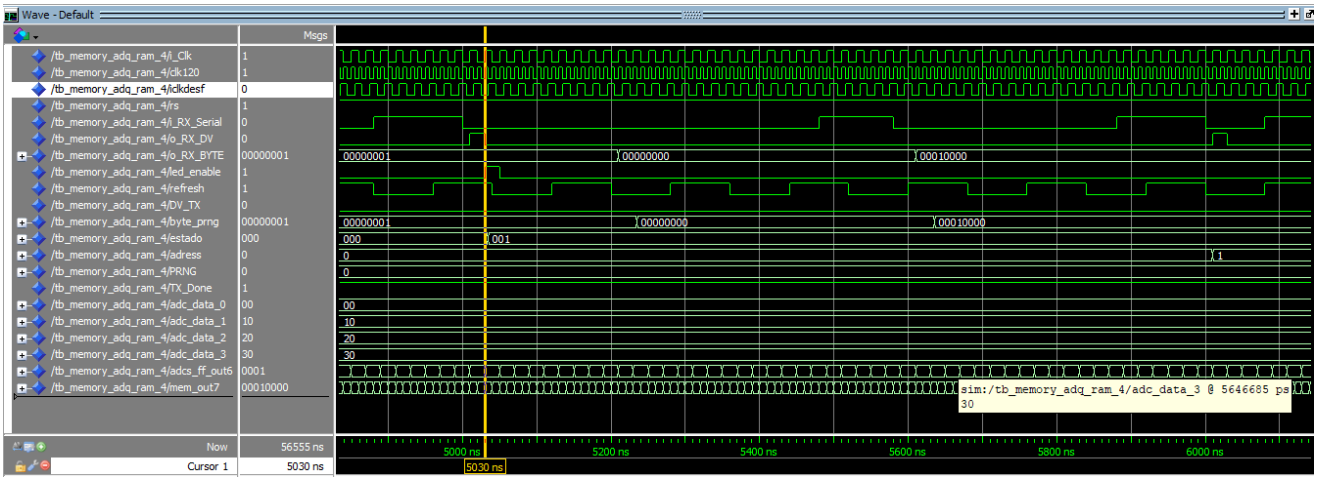


Figura 3.11 Simulación en testbench utilizando modelsim.

En la figura 3.11 se encuentra el bloque **adq_ram** en el estado 001 que quiere decir que ese está almacenando la PRNG. En la figura 3.12 se encuentra el bloque **adq_ram** en el estado 010 que quiere decir que la memoria está entregando la PRNG a los bloques siguientes.

Para las entradas de adquisición se utilizaron datos fijos, y en las últimas dos señales se puede ver cómo a partir de el bit de PRNG recibido, el bloque **adc_sel** elige el ADC a muestrear y envía a la memoria SDRAM el dato y el ID del adc. Este proceso sucede en todo momento. Al unir todo el sistema se deberá avisar al bloque que controla la memoria SDRAM que solo adquiera los datos procedentes del bloque **adc_sel** cuando el bloque **adq_ram** esté en el estado de enviar la PRNG.

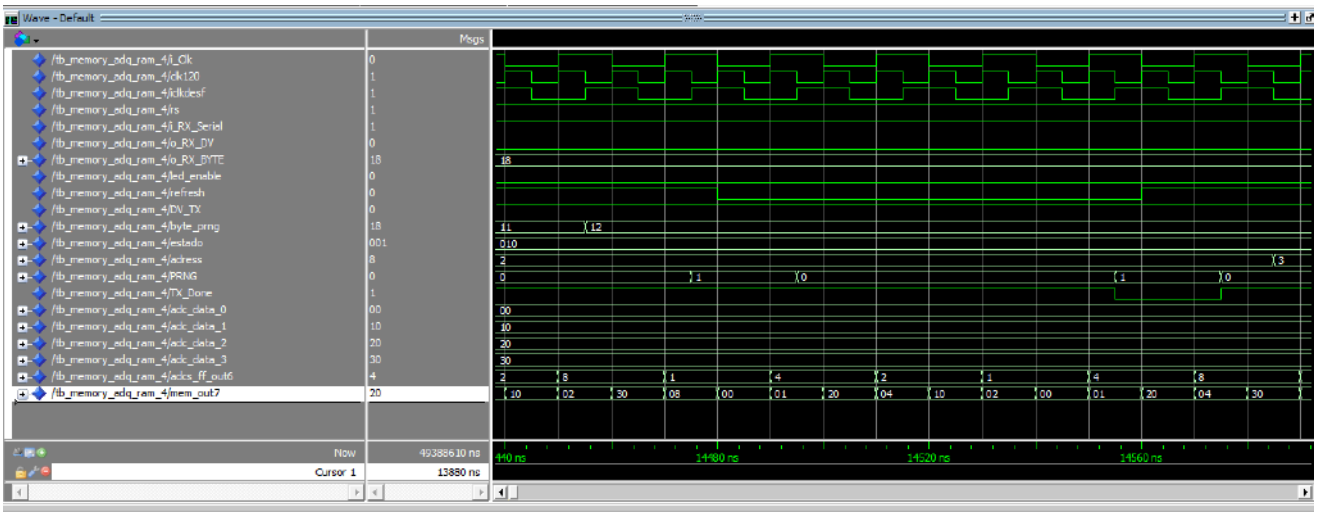


Figura 3.12. Simulación en testbench utilizando modelsim.

Finalmente, se debían integrar los dos sistemas en uno solo. La dificultad en este punto es sincronizar las señales de clock y sincronizar la escritura de la memoria SDRAM. El sistema está sincronizado a una frecuencia de 60 MHz de clock, que corresponde a la frecuencia a la que se adquiere cada muestra. Pero el bloque **adc_sel**, al tener que enviar la muestra y el ID del ADC utilizado, entrega los datos en su salida al doble de velocidad (120 MHz). Por lo tanto, el bloque **sdram_controller** debe funcionar a 120 MHz para no perder ningún dato en su escritura.

Aquí surgen dos problemas:

1) La memoria tiene la posibilidad de escribir por ráfagas. De a 2, 4, 8 o ocupar todas las posiciones de una página (full page). Pero la cantidad de datos a adquirir será mayor al máximo posible, que en este caso es una página completa, que quiere decir una columna entera (512 posiciones). Además, la memoria, luego de escribir

una ráfaga, necesita más de un periodo de clock para volver al estado de escritura, lo que causaría perder muestras en el medio.

2) Por otro lado, la memoria al ser dinámica, necesita refrescarse para no perder sus datos y necesita de 11 ciclos de reloj para lograrlo.

Para solucionar los dos problemas se procedió a realizar un bloque (**wr_and_r_2**) que funcione como buffer para almacenar una cantidad de muestras suficientes de tal forma que le de al tiempo a la memoria de, por un lado refrescarse (manualmente) y por otro lado, que la memoria llegue al estado previo a la escritura. Este bloque que está entre el bloque **adc_sel** que entrega las muestras y los ID y el bloque que funciona de interfaz a la memoria SDRAM (**sdram_controller**) funcionará a velocidad de reloj de 120 MHz, para poder adquirir la muestra e ID por separado. Además, se cambió la entrada de datos a la interfaz de la memoria para tomar 16 bits de datos.

La memoria necesita 6 periodos de clock entre la escritura y el estado IDLE (estado cero), 11 periodos de clock por el refresh y necesita 5 periodos de clock entre IDLE y llegar a escribir el primer dato. Por lo tanto la memoria necesita 6 +5+11+ 512(datos)= 534 periodos de clock para llenar una página completa de la memoria y empezar a escribir la página siguiente. Se decidió usar un buffer de 512 posiciones de 16 bits cada uno como se ve en la figura siguiente.

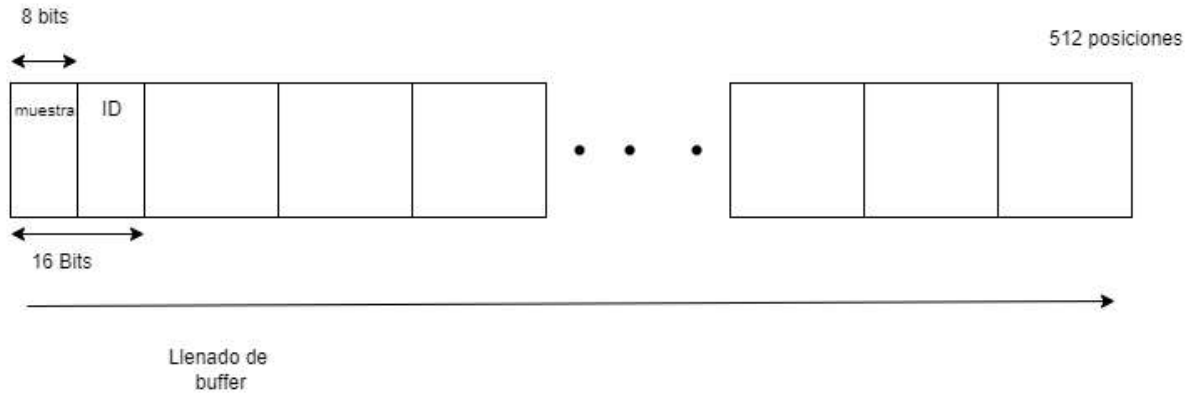


Figura 3.13. Buffer para escribir la memoria SDRAM.

El buffer empezará a llenarse (8 bits por cada periodo de clock) y en el periodo 1016 se señalizará al bloque sdram controller para activar la señalización de escritura. En el periodo 1021 se empezará a escribir la memoria entregando 16 bits por periodo de clock. El bloque actúa con dos punteros, uno que funcione para apuntar a la dirección que se tiene que llenar en el buffer y otro que apunte a la posición que está entregando los datos al bloque sdram controller. Al llegar al periodo 1023, empieza a sobreescribirse desde la posición 1 el buffer. Como el desagotamiento es el doble de rápido que el llenado del buffer,

al llegar de nuevo a la posición 1021, la memoria ya se refresca y está lista para volver a escribir.

Para comprobar la escritura de la memoria en modo full page, se realizó un sistema VHDL donde, utilizando un pushbutton que contiene la FPGA se decidirá escribir y leer recibiendo los datos en el software Docklight en la PC como en la figura 3.14.

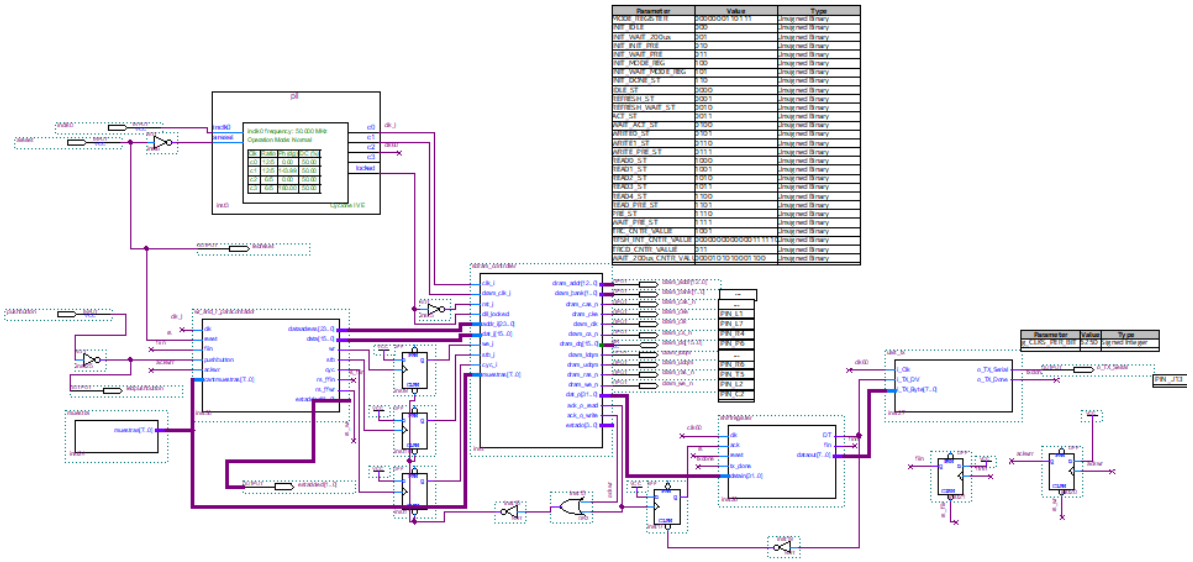


Figura 3.14. Escritura y lectura en modo full page.

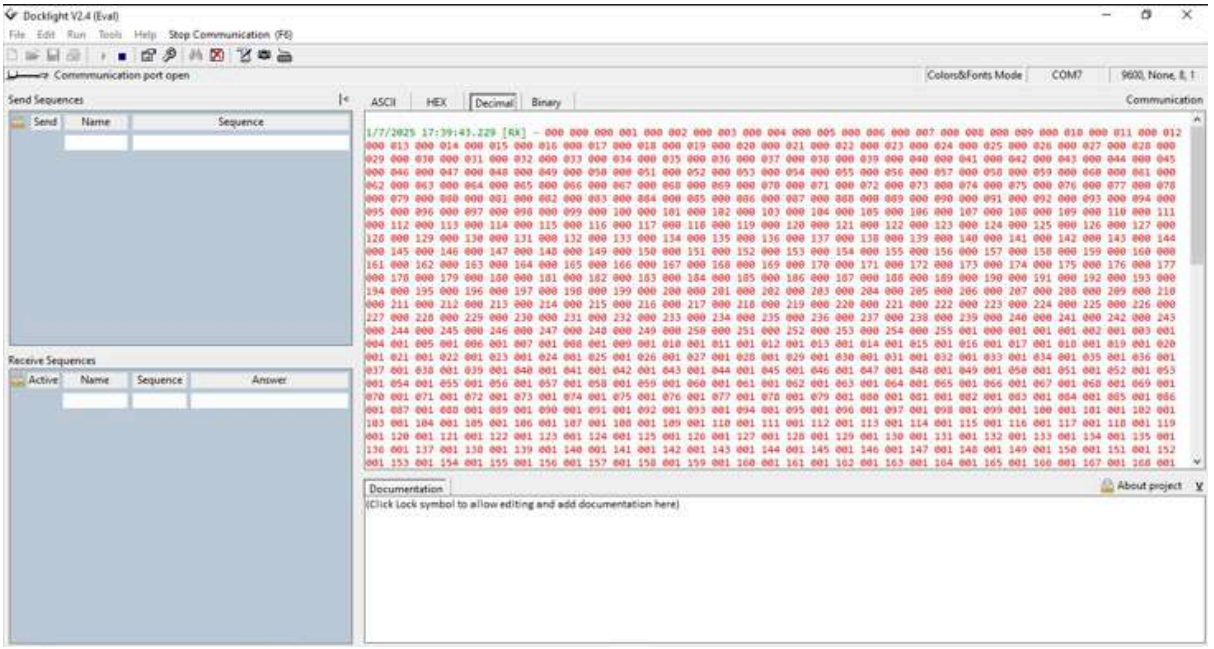


Figura 3.15. Recepción de datos en docklight.

En las imágenes siguientes se pueden observar como se acoplaron los dos sistemas, generando el código VHDL final con la solución propuesta que luego será comprobado en conjunto con la interfaz gráfica en la sección de pruebas.

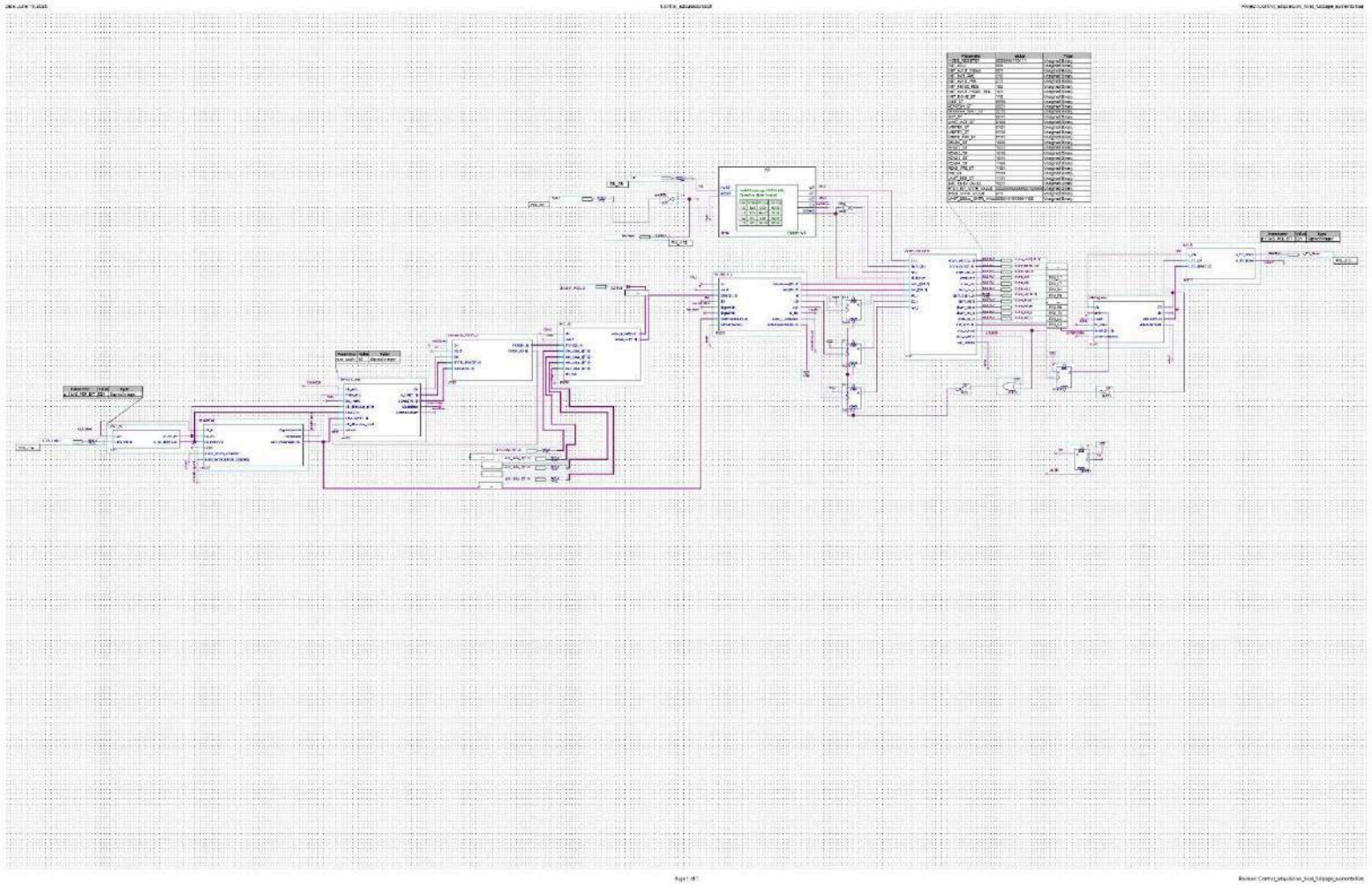


Figura 3.16. Sistema VHDL Final.

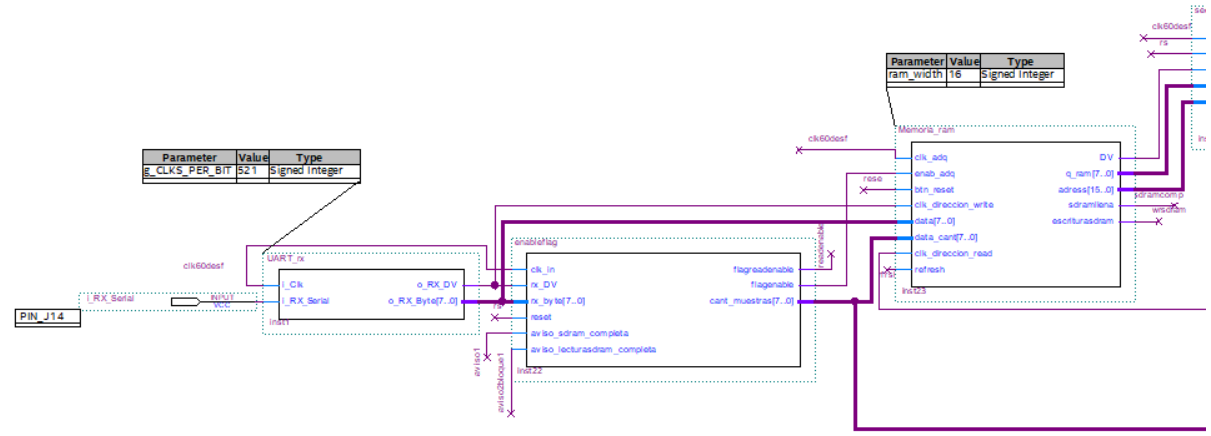


Figura 3.17. Sistema VHDL Final (Parte 1).

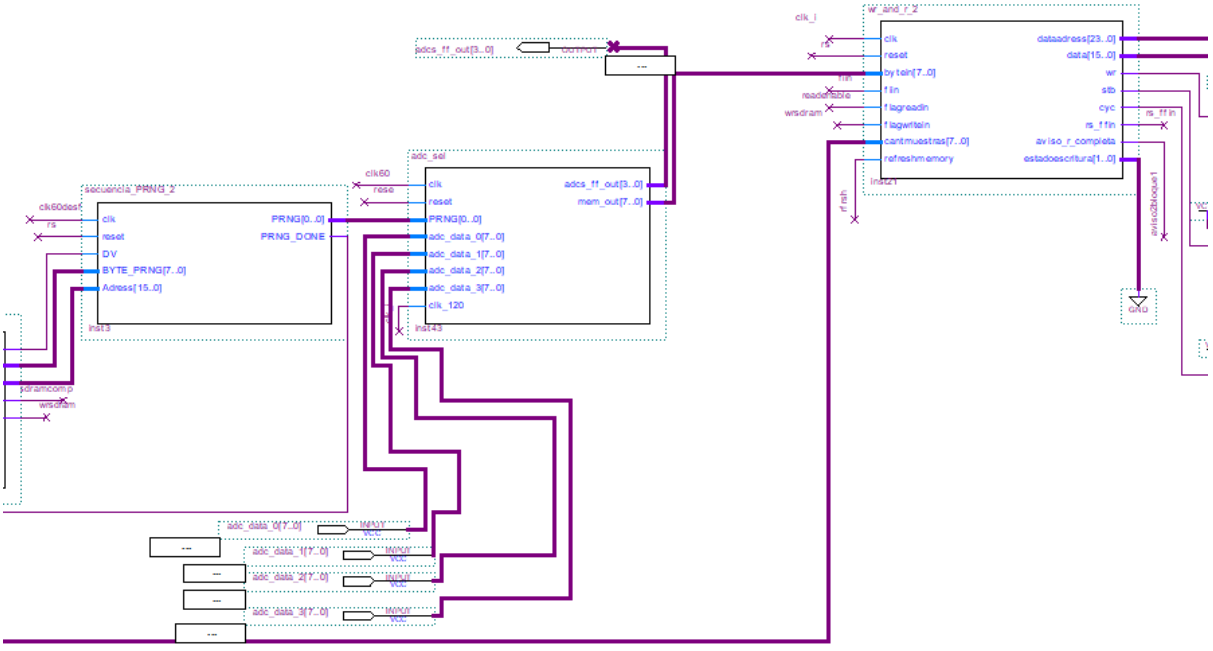


Figura 3.18. Sistema VHDL Final (Parte 2).

