Instituto Tecnológico de Costa Rica
Vicerrectoría de Investigación y Extensión
Dirección de Proyectos

Informe final de proyecto de investigación
Documento I

Sistema electrónico integrado en chip (SoC) para el reconocimiento de patrones de disparos y motosierras en una red inalámbrica de sensores para la protección ambiental

5402-1360-3101

Adscrito a:
Escuela de Ingeniería Electrónica

Investigador principal:
Dr. Alfonso Chacón Rodríguez, EIE

Investigador participante:
Dr. José Pablo Alvarado Moya, EIE

12 de junio de 2014

Agradecimientos

Agradecemos a la Escuela de Ingeniería Electrónica, por proveer de los fondos, equipamiento y tiempo extra necesarios para completar varias etapas de este proyecto.

A la Vicerrectoría de Investigación y Extensión por el aporte en horas y recursos para gran parte de este proyecto.

Al servicio MOSIS de la Universidad de California del Sur por proveer de la fabricación de los prototipos de circuito integrados generados en este proyecto.

A la Vicerrectoría de Docencia y a National Instruments, por dotar al DCILab de la Escuela de Ingeniería Electrónica, del sistema PXIe-1078, usado para las pruebas de varios de los circuitos fabricados.

Al Consejo de Investigación y Extensión en ejercicio durante el año de 2013, por su comprensión para con el investigador principal por el atraso sufrido, al tener éste que ejercer la Dirección de Posgrado a solicitud expresa del señor Rector durante el periodo 2012-2013.

El desarrollo del proyecto y sus alcances no hubiesen sido posibles sin la colaboración de los ingenieros y estudiantes asistentes de grado y posgrado, cuyas tesis de licenciatura, maestría y doctorado sirvieron de base para el proyecto y fueron la fuente de datos indispensable para este informe: Ing. Roberto Pereira, Ing. Ronny García, Ing. Jorge Cárdenas, Ing. Roberto Cerdas, Ing. Roberto Molina, Ing. Frank Nicaragua, Ing. Carlos Adrián Salazar, Ing. Mauricio Carvajal, Ing. Luis Abrahams, Ing. Jairo Valverde, Ing. Dave Porras, Ing. José Ibarra, Ing. Berny Dinarte, Ing. Jordan Montero, Ing. Luis Adolfo Alfaro, Ing. Carlos Salazar, Ing. Andrey Villegas, Ing. Óscar Zúñiga, Ing. Mario Sequeira, Ing. Erick Salas, Ing. Esteban Smith, Ing. Gabriela Sáenz e Ing. Allan Zúñiga, quienes aportaron todo su conocimiento y esfuerzo en las distintas etapas de este proyecto. Además se agradece el aporte técnico del estudiante asistente del DCILab, Sr. Reinaldo Castro, y al estudiante Sr. Carlos Mata.

Resumen

Se ha concluido un proyecto de investigación y desarrollo financiado y apoyado por la Vicerrectoría de Investigación y Extensión, la Vicerrectoría de Docencia y la Escuela de Ingeniería Electrónica del Tecnológico de Costa Rica, y el consorcio MOSIS de la Universidad del Sur de California.

Este proyecto se ha enfocado en desarrollar un Sistema integrado en Chip (SoC) que sirva de nodo inteligente en una red inalámbrica de sensores, abocada a la detección y localización de cacería y tala ilegal, a través de la detección del sonido de armas de fuego y motosierras, en zonas boscosas y selváticas protegidas. Para ello se ha trabajado en el diseño y comprobación de distintas partes de la estructura de clasificación necesaria para cumplir esta función, con resultados mixtos: un sistema de acondicionamiento y adquisición de información acústica, una etapa de detección inicial, una etapa de clasificación de patrones acústicos y la red base de sensores en que se integrarían los módulos.

Se ha desarrollado y caracterizado una unidad funcional de adquisición de datos. Se han fabricado dos propuestas microelectrónicas de detectores de disparos, uno de los cuales ha superado las pruebas de caracterización eléctrica y será sometido a pruebas estadísticas de efectividad de detección de datos reales en los próximos meses. Se cuenta con un algoritmo eficiente, verificado con datos reales y realizable en hardware, para la detección de motosierras. Se ha finalizado la implementación y verificación con datos reales del sistema clasificador en una plataforma embebida y se cuenta con el software necesario para entrenar el sistema para distintos ambientes de reconocimiento de patrones acústicos. Se ha finalizado la prueba estructural en FPGA de la etapa de extracción de características de patrones acústicos del sistema anterior, y se encuentra en las etapas de verificación la etapa final de reconocimiento de patrones, para contar así con una descripción RTL fabricable en un ASIC digital de todo el sistema clasificador de patrones. En los próximos meses se irán a fabricación varias de estas estructuras que serán probadas posteriormente en el DCILab. Se cuenta con una red inalámbrica base ya para soportar los SoCs, y se ha desarrollado un protocolo ad-hoc eficiente para esta red inalámbrica.

Este informe presenta y discute los resultados obtenidos en el desarrollo del hardware y software de todo el proyecto.

Abstract

A research and development project has been completed. The project has been financed and supported by Vicerrectoría de Investigación y Extensión, Vicerrectoría de Docencia and Escuela de Ingeniería Electrónica from Tecnológico de Costa Rica, and the MOSIS Service from the University of Southern California.

The project aimed at the development of a intelligent wireless sensor network node integrated on a SoC. The network is intended to serve as surveillance system for the detection and location of poaching and illegal timbering, through the detection of the sound coming from firearms and chainsaws in protected wildlife reserves. Prototypes for different parts of the system have been designed and tested, with mixed results, including: a signal conditioning and acquisition system, an initial detection stage, an acoustic classification stage, and the wireless sensor network basic setup.

A functional signal conditioning and acquisition unit has been designed and characterized. Two versions of microelectronic gunshot acoustic detectors have been fabricated, and one of them has already passed the electric characterization, and its efficiency as a detector is to be statistically tested with real data in the next months. An efficient and hardware feasible chainsaw detection algorithm has been validated with real data. An acoustic classification system has been implemented on a commercial embedded platform and tested with real data. The system is provided with a training software tool to adapt it to different acoustic settings. A FPGA version of the acoustic characteristics extraction stage of this same system has been successfully tested. The final classification stage is under testing, in order to have a complete RTL version of the system, suitable for its integration on a digital ASIC. Some os this structures are be sent to fabrication in the next months, to be tested later at the DCILab. A basic setup of a wireless sensor network is ready, and an efficient ad-hoc communications protocol has been developed for this network.

This report presents and discusses the results obtained in the hardware and software implementation of the project's objectives.

Índice general

Índice de figuras

Índice de tablas

Índice de abreviaturas y palabras clave

ADC Convertidor analógico a digital (*Analog-Digital Converter*)

AGC Control automático de ganancia (*Automatic Gain Control*)

ASIC Circuito integrado de aplicación específica (*Application Specific Integrated Circuit*)

DCILab Laboratorio de Diseño de Circuitos Integrados (Escuela de Ingeniería Electrónica)

D2ARS Diseño y desarrollo de aplicaciones basadas en redes de sensores.

DSP Procesamiento digital de señales

FPGA Arreglo lógico programable de compuertas (*Field Programmable Gate Array*)

MOM Modelo oculto de Markov (HMM, *Hidden Markov Model*)

RIS Red Inalámbrica de Sensores

SiRPA Sistema de Reconocimiento de Patrones Acústicos

SoC Sistema en chip (*System on Chip*)

VLSI Circuito de muy alta integración (*Very Large Integrated Circuit*)

Palabras clave:
Redes de sensores, sistema en chip, sensores inteligentes, reconocimiento de disparos y motosierras, circuitos integrados, FPGA, procesamiento digital de señales.
Keywords:
Wireless sensor networks, System-on-Chip, intelligent sensors, gunshot and chainsaw recognition, integrated circuits, FPGA, digital signal processing

1. Introducción

Los avances tecnológicos de los últimos años han permitido extender el rango de aplicaciones de la electrónica y las redes de sensores con nodos inteligentes en actividades como la vigilancia de grandes zonas agrícolas y forestales, acciones de coordinación en áreas extensas en caso de desastres naturales, seguimiento de cauces de ríos, vigilancia de seguridad, en sistemas de adquisición de energías alternativas, etc.

Estas nuevas tecnologías introducen nuevos retos inducidos porque estos nodos inteligentes

- se distribuyen en áreas geográficas e interaccionan con el entorno ambiental
- generan y almacenan la información sensorial y utilizan sus conexiones inalámbricas para hacerla llegar a otros nodos
- pueden ser móviles
- están aislados y deben satisfacer sus propias necesidades energéticas para tareas de cálculo y transmisión de datos.

Este informe final presenta resultados parciales del proyecto *Sistema electrónico integrado en chip (SoC) para el reconocimiento de patrones de disparos y motosierras en una red inalámbrica de sensores para la protección ambiental*, No. 5402-1360-3101, desarrollado en la Escuela de Ingeniería Electrónica. El proyecto pretendía usar como entrada resultados de otros proyectos realizados por los investigadores en diseño de circuitos integrados, redes de sensores y procesamiento de señales acústicas. Así, se esperaba desarrollar prototipos funcionales de circuitos integrados como base de nodos inteligentes de una red de sensores de vigilancia aplicada a la protección del ambiente.

El proyecto ha tenido resultados mixtos con respecto a los resultados obtenidos y el cumplimiento de objetivos, y en este informe se analizan no solo los resultados del proyecto, sino las principales causas que a criterio de los investigadores fueron determinantes en la no consecución de lo planeado.

Este informe está estructurado de la siguiente manera: en el capítulo 2 se plantean los objetivos que buscaba este proyecto. En el capítulo 3 se resumen los antecedentes de este proyecto, junto con una guía sobre el estado del arte actual en las áreas temáticas exploradas en el proyecto. En el capítulo 4 se analizan los resultados de las distintas etapas del proyecto. El capítulo 5 resume las principales conclusiones y recomendaciones y en el capítulo 6 se rescatan los principales aportes del proyecto.

2. Objetivos

Se describen en esta sección los objetivos finales que fueron planteados al iniciar el proyecto, según constan en el comunicado VIE-1031-10, que transcribe el acuerdo tomado por el Consejo de Investigación y Extensión de la Vicerrectoría de Investigación y Extensión en su sesión No. 20-2010, artículo 12 del 25 de octubre del 2010. En la sección de análisis de resultados se evaluarán los logros de cada uno de los objetivos específicos.

2.1 Objetivo General

El objetivo general propuesto era el de construir sistemas electrónicos integrados de muy bajo consumo de potencia, que implementasen métodos y algoritmos de procesamiento de señales acústicas, aptos para la detección y clasificación de disparos de armas de fuego y motosierras. Estos dispositivos serían parte de una red inalámbrica de sensores para la protección contra caza y tala ilegales en zonas forestales protegidas. La metodología de comprobación de dichos métodos y algoritmos sería guiada mediante el desarrollo de varios prototipos de hardware y software.

2.2 Objetivos Específicos

1. Finalizar la implementación en circuito integrado de dos estructuras de detección de disparos basadas en análisis tiempo frecuencia de señales, que consuman menos de $1\,\mu$W, con una eficiencia cercana al 90% de detección.
2. Evaluar algoritmos de detección acústica de una complejidad lo suficientemente sencilla para la detección de motosierras, con el objeto de determinar aquel más apto en términos de bajo consumo de potencia versus eficiencia de detección.
3. Generar una descripción mixta, verificada lógicamente y simulada eléctricamente, de una estructura de detección de motosierras con el algoritmo escogido, con una eficiencia cercana al 90% de detección.
4. Generar una descripción verificada en HDL, y sintetizada de una etapa de clasificación digital, con el menor consumo de potencia posible y con una tasa de clasificación de un 98%, que sea posible de integrar a futuro en un sistema embebido como parte de una red inalámbrica de sensores.
5. Proponer, a nivel de bloques al menos, una arquitectura final de comunicación del SoC con el nodo de red final.
6. Generar una publicación en revistas indexadas de alto prestigio, a partir de los resultados obtenidos.
7. Documentar y publicar los avances y resultados de la investigación.

3. Antecedentes

3.1 Estudios preliminares del estado del arte

Una red inalámbrica de sensores (RIS) consiste en un conjunto de nodos con poder computacional restringido, equipados con capacidades de comunicación inalámbrica y de sensado [1, 96]. Estos nodos de sensores generalmente se diseminan sobre la región en estudio, donde cada nodo sensor es responsable de extraer datos del entorno tales como humedad, temperatura, presión, luminosidad, etc., procesando y enviando estos datos a través de uno o más nodos *sumideros*, los cuales se encargan de la transmisión de los datos al usuario final. Los retos en el desarrollo del hardware y el software en una red de este tipo se acentúan al hacerse necesarias consideraciones respecto al consumo energético, puesto que los nodos deben subsistir el mayor tiempo posible con baterías o con fuentes de energía de prestaciones limitadas, como celdas solares [70]. Esto tiene implicaciones sobre los tipos de sensores y transductores que pueden ser operados, los tiempos utilizados en la captura de los datos, los cálculos que pueden realizarse en cada nodo, y particularmente en los tiempos y rangos presentes en los procesos de comunicación.

En proyectos anteriores se ha abordado el caso general de aplicación de RIS [6, 4] y un caso particular ha sido el detección de sonidos acústicos de motosierras y disparos como indicador indirecto de actividades de caza y tala ilegal en zonas protegidas [33, 14].

Hasta el momento se han logrado implementar algoritmos en lenguajes de alto nivel [86, 82] para reconocer los patrones acústicos en una cadena de procesamiento digital que culmina con clasificadores basados en Modelos Ocultos de Markov. Los conceptos allí plasmados se han implementado en hardware en [77]. Para contrastar, se ha realizado además la implementación parcial de los módulos en software para sistemas empotrados [5]. La complejidad involucrada en los algoritmos de detección propuestos está acompañada de un inevitable consumo energético no apto para su uso en RIS con acceso restringido a fuentes de energía [96]. Todo lo anterior ha motivado continuar la investigación en dos aspectos claves:

1. Jerarquizar la detección, de modo que un primer detector con consumo energético en niveles inferiores a de $10\,\mu$W realice una monitorización continua del entorno, y únicamente en detecciones positivas a éste nivel (aunque eventualmente falsas), se desinhiba a una segunda etapa con mayores prestaciones computacionales y de precisión de la detección.

2. Optimización e integración de los prototipos implementados anteriormente a nivel de silicio para reducir el consumo posible de energía.

Ambos puntos constituían los retos tecnológicos que acercarían los resultados de proyectos anteriores cada vez más a un producto.

Este proyecto se concentró por tanto en el diseño de varios prototipos con miras a integrar

en el mediano plazo un sistema que permitiera:

1. Detectar la presencia de cazadores o taladores ilegales en zonas protegidas mediante el análisis acústico en línea de sonidos ambientales, como parte de un nodo que consuma tan poca energía como para funcionar por años sin necesidad de reemplazarse.
2. Clasificar y localizar los sonidos detectados para poder dar una alarma de alta certeza sobre la presencia de cazadores o taladores ilegales en determinada localización de la zona protegida.

El desarrollo necesario del marco teórico para desarrollar las metas anteriores ya fue expuesto con detalle en [5, 13]. En estos textos se analizaron gran cantidad de investigaciones alrededor de la teoría sobre disparos de armas de fuego y sus necesidades de estudio por razones de seguridad ocupacional, seguridad ciudadana y protección ambiental (ver [94, 84, 60, 97]), y varias implementaciones en software y hardware de métodos computacionalmente eficientes para procesamiento digital de señales de disparos, con el objetivo de detectar, clasificar y localizar los mismos o las armas que los producen [22, 54, 76, 28, 21, 46, 43, 97]. Estas soluciones se basan en algorimos complejos tales como:

- Transformada de tiempo corto de Fourier (STFT) [54].
- Transformadas de onditas para detección de la onda de choque [76] y como preprocesamiento para un algoritmo de detección por umbral adaptivo y posterior procesamiento por modelos de máxima verosimilitud y mezclas gaussianas [43].
- Detecciones de cruce por cero para detección inicial [46]
- Modelos ocultos de Markov para clasificación final tomando coeficientes obtenidos de una STFT [46].
- Filtrado de mediana para detección inicial, y clasificación por modelos de máxima verosimilitud y mezclas gaussianas [22].
- Transformadas de Fourier para la extracción de coeficientes MFCC (Mel Frequency Cepstral Coefficients) usados en clasificadores bayesianos [97].
- Localización por triangulación de sensores que proveen de información de ángulo de arribo de fuentes sonoras [28].

En la mayoría de estas propuestas se reportan eficiencias superiores al 90 % en la detección, clasificación y localización de disparos de distintas armas de fuego, con distintos rangos de alcance: entre 30-150m [22, 46] y kilómetros [28]. Sin embargo, todas son computacionalmente pesadas, no aptas para trabajar con fuentes de energía limitadas y en zonas remotas.

Por otra parte, en [13] se reportaron investigaciones dedicadas a la detección y localización de motores de combustión que pueden adaptarse a las necesidades de detección de motosierras. De esas investigaciones han derivado algunas implementaciones VLSI eficientes

en términos de consumo y eficiencia [31, 83, 44, 73], pero que no han sido comprobadas en ambientes selváticos tropicales.

3.2 Estado del arte para este proyecto

Este proyecto encaró áreas multidisciplinarias en el desarrollo de cada uno de sus objetivos. Estas áreas incluyeron: microelectrónica digital y analógica de bajo consumo, acondicionamiento y procesamiento de señales, desarrollo de aplicaciones para sistemas empotrados, redes inalámbricas y protocolos de comunicación. Dada la extrema heterogeneidad de dichos temas, se ha preferido por tanto incluir el estudio de antecedentes de cada objetivo dentro del análisis de resultados de cada uno, para facilitar el seguimiento de su desarrollo.

4. Resultados y análisis

4.1 Etapa de adquisición y acondicionamiento de señales

.

El objetivo asociado a la construcción de una etapa de acondicionamiento y adquisición de las señales tomadas de micrófonos, para las pruebas del sistema completo, se eliminó al no aprobarse el presupuesto solicitado a la VIE para la compra de los componentes necesarios, este resultaba de vital importancia tanto para las pruebas de laboratorio como las que se realicen en campo al sistema final. La unidad, descrita a nivel de bloques en la Fig. 4.1, fue finalmente construida durante un trabajo de graduación (ver [63]) con aporte de la Escuela de Ingeniería Electrónica y con fondos remanentes de algunos proyectos de esta misma Escuela en FundaTec (ver Fig. 4.5 para el prototipo ensamblado ya conectado para pruebas del sistema completo en FPGA). Esta misma unidad provee de varias fuentes de alimentación para futuras tarjetas o el circuito final de clasificación.

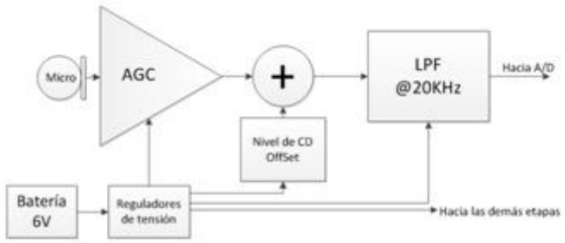


Figura 4.1: Diagrama de bloques del circuito acondicionador de señal, con control de ganancia automático. Tomada de [63]

4.1.1 Resultados de la implementación en hardware

Control de ganancia

En la Fig. 4.2 se detalla el esquemático del circuito construido para acondicionar la señal (no se muestran las fuentes de alimentación). La ganancia controlada automáticamente (AGC) se ajustó para ir desde un mínimo de 20dB a un máximo de 40dB. La máxima excursión de esta etapa es de 3,3V, para acoplarla a la etapa de adquisición. El filtro de esta unidad es el que sirve para evitar el efecto de *aliasing* a la hora de capturar digitalmente los datos. En la Fig. 4.3 se muestra el ancho de banda medido de la unidad de acondicionamiento. La ganancia AGC es provista por el circuito MAX9814 de Maxim (ver [41]).

El circuito tiene una baja distorsión asociada, mucho menor al 1%. (ver análisis de THD en 4.4). El circuito soporta entradas de hasta 100mV, típicas para los micrófonos

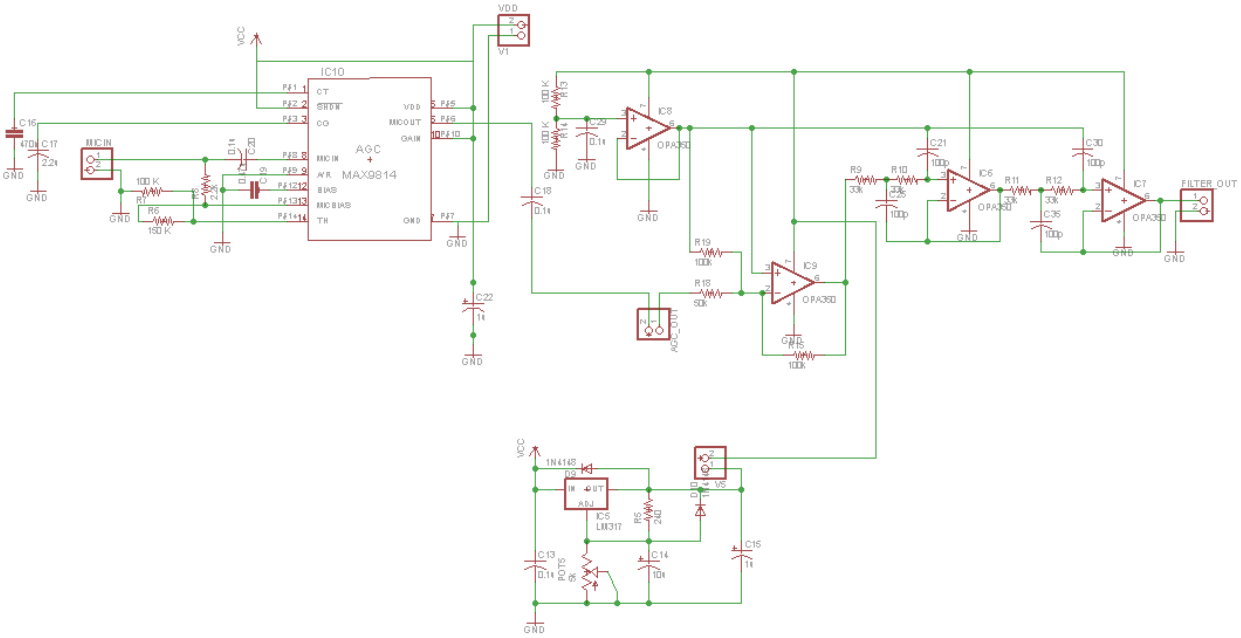


Figura 4.2: Esquemático del acondicionador de señal. El AGC es implementado por un MAX9814 de Maxim, ajustado para ir de 20dB a 40dB. Se utiliza un filtro pasabajo de cuarto orden Butterworth, configuración Sallen-Key, con un ancho de banda de 20kHz.

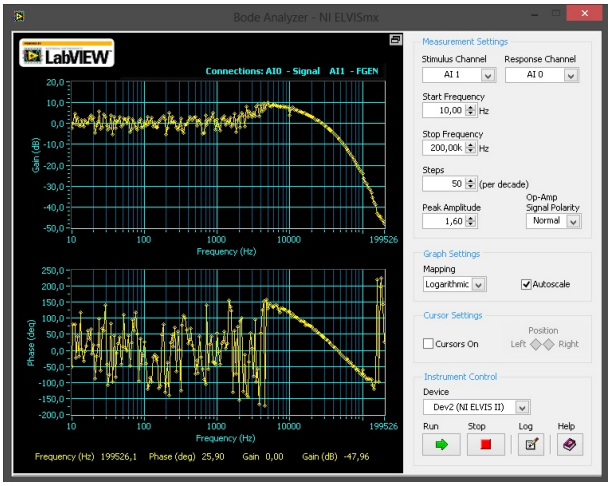


Figura 4.3: Respuesta de frecuencia del acondicionador de señal.

a usar (EMKAY MD9745APZ-F, con sensibilidad de -44dB) para presiones acústicas aproximadas de 115dB$_{SPL}$. El análisis detallado de valores de tensión y presión acústica del sistema, puede hallarse en [63, 13]. Las pruebas completas del diseño y funcionamiento del AGC están disponibles en [63].

Para la adquisición de datos se utilizó el módulo PMOD-AD de Digilent, que se incorpora a las tarjetas FPGA usadas en el desarrollo del objetivo 4. Este módulo tiene un convertidor flash ADC de 12 bits, que corre a una frecuencia de muestreo de 44,1 kHz. La señal de 12 bits es posteriormente mapeada a 16 bits (ver [17, 18]).

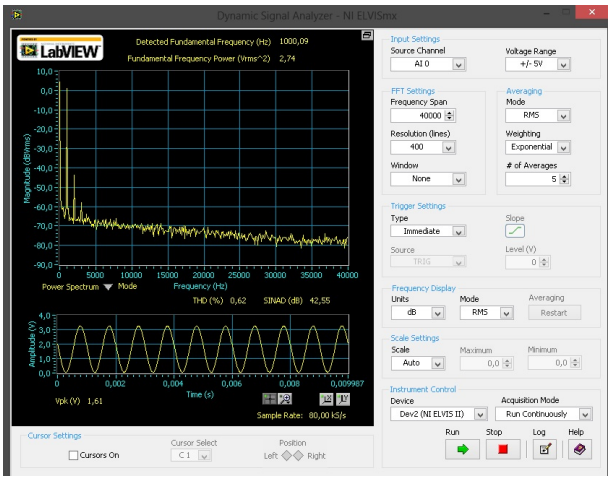


Figura 4.4: Análisis de distorsión total de armónicas, muy inferior al 1%.

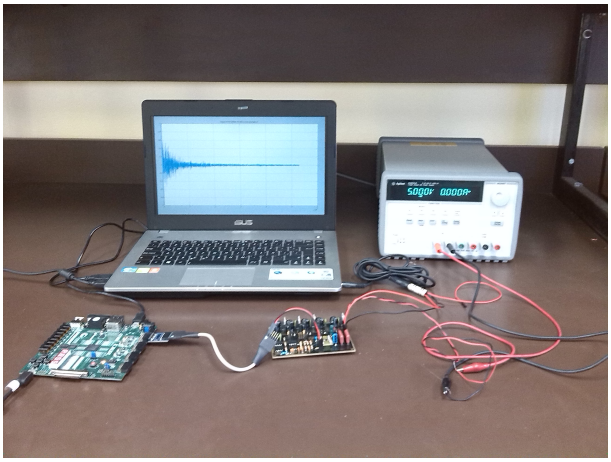


Figura 4.5: Conexión del sistema de acondicionamiento de señal con la etapa de adquisición y la FPGA para las pruebas realizadas en la sección 4.5. Ambas tarjetas pueden ser conectadas en el gabinete protector diseñado para este fin, que no se muestra en la imagen (ver [63] para imágenes del sistema completo mostrado). Este mismo circuito acondicionador se usó para conectar al sistema empotrado desarrollado en la sección 4.4, para la etapa de entrenamiento.

4.2 Estructuras integradas de detección de patrones acústicos de disparos

La unidad de clasficación que se pretende desarrollar en el objetivo 4 necesita despertarse de alguna manera, para iniciar la clasificación. Esta labor le compete a la unidad de detección, tanto de disparos como de motosierras. La misma debe proveer de una alta tasa de detección (verdaderos positivos) pero sin que la tasa de falsos positivos sea demasiado alta. En [11] se ofrece una explicación detallada de la labor que debe ejecutar esta unidad, y se proponen dos alternativas para la detección de disparos. Ambas fueron fabricadas y evaluadas, con resultados parcialmente satisfactorios que se explicitan en [11]. En este proyecto, se propuso mejorar la implementación del circuito integrado, corrigiendo

aquellos errores hallados en las evaluaciones anteriores.

4.2.1 Detección de patrones acústicos de un disparo mediante un filtro de onditas continuas

Diseño y resultados del filtro de tres bandas

El filtro de onditas continuas fabricado en [13] presentó varios problemas, entre ellos el corrimiento de los polos debido a la alta capacidad parásita en la entrada de los amplificadores operacionales de transconductancia (en adelante OTA), además de problemas de desapareamiento entre los condensadores. Esto significó un corrimiento de frecuencia en los polos del filtro que debía corregirse variando la corriente de polarización de los OTA; esto sin embargo significa un aumento en el consumo de energía deseado..

En primer lugar, se trataron de optimizar los OTA, para disminuir tanto su consumo como los efectos de su capacitancia parásita que afectaban el filtro. Como ejemplo de los varios OTA desarrollados, se muestra el trabajo hecho con uno de 192ns (para más detalles, por favor referirse al proyecto de graduación de José Ibarra [30] y a la tesis doctoral de Roberto Pereira Arroyo [64], donde se explica el proceso de optimización empleado para el diseño de los OTA. Más adelante en este informe se explicará un poco más la metodología de diseño basado en algoritmos genéticos implementada en [64]). Se utilizó la técnica empleada en [58], que consiste en realizar un escalamiento de la corriente por medio de la utilización de espejos de corriente por un factor de m=0.3 para obtener la mencionada transconductancia. Para realizar el escalamiento, se modificó el circuito del OTA de 64nS, mostrado en la figura 4.6. En la tabla 4.1, se muestran las dimensiones de los transistores empleados.

Tabla 4.1: Dimensiones de los transistores empleados en los OTA. Tomada de [30]

Transistor	W(μm)	L(μm)
M_1	0.8	7.2
M_2	0.8	8
M_3	5	2
M_5	5	2

Las medidas indicadas en la tabla 4.1 corresponden a los transistores de tamaño mínimo con que se construyeron los arreglos de transistores equivalentes del circuito de la Fig. 4.6. Los transistores M_1 de la figura 4.6, están formados por la conexión en serie de 3 transistores de tamaño mínimo; lo mismo ocurre con los transistores M_2, M_3 y M_5, que están hechos con la conexión en serie de 18 transistores para M_2 y tres transistores conectados en serie para los otros dos casos.

La figura 4.7, muestra el esquemático del OTA de 192nS, luego de agregar los transistores

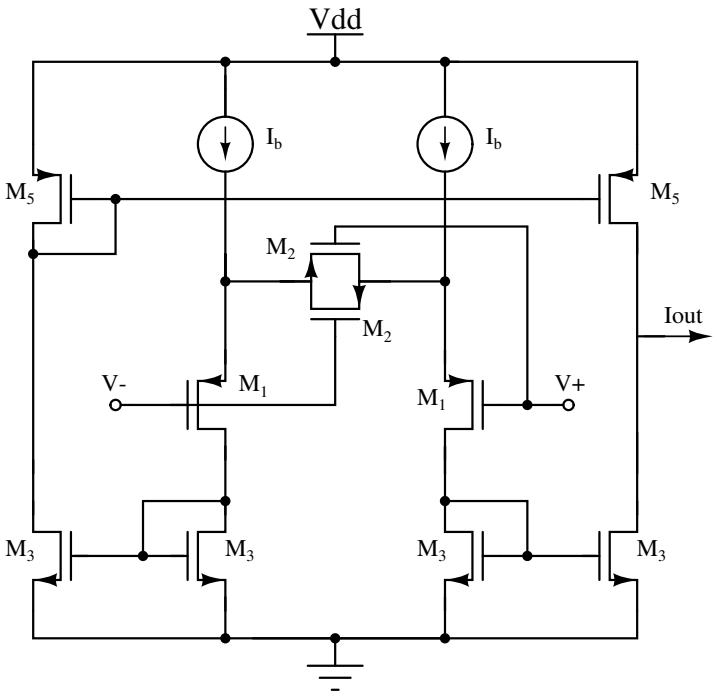


Figura 4.6: Esquemático del OTA de 64nS. Tomado de [30]

necesarios para realizar el escalamiento de corriente. Las modificaciones se realizaron en los copiadores de corriente inferiores (transistores M_3). La composición de los transistores es igual a la mencionada anteriormente. La corriente de polarización de esta unidad es de 20.16nA.

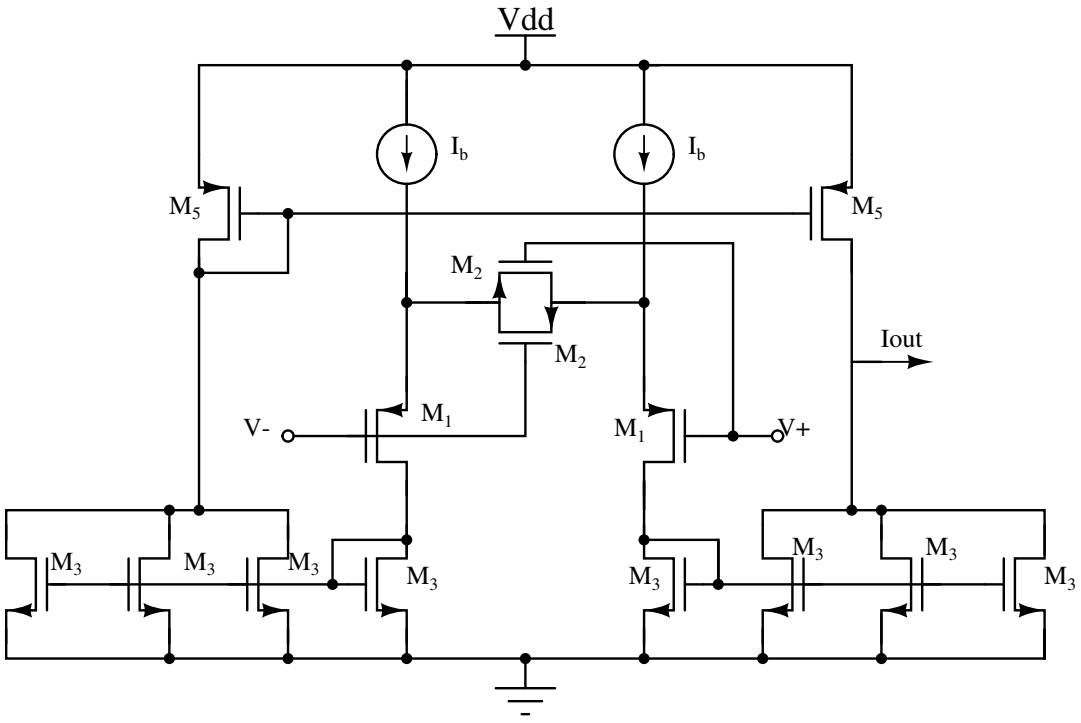


Figura 4.7: Esquemático del OTA de 192nS. Tomado de [30]

Los resultados de simulación se muestran en la Fig. 4.8, donde puede apreciarse a corriente de salida en función de la tensión diferencial de entrada en un rango de ± 1V.

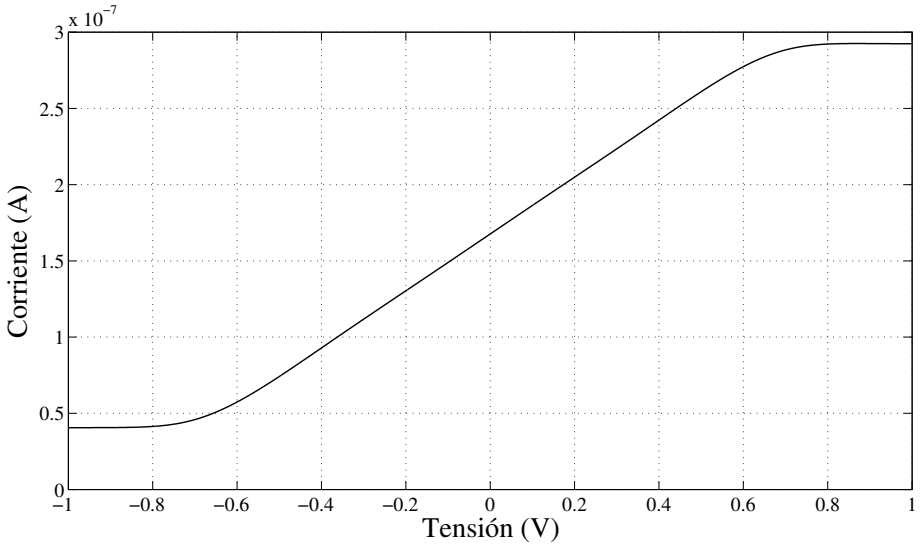


Figura 4.8: i_{out} vrs V_d para el OTA de 192nS. Tomado de [30]

La Fig. 4.9 muestra la curva de transconductancia obtenida.

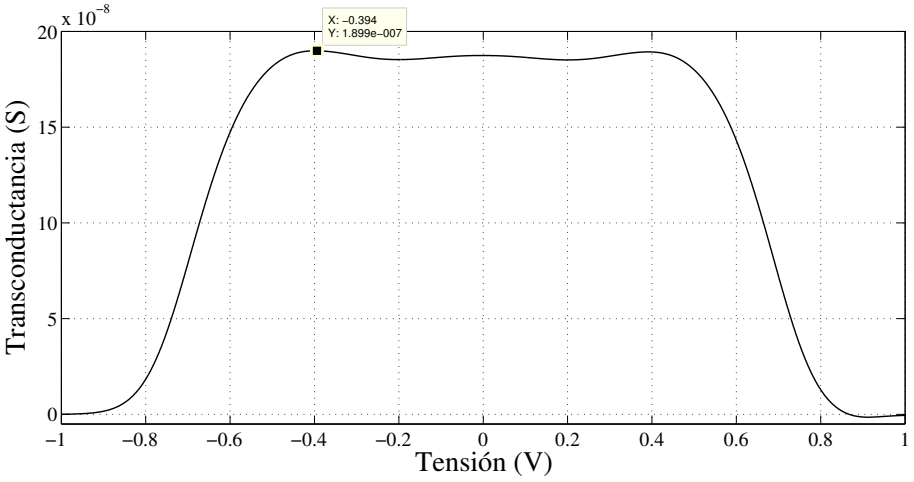


Figura 4.9: Curva de transconductancia para el OTA de 192nS. Tomada de [30]

La tabla 4.2 presenta un resumen de las características de interés del OTA de 192nS.

Tabla 4.2: Características del OTA de 192nS. Tabla tomada de [30]

OTA	G_{mmax}(nS)	C_{ent} (fF)	Potencia (nW)	ΔV(mV)	SR (mV/μs)
192	189.89	89.63	914.54	±502	0.485

Los datos mostrados en la tabla 4.2, confirman lo mencionado en [58], en donde se asevera que el escalamiento por medio de espejos de corriente no afecta el rango lineal ni la

capacitancia, ya que como se puede observar para el OTA de 192nS estas características se mantuvieron igual a los OTA diseñados en [58]. La transconductancia máxima alcanzada para el OTA fue de 189.89nS, tal y como se puede observar en la figura 4.9, lo cual implica un porcentaje de error de 1.09% con respecto al valor teórico. El consumo de potencia del OTA fue mayor a los otros tres OTA, ya que al aumentar la transconductancia los espejos de corriente provocan que se agreguen más ramas conectadas a VDD que suman al consumo.

El mismo proceso de diseño se usó para ajustar los otros cuatro OTA necesarios para implementar el banco de filtros analógicos. Los detalles de este diseño pueden verse en [30] y [64].

Los condensadores del banco de filtros analógicos debieron calcularse nuevamente debido a la modificación de los OTA. Las frecuencias de corte del banco de filtros analógicos se deben mantener como en la respuesta en frecuencia deseada según [13]. La ecuación (4.1) fue la utilizada para obtener los nuevos valores de capacitancia, G_m corresponde a la transconductancia del OTA asociado al condensador y f_c corresponde a la frecuencia de corte respectiva.

$$C = \frac{G_m}{2\pi f_c} \tag{4.1}$$

La tabla 4.3 muestra los nuevos valores de capacidad en el banco. Los valores de frecuencia de corte mostrados corresponden a la frecuencia de corte de los filtros de primer orden por los que está compuesto cada uno de los filtros; cada coeficiente del filtro está formado por dos filtros pasabaja de primer orden y un filtro pasaalta de primer orden a la salida. Para el caso del filtro de entrada, el mismo es un filtro pasaalta primer orden, seguido de una etapa de amplificación.

Tabla 4.3: Valores de los condensadores usados en el banco de filtros analógico. Tabla tomada de [30]

Condensador	f_c(Hz)	G_m (nS)	Capacidad (pF)
C1	875	64	12
C2	875	64	12
C3	437	32	12
C4	437	64	24
C5	437	64	24
C6	219	32	24
C7	219	32	24
C8	219	16	12
C9	109	16	24
C10	109	16	24

Con los OTA optimizados y los valores de capacidad calculados, se procedió a rearmar el filtro para obtener un nuevo circuito, ver Fig. 4.10. Se debe notar que el valor del condensador C10 mostrado no coincide con el reportado en la tabla 4.3Ello pues para debió equipararse la respuesta en frecuencia del filtro con la respuesta analítica que se usó para evaluar el algoritmo CWT [13].

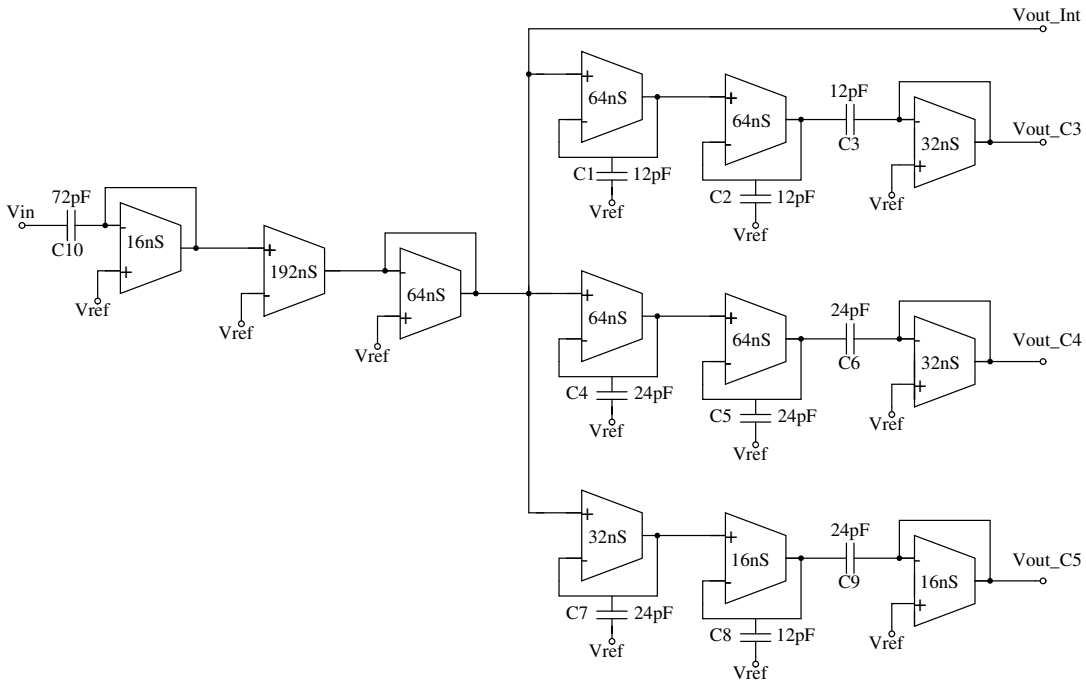


Figura 4.10: Banco de filtros analógico después de la optimización de los OTA. Tomado de [30]

Con el banco de filtros analógico optimizado se procedió a probar su funcionamiento a través del cálculo de la función de transferencia para cada uno de los coeficientes de salida del filtro. La función de transferencia del filtro para el coeficiente 3, se muestra en la ecuación (4.2). Esta función tiene un polo de segundo orden en la frecuencia de 424Hz y un polo simple en 845Hz.

$$CD_3(s) = \frac{sC3(64nS)^2}{(sC1 + 64nS)^2(sC3 + 32nS)} \tag{4.2}$$

El mismo procedimiento se siguió para los polos ya determinados en [13].

Para el caso del nodo intermedio, se tiene la función de transferencia presentada en la ecuación (4.3), que incluye el efecto de amplificación entre los OTA de 192nS y 64nS y el polo del filtro pasaalta dado por C10.

$$Vout_{NI}(s) = \frac{sC10(\frac{192nS}{64nS})}{sC10 + 16nS} \tag{4.3}$$

Conociendo el valor de la función de transferencia del filtro, se generó una simulación del modelo teórico del mismo, para comprobar su correcto funcionamiento. La figura 4.11,

muestra la magnitud de la respuesta en frecuencia del banco de filtros, y la figura 4.12 presenta la fase.

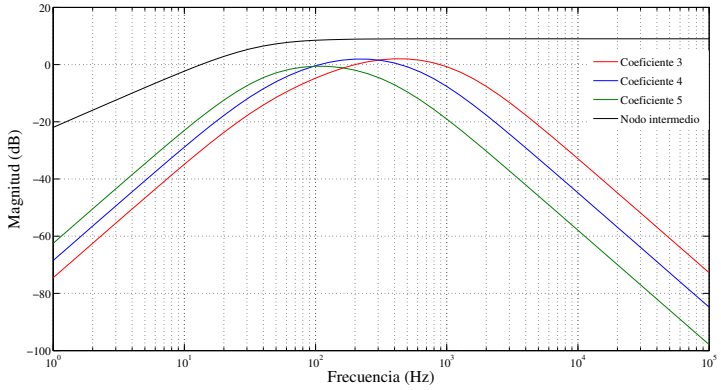


Figura 4.11: Magnitud de la respuesta en frecuencia teórica del banco de filtros analógico. Tomado de [30]

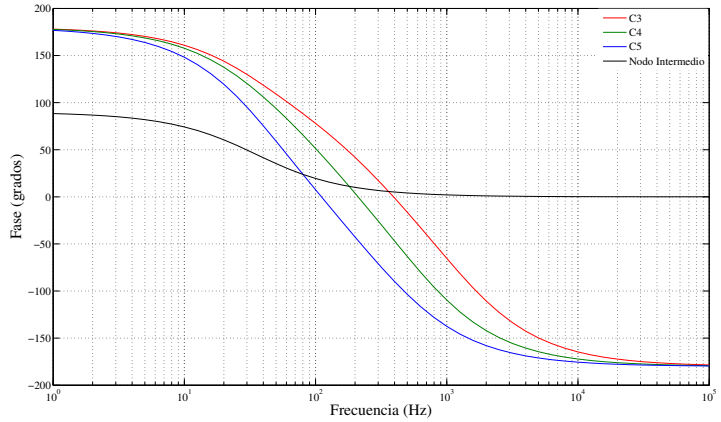
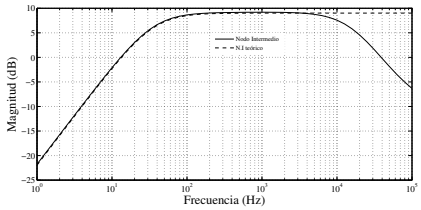


Figura 4.12: Fase de la respuesta en frecuencia teórica del banco de filtros analógico. Tomado de [30]

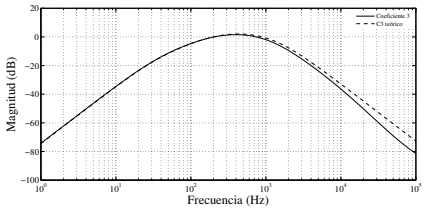
Espejos de corriente para polarizar los OTA

Todos los OTA trabajan con la misma corriente de polarización, de 20,16nA, a partir del escalamiento de una corriente de referencia (ver más adelante sobre la construcción de esta corriente de referencia).

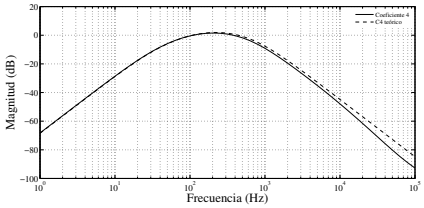
La figura 4.13 muestra la respuesta en frecuencia del banco de filtros analógico, y la figura 4.14 muestra el análisis de transitorios del mismo.



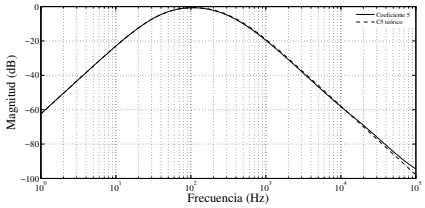
(a) Respuesta en frecuencia para el nodo intermedio.



(b) Respuesta en frecuencia para coeficiente 3.



(c) Respuesta en frecuencia para coeficiente 4.



(d) Respuesta en frecuencia para coeficiente 5.

Figura 4.13: Magnitud de la respuesta en frecuencia de los coeficientes 3, 4, 5 y el nodo intermedio. Tomado de [30]

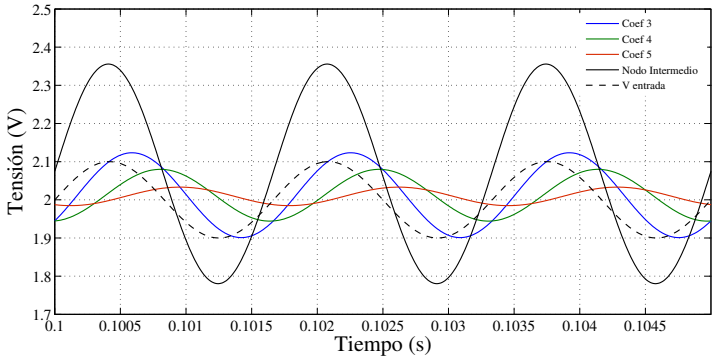


Figura 4.14: Respuesta de transitorios del banco de filtros analógico para una tensión de entrada con frecuencia de 600Hz.

Al observar los valores de magnitud de la respuesta de frecuencia de cada uno de los coeficientes, se ve que estos presentan una ligera diferencia entre sus valores máximos. En la figura 4.13, las diferencias se han reducido especialmente entre los coeficiente 3 y

4, no tanto así con el coeficiente 5. En las pruebas de los polinomios originales en [13], la evaluación se hizo sobre un filtro cúbico pasabanda. Al añadir el filtro de acondicionamiento para eliminar los efectos de CD a la entrada, este polinomio se ve afectado. El aumentar C10 mueve este polo de acondicionamiento hacia la izquierda y disminuye su efecto al menos sobre los coeficientes 3 y 4. Aunque el efecto en el polo 5 sigue siendo sustancial, se comprobó en [13] que el coeficiente 5 no es tan determinante como los otros coeficientes. En todo caso, el filtro será evaluado estadísticamente para verificar que esto sea así. Los resultados obtenidos al cambiar C10 se presentan en la tabla 4.4.

Tabla 4.4: Variaciones en la magnitud de la respuesta en frecuencia de cada uno de los coeficientes ante cambios en el valor del condensador C10. Tabla tomada de [30]

C10 (pF)	Coef. 3 (dB)	Coef. 4 (dB)	Coef. 5 (dB)	Nodo Interm. (dB)
24	1.19	0.68	-2.71	9.08
48	1.49	1.31	-1.16	9.15
72	1.54	1.48	-0.73	9.19
96	1.59	1.57	-0.49	9.20

Como se puede observar en la tabla 4.5 el valor de la capacitancia del nodo intermedio se disminuyó considerablemente – un 90% – con respecto al valor presentado en el filtro anterior. Dicha mejora evita que se traslade el polo de las primeras etapas de los filtros al sumarse esta capacidad a la de la primera etapa. Con los cambios realizados y como se puede observar en la figura 4.13, los polos de más alto valor se desplazan hacia la derecha debido a que al disminuir el tamaño de los OTA, disminuyen las capacidades parásitas de los transistores, especialmente la de la compuerta, que es la que afecta más la capacidad parásita en el nodo intermedio.

Con la optimización de los OTA hecha el consumo de potencia obtenido fue de 3.66μW, logrando por tanto una reducción del 34.29% respecto al consumo de potencia obtenido en [13], con mejores prestaciones del circuito, tales como menor área y menores capacidades parásitas asociadas. Hay que recordar que en [13], el consumo del filtro fue mayor debido al ajuste necesario para compensar los errores del circuito, como se muestra en la tabla 4.6, con lo cual el porcentaje de mejora es mayor si se toma en cuenta que al minimizarse los errores,no se va a requerir de ajustes posteriores que aumenten el consumo.

Tabla 4.5: Comparación entre características obtenidos y los valores anteriores del filtro. Tabla tomada de [30]

Característica	Inicial	Obtenido	Porcentaje de mejora (%)
Potencia (μW)	5.57	3.66	34.29
Capacitancia nodo intermedio (pF)	11.21	1.03	90.77

Tabla 4.6: Consumo del banco de filtros con alimentación de 4V. CI implementado en [13]. Tabla tomada de [30]

Circuito	Consumo sin ajuste (μA)	Consumo con ajuste (μA)
Banco de filtros	2.26	5.64

La tabla 4.7 presenta los valores de *offset* de salida para las etapas de los coeficientes y el nodo intermedio.

Tabla 4.7: Tensión *offset* de salida para el banco de filtros analógico. Tabla tomada de [30]

Etapa	Tensión de *offset* (mV)
Coeficiente 3	12.06
Coeficiente 4	12.06
Coeficiente 5	9.04
Nodo Intermedio	68.06

El *layout* de los OTA, se basó en el diseño realizado en [58], el cual tiene como referencia el método propuesto por Arnaud en [7] de apilar en columnas, de forma intercalada, los transistores de las diferentes partes del circuito. Esta técnica busca disminuir el desapareamiento entre los transistores, ya que provee de una dispersión satisfactoriamente baja, sin un impacto prohibitivo en el área[13]. Ésta técnica fue la utilizada por [58] para realizar el *layout* de los OTA. Sin embargo, se detectaron varios problemas en este trazado,pues se apilaron transistores no uniformes y de tamaño distinto lo que podría aumentar el desapareamiento entre los transistores. Fue así que para corregir esos errores se optó por dibujar los transistores para el par diferencial (M_1) y el difusor (M_2), y no utilizar los que automáticamente da el programa editor de *layout*. La figura 4.15 muestra el transistor utilizado para generar el layout del difusor.

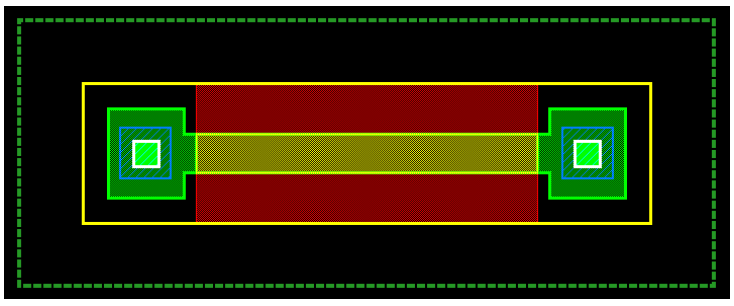


Figura 4.15: Layout utilizado para los transistores del difusor. Los transistores usados para el par diferencial son similares, pero con el respectivo cambio en las dimensiones. Figura tomada de [30]

Con los transistores corregidos, se procedió a realizar las conexiones de los mismos para

obtener el *layout* para cada uno de los OTA. El primer cambio que se hizo fue separar los transistores del diferencial y el difusor, ya que estos presentan diferentes dimensiones, como se muestra en la tabla 4.1. Ambas secciones del *layout* se conectaron usando transistores apilados en columnas, con transistores cortocircuitados en los extremos (transistores *dummy*) de cada columna. Las conexiones se realizaron usando capas de metal, y se realizaron de forma que se asegure que la corriente fluya siempre en la misma dirección. De igual forma se realizaron las conexiones para los transistores de los espejos de corriente, tanto el superior (M_5) como el inferior (M_3).

En la figura 4.16 hay un ejemplo de trazado de los OTA (las figuras están rogadas 90 grados para facilitar su inclusión en el informe).

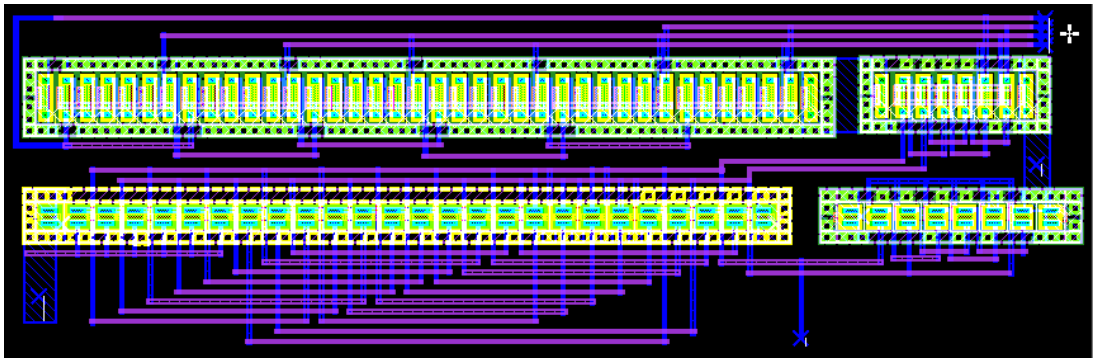


Figura 4.16: Layout del OTA de 16nS.(Trazado rotado 90 grados para facilitar su inserción en el texto). Figura tomada de [30]

La tabla 4.8 muestra las dimensiones que presentaron cada uno de los trazados.

Tabla 4.8: Tamaño físico de los OTA. Tabla tomada de [30]

OTA	W(μm)	L(μm)	Área (nm^2)
16nS	100.44	321.90	32.65
32nS	95.25	312.66	29.78
64nS	98.19	312.53	30.69
192nS	99.75	327.3	32.65

Con la optimización de los OTA, y la reducción del tamaño de los transistores, se logró una reducción en las dimensiones del trazado, que a su vez va a dar por resultado una disminución en el área final del circuito. Para dar una idea de la significativa reducción que se logró, basta con mencionar que el OTA de 137nS del circuito fabricado en [13], posee un área de 72μm^2, un área superior en casi un 58% al área de los OTA implementados en este trabajo.

El trazado de los espejos de corriente se hizo con la técnica empleada por [13]. Se usa también la técnica de apilamiento de transistores intercalados para reducir la dispersión por desapareamiento. La figura 4.17, presenta el trazado de los espejos de corriente para copiar la corriente que se va a utilizar para dar la corriente de polarización del OTA de 192nS.

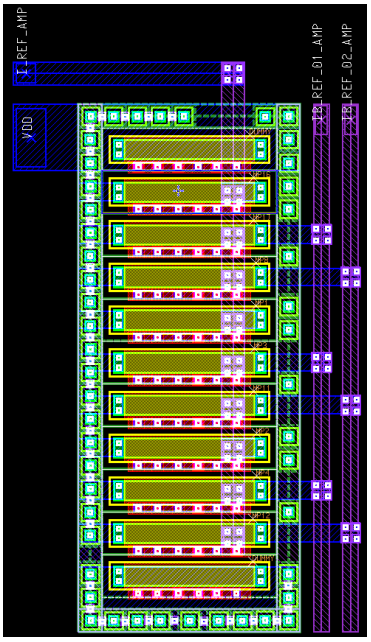


Figura 4.17: Trazado del espejo de corriente para la corriente de polarización del OTA de 192nS. Tomado de [30]

Los capacitores utilizados están hechos con dos capas de polisilicio (poly2 sobre poly),que se acomodaron en un arreglo rectangular de capacitores unitarios de 3pF. La conexión en

paralelo de estos capacitores genera las capacidades deseadas de 12pF, 24pF y 72pF; la distribución de la capacidad total en capacitores unitarios ayuda a disminuir los efectos de variabilidad[72]. Alrededor del banco de capacitores, se utilizan estructuras de capacitores cortocircuitadas para la geometría circundante de cada capacitor[13], llamadas capacitores *dummy* . Alrededor de la estructura se colocó un pozo N, conectado a la tensión VDD, además de un anillo P+ conectado a tierra.

Las dimensiones que presentan cada uno de los capacitores es de 57,3μm de lado. Los capacitores *dummy* colocados alrededor son para disminuir el desapareamiento. El banco de capacitores se puede observar en la figura 4.18.

Utilizando la *C5 Double Poly Capacitor Mismatch Calculator* para la tecnología CMOS de 0.5μm se calculó el *mistmatch* para los condensadores. Los resultados indican que el 95.5% de los capacitores van a presentar un *mistmatch* de 0.068%, y que un 68.3% de los mismos tendrán un *mismatch* del 0.034%.

La figura 4.18, enseña el *layout* completo del banco de filtros analógicos. El mismo presenta dimensiones de 1407μm de ancho, 1213μm de largo, para un área total de 1,70μm^2.

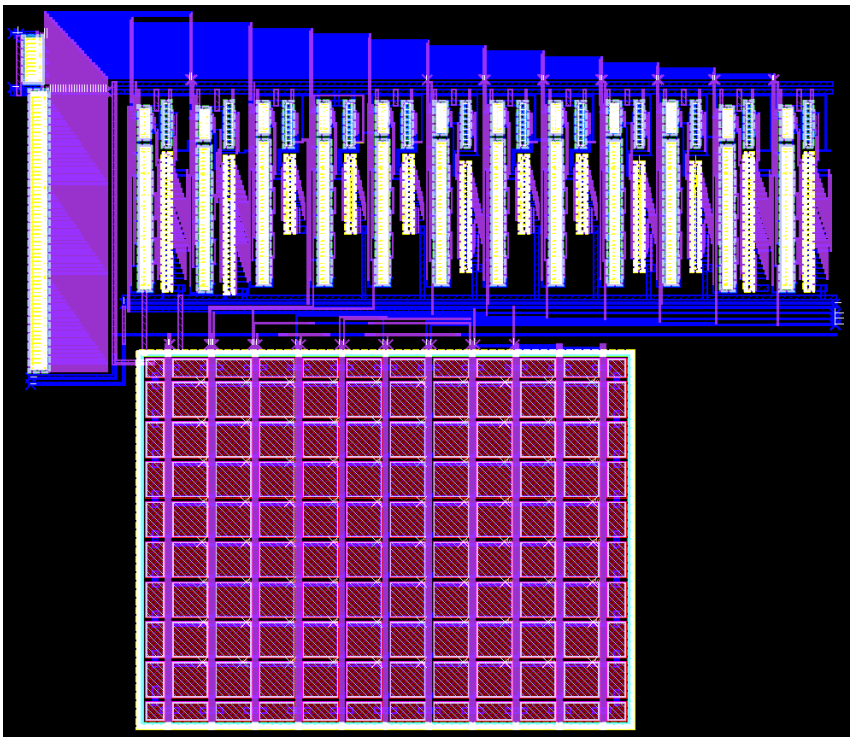


Figura 4.18: Trazado final de todo el circuito del banco de filtros analógico, en tecnología estándar CMOS de 0.5μm. Tomado de [30]

Con los datos del *netlist* generado, se pudo observar que las dimensiones de los transistores variaron con respecto a las reportadas en la tabla 4.1. Los cambios en las dimensiones se muestran en la tabla 4.9. Estos cambios presentados se deben al redondeo que produce el escalamiento del proceso, que debe hacerse en múltiplos de una constante llamada lambda, que para el proceso usado es de 0,3. Las dimensiones de los transistores, por

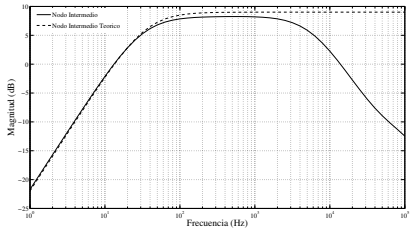
tanto, solo pueden ser múltiplos de dicho factor de escala (para más información sobre las reglas escalables, ver el sitio de MOSIS http://www.mosis.org)

En el *netlist* también se reporta el valor de capacitancia que tienen los capacitores en el *layout*: 3.06pF para cada capacitor unitario.

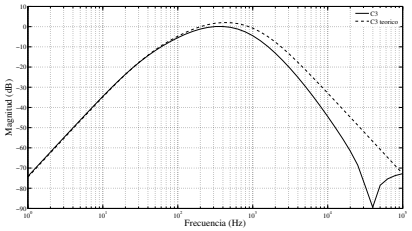
Tabla 4.9: Dimensiones de los transistores en el trazado del circuito. Tomados de [30]

Transistor	W(μm)	L(μm)
M_1	0.9	7.2
M_2	0.9	8.1
M_3	5.1	2.1
M_5	5.1	2.1

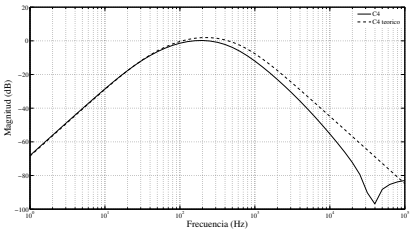
Se realizaron simulaciones *post-layout* para comprobar el funcionamiento del *layout* del banco de filtros analógico. En la figura 4.19 se presenta la respuesta en frecuencia del filtro.



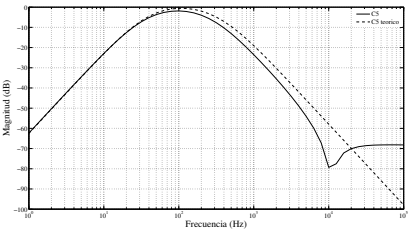
(a) Respuesta en frecuencia *post-layout* para el nodo intermedio.



(b) Respuesta en frecuencia *post-layout* para coeficiente 3.



(c) Respuesta en frecuencia *post-layout* para coeficiente 4.



(d) Respuesta en frecuencia *post-layout* para coeficiente 5.

Figura 4.19: Magnitud de la respuesta en frecuencia de los coeficientes 3, 4, 5 y el nodo intermedio, para la simulación *post-layout*. Tomada de [30]

La respuesta en frecuencia del banco de filtros analógico obtenida luego de realizar la simulación con parásitas incluidas, sufrió algunos cambios, respecto a las obtenidas en la simulación del esquemático del filtro. En la figura 4.19 se observa como la respuesta del nodo intermedio se vio afectada al correrse el polo hacia la izquierda, cerca de los

6KHz; este polo está más a la derecha respecto a la versión anterior del filtro cuyo polo se ubicó cerca de los 2,5KHz. Hay también diferencias respecto a la respuesta teórica especialmente cerca de la frecuencia de corte superior, especialmente en el coeficiente 3, en donde la diferencia de la amplitud como en corrimiento hacia la izquierda de la respuesta es notoria. A pesar de los cambios mencionadas, el filtro es totalmente funcional, ya que los cambios se presentaron en zonas que se consideran ya no están dentro del ancho de banda en el que fue diseñado el filtro. Notar además que el problema con el coeficiente 5, cuya magnitud es menor que la de los coeficientes 3 y 4 se sigue manteniendo, siendo este un problema mayor al abarcado en este proyecto, el cual debería ser corregido en versiones posteriores del circuito.

La figura 4.20, muestra la respuesta transitoria del banco de filtros analógico ante una tensión de entrada de 600Hz,y la figura 4.21 da el comportamiento que tiene el filtro cuando se varía la frecuencia de la tensión de entrada.

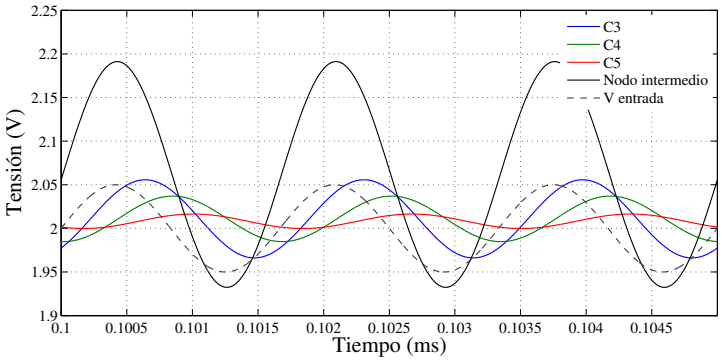


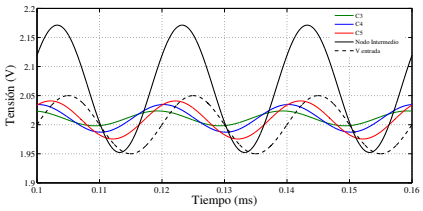
Figura 4.20: Respuesta transitoria del banco de filtros analógico *post-layout* ante una tensión de entrada de 600Hz. Tomada de [30]

La tabla 4.10, da el comportamiento que presentan las tensiones de *offset* luego de realizar la simulación con parásitas incluidas.

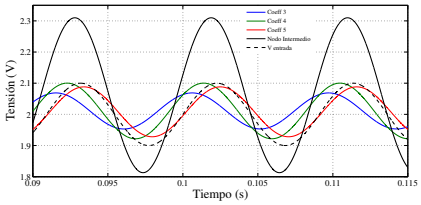
Tabla 4.10: Tensión *offset* de salida para el banco de filtros analógico para la simulación con parásitas incluidas. Tomada de [30]

Etapa	Tensión de *offset* (mV)
Coeficiente 3	10.88
Coeficiente 4	10.88
Coeficiente 5	8.14
Nodo Intermedio	61.51

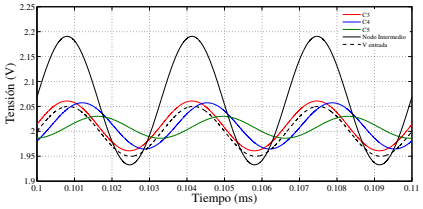
El *offset* de salida del banco de filtros analógico, en la simulación con parásitas incluidas fue menor al obtenido en la simulación del circuito esquemático. Estos resultados se muestran en la tabla 4.10, en donde se aprecia que para los coeficientes 3 y 4 hubo una



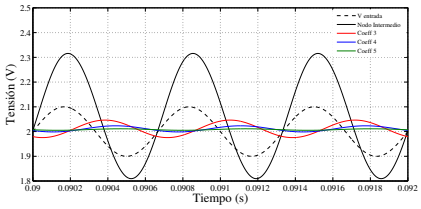
(a) Respuesta en transitorio para una tensión de entrada con frecuencia de 50Hz.



(b) Respuesta en transitorio para una tensión de entrada con frecuencia de 110Hz.



(c) Respuesta en transitorio para una tensión de entrada con frecuencia de 300Hz.



(d) Respuesta en transitorio para una tensión de entrada con frecuencia de 1500Hz.

Figura 4.21: Comportamiento *postlayout* del filtro ante una tensión de entrada con diferente frecuencia. Tomada de [30]

disminución de 1.18mV, para el coeficiente 5 de una de 0.9mV y para el nodo intermedio una de 6.55mV.

Diseño y resultados de la unidad de cálculo energético

Para efectos de minimizar aún más el consumo de potencia, y minimizar el efecto del offset en la etapa de cálculo energético desarrollada en [13], se replanteó el diseño de un comparador con un *offset* sistemático menor a $1mV$ y con un trazado que minimizara el *offset* aleatorio introducido por la variabilidad del proceso de fabricación (ver [3] y [64] para un detalle más amplio de la discusión aquí desarrollada). Se evaluaron varios esquemas de circuito bates de escoger el definitivo. Debido a que este comparador se utiliza para activar o desactivar un par de llaves de paso, otra característica importante en su diseño es el valor de *slew rate*, porque entre más rápido el comparador, más exacta será la rectificación.

La expresión 4.2.1 permitió determinar el *slew rate* de salida mínimo necesario en la configuración seleccionada. Este será igual al *slew rate* máximo de la señal aplicada a la entrada y no será fuertemente afectado por la carga conectada a la salida ya que esta va a ser la capacitancia equivalente de la compuerta de los transistores de las llaves de paso; esta equivalencia se elige con el fin de evitar retardo en el comparador debido a la velocidad de conmutación del mismo. Dado que se diseña el comparador para funcionar con una señal senoidal de entrada de $500mV$ de amplitud con una frecuencia de $10kHz$ el valor mínimo de *slew rate* buscado en este diseño será de $31,5V/ms$.

Se evaluaron dos tipos de circuito para implementar los comparadores. En la Fig. 4.22 se muestra una etapa de amplificador diferencial con transistores tipo n en la entrada con una única salida. El atributo de esta configuración es la capacidad de amplificar la diferencia de tensión entre la entrada inversora v_n y la no inversora v_p. Como resultado, el punto de transición del comparador puede hacerse independiente de las variaciones del proceso y de la tensión de alimentación.

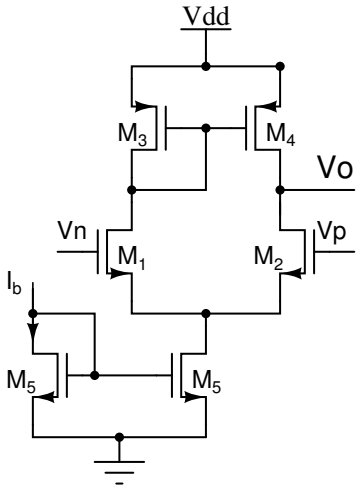


Figura 4.22: Esquema de comparador diferencial

La Fig. 4.23 muestra un comparador de dos etapas que combina las características del amplificador diferencial con las cualidades de una etapa inversora. La pobre ganancia de la etapa diferencial es aumentada por la ganancia de la etapa inversora. La salida de la etapa diferencial, que se encuentra cerca de VDD, está en la vecindad del punto de transición de la etapa inversora que le sigue. Por lo tanto, el limitado rango de salida, que es un problema para la etapa diferencial, ahora es una buena característica en la configuración de dos etapas.

