IMPLEMENTACIÓN DE UN SISTEMA PARA ADQUISICIÓN Y PROCESAMIENTO DE SEÑALES DE AUDIO A TRAVÉS DE LAS TARJETAS ZEDBOARD

Jeison Julián Suarez Ríos

Ingeniería Electrónica

Director(es) del trabajo de grado

Luis Fernando Castaño

INSTITUTO TECNOLÓGICO METROPOLITANO

1 de agosto de 2016

RESUMEN

En este trabajo de grado se presenta una aplicación de procesamiento de audio. La implementación del proyecto se realiza en un sistema de desarrollo ZedBoard. Se emplea un diagrama de bloques para la descripción del sistema en el entorno *Vivado Design Suite* de *Xilinx.* La señal de audio se ingresa a través de la línea de entrada de la ZedBoard. La conversión de la señal es realizada por un CODEC de audio que se comunica con el procesador a través de un bus I2C. El control del CODEC es ejecutado a través de un programa en lenguaje C desarrollado sobre el *Software Development Kit* (SDK) de *Xilinx*. Esta aplicación es ejecutada por un sistema procesamiento ZYNQ-7000 en uno de los núcleos ARM Cortex-A9 de la SoC FPGA. La señal digitalizada se almacena en una tarjeta SD y se le aplica un filtro digital FIR pasa altas empleando el *IP Core* disponible en el entorno de desarrollo. Los resultados obtenidos son comparados con la simulación realizada en MATLAB.

Palabras clave: FPGA, Adquisición de datos, Procesamiento de audio, Filtro digital, almacenamiento de datos, *script*

RECONOCIMIENTOS

Quiero agradecer a la Institución Universitaria ITM por brindarme la oportunidad de desarrollarme no solo profesional sino también personalmente. Agradezco al laboratorio de Sistemas de Control y Robótica por la posibilidad de desarrollar este trabajo de grado bajo la modalidad de Producto en Laboratorio de Investigación sin olvidar en especial a los profesores Luis Fernando Castaño y David Márquez por el apoyo y asesorías brindadas para la realización de este trabajo. También quiero hacer un reconocimiento y expresar la más sincera gratitud a mi empresa EPM por brindarme el apoyo no solo económico sino también por el tiempo laboral cedido para poder realizar mis estudios. Igualmente agradezco a mis compañeros de estudio, familiares y amigos que de una u otra forma aportaron para el cumplimento de mi pregrado. Finalmente, y lo más importante de todo es agradecer a Dios por todas las bendiciones y oportunidades que envió en mi vida para que hoy día y en el futuro sea un profesional que aporte al desarrollo de esta sociedad.

ACRÓNIMOS

FPGA Field Programmable Gate Array

DAQ Data Adquisition

SD Secure Digital

ADC Anolog Digital Converter

DAC Digital Anolog Converter

ARM Advanced RISC Machine

RISC Reduced Instruction Set Computer

DSP Digital Signal Processing

FIR Finite Impulse Response

PC Personal Computer

TABLA DE CONTENIDO

1. INTRODUCCIÓN

Existen diferentes arquitecturas de hardware para la implementación de algoritmos de procesamiento digital de señales. Entre estas, el empleo de sistemas embebidos ha tenido gran acogida por las características de flexibilidad, consumo de potencia y procesamiento. En este trabajo se presenta la implementación de un filtro digital FIR pasa alto por medio del sistema de desarrollo ZedBoard, el cual contiene una FPGA de la familia Zynq7000 de Xilinx y dos núcleos ARM Cortex-A9.

A pesar de las bondades de esta tecnología los estudiantes de Electrónica y Telecomunicaciones de la Institución Universitaria ITM poco la emplean debido al desconocimiento de la misma, motivo por el cual este trabajo de grado quiere divulgar y dar a conocer a los estudiantes las capacidades de usar estos dispositivos para invitarlos a iniciar el desarrollo de sus ideas basados en esta tecnología. Además, se quiere optimizar el uso de los recursos brindados por la Institución, ya que están siendo subutilizados en cuanto a estos dispositivos se refiere. Este trabajo de grado se enfoca a la rama de adquisición y procesamiento de señales digitales. Sin embargo, se deja claro que esta tecnología no solo es aplicable a la DAQ sino también a un sin fin de aplicaciones que solo la mente le pone límites.

El desarrollo de este trabajo servirá como insumo en el laboratorio de Sistemas de Control y Robótica, para el desarrollo de trabajos de grado y proyectos de investigación que requieran el uso de sistemas basados en FPGA para la adquisición y procesamiento digital de señales. Igualmente permitirá divulgar a los estudiantes del Departamento de Electrónica y Telecomunicaciones del ITM, el uso de FPGA en el procesamiento digital de señales audio. Este trabajo puede beneficiar particularmente a los estudiantes de la asignatura Diseño Digital y de Trabajo de Grado que deseen incursionar en el área de las DAQ y procesamiento de señales digitales. Para el desarrollo de este trabajo se hará uso de

los recursos disponibles en el laboratorio de Sistemas de Control y Robótica y de Microelectrónica y Nanotecnología del ITM.

En este trabajo de grado se realizará la implementación de un filtro digital sobre FPGA. Esta se realizará sobre el sistema de desarrollo Zedboard, el cual contiene una FPGA de la familia Zynq7000 de Xilinx. Para la evaluación del funcionamiento del algoritmo se emplearán señales discretas almacenadas en formato de texto en la memoria SD del sistema de desarrollo. Igualmente se realizará la adquisición de señales por medio de la interface de audio. La validación se realizará mediante la comparación de los resultados obtenidos empleando FPGA con los obtenidos para el mismo algoritmo con MATLAB.

En el capítulo 2 se presenta el marco teórico que aclara los conceptos básicos para el desarrollo del presente proyecto a través de la ZedBoard. En el capítulo 3 se presenta la metodología utilizada para lograr la adquisición, almacenamiento y procesamiento de señales de audio en la ZedBoard. En el capítulo 4 se presentan los resultados obtenidos en la ejecución este trabajo. El capítulo 5 muestra las conclusiones y recomendaciones para tener en cuenta a la hora de trabajar en DAQ, almacenamiento y procesamiento de señales. Por último, se citan las referencias y apéndices que dan soporte a este trabajo.

2. MARCO TEÓRICO

ADQUISICION DE DATOS (DAQ)

Es el proceso de medir un fenómeno físico o eléctrico tal como voltaje, corriente, temperatura, presión o sonido (NI.com, 2016). Un sistema DAQ se compone de tres partes:

- Sensores
- Convertidores ADC y DAC
- Hardware para el procesamiento de señales (FPGA, DSP, PC)

Sensores:

Los sensores son los dispositivos encargados de medir la variable física-eléctrica y convertir esa variable a una señal eléctrica medible que pueda ser interpretada por un dispositivo electrónico. Algunas de las necesidades que han surgido para medir variables físicas son:

- Se requiere determinar el comportamiento de un sistema para optimizar el rendimiento.
- Las señales de los sensores indican como está funcionando el sistema.
- Se puede usar esta información para modificar el sistema y mejorarlo.

Las señales que transmiten los sensores pueden ser digitales o analógicas. En el caso de este trabajo de grado la señal que representaría el sonido es eléctrica y analógica. Esta señal proviene de cualquier dispositivo que reproduzca audio.

Convertidores ADC y DAC:

Los convertidores ADC sirven para transformar la señal eléctrica analógica en un formato que pueda ser procesado por un dispositivo electrónico programable, señal digital. Por el contrario, un convertidor DAC sirve para realizar el proceso inverso al ADC, es decir, parte de una señal digital y la convierte a analógica. Una señal analógica es continua en el tiempo,

puede tomar cualquier valor y tiene infinitos valores, mientras que una señal digital discreta no es continua en el tiempo y tiene valores finitos. El proceso de digitalización de una señal analógica consiste en la toma de una muestra de la señal cada determinado tiempo, motivo por el cual en el proceso de digitalización se pierde información de la señal. Para una correcta digitalización se debe tener en cuenta la resolución, rango y tasa de muestreo y así garantizar una correcta representación digital de la señal analógica.

- **Resolución:**

  Es el número de niveles binarios discretos que se utilizan para representar la señal analógica como muestra la ***Figura 1***:

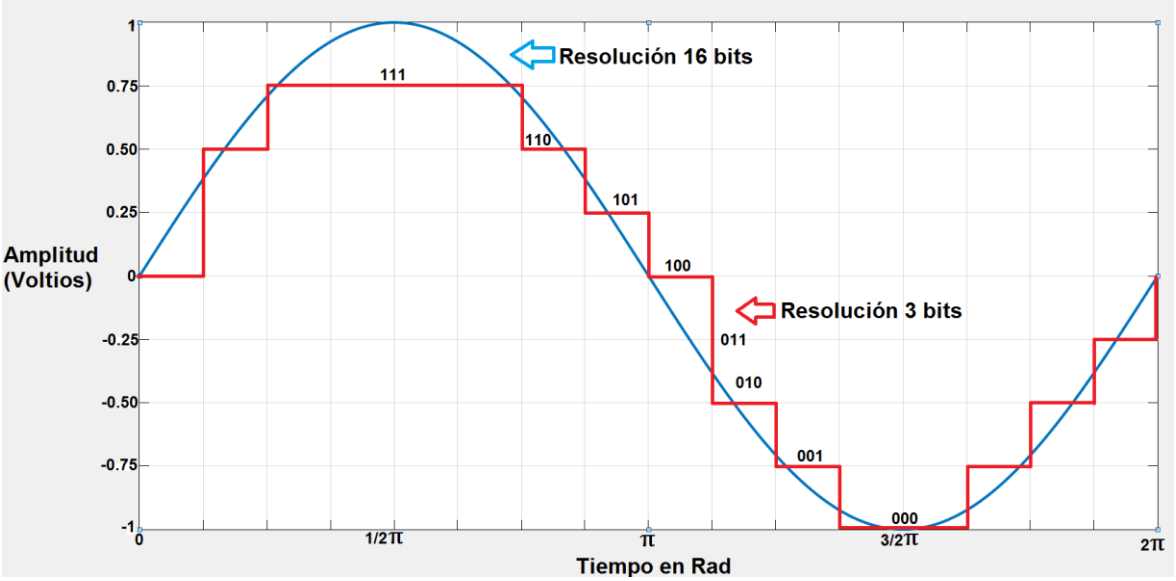

Figura 1. **Niveles binarios para resolución de 3 bits**

A mayor resolución binaria, es mejor la representación de la señal original. Teniendo como ejemplo una señal senoidal analógica que se quiere digitalizar, en la **Figura 1** se observa que para una digitalización de resolución de 3 bit existen 8 niveles de representación de voltaje, lo que genera una señal escalonada y para una digitalización de resolución de 16 bit existen 65536 niveles lo que permite una representación más fiel de la señal original.

Rango:

Son los niveles de voltaje que puede representar una señal digital. Este parámetro está determinado según las características del circuito ADC o el DAC. Por ejemplo, para un rango de 0 a 10V y una resolución de 3 bit, cada nivel equivale a 1.25V. Mientras que, para el mismo caso, pero una resolución de 16 bit sería de 0.000152588V o 152.58µV.

Tasa de muestreo:

Es la frecuencia con la que se toma una muestra de la señal original. Este parámetro es el más importante para una correcta representación digital de la señal original porque de la frecuencia de muestreo depende la calidad de la señal digitalizada, a mayor frecuencia de muestreo la señal digital es más parecida a la real. Sin embargo, una señal digital discreta nunca es igual a la señal analógica que intenta representar, esto se debe a que en el proceso de digitalización se pierde información, pero con una correcta frecuencia de muestreo, como lo describe el teorema de Nyquist, se puede lograr una muy buena representación digital de la señal analógica. En la **Figura 2** se observa el proceso de digitalización.

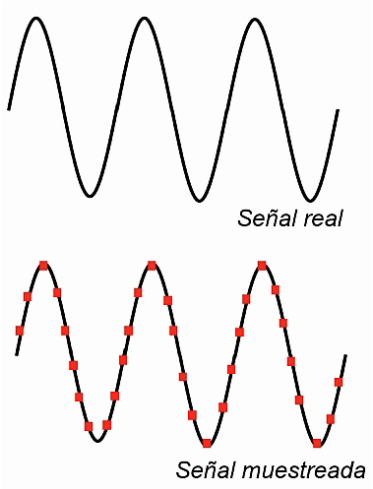

Figura 2. **Proceso de muestreo de una señal analógica. Tomada de** *(NI.com, 2016)*

Según el teorema de Nyquist, para replicar una señal analógica, la frecuencia de muestreo debe ser por lo menos igual o superior a dos veces la máxima frecuencia de la señal a muestrear. En la **Figura 3** se observa la reconstrucción de una señal analógica a diferentes frecuencias de muestreo. A pesar de que muestrear a dos veces la frecuencia máxima de la señal analógica es suficiente para replicar la señal, se observa que una frecuencia de muestreo de cinco o más veces la máxima + frecuencia, la señal que se obtiene es más similar a la original.

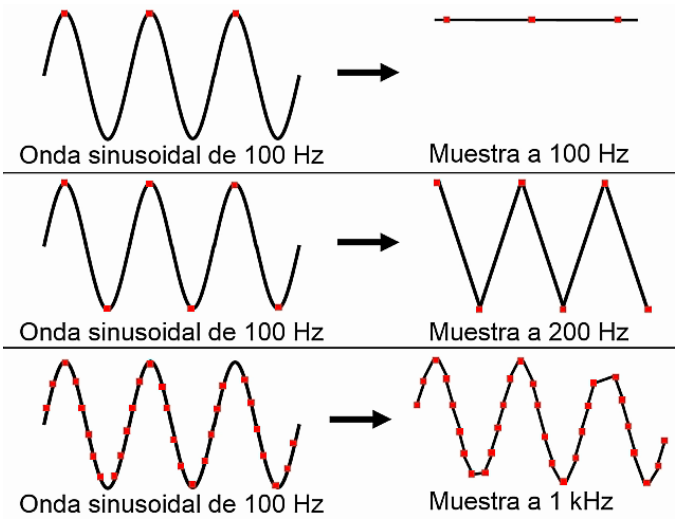

Figura 3. Señal senoidal de 100Hz muestreada a una frecuencia de 100Hz, 200Hz y 1KHz. **Tomada de** *(NI.com, 2016)*

Procesador de señales:

Son los dispositivos lógicos programables que permiten modificar y procesar señales muestreadas. En este caso se empleará la tarjeta ZedBoard con el procesador Zynq-7000 de Xilinx. Los sistemas de procesamiento de datos poseen un dispositivo programable y periféricos de entrada y salida que facilitan la interpretación y exportación de los datos.

PROCESAMIENTO DIGITAL DE SEÑALES (DSP)

Consiste en la manipulación matemática de una señal en el dominio del tiempo discreto para modificarla o mejorarla en algún aspecto. El procesamiento de las señales se hace mediante un sistema basado en un procesador que posee un juego de instrucciones, un hardware y un software optimizado para aplicaciones que requieran operaciones numéricas a muy alta velocidad tal como las características de la ZedBoard.

Se puede trabajar con señales analógicas, pero es un sistema digital, por lo tanto, necesitará un conversor analógico/digital a su entrada y digital/analógico en la salida. Como todo sistema basado en procesador programable necesita una memoria donde almacenar los datos con los que trabaja y el programa que ejecuta. En la ***Figura 4*** se observa el diagrama de un sistema de procesamiento digital de señales.

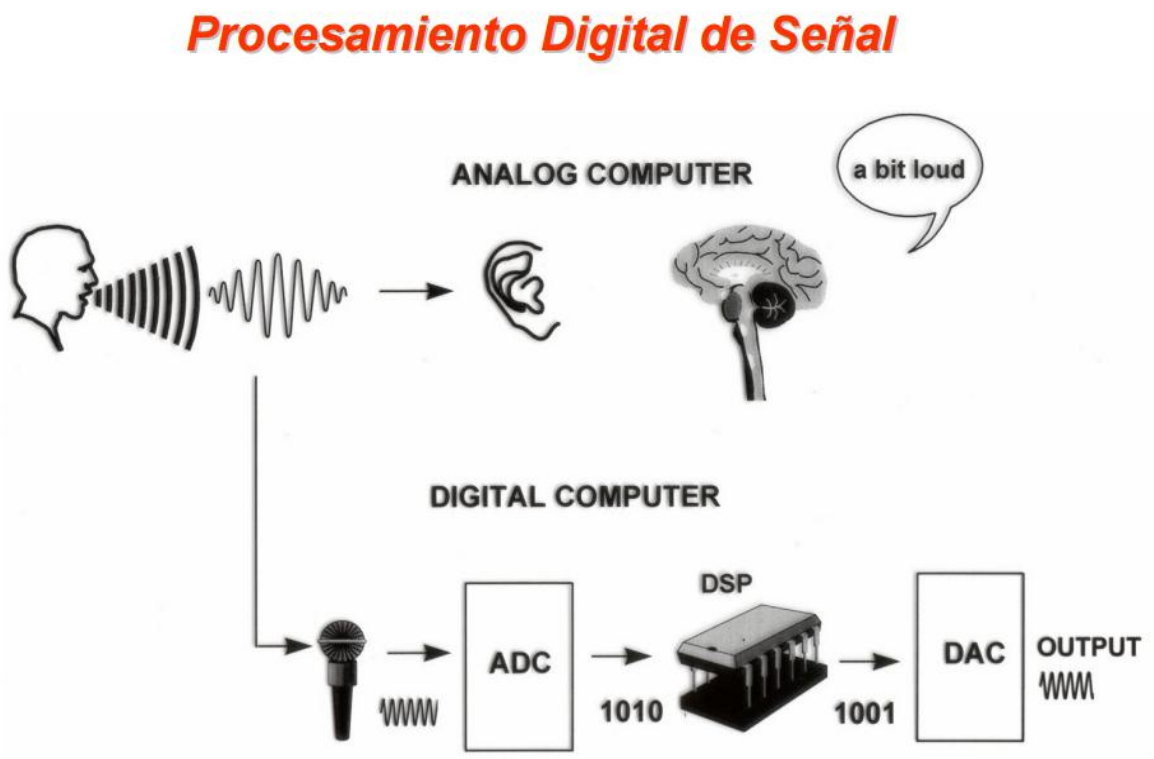

Figura 4. Esquema procesamiento digital de señales. Tomada de http://arantxa.ii.uam.es/~taao1/teoria/tema1/pdf/tema1.pdf

Ventajas del Procesamiento digital de señales:

Al trabajar con sistemas digitales y procesadores se obtienen las siguientes ventajas:

- El sistema se puede programar y modificar fácilmente dando un alto grado de flexibilidad en el diseño.
- Se puede mitigar el ruido, ya que la señal al estar digitalizada no sufre alteraciones por otros componentes o etapas del sistema.
- Un sistema digital funciona igual en toda su vida útil. A diferencia de los sistemas analógicos que los componentes activos y pasivos varían sus tolerancias con el paso del tiempo logrando así cambios en el comportamiento del diseño.
- Se puede realizar más fácilmente funciones especiales en la programación como operaciones matemáticas, funciones trigonométricas, et; que a comparación de un sistema analógico puede no existir un componente que haga la misma función.

Desventajas del procesamiento digital de señales:

También existen algunas desventajas en este tipo de procesamiento. Algunas de ellas son:

- Los sistemas analógicos son de menos consumo que los digitales.
- Al convertir una señal analógica a digital se pierde información. La señal digital no va ser igual a una analógica.
- Para señales analógicas de muy alta frecuencia se requieren conversores ADC de muy alta tasa de muestreo. Además, se requiere de un procesador de muy alta velocidad que pueda realizar el tratamiento de la señal de acuerdo a la tasa de muestreo a la cual se desea trabajar.
- El diseño es más complejo ya que se requiere de hardware y software para su implementación.

Aplicaciones del procesamiento digital de señales:

El procesamiento digital de señales ha incursionado en muchos campos de la ingeniería. Algunos ejemplos de aplicaciones son:

- Voz: Se usa para identificación y reconocimiento de vos.
- Imágenes: Mejorar el brillo, contraste o nitidez de la imagen.
- Domótica: Para automatización de las tareas del hogar.
- Medicina: Para interpretación de electrocardiogramas.
- Televisión: Televisión digital terrestre, IPTV, HDTV.
- Audio: Ecualización, compresión de la información (MP3), cancelación activa de ruido ambiente (inyectando ruido en contrafase).

FILTROS DIGITALES FIR

Una de las principales aplicaciones que se tiene en el procesamiento de señales son los filtros. A pesar de que existen los filtros analógicos, un filtro digital permite un fácil diseño y una calidad de respuesta a la frecuencia que en la actualidad está reemplazando los filtros analógicos en una gran variedad de aplicaciones industriales.

Un filtro digital es un proceso realizado por algoritmos a través del cual una señal digital se transforma en otra, es decir, afecta el contenido frecuencial de la señal de entrada y la transforma en otra dependiendo de la regla preestablecida en el filtro diseñado.

Un filtro FIR o filtro de respuesta finita al impulso unitario, es un tipo de filtro que como respuesta a un impulso tendrá un numero finito de términos no nulos. Los filtros FIR usan la muestra actual y muestras pasadas para obtener una salida, es decir, no es un filtro con realimentación. Por esta razón también se le llama filtros no recursivos.

La señal de entrada y salida de un filtro FIR está relacionado con la sumatoria de la convolución de dos vectores:

$$y[n] = \sum_{k=0}^{N-1} h[k]x[n-k] \qquad (1)$$

Donde:

h[k]= Coeficientes del filtro en función de frecuencia de muestreo, tipo de filtro y frecuencias de corte.

N = Orden del filtro.

X[n-k] = muestras del audio.

Aplicando la transformada Z a la ecuación (1):

$$H(z) = \sum_{k=0}^{N-1} h_k z^{-k} = h_0 + h_1 Z^{-1} + \cdots + h_{N-1} Z^{-(N-1)} \qquad (2)$$

La forma gráfica de ver un filtro FIR se presenta en la **Figura 5**.

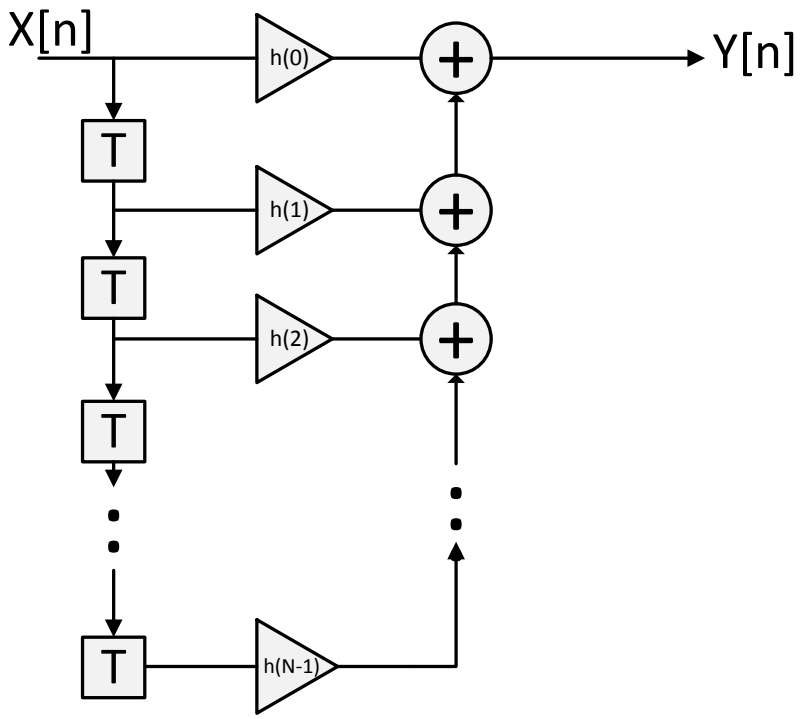

Figura 5. Esquema de un filtro FIR.

3. METODOLOGÍA

Para lograr el objetivo general de este trabajo, primero se debe entender y adecuar el ejemplo "ADC/DAC and Digital Audio Processing" de la página https://embeddedcentric.com/adc-dac-and-digital-audio-processing/ que brinda el insumo para la adquisición de audio. Segundo, Implementar un código en lenguaje C que permita almacenar los datos de audio adquiridos en la tarjeta SD de la ZedBoard. Tercero, desarrollar un algoritmo que realice o aplique un filtro digital FIR pasa alto al audio almacenado en la tarjeta SD y que el resultado sea almacenado nuevamente en la SD. Cuarto, implementar el mismo algoritmo de procesamiento de señal en MATLAB. Por último, graficar y comparar el audio filtrado por la ZedBoard y MATLAB en un PC por medio del mismo software MATLAB.

Todo el proceso de adquisición, discretización y procesamiento de señal lo realiza la tarjeta ZedBoard por medio de sus componentes, así:

- La señal de entrada de audio se genera por medio de un dispositivo móvil o PC. Estas es la señal que se desea procesar.
- El audio ingresa por la línea de entrada de la ZedBoard, que corresponde al Jack de color azul.
- La digitalización de la señal se hace por medio del chip ADAU1761, que es un chip integrado que posee dos conversores ADC con resolución de 24 bits cada uno y dos conversores DAC con resolución de 24 bits cada uno. Este chip integrado tiene la capacidad de trabajar a una frecuencia de muestreo desde 8KHz hasta 96KHz.
- El procesamiento de la señal se hace por medio de uno de los núcleos ARM Cortex-A9 de la ZedBoard que puede trabajar hasta una frecuencia de 667 MHz.

- La interface de tarjeta SD, se encarga de almacenar los datos procesados. La memoria SD es una memoria externa no volátil que se usa para almacenamiento de datos ya que posee una gran capacidad de almacenamiento.
- Los datos procesados se pueden escuchar por la línea de salida de la ZedBoard, que corresponde al Jack de color verde. Adicionalmente, si los datos se quieren interpretar en otro software y por medio de un PC, se hace uso de los datos almacenados en la memoria SD.

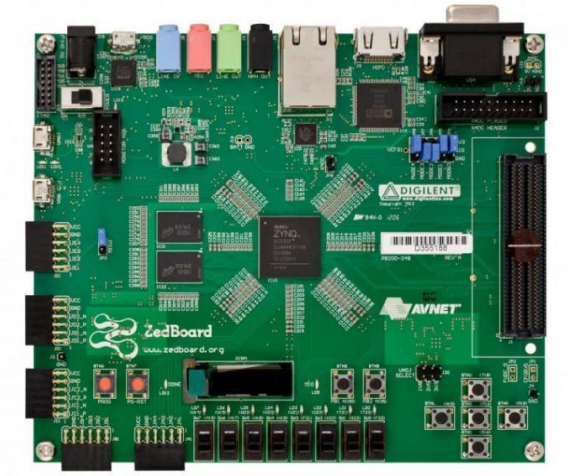

Figura 6. Tarjeta ZedBoard

Hardware necesario:

- Tarjeta de desarrollo ZedBoard Zynq-7000 ARM/FPGA SoC, usada para la adquisición, procesamiento y almacenamiento de datos como *Standalone*.
- Dispositivo móvil, usado como fuente de audio
- Tarjeta SD, usada para almacenamiento de datos
- Computador, usado para ejecutar programas necesarios para configurar la ZedBoard.

Software necesario:

- MATLAB versión R2015a, este software se emplea para crear scripts que permiten comparar los resultados obtenidos a través de la tarjeta ZedBoard y los resultados que brinda el mismo software MATLAB.

- Vivada versión 2015.3, este software es empleado para configurar el hardware embebido en la tarjeta ZedBoard tal como, conversores ADC Y DAC, el procesador, velocidad de procesamiento, la tarjeta SD y los drivers necesarios para que la tarjeta ZedBoard funcione correctamente.
- SDK versión 2015.3, este software es empleado para crear la aplicación de procesamiento digital deseado a través de los componentes de hardware de la tarjeta ZedBoard.

3.1. ADQUISICIÓN DE AUDIO A TRAVÉS DE LA TARJETA ZEDBOARD

Para realizar la adquisición de audio se debe seguir los pasos descritos a continuación:

a. Crear un proyecto nuevo en Vivado:

- Se abre el software Vivado y en la pantalla de inicio se da clic en "Create New Project"

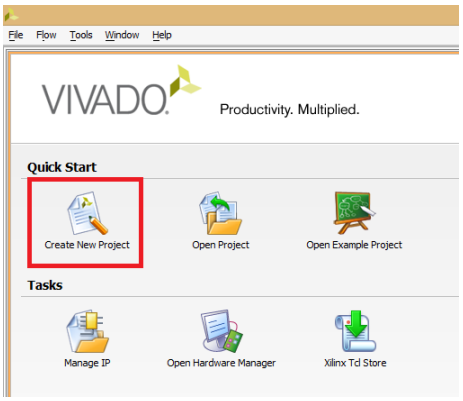

Figura 7. Vista de inicio del software Vivado.