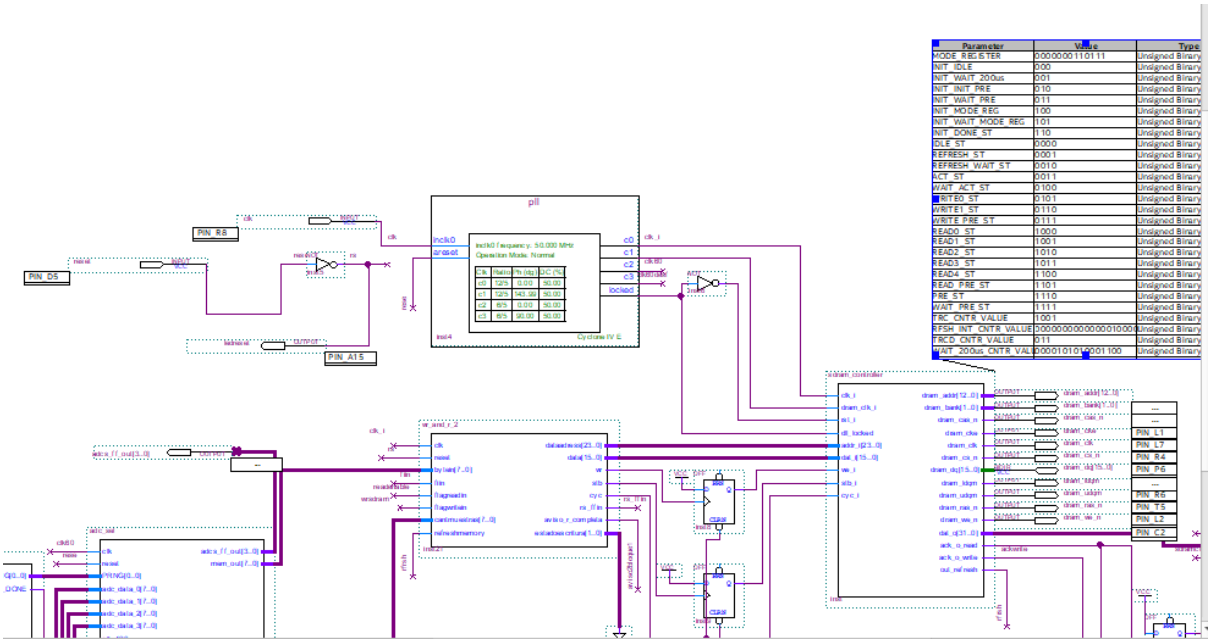


Figura 3.19. Sistema VHDL Final (Parte 3).

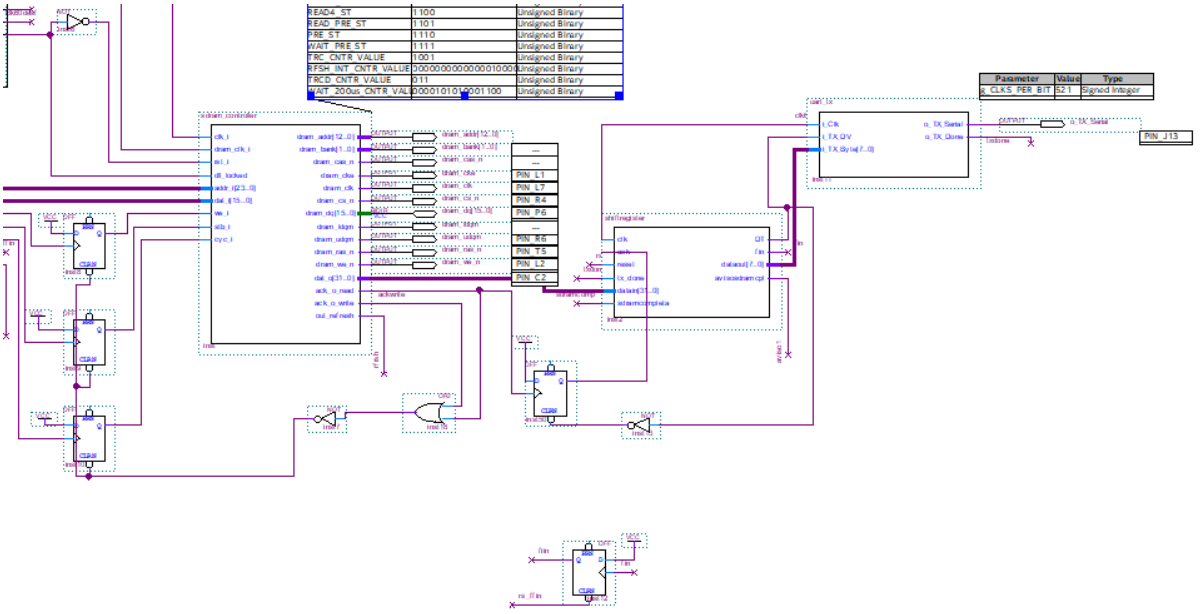


Figura 3.20. Sistema VHDL Final (Parte 4).

En rasgos generales, el sistema contiene los mismos bloques que se vieron previamente. Se realizó el cambio del bloque **wr_and_r** por **wr_and_r_2**. Además, el bloque **enable_flag**, cumple con más funciones. Recibe los comandos desde la PC para:

1) Avisar al bloque de **Memoria_ram (en las pruebas previas fue llamado adq_ram)** que comenzará a llegar la secuencia PRNG.
2) Comunicar al resto de los bloques la cantidad de muestras a tomar.

3) Avisar al bloque **shift_register** que se debe comunicar a la PC que la memoria SDRAM ya está llena.

Finalmente, se agregó una salida al bloque **sdram_controller**, para avisar el estado de refresco al resto de los bloques.

El bloque **Menoria_ram** instancia la memoria RAM de la FPGA y donde se almacenará la secuencia PRNG. El bloque **secuencia_PRNG_2** se encarga principalmente de convertir las palabras de secuencia de 8 bits en bits en serie, para comunicarse con el bloque **adc_sel** que se encarga de seleccionar los ADC y tomar las muestras.

El bloque **shift_register** convierte los 32 bits de datos que llegan del bloque sdram_controller en palabras de 8 bits para poder transmitirlo por el bloque **uart_tx**

En la sección apéndice se encuentra el documento Especificaciones técnicas, que explica cada bloque con más detalle.

3.2 Interfaz Gráfica

Para realizar la interfaz gráfica, se decidió utilizar el lenguaje de programación *Python* y el software *QTdesigner*. *QTdesigner* permite diseñar interfaces de usuario, de forma visual arrastrando y soltando widgets en un entorno de trabajo. De esta manera se puede darle la forma necesaria a la interfaz. Luego, mediante el comando *pyuic5* de *python*, se convirtió el archivo .ui generado por *QTdesigner*, a un archivo .py que puede interpretar *python* y así, darle funcionalidad a cada widget de la interfaz. Para darle funcionalidad es necesario tener la librería PyQT5 para trabajar con el marco de trabajo QT con el que funciona *QTdesigner*.

Por lo tanto se tendría el diseño de la interfaz en un archivo .py y en otro archivo .py se le dará funcionalidad a cada widget, llamando al archivo .py de diseño.

En la figura 3.21 se puede observar el entorno de trabajo del software *QTdesigner*, con algunos widgets colocados en la ventana.

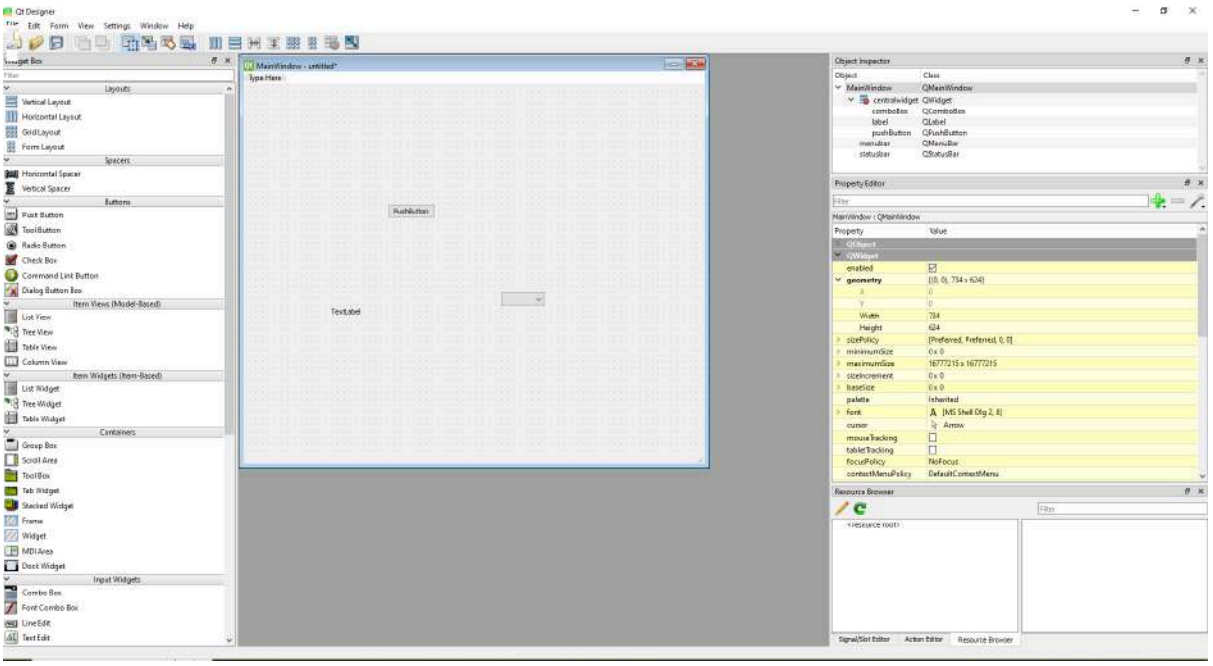


Figura 3.21. Entorno Qtdesigner.

En el sector izquierdo se encuentran todas las herramientas que se pueden utilizar. En el sector derecho se puede configurar el nombre, el tamaño, forma y colores de cada widget. En el centro se encuentra la ventana que representa la interfaz gráfica, y dentro de ella se arrastraran cada uno de los widgets a utilizar.

El diseño de la interfaz gráfica generado en *Qtdesigner* se ve en la imagen siguiente.

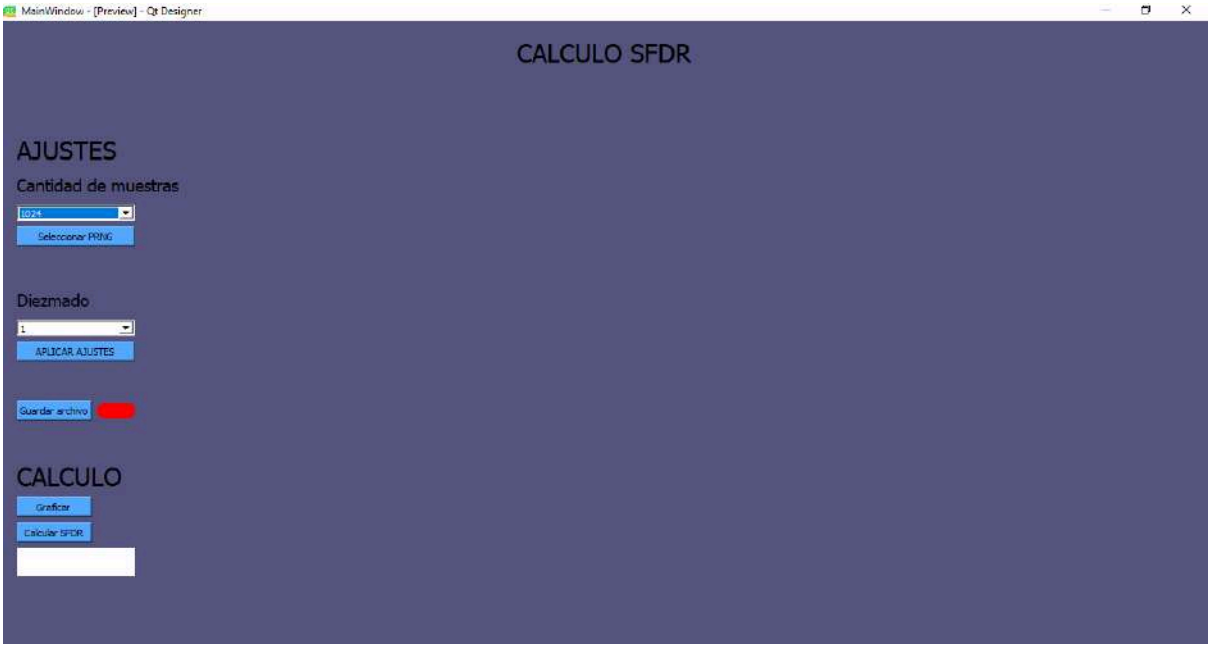


Figura 3.22. Interfaz gráfica desde vista previa QTdesigner.

Para la elección de los widgets se tuvo en cuenta los requerimientos que debe cumplir la interfaz. En la etapa de ajustes se encuentran los widgets necesarios para configurar el banco de pruebas . Un *combobox* para elegir la cantidad de muestras entre 5 posibilidades (1024, 4096, 16384 ,65536, 262144 y 524288). El *pushbutton* Seleccionar PRNG abre un cuadro de diálogo para seleccionar el archivo de texto que contiene la PRNG, desde cualquier ruta de la PC. En el recuadro de Diezmado se puede elegir un factor de diezmado de 1, x2, x4 y x8.

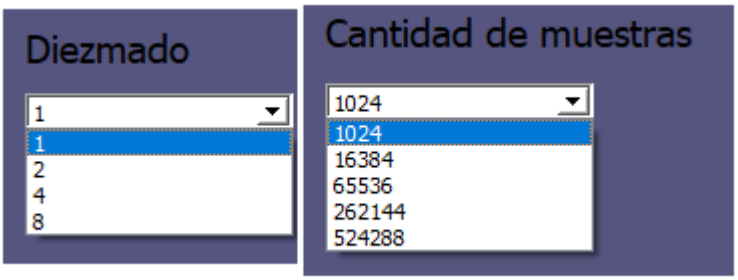


Figura 3.23. Combobox para diezmado y cantidad de muestras.

Luego hay otro *pushbutton* para transmitir las muestras a la PC con el indicador de led para notificar cuando termine el proceso. En el sector **Cálculo**, el primer *pushbutton* grafica la señal en el tiempo y el segundo calcula el SFDR y se grafica la señal en frecuencia con el SFDR. La ventana mostrará el valor de la SFDR. Finalmente, a la derecha se ve un espacio que contiene dos layouts (contenedores) para graficar las señales y el menú de gráficos en ese espacio. En los espacios vacíos existen ventanas de texto para notificar al usuario sobre el proceso. En la imagen siguiente se puede ver los layouts y las ventanas de textos seleccionados.

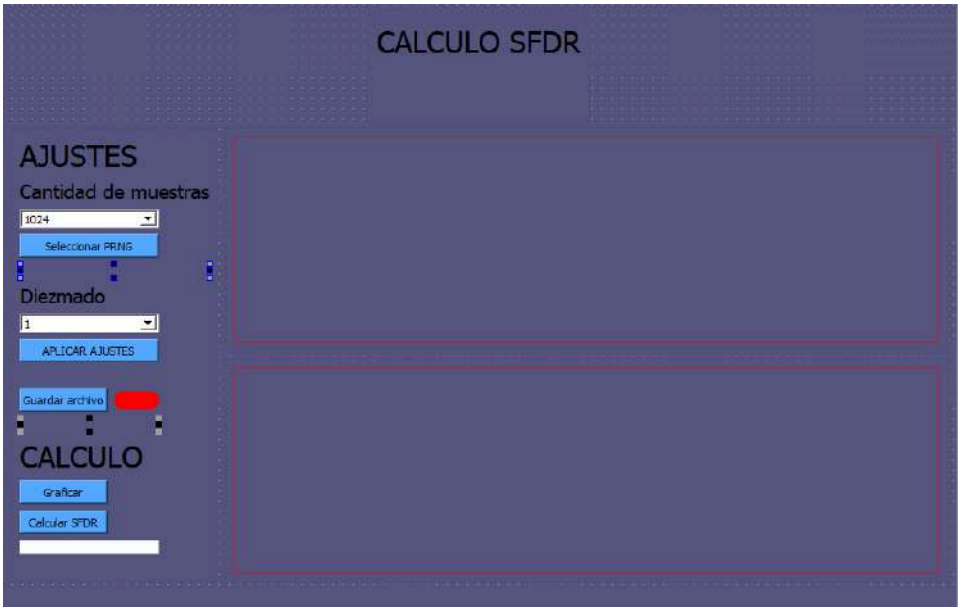


Figura 3.24. Interfaz desde Qtdesigner.

En paralelo con el diseño de la interfaz, se fueron probando por separado las funcionalidades de cada widget realizadas en interfaces diferentes y de los requerimientos que debía cumplir la interfaz, para luego incluirlo en el código principal.

Para comprobar si la placa de adquisición está conectada, se creó la interfaz que se ve a continuación donde solo se ve una ventana de diálogo que comunica si el dispositivo está conectado o no. Para utilizar esta funcionalidad se utilizó el comando *serial.tools.list_ports.comports* de la librería *pyserial* que lista todos la información de todos los puertos. Luego se pregunta si alguno de los puertos coincide con la información del módulo USB y si es así cambia el cuadro de diálogo. Esta funcionalidad se incorporara en una ventana de texto en la parte inferior de la interfaz. Si el dispositivo no está conectado se bloquean el resto de los pushbutton, bloqueandolos desde el código.

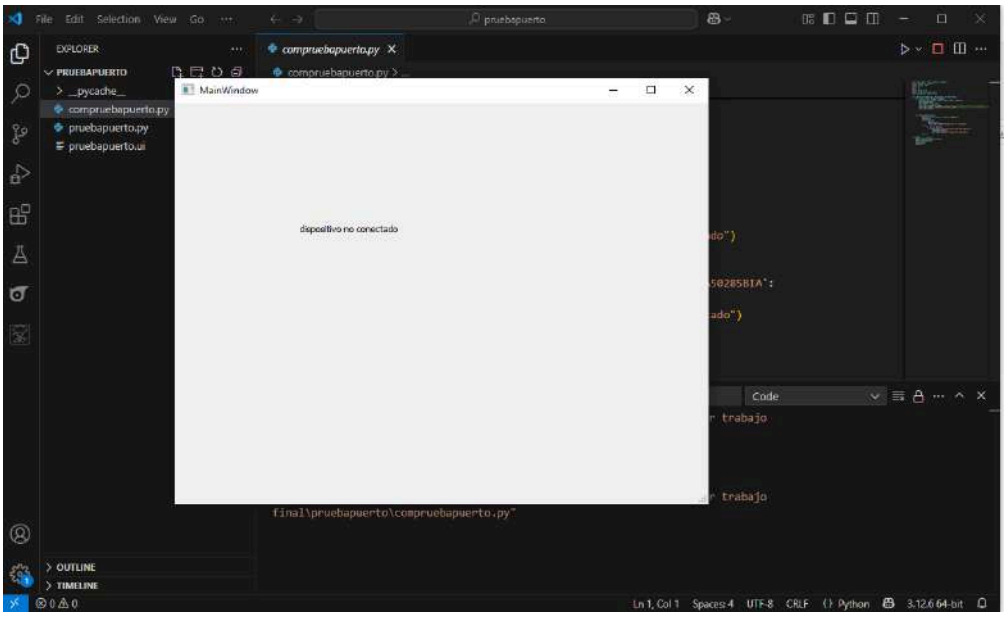


Figura 3.25. Interfaz ejemplo para dispositivo conectado.

Para poder seleccionar la PRNG desde un archivo de texto, desde el *pushbutton* **SELECCIONAR PRNG** se abre un cuadro de diálogo que te permite elegir un archivo de texto como se ve en la figura 3.26.

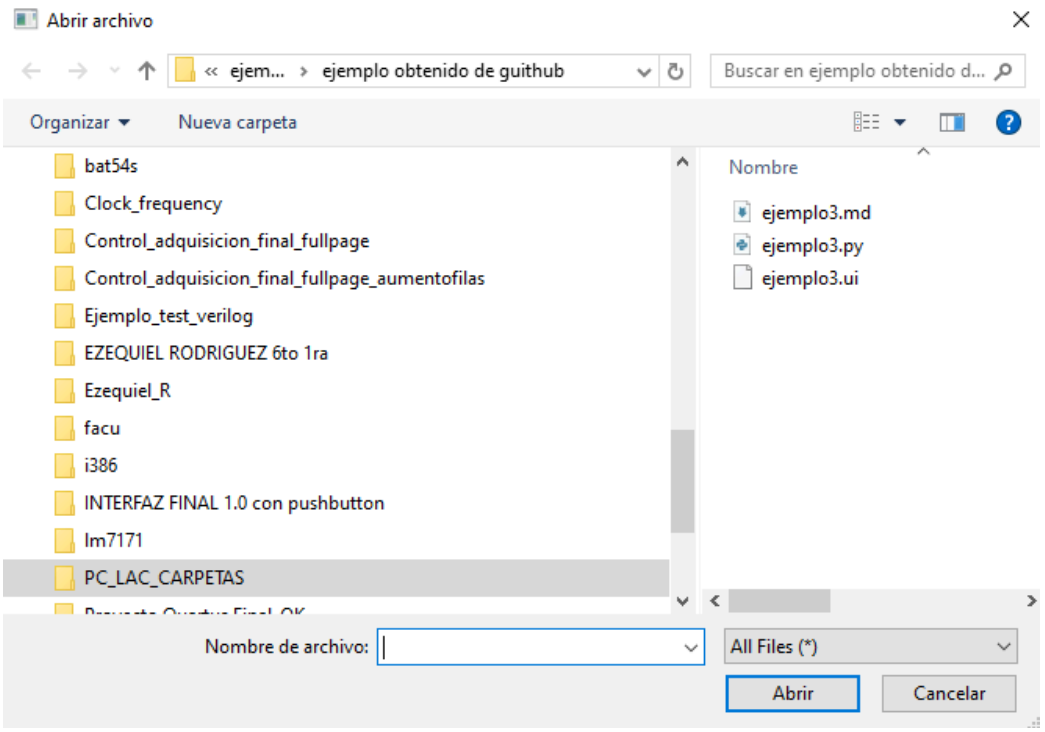


Figura 3.26. Interfaz ejemplo para selección de archivo.