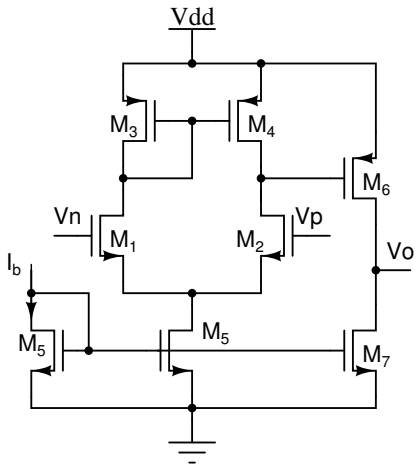
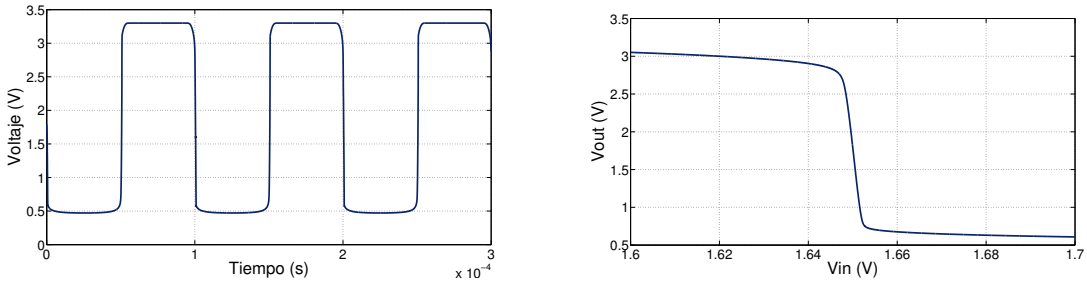


Figura 4.23: Esquema de comparador de dos etapas.

Las simulaciones de la primera configuración se presentan en la Fig. 4.24, donde se aprecian la respuesta transitoria y CD.



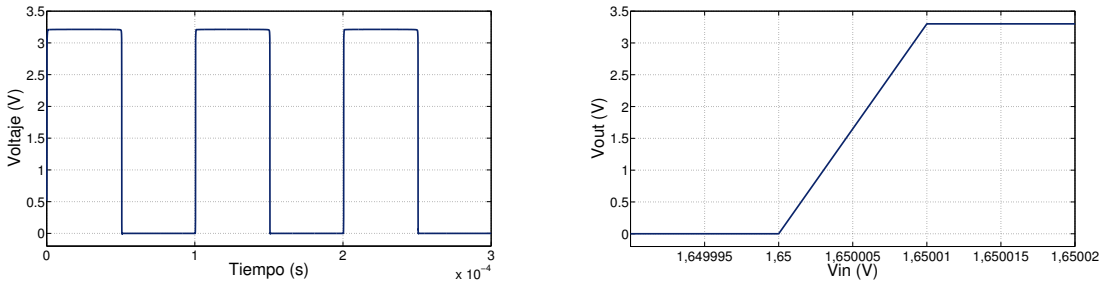
(a) Respuesta transitoria.

(b) Respuesta en CD de la tensión de salida $Vout$ versus la tensión de entrada Vin.

Figura 4.24: Salida del comparador diferencial para una señal senoidal de entrada de $0,15V$ de amplitud y frecuencia $10kHz$. Tomado de [3]

El consumo de potencia de fue excesivo: $14\mu W$.

En la Fig. 4.25 se muestra los resultados de la simulación transitoria y CD del circuito comparador de dos etapas. Esta configuración presenta un *offset* menor a $10\mu V$ y un comportamiento transitorio que cubre las características deseadas: ciclo de trabajo simétrico (cercano a 50%-50%) y una alta tasa de cambio entre los dos niveles de salida deseados; como contraste, el consumo de potencia resultó de $20\mu W$, demasiado alto para lo deseado.



(a) Respuesta Transitoria

(b) Respuesta en CD de la tensión de salida $Vout$ versus la tensión de entrada Vin.

Figura 4.25: Salida del comparador de dos etapas para una señal senoidal de entrada de $0,15V$ de amplitud y frecuencia $10kHz$. Tomado de [3]

Para la tercera opción de comparador, se escogió el de la Fig. 4.27. Este comparador tiene una buena simetría en el ciclo de trabajo, una alta tasa de cambio entre alto y bajo (y viceversa), además de un reducido *offset* sistemático; el resumen de estas características se muestran en la tabla 4.11.

En la Fig. 4.26 se muestra el diagrama de bloques de este comparador de alto desempeño (ver [3]). El comparador consiste de tres etapas: el preamplificador de entrada, una realimentación positiva o etapa de decisión, y un seguidor de salida. La primera etapa amplifica la señal de entrada para mejorar la sensibilidad del comparador y aisla la entrada del comparador del ruido por conmutación proveniente de la etapa de realimentación

positiva. La etapa de realimentación determina cual de las señales de entrada es mayor. El seguidor de salida amplifica esta información y da la señal digital de salida (más detalles de las distintas etapas de este comparador pueden hallarse en [3]).

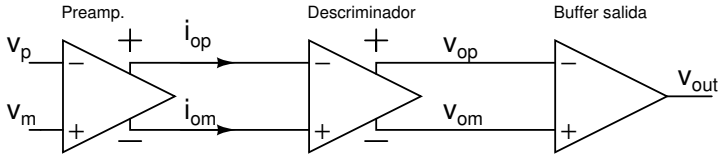


Figura 4.26: Diagrama de bloques del comparador con circuito de decisión y seguidor de salida.

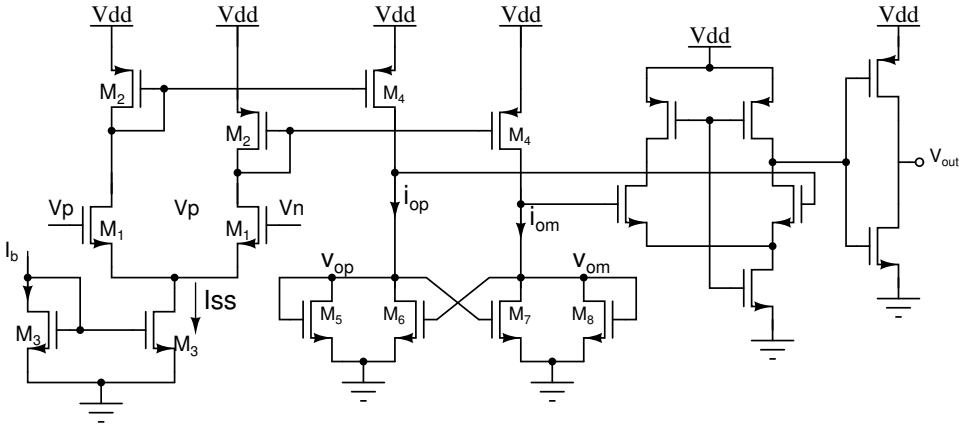
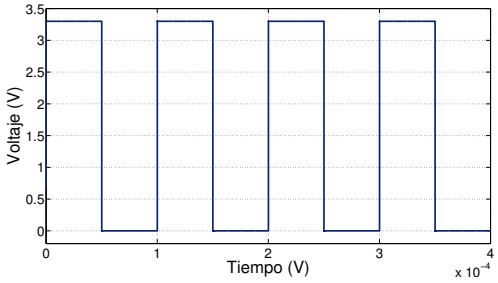
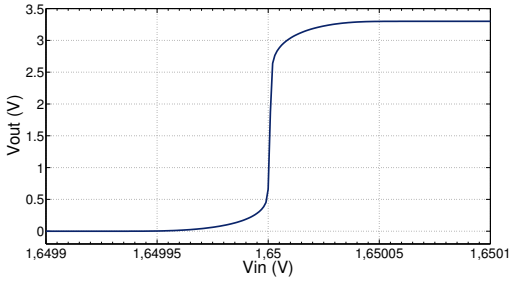


Figura 4.27: Esquema de comparador con circuito de decisión y seguidor de salida.



(a) Respuesta Transitoria.



(b) Respuesta en CD de la tensión de salida $Vout$ versus la tensión de entrada Vin.

Figura 4.28: Salida del comparador diferencial con circuito de decisión y *buffer* de salida para una señal senoidal de entrada de $0,15V$ de amplitud y frecuencia $10kHz$. Figuras tomadas de [3]

Todas las alternativas evaluadas de comparadores cumplen con el requisito de un *slew rate* superior a $31,5V/ms$. El comparador con circuito de decisión es el mejor respecto al parámetro de *offset* sistemático con únicamente $815nV$, tiene un comportamiento similar a las otras dos opciones en cuanto a ciclo de trabajo, pero tiene *slew rate*,un elevado consumo de potencia ($412\mu W$) comparado con las otras dos opciones con $20\mu W$. Si bien es cierto el comparador diferencial presenta el menor consumo de potencia, este tiene

Tabla 4.11: Características obtenidas de la simulación del comparador con circuito de decisión y buffer de salida, para una señal de entrada de $0,15V$ a $10kHz$. Tomada de [3]

Offset (nV)	**Ciclo de Trabajo** (%)	**Consumo** (μW)	**Slew Rate** $V/\mu s$
815	49,99	412,2	450

limitaciones de ganancia y la tensión del comparador no alcanza las tensiones de riel. Por otra parte, el comparador de dos etapas presenta un *offset* sistemático de $10\mu V$, no tan bueno como el del comparador con circuito de decisión, pero lo suficientemente bajo para cumplir con la necesidades dentro de la unidad cálculo y con un costo en potencia mucho menor. Por estas razones se seleccionó al comparador de dos etapas como la configuración a optimizar.

El proceso de optimización por algoritmo genético del comparador siguió el procedimiento que se detalla en [64]. Para el diseño inicial se utilizaron los parámetros de las simulaciones preliminares, que muestran en la tabla 4.12.

Tabla 4.12: Parámetros iniciales del comparador seleccionado para la optimización, para una corriente de polarización de $20\mu A$. Tabla tomada de [3]

Transistor	**Dimensión** W/L (μm)
M1	20/10
M2	20/10
M3	10/10
M4	40/10
M5	10/10

Los cuatro valores de aptitud deseados para el comparador y alimentados a la herramienta de optimización se presentan en la tabla 4.13. El *slew rate* se calcula según

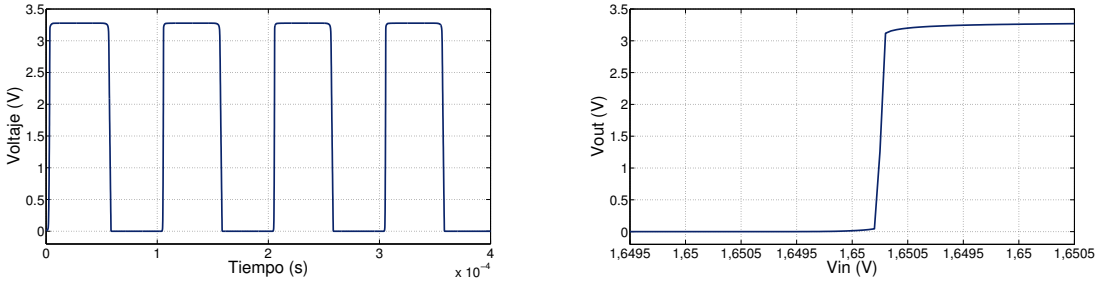
$$SR_{(i)} = dV_{out(i)} = \frac{V_{out(i+1)} - V_{out(i-1)}}{2d} \tag{4.4}$$

donde i es la posición en la que se evalúa la derivada, $V_{out(i-1)}$ y $V_{out(i+1)}$ son los valores de tensión anterior y posterior al evaluado, y d es el tiempo entre dos muestras que según lo configurado en el ambiente de simulación es de $10\mu V$. Se obtiene la máxima tasa de cambio de bajo a alto y viceversa y de esos dos valores se escoge el menor como el valor de *slew rate* a reportar para valor de aptitud.

Luego de correr la optimización se obtuvo un comparador que cumple con las especificaciones. Los resultados de la simulación tanto transitoria como CD se muestran en la Fig. 4.29.

Tabla 4.13: Valores de aptitud para el comparador. Tomada de [3]

Especificación	Valor de aptitud
slew rate	$\frac{dV_{out}}{dt}$
Ib	$\frac{1}{Ib}$
consumo	$\frac{1}{consumo}$
offset	$\frac{1}{offset}$



(a) Respuesta Transitoria

(b) Respuesta en CD de la tensión de salida $Vout$ versus la tensión de entrada Vin.

Figura 4.29: Salida del comparador con los parámetros obtenidos de la optimización, para una señal senoidal de entrada de $0,15V$ de amplitud y frecuencia $10kHz$.Figuras tomadas de [3]

Se utilizaron los parámetros obtenidos de esta primera optimización para construir un nuevo comparador que se encuentre compuesto por bloques de transistores, de manera tal que durante el proceso de elaboración de trazado se puedan aplicar técnicas de apareamiento para disminuir efectos de variabilidad (ver Fig. 4.30).

Dos inversores extra se colocaron a la salida del circuito de la Fig. 4.23 con el fin de obtener una salida dual y aumentar el *slew rate* de salida del comparador según lo propone [2].

Cada uno de los transistores del circuito de la figura 4.23 se sustituyen por bloques de arreglos en serie o paralelo con la intención de poder intercalar cada uno ellos de manera apilada durante el proceso de elaboración del trazado, tal como se aprecia en la .

La figura 4.31 muestra la composición de cada uno de los bloques del comparador apilado con los respectivos parámetros que se introducen dentro de la herramienta de optimización.

Para la preselección de comparadores, se buscaron aquellos casos cuyo *offset* sistemático fuera menor a $1mV$.

El segundo criterio de selección es una combinación entre la corriente de polarización y el consumo de potencia. El valor de *slew rate* no fue necesario utilizarlo como criterio de selección debido a que todas las configuraciones presentaban características semejantes (cerca de $400V/\mu s$) y mucho mayores (en tres ordenes de magnitud) que el valor mínimo necesario de $31,5V/ms$.

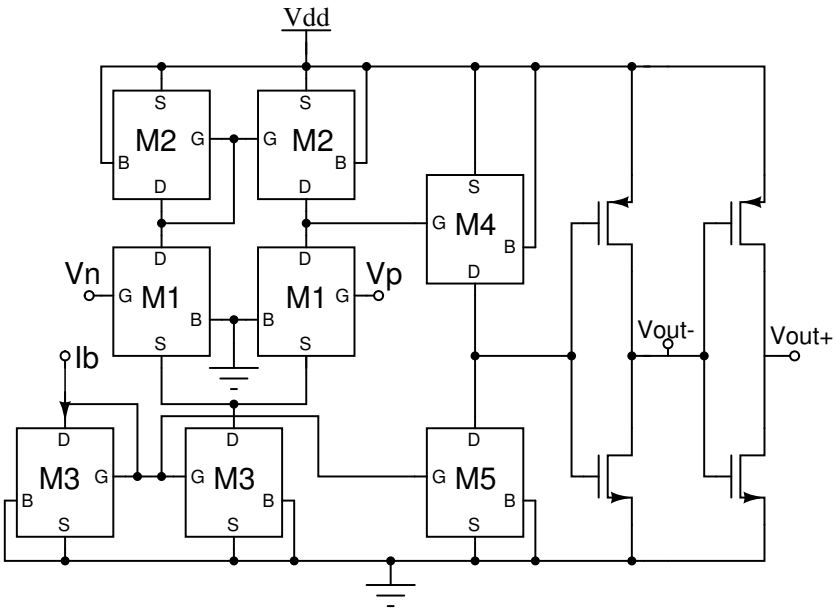


Figura 4.30: Esquema de comparador de dos etapas apilado, los transistores de los inversores son $3/0,6$ para los tipo p y $1,8/0,6$ para los tipo n. Tomada de [3]

Además, se introdujo un tercer criterio de selección que es el ciclo de trabajo del comparador. La asimetría de la tensión de salida del comparador produce diferencia en la tensión pico de dos semi-ciclos consecutivos de la salida del rectificador. Este criterio es determinante junto con el *offset* sistemático para la configuración a escoger.

Se simularon varias de las configuraciones óptimas y dentro de ellas se escogieron la cuatro mejores; el resumen de las características de estos comparadores se muestra en la tabla 4.14.

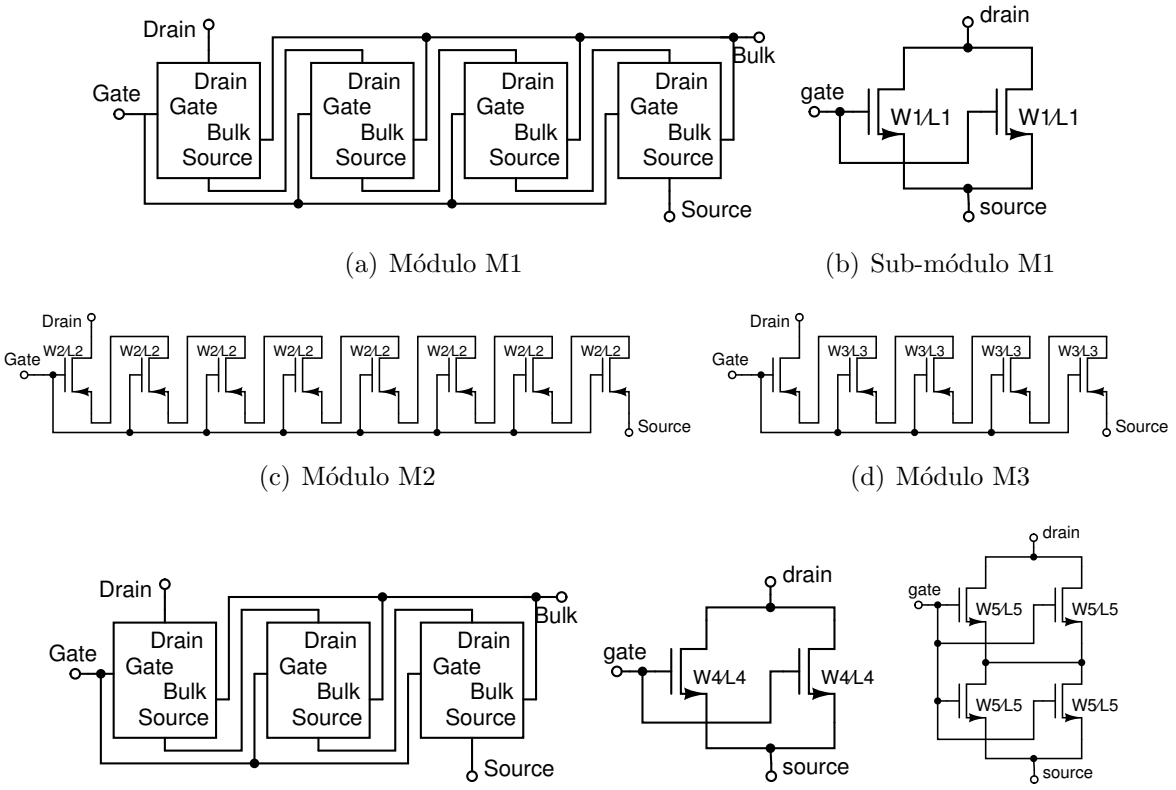
Tabla 4.14: Características de los mejores comparadores apilados obtenidos con la optimización. Tomada de [3]

Configuración	**Ciclo de trabajo (%)**	**Offset (μV)**	**Consumo (nA)** @$3,3V$
1	48,98	25	34
2	48,84	35	28
3	49,05	25	30
4	48,90	45	28

La configuración escogida fue la tercera. El resumen de las características para este se presentan en la tabla 4.15.

Los valores de los parámetros obtenidos por el optimizador para la configuración seleccionada se presentan en la tabla 4.16.

En la Fig. 4.32 se muestran tanto la respuesta transitoria como de CD del comparador seleccionado. Se aprecia la simetría de los semi-ciclos positivos y negativos, con excepción



(a) Módulo M1 (b) Sub-módulo M1

(c) Módulo M2 (d) Módulo M3

(e) Módulo M4 (f) Sub-módulo M4 (g) Modulo M5

Figura 4.31: Estructura de los bloques del comparador apilado.Se utilizan configuraciones serie-paralelo para aprovechar las ventajas en términos de r_o y de apareamiento descritas en [7]. Tomada de [3]

Tabla 4.15: Características del comparador apilado seleccionado de la optimización. Tomada de [3]

Offset (μ**V**)	**Ciclo de Trabajo (%)**	**Ib** (nA)	**Consumo** (nA) @$3,3V$	***slew rate*** $V/\mu s$
25	49,05	14,5	30	411,5

el primer ciclo que introduce un retardo de $11\mu s$ en la tensión de salida.

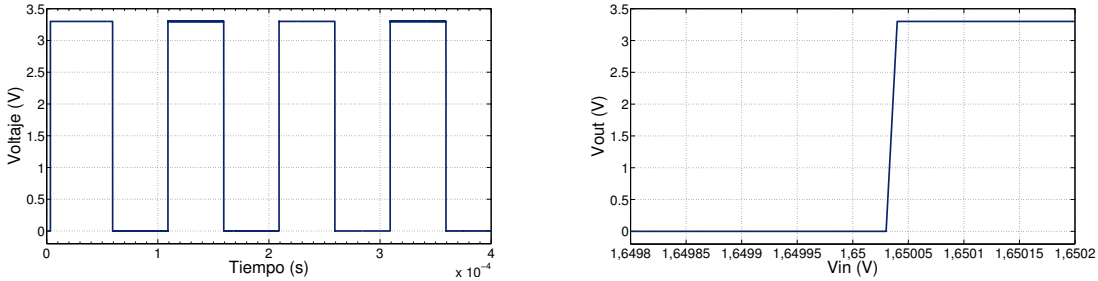
Una vez desarrollado el comparador, se pudo diseñar el resto de la unidad de cálculo. El comparador de dos etapas con salida dual se incorporó en el circuito rectificador de la Fig. 4.33. Para obtener la salida de la rectificación como una tensión se utilizó un OTA con una transconductancia de $64nS$ para la conversión de corriente a tensión. Los resultados se muestran en la Fig. 4.34 y el resumen de la características obtenidas se brindan en la tabla 4.17.

Los amplificadores de transconductancia utilizados en el rectificador son los diseñados en [58].

El bloque de copiado de corriente se construyó mediante un espejo de corriente PMOS

Tabla 4.16: Parámetros del comparador apilado seleccionado de la optimización. Tomada de [3]

Parámetro	Valor
L1	9,6 μm
W1	3,7 μm
L2	6,4 μm
W2	3,2 μm
L3	9,6 μm
W3	4,1 μm
L4	1,2 μm
W4	0,8 μm
L5	6,1 μm
W5	4,1 μm
Ib	14,5 nA



(a) Respuesta Transitoria

(b) Respuesta en CD de la tensión de salida $Vout$ versus la tensión de entrada Vin.

Figura 4.32: Salida del comparador seleccionado, para una señal senoidal de entrada de $0,15$ V de amplitud y frecuencia $10kHz$.

según [13]. La suma de energías se hizo mediante espejos cascodo (ver figura 4.35) a la salida de cada rectificador, unidos en un mismo punto. Un OTA de $64nS$ se usó como conversor corriente-tensión la relación de transconductancia mantiene las señales dentro de los rangos dinámicos de los transconductores sin exceder su rango lineal.

La figura 4.36 muestra la respuesta transitoria de la unidad de cálculo completa ante tres señales senoidales de entrada idénticas, lo que permite observar la simetría entre los semiciclos de la señal rectificada de salida, indicador de la reducción del *offset* sistemático.

Las especificaciones medidas para la unidad de cálculo completa se presentan en la tabla 4.18.

Cuando se prueba el rectificador con una señal de $875Hz$, el retardo de $11\mu s$ es menor que el 1% del periodo de la señal. La distorsión introducida revela potenciales errores en

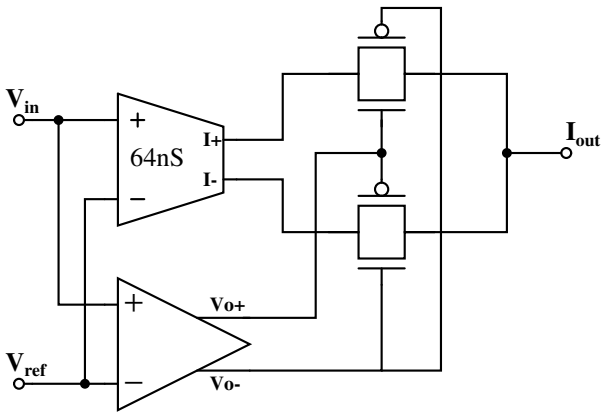


Figura 4.33: Esquema del circuito rectificador de corriente.

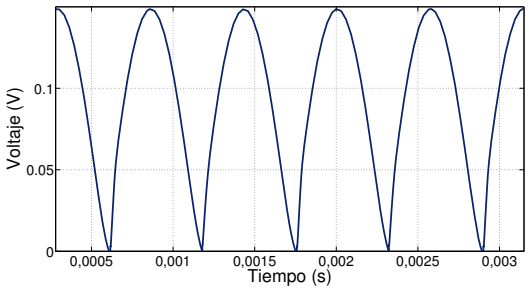


Figura 4.34: Simulación del rectificador con salida en tensión, para una entrada de $0,15V$ a $875Hz$, la linea punteada representa la curva ideal esperada. Tomada de [3]

el cálculo real del *slew rate* del circuito (que no se apreciaron a su debido tiempo), que luego serían verificados en las mediciones. La tensión de salida es una copia rectificada de la señal de entrada tal como se muestra en la figura 4.34. Aquí, el *offset* sistemático es de $25\mu V$ con un consumo de potencia de $762,3nW$ para una corriente de polarización de $14,5nA$ para el comparador y $20,2nA$ para los transconductores.

La simulación de la unidad completa revela un *offset* sistemático de la unidad de cálculo completa de $332\mu V$. Esto es apenas un $0,22\%$ de la señal de entrada con la cual se realizaron las pruebas. Además, el consumo de potencia medido para esta implementación es de $1,94\mu W$, siendo este un 32% del consumo obtenido por [7], superando con creces la reducción del 50% planteada dentro de los objetivos.

Para las pruebas hechas con $4V$ de alimentación (ver tabla 4.18) se obtuvo un *offset* sistemático de $331,8\mu V$ (semejante para $3,3V$). El consumo simulado es de$2,384\mu W$, un $8,76\%$ del consumo obtenido por [13] y claramente supera por mucho los $13,6\mu W$ planteados como los objetivo.

Si bien las simulaciones en esquemático arrojan resultados satisfactorios que cumplen con todo lo planteado en los objetivos; se realizaron simulaciones post-trazado para verificar posibles desviaciones. Los resultados se muestran en la Fig. 4.37, donde se aprecia una variación en el ciclo de trabajo del comparador de 40%. Este comportamiento anómalo

Tabla 4.17: Características del rectificador completo. Tomada de [3]

Offset (μV)	**Consumo** (nA) @$3,3V$
25	231

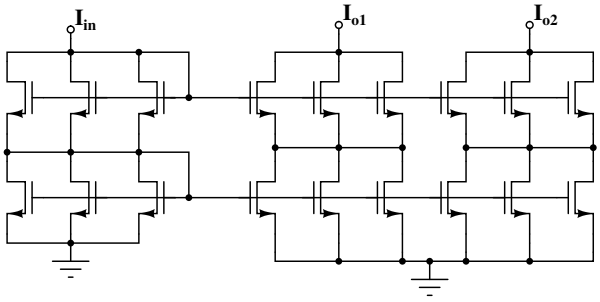


Figura 4.35: Esquema del circuito de copiadores cascodo utilizado para la suma de corrientes, todos los transistores de 3/8. Tomada de [3]

provoca deformación en la rectificación y error en la estimación de la energía.

Esta disminución en el ciclo de trabajo se debe al aumento en el tiempo de transición de alto a bajo del comparador de dos etapas (ver Fig. 4.38), ocasionado por el cambio en las dimensiones los transistores del bloque M5 para intercalarlos con los del bloque M3. Además el *layout* del bloque M3, por estar compuesto de muchos transistores (15 para cada bloque), quedó como una columna de más de $250\mu m$ de alto y con estas dimensiones las variaciones por gradientes de dopado, grabado y chorreo de las distintas capas significarán transistores muy distintos entre sí.

Se disminuyó el número de transistores del bloque M3 de quince a cinco tal y como se muestra en la Fig. 4.31 y se rehizo el trazado del bloque M5 y M4 con una configuración interdigitada para seguir con la consigna de obtener una implementación en la que se minimice el offset aleatorio. El esquema del *layout* del comparador con las modificaciones se muestra en la Fig. 4.39.

Luego de los cambios realizados se simuló de nuevo el comparador, obteniéndose el comportamiento de la Fig. 4.40 donde se aprecia la simetría del ciclo de trabajo. El resumen de las características *post-layout* se muestra en la tabla 4.19.

Al implementar el diseño en *layout* del comparador dentro del rectificador se observa que el *delay* de $11\mu s$ se mantiene y que igualmente que en la implementación en esquemático es influyente en las pruebas con una señal de entrada de $10kHz$ pero despreciable cuando la señal tiene una frecuencia de $875Hz$ (ver Fig. 4.41).

El detalle de los trazados del resto de las secciones puede hallarse en [3]. En la Fig. 4.42 se observa la simetría de los semi-ciclos rectificados.

La tabla 4.20 resume el comportamiento del circuito.

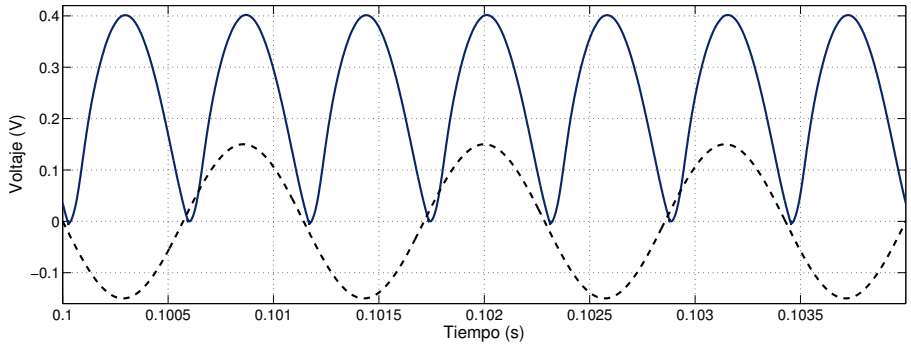


Figura 4.36: Simulación en esquemático de la tensión de salida de la unidad de cálculo completa, para tres entradas senoidales idénticas de 0.15 V a 815 Hz. Tomada de [3]

Tabla 4.18: Características de la unidad de cálculo completa obtenidas de la simulación del esquemático, para tres entradas senoidales idénticas de $0,15V$ a $815Hz$. Tomada de [3]

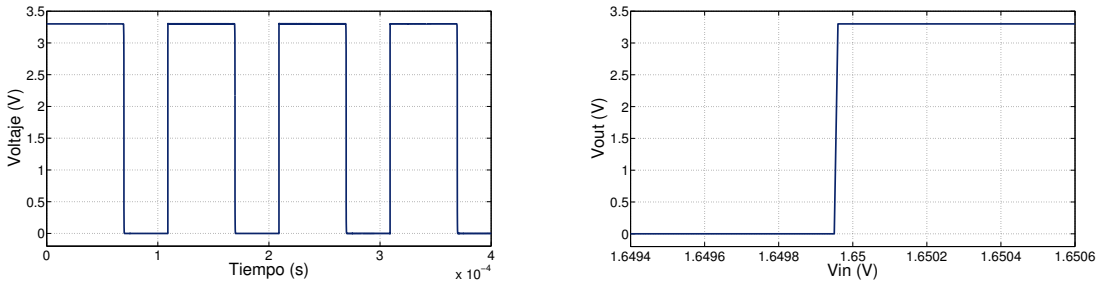
Alimentación (V)	Offset (uV)	Consumo (nA) @Ib =14,5 nA
3,3	332	587,9
4	331,8	596,1

Las simulaciones con la integración del *layout* de todos los bloques (excepto los OTAs de salida dual) se muestran en la figura 4.42. El *offset* sistemático fue de $548\mu V$, lo que representa un $0,14\%$ de la tensión de salida rectificada ($400mV$ pico) y un $5,5\%$ del offset obtenido por [13] (un 55% del planteado en los objetivos). El consumo de potencia obtenido fue $1,995\mu W$ lo cual representa un $33,3\%$ del consumo de la unidad diseñada en [13], para una alimentación de $3,3V$.

Diseño y resultados de la unidad corriente de polarización

La fuente de corriente auto-polarizada propuesta se basa en el diseño de topología simétrica de *SBCS* presentando por [47]. Al ser una configuración auto-polarizada no presenta problemas de *arranque* gracias a las tensiones *PTAT*. El diseño de este bloque de polarización a partir de únicamente *MOSFETs* operando en inversión moderada y débil lo convierte en un circuito de ultra-baja potencia. Requisitos de corriente de referencia, sensibilidades, áreas y otros aspectos concernientes a fuentes de corriente se analizan con detalle en [20] y [64].

Se implementó la fuente de corriente *SBCS* de la Fig 4.43, debido a la estabilidad de la topología simétrica (interruptor conectado al nodo $V_{X(M.I)}$). La topología simétrica (figura 4.43) emplea espejos de corriente *pMOS* de ganancia unitaria, motivo por el cual se mejora el *matching* o *apareamiento* de la estructura. Además espejos de corriente de



(a) Respuesta transitoria

(b) Respuesta en CD de la tensión de salida $Vout$ versus la tensión de entrada Vin.

Figura 4.37: Simulación *post-layout* del comparador seleccionado, primera implementación, para una señal senoidal de entrada de $0,15V$ de amplitud y frecuencia $10kHz$. Tomada de [3]

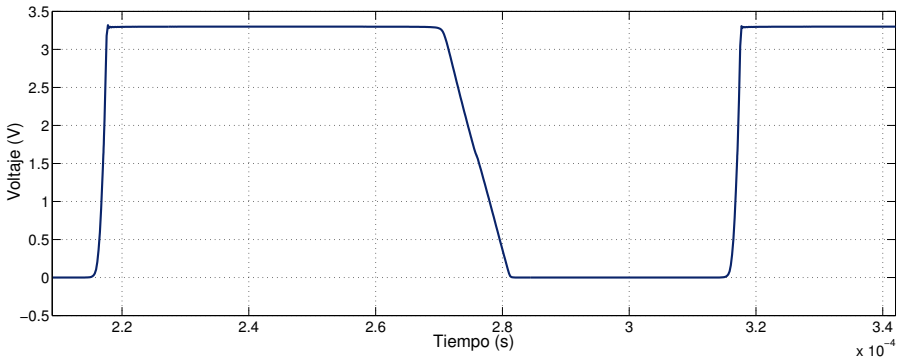


Figura 4.38: Tensión de salida del comparador antes de los inversores. Tomada de [3]

ganancias elevadas degradan la eficiencia de potencia.

En este documento se hará referencia a los transistores M_8 y M_9 de la figura 4.43 como M_X puesto que han sido definidos como equivalentes; de igual manera se hará referencia a los transistores tipo $pMOS$ (M_5, M_6, M_7, M_10 y M_{11}) como transistores M_P.

La polarización de los transistores en inversión débil garantiza un menor consumo de potencia y por ende un menor área de silicio. El diseño de la fuente se basa en el Modelo ACM, de manera tal que se dimensionan los transistores con base a una corriente de referencia y un nivel de inversión definido. A pesar de la validez del *Modelo ACM* para todas las regiones de inversión, idealmente se desea diseñar una fuente con transistores que operen solo en inversión débil *(WI)*, o inclusive en inversión moderada *(MI)*. La operación de transistores en inversión fuerte *(SI)* requiere voltajes de alimentación altos para el arranque y transistores sumamente largos. Mayores dimensiones en los transistores se traducen en mayor área de silicio.

Las ecuaciones (4.5) a (4.7) pertenecientes al desarrollo del *MOSFET auto-cascodo* se emplean en la metodología de diseño de las fuentes *SBCS*.

$$i_{f_1} = i_{f_2}\left[1 + \frac{S_2}{S_1}\left(1 + \frac{1}{N}\right)\right] \tag{4.5}$$

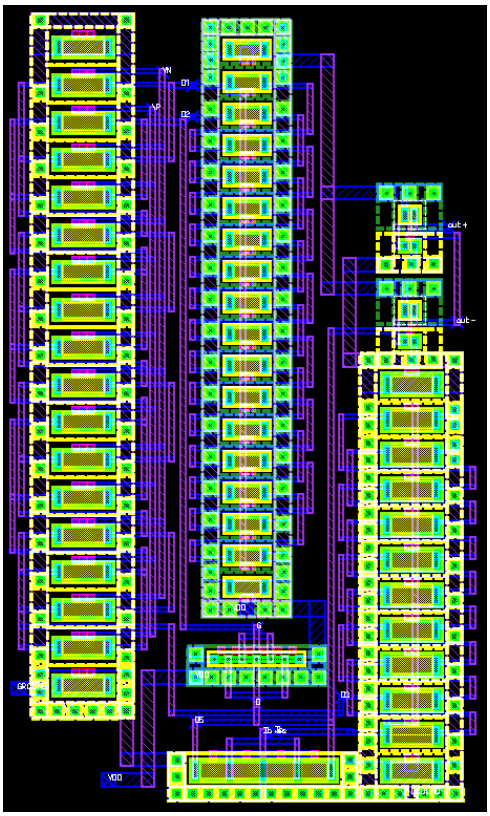
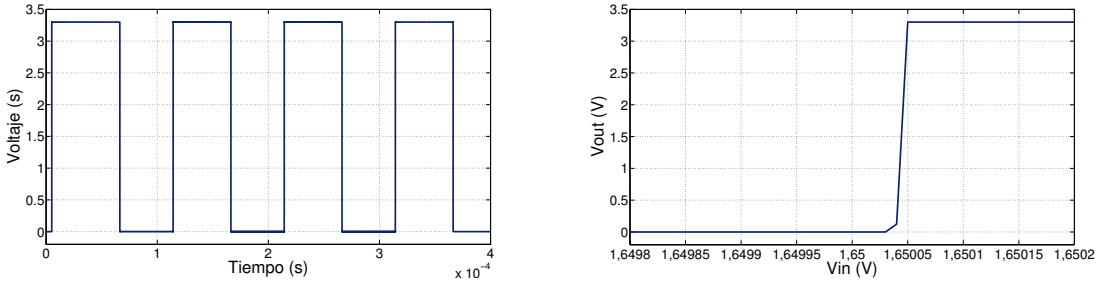


Figura 4.39: Trazado funcional de comparador. Tomado de [3]



(a) Respuesta transitoria

(b) Respuesta en CD de la tensión de salida $Vout$ versus la tensión de entrada Vin.

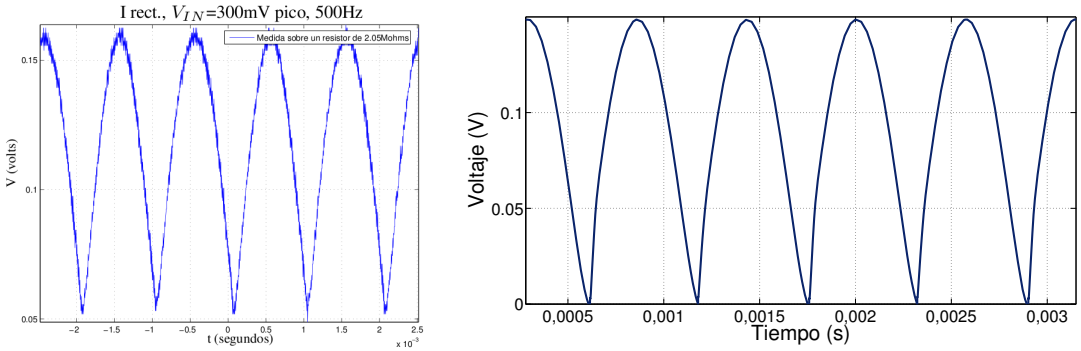
Figura 4.40: Simulación *post-layout* del comparador seleccionado, segunda implementación, para una señal senoidal de entrada de $0,15V$ de amplitud y frecuencia $10kHz$. Tomada de [3]

$$\frac{V_P - V_X}{\phi_t} + 1 = \left(\sqrt{1 + i_{f_2}} - 1\right) + ln\left(\sqrt{1 + i_{f_2}} - 1\right) \tag{4.6}$$

$$\frac{V_P}{\phi_t} + 1 = \left(\sqrt{1 + i_{f_1}} - 1\right) + ln\left(\sqrt{1 + i_{f_1}} - 1\right) \tag{4.7}$$

Tabla 4.19: Características del comparador después de las modificaciones al trazado. Tabla tomada de [3]

Offset (μV)	**Ciclo de Trabajo** (%)	**Ib** (nA)	**Consumo** (nA) @$3,3V$	***slew rate*** $V/\mu s$
35	49,9	14,5	35,4	401



(a) Respuesta obtenida en la unidad en [13]

(b) Respuesta *post-layout* la unidad trazada

Figura 4.41: Respuesta a mejorar y simulación *post-layout* del rectificador, para una señal senoidal de entrada de $0,15V$ y $875Hz$.

A partir de las anteriores ecuaciones se deriva la expresión (4.8) según [47] que describe la sensibilidad de la corriente de referencia (I_{REF}) al voltaje $PTAT$ denominado V_{REF}. Para obtener (4.8) se asume una desviación en V_X igual a δV_X y corrientes iguales a través de los transistores M_5 y M_6.

$$\frac{\delta i_{ref}}{i_{ref}} = 2\frac{\delta V_X}{\phi_t}\left[\sqrt{1+i_{f1}} - \sqrt{1+\frac{i_{f1}}{\left(1+\frac{2S_2}{S_1}\right)}}\right]^{-1} \tag{4.8}$$

La Fig 4.44 representa gráficamente la sensibilidad de I_{REF} a V_X en función del nivel de inversión del transistor M_1. Para diferentes relaciones de S_2/S_1 se observa que la sensibilidad de I_{REF} a V_X es extremadamente alta en valores de niveles de inversión bajos ($i_{f1} \to 0$). Así por ejemplo, si $i_{f1} < 3$, la corriente de referencia cambia por más de un 10% por mV de variación en V_X. En orden a minimizar la sensibilidad de I_{REF} a los voltajes $PTAT$ se debe dimensionar M_1 para que opere en inversión moderada.

Reescribiendo la expresión (4.5) como

$$\frac{i_{f1}}{i_{f2}} = \alpha = \left[1+\frac{S_2}{S_1}\left(1+\frac{1}{N}\right)\right] \tag{4.9}$$

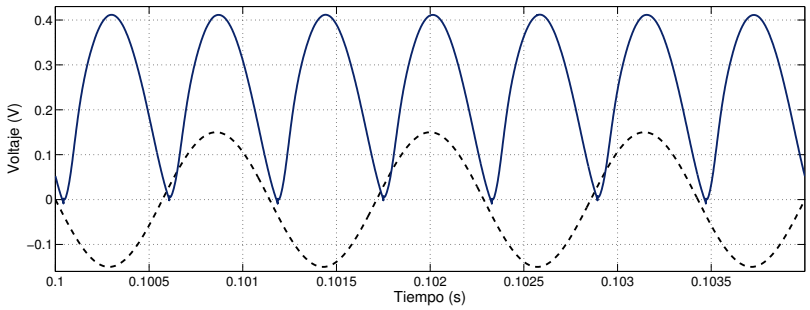


Figura 4.42: Simulación post-trazado de la tensión de salida de la unidad de cálculo completa, para tres entradas senoidales idénticas de $0,15V$ a $815Hz$. Tomada de [3]

Tabla 4.20: Características de la unidad de cálculo completa obtenidas de la simulación *post-layout*, para tres entradas idénticas de $0,15V$ a $815Hz$. Tomada de [3]

Alimentación (V)	**Offset (uV)**	**Consumo (nA) @Ib =14,5 nA**
3,3	548	604,5
4	547	612,9

y aplicando la expresión (4.10) a los transistores M_1 y M_2 según lo expuesto por [48] se puede ver que aplicando

(4.10)

se obtiene

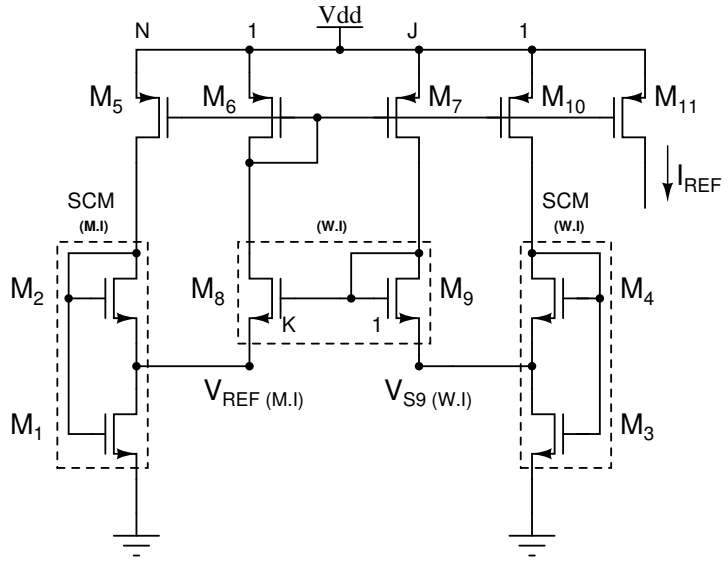


Figura 4.43: Topología simétrica de fuente de corriente auto-polarizada. Tomada de [20].

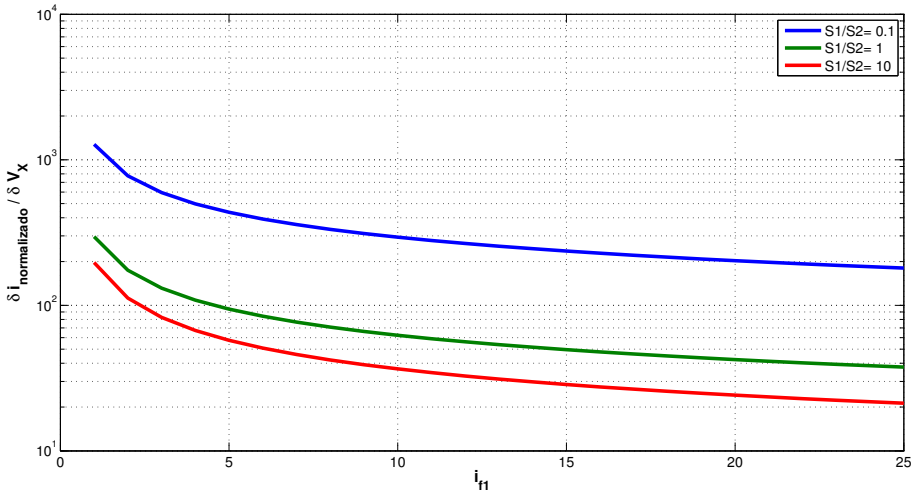


Figura 4.44: Sensibilidad de corriente de referencia (I_{REF}) ante nivel de inversión i_{f1} del transistor M_1 para una fuente de corriente auto-polarizada y distintas relaciones S_1/S_2. Tomada de [20]

$$\frac{V_X}{\phi_t} = \sqrt{1+\alpha i_{f2}} - \sqrt{1+i_{f2}} + ln\left[\frac{\sqrt{1+\alpha i_{f2}}-1}{\sqrt{1+i_{f2}}-1}\right] \tag{4.11}$$

La Fig 4.45 muestra una familia de curvas que representan la variación de V_X según el índice de inversión de M_2 para valores de α de acuerdo con la expresión (4.11).

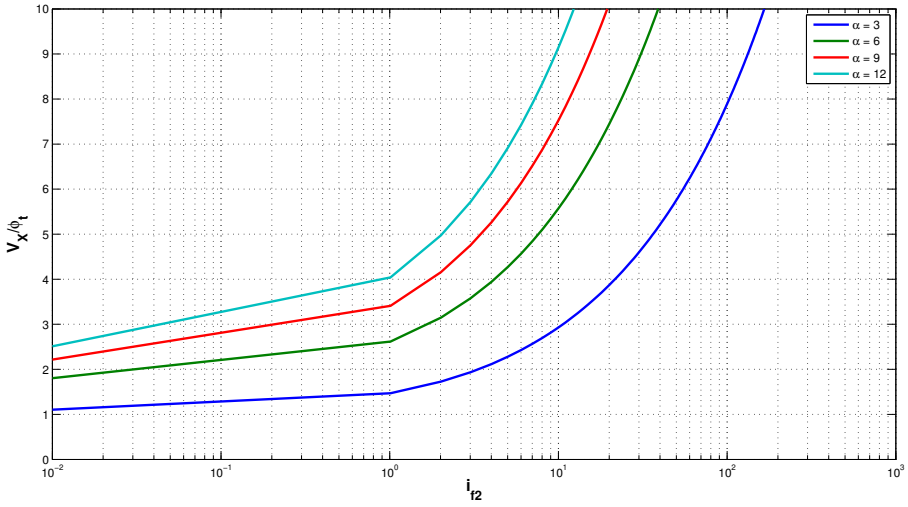


Figura 4.45: Sensibilidad de corriente de referencia (I_{REF}) ante nivel de inversión i_{f2} del transistor M_2 para una fuente de corriente auto-polarizada y distintos valores de α. Tomada de [20]

Para valores de i_{f2} inferiores a diez la variación del voltaje V_X es casi independiente de la corriente de polarización pero dependiente de α, o mejor dicho el valor de i_{f2} es

extremadamente sensible a V_X. Para valores de i_{f2} mayores a la unidad la dependencia de i_{f2} sobre V_X decrece progresivamente. De acuerdo con lo concluido previamente, M_2 debe operar en inversión moderada y así el *SCM* a la izquierda del circuto *SBCS* debe operar completamente en inversión moderada (ver figura 4.43).

El rendimiento de una fuente de corriente se mide por sus sensibilidades a los parámetros que influyen en su corriente de salida, tales como las variaciones en el voltaje de alimentación y las de temperatura.

La fuente de corriente propuesta extrae la corriente específica I_{SQN} de un transistor *nMOS*. Un transistor *nMOS* polarizado con una réplica escalada de la corriente específica presenta un nivel de inversión independiente de la temperatura. Sin embargo, la corriente específica es un parámetro intrínseco que tiene información sobre el proceso y la temperatura; y que se define como:

$$I_{SQ} = \mu C_{ox} n \frac{\phi_t^2}{2} \tag{4.12}$$

La movilidad (μ) y la capacitancia del óxido por unidad de área (C_{ox}) son parámetros dependientes de la temperatura. La variación de C_{ox} es menor a $40ppm/^\circ K$, por lo cual se desprecia la misma. La variación de la movilidad no es despreciable y se define en 4.13 como:

$$\mu = \mu_o \left(\frac{T}{T_o}\right)^2 \tag{4.13}$$

donde T_o es la temperatura de referencia y m es una constante positiva en el rango de $1.2 - 2$ de acuerdo con [91]. Despreciando la dependencia débil del *slope factor* en la temperatura, la corriente específica puede reescribirse según [48] como:

$$I_{SQ} = \mu_o C_{ox} n_o \frac{\phi_{to}^2}{2} \left(\frac{T}{T_o}\right)^{2-m} \tag{4.14}$$

Los parámetros μ_o, n_o y ϕ_{to}^2 está definidos a la temperatura ambiente y $2 - m \cong 0.5$. La Fig 4.46, muestra la variación de la corriente específica de transistores *nMOS* y *pMOS* con respecto a la temperatura.

El voltaje mínimo para polarizar los transistores de la fuente de corriente está dado por las limitaciones impuestas por las dos ramas más a la izquierda del circuito de la Fig 4.43. Según [47] el voltaje de alimentación (V_{DD}) mínimo se define como:

$$V_{DD} \geq max\left\{|V_{DSsat,P}| + V_{GS,M_1}, |V_{GS,P}| + V_{DSsat,M_8} + V_X\right\} \tag{4.15}$$

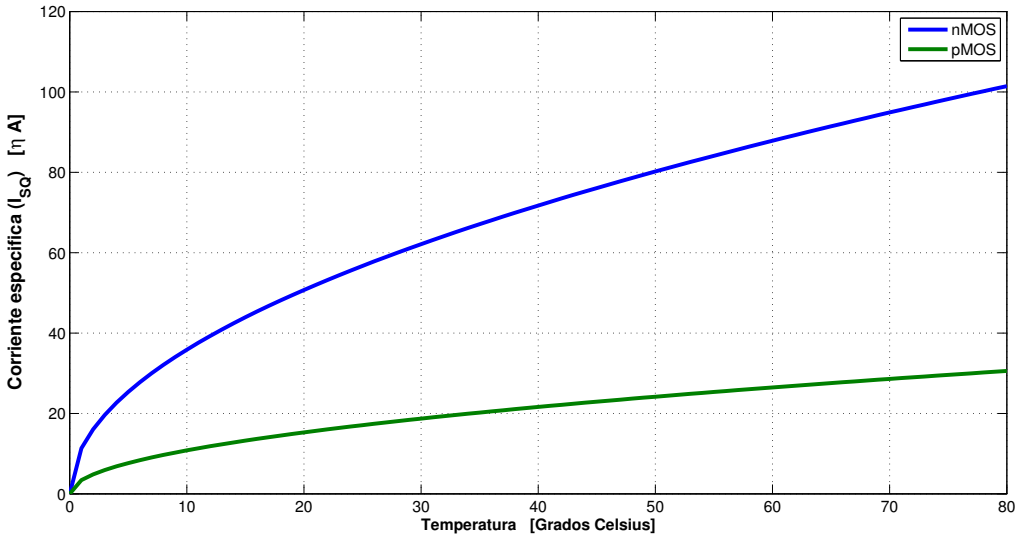


Figura 4.46: Corriente específica (I_{SQ}) según la temperatura para transistores *CMOS*. Figura tomada de [20]

En (4.15) $V_{DSsat,M_8} \cong 100mV$ $(5\phi_t)$ debido a que M_8 opera en inversión débil. Los transistores $pMOS$ de los espejos de corriente operan en inversión débil, por lo cual $i_{fP} < 1$; y por eso $|V_{DSsat,P}| \cong 100mV$ y $V_{GS,P} \cong V_{TP}$ o menos. V_X debe ser menor a $100mV$ mientras M_2 debe operar en inversión moderada de acuerdo con lo expuesto por [47] y [48]. El índice de inversión de M_2 debe estar entre $3-8$, puesto que si opera en inversión débil aumenta la sensibilidad de la corriente de referencia ante el *mismatch* y el voltaje de alimentación. Además, i_{f2} debe tener un valor pequeño en inversión moderada, para que el voltaje *gate-source* sea ligeramente mayor al voltaje umbral. Por tal motivo, $V_X < 100mV$ y luego $V_{GS,M_1} = V_{GS,M_1} \cong V_{TN} + 100mV$. Según lo expuesto anteriormente, y basado en [47] se obtiene una expresión más compacta para V_{DD}.

$$V_{DD} \geq max\left\{|V_{TP}|, V_{TN}\right\} + 200mV \tag{4.16}$$

Las tensiones umbrales (V_{TN} y V_{TP}) definen el voltaje mínimo de alimentación para la fuente de corriente; es más crítica sin embargo la variación introducida por V_{DD}, puesto que el valor nominal es de $3.3V$.

El voltaje térmico (ϕ_t) según [91] se expresa como:

$$\phi_t = 25.9mV\frac{T}{300K} \tag{4.17}$$

A partir de (4.17) se puede reescribir la expresión (4.8) de la siguiente manera:

$$\frac{\delta i_{ref}}{i_{ref}} = \frac{\delta 2V_X}{25.9mV\frac{T}{300K}}\left[\sqrt{1+i_{f1}} - \sqrt{1+\frac{i_{f1}}{\left(1+\frac{2S_2}{S_1}\right)}}\right]^{-1} \tag{4.18}$$

Para un valor definido de S_1/S_2 se puede obtener una gráfica que describa el comportamiento de la sensibilidad de corriente de referencia, ante valores del índice de inversión de M_1 para distintas temperaturas (ver Fig 4.47).

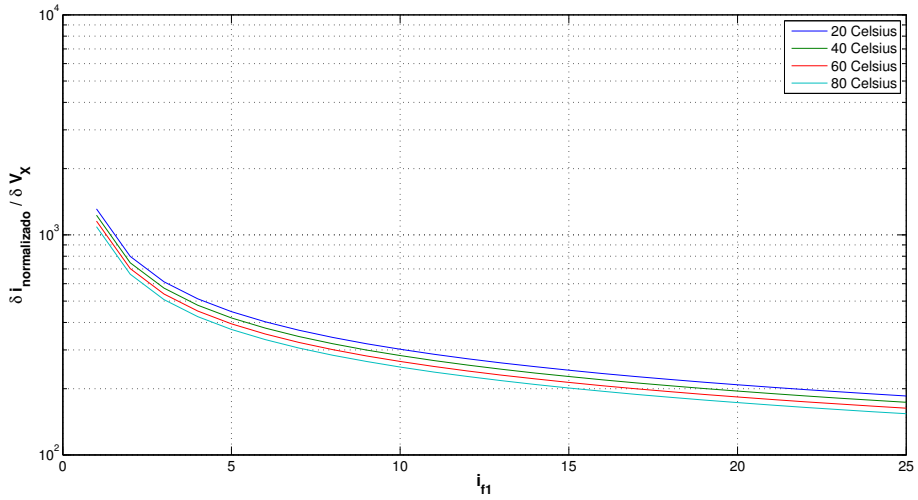


Figura 4.47: Sensibilidad de corriente de referencia (I_{REF}) ante nivel de inversión i_{f1} del transistor M_1 para una fuente de corriente auto-polarizada y distintas temperaturas. Figura tomada de [20]

De las curvas de las figuras 4.44 y 4.47 se desprende que la sensibilidad de la corriente ante el voltaje de alimentación es proporcional a la sensibilidad ante la temperatura. Ambas sensibilidades se reducen al aumentar el índice de inversión del transistor M_1.

Se dimensionaron varios prototipos para comprobar la metodología de diseño expuesta por [47]. En la tabla 4.21 se muestra el esquema de apilamiento utilizado para construir el equivalente de cada uno de los 11 transistores del circuito de *SBCS* presentando en la Fig. 4.43. Los transistores M_1 y M_2 necesitan ser muy largos, puesto que deben operar en inversión moderada; por consiguiente, ambos arreglos son representados respectivamente por 50 y 20 transistores en serie en los esquemas de apilados *Apilado I* y *Apilado II*. Los demás transistores operan en inversión débil y son construidos por arreglos pequeños, dos en paralelo por cuatro en serie para el *Apilado I* y dos en paralelo por dos en serie para el *Apilado II*.

El diseño del primer prototipo de fuente *SBCS* se basó en el esquema *Apilado I*, para una corriente de referencia de $250pA$. En la tabla 4.22 se presenta un resumen de las relaciones de aspecto y los índices de inversión para los transistores, según las fórmulas expresadas anteriormente. Para el diseño se asumió un índice de inversión $i_{f2} = 3$ como punto de

Tabla 4.21: Esquemas de transistores apilados para la SBCS. Tomada de [20]

	Apilado I		**Apilado II**	
M	Paralelo	Serie	Paralelo	Serie
M_1	1	50	1	20
M_2	1	50	1	20
M_3	2	4	2	2
M_4	2	4	2	2
M_P	2	4	2	2
M_X	2	4	2	2

partida según [47], y se procedió a variar la tensión *PTAT* de los *SCMs* de manera tal que permanecieran en *triodo*; desencadenando también en distintos índices de inversión para los transistores M_1 y M_3. Se obtuvieron prototipos de diseño de fuente *SBCS* para tres valores de tensión *PTAT*, $50mV$, $25mV$ y $75mV$; las cuales equivalen a niveles de inversión para M_1 de 10.15, 5.42 y 13.75 respectivamente.

Tabla 4.22: Relaciones de aspecto e índices de inversión para diseños de fuentes SBCS según un voltaje *PTAT* definido (V_{S9}). Corriente de referencia $I_{REF} = 250pA$. Tomada de [20]

	$V_{S9} = 50mV$		$V_{S9} = 25mV$		$V_{S9} = 75mV$	
M	S	i_f	S	i_f	S	i_f
M_1	1.18E-03	10.150	3.50E-003	5.420	7.89E-004	13.750
M_2	1.41E-03	3.000	1.40E-003	3.000	1.41E-003	3.000
M_3	2.20E-01	0.043	1.00E+000	0.013	1.00E-001	0.089
M_4	8.40E-01	0.005	8.48E-001	0.005	8.48E-001	0.005
M_P	1.40E-01	0.100	1.40E-001	0.100	1.40E-001	0.100
M_X	4.24E-01	0.010	4.24E-001	0.010	4.24E-001	0.010

La tabla 4.23 presenta las dimensiones de los transistores para cada circuito considerando el sistema de *Apilado I*.

El procedimiento anterior se repitió para obtener las dimensiones de los transistores de las fuentes diseñadas para el esquema de *Apilado II*. Los resultados se aprecian en la tabla 4.24.

Se implementaron esquemáticos en Design Architech de Mentor Graphics para los tres prototipos diseñados para el sistema de *Apilado I* y para los tres prototipos diseñados para esquema de *Apilado II*. Se realizó una simulación de transitorio de dos segundos para analizar el comportamiento de la corriente de referencia (I_{REF}) de la fuente, sus tensiones *PTAT* (V_{REF} y V_{S9}) y medir su consumo de potencia. El voltaje de alimentación (V_{DD})

Tabla 4.23: Dimensiones de transistores para los prototipos de fuente calculados, según el esquema de *Apilado I*. Figura tomada de [20]

	Fuente 1 - 3	**Fuente 1**	**Fuente 2**	**Fuente 3**
M	W [μ m]	L [μ m]	L [μ m]	L [μ m]
M_1	2	33.9	11.43	50.7
M_2	2	28.37	28.57	28.37
M_3	10	22.73	5	50
M_4	10	5.95	5.95	5.95
M_P	10	35.71	35.71	35.71
M_X	10	11.79	11.79	11.79

Tabla 4.24: Dimensiones de transistores para los prototipos de fuente calculados, según el esquema de *Apilado II*. Tabla tomada de [20]

	Fuente a - c	**Fuente a**	**Fuente b**	**Fuente c**
M	W [μ m]	L [μ m]	L [μ m]	L [μ m]
M_1	2	42.37	14.29	63.37
M_2	2	35.46	35.71	35.46
M_3	10	18.18	4	40
M_4	10	4.72	4.72	4.72
M_P	10	28.57	28.57	28.57
M_X	10	9.43	9.43	9.43

aplicado es de $3.3V$.

Las figuras 4.48 y 4.49 muestran las curvas de la corriente de referencia I_{REF} para los prototipos de fuentes diseñados. La curva para I_{REF} en cada uno de los casos, cumple con ser constante en el tiempo, y no presentar problemas de arranque. La corriente de referencia para el esquema de *Apilado I* fue más precisa sobre el valor teórico de $250pA$.

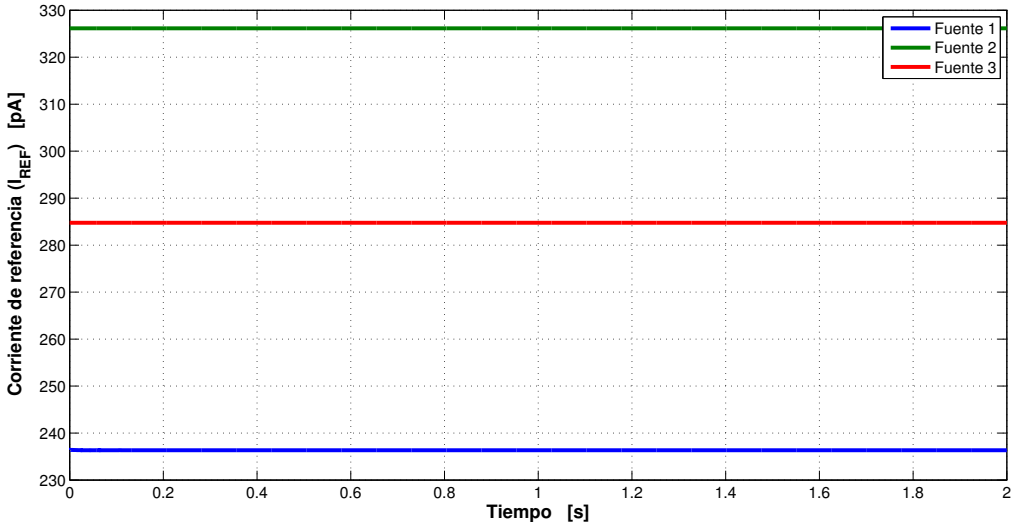


Figura 4.48: Corriente de referencia. Prototipos *SBCS*: 1, 2 y 3. Tomada de [20]

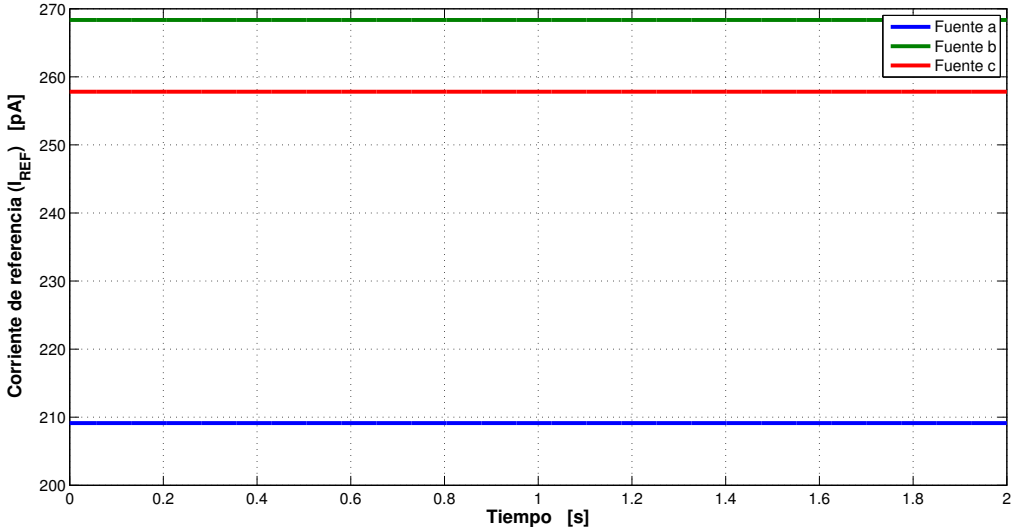


Figura 4.49: Corriente de referencia. Prototipos *SBCS*: a, b y c. Tomada de [20]

Se implementó una simulación *CD* para analizar el comportamiento de la corriente ante la variación de la tensión de alimentación, (V_{DD}). Se varió esta tensión desde los $2V$ hasta los $4V$, y se analizó la pendiente de cada una de estas curvas. Las figuras 4.50 y 4.51 muestran la variación de I_{REF} en función de V_{DD}.

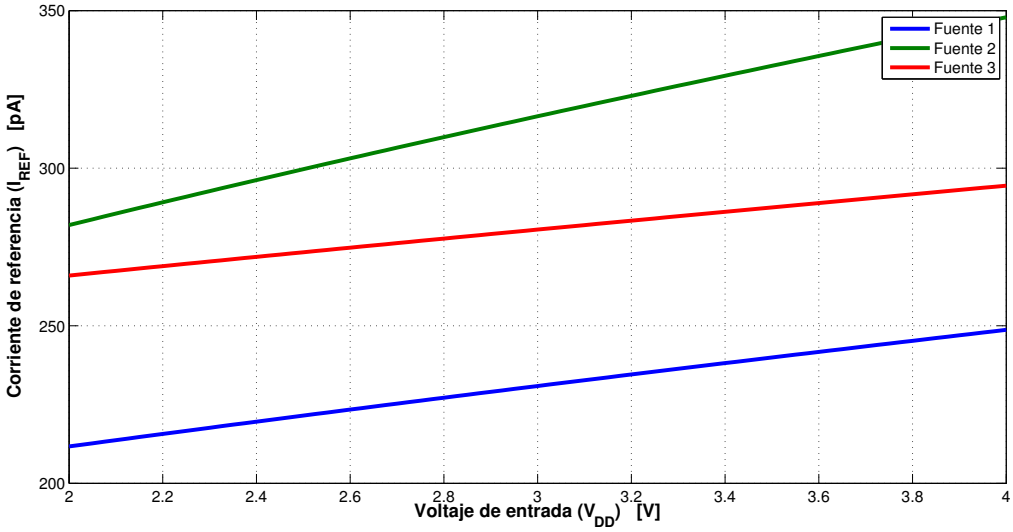


Figura 4.50: Corriente de referencia en función del voltaje de entrada. Prototipos *SBCS*: 1, 2 y 3. Tomada de [20]

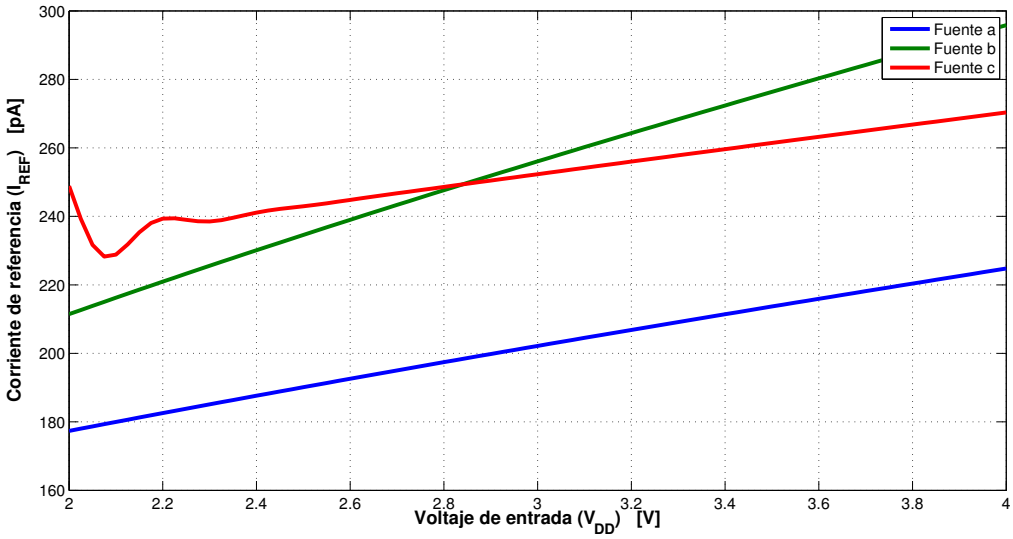


Figura 4.51: Corriente de referencia en función del voltaje de entrada. Prototipos *SBCS*: a, b y c. Tomada de [20]

Un nuevo análisis *CD* se llevó a cabo, pero esta vez en función de la temperatura para determinar el comportamiento de la corriente de referencia. Como se esperaba y lo describe la teoría, la corriente I_{REF} es proporcional a la temperatura; sin embargo, su pendiente la hace variar en menos de $1\%/°C$ para todos los casos debido que M_1 opera en inversión moderada. Las figuras 4.52 y 4.53 muestran los resultados de estas simulaciones para los prototipos de fuentes diseñados según los esquemas de apilado.