- En la ventana emergente que se abre dar clic en "Next". Luego se da un nombre y ruta de ubicación al proyecto como se muestra en la **Figura 8** y clic en "Next".

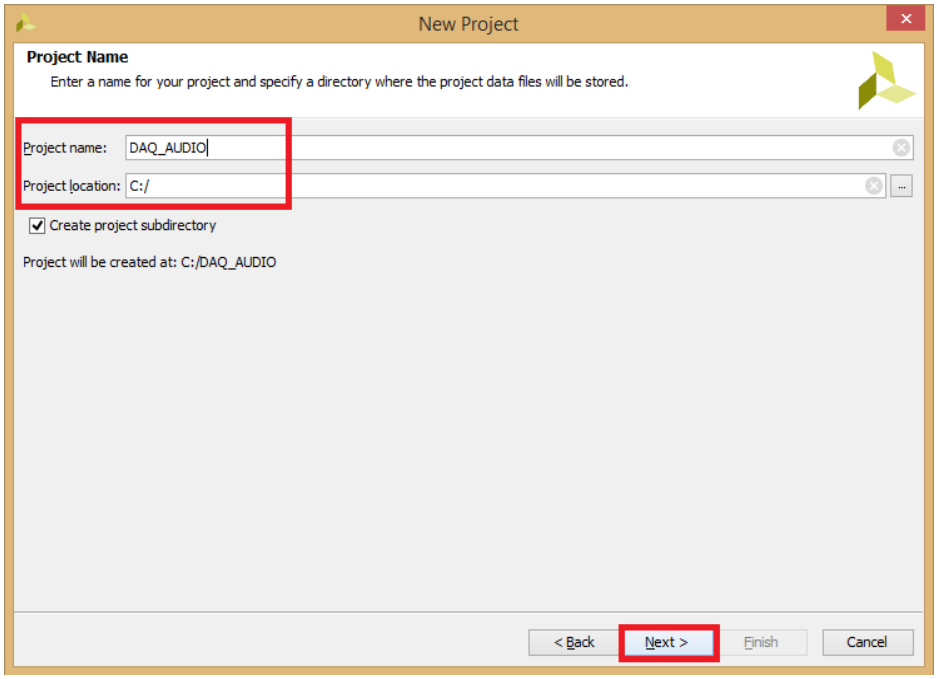

Figura 8. Ventana de creación de un proyecto en Vivado página 1.

- En la ventana siguiente se deja por defecto y se da clic en “Next”.

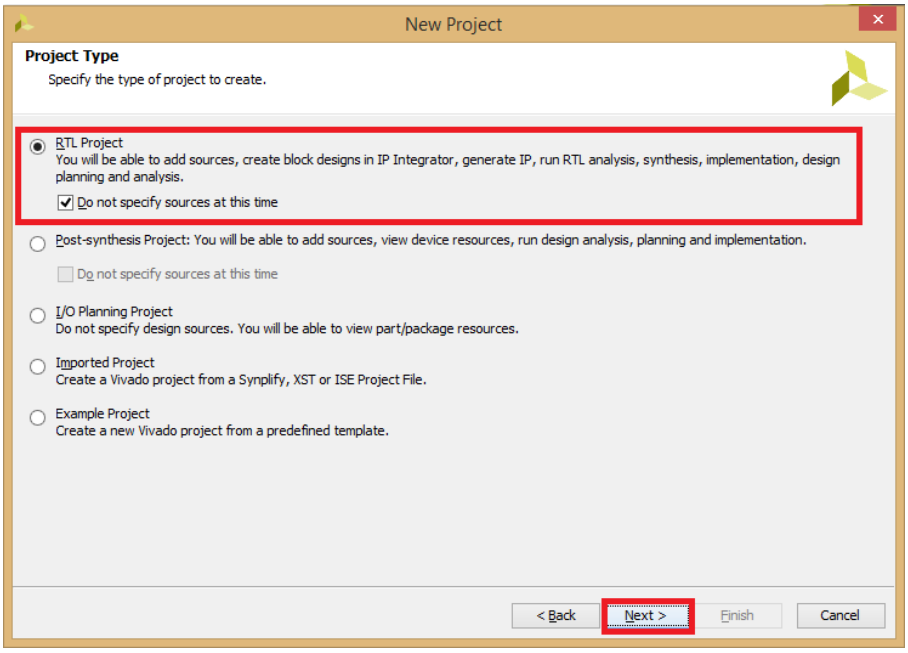

Figura 9. Ventana de creación de un proyecto en Vivado página 2.

- En la ventana siguiente se da clic en “Boards” y se selecciona “ZedBoard Zynq…….”, luego clic en “Next” y finalmente en “Finish”. De esta manera queda el proyecto creado.

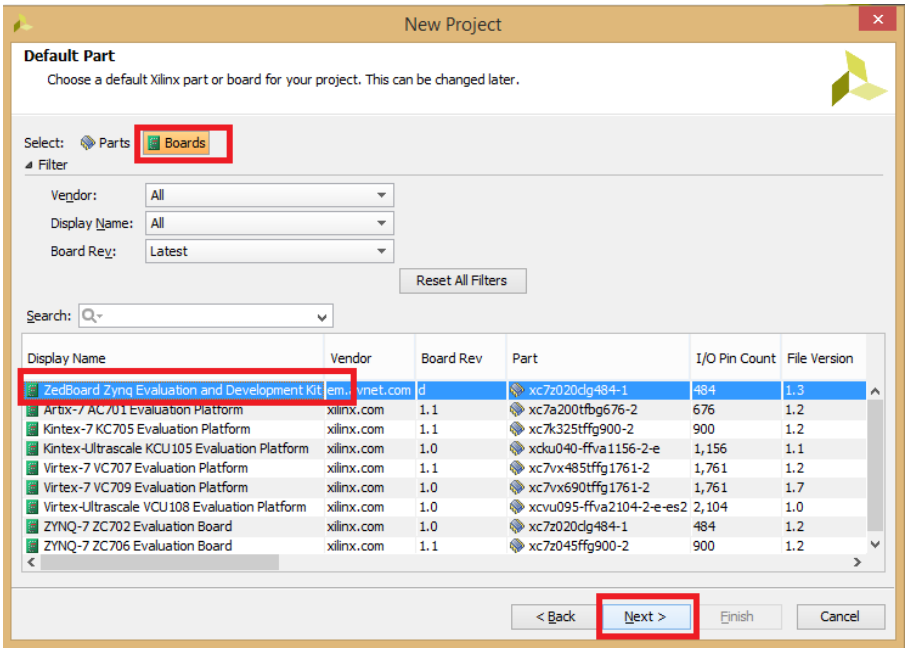

Figura 10. Ventana de creación de un proyecto en Vivado página 3.

b. Crear el bloque de diseño:

- Estando en la pantalla principal del proyecto creado en vivado ubicar el menú “Flow Navigator”, luego el submenú “IP Integrador” y se da clic en “Create Block Design”.

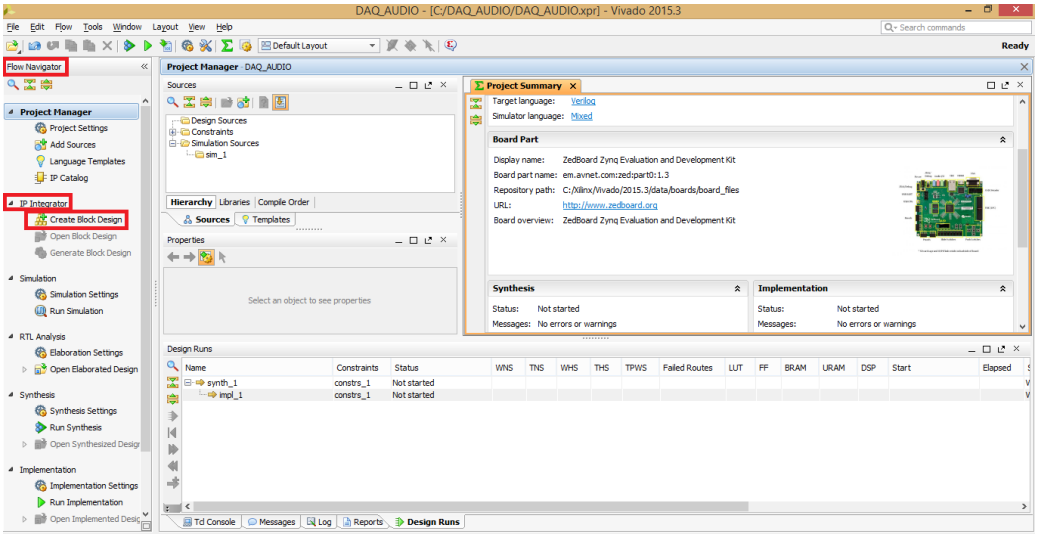

Figura 11. Ventana principal de un proyecto creado en Vivado.

- En la ventana que se abre se le da nombre al diseño y luego clic en “OK”. En este caso el nombre del diseño es “DAQ_AUDIO”.

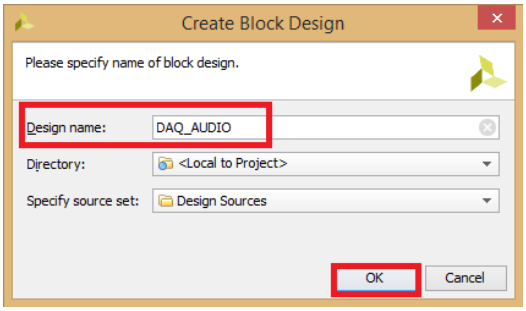

Figura 12. Ventana de creación del bloque de diseño.

- Agregar el “IP” que corresponde al sistema de procesamiento “ZYNQ7”. Para esto hacer clic en “add ip”. En la ventana que se abre escribir “Zynq” y seleccionar la opción “ZYNQ7 Proccessing System” con doble clic.

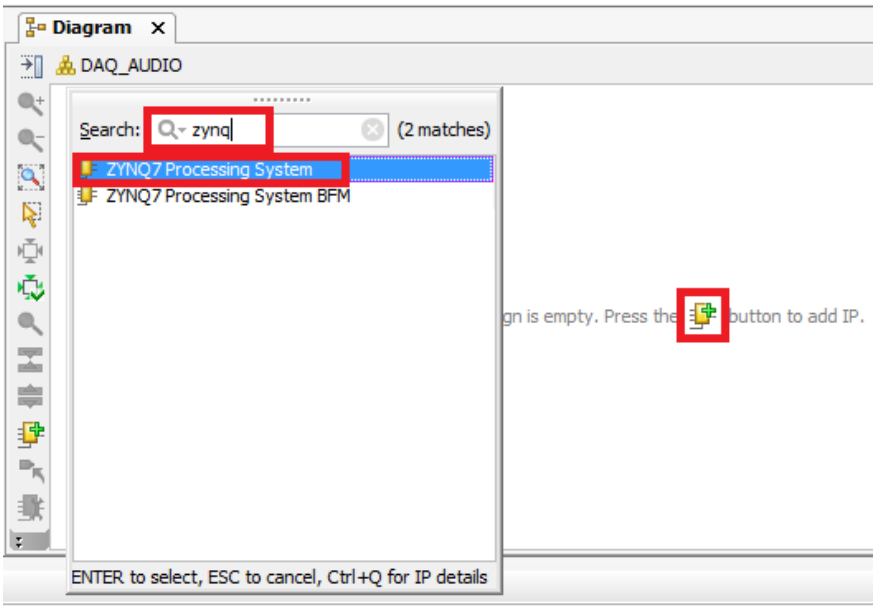

Figura 13. Ventana para agregar bloques IP al bloque de diseño.

- Se debe agregar el bloque llamado “ZYNQ7 proccesing system”.

Figura 14. Bloque IP ZYNQ7

- Hacer clic en “Run Block Automation”.

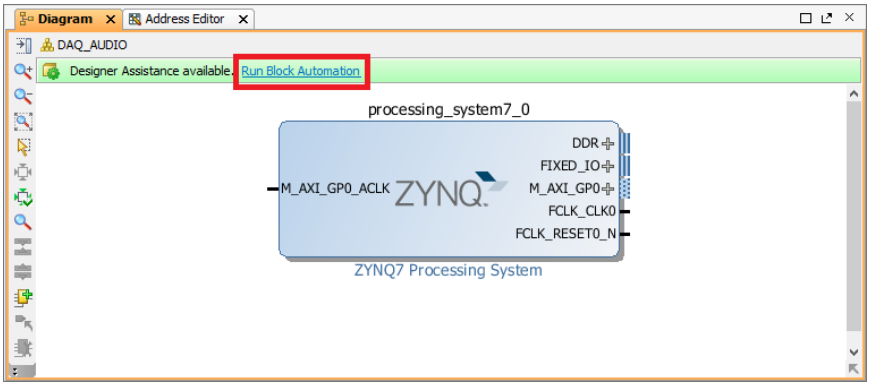

Figura 15. Run Block Automation.

- En la ventana que se abre, dejar las configuraciones como se muestran en la **Figura 16** y hacer clic en “OK”.

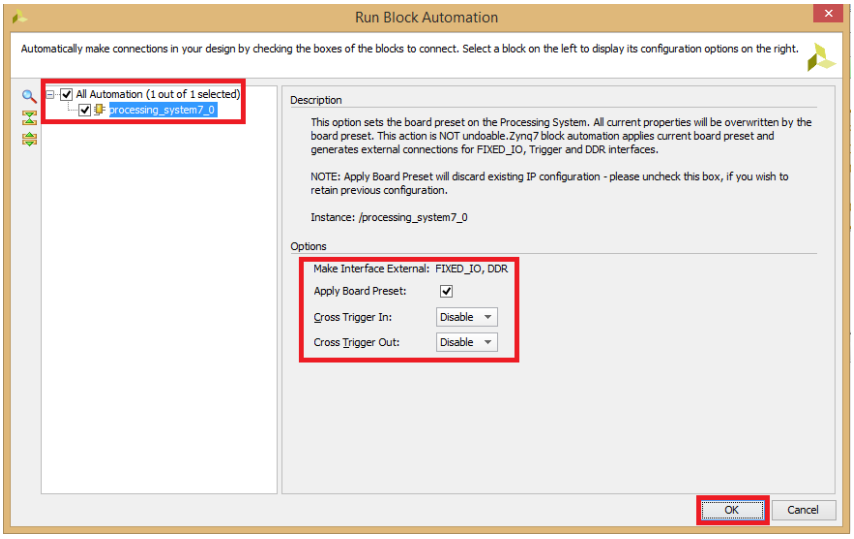

Figura 16. Ventana de configuración del Run Block Automation.

c. Crear la ruta “C:\Ip_cores” en el computador. Luego guardar las carpetas “Xilinx_com_hls_nco_1_0” y “zed_audio_ctrl”, que se dejan incluidas en este trabajo

d. Se debe configurar el bloque “ZYNQ7 proccessing system” de la siguiente manera:

- Dar doble clic al bloque “ZYNQ7 proccessing system” para ingresar en su configuración. Hacer clic en “MIO Configuration”, desplegar las opciones de “I/O Peripherals” y seleccionar las opciones como se muestra en la **Figura 17**.

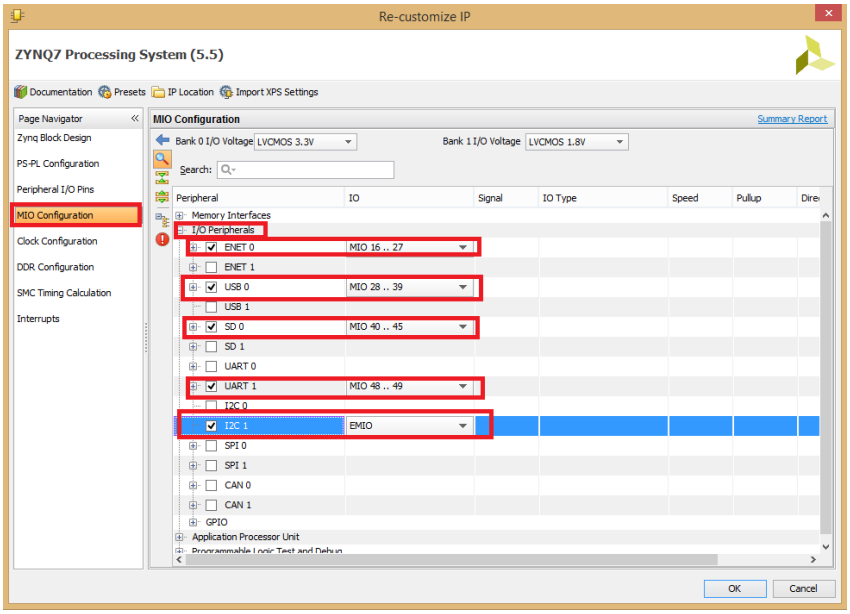

Figura 17. Configuración de periféricos de entrada y salida para adquisición de audio.

- Hacer clic en “Clock Configuration”, desplegar las opciones de “PL Fabric Clocks”, seleccionar las opciones mostradas como se muestra en la **Figura 18** y hacer clic en “OK”.

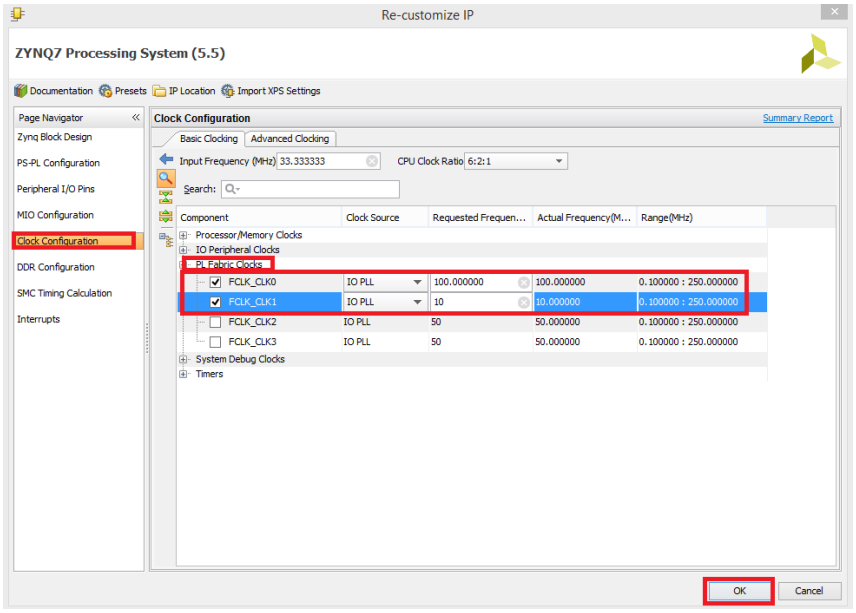

Figura 18. Configuración de los relojes del procesador.

- El bloque “ZYNQ7 proccessing system” se modifica. Hacer clic derecho en el pin “IIC_1” y seleccionar “Make External”. Hacer lo mismo con el pin “FCLK_CLK1”. El bloque queda como se muestra en la **Figura 19**.

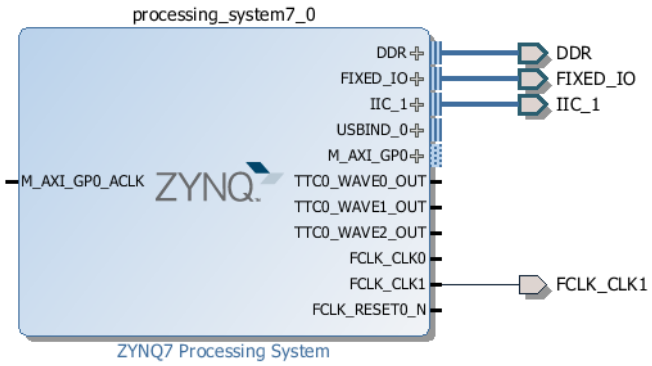

Figura 19. Bloque ZYNQ7 configurado.

e. Agregar los archivos de las carpetas que se guardaron previamente en la ruta “C:\Ip_cores” de la siguiente manera:

- Ubicar la ventana “Flow Navigator” y seleccionar “Project Setting” en el menú “Project Manager”.

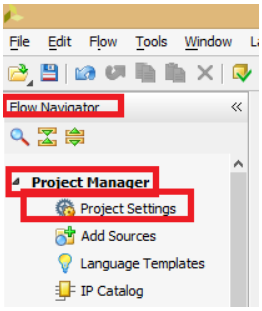

Figura 20. Opción Project Settings.

- En la ventana que se abre hacer clic sobre “IP”, ubicar y hacer clic en la pestaña “Repository Manager” y hacer clic en el botón “Add Repository” identificado con el símbolo “+” verde.

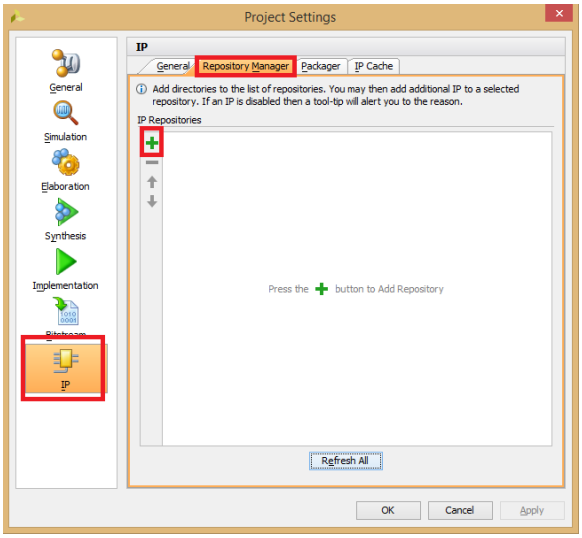

Figura 21. Ventana para agregar nuevos bloques IP.

- En la ventana que se abre ubicar la ruta que se creó anteriormente “C:\Ip_cores” y hacer clic en “Select”.

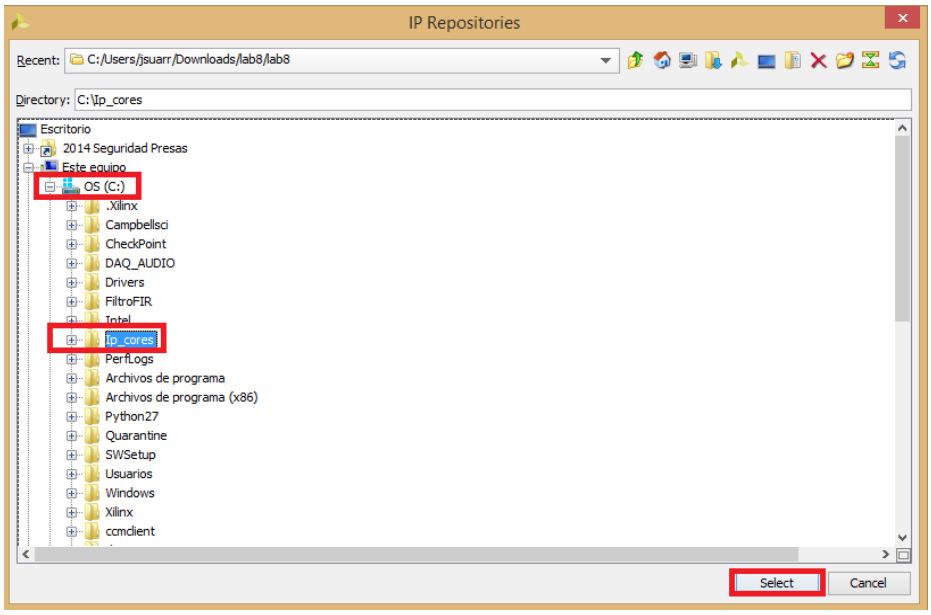

Figura 22. Ubicación de la carpeta Ip_cores.

- Hacer clic en “Apply” y después en “OK”. De esta manera, quedan agregados al Vivado los dos bloques que contienen la carpeta Ip_cores.

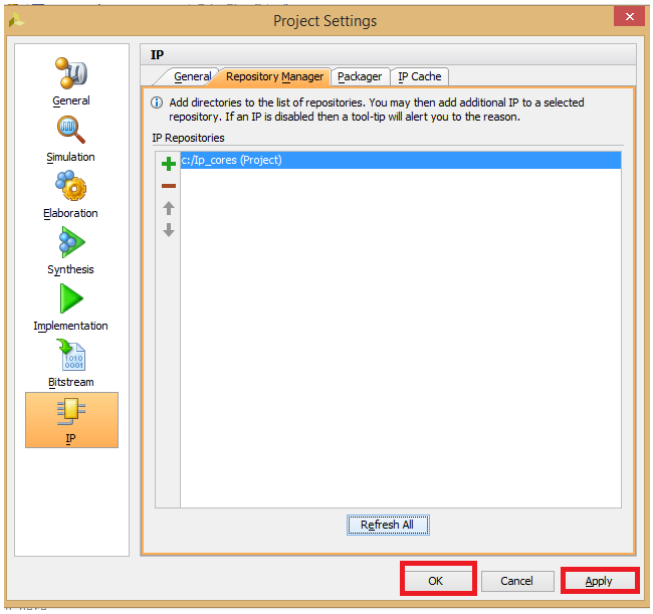

Figura 23. Ventana para confirmar la creación de bloques IP.

f. Agregar y cablear los bloques IP de la carpeta Ip_cores de la siguiente manera:

- Ubicar la ventana “Diagram” del diseño de bloques y hacer clic en “Add IP”. En la ventana que se abre escribir “nco” y dar doble clic para agregarlo.

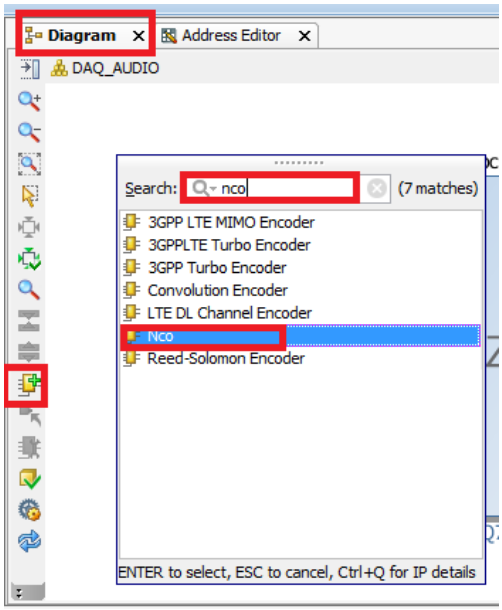

Figura 24. Ventana para agregar el bloque IP NCO al diseño.

- Hacer clic en “Run Connection Automation” y en la ventana que se abre dejar todo por defecto y hacer clic en “OK”.

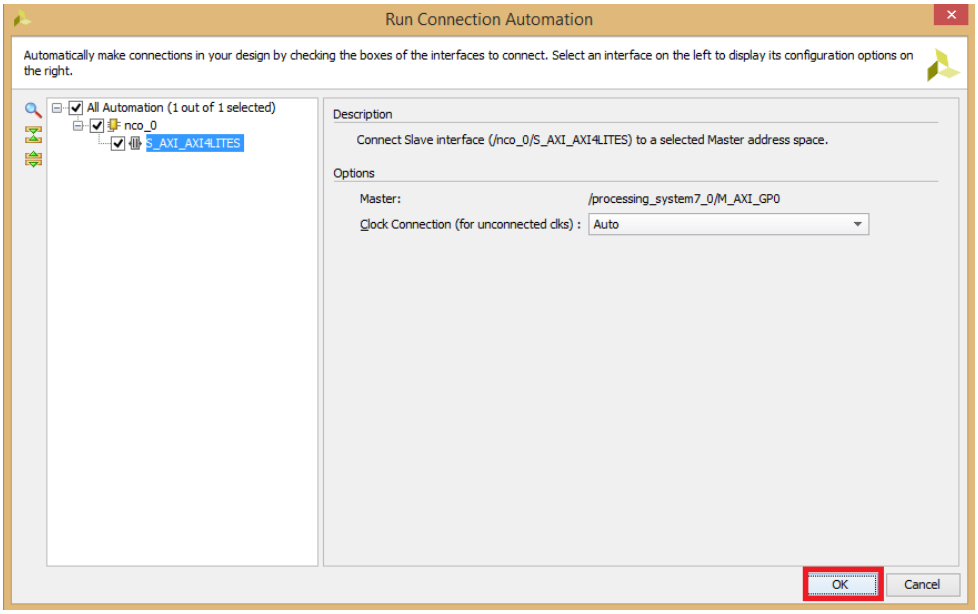

Figura 25. Ventana de configuración de Run Connection Automation.