Con esta información, se lee el archivo desde *python* y se guarda la secuencia en una variable. El archivo debe contener la secuencia sin saltos de línea y debe ser de tipo texto. Si la secuencia no es de tipo texto, se notificará en una ventana de diálogo que hubo un error al seleccionar el archivo y seguirá bloqueado el pushbutton siguiente. Una vez almacenada la secuencia en una variable interna, se compara la longitud de la variable con la cantidad de muestras seleccionadas desde el *combobox.* Si la longitud de la secuencia es distinta a la cantidad de muestras elegidas, se comunica el error y no se envían los parámetros. Se debe editar la cantidad de muestras a tomar o el archivo seleccionado.

Una vez bien seleccionadas las muestras y la PRNG a enviar, se pueden enviar los parámetros mediante el *pushbutton* **aplicar ajustes**.

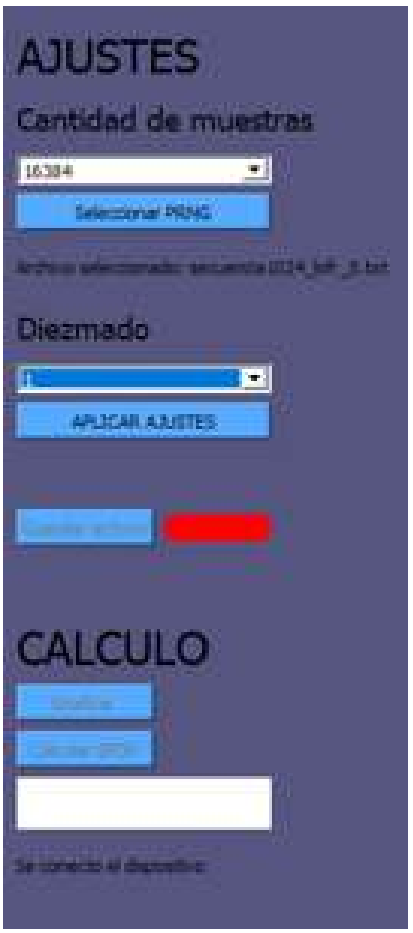


Figura 3.27. Captura de parte de la interfaz.

El resto de los *pushbutton* quedan bloqueados, hasta que se reciba desde la FPGA la notificación de que se terminó de llenar la memoria SDRAM y se habilita el *pushbutton* **Guardar archivo** como se observa en la siguiente figura.

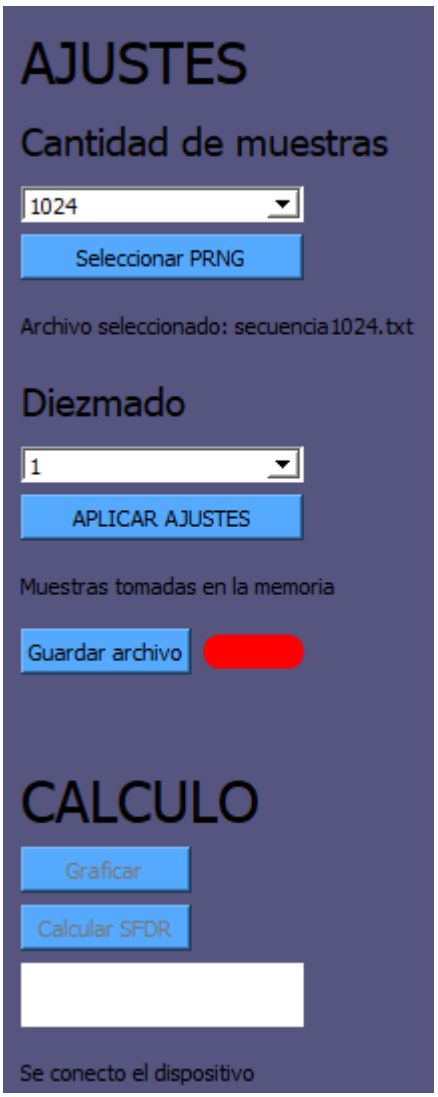


Figura 3.28. Captura de parte de la interfaz.

Para transmitir y recibir datos desde la FPGA se utiliza la librería *serial* y los comandos *write* y *read*. Se ajustan los parámetros de lectura de tal forma de que el puerto espere a que termine de recibirse o transmitirse los datos. Luego de cada transmisión y recepción se cierra el puerto de la PC con el comando close.

Al presionar el *pushbutton* **Guardar archivo** se abre un cuadro de diálogo como ya vimos anteriormente para seleccionar dónde almacenar el archivo con las muestras. Los datos recibidos, muestras e ID, además, se almacenan en una variable interna. Estos datos pasan por un proceso de separación, ordenado y diezmado. Para el ordenamiento se utilizó la misma premisa que en el trabajo final del Ing.Medina. El ID que se obtiene en una posición coincide con la muestra que llegará tres períodos de clock adelante como se observa en la figura 3.29.

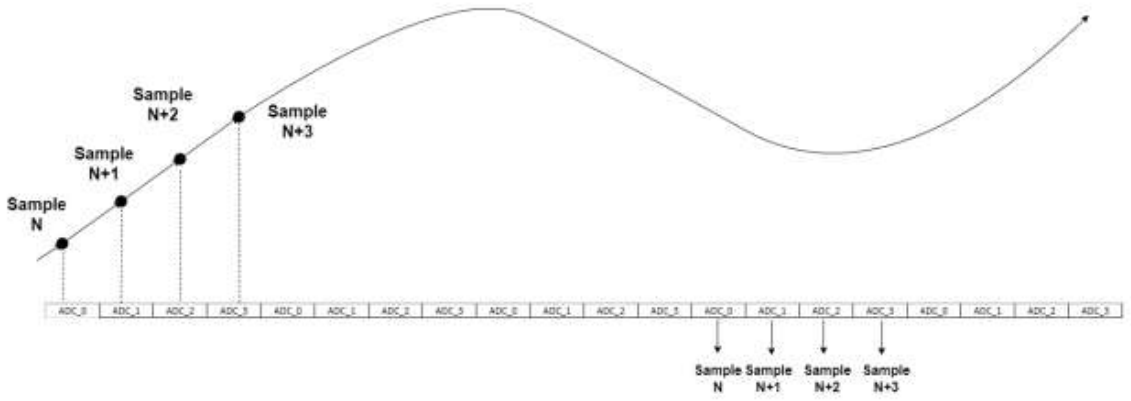


Figura 3.29.Secuencia de muestreo ADC.

Una vez realizado este proceso, se habilitan los *pushbutton* de **graficar** y **calcular SFDR.** Además,se bloquea el pushbutton de **guardar archivos**.

Figura 3.30.Captura de parte de la interfaz.

.

El pushbutton **graficar** genera la señal en el tiempo utilizando las muestras almacenadas. Se utiliza la librería *matplotlib* para el gráfico, además de un toolbar para trabajar sobre el gráfico como se ve en la figura 3.31.

Figura 3.31.Captura de gráfico de la interfaz.

Luego, al presionar **Calcular SFDR** las muestras pasan por un proceso igual al que se utiliza en la función SFDR del software *matlab* para calcularla y graficarla.

Se utiliza parte de una librería llamada *pysnr* (Paul, 2022) libre que se encuentra en un repositorio de github (ver bibliografía), para lograr el cálculo. El proceso consta de remover la continua de las muestras, calcular la fft, obtener el bin donde se encuentra la amplitud máxima de potencia y el segundo valor máximo de potencia y calcular la relación entre ellas. Finalmente, se grafica la potencia espectral y se muestran los puntos de interés con la zona que corresponde a la SFDR.

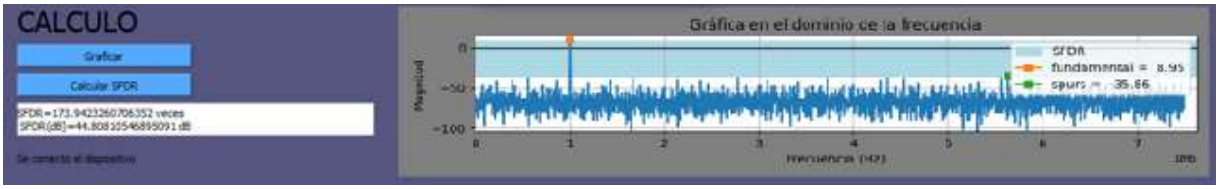


Figura 3.32.Captura de cálculo y gráfico de SFDR.

Utilizando el comando *Py_installer* desde la ventana de comandos, se comprimió toda la carpeta utilizada con los archivos .py dentro en un mismo archivo.exe portable para utilizar la interfaz en cualquier PC. Para conocer más en detalle el código realizado para lograr la interfaz se puede observar en la sección *apéndice* un enlace que llevará a github donde se encuentra todo el código. El repositorio cuenta con 7 archivos. Los archivos .py Separación y Ordenar contienen el proceso realizado para la identificación y ordenamiento de las muestras. El archivo Diezmar contiene el proceso de diezmado, mientras que el archivo Graficar contiene el desarrollo para los gráficos. Estos archivos son usados como módulos para llamar a sus funciones desde el código principal contenido en el archivo principal **INTERFAZ_SFDR_MAIN_2_1.** Los dos archivos restantes corresponden al diseño de la interfaz, generados con el software Qtdesigner.

3.3 Mejoras de hardware

Al realizar una revisión y análisis sobre el diseño de la placa de adquisición e informe del proyecto final "Desarrollo de placa de adquisición basada en "Time Interleaving"" del Ing. M.Medina, se concluyó que se pueden realizar tres mejoras favorables en un nuevo diseño para la placa de adquisición que serán descriptas a continuación.

Integridad de señal de clock a cada ADC

En primer lugar, dentro del informe se explica, que la integridad de la señal de clock proveniente de la FPGA hacia cada ADC se ve degradada como se ve en la figura siguiente.

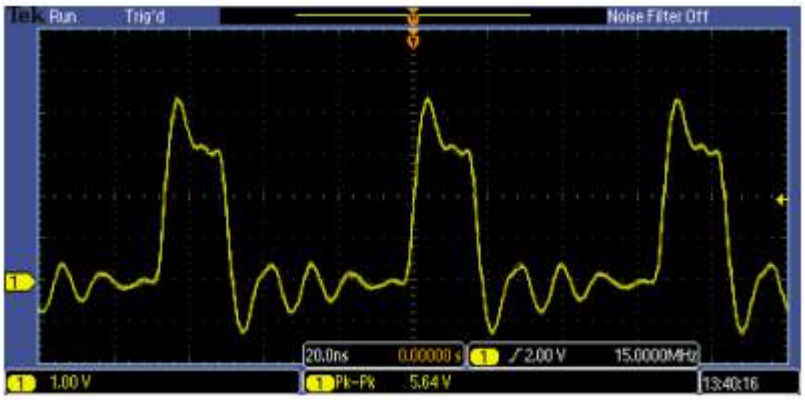


Figura 3.33. Señal de clock a la entrada de cada ADC.

En la imágen se observa la señal de clock que ingresa a uno de los 4 ADC de la placa de adquisición, para un ensayo de muestreo a 60 MHz de forma no aleatoria (cada ADC recibe una señal de clock a 15 MHz). Cada ADC no obstante, tiene la capacidad de funcionar a una frecuencia de hasta 50 MHz. Sin embargo al intentar aumentar la frecuencia de muestreo del sistema se comienzan a detectar errores producto de la pérdida de integridad en las señales de reloj.

Entonces, se planteó la posibilidad de colocar una etapa entre la salida de la FPGA y la entrada de los ADC que permitan reducir el ringing y overshoot en la señal de clock.

Una resistencia serie podría permitir mejorar el efecto de ringing y poder aumentar la frecuencia de muestreo general, lo que permitiria agrandar el rango de las señales de entrada al sistema.

Ahora, para calcular el valor de esa resistencia, se procedió a medir la señal que se obtiene en la entrada de cada ADC, modelar mediante un circuito RLC en el software de simulación *LTspice*, y a partir de allí calcular el valor de resistencia necesaria para mitigar el efecto, y tratar de comprobarlo de forma experimental.

Se diseñó el siguiente circuito PCB para tomar una medición correcta ya que si se toma la medición directamente desde la placa de adquisición, se está cargando el circuito con la capacidad de punta del osciloscopio, lo que provoca que la medición sea incorrecta. Entonces se diseñó un PCB que tenga en cuenta la capacidad de entrada del ADC, y con un divisor capacitivo desacoplar la carga que afecta la medición desde el osciloscopio. La medición se verá atenuada, pero su forma se mantendrá.

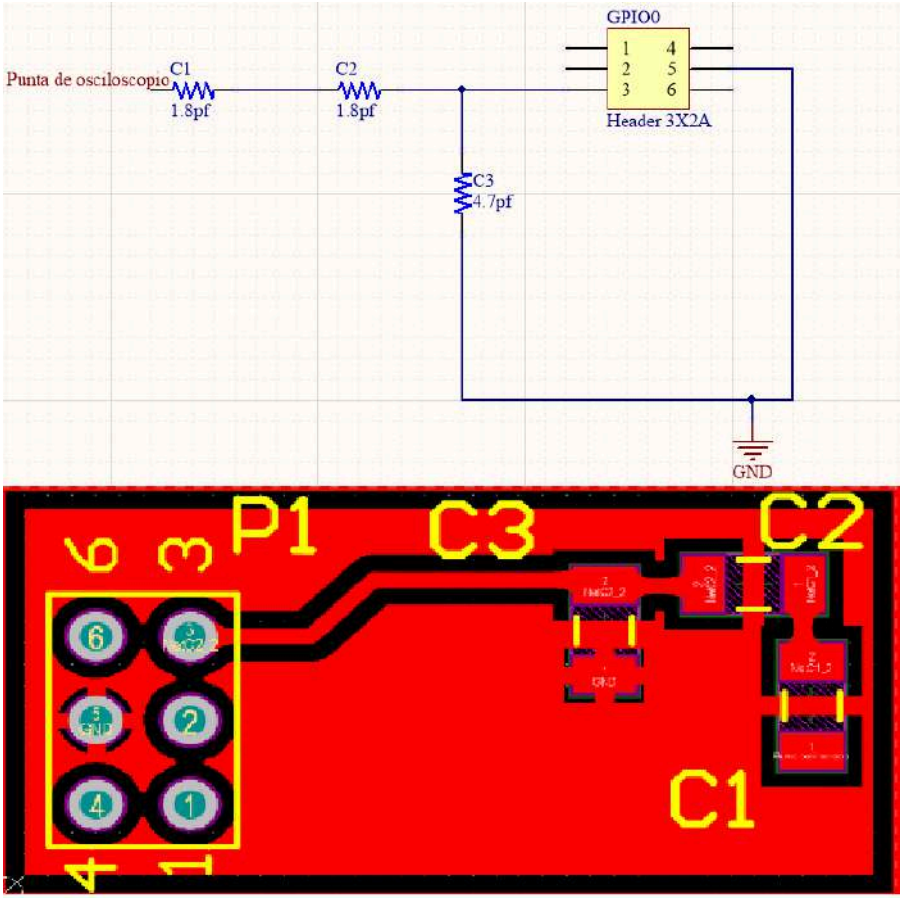


Figura 3.34.PCB para medición de Clock.

Con la FPGA se generarían señales de clock de 20 MHz y 30 MHz de 3.3V de amplitud y 30% de ciclo de trabajo, que llegan al capacitor C3 que representa la capacidad de entrada del ADC (entre 4 y 5 pf). Los capacitores C1 y C2 son dos capacitores en serie (de 1.8pf cada uno) a C3 para que la capacidad de entrada del osciloscopio no afecte la medición. De esta forma se obtendrá una señal disminuida en amplitud (debido al divisor

capacitivo entre la capacidad de 1pf en serie y los 10 pf de la punta de entrada), pero con la forma correcta. P1 es un zócalo tipo hembra para conectar los GPIO de la FPGA.

De esta manera se podría ver las 2 señales para ver el grado de degradación al aumentar la frecuencia. A continuación, se observan las mediciones realizadas.

Figura 3.35. Medición señal de clock (20MHz).

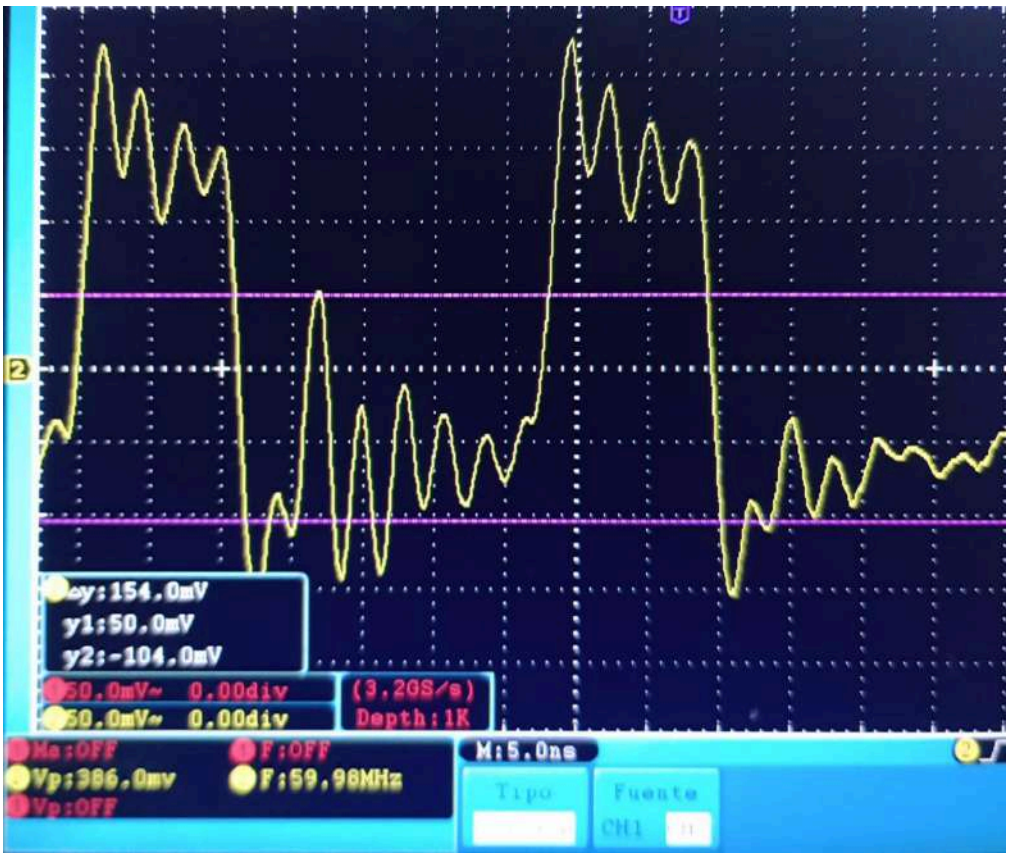


Figura 3.36. Medición señal de clock(30 MHz).

Como se ve en la imagen 3.36, la oscilación en el semiciclo negativo para una frecuencia de clock de 30 MHz tiene una amplitud de 155 mV, que dividida por un factor de 1/11 que es la correspondiente al divisor capacitivo nos lleva a una amplitud de 1.7V. Como se ve en la hoja de datos del ADC en la figura 3.37, a partir de VIH y VIL, el ADC interpreta esa diferencia de tensión como un cambio de estado y toma la decisión de muestrear.

CLK, PD DIGITAL INPUT CHARACTERISTICS					
V_{IH}	Logical High Input Voltage			**2.0**	V (min)
V_{IL}	Logical Low Input Voltage			**0.8**	V (max)
I_{IH}	Logical High Input Current	$V_{IH} = DV_{DD} = AV_{DD} = +5.25V$		**±5**	µA (max)
I_{IL}	Logical Low Input Current	$V_{IL} = 0V, DV_{DD} = AV_{DD} = +5.25V$		**±5**	µA (max)
C_{IN}	Digital Input Capacitance		4		pF

Figura 3.37. hoja de datos ADC.

A partir de la señal medida a 30 MHz, se realizó un circuito RLC equivalente como se ve en la imagen 3.38, en el software LTspice para emular la señal y calcular una resistencia que ayude a mitigar el efecto. Para calcular el circuito RLC se tuvo en cuenta, que la resistencia de salida del GPIO varía entre 15 y 50 ohms, la capacidad en paralelo pertenece a la del ADC y las ecuaciones que relacionan una respuesta amortiguada con el circuito de segundo orden.

$w0 = \frac{1}{\sqrt{LC}}$; frecuencia de natural

$\xi = \frac{R}{2}.\sqrt{\frac{C}{L}}$;factor de amortiguamiento

$overshoot = e^{-\pi\xi/(1-\xi^2)}.100\%$

La frecuencia natural se reemplazó por 376 MHz teniendo en cuenta la cantidad de oscilaciones en el semiciclo de la señal positiva de 30 MHz. A partir de ahí se obtuvo L=40 nHy. Con un overshoot del 70 por ciento se obtuvo ξ, y así R. Se probaron R cercanas hasta dar con el valor óptimo de 15 ohm para obtener una señal muy similar a la medida como se observa en la simulación de la figura 3.40.