Las tablas 4.25 y 4.26 detallan las corrientes de *drain* y los índices de inversión para cada transistor de los seis diseños propuestos. La corriente de *drain* se obtuvo mediante

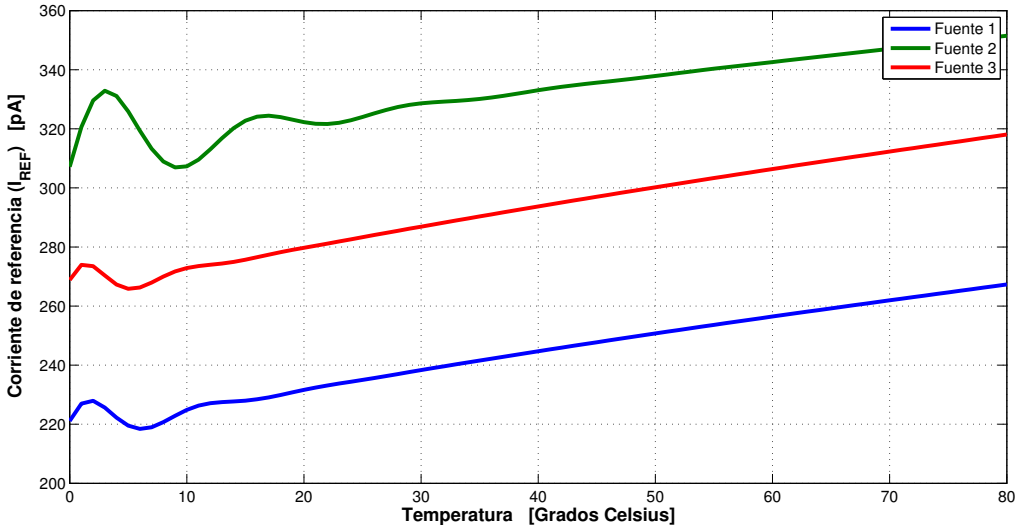


Figura 4.52: Corriente de referencia en función de la temperatura. Prototipos *SBCS*: 1, 2 y 3. Tomada de [20]

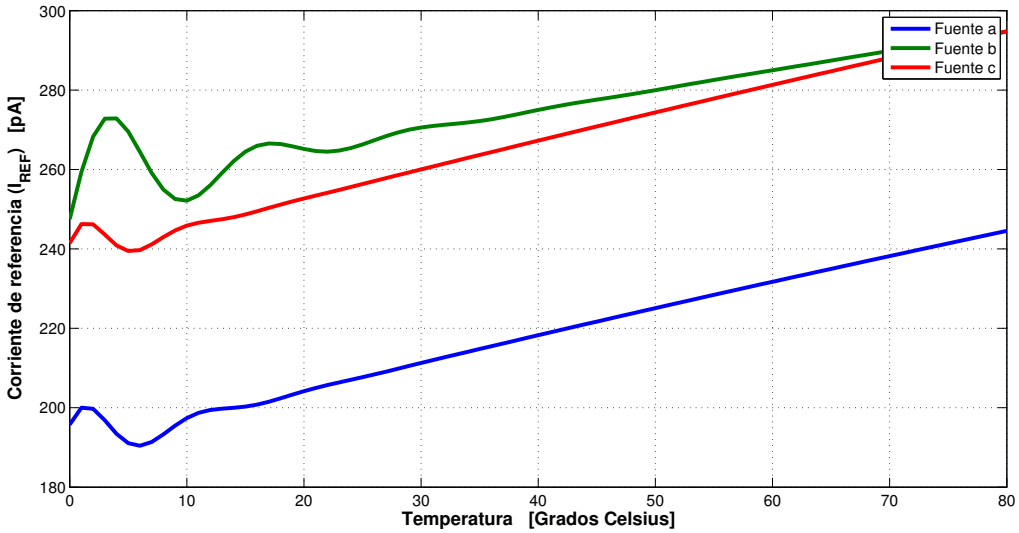


Figura 4.53: Corriente de referencia en función de la temperatura. Prototipos *SBCS*: a, b y c. Tomada de [20]

un análisis de transitorio, y el índice de inversión fue calculado según las ecuaciones del modelo ACM.

Las fuentes propuestas han sido diseñadas para analizar el impacto del índice de inversión del transistor M_1. Se aprecia que para i_{f1} ¿ 10, la corriente de referencia es más precisa con respecto a la corriente planteada de $250pA$. La *fuente 1* es homóloga a la *fuente a*, la *fuente 2* a la *fuente b* y la *fuente 3* a la *fuente c*; lo que cambia entre prototipos es el esquema de apilamiento. Las *fuente 3 y c* tienen un $i_{f1} > 10$ y muestran la mínima sensibilidad de I_{REF} al voltaje de alimentación; sin embargo, este valor de índice de

Tabla 4.25: Corrientes de drain (I_D) e índices de inversión (i_f) experimentales para prototipos de fuentes 1, 2 y 3. Tomada de [20]

	Fuente 1		**Fuente 2**		**Fuente 3**	
M	I_D [pA]	i_f	I_D [pA]	i_f	I_D [pA]	if
M_1	446.980	6.428	629.730	3.053	540.942	11.634
M_2	233.720	2.813	323.470	3.921	281.921	3.393
M_3	465.040	0.036	645.800	0.011	560.929	0.095
M_4	234.730	0.005	324.640	0.006	283.071	0.006
M_P	236.330	0.029	326.150	0.040	284.759	0.035
M_X	229.490	0.009	319.140	0.013	277.807	0.011

inversión ubica al transistor M_1 en inversión fuerte lo que desencadena en mayor área de silicio para éste.

Él cálculo del factor de inversión (i_f) de los transistores M_1 y M_3 por estar polarizados en la región de *triodo* involucra el índice de inversión en inversa (i_f) y la corriente en inversa (I_R) como lo expresa (4.19); sin embargo, se ha despreciado el efecto de este par de parámetros por la imposibilidad de determinar el valor de la corriente en inversa; de esta manera el cálculo experimental de (i_{f1}) e (i_{f3}) es una aproximación. La determinación experimental de los restantes (i_f), se realizó como lo indica la teoría según la ecuación 4.20.

$$I_D = I_F - I_R = I_S\,(i_f - i_r) \tag{4.19}$$

$$i_{f(r)} = \frac{I_{F(R)}}{I_S} \tag{4.20}$$

Finalmente, en la tabla 4.27 se presenta un resumen de los parámetros obtenidos posterior a la simulación de las fuentes diseñadas para ambos esquemas de apilamiento. Por precisión en corriente de referencia domina la *fuente c* presentando el mínimo porcentaje de error (3.12%) sobre el valor teórico ($250pA$). La potencia promedio de salida (P_{OUT}) para todas las fuentes es $4.3\eta W$, por lo cual se desprende que ningún diseño domina a otro. Lo mismo sucede con la sensibilida de I_{REF} a la temperatura, donde el valor promedio es de $0.23\%/°C$.

Los voltajes $PTAT$ experimentales ($V_{S9} - V_{REF}$) difieren de los valores teóricos por más de un 10% en la mayoría de los casos, lo cual afecta la estabilidad del proceso de extracción de corriente, y por ende las sensiblidades de la corriente de referencia. La *fuente 3* presenta el menor valor de sensibilidad a la tensión de alimentación ($5\%/V$), mientras que la *fuente b* posee la máxima ($15.67\%/V$).

Tabla 4.26: Corrientes de drain (I_D) e índices de inversión (i_f) experimentales para prototipos de fuentes a, b y c. Tomada de [20]

	Fuente a		**Fuente b**		**Fuente c**	
M	I_D [pA]	i_f	I_D [pA]	i_f	I_D [pA]	i_f
M_1	395.796	5.692	515.513	2.499	491.601	10.573
M_2	205.377	2.472	264.604	3.207	253.797	3.054
M_3	410.726	0.032	529.841	0.009	507.250	0.086
M_4	207.064	0.004	266.656	0.005	255.551	0.005
M_P	209.138	0.025	268.360	0.033	257.815	0.031
M_X	200.589	0.008	259.599	0.010	249.090	0.010

El área de silicio mínima es estimada según la cantidad de transistores para ambos esquemas de apilado. Se tienen 172 transistores para el esquema de *Apilado I* y 76 para el esquema de *Apilado II*; el área se aproximó para las dimensiones de cada fuente. Por cantidad de transistores, el esquema de *Apilado II* proporcionó los diseños con menor área.

Tabla 4.27: Resumen de parámetros experimentales para prototipos de fuentes diseñadas. Tomada de [20]

Fuente	**1**	**2**	**3**	**a**	**b**	**c**
I_{REF} $[pA]$	236.33	326.15	284.75	209.14	268.36	257.81
P_{OUT} $[\eta W]$	3.95	5.33	4.65	3.39	4.37	4.19
Sens a V_{DD} $[\%/V]$	7.82	10.09	5	11.31	15.67	9.07
Sens a T $[\%/°C]$	0.25	0.14	0.22	0.32	0.19	0.28
Área mínima $[\mu m^2]$	24693.59	21044.2	28550.79	4511.38	3727.07	5279.72
V_{REF} $[V]$	58.11	27.55	79.78	59.75	26.82	82.84
V_{S9} $[V]$	56.66	26.42	78.76	57.62	25.04	81.07

En términos generales, la *fuente 3* demostró tener el mejor desempeño, teniendo la mejor aproximación de corriente de referencia y bajas sensibilidades. Además mediante la comparación de los esquemas de apilado, se verifica que a pesar de que arreglos grandes de transistores disminuyen efectos en el proceso de modelado y fabricación, incrementan sustancialmente el área de un eventual *layout*.

La optimización de la fuente de corriente se realizó siguiendo la metodología expuesta en [64], usando la herramienta de optimización *PESA*. Para este caso se emplean tres *valores de aptitud*: corriente de referencia (I_{REF}), consumo de potencia (P_{OUT}) y sensibilidad de I_{REF} al voltaje de alimentación (V_{DD}). La sensibilidad de I_{REF} a la temperatura no se consideró como valor de aptitud puesto que es directamente proporcional a la sensibilidad

de I_{REF} a V_{DD}.

Los esquemas de apilamiento propuestos en la tabla 4.21 han sido retomados; de manera tal que se han implementado y parametrizado dos circuitos para ser sometidos a optimización según dichos esquemas. En la figura 4.54 se aprecia la parametrización empleada para las dimensiones de los transistores.

Se han ajustado dos análisis en simulación para medir el comportamiento de los valores de aptitud o *fitness* ante el avance de la optimización. Un análisis en transitorio de dos segundos para medir la corriente de referencia y el consumo de potencia en el nodo de la fuente de alimentación y un otro análisis *CD* para medir la variación de la corriente de referencia; de acuerdo con un barrido lineal del voltaje de alimentación desde los $2V$ hasta los $4V$.

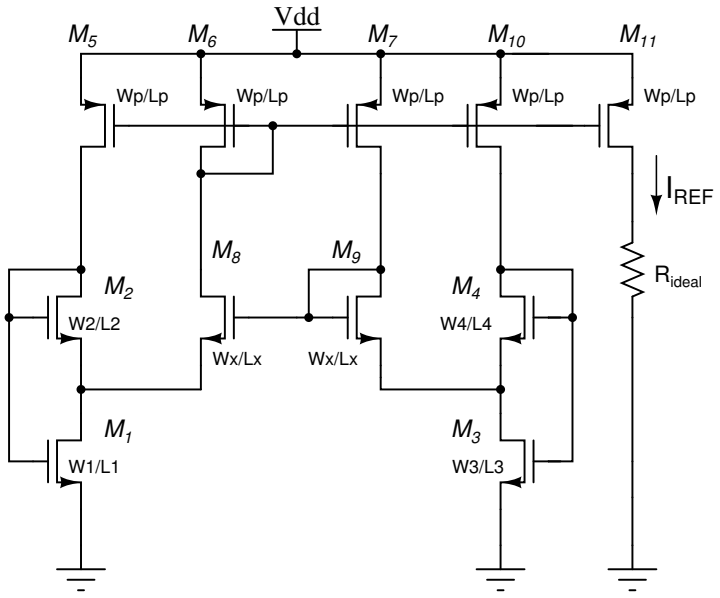


Figura 4.54: Esquemático de fuente de corriente *SBCS* implementado en Design Architech para proceso de optimización. Tomado de [20]

El flujo que sigue el optimizador *PESA* en el proceso de la construcción de un *frente de Pareto* se inicia cuando la *clase* de algoritmos genéticos comienza la evolución de un conjunto de parámetros para las longitudes de los transistores. Estos parámetros son tomados como insumos por el simulador para generar la simulación del esquemático. Los resultados de la simulación son analizados, y si representan un diseño funcional de una fuente de corriente auto-polarizada los tres fitness (I_{REF} , P_{OUT} y Sens a V_{DD}) son introducidos al *frente de Pareto*, y se reinicia el proceso de evolución de nuevos parámetros, tomando como guía los ya generados que han dado buenos resultados.

Una vez que se tiene el *frente de Pareto* después de una n cantidad de simulaciones, se proceden a analizar los puntos del frente que mejor se ajusten a los requisitos impuestos para I_{REF} , P_{OUT} y sensibilidad a V_{DD}. Cuando se determina el punto o los puntos que dan las mejores optimizaciones multiobjetivos, se ubican los parámetros del circuito que arrojaron esas respuestas para los *fitness*; y se da fin al proceso de dimensionamiento de

una fuente de corriente.

Para iniciar el proceso, se implementaron dos circuitos para los dos esquemas de apilado mencionados en 4.21, y los valores de las longitudes de sus canales fueron sustituidos por los parámetros correspondientes. Se implementó una rutina en *C++* que sobreescribe los valores de los parámetros provenientes de los algoritmos genéticos en los archivos de *spice* del simulador así como una rutina que analiza la forma de onda de la corriente de referencia.

El análisis de la curva de I_{REF} determina la validez de un diseño de una fuente de corriente.

En caso de obtenerse una fuente de corriente con una corriente de referencia constante en el dominio del tiempo, se procedía a analizar su sensibilidad ante la variación del voltaje de alimentación mediante la simulación *CD*. Al variar el voltaje de alimentación (V_{DD}) se obtiene una recta con pendiente positiva, en la curva I_{REF} versus V_{DD}. Dicha pendiente permite calcular la sensibilidad de I_{REF} ante V_{DD}. El consumo de potencia es tomado de los reportes generados por el simulador por una rutina en software.

Luego de que se han obtenido los tres valores de aptitud, estos son tomados por la biblioteca *LTI-Lib* para construir el *frente de Pareto*, según análisis de puntos dominados y no dominados.

Los valores mínimos son establecidos según le tecnología *CMOS* de $0.5\mu m$ usada en el proyecto.

Al concluir el proceso de optimización de cada una de las topologías de apilado, la herramienta facilitó un *frente de Pareto*. Este frente contiene los valores de aptitud más óptimos para cada configuración.

La figura 4.55 y la figura 4.56 contienen la representación gráfica de los frentes. Se detalla que la región que contiene resultados válidos es mucho mayor en la figura 4.56 que pertenece al esquema de *Apilado II*. Con la optimización se pretende que tanto la corrientes de referencia, la potencia de salida y la sensibilidad al voltaje de alimentación sean las mínimas posibles. La herramienta diseñada por [65] opera con valores crecientes, por lo cual se trabajó con los inversos de los valores de aptitud y por eso las figuras de *frente de Pareto* muestran valores tan altos en sus escalas.

Después de analizar los resultados de los *fitness* en los frentes, fue posible elegir tres prototipos de fuentes para cada esquema de apilado. La tabla 4.28 resume los principales parámetros de las fuentes. Las fuentes *I* , *II* y *III* pertenecen al esquema de *Apilado I*, mientras que las fuentes *IV*, *V* y *VI* obedecen al sistema de *Apilado II*. La sensibilidad de I_{REF} ante la variación de la temperatura, se calculó una vez finalizado el proceso de optimización.

La tabla 4.29 presenta las dimensiones de los transistores para los dos diseños de fuentes *SBCS* elegidos; se seleccionó una fuente por sistema de apilado. Las fuentes se eligieron, con base a sus sensibilidades y sus áreas estimadas. Para el esquema de *Apilado I* se eligió la *fuente I* (ver convención de tabla 4.28), y para el esquema de *Apilado II*, la *fuente IV*

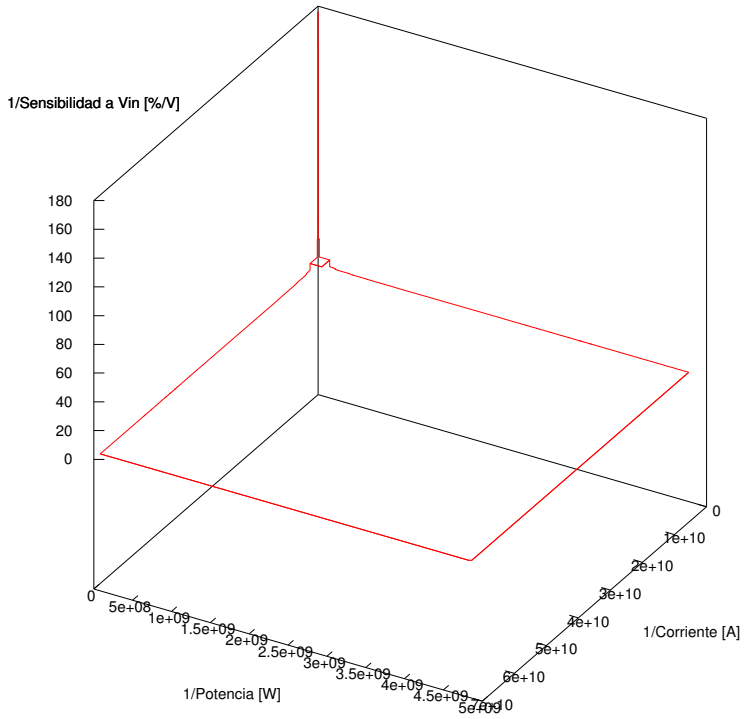


Figura 4.55: Frente de Pareto para optimización según *Apilado I*. Tomada de [20]

Tabla 4.28: Parámetros de mejores *SBCS's* obtenidas mediante la optimización con base a esquemas de *Apilado I* y *Apilado II*. Tomada de [20]

Fuente	I	II	III	IV	V	VI
I_{REF} $[pA]$	245.64	303.95	337.99	254.68	217.33	218.46
P_{OUT} $[\eta W]$	4	4.96	5.52	4.13	3.11	3.54
Sens a V_{DD} $[\%/V]$	4.32	4	3.14	6.19	7.59	7.8
Sens a T $[\%/°C]$	0.3	0.26	0.49	0.37	0.77	0.81
Área mínima $[\mu m^2]$	28013.28	22890.04	48502.32	7902	22731.36	22776.4

fue la escogida.

Los índices de inversión del diseño *SBCS* se muestran en la tabla 4.30 para ambos procesos de optimización. Cabe destacar, que ambos diseños concordaron con el análisis teórico en el hecho de que M_1 debe operar en inversión moderada, de ahí que se hayan obtenido sensibilidades bajas ante el voltaje de alimentación y ante la temperatura en ambos prototipos de fuentes. Los restantes niveles de inversión ubican a todos los transistores en inversión débil, y esto establece una discrepancia en cuanto a lo expuesto por el análisis teórico anterior. Los resultados de la optimización de la topología de una fuente de corriente auto-polarizada han demostrado que el transistor M_2 no necesita operar en

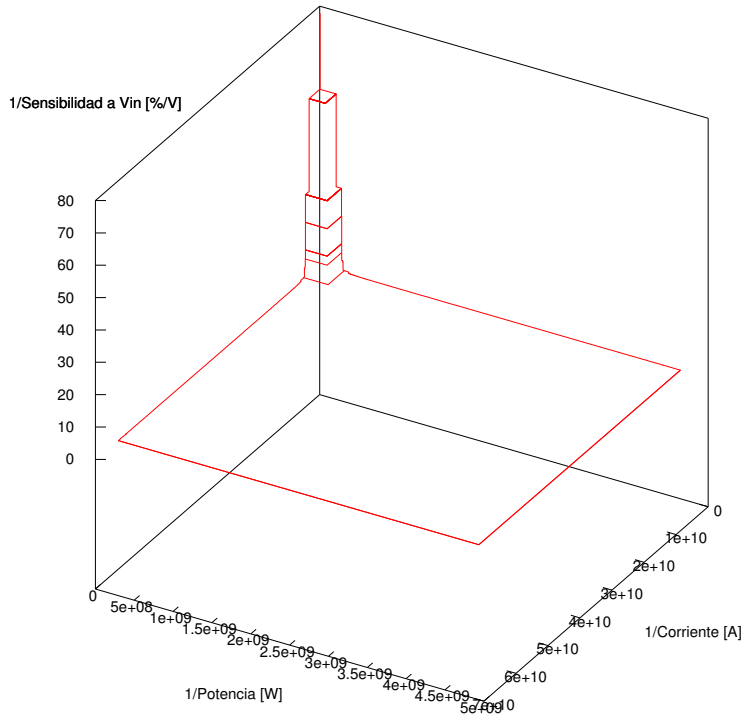


Figura 4.56: Frente de Pareto para optimización según *Apilado II*.
Tomada de [20]

Tabla 4.29: Dimensiones de transistores para *SBCSs* obtenidas mediante optimización, usando ambos esquemas de apilado. Tomada de [20]

	Apilado I		**Apilado II**	
Transistor	W [μm]	L [μm]	W [μm]	L [μm]
M_1	1.8	69.2	0.9	45
M_2	2.9	13.6	7	23
M_3	1.7	10.4	0.9	25
M_4	22.7	24.6	22	6
M_P	18.3	19.4	10	10
M_X	5.1	12.3	10	10

inversión moderada para obtener una corriente de referencia constante en el dominio del tiempo con sensibilidades bajas.

Las tensiones *drain-source* mínimas fueron registradas para las dos prototipos de fuentes, y en ambos circuitos corresponden a las tensiones V_{DS} de los transistores M_1 y M_3. Para la fuente según el sistema de *Apilado I*, las tensiones están en el orden de los $188mV$, y para la fuente según el esquema de *Apilado II* tienen valores aproximados a los $344mV$. Por

Tabla 4.30: Corrientes de *drain* (I_D) e índices de inversión (if) para diseño *SBCS* de *Apilado I* y *Apilado II*. Tomada de [20]

	Apilado I		**Apilado II**	
Transistor	I_D $[pA]$	i_f	I_D $[pA]$	i_f
M_1	450.39	14.69125	472.84	8.02390
M_2	242.71	0.96575	250.76	0.27963
M_X	477.90	0.09922	495.18	0.23342
M_3	243.49	0.00896	251.61	0.00116
M_4	245.64	0.02933	255.04	0.00433
M_P	238.40	0.01951	244.93	0.00260

lo tanto, la herramienta de optimización obtuvo diseños de fuentes funcionales en donde ningún transistor opera en zona de *triodo*; puesto que sus tensiones V_{DS} son superiores al límite entre entre las regiones de *triodo* y saturación ($100mV \cong 5\phi_t$). Los diseños de fuentes optimizados prescinden de tensiones $PTAT$, y demuestran que la generación de estas tensiones no es necesario; al establecer tensiones de V_{DS} para los *SCMs* en la región de *saturación*.

El hecho de que el transistor M_2 opere en inversión débil y que ningún transistor opere en *triodo* permite constatar que el espacio de análisis con el cual consta una herramienta de optimización permite contemplar supuestos que desde un inicio según desarrollos teóricos se hubiesen desechado, y que siempre es posible obtener resultados satisfactorios. Cuando se estableció en el optimizador solo el requisito de verificar la forma de onda de la corriente se abrió la oportunidad a la obtención de resultados como los anteriores, que ponen en contraste la teoría con la práctica experimental.

Las diferencias entre los resultados anteriores y los de la optimización pueden atribuirse a la mayor capacidad de análisis que posee la herramienta, puesto que sus algoritmos le permiten evolucionar en pro de conseguir su objetivo. El análisis teórico demostró ser de gran utilidad y acertado con sus resultados, sin embargo la herramienta brinda una mayor capacidad de exploración del conjunto de parametrizaciones para dimensionar una fuente y obtener los requisitos más óptimos.

La fuente de corriente resultante de la optimizacón del esquema de *Apilado II*, màs específicamente la *fuente IV* (ver tabla 4.28), demostró ser el diseño más robusto, por el balance que posee entre sus parámetros. A pesar de que otros diseños, poseen menores sensibilidades, este prototipo domina y cumple a cabalidad con las especificaciones, y se consideró como el mejor diseño para la topología *SBCS*.

El trazado de la fuente de corriente auto-polarizada se realizó principalmente empleando las técnicas de apilado y multidedo. Se aplicó un diseño modular para facilitar los procesos de verificación de reglas de diseño (DRC) y comparación entre trazado versus esquemático (LVS).

En la siguiente tabla se resumen los parámetros de la fuente escogida para trazar:

Tabla 4.31: Parámetros de *SBCS* obtenida mediante la optimización con base a esquema de *Apilado II*. Tomada de [20]

I_{REF} $[pA]$	P_{OUT} $[\eta W]$	Sens a V_{DD} $[\%/V]$	Sens a T $[\%/°C]$	Área mínima $[\mu m^2]$
254,68	4,13	6,19	0.37	7902

En la tabla 4.32 se presenta un resumen de las dimensiones de los transistores que conforman la fuente de corriente.

Tabla 4.32: Dimensiones de transistores para *SBCS* a implementar en layout. Tomada de [20]

Transistor	W $[\mu m]$	L $[\mu m]$
M_1	0.9	45
M_2	7	23
M_3	0.9	25
M_4	22	6
M_P	10	10
M_X	10	10

Se usó la técnica de apilado para los transistores M_1 y M_2, por la cantidad de los transistores que los conforman (20 por cada uno) y por su carácter de transistores largos. La técnica de trazado interdigitado se utilizó para los transistores M_3 a M_{11}.

Los transistores M_1 y M_2 se dividieron cada uno en dos columnas, tomando como referencia técnicas de asociaciones trapezoidales sugeridas por [47]. La figura 4.57 muestra el diseño del trazado.

La fuente tiene un área de $0.052mm^2$, lo que para un CI de $9mm^2$ equivale a 0.57% del área total. En la figura 4.57 son visibles los puertos del bloque de trazado definitivo de la fuente *SBCS*. Los puertos son voltaje de alimentación (VDD), corriente de referencia ($Iref$) y tierra (GND).

La tabla 4.33 muestra las longitudes de los transistores, una vez que han sido extraídas. Estas longitudes difieren como máximo en un 0.71% para el caso de $W2$, mientras que en los otros casos permanecen constantes. La extracción de parámetros fue el paso previo a la simulación post-layout.

Los mismos análisis implementados para simulaciones de esquemático se aplicaron al trazado de la fuente. Se extrajo un modelo *spice* del trazado final de la fuente; dicho modelo contiene aproximaciones de variables físicas del proceso y brinda una excelente representación del comportamiento post-fabricación de la fuente de corriente. Para el layout de la Fig 4.57 se realizó el análisis de transitorio y CD. La implementación del layout

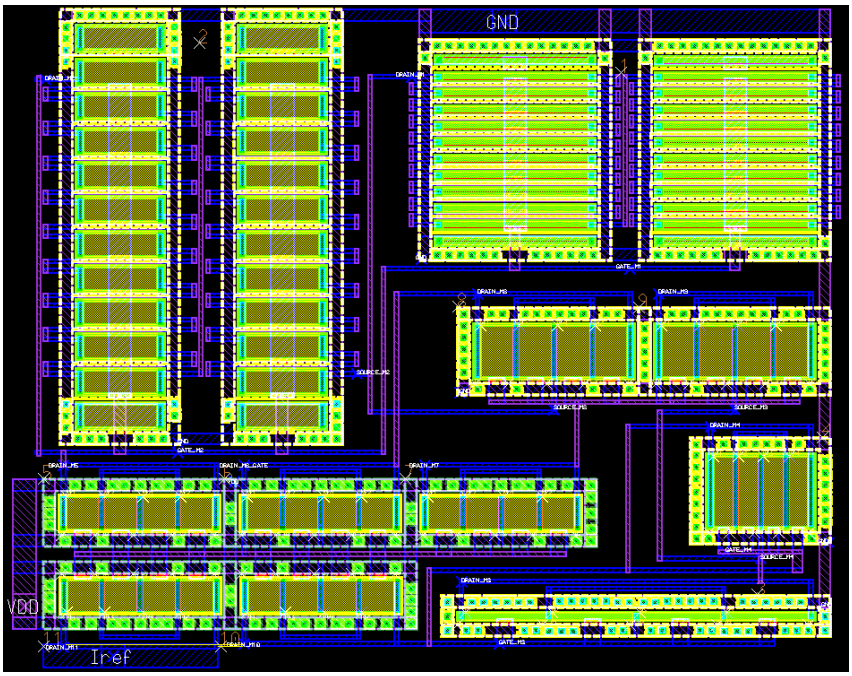


Figura 4.57: Layout de fuente de corriente *SBCS*, 258.77μm por 204.33μm. Tomada de [20]

Tabla 4.33: Dimensiones de transistores para *SBCS* post-layout.

Transistor	**W** [μm]	**L** [μm]
M_1	0.9	45
M_2	7.05	23.1
M_3	0.9	25.05
M_4	22.05	6
M_P	10.05	10.05
M_X	16.05	10.05

del transistor M_3 inicialmente se realizó mediante la técnica de transistores apilados. Los resultados de las simulaciones post-layout no fueron funcionales, como se aprecia en la figura 4.58.

Se extrajo el modelo *spice* del layout de cada transistor, para realizar simulaciones y analizar el impacto del trazadode cada transistor sobre el desempeño de la fuente. El transistor M_3 provocó la mayor desviación sobre la corriente de referencia, resultando un valor de 189pA de 254.68pA esperados. Los restantes transistores produjeron variaciones menores al 2% en la corriente de referencia de 254.68pA. Se determinó que el modelo *spice* de M_3 presentó valores muy altos de resistencias (mayores a 80Ω) y capacitancias, que impidieron un desempeño correcto de la simulación post-layout; como se demostró en la figura 4.58. Por este motivo, se decidió implementar el trazado del transistor M_3 mediante la técnica multidedo.

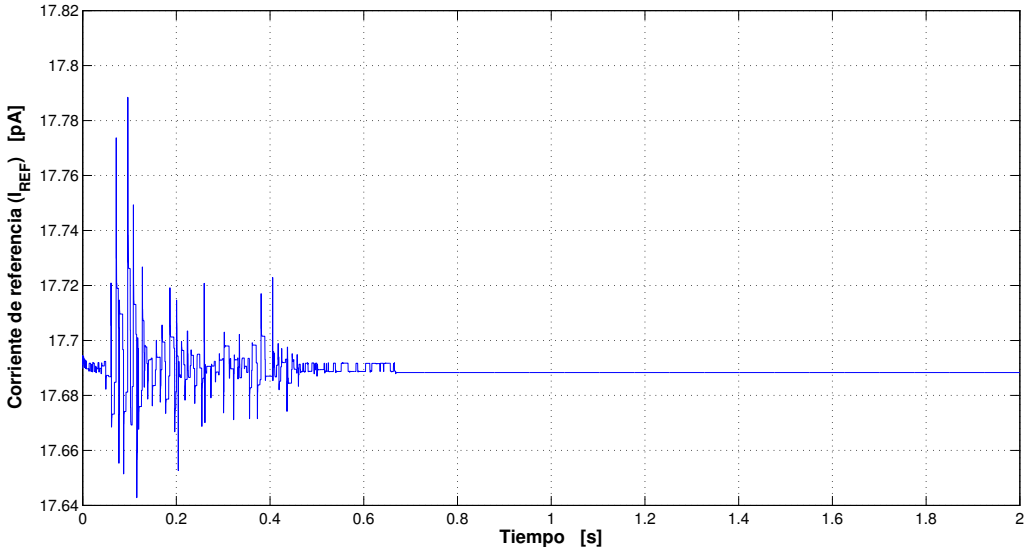


Figura 4.58: Corriente de referencia post-layout para *SBCS*, con transistor M_3 implementado mediante layout apilado. Tomada de [20]

Las curvas del modelo *spice* del esquemático de la fuente y las curvas del modelo *spice* del layout han sido presentadas en una misma figura para facilitar la comparación. La corriente de referencia (I_{REF}) para un transitorio de dos segundos se presenta en la figura 4.59. La corriente de referencia post-layout tiene un porcentaje de error respecto a la corriente de referencia del esquemático de 0.368%. Las figuras 4.60 y 4.61 presentan las sensibilidades de la corriente de referencia ante variaciones del voltaje de alimentación y la temperatura.

La sensibilidad de la corriente de referencia ante el voltaje de alimentación del análisis del modelo *spice* del layout se mantiene por debajo de un $6.49\%/V$, lo mismo sucedió con la sensibilidad del modelo *spice* del esquemático. La regulación de la corriente de referencia con respecto a la temperatura para ambos modelos está por debajo del $1\%/°C$.

La tabla 4.34 contiene un resumen de las especificaciones de los parámetros del layout de la fuente de corriente auto-polarizada, mismos que cumplen con todos los requisitos impuestos en los objetivos.

Análisis de resultados de filtro y la fuente fabricados

Este análisis está basado en el trabajo realizado en [15] y [64]. La prueba y caracterización del ASIC se hizo en una plataforma de medición de National Instruments. Esta se bass en el concepto de instrumentación virtual (VI).

La plataforma utilizada fue un PXI de National Instruments. En su utilización fue necesario definir aspectos tales como tasas de muestreo, resolución, exactitud, blindaje y

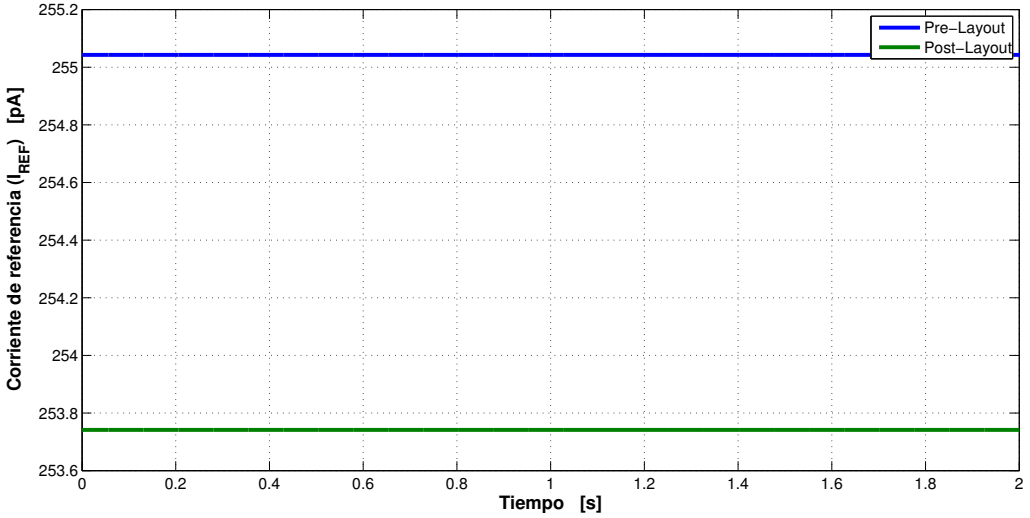


Figura 4.59: Corriente de referencia. Tomada de [20]

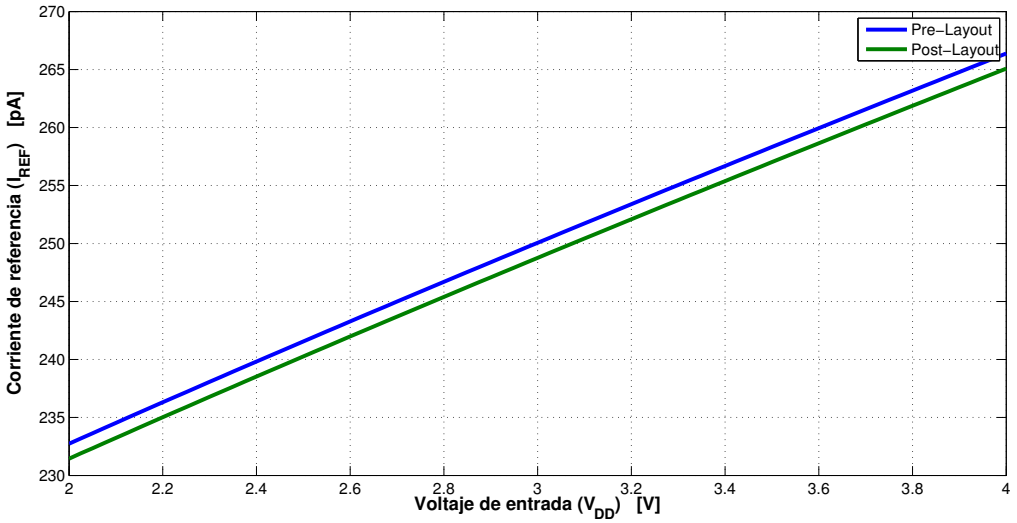


Figura 4.60: Corriente de referencia en función del voltaje de entrada. Tomada de [20]

acondicionamiento de señales. La plataforma se muestra en Fig. 4.62.

Para realizar las mediciones de construyó un PCB especial, donde se minimizaran a las pérdidas por las pérdidas parásitas. La Fig. 4.63 muestra un ejemplo del blindaje desarrollado en el PCB para no afectar las señales provenientes de las SMUs (Source Measurement Unit), ya que debían generar tensiones de referencia muy precisas, y fuentes de corriente muy bajas.

En la Fig. 4.64 tenemos ya el PCB fabricado y colocado en la carcaza de protección, debidamente aterrizara para minimizar el ruido externo.

Se utilizó LabVIEW para generar las configuraciones de hardware, y las distintas rutinas de medición y almacenamiento de datos. Las Figs. 4.65 y 4.66 muestran las mediciones de

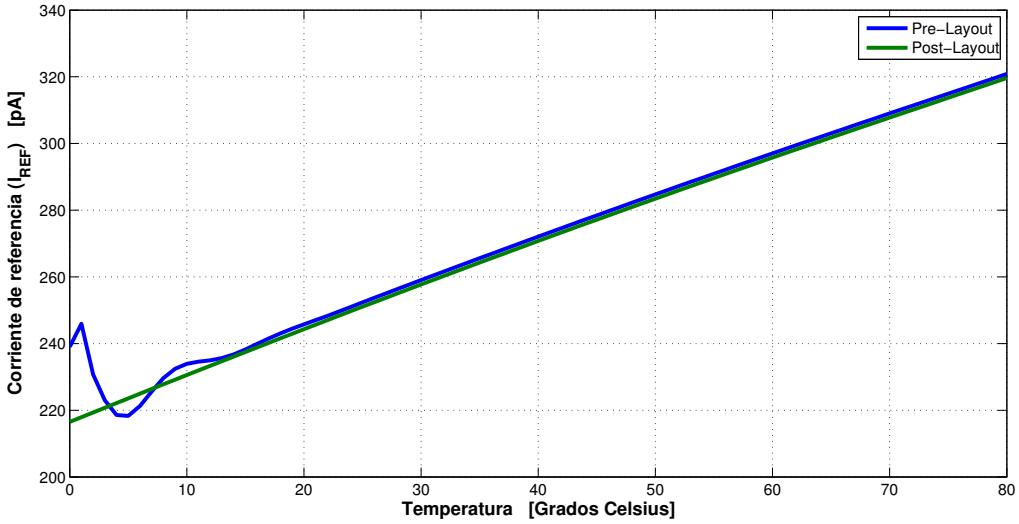


Figura 4.61: Corriente de referencia en función de la temperatura. Tomada de [20]

Tabla 4.34: Parámetros del diseño final de layout de fuente de corriente auto-polarizada. Obtenida mediante optimización con base a esquema de *Apilado II*. Tomada de [20]

Parámetro	**Valor**	**Unidad**
V_{DD}	3.3	V
I_{REF}	253.741	pA
P_{OUT}	4.116	ηW
Sensibilidad de I_{REF} a V_{DD}	6.62	$\%/V$
Sensibilidad de I_{REF} a T	0.5	$\%/°C$
Área Layout	0.052	mm^2

transconductancia versus tensiones de entrada, y la corriente de salida de uno de los OTAs implementados, como función de la tensión de entrada. Los resultados fueron satisfactorios, pues la transconductancia (considered como un 5% sobre el valor obtenido cuando la entrada es igual a la tensión de referencia) es de 84.38nS (apenas un error de -0.73% con respecto al valor diseñado en las secciones anteriores (ver [30]). Puede apreciarse no obstante un offset de 20mV en la Fig. 4.66. La metodología de optimización no consideró el desapareamiento en los transistores, producto de la variabilidad del proceso, a la hora de buscar lo valores óptimos. Vemos que el rango lineal es aproximadamente 1V.

La Fig. 4.65 muestra la respuesta temporal de los coeficientes 3, 4 y 5. En la tabla 4.35 se observar el *offset* final de cada coeficiente. Este *offset* sigue siendo alto, y es un producto de la acumulación inevitable por desapareamiento del *offset* de cada OTA. El no haber utilizado una configuración realimentada en estos filtros (como una estructura biquad) no ayuda a filtrar esta acumulación de *offset*. Este problema impactará en la etapa siguiente de cálculo energético.

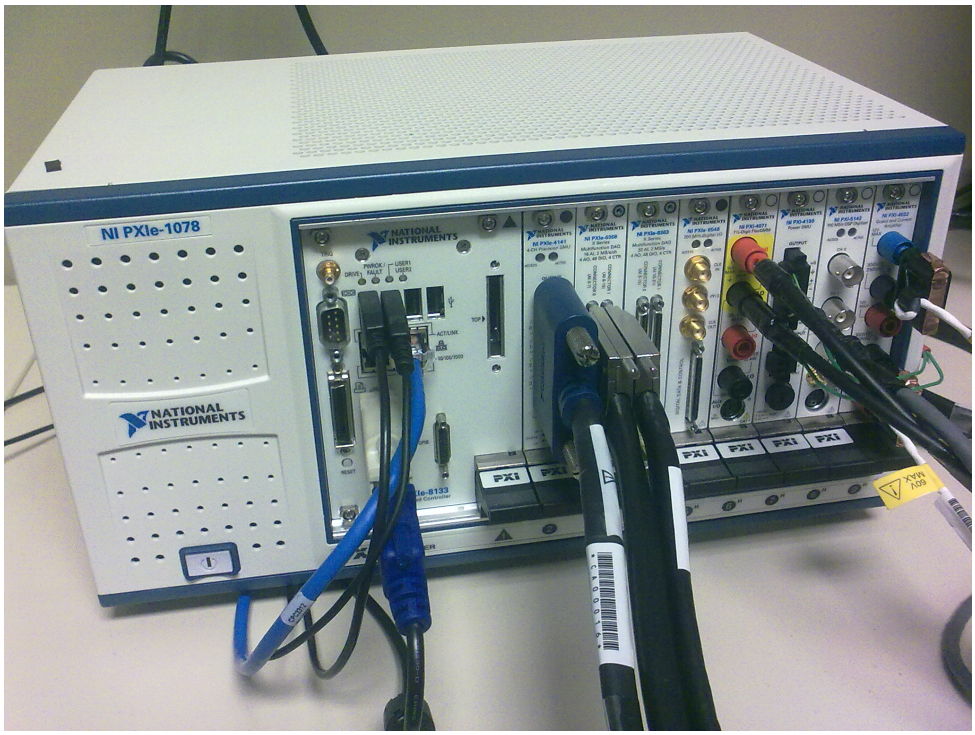


Figura 4.62: Plataforma PXI usada para las mediciones de los circuitos fabricados. Tomada de [15].

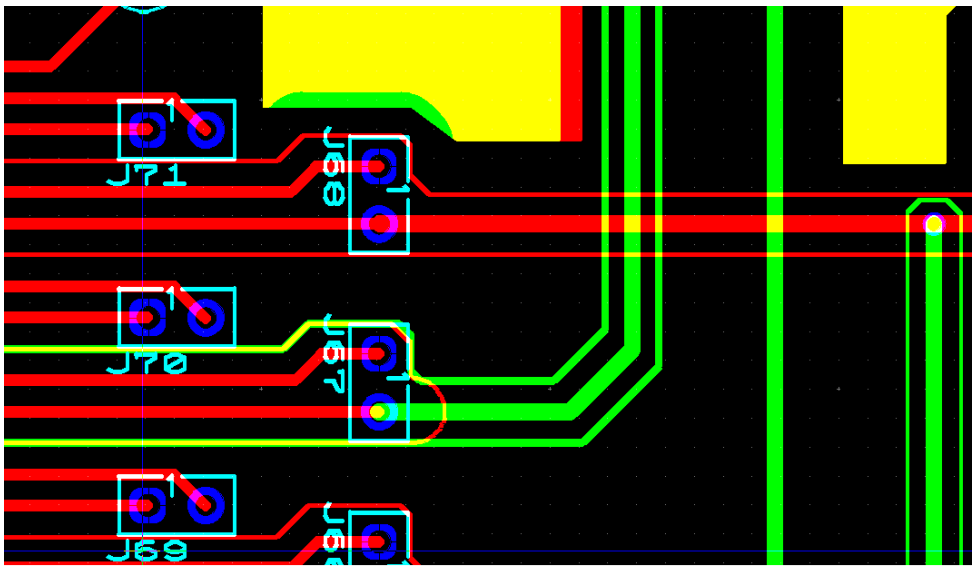


Figura 4.63: Técnica de blindaje usada en el PCB (rojo: pistas en capa superior top layer; verde: pistas en capa inferior; amarillo: traslape de capas.). Tomada de [15].

La Fig.4.68 muestra la respuesta en frecuencia de los tres filtros GmC. La respuesta de frecuencia revela que los filtros están relativamente bien ajustados dentro de las bandas preestablecidas. Es conveniente explorar sin embargo un métodos dechado de ajuste fino para estos filtros, sin que el mismo impacte el consumo de potencia esperado.

La respuesta de la unidad de cálculo se muestra en la Fig.4.69. Se barrió una tensión para

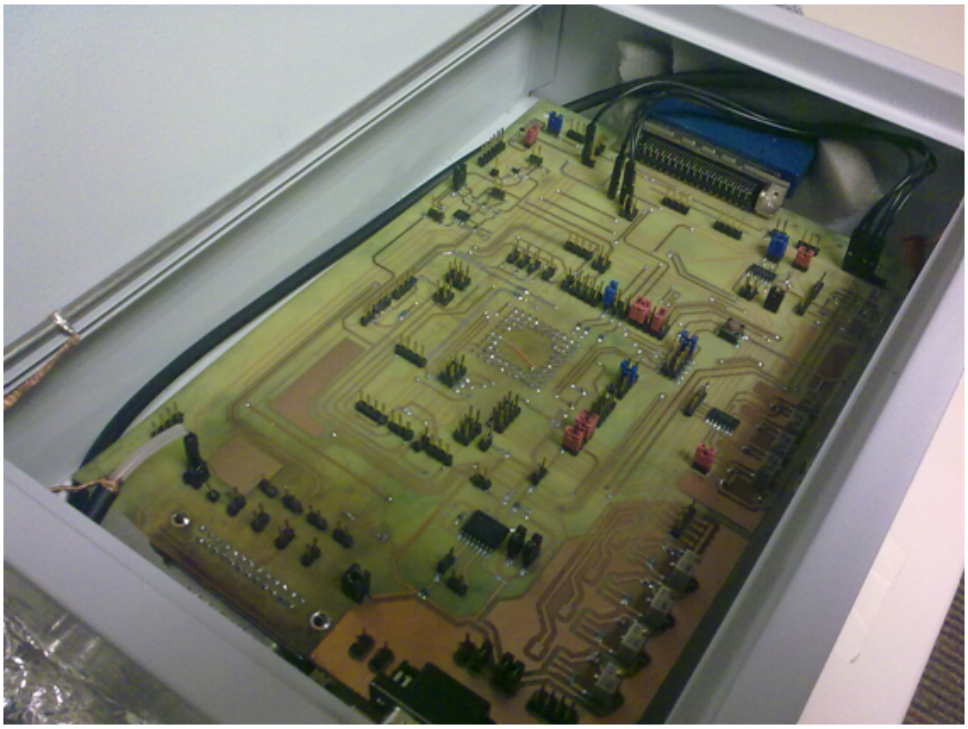


Figura 4.64: PCB para pruebas, ya en su carcasa. Tomada de [15].

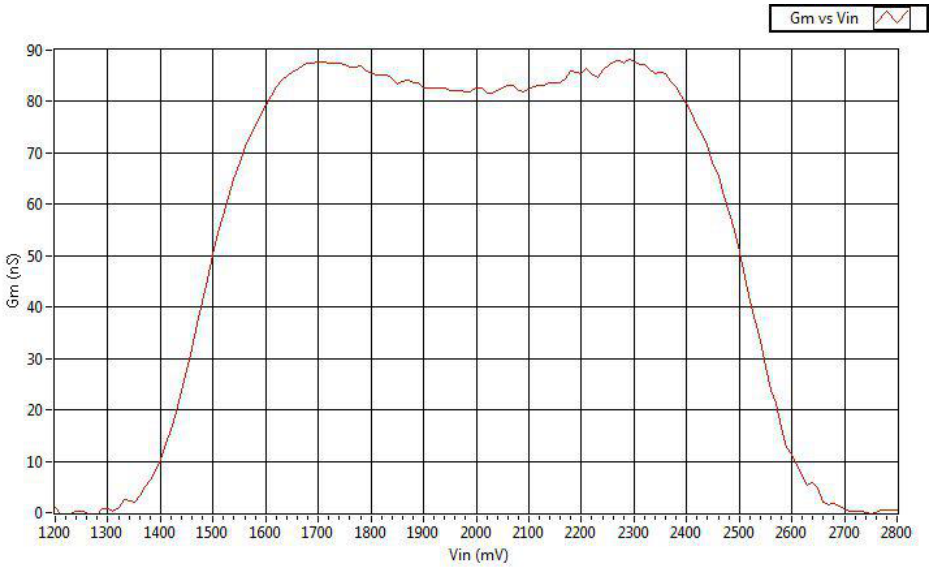


Figura 4.65: G_m medido como función de la tensión de entrada. Tomada de [15].

cada entrada de los coeficientes 3, 4, y 5. A la izquierda de la figura se tiene la salida cuando solo se excita una sola de las entradas. La de la derecha muestra la salida para la misma señal aplicada a los tres coeficientes. El rango lineal de la etapa de rectificación es adecuado a lo que se esperaba (entre 980mv en el caso de una única entrada, a 750mV para todas las entradas conjuntas), pese a un leve *offset* apreciable en ambas figuras.

La Fig .4.70 muestra la salida de las fuentes SBCS fabricadas. Como vemos en la Fig. 4.71

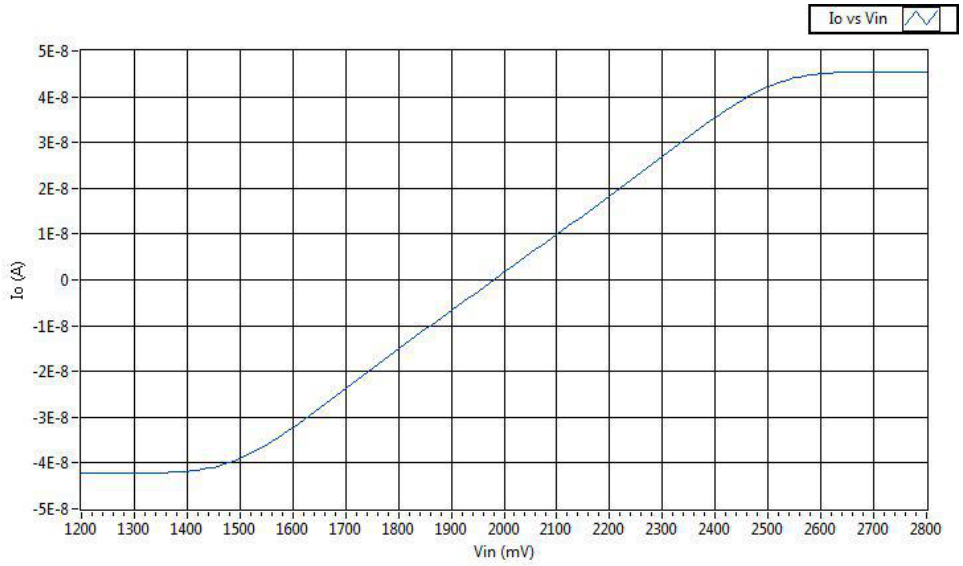


Figura 4.66: Corriente de salida media, como función de la tensión de entrada [15].

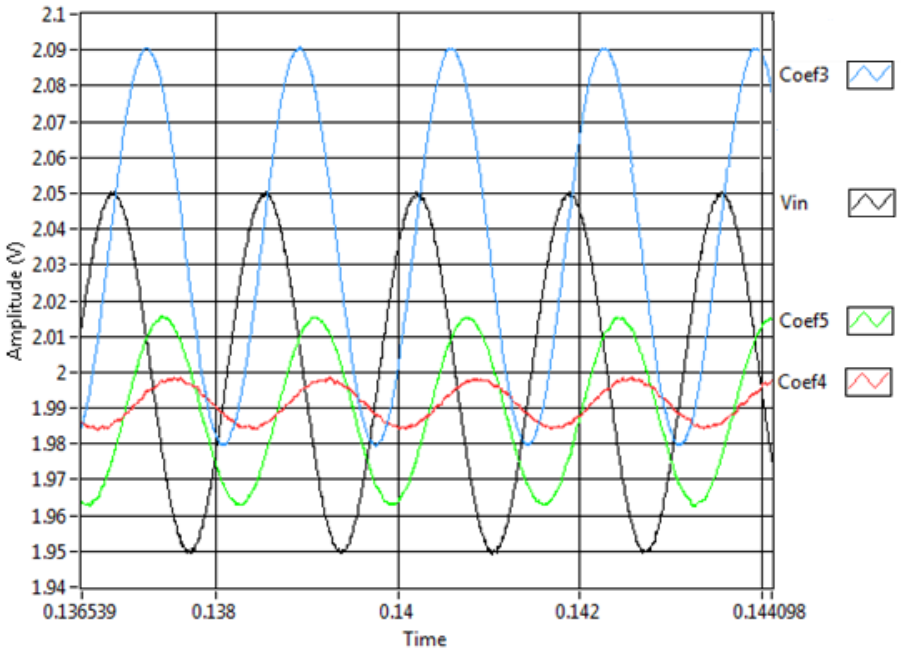


Figura 4.67: Respuesta temporal de los filtros de coeficientes 3, 4, y 5. Tomada de [64]

Tabla 4.35: Tensiones de *offset* de cada coeficiente. Tomada de [64]

Signal	Amplitude (mV)	Offset (mV)
Coeff3	56,14	34,63
Coeff4	7,74	-8,72
Coeff5	26,88	-11,09

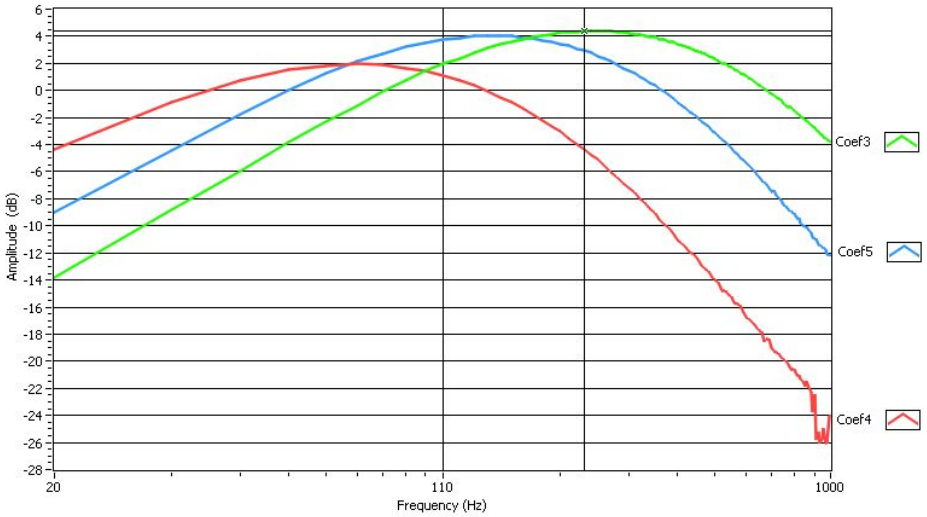


Figura 4.68: Respuesta de frecuencia de los tres coeficientes. Tomada de [64]

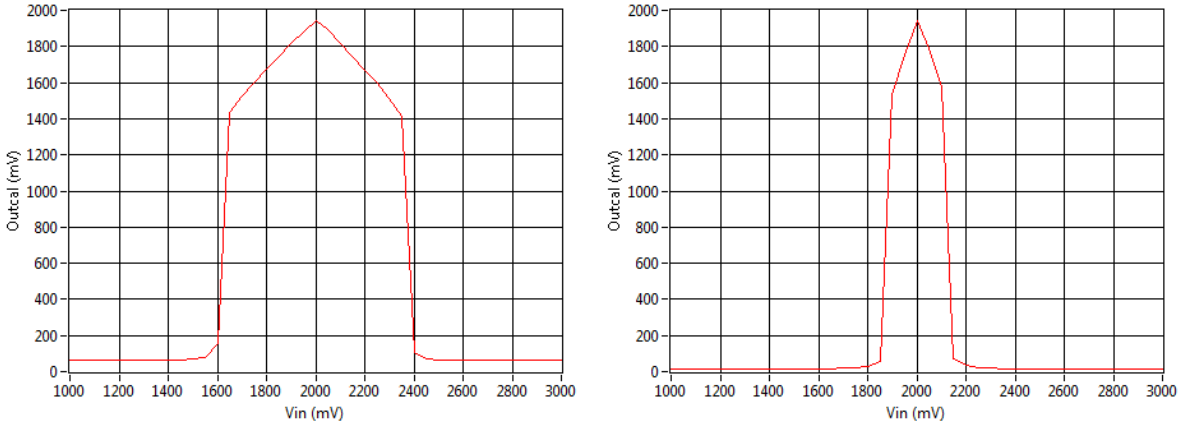


Figura 4.69: Respuesta de la unidad de cálculo. Tomada de [64].

hay una diferencia en las mismas que puede explicar la diferencia a su vez en las corriente de salida, que son versiones escaladas de las corrientes de referencia.

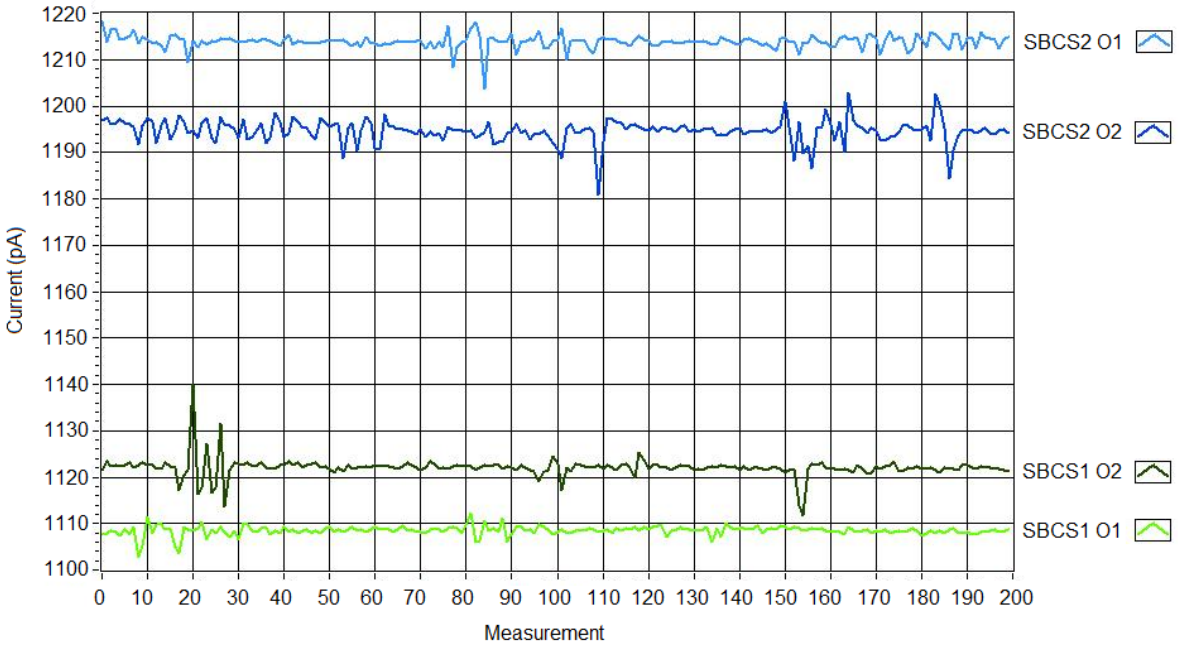


Figura 4.70: Salida de corriente de las SBCS. Tomada de [15].

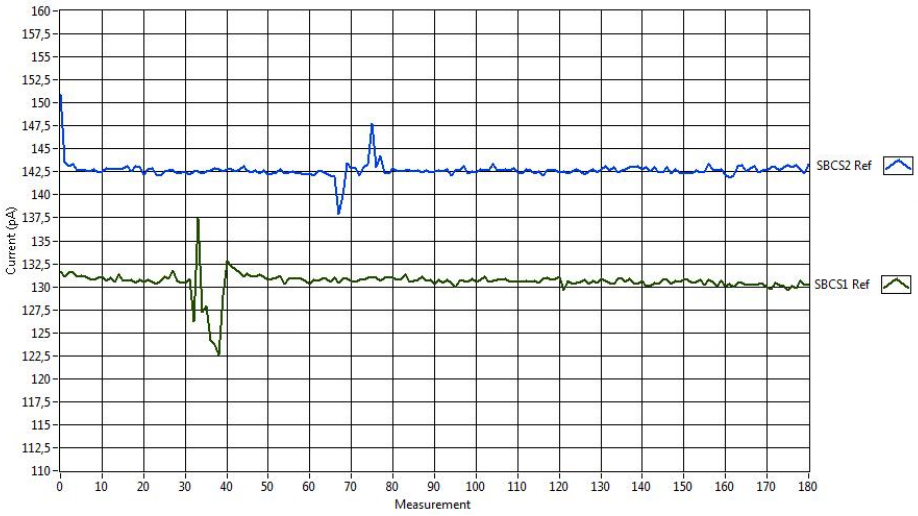


Figura 4.71: Corrientes de referencia de las SBCS. Tomada de [15]

En la tabla 4.36 se resumen las tables de corriente medidas, con respecto a los 2 nA esperados.

La tabla 4.37 muestra los altos porcentajes de error en ambas fuentes. Una razón posible sobre estas divergencias puede obtenerse del análisis de los frentes de Pareto, según muestran las Figs. 4.55 y 4.56. Los cambios drásticos en los frentes dan a entender que

Tabla 4.36: Salidas para cada *SBCS*. Tomada de [15]

Fuente	I Ref (pA ± 10pA)	Iout_1 (pA ± 10pA)	Iout_2 (pA ± 10pA)
SBCS1	130,5	1108,5	1122,0
SBCS2	142,4	1214,0	1195,0

los especímenes hallados son bastante inestables, en el sentido de que una pequeña variación en sus parámetros (como el W o el L en el transistor) puede modificar el punto óptimo de una forma extrema (ver [64] para un análisis más minuciosos). Ambas fuentes también mostraron problemas de arranque que no se presentaron durante la simulación. Sin duda, es necesario retomar el estudio de esta fuente para mejorar su implementación. No obstante, la fuente no era un elemento clave de este proyecto, por lo que se puede considerar que no se afectan los objetivos a alcanzar.

Tabla 4.37: Porcentaje de error para cada *SBCS*. Tomada de [15]

Error	I Ref (%)	Iout_1 (%)	Iout_2 (%)
SBCS1	-48,41	-44,80	-43,90
SBCS2	-43,71	-39,31	-40,27

Por último era necesario hacer una validación estadística de toda la unidad (cálculo y filtros). Para ello, en este momento se está construyendo un banco de pruebas en LabView, sobre el que se montará una colección de sonidos reales, que serán alimentados a la unidad. Será necesario construir en LabView la etapa final del proceso: un filtro promediador y un detector de umbral, tal como se muestra en la estructura general de detección (ver Fig.4.72). Se espera tener resultados de esta prueba para junio. El atraso es debido a que en el momento de generar este informe, el equipo completo se encuentra trabajando en el cierre de la fabricación del clasificador digital SiRPA, para cumplir con el objetivo 4.

4.2.2 Conclusiones parciales sobre detección de patrones acústicos de un disparo mediante un filtro de onditas continuas

Las pruebas de las unidades corroboraron la mejora en términos de consumo de potencia al utilizar la optimización genética. Además, se disminuyó el efecto de las capacidades parásitas de entrada, lo que redundó en un filtro más cerca de las bandas deseadas. Sin embargo, se siguieron teniendo problemas de offset que podrían afectar el desempeño final del detector. Además, el bajo slew-rate de los amplificadores usados en los rectificadores, introducen también no linealidades en el circuito de cálculo de energía.

Como se ha indicado, queda pendiente la prueba estadística final del circuito, que debería determinar en que grado estas imperfecciones afectan la viabilidad de la unidad de detección. Quedan además muchas recomendaciones clave por hacer si se desea explorar más esta opción para proponer un detector fiable:

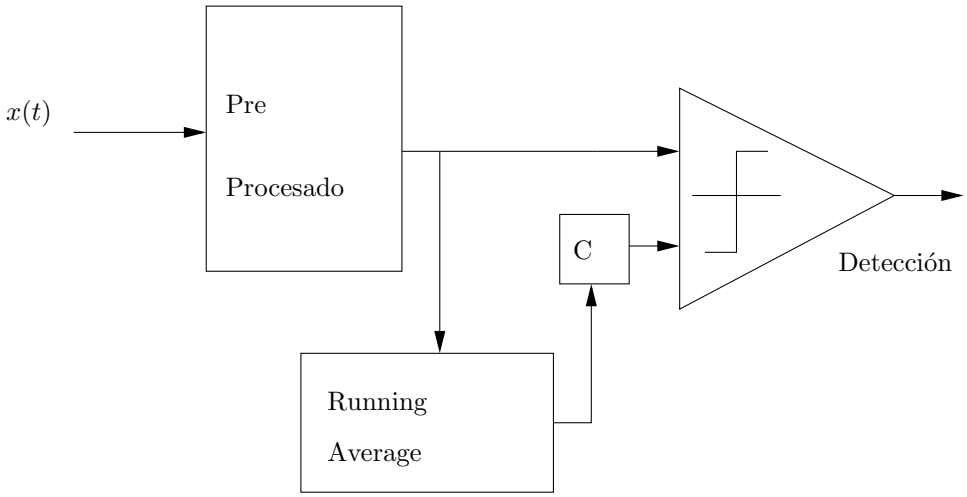


Figura 4.72: Esquema de detección que fue propuesto. El subsistema de procesamiento sería el circuito a verificar (banco de filtros y unidad de cálculo). El resto del esquema de detección se implementará en LabView. Las señales de audio entrarán por $x(t)$.

a Incluir los efectos de variabilidad de transistores MOS en la herramienta de optimización, y vincular estos con los efectos típicos del desapareamiento de transistores en el diseño de amplificadores tales como el offset y la baja del rechazo de modo común. La idea es que al obtener las propuestas de diseños optimizados, ya se tenga una previsión de cómo afectará la variabilidad a los circuitos propuestos. Ya se tiene a un estudiante de maestría que trabaja con los autores de [64] en añadir estas capacidades.

b Utilizar estructuras biquad para construir los filtros, de manera que la realimentación típica de estas estructuras atenúe los efectos del offset en los OTAs.

c Mejorar las técnicas de trazado de los circuitos. El investigador principal estará tomando un curso en el mes de julio en la Universidad Católica del Uruguay para precisamente mejorar su habilidades en esta área.

4.2.3 Detección de patrones acústicos de un disparo mediante un filtro de onditas discretas

Diseño y resultados del filtro de tres bandas

La primera versión del circuito integrado del filtro Haar de capacitores conmutados presenta un comportamiento erróneo, al presentarse ruido en los coeficientes de salida de los filtros (ver Fig. 4.73 y [8, 13]).

Estas diferencias se atribuyeron a dos posibles factores:

1. Existe ruido que se filtra de los pads de conexión.

2. El circuito integrado no se encuentra optimizado para la protección contra el ruido.

Se hicieron pruebas al circuito integrado, tratando de filtrar el ruido en el circuito (ver [8]). Estas mostraron que la causa del error debía provenir de que el circuito no estaba bien protegido contra el ruido.

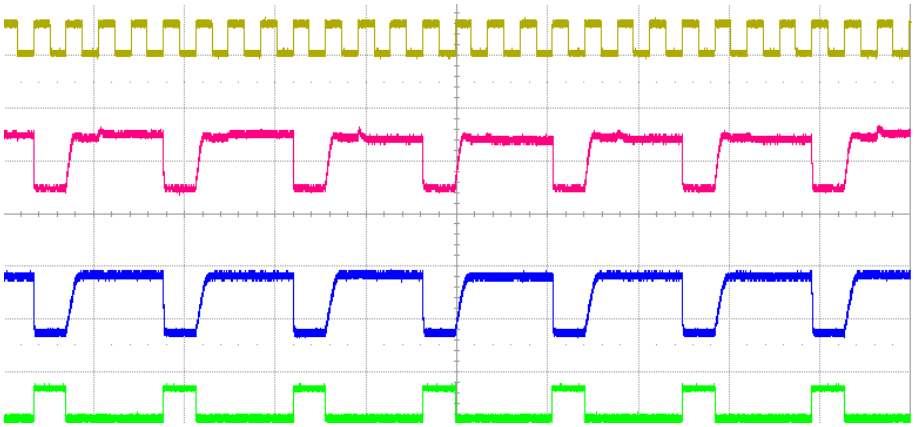


Figura 4.73: Coeficiente de aproximación dos para la respuesta del circuito: señal de reloj $f = 14KHz$, rojo: coeficiente n, azul: coeficiente p, verde: máquina de estados. Nótese la muesca en la señal. Tomada de [8].

La versión 1 del circuito integrado posee una estructura como la que se muestra en la Fig. 4.74 En este diseño los amplificadores operacionales se encuentran junto con los interruptores y en la parte inferior se encuentran los circuitos generadores de reloj. La máquina de estados (FSM) se encuentra cerca del banco de capacitores y de los generadores de reloj, con una orientación de 90 grados.

El rediseño tomó como base las buenas prácticas en el diseño de circuitos de señal mixta propuestas en [32] [45] y [72]. La estructura general del nuevo diseño se puede observar en la Fig. 4.75.

La Fig. 4.75 permite la separación de las señales analógicas y las señales digitales que se van a enrutar por diferentes caminos. También permite la separación de la sección digital y analógica del circuito.

El objetivo principal de estas modificaciones es el de mantener el diseño modular y los niveles de jerarquía correctos en el circuito.

Por otro lado, los amplificadores operacionales son la unidad funcional que opera dentro de estos circuitos. En [13] se usa como amplificador operacional a un cascodo y su diseño se hizo bajo una configuración multi-dedo. Esta configuración le permite colocarse junto a los interruptores en mismo circuito.

Siguiendo los criterios de diseño de [45] se separó el amplificador operacional del circuito original, para colocarlo individualmente. El objetivo era tener un arreglo de amplificadores operacionales separados en el diseño final. Además la estructura interna del amplificador operacional se modificó según se recomienda en [32] a una estructura que redujera el

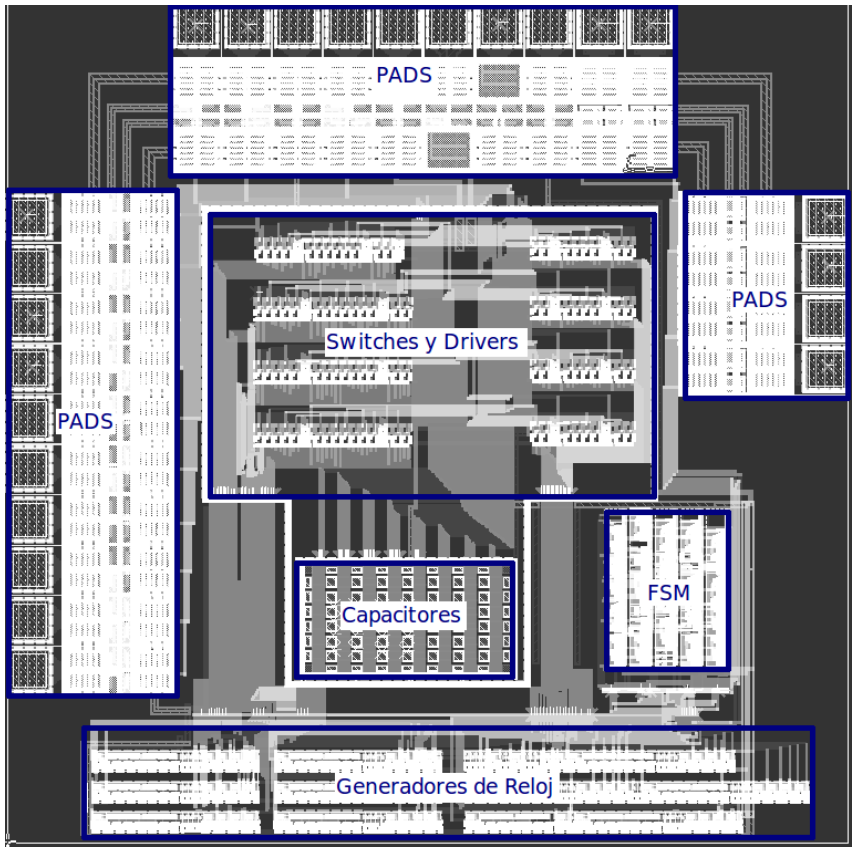


Figura 4.74: Versión uno del circuito integrado. Tomada de [13].

desapareamiento por variabilidad del proceso CMOS. Se usó una configuración de transistores apilados, en la que se separan los transistores n y p en arreglos serie/paralelo. La Fig. 4.76 muestra el esquemático del nuevo diseño del amplificador cascodo. La Fig. 4.77 muestra el diseño del circuito integrado del amplificador cascodo implementado.