- Repetir los pasos anteriores para agregar el bloque “zed_audio_ctrl”.
- Con los bloques agregados al diseño, se debe hacer clic derecho sobre el área de trabajo del diagrama de bloques y hacer clic en “Regenerate Layout”.

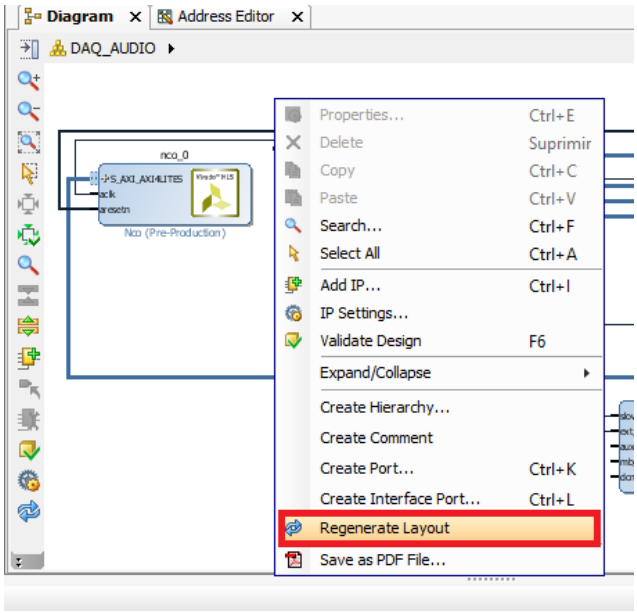

Figura 26. Ventana para regenerar las conexiones.

- El diagrama debe quedar como se muestra en la **Figura 27** , de lo contrario revisar los pasos anteriores hasta conseguir el diagrama correcto.

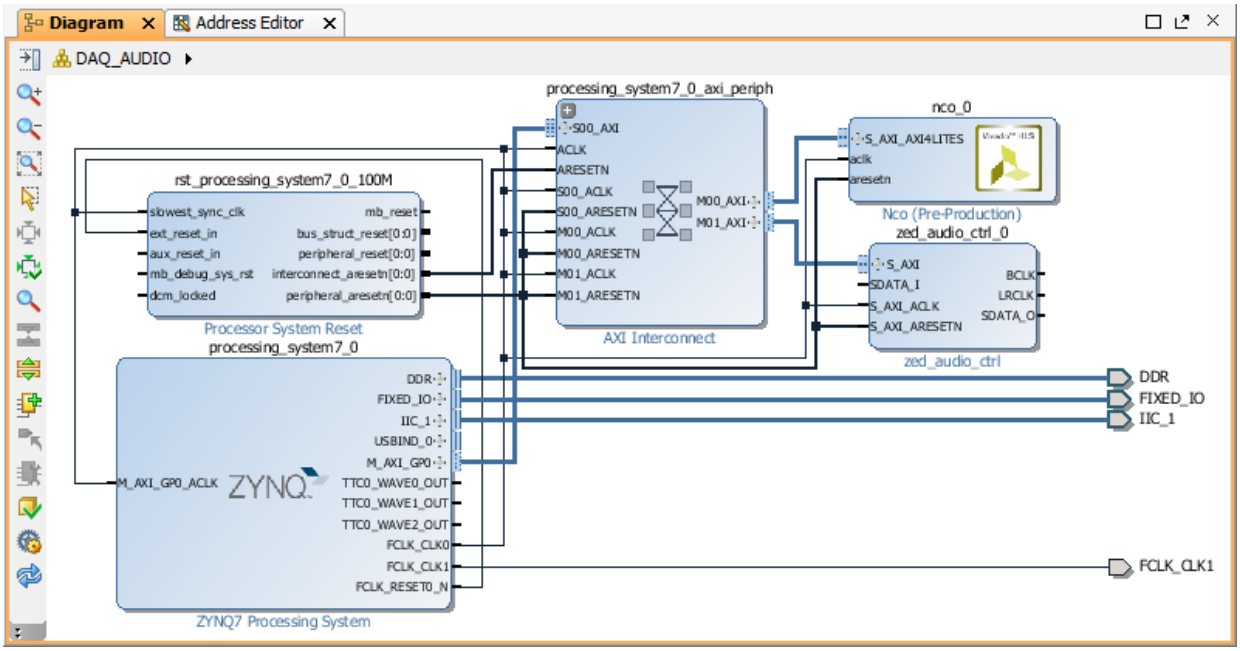

Figura 27. Diagrama de bloques del diseño.

- Hacer clic derecho en el pin “BCLK” del bloque “zed_audio_ctrl_0” y seleccionar “Make external”. Repetir lo mismo para los pines “LRCLK”, “SDATA_0” y “SDATA_1”. El diagrama debe quedar igual a como se muestra en la **Figura 28**.

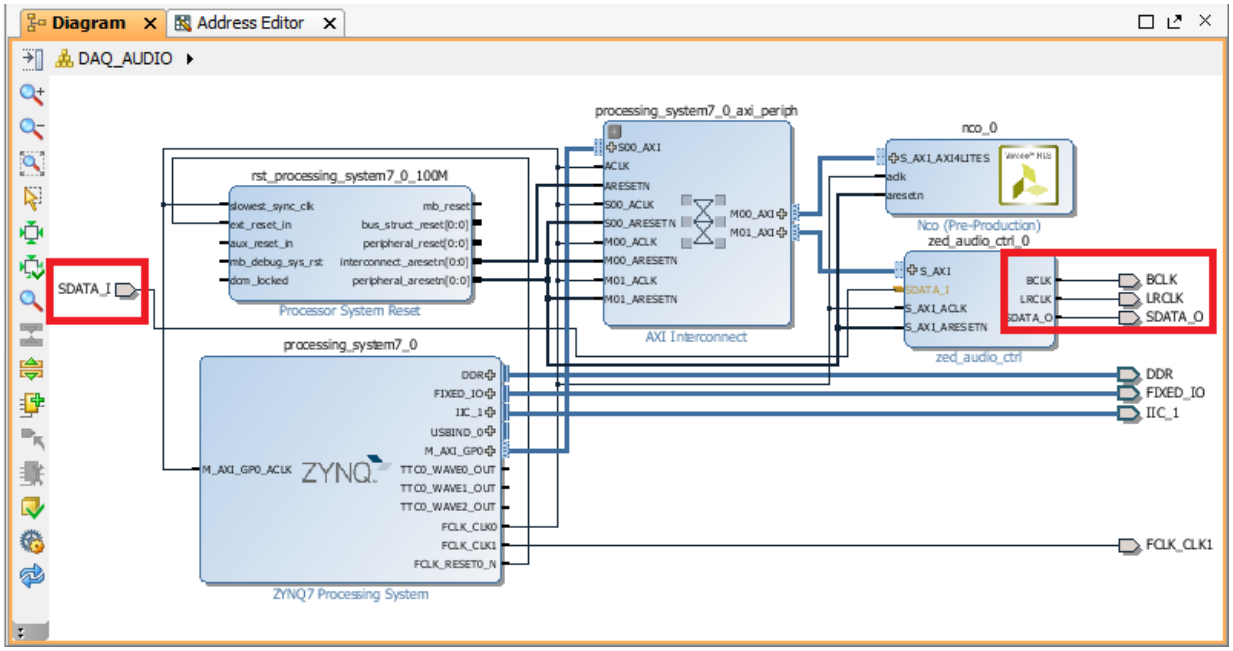

Figura 28. Conexiones externas del bloque zed_audio_ctrl_0.

g. Agregar el bloque IP que controla los interruptores y leds de la tarjeta ZedBoard de la siguiente manera:

- Ubicar la ventana “Diagram” del diseño de bloques y hacer clic en “Add IP”. En la ventana que se abre escribir “gpio” y dar doble clic en “AXI GPIO” para agregarlo.

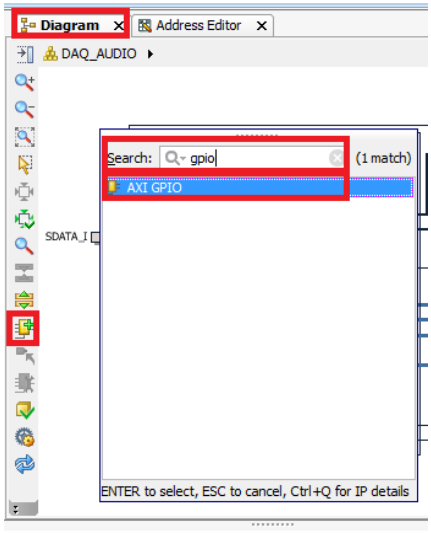

Figura 29. Ventana para agregar el bloque IP AXI GPIO.

- Hacer doble clic sobre el bloque que se generó para acceder a su configuración. Dejar los parámetros como se muestra en la **Figura 30** y hacer clic en “OK”.

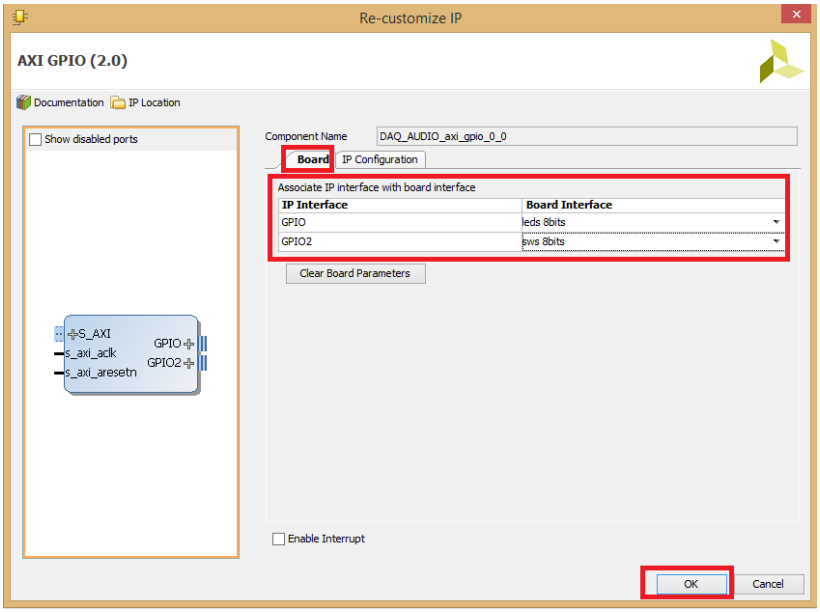

Figura 30. Configuración del GPIO.

- Para conectar este bloque hacer clic en “Run Connection Automation”. Seleccionar todas las opciones y hacer clic en “OK”. Como se muestra en la **Figura 31**.

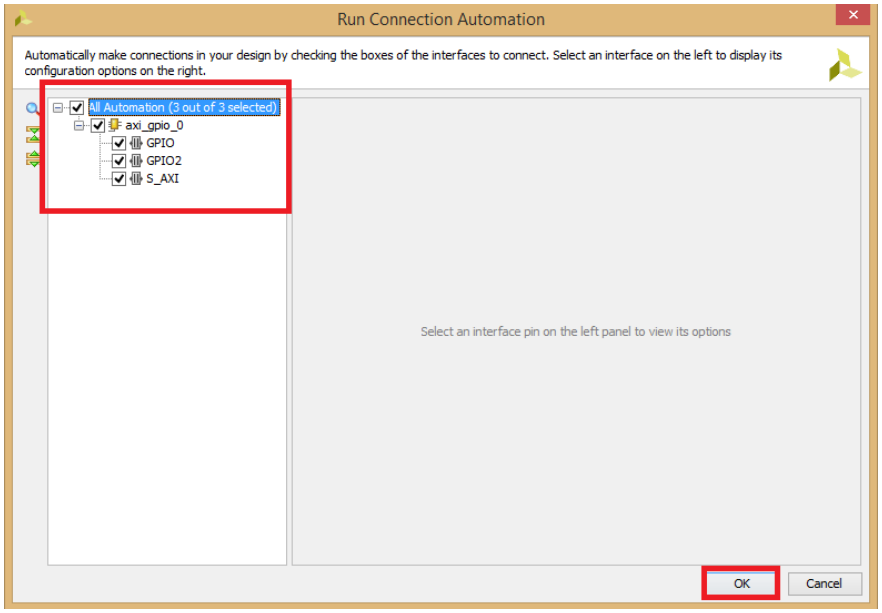

Figura 31. Configuración de Run Connection Automation.

h. Asignar los valores para los bits menos significativos de la dirección del I2C ADAU1761 de la siguiente manera:

- Hacer clic derecho sobre el área de trabajo del diagrama de bloques y seleccionar “Create Port”.

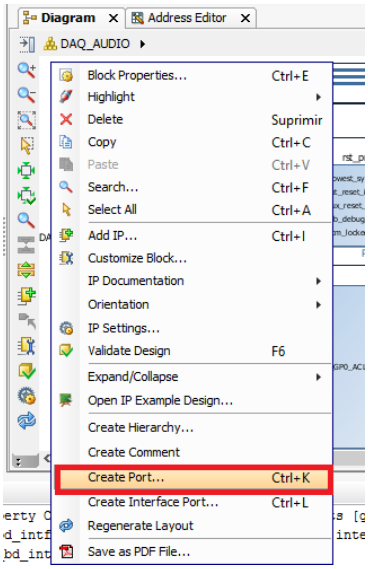

Figura 32. Ventana para creación de puerto.

- En la ventana que se despliega, configurarla como se muestra en la **Figura 33**.

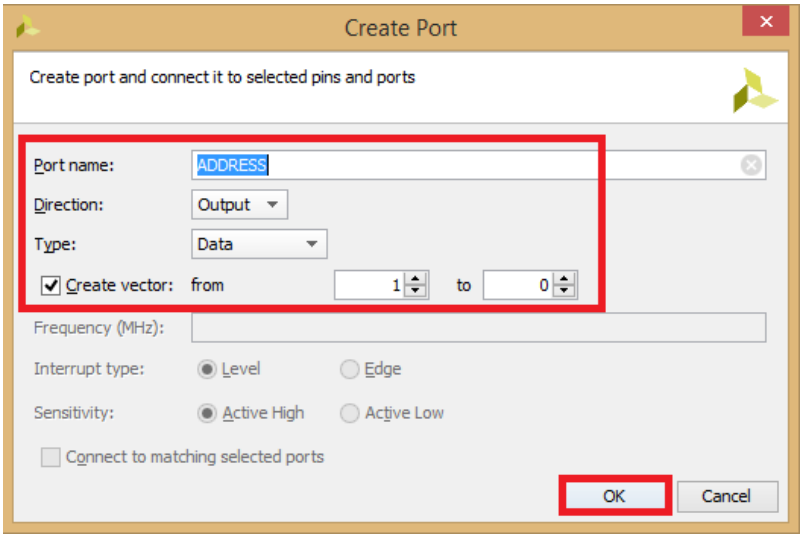

Figura 33. Ventana de configuración de puerto.

- Ubicar la ventana "Diagram" del diseño de bloques y hacer clic en "Add IP". En la ventana que se abre escribir "cons" y dar doble clic sobre "Constant" para agregarlo.

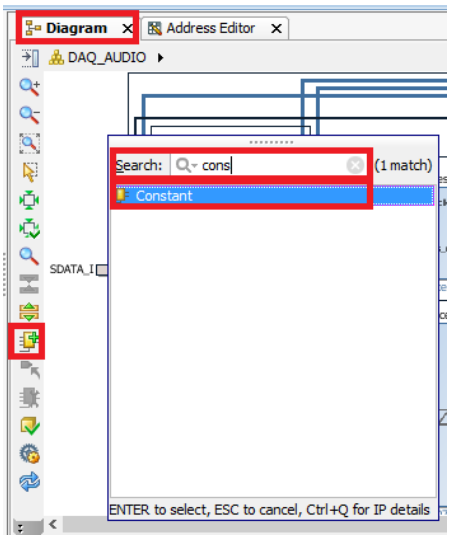

Figura 34.Ventana para agregar el bloque IP Constant.

- Dar doble clic en el bloque que se creó para acceder a su configuración, adecuar las opciones como se presentan en la **Figura 35** y hacer clic en "OK".

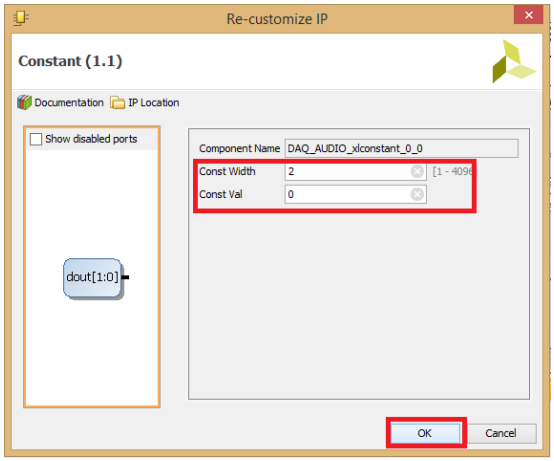

Figura 35. Configuración del bloque Constant

- Conectar el pin “dout[1:0]” del bloque “Constant” al pin “Address[1:0]” que fue el puerto creado. Para esto hacer clic sostenido en uno de los pines y arrastrarlo hasta el otro pin.

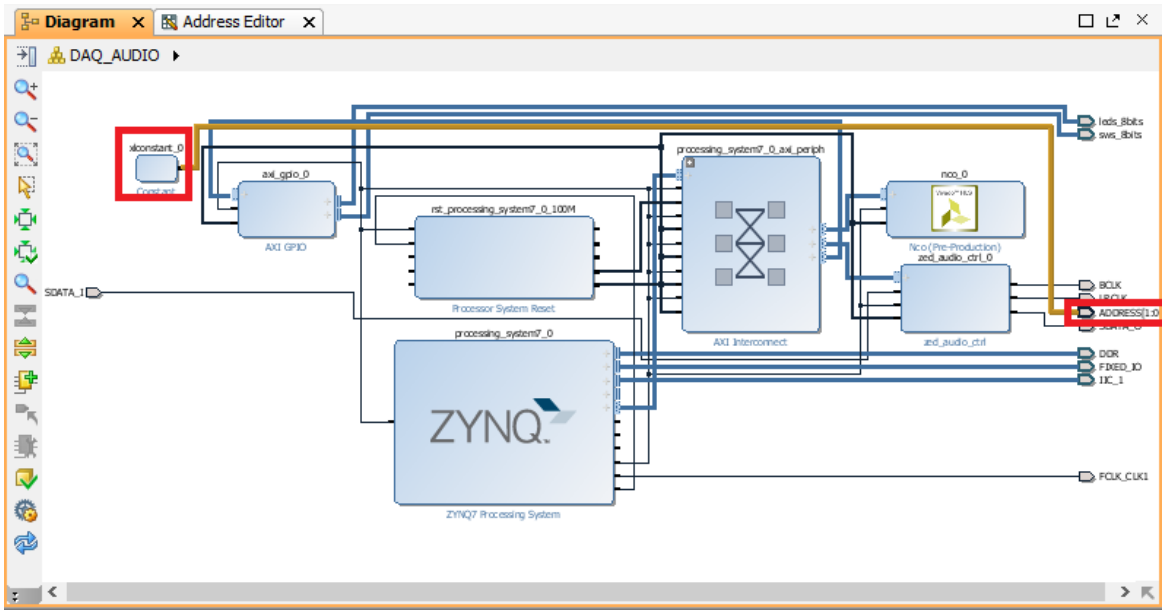

Figura 36. Conexión del puerto Address[1:0] al bloque Constant.

- Aplicar el “Regenerate Layout”. El diagrama debe quedar como se muestra en la **Figura 37**. De lo contrario se debe verificar los pasos anteriores hasta llegar al mismo diagrama.

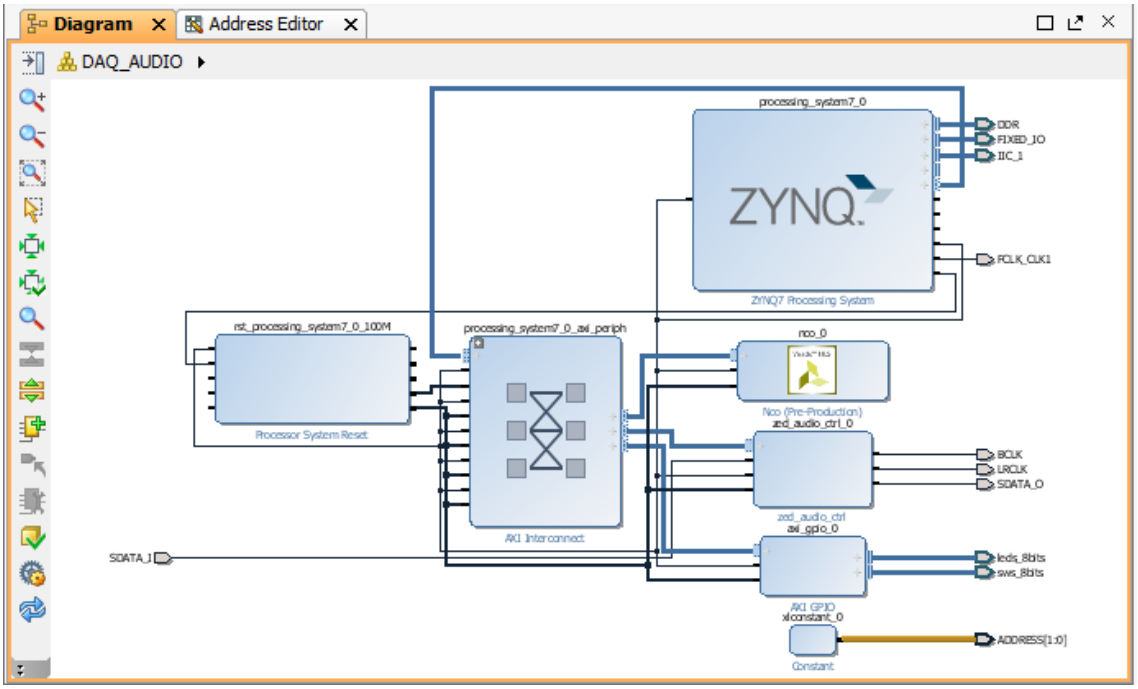

Figura 37. Diagrama de bloques final para adquisición de audio.

i. Agregar un archivo “Constraints” al proyecto de la siguiente manera:

- Ubicar la pestaña “Flow Navigator” en el menú “Project Manager” hacer clic en “Add Sources”.

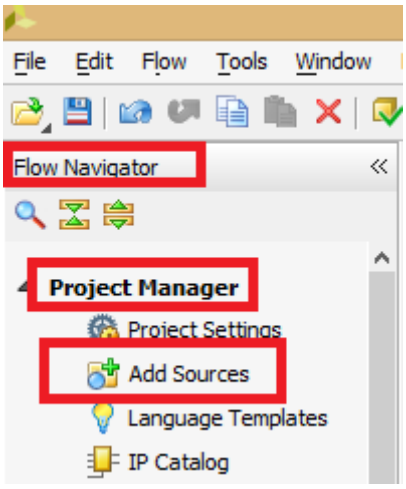

Figura 38. Ventana para agregar fuentes.

- En la ventana que se abre seleccionar “Add or Create Constraints” y hacer clic en “Next”.

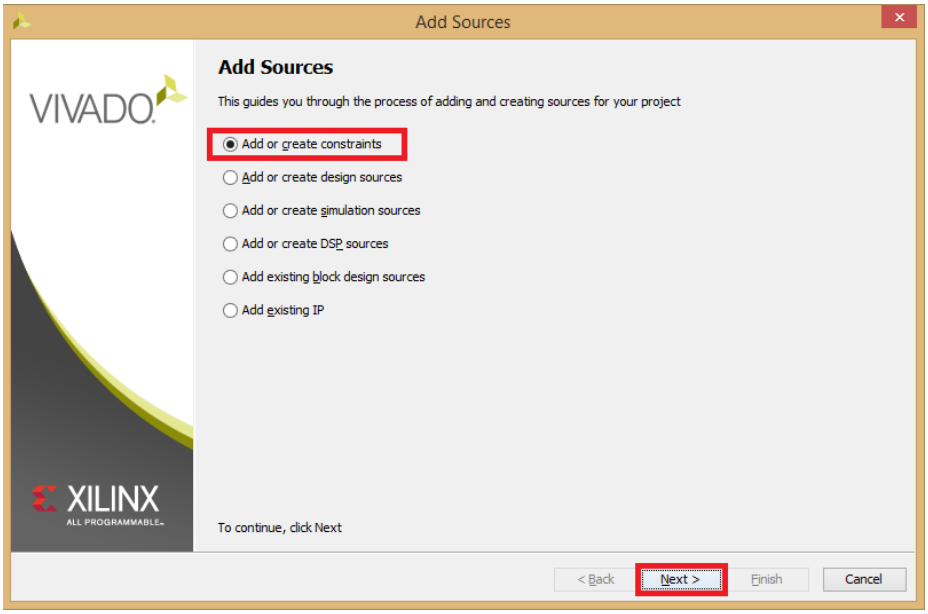

Figura 39. Ventana para creación de Constraints.

- En la ventana que se abre hacer clic en el icono "+" de color verde y seleccionar "Create File".

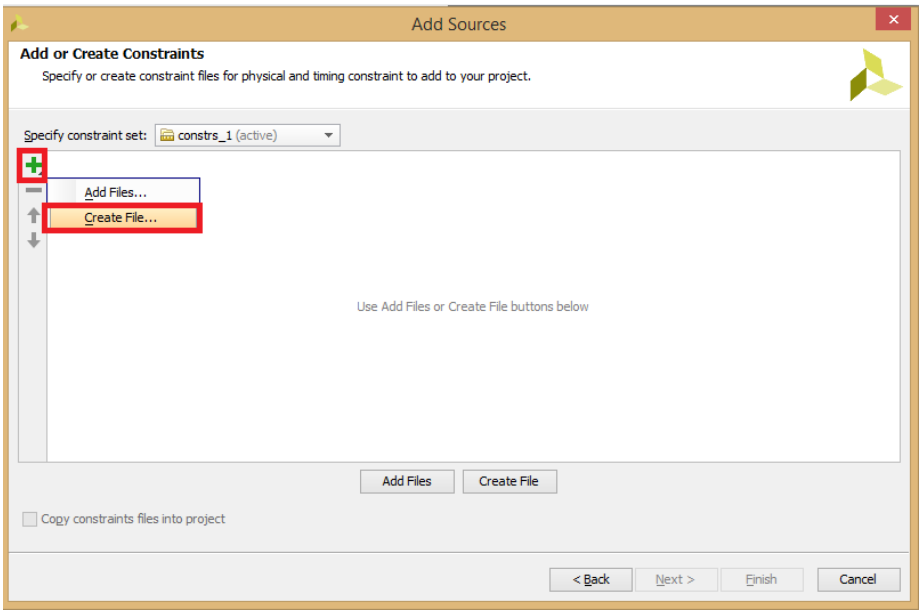

Figura 40. Ventana para agregar el archivo del Constraint.

- En la ventana que se abre escribir las opciones que se muestran en la **Figura 41**. Luego hacer clic en "OK" y en "Finish"

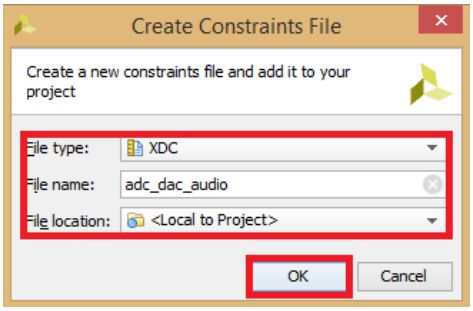

Figura 41. Atributos del archivo Constraint.