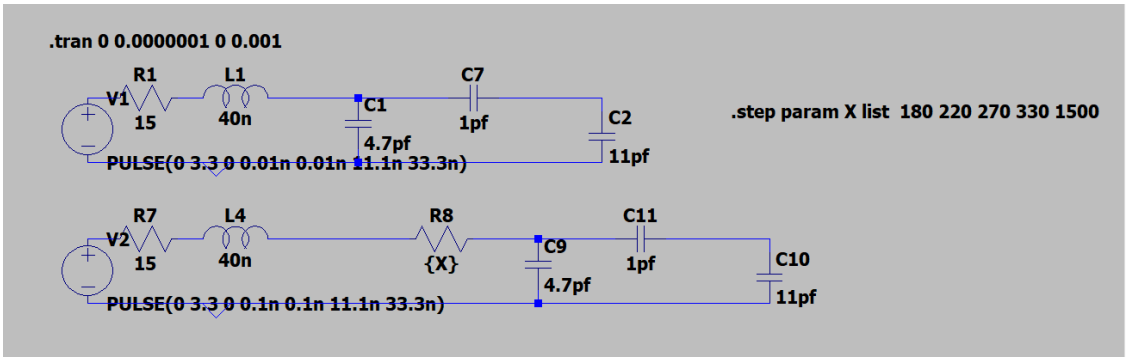


Figura 3.38. Circuito RLC equivalente + etapa de medición.

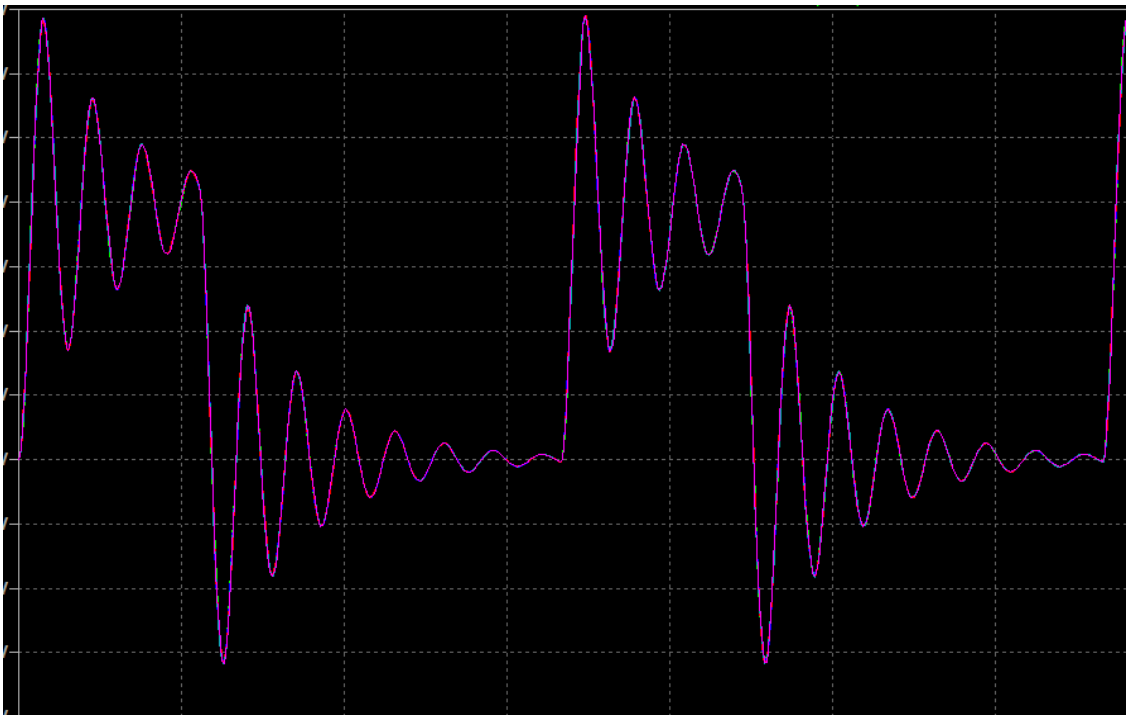


Figura 3.39. Señal de clock equivalente sin resistencia en serie.

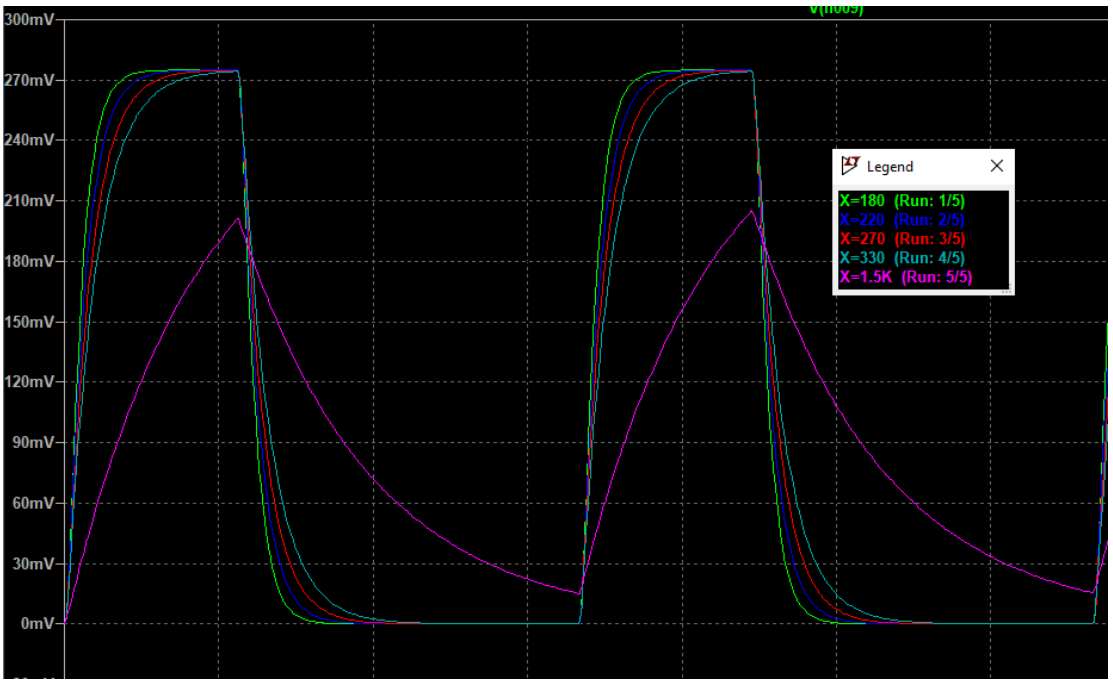


Figura 3.40. Señal resultante con resistencia serie agregada.

Como se observa, en la respuesta con la resistencia en serie, la respuesta mejora notoriamente, siendo la mejor opción la de la resistencia de 180 ohms.

Para comprobar esta medición se realizó el siguiente PCB, teniendo en cuenta la resistencia en serie al circuito.

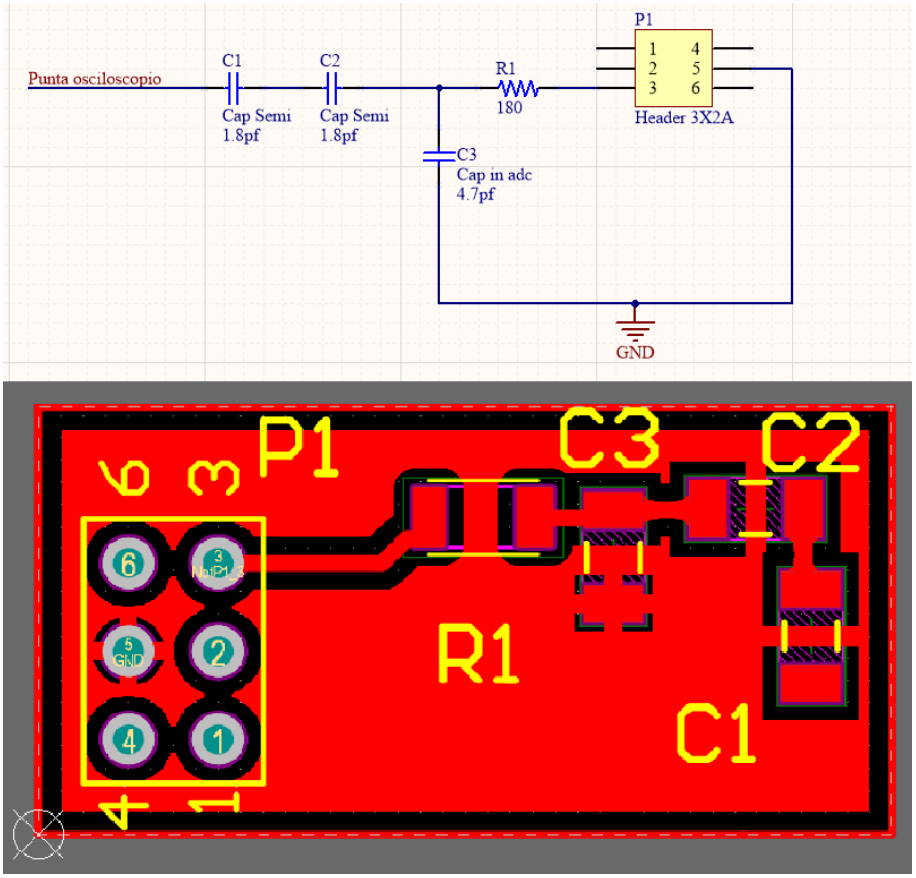


Figura 3.41. Diseño experimental PCB con resistencia serie.

Al medir la señal de clock con la resistencia en serie se logró mitigar el ringing como se ve en la figura 3.42. Además, el pico que se ve en el ciclo negativo es de alrededor de 70 mV, que al tratarlo con el factor del divisor capacitivo nos da una amplitud de 0.770V, que es menor a la amplitud que puede tomar el ADC como indicación de muestreo. De esta forma el ADC puede muestrear hasta por lo menos 30 MHz, y así, la placa de adquisición en general puede aumentar su frecuencia de muestreo hasta 90 MHz.

Figura 3.42. Medición experimental con resistencia serie.

Etapa de adecuación y filtrado de la señal de entrada

En segundo lugar, se buscó mejorar la etapa de adecuación y filtrado de la señal de entrada.

En el diseño original de la placa de adquisición, se quiso replicar la punta de osciloscopio en la primera etapa de entrada donde se atenúa la señal 10 veces. Para ello se pensaba utilizar un divisor resistivo formado por las resistencias de 820 Kohm, 82 Kohm Y 100 Khm (figura 3.43). Pero debido al drift de bias de corriente con la variación de la temperatura que se tiene utilizando el amplificador operacional LM 7171 (figura 3.44), esa elección de resistencias causaba que el drift de offset de tensión de continua en la salida de la etapa de entrada fuera mayor a 4mV. Este valor representa el máximo error permitido para que el ADC no tenga un error de escalón al convertir la señal analógica a digital, ya que su LSB es de 8 mV. Por lo tanto, el Ing Medina debió achicar el valor de resistencias del divisor, utilizando ahora los valores 22 Kohm, 10 Kohm y 3.9 Kohm.

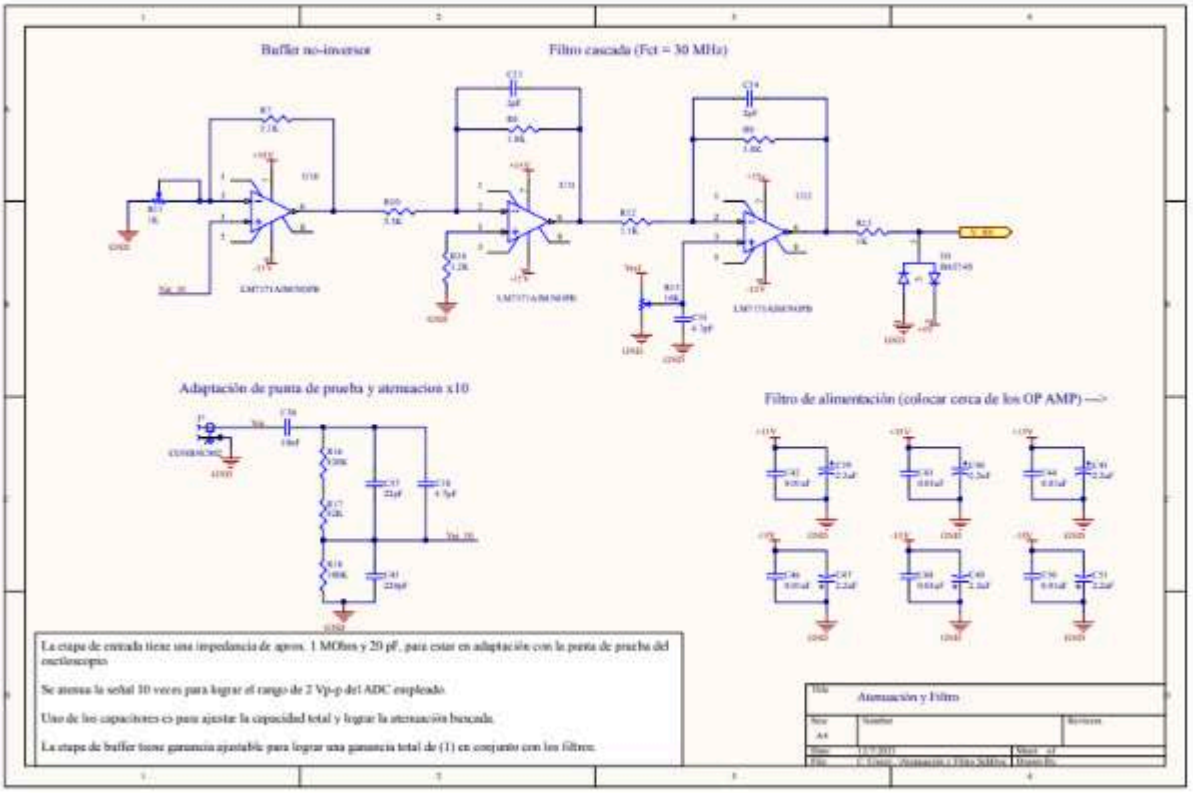


Figura 3.43. esquema original de la etapa de adecuación de la señal .

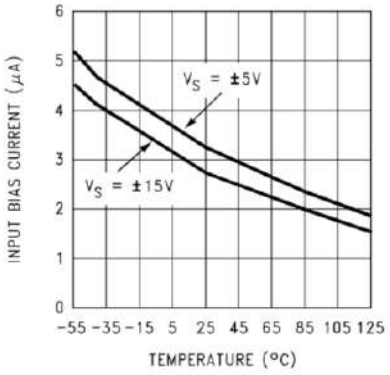


Figura 3.44. IB vs Temperature LM 7171.

Con el fin de mejorar la impedancia del divisor resistivo de entrada, se busco la utilización de otro amplificador operacional que cumpla con las características de buffer y además tenga mejor performance de variación de offset con respecto a la temperatura. Por otro lado, se buscó mejorar la etapa de filtrado utilizando un solo amplificador operacional con una topología multi-feedback de orden 2. El filtro deberá tener una frecuencia de corte de 36 MHz. Esta frecuencia se da ya que, al mejorar la integridad de la señal de Clock de los ADC, se puede aumentar la frecuencia de muestreo del sistema y así, mejorar el rango de la señal de entrada. Nuestro limitante, ahora, es la frecuencia de procesamiento de la

SDRAM que es de 143 MHz. Como la frecuencia de clock de memoria es del doble de la del sistema, nuestro sistema ahora puede adquirir a 71.5 MHz. Cada ADC, con una secuencia aleatoria muestreara a fs/3= 23.5MHz.

El amplificador operacional elegido en primer lugar para la etapa de adecuación fue el LMH 6732 que tiene las siguientes características. Este operacional, es realimentado por corriente y tiene la particularidad de cambiar su respuesta en frecuencia de acuerdo a una resistencia que se coloca en el pin 8 que modifica la corriente interna del operacional.

FEATURES

- **Exceptional Performance at Any Supply Current:**
  V_S = ±5V, T_A = 25°C, A_V = +2V/V, V_{OUT} = 2V_{PP}, Typical unless Noted:

I_{CC} (mA)	-3dB BW (MHz)	DG/DP (%/deg.) PAL	Slew Rate (V/µs)	THD 1MHz (dBc)	Output Current (mA)
1.0	55	0.20 / 0.036	400	-70.0	9
3.4	180	0.022 / 0.017	2100	-78.5	45
9.0	540	0.025 / 0.010	2700	-79.6	115

- **Ultra High Speed (−3dB BW) 1.5GHz (I_{CC} = 10mA, 0.25V_{PP})**
- **Single Resistor Adjustability of Supply Current**
- **Fast Enable/ Disable Capability 20ns (I_{CC} = 9mA)**
- **"Popless" Output on "Enable" 15mV (I_{CC} = 1mA)**
- **Ultra Low Disable Current <1µA**
- **Unity Gain Stable**
- **Improved Replacement for CLC505 & CLC449**

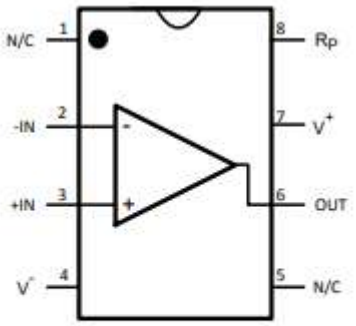
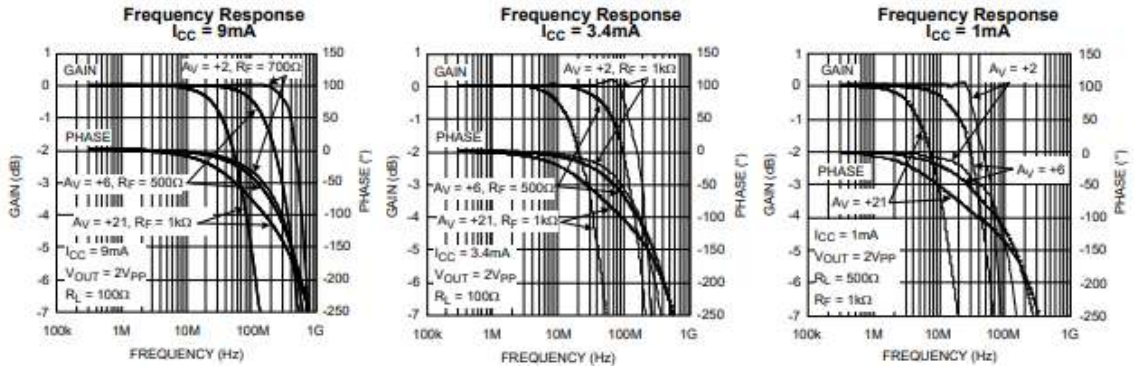


Figure 3. 8-Pin SOIC (Top View)
See Package Number D (R-PDSO-G8)



I_{BN}	Input Bias Current	Non Inverting[3]		−2	±11 **±12**	µA
DI_{BN}	Input Bias Current Average Drift	Non-Inverting[4]		5		nA/°C
I_{BI}	Input Bias Current	Inverting[3]		−9	±20 **± 30**	µA
DI_{BI}	Input Bias Current Average Drift	Inverting[4]		−14		nA/°C

Figura 3.45. LMH 6732.

Pero al probar la etapa de adecuación por separado con este operacional con la configuración que se ve en la figura 3.43, nunca se pudo lograr la estabilidad de la señal a la salida ya que se introducía mucho ruido y oscilaciones. Debido a lo mencionado y a que el operacional se quemó, se optó por otro amplificador operacional, el LMH 6611/2.

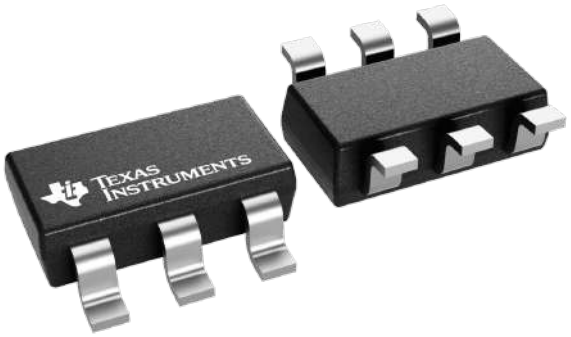
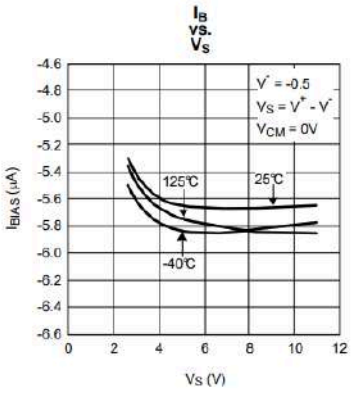


Figura 3.46. LMH 6611.

FEATURES

- V_S = 5V, R_L = 1 kΩ, T_A = 25°C and A_V = +1, Unless Otherwise Specified.
- Operating Voltage Range 2.7V to 11V
- Supply Current Per Channel 3.2 mA
- Small Signal Bandwidth 345 MHz
- Open Loop Gain 103 dB
- Input Offset Voltage (Limit at 25°C) ±1.5 mV
- Slew Rate 460 V/µs
- 0.1 dB Bandwidth 45 MHz
- Settling Time to 0.1% 67 ns
- Settling Time to 0.01% 100 ns
- SFDR (f = 100 kHz, A_V = 2, V_{OUT} = 2 V_{PP}) 102 dBc
- Low Voltage Noise 10 nV/√Hz
- Output current ±100 mA
- CMVR −0.2V to 3.8V
- Rail-to-Rail Output
- −40°C to +125°C Temperature Range



Input DC Performance						
V_{OS}	Input Offset Voltage (LMH6611)	V_{CM} = −4.5V		0.074	±1.5 ±2	mV
	Input Offset Voltage (LMH6612)	V_{CM} = −4.5V		0.095	±1.5 ±2	
TCV_{OS}	Input Offset Voltage Average Drift	See(3)		4		µV/°C

Figura 3.47. Características principales y respuesta de tensión y corriente de offset del LMH 6611.

El diseño realizado para la nueva etapa de entrada es la siguiente y para ver más detalle en los cálculos realizados se puede acercar al documento de Especificaciones Técnicas que se encuentra en el apéndice.