Cada unidad de cálculo de coeficientes (tanto de aproximación como de detalle) y cada unidad de muestreo requieren en su diseño un par diferencial de amplificadores operacionales. Se diseñó una unidad que contuviera el par de amplificador operacional necesarios, y los puertos para la conexión con otras unidades. El diseño de esta unidad se muestra en la Fig. 4.78, que corresponde a colocar dos amplificadores como los de la figura 4.77 y posteriormente estandarizar los puertos de entrada y de salida en la parte superior e inferior del circuito, para facilitar la interconexión con otras unidades del circuito.

El circuito general consiste de cuatro unidades de cálculo de coeficientes, y cuatro muestreadores, lo que conlleva a un total de ocho unidades como la mostrada en la figura 4.78. Siguiendo la topología recomendada en [45] todos los amplificadores operacionales debían encontrarse en una única unidad y se debían colocar en la parte superior del circuito total. De igual manera los puertos de conexión se colocaron en la parte superior e inferior para poder facilitar el acceso de los buses para interconectar los circuitos. La figura 4.79 muestra el diagrama de todos los amplificadores operacionales.

La figura 4.79 corresponde al primer bloque definido en la figura 4.75.

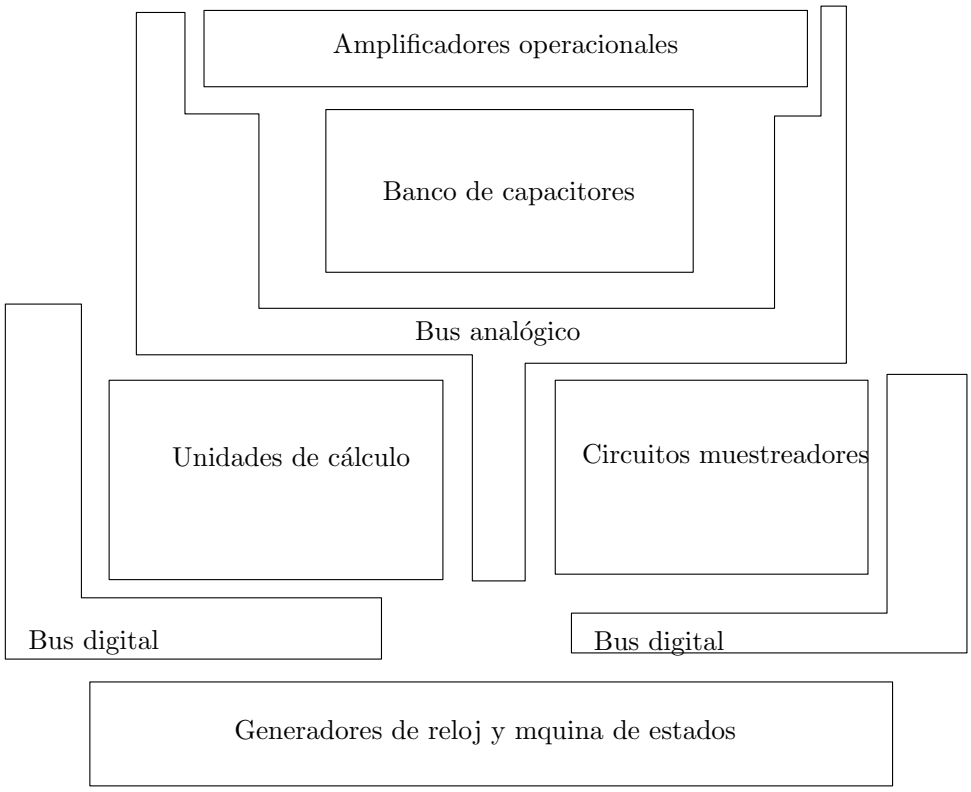


Figura 4.75: Diagrama propuesto para la version dos del circuito integrado. Tomada de [8].

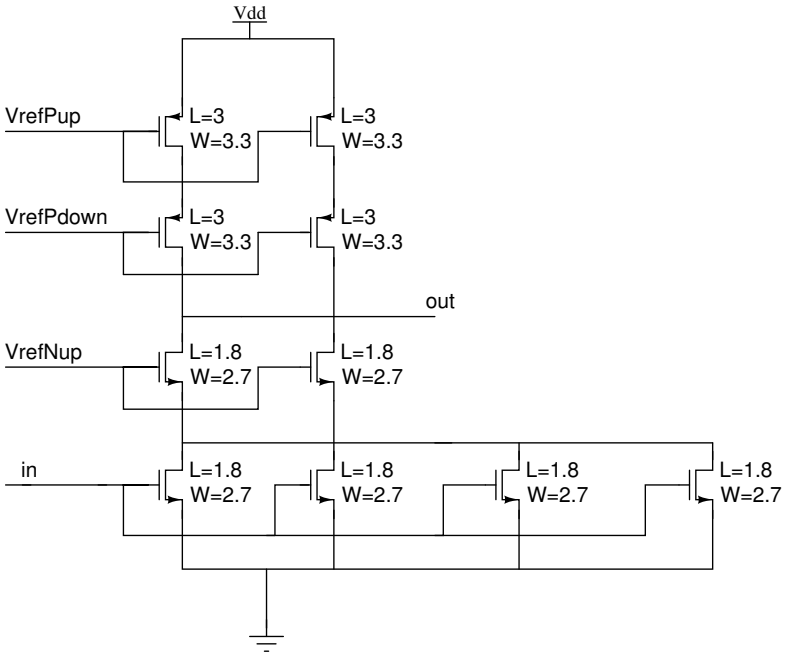


Figura 4.76: Diseño del esquemático del circuito del amplificador operacional cascodo. Tomada de [8].

Diseño de las unidades de cálculo de coeficientes y muestreo

El filtro Haar consta de un arreglo de circuitos que calculan coeficientes; uno de ellos es el coeficiente de aproximación dos, y con base en éste se calculan los coeficientes de detalle

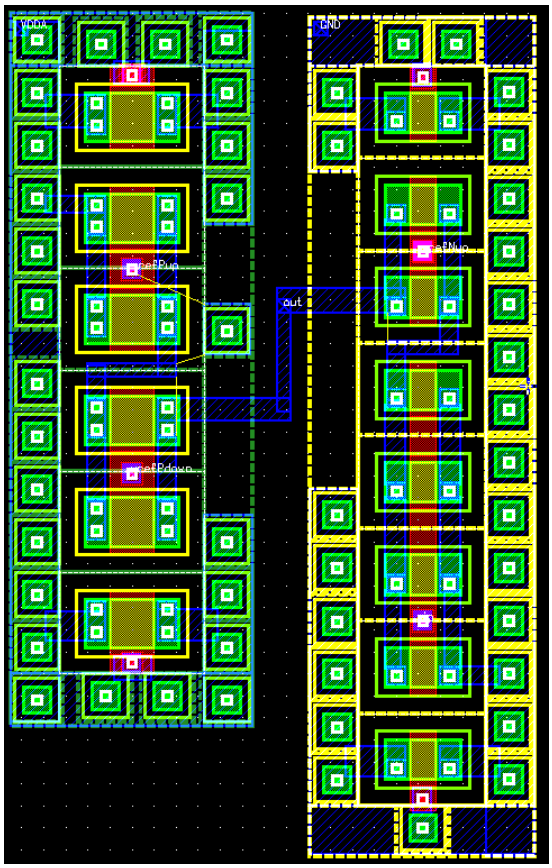


Figura 4.77: Diseño del circuito del amplificador operacional cascodo usando transistores apilados. Tomada de [8].

3, 4 y 5. Estas unidades de cálculo están diseñadas mediante capacitores conmutados. La Fig.4.80 muestra el esquemático de la unidad de cálculo del coeficiente de aproximación dos. El resto de circuitos sigue una lógica similar, donde solo varían los valores de los capacitores y la secuencia de activación de los interruptores.

Como se menciona en [32, 45], a la hora de diseñar el circuito de la figura 4.80 se deben separar los elementos, es decir, en un mismo sector no pueden convivir interruptores, amplificadores operacionales y capacitores. El esquemático de los circuitos implementados para calcular los coeficientes de aproximación y detalle se muestran en las Figs. 4.80 y 4.81, teniendo claro que los interruptores se colocarán lejos de los amplificadores y los capacitores.

El esquemático de los circuitos implementados para calcular los coeficientes detalle se muestra en las Fig. 4.81, teniendo claro que los interruptores se colocarán lejos de los amplificadores y los capacitores al igual que en el de la Fig. 4.80, y en los circuitos de muestreo.

Al dejar por fuera de la unidad de cálculo los amplificadores y los capacitores, éstos son ubicados en zonas lejanas. La figura 4.82 muestra los caminos que deben seguir las señales para llegar a los respectivos componentes. El alambrado de las secciones debió también hacerse cuidadosamente para evitar el acople de ruido entre las distintas secciones.

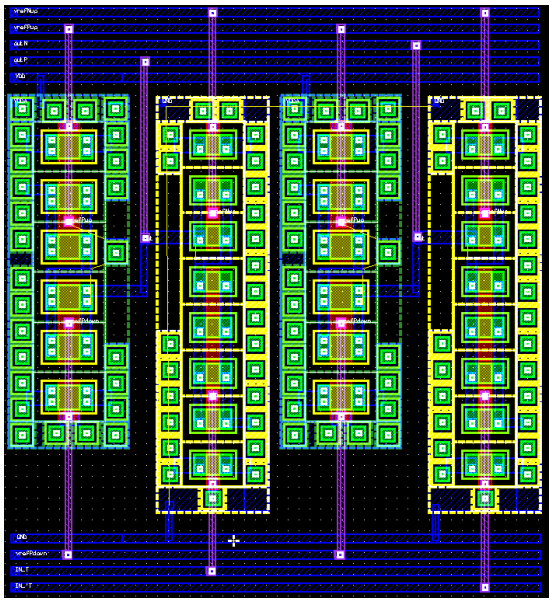


Figura 4.78: Diseño del cascodo dual para las unidades de calculo de coeficientes y muestreadores. Tomada de [8].

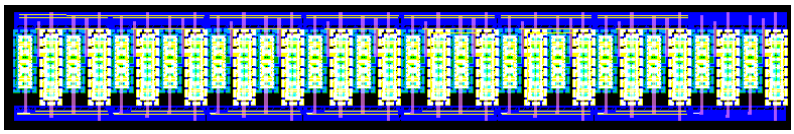


Figura 4.79: Diseño del bloque completo con todos los amplificadores operacionales del filtro Haar. Tomada de [8].

El circuito final se muestra en la figura 4.83 Las modificaciones realizadas fueron la eliminación de los amplificadores operacionales y las líneas de alimentación y de entrada y salida de los mismos, y agregar las señales de enrutado. En el circuito quedaron únicamente los interruptores que operan con las señales digitales que llegan desde los circuitos generadores de reloj. Estas señales fueron protegidas con una capa de poly y se alimentaron por la parte inferior del circuito. Las señales analógicas están en la parte superior del circuito. Se aislaron de la señales digitales con los interruptores y las conexiones a tierra que poseen los mismos; son estas señales las que se comunican con los capacitores y los amplificadores operacionales y se enrutan por caminos diferentes a las señales digitales según se recomienda en [45].

Se agregaron salvaguardas entre las conexiones digitales y analógicas para evitar la filtración de ruido como se describe en [32]. Estas salvaguardas consisten en capas de metal conectadas continuamente a un pozo p que esta conectado al potencial de referencia. Esta conexión permite que excesos de electrones o huecos sean redireccionados al potencial y no alteren señales que pasan cerca. la figura 4.84 muestra un acercamiento a las salvaguardas que se implementaron en el circuito.

El trazado final se basó en la figura 4.75: para su diseño se tomaron las siguientes consideraciones:

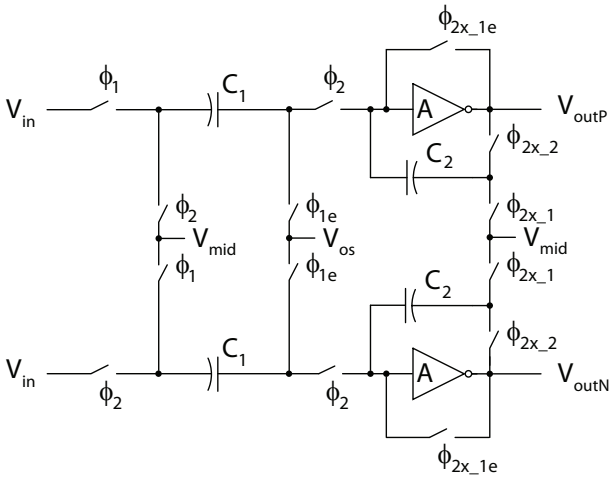


Figura 4.80: Esquemático del circuito del coeficiente de aproximación 2 (cA_2). Tomada de [12].

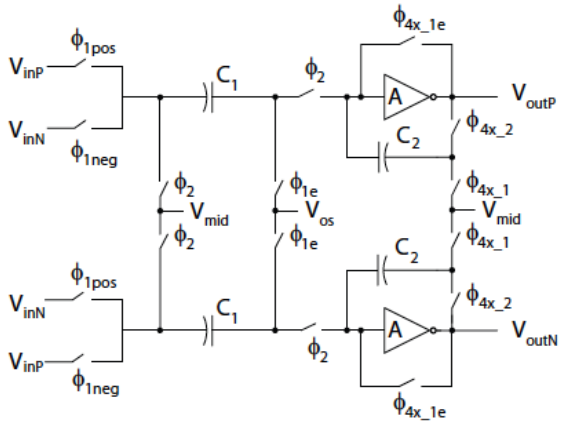


Figura 4.81: Esquemático del circuito del coeficiente de detalle 3 (cD_3). Los coeficientes de detalle 4 y 5 tienen la misma estructura, variando solo la secuencia de temporizado. Tomada de [12].

1. La sección digital debía estar separada y aislada de la sección analógica
2. Los enrutamientos horizontales fueron hechos con metal 1 y metal 3.
3. Los enrutamientos verticales fueron hechos con metal 2.
4. La sección digital del circuito se alimentó con una fuente de tensión diferente de la tensión de alimentación de la sección analógica

El diagrama final del filtro Haar se muestra en la figura 4.85

En resumen, para el diseño final mostrado en la figura 4.85 se tomaron las siguientes consideraciones:

1. Se siguió el modelo propuesto en la figura 4.75

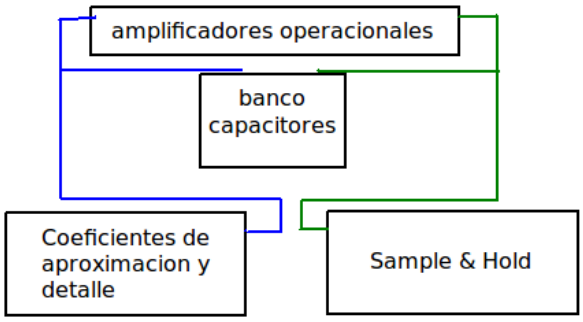


Figura 4.82: Conexiones para integrar interruptores, amplificadores operacionales y banco de capacitores. Tomada de [8].

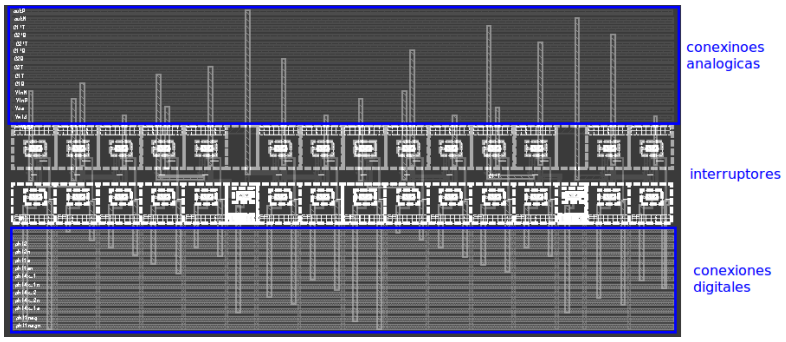


Figura 4.83: Unidad de cálculo sin los amplificadores cascodo, siguiendo el diseño de la figura 4.80. Tomada de [8].

2. Se utilizó el banco de capacitores que ya estaba diseñado, ya que no requería modificaciones.

3. La máquina de estados se colocó con una orientación de 0 grados, para evitar variaciones en la lógica que alteraran el funcionamiento de las señales de sincronía.

4. Se colocaron salvaguardas a lo largo del bus digital para minimizar su influencia sobre las señales analógicas.

5. Los buses analógicos y digitales viajan por rutas separadas.

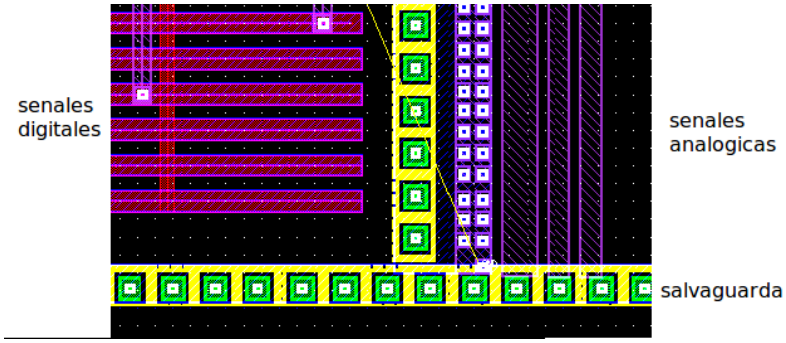


Figura 4.84: Ejemplo de salvaguardas utilizadas en el diseño del filtro Haar. Tomada de [8].

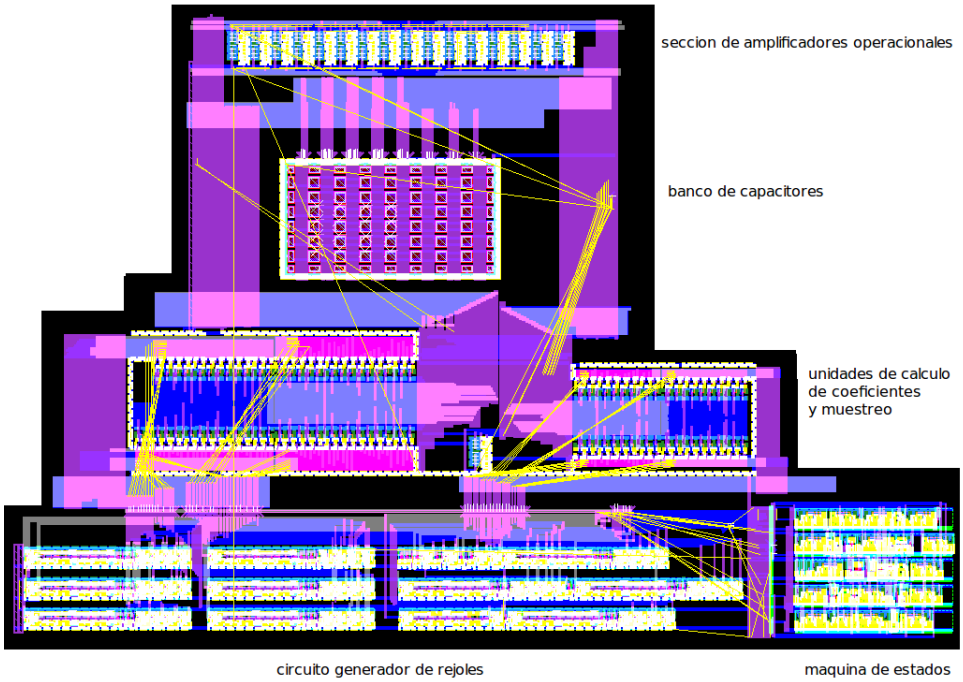


Figura 4.85: Diagrama total del filtro Haar. Tomada de [8].

6. Se previó la colocación de las terminales al lado derecho e inferior del circuito.

7. Las capas de metales se implementaron de forma que dos capas adyacentes no viajaran en la misma dirección, o que fueran perpendiculares entre sí.

Otra modificación que se hizo al circuito, fue la separación de las fuentes de alimentación del circuito digital como del circuito analógico en los pads de conexión.

La figura que se muestra en 4.86 explica mejor cada uno de los segmentos del circuito final.

Pruebas realizadas en simulación

La Fig. 4.87 muestra el comportamiento de la versión uno del circuito integrado. Esta gráfica muestra las señales de alimentación, y la interferencia que existe entre las mismas. Esta interferencia es causada por los fenómenos descritos en [32] sobre el manejo del ruido en las señales de alimentación.

Luego de las modificaciones necesarias al circuito se observa que dicho ruido ha desaparecido (ver Fig. 4.88)

El consumo de potencia fue de $40.464\mu W$ en simulaci[on.

La tabla 4.38 muestra una comparación entre el consumo de las dos versiones del circuito. La mejora fue de un 14%.

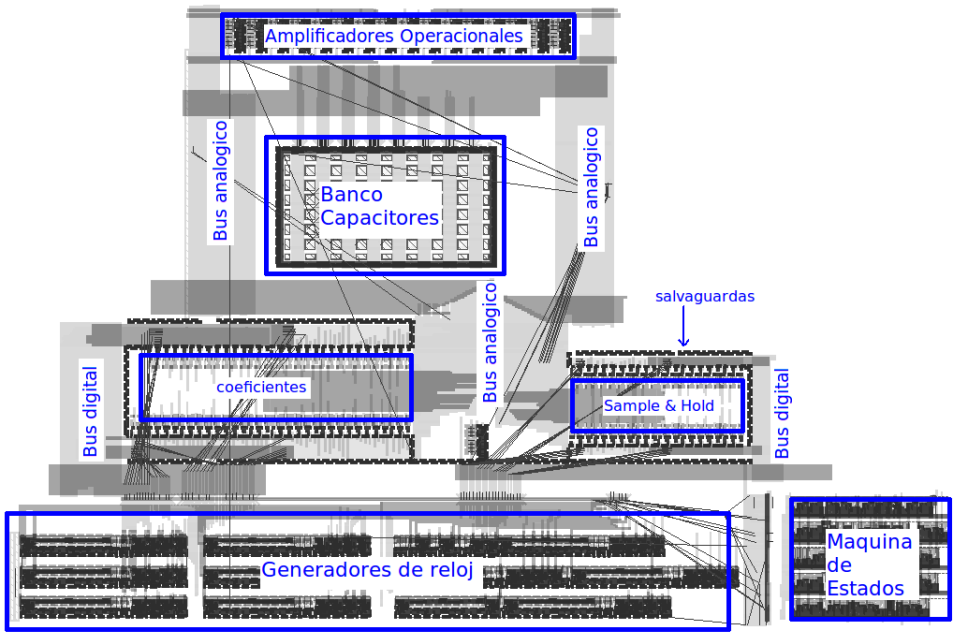


Figura 4.86: Diagrama total del filtro Haar implemetado y enviado a fabricar (sin pads). Tomada de [8].

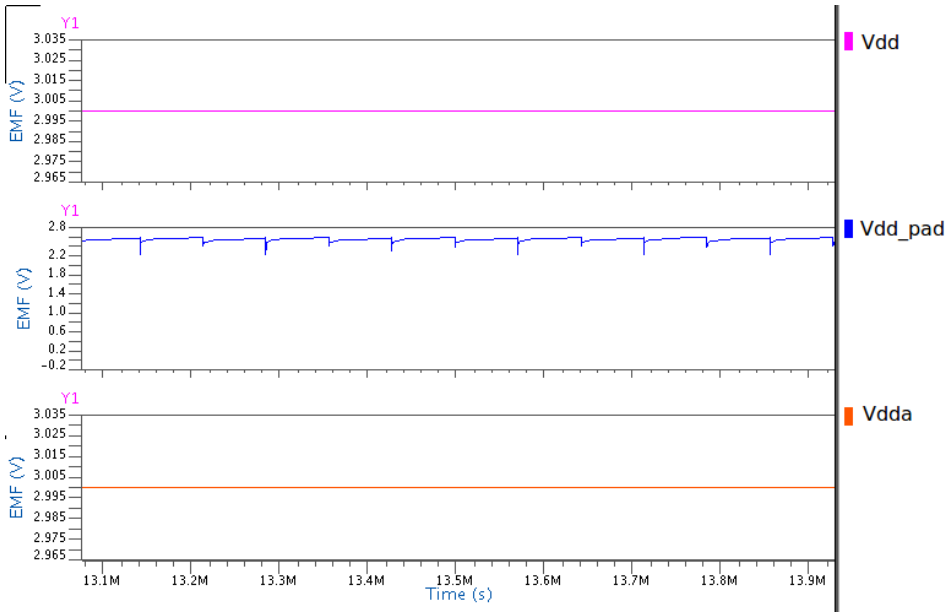


Figura 4.87: Tensiones de alimentación del circuito versión uno. Tomada de [8].

Con respecto al área, la tabla 4.39 compara el área de ambas versiones del circuito. El área se incrementó en un 33% básicamente por el cambio en el formato del diseño.

El aumento en área es justificable, ya que responde a un diseño que está mejor estructurado, sigue un diseño especialmente definido para circuitos de señal mixta.

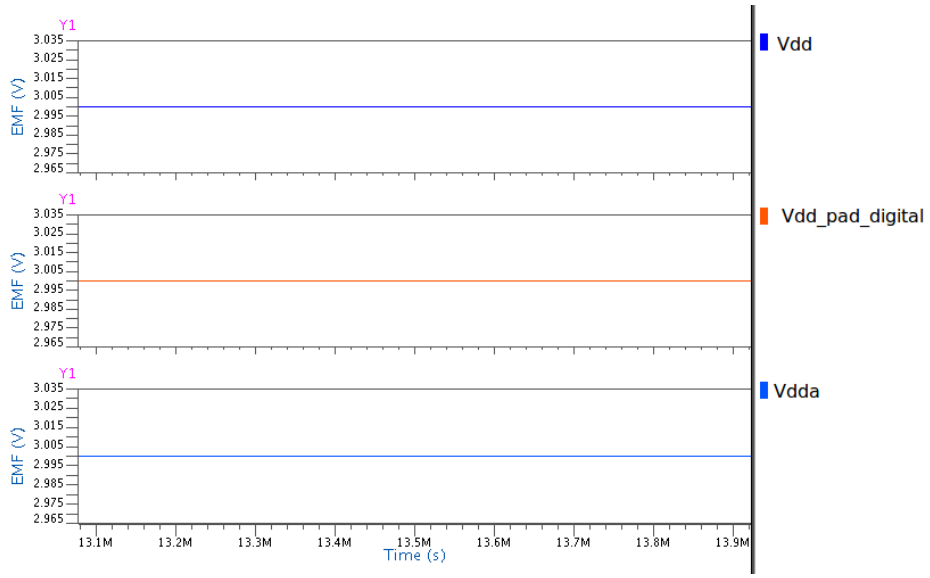


Figura 4.88: Tensiones de alimentación del circuito versión dos. Tomada de [8].

Versión del circuito	Consumo de potencia (μW)
Uno	47.09
Dos	40,46

Tabla 4.38: Consumo de potencia de las dos versiones del circuito integrado. Tomada de [8].

Resultados de mediciones

Para la prueba del filtro Haar se utilizó el circuito de pruebas propuesto en [13, 8]. El banco de pruebas fue el mismo ya mencionado que se usó para el filtro de onditas analógico y la unidad anteriormente probados (ver [15] para más detalles). Fue necesario necesario utilizar un seguidor de voltaje en la salida de todos los coeficientes, pues es generada por un amplificador cascodo de muy alta impedancia. Desgraciadamente, tal y como puede observarse en la Fig. 4.89, los coeficientes P y N muestran aún interferencia debida a las señales de reloj. Todas las pruebas sub-secuentes mostraron el mismo patrón para los coeficientes P y N, tanto para las pruebas con entradas CD como para entradas sinusoidales. Distintas pruebas fueron realizadas tratando de averiguar la fuente de dicho error, que no obstante no pudo resolverse.

Versión del circuito	Área (mm^2)
Uno	1,464
Dos	1,944

Tabla 4.39: Comparación de las áreas de las dos versiones del circuito integrado. Tomada de [8].

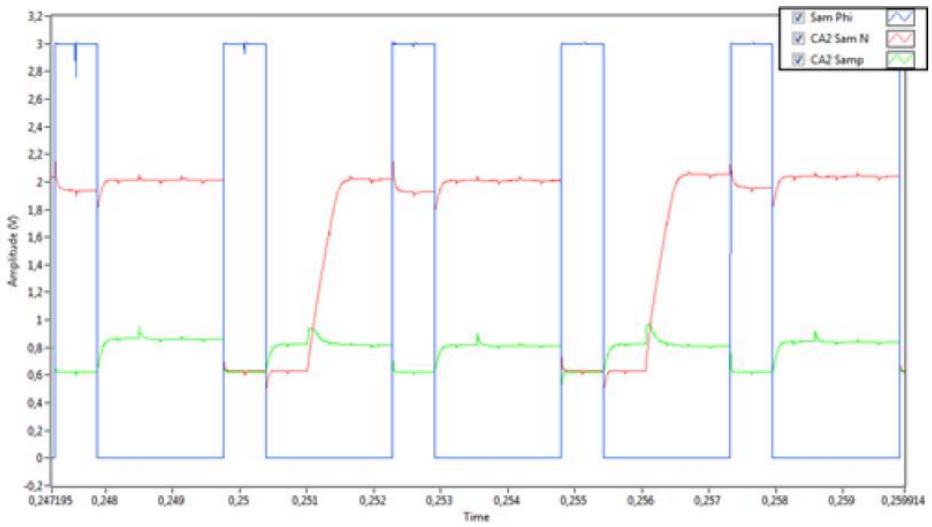


Figura 4.89: Tensiones de salida del coeficiente de aproximación 2 del circuito. Nótense las muescas que produce el reloj sam_{phi} en los coeficientes. No fue posible hallar una explicación para este error. Tomada de [15].

4.2.4 Conclusiones parciales sobre detección de patrones acústicos de un disparo mediante un filtro de onditas discretas

Las pruebas de las unidades siguieron mostrando un problema de ruido en los coeficientes del filtro. No fue posible hallar la causa del mismo, pese a la consulta con el Dr. Milutin Stanacevic y su estudiante doctoral. Dados dichos problemas, y la falta de tiempo del investigador principal para trabajar más sobre este diseño, se decidió descartar esta opción de implementación del detector.

Como recomendaciones principales para próximos proyectos, se sugiere:

a Desarrollar habilidades en los investigadores sobre topologías de capacitores conmutados, y su implementación microelectrónica.

b Fortalecer el dominio matemático del procesamiento discreto mediante estructuras de capacitares conmutados. Ya se ha recibido aprobación por parte de la VIE para una actividad de fortalecimiento a partir del segundo semestre del 2014, para precisamente fortalecer conocimientos en esta área.

c Al igual que para el caso anterior, se deben mejorar las técnicas de trazado de los circuitos del investigador principal. Se espera que el curso que éste estará tomando en el mes de julio ayude en esta área también.

4.3 Estructuras integradas de detección de patrones acústicos de motosierras

Los objetivos 2 y 3 explicitan la necesidad de desarrollar la unidad de detección de motosierras, encargada de despertar a la unidad de clasificación, según la jerarquía definida en [13]. Esta sección se dedica a describir tanto el estudio preliminar de algoritmos propuestos como los avances logrados en la implementación de los mismos.

4.3.1 Análisis de algoritmos propuestos para la detección preliminar de motosierras

Se propusieron inicialmente para la posible detección de sonidos de motosierra los siguientes algoritmos:

1. Detección por transformada de onditas.
2. Detección por análisis de periodicidad.

Los sonidos de motosierra son generados por motores eléctricos o de combustión interna, por lo que presentan patrones periódicos cuando se están utilizando. Los periodos de los sonidos pueden variar a diferentes velocidades de la motosierra. Para evaluar los algoritmos propuestos se usaron las grabaciones a indicadas en los apartados anteriores. Para más detalles sobre este análisis, ver [93].

Utilización de transformadas wavelets o de onditas

Al igual que en los objetivos relacionados con la detección de disparos, se propuso en [93] primero utilizar un banco de filtros de onditas como etapa de preprocesamiento para la detección. Tal como se conoce de la teoría de las transformadas de onditas, estas intentan expresar una señal $x(k)$ con K muestras en total, mediante una expansión de términos o coeficientes proporcionales al producto interno entre la señal y diferentes versiones escaladas y trasladadas de una función prototipo ψ más conocida como *ondita madre*. Para asemejar la ondita madre al segmento de la señal analizada se producen traslaciones enteras y escalamientos de una única función ondita ψ [77].

La motivación para usar onditas en la detección de disparos se basó en el hecho de que estas son buenas la detección de transiciones abruptas, al producir coeficientes de onditas grandes centradas en las discontinuidad de la señal. En la detección de señales periódicas suaves se puede detectar algunas características pero no de manera sencilla, ya que los coeficientes producidos no son de gran magnitud; además que hay que utilizar una ondita generadora muy especifica para cada caso que se desee analizar. La detección de un disparo mediante la transformada de onditas es entonces relativamente eficiente ya que el disparo se presenta como una cambio abrupto de la señal, además de que presenta altos niveles

energéticos. Este estudio se llevó a cabo en otra tesis de licenciatura, cuyos resultados no se incluyen en este proyecto por razones de espacio, donde se evaluaron alternativas digitales a la detección de sonidos de disparos por medios analógicos estudiada en este proyecto (ver [66] para más detalles). Para señales periódicas este proceso es mucho más complicado, por lo que una transformada de onditas pudiera no ser práctica para analizar ondas de este tipo. Esto motivó a no desarrollar mucho el análisis de esta modalidad de detección (para más detalles ver [93]).

Medición por periodicidad de señal

El método de medición de periodicidad consiste en poder analizar una señal $x(k)$ con diferentes valores de periodicidad "P", para así obtener cuales valores contiene la señal (para más detalle de la teoría detrás de esta implementación, ver [93, 31].

Una vez que se conoce los valores de las posibles periodicidades a detectar, se puede diseñar un detector que analice las señales y encuentre un aproximado al valor de periodicidad principalmente de la onda fundamental para determinado sonido. Ya que los motores de combustión interna generan sonidos periódicos por causa de sus mecanismos internos, el análisis de periodicidad parece ser un método eficiente para realizar detecciones para estos sonidos. Para llevar a cabo este análisis se podrían utilizar algoritmos como el de autocorrelación, que tiene como función la búsqueda de similitud entre los mismos datos de una señal utilizando un índice de desplazamiento o retado de tiempo. Una vez obtenidas las relaciones de autocorrelación, se analizaría el resultado mediante el algoritmo de periodicidad para detectar niveles bajos o altos de energía dentro de las muestras. Cuando se utilizan niveles altos valores en los retrasos de prueba en el algoritmo para medir periodicidad, se puede obtener valores altos o bajos como resultado. Si existen niveles altos en el cálculo, estaríamos en presencia de niveles periódicos en la señal de entrada analizada; en cambio, cuando se obtienen niveles bajos en el algoritmo se puede suponer la no presencia de muestras periódicas en esa porción de señal. De esta manera al tener un nivel cuando se detectan o no valores de periodicidad en la señal, se puede determinar un umbral de detección para cuando existe cierto nivel en el resultado de medición de periodicidad.

Como primer paso fue necesario medir los valores de periodicidad presente en las grabaciones disponibles. Para poder realizar la estimación de máxima verosimilitud, se definió a la señal que se va a analizar como

$$x[k] = s[k] + n[k] \tag{4.21}$$

donde

$s[k]$: Señal periódica.

$n[k]$: Ruido.

La señal $s[k]$ se puede considerar con una repetición de un subsegmento $q[k]$ que depende de los valores de periodicidad a los que se les realizará el análisis.

La función que porta la periodicidad está compuesta por los segmentos $q[k]$ para los deferentes valores de periodicidad como se muestra en (4.22).

$$s[k] = q[k \bmod P] \tag{4.22}$$

Para cada valor de "P" se obtiene.

$$q_P[k] = \frac{1}{L}\sum_{l=0}^{L-1} x[k + lP] \tag{4.23}$$

con $L = K/P$. De esta manera cuando se obtienen los diferentes valores $q_P[k]$, se analizan los vectores que maximizan la energía mediante (4.24) para cada 'P".

$$\mathscr{E}_x(P) = L\sum_{k=0}^{P-1}(q_P[k])^2 \tag{4.24}$$

Así, con el valor de periodicidad que maximizó la energía en (4.24), se procede calcular la estimación de máxima verosimilitud

$$\mathscr{P}_{MLE} = \frac{\mathscr{E}_S}{\mathscr{E}_N} = \frac{\sum_{k=0}^{K-1}(s[k])^2}{\sum_{k=0}^{K-1}(x[k] - s[k])^2} \tag{4.25}$$

donde "K" es la cantidad de muestras a analizar y el vector $s[k]$ es el que contiene las muestras con el valor "P" determinado por la máxima energía en (4.24).

Se usaron vectores para las pruebas equivalentes con las señales de audio muestreadas a 48kHz, subdivididos en bloques de 48000 muestras, es decir, bloques de un segundo (ver [5] para el detalle de las muestras utilizadas, a partir de las grabaciones realizadas en el bosque lluvioso tropical). Se realizó un barrido de valores de periodicidad utilizando las ecuaciones anteriormente mostradas, en los sectores de señales escogidas donde se distinguen perfectamente los sonidos de motosierra y de lluvia. Se trató de buscar esos valores donde la energía se maximiza para los valores de periodicidad analizados. Una vez que se tuvieron los vectores analizados, se generó un espectrograma de los mismo. Las tonalidades claras representan una alta periodicidad de las muestras para cada valor "P" analizado. Las tonalidades oscuras representan baja periodicidad.

La Fig. 4.90 muestra un análisis de estimación de máxima verosimilitud para una grabación con sonido de motosierra con una duración de 21 segundos. La figura 4.91 presenta el mismo análisis para la grabación de sonido de lluvia con una duración de 14 segundos.

Analizando la Fig. 4.90 se pueden encontrar ciertos patrones de líneas color claro y que son horizontales que representan alta estimación de periodicidad para cada sector de 1

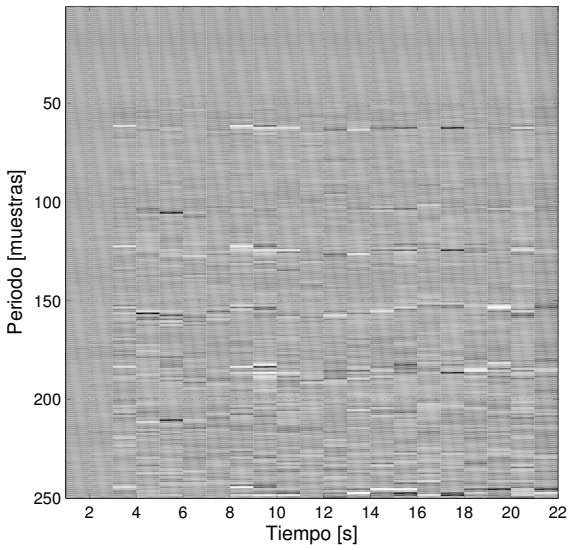


Figura 4.90: Estimación de periodicidad para grabación de sonido de motosierra. Tomada de [93].

segundo de longitud. Por ejemplo para un valor "P" de aproximadamente 150, se puede ver que presenta uno de estos patrones alrededor de los 22 segundos analizados del vector con sonido de motosierra. Analizando la totalidad de la gráfica, obtenemos los valores de periodicidad en unidades de muestras, con valores de 60, 105, 125, 155, 180, 210 y 240.

Si se transforman estos valores a unidades en Hertz, tenemos los valores de las armónicas representativas de los sonidos de motosierra y de esta manera se sabe en qué sector del espectro se localizan estos. Para obtener estos valores será necesario hacer uso de la ecuación (4.26).

$$P(Hz) = \frac{1}{P_N X \frac{1}{48000}} \tag{4.26}$$

donde P_N es el periodo obtenido del gráfico de estimación de periodicidad presentado en la Fig. 4.90.

Con los valores obtenidos, se tabulan los resultados para los diferentes valores "P" analizados. Estos datos se muestran en la Tabla 4.40.

Mediante la ecuación (4.26) se construye la Tabla 4.40 donde se muestra la equivalencia de periodo en unidades de muestras con periodos pero en unidades de Hertz. Estos valores representan ondas que poseen características de periodicidad en los sonidos de motosierra definidos y estos se pueden utilizar para su posible reconocimiento en rangos de frecuencia específicos.

Por otro lado, para la gráfica de la Fig. 4.91, no se puede ver a simple vista un patrón de líneas de color horizontales durante la mayoría de tiempo de la porción de grabación analizada. Los bloques de colores se encuentran dispersos y desordenados a diferencia de

Tabla 4.40: Conversión de periodo para las estimaciones de periodicidad obtenidas de la grabación de sonidos de motosierra de la Fig. 4.90. Tomada de [93].

Estimación de periodicidad [Muestras]	**Periodo [Hz]**
60	800.0
105	457.1
125	384.0
155	309.7
180	266.7
210	228.6
240	200.0
250	176.0

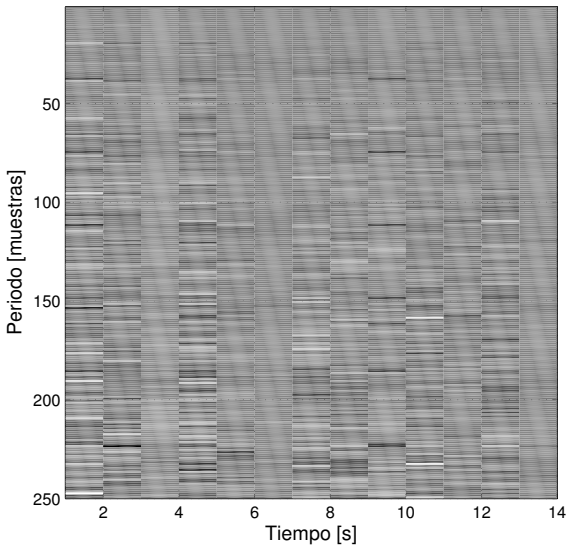


Figura 4.91: Estimación de periodicidad para grabación de sonido de lluvia. Tomada de [93].

los resultados obtenidos para la grabación con sonidos de motosierra, por lo que se puede inferir que los sonidos de lluvia no poseen dentro del espectro de frecuencia un patrón definido de señales periódicas.

Es por tanto factible detectar la señales periódicas incluidas en los sonidos de motosierra y diferenciarlos de todos los sonidos que se pueden encontrar en la naturaleza.

No obstante, el algoritmo utilizado para realizar la estimación de periodicidad conlleva ecuaciones avanzadas y de alto cálculo computacional, por lo que posiblemente no pueda ser utilizado para el propósito que se anda buscando como un futuro diseño de hardware sencillo con un consumo bajo de potencia. Por esto fue necesario modificar el algoritmo para disminuir sus necesidades de procesamiento computacional.

Propuesta de filtro para el paso de armónicas fundamentales

Para aumentar el porcentaje de eficiencia del detector, su usó un filtro que no dejara pasar las partes de la señal capturada que no interesan y permitir el paso principalmente la armónica fundamental de los sonidos periódicos producidos por los motores de las motosierras.

Mediante la herramienta LabVIEW de National Instruments se llevó acabo un análisis para determinar la armónica fundamental y los diferentes tonos que poseen las grabaciones que se tenían para el proyecto, para así determinar la banda de frecuencia donde estaban presentes las armónicas fundamentales de los sonidos de motosierra y dejar afuera de la banda las armónicas que contienen el resto de los sonidos como los de disparos. Los resultados obtenidos para los sonidos de motosierra se presentan en la Tabla 4.41.

Tabla 4.41: Armónicas fundamentales para sonidos de motosierra obtenidas con funciones de procesamieto digital de señales. Tomada de [93].

Sonido	Frecuencia de armónica fundamental [Hz]
D030m_Saw_000	125.34
D030m_Saw_090	127.13
D030m_Saw_180	139.68
D090m_Saw_000	148.42
D090m_Saw_090	140.59
D090m_Saw_180	147.82
D250m_Saw_000	20.76
D250m_Saw_090	20.16
D250m_Saw_180	17.89
D600m_Saw_000	22.13
D600m_Saw_090	21.21
D600m_Saw_180	19.31

La nomenclatura para los nombres de la Tabla 4.41 representan la distancia a la que fueron hechas cada una de las grabaciones, además del ángulo con respecto a dirección normal en la que se posicionó el micrófono del grabador digital. El número posicionado

después de la "D" mayúscula representa la distancia del micrófono a la motosierra. El número al final del nombre representa el ángulo con el que se hizo la grabación.

Las frecuencias de las armónicas fundamentales para los sonidos de motosierra para distancias de 250 y 600 metros a diferentes ángulos presentan un gran cambio con respecto a las frecuencias para distancias de 30 y 90 metros. Esto se debe a que a distancias largas los sonidos se distinguen menos y se confunden o esconden dentro de los sonidos de ruido generados por el bosque.

Los resultados para las armónicas fundamentales de los demás sonidos se presentan en la Tabla 4.42.

Tabla 4.42: Armónicas fundamentales obtenidas para sonidos de lluvia, viento, aeroplanos y disparos de diferentes tipos de armas a variadas distancias y ángulos de adquisición. Tomada de [93].

Sonido	Frecuencia de armónica fundamental [Hz]
A_Birds_001	20.74
A_Plane_001	87.13
A_Rain_001	21.32
A_Wind_001	20.42
D030m_C22_000	956.57
D030m_C22_090	1540.00
D030m_C22_180	20.24
D030m_Pi9_000	20.75
D030m_Pi9_090	20.06
D030m_Pi9_180	20.68
D030m_R32_000	21.52
D030m_R32_090	20.92
D030m_R32_180	19.99
D030m_R38_000	21.08
D030m_R38_090	455.70
D030m_R38_180	19.36
D090m_C22_000	18.85
D090m_C22_090	18.57
D090m_C22_180	18.97
D090m_Pi9_000	478.77
D090m_Pi9_090	19.27
D090m_Pi9_180	25.51
D090m_R32_000	22.53
D090m_R32_090	19.83
D090m_R32_180	23.71
D090m_R38_000	18.40
D090m_R38_090	18.84
D090m_R38_180	20.07

Analizando las frecuencias de la armónica fundamental obtenidas para sonidos que no son de motosierra, se puede observar que la mayoría se mantienen en las frecuencias cercanas al valor de los 20Hz. Algunas otras presentan valores cercanas a 500Hz y 1KHz como lo

son las grabaciones para las armas C22 a 30 metros, la R38 a 30m y la Pi9 a 90m.

Una vez analizados los resultados de la Tabla 4.41 y 4.42, es posible proponer un filtro digital que deje pasar las frecuencias de las armónicas fundamentales de los sonidos de motosierra y que bloquee los demás sonidos que no nos interesan y que podrían causar alguna falsa detección en los bloques de análisis para encontrar las señales periódicas de las que se componen los sonidos de motosierra. Un punto importante es que las grabaciones utilizadas se realizaron con un grabador digital que presenta en su configuración una frecuencia de muestreo de 48KHz. Este valor produce grabaciones de alta calidad pero con gran cantidad de muestras por segundo (48000), que son difíciles de procesar para un hardware limitado y sencillo, por lo cual se ha propuesto realizar un submuestreo de las grabaciones de 48KHz a 1KHz para sobrellevar este inconveniente. Al tratar de realizar este submuestreo es necesario tener en cuenta que para realizarlo se debe de cumplir con el teorema de muestreo o muestreo de Nyquist. Según este teorema, es posible reconstruir una señal si se utiliza como frecuencia de muestreo al menos el doble de la frecuencia máxima encontrada en la señal, como se denota en la ecuación (4.27).

$$F_s > 2F_c + B \tag{4.27}$$

donde:

F_s: Frecuencia de muestreo.

F_c: Frecuencia central.

B: Ancho de banda.

Así se puede concluir que la mayor frecuencia que se puede contener en una grabación muestreada a 1KHz es de 500Hz como máximo. Como las frecuencias de las armónicas fundamentales de los sonidos de motosierra sen encuentran entre los valores de 100-200Hz, es posible albergar el conjunto de señales que nos interesan y discriminar el ruido con un filtro. Se propuso por tanto un filtro digital pasa banda de respuesta de impulso finita (FIR), entre 100 y 250Hz, seguido de un submuestreador que pase de 48KHz a 1KHz en las muestras de las grabaciones a analizar.

Cálculo de autocorrelación y medición de periodicidad

El primer análisis que se llevó a cabo para la detección de sonidos de motosierra fue el de autocorrelación.

En las Fig. 4.92 se presenta el diagrama de bloques o programación gráfica realizado en Labview para realizar el cálculo de la autocorrelación y la medición de la periodicidad para una grabación seccionada en ventanas de datos.

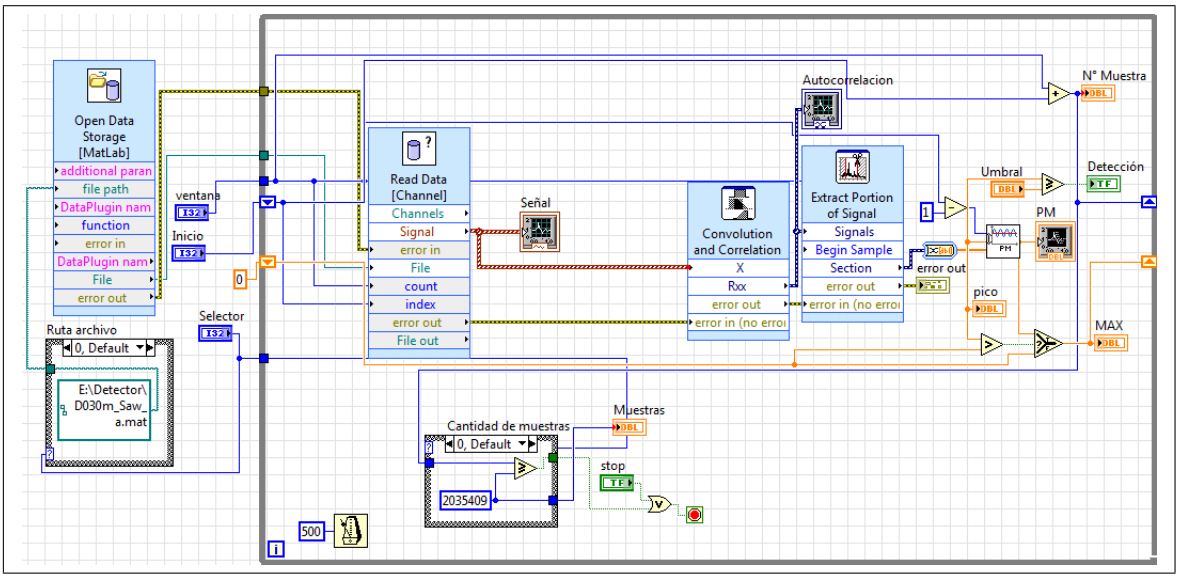


Figura 4.92: Diagrama de bloques para el cálculo de autocorrelación y medición de periodicidad con desplazamiento en ventanas de datos. Tomada de [93].

Se procedió a realizar el análisis de periodicidad para las grabaciones de sonidos de motosierra, lluvia y disparos. La autocorrelación fue obtenida para una ventana característica donde el sonido es el esperado, con una ventana de 5000 datos de longitud que representa una fracción de segundo en la grabación real. Las gráficas se presentan en las figuras 4.93, 4.94 y 4.95.

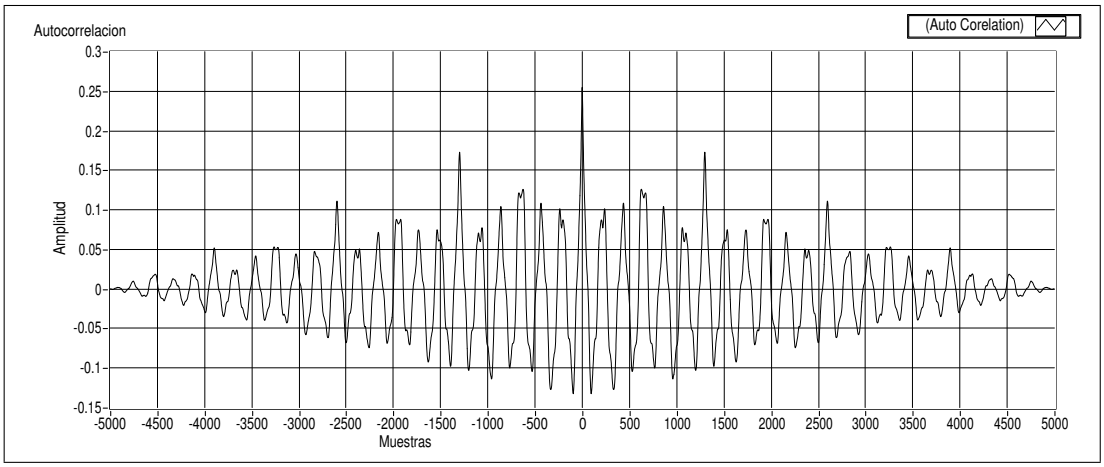


Figura 4.93: Análisis de autocorrelación para sonido de motosierra. Tomada de [93].

Realizando un análisis a las figuras presentadas anteriormente, se observa en la Fig. 4.93 una forma regular periódica y que va tendiendo a cero conforme se realizaron los cálculos con índices de retraso más grandes. Es posible ver en la gráfica la periodicidad de la señal analizada, pues el valor de autocorrelación es alta a valores distintos del valor de retardo cero (que sería la señal correlacionada con una copia suya en fase). Se nota el caso en los valores 200, 400 y 700 aproximadamente, por ejemplo. Es posible por tanto detectar valores de periodicidad en señales físicas originales que contienen alteraciones

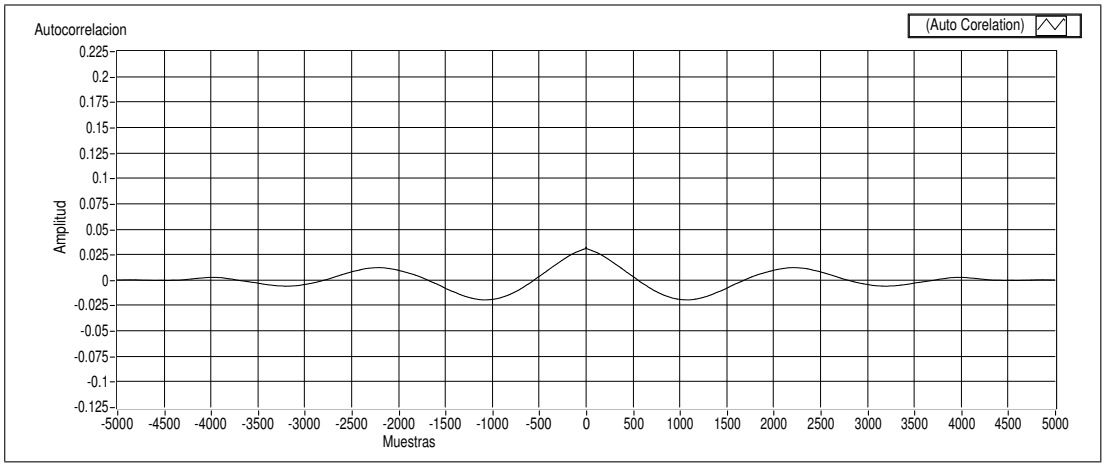


Figura 4.94: Análisis de autocorrelación para sonido de lluvia. Tomada de [93].

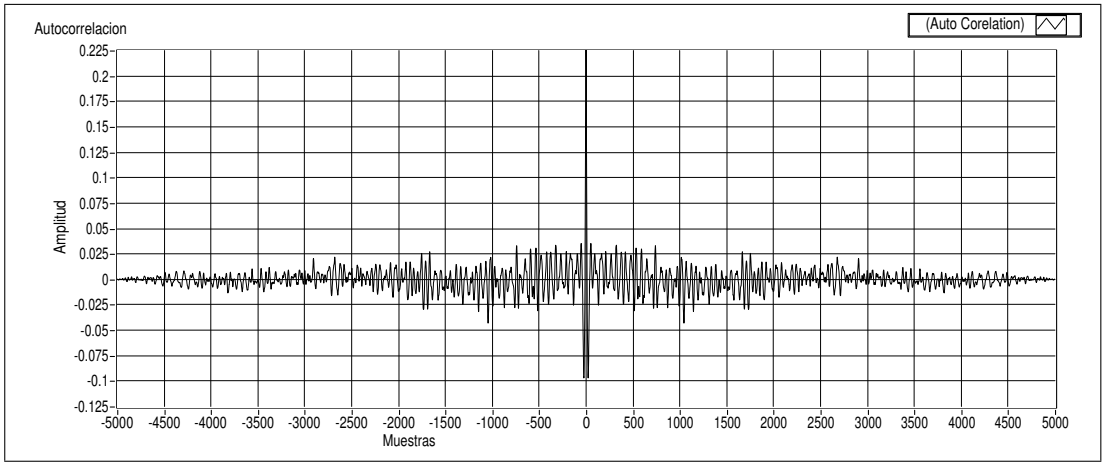


Figura 4.95: Análisis de autocorrelación para disparo de arma R38. Tomada de [93].

con interferencias aleatorias.

En contraste, en la Fig. 4.94, se muestra la autocorrelación de una grabación de sonido de lluvia para una ventana igualmente de 5000 muestras. La gráfica de autocorrelación es mucho menos definida y rápidamente tiende a cero, lo que sugiere que no existe periodicidad en la señal analizada.

En la Fig. 4.95 tenemos la autocorrelación para un disparo de arma calibre 38. La gráfica permite deducir la presencia de cierta periodicidad, si bien bastante baja, dada la pequeña magnitud de los picos altos de correlación lejos del origen, en contraste con la Fig. 4.93.

Es posible por tanto realizar cálculos que diferencien el comportamiento de estas gráficas por medio de análisis de periodicidad que consistiría en formular una sumatoria de valores con cierto índice de retardo, aprovechando la diferencia entre valores máximos que se encuentra de la autocorrelación de sonido de motosierra contra los demás tipos de sonidos que se tiene a disposición. A este siguiente paso se le llamará medición de periodicidad y con este se podrá realizar análisis para varios valores de umbral para generar una detección.

En la ecuación (4.28) se describe la forma clásica para el cálculo de la medición de periodicidad que consiste en la sumatoria de cada unos de los valores producidos por el paso de autocorrelación.

$$\mathscr{P}_{alt} = \sum_{n=N_{min}}^{N_{max}} (\overline{R}_{xx}[n])^2 \tag{4.28}$$

Como se menciona en [31], para utilizar la ecuación (4.28) es necesaria una señal estacionaria, donde no ocurran fluctuaciones en baja frecuencia que introduzcan picos adicionales a la función de autocorrelación. Estos pico producen un *offset* en el cálculo de autocorrelación, generando un error que no permite la detección de periodicidad en la señal. Para esto fue necesario realizar una modificación a la ecuación de medición de periodicidad plateada anteriormente, realizando una derivación en el tiempo y calculando luego la suma de cuadrados. Entonces la ecuación (4.28) se convierte en la ecuación (4.29) después de que se tomara en cuenta lo dicho en [31].

$$\mathscr{P}_{alt} = \sum_{n=N_{min}}^{N_{max}-1} (\overline{R}_{xx}[n+1] - \overline{R}_{xx}[n])^2 \tag{4.29}$$

Se aplicó esta ecuación a los sonidos de motosierra y el disparo de arma calibre 38, y se traficaron los resultados. Estos se presentan en las Figs. 4.96 y 4.97. Para el sonido de lluvia no se presenta gráfico de medición de periodicidad ya que los valores obtenidos están muy por debajo de los valores utilizados para la detección en los demás sonidos.

En la Fig. 4.96 se observa que, conforme a la predicción visual hecha, el nivel de periodicidad es alto. De esta manera es posible determinar un nivel fijo de detección para

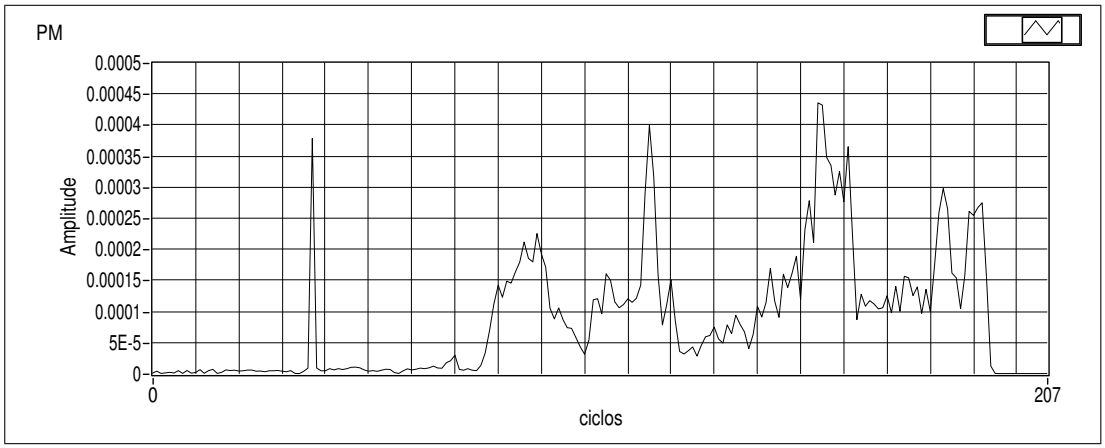


Figura 4.96: Análisis medición de periodicidad para sonido de motosierra. Tomada de [93].

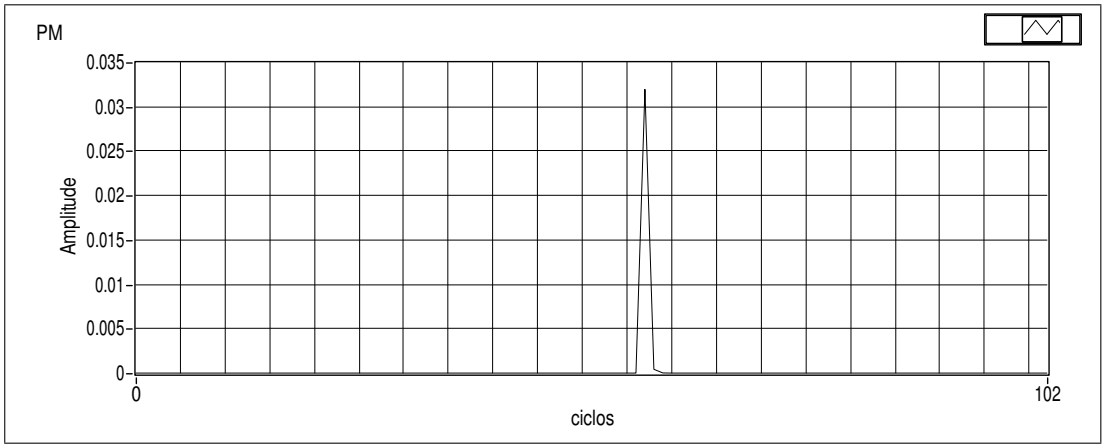


Figura 4.97: Análisis medición de periodicidad para sonido disparo de arma calibre 38. Tomada de [93].

generar un señal alarma de detección de señales periódicas. En la Fig. 4.97 se observa en contraste que la periodicidad es alta solo en el momento que se produce el disparo, para luego desvanecerse. Es posible por tanto establecer un criterio en función de la duración del valor de periodicidad (o establecer patrones de subdivisión de la señal mayores) para discriminar entre señales impulsivas y aquellas de mayor duración.

Con los vectores generados para el cálculo de la medición de periodicidad, se puede establecer un valor umbral para discernir cuando existe y cuando no una detección de periodicidad. Para una prueba se propuso un valor de $1E^{-5}$ después de haber analizado cuáles son los valores promedios que se obtuvieron con el proceso de anterior. Las Figs. 4.98 y 4.99 representan la matriz de detección según la localización de las muestras analizadas que depende estrictamente de los resultados obtenidos en las figuras 4.96 y 4.97. Cada impulso que se presentan en estas dos gráficas representa una detección en la ventana procesada de los sonidos de motosierra y del arma calibre 38.

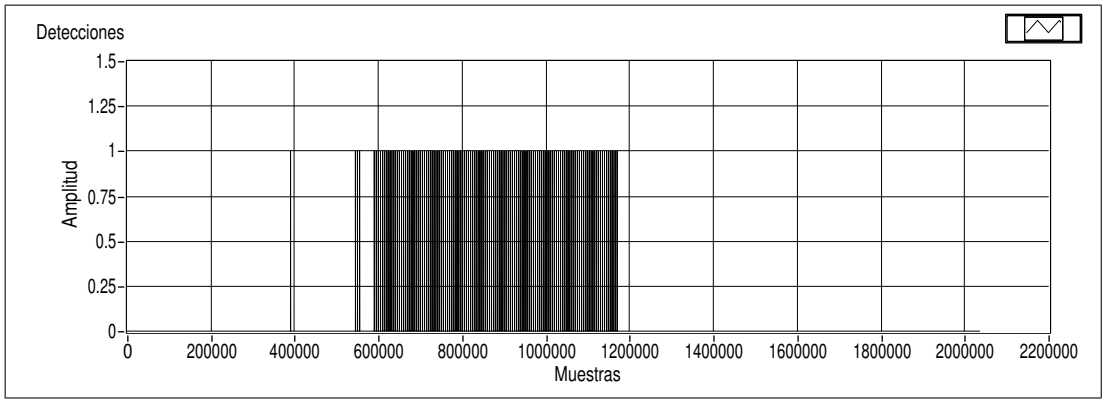


Figura 4.98: Matriz de detecciones para sonido de motosierra. Tomada de [93].

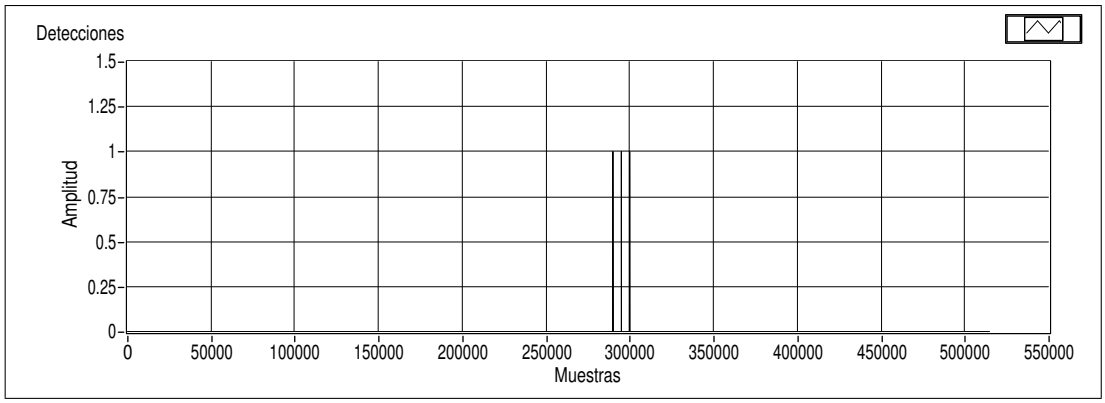


Figura 4.99: Matriz de detecciones para sonido disparo de arma calibre 38. Tomada de [93].

Para esta prueba en particular se presenta la tabla 4.43 con la cantidad de detecciones obtenidas para par el umbral establecido de $1E^{-5}$.

Con estos resultados provisionales se puede concluir que los algoritmos utilizados son aptos para nuestro interés de detectar sonidos generados por motosierras y poder discernirlos

Tabla 4.43: Detecciones obtenidas para umbral seleccionado de $1E^{-5}$. Tomada de [93].

Sonido	Detecciones
Motosierra	121
Disparo R38	3
Lluvia	0

del ruidos ambiente que se presenta en el bosque y de otros sonidos generados por el hombre.

4.3.2 Evaluación del algoritmo detector de sonidos incluyendo filtro y submuestreador

Se implementó un filtro con una banda pasante de entre 100-250Hz y con con una atenuación del 99% fuera de la banda pasante. En la Fig. 4.100 se presenta el diagrama de bloques algorítmico para realizar los análisis de detecciones para las grabaciones a distintas distancias y ángulos con respecto al micrófono del grabador digital utilizado.

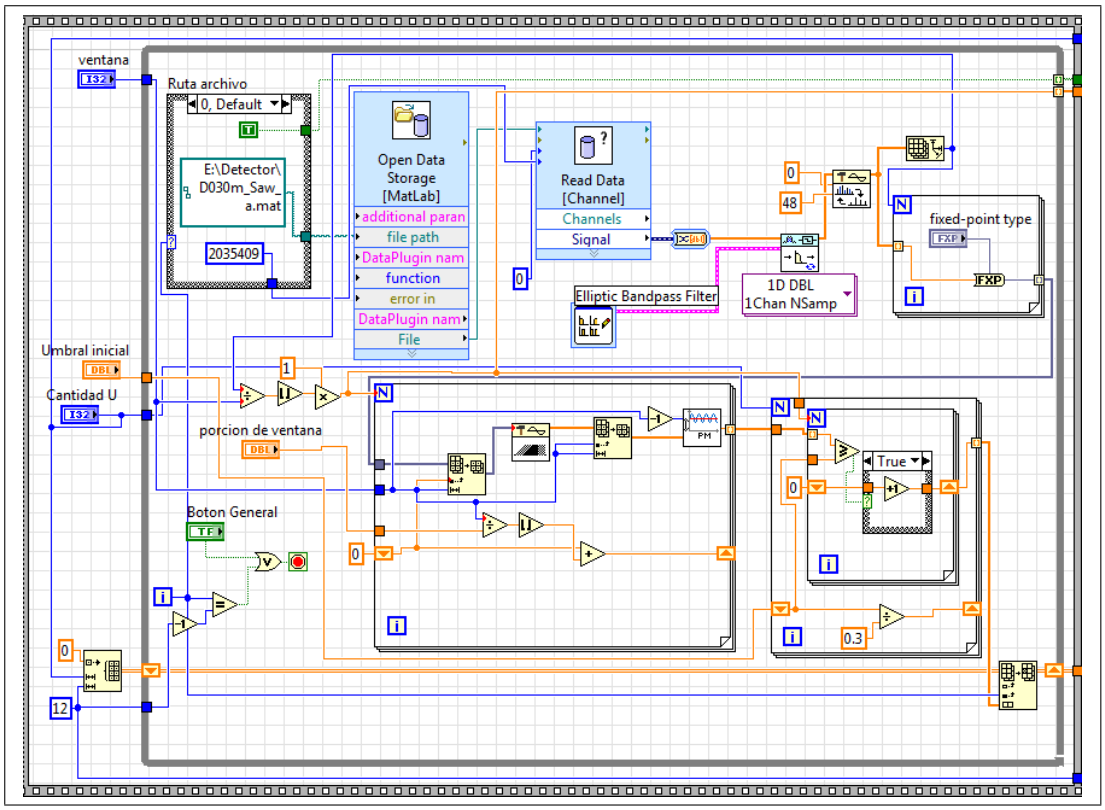


Figura 4.100: Diagrama de bloques para la detección de sonidos con presencia de periodicidad con el filtro y submuestreador propuesto. Tomada de [93].

Se utilizaron los bloques de Labview llamados "DFD Filtering" para el proceso de filtrado y "Resample" para el submuestreador. Se integran además los bloques diseñados para los cálculos de autocorrelación y medición de periodicidad, así como registro de las cantidades

de detecciones producidas por el algoritmo, para todas la grabaciones disponibles. Ya que a la hora de obtener la medición de periodicidad se presenta un nivel que depende del cálculo de periodicidad encontrado con ayuda del análisis de autocorrelación, es posible definir umbrales de detección para esos niveles en la medición. Los resultados se presentan en las Tablas 4.44 y 4.45.

Puede verse que existe una alta cantidad de detecciones de periodicidad en los sonidos de motosierra, con una máximo de 96 en el sonido "D090m_Saw_000b" que fue generado a 90 metros y con un ángulo de 0^o con respecto al grabador digital. También se presenta una cantidad de detecciones considerables en los sonidos de aeroplanos y del arma S12 para todas las distancias estudiadas en este caso particular. El sonido "A_Plane_002" presenta una cantidad máxima de 20 detecciones cuando se utiliza un umbral de medición de $20E^{-9}$ y $60E^{-9}$. El sonido de aeroplano es generado por un motor similar al de una motosierra, por lo que resulta lógico que sea detectado. Se puede notar en la Tabla 4.45 que las detecciones de motosierras a 600 metros se degradan en gran manera ya que a esa distancia el ruido ambiente produce interferencia con el sonido periódico que se trata de discernir. Esto producirá en los análisis una disminución de la efectividad del algoritmo para realizar detecciones certeras.