- Digerirse a la pestaña “Sources”, desplegar la carpeta “Constraints” y hacer doble clic en “adc_dac_audio.xdc”

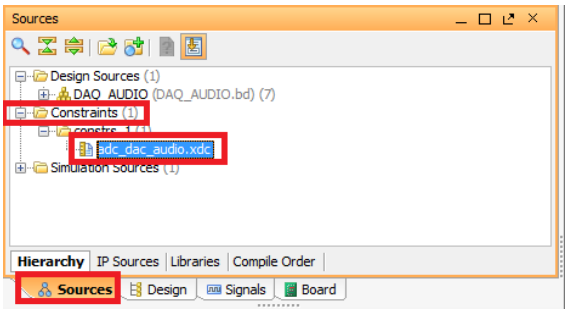

Figura 42. Apertura del archivo Constraints.

- En la ventana que se abre en blanco pegar el código que se muestra a continuación.

```
# ZedBoard Audio Codec Constraints
set_property PACKAGE_PIN AA6 [get_ports BCLK]
set_property IOSTANDARD LVCMOS33 [get_ports BCLK]

set_property PACKAGE_PIN Y6 [get_ports LRCLK]
set_property IOSTANDARD LVCMOS33 [get_ports LRCLK]

set_property PACKAGE_PIN AA7 [get_ports SDATA_I]
set_property IOSTANDARD LVCMOS33 [get_ports SDATA_I]

set_property PACKAGE_PIN Y8 [get_ports SDATA_O]
set_property IOSTANDARD LVCMOS33 [get_ports SDATA_O]

#MCLK
set_property PACKAGE_PIN AB2 [get_ports FCLK_CLK1]
set_property IOSTANDARD LVCMOS33 [get_ports FCLK_CLK1]

set_property PACKAGE_PIN AB4 [get_ports iic_1_scl_io]
```

```
set_property     IOSTANDARD     LVCMOS33     [get_ports
iic_1_scl_io]

set_property PACKAGE_PIN AB5 [get_ports iic_1_sda_io]
set_property     IOSTANDARD     LVCMOS33     [get_ports
iic_1_sda_io]

set_property PACKAGE_PIN AB1 [get_ports {ADDRESS[0]}]
set_property     IOSTANDARD     LVCMOS33     [get_ports
{ADDRESS[0]}]

set_property PACKAGE_PIN Y5 [get_ports {ADDRESS[1]}]
set_property     IOSTANDARD     LVCMOS33     [get_ports
{ADDRESS[1]}]
```

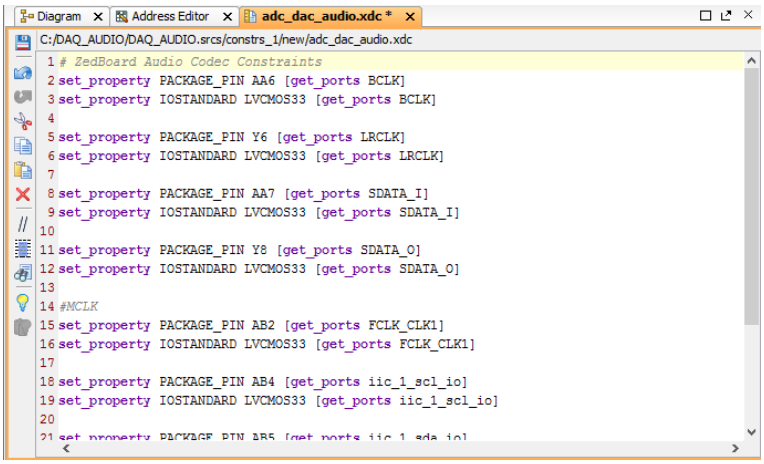

Figura 43. Ventana del código del Constraints.

- Hacer clic en “save” para salvar las asignaciones realizadas en el “Constraints”.

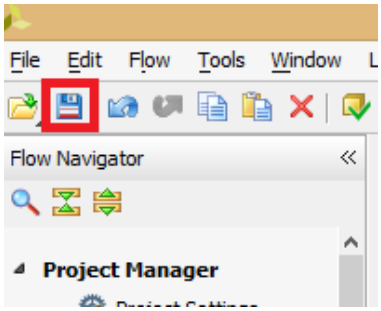

Figura 44. Icono para guardar los cambios realizados en el proyecto.

j. Creado el diseño del diagrama de bloques se debe generar el archivo de programación “Bitstream” para poder llevar el proyecto al software SDK. Para esto seguir los pasos descritos a continuación:

- Ubicar la pestaña “Sources” y hacer clic derecho en “DAQ_AUDIO” y seleccionar “Create HDL Wrapper”.

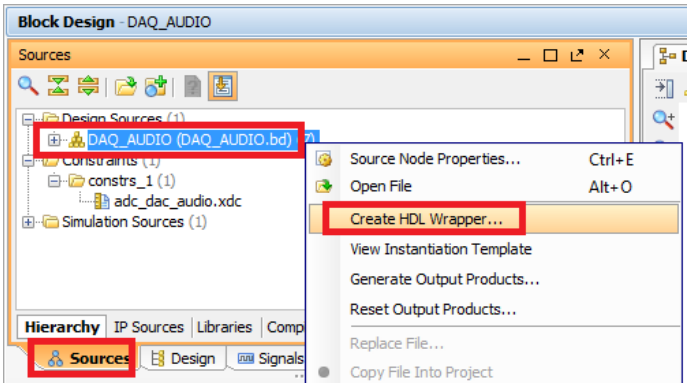

Figura 45. Opciones para crear el HDL Wrapper.

- En la ventana que se abre seleccionar “Let Vivado manage wrapper and auto-update” y clic en “OK”.

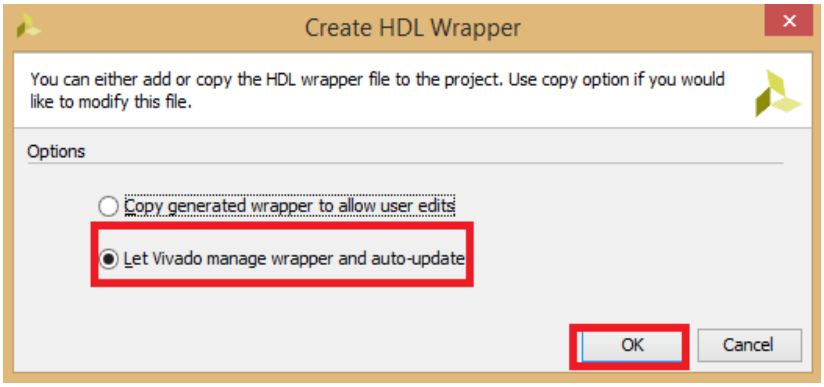

Figura 46. Opciones de creación del HDL Wrapper.

- Ubicar la pestaña “Flow Navigator” en el menú “Program and Debug” hacer clic en “Generate Bitstream”. En la ventana que se abre seleccionar “save”. El proceso de generar el bitstream puede demorar entre dos hasta diez minutos, todo depende de la capacidad de procesamiento del PC con el que se trabaje.

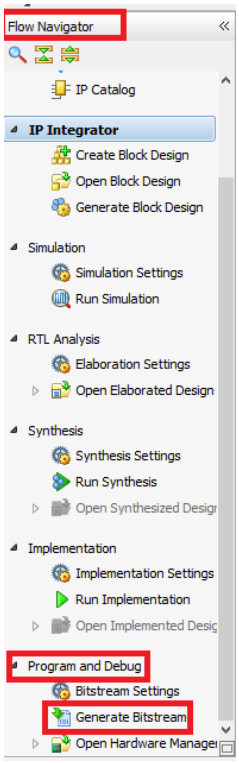

Figura 47. Secuencia para generar el archivo de programación Bitstream.

- Cuando se termina de crear el bitstream sale una ventana en la cual se debe seleccionar “View Reports” y clic en “OK”. Si el software pregunta por calificación queda a voluntad propia la opción a escoger.

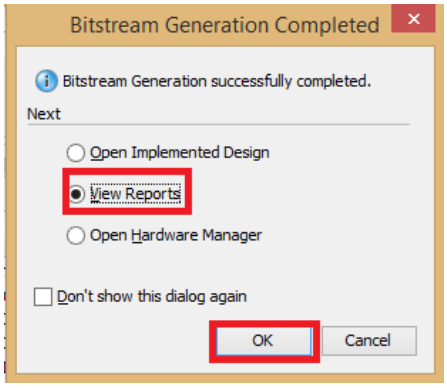

Figura 48. Ventana de confirmación exitosa de generación de Bitstream.

k. Con el bitstream generado se debe exportar el hardware para llevarlo al software SDK. Para esto seguir los pasos descritos a continuación:

- Hacer clic en “File”, luego en “Export” y hacer clic en “Export Hardware”

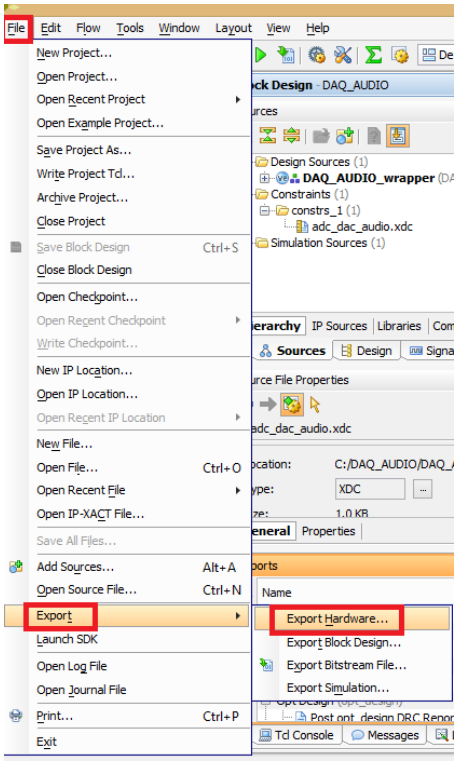

Figura 49. Secuencia para exportar el Hardware al SDK.

- En la ventana que se abre activar la casilla “Include bitstream” y clic en “OK”.

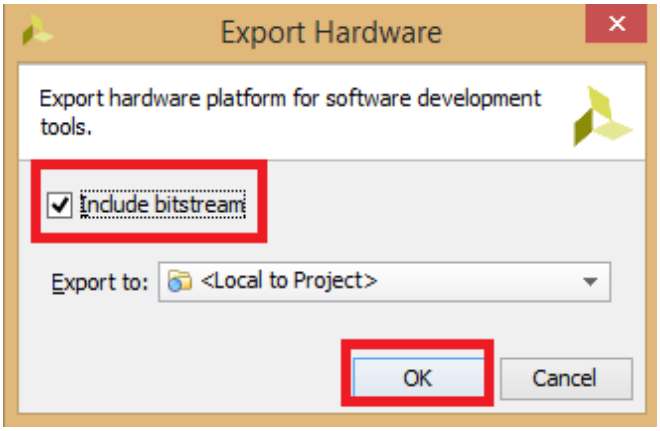

Figura 50. Ventana para exportar el Hardware.

- Desplegar la pestaña “File” y hacer clic en “Launch SDK”. En la ventana que se abre hacer clic en “OK”.

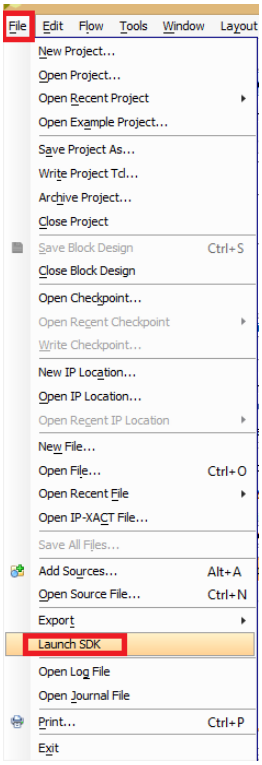

Figura 51. Secuencia para enviar el proyecto al SDK.

l. Con en el software “SDK” abierto, se debe crear una nueva aplicación de la siguiente manera:

- Desplegar el menú “File”, luego el menú “New” y hacer clic en “Application Project”.

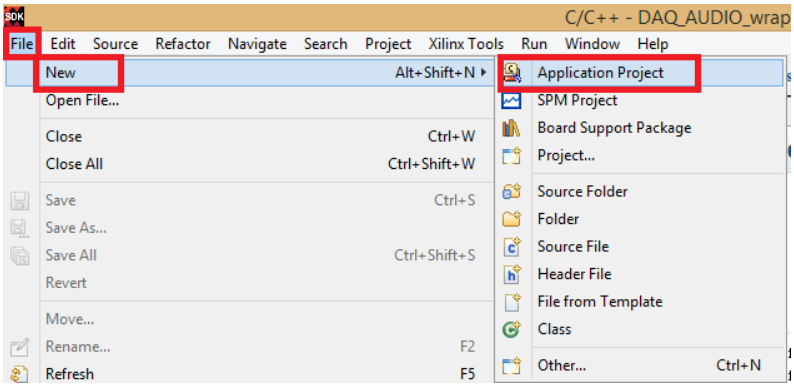

Figura 52. Secuencia para crear una nueva aplicación.

- En la ventana que se abre darle nombre al proyecto, verificar que las opciones marcadas sean las correctas como se muestra en la **Figura 53** y hacer clic en “Next”.

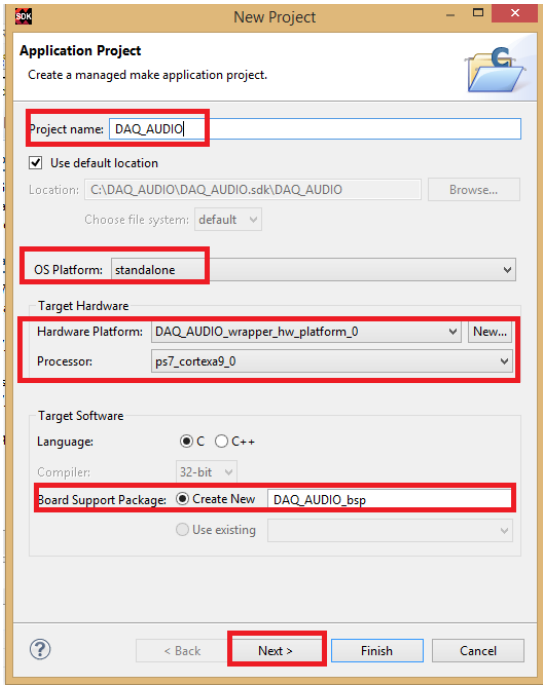

Figura 53. Opciones para creación de nueva aplicación.

- En la ventana que se muestra en la **Figura 54** seleccionar “Empty Application” y clic en “Finish”.

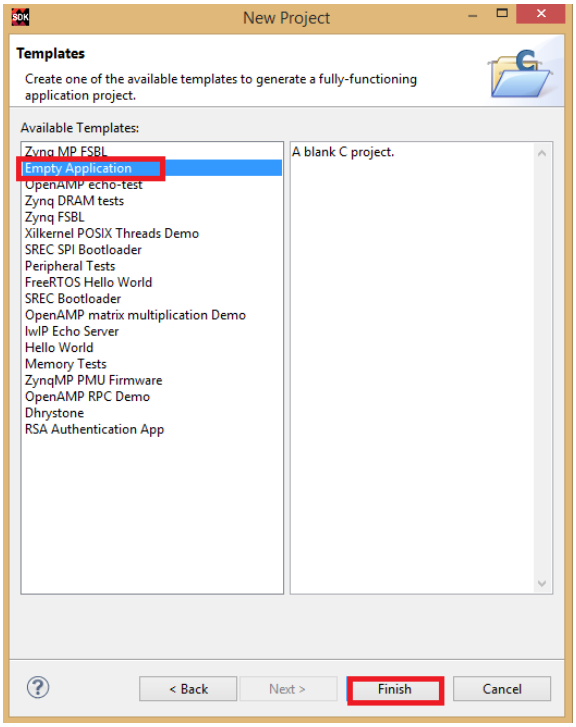

Figura 54. Opción para crear una aplicación en blanco.

- Se debe crear el código en lenguaje C. Para esto ubicar la pestaña “Project Explorer”, expandir la carpeta “DAQ_AUDIO”, hacer clic derecho en la carpeta “src”, desplegar “New” y hacer clic en “Source File”.

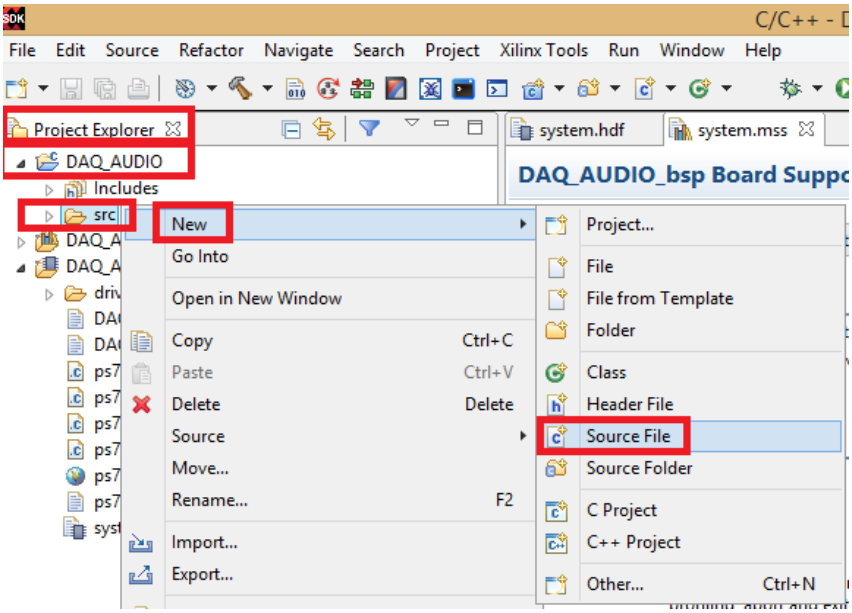

Figura 55. Secuencia para crear un archivo para el código en lenguaje C.

- En la ventana que se abre, darle nombre al archivo como se muestra en la **Figura 56**. Hacer clic en “Finish”.

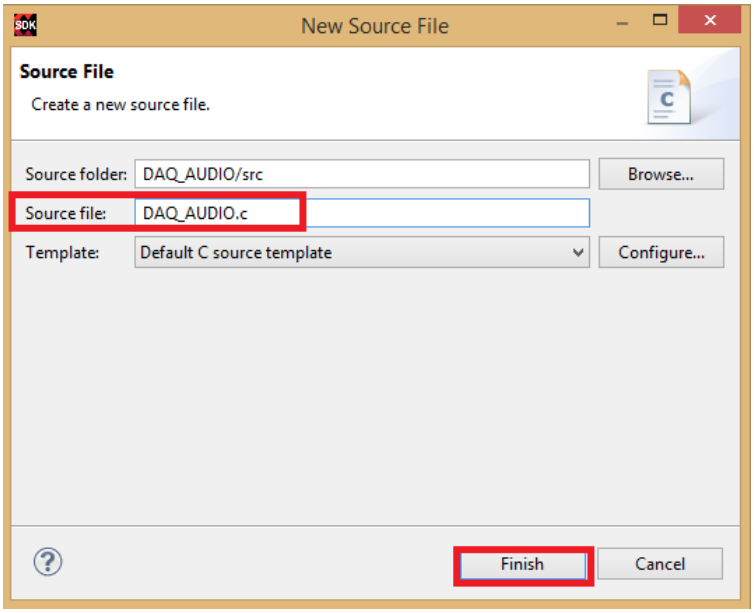

Figura 56. Atributos para el archivo del código.

- Se debe crear un archivo de cabecera con extensión ".h". Para esto ubicar la pestaña "Project Explorer", expandir la carpeta "DAQ_AUDIO", hacer clic derecho en la carpeta "src", desplegar "New" y hacer clic en "Header File".

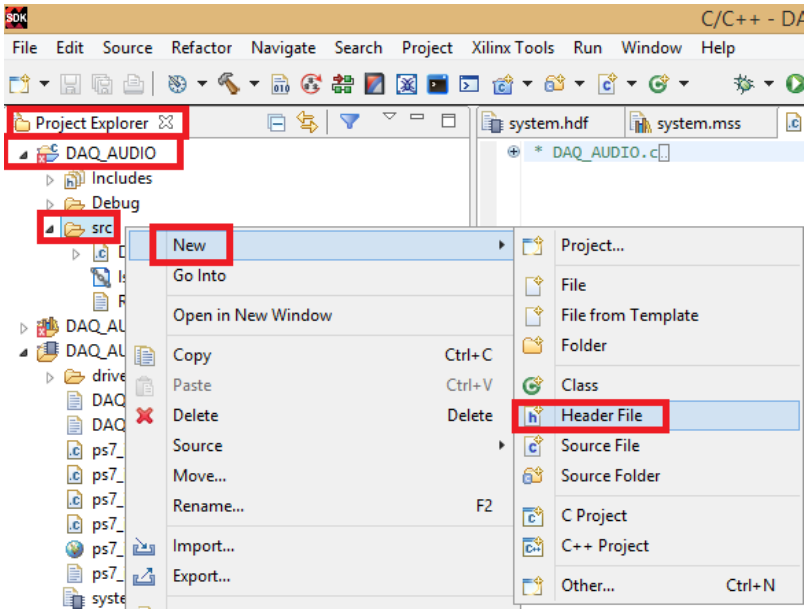

Figura 57. Secuencia para creación del archivo cabecera ".h".

- En la ventana que se abre, darle nombre al archivo como se muestra en la **Figura 58**. Hacer clic en "Finish".

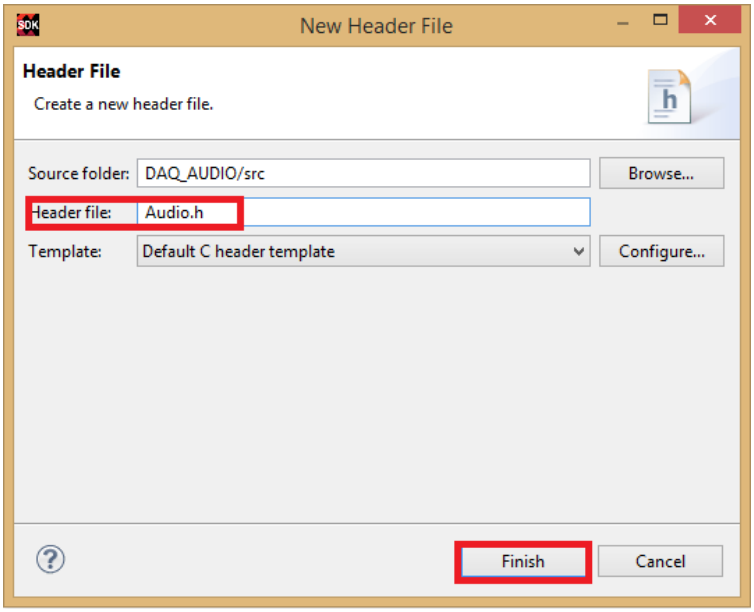

Figura 58. Atributos para el archivo de cabecera.

- Se debe pegar el código en lenguaje C para el archivo “DAQ_AUDIO.c”. Para esto ubicar la pestaña “Project Explorer”, expandir la carpeta “DAQ_AUDIO”, expandir la carpeta “src” y hacer doble clic en “DAQ_AUDIO.c”.

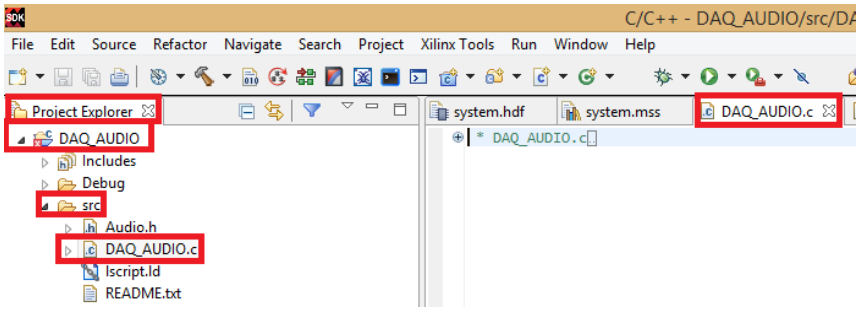

Figura 59. Apertura del archivo "DAQ_AUDIO.c".

- Copiar el código del apéndice A de este trabajo y pegarlo en el área de trabajo de archivo “DAQ_AUDIO.c” y hacer clic en “save”.

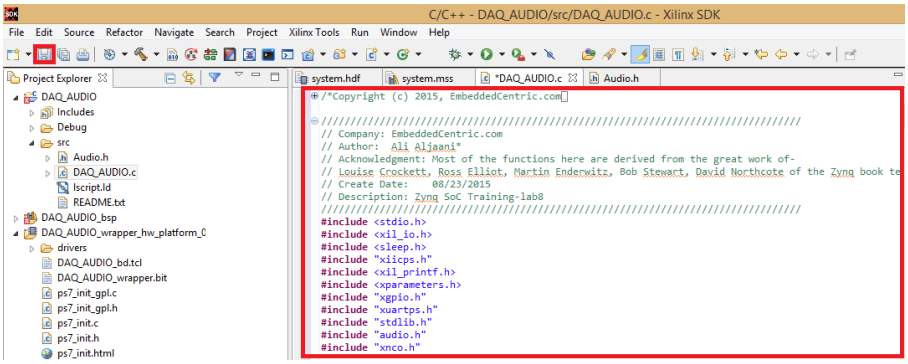

Figura 60. Código en lenguaje C.

- De una manera similar se debe copiar el código para el archivo “Audio.h” que se encuentra en el apéndice B de este trabajo.

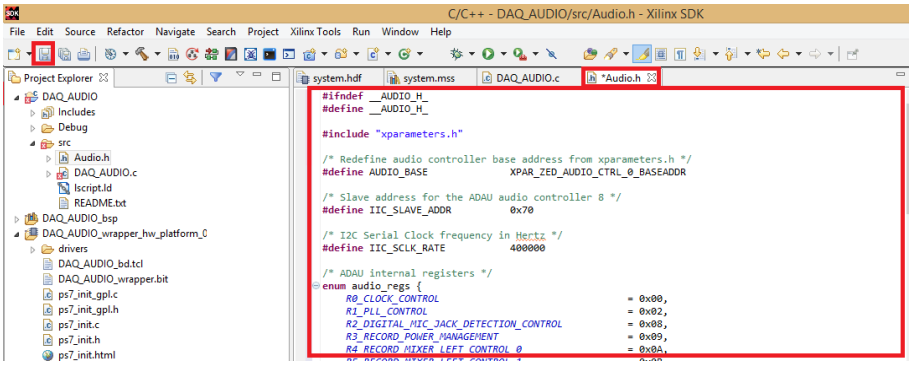

Figura 61. Código para el archivo de cabecera.

m. Terminados todos los procesos anteriores, el proyecto está listo para programarse en la tarjeta ZedBoard y ejecutarse la aplicación. Para esto seguir los pasos:

- Se requiere de dos cables USB cada uno con un conector USB tipo A estándar y el otro micro USB tipo B. como se muestra en la **Figura 62**.

Figura 62. Tipo de conectores USB.

- Se conectan las dos puntas micro USB tipo B a los puestos “UART” y “PROG” de la tarjeta ZedBoard y las dos puntas USB estándar se conectan al computador. También se debe conectar la tarjeta SD al puerto indicado para esta en la ZedBoard. Se conecta el cable de alimentación a la ZedBoard y se enciende por medio del interruptor “SW8”.