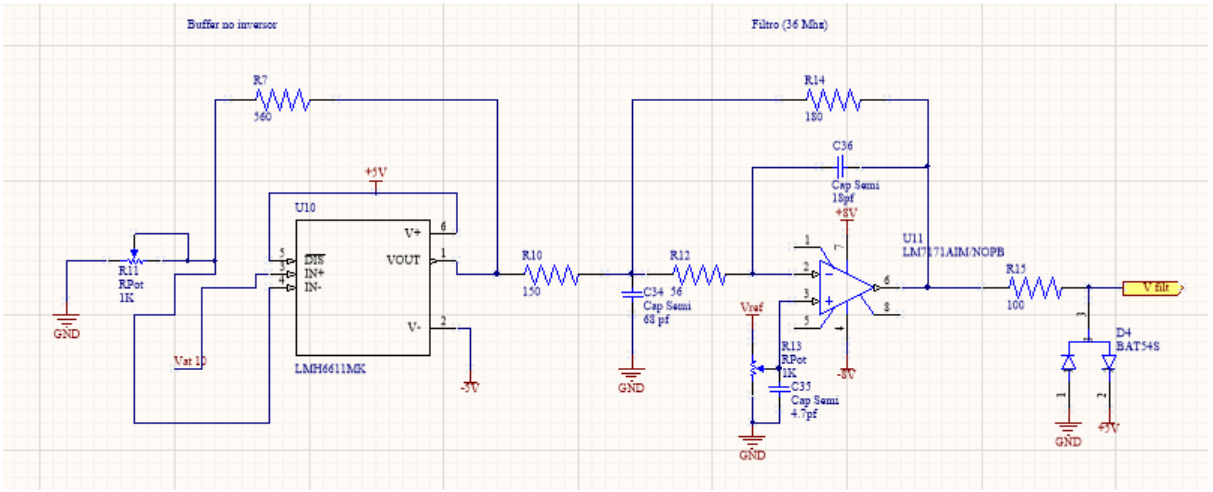


Figura 3.48. Etapa de adecuación y filtrado nueva.

A la salida se ve la resistencia en serie y el diodo de protección que se encuentran antes de los ADC. La resistencia en serie cumple la función de limitar la corriente para proteger el dispositivo. El amplificador operacional utilizado experimentalmente fue el LMH 6612 en la etapa de adecuación, aunque para el diseño, para disminuir el espacio, se utilizará el LMH 6611. La única diferencia entre ellos es que el LMH 6612 cuenta con dos operacionales dentro.

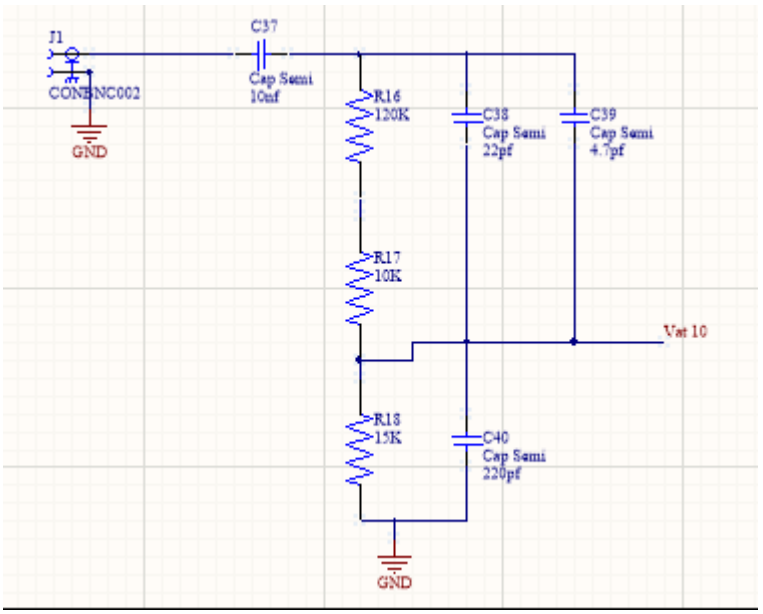


Figura 3.49. Etapa de atenuación con la impedancia de entrada mejorada.

Para realizar las mediciones y comprobar el funcionamiento se optó primero por realizar una placa experimental como la que se ve en la siguiente imagen.

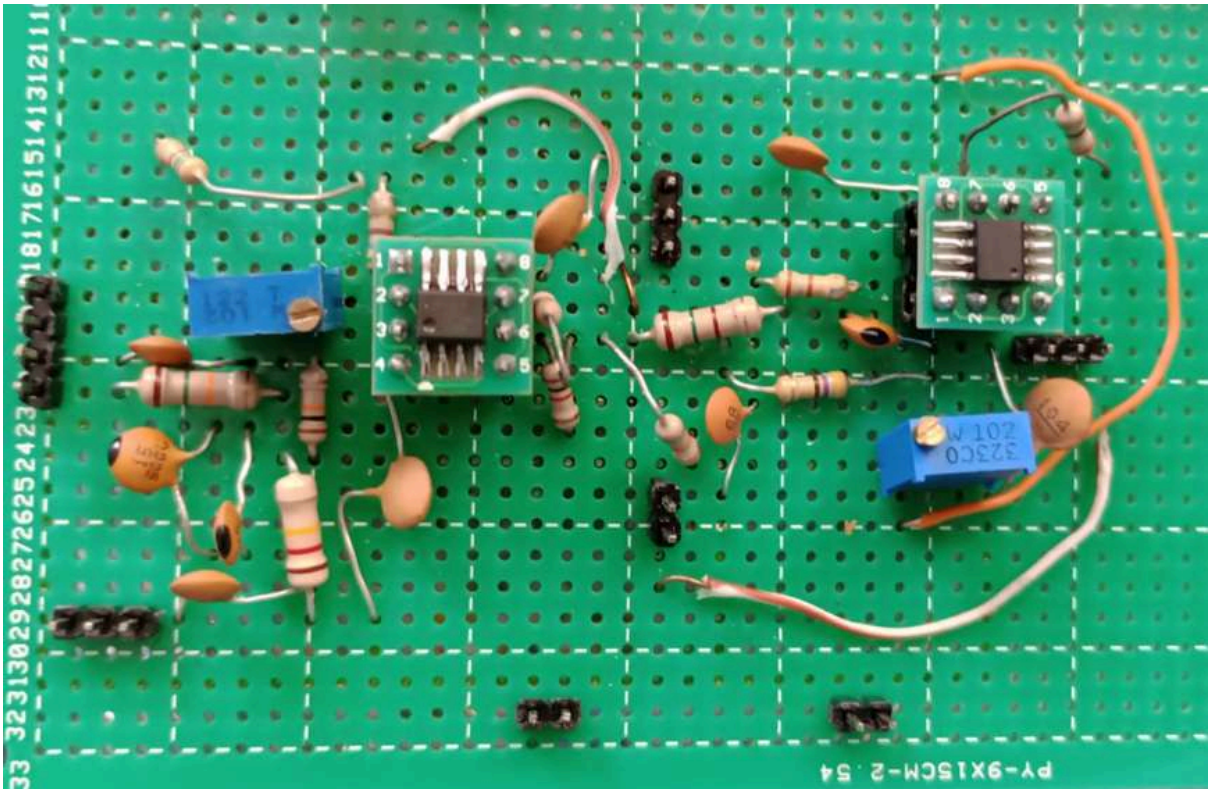


Figura 3.50. Placa experimental para la etapa de adecuación/filtrado.

El circuito tenía una buena respuesta al realizar las mediciones con el generador de señales y el osciloscopio. Pero el generador de señales tiene la limitación de llegar hasta 25 MHz en frecuencia y nosotros debíamos comprobar la medición hasta la frecuencia de corte en 36 MHz. Por lo tanto, se decidió utilizar el analizador de espectro para ver la respuesta en frecuencia. Al realizar la medición, midiendo en primera instancia la etapa de atenuación, antes de llegar al primer operacional nos encontramos con la respuesta que se ve en la imagen 3.52.

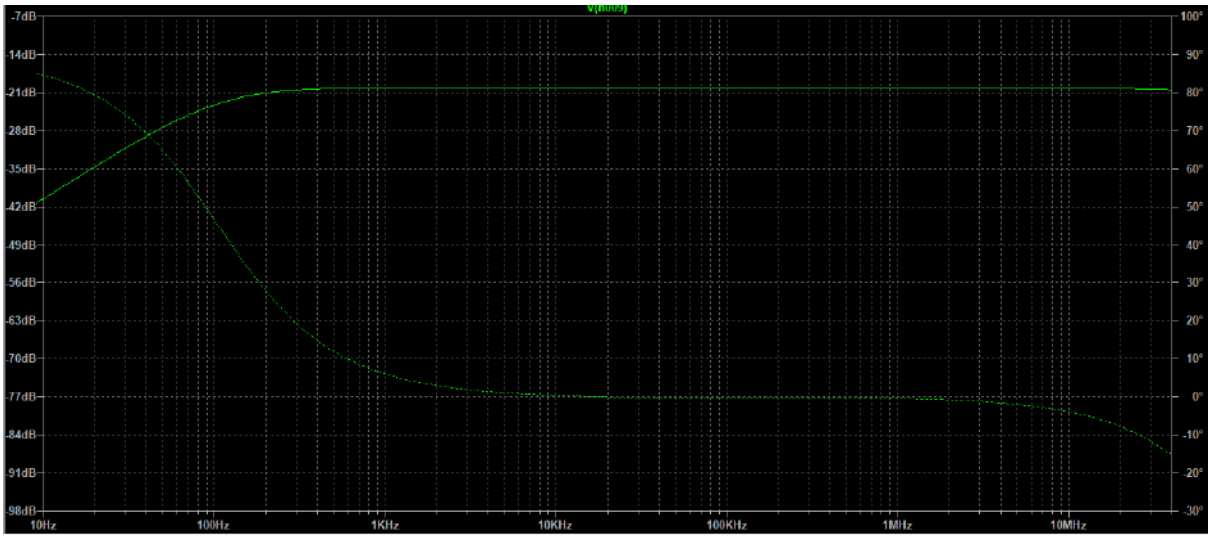


Figura 3.51. Respuesta en frecuencia de etapa de atenuación (LTspice) .

Figura 3.52. Respuesta en frecuencia de etapa de atenuación (experimental)

.

Al comparar las respuestas en simulación y de forma experimental, se observa que la respuesta no es plana como se esperaba. Una posible causa de esta diferencia es la variación del valor nominal de las resistencias en la etapa de atenuación a medida que aumenta la frecuencia. En la Figura 3.53 se muestra un gráfico de impedancia en función de la frecuencia para una resistencia through-hole. Como puede observarse, para resistencias del orden de los 10 kΩ, el valor nominal disminuye notablemente a partir de los 10 MHz. La imagen fue extraída de la hoja de datos proveniente de la empresa Vishay Intertechnology. (s.f.) y en la bibliografía se encuentra el enlace.

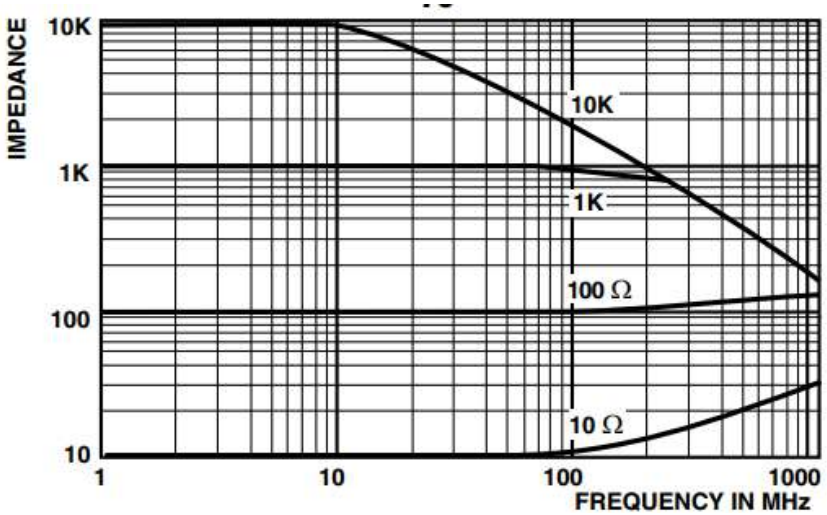


Figura 3.53. Impedancia vs frecuencia para resistencia through hole

Por otro lado, el uso de caminos de soldadura para la conexión de los nodos del circuito (en lugar de pistas optimizadas) así como la ausencia de un plano de masa, también puede afectar negativamente la respuesta del circuito.

Por lo tanto, para tener una medición precisa, se optó por realizar el diseño en PCB con componentes SMD 1206 para tener una respuesta más fiel.

El diseño y la placa desarrollada con la CNC son las imágenes que se ven a continuación.

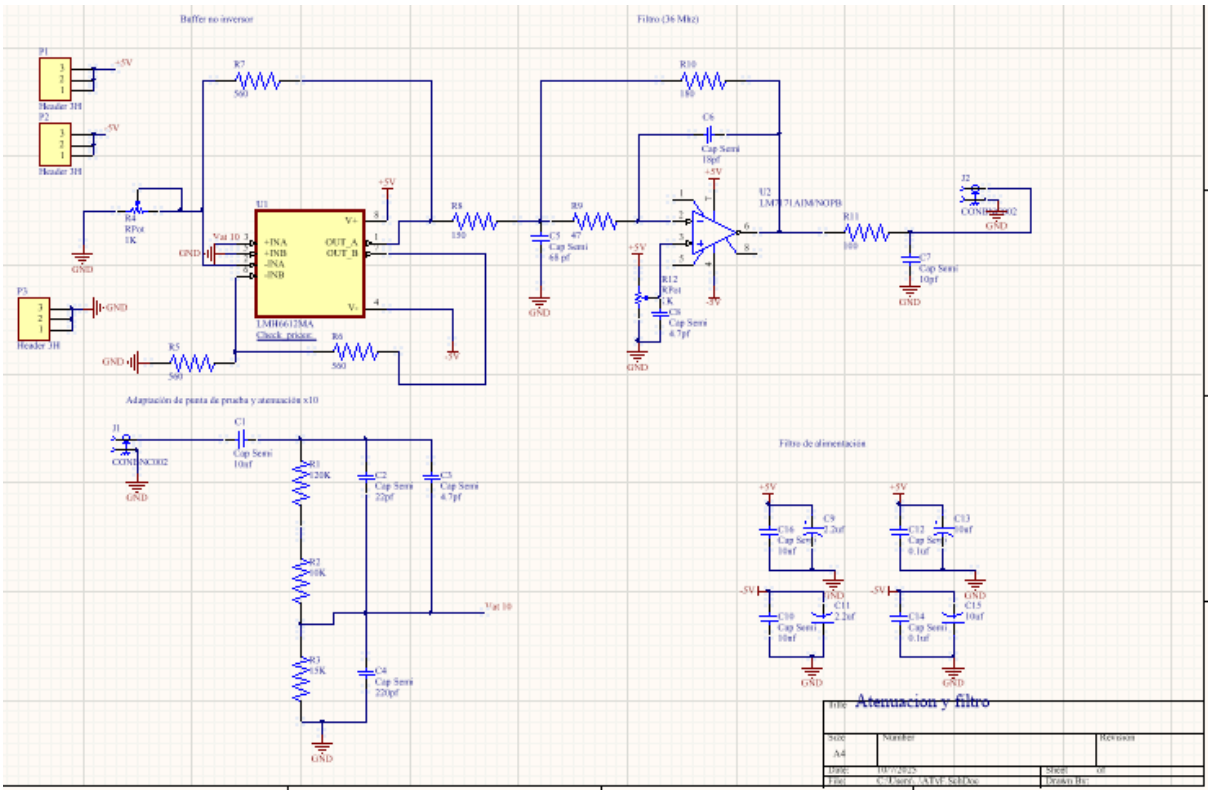


Figura 3.54 Esquema de adecuación y filtro de entrada

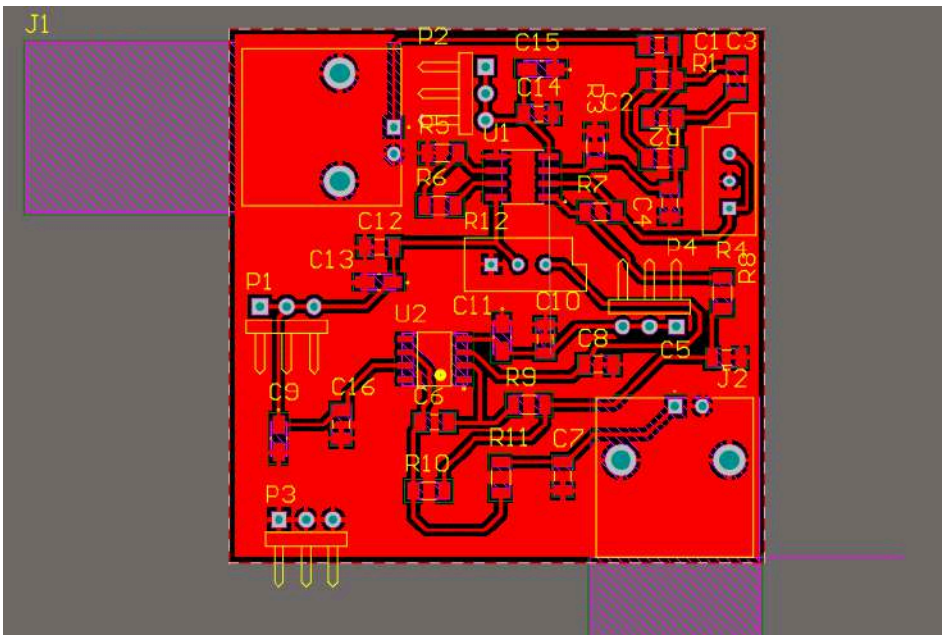


Figura 3.55 Diseño PCB de adecuación y filtro de entrada

A la salida del circuito se colocó en paralelo una capacidad de 10 picofaradios que corresponde a la capacidad que tiene el diodo de protección.

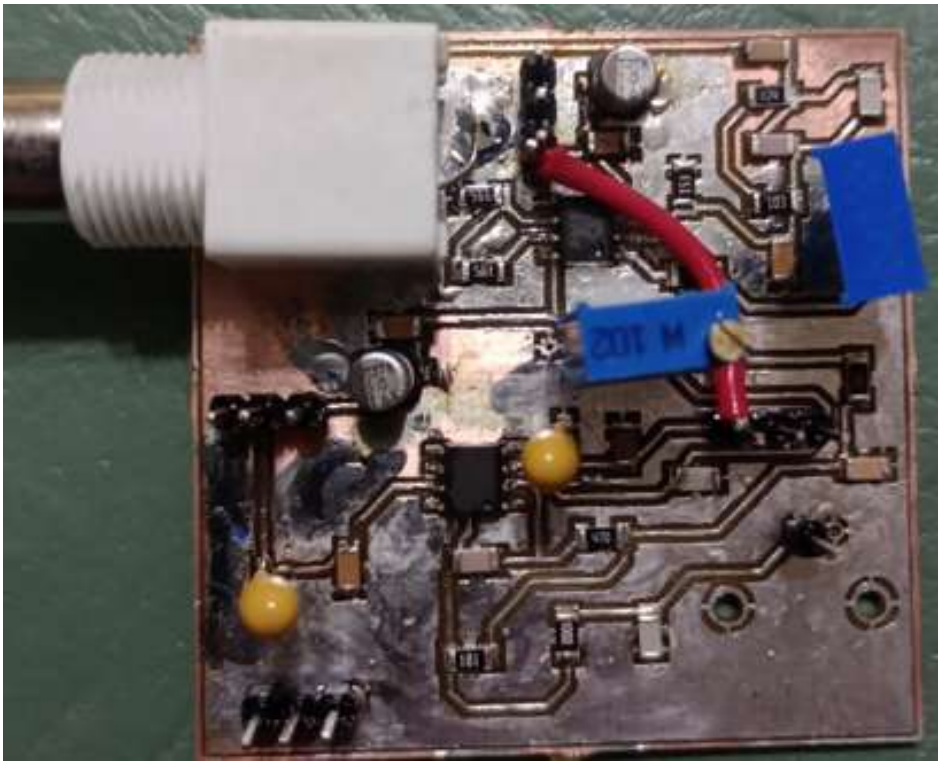


Figura 3.56.PCB de adecuación y filtro de entrada

El cable une la línea de alimentación de -5v. Para recibir la señal se utilizó un pin de tira de poste ya que no se contaba con otro BNC horizontal.

En primer lugar, se verificó que la respuesta midiendo con el osciloscopio y el generador de señales sea acorde a lo esperado en el rango de frecuencias de 0 a 25 MHz que es el que tiene el generador de señales.

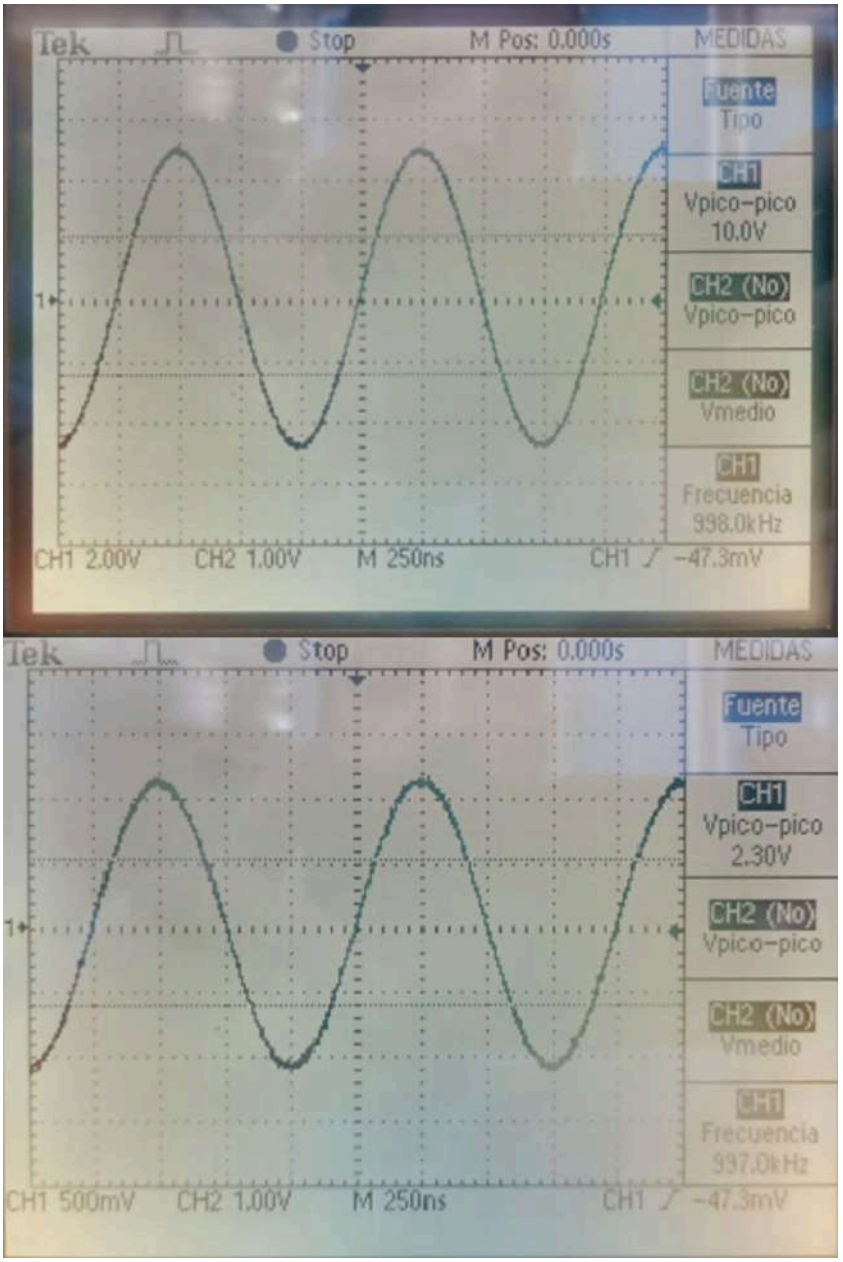


Figura 3.57. Respuesta del circuito midiendo con el osciloscopio en 1MHz.