Los resultados se obtuvieron utilizando una ventana entera de 250 muestras, por ser la que mejor cantidad de detecciones produjo con la menor cantidad de muestras en las grabaciones analizadas. También se realizaron análisis con desplazamientos de media, tercio y cuarto de ventana, produciéndose cantidad de detecciones similares y de parecida calidad. La utilización desplazamientos no enteros de la ventana en las muestras de las grabaciones, genera la necesidad de un hardware más veloz y por ende de mayor consumo.

Eficiencia del algoritmo detector de periodicidad mediante curvas ROC. Tomada de [93].

La evaluación de la efectividad de los algoritmos se realizó mediante el análisis de curvas ROC (receiver operating characteristic) muy usadas en teoría de señales (ver [13] para más detalles). El análisis incluyó también las consideraciones sobre la degradación de las detecciones al disminuir la resolución en punto fijo de los datos. Para generar los análisis que se detallan en esta sección, se hizo uso de la herramienta LabVIEW ya que partes de los módulos utilizados en las secciones anteriores 4.3.1 y 4.3.2 son útiles nuevamente para la generación de los análisis ROC. La Fig. 4.101 presenta el proceso utilizado para realizar el análisis integral de la grabaciones y luego generar la curva ROC con los resultados de las detecciones encontradas después de pasa por el filtrado, submuestreo, división por ventanas, autocorrelación y cálculo de periodicidad. El bloque algorítmico de más a la derecha representa la sección cálculo de los datos para luego generar la gráfica ROC. En esta sección también se agrega el bloque funcional ya proporcionado por el software para el cálculo del área bajo la curva, para parametrizar la eficiencia que presenta el algoritmo de detección utilizado (a mayor área, mejor eficiencia).

Tabla 4.44: Detecciones obtenidas para las grabaciones a una distancia de 30 y 90 metros en función del umbral de detección. Tomada de [93].

	Umbral de detección $[E^{-9}]$										
Sonido	**20**	**60**	**120**	**180**	**240**	**300**	**360**	**420**	**480**	**540**	**600**
A_Birds_001	0	0	0	0	0	0	0	0	0	0	0
A_Birds_002	0	0	0	0	0	0	0	0	0	0	0
A_Birds_003	0	0	0	0	0	0	0	0	0	0	0
A_Plane_001	5	5	4	4	4	3	3	2	1	0	0
A_Plane_002	20	20	19	15	12	10	5	1	0	0	0
A_Rain_001	0	0	0	0	0	0	0	0	0	0	0
A_Rain_002	42	13	5	1	1	0	0	0	0	0	0
A_Wind_001	0	0	0	0	0	0	0	0	0	0	0
D030m_C22_000	0	0	0	0	0	0	0	0	0	0	0
D030m_C22_090	0	0	0	0	0	0	0	0	0	0	0
D030m_C22_180	0	0	0	0	0	0	0	0	0	0	0
D030m_Pi9_000	2	2	2	1	1	0	0	0	0	0	0
D030m_Pi9_090	3	2	1	1	0	0	0	0	0	0	0
D030m_Pi9_180	1	1	0	0	0	0	0	0	0	0	0
D030m_R32_000	2	1	1	1	0	0	0	0	0	0	0
D030m_R32_090	1	1	0	0	0	0	0	0	0	0	0
D030m_R32_180	0	0	0	0	0	0	0	0	0	0	0
D030m_R38_000	2	2	2	1	0	0	0	0	0	0	0
D030m_R38_090	1	1	1	0	0	0	0	0	0	0	0
D030m_R38_180	1	0	0	0	0	0	0	0	0	0	0
D030m_S12_000	6	6	5	3	3	2	2	2	2	1	1
D030m_S12_090	9	9	7	6	6	6	5	5	4	3	3
D030m_S12_180	7	7	6	6	5	4	3	2	1	1	1
D030m_Saw_000	91	90	88	86	86	84	77	66	51	48	38
D030m_Saw_090	69	67	65	62	60	55	52	45	20	8	3
D030m_Saw_180	60	57	53	47	33	12	0	0	0	0	0
D090m_C22_000	1	1	1	1	0	0	0	0	0	0	0
D090m_C22_090	0	0	0	0	0	0	0	0	0	0	0
D090m_C22_180	0	0	0	0	0	0	0	0	0	0	0
D090m_Pi9_000	3	3	2	1	1	1	1	1	0	0	0
D090m_Pi9_090	2	2	1	1	0	0	0	0	0	0	0
D090m_Pi9_180	0	0	0	0	0	0	0	0	0	0	0
D090m_R32_000	1	1	0	0	0	0	0	0	0	0	0
D090m_R32_090	1	1	0	0	0	0	0	0	0	0	0
D090m_R32_180	1	0	0	0	0	0	0	0	0	0	0
D090m_R38_000	2	1	0	0	0	0	0	0	0	0	0
D090m_R38_090	1	0	0	0	0	0	0	0	0	0	0
D090m_R38_180	1	1	0	0	0	0	0	0	0	0	0
D090m_Saw_000a	72	72	72	71	70	70	69	67	59	53	48
D090m_Saw_000b	96	96	96	96	96	96	94	82	59	51	3
D090m_Saw_090	60	57	46	37	30	19	11	2	0	0	0
D090m_Saw_180	43	39	36	31	27	26	14	3	1	0	0

Tabla 4.45: Detecciones obtenidas para las grabaciones a una distancia de 250 y 600 metros en función del umbral de detección. Tomada de [93].

	Umbral de detección $[E^{-9}]$										
Sonido	**20**	**60**	**120**	**180**	**240**	**300**	**360**	**420**	**480**	**540**	**600**
D250m_C22_000	0	0	0	0	0	0	0	0	0	0	0
D250m_C22_090	0	0	0	0	0	0	0	0	0	0	0
D250m_C22_180	0	0	0	0	0	0	0	0	0	0	0
D250m_Pi9_000	2	0	0	0	0	0	0	0	0	0	0
D250m_Pi9_090	1	0	0	0	0	0	0	0	0	0	0
D250m_Pi9_180	0	0	0	0	0	0	0	0	0	0	0
D250m_R32_000	1	0	0	0	0	0	0	0	0	0	0
D250m_R32_090	0	0	0	0	0	0	0	0	0	0	0
D250m_R32_180	0	0	0	0	0	0	0	0	0	0	0
D250m_R38_000	0	0	0	0	0	0	0	0	0	0	0
D250m_R38_090	0	0	0	0	0	0	0	0	0	0	0
D250m_R38_180	0	0	0	0	0	0	0	0	0	0	0
D250m_S12_000	5	4	4	4	3	3	2	1	1	0	0
D250m_S12_090	5	5	4	4	4	2	2	1	1	0	0
D250m_S12_180	5	4	4	4	2	1	0	0	0	0	
D250m_Saw_000	17	7	0	0	0	0	0	0	0	0	0
D250m_Saw_090	14	2	0	0	0	0	0	0	0	0	0
D250m_Saw_180	0	0	0	0	0	0	0	0	0	0	0
D600m_C22_000	0	0	0	0	0	0	0	0	0	0	0
D600m_C22_090	0	0	0	0	0	0	0	0	0	0	0
D600m_C22_180	0	0	0	0	0	0	0	0	0	0	0
D600m_Pi9_000	0	0	0	0	0	0	0	0	0	0	0
D600m_Pi9_090	1	1	1	1	0	0	0	0	0	0	0
D600m_Pi9_180	1	1	0	0	0	0	0	0	0	0	0
D600m_R32_000	1	1	1	0	0	0	0	0	0	0	0
D600m_R32_090	0	0	0	0	0	0	0	0	0	0	0
D600m_R32_180	0	0	0	0	0	0	0	0	0	0	0
D600m_R38_000	0	0	0	0	0	0	0	0	0	0	0
D600m_R38_090	1	0	0	0	0	0	0	0	0	0	0
D600m_R38_180	0	0	0	0	0	0	0	0	0	0	0
D600m_S12_000	5	5	4	4	4	4	3	2	2	1	0
D600m_S12_090	3	3	2	2	2	2	0	0	0	0	0
D600m_Saw_000	5	5	3	3	1	1	0	0	0	0	0
D600m_Saw_090	7	0	0	0	0	0	0	0	0	0	0
D600m_Saw_180	8	6	1	0	0	0	0	0	0	0	0

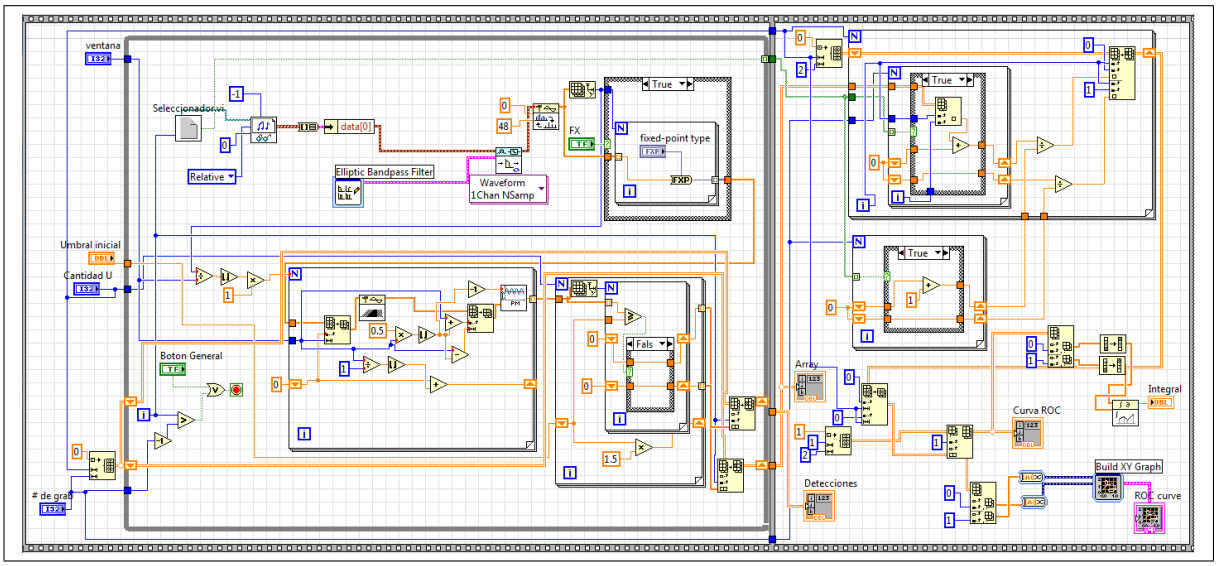


Figura 4.101: Diagrama de bloques para la detección de sonidos con presencia de periodicidad y generación de análisis ROC. Tomada de [93].

Para crear los ROC se definieron las tasas de detección de la siguiente manera:

$$Razón\ verdaderos\ positivos = \frac{Verdaderos\ positivos}{Verdaderos\ positivos\ +\ Falsos\ negativos} \tag{4.30}$$

donde

$$Razón\ verdaderos\ positivos = Sensibilidad \tag{4.31}$$

$$Razón\ falsos\ positivos = \frac{Falsos\ positivos}{Falsos\ positivos\ +\ Verdaderos\ negativos} \tag{4.32}$$

donde

$$Razón\ falsos\ positivos = 1\text{-}Especificidad \tag{4.33}$$

Con la matriz de detecciones, se calculó la tasa de verdaderos positivos y tasa de falsos positivos mediante las ecuaciones anteriores, construyendo otra matriz que contiene estos resultados para cada uno de los umbrales utilizados .

Eficiencia obtenida para el algoritmo con diferentes ventanas

Los resultados que se presentan a continuación representan los análisis ROC producidos para las grabaciones de 30, 90, 250 y 600 metros.

Las curvas ROC generadas para ventanas de análisis de 100, 250 y 400 muestras se presentan en las Figs. 4.102, 4.103 y 4.104 respectivamente.

Para las curvas obtenidas con la ventana que contiene 100 muestras se presentaron resultados alrededor del 62% para sonidos obtenidos a 30 y 90 metros, tal como se puede corroborar en las gráficas 4.102(a) y 4.102(b). Para distancias de 250 y 600 metros se obtuvieron resultados más bajos de 55.8% y 51% respectivamente, que se presentan en las gráficas 4.102(c) y 4.102(d).

Con una ventana de *250 muestras* para el cálculo de medición de periodicidad se obtuvieron mejores valores de área bajo la curva con respecto a la ventana de 100 muestras. Los resultados se puede ver en las gráficas de la figura 4.103. Para una distancia de 30 metros se obtuvo una eficiencia del 86.8% y para 90 metros esta fue de 90.02%. Para las distancias de 250 y 600 metros se tienen resultados de 80% y 74.5% respectivamente. Puede concluirse que manejar una ventana de 250 muestras es razonable para realizar la detección con una efectividad aceptable.

Los resultados para una ventana de 400 muestras se presentan en la Fig. 4.104. Los resultados no superaron los de la ventana de 250 muestras.

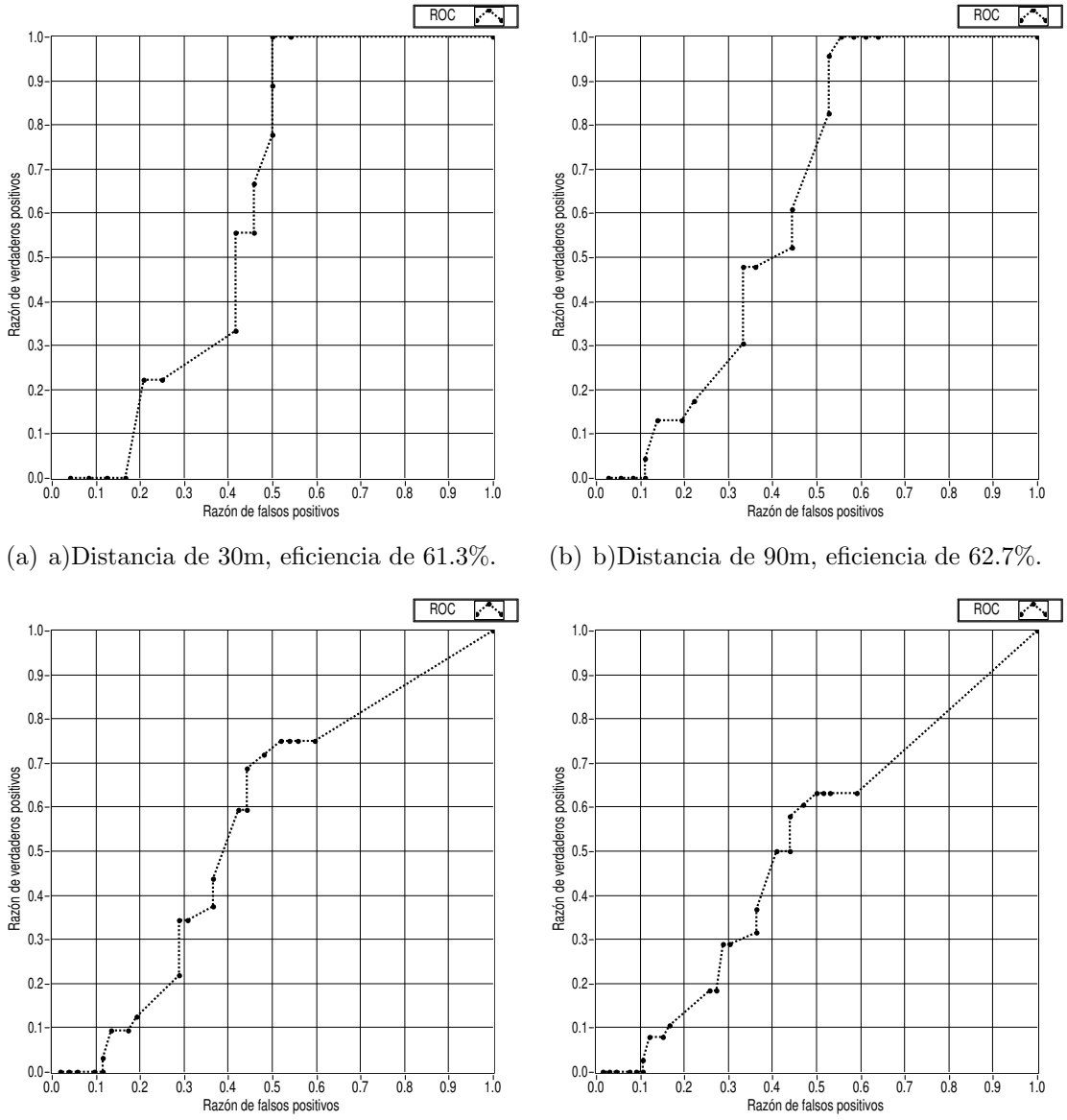
Con los resultados analizados anteriormente se llegó a la conclusión de que conviene utilizar para el procedimiento de detección de sonidos periódicos de motosierra, un hardware de cálculo para los algoritmos de autocorrelación y medición de periodicidad con una longitud de ventana de 250 muestras, que procese principalmente sonidos que provengan de distancias no mayores de 100 metros con respecto a el micrófono a utilizar y permitiéndose variar el ángulo de llegada de las ondas sonoras con respecto a la normal del mismo micrófono.

Eficiencia obtenida en punto fijo para la ventana seleccionada

Una vez presentados los resultados obtenidos en la Sección 4.3.2, donde se escogió una ventana de 250 muestras por sus buenos resultados y conveniente tamaño, se procedió a realizar algunas pruebas de cómo cambian los resultados para los análisis ROC cuando se recorta la cantidad de bits de precisión con el sistema digital de cálculo en punto fijo. Se realizaron varia pruebas con una representación numérica de 16, 10, 8 y 6 bits.

El formato usado para el análisis fue de signo magnitud, con un bit de signo, un bit de parte entera, y el resto de bits de parte fraccionaria. En las Fig. 4.105, 4.106, 4.107 y 4.108 se presentan los resultados de las pruebas para diferentes valores en punto fijo. Por ejemplo, en la Fig. 4.105 de 16 bits de palabra, la sección fraccionaria está representada por 14 bits de precisión. Posteriormente, se notó que utilizar una representación signo magnitud, con representación solo de parte fraccionaria (dando por hecho que la parte entera de la representación será siempre cero) ofrecía valores satisfactorios.

Como es de esperar, al disminuir el tamaño de la representación, se produjo una decaimiento de la efectividad del algoritmo utilizado. Las gráficas para las demás pruebas se presentan en las figuras 4.106, 4.107 y 4.108 para longitudes de palabra de 10, 8 y 6 bits respectivamente.



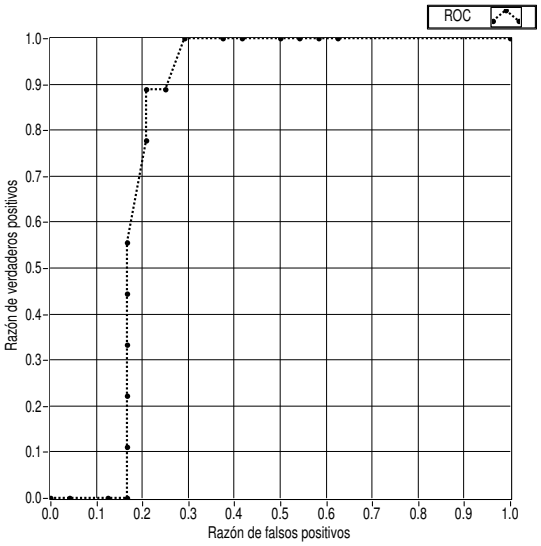
(a) a)Distancia de 30m, eficiencia de 61.3%.

(b) b)Distancia de 90m, eficiencia de 62.7%.

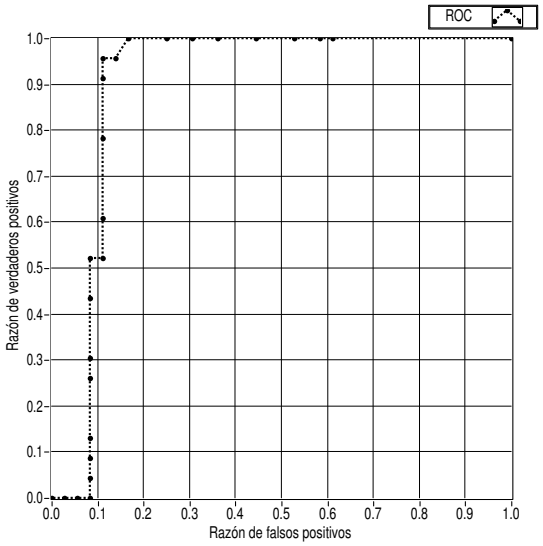
(c) c)Distancia de 250m, eficiencia de 55.8%.

(d) d)Distancia de 600m, eficiencia de 51.1%.

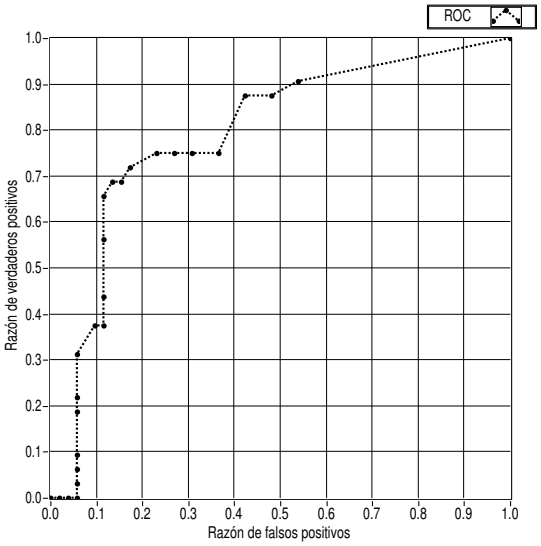
Figura 4.102: Análisis ROC para diferentes distancias con una una ventana de análisis de 100 muestras (figuras tomadas de [93].



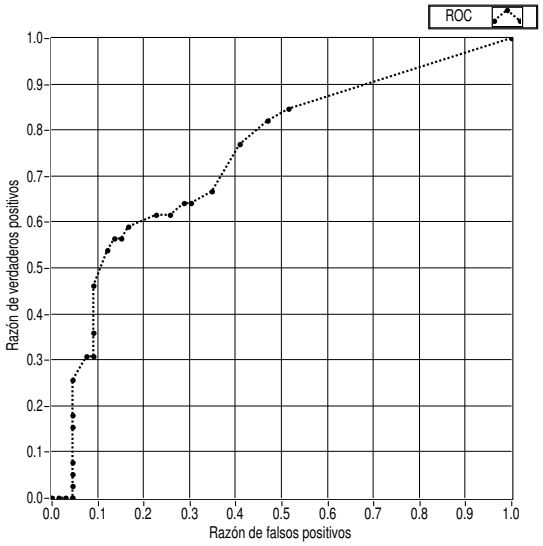
(a) a)Distancia de 30m, eficiencia de 86.8%.



(b) b)Distancia de 90m, eficiencia de 90.02%.

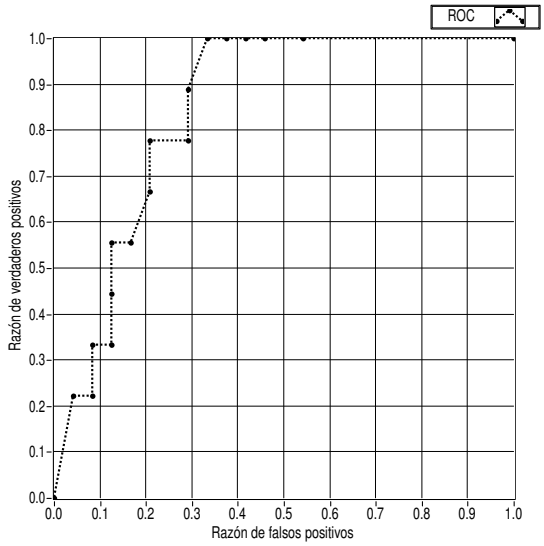


(c) c)Distancia de 250m, eficiencia de 80%.

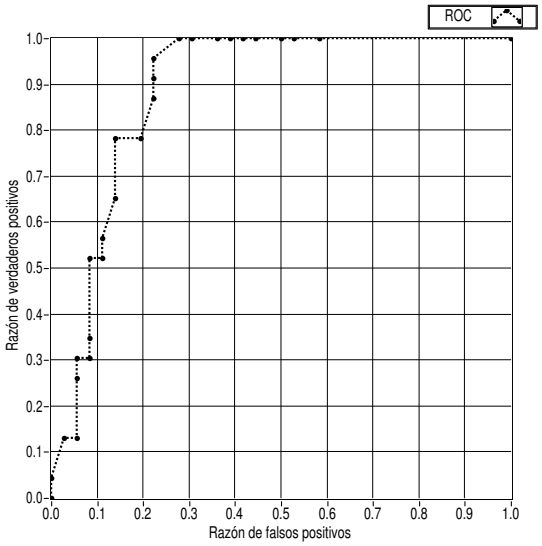


(d) d)Distancia de 600m, eficiencia de 74.5%.

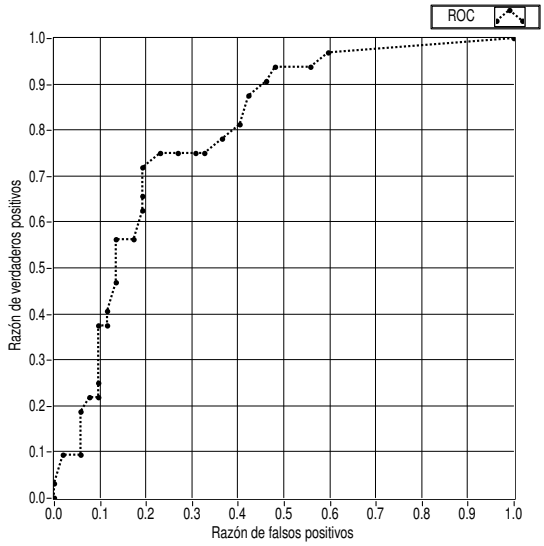
Figura 4.103: Análisis ROC para diferentes distancias con una ventana de análisis de 250 muestras (figuras tomadas de [93]).



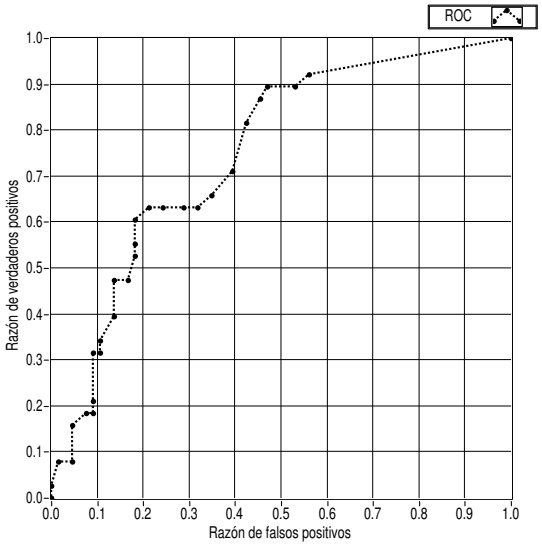
(a) a)Distancia de 30m, eficiencia de 84.7%.



(b) b)Distancia de 90m, eficiencia de 88.9%.

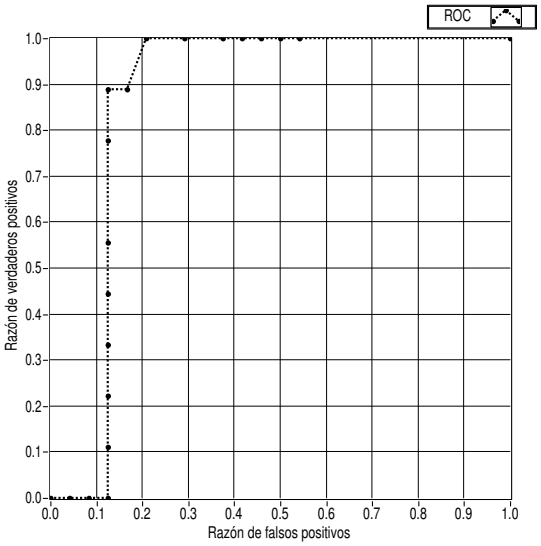


(c) c)Distancia de 250m, eficiencia de 79.6%.

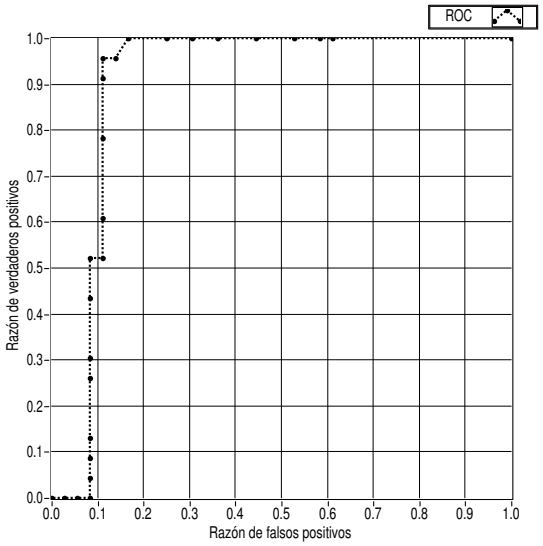


(d) d)Distancia de 600m, eficiencia de 75.1%.

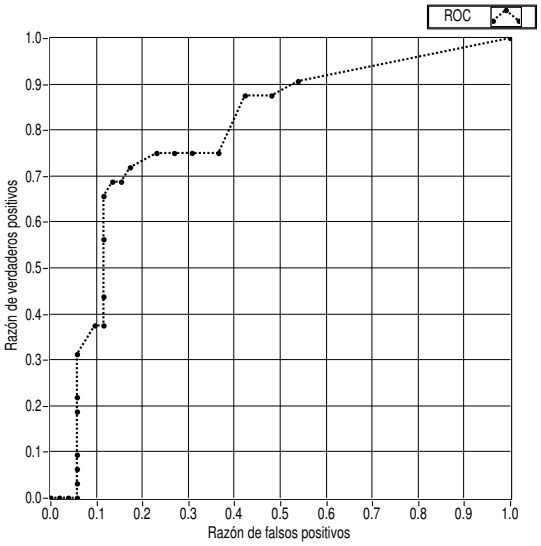
Figura 4.104: Análisis ROC para diferentes distancias con una ventana de análisis de 250 muestras (figuras tomadas de [93]).



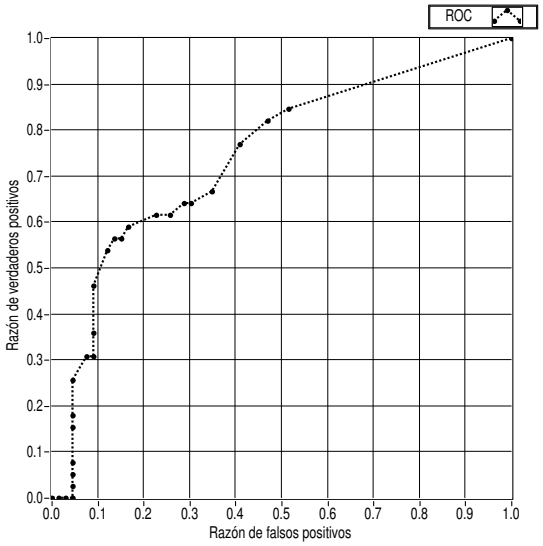
(a) a)Distancia de 30m, eficiencia de 86.8%.



(b) b)Distancia de 90m, eficiencia de 90.02%.

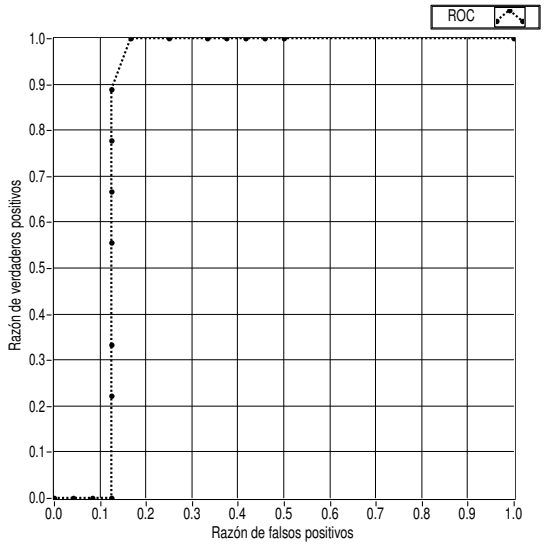


(c) c)Distancia de 250m, eficiencia de 80%.

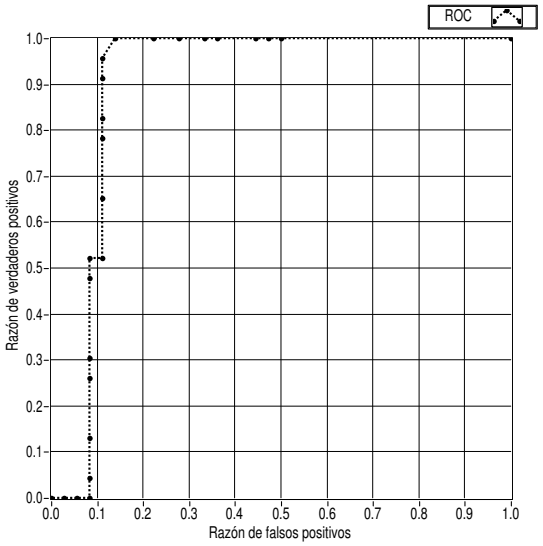


(d) d)Distancia de 600m, eficiencia de 74.5%.

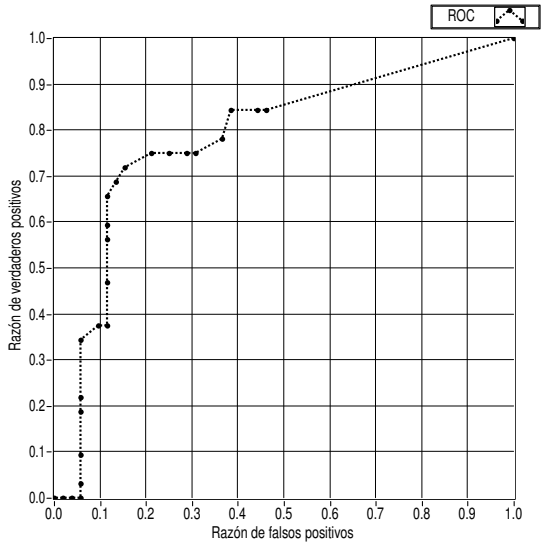
Figura 4.105: Análisis ROC para diferentes distancias con 16 bits de longitud de palabra (figuras tomadas de [93]).



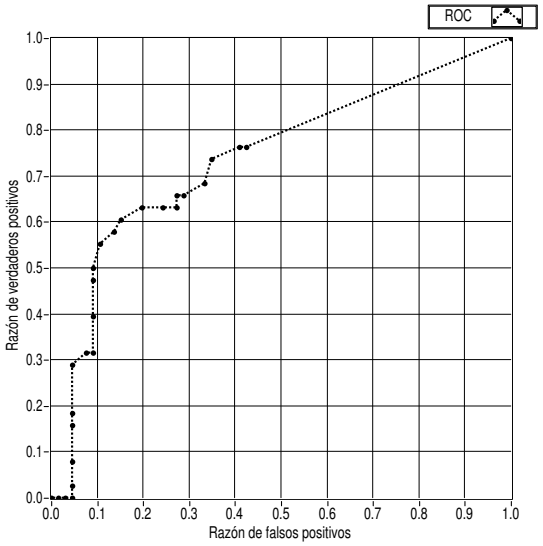
(a) a)Distancia de 30m, eficiencia de 87%.



(b) b)Distancia de 90m, eficiencia de 90.02%.

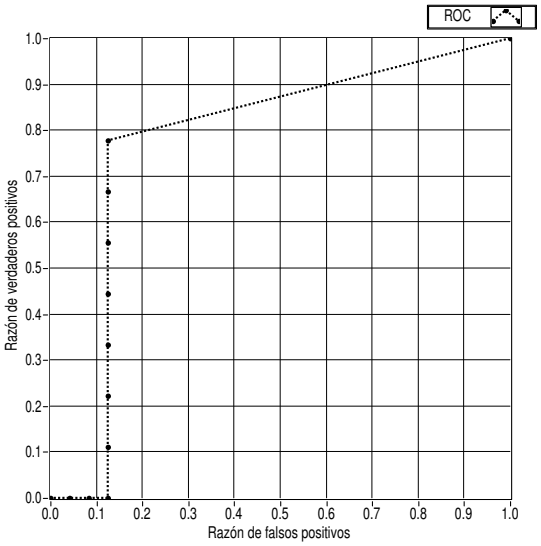


(c) c)Distancia de 90m, eficiencia de 90.2%.

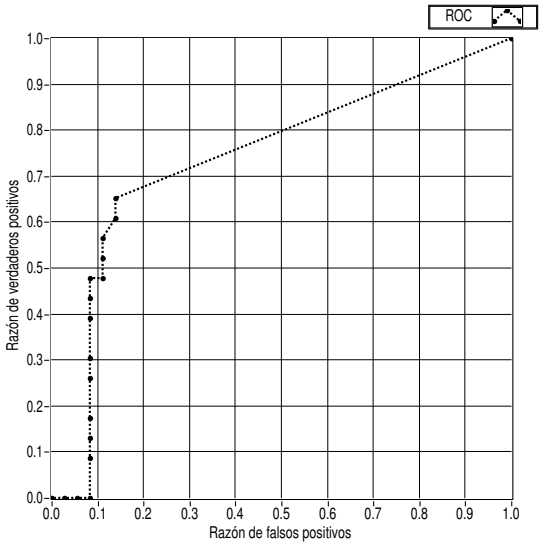


(d) d)Distancia de 600m, eficiencia de 73.9%.

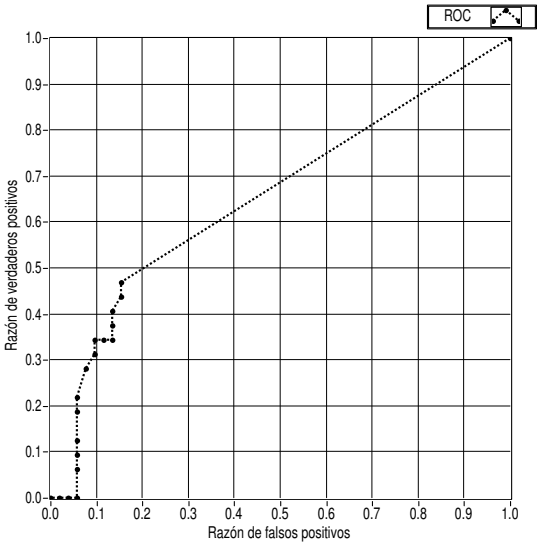
Figura 4.106: Análisis ROC para diferentes distancias con 10 bits de longitud de palabra (figuras tomadas de [93]).



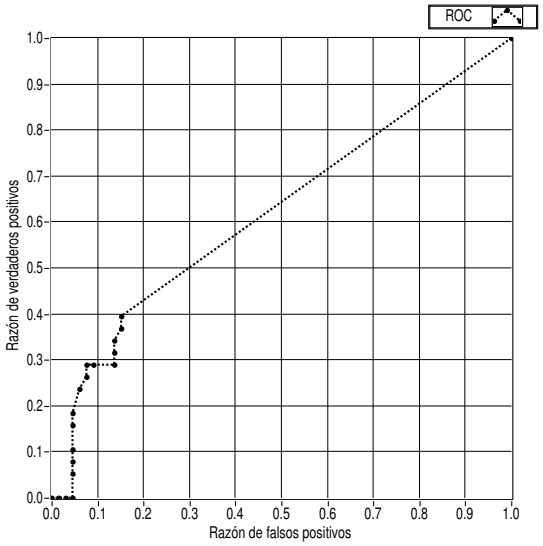
(a) a)Distancia de 30m, eficiencia de 77.8%.



(b) b)Distancia de 90m, eficiencia de 74.1%.

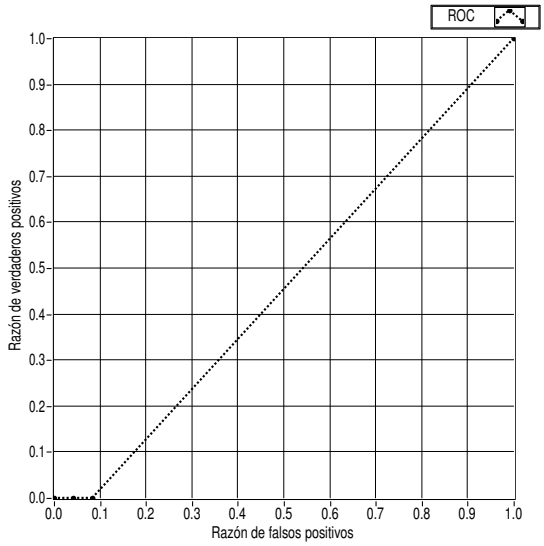


(c) c)Distancia de 250m, eficiencia de 65.3%.

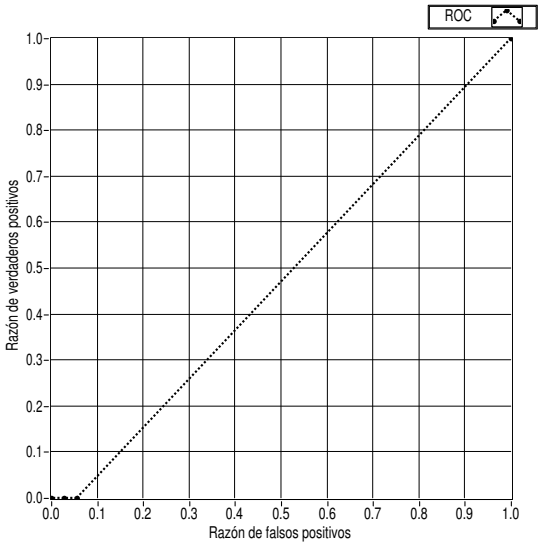


(d) d)Distancia de 600m, eficiencia de 62.2%.

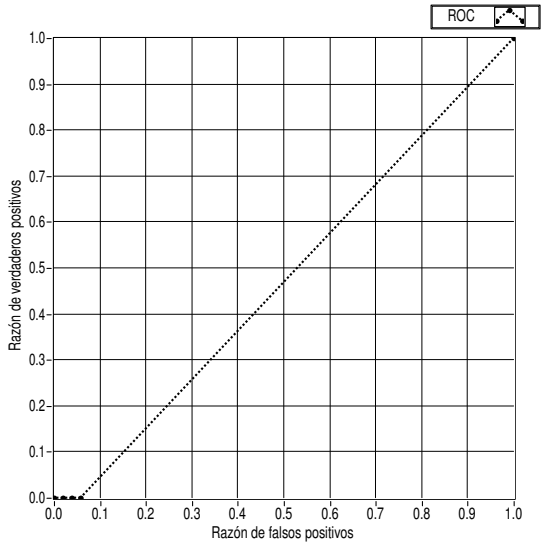
Figura 4.107: Análisis ROC para diferentes distancias con 8 bits de longitud de palabra (figuras tomadas de [93]).



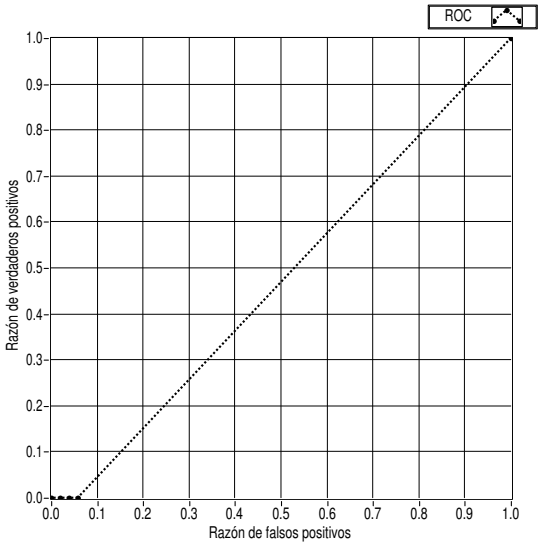
(a) a)Distancia de 30m, eficiencia de 45.8%.



(b) b)Distancia de 90m, eficiencia de 47.2%.



(c) c)Distancia de 250m, eficiencia de 47.1%.



(d) d)Distancia de 600m, eficiencia de 47.7%.

Figura 4.108: Análisis ROC para diferentes distancias con 8 bits de longitud de palabra (figuras tomadas de [93]).

Como conclusión, para obtener buenos resultados en la detección de los sonidos periódicos será necesario la utilización de por lo menos un sistema digital en punto fijo de 9 bits de longitud de palabra (1 bit de signo, 8 de magnitud fraccionaria).

4.3.3 Conclusiones parciales sobre diseño de un sistema de detección preliminar de motosierras

Se logró obtener el diseño preliminar del algoritmo para un detector de sonidos de motosierra, cuyos sonidos tienen la particularidad de poseer un comportamiento periódico característico en la señal que proviene del aire capturada por el micrófono.

Se comprobó que el usar un filtro pasabanda de 100Hz- los 250Hz, mejora la eficiencia del algoritmo de detección. Las pruebas se realizaron con un filtro digital, pero este filtro bien puede ser el filtro analógico de antialias antes del convertidor ADC. Esto disminuiría el tamaño del circuito digital.

Los mejores resultados se obtuvieron al utilizar una ventana de muestras de sonidos entrantes de 250 valores, con 8 bits de resolución para la precisión fraccionaria más 1 bit de signo. Mediante esta configuración se obtuvieron resultados de eficiencia del algoritmo por medio de las curvas ROC de hasta un 90.02% cuando se utilizan sonidos de alrededor de 90 metros de distancia, y para un 74.5% para sonidos de hasta 600 metros. Es eso sí necesario ampliar el lapso de medición de periodicidad (que esta se mantenga encima de un umbral por un tiempo determinado) para evitar que sonidos cortos periódicos generen un falso positivo.

También se ha mostrado que el re-muestreo a 1kHz de la señal entrante de 48kHz no produce una pérdida significativa en el análisis en la medición de periodicidad. Con esto se puede entonces disminuir las necesidades de conversión de la señal a detectar, y utilizar un convertidor de baja frecuencia de muestreo, y de una resolución de solo 9 bits.

No existieron ya recursos de tiempo ni de asistentes para llevar este algoritmo a una implementación en hardware, por lo que el objetivo 3 no pudo cumplirse. Se sugiere realizar pruebas mediante una FPGA a través de LabVIEW para llevar más rápido al hardware la solución propuesta, que es prometedora. Para ello también puede aprovecharse el trabajo que se hace en este momento en el DCILab en el que se está implementando ya una unidad de cálculo de autocorrelación con FFT, para un proyecto internacional.

4.4 Clasificador digital para la clasificación de patrones acústicos de disparos y motosierras en un sistema empotrado.

El Sistema de Reconocimiento de Patrones Acústicos (SiRPA) es la etapa final del nodo o sistema integrado, que debe implementar la clasificación final de los sonidos detectados. Una solución a nivel de software C/C++ es provista en el trabajo del proyecto D2ARS realizado por el Dr Alvarado Moya y otros investigadores (ver [5]), donde ya se propone

e implementa una estructura a nivel de bloques funcionales. Se plantearon dos opciones de implementación de este sistema:

a Una plataforma integrada embebida que replique los resultados obtenidos en [5]. Esta ha sido pensada como un prototipo para evaluar cambios de manera flexible, realizar pruebas de campo y para contar con un punto de comparación con la implementación de hardware RTL, en cuanto a funcionamiento y rendimiento. La cadena de procesamiento realizada por el sistema embebido se muestra en la Fig.4.109.

b Una implementación en código RTL para portar primero a una FPGA, y luego trasladar al objetivo final: un ASIC o circuito integrado digital. Esta etapa se basa en la misma estructura de bloques de la Fig.4.109, pero descrita de una manera óptima para construir un circuito específico.

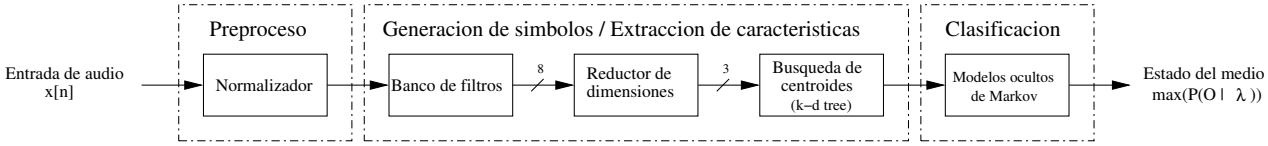


Figura 4.109: Cadena de procesamiento del SiRPA para ejecución, tal como fueron definidos en [5, 55].

Ahora, dentro del SiRPA existen diferentes bloques que deben ser entrenados previo a su utilización para reconocimiento en línea. Esto se hace por medio de archivos de audio de ejemplo para las tres clases mencionadas, y que fueron capturados para su uso en trabajos previos. Específicamente, deben entrenarse los bloques del reductor de dimensiones, árbol k-d y los HMM dentro del módulo de clasificación, para lo que se toman muestras en los puntos mostrados en la figura 4.110. Los resultados del entrenamiento pueden ser posteriormente exportados para usarlos en la implementación de ejecución.

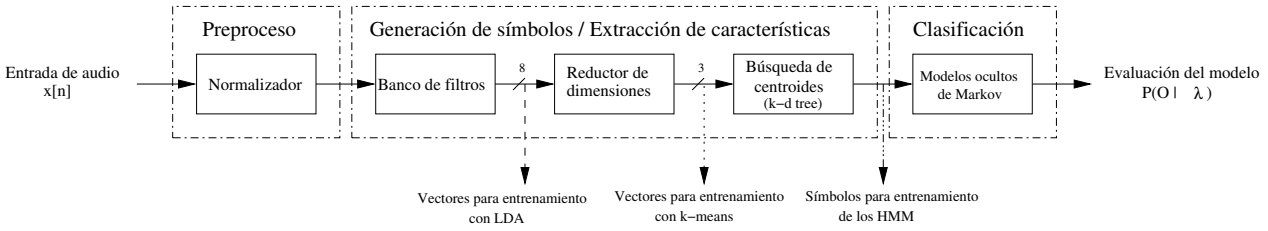


Figura 4.110: Cadena de procesamiento del SiRPA para entrenamiento, tal como fueron definidos en [5, 55].

El entrenamiento se encuentra desarrollado en C/C++ e integra una interfaz gráfica para facilitar la extracción de patrones acústicos desde archivos de audio, almacenamiento y administración de las muestras utilizadas por cada algoritmo de entrenamiento, y evaluación de los resultados obtenidos para cada uno de los bloques que integran el sistema.

La sección de ejecución es implementada en un sistema embebido Beagleboard-xM, y se encuentra desarrollada en el lenguaje de programación C.

A continuación se ofrece un resumen del análisis teórico necesario que establece los fundamentos del sistema de clasificación. Posteriormente, se detallan las etapas de entrenamiento del sistema, y de su implementación primero en un sistema embebido BeagleBoard. En una sección aparte se han colocado los detalles del desarrollo del RTL para su comprobación en FPGA para una posterior fabricación en ASIC.

4.4.1 Análisis teórico para la implementación del sistema de clasificación SiRPA

Para concluir este proyecto, es necesario portar código desarrollado en [5] a una plataforma integrada. Ello requiere una comprensión adecuada de los fundamentos teóricos de los algoritmos propuestos. Los resultados de esta sección son tomados del trabajo hecho en [5] y, especialmente, de un proyectos de graduación de licenciatura [55] y una tesis de maestría [16].

Reconocimiento de patrones

En general, en teoría de reconocimiento de patrones, se busca asignar un objeto, también referido como *patrón*, a una categoría o *clase* específica dentro de un determinado número de posibilidades [88]. Estos patrones pueden ser cualquier tipo de señal que necesite clasificarse, tal como señales acústicas capturadas en un bosque.

Los patrones, denotados como $\underline{\mathbf{x}}$, se componen de elementos denominados *características*, que corresponden a cada uno de los elementos del vector. Así, al patrón se le puede llamar también vector de características. De esta forma, el problema matemático principal del reconocimiento de patrones sería asignar $\underline{\mathbf{x}}$ a una de M posibles clases. Para esto se utiliza la probabilidad de que el vector pertenezca a cada una de las clases, que se expresa como

$$P(\omega_i|\underline{\mathbf{x}}) \quad , i = 1, 2, .., M \tag{4.34}$$

Para realizar esta clasificación se requiere primero aprender sobre las diferentes clases, para lo que usualmente se utilizan patrones de ejemplo similares a los que se busca reconocer. Esto se conoce como *aprendizaje supervisado*. En el caso de este trabajo, las señales acústicas de ejemplo corresponden a sonidos normales en el bosque, así como sonidos de disparos y motosierras. El SiRPA, entonces, es básicamente la implementación algorítmica software-hardware del proceso de clasificación de señales, una vez que este sistema ha sido entrenado para reconocer señales específicas..

Estimación de energía de señales

En la implementación inicial del sistema, no existía la unidad de AGC descrita en la primera sección de este análisis. Por ello, se implementó en software una unidad de

normalización de señales. Además, a la salida del bloque del banco de filtros en laFig.4.131 es necesario también estimar la energía resultante del proceso. Para una señal de variable discreta, esta energía es según [62]

$$E_x = \sum_{n=-\infty}^{\infty} |x(n)|^2 \tag{4.35}$$

o en caso de limitarla a un rango finito de tiempo

$$E_N = \sum_{n=-N}^{N} |x(n)|^2 \tag{4.36}$$

En el dominio de la frecuencia, y utilizando la relación de Parseval, este energía puede representarse como

$$E_x = \frac{1}{2\pi} \int_{-\pi}^{\pi} |X(\omega)|^2 d\omega \tag{4.37}$$

Si bien (4.35) y (4.37) permiten obtener la energía de la señal de manera exacta, presentan el problema de que se requiere la totalidad de la señal disponible para realizar el cálculo, lo que en un sistema que opera en línea no es posible. Además, aun cuando (4.36) permite obtener la energía de la señal en un intervalo que incluya sólo muestras previas, resulta computacionalmente pesada.

Dentro de la cadena de procesamiento del SiRPA existen diversos bloques que requieren obtener un estimado de la energía de una señal. Es por esto que se han buscado estrategias de estimación de energía que sean más sencillas de implementar, y presenten menores requisitos de tiempo de cálculo, aunque ello signifique una pérdida de precisión. Una de las opciones corresponde al uso de un seguidor de envolvente, también conocido como demodulador AM, que se muestra en la Fig.4.111.

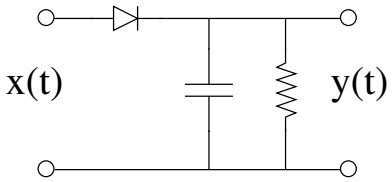


Figura 4.111: Circuito seguidor de envolvente (equivalente a un filtro de promedio móvil). Tomada de [55].

Este filtro presenta un comportamiento de promedio móvil, lo que permite obtener un estimado de la energía de la señal en cada momento [68].

La figura 4.112 muestra el proceso de obtención del valor promedio, donde muestra el efecto de la constante RC del filtro pasa bajo.

La salida del circuito, una vez discretizadas las ecuaciones que la describen, es

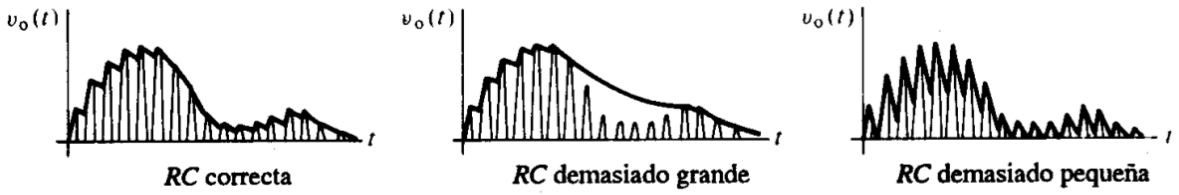


Figura 4.112: Proceso de demodulación con detector de envolvente. Tomada de [85].

$$y[n] = \begin{pmatrix} |x[n]| & \text{si } |x[n]| \geq y[n-1] \\ \alpha \cdot y[n-1] & \text{si } y[n-1] > |x[n]| \end{pmatrix} \tag{4.38}$$

donde el parámetro α se relaciona con la constante RC como

$$\alpha = \exp\left(-\frac{T}{RC}\right) \tag{4.39}$$

Normalización de señales

El proceso de normalización consiste en llevar una señal pasada como entrada a un nivel de referencia común, con el fin de prepararla para su posterior procesamiento y análisis probabilístico [5]. El objetivo de este proceso es sobre todo acomodar el amplio rango dinámico que tienen las señales acústicas, al limitado rango de un sistema computacional de punto fijo. En las primeras etapas del proyecto, esta normalización se realizó vía el promediador descrito. No obstante, para las etapas finales, se introdujo un AGC a la etapa de adquisición de señales, tal como ya fue descrito en la sección 4.1.1. Esto eliminó la necesidad de este proceso de normalización para la etapa inicial del sistema.

Filtros digitales y bancos de filtros

En el sistema propuesto (Fig.4.131) se utiliza un banco de filtros digitales para separar la señal capturada en bandas espectrales.

Filtros

Un filtro es un tipo de sistema que discrimina o extrae algún atributo de una señal aplicada a su entrada [62]. En procesamiento de señales, filtro es un término utilizado usualmente para indicar un sistema lineal e invariante en el tiempo (LTI) capaz de filtrar diferentes componentes de frecuencia de la señal aplicada a su entrada, de acuerdo a su respuesta de frecuencia $H(\omega)$.

Los filtros digitales pueden ser separados en dos categorías [62], los de respuesta finita al impulso (FIR) y los de respuesta infinita al impulso (IIR). Los filtros IIR tienen un $h(n)$

con extensión infinita, por lo que se implementan por medio de *ecuaciones de diferencias*, que son el equivalente discreto de las ecuaciones diferenciales; estas describen los sistemas IIR de manera recursiva como

$$y(n) = -\sum_{k=1}^{N} a_k y(n-k) + \sum_{k=0}^{M} b_k x(n-k) \tag{4.40}$$

donde N recibe el nombre de orden del sistema e indica el número de polos, mientras que M indica el número de ceros. Los coeficientes a_k y b_k provienen de la denominada función de transferencia del sistema $H(z)$, y de ellos depende la ubicación de cada uno de los polos y los ceros que describen al sistema. Es posible obtener $H(z)$ al aplicar la herramienta matemática conocida como transformada z a la ecuación de diferencias del sistema analizado.

Los filtros IIR son más simples de implementar que los FIR y, aunque introducen un comportamiento de fase no lineal [62], esto no es determinante en aplicaciones de audio, o en aquellas donde solamente interesa determinar características como la energía de la señal, tal como el caso de este proyecto.

Existen diferentes formas de implementar los filtros IIR de acuerdo a como se manipule (4.40); estas buscan reducir requisitos de memoria, efecto de errores o ruido de cuantización de los coeficientes en la ubicación de los polos y ceros, entre otros factores. La denominada forma directa II [62] permite reducir el número de elementos de memoria requeridos al mínimo, al introducir en (4.40) una variable intermedia $\omega(n)$. El diagrama de bloques que describe esta implementación se muestra en la figura 4.113, donde el bloque z^{-1} representa un retardo de una muestra y deriva de las propiedades de la transformada z.

Bancos de filtros

A la organización de un conjunto de filtros, digitales en este caso, se le denomina banco de filtros. Esta estructura toma una señal de entrada y la separa en diferentes bandas de frecuencia, que pueden o no traslaparse. Así, la salida de cada uno de los elementos que componen el banco representa una banda específica.

Existen diferentes estructuras para construir el banco [78], así como diferentes maneras de segmentar el espectro de frecuencia. Una de las alternativas es la separación en octavas, que consiste en realizar una segmentación del espectro de manera tal que cada banda tenga la mitad del ancho de banda de la anterior. Esta separación se muestra en la figura 4.114.

Este tipo de separación [78] se ha usado en aplicaciones de compresión de señales de audio, y se implementa mediante el uso de filtros espejo en cuadratura (QMF). Estos últimos se dimensionan con el fin de que tengan una respuesta que abarca en conjunto todo el espectro de frecuencias, permitiendo así separar el espectro de la señal de entrada en su

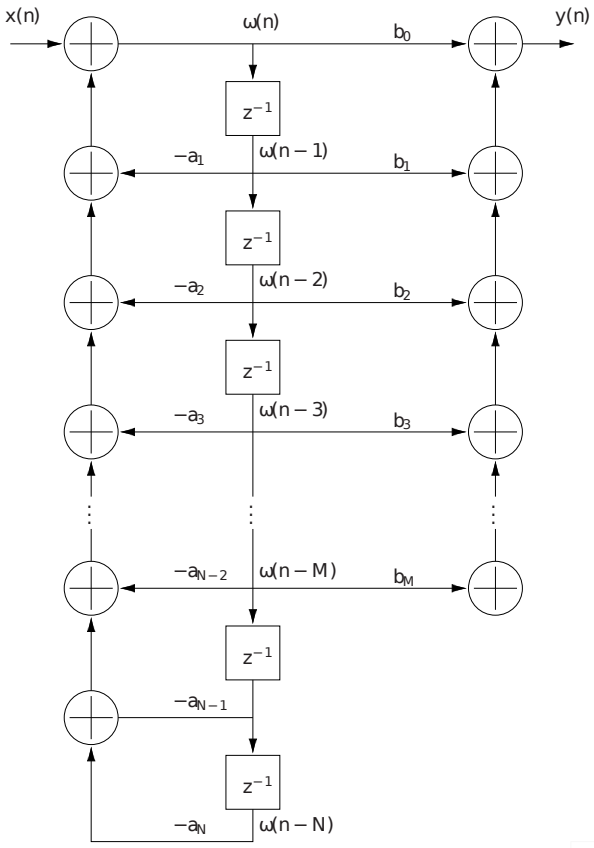


Figura 4.113: Estructura de forma directa II. Tomada de [55].

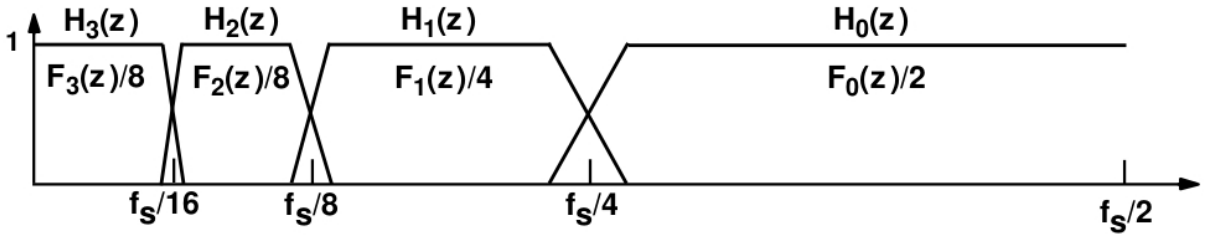


Figura 4.114: Separación del espectro en octavas. Tomada de [78].

mitad superior e inferior. El diagrama de bloques para este tipo de banco se muestra en la figura 4.115.

Este representa un sistema multitasa, dado que no todos los filtros trabajan a la misma frecuencia de muestreo. Esta es dividida en dos por el bloque de diezmado intermedio cada vez que se pasa a la siguiente banda.

Este banco funciona separando el espectro de la señal de entrada en dos; la mitad superior es la generada por el filtro pasa altas (HPF), y corresponde a la octava de salida deseada. La mitad inferior corresponde a la salida del filtro pasa bajas (LPF), y es pasada como entrada al siguiente par de filtros del banco, repitiendo el proceso hasta generar el número de bandas deseadas. Los bloques de diezmado tienen la finalidad de tomar el espectro de la mitad inferior, que abarca idealmente el rango de frecuencias normalizadas [62] de $-\frac{1}{4}$

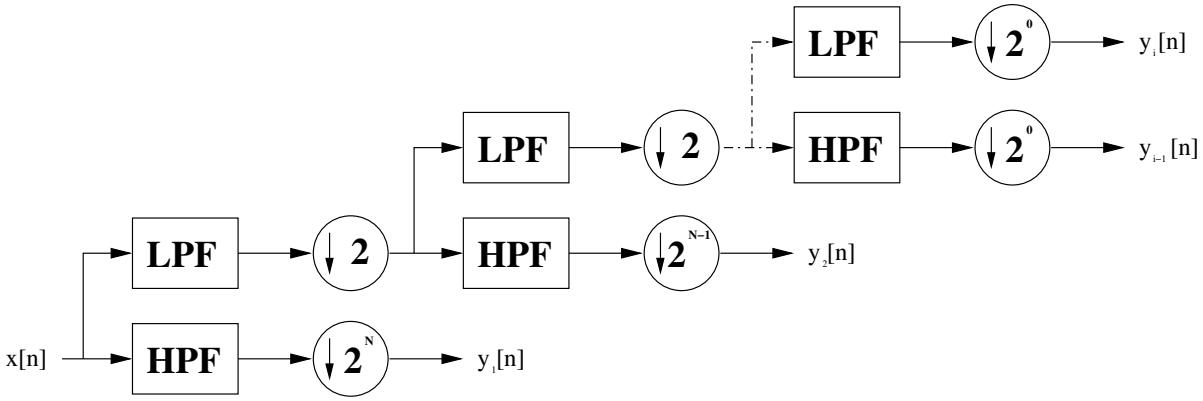


Figura 4.115: Banco de filtros para codificación subbanda. Tomada de [55].

a $\frac{1}{4}$, y expandirlo al rango de $-\frac{1}{2}$ a $\frac{1}{2}$ para que divisiones subsiguientes den los resultados correctos.

La ventaja de este tipo de banco es que gracias al diezmado y a la división del espectro realizados, todos los pares de filtros LPF y HPF son iguales, por lo que sólo debe implementarse uno de cada uno, lo que es útil en la implementación de hardware. Además, la reducción del número de muestras que se pasan a la salida del banco permite que etapas posteriores trabajen a frecuencias menores, reduciendo así las exigencias de procesamiento. El banco tiene también un tiempo de procesamiento menor, dado que no todos los pares de filtros del banco trabajan continuamente.

Reducción de dimensiones

El proceso de reducción de dimensiones busca pasar un conjunto de datos descritos por vectores de n dimensiones a un espacio de m dimensiones, por medio de una transformación lineal del vector de entrada $\underline{\mathbf{x}}_i$ al vector de salida $\underline{\mathbf{y}}_i$ [80]. Para esto se realiza el producto matriz-vector expresado como

$$\underline{\mathbf{y}}_i = \mathbf{W}\underline{\mathbf{x}}_i \tag{4.41}$$

donde $\mathbf{W}$ se conoce como matriz de transformación, y tiene tamaño $n \times m$.

Esta transformación asume que los datos tienen media cero, lo que se puede forzar restando el promedio $\underline{\boldsymbol{\mu}}$ a cada dato como

$$\underline{\mathbf{x}}_i = \underline{\mathbf{x}}_i - \underline{\boldsymbol{\mu}}; \quad \underline{\boldsymbol{\mu}} = \frac{1}{N}\sum_{i=1}^{N}\underline{\mathbf{x}}_i \tag{4.42}$$

lo que es necesario realizar con el fin contribuir a la estabilidad numérica del proceso.

El objetivo del procedimiento de reducción de dimensiones es simplificar el procesamiento en etapas posteriores del sistema, dado que según lo predice la *maldición de la dimen-*

sionalidad [80] el conjunto de símbolos necesarios para describir de manera adecuada las observaciones realizadas es 3 órdenes de magnitud mayor al trabajar en un espacio de 8 dimensiones en comparación a uno de 3 dimensiones. Además de los requisitos de memoria adicionales, se requeriría de alrededor de 3 veces más tiempo de procesamiento para realizar búsquedas en un árbol binario k-d.

El análisis de discriminantes lineales, conocido también como discriminantes lineales de Fisher, busca realizar una proyección del vector $\underline{\mathbf{x}}$ a otro vector $\underline{\mathbf{y}}$ en un espacio de menores dimensiones, de manera tal que se maximice la varianza entre los vectores de diferentes clases, a la vez que se minimiza la varianza entre vectores de una misma clase.

Este análisis identifica la matriz $\mathbf{W}$ que realiza de manera óptima esta transformación, al resolver un problema de eigenvalores y eigenvectores que da por resultado cada uno de los coeficientes que la componen. Los detalles para este análisis pueden ser consultados en [80, 23].

Organización de datos y árboles k-d

Para encontrar símbolos discretos asociados a la señal acústica, se deben identificar centroides en la etapa de entrenamiento, para lo que se usa un algoritmo de aglomeración (k-medias). Posteriormente, debe identificarse eficientemente cuál de los centroides es el más cercano a la señal de salida del módulo de reducción de dimensiones, proceso de búsqueda que puede realizarse mediante un árbol binario.

Los árboles k-d (k-d tree en inglés) son árboles binarios donde cada uno de sus nodos tiene componentes en k dimensiones diferentes. Esta es una estructura de datos que permite ordenar un conjunto de N elementos en un árbol balanceado con profundidad $\log_2(N)$ (redondeado a la unidad superior). Este ordenamiento permite agilizar los procedimientos de búsqueda, que resultan en algoritmos que exhiben un comportamiento $O(\log n)$, y que reducen significativamente el número de comparaciones requeridas para alcanzar la solución buscada en comparación al conjunto de datos sin ordenar.

Existen diferentes maneras de construir un árbol k-d balanceado a partir del conjunto de datos introducido [57]. Una de ellas consiste en ir alternando entre cada una de las k dimensiones en orden secuencial, conforme se cambia entre los niveles de profundidad del árbol; en cada caso, se toma como punto de pivote la mediana de los datos en esa dimensión, se introduce el nodo en el árbol y se divide el conjunto de datos en dos, con los elementos menores al pivote en un grupo y los mayores en el otro. El proceso se repite hasta insertar todos los datos en el árbol. Otra alternativa consiste en tomar el conjunto de datos, determinar la dimensión con mayor variación de valores y tomar la mediana en ella. Se subdivide el conjunto igual que antes y se repite el procedimiento.

La última alternativa presentada, y que es la utilizada en esta implementación, consiste en determinar la dimensión de mayor varianza, tomar la mediana en ella y agregarla al árbol, y dividir luego el conjunto de acuerdo a si cada elemento es mayor o menor al valor

del pivote en la dimensión determinada, procediendo a repetir el procedimiento en cada subconjunto. El objetivo de este tipo de construcción del árbol es reducir el número de comparaciones requeridas para obtener el nodo más cercano al punto de entrada en los algoritmos de búsqueda, dado que al segmentar el espacio de esta manera se reduce la probabilidad de que se tenga que visitar innecesariamente ramas del árbol y por ende se reduce el número de pasos a realizar para obtener el resultado final [61].

Las búsquedas en el árbol se realizan por la denominada *búsqueda del vecino más cercano*, o "nearest neighbor search" en inglés. Esta consiste en determinar cuál de los N elementos que componen el árbol se encuentra más cercano a un punto pasado como entrada, de acuerdo a una métrica predefinida tal como la distancia euclidiana. Se comparan los resultados obtenidos contra todos aquellos elementos del árbol que deben ser revisados, y se selecciona como resultado final aquel que minimiza esta métrica.

Al igual que con la construcción del árbol, existen diferentes maneras de realizar la búsqueda [89, 57], donde el objetivo final de cada una es hacerla más eficiente. La elegida acá hace uso de un hiper-rectángulo que encierra o contiene todo el conjunto de datos, donde sus límites vienen dados por los nodos con el menor y mayor valor en cada dimensión.

Al realizar la búsqueda, este hiper-rectángulo es dividido de manera recursiva por un hiper-plano, hasta llegar a un nodo hoja. Una vez hecho esto se calcula la distancia del punto hasta el nodo y se van deshaciendo las divisiones realizadas. En cada caso se compara el mejor resultado actual contra la distancia que se tiene desde el punto de entrada hasta los otros nodos, para verificar cuál de ellos presenta la distancia mínima. Además, se compara la mejor distancia contra la que se tiene entre el punto y los lados del hiper-rectángulo, donde en caso que la última sea mayor se pueden descartar ramas enteras para la búsqueda. Esto hace que el algoritmo reduzca el número de comparaciones requeridas para obtener la salida, lo que reduce tiempos de procesamiento.

Análisis de conglomerados de k-medias

El análisis de conglomerados de k-medias (k-means clustering en inglés) permite obtener los centroides que mejor agrupan los vectores o datos de entrada denotados como $\underline{\mathbf{x}}_i$, donde cada uno tiene I componentes. Cada uno de los aglomerados, o "cluster" en inglés, es descrito por un vector $\underline{\mathbf{m}}_k$ que describe su media en cada dimensión.