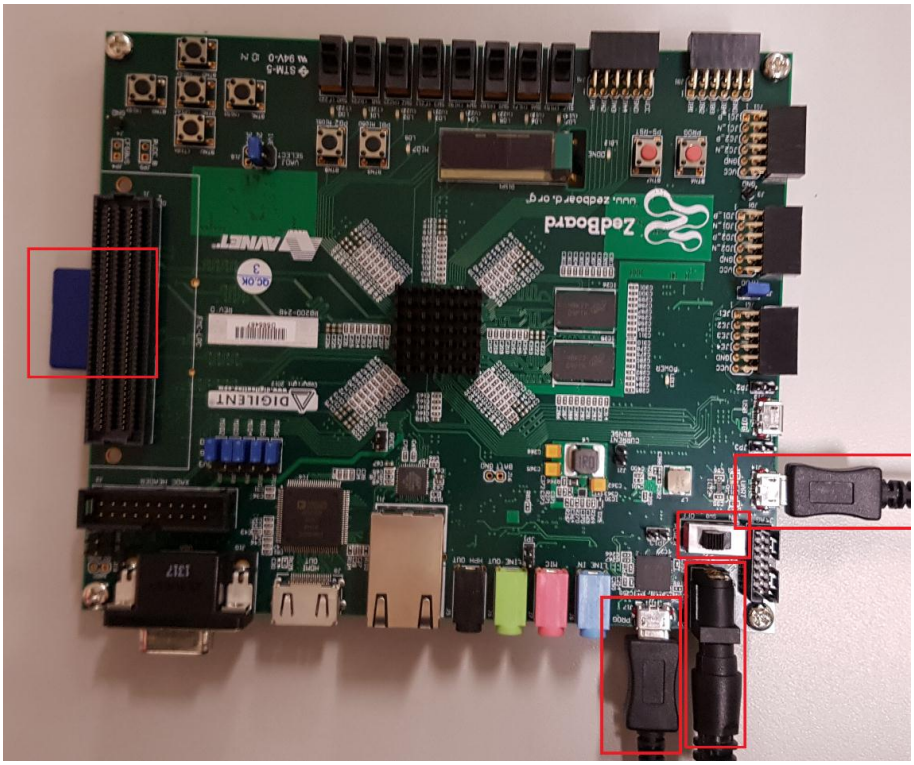

Figura 63.Conexión de la tarjeta ZedBoard.

- Con la tarjeta encendida verificar que puerto COM se habilitó para la conexión de la misma. Para esto ir al administrador de dispositivos del PC en que se está trabajando y desplegar el menú “Puertos (COM y LTP)” y verificar el puerto. Como se muestra en la

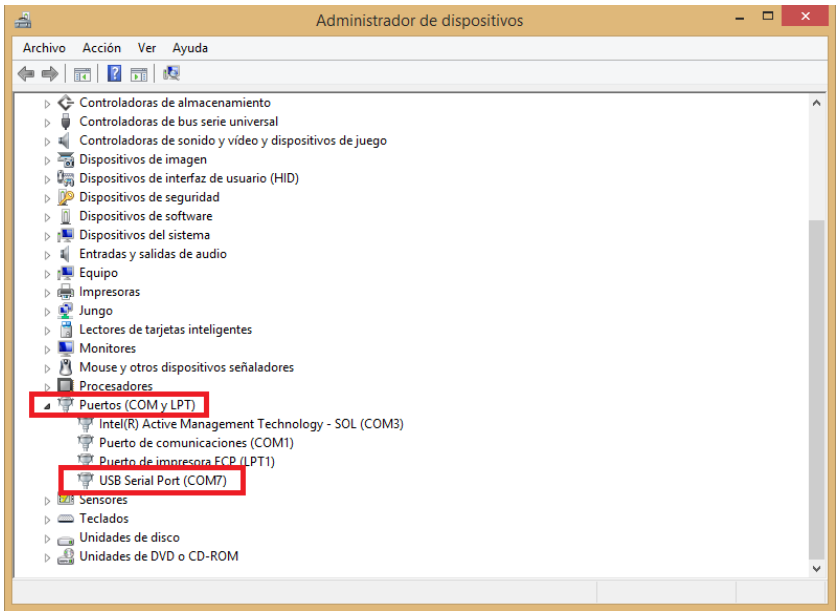

Figura 64. Verificación del puerto de comunicación de la tarjeta ZedBoard.

- Identificado el puerto, se procede a programar la tarjeta ZedBoard. Hacer clic en “Xilinx Tools” y luego en “Program FPGA”.

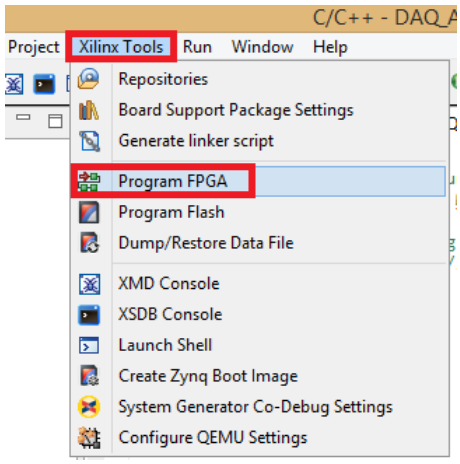

Figura 65. Secuencia para programar la tarjeta ZedBoard.

- En la ventana que se abre asegurarse que estén seleccionados los parámetros correctos como se muestra en la **Figura 66** y hacer clic en "Program". Realizado este procedimiento queda la tarjeta programada.

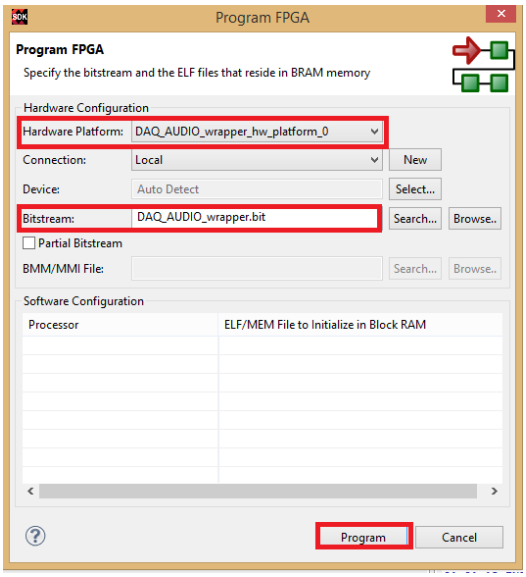

Figura 66. Opciones para programación de la tarjeta ZedBoard.

- Con la tarjeta programada se debe crear una consola "terminal" y conectarla a la tarjeta para observar si el código en lenguaje C si se está ejecutando. Para esto desplegar la pestaña "Window", desplegar "Show View" y hacer clic en "Other"

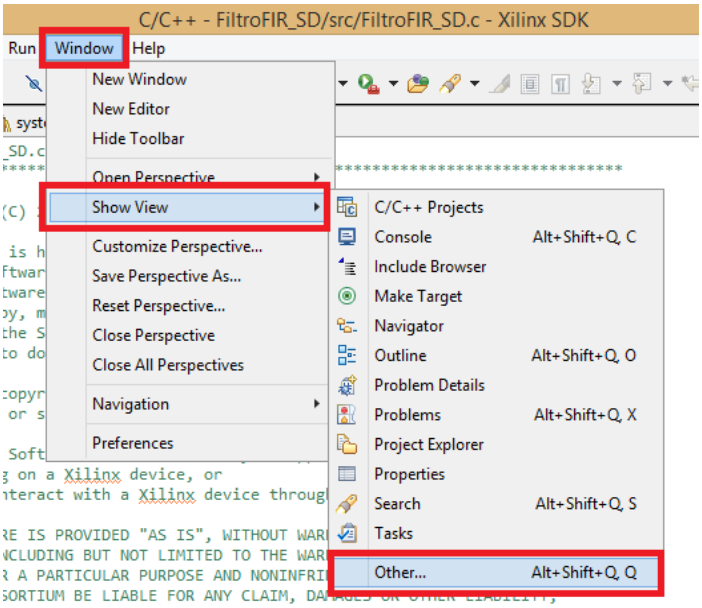

Figura 67. Secuencia para crear la consola terminal.

- En la ventana que se abre, desplegar el menú “Terminal”, seleccionar la opción “Terminal” y hacer clic en “OK”.

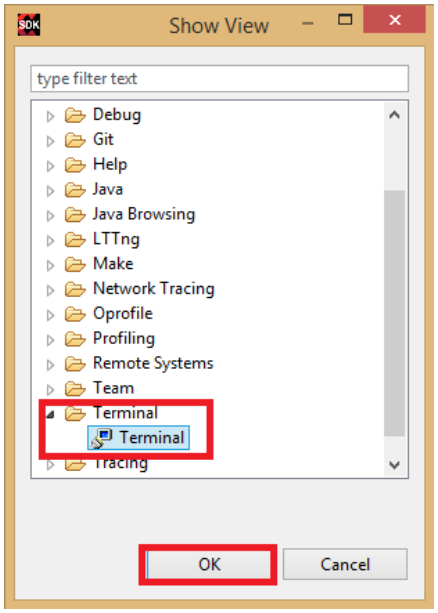

Figura 68.Opcion para abrir la consola terminal.

- En la ventana que se abre, hacer clic en el botón “Settings”.

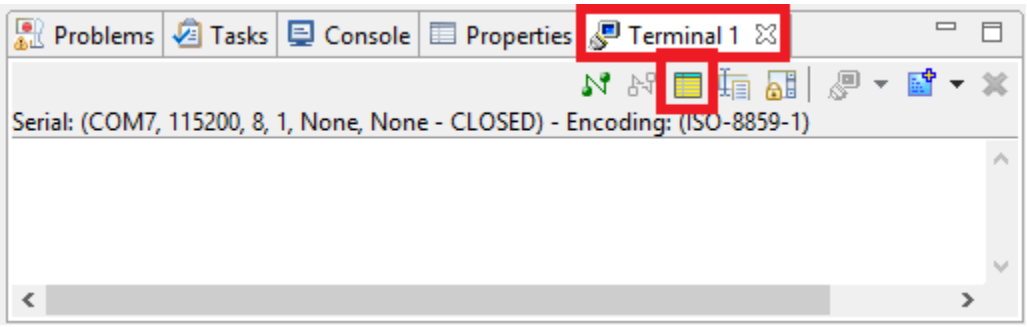

Figura 69. Ubicación del icono "Settings".

- Seleccionar en “Connection Type” la opción “Serial”, en el campo “Port” seleccionar el puerto COM que está asignado a la tarjeta, que en este caso es “COM7”, en el campo “Baut Rate” seleccionar “115200” y hacer clic en “OK”. Con esta acción el terminal está conectado y preparado para visualizar cuando se ejecute el código en lenguaje C.

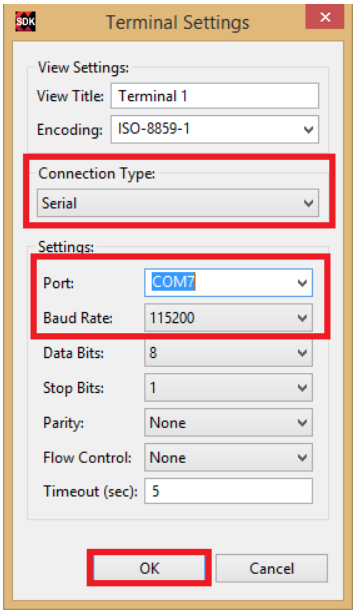

Figura 70. Ventana de configuración de la consola terminal.

- Para ejecutar el código en C, dirigirse a la pestaña “Project Explorer”, hacer clic derecho en “DAQ_AUDIO”, desplegar el menú “Run As” y hacer clic en “Launch on Hardware (GDB)”. Al realizar este procedimiento se envía el código a la ZedBoard para que se ejecute.

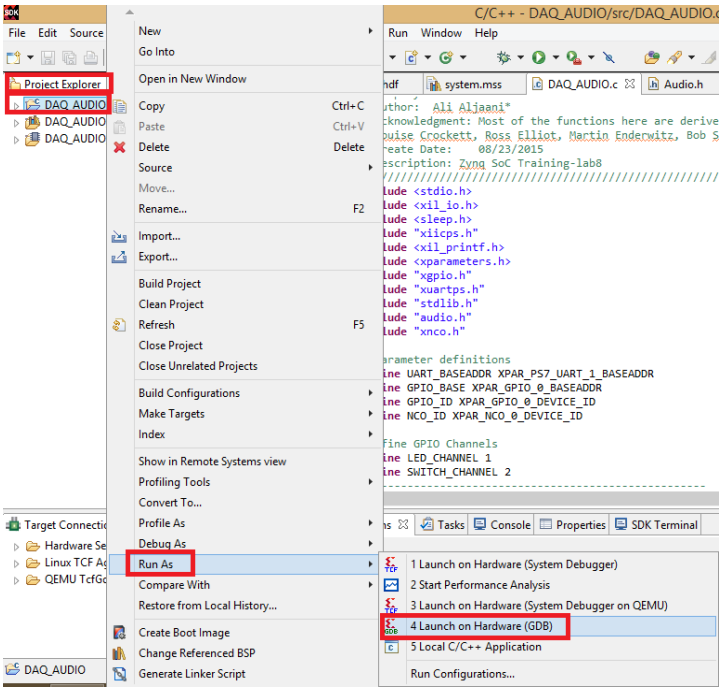

Figura 71. Secuencia para ejecutar el código en C.

n. Para verificar que la aplicación está funcionando correctamente hay tres maneras de comprarlo. Primera, hacer clic en la pestaña “Terminal 1” y observar que cuando se accionan los diferentes interruptores de la tarjeta se imprime “Step = %d, nco_in = %d”, donde “%d” es un número. Segunda, si los LED de la tarjeta ZedBoard encienden cuando se activan los Interruptores. Tercera, conectando audífonos a la línea de salida de audio (Jack Verde), se deben escuchar tonos de diferentes frecuencias al accionar los interruptores de la tarjeta ZedBoard.

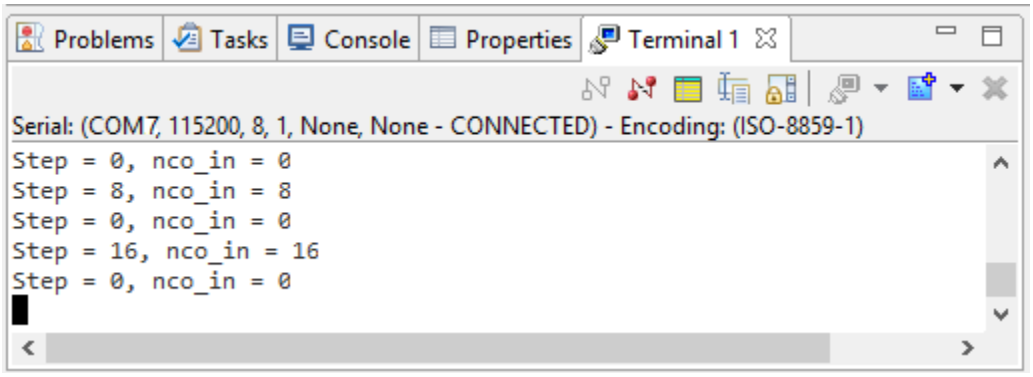

Figura 72. Ventana terminal en ejecución del código en C.

3.2. ALMACENAMIENTO DE AUDIO EN FORMATO DE TEXTO EN LA TARJETA SD.

Este procedimiento describe como se debe almacenar el audio adquirido por medio de la tarjeta ZedBoard. Para esto se hace uso del proyecto creado anteriormente para adquisición de audio. Seguir los pasos descritos a continuación:

a. Ubicar la pantalla principal del proyecto en el software Vivado y abrir el bloque diseño. Para esto ubicar la pestaña "Flow Navigator" y hacer doble clic en "Open Block Design" del menú "IP Integrator".

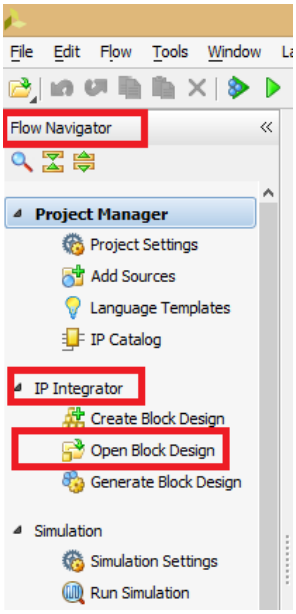

Figura 73. Secuencia para abrir el diseño de bloques.

b. Dar doble clic al bloque "ZYNQ7 proccesing system" para abrir la configuración de dicho elemento.

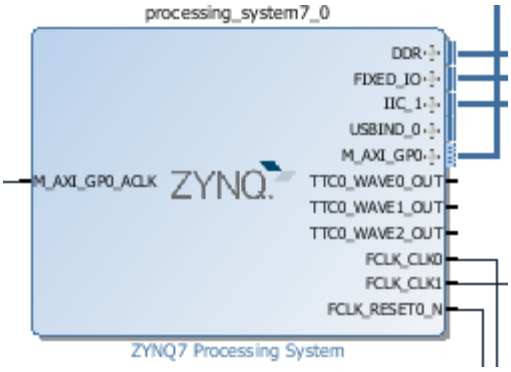

Figura 74. Bloque ZYNQ7.

c. Desplegar la pestaña “MIO Configuration” y abrir el menú “I/O Peripherals. En esta pantalla realizar las siguientes modificaciones, como se muestra en la **Figura 75**:

- En “Bank 1 I/O Voltage” poner la opción “LVCMOS 1.8V”.
- Habilitar la casilla de “SD 0”.
- Habilitar la casilla de “CD” y poner la opción “MIO 47”.
- Habilitar la casilla de “WP” y poner la opción “MIO 46”.
- Desde el MIO 40 al 45 en la opción “IO type” poner “LVCMOS 1.8V”, en la opción “Speed” poner “fast” y en la opción “Pullup” poner “disabled”.
- Habilitar la casilla “UART 1”.
- Las demás configuraciones se dejan por defecto. Finalmente, hacer clic en OK para cerrar.

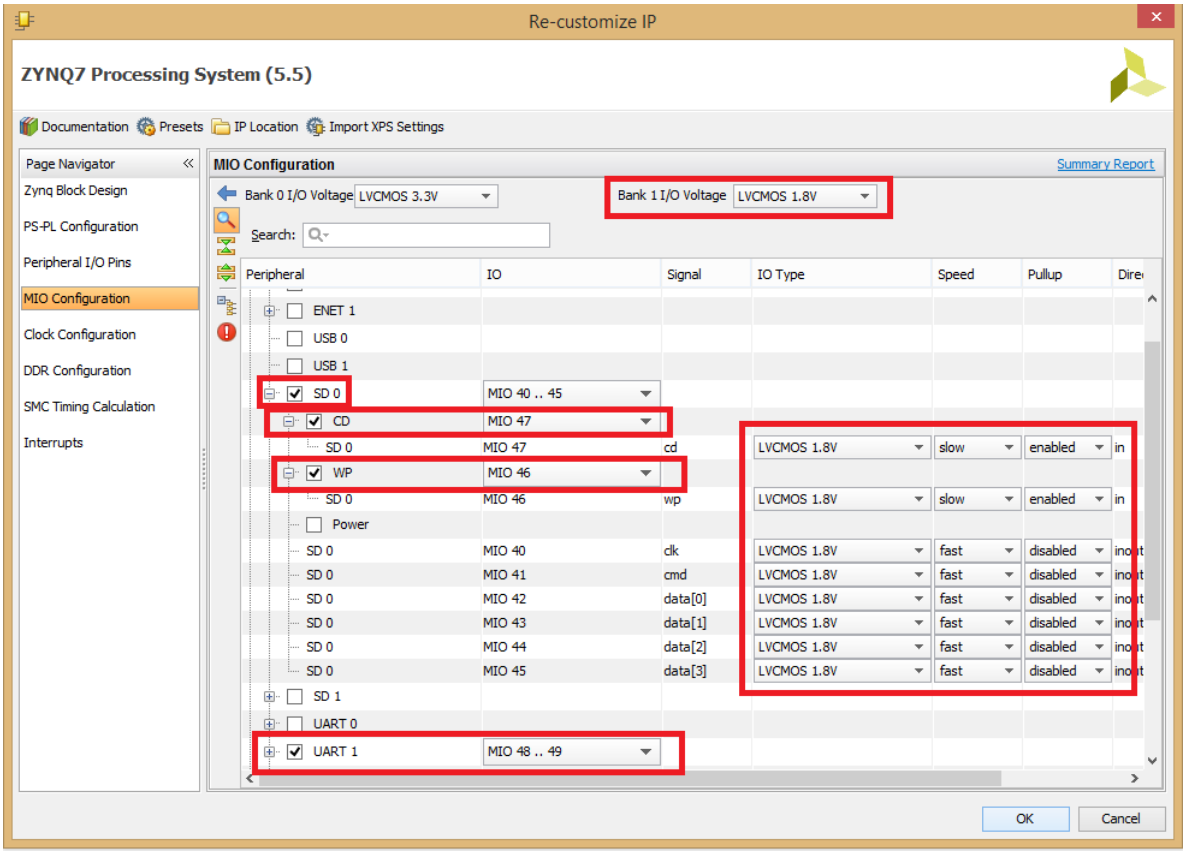

Figura 75. Configuración de periféricos para activar el uso de la tarjeta SD.

d. Se debe actualizar el “Bitstream”. Para eso repetir los pasos “j” y “**¡Error! No se ncuentra el origen de la referencia.**” de la sección **3.1.**

e. Con el proyecto en el SDK, se debe configurar de manera que trabaje el almacenamiento de datos en la tarjeta SD de la siguiente manera:

- Ubicar la pestaña “Project Explorer”, desplegar el menú “DAQ_AUDIO”, desplegar la carpeta “src” y hacer doble clic sobre “DAQ_AUDIO.c” para abrir el programa en lenguaje C. Agregar las librerías necesarias para el almacenamiento de datos en la tarjeta SD las cuales son "xsdps.h", "sd_card/ff.h" y "xil_cache.h".

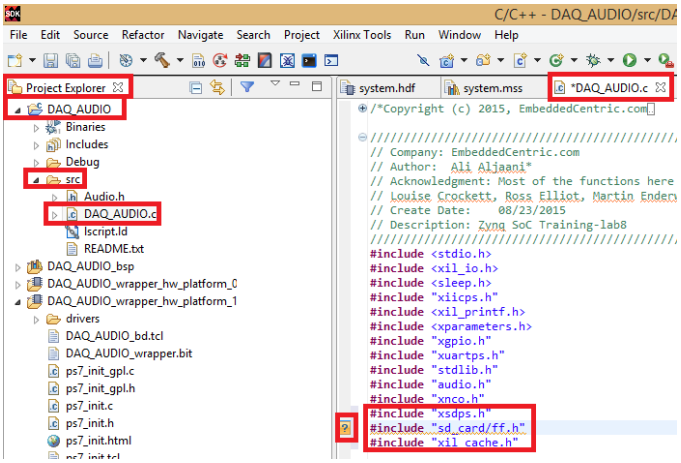

Figura 76. Librerías para uso de la tarjeta SD.

- Al observar la **Figura 76**, se nota que hay un error en el código. Esto se debe a que la librería “sd_card/ff.h” no está incluida en el compilador, ya que la librería es creada. Por este motivo se debe copiar la carpeta llamada “sd_card” en la carpeta “scr”, que se incluye en este trabajo. Lo primero es buscar la carpeta “sd_card” y darle clic derecho-copiar, dirigirse a la pestaña “Project Explorer” desplegar el menú “DAQ_AUDIO”, hacer clic derecho en “src” y hacer clic en “Paste”.

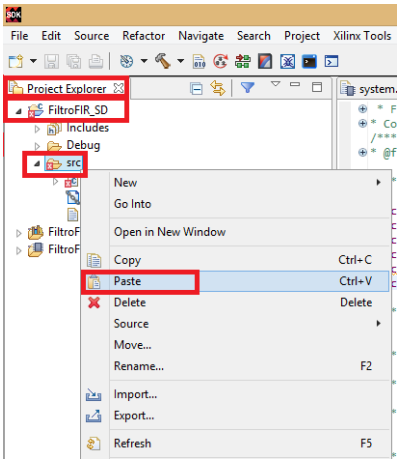

Figura 77. Secuencia para pegar la carpeta "sd_card".

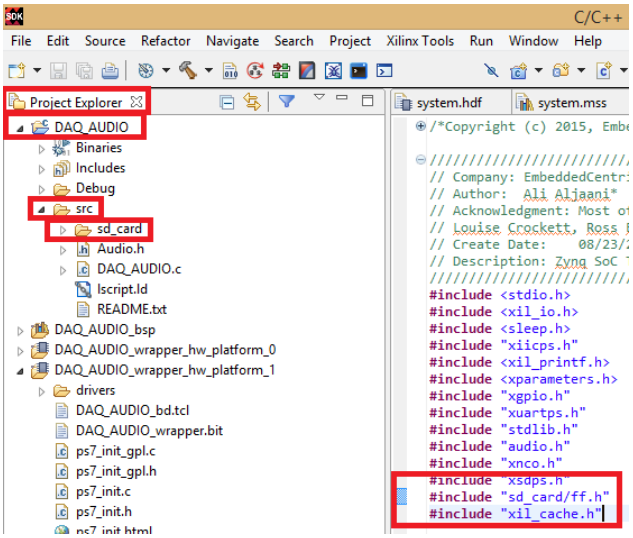

Figura 78. Carpeta "sd_card" copiada.

- La **Figura 78** muestra que el error desaparece y la librería es reconocida por el compilador. Abrir el código “DAQ_AUDIO.c” y agregar el código necesario para que el audio adquirido por la tarjeta ZedBoard se almacene en la tarjeta SD. Como se describe a continuación (código completo en el apéndice C):
    - Se agregan las variables y definiciones necesarias para almacenar datos en la tarjeta SD.

```
#include "xsdps.h"
#include "sd_card/ff.h"
#include "xil_cache.h"

static FIL filin;
static FATFS fatfs;
static char FileNamein[32] = "audioin.txt";
static char *SD_File;
int n,len;
long int accum=0;
double samplein;

#ifdef __ICCARM__
#pragma data_alignment = 32
u8 DestinationAddress[13];
#pragma data_alignment = 4
#else
u8 DestinationAddress[13] __attribute__ ((aligned(32)));
#endif

// Parameter definitions
#define UART_BASEADDR XPAR_PS7_UART_1_BASEADDR
#define GPIO_BASE XPAR_GPIO_0_BASEADDR
#define GPIO_ID XPAR_GPIO_0_DEVICE_ID
#define NCO_ID XPAR_NCO_0_DEVICE_ID
```

Figura 79. Líneas de código para variables de almacenamiento de datos en la SD.

- Líneas que hacen que la tarjeta SD sea reconocida por la ZedBoard y crea el archivo “audioin.txt” donde se almacenan los datos.

```
void read_superpose_play(void)
{
    FRESULT Res;
    UINT NumBytesWritten;

    Res = f_mount(&fatfs,"",1);
    if (Res != FR_OK) {
        return XST_FAILURE;
    }

    SD_File = (char *)FileNamein;
    Res = f_open(&filin, SD_File, FA_CREATE_ALWAYS | FA_WRITE );
    if (Res) {
        return XST_FAILURE;
    }

        u32 nco_in, nco_out, in_left, in_right, out_left, out_right,step, temp;
        // step is associated with the frequency of the sin wave
        /* Read step size value from DIP switches */
        step = XGpio_DiscreteRead(&Gpio, SWITCH_CHANNEL);
```

Figura 80. Líneas de código para que la SD sea reconocida por la ZedBoard.

- Líneas que graban las muestras de audio cuando el Interruptor “sw0” se acciona. Si el interruptor “sw0” no está accionado no se graba audio ni se adquiere.

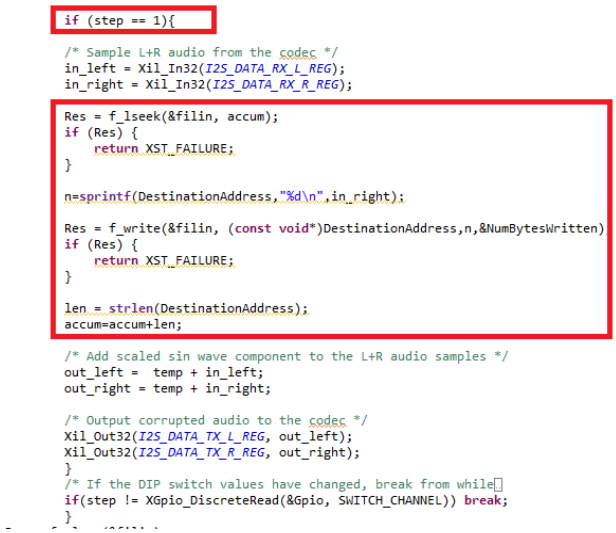

Figura 81. Líneas de código para control de almacenamiento de datos.

- Para que los datos queden grabados en la tarjeta SD se debe cerrar el archivo "audioin.txt". Esto se hace con las líneas mostradas en la **Figura 82** .