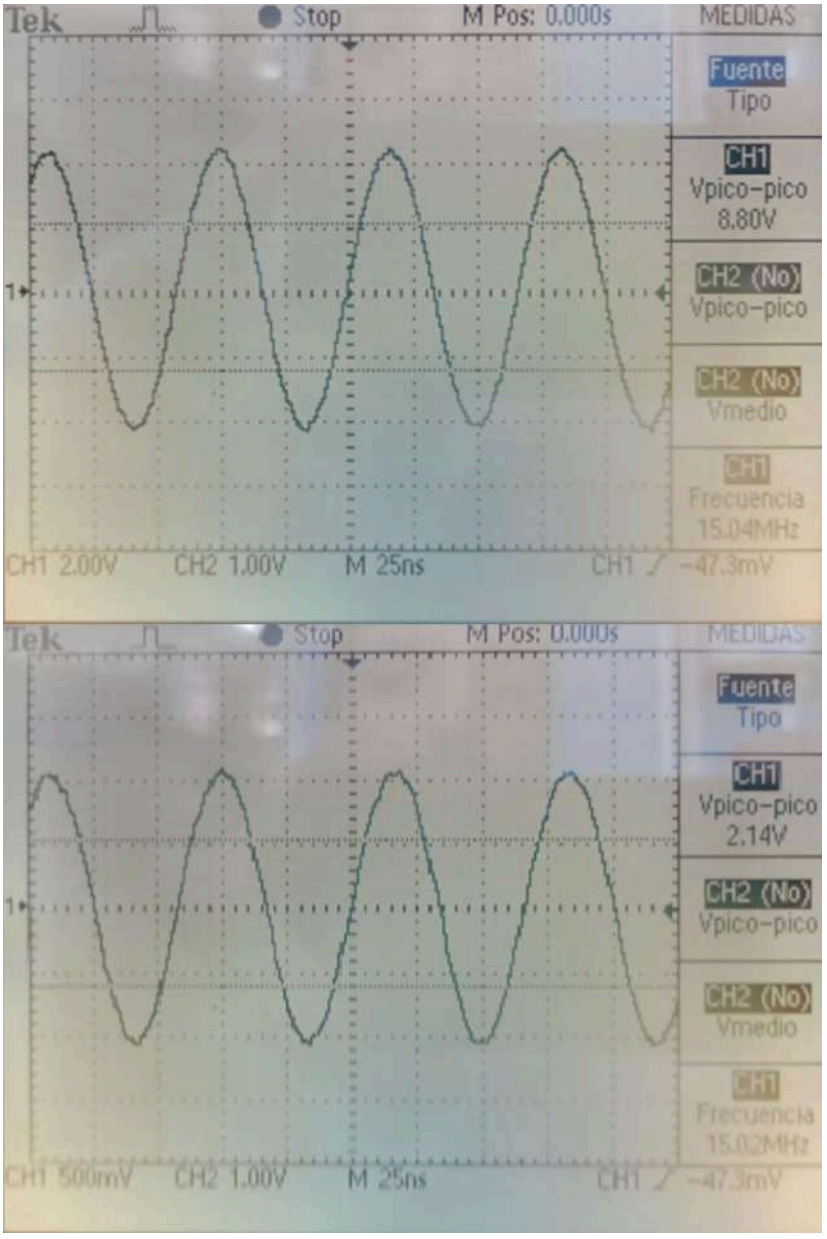


Figura 3.58. Respuesta del circuito midiendo con el osciloscopio en 15MHz.

Entrando con una señal de 9.5V se midió a la entrada y a la salida del circuito. Calculando la ganancia en dB, se obtiene que en 1 MHz tenemos una ganancia de -12.76 dB mientras que en 15 MHz tenemos una ganancia de -12.28 dB. Lo que indican un sobrepico de 0.6 dB. La ganancia de forma teórica debería darnos -14 dB (-20 de la etapa de atenuación +6dB de la etapa de ganancia de los operacionales). La diferencia que obtenemos se debe a que la ganancia de la etapa de adecuación es un poco mayor debido al límite de excursión del preset de 1K, y que la atenuación de la primera etapa es menor a 10 veces.

Midiendo con el analizador de espectro, utilizando el tracking generator para barrer en frecuencia, obtenemos la siguiente respuesta.

Figura 3.59. Respuesta en frecuencia real.

Figura 3.60. Frecuencia de corte .

Los resultados obtenidos fueron diferentes a los obtenidos con la placa experimental tipo true hole.

Por un lado, en baja frecuencia se ve una ganancia de -17.2dBm como se ve en la figura 3.58.,con un sobrepaso de 1.1dB. La caída de 3 dB se produce a los 35 MHz. La ganancia en baja frecuencia debería estar alrededor de los -12.76 dBm según los cálculos a partir de las mediciones con el osciloscopio, pero la medición que se observa en el analizador es de -17.2 dBm. Esto se debe a 2 motivos:

1) La resistencia de salida del circuito es de 100 ohms (se deprecia la resistencia de salida del operacional LM 7171 debido a la realimentación), y la entrada del analizador es de 50 ohms. Esto forma un divisor resistivo que tiene la siguiente atenuación.

$$At = 20\,log(\frac{50}{50+100}) = -\,9.54dB$$

Por lo tanto se verían reflejados -22.3 db. Pero sigue siendo distinto a los -17.2 dB obtenidos. La resistencia de 100 ohms que en este caso está causando la desadaptación, es utilizada para limitar la corriente y proteger el dispositivo.

2) El segundo motivo es que la entrada del circuito tampoco está adaptada para el analizador. El analizador espera que en la entrada del circuito se tenga:

$$Vi = VS.\frac{Zin}{Zin+50} = VS.\frac{50}{50+50} = VS\frac{1}{2}$$

Pero lo que realmente ocurre es:

$$Vi = VS.\frac{Zin}{Zin+50} = VS.\frac{150k}{150k+50} = VS.0.999$$

La diferencia es de 6 dB y en el analizador se ve reflejado como ganancia. Entonces, deberíamos ver -16.3 dB que es cercano al valor que observamos. La longitud de los cables para la conexión del instrumento y el no utilizar un bnc para la salida del circuito pueden causar la atenuación que tenemos para llegar a los -17.2 dB.

Por lo tanto, los resultados en el analizador de espectro están justificados.

Colocando un capacitor en paralelo a la resistencia de realimentación del LMH 6612 y ajustando el valor del capacitor C3 que se encuentra en paralelo en la etapa de atenuación, se puede mejorar el sobrepaso observado con el osciloscopio, además se reemplazará el preset multivuelta de 1K por uno de 2K para tener mejor rango de variación de ajuste en la ganancia.

Finalmente, se tomó una medición del de offset de tensión en la salida del circuito con un multímetro de banco, variando la temperatura con una pistola de calor, obteniendo una variación de alrededor de 5 mV para un cambio en temperatura de 29 a 66 grados centígrados. En las siguientes figuras se pueden observar las mediciones.

Figura 3.61. Medición ΔDC vs ΔTemp.

Figura 3.62. Medición ΔDC vs ΔTemp.

Disminución de 10% del área del diseño PCB de la placa de adquisición.

Finalmente, la última mejora para la nueva placa de adquisición consiste en disminuir el área del diseño PCB en un 10 %. Al realizar la interfaz gráfica para ajustar el banco de medición, se podría eliminar dos pushbutton y los dipswitch que funcionaban como ajuste para la placa original, que para el nuevo sistema no son utilizados.

Por otro lado se debía agregar las 4 resistencias para mejorar la integridad de la señal de cada adc y los cambios realizados en la etapa de adecuación y filtrado.

El diseño se realizó en el software *Altium Designer*.

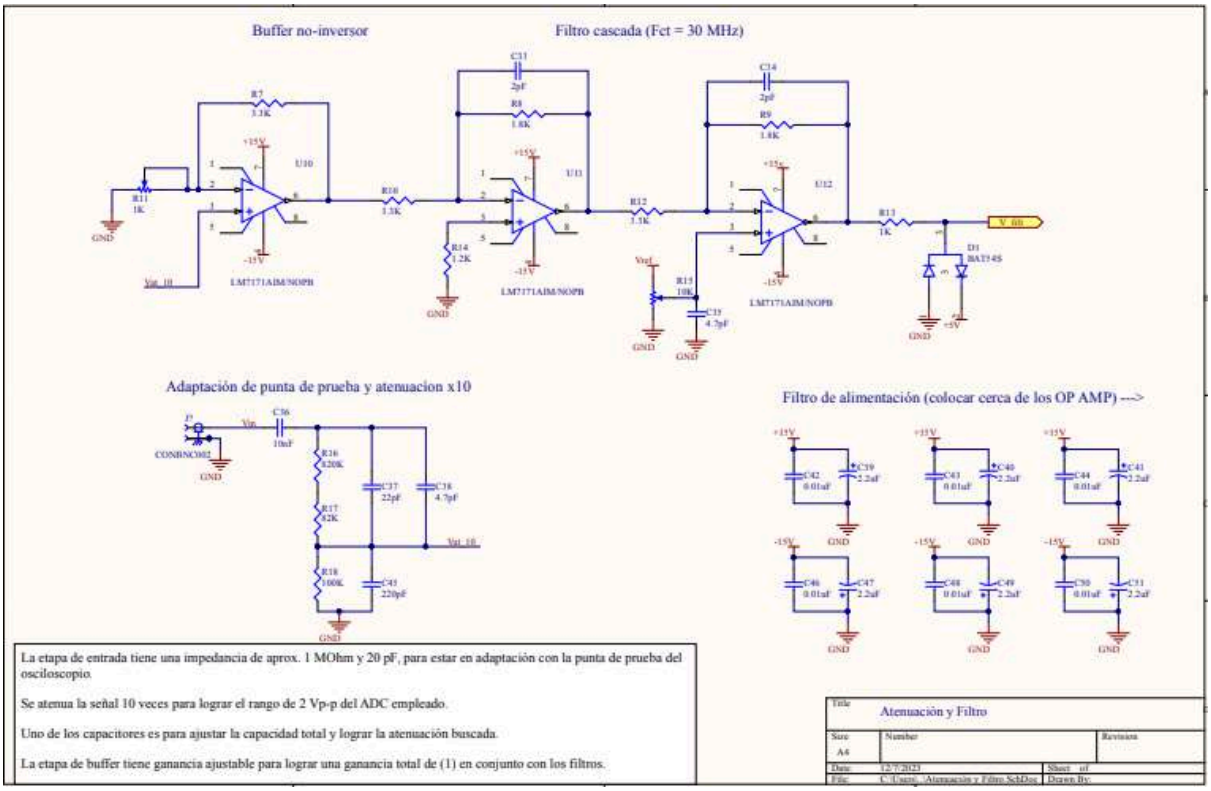


Figura 3.63. Etapa de adecuación y filtro del diseño original.

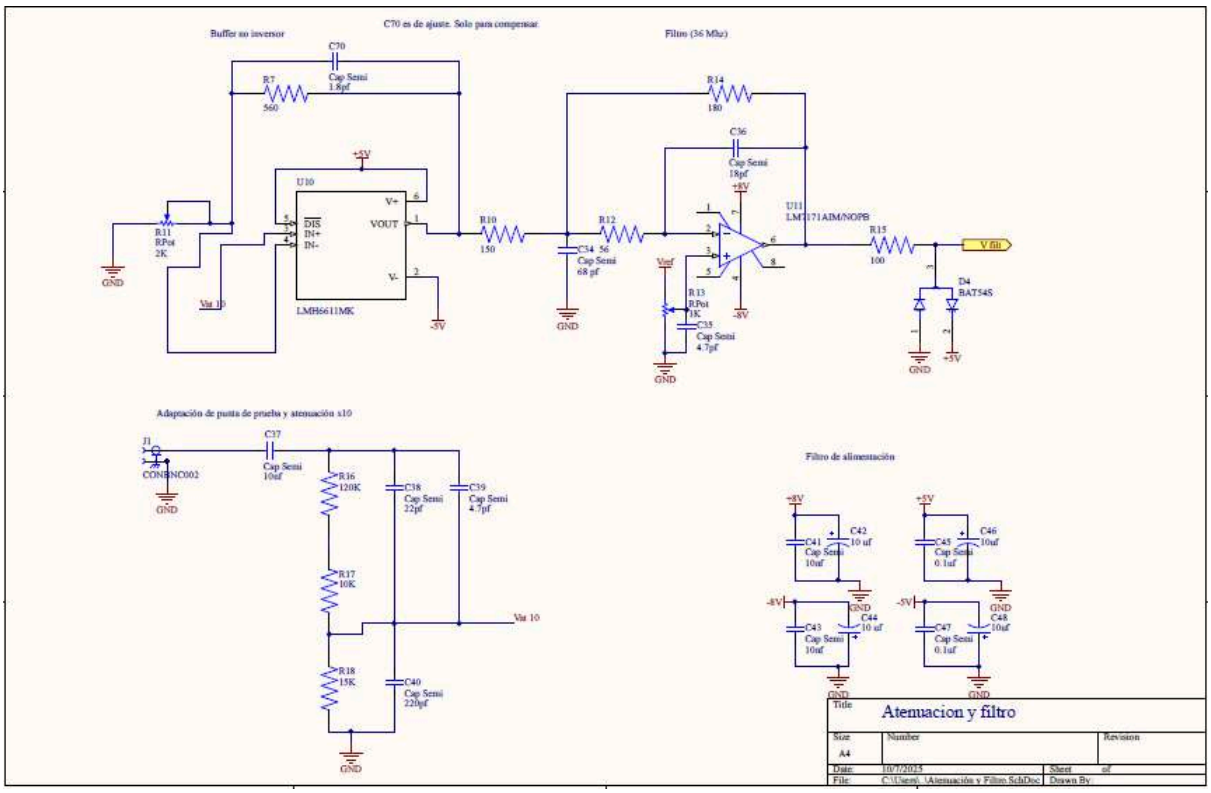


Figura 3.64. Etapa de adecuación y filtro del diseño original.

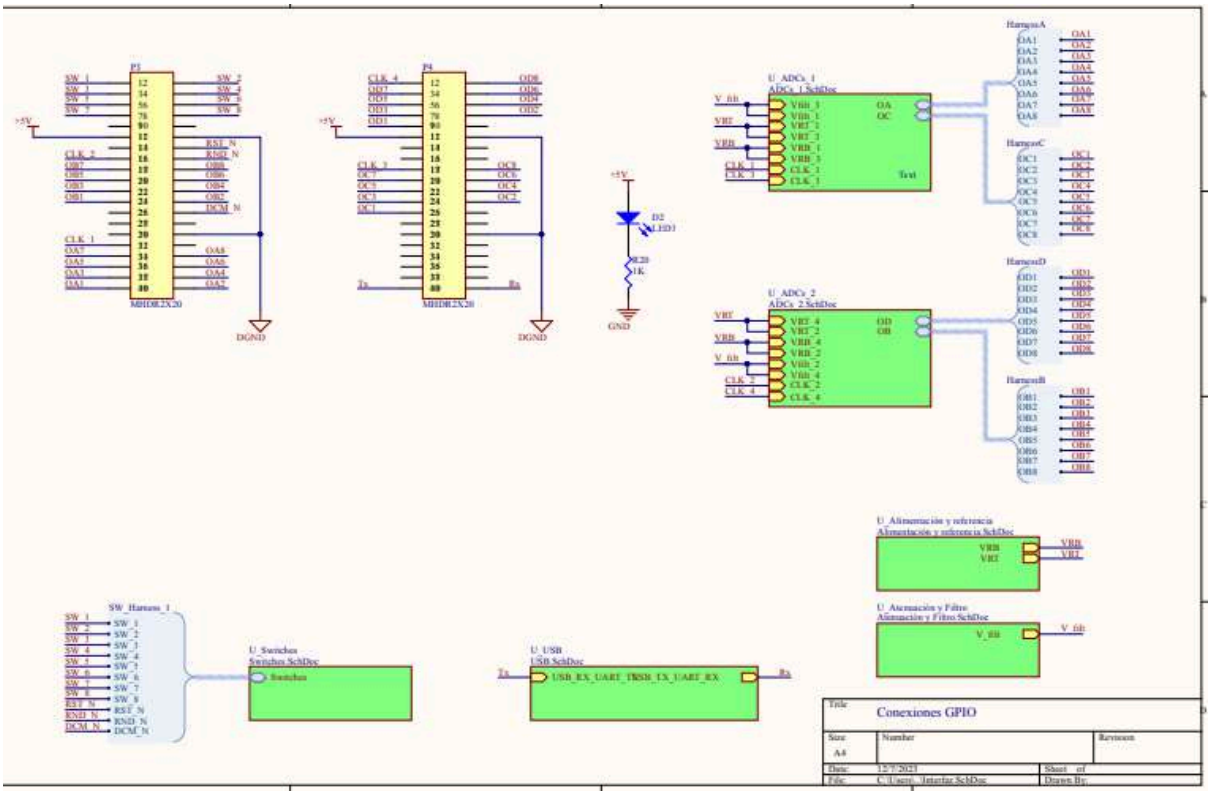


Figura 3.65. Conexión de GPIO con adc del diseño original.

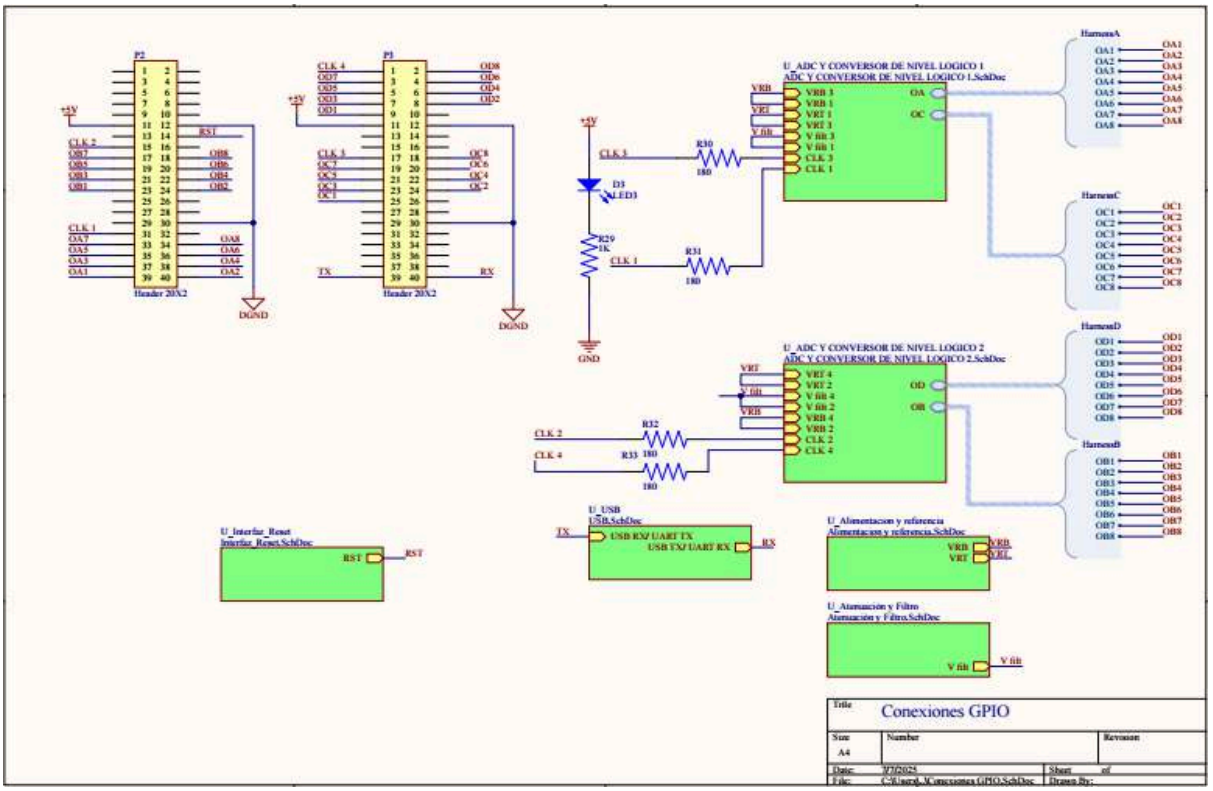


Figura 3.66. Conexión de GPIO con adc del diseño original.