Para realizar la agrupación de cada uno de los datos de entrada se utiliza algún tipo de métrica, tal como la mínima distancia euclidiana desde el punto dado por sus vectores hasta cada uno de los grupos. Definiendo este criterio se realiza el agrupamiento en dos pasos: *asignación* y *actualización*.

La *asignación* toma cada uno de los vectores de $\underline{\mathbf{x}}_i$ y los asigna a uno de los grupos descritos por $\underline{\mathbf{m}}_k$, de acuerdo a la métrica utilizada. La *actualización* utiliza todos los valores asignados a cada grupo y por medio de las componentes individuales actualiza la

media de los vectores de $\underline{\mathbf{m}}_k$. Ambos pasos se repiten hasta que no existan cambios en los valores asignados a cada uno de los grupos, pues luego del paso de actualización podrían variar. Además, al inicio del proceso se generan valores aleatorios para cada uno de los grupos, que son actualizados posteriormente.

Una descripción detallada para este análisis, así como de las ecuaciones involucradas en cada uno de los pasos del proceso, pueden ser consultados en [49].

Modelos Ocultos de Markov

Los modelos ocultos de Markov (HMM) son una herramienta estadística que permite modelar sistemas y procesos, y que han sido utilizados en aplicaciones de reconocimiento del habla, reconocimiento de escritura, reconocimiento facial, análisis de secuencias de ADN, estimación o identificación de acordes musicales, entre otras. Estos constituyen el último paso del modelo SiRPA (Fig.4.131). Para detalles teóricos puede consultarse [69], mientras que para las diferentes estrategias de entrenamiento puede verse también [51, 16].

Los HMM se definen a partir de los siguientes elementos, que describen cada una de sus características:

- N, que corresponde al número de estados que componen el modelo. Los estados pueden o no tener algún significado físico. Usualmente un estado puede ser alcanzado por cualquier otro (modelo ergódico), aunque no necesariamente.
- M, que indica el número de símbolos diferentes que se pueden emitir u observar por el modelo. Estos símbolos corresponden a la salida del sistema que se está modelando. Este parámetro indica el tamaño del alfabeto discreto, donde cada elemento se identifica como $V = \{v_1, v_2, \ldots, v_M\}$.
- La matriz $\mathbf{A}$, que indica la distribución de probabilidad de transición de estados. Esta matriz es descrita por sus coeficientes como $\mathbf{A} = \{a_{ij}\}$, donde cada uno de ellos define la probabilidad de pasar del estado i al estado j. Esta matriz es de tamaño $N \times N$.
- La matriz $\mathbf{B}$, que indica la distribución de probabilidad de observación de símbolos, y que se describe como $\mathbf{B} = \{b_j(k)\}$, donde cada coeficiente indica la probabilidad de observar el símbolo k en el estado j. Esta matriz es de tamaño $N \times M$.
- El vector $\underline{\boldsymbol{\pi}}$, que da la distribución inicial de estados, es de tamaño $1 \times N$, y se describe como $\underline{\boldsymbol{\pi}} = \{\pi_i\}$. Así, cada coeficiente indica la probabilidad de que al inicializarse el modelo se esté en el estado i.

Los HMM se describen en su totalidad por las matrices $\mathbf{A}$, $\mathbf{B}$ y el vector $\underline{\boldsymbol{\pi}}$, que suelen resumirse como $\lambda = (\mathbf{A}, \mathbf{B}, \underline{\boldsymbol{\pi}})$ [69]. Los modelos tienen como entrada una cadena de observaciones, que se denota como $O = O_1, O_2, \ldots, O_T$.

Existen tres problemas que se presentan a la hora de utilizar los HMM para modelar un proceso o sistema, y que deben resolverse para que sean útiles en aplicaciones prácticas

[69]. En este caso, el interés se centra en el primero de ellos, que plantea lo siguiente: *dada una cadena de observaciones O y un modelo λ, ¿cómo se puede determinar de manera eficiente $P(O|\lambda)$, la probabilidad de que la cadena haya sido generada por este modelo?*

Para resolver este problema se utiliza el denominado algoritmo de evaluación hacia adelante (forward en inglés). Este se divide en tres pasos, empezando por la *inicialización*, dada por

$$\alpha_1(i) = \pi_i b_i(O_1) \ , \ 1 \leq i \leq N \tag{4.43}$$

Una vez realizado este paso, se realiza la *inducción*, expresada como

$$\alpha_{t+1}(j) = \left[\sum_{i=1}^{N} \alpha_t(i) a_{ij}\right] b_j(O_{t+1}) \ , \ 1 \leq t \leq T-1 \ , \ 1 \leq j \leq N \tag{4.44}$$

Finalmente, se realiza el paso denominado *terminación*, dado por

$$P(O|\lambda) = \sum_{i=1}^{N} \alpha_T(i) \tag{4.45}$$

Este procedimiento presenta un problema numérico para su implementación, dado que conforme crece el número de observaciones de la cadena (T), el número de estados del modelo (N) o el tamaño del alfabeto discreto (M), las probabilidades de salida resultantes tienden rápidamente a cero. Esto se debe a que las matrices que describen al sistema se componen de coeficientes con valores pequeños (mucho menores a la unidad por lo general), por lo que al realizar todos los productos necesarios se llega a este problema. Por ello, debe realizarse un procedimiento de escalado [69, 52] en cada una de estas ecuaciones.

Se introduce entonces el factor de escala

$$c_t = \frac{1}{\sum_{i=1}^{N} \alpha_t(i)} \tag{4.46}$$

Así, (4.43) y (4.44) se modifican solamente cambiando α por α', con

$$\alpha_t'(i) = c_t \alpha_t(i) \ , \ 1 \leq i \leq N \tag{4.47}$$

Así, se efectúan los cálculos tal como se muestra en (4.43) y (4.44), pero aplicando el escalado dado por (4.47) *antes* de pasar al siguiente paso o realizar la siguiente iteración, con el fin de que los valores α sean los correctos. Con este procedimiento ya no es posible utilizar (4.45), y en su lugar se utiliza el valor de su logaritmo

$$\log(P(O|\lambda)) = -\sum_{t=1}^{T} \log(c_t) \tag{4.48}$$

por lo que los resultados de la evaluación han pasado de estar en el rango de 0 a 1, y se ubicarán ahora entre -∞ y 0.

Matrices de confusión

Las matrices de confusión son una herramienta que permite validar la precisión de un clasificador multiclase. Estas ofrecen tanto la información sobre las tasas de reconocimiento de los HMM utilizados en la etapa de clasificación, como la necesaria durante el entrenamiento para verificar el efecto de las variaciones de los diferentes parámetros de los modelos en sus tasas de reconocimiento.

La matriz de confusión [95], conocida como $\mathbf{C}$, es una matriz de tamaño $N \times N$, donde N es el número de clases usadas en la etapa de clasificación. Las columnas de esta matriz indican la clase *real* a la cual pertenecen los objetos, mientras que las filas indican a cuál de estas clases fueron asignados por el clasificador. Así, el elemento c_{ij} de la matriz de confusión indica un objeto que fue asignado a la clase i, pero pertenece realmente a la clase j.

La matriz de confusión puede resumir los resultados de la clasificación de diferentes formas. La primera es la matriz de confusión *en bruto*, donde se llena $\mathbf{C}$ indicando cuantas veces se dio una determinada situación de clasificación; es decir, cuantas veces el resultado del clasificador fue cada elemento c_{ij}. A partir de esta matriz se pueden obtener dos valores que aportan más información sobre los resultados, que son la *sensitividad* y el *valor predictivo positivo* (VPP).

La sensitividad indica cuál es la probabilidad de que un objeto que pertenece a una determinada clase sea correctamente clasificado. Este valor puede ser estimado al dividir los elementos de la matriz de confusión en bruto entre el total de objetos que componen cada clase real. Es decir, se divide cada elemento de una columna entre la suma del total de elementos de esa columna. De esto último se desprende que la suma de los elementos de cada columna es la unidad.

El VPP indica cuál es la probabilidad de que un objeto asignado a una clase pertenezca verdaderamente a esa clase. Este valor se puede estimar tomando los elementos de la matriz de confusión en bruto y dividiéndolos entre el número total de elementos que se asignaron a cada clase. Es decir, se dividen los elementos de cada fila entre la suma del total de elementos de su fila respectiva, por lo que la suma de los elementos de cada fila suma uno.

Tanto la sensitividad como el VPP son útiles para estudiar la efectividad del clasificador, y lo ideal sería que ambos valores sean altos para todas las clases o elementos $c_{11}, c_{22}, \ldots, c_{NN}$ de la matriz. En este trabajo, la sensitividad indica qué tan efectiva es la

clasificación de las cadenas de observaciones, mientras que el VPP indica qué tan efectivo es el modelo para reconocer las cadenas de su propia clase.

4.4.2 Desarrollo de la sección de entrenamiento del SiRPA

El entrenamiento busca determinar los valores de las constantes que deben utilizarse en cada uno de los bloques del sistema, y que permitan maximizar las tasas de reconocimiento para cada evento que se busca detectar, permitiendo determinar de manera más precisa el estado del medio y evitar así falsos positivos y falsos negativos.

Como se verá posteriormente, la sección de ejecución (que corre en el sistema embebido, ver 4.4.3) comparte la mayor parte de los bloques explicados en esta subsección, por lo que luego se explicarán sólo las diferencias de implementación entre ambas. A continuación se presenta de manera detallada la solución completa realizada con miras a la generación del software de entrenamiento del sistema. Se explicacada elemento de manera individual; en aquellas secciones que involucran entrenamiento, se presenta el algoritmo que lo realiza antes de presentar el bloque a entrenar.

Todos los bloques llevan a cabo sus cálculos utilizando valores en punto flotante, y la implementación es desarrollada en C/C++ usando el entorno de desarrollo Qt Creator [59]. La aplicación base para este trabajo es la propuesta en [16].

Promediador de señal y estimador

Aunque se implementaron rutinas para desarrollar estas etapas, las mismas fueron luego retiradas al introducirse un AGC en la sección de adquisición (ver 4.1.1. No obstante, los detalles de esta implementación pueden verse con detalle en [55].

Banco de filtros

El banco de filtros implementado utiliza codificación subbanda, separando la señal de entrada en diferentes bandas de frecuencia, cada una con un tamaño de una octava menor a la anterior. En la figura 4.116 se muestra el diagrama de bloques del sistema, en donde se han colocado estimadores de energía a la salida de cada banda.

La tabla 4.46 muestra la distribución de frecuencias por banda.

En este caso, y siguendo el diagrama de la figura 4.116, se tendría que $i = 9$ pues se tienen 9 salidas del banco, una por cada banda; además, $N = 7$, pues existen 7 bloques de diezmado desde el punto donde entra la señal hasta el último par de filtros del banco. Sin embargo, la salida del último LPF del banco, en la banda 9, es descartada, pues un análisis previo [82] mostró que no aporta información relevante para la transformación realizada por el siguiente bloque en la cadena de procesamiento.

Los estimadores de energía mostrados fueron descritos ya en la sección 4.4.1. El estimador

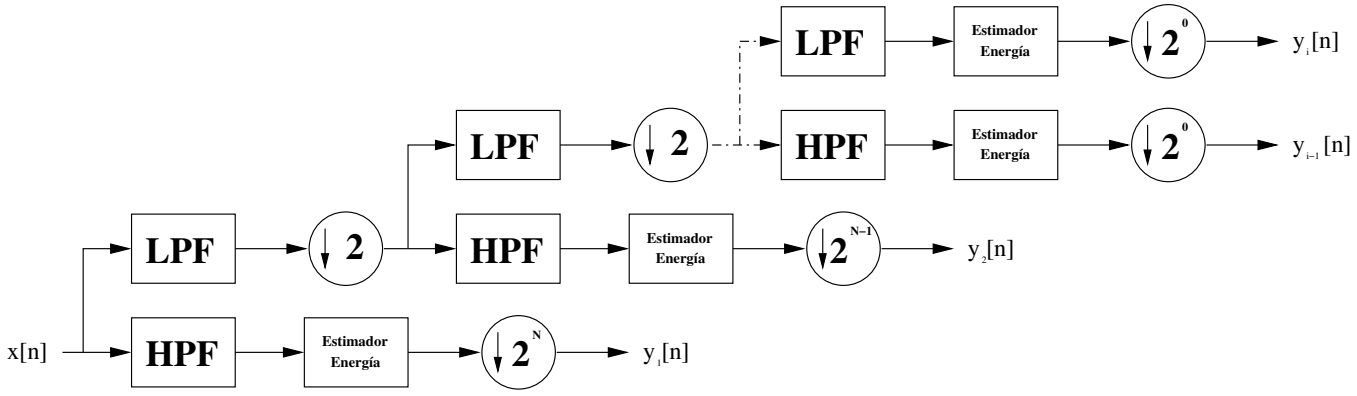


Figura 4.116: Diagrama de bloques para el banco de filtros. Tomada de [55].

Tabla 4.46: Distribución de frecuencias en el banco de filtros. Tomada de [55].

Banda	Frec. Muestreo (Hz)	Frec. Superior (Hz)	Frec. Inferior (Hz)
1	44100	22050	11025
2	22050	11025	5512,5
3	11025	5512,5	2756,25
4	5512,5	2756,25	1378,125
5	2756,25	1378,125	689,0625
6	1378,125	689,0625	344,53125
7	689,0625	344,53125	172,265625
8	344,53125	172,265625	86,1328125
9	344,53125	86,1328125	0

conectado a la salida del primer HPF usa la misma k_{decrec} que se utiliza para el bloque promediador del normalizador, pero este valor se incrementa en un factor de 2 conforme se avanza en cada banda (donde la frecuencia de muestreo disminuye en el mismo factor). Esto se hace con el fin de que el efecto del estimador de energía sea similar en todas las bandas, pues conforme se disminuye la frecuencia de muestreo, cada banda tiene un menor número de muestras para procesamiento, y sin este ajuste no se obtendrían niveles de energía equivalentes en cada salida del banco. Los estimadores son muestreos a la frecuencia más baja del banco de filtro. Así, se muestrea de una sola vez el contenido energético en cada banda.

En la implementación inicial se olvidó incluir a la salida del banco estos bloques de diezmado, lo que provocaba que la salida se actualizara a la misma frecuencia de muestreo de la entrada. Los bloques del reductor de dimensiones y del árbol k-d también funcionaban a esta frecuencia, provocando que se diera una repetición de símbolos en las cadenas de observaciones por periodos prolongados de tiempo, al no capturar la variación en todas las bandas de frecuencia. Esto fue corregido al incluir los bloques de diezmado.

LDA

El entrenamiento con LDA se realiza mediante una aplicación externa de señales al programa de entrenamiento implementado, utilizando los vectores de salida del banco de filtros. Cada uno se toma a la misma frecuencia con la que se actualiza la salida del banco, y son vectores en un espacio de 8 dimensiones.

Los vectores usados para entrenamiento son extraídos y almacenados en archivos de texto, utilizando una base de datos basada en MySQL para realizar su administración y consulta. Cuando se desea realizar el entrenamiento se lleva a cabo la preparación de matrices individuales para cada clase, utilizando las muestras seleccionadas. La matriz de transformación resultante es almacenada en la misma base de datos, y una vez obtenida puede ser utilizada en el módulo de reducción de dimensiones. La aplicación externa, basada en la LTI-Lib [26], fue el resultado de [80].

Reductor de dimensiones

El objetivo del reductor de dimensiones es transformar el espacio de valores de salida del banco de filtros, de ocho dimensiones, a un espacio tridimensional. La transformación se hace mediante un producto matriz-vector, ejemplificada en el diagrama de bloques de la figura 4.117.

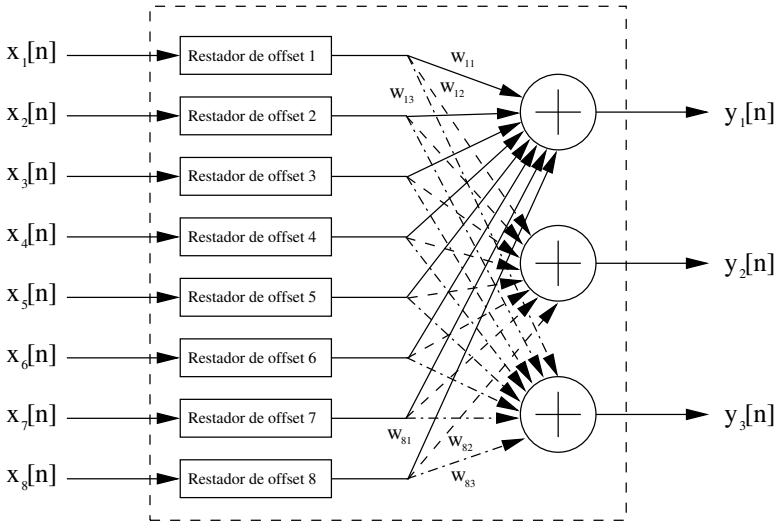


Figura 4.117: Diagrama de bloques para el reductor de dimensiones. Tomada de [55].

Este bloque carga los valores usados por la matriz de transformación y por el vector de media desde archivos generados a partir del entrenamiento con LDA, en caso que hayan sido exportados primero. Estos valores son almacenados en arreglos internos con el fin de realizar primero la resta de la media y luego aplicar el producto matriz-vector a todas las muestras de salida del banco de filtros.

k-medias

El algoritmo de análisis de conglomerados de k-medias (k-means en inglés), es utilizado con el fin de generar los centroides en el espacio tridimensional que identifican a cada uno de los símbolos que componen el alfabeto discreto utilizado en la etapa de clasificación.

Para la administración y extracción de los vectores se utilizan archivos de texto y una base de datos basada en MySQL. El entrenamiento se realiza llamando a una aplicación externa a la que se le pasa como parámetro un archivo que contiene la totalidad de los vectores seleccionados para entrenar. La aplicación genera un archivo que contiene los centroides calculados, que son posteriormente cargados y almacenados en la base de datos, y que además pueden ser vinculados con el sistema y utilizarlos para generar el árbol k-d.

Árbol k-d

En esta implementación se utiliza un árbol k-d para organizar los 32 centroides, de 3 dimensiones cada uno, generados a partir del entrenamiento con el algoritmo de k-medias, permitiendo así realizar la búsqueda del centroide cuyas coordenadas se encuentran más cerca del punto pasado como entrada, que es la salida del reductor de dimensiones. Se utiliza como métrica la distancia euclidiana al cuadrado desde cada uno de ellos hasta el punto en análisis, expresada como

$$d = \sum_{i=1}^{k} (\underline{\mathbf{m}}_i - \underline{\mathbf{x}}_i)^2 \tag{4.49}$$

donde $\underline{\mathbf{x}}$ es el vector de entrada y $\underline{\mathbf{m}}$ el vector del centroide en análisis. Se usa esta métrica en lugar de la distancia euclidiana para optimizar el proceso al evitar el cálculo de la raíz cuadrada, dado que los resultados de las comparaciones no se ven afectados.

Una vez que se ha localizado el nodo, se obtiene como salida su identificador o símbolo, que es utilizado para generar la cadena de observaciones que se pasa a los HMM. Cada nodo del árbol almacena los siguientes datos: coordenadas del centroide en 3D, identificador único del nodo, dirección de partición del espacio (que en este caso puede ser x, y ó z) y vecinos a su izquierda y derecha (de acuerdo a la dimensión de partición actual).

El algoritmo de búsqueda del vecino más cercano trabaja de manera recursiva, realizando una división del hiper-rectángulo (o al ser en 3D, el cubo) que contiene todos los centroides conforme se profundiza en cada nodo hasta alcanzar una hoja. Esto se realiza verificando si la coordenada del punto de entrada es menor o mayor a la del centroide con el que se compara en cada momento, de acuerdo a la dimensión de partición de este último, decidiendo así si se mueve hacia la rama izquierda o derecha.

Cuando se alcanza la hoja, se calcula (4.49) usando las coordenadas del centroide asociado a este nodo y las del punto de entrada, y se comienzan a deshacer las particiones realizadas al hiper-rectángulo, verificando nuevamente la métrica y comparándola contra los otros

nodos del árbol para obtener cuál centroide la minimiza, que determinaría el símbolo de salida.

La partición del hiper-rectángulo se realiza con el fin de determinar si debe descartarse o no la revisión de las ramas no visitadas, calculando (4.49) con el punto en análisis y los "lados" del hiper-rectángulo que encierra el subespacio no visitado, que contiene todos los centroides de la rama analizada. Si la métrica resultante es menor a la mejor calculada hasta el momento debe revisarse esta rama, al existir la posibilidad de que el subespacio que ella encierra contenga un centroide que esté más cerca del punto de entrada.

Los algoritmos utilizados por el k-d tree fueron tomados de [90], pero adaptados para los requisitos de esta aplicación. El algoritmo de construcción del árbol en la implementación en que se basa este trabajo presentaba el problema de que no se actualizaban los "lados" del hiper-rectángulo al introducir un nuevo nodo al árbol, lo que provocaba que al realizar las búsquedas se descartaran ramas que no debían descartarse. Esto fue corregido para que el hiper-rectángulo se actualice conforme se insertan los nodos al árbol.

Entrenamiento de los HMM

Con el fin de determinar los valores de las matrices $\mathbf{A}$, $\mathbf{B}$ y el vector $\underline{\boldsymbol{\pi}}$ que caracterizan a cada modelo λ, se utilizan algoritmos de entrenamiento simple, múltiple y negativo. El objetivo de estos es maximizar las tasas de reconocimiento del modelo para eventos de su propia clase, a la vez que se disminuyen para los de las otras clases.

El entrenamiento se lleva a cabo capturando muestras de observaciones, utilizando archivos de audio de ejemplo para las diferentes clases o eventos que se busca reconocer. En la Fig.4.110 se muestra el punto desde el cual se extraen los símbolos para realizar el entrenamiento, los cuales son almacenados en archivos de texto, y administrados por la base de datos implementada en MySQL.

Se utiliza el algoritmo de Baum-Welch [51] para el caso de entrenamiento simple y múltiple, donde para este último se debe modificar el algoritmo tal como se explica en [69]. Para el caso de entrenamiento negativo se utiliza el denominado Bold's Smooth Algorithm [51]. Todas estas variantes de entrenamiento fueron implementadas en un trabajo previo [16]. Sin embargo, tenían problemas de estabilidad al intentar utilizar para entrenamiento o evaluación cadenas de observaciones que no contaban con una longitud suficiente de acuerdo a determinados valores de muestreo y/o longitud de cadena, lo que llevaba al cierre de la aplicación. Esto se corrigió para que aquellas cadenas que no cumplan con la longitud requerida sean descartadas de estos procedimientos.

Evaluación de los HMM

Con el fin de realizar la evaluación de los HMM se utilizan las matrices de confusión, prestando especial atención a los valores de sensitividad y VPP obtenidos. Estos valores permiten estudiar el efecto en las tasas de reconocimiento de la variación de los parámetros

de número de estados, muestreo, longitud de cadena y razón de aprendizaje (para el entrenamiento negativo), así como de las cadenas seleccionadas para entrenar. La idea es maximizar los casos donde la clase asignada i y la clase real j sean iguales, lo que denota una clasificación correcta.

Para formar las matrices se toma una cadena de observaciones para una clase determinada j y se pasa al módulo de clasificación, donde se tienen todos los HMM que se busca evaluar. Estos utilizan el algoritmo de evaluación hacia adelante para obtener $\log(P(O|\lambda))$, y se selecciona como clase asignada i a la clase del modelo que tenga un valor más cercano a cero, pues por el mapeo realizado al introducir los factores de escala, valores cercanos a cero indican una mayor probabilidad de que el modelo haya sido el que generó esa cadena de observaciones. De esta forma se completa cada elemento c_{ij} de la matriz de confusión en bruto $\mathbf{C}$, a partir de la cual se obtienen luego los valores de sensitividad y VPP.

La implementación de la función de evaluación, junto a otras de entrenamiento y administración para los HMM, se realizó utilizando la biblioteca provista en [24], que se encuentra desarrollada en C.

Clasificación

La etapa de clasificación es la que contiene los HMM utilizados para evaluación, los cuales son el resultado del entrenamiento realizado con los algoritmos previamente mencionados. En la implementación en el computador de propósito general, o de entrenamiento, este bloque no existe como parte de la cadena de procesamiento del SiRPA, pero se incluye en el diagrama de la figura 4.110 por claridad.

Cada vez que se lleva a cabo la evaluación de un HMM, o un grupo de ellos, se realiza la carga de los parámetros para cada uno desde la base de datos, junto con las cadenas que hayan sido elegidas. Se procede luego a verificar las probabilidades de salida para cada modelo de acuerdo a la cadena que se le pasa como entrada. Una vez que se realiza este procedimiento con todas las observaciones, se almacenan los valores resultantes en la base de datos, se generan las matrices de confusión por modelo y se generan las estadísticas de reconocimiento, quedando estos resultados disponibles para la revisión por el usuario.

Ejecución

La sección de ejecución corresponde a una versión reducida de la cadena de procesamiento utilizada para entrenamiento, la cual se optimiza para reducir el tiempo de procesamiento por cada bloque de señal, permitiendo al sistema embebido operar en línea a una frecuencia de muestreo definida. En la implementación de hardware del SiRPA, la reducción en los tiempos de procesamiento y el número de pasos requeridos para obtener la salida permiten reducir a su vez el consumo energético del sistema.

La implementación de ejecución hace uso de las diferentes matrices y vectores generados durante el entrenamiento, y solamente tiene la salida de la etapa de clasificación. Dado que

la mayor parte de los bloques ya fueron detallados, se explican a continuación solamente las diferencias con respecto a la implementación de entrenamiento.

4.4.3 Implementación en plataforma embebida Beagle Board

La implementación en el sistema embebido cuenta con dos modos de reconocimiento: utilizando archivos de audio (en formato WAV), o mediante la entrada de audio (de línea) que incluye la plataforma Beagleboard-xM utilizada [9]. La selección entre un modo u otro se da de acuerdo al número de argumentos pasados a la aplicación a la hora de llamarla desde la línea de comandos.

El modo de reconocimiento en base a archivos de audio se incluye con el fin de realizar pruebas de reconocimiento de manera rápida y flexible. Este modo permite comparar los resultados de salida de cada etapa y de las probabilidades finales contra la implementación en el computador de propósito general. La aplicación lee la totalidad del archivo o sólo una parte de él, de acuerdo a la configuración elegida a la hora de compilarla, y genera archivos de texto donde se almacenan los valores de salida de cada una de las etapas de la cadena de procesamiento. Además, imprime en consola el estado determinado a partir de la evaluación de los modelos pasados como parámetros. Se almacena también información sobre tiempos de ejecución por etapa o por bloque procesado (cuyo tamaño es también configurable), lo que permite estimar los tiempos de procesamiento que se tendrán al utilizar el modo de reconocimiento en línea.

El modo de reconocimiento mediante la entrada de audio corresponde al modo de reconocimiento en línea. Este permite que se conecte un micrófono (que debe ser preamplificado, pues el sistema embebido no realiza amplificación alguna) a la plataforma, para realizar pruebas de campo. La captura de los datos desde esta entrada se hace mediante un cliente de JACK [27], que se encarga de tomar bloques de muestras de audio y realizar llamadas a una función de procesamiento específica, en un entorno de baja latencia. Parámetros como la frecuencia de muestreo, tamaño de bloque, número de búferes usados, entre otros, son configurables al iniciar el cliente.

En esta implementación se utiliza un tamaño de bloque de 256 muestras (que por limitaciones del sistema embebido es el máximo utilizable), con una frecuencia de muestreo de 48 kHz (cuyo valor no es configurable en este caso, y es impuesto por el hardware de audio de la plataforma), lo que implica una ventana de tiempo para procesamiento de menos de 5,3 ms, luego de la cual se llama a procesar el bloque siguiente.

Este modo de operación no almacena información en archivos, pues la apertura, escritura y cierra de ellos conlleva tiempos en el rango de los milisegundos, que por las limitaciones de tiempo mencionadas no son tolerables. En su lugar, se imprime solamente en consola información sobre el estado actual del medio con su probabilidad asociada (de acuerdo a evaluación del modelo), actualizando el estado sólo en caso de que éste cambie.

Procesamiento en punto fijo

Con el fin de estudiar maneras de acelerar el procesamiento de los datos y aprovechar los elementos de hardware provistos por el sistema en chip (SoC) DM3730 [39] de la plataforma utilizada, se hace uso del DSP (TMS320C64x+) incluido. Este procesador realiza sus cálculos mediante una representación numérica en punto fijo, por lo que es necesario adaptar los algoritmos que se desean ejecutar en el procesador a este formato. De no hacerlo así el compilador generará el código requerido para realizar la emulación del formato en punto flotante, lo que conlleva tiempos de procesamiento excesivos (decenas de veces más lentos).

Para determinar cuál sección de la cadena de procesamiento del SiRPA transferir al DSP se analizó la frecuencia a la que opera cada uno de los bloques de la etapa de *extracción de características*, que se presentó en la figura 4.110. Estas frecuencias se muestran en la tabla 4.47. La frecuencia de muestreo utilizada como referencia es la de la implementación en ASIC (44,1 kHz), aunque podría utilizarse también la del sistema embebido (48 kHz).

Tabla 4.47: Frecuencia de operación de bloques de la cadena del SiRPA. Tomada de [55].

Bloque	Frec. de operación (Hz)
Banco de filtros	44100
Reductor de dimensiones	344,53125
Árbol k-d	344,53125

A partir de esto se decide transferir el bloque del banco de filtros al DSP, ya que es el único bloque de esta etapa que opera a la misma frecuencia con que se toman las muestras de la señal de audio, por lo que se le transfieren al procesador bloques de señal de gran longitud (256 muestras en este caso) y se evita así el intercambio frecuente de datos entre procesadores, que conlleva un *overhead*. Así, se hace el cambio de todos sus algoritmos para operar usando una representación en punto fijo de 32 bits, con un número de bits configurable para la parte fraccionaria y entera. No es posible utilizar un menor número de bits, pues conllevaría una pérdida de precisión importante respecto a la implementación en punto flotante.

Con el fin de llevar a cabo la compilación de los ejecutables para el DSP, se utiliza código desarrollado en C que se compila mediante la herramienta C6RunLib [38], la cual se encarga de que a la hora de ejecutar la aplicación en el sistema embebido se realice la carga del código necesario en el DSP. Este también introduce *stubs* con el fin de que las llamadas a las funciones transferidas al DSP sean desviadas hacia éste, pasándole el control de los datos y punteros a ellos; los resultados son almacenados también en memoria, retornando el DSP punteros hacia ellos.

La memoria utilizada por el DSP debe ser colocada en un espacio de memoria específico, con el fin de que pueda ser utilizada por el DSP, por lo que C6Run cambia las funciones de solicitud y liberación de memoria por sus propias versiones. Esta necesidad surge a

partir del hecho de que el procesador de propósito general (GPP) ARM cuenta con una unidad de administración de memoria (MMU), que es utilizada por el sistema operativo para operar en un contexto de memoria virtual, mientras que el DSP no cuenta con una de estas unidades y trabaja por lo tanto con memoria física. De esta forma, debe realizarse algún tipo de conversión entre los punteros usados por ambos procesadores, pues de otra forma una dirección de memoria apuntada por el GPP por lo general no concordaría con la apuntada por el DSP, resultando en errores de acceso a los datos.

Generación y optimización de código para el DSP

Para el desarrollo de la sección de código que se ejecuta en el DSP se utiliza el lenguaje de programación C, donde los archivos fuente que se generan se compilan por medio de la herramienta C6RunLib. Además de cambiar todos los algoritmos usados por el banco de filtros y los estimadores de energía conectados a su salida al formato de representación numérica en punto fijo, se utilizaron los denominados *intrinsic* [40] del DSP para mejorar el rendimiento de las funciones implementadas. Estos son instrucciones de ensamblador del DSP que pueden ser llamadas directamente desde el código en C, y que usualmente se ejecutan en un solo ciclo de reloj de este procesador.

En [40] se presentan otras alternativas para el desarrollo de aplicaciones para el DSP, que son el uso de ensamblador lineal y el ensamblador propio del procesador. La diferencia entre ambos es que para el primero debe especificarse solamente las instrucciones específicas del DSP que se desean utilizar, y el compilador se encarga automáticamente de los detalles de paralelismo, asignación de registros y uso de las unidades de ejecución (al tener una arquitectura *pipeline*). Por otro lado, el uso de ensamblador propio del sistema requiere que el programador indique explícitamente estos detalles.

El problema principal con estas alternativas es que la herramienta de compilación (C6RunLib) no permite utilizarlas, obligando a desarrollar el código tal como se describió previamente, donde el compilador genera el código ensamblador a partir del código desarrollado en C.

Dado que se desea estudiar el uso de las unidades del DSP para explotar el paralelismo entre ellas, se agregan banderas al compilador (ver configuración en la sección de apéndices) para que al crear el archivo ejecutable se genere también el archivo que contiene las instrucciones de ensamblador requeridas para realizar cada instrucción de código C. Así, puede estudiarse de qué manera se utilizan las unidades de ejecución del procesador al cambiar el código C y/o al utilizar los *intrinsics*. Esta es la estrategia recomendada en [40] para el desarrollo de aplicaciones, antes de recurrir al uso de código ensamblador que es menos "portátil", y que si no se utiliza correctamente no tendría un mejor rendimiento respecto al obtenido con herramientas automatizadas.

Un ejemplo de código ensamblador generado de manera automática se muestra en la figura 4.118. En este caso, la primera columna indica la instrucción de ensamblador del DSP utilizada, mientras que la segunda columna indica cuál unidad de ejecución se está utilizando. La tercera columna indica los registros utilizados en esa operación, y la última

columna indica cuál es la instrucción de código C asociada a esta instrucción de ensamblador. Por medio de esto se puede reestructurar el código C y utilizar *intrinsics* para lograr que al pasar el código al compilador se utilicen mejor las unidades del procesador, y se pueda paralelizar funciones. En [36] puede consultarse más información sobre la arquitectura interna del DSP utilizado y sus unidades de ejecución.

```
EXCLUSIVE CPU CYCLES: 33
LDW     .D2T2   *B10,B4          ; |dsp_lib/filterDSP.dsp_stub.c:55|
ADD     .D2     SP,16,B5         ; |dsp_lib/filterDSP.dsp_stub.c:55|
ADD     .L2     4,B10,B31        ; |dsp_lib/filterDSP.dsp_stub.c:56|
ADD     .L2     12,SP,B30        ; |dsp_lib/filterDSP.dsp_stub.c:56|
ADD     .L1X    8,B10,A3         ; |dsp_lib/filterDSP.dsp_stub.c:56|
STW     .D2T2   B4,*B5           ; |dsp_lib/filterDSP.dsp_stub.c:55|
LDW     .D2T2   *B31,B5          ; |dsp_lib/filterDSP.dsp_stub.c:56|
ADD     .L1X    8,SP,A4          ; |dsp_lib/filterDSP.dsp_stub.c:57|
ADD     .L2     12,B10,B10       ; |dsp_lib/filterDSP.dsp_stub.c:56|
NOP             2
STW     .D2T2   B5,*B30          ; |dsp_lib/filterDSP.dsp_stub.c:56|
LDW     .D1T1   *A3,A3           ; |dsp_lib/filterDSP.dsp_stub.c:57|
ADD     .L2     4,SP,B5          ; |dsp_lib/filterDSP.dsp_stub.c:58|
NOP             3
STW     .D1T1   A3,*A4           ; |dsp_lib/filterDSP.dsp_stub.c:57|
LDW     .D2T2   *B10,B4          ; |dsp_lib/filterDSP.dsp_stub.c:58|
NOP             4
STW     .D2T2   B4,*B5           ; |dsp_lib/filterDSP.dsp_stub.c:58|
LDW     .D2T2   *+SP(4),B6       ; |dsp_lib/filterDSP.dsp_stub.c:59|
LDW     .D2T1   *+SP(8),A6       ; |dsp_lib/filterDSP.dsp_stub.c:59|
LDW     .D2T1   *+SP(16),A4      ; |dsp_lib/filterDSP.dsp_stub.c:59|
```

Figura 4.118: Código ensamblador generado por herramienta C6RunLib. Tomada de [55].

Optimización de algoritmos para ejecución en el DSP

Dado que el bloque normalizador previo al banco de filtros y el reductor de dimensiones que se ubica luego de él trabajan con una representación numérica en punto flotante, se deben introducir bloques para realizar la conversión de punto flotante a punto fijo y viceversa. El diagrama resultante de esto se muestra en la figura 4.119.

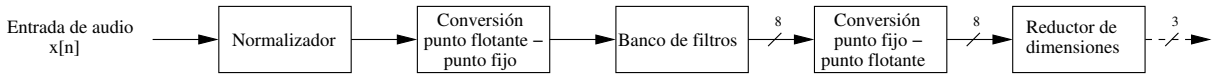


Figura 4.119: Cadena modificada para conversión entre formatos numéricos. Tomada de [55].

Lo primero que se realizó antes de comenzar con las modificaciones del resto del sistema fue determinar el número de bits requeridos para la parte entera y fraccionaria de la representación en punto fijo, para lo que se realizó un *profiling* del código desarrollado en formato punto flotante, y se estudiaron los valores máximos esperados y el tipo de precisión requerida.

Por el tipo de entradas que se esperan desde el bloque normalizador se determinó que se requiere como mínimo de 8 bits en la parte entera, y podrían requerirse más en caso de que se modifiquen las constantes usadas en él. De no usarse esta cantidad de bits se corre el riesgo de que se presente un desbordamiento u *overflow* de los datos, dando lugar a valores erróneos en caso que el resultado salga del rango de representación que da el número de bits elegido.

Por otro lado, se realizaron pruebas para determinar de qué manera varían los símbolos de salida de la etapa de extracción de características con respecto a la implementación de entrenamiento, de acuerdo al número de bits usados para la parte fraccionaria. Se encontró que como mínimo deben utilizarse 20 bits en la parte fraccionaria para que no varíen las cadenas de observaciones generadas y se provoque por lo tanto una clasificación incorrecta de la señal debido a esta variación.

Así, se utiliza una representación en punto fijo de 32 bits, lo que repercute sobre las posibilidades de optimización del código en el DSP dado que este tiene una arquitectura con un tamaño de bus de 16 bits [36], y se encuentra optimizado para trabajar con paquetes de datos ya sea de 8 o 16 bits. Esto afecta también el paralelismo, dado que el mayor número de bits obliga a utilizar varias unidades de ejecución para una sola operación aritmética, que de otra forma hubieran podido ser utilizadas por dos operaciones en paralelo. Por ejemplo, una multiplicación entre dos valores de 32 bits tomará cuatro veces más tiempo que el que tomaría multiplicar dos valores de 16 bits [40].

Definiendo esto, se realizaron las modificaciones requeridas para optimizar el código en C, a la vez que se introdujeron secciones de código para que el compilador optimice secciones de la aplicación. Para esto último se introducen directivas para indicarle al compilador el número de veces mínimo y máximo que se espera se ejecute un ciclo en el código, lo que permite estimar cuándo se presentará una ramificación en su ejecución. Para esto se requiere analizar primero el código y estimar estos valores. Por ejemplo, la siguiente directiva indica que el código se ejecutará exactamente 256 veces, lo que se sabe previamente por el tamaño de bloque elegido.

```
#pragma MUST_ITERATE (256, 256);
```

Además de esto, se le indica al compilador de qué manera se utilizan variables de la función, con el fin de que puedan ser optimizadas en cuanto a colocación en espacios de memoria, acceso a los datos y uso de sus punteros, utilización de la caché en cuanto a localidad espacial y temporal, entre otras cosas. Por ejemplo, en la declaración de la función se colocan los argumentos como

```
INBUF int * restrict inBuffer, OUTBUF int * restrict outBuffer
```

donde INBUF indicaría que el búfer es sólo de entrada, mientras que OUTBUF indica que es de sólo lectura. Esto evitaría que el compilador intente agregar instrucciones y crear variables para escribir o leer de ellos, respectivamente. El término *restrict* indica además que solo el puntero pasado como argumento, y aquellos que se puedan derivar de él, tiene acceso a los datos en todo momento. De esta manera, se dispone libremente de los datos y el compilador hace optimizaciones adicionales.

El código en C se reestructuró además para reducir el número de variables requeridas en las operaciones, y evitar así la asignación innecesaria a registros. Sin embargo, existen casos

donde se dejan variables temporales para ciertas operaciones aritméticas de multiplicación y acumulación, que evitan que se acceda de manera constante a memoria y se utilice en su lugar un registro del procesador, lo que disminuye tiempos de operación. Además, los "objetos" de filtros y el banco de filtros en sí fueron modificados para que a la hora de compilar la aplicación se sepa previamente el tamaño en memoria que ocupan, pues de otra forma la aplicación no puede ser compilada.

Finalmente, se introducen los *intrinsic* para optimizar ciertas operaciones realizadas. Por ejemplo, al realizar la multiplicación de dos valores de 32 bits, el resultado es un número de 64 bits que luego debe ser desplazado para obtener nuevamente un valor de 32 bits que represente el resultado. Esto se puede realizar con dos *intrinsic* como

```
tmpM = _mpy32ll(tmpOp1, tmpOp2);
tmpMul  = _hill((tmpM << INT_BITS));
```

de manera que en la variable tmpMul se tiene el resultado de multiplicar las variables de 32 bits tmpOp1 y tmpOp2. Esto se traduce en las dos instrucciones de ensamblador

```
MPY32
SHRU
```

que toman 4 ciclos y 1 ciclo de reloj del procesador para ejecutarse, respectivamente [37].

Es posible realizar optimizaciones adicionales al código en caso que se reduzca el número de bits usados. Si se utilizan 16 bits en total para la parte entera y fraccionaria se podrían utilizar un mayor número de *intrinsic*, además de que se podrían paralelizar un mayor número de instrucciones. Sin embargo, debe evaluarse la pérdida de precisión que se tendrá en los resultados con respecto a la implementación en punto flotante, y también considerarse los efectos de desbordamiento. Para más información respecto a las posibles optimizaciones se puede consultar [40].

Ordenamiento del árbol k-d

La implementación para la construcción del árbol k-d utilizada en la sección de entrenamiento produce estructuras desbalanceadas, lo que no es un factor crítico debido a la naturaleza de operación fuera de línea de esta etapa. Sin embargo, dado que el sistema embebido trabaja en línea se deben disminuir los tiempos de procesamiento al máximo, por lo que no son tolerables los retardos adicionales debido a esto.

Por ello, se implementa un algoritmo que realiza el ordenamiento del árbol de manera recursiva, ejecutándose al momento de inicializar los parámetros del sistema para que cuando se inicie la captura y procesamiento de datos ya se encuentre balanceado. Esto permite que el algoritmo de búsqueda sea efectivamente del tipo $O(\log n)$, haciendo que

se requiera de tan sólo alrededor de 5 comparaciones como máximo para encontrar el centroide más cercano al punto de entrada [80].

El algoritmo de ordenamiento realiza la partición en base a la dimensión de mayor varianza, utilizando el algoritmo Quicksort para organizar el conjunto de datos. El funcionamiento es el siguiente: se toma el conjunto de datos y se determina cuál de las k dimensiones (tres en este caso) presenta la mayor varianza, usando las coordenadas de todo el conjunto para realizar el cálculo. Se ordenan los centroides de menor a mayor según la coordenada de cada uno en la dimensión determinada y se toma la mediana del conjunto, que se inserta inmediatamente al árbol. El algoritmo se repite de manera recursiva sobre cada uno de los subconjuntos generados, hasta que se hayan insertado todos los nodos en el árbol, resultando en un árbol balanceado.

Clasificación

La etapa de clasificación consiste en un modelo para cada clase a identificar, donde cada modelo estima la probabilidad de que la secuencia de observaciones a su entrada sea emitida u observada por él. Se asume que la clase correcta para la cadena de observaciones es aquella cuyo modelo produzca la mayor probabilidad; en este caso, la que se encuentre más cercana a cero al utilizarse $\log(P(O|\lambda))$.

Al inicializar se cargan todos los parámetros de los modelos desde un archivo que contiene las matrices $\mathbf{A}$, $\mathbf{B}$ y el vector $\underline{\boldsymbol{\pi}}$, junto a los parámetros N, T, F y V. Todos estos valores son generados de manera automática al exportar los modelos desde la implementación de entrenamiento. La figura 4.120 ilustra la estructura interna que se tendría en este bloque al exportar un modelo de cada clase.

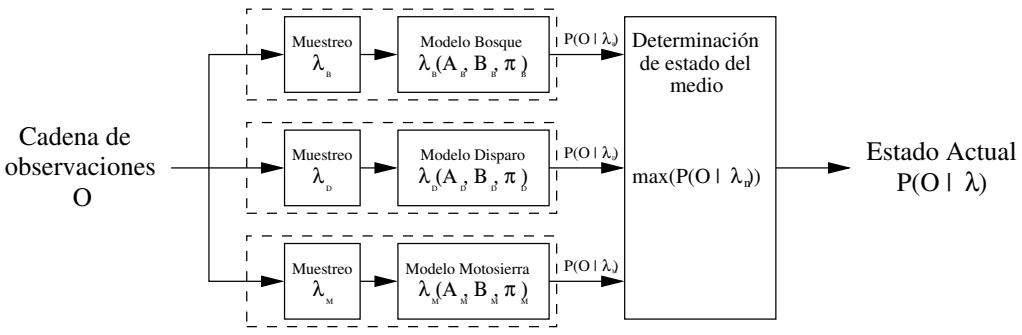


Figura 4.120: Etapa de clasificación en implementación de ejecución. Tomada de [55].

Para realizar la clasificación se utiliza el algoritmo hacia adelante, utilizando las modificaciones presentadas en [69], las cuales introducen factores de escala dentro de los cálculos. El algoritmo original fue tomado de [25], pero fue luego modificado para reducir tiempos de procesamiento y uso de memoria, así como para orientarlo al uso de objetos para facilitar administración y evaluación de los modelos. Se removieron también funciones y secciones de código no requeridas.

Con el fin de realizar la operación en línea, se realiza una adaptación a la forma en que

se realiza el proceso de evaluación [81]. Se realiza el muestreo de símbolos de manera normal, hasta que se alcanza la longitud requerida por el modelo (T); una vez que se cumple esto se llama a la función de evaluación, la cual realiza los tres pasos presentados anteriormente para obtener $\log(P(O|\lambda))$. Luego de calcular esta salida, se "reinicia" el tiempo, procediendo a muestrear símbolos nuevamente hasta que se vuelva a contar con la cantidad de muestras requeridas para evaluar, repitiendo todos los pasos de evaluación.

Este procedimiento puede verse como la aplicación de una ventana a la cadena de observación, la cual inicia en cero y finaliza en $T-1$. Una vez evaluada esta cadena, se desplaza la ventana para iniciar en T y finalizar en $2T-1$. Se muestrea la cadena y se evalúa, repitiendo el proceso hasta llegar al final de la cadena de observación completa, o hasta que se detenga ejecución de la aplicación que opera en línea.

Este método de evaluación tiene sus desventajas. La primera de ellas corresponde al hecho de que los HMM son sensibles a variaciones en los puntos donde se toman los símbolos que componen la cadena. Esto haría que si se muestrea la cadena unos cuantos símbolos antes o después de que se presente un evento de alarma las probabilidades de salida resultantes provoquen una clasificación incorrecta del medio, llevando a falsos positivos o a falsos negativos. Otra desventaja es el hecho de que el proceso de evaluación completo se puede repetir con mucha frecuencia, por lo que en caso de que se usen configuraciones de modelos que sean exigentes en cuanto a tiempo de procesamiento (con muchos estados y longitudes de cadena grandes), podrían llegar a presentarse violaciones al tiempo de procesamiento disponible.

Sin embargo, se elige este método de evaluación dado que es más sencillo y eficiente que otras alternativas, y dado que las limitaciones para la implementación en ASIC impiden el uso de algoritmos más complejos.

Resultados del entrenamiento

Para realizar el entrenamiento de cada una de las etapas que componen el SiRPA, así como la evaluación de las tasas de reconocimiento y la generación de las matrices de confusión en la etapa de clasificación se cuenta con un total de 192 archivos de audio. Estos se clasifican de acuerdo a su clase, y se separan entre cadenas de entrenamiento y evaluación. La tabla 4.48 muestra la distribución de todos los archivos.

Tabla 4.48: Distribución de archivos para entrenamiento y evaluación. Tomada de [55].

Clase	Entrenamiento	Evaluación	Total
Bosque	43	38	81
Disparo	24	21	45
Motosierra	36	30	66
Total	103	89	192

La selección de los archivos usados para entrenamiento tiene efectos en las tasas de re-

conocimiento obtenidas, por lo que deben realizarse pruebas empíricas para mejorarlas. Factores como el muestreo y longitud de cadena son también relevantes, pues además de afectar estas tasas, ciertas combinaciones de estos valores provocarían que deban descartarse cadenas que no cuentan con la longitud suficiente para ser usadas en el entrenamiento o evaluación, reduciendo el número de ejemplos disponibles.

Constantes del normalizador

Las constantes utilizadas por el normalizador son dimensionadas en base a la clase de disparo, con el fin de que se pueda capturar el pico de amplitud producido por la detonación de un arma, pero que la salida del bloque no se vea saturada a un nivel de amplitud alto por un tiempo prolongado. De no cumplirse esto la salida de la etapa de generación de símbolos podría verse saturada a uno o dos valores, lo que no permitiría reconocer el momento en que se dio la transición del estado de disparo al estado normal del medio, dando lugar a un entrenamiento inadecuado de los HMM, y por lo tanto a tasas de reconocimiento pobres.

Así, se seleccionan las constantes k_{crec} y k_{decrec} del promediador con el fin de que se alcance la amplitud máxima del pico, que en este caso es 1, en el tiempo de duración total de la detonación del disparo, y no del eco remanente. En [50] se explica que la duración de esta explosión sónica es de 3-5 ms, por lo que en esta implementación se trabaja con una duración de 5 ms. La figura 4.121 muestra el comportamiento que se busca para el promediador, donde se asume que una vez que se alcanza el pico de amplitud de entrada de 1, éste desaparece y se procede con la descarga lineal.

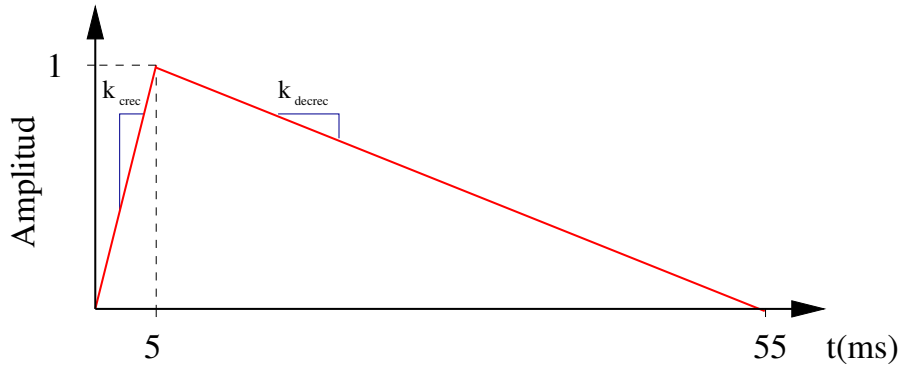


Figura 4.121: Curva usada para determinación de constantes de normalizador. Tomada de [55].

Otra constante importante para el normalizador es k_{nor}, que evita la división por cero, y también disminuye el efecto de ruido de alta frecuencia. Para su determinación se usaron archivos de audio de ejemplo y se empleó la FFT para estimar el valor del piso de ruido en ellos. La constante se selecciona para ser un poco mayor a este valor.

Los valores obtenidos para cada una de las tres constantes se resumen en la tabla 4.49. Se asume que la frecuencia de muestreo es de 44,1 kHz, que es la utilizada por los archivos

de audio de entrenamiento y en la implementación en ASIC.

Tabla 4.49: Constantes utilizadas en el normalizador, con $F_S = 44,1$ kHz. Tomada de [55].

Constante	Valor
k_{crec}	0,004524887
k_{decrec}	0,000452489
k_{nor}	0,000030518

Se calculan además estas constantes para el caso donde la frecuencia de muestreo sea de 48 kHz, que es la utilizada por el sistema embebido. Estas se muestran en la tabla 4.50. En caso que se utilice el modo de reconocimiento en línea deben utilizarse estas constantes, mientras que si se utiliza el modo de reconocimiento sobre archivos, y estos fueron muestreados a 44,1 kHz, deben utilizarse las constantes que se presentaron en la tabla 4.49.

Tabla 4.50: Constantes utilizadas en el normalizador, con $F_S = 48$ kHz. Tomada de [55].

Constante	Valor
k_{crec}	0,004166667
k_{decrec}	0,000416667
k_{nor}	0,000030518

Resultados para LDA

La siguiente matriz fue obtenida mediante el entrenamiento con la totalidad de los archivos disponibles para todas las clases, y utilizando el muestreo por defecto de 1 con el fin de contar con un mayor número de muestras.

$$\mathbf{W} = \begin{bmatrix} -0.407052 & 0.247081 & -0.890227 \\ -0.337315 & 0.071749 & 0.139704 \\ -0.049092 & 0.064765 & 0.043157 \\ 0.205715 & -0.233988 & -0.165127 \\ 0.778149 & 0.022167 & -0.243426 \\ 0.018490 & -0.297322 & -0.130926 \\ 0.051397 & 0.258108 & 0.091751 \\ 0.259423 & 0.848143 & 0.272091 \end{bmatrix}$$

Esta matriz tiene asociado el siguiente vector de media en 8 dimensiones.

$$\underline{\boldsymbol{\mu}} = \begin{bmatrix} 0.648541 & 0.537859 & 0.547184 & 0.526413 & 0.494256 & 0.405472 & 0.308410 & 0.279029 \end{bmatrix}$$

Resultados para k-medias

El objetivo del entrenamiento con el algoritmo de k-medias es obtener la matriz de centroides en 3 dimensiones que componen el alfabeto discreto utilizado por los HMM en la etapa de clasificación. Estos componen también los diferentes nodos del árbol k-d, utilizado como parte de la cadena de procesamiento del SiRPA.

Para fines de este proyecto se utiliza un alfabeto discreto compuesto por un total de 32 centroides, pues en un trabajo previo [82] se determinó que este valor es suficiente para realizar un reconocimiento adecuado, lo que se fundamenta en la reducción de la distorsión de los datos expuesta en [69]. Así, el resultado del entrenamiento será una matriz de 32 por 3.

La matriz obtenida se muestra en la tabla 4.51, donde se ha organizado de esta manera con el fin de mostrar el número de identificación del símbolo asociado a cada uno de los centroides. El entrenamiento se realizó utilizando todos los archivos, y con un muestreo de 1 para contar con un mayor número de muestras.

Con el fin de observar de manera gráfica la distribución de los centroides en 3 dimensiones se generó la figura 4.122, donde se muestran los puntos descritos en la tabla 4.51.

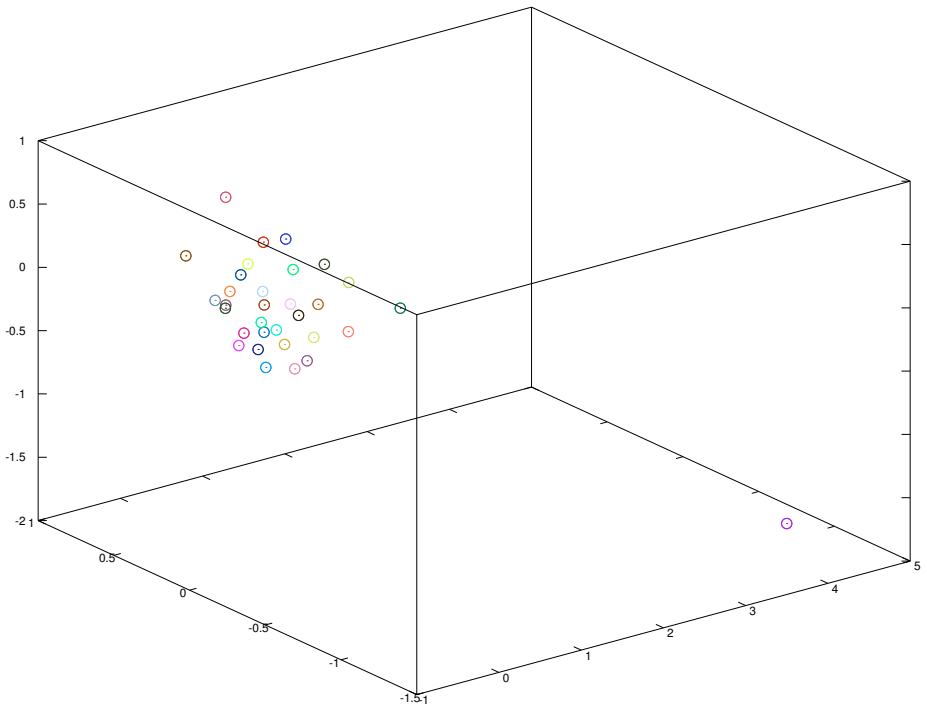


Figura 4.122: Distribución de los centroides generados en espacio de 3 dimensiones. Tomada de [55].

Tabla 4.51: Centroides que componen alfabeto discreto **V**. Tomada de [55].

ID	Coordenadas (x, y, z)
0	-0.200390, 0.095361, 0.297638
1	0.128159, 0.227361, 0.253334
2	0.182770, 0.006590, 0.561523
3	-0.470512, 0.048084, 0.132072
4	-0.407849, -0.004194, -0.169165
5	0.016712, -0.143028, -0.351365
6	0.015306, 0.314037, -0.124772
7	0.318917, 0.449405, -0.119228
8	0.281760, 0.203966, -0.300115
9	0.026294, 0.104181, -0.336463
10	-0.157564, 0.097897, -0.171364
11	0.353007, 0.068058, -0.015132
12	-0.125777, -0.152615, -0.129345
13	0.088305, -0.230108, -0.067904
14	-0.277249, -0.081856, 0.031854
15	-0.211896, -0.075756, -0.337535
16	0.267882, -0.088447, -0.362031
17	0.096316, 0.021522, -0.148992
18	0.027619, 0.581800, 0.139942
19	-0.243702, 0.241544, 0.022696
20	-0.025749, 0.035382, 0.062096
21	0.139876, -0.229726, 0.183623
22	-0.095523, -0.228818, 0.137479
23	-0.327123, -0.117944, 0.305483
24	4.123076, -1.160674, -1.734850
25	0.380117, -0.299191, -0.035840
26	0.550334, 0.353983, 0.281069
27	0.684771, -0.477256, 0.193790
28	0.316508, 0.476075, 0.609859
29	0.476441, -0.088568, 0.362999
30	0.260909, -0.366663, 0.410339
31	-0.133469, -0.213828, 0.497775

Resultados de clasificación

Se realizaron pruebas no exhaustivas (usando todas las cadenas disponibles para el entrenamiento, y variando parámetros en alrededor de 3-5 valores diferentes) con los algoritmos de entrenamiento múltiple y negativo, variando el número de estados, la longitud de cadena y el muestreo para verificar efectos sobre la tasa de reconocimiento de las cadenas.

Para cada una de las pruebas se obtiene la matriz de confusión, a partir de la cual se extraen los valores de sensitividad y el VPP para cada clase individual, que corresponden a los elementos c_{11}, c_{22} y c_{33} de la matriz **C**. Dado que los resultados son numerosos, se coloca la tabla con todos los valores en la sección de apéndices, mostrando en la tablas 4.52 y 4.53 solamente un resumen de los mejores valores de reconocimiento obtenidos en cada entrenamiento. Además, tal como se mencionó previamente, conforme se incrementa el valor del muestreo y/o la longitud de cadena, el número de archivos útiles para entrenar y evaluar disminuye, y de ahí que en algunos casos se indique con N.A. que no existe una cadena para evaluar esta clase. Si un valor es cero, quiere decir que no se clasificó ninguna cadena de una clase determinada correctamente.

Tabla 4.52: Tasas de reconocimiento obtenidas mediante entrenamiento múltiple. Tomada de [55].

			Sensitividad (%)			VPP (%)		
Estados	Muestreo	Long. Cadena	Bosque	Disparo	Motosierra	Bosque	Disparo	Motosierra
3	10	25	89,47	90	86,67	89,47	100	83,87
3	10	10	57,89	**94,44**	73,33	73,33	**94,44**	57,89
10	10	50	94,74	0	**93,33**	94,74	0	**87,5**
15	20	25	**97,37**	0	60	**74**	0	90

Tabla 4.53: Tasas de reconocimiento obtenidas mediante entrenamiento negativo. Tomada de [55].

			Sensitividad (%)			VPP (%)		
Estados	Muestreo	Long. Cadena	Bosque	Disparo	Motosierra	Bosque	Disparo	Motosierra
3	1	10	44,74	**4,76**	0	30,36	**3,23**	0
10	1	5	**97,37**	0	6,67	**46,84**	0	50
10	20	50	18,42	N.A.	**58,62**	36,84	N.A.	**62,96**

En la tabla 4.52 se observa que los resultados con entrenamiento múltiple dan valores de sensitividad y VPP por encima del 80% para las tres clases, aún cuando se utilizan pocos estados. Además, es posible obtener valores de sensitividad por encima del 93% para cada clase de manera individual, al variar los parámetros del modelo. Si se realizan pruebas exhaustivas es posible que se obtengan valores aún mayores a estos.

A partir de los resultados de la tabla 4.53 podría pensarse que el entrenamiento negativo tiene un rendimiento inferior, pero este no es el caso. Tal como se explica en [16], el entrenamiento negativo requiere de la utilización de cadenas de observación con longitudes cortas, pues de otra forma el entrenamiento agrega ruido a los modelos y se disminuyen las tasas de reconocimiento. Además, la selección de las cadenas de observación utilizadas para entrenar es más delicada que con el entrenamiento múltiple, y se requiere evaluar los resultados y agregar o quitar cadenas que los afecten, hasta obtener tasas satisfactorias, lo que aunado a la variación que debe realizarse de los otros parámetros de los modelos hace el proceso más trabajoso.

En la figura 4.123 se muestra un ejemplo de diagrama de transición de estados para las 3 clases involucradas, utilizando solamente 3 estados. Estos fueron obtenidos mediante entrenamiento múltiple, y tal como se presentó en la tabla 4.52, ofrece tasas de reconocimiento aceptables (mayores al 80% en las tres clases) con pocos estados, lo que en la implementación de ejecución es importante, ya que el tiempo de procesamiento aumenta conforme se incrementa el número de estados y la longitud de las cadenas de observación.

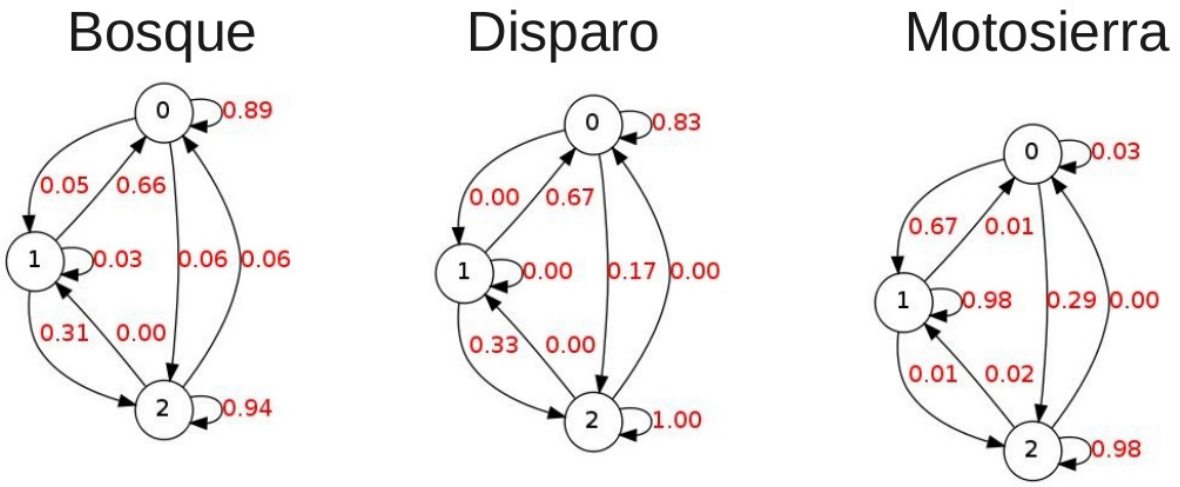


Figura 4.123: Ejemplos de HMM de 3 estados, entrenados con múltiples cadenas. Tomada de [55].

El número de estados, el muestreo y la longitud de las cadenas de observación no tienen que ser necesariamente igual entre los modelos, por lo que es posible utilizar combinaciones de ellos con el fin de maximizar las tasas de reconocimiento. Para ello, las tablas en los apéndices pueden dar una guía sencilla de los efectos de cada uno de estos valores sobre un modelo en específico. Sólo es importante evitar utilizar valores que provoquen tiempos de procesamiento excesivo, que harían que la implementación de ejecución llegue a fallar en algunos casos.

Resultados gráficos por etapa

En las siguientes figuras se presenta de manera gráfica el efecto de algunas de las etapas del SiRPA, con el fin de observar el efecto del normalizador, estimación de energía por banda y símbolos generados. Se presentan los resultados para los primeros 100 ms de audio para un disparo de un revólver calibre 32 a una distancia de 30 m, con angulación de 90° respecto al dispositivo de captura. Se toman las primeras 4411 muestras de audio, que producen en total 36 símbolos discretos. En la figura 4.124 se muestra la señal de entrada tal a como se lee del archivo (en la parte superior) y la señal resultante luego de la normalización (en la parte inferior).

Puede observarse la presencia de picos de gran amplitud (cercanos a ± 1) en los primeros

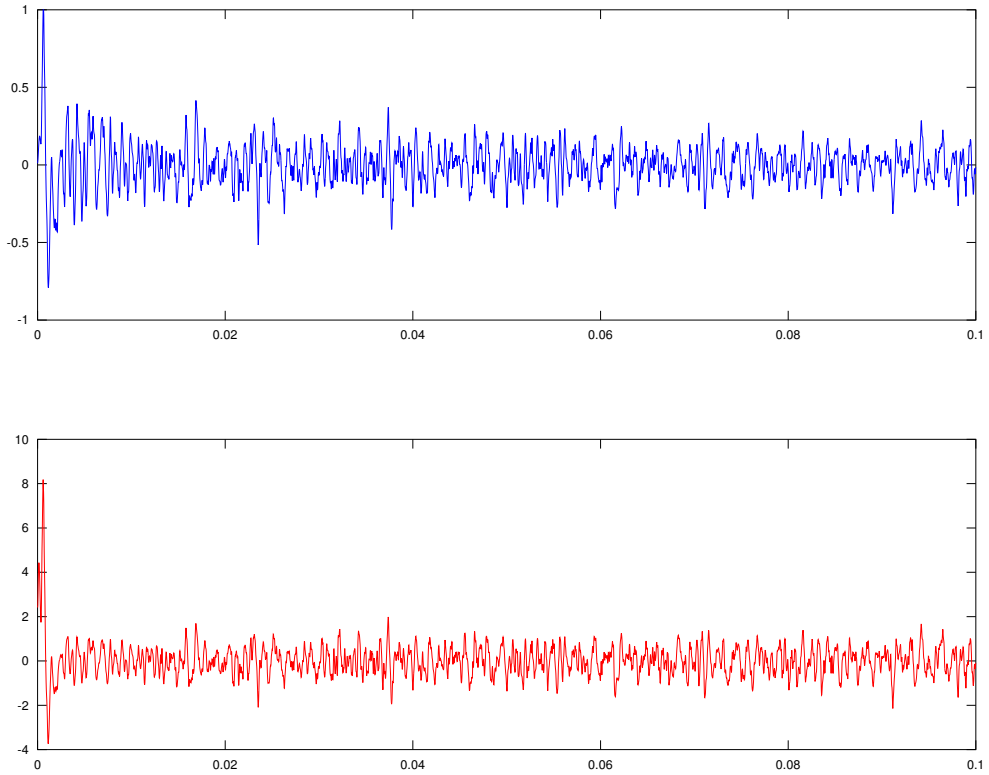


Figura 4.124: Señal de entrada (arriba) y señal normalizada resultante (abajo). Tomada de [55].

5 ms del gráfico de la entrada, que corresponde a la detonación del arma. Estos picos generan que la salida del normalizador se dispare, alcanzando en este caso valores 8 veces mayores a la máxima amplitud de la entrada. Posteriormente, la amplitud de la entrada baja, procediendo el normalizador a seguirla pero con una amplitud mayor. Una vez que desaparecen estos picos, la señal de entrada disminuye y se mantiene en niveles menores a $\pm 0{,}5$. Esto hace que la salida del normalizador se mantenga en niveles cercanos a ± 1, que serían los límites esperados cuando el medio se encuentra en un estado normal.

La presencia de los picos de gran amplitud a la salida del normalizador sería el comportamiento esperado cuando el medio se encuentra en un estado normal y se presenta súbitamente el disparo de un arma de fuego. Dado que el valor promedio en el estado normal sería aproximadamente constante, y de amplitud baja en comparación al pico para la detonación, la salida del normalizador se eleva y se mantiene en un valor alto hasta que el promediador reduzca su ganancia y por lo tanto la amplitud de salida.

La señal normalizada es luego pasada al banco de filtros, donde la figura 4.125 muestra las salidas obtenidas para los estimadores de energía que lo componen.

En esta figura puede notarse cómo, para este caso, las bandas 4, 5 y 6 son las que presentan

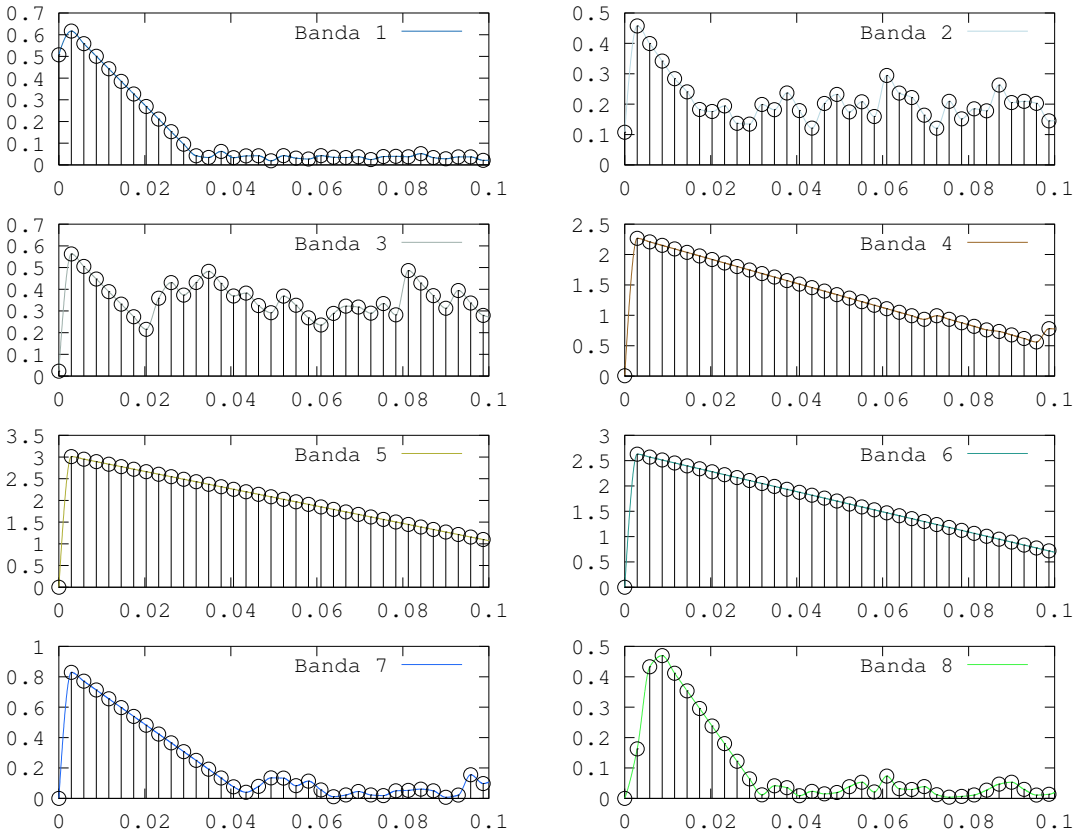


Figura 4.125: Salida del banco de filtros con estimación de energía por banda. Tomada de [55].

los mayores niveles de energía, alcanzando valores máximos en las primeras muestras y procediendo luego a descargarse de manera lineal, lo que corresponde al seguimiento de la envolvente. En cada gráfica se indican los puntos de muestreo en color negro, y superpuesta se tiene la señal continua interpolada a partir de estas muestras, utilizando interpolación *cubic* o *pchip*.