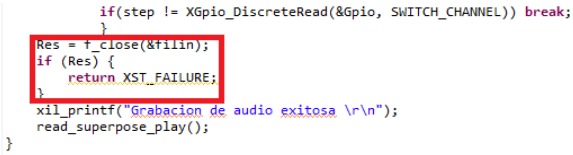

Figura 82. Líneas de código para cerrar el archivo de almacenamiento.

- En el terminal 1 se observa que el proceso de grabación fue exitoso cuando sale el mensaje "Grabación de audio exitosa".

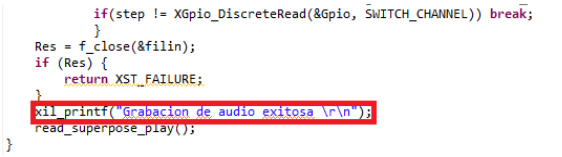

Figura 83. Líneas de código para confirmación exitosa de almacenamiento.

f. Se debe programar la tarjeta ZedBoard y ejecutar el código. Para esto seguir el paso "m" de la sección **3.1**.

g. Para comprobar el funcionamiento del algoritmo seguir los pasos descritos a continuación:

- Con la tarjeta programada y el código en ejecución, hacer clic en la pestaña "terminal 1" para observar los mensajes de confirmación del proceso de almacenamiento.

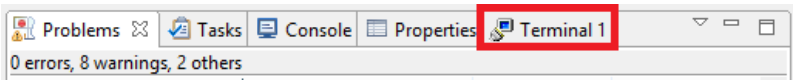

Figura 84. Pestaña "Terminal 1".

- Para que el audio se almacene, se debe activar el interruptor “sw0” de la ZedBoard y para que deje de almacenar el audio se debe desactivar el interruptor “sw0”. Cuando en el terminal se despliegue el mensaje “Grabación de audio exitosa” significa que el audio ha sido guardado en la tarjeta SD. El nombre del archivo generado es “audioin.txt”.

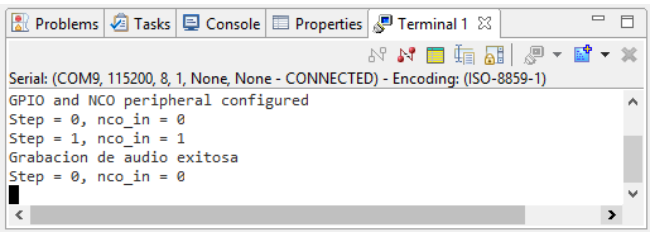

Figura 85. Vista en el terminal del mensaje de confirmación de grabación de datos.

- Para verificar si el archivo se creó y almacenó correctamente seguir los pasos descritos a continuación:
    - Introducir la tarjeta SD en un computador.
    - Abrir el archivo “audioin.txt” y verificar que tenga los datos de audio que se grabaron anteriormente.

3.3. PROCESAMIENTO DE FILTRADO DE AUDIO

El procesamiento aplicado al audio es un filtro FIR pasa alto con frecuencia de corte de 3KHz. Para este procesamiento se va a crear un proyecto nuevo en Vivado para no mezclar la parte de adquisición de audio y simplificar el diseño. Para probar el funcionamiento del filtro se hace uso de una señal de audio grabada en formato de texto en la tarjeta SD que tiene las siguientes características:

- Frecuencia de muestreo de 44100Hz.
- Cantidad de muestras 663552 equivalentes a 15.04 segundos de audio.
- Se le da nombre al archivo de “audioin.txt”.

A continuación, se describe cómo lograr el procesamiento:

a. Como primer paso se debe asegurar que en la tarjeta SD de la ZedBoard este almacenado el archivo de audio original “audioin.txt”. Este procedimiento se debe hacer por medio del computador.
b. Repetir el paso “a” y “b” de la sección **3.1**. Tener en cuenta que en este caso el proyecto se va a llamar “FiltroFIR_SD”. Cuando se esté realizando el paso “b” omitir la parte de hacer clic a “Run Block Automation”.
c. Repetir los pasos “b” y “c” de la sección ¡Error! No se encuentra el origen de la eferencia..
d. Realizar la conexión del bloque “ZYNQ7 Proccesing System” dando clic sostenido en el pin “FCLK_CLK0” y se arrastra hasta el pin “M_AXI_GP0_ACLK”. Después hacer clic en “Run Block Automation”.

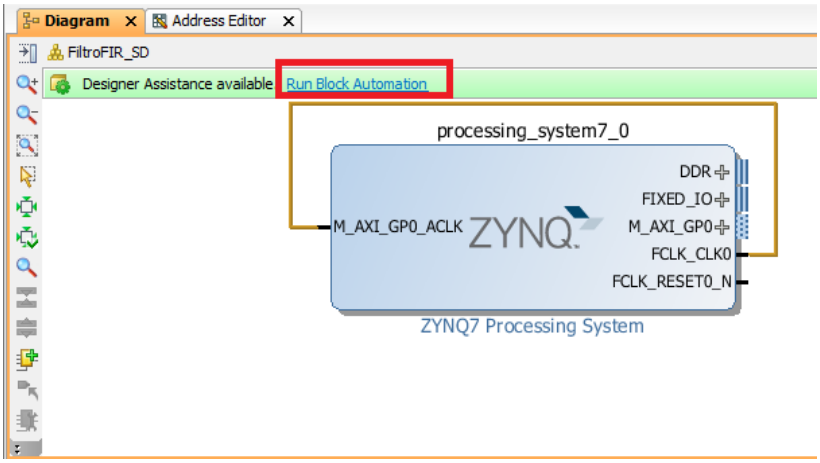

Figura 86. Conexión del bloque ZYNQ7.

e. En la ventana que se abre asegurarse que no esté seleccionada la opción "Apply Board Preset" y luego clic en "OK".

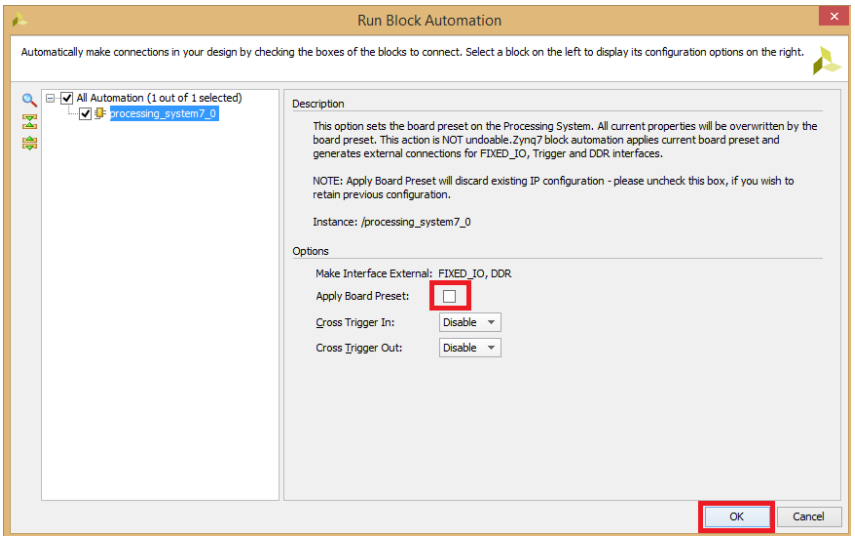

Figura 87. Configuración de Run Block Automation.

- El diagrama del diseño debe quedar como se muestra en la **Figura 88**:

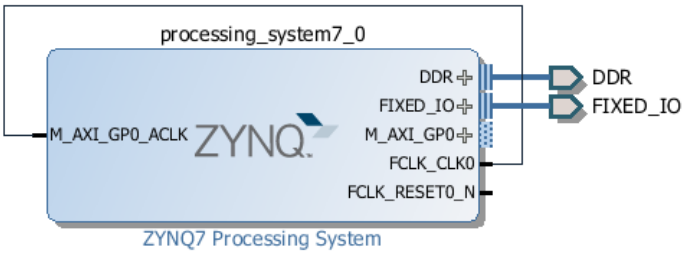

Figura 88. Conexión final del diagrama de bloques.

f. Grabar el diseño realizado desplegando el menú “File” y hacer clic en “Save Block Design”.

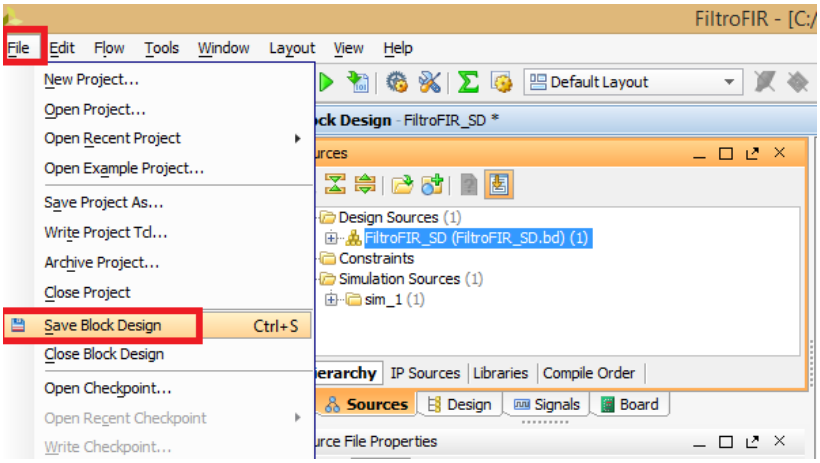

Figura 89. Secuencia para grabar el diseño del diagrama.

g. Generar el archivo de programación “Bitstream”, exportar el hardware y llevarlo al SDK. Para esto repetir los pasos “j” y “**¡Error! No se encuentra el origen de la referencia.**” e la sección **3.1**.

h. Se puede cerrar el software Vivado y seguir trabajando en el SDK. Crear una aplicación nueva en el SDK de la siguiente manera:

- Para esto desplegar la pestaña “File”, desplegar el menú “New” y hacer clic en “Application project” como se muestra en la **Figura 52**.
- En la ventana que se abre se le da nombre al proyecto. Además, se debe asegurar que el “OS Platfom” sea “Standlone” y que el “Hardware Plataform” sea el correcto como se muestra en la **Figura 90**. Hacer clic en “Next”, asegurarse de seleccionar “Empty Application” y hacer clic en “Finish”.

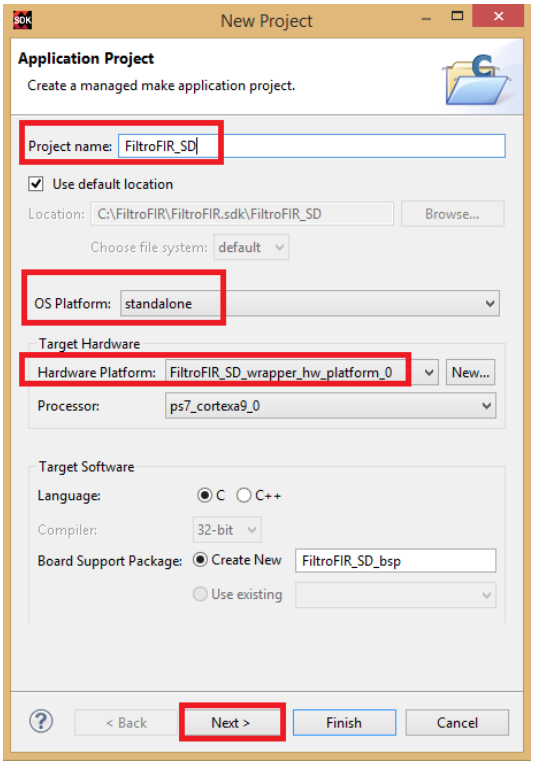

Figura 90. Opciones para creación de la nueva aplicación.

- Se debe crear el archivo en lenguaje C que va a realizar el algoritmo de procesamiento de audio. Para esto se dirige a la pestaña “Project Explorer” desplegar el menú “FiltroFIR_SD”, hacer clic derecho en “src”, desplegar el menú “New” y hacer clic en “Source File”.

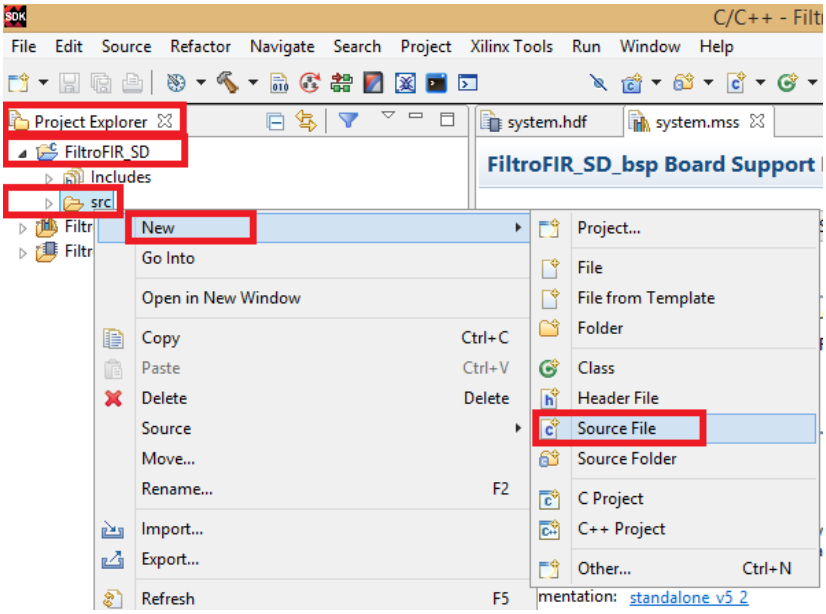

Figura 91. Secuencia para creación de archivo para el código en lenguaje C.

- En la ventana que se abre, dar nombre al archivo que en este caso se llama "FiltroFir_SD.c" y hacer clic en "Finish".

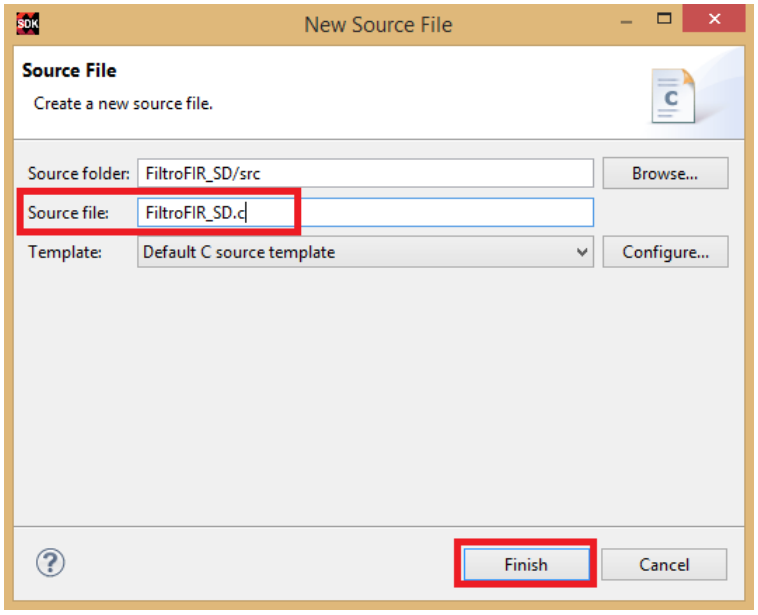

Figura 92. Atributos para el archivo del código en C.

- Se debe pegar el algoritmo de programación que realiza el procesamiento de datos. Para esto, primero se copia el código que se describe en el apéndice D de este trabajo, se pega en el archivo que se creó y hacer clic en "save".

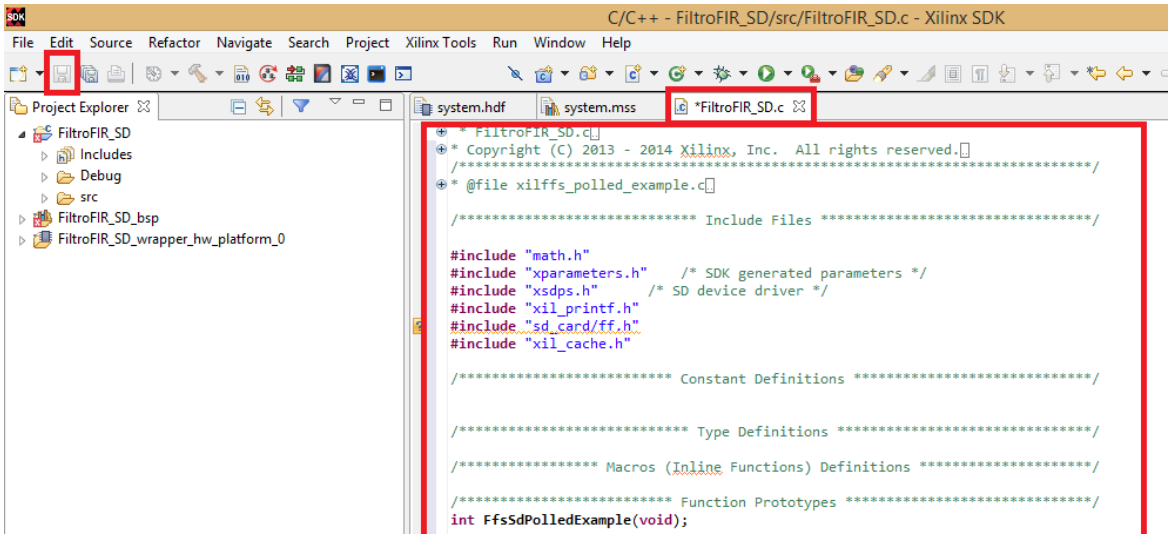

Figura 93. Código en lenguaje C.

- Al observar la imagen anterior, se nota que hay un error en el código. Esto se debe a que la librería “sd_card/ff.h” no está incluida en el compilador, ya que esta librería es creada. Por este motivo se debe copiar la carpeta llamada “sd_card”, que se incluye en este trabajo de grado. Lo primero es buscar la carpeta “sd_card” y hacer clic derecho y copiar, luego se dirige a la pestaña “Project Explorer” desplegar el menú “FiltroFIR_SD”, hacer clic derecho en “src” y clic en “Paste”.

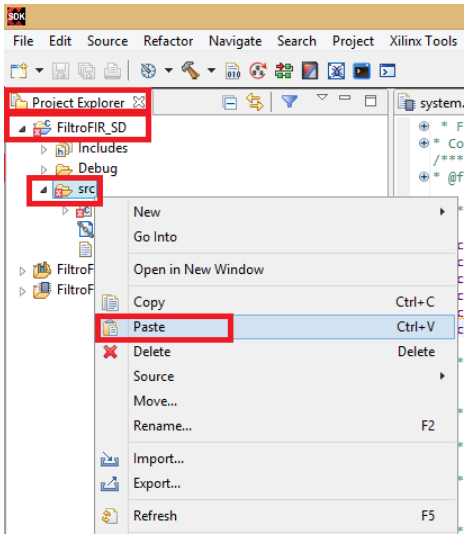

Figura 94. Secuencia para pegar la carpeta “sd_card”.

- En la **Figura 95** se observa que la librería “sd_card/ff.h” es reconocida por el compilador.

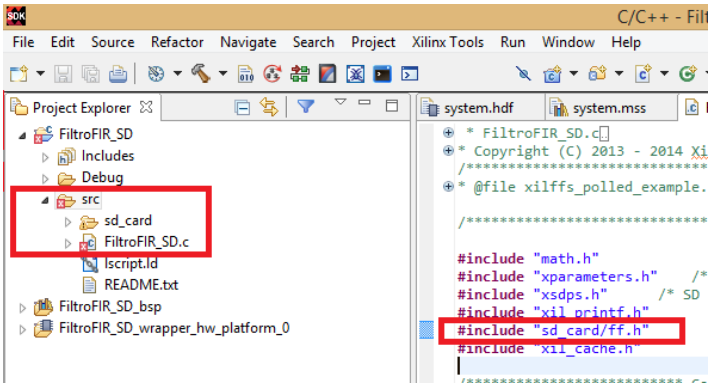

Figura 95. Carpeta "sd_card" copiada.

- En la **Figura 96** se observa que existe un error en el código. El error se presenta en las líneas que ejecutan la función seno que pertenece a la librería “math.h”.

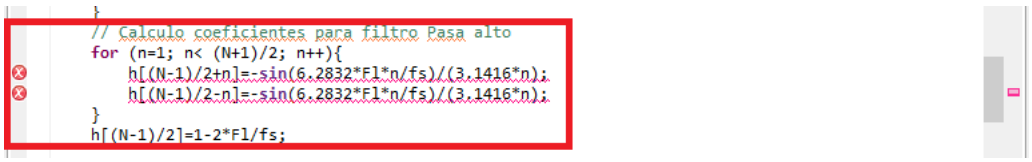

Figura 96. Error en las líneas que ejecutan la función seno.

- Para solucionar el error que se observa en la **Figura 96** se realiza una configuración en la librería “math.h” para que trabaje correctamente de la siguiente manera:
    - Hacer clic en “FiltroFIR_SD” para asegurarse que este seleccionado, desplegar la pestaña “Project” y hacer clic en “Properties”.

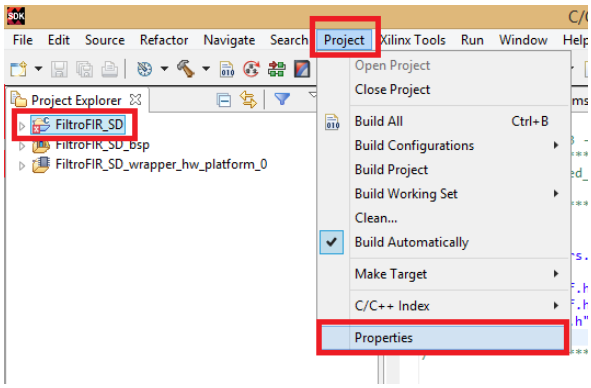

Figura 97. Secuencia para abrir las propiedades.

- En la ventana que se abre desplegar el menú “C/C++ Build” y hacer clic en “Settings”. En la ventana de configuración que se despliega hacer clic en “Libraries”. Observar que se abren dos recuadros, el que interesa es “Libraries (-l)”.

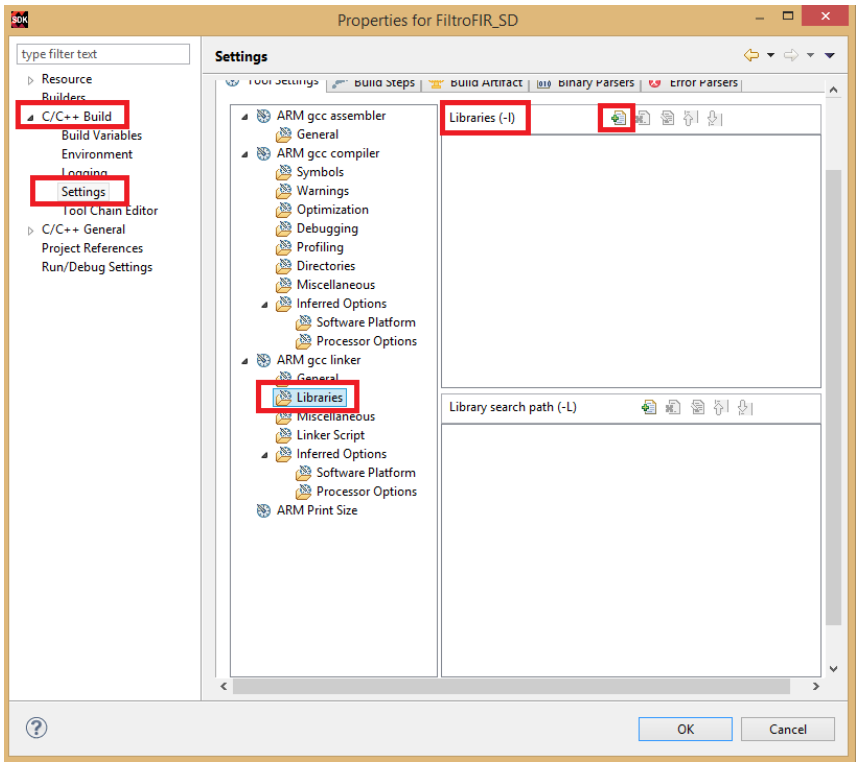

Figura 98. Opciones de configuración de las propiedades.

- Hacer clic en el botón agregar que está identificado con el símbolo “+” de color verde, en la ventana que se abre escribir la letra “m” y hacer clic en “OK” y nuevamente en “OK”. Con este proceso se corrige el error presentado en la función seno de la librería “math.h”

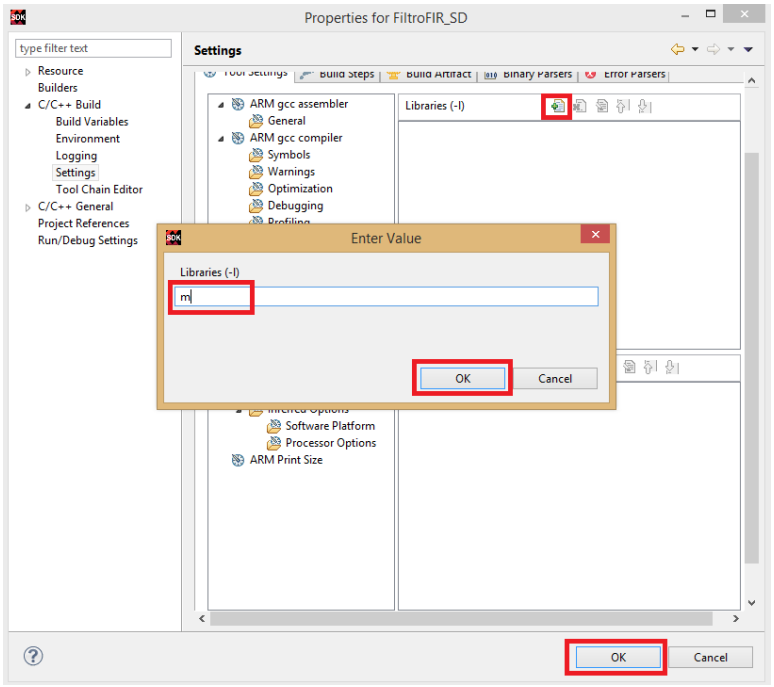

Figura 99. Opción para corregir la librería "math.h".

- en la se observa que ya no existen errores en el código en C, solo existen “Warnings” que se pueden ignorar.

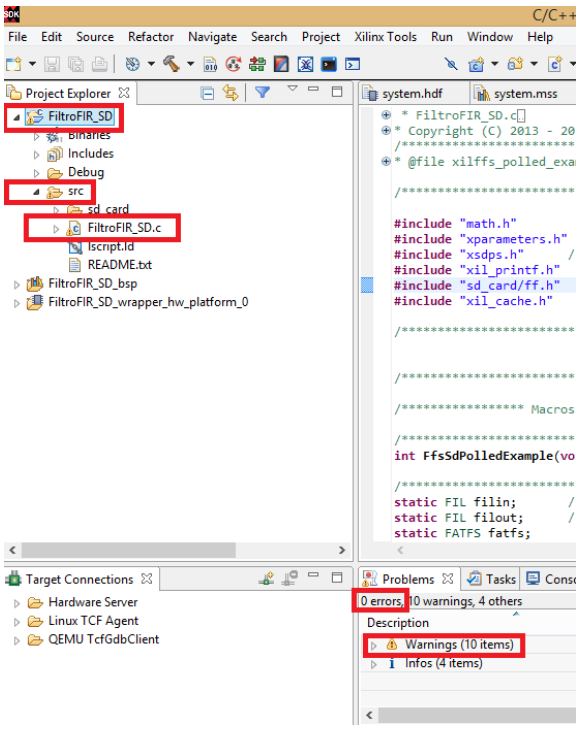

Figura 100. No existen errores en el código en C.

i. Con el código corregido y funcionando, se debe programar la tarjeta ZedBoard y ejecutar el código. Para esto repetir el paso "m" de la sección **3.1**. Tener en cuenta que algunos nombres van a cambiar porque el proyecto se llama de una manera diferente al proyecto de la sección **3.1**.

j. Para observar que proceso está realizando el algoritmo de procesamiento de datos se debe dar clic en la pestaña "Terminal 1". Cuando se despliega el mensaje "El procesamiento del audio se cumplió satisfactoriamente" es porque el algoritmo se ejecutó con éxito. En la tarjeta SD se cuenta con el archivo en forma de texto con el audio filtrado y se identifica con el nombre "audioout_Zed.txt". Para verificar que este archivo se creó correctamente se debe ingresar la tarjeta SD al computador, abrir el archivo y verificar que los datos estén almacenados. Se recomienda guardar este archivo en el computador para su posterior análisis a través del software MATLAB.