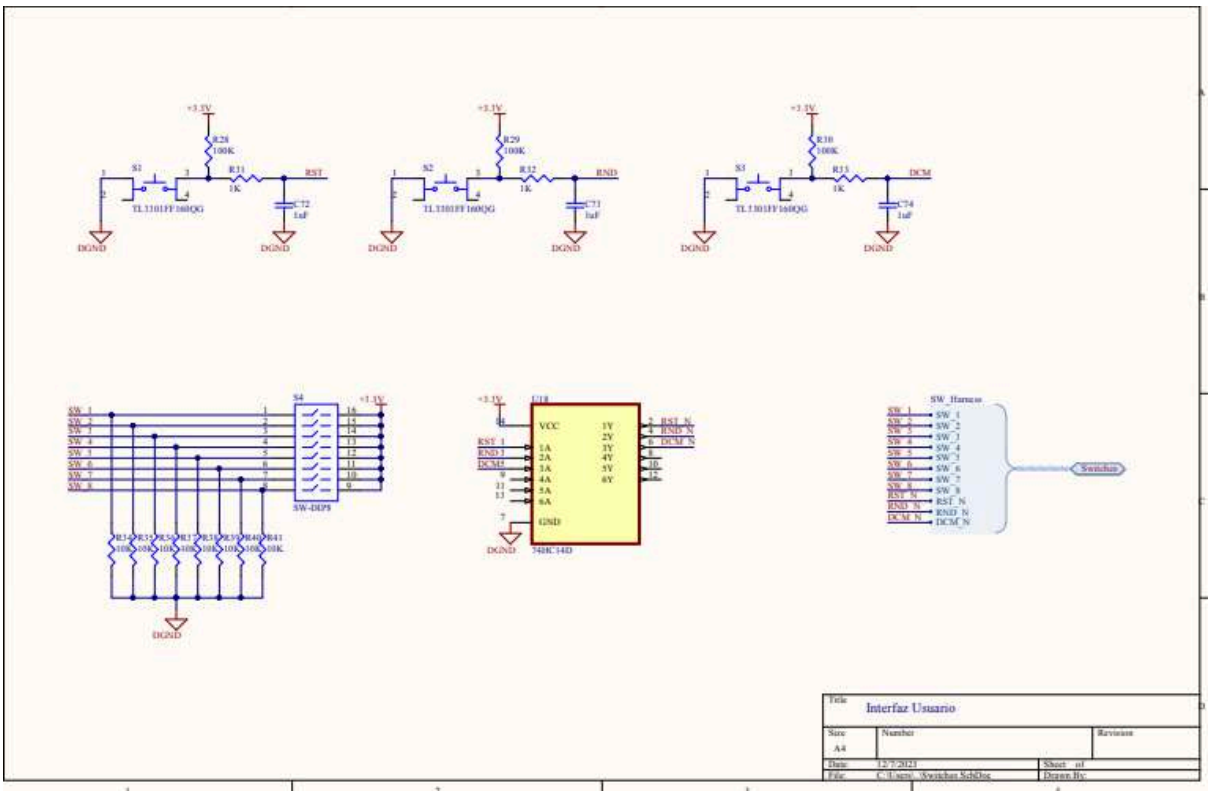


Figura 3.67. Etapa de interfaz con usuario del diseño original.

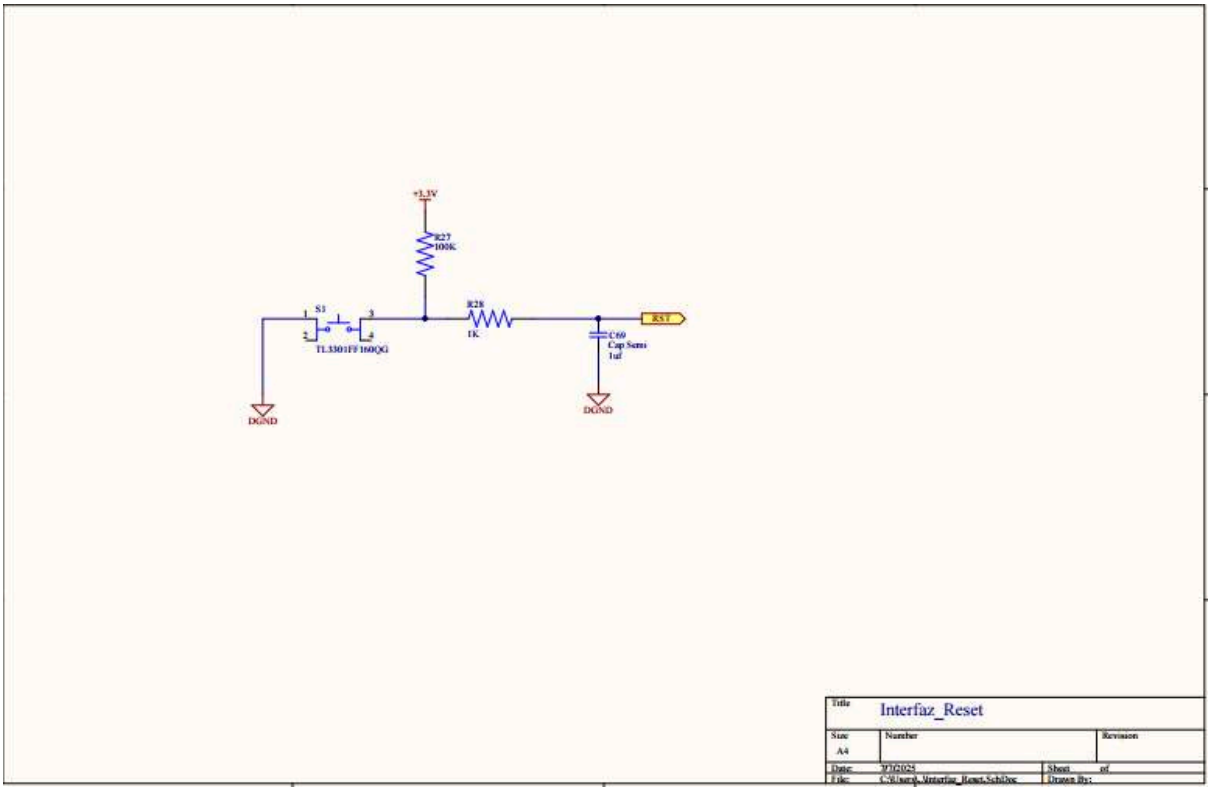


Figura 3.68. Etapa de interfaz con usuario del diseño original.

En las figuras siguientes se puede ver el diseño de PCB nuevo con un área de 71,8 cm^2, siendo antes el área de $12x7 = 84cm^2$. En la sección de Especificaciones Técnicas se pueden observar los cambios realizados con más detalle.

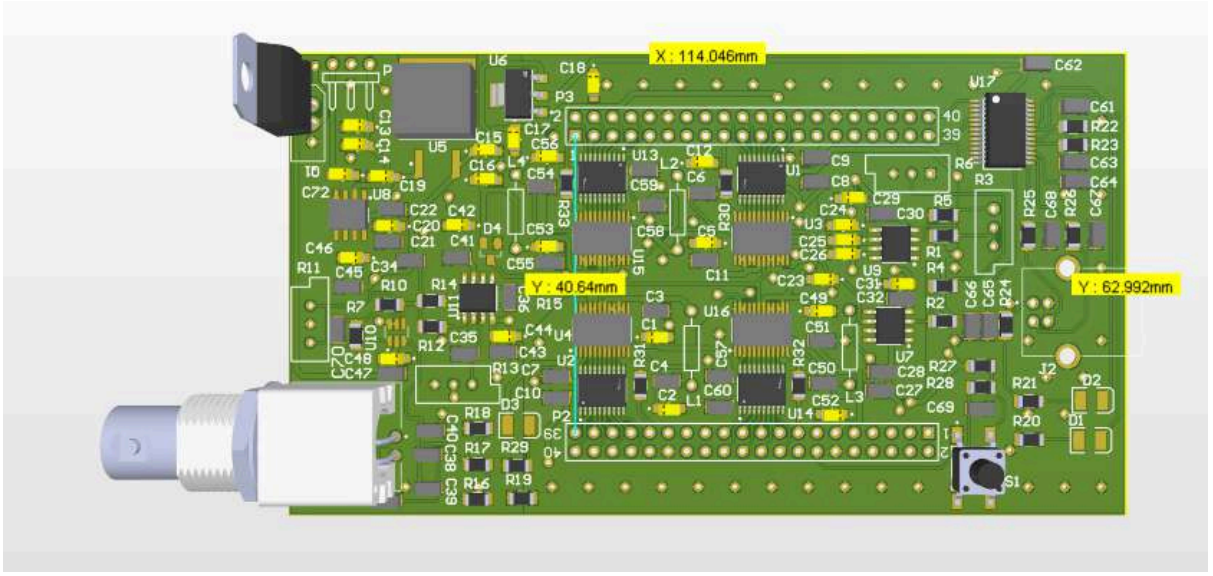


Figura 3.69. PCB en vista 3D.

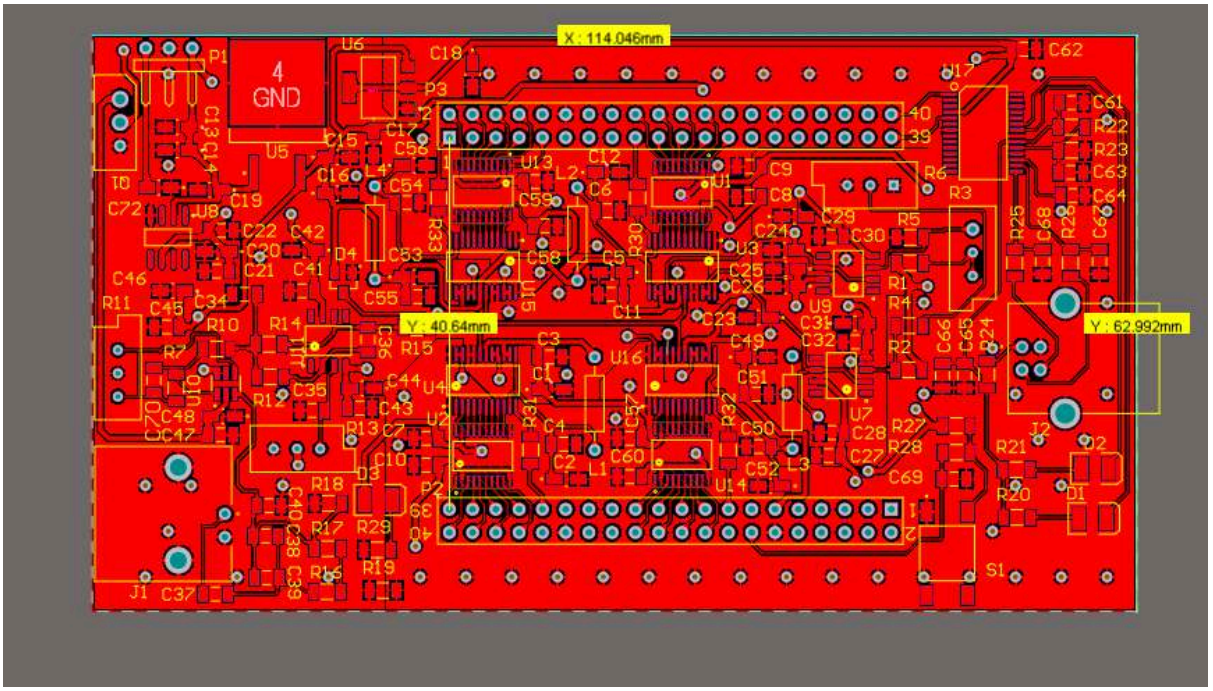


Figura 3.70. PCB Top Layer.

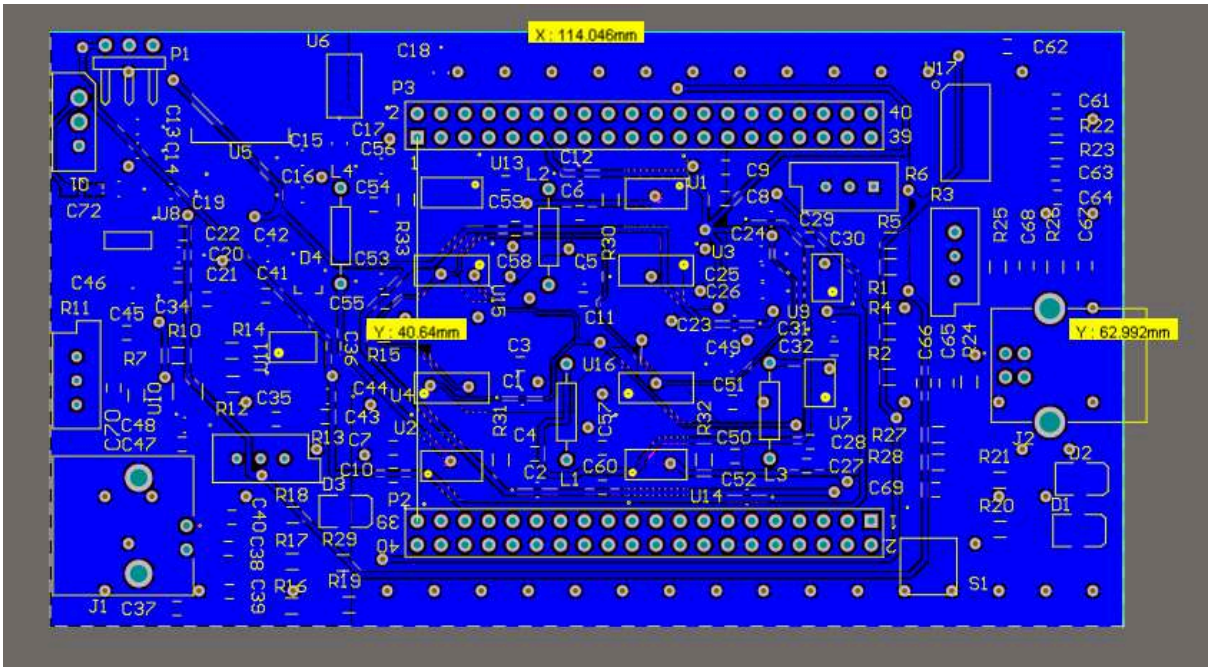


Figura 3.71. PCB Bottom Layer.

4.Pruebas y mediciones del banco de prueba

Para probar el banco de prueba, que une el sistema realizado en VHDL y la interfaz gráfica portable, se utilizó la placa de adquisición realizada por el Ing.Medina. Se cargó en la memoria de programación de la DE0-Nano el archivo del proyecto VHDL. Se conectó la placa DE0-Nano a la placa de adquisición y se conectó la alimentación a la fuente de alimentación. Se conecta la placa a la PC mediante un cable USB y la placa al generador de señales. Luego, se abre la interfaz gráfica y se resetea manualmente la placa de adquisición para reiniciar el sistema vhdl de la FPGA.

Se probó adquirir señales senoidales y cuadradas a distintas frecuencias, aplicando distintas cantidades de muestras, diezmados y secuencias pseudo-aleatorias. Para demostrar el funcionamiento en esta sección se decidió usar 4 tipos de pruebas fundamentales, adquiriendo una señal de 1 MHz de 8 Vpp de amplitud:

1. Adquisición de senoidal utilizando secuencia de unos.
2. Adquisición de senoidal utilizando secuencia pseudo-aleatoria.
3. Adquisición de señal cuadrada.
4. Adquisición de senoidal utilizando más cantidad de muestras y una secuencia pseudo-aleatoria.

Las secuencias pseudo-aleatorias se obtuvieron a partir de un código utilizado en python que simula un LFSR (Registro de corrimiento con realimentación lineal).

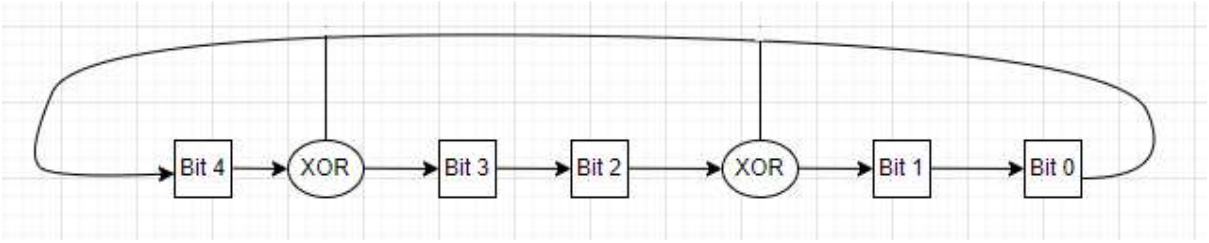


Figura 4.01. Esquema general de LFSR.

Cada recuadro que almacena un bit está constituido por un Flip Flop. Modificando la cantidad de FF's y el tipo de realimentación se pueden realizar secuencias más o menos aleatorias.

En algunos casos, se comparará la SFDR obtenida con la que se obtiene en *matlab* utilizando su función SFDR. Para ello, se utilizarán los archivos de las muestras almacenadas en la PC.

1)**Secuencia de unos**

En primer lugar, se ingresó una señal senoidal de 8 Vpp de 1 MHz de frecuencia. La cantidad de muestras elegidas son 16384. La secuencia elegida es de todos unos. Como se ve en la respuesta, la SFDR nos da un valor de 32.2 dB. Comparándolo con el resultado obtenido en *matlab* se observan resultados iguales. Además, se observa, que el bin espurio se encuentra alrededor de los 20 MHz que es la frecuencia a la que muestrea cada ADC individualmente.

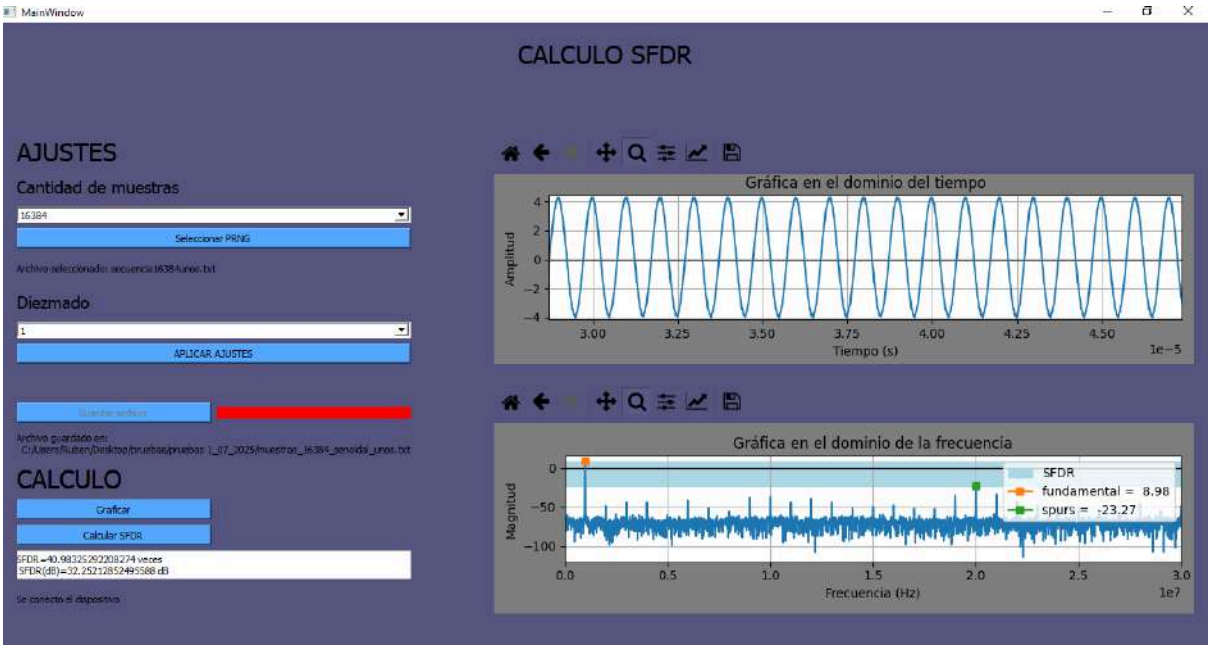


Figura 4.02. adquisición secuencia de unos en interfaz.

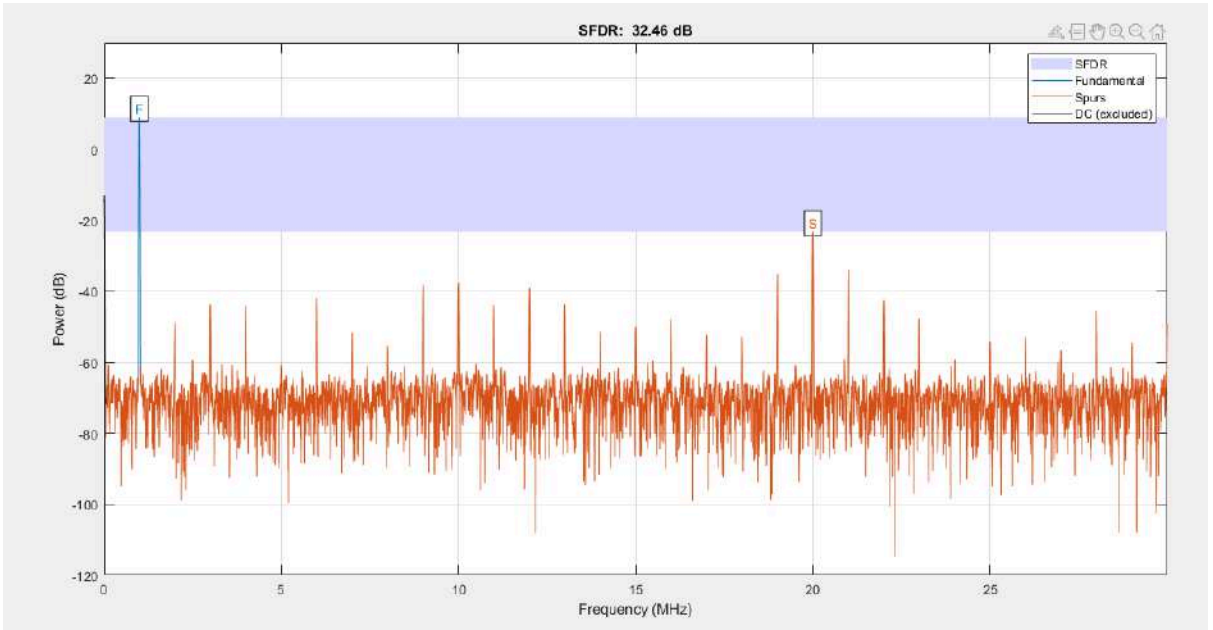


Figura 4.03. adquisición secuencia de unos en matlab.

2)**Secuencia de pseudo-aleatoria**

En este caso, se adquirió la misma señal pero con una secuencia pseudo-aleatoria. Se observa la mejora en el valor de la SFDR de 46.18 dB. La cantidad de muestras elegidas son 16384.