Los vectores de ocho dimensiones de los estimadores de energía son luego pasados por el reductor de dimensiones, y el vector de tres dimensiones resultante se utiliza para realizar la búsqueda en el árbol k-d. Esto permite determinar el centroide que se encuentra más cercano a este vector, y por lo tanto la salida del árbol sería el identificador para ese centroide, que corresponde también al símbolo de observación usado como entrada del módulo de clasificación. El tren de símbolos generado por los vectores mostrados en la figura 4.125 se muestra en la figura 4.126.

Estos símbolos serían los que se pasan posteriormente a la etapa de clasificación para realizar la evaluación, y podría aplicarse un muestreo adicional sobre ellos dependiendo de la configuración de los modelos. La distribución de los símbolos es similar para los otros ejemplos de disparos con los que se cuenta, pero varía de acuerdo a la distancia y

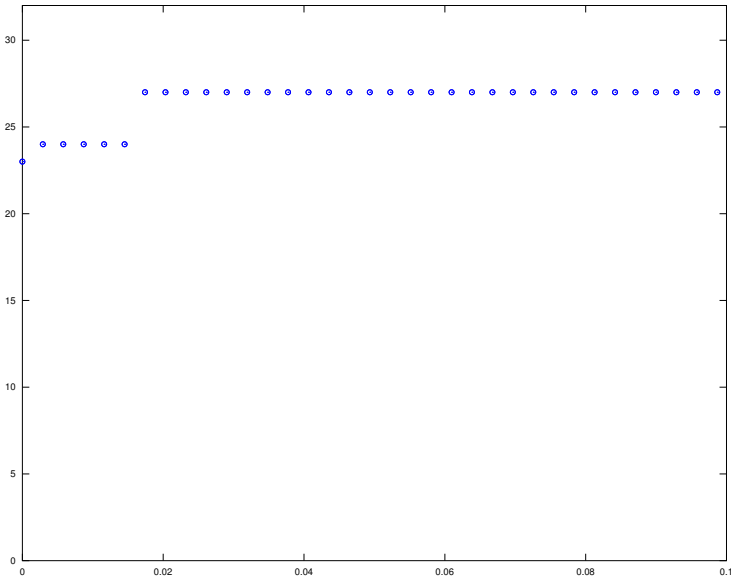


Figura 4.126: Tren de símbolos durante ventana de tiempo elegida. Tomada de [55].

la inclinación con respecto al dispositivo de captura [50].

Resultados gráficos para diferentes clases

Con el fin de observar la separación de los datos de diferentes clases que se obtiene luego de realizar la reducción de dimensiones, se presenta la figura 4.127. En esta se muestran puntos en tres dimensiones para las clases de disparo, motosierra y bosque, donde se utilizó un archivo de audio de ejemplo para cada una.

Puede notarse que los datos en tres dimensiones se separan espacialmente para vectores de diferentes clases, mientras que los vectores de la misma clase se encuentran agrupados. Este es justamente el efecto buscado, pues de esta forma las cadenas de observaciones generadas para cada clase facilitan la clasificación del patrón.

En la figura 4.127 se observa que la clase de disparo es la que presenta mayor variación en sus valores, lo cual se debe a que sus vectores de salida en ocho dimensiones del banco de filtros presentan altos valores de energía en diferentes bandas, tal como se observa en la figura 4.125. Conforme la salida de los estimadores de energía se descarga, los datos comienzan a "desplazarse" hacia la región de datos del bosque, que es el efecto buscado y para el cual se dimensionaron las constantes mostradas en la sección 4.4.3. De esta forma se captura el sonido de la detonación del disparo, se extraen sus características y se vuelve nuevamente a un estado normal del bosque.

La separación presente entre los datos de la clase de disparo con respecto a las otras dos es lo que permite que en las matrices de confusión esta clase muestre valores de sensitividad

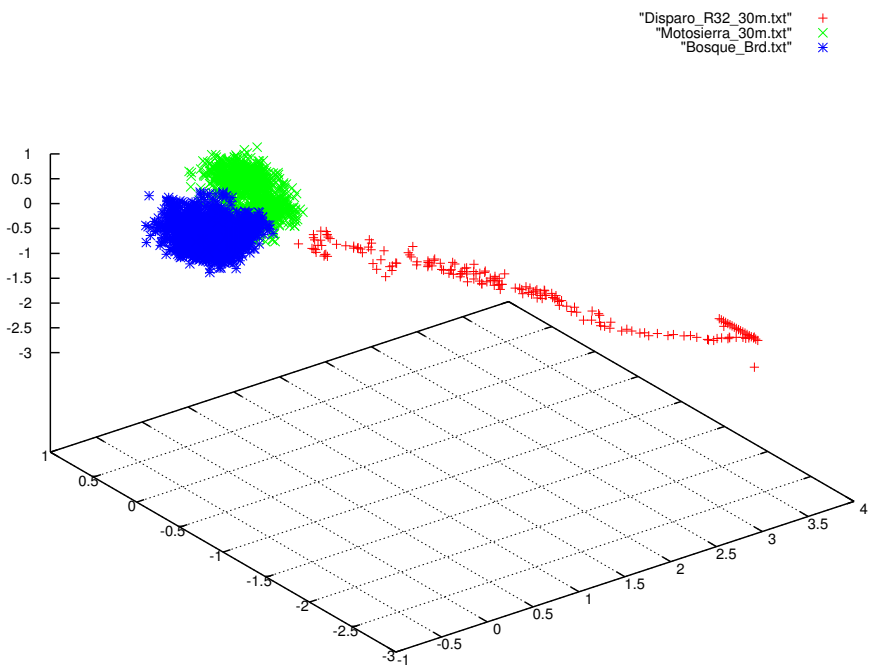


Figura 4.127: Datos en tres dimensiones para las diferentes clases. Tomada de [55].

por encima del 90%, y un VPP de inclusive el 100%. Sin embargo, conforme los sonidos capturados se ubican a una mayor distancia, los niveles de energía obtenidos en cada banda son más bajos y se da un traslape de los vectores de diferentes clases.

Este efecto es más notable entre las clases de bosque y motosierra, pues si bien en la figura 4.127 se encuentran debidamente separadas, la distancia entre los datos de ambas clases no es tan notable como para la clase de disparo, y conforme los sonidos se ubiquen más lejos del dispositivo de captura comienzan a traslaparse los datos entre ambas. Para ilustrar esto, se muestra en la figura 4.128 los datos obtenidos para sonidos de las clases de disparo y motosierra a 600 m de distancia.

Finalmente, se muestra en la figura 4.129 un ejemplo de tren de símbolos para las tres diferentes clases. Estos son los símbolos que componen las cadenas de observaciones que se pasan posteriormente al módulo de clasificación. Se muestran los primeros 100 símbolos (equivalente a aproximadamente 300 ms de audio) generados a partir de los datos que se mostraron en la figura 4.127, y utilizando los mismos colores para cada clase que se utilizaron en ella.

En esta figura puede notarse que el tren de símbolos para la clase de disparo (en rojo) es repetitivo, lo cual se debe a que sus datos, según se muestran en la figura 4.127 se ubican en su mayoría cerca de los centroides con el identificador 24 y 27, mostrados en la tabla 4.51. Los trenes para las clases de bosque y motosierra cambian repetidamente de valores, pues tal como se observa en la figura 4.127 sus datos se concentran en una región del espacio tridimensional donde se tienen múltiples centroides cerca, por lo que la

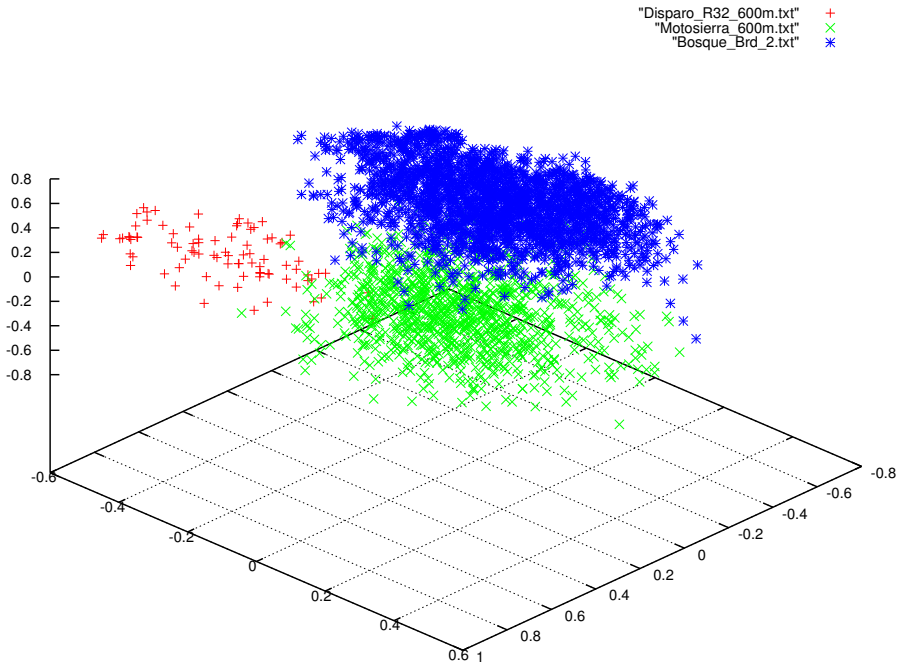


Figura 4.128: Datos en tres dimensiones con sonidos a mayor distancia. Tomada de [55].

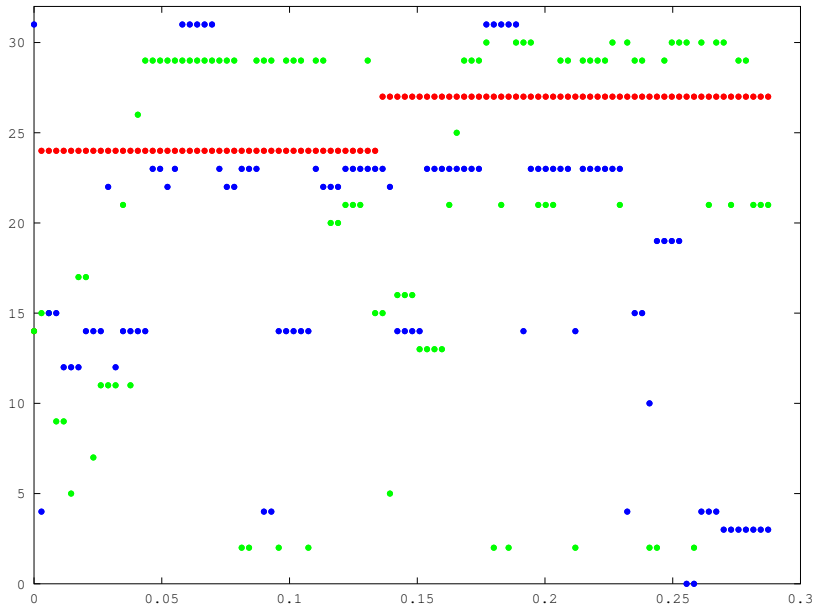


Figura 4.129: Ejemplo de tren de símbolos para las tres clases. Tomada de [55].

búsqueda del vecino más cercano no siempre da el mismo resultado.

A partir de esto es posible notar que se requiere de un mayor número de centroides para describir las clases de bosque y motosierra, dado que los vectores para estas se ubican en regiones concentradas del espacio, y de ahí que 31 de los 32 centroides mostrados en la tabla 4.51 se ubiquen dentro de un cubo delimitado en todas las dimensiones por -1 y 1,

con sólo un centroide fuera de esa región. Este último describe por lo general salidas de alta energía del banco de filtros, tal como son los sonidos de disparos.

Resultado de ordenamiento del árbol k-d

Tal como se mencionó en un capítulo previo, el árbol k-d de la implementación de entrenamiento previa se encuentra no balanceado. Dado que la aplicación para el sistema embebido opera en línea, el retraso adicional que se genera por esto es un factor crítico, y de ahí la inclusión de algoritmos para balancearlo. La construcción se hace en base a la división del espacio en la dimensión de mayor varianza, para que los algoritmos de búsqueda reduzcan así el número de comparaciones requeridas para obtener el símbolo de salida.

El resultado de esto se muestra de manera gráfica en la figura 4.130, donde es posible observar que el árbol se encuentra balanceado con un total de 6 niveles. El número en cada nodo corresponde al identificador, o símbolo, que le fue asignado en la tabla 4.51.

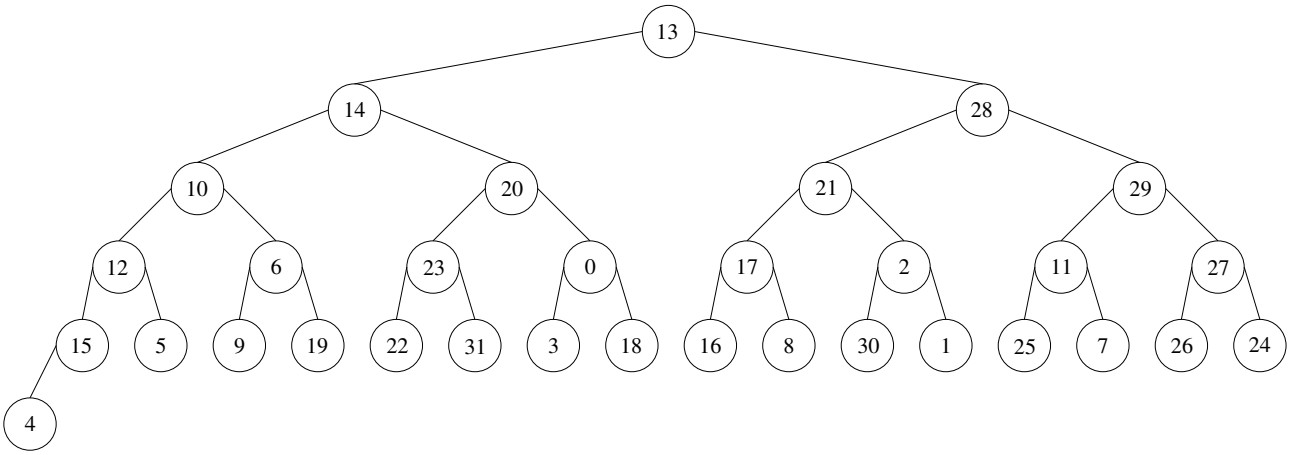


Figura 4.130: Ordenamiento resultante de balancear el árbol k-d. Tomada de [55].

Resultados de clasificación

Con el fin de evaluar los resultados de la etapa de clasificación, se exportaron dos juegos de modelos diferentes, obtenidos ambos utilizando entrenamiento múltiple. El primero consiste en modelos de 3 estados, muestreo de 10 y longitud de cadena de 25, mientras que el segundo son modelos de 10 estados, muestreo de 5 y longitud de cadena de 50. Se eligieron estos 2 juegos con el fin de evaluar tanto los efectos de la variación de los parámetros mencionados sobre el reconocimiento, como para determinar la variación en el tiempo de procesamiento requerido para obtener la salida en cada caso, pues los modelos del segundo juego requieren considerablemente más cálculos para realizar la evaluación.

Se utilizan las misma cadenas de evaluación que para la implementación de entrenamiento, y en base a los resultados de clasificación se construyen las matrices de confusión, cuyos

resultados se organizan en la tabla 4.54.

Tabla 4.54: Tasas de reconocimiento obtenidas mediante con sistema embebido. Tomada de [55].

			Sensitividad (%)			VPP (%)		
Estados	Muestreo	Long. Cadena	Bosque	Disparo	Motosierra	Bosque	Disparo	Motosierra
3	10	25	81,58	90	83,33	86,11	100	75,76
10	5	50	86,84	80	83,33	84,62	100	80,65

Alvarado10

Tiempos de ejecución por bloque

Se llevaron a cabo mediciones del tiempo de ejecución requerido para procesar cada bloque de señal, así como el tiempo que toma en promedio cada etapa para realizar sus operaciones. Dado que uno de los objetivos principales de la implementación en el sistema embebido es reducir estos tiempos para realizar la operación en línea de manera adecuada, varios de los bloques y sus funciones asociadas han sido optimizadas, verificando siempre que los resultados no se vean afectados.

Para fines de comparación se tienen tres aplicaciones diferentes; dos de ellas realizan el procesamiento utilizando solamente el procesador ARM, pero una es compilada utilizando gcc [67] y el entorno de desarrollo Eclipse [29], mientras que la otra se compila con un Makefile y herramientas propias de Texas Instruments. La tercera utiliza la combinación del procesador ARM y el DSP, ejecutando en este último todo lo relativo al banco de filtros y estimación de energía en cada banda. Se utilizan los mismos modelos de la subsección previa para verificar cambio en tiempos de procesamiento para cada uno, así como un tamaño de bloque de 256 muestras para estimar tiempos de procesamiento que se presentarán en la operación en línea.

La implementación inicial del sistema contenía errores estructurales heredados de la aplicación para entrenamiento, que fue desarrollada originalmente en un proyecto previo [16]. Estos errores fueron detallados en el capítulo anterior. Sin embargo, las mediciones de tiempo realizadas se llevaron a cabo conforme se corregían y optimizaban diferentes partes del sistema, lo que permite observar la mejora en los tiempos de procesamiento conforme se finalizaba la aplicación.

En la tabla 4.55 se muestran los resultados de tiempo de procesamiento promedio obtenidos para cada una de las aplicaciones previamente mencionadas. En cada nueva fila se específica cuál era la condición de la implementación del sistema, para verificar así el efecto de las correcciones y optimizaciones realizadas. En orden se tendrían las siguientes condiciones: *Inicial*, donde se tenían las tasas de muestreo incorrectas para el banco de filtros, y no se tomaba la salida en los estimadores de energía. *Corrección 1* donde se atacó este problema y a su vez se llevó a cabo la implementación de los algoritmos de

filtrado en el DSP. *Corrección 2* donde se obtuvo un árbol k-d balanceado, y por último la implementación *Final*, donde se realizaron últimas modificaciones al código en diferentes bloques para mejorar los tiempos de procesamiento.

Tabla 4.55: Tiempos de procesamiento para aplicaciones en sistema embebido. Tomada de [55].

	Tiempo por tipo de aplicación (ms)		
Estado de implementación	ARM (GCC)	ARM (TI)	ARM + DSP
Inicial	4,4768	1,5595	1,7912
Corrección 1	1,2771	0,4274	0,5492
Corrección 2	1,0997	0,3142	0,4576
Final	0,8851	0,3051	0,3660

Al realizar mediciones de tiempo etapa por etapa, se observa que las implementaciones compiladas con herramientas de TI tienen tiempos similares en todas ellas excepto en el banco de filtros, que toma en promedio 80 μs más para obtener la salida. Este tiempo viene asociado al *overhead* generado al llamar a la función de filtrado que se ejecuta en el DSP, ya que debe transferirse el control de los datos a éste y esperar posteriormente los resultados. Esto, junto al tiempo que toma realizar la conversión de memoria virtual a física y viceversa, provoca que el tiempo de procesamiento se vea afectado. Si el tamaño de bloque utilizado fuera mayor, estos efectos podrían verse mitigados, al realizar menos llamadas a esta función.

Para la medición del tiempo de procesamiento de la etapa de clasificación se utilizó la aplicación ARM compilada con las herramientas de TI. En el caso del modelo de 3 estados, el tiempo de procesamiento para el bloque donde se alcanza el número de muestras requeridas para realizar la evaluación es de 549 μs, por lo que tomando como referencia el tiempo de procesamiento de bloque de 305 μs indica que la evaluación para los 3 modelos toma alrededor de 250 μs. Por otro lado, para el modelo de 10 estados se obtiene un tiempo de 1,312 ms, por lo que en este caso la evaluación toma un poco más de 1 ms, lo que es de esperar dado que con este modelo se tienen matrices más grandes, así como una cadena de observación el doble de larga.

Estos resultados muestran que el prototipo de software del SiRPA implementado en el sistema embebido permite realizar pruebas con modelos de un tamaño considerable, manteniéndose por debajo del tiempo límite para el procesamiento en línea que es de aproximadamente 5,3 ms. Esto permite estudiar sin problema las tasas de reconocimiento para modelos de gran tamaño. Solamente debe garantizarse que no se exceda el tiempo indicado, pues conllevaría a que no se atiendan las respuestas a tiempo en el cliente de JACK, provocando que la aplicación falle.

4.4.4 Conclusiones parciales sobre un casificador digital para la clasificación de patrones acústicos de disparos y motosierras en un sistema embebido

Se ha conseguido integrar todas las fases de procesamiento del SiRPA en un sistema de bajos recursos. Esta integración permite para procesar cadenas de observaciones en tiempo real.

La aplicación de entrenamiento desarrollada permite obtener de manera correcta la matriz de transformación y el vector de media para el reductor de dimensiones. Además, permite el cálculo adecuado de los 32 centroides que describen el alfabeto discreto para los HMM.

Los mejores valores se sensitividad obtenidos con entrenamiento múltiple fueron de 89,47%, 90% y 86,67%, para las clases de bosque, disparo y motosierra, respectivamente. Estas se obtuvieron con modelos de 3 estados, muestreo de 10 y longitud de cadena de 25. Sin embargo, debe notarse que podrían existir valores de estos parámetros que lleven a tasas de reconocimiento mayores, para lo que es necesario hacer un mayor número de pruebas.

Los parámetros de número de estados, muestreo y longitud de cadena afectan de diferente manera las tasas de reconocimiento de los modelos. Las tasas suelen mejorar para todas las clases conforme se incrementa la longitud de la cadena, hasta que se alcanza un punto máximo alrededor de longitudes de 15 o 20 símbolos, a partir de donde un incremento beneficia el reconocimiento para clases como la de bosque, y en ocasiones la de motosierra, pero suele afectar negativamente a la de disparo.

El muestreo también influye sobre las tasas de reconocimiento, pues conforme se incrementa su valor se mejoran estas tasas para las tres clases, hasta un punto máximo alrededor de valores de muestreo de 5 o 10. Luego de este último valor las tasas de reconocimiento disminuyen para las tres clases. Esto es de esperar dado que con valores superiores a 10 se descartan símbolos que aportan información característica de los patrones, además de que se reduce el número de cadenas útiles para entrenar y evaluar dependiendo también de la longitud de cadena usada. Así, en futuros entrenamientos el valor de muestreo debe ser igual o menor a 10.

Las tasas de reconocimiento para las tres clases se incrementan conforme se aumenta el número de estados, hasta alcanzar valores máximos para modelos con 10 estados.

El entrenamiento negativo dio tasas de reconocimiento bajas, pero los procedimientos de entrenamiento y evaluación realizados no fueron exhaustivos. Este entrenamiento requiere de una selección delicada de las cadenas de observación utilizadas y de los parámetros de longitud de cadena, muestreo, número de estados y razón de aprendizaje. Si todos estos no se seleccionan con cuidado se obtendrán tasas como las mostradas, por lo que los resultados obtenidos en este caso no son concluyentes respecto a la efectividad de esta variante de entrenamiento.

La exportación de los modelos pudo realizarse correctamente, dando por resultado las mismas probabilidades al realizar la evaluación en el sistema embebido con respecto a la aplicación de entrenamiento. Los modelos pudieron ser cargados por el sistema embebido

sin ningún tipo de error, y sin importar el número de estados, longitud de cadena y muestreo elegidos. No existen además limitaciones en cuanto al número de modelos que se pueden cargar desde un mismo archivo. Sin embargo, lo ideal es exportar sólo uno de cada clase.

Los tiempos de procesamiento obtenidos en la implementación final del sistema son de apenas 305 μs y 366 μs, al utilizar sólo el procesador ARM o la combinación ARM + DSP, respectivamente. Este es el tiempo que toma procesar un bloque completo de 256 muestras hasta la salida de la etapa de generación de símbolos. Para la etapa de clasificación el tiempo de procesamiento varía de acuerdo a los parámetros elegidos para cada HMM, donde se obtuvieron tiempos de 250 μs para evaluar modelos de 3 estados con longitud de cadena de 25, y 1 ms al evaluar modelos de 10 estados y longitud de cadena de 50.

Las tasas de reconocimiento obtenidas por el sistema embebido son las mismas que para la implementación en el computador de propósito general, donde las diferencias surgen a partir de que la primera permite evaluar la totalidad de la cadena de observación generada a partir de los archivos de ejemplo al desplazar una ventana sobre las muestras, mientras que la última sólo considera las primeras T muestras adquiridas. Esto provoca que al evaluar en una ventana diferente a la inicial exista la posibilidad de que se den cambios en la clasificación en algunos casos.

4.5 Implementación en RTL-FPGA del SiRPA

A la hora de generar este informe, todavía se encuentra en proceso de pruebas el código RTL, con miras a su envío a fabricación en un proceso CMOS de 130nm a través de MOSIS, para el 18 de agosto del 2014. Los resultados mostrados en esta sección son por tanto parciales y sujetas a modificaciones. No se muestran los detalles tampoco de la generación de los archivos para dicha fabricación en las herramientas EDA de la Escuela, pues esta etapa se halla también en proceso.

Una vez lista la implementación en sistema embebido desarrollada en la sección anterior 4.4, se contó ya con un patrón de comparación para evaluar la descripción del SiRPA a nivel de RTL. Se persigue en esta sección la implementación primero a nivel de FPGA, para tener luego una representación válida que pueda sintetizarse finalmente en un ASIC digital. Esto con miras a cumplir con el objetivo 4 del proyecto. La Fig. 4.131 resume el mismo diagrama de bloques ya visto, pero ahora visto desde la perspectiva de un circuito independiente. El proceso de diseño y prueba de esta etapa del proyecto puede verse con más detalle en [34, 10].

En el mismo, se muestran las señales de entrada:

- **audio_in:** es la señal digital de entrada del sistema, es decir los datos digitales que representan el audio que se va a analizar. Para representar a los mismos, se emplean dieciséis bits.
- **clk:** la señal de reloj para el SiRPA.

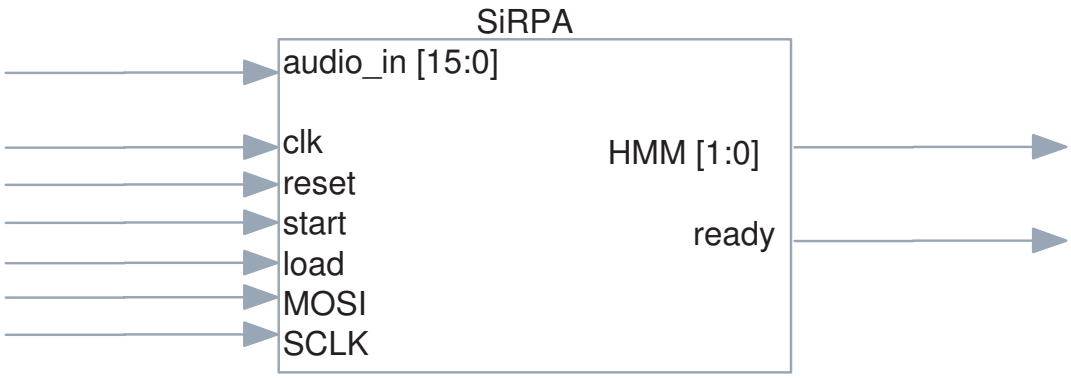


Figura 4.131: Diagrama de entradas y salidas del SiRPA. Tomada de [34].

- **reset:** es la señal de reinicio del sistema,a ctiva en alto.
- **load:** con este puerto de entrada se le indica al sistema que se desea realizar una carga de coeficientes, es decir se va a reconfigurar los coeficientes del SiRPA.
- **MOSI:** es la salida de datos del módulo maestro del protocolo serial (SPI) y la entrada de los datos al SiRPA. Los datos que se introducen al SiRPA corresponden a los coeficientes del sistema.
- **SCLK:** es la señal de reloj de los datos del protocolo serial (SPI).

Las señales de salidas del sistema:

- **HMM:** con esta señal de dos bits se indica que la muestra de audio representa con mayor probabilidad el bosque (con un cero), una motosierra (con un uno) o un disparo (con un dos).
- **ready:** cuando el sistema ha concluido de analizar el audio y ya identificó cual modelo es mejor representado por la muestra de audio, lo indica mediante esta señal al colocar en un valor alto.

Para poder llevar a cabo el procedimiento completo del SiRPA, éste se descompuso en distintos bloques, que ejecutan cada una de las funciones descritas en la Fig. 4.109. En la Fig. 4.132, se muestra el diagrama de bloques interno del sistema.

Las funciones de todas estas unidades ya han sido exhaustivamente explicadas en la sección 4.4, por lo que solo se describen someramente aquí, añadiendo las modificaciones necesarias para una construcción hardware concurrente. En la figura 4.132, se muestra el módulo denominado filtro segmentado [77], cuya función principal es dividir la señal de entrada (el audio proveniente del exterior) en ocho bandas de frecuencia, para que las mismas sean procesadas por el módulo reductor de dimensiones. Esta sección fue ampliamente modificada para corregir errores de sincronía, de resolución y de desborde que afectaban los resultados, pero la estructura funcional es la misma documentada en [77, 55].

En las siguientes subsecciones se detalla el desarrollo de los módulos de hardware correspondientes al reductor de dimensiones, el árbol de comparadores, la unidad de modelos

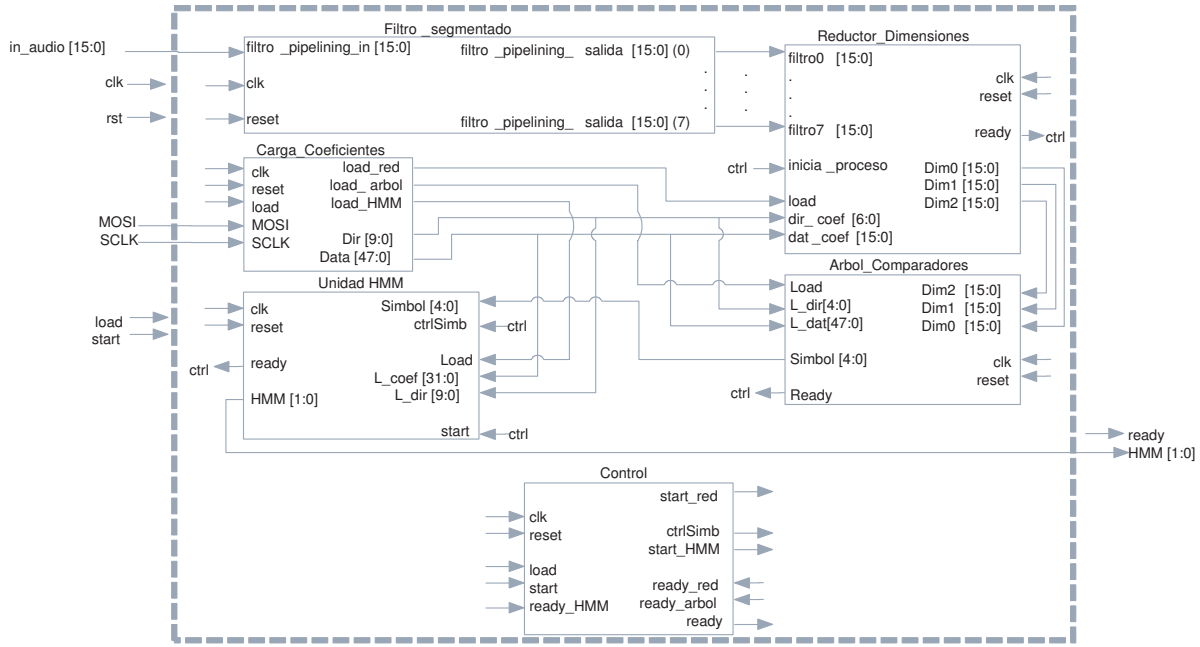


Figura 4.132: Diagrama de segundo nivel del SiRPA. Tomada de [34]

ocultos de markov (unidad HMM), el control y el sistema de carga de coeficientes.

4.5.1 Diseño del Reductor de Dimensiones

El módulo de reducción de dimensiones realiza una conversión de un espacio de entrada de ocho bandas de frecuencia a uno de salida de tres. Esta transformación se lleva a cabo a partir de una combinación lineal de los parámetros de entrada con los coeficientes del sistema: los coeficientes son calculados por un software de entrenamiento [16, 55]. Todos estos datos se encuentran en formato punto fijo.

El diagrama de bloques del módulo reductor de dimensiones se muestra en la Fig. 4.133.

El bloque reductor de dimensiones recibe las señales de **filtro0** hasta **filtro7**, las cuales son la salida del banco de filtros. La señal **start** indica al bloque cuándo debe iniciar el proceso de reducción de dimensiones. La entrada **load** activa el módulo de carga de coeficientes, con lo que los datos que están en la señal **coeficiente** se pasan al banco de registros en la dirección dada por **dir_coef**.

Las salidas son el resultado de la reducción de dimensiones y están identificadas como **Dim0**, **Dim1** y **Dim2**. La señal **ready** en alto indica el estado del sistema, es decir que el módulo se encuentra en estado de inactividad y puede realizar la reducción de dimensiones o la carga de coeficientes del exterior, o que está procesando una reducción de dimensiones.

Después de un reinicio, el reductor procede a cargar la matriz de coeficientes, definida como **W**, en el banco de registros (bloque llamado ***"Registers"***), a través del ***"Inicia_coef"***. Los números y nombres de los registros se muestran en la tabla 4.56.

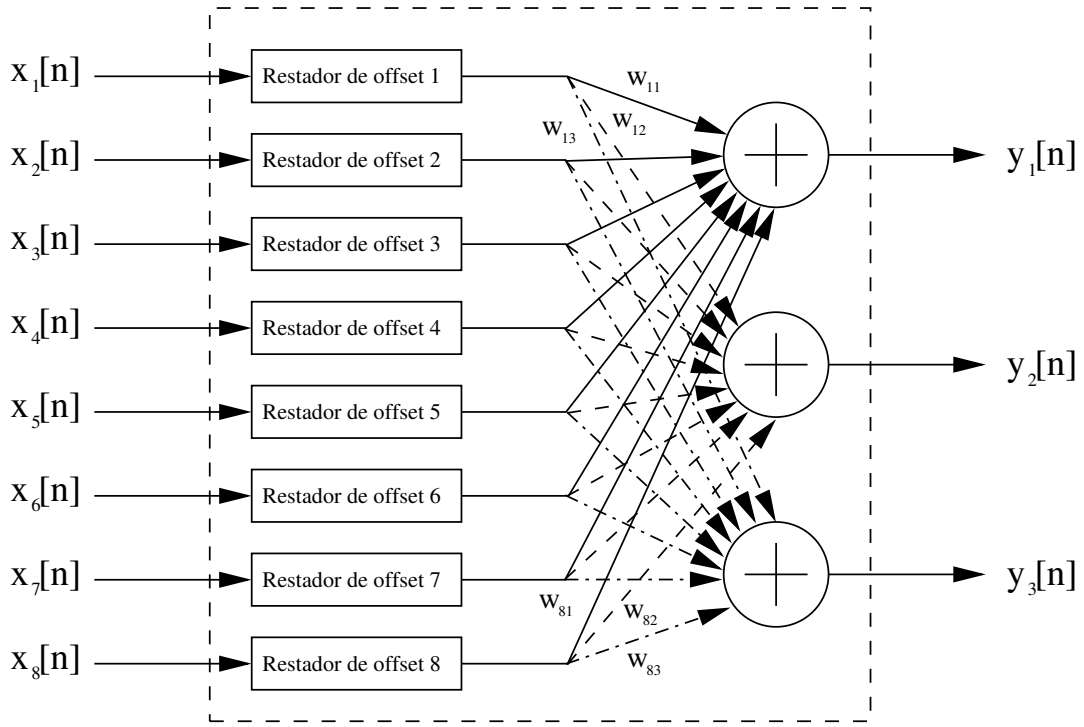


Figura 4.133: Diagrama del reductor de dimensiones. Tomada de [34].

Dirección	Registro	Dirección	Registro	Dirección	Registro
0	Offset entrada 1	13	coeficiente w_{23}	26	coeficiente w_{71}
1	Offset entrada 2	14	coeficiente w_{31}	27	coeficiente w_{72}
2	Offset entrada 3	15	coeficiente w_{32}	28	coeficiente w_{73}
3	Offset entrada 4	16	coeficiente w_{33}	29	coeficiente w_{81}
4	Offset entrada 5	17	coeficiente w_{41}	30	coeficiente w_{82}
5	Offset entrada 6	18	coeficiente w_{42}	31	coeficiente w_{83}
6	Offset entrada 7	19	coeficiente w_{43}	32	Temporal 0
7	Offset entrada 8	20	coeficiente w_{51}	33	Temporal 1
8	coeficiente w_{11}	21	coeficiente w_{52}	34	Dimensión 2
9	coeficiente w_{12}	22	coeficiente w_{53}	35	Dimensión 1
10	coeficiente w_{13}	23	coeficiente w_{61}	36	Dimensión 0
11	coeficiente w_{21}	24	coeficiente w_{62}		
12	coeficiente w_{22}	25	coeficiente w_{63}		

Tabla 4.56: Banco de registros del reductor de dimensiones. Tomada de [34].

Una vez que se concluye con la inicialización, el sistema se encuentra listo para realizar una carga externa de coeficientes, o un proceso de reducción de dimensiones.

Si se inicia el proceso de reducción de dimensiones, el sistema selecciona la primera señal de entrada, le resta el valor de *offset 1* y lo almacena en el registro Temporal 0, posteriormente, multiplica el contenido del registro Temporal 0 con el coeficiente correspondiente (en este caso el w_{11}) y lo almacena en el Temporal 1. Luego, selecciona la entrada dos

Coordenada "X"	Coordenada "Y"	Coordenada "Z"
47 ··· 32	31 ··· 16	15 ··· 0

Tabla 4.57: Formato del vector de datos del árbol de clasificación. Tomada de [34].

y le resta el *offset 2* y lo almacena en el Temporal 0. Después, multiplica el Temporal 0 con el coeficiente w_{21} y el resultado lo suma al Temporal 0, para finalmente almacenar la suma en el mismo Temporal 0. Este proceso se repite hasta que se hayan abarcado las ocho entradas; al final no se almacena la suma en el Temporal 0, sino que se almacena en el registro de **Dim0**.

El proceso descrito anteriormente se lleva a cabo de manera similar para calcular las salidas de **Dim1** y **Dim2**, pero se emplean los coeficientes particulares.

Si se procede a realizar una carga de coeficientes externa para reconfigurar el sistema, la entrada **load** se debe colocar en alto: así, el sistema habilita la escritura en el banco de registros y procede a escribir los datos que se tienen en la señal **coeficiente** en la dirección **dir_coef**.

4.5.2 Diseño del Árbol de Clasificación / Generador de Símbolos

La función del bloque árbol de clasificación (o generador de símbolos), es determinar dentro de treinta y dos puntos ya definidos (determinados por el software de entrenamiento [16]), cuál es el que se encuentra más cerca al punto de entrada. Es decir, se calcula la distancia tipo L1 entre el punto de entrada y cada uno de los treinta y dos nodos, para poder definir cuál de los nodos se encuentra a menor distancia.

Para realizar la función antes mencionada, se diseñó el sistema que se muestra en la figura 4.134.

Al igual que el módulo reductor de dimensiones, al iniciar el funcionamiento, el sistema realiza una carga de coeficientes al banco de registros, por medio del módulo denominado ***Inicializacion_Nodo***, que se encarga de enviar los datos a los treinta y dos nodos (llamados ***Nodo***). Los datos se encuentran organizados en un arreglo de cuarenta y siete bits en formato punto fijo, en donde los dieciséis primeros son la coordenada "X", los siguientes dieciséis la coordenada "Y" y finalmente los dieciséis menos significativos son los correspondientes a la coordenada "Z", tal como se muestra en 4.57.

Una vez terminada la inicialización, el módulo indica que se encuentra listo para cargar coeficientes del exterior o ejecutar el proceso de cálculo del símbolo (que en realidad es la identificación del nodo que se encuentra más cercano al punto de entrada, o sea, el resultado del funcionamiento del módulo es un número del cero al treinta y uno). Este estado lo indica la unidad al colocar en alto la señal **ready**.

Para iniciar la carga de coeficientes, se coloca en estado alto la señal **load**, con lo que el sistema permite la escritura en uno de los treinta y dos nodos; los datos que se encuentran

en **load_coef** se cargan en el nodo que se indica en la dirección de la señal **load_dir**. Además, se coloca en estado bajo la señal **ready**, con lo que se indica que se esta realizando la carga, una ve finalizada la operación, la señal vuelve al estado alto.

En el caso del proceso de cálculo del nodo más cercano, el sistema funciona de manera combinacional. Se colocan los datos válidos en las entradas, denominadas como **in_X**, **in_Y** y **in_Z**, y una vez que se hayan propagado las señales correspondientes, se encontrará una salida válida en el sistema, que indica cuál nodo es el más cercano a la entrada.

Para realizar el cálculo del nodo mas cercano, primeramente se calcula la distancia Manhattan (o distancia L1) entre el punto de entrada y los nodos del sistema, a través de los módulos **Cálculo_L1**. Una vez que se tiene la distancia a cada nodo, se realizan comparación en parejas; por ejemplo, se comparan las distancias del **Nodo_0** y la del **Nodo_1**, las del **Nodo_2** y la del **Nodo_3** y así de manera sucesiva. Del primer nivel de comparación, resultan dieciséis distancias seleccionadas, las cuales pasan al siguiente nivel de comparación.

El segundo nivel de comparación funciona de igual manera, en este caso se operan comparaciones sobre dieciséis distancias, para dar como resultado ocho distancias. Posteriormente, se continúa con un tercer, cuarto y quinto nivel de comparación, lo que finalmente permite identificar cual es el nodo más cercano al punto de inicio. Finalmente, con las comparaciones y un conjunto de módulos de multiplexación (denominados **Mux 2x1** en el diagrama de la figura 4.134) se produce a la salida el número que identifica el nodo más cercano a la entrada. Las señales de **Dir0** hasta **Dir4** son las salidas del sistema.

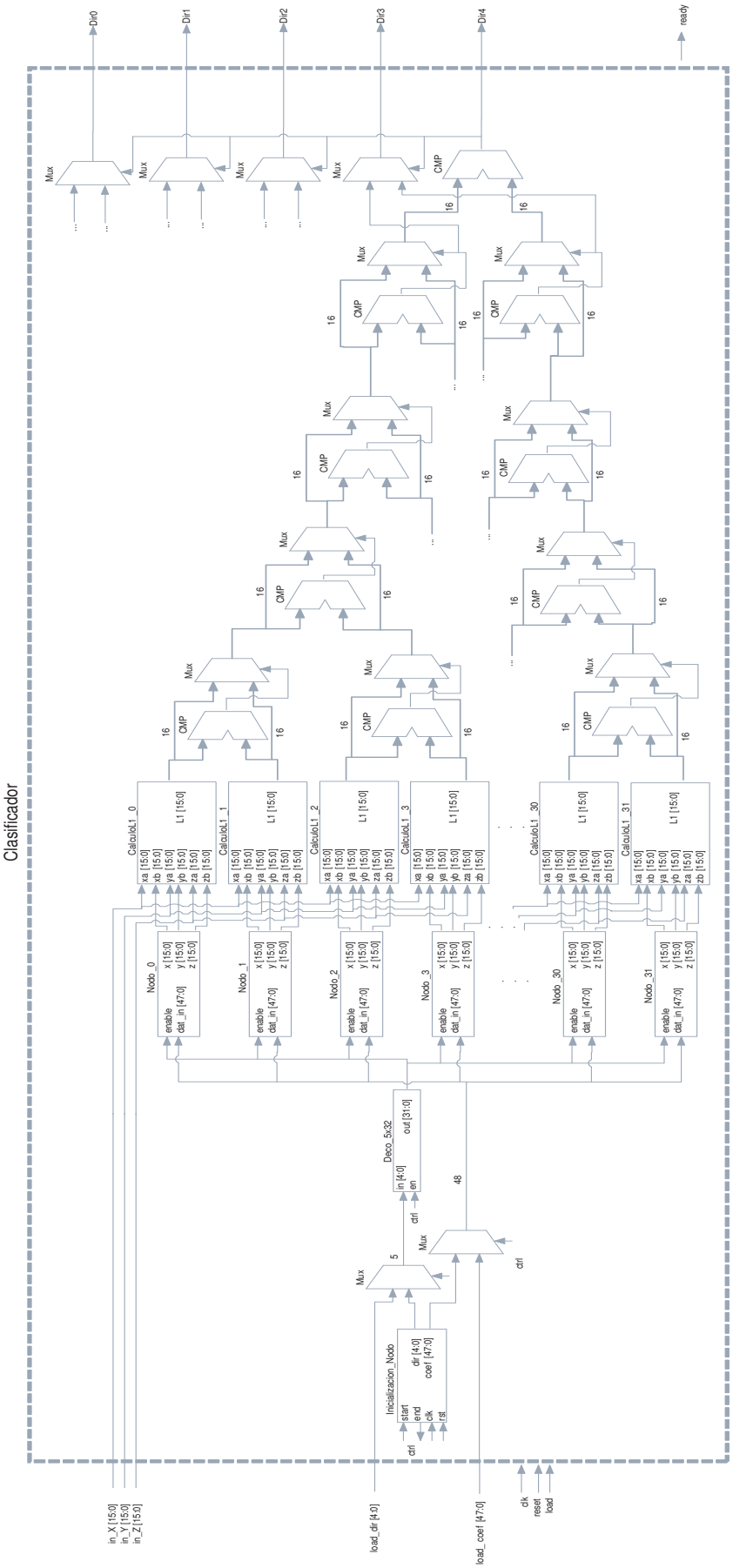


Figura 4.134: Diagrama del árbol de clasificación. Tomada de [34].

4.5.3 Diseño de la Unidad de Modelos Ocultos de Markov (HMMU)

El sistema de salida del SiRPA ejecuta el procedimiento descrito en la sección , es decir el algoritmo *Forward-Backward Procedure*, delimitado por los siguientes parámetros: veinte símbolos de largo de cadena, cinco estados para cada modelo oculto de Markov y un alfabeto de treinta y dos símbolos. Para poder llevar a cabo el procedimiento se diseñó el módulo de la figura 4.136.

Para controlar la unidad, se implementó una máquina de estados con el diagrama de la figura 4.135.

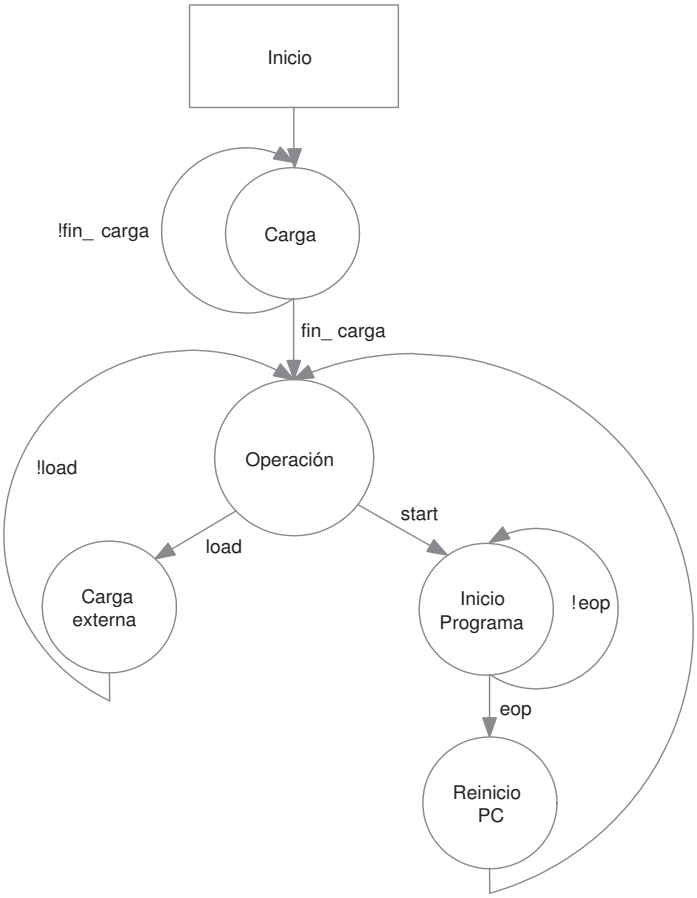


Figura 4.135: Máquina de estados del control del HMMU. Tomada de [34].

Después de un reinicio, el sistema procede a cargar los coeficientes por defecto en la memoria, a través del módulo **Inicia_coef**. Los datos se encuentran distribuidos según se indica en la tabla 4.58.

Antes de iniciar el procedimiento de cálculo, se deben cargar los veinte símbolos en el módulo denominado **Simbol_Reg**. El control general del SiRPA se encarga de cargar individualmente cada símbolo generado en el registro mediante la señal **ctrlSimb** y de dar la señal de inicio (*start* en la figura 4.135). Por otra parte, se puede proceder a realizar una carga externa de coeficientes al banco de memoria, mediante la activación de la señal **load**.

Dirección de Memoria	Matriz	Modelo
0 ⋮ 4	Inicialización	Bosque
5 ⋮ 29	Transición	
30 ⋮ 189	Observación	
190 ⋮ 194	Inicialización	Motosierras
195 ⋮ 219	Transición	
220 ⋮ 379	Observación	
380 ⋮ 384	Inicialización	Disparos
385 ⋮ 409	Transición	
410 ⋮ 569	Observación	

Tabla 4.58: Direcciones de memoria de los coeficientes del HMMU. Tomada de [34].

Dirección de Memoria	Registros
570	temporal 0
⋮	⋮
585	temporal 15
586	acumulador 0
587	acumulador 1
588	acumulador 2
589	zero

Tabla 4.59: Registros de temporales del HMMU. Tomada de [34].

Si se inicia el proceso del algoritmo *Forward-Backward Procedure*, el sistema recorre la memoria de programa (**Rom_Mem**) hasta que se alcanza la instrucción "eop" (su estructura y funcionamiento se detallan más adelante) con lo que la máquina se detiene y regresa al estado de Operación.

Durante la operación de la unidad, además de los registros con las matrices de coeficientes de cada modelo oculto de Markov, se emplean registros para los cálculos intermedios del proceso. La dirección de memoria de los mismos, así como los nombres de los mismos se encuentran en la tabla 4.59.

El conjunto de instrucciones que ejecuta el módulo se encuentra descrito en la tabla 4.60. La función de cada una se explica a continuación:

- **nope:** es la instrucción que se emplea para que el sistema no realice ninguna acción.
- **add:** la instrucción realiza una suma del registro A con el registro B y lo almacena en el registro destino.
- **addi:** suma el contenido del registro A con el valor del inmediato, para almacenarlo en el registro destino.
- **sub:** al contenido del registro A le resta el registro B y lo almacena en el registro destino.
- **mult:** multiplica el contenido del registro A y el registro B y lo almacena en el registro destino.
- **div:** divide el registro A entre el registro B, el resultado lo almacena en el registro destino.
- **push:** esta instrucción toma los datos del registro indicado en A y los almacena en un registro para la comparación (módulo llamado **Comp_HMM**). Unicamente los registros acumulador 0, acumulador 1 o acumulador 2 son válidos para la comparación, en los mismos se almacenan las probabilidades de que la cadena de símbolos represente al modelo del Bosque, Motosierras o Disparos según corresponde, y por ende sólo éstos registros se debe usar como parámetros (el módulo de **Ctlr_HMM major** se encarga del control).
- **eop:** con esta instrucción se indica al sistema que se encuentra en el final del programa, y por ende se detiene el conteo del PC (*program counter*).

Ins	Composición						
nope	Datos sin efecto						
add	Reg A	Reg B	Reg Dest	Innecesario	Simbolo	Fila	Dirección
addi	Reg A	Innecesario	Reg Dest	Inmediato	Simbolo	Fila	Dirección
sub	Reg A	Reg B	Reg Dest	Innecesario	Simbolo	Fila	Dirección
mult	Reg A	Reg B	Reg Dest	Innecesario	Simbolo	Fila	Dirección
div	Reg A	Reg B	Reg Dest	Innecesario	Simbolo	Fila	Dirección
push	Reg A	Innecesario	Innecesario	Innecesario	Simbolo	Fila	Dirección
eop	Datos sin efecto						
Cantidad de bits de cada elemento de la instrucción							
3	10	10	10	32	5	3	2

Tabla 4.60: Conjunto de instrucciones del HMMU. Tomada de [34].

Cada instrucción tiene los parámetros: **Simbolo**, **Fila** y **Dirección**, los cuales determinan la dirección de memoria que se lee en el puerto cero del banco de memoria durante la ejecución de cada instrucción. La señal **Simbolo** se encarga de seleccionar cuál de los símbolos de entrada se pasa al módulo **Dir_Deco**; la combinación del símbolo con **Fila** produce las direcciones de memoria del coeficiente para los tres modelos ocultos. Finalmente, según se indique en el parámetro **Dirección** se pasa la dirección correspondiente al parámetro "registro A" de la instrucción, el modelo del Bosque, Motosierra o Disparo.

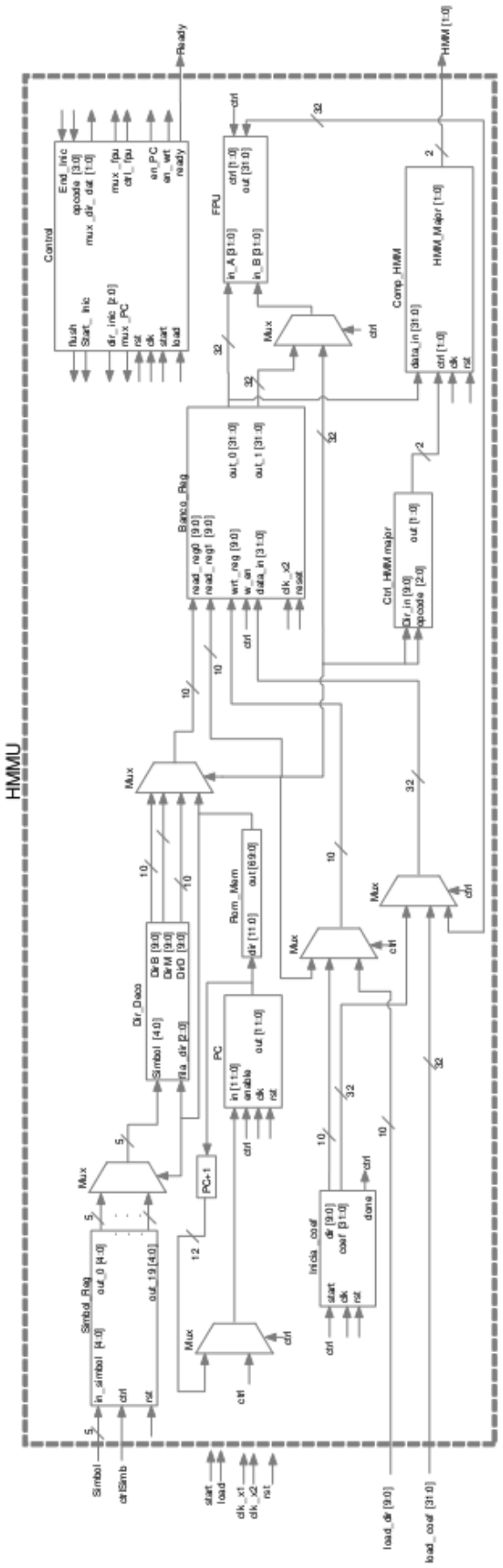


Figura 4.136: Diagrama del HMMU. Tomada de [34].

4.5.4 Desarrollo de pruebas de banco en FPGA de las unidades del SiRPA

Interfaz de recepción y envío de datos para pruebas de banco en FPGA de las unidades

La implementación final del SiRPA se muestra en la figura 4.132. En la etapa de preproceso la señal es captada por el micrófono; normalizada por el AGC y digitalizada por el ADC, tal como se explicó en la sección 4.1. Las restantes etapas ya han sido descritas exhaustivamente. Cada etapa se trasladó a una FPGA embebida en una tarjeta de desarrollo Nexys de Digilent [19]. Para más detalles del desarrollo y análisis aquí resumido, ir a [10].

Controlador SPI para ADC

Como parte de la comunicación entre el ADC y el resto de los módulos del SiRPA, se escribió en Verilog HDL un controlador SPI genérico y compatible con el PmodAD1 [17]. El controlador se implementó en modo máster y es además es el encargado de iniciar la transferencia de datos. En la Fig. 4.137 se muestra el diagrama de bloques del controlador.

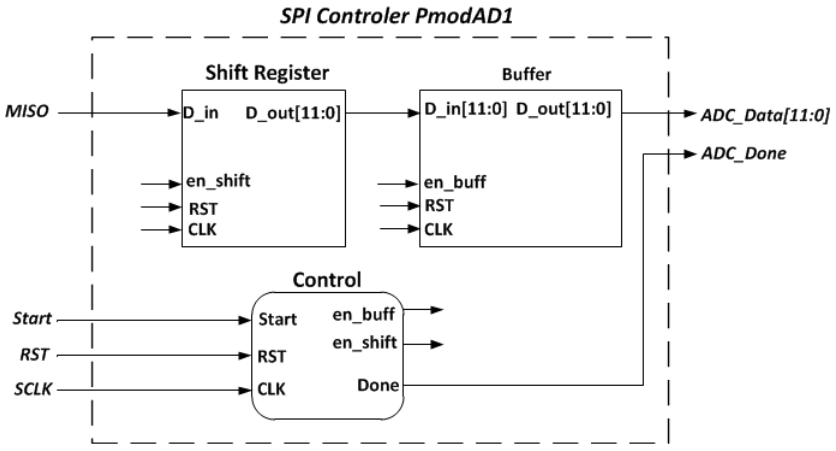


Figura 4.137: Diagrama de bloques de controlador SPI. Tomada de [10].

El modo 0 fue el modo utilizado por el controlador del SPI. Tomando en cuenta que el uso de este controlador es específicamente para uso y transmisión del ADC hacia el SiRPA, la patilla MISO no se implementó.

El estándar SPI trabaja con 16 bits y el PmodAD1 tiene un resolución de 12 bits, por lo cual los primeros 4 bits son ignorados.

Controlador RAM MT45W8MW16BGX

Se diseñó en Verilog HDL un controlador para la RAM externa asíncrona que poseen las tarjetas de desarrollo Digilent Nexys usadas en las pruebas (ver [19]). Esta RAM se utilizó

durante las pruebas de cada etapa, para almacenar tanto los vectores de estímulo como los resultados de la etapa. No será implementada en el circuito final. En la Fig 4.138 se muestra el controlador diseñado.

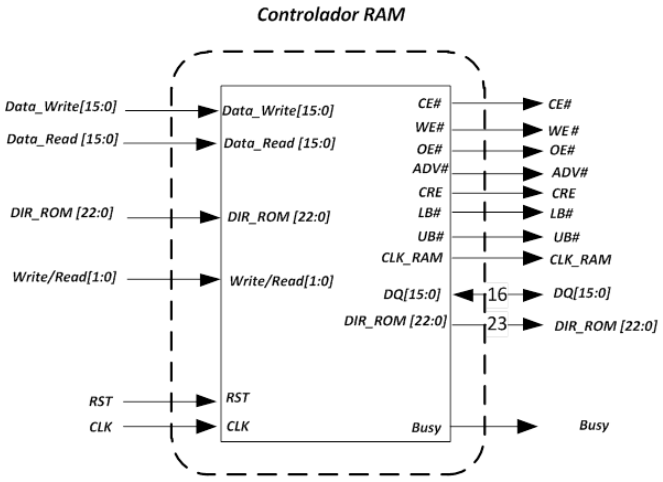


Figura 4.138: Diagrama del controlador de la RAM MT45W8MW16BGX implementado en Verilog HDL. Tomada de [10].

Normalización de la tarjeta de audio PC para envío y adquisición de datos

Para utilizar la etapa de preproceso y normalizar los sonidos de bosque, motosierra y disparos, se requiere acondicionar el volumen de la tarjeta de audio de la PC con los parámetros del micrófono que se utilizará (ver sección 4.1 para más detalles). Se creó un scritp en Matlab con una onda senoidal de amplitud "1", que se reprodujo a través de la tarjeta de audio de la PC, conectada a la entrada del AGC. Como segundo paso se midió con un osciloscopio la salida del AGC ajustando el volumen hasta obtener el máximo que soportaba el AGC a la entrada y que representa aproximadamente 117dB$_{SPL}$.

En la figura 4.139, se muestra la onda obtenida en las primeras pruebas, en donde se observó que dicha señal tiene un valor de excursión máximo de 2.417V_{pp}.

Debido a que al rango de operación del ADC de 0 a 3.3V y la máxima excursión de salida del AGC es hasta 2.417V_{pp}, se perdía un 26.75% de resolución, lo que se resolvió modificando los filtros de salida del acondicionador, dotándolo de una ganancia de unos 5B, lo que llevó la salida de la etapa de acondicionamiento a prácticamente 3.3V (ver sección 4.1).

La FIg. 4.140 muestra el diagrama de segundo nivel de la etapa de adquisición. Cabe destacar que debido a la velocidad de muestreo que utiliza el SiRPA (44100Hz) no es posible utilizar el puerto serie para enviar los datos *en linea*, por las limitaciones de velocidad del protocolo RS232. Por ello la interfaz almacena temporalmente sus datos en la RAM externa de la tarjeta de desarrollo Nexys; al finalizar el sonido de entrenamiento, los datos son enviados de manera *fuera de línea* por el puerto serie.

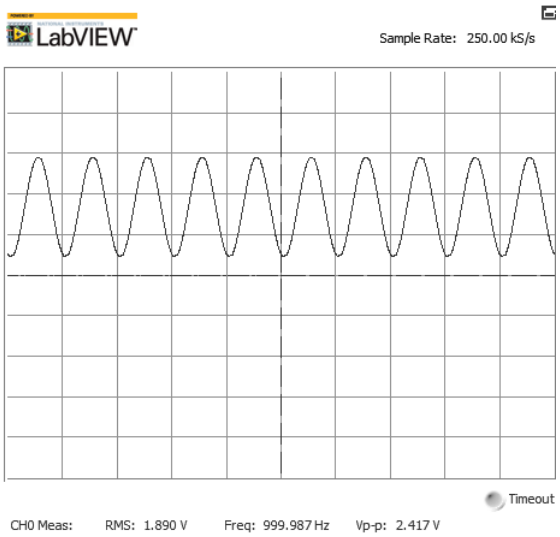


Figura 4.139: Máxima excursión permitida por el AGC

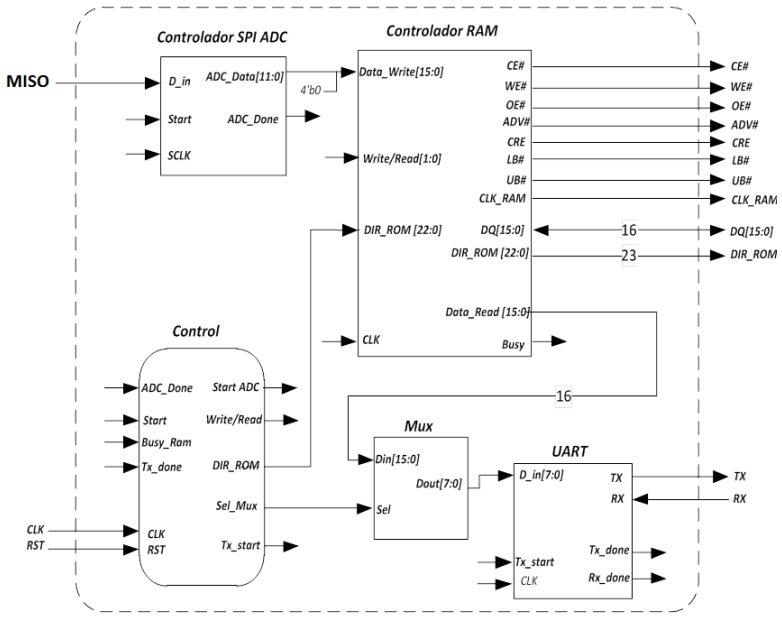


Figura 4.140: Diagrama de interfaz de recepción y envío de datos para entrenamiento. Tomada de [10].

En la figura 4.141 se presenta una muestra la señal obtenida por esta interfaz.

Para la recepción y procesamiento de datos se utilizó *Matlab*. Para preprocesar los datos se necesita que estos se encuentren sincronizados con el momento en que se reproduce el sonido, para empezar a almacenar los valores muestreados por el ADC en la interfaz de la FPGA. En la figura 4.142 se muestra parte del script que se utilizó para habilitar el puerto serie y permitir la recepción de datos.

Comprobada la interfaz de adquisición y ajustada la tarjeta de audio de la PC se realizó el preprocesamiento de los sonidos de entrenamiento. En la figura 4.143 se muestra un

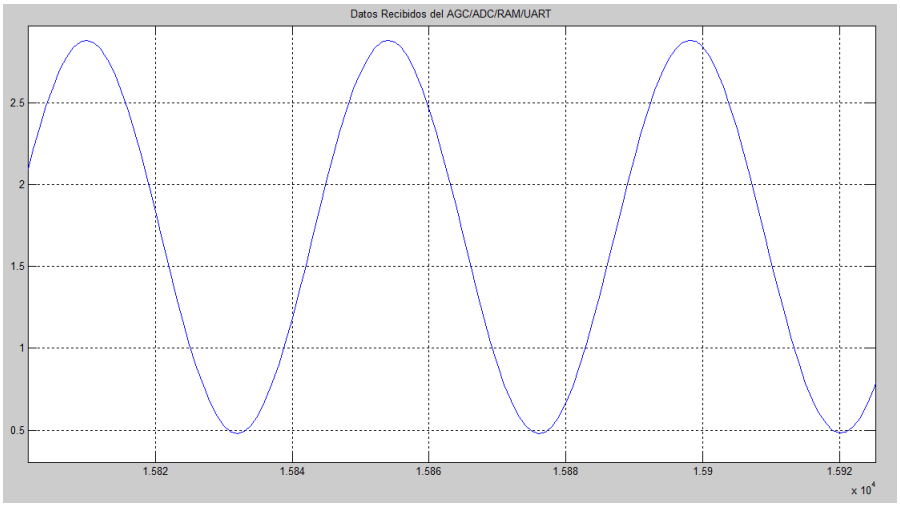
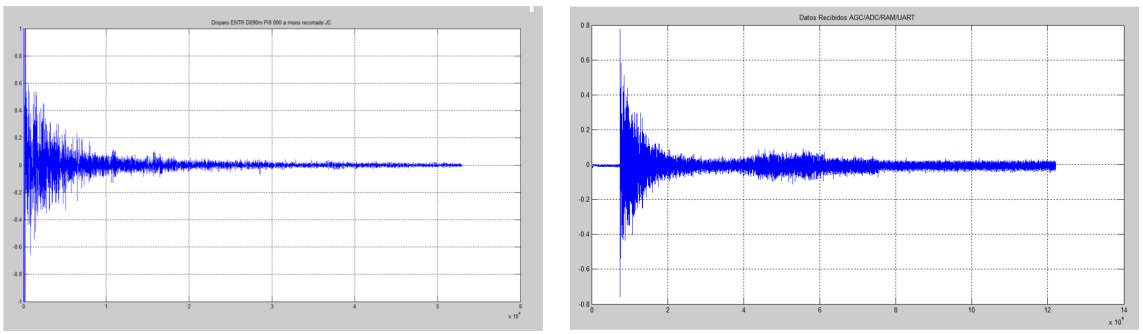


Figura 4.141: Señal senodial de prueba obtenida por la interfaz de adquisición. Tomada de [10].

```
s=serial('COM8', 'BaudRate', 19200); % Habilita el puerto 8 y velocidad
s.InputBufferSize=256;               % Especificar buffer de entrada
s.Timeout=5;                         % Tiempo de espera
fopen(s);                            % Abriendo el puerto
DataRecibido=fread(s);               % Habilita lectura del puerto
fwrite (s,255);                      % Escribe datos en el puerto
fclose(s);                           % cerrando el puerto
delete(s)                            % borrando el objeto de memoria
clear s
```

Figura 4.142: Script para habilitar puerto serie con Matlab. Tomada de [10].

ejemplo del preproceso (normalización con el AGC y muestreo con el ADC), para un disparo de una pistola 9mm.



a. b.

Figura 4.143: a. Disparo de entrenamiento b. Disparo de entrenamiento obtenido. Tomadas de [10].

Los datos recibidos son normalizados por un script creado en Matlab. Estos datos son bytes concatenados para formar el dato original. Se eliminó el nivel de offset brindado por el AGC y se normalizó utilizando un mapeo en un rango de $[-1:1]$ y finalmente se guardó en el .wav, que es el formato de audio que utiliza el HMMSoft para el entrenamiento.

Entrenamiento de coeficientes utilizando el HmmSoft

El *HMMSoft* es una aplicación de entrenamiento para la etapa de clasificación elaborado en [16] y modificado por [55] (ya descrita en la sección en la sección 4.4).

En [55] se entrenó el sistema utilizando los archivos de audio descritos anteriormente. La diferencia que se presenta ahora consiste en que los archivos de audio con los que se re-entrenará han sido preprocesados (normalizados y muestreados) por los componentes físicos que utilizará el SirPA cuando se encuentre en funcionamiento (el módulo de acondicionamiento AGC y el convertidor ADC). .

En el entrenamiento de los módulos con los nuevos archivos de audio se realizó una modificación al HMMSoft, que consistió en suprimir el módulo equivalente del normalizador, haciendo que la entrada de audio de los archivos pase directamente al banco de filtros en el código. El procedimiento, sin embargo, es el mismo al descrito en la sección 4.4.

Pruebas del banco de filtros de entradas

Se utilizó un filtro recursivo que se muestra en la Fig.4.144, que fue desarrollado y comprobado a nivel RTL en [77] y modificado para que cumpliera con la integración física "correcta por construcción (CBC)" en [92]. Con este filtro se implementa el equivalente al banco de filtros encargado de la separación en ocho bandas tiempo-frecuencia de los datos de audio.

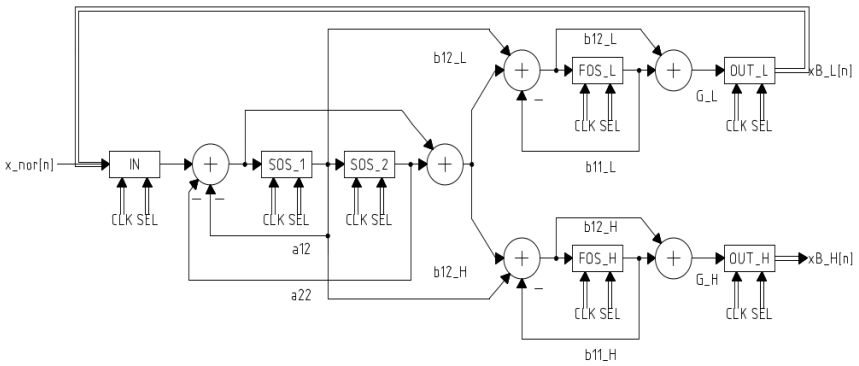


Figura 4.144: Diagrama interno del filtro segmentado. Tomado de [77].

Las características de dicho filtro son las siguientes:

- Filtros QMF elíptico o de Cauer de orden 3.
- IIR.
- Multiplicadores CSD.
- Ancho del bus de datos interno 19 bits.

La corroboración del funcionamiento del filtro consistió en diseñar un banco de pruebas implementado en la FPGA, en donde se excitó el módulo con señales de entradas conocidas

y comparar entre los datos experimentales obtenidos y los datos generados en la sección 4.4.

Se diseñó un módulo en HDL compatible con el filtro mostrado en la figura 4.145, con una ROM donde se cargan los datos de entrada del filtro, ingresados cada $44,100Khz$; el control cumple la función de tomar dichos datos y almacenarlos en un banco de registros a la salida del filtro segmentado, para luego enviarlos por el puerto serie y recibirlos en la PC para procesarlos y compararlos con los datos de las pruebas en sistema embebido.