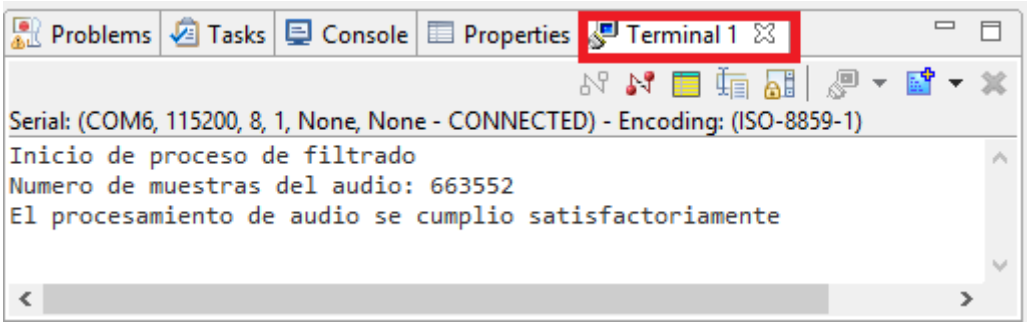

Figura 101. Visualización en el terminal de confirmación exitosa de procesamiento.

3.4. PROCESAMIENTO DE AUDIO A TRAVÉS DE MATLAB

Implementar un algoritmo en MATLAB que haga el mismo procesamiento de audio que la tarjeta ZedBoard para comparar ambos procesamientos. Se debe usar el mismo archivo de audio original almacenado en forma de texto que se usó en el procesamiento anterior. El nombre que se le dio a este archivo "audioin.txt".

Para crear el algoritmo en MATLAB se siguen los pasos descritos a continuación:

- Se abre el programa MATLAB.

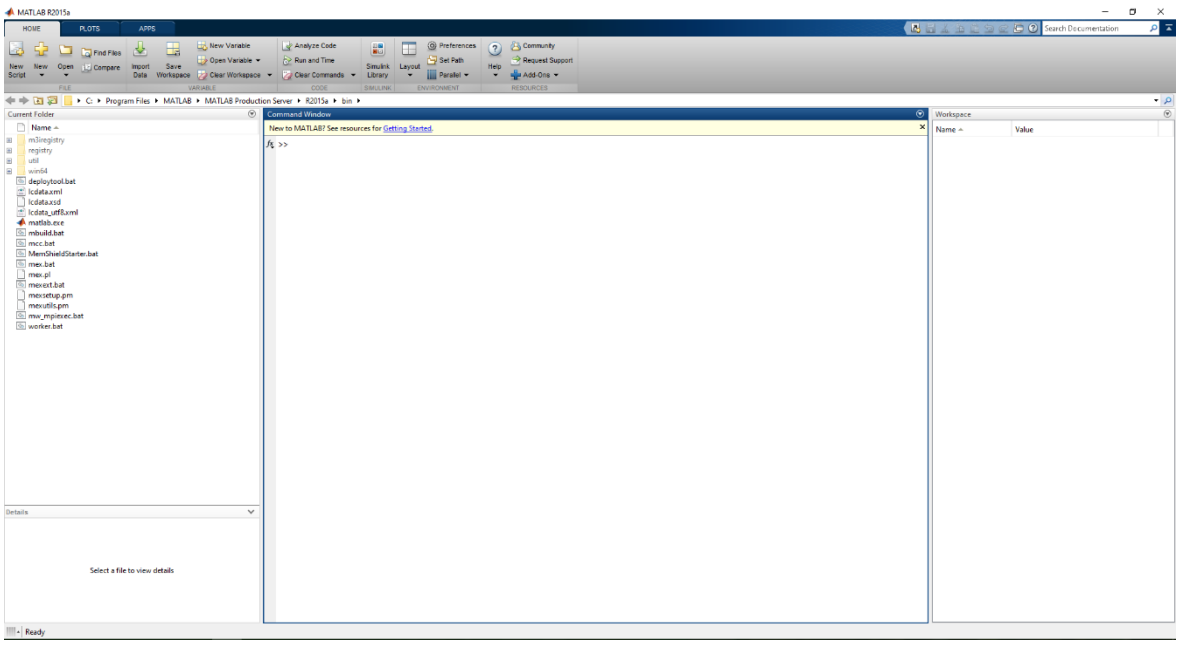

Figura 102. Ventana principal del software MATLAB.

- Se recomienda crear una carpeta de trabajo y allí almacenar todos los archivos concernientes al procesamiento de audio de este trabajo. La ruta definida en este caso es "C:\Users\User1\Documents\MATLAB\Procesamiento_Audio". Se debe garantizar que al ejecutar el MATLAB se esté situado en la ruta definida anteriormente.

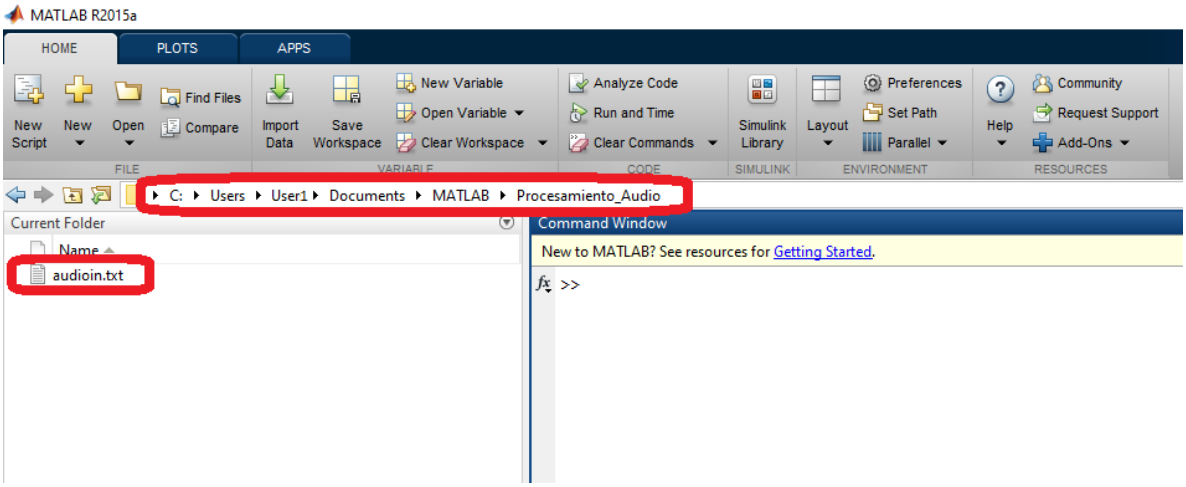

Figura 103. Carpeta designada para los archivos del procesamiento de audio.

Como se ve en la **Figura 103**, se está situado en la ruta especificada y se tiene el archivo de audio original en formato txt almacenado en dicha ruta.

- Para escribir el código que se encarga del procesamiento de audio se recomienda crear un script. En este caso se llamará "FiltroFIR.m". Para esto se hace clic en "New Script", hacer clic en "Save" para guarda el *script* y ponerle el nombre indicado. Recordar guardarlo en la carpeta creada anteriormente.

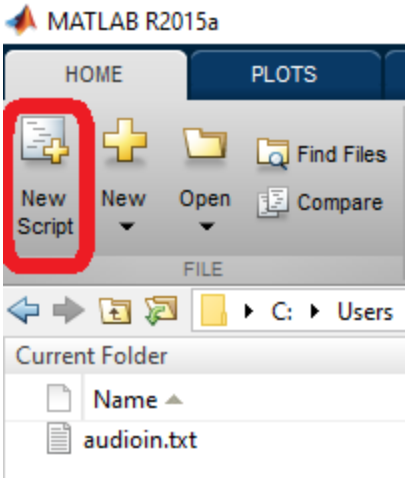

Figura 104. Icono para creación de scripts.

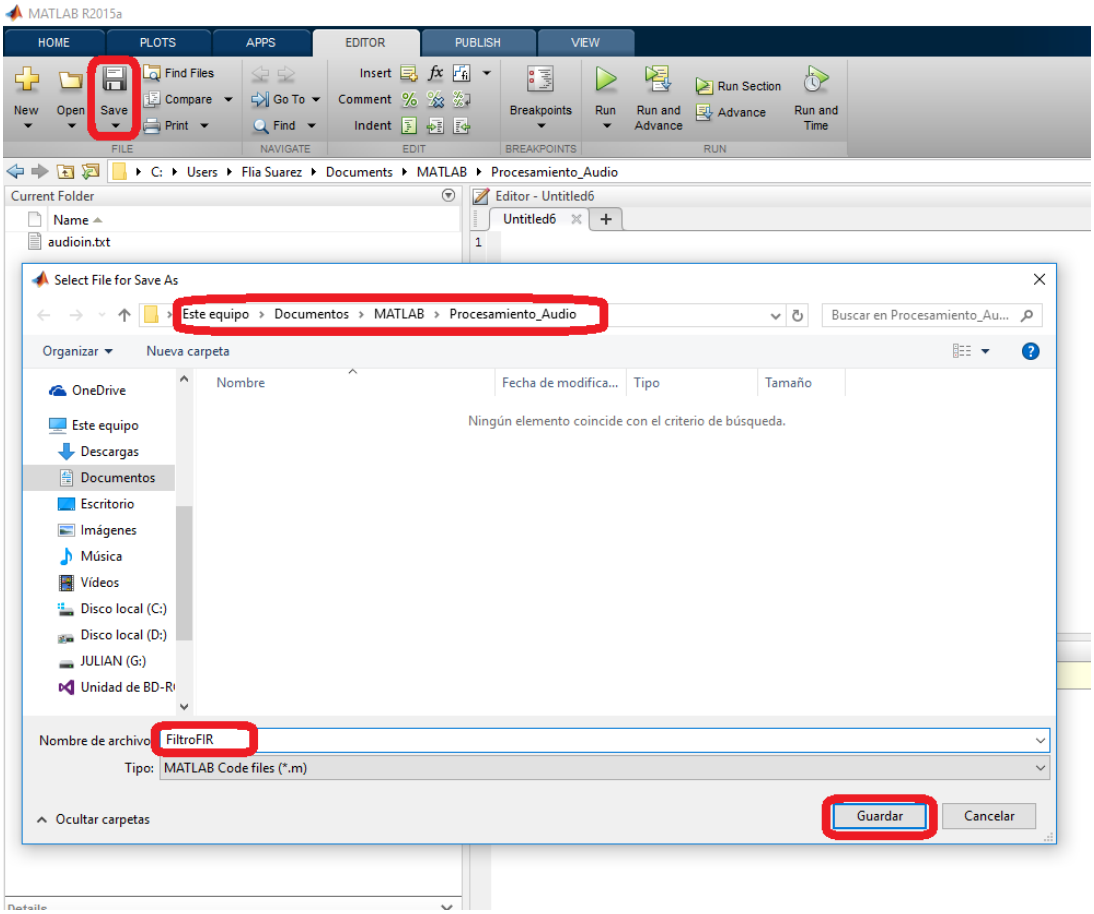

Figura 105. Nombre y ruta de almacenamiento del script.

- Realizado el procedimiento anterior, se puede ver que a la carpeta creada se agrega un nuevo archivo con extensión ".m" que es el script creado anteriormente con el nombre que se definió. La pestaña de ejecución del script también toma el mismo nombre.

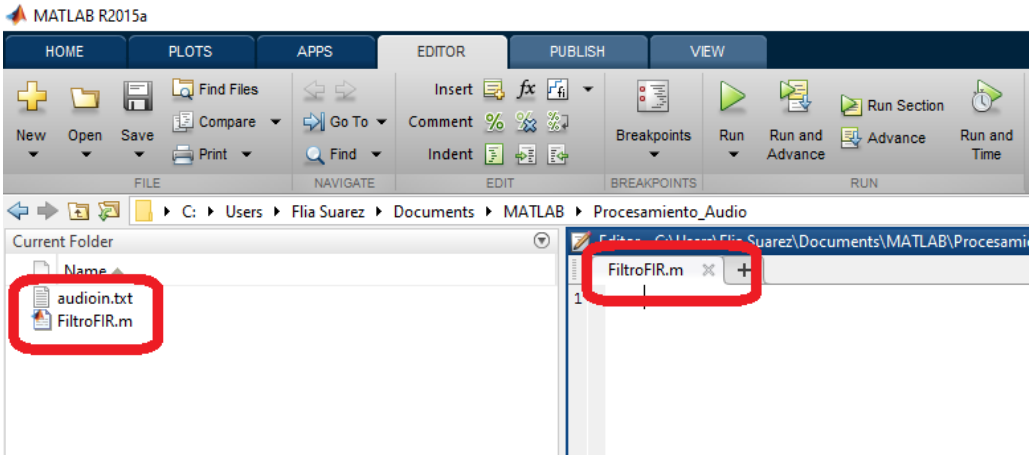

Figura 106. Script FiltroFIR.m en la carpeta de trabajo.

- Se escribe el script para realizar el procesamiento de audio disponible en el apéndice E de este trabo que consiste básicamente en seguir la rutina descrita a continuación:
    - Cargar el audio original en txt a un vector en MATLAB, llamado “audioin”. Esto se hace por medio de la instrucción “importdata”
    - Definir los parámetros del filtro tales como N, Fc y Fs
    - Crear los coeficientes del filtro h(n).
    - Aplicar el filtro y almacenarlo en un vector llamado “audioout_MATLAB”
    - Graficar el audio original y el filtrado.
    - Si se desea, escuchar el resultado del audio filtrado.

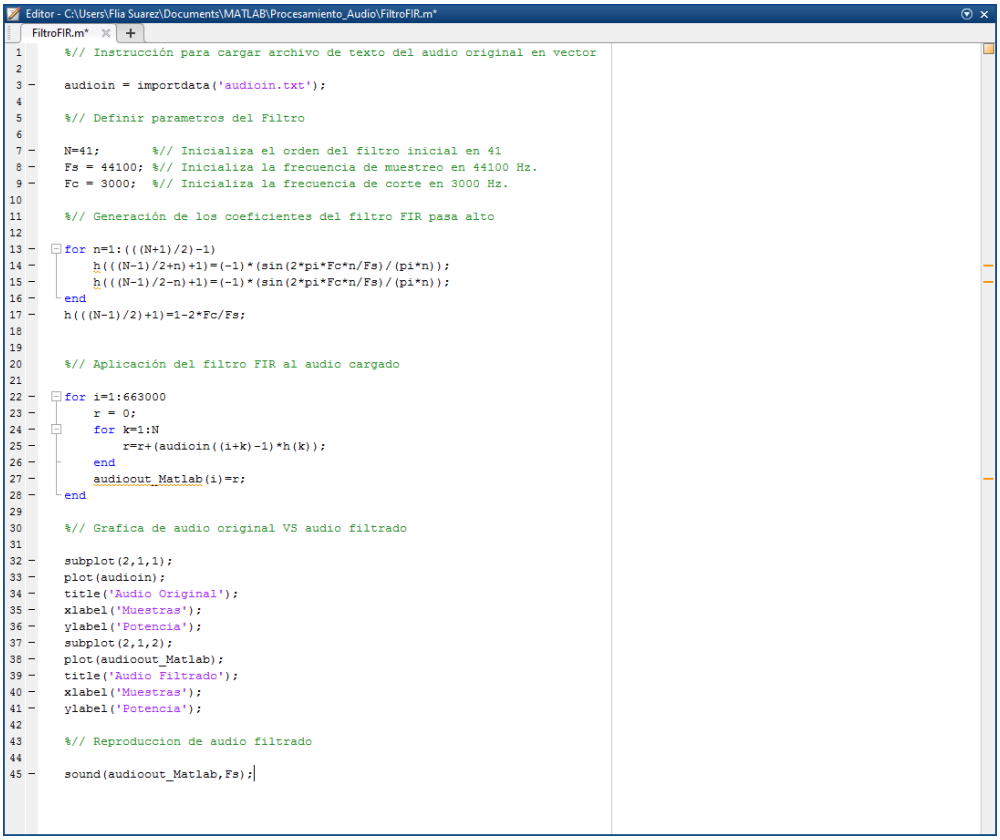

Figura 107. Código del script.

- Para ejecutar el script se da clic en “Run”.

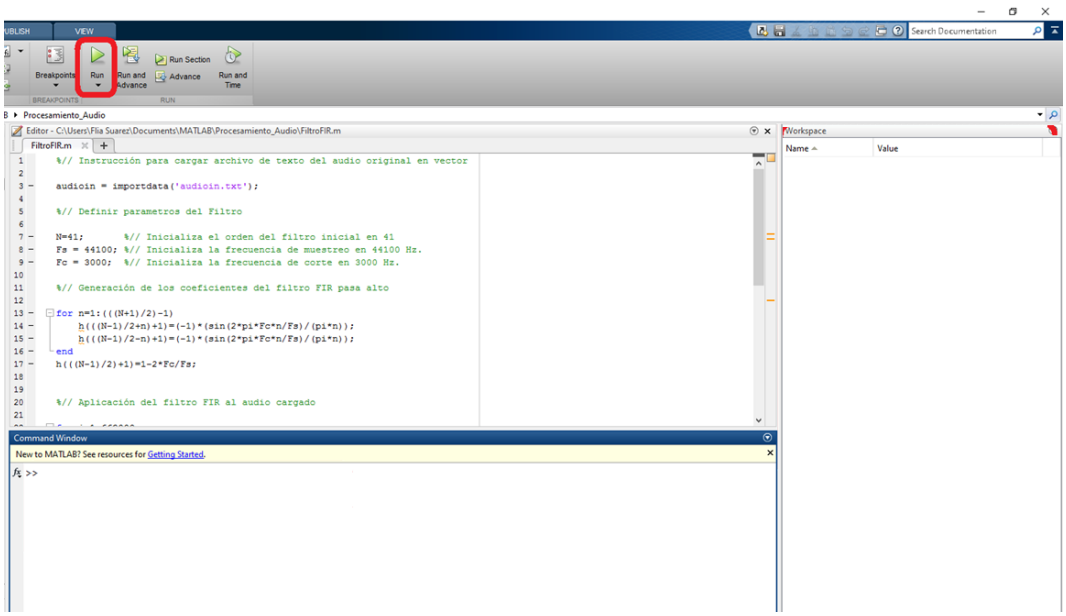

Figura 108. Icono "Run" para ejecutar el script.

- Para comprobar que el *script* se ejecutó, observar que en la ventana "Command Window" debe aparecer el texto "FiltroFIR" como se muestra en la **Figura 109**. Además, se debe crear una ventana con la gráfica del audio original comparada con el audio filtrado como se muestra en la **Figura 110** e inmediatamente se debe escuchar el audio filtrado a través de los parlantes del computador.

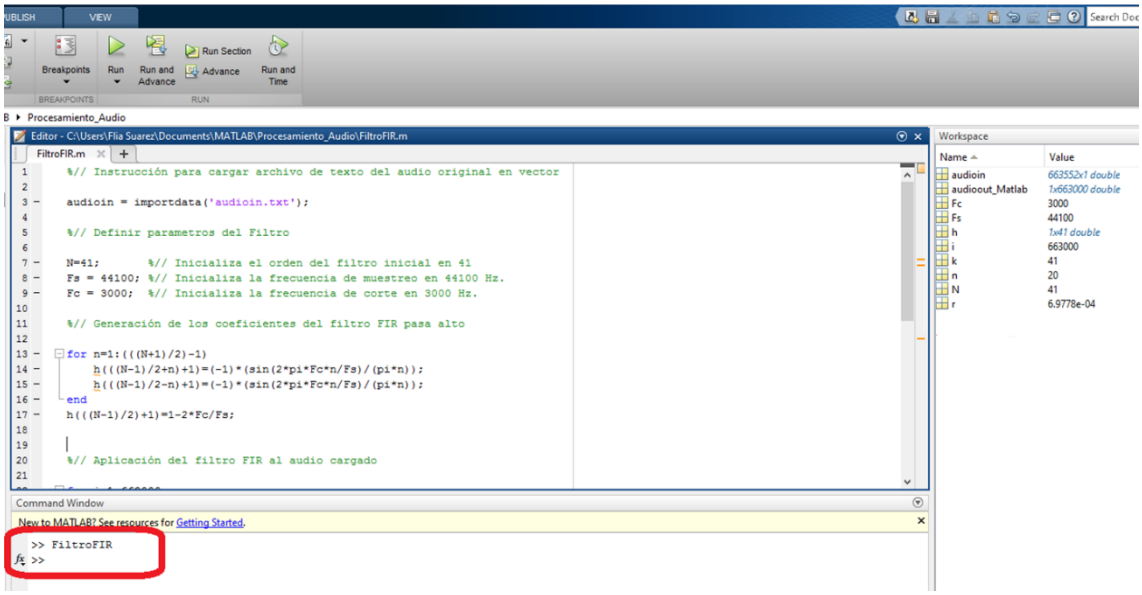

Figura 109. Ejecución del script.

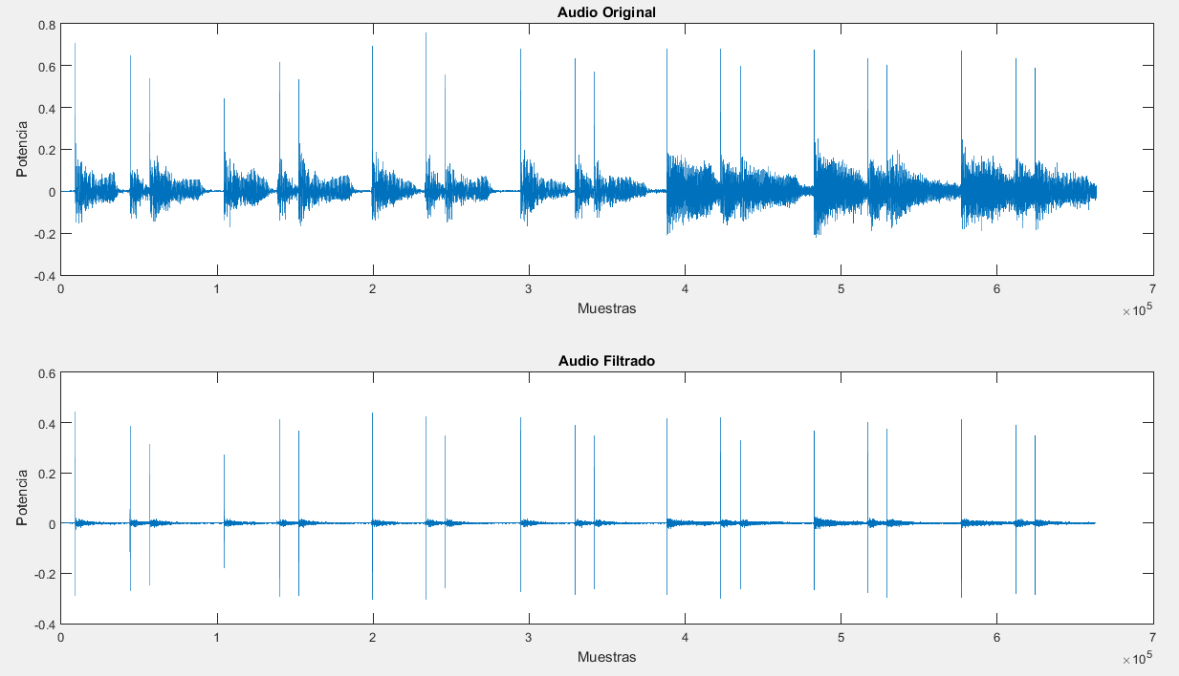

Figura 110. Gráfica de audio original VS audio filtrado.

Se puede ver en la **Figura 110** que el audio filtrado tiene una disminución de amplitud. Esto se debe a que las amplitudes de las frecuencias por debajo de 3KHz, fueron eliminadas. Al reproducir el audio filtrado por los parlantes del computador se escucha el audio, pero con tonos agudos que corresponden a frecuencias altas.

3.5. COMPARACIÓN DE RESULTADOS

Para comparar los resultados de procesamiento de audio obtenidos por la tarjeta ZedBoard y MATLAB, se debe crear otro script en MATLAB en la misma carpeta que se describe en la sección **3.4.** Para esto seguir el procedimiento descrito a continuación:

- Guardar el audio procesado por la ZedBoard en la carpeta “Procesamiento_Audio”. Este audio tiene el nombre de “audioout_Zed.txt”.

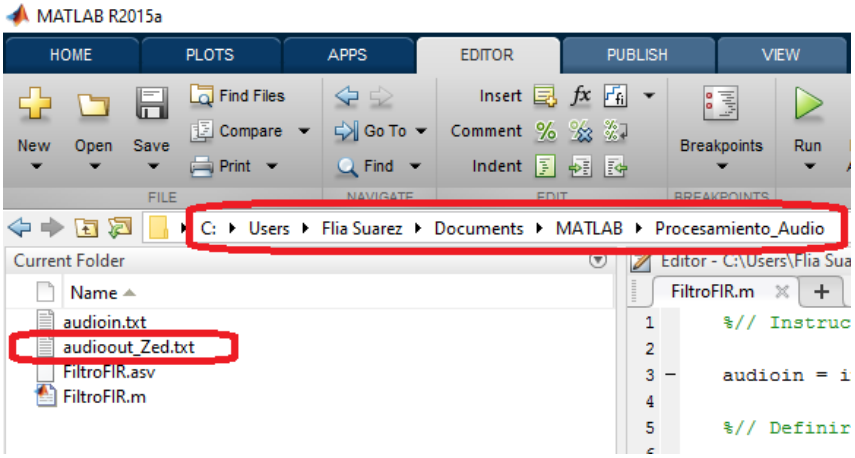

Figura 111. Archivo en txt de audio procesado por la ZedBoard.

- Con el MATLAB abierto, ubicar la ruta de la carpeta “Procesamiento_Audio”. Crear un nuevo script como se describe en la sección **3.4** y darle el nombre de “Comparación”. Pegar el código del apéndice F en el *script*.

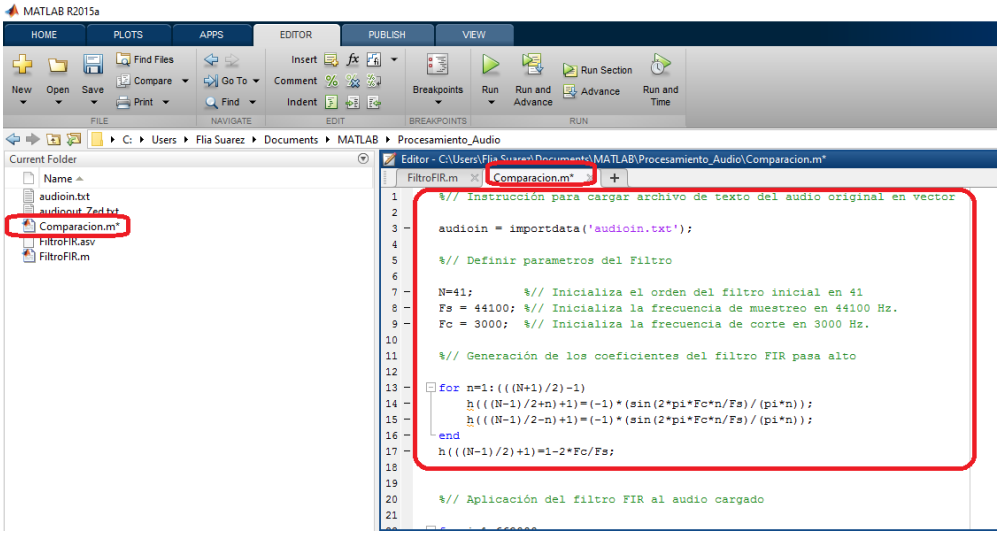

Figura 112. Script de comparación.

- Ejecutar el script desde el botón “Run”. Inmediatamente debe aparecer una ventana con tres gráficas como se observa en la **Figura 113**.