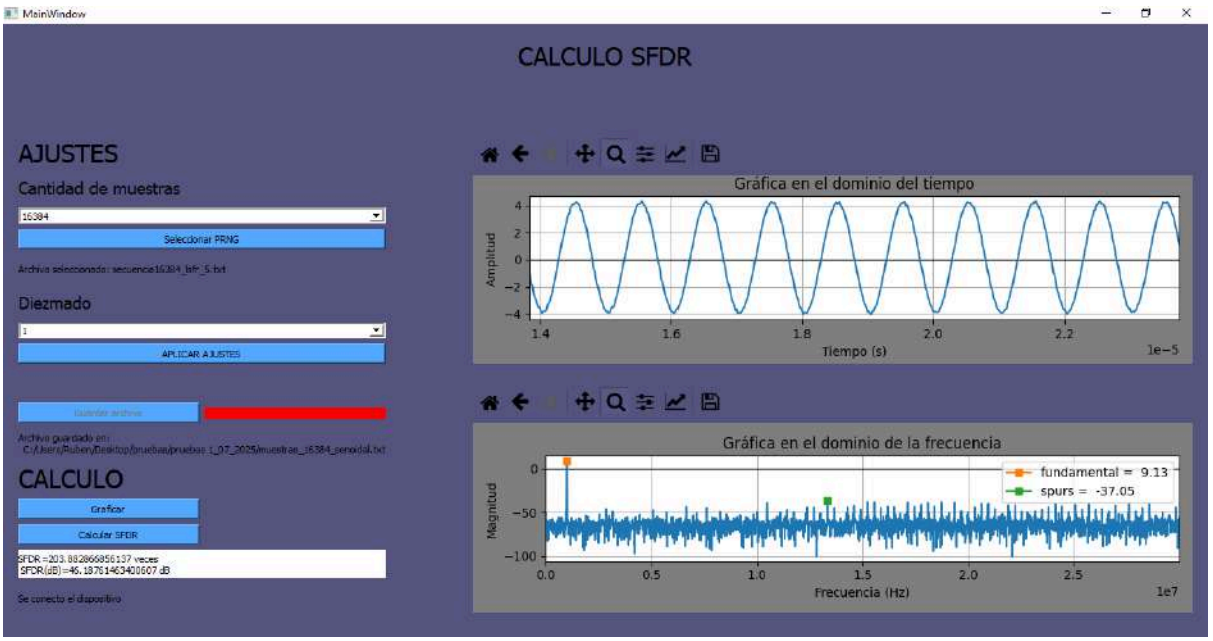


Figura 4.04. adquisición secuencia de pseudo-aleatoria en interfaz.

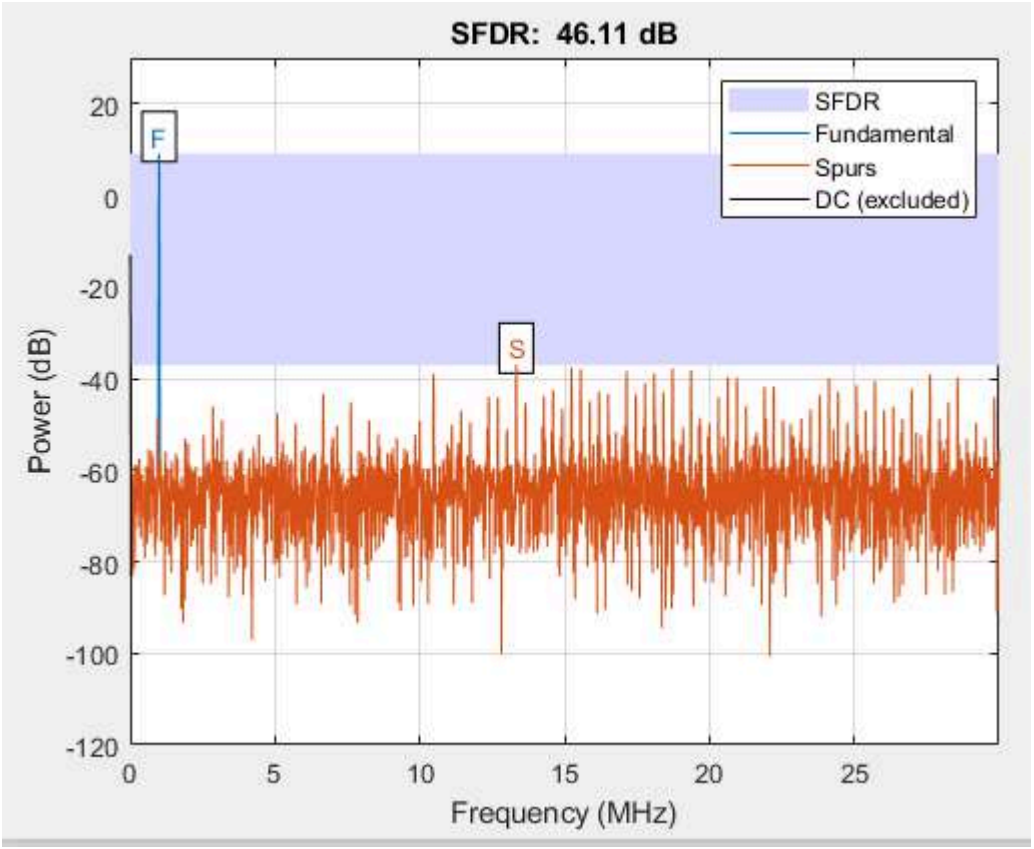


Figura 4.05. adquisición secuencia de pseudo-aleatoria en matlab.

3)**Adquisición de señal cuadrada**

En este prueba, no se busca calcular la SFDR ya que el bin con más amplitud que le sigue a la componente fundamental, es la primera armónica. Lo que se busca analizar es la fidelidad de la representación de la señal cuadrada. La cantidad de muestras elegidas son 16384.

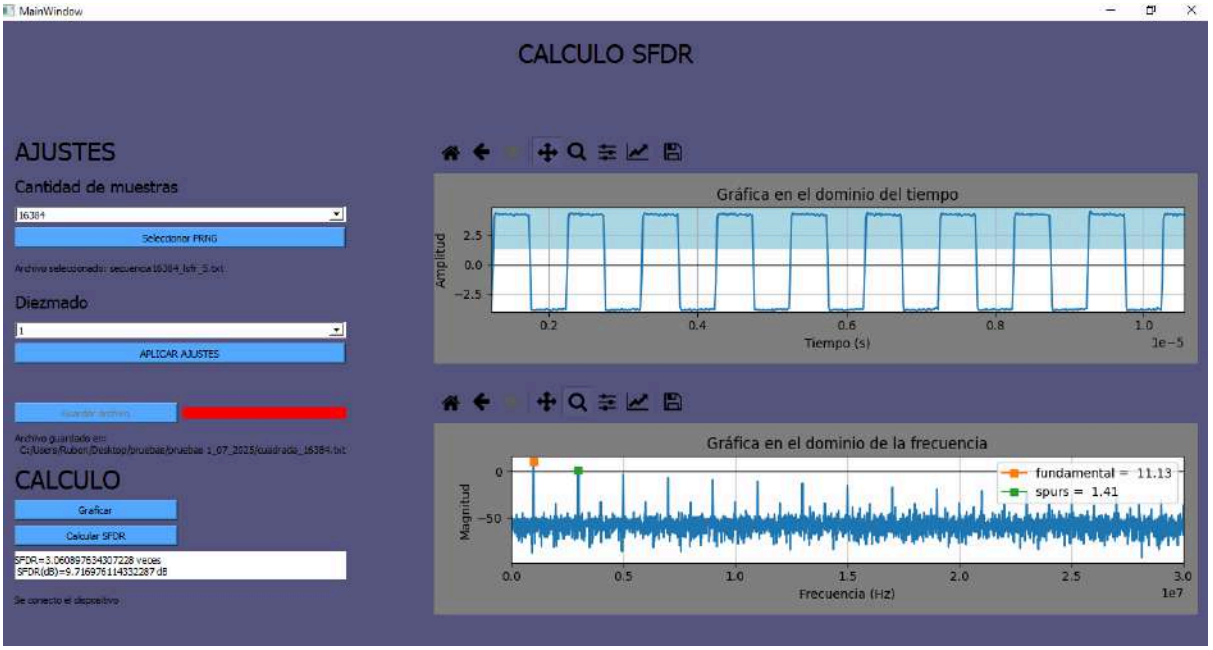


Figura 4.06. adquisición señal cuadrada en interfaz.

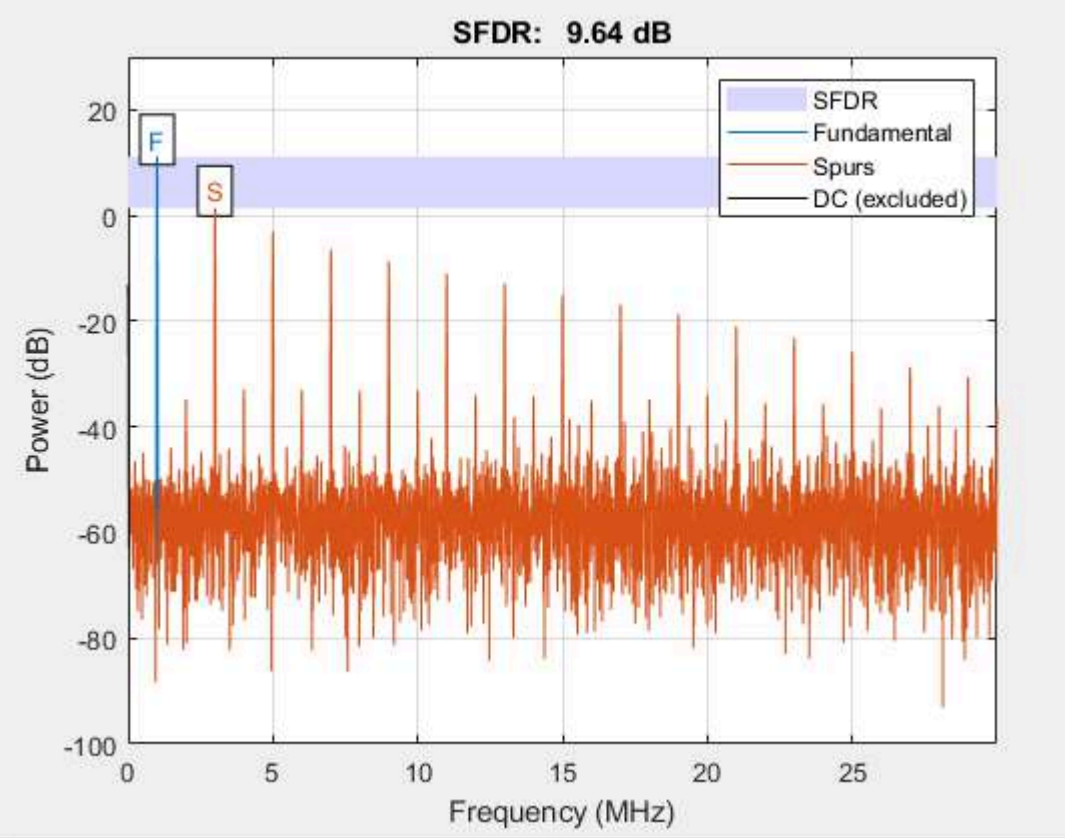


Figura 4.07. adquisición señal cuadrada en interfaz.

4) **Adquisición de senoidal utilizando más cantidad de muestras y una secuencia más aleatoria.**

En esta última prueba se quiso demostrar la diferencia entre usar diferente tipos de secuencias pseudo-aleatorias, de tal forma de que al aumentar la aleatoriedad, se mejora el valor de la SFDR. Aquí nos encontramos con un problema. Al utilizar una frecuencia más aleatoria, el valor de la componente espuria debido al uso de los ADC disminuye y aparece otra componente alrededor de los 20 KHz que tiene más amplitud y no permite calcular la SFDR correctamente.

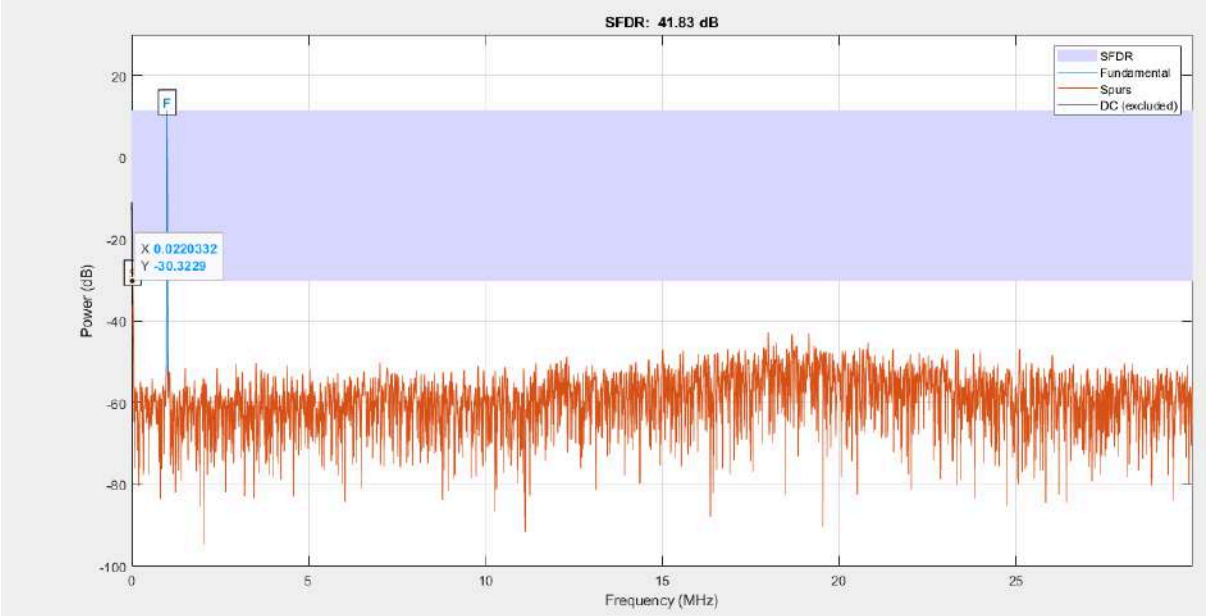


Figura 4.08. Captura de sfdr en matlab.

Luego de investigar exhaustivamente de donde provenía esa componente, midiendo cada sector de la placa de adquisición, probando el sistema con otras PC y utilizar distintos cables USB, se decidió hablar con el laboratorio de sistemas caóticos para tomar una decisión. La solución fue, no tomar en cuenta las componentes hasta los 50 KHz ya que no era de su interés lo que ocurre en esa rango de frecuencias. Esto se realizó editando el código de cálculo de la SFDR.

Ahora si, luego de solucionar el problema se puede realizar cualquier tipo de prueba. En las imágenes siguientes se puede ver la diferencia del valor de la SFDR viendo las distintas amplitudes que tiene la componente espuria, mejorando para una secuencia más aleatoria.

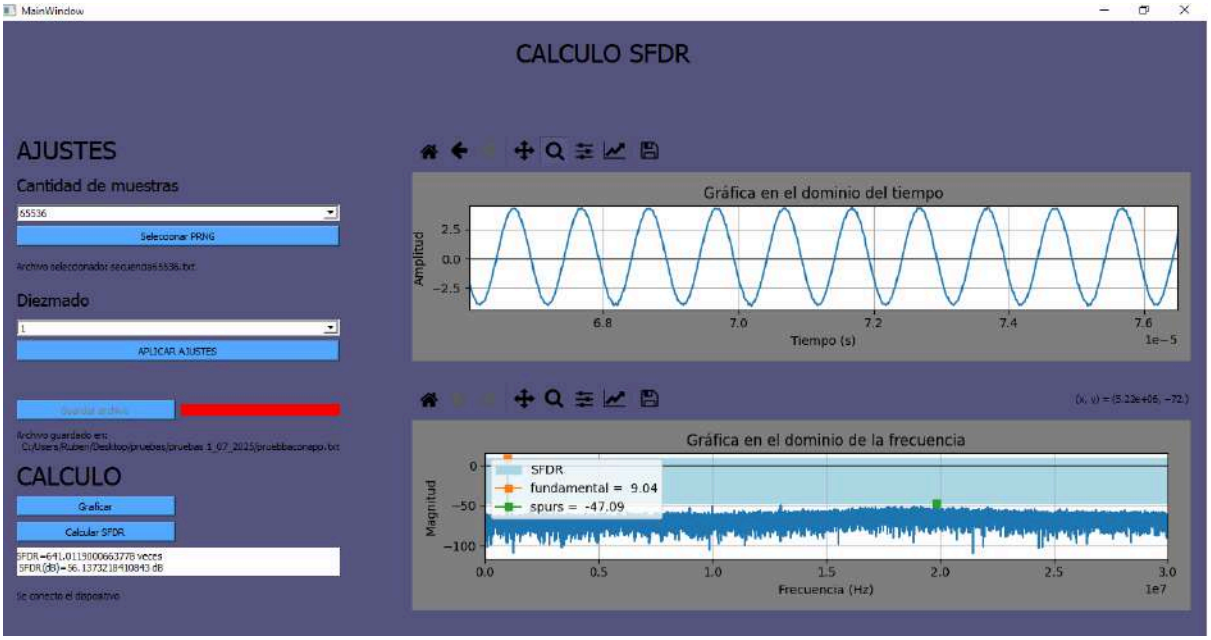


Figura 4.09. Adquisición señal con secuencia más aleatoria.

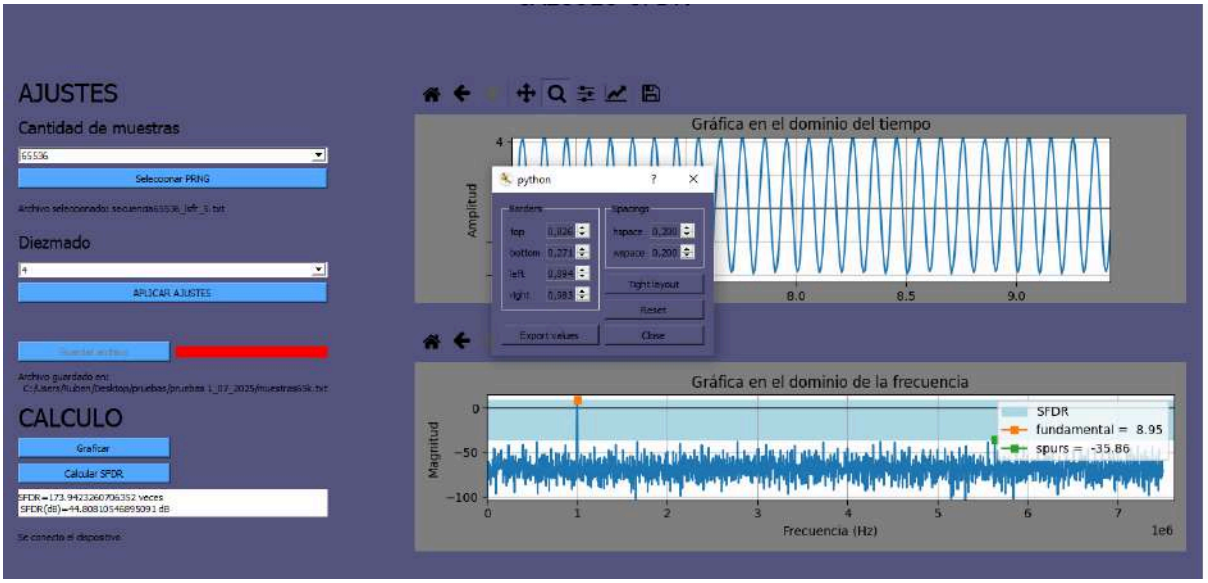


Figura 4.10. adquisición señal con secuencia menos aleatoria.

5.Conclusiones

El trabajo final de la carrera se logró realizar a lo largo de alrededor de 14 meses. Dentro del proceso, se pudieron aplicar conceptos vistos a lo largo de la carrera. La temporización y sincronización de sistemas digitales, la utilización de lenguaje VHDL para diseñar cualquier tipo de circuito digital en la FPGA, la realización de filtros activos y aplicación de distintas etapas analógicas son ejemplos claros de estos conceptos.

También, la posibilidad de aprender conceptos nuevos y estar en contacto con sistemas y procesos con los que nunca se había experimentado como la realización de una placa PCB utilizando un dispositivo como la CNC que permite darle forma al diseño o la investigación y el uso del lenguaje *Python* para la realización de la interfaz gráfica.

Desde el punto de vista de gestión del proyecto, se puede concluir que se debió tener más precaución a la hora de elegir los tiempos de proceso en el plan del proyecto. Más allá de que hay muchos tiempos que no dependen del diseñador como la llegada de los componentes necesarios, se debe tener un margen de tiempo mayor en ese tipo de tareas. El proyecto finalizó alrededor de dos meses después del punto final que se había calculado originalmente.

Más allá de este último detalle, con el desarrollo del banco de prueba utilizando el VHDL, la interfaz gráfica portable, sumado a las mejoras realizadas probadas de forma experimental y el diseño de una placa de adquisición nueva se puede concluir que el trabajo final fue un éxito donde se pudieron abordar las distintas ramas que abarca la carrera.

6.Bibliografía

[1] Gabriele Manganaro y Dave Robertson. “Interleaving ADCs : Unraveling the Mysteries”. En: Analog Dialogue 49 (07 2015).

[2] Howard Johnson y Martin Graham . High-speed digital design. 1993.

[3] Alan Oppenheim y Schafer Ronald. Tratamiento de Señales en Tiempo Discreto. 2009.

[4] Wai-Kai Chen. Circuits and Filters. Second edition. 2003.

[5] Python Software Foundation. (2024). *Python 3 documentation*. https://docs.python.org/3/

[6] Hardi Electronics AB. VHDL Handbook. 2000.

[7]Vishay Intertechnology. (s.f.). *LCA series – Carbon composition resistors* (Hoja de datos n.º 20135). https://www.vishay.com/docs/20135/lca.pdf

[8]Paul, S. [psambit9791]. (2022, junio). *pysnr* (versión 0.0.1) [Repositorio en GitHub]. GitHub. https://github.com/psambit9791/pysnr

7.Apéndice

- Especificación de requerimientos
- Plan de proyecto
- Especificaciones funcionales
- Especificaciones técnicas
- Plan de pruebas
- Github del código de la interfaz
- Diseño de placa de adquisición original
- Nuevo diseño de placa de adquisición