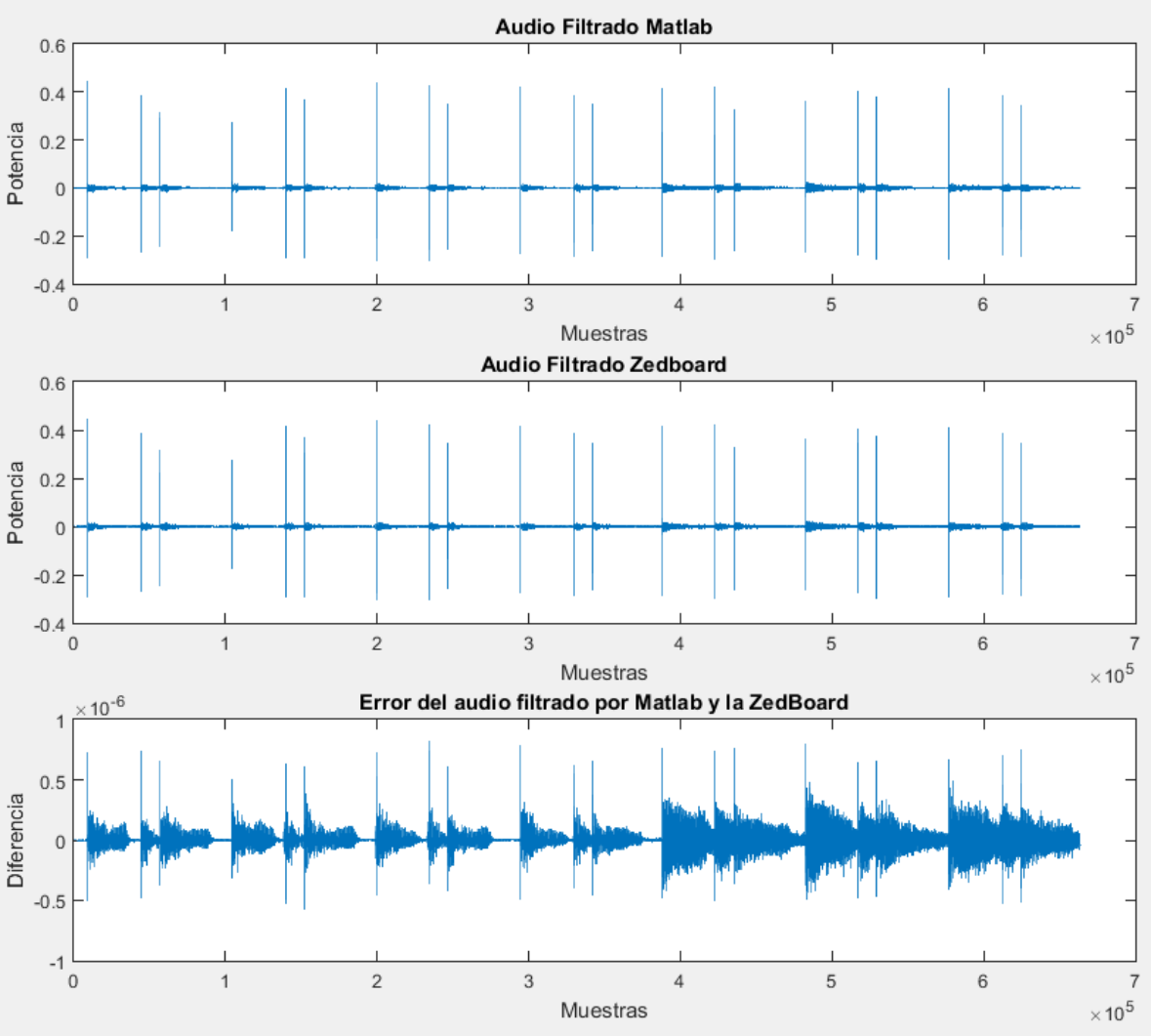

Figura 113. Gráficas de comparación de audio procesado en MATLAB y en la ZedBoard.

- Si se desea escuchar, por medio de los parlantes del computador, el audio filtrado en MATLAB y el filtrado por la ZedBoard se aprovecha que los vectores y variables trabajadas en el *script* están disponibles en el “Workspace” o espacio de trabajo como se muestra en la **Figura 114**. Para esto ir área de trabajo del “Command Window” y escribir el comando “soundsc(audioout_MATLAB,Fs);”, esperar a que el audio termine de reproducirse y escribir el comando “soundsc(audioout_Zed,Fs);”.

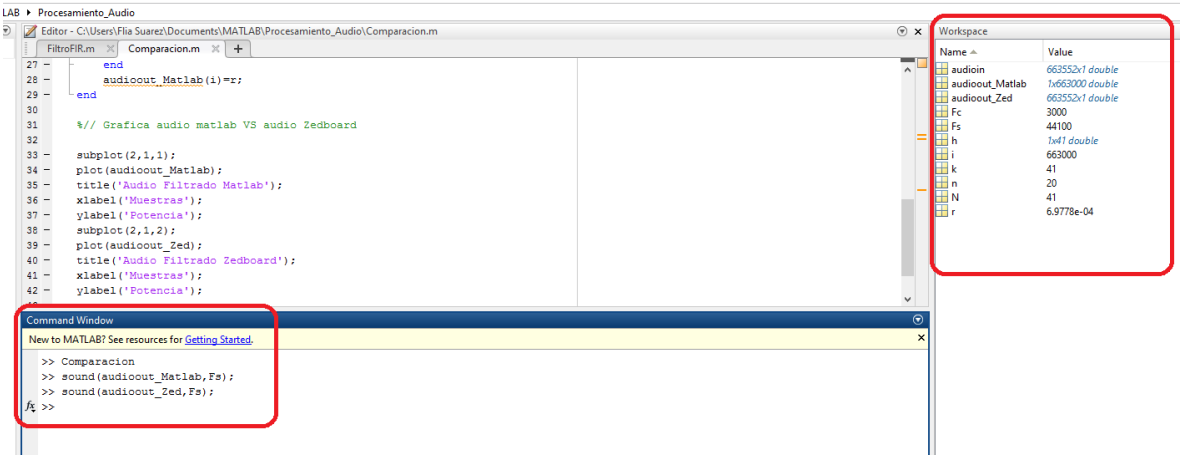

Figura 114. Instrucciones para reproducir el audio filtrado.

4. RESULTADOS Y DISCUSIÓN

En el proceso de adquisición y almacenamiento de audio el formato de los datos se da en números enteros como se muestra en la **Figura 115**. Pero el algoritmo de procesamiento (Filtro Digital FIR) trabaja con números en formato flotante. Por este motivo se debe realizar un proceso matemático para convertir los datos del formato de números enteros a flotantes teniendo en cuenta las especificaciones de rango y resolución de los conversores ADC Y DAC.

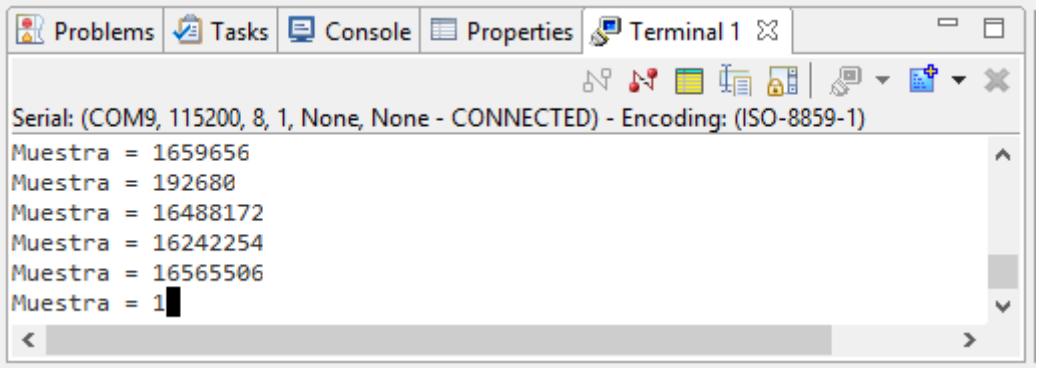

Figura 115. Formato de los datos adquiridos por la tarjeta ZedBoard visto en el Terminal.

Para probar el algoritmo de procesamiento (Filtro Digital FIR) se hace uso de un audio en formato de texto almacenado en la tarjeta SD de la ZedBoard. El audio que se utilizó fue llamado “audioin.txt” y tiene las siguientes características:

- Formato de almacenamiento en texto.
- Valor decimal de mayor y menor valor 0.7595214844 y -0.2106323242 respectivamente.
- Frecuencia de muestreo Fs = 44100Hz
- Cantidad de muestras 663552, lo que equivale a 15.04 segundos de audio.

Por medio de un nuevo proyecto en el software Vivado, se realiza un algoritmo de programación que lee el archivo de audio original en texto llamado “audioin.txt”, lo

transforma a formato flotante, realiza el procedimiento de filtrado y finalmente guarda el audio filtrado en forma de texto en la tarjeta SD con el nombre de “audioout_Zed.txt”. El algoritmo de procesamiento trabaja muestra a muestra y realiza el siguiente procedimiento por muestra:

- Lee la muestra de texto en la SD.
- Convierte la muestra en formato flotante.
- Aplica el filtro FIR a la muestra.
- Convierte la muestra filtrada nuevamente a texto.
- Guarda la muestra filtrada en la tarjeta SD.
- Se repite la misma secuencia por cada muestra.

Según la descripción anterior se realiza muchos procesos para poder filtrar una muestra, lo que representa una demora en el procesamiento. Además, el proceso de leer de la tarjeta SD y almacenar son los que toman más tiempo.

Al realizar la comparación del procesamiento de audio tanto de MATLAB como de la tarjeta ZedBoard se tienen resultados muy similares. La **Figura 116** muestra la gráfica del audio procesado por MATLAB “Audio Filtrado Matlab”, la gráfica del audio procesado por la tarjeta Zedboard “Audio Filtrado Zedboard” y la gráfica de error entre ambos procesamientos “Error del audio filtrado por Matlab y Zedboard”. En la gráfica “Error del audio filtrado por Matlab y Zedboard” se observa que existen errores del orden 10^{-6}.

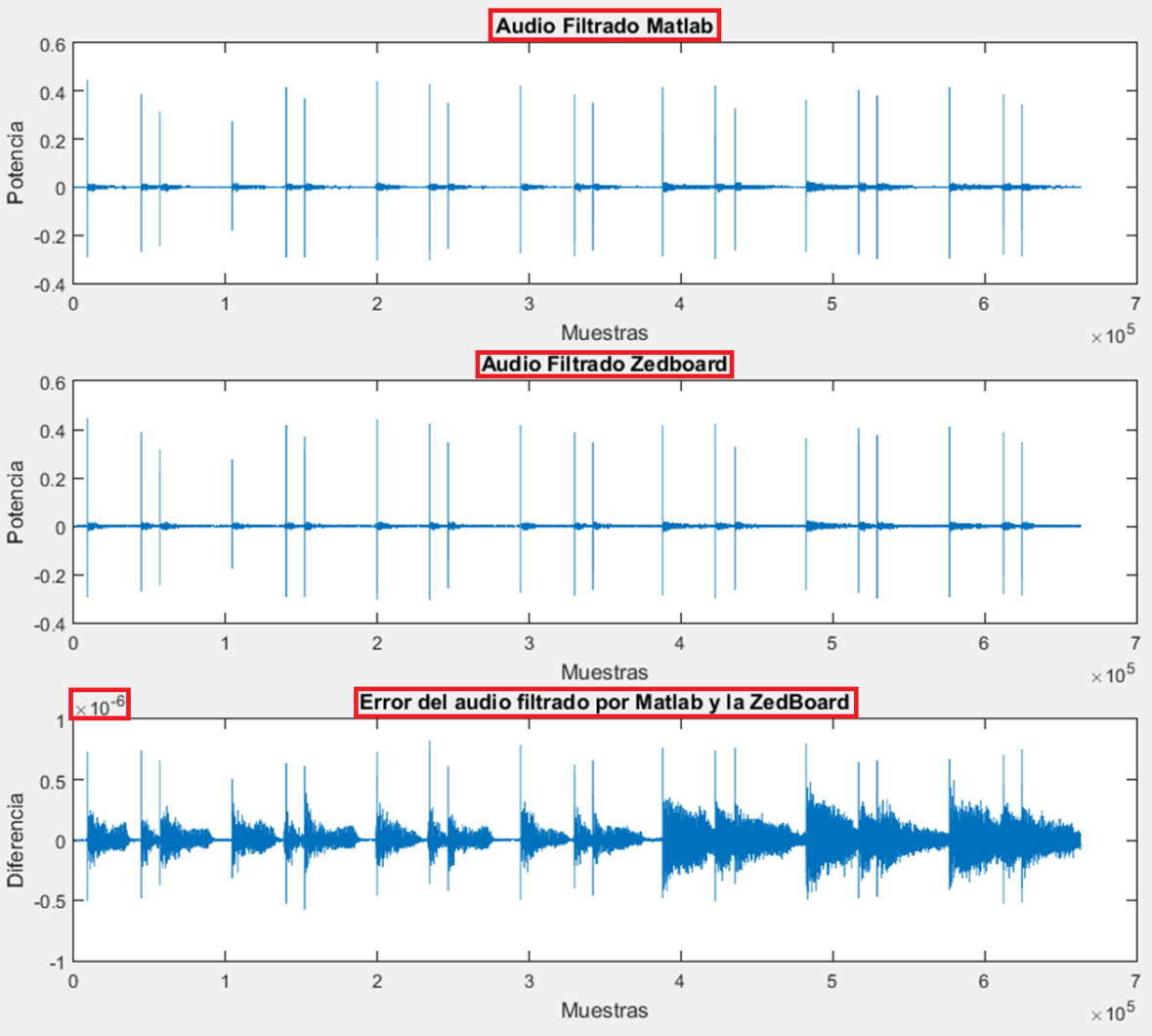

Figura 116. Grafica de resultados de comparación del procesamiento.

Finalmente se tienen ventajas y desventajas de trabajar con uno u otro medio, las cuales son:

Ventajas de trabajar en MATLAB:

- Se puede analizar gráficamente en el mismo medio, ya que el computador cuenta con pantalla para visualizar datos.
- Una comunidad muy extendida, por lo cual es fácil encontrar ayuda en foros sobre funciones y aplicaciones de MATLAB.

Desventajas de trabajar en MATLAB:

- Adquirir una licencia es muy costoso en el ámbito industrial.

- A pesar de que el software corre en la mayoría de los PC, existen unos requerimientos mínimos para que le software funcione bien como se describe en https://www.mathworks.com/support/sysreq/current_release/?requestedDomain=www.mathworks.com. Pero si se desea obtener un mejor rendimiento se deben superar esas capacidades, lo que hace que el PC sea costoso.

Ventajas de trabajar en la ZedBoard:

- Es un sistema de desarrollo más económico que un computador con especificaciones óptimas para ejecutar el MATLAB.
- Es un sistema embebido que puede trabajar en modo autónomo, es decir, funciona sin la necesidad de una conexión a internet o un software maestro.
- Se puede configurar la ZedBoard para que realice una aplicación específica, sin importar que la tarjeta sea desenergizada, al momento de volverla a energizar se ejecuta la aplicación sin necesidad de programarla nuevamente.

Desventajas de trabajar en la ZedBoard:

- No existe una comunidad muy extendida, los foros para ayuda son limitados.
- Para programar la tarjeta y crear aplicaciones se requiere de un computador con el software Vivado.
- Si la tarjeta se daña y es reparable, los repuestos son difíciles de encontrar en Colombia, por lo general se deben importar.

5. CONCLUSIONES, RECOMENDACIONES Y TRABAJO FUTURO

Se logra realizar el procesamiento digital de señales propuesto en el presente trabajo de grado a través de la tarjeta de desarrollo ZedBoard.

Se realiza la configuración para la adquisición de audio a través de la tarjeta ZedBoard y almacenamiento del mismo en la tarjeta SD.

Se implementa un algoritmo de programación que aplica un filtro digital FIR pasa alto con frecuencia de corte de 3KHz a un audio almacenado en formato de texto y guarda el audio filtrado en formato de texto.

Se implementa un *script* en el software MATLAB que hace el mismo procesamiento de señales realizado a través de la tarjeta ZedBoard.

Se compara los resultados obtenidos a través de MATLAB y la tarjeta ZedBoard logrando resultados similares como se muestra en la **Figura 116**.

Se recomienda mejorar el algoritmo de procesamiento digital de tal manera que dicho procesamiento se paralelice y se puedan obtener resultados en menos tiempo que le procesamiento por secuencias.

Es interesante implementar un algoritmo para que el procesamiento digital no se haga con datos almacenados, sino en tiempo real.

Cuando se quiera replicar el ejemplo “ADC/DAC and Digital Audio Processing” de la página *embedded centric* se aconseja trabajar con los “Ip_cores” y el código en lenguaje C “DAQ_AUDIO.c” entregados en este trabajo debido que seguir el ejemplo de la página al pie de la letra no funciona.

El presente trabajo es un insumo para estudiantes de ingeniería electrónica y en especial para los estudiantes del curso Diseño Digital. Por medio de este trabajo los estudiantes pueden basarse para realizar proyectos futuros que tienen que ver con el procesamiento digital de señales y que se puede hacer a través de tarjetas de desarrollo ZedBoard. Ya que en la actualidad el campo del procesamiento digital de señales cuenta con mucho recorrido y sirve para el desarrollo de muchas áreas de la ciencia y la ingeniería tales como la medicina, la astronomía, electrónica, etc.

REFERENCIAS

Albarado Moya, J. (2011). Procesamiento Digital de señales. [Online] Available at: http://www.ie.itcr.ac.cr/palvarado/PDS/pds.pdf [Accessed 1 Aug. 2016].

Alvarez, J. A., Lindig, K. M., & Martinez, G. (2008). Implementación de Filtros Digitales Tipo FIR en FPGA. [Online] Available at: http://www.scielo.org.mx/pdf/poli/n37/n37a12.pdf [Accessed 5 Jul. 2016].

DeFatta, L. (2000). Digital Signal Processing. New Jersey: Prentice Hall.

Docentes.unal.edu.co. (2005) Filtros digitales FIR. [Online] Available at: http://www.docentes.unal.edu.co/hmorenom/docs/diseno_dsp_2012_2/FILTROS%20DIGITALES%20FIR.doc [Accessed 1 Aug. 2016].

Embedded Centric. (2015). ADC/DAC and Digital Audio Processing. [Online] Available at: https://embeddedcentric.com/adc-dac-and-digital-audio-processing/ [Accessed 1 Jul. 2016].

Ni.com. (2016). ¿Qué es adquisición de datos? [Online] Available at: http://www.ni.com/data-acquisition/what-is/esa/ [Accessed 1 Jul. 2016].

Ni.com. (2016). Serie de fundamentos de mediciones con sensores. [Online] Available at: https://ni.adobeconnect.com/p1swy7yvg53/ [Accessed 1 Jul. 2016].

Udistrital.edu.co (2005) Filtros digitales. [Online] Available at: ftp://ftp.udistrital.edu.co/Documentacion/Electronica/Dsp/capitulo5.PDF [Accessed 1 Aug. 2016].

Xilinx.com. (2016). Zynq-7000 All Programmable SoC. [Online] Available at: http://www.xilinx.com/support/documentation/user_guides/ug585-Zynq-7000-TRM.pdf [Accessed 5 Jul. 2016].

Zedboard.org. (2014) Standalone SD card SDIO. [Online] Available at: http://zedboard.org/content/standalone-sd-card-sdio [Accessed 25 Jul. 2016].

APÉNDICE

APÉNDICE A: Programa en lenguaje C para adquisición de audio a través de la Zedboard.

DAQ_AUDIO.c.docx

APÉNDICE B: Encabeza Audio.h.

Audio.h.docx

APÉNDICE C: Programa en lenguaje C para adquisición y almacenamiento de audio a través de la Zedboard.

DAQ_AUDIO_SD.docx

APÉNDICE D: Programa en lenguaje C del algoritmo de procesamiento de audio.

FiltroFIR_SD.c.docx

APÉNDICE E: Script para procesamiento de audio por medio de Matlab.

Script_Proce_Matlab.docx

APÉNDICE F: Script para comparación de procesamiento en Matlab y Zedboard.

Scrit_Comp_Matlab.docx

APÉNDICE G: archivo “audioin.txt”.

audioin.txt

APÉNDICE H: Carpeta “Ip_cores”.

Ip_cores.7z

APÉNDICE I: Proyecto “DAQ_AUDIO”.

DAQ_AUDIO.7z

APÉNDICE J: Proyecto “FiltroFIR_SD”.

FiltroFIR_SD.7z

Institución Universitaria	INFORME FINAL DE TRABAJO DE GRADO	Código	FDE 089
		Versión	03
		Fecha	2015-01-22

FIRMA ESTUDIANTES Jeison Suarez

FIRMA ASESOR Luis Fernando Cortés Londoño

FECHA ENTREGA: 20-10-2016

FIRMA COMITÉ TRABAJO DE GRADO DE LA FACULTAD _____

RECHAZADO__ ACEPTADO____ ACEPTADO CON MODIFICACIONES_____

ACTA NO._____

FECHA ENTREGA: _____

FIRMA CONSEJO DE FACULTAD_____

ACTA NO._____

FECHA ENTREGA: _____

88

Institución Universitaria	MODALIDAD TRABAJO DE GRADO PRODUCTO OBTENIDO EN TALLERES O LABORATORIOS DEL ITM Registro de actividades y cumplimiento de horas / Talleres o Laboratorios de DOCENCIA	Código	FDE 146
		Versión	02
		Fecha	2015-09-30

Documento de identidad:	*1128391652*		
Nombre completo del estudiante:	*Jeison Julian Suarez Rios*		
Programa académico ITM:	*Ingeniería en Electronica*		
Nombre completo del Docente Asesor:	*Luis Fernando Castaño Londoño*		
Fecha de iniciación del producto (aaaa/mm/dd):	*29/02/2016*	**Fecha de terminación del producto (aaaa/mm/dd):**	*30/06/2016*
Nombre Taller o Laboratorio:	*Sistemas de control y robótica*		
Ubicación:			
Campus:	*Fraternidad*		

Fecha A	Fecha M	Fecha D	Actividad desempeñada por el estudiante	Hora ingreso	Hora salida	Total horas	Firma Laboratorista	Firma Estudiante
16	2	29	Socialización del producto a ejecutar	6:00pm	8:00pm	2	Luis Fernando Castaño L	Jeison Suarez
16	3	5	Socialización de cronograma y de plan de trabajo	10:00am	12:00pm	2	Luis Fernando Castaño L	Jeison Suarez
16	3	7	Investigacion sobre el tema del producto	6:00pm	8:00pm	2	Luis Fernando Castaño L	Jeison Suarez
16	3	12	Ambientación con las Zedboard (componentes, funcionalidades, usos, servicios...)	10:00am	12:00pm	2	Luis Fernando Castaño L	Jeison Suarez
16	3	14	Prueba de las tarjetas Zedboard con ejemplos tipicos para verificar el perfecto estado y funcionalidad de las mismas	6:00pm	8:00pm	2	Luis Fernando Castaño L	Jeison Suarez
16	3	19	Comprobación y validación del contenido del GPIO-demo (encedido y apagado de led, mediante pulsaores, switchs..)	10:00am	12:00pm	2	Luis Fernando Castaño L	Jeison Suarez
16	3	28	Estudio de guía de laboratorio propuesta en (embeddedcentric - Lab 8 ADC/DAC and Digital Audio Processing)	6:00pm	8:00pm	2	Luis Fernando Castaño L	Jeison Suarez
16	4	2	Prueba del Lab 8 de embeddedcentric con vivado 2014.4 (No funciona)	10:00am	12:00pm	2	Luis Fernando Castaño L	Jeison Suarez
16	4	4	Buscar soluciones a las fallas de la guia Lab 8 de embeddedcentric	6:00pm	8:00pm	2	Luis Fernando Castaño L	Jeison Suarez
16	4	9	Buscar soluciones a las fallas de la guia Lab 8 de embeddedcentric	10:00am	12:00pm	2	Luis Fernando Castaño L	Jeison Suarez

16	4	11	Al no encontrar solucion a la Guia Lab 8 de embeddedcentric, se opta por buscar otros ejemplos y guias para adquisicion de audio con la Zed board	6:00pm	8:00pm	2	Luis Fernando Cortes	Jeison Suarez
16	4	16	Pruebas a guia de laboratorio encontrada en Internet basada en ISE, se hace la migracion a Vivado 2014.4	10:00am	12:00pm	2	Luis Fernando Cortes	Jeison Suarez
16	4	18	Pruebas a guia de laboratorio encontrada en Internet basada en ISE, se hace la migracion a Vivado 2014.4	6:00pm	8:00pm	2	Luis Fernando Cortes	Jeison Suarez
16	4	23	No se obtiene resultados satisfactorios con la guia encontrada en internet basada en ISE, se busca posible solucion.	10:00am	12:00pm	2	Luis Fernando Cortes	Jeison Suarez
16	4	25	No se obtiene resultados satisfactorios con la guia encontrada en internet basada en ISE, se busca posible solucion.	6:00pm	8:00pm	2	Luis Fernando Cortes	Jeison Suarez
16	4	30	Se prueba conexión entre la tarjeta zedboard y el PC por terminal, se encuentra resultado satisfactorio.	10:00am	12:00pm	2	Luis Fernando Cortes	Jeison Suarez
16	5	2	Se opta por busca solucion al error de la guia Lab 8 de embeddedcentric.	6:00pm	8:00pm	2	Luis Fernando Cortes	Jeison Suarez
16	5	7	Se opta por busca solucion al error de la guia Lab 8 de embeddedcentric.	10:00am	12:00pm	2	Luis Fernando Cortes	Jeison Suarez
16	5	14	Se encuentra el error a la guia Lab 8 de embeddedcentric.	10:00am	12:00pm	2	Luis Fernando Cortes	Jeison Suarez
16	5	16	Se realizan pruebas en la tarjeta zedboard de la guia Lab 8 de embeddedcentric.	6:00pm	8:00pm	2	Luis Fernando Cortes	Jeison Suarez
16	5	21	Se realizan pruebas en la tarjeta zedboard de la guia Lab 8 de embeddedcentric. El proyecto funcionó correctamente.	10:00am	12:00pm	2	Luis Fernando Cortes	Jeison Suarez
16	5	23	Se hacen pruebas de un proyecto en vivado para almacenamiento de datos en la tarjtea SD.	6:00pm	8:00pm	2	Luis Fernando Cortes	Jeison Suarez
16	5	28	El proyecto de almacenamiento no funciona bien, se opta por buscar otras metodologias	10:00am	12:00pm	2	Luis Fernando Cortes	Jeison Suarez
16	6	2	Consulta de otras metodologias para almacenamiento de datos en SD	7:00pm	9:00pm	2	Luis Fernando Cortes	Jeison Suarez
16	6	3	Consulta de otras metodologias para almacenamiento de datos en SD	7:00pm	9:00pm	2	Luis Fernando Cortes	Jeison Suarez
16	6	7	Conexión Remota a la Board por TeamViewer	5:00pm	7:00pm	2	Luis Fernando Cortes	Jeison Suarez
16	6	10	Hacer Pruebas con Vivado 2015.3	7:00pm	9:00pm	2	Luis Fernando Cortes	Jeison Suarez

16	6	11	Se encuentra solucion al almacenamieto de datos en la tarjeta SD	10:00am	12:00pm	2	Luis Fernando Cortés	Jeison Suarez
16	6	12	Se realizan pruebas de almacanamiento de audio en la tarjeta SD	6:00pm	8:00pm	2	Luis Fernando Cortés	Jeison Suarez
16	5	13	Se busca implementacion de algortimos de procesamiento de audio.	10:00am	12:00pm	2	Luis Fernando Cortés	Jeison Suarez
16	6	17	Se opta por aplicar filtro FIR al audio almacenado	2:00pm	4:00pm	2	Luis Fernando Cortés	Jeison Suarez
16	6	18	Se crea proyecto en vivado para implementar el algoritmo de procesamiento de audio	2:00pm	4:00pm	2	Luis Fernando Cortés	Jeison Suarez
16	6	19	Se realizan pruebas del proyecto de procesamiento de audio y se realizan ajustes al algoritmo	2:00pm	4:00pm	2	Luis Fernando Cortés	Jeison Suarez
16	6	20	Se realizan algoritmos en Matlab para comparacion de procesamiento de audio	2:00pm	4:00pm	2	Luis Fernando Cortés	Jeison Suarez
TOTAL HORAS								68

Jeison Suarez

Firma Estudiante

Luis Fernando Cortés Londoño

Nombre y firma Laboratorista

Nombre y firma Profesional Universitario - Centro de Laboratorios