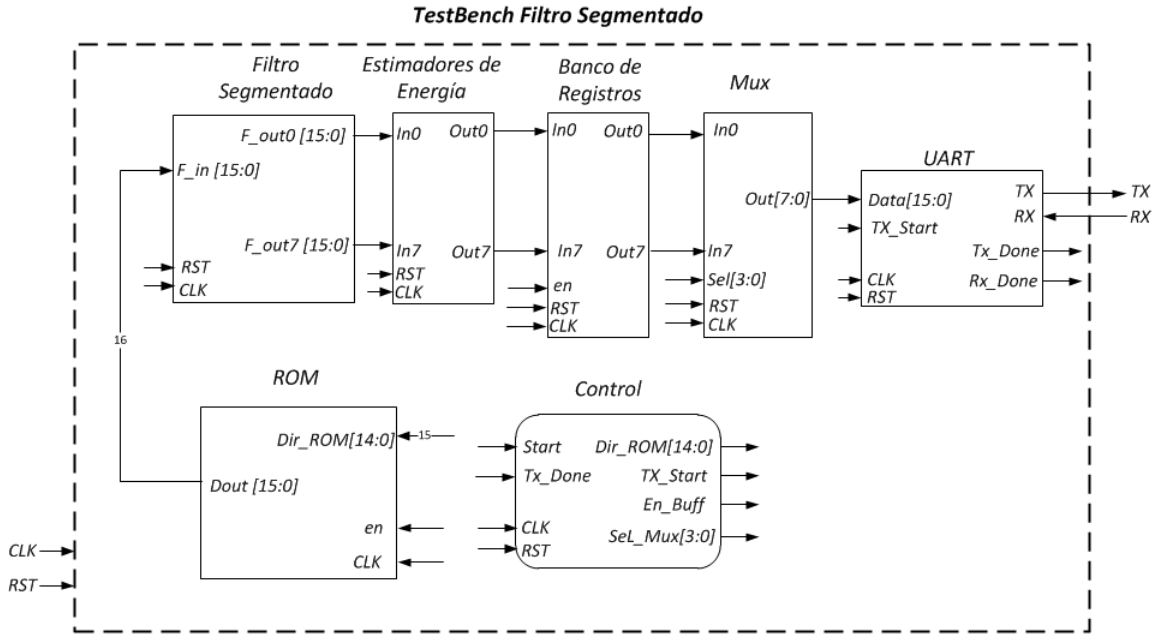


Figura 4.145: Diagrama de módulo para testbench del filtro segmentado. Tomada de [10].

A este filtro se le realizó una serie de modificaciones para lograr un funcionamiento correcto, pues el mismo presentaba gruesos errores de respuesta con respecto al modelo teórico.

El banco de filtros utiliza una codificación subbanda, lo que realiza una separación de la señal de entrada en diferentes bandas de frecuencia, con un tamaño una octava menor a la anterior. Así existía un error de concepto al tomar los datos de la salida del filtro a una frecuencia de $172,265625Hz$, siendo la correcta $344,53125Hz$ tal como se definió en la tabla de frecuencias para el banco en la sección 4.4.

Según [92], en la implementación del filtro segmentado era necesario agregar una señal de reinicio, esto para que cumpliera con las reglas de integración física "correcta por construcción (CSB). Dicha señal reinicio afectaba tres módulos principales: el multireloj, el registro de pila y el reloj de prioridad.

En el caso del reloj de prioridad, dicho reinicio no se encontraba bien conectado, tal como se muestra en la figura 4.146. Con respecto a los siguientes dos módulos el reinicio era activo en bajo, algo que se corrigió para compatibilidad con el reinicio activo en alto de las demás etapas.

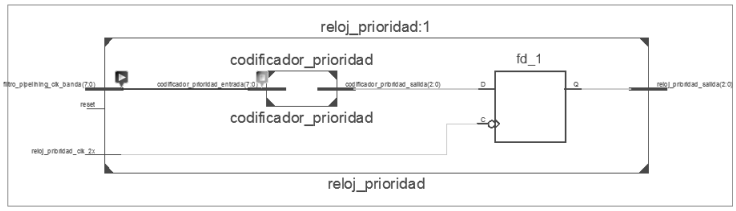


Figura 4.146: Error en la implementación de reinicio del filtro. Tomada de [10].

Con la ayuda del Ing. Carlos Adrián Salazar se corrigieron los errores de codificación y estructura que afectaban a la unidad: se eliminaron los módulos de múltiples relojes desarrollados en [77] y se dejó el circuito funcionando con un único reloj (como es preferible en un sistema sincrónico digital), con unidades de chequeo y corrección de desborde, y con las frecuencias de muestreo adecuadas. Además, se ajustó la velocidad de procesamiento de la unidad recursiva, llevándola a 790kHz, para poder procesar los ochos bandas necesarias en el tiempo disponible dado por la velocidad de muestreo de 44.1kHz. El diagrama interno funcional del circuito sigue siendo el mismo, no obstante. Se diseñó además un módulo estimador de energía, ausente en la implementación de [77] (aunque descrita teóricamente su necesidad ya en esa misma referencia, y en la sección 4.4) y que se muestra de nuevo en la Fig 4.147).

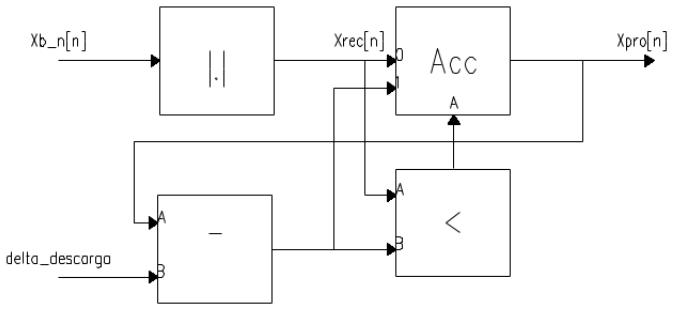


Figura 4.147: Diagrama de bloques del estimador de energía implementado. Tomado de [77]

El resultado de las pruebas del banco de filtros, una vez agregado el estimador de energía, se muestra en la Fig. 4.148, en donde se observan claramente errores debidos al desborde las unidades aritméticas.

En la Fig. 4.149 se muestra ya la respuesta completa del banco de filtros. Nótese la disminución del error, que es producto solo de la reducción de resolución numérica desde 128bits en punto flotante para las pruebas en sistema embebido, a 18 bits en punto fijo, para las pruebas en la FPGA.

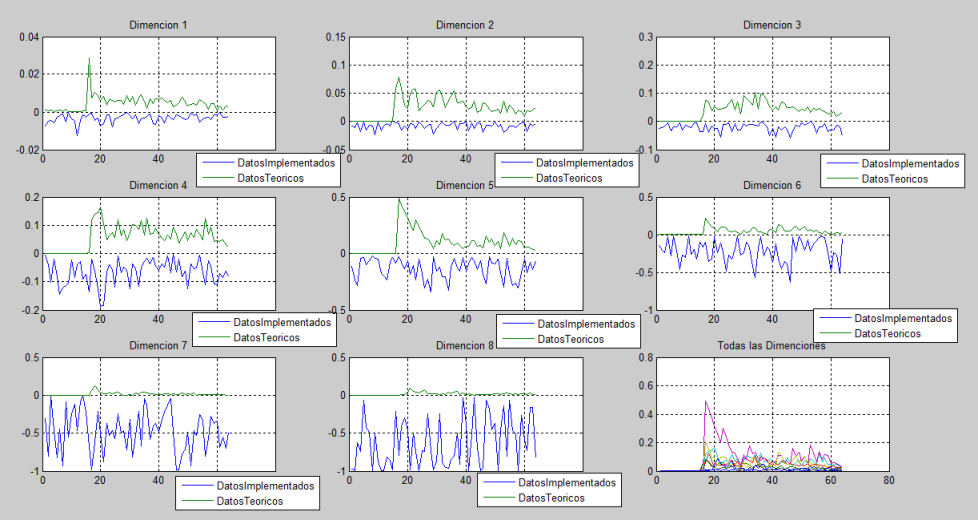


Figura 4.148: Respuesta del banco de filtros por coeficiente o dimensión, contra la respuesta teórica. Nótese el gran error en la salida de cada banda. Los errores descubiertos eran debidos al desborde las unidades aritméticas. Tomada de [10].

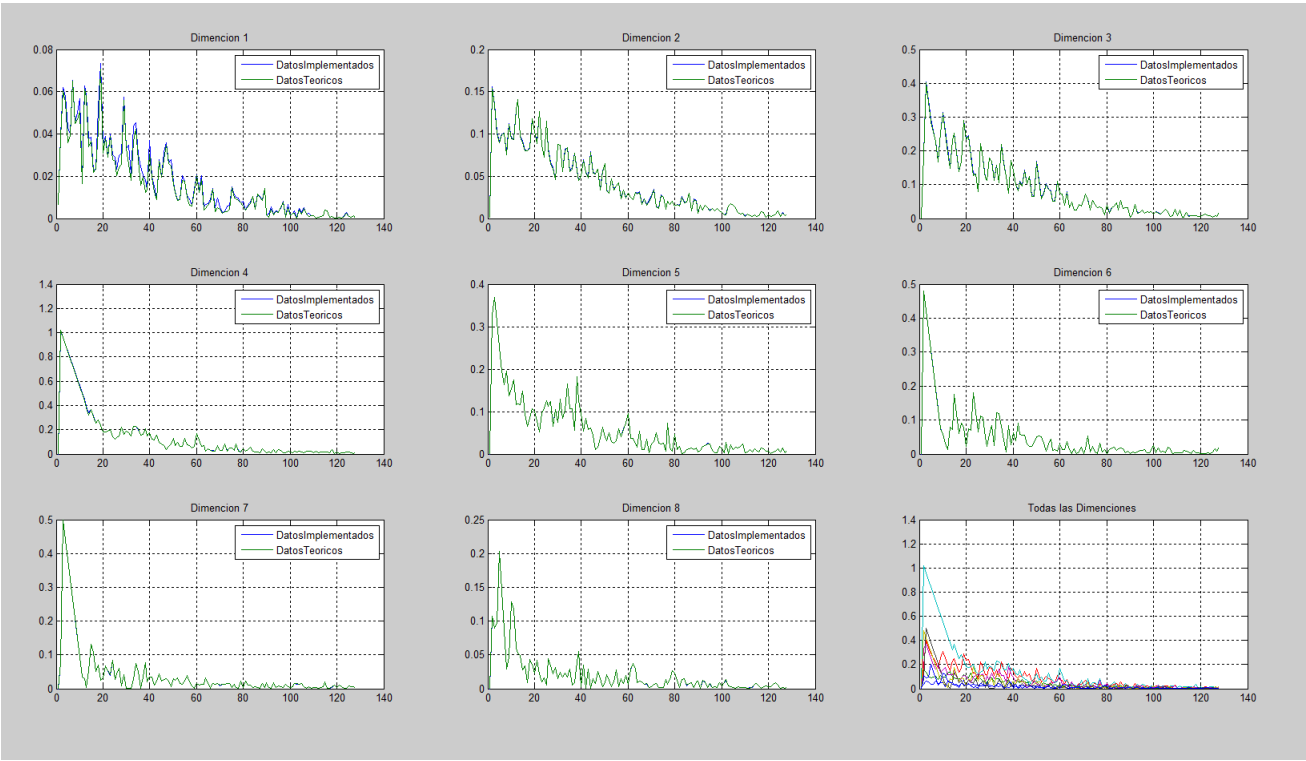


Figura 4.149: Respuesta del banco de filtros por coeficiente o dimensión, contra la respuesta teórica, después de las correcciones. El error existente ahora es producto solo de a reducción de resolución numérica (de 128bits en punto flotante para las pruebas en sistema embebido, a 18 bits en punto fijo, en la FPGA).

Pruebas del reductor de dimensiones o LDA

La comprobación de funcionamiento del reductor de dimensiones ya descrito en 4.5.1 se realizó también en la FPGA. Se obtuvo la respuesta del reductor una vez excitado

con una serie de entradas conocidas, y enviadas a través del puerto serie de la PC para ser analizadas con respecto a los datos de respuesta obtenidos de la sección 4.4. (ver Fig.4.150). Se tiene una memoria ROM principal encargada de almacenar todos los datos de entrada y un banco de 8 registros habilitados por el control para guardar los datos temporales para las entradas del reductor.

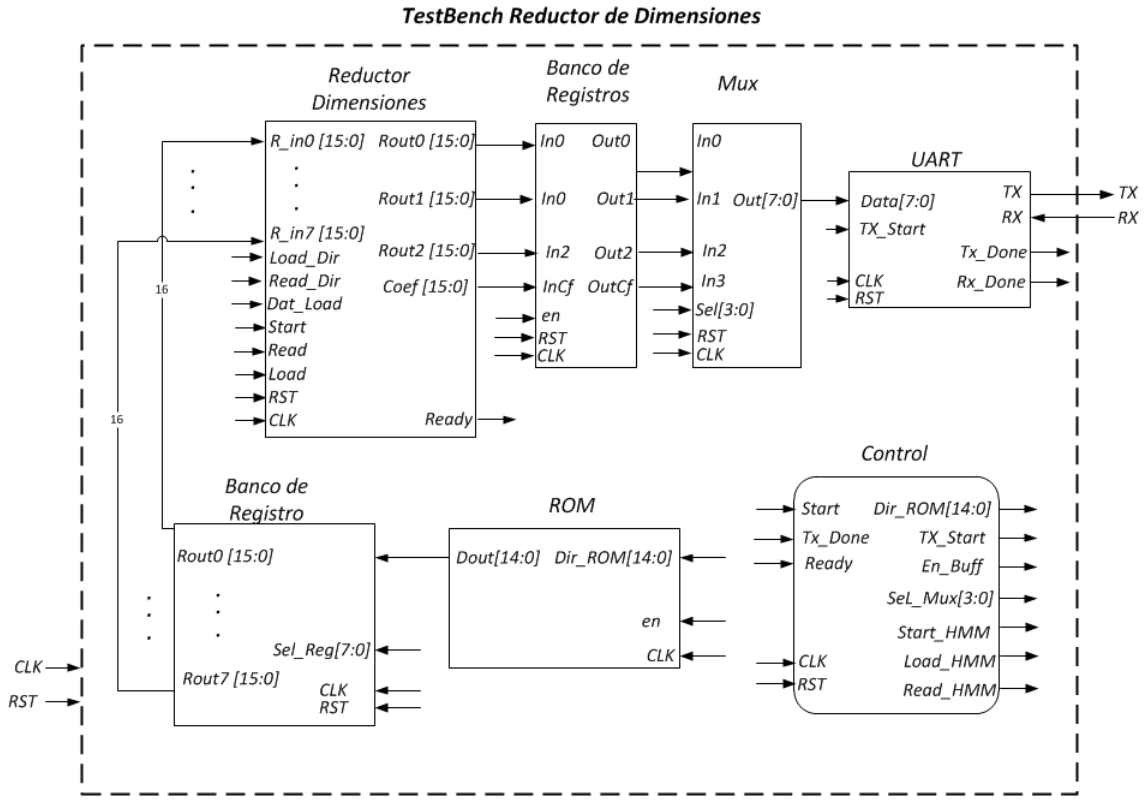


Figura 4.150: Diagrama de bloques del banco de pruebas para el reductor de dimensiones. Tomada de [10].

Cabe destacar que el módulo de pruebas se diseñó con la capacidad de leer y escribir los coeficientes del reductor, con el fin de garantizar su funcionamiento en la implementación en la FPGA. Utilizando Matlab y aprovechando las características que posee este, se creó un script encargado de generar un archivo .bin con los datos en binario necesarios para la prueba en FPGA.

Se creó otro script para la recepción de datos a través del puerto serie, el cual recibe los datos en bytes, los concatena y los mapea al valor decimal correspondiente. Posteriormente utiliza estos valores para graficarlos contra los valores teóricos. Los valores generados despliegan en Matlab.

Para garantizar el correcto funcionamiento del reductor de dimensiones, se realizaron modificaciones, principalmente en el módulos de la ALU(sumador, restador y multiplicador) y en el control de la lectura de los coeficientes. El diagrama final de segundo nivel se muestra en la Fig.4.151.

Se corrigió un error en el manejo de la lectura de los coeficientes y otro en la generación

de la dirección de lectura de los coeficientes.

El uso del LDA permite obtener la matriz de transformación **W** y el vector de offset μ, que son utilizados para realizar la transformación lineal, y poder proyectar un espacio octadimencional a uno tridimensional reduciendo así las dimensiones del espacio a explorar.

Se utilizaron los 193 archivos de audio preprocesados de las tres clases disponibles, utilizando un muestreo de uno, lo que permitió utilizar todas las muestras de cada uno de los sonidos, la matriz **W** obtenida en (4.50).

$$W = \begin{bmatrix} -0.638745 & -0.977134 & 0.949478 \\ -0.010113 & -0.109405 & -0.150462 \\ -0.0046262 & -0.030345 & -0.049580 \\ 0.394740 & 0.100250 & 0.207472 \\ 0.459854 & -0.012702 & 0.065774 \\ -0.407438 & 0.0041747 & -0.016849 \\ 0.234205 & -0.102005 & -0.096295 \\ 0.040233 & -0.099800 & -0.128322 \end{bmatrix} \tag{4.50}$$

y el vector de offset μ en (4.51).

$$\mu = \begin{bmatrix} 0.010154 & 0.007922 & 0.008348 & 0.007613 & 0.009977 & 0.013422 & 0.009212 & 0.019448 \end{bmatrix} \tag{4.51}$$

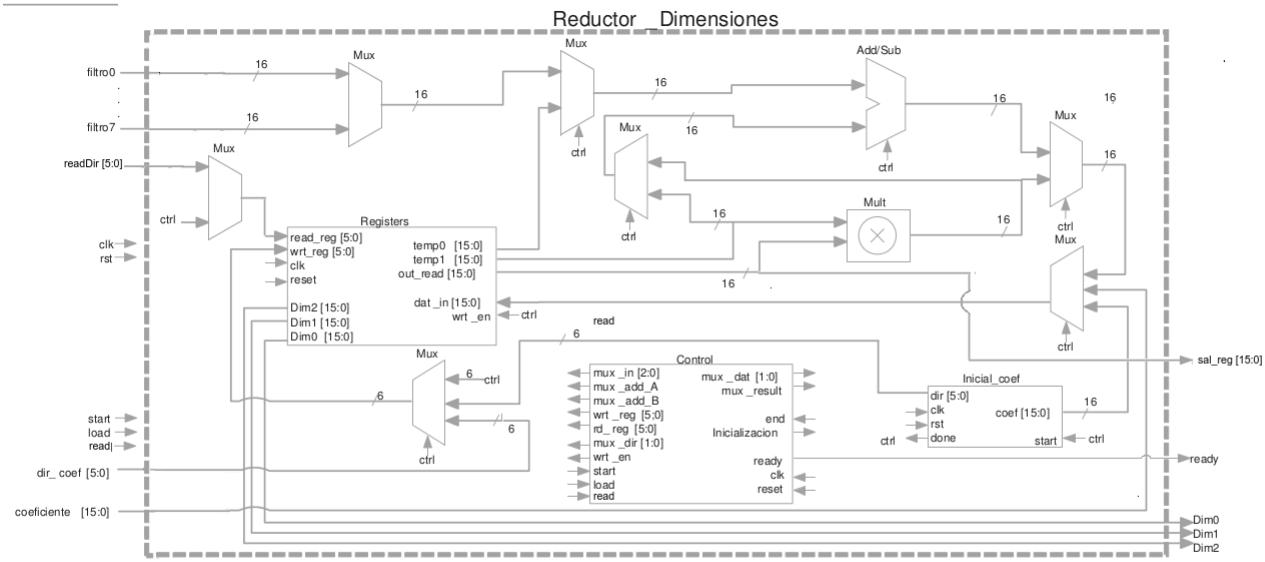


Figura 4.151: Diagrama de 2 segundo nivel de reductor de dimensiones. Modificado de [34].

Se resincronizaron además el acumulador, los coeficientes y offsets con los datos de entrada. Se ajustó la ALU del reductor para operaciones con signo y se creó un script que

permite ingresar el valor decimal de los coeficientes y generar el valor representativo en hexadecimal.

Al realizar las modificaciones correspondientes al reductor de dimensiones se alcanzó un comportamiento aceptable, que se muestra en la FIg. 4.152. Cabe destacar que por fines ilustrativos dicha figura solo muestra hasta 200 reducciones, pero la interfaz implementada permite realizar hasta 2000 reducciones.

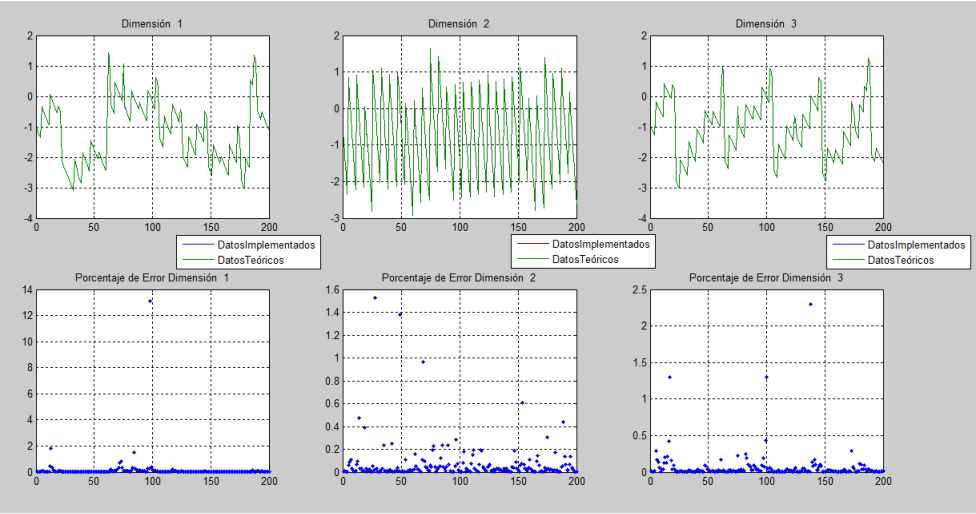


Figura 4.152: Respuesta del Reductor de dimensiones teóricas y experimentales con su porcentaje de error. Tomada de [10].

El porcentaje de error obtenido es menor al 0.1%, un error más que aceptable, debido a la utilización de un formato punto fijo en vez de un formato de punto flotante.

Pruebas del árbol-clasificador en la FPGA

En el caso del árbol-clasificador o generador de símbolos (ver sección 4.5.2), se diseñó un módulo compatible para implementarlo de igual manera que los módulos anteriores en la FPGA, y el cual permitió agregarle una serie de entradas conocidas para obtener su respuesta para un posterior análisis en la PC.

El módulo implementado a nivel de software en el sistema embebido presenta una diferencia con respecto al implementado en HDL. La diferencia se encuentra en que en el sistema embebido utiliza el algoritmo de árbol k-dimensional, que constantemente se encuentra calculado distancia de hiperrectángulos por lo que su implementación a nivel de hardware se complica y aumenta el consumo de recursos, por lo que en [34] se tomó la decisión de realizar un árbol comparador utilizando la distancia L1 o Manhattan. La matriz obtenida como resultado del kd-medias se muestra en la tabla 4.61.

En la Fig.4.153, se muestra el módulo hecho con el fin de probar el comportamiento del módulo en la FPGA. Se crearon tres memorias ROM con entradas conocidas para el árbol, que se cargaban vía puerto serie.

Tabla 4.61: Resultado de los centroides re-calculados. Tomada de [10].

Posición	Coordenadas		
	X	Y	z
0	0.002268	0.010565	-0.006228
1	0.012963	0.008158	-0.004006
2	0.031456	-0.003086	0.014269
3	0.006512	0.001633	0.001942
4	0.000209	0.005444	-0.002275
5	-0.007886	-0.006790	0.009055
6	0.007485	-0.000944	-0.016842
7	-0.010738	-0.002009	-0.018987
8	-0.031306	-0.003282	-0.026480
9	0.021365	-0.020481	-0.041924
10	0.158611	0.006225	0.039026
11	0.030121	0.006433	0.002433
12	-0.000042	-0.001833	0.004175
13	0.012859	-0.011221	-0.028415
14	0.017521	-0.000148	0.008304
15	0.018882	-0.005833	-0.002546
16	-0.036999	0.005681	-0.008957
17	0.008277	-0.013302	-0.010493
18	-0.001942	0.002633	0.000318
19	-0.011440	-0.013152	0.014416
20	-0.005824	-0.002937	0.005667
21	0.025651	-0.010788	-0.015602
22	-0.006238	-0.012604	-0.034580
23	-0.004488	-0.008329	-0.003067
24	0.007380	-0.008737	0.008847
25	0.003702	-0.001714	-0.005110
26	0.042820	-0.007985	-0.003063
27	-0.004251	0.000185	0.002694
28	0.055623	0.005255	0.013811
29	0.093956	0.006490	0.021817
30	-0.018345	0.004570	-0.003732
31	-0.060554	0.006202	-0.017201

En la figura 4.154 se muestran los resultados del árbol clasificador o generador de símbolos, para los mismos vectores de datos con que se evaluaron las pruebas en el sistema embebido, de la sección 4.4.

El porcentaje de error total obtenido es $0,06109375\%$, un porcentaje sumamente bajo pese a la representación en punto fijo usada.

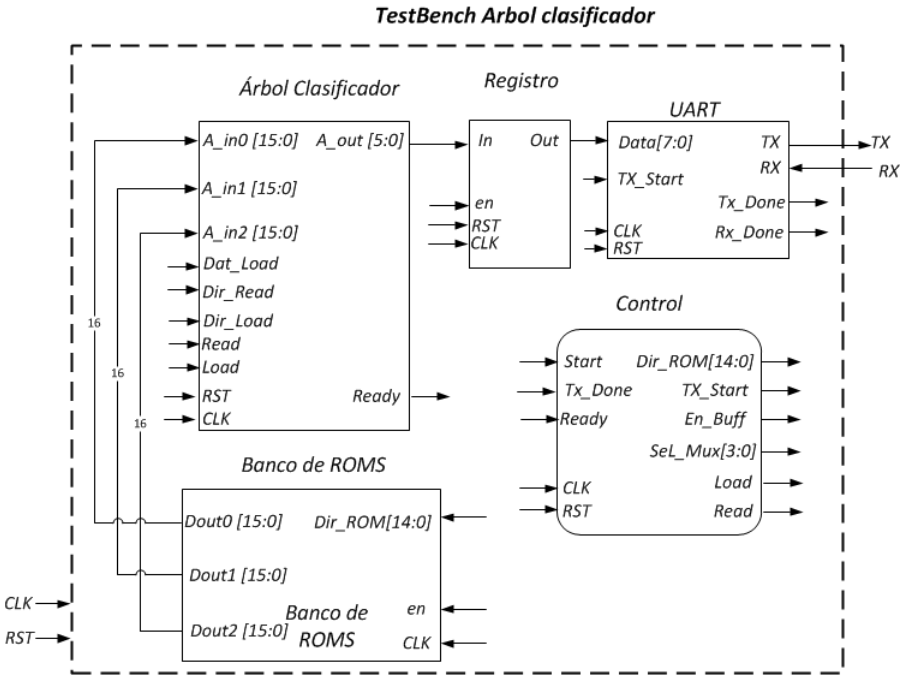


Figura 4.153: Diagrama de bloques del banco de pruebas para el Árbol-clasificador. Tomada de [10].

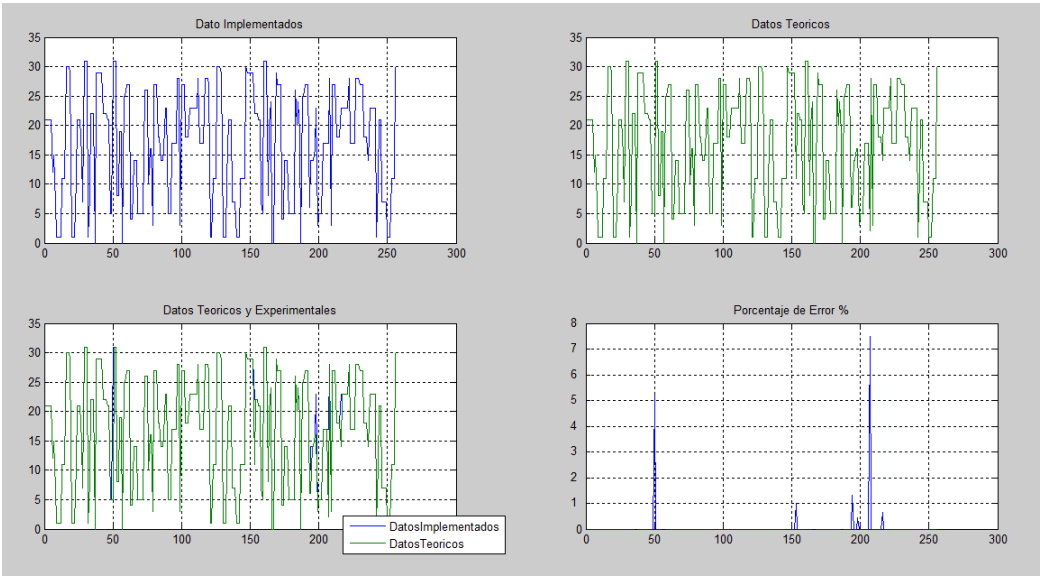


Figura 4.154: Estímulos para las entradas del reductor de dimensiones. Tomada de [10].

Pruebas en simulación de la HMM

Resultados para el reentrenamiento y pruebas de la unidad HMM

El resultado del entrenamiento de las etapa de clasificación se encuentra en resumido en matrices de confusión mostrada en la figura 4.155. Las matriz de confusión es una herramienta que permite obtener una mejor idea de la clasificación obtenida. Es de tamaño $N \times N$, donde N es el número de clases usadas en la etapa de clasificación. Las

columnas de esta matriz representan la clase real, mientras que las filas indican la clase que fue asignada por el clasificador. En la primera *matriz de sensitividad* se indica cuál es la probabilidad de un objeto que pertenece a una clase sea correctamente clasificado, y la matriz de *valor predictivo positivo* (Vpp) indica cuál es la probabilidad de que un objeto asignado a una clase, pertenezca verdaderamente a esa clase.[55]

Sensitividad

	Bosque	Disparo	Motosierra
Bosque	**91.89 %**	6.67 %	46.67 %
Disparo	2.70 %	**93.33 %**	3.33 %
Motosierra	5.41 %	0.00 %	**50.00 %**

Valor predictivo positivo

	Bosque	Disparo	Motosierra
Bosque	**69.39 %**	2.04 %	28.57 %
Disparo	6.25 %	**87.50 %**	6.25 %
Motosierra	11.76 %	0.00 %	**88.24 %**

Figura 4.155: Matriz de confusión del resultado de entrenamiento HMM, Tomada de [10].

La clasificación de un sonido en la clase disparo tiene 93% de posibilidad de ser clasificado correctamente y alrededor de 87% de posibilidad de que un sonido que haya sido clasificado como disparo, realmente lo sea.

Las clases motosierras y bosque no presentan la alta probabilidad de ser clasificadas correctamente como la de disparo. Basándonos en las matrices de confusión en 4.155, se concluye uno de cada dos sonidos de motosierra son clasificados en la clase motosierra. La mayoría de estos sonidos se encuentra siendo clasificada como bosque, al rededor de un (46.46%).

Los sonidos de motosierras y bosque se encuentran grabados a 30, 90, 250 y 600 metros de distancia de la fuente emisora de sonido, algo que puede ser determinante en el resultado de dicha clasificación. Esto a que los sonidos muy alejados (250 y 600 metros) tienen a ser captados muy bajos y probablemente el normalizador (AGC) termine amplificando ruido natural de bosque. Este resultado no es en realidad negativo, en el sentido de que el sistema no se encontrará brindando *falsos positivos*, si no más bien que a dichas distancias el sistema ya no es eficiente detectando motosierras.

Se llevó a cabo una segunda iteración de entrenamiento, en la que se descartaron los sonidos grabados a más de 250 metros. En la figura 4.156 se muestran la matrices de confusión obtenidas.

Estas matrices evidencian como mejoró el comportamiento descrito anteriormente, pues ahora para distancias largas el HMM tuvo una eficiencia de clasificación de sonidos de disparos de un 100% de probabilidad, y un 75% de probabilidad para motosierras (una mejora casi un 50 % con respecto al caso anterior).

Sensitividad

	Bosque	Disparo	Motosierra
Bosque	**97.30 %**	0.00 %	25.00 %
Disparo	0.00 %	**100.00 %**	0.00 %
Motosierra	2.70 %	0.00 %	**75.00 %**

Valor predictivo positivo

	Bosque	Disparo	Motosierra
Bosque	**94.74 %**	0.00 %	5.26 %
Disparo	0.00 %	**100.00 %**	0.00 %
Motosierra	14.29 %	0.00 %	**85.71 %**

Figura 4.156: Matriz de confusión del resultado de entrenamiento HMM sin sonidos de 600m. Tomada de [10].

Simulaciones del código HDL de la HMM

Antes de proceder a las pruebas en FPGA, se realizaron simulaciones exhaustivas de la unidad de HMM descrita en Verilog. Desgraciadamente, pudo comprobarse mediante estas simulaciones que la unidad no es efectiva por los efectos de falta de resolución de la unidad de cálculo de punto flotante. Esta fue implementada con una mantisa de solo 23 bits, y ocho de exponente. Desgraciadamente, al tratarse el algoritmo adelante-atrás de una sumatoria de múltiples productos de probabilidades, los valores de cálculo tienen rápidamente a cero. En todos los casos evaluados, no llegaba a terminarse la evaluación de la cadena de símbolos de cada sonido sin que los resultados parciales ya quedaran fijos en cero. Hay solo dos posibles soluciones, o incrementar el tamaño de la representación numérica (llevarla a 128 bits, como se hizo en las pruebas en el sistema embebido) o aplicar el HMM utilizando logaritmos (este problema ya fue analizado en la sección 4.4.1). Por estarse precisamente en las etapas finales del proyecto antes de enviar a fabricar esta unidad, no hay tiempo para realizar estos cambios antes de la fecha límite, ni ha sido posible documentar los cambios propuestos en este informe.

4.5.5 Conclusiones parciales sobre la implementación en RTL-FPGA del sistema de reconocimiento de patrones acústicos

A partir de los resultados alcanzados y comparados contra la implementación en sistema embebido, se puede decir que se cuenta con un código verificado y funcional de un 75% del circuito de clasificación. Se trabaja precisamente en los detalles finales para el envío a fabricación de las unidades de adquisición de datos, banco de filtros, reductor de dimensiones y árbol clasificador. No obstante, no podrá enviarse a fabricar la etapa HMM, que deja al circuito inconcluso, aunque será posible comprobar el funcionamiento de las etapas anteriores de extracción de características posteriormente, al incorporar la etapa faltante de HMM sobre una FPGA.

Es posible por tanto concluir que es factible y mucho más compacto realizar una implementación en hardware concurrente del SiRPA. Sin embargo, debe admitirse que la

escasez de recursos humanos y técnicos para realizar esta implementación han influido enormemente en no poder culminar este objetivo, y esto debe servir de advertencia para futuros proyectos microelectrónicos demasiado ambiciosos, que pretendan construir circuitos de similar complejidad a los buscados en este proyecto en un lapso de dos años, algo que a nivel industrial lleva casi el doble del tiempo, con amplios recursos tanto de personal como financieros.

4.6 Propuesta a nivel de bloques de la arquitectura de comunicación del SoC con los demás nodos de la red

Uno de los objetivos finales de este proyecto era establecer al menos a nivel de bloques la arquitectura final de comunicación del SoC con el resto de la red. Se dividió este problema en dos partes: por un lado la búsqueda de un protocolo de comunicación eficiente en términos energéticos y, por otro, la implementación de la red básica en una plataforma comercial.

4.6.1 Implementación y pruebas de una plataforma básica de red de sensores

Este segundo objetivo fue reportado en [56]. Para cumplirlo, se diseñó y construyó una red de sensores, que utiliza una "topología tipo malla" la cual permite el envío de las alertas producidas entre todos los nodos que se encuentran dentro del radio de alcance.

Esta red utiliza un protocolo de comunicación que permite vincular cada dispositivo (nodo) en forma descentralizada a efecto de que esté en posibilidad de enviar información o mensajes a partir de saltos múltiples de la información hacia los demás dispositivos (nodos o terminales) conectados a la red hasta llegar al nodo destino (sumidero). Con esta red se pretende la monitorización de eventos en zonas protegidas Este proceso se puede resumir en cuatro etapas:

1 Monitorización el entorno y la notificación en caso de presentarse un evento; para esto se requiere de sensores o dispositivos capaces de identificar las variables involucradas.

2 Transmisión de la notificación del evento desde el punto de origen hasta el punto central desde donde se monitoriza la actividad, mostrando al operador, además, la información correspondiente al evento.

3 Búsqueda y detención del causante de la alerta.

4 Notificación a las autoridades policiales, seguido del desplazamiento a la zona del evento.

Para la implementación de la solución desarrollada se utilizaron sistemas que integran varias secciones del hardware en un solo paquete. Se utilizó el módulo de hardware AVR

Raven, que incluye el microcontrolador de procesamiento de información, microcontrolador de control de antena, dispositivos de IO, la antena, dispositivos de memoria y entradas para el sensor. El RZUSBSTICK incluye la antena, el microcontrolador y también el puerto para conexión USB. Más información técnica sobre el hardware probado puede hallarse en [56].

Se construyó el software necesario para implementar una red de cuatro sensores, que fueron probados en campo con el protocolo adecuado. Se verificó el funcionamiento de la red y se contrastó contra simulaciones de la misma red en el programa Radio Mobile. En la Fig.4.157 se muestra la zona de pruebas real y simulada en el Campus del TEC.

Figura 4.157: Campo de pruebas para la red de cuatro enlaces y un sumidero, usado para las pruebas de la red y su contraste por simulación mediante Radio Mobila. Tomada de [56].

Se verificó tanto el protocolo como la funcionalidad de los radio enlaces. Además se desarrolló un software de control desde una PC para la red total, y se verificó la robustez del protocolo implementado. Para más detalles sobre las pruebas y evaluaciones hechas a esta red, ir a [56].

4.6.2 Propuesta del protocolo de comunicación

Se propuso el diseño del protocolo de comunicación utilizado en una red de sensores para la monitorización de variables diversas en el bosque; el protocolo debía abarcar funciones correspondientes a las capas de red, control de acceso al medio y de enlace de datos según el modelo de interconexión de sistemas abiertos (OSI). Esta propuesta fue desarrollada

por el Ing. Ronny García como su tesis de maestría en ciencias de la computación. Esta sección es un resumen de los logros alcanzados en dicha tesis (para más detalles ver [71]).

Entre los principales problemas que se enfrentaron estaba la reducción del consumo energético del módulo de comunicación con respecto del protocolo IEEE802.15.4, obtener una latencia aceptable en la transmisión de la información y lograr que cada nodo de la red tenga la capacidad de configurarse por sí mismo de manera que los módulos no tengan que ser modificados en procesos de expansión o falla. Con el objetivo de reducir el consumo energético y cumplir con los objetivos de la aplicación, el protocolo propuesto debería tener las siguientes características:

- **Reducir las colisiones y minimizar del tráfico en la red:** Se propuso la creación de conglomerados que recibieran la información de los nodos vecinos, y que generaran tramas para comprimir y mezclar la información, de forma que se reduzca la cantidad de tramas circulando en la red y la probabilidad de retransmisión debida a colisiones. Al usar un protocolo multicapa se propuso el uso de las mismas tramas de control usadas para la administración del canal para la administración de la capa de red.

- **Debe procurar el éxito del envío de los mensajes:** El protocolo propuesto debía contener mecanismos que garantizaran que el mensaje fuera recibido de manera exitosa en el siguiente nodo. Se propuso el uso de tramas de reconocimiento de recibido, llamadas en adelante ACK, con un chequeo de suma $CHECK\_SUM$.

- **Debe apagar la etapa de radio por la mayor cantidad de tiempo posible:** Se supone que la mayoría del tiempo la red se va a encontrar en reposo y monitorizando el ambiente, por lo que os nodos apagarán la etapa de radio si no es necesaria. Se propuso el uso de ciclos periódicos de hibernación en todos los nodos

- **Debe tener una latencia aceptable:** Por lo anterior, hay un impacto directo en la latencia de las tramas que es importante acortar para que quede dentro de un margen aceptable para la aplicación planteada. Se propuso modelar el retardo de las tramas con respecto al tiempo de hibernación de los nodos.

- **Selección de la ruta más corta posible:** Es necesario que los paquetes transiten por la ruta más corta posible. Se propuso un protocolo en el que los nodos supieran su distancia en cantidad de saltos a la base y sean capaces de escoger la ruta más corta.

- **Auto-reconfigurable:** El protocolo de red debe ser reconfigurable, conforme algunos de los miembros de la red se queden sin energía de forma que sea capaz de mantener la funcionalidad de la red por más tiempo. Para cumplir con este requerimiento cada nodo monitoriza el ambiente de forma periódica para detectar cualquier cambio en la topología de la red.

Descripción del protocolo de control de acceso al medio propuesto

Se decidió evitar el uso de protocolos llamados "libres de colisiones", en los que se le asigna a cada uno de los nodos en un dominio de colisión un canal, o un espacio de tiempo en el que puede transmitir sin competidores; un ejemplo de estos protocolos son los que usan TDM para la administración del canal. Estos protocolos suponen un mayor consumo de energía en los nodos encargados de regular el uso del canal, o el espacio de tiempo que corresponde a cada uno de los nodos para transmitir, esto debido a que estos nodos deben sincronizar la transmisión de los nodos que se encuentran alrededor. Si bien es cierto que este tipo de protocolos reducen el desperdicio de energía por retransmisiones causadas por colisiones en los nodos que no están administrando el canal, también es cierto que la aplicación específica para la cual se está diseñando es una aplicación de bajo tráfico, en la que se espera que se transmitan tramas de forma esporádica, por lo que se considera de mayor ganancia mantener un protocolo donde todos los nodos sean capaces de mantener periodos de sueño tan prologados como sea posible y no transmitir tramas a menos que sea estrictamente necesario.

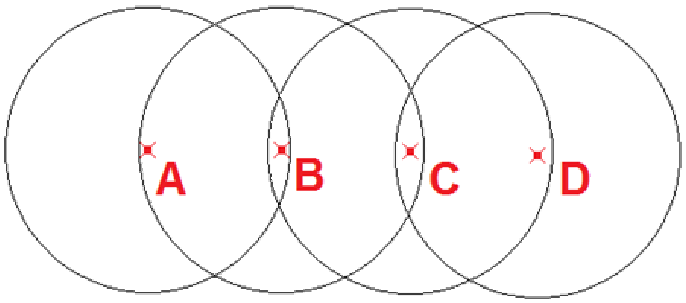


Figura 4.158: Problema de la estación expuesta y la estación oculta, (tomada de[87]).

Un protocolo CSMA no es realmente adecuado porque lo que importa en las comunicaciones inalámbricas es la interferencia en el receptor, no en el emisor. Para entender la naturaleza de este problema, en la figura 4.158 se ilustran cuatro estaciones inalámbricas.

El alcance de radio es tal que A y B están en el mismo alcance y potencialmente pueden interferir entre sí. C también podría interferir tanto con B como con D, pero no con A. Cuando A está transmitiendo hacia B. Si C detecta el medio no podría escuchar a A porque esta fuera de su alcance y por tanto deducirá falsamente que puede transmitir a B. Si C comienza a transmitir, interferirá en B eliminando la trama de A. El problema de que una estación no pueda detectar a un competidor potencial por el medio, puesto que dicho competidor esta demasiado lejos se denomina problema de estación oculta.

Considérese ahora la situación opuesta: B transmitiendo a A. Si C detecta el medio, escuchará una transmisión y concluirá equivocadamente que no puede enviar a D, cuando de hecho tal transmisión causaría una mala recepción solo en la zona entre B y C, en la que no esta localizado ninguno de los receptores pretendidos. Esta situación se denomina problema de estación expuesta.

El problema es que antes de comenzar una transmisión, una estación realmente necesita saber si hay actividad o no alrededor del receptor. El CSMA simplemente le indica si hay o no actividad alrededor de la estación que esta detectando la portadora. Con un cable, todas las señales se propagan a todas las estaciones de manera que solamente puede llevarse a cabo una transmisión en un momento dado en un lugar del sistema. En un sistema basado en ondas de radio de corto alcance, pueden ocurrir transmisiones simultáneas si las ondas tienen destinos diferentes y estos están fuera de alcance entre sí.

Tomando en cuenta lo anterior se plantea un protocolo cuya estrategia sea que el emisor estimule al receptor a enviar una trama corta de manera que las estaciones cercanas puedan detectar esta transmisión y eviten ellas mismas hacerlo durante la siguiente trama de datos. Este protocolo es el que se conoce como MACAW (Acceso múltiple con prevención de colisiones inalámbrico), y fue desarrollado inicialmente por Karn, Bharghavan, y Cols entre 1990 y 1994[87].

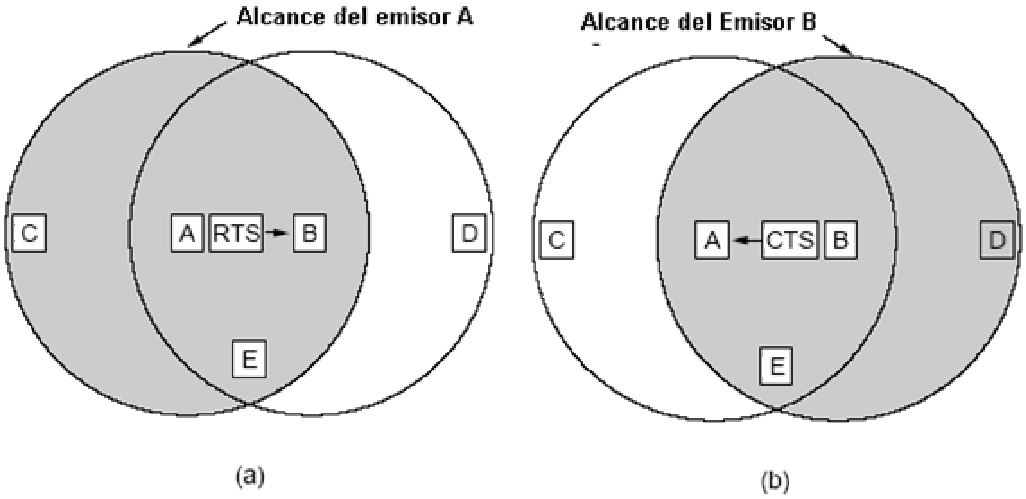


Figura 4.159: Ilustración del funcionamiento del protocolo planteado durante la etapa de reconocimiento del problema. (tomada de[87])

En la figura 4.159 se ilustra el concepto. Considere la manera en que A envía una trama a B. A comienza por enviar una trama RTS (solicitud de envío) a B, como se muestra en la figura 4.159 (a), esta trama contiene la longitud de la trama que seguirá posteriormente; se puede hacer la trama más corta si se hace que esta longitud tenga un valor fijo. Después B contesta con una trama CTS (Libre para el envío) como se muestra en la figura 4.159 (b) la trama CTS contiene la longitud de los datos copiada de la trama RTS. Una vez que sucede la recepción de la trama CTS, A comienza a transmitir. Cualquier estación que escuche el RTS evidentemente está bastante cerca de A y debe permanecer en silencio el tiempo suficiente para que el CTS se transmita de regreso a A sin conflicto. Cualquier estación que escuche el CTS está bastante cerca de B y debe permanecer en silencio durante la siguiente transmisión de datos, cuya longitud puede determinar examinando la trama CTS . En la Fig. 4.159, C está en el alcance de A, pero no en el de B. Por lo tanto escucha el RTS de A pero no el CTS de B. Mientras no interfiera con el CTS está libre para transmitir mientras se está enviando la trama de datos. En contraste, D está

en el alcance de B pero no de A. No escucha el RTS pero si el CTS . Al escuchar el CTS se le indica que está cerca de una estación que esta a punto de recibir una trama, por lo que difiere el envío de cualquier cosa hasta el momento en que espera la terminación de esta trama. La estación E escucha ambos mensajes de control y, al igual que D, debe permanecer en silencio hasta que haya completado la trama de datos.

A pesar de estas precauciones, aún pueden ocurrir colisiones. Se plantea el uso de una trama ACK (reconocimiento) tras cada trama de datos transmitida exitosamente que provea al transmisor de una confirmación de que la trama se recibió de manera exitosa y no es necesario enviar nuevamente la trama.

Este protocolo supone que las estaciones inalámbricas se encuentran encendidas todo el tiempo de manera que las estaciones cuando no se encuentran transmitiendo se encuentran escuchando el medio. Como en el caso de la red de monitorización los nodos no se encuentran siempre encendidos, antes de iniciar el proceso de transmisión de tramas es necesario que los nodos se escuchen por un tiempo prudencial para asegurar que no se perdieron de escuchar una trama de CTS o una de RTS.

Es necesario utilizar algún tipo de señalización para avisar a los nodos vecinos cuando uno de los nodos se encuentra listo para retransmitir tramas provenientes de nodos que se encuentra más alejados de la base. Por ello se propone el uso de una trama a la que se llamará PT que son las siglas de "petición de transmisión"; esta trama se transmitirá después de que el nodo se despierte, una vez terminado el tiempo prudencial anteriormente mencionado para escuchar las tramas de RTS y CTS. El tiempo que se debe escuchar después de que uno nodo despierta y antes de transmitir la trama de PT no debe ser menor que el tiempo necesario para transmitir la trama más larga posible en el protocolo, a este tiempo se le llamará en adelante B y será un parámetro fundamental para la configuración de la red.

Se decidió diseñar un protocolo en el que la implementación se hace en un solo bloque de lógico de software, en lugar de que el protocolo sea multi-capa. Ello pues el uso de las capas si bien es cierto facilita le proceso de codificación y desarrollo de las redes, también añade un esfuerzo extra en términos de tareas duplicadas y procesamiento de datos que conlleva un mayor gasto energético.

Descripción del protocolo de red propuesto

La conexión de los nodos en una red de sensores inalámbrica es aproximadamente una estrella extendida, en la que la conexión se encuentra condicionada al alcance de la etapa de radiofrecuencia de los nodos; como se muestra en la Fig.4.160

Partiendo de esto se propone una red organizada jerárquicamente en niveles, en la que se pueden definir como nodos de nivel 1 todos los nodos que se encuentren en el alcance de la estación base (zona gris), los nodos de nivel 2 serán los que se encuentren en el alcance de algún nodo de nivel uno y no de uno de nivel cero(zona amarilla) y así sucesivamente.

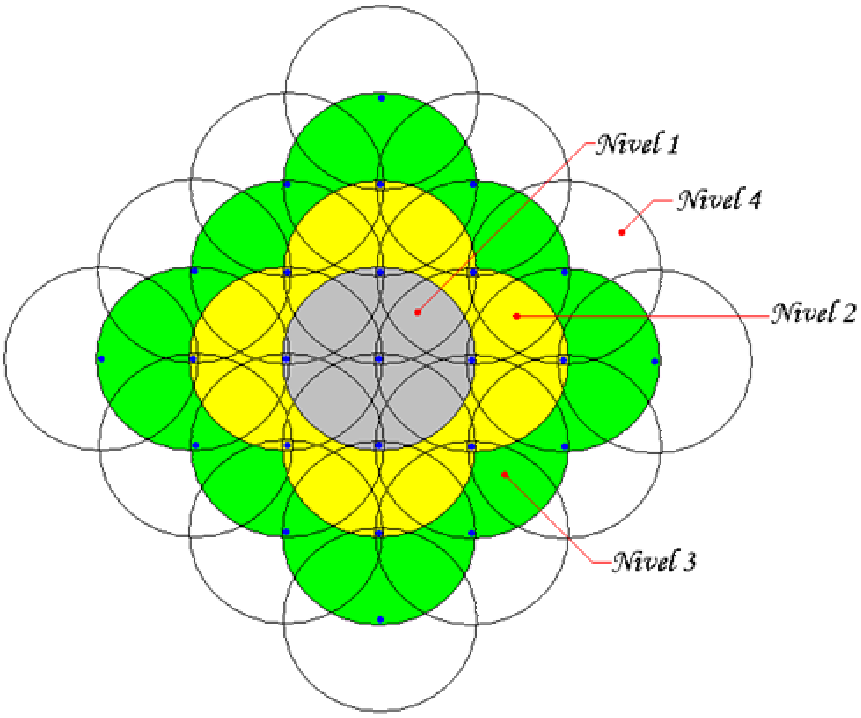


Figura 4.160: Ilustración Conectividad de nodos en una red de sensores. Tomada de[71].

Es necesario que los mensajes transmitidos por un nodo de nivel N pasen en el mejor de los casos por $N-1$ saltos antes de llegar a un sumidero. Además si se pretende cubrir el 100% del área sin dejar regiones sin cubrir en medio de los sensores, la posición geográfica de cada uno de los nodos no dependerá del alcance del módulo de radiofrecuencia del nodo solamente, sino del alcance de los sensores en los nodos.

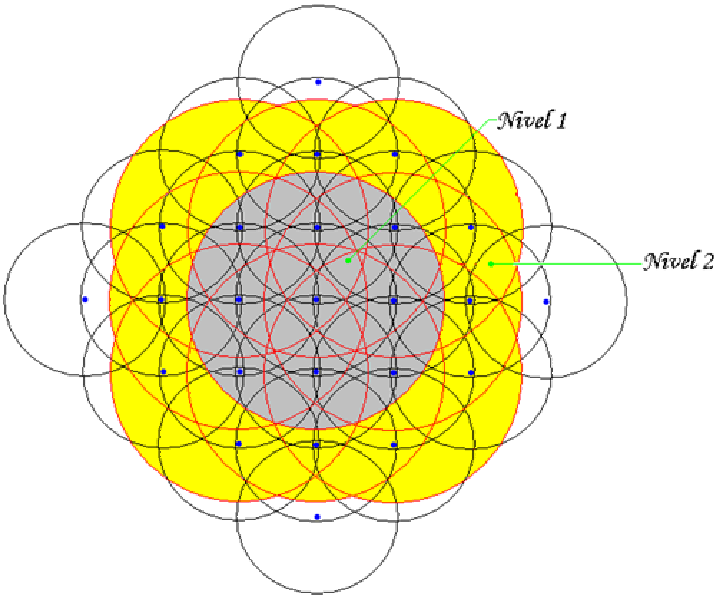


Figura 4.161: Ilustración del caso en que los nodos tienen mayor alcance de comunicación que los sensores tienen alcance. Tomada de[71].

La principal consecuencia de esto es que pueden haber N nodos de nivel k dependiendo de la ubicación que se le asigne a cada uno de los nodos, y pueden haber N nodos en un dominio de colisión dado. Por ejemplo en la Fig. 4.161 los nodos se colocan de acuerdo con el radio de alcance del sensor que abarca un área menor (este radio se representa por medio de los círculos negros) mientras que los niveles de la red siguen siendo definidos en función del alcance del módulo de comunicación (los cuales se representan por medio de los círculos rojos).

El algoritmo propuesto se puede dividir en dos secciones concurrentes. Para simplificar su comprensión se explicarán estas dos partes por separado y luego se unirán para formar el algoritmo completo.

Algoritmo de selección de ruta de datos mezclados (ADM)

Como su nombre lo indica este es el algoritmo con el que se enviarán los datos que ya han sido mezclados hacia los sumideros de la red.

Se propone que cada uno de los nodos pase a través de una serie de estados que contemplan un estado de hibernación para reducir el consumo de energía en el módulo de comunicación; Además se procura que el protocolo tenga la capacidad de configurar de manera automática la arquitectura de la red evitando de esta manera la intervención externa en el proceso de instalación y/o ampliación del sistema.

El protocolo propuesto se describe de forma gráfica en la Fig. 4.162. La función de cada uno de los estados del protocolo se explica detalladamente en [71], desde el punto de vista del algoritmo de selección de ruta de datos mezclados ADM. Las funciones del mismo son en forma resumida:

- Descubrimiento de nivel de pertenencia. El nodo descubre su nivel de pertenencia en la red, basándose en la definición de nivel. En este estado se escucha el medio durante un tiempo suficiente para detectar las tramas PT de todos los nodos cercanos y comparar su nivel en la red. En caso de que el nodo escuche por el tiempo predeterminado para este periodo sin oír una sola trama de PT, el nodo pasará al estado de hibernación para ahorrar energía para pasar nuevamente al estado de descubrimiento de nivel de pertenencia, repitiendo el ciclo hasta que se tenga un nivel de red en el nodo.
- Hibernación. En este estado no se reciben ni se transmiten datos. Este estado finaliza cuando termina el tiempo destinado al mismo. Una vez que finaliza se revisa si existen alarmas por enviar. Si hay alarmas se pasa al primer estado de espera y en caso contrario a la primera fase de petición de trama. Además, tiene un contador que lleva el control de cuántas veces se ha pasado por este estado. Si este contador detecta que el protocolo ha pasado más de X veces por este estado entonces el nodo regresa al estado de descubrimiento del nivel de pertenencia para verificar la arquitectura de la red y redefinir su nivel de pertenencia en la red.
- Primera fase de petición de trama. El nodo escucha por un periodo de tiempo $2B$

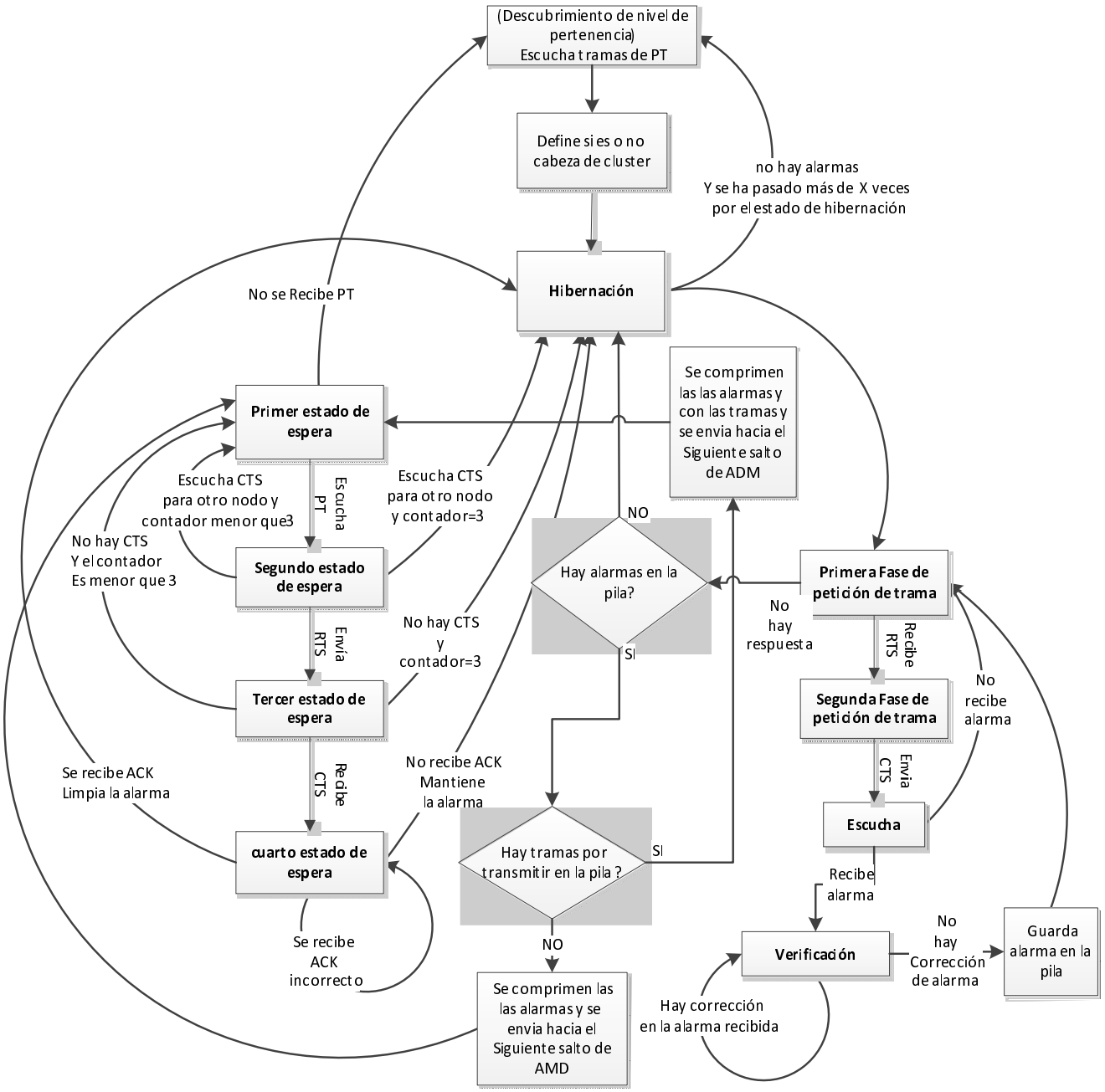


Figura 4.162: Diagrama de flujo del algoritmo multicapa propuesto. Tomada de [71].

para evitar colisiones con tramas que se encuentren en proceso de transmisión por medio de la escucha del canal en busca de tramas RTS o CTS. En caso de no escuchar tramas de CTS o RTS envía una trama PT preguntado a los nodos de nivel superior si tienen una trama para retransmitir en caso de no recibir respuesta el nodo vuelve al estado de hibernación.

- Segunda fase de petición de trama. En caso de que alguna de las tramas PT descritas en el estado anterior sea contestada con una trama RTS, el nodo pasa a la segunda fase de petición de trama; en este estado el nodo envía una trama CTS para informar al primer nodo que envió el RTS que tiene el canal asignado para la transmisión de la trama, los nodos que escuchen el CTS se mantienen en silencio mientras se transmite el mensaje con el objetivo de evitar que las tramas colisionen.
- Escucha. Cuando el nodo envía el CTS, espera para recibir el mensaje del nodo que solicitó el canal; si el tiempo de espera se acaba, el nodo supone que el transmisor tuvo un fallo y retorna al estado de hibernación. Si se recibe la trama a retransmitir, el nodo pasa al estado de verificación.
- Verificación. Se envía una trama de ACK y se espera un tiempo para determinar que la trama se haya recibido correctamente; si el nodo emisor de la trama no corrige el mensaje enviado se supone que el mismo se recibió correctamente y se pasa al primer estado de espera, en caso de que la trama sea corregida se guarda la nueva información de la alarma y se inicia nuevamente el estado.
- Primer estado de espera. El nodo tiene alarmas por transmitir a al base y espera por una trama PT de algún nodo de nivel inferior; en el momento en que dicha trama es escuchada el nodo pasa al segundo estado de espera. Si el nodo escucha una trama PT de un nodo de nivel inferior al inmediatamente inferior al propio, cambia su identificador de nivel para ajustarse a la nueva configuración de la red. En caso de no recibir una trama de PT válida después de un tiempo el nodo asume que hubo un cambio en la red y vuelve al estado de descubrimiento de nivel sin borrar la memoria de alarma. En caso de escuchar un CTS o un RTS el nodo esperará por el tiempo requerido para que la transmisión en progreso sea completada e ignorara cualquier PT que escuche en este tiempo.
- Segundo estado de espera. el nodo aguarda un tiempo aleatorio corto, esto con el objetivo de evitar colisiones con algún otro nodo que desee contestar la transmisión PT, y envía una trama de RTS para luego pasar al tercer estado de espera. En caso de escuchar un CTS proveniente del nodo del que escuchó la trama de PT en el periodo de espera, se incrementa un contador y el nodo regresa al primer estado de espera. Si el contador indica que el nodo ha pasado por este estado 3 veces entonces en lugar de pasar al primer estado de espera el nodo supone que hay muchos nodos que desean transmitir sus tramas y pasa al estado de hibernación sin borrar la alarma.
- Tercer estado de espera. El nodo espera por un CTS dirigido hacia él mismo. En caso de ser recibido el nodo pasa al cuarto estado de espera. En caso de que no reciba la trama CTS el nodo incrementa un contador que indica la cantidad de veces que el nodo ha pasado por este estado y pasa al primer estado de espera. Si el

contador indica que el nodo ha pasado por este estado 3 veces entonces en lugar de pasar al primer estado de espera el nodo asume que hay muchos nodos encendidos y pasa al de hibernación sin borrar la alarma

- Cuarto estado de espera. El nodo envía la trama de alarma y espera por una trama de ACK (de reconocimiento de envío) que indica que la información llegó correctamente al nodo receptor; en caso de no recibirse después de un tiempo de espera B, el nodo pasa al estado de hibernación sin limpiar la memoria de la alarma. En caso de que el nodo reciba la trama de ACK se verifica que la información recibida sea correcta. Si no es así se retransmite la trama y se espera nuevamente el ACK (o sea se vuelve a iniciar el cuarto estado de espera). En caso de que la información sea recibida correctamente se limpia la memoria de la alarma y se pasa al estado de hibernación.
- Nodo base. La estación base solamente escucha para verificar las transmisiones de los nodos de nivel uno y no detecta eventos.

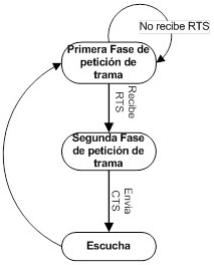


Figura 4.163: Ilustración de los estados seguidos por el nodo base. Tomada de [71].

El diagrama de estados que se sigue en este nodo se muestra en la Fig. 4.163.

Algoritmo agrupamiento y mezcla de datos (AMD)

La naturaleza de la aplicación de la red hace que sea posible que más de un nodo de la red detecte un evento al mismo tiempo, de manera que ante por ejemplo el disparo de un arma de fuego, sean muchos los nodos que deseen informar de la alarma en la misma área geográfica.

Con el objetivo de minimizar la cantidad de datos transitando en la red y por ende reducir el consumo energético, se propuso que los nodos compriman y mezclen la información que se enviará a los sumideros en conglomerados locales. Para lograr lo anterior es necesario definir:

- Un algoritmo para formar los conglomerados locales y la cabeza del conglomerado.
- Un mecanismo para la compresión de los datos recolectados en las cabezas de conglomerados.

El algoritmo debe ser compatible con el protocolo de selección de ruta de datos mezclados, para reducir el uso de tramas de control adicionales. Se propuso entonces que el algoritmo de agrupamiento y mezcla de datos funcionara igual que el algoritmo de selección de

ruta de datos mezclados, con la diferencia que cada una de las cabezas de conglomerado funcionará como un sumidero local en el que se acopiarán las alarmas, y se enviarán los mensajes usando algoritmo de selección de ruta de datos mezclados. El mejor algoritmo de compresión para esta labor deberá ser aquel que se ajuste a las necesidades de la aplicación particular.

Algoritmo para formar los conglomerados locales y seleccionar la cabeza del conglomerado:

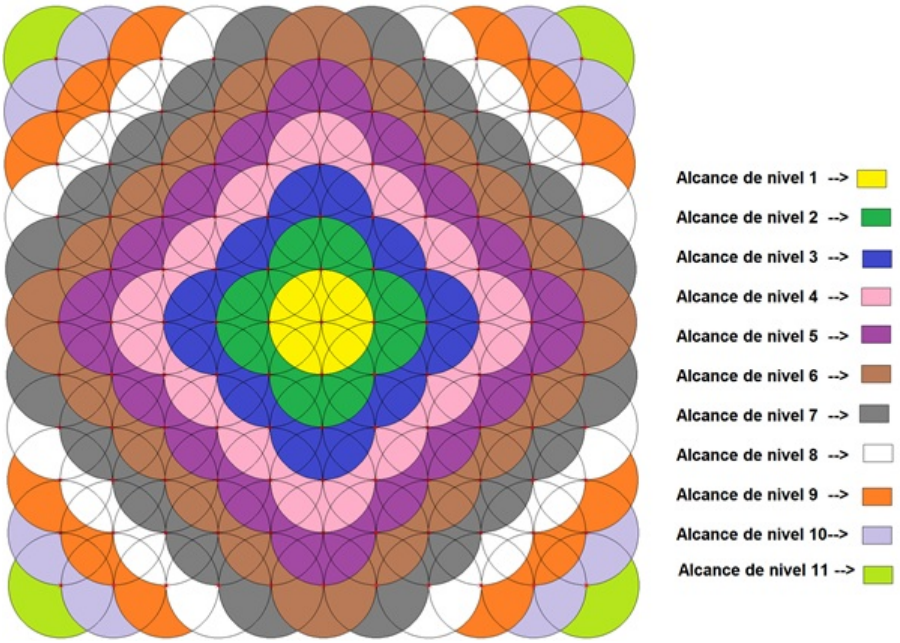


Figura 4.164: lustración de niveles del protocolo de selección de ruta de datos mezclados. Tomada de [71].

Supóngase que se tiene una red en la que los nodos se encuentran posicionados como se muestra en la Fig. 4.164. Lo primero que debe ocurrir en esta red es que cada nodo debe descubrir su nivel en la red.

Una vez que cada nodo conoce su nivel de pertenencia en la red para el algoritmo de selección de ruta de datos mezclados es necesario seleccionar las cabezas de conglomerados, e iniciar el proceso de descubrimiento de nivel de pertenencia en el algoritmo de agrupamiento y mezcla de datos. Para esto es necesario seleccionar un nivel máximo de saltos permitidos en los conglomerados, al cual se le llamará: "$Max\_cluster\_level$" y de aquí en adelante se considerará como parámetro de la red.

Dependiendo de la cantidad de saltos máxima seleccionada se usarán los nodos de "nivel de selección de ruta de datos mezclados" múltiplo de $Max\_cluster\_level \times 2$ como cabezas de conglomerado.

En caso de que una cabeza de conglomerado sea capaz de escuchar a otra cabeza de conglomerado con mayor disponibilidad de carga que la propia y esta tenga una trama por transmitir de tamaño menor que "$limite\_de\_mezcla$", esta retransmitirá la trama hacia

la cabeza de conglomerado con mayor disponibilidad de energía para que sea mezclada con la información por transmitir de esta trama y reducir aun más el tráfico en la red. Es importante destacar que para reducir la latencia de las tramas se establece que las tramas solamente pueden ser retransmitidas una vez.

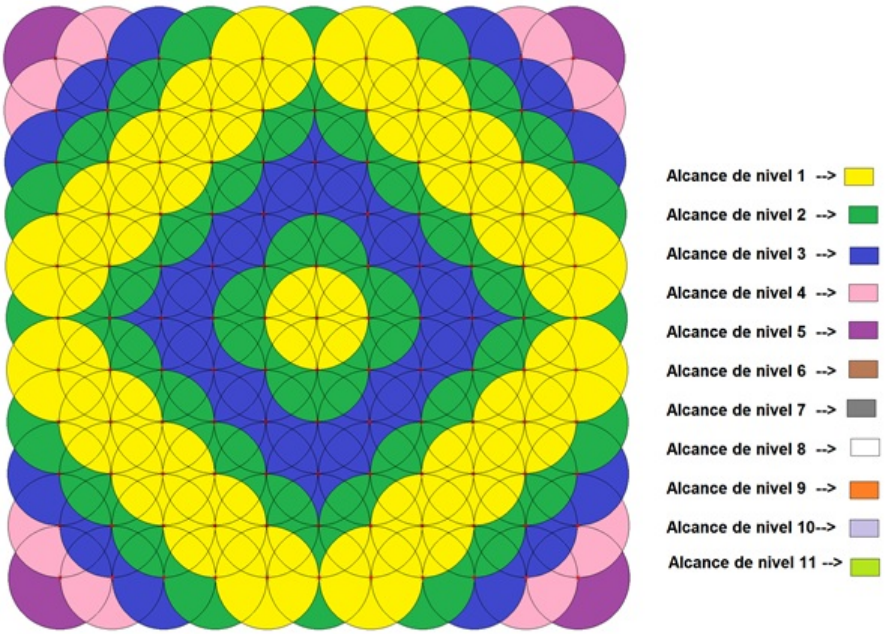


Figura 4.165: Ilustración de distribución de niveles del algoritmo de agrupamiento y mezcla de datos para una red que usa $max\_cluster\_level = 3$. Tomada de [71].

En la Fig. 4.165 se muestra la manera en que se espera se distribuyan los niveles del algoritmo de agrupamiento y mezcla de datos. En caso de que un nodo tenga varios vecinos de nivel inferior en AMD, sr usará para retransmitir al que tenga menor nivel en ADM.

Algoritmo de compresión de datos: Existen en la literatura diversas técnicas de compresión de datos que se pueden utilizar, por lo que se decidió simplemente agrupar el tipo de alarma, seguido por el número de nodos que reportaron la alarma y los identificadores; por ejemplo se asume que los nodos solamente tienen ocho bits de ID y que existen cinco alarmas posibles a transmitir en la red. Esto quiere decir que el campo de tipo de alarma solamente requerirá tres bits, no obstante se utilizarán ocho para dar la opción de agregar hasta 256 diferentes tipos de alarmas el del número de nodos que detectaron la alarma tendrá ocho bits y el de cada uno de los identificadores de nodo tendrá ocho bits respectivamente. En caso de que una cabeza de conglomerado reciba el reporte de cuatro alarmas tipo uno de los nodos 01h, 03h, 05h y 07h, y además reciba dos alarmas de tipo tres de los nodos 01h y 09h, los datos de la trama tendrían el siguiente formato:

01h	04h	01h	03h	05h	07h	03h	02h	01h	09h

En el algoritmo propuesto cada nodo tiene la capacidad de comprimir los datos recibidos y no solamente las cabezas de aglomerado. No obstante, al existir un mecanismo mediante el cual los datos son todos enviados a un punto específico, esto dá más oportunidad para

que los datos sean comprimidos-mezclados y se reducirá la cantidad de tramas transitando la red, reduciendo de esta forma la probabilidad de colisión.

Algoritmo multicapa propuesto

Sumando lo expuesto en las secciones 4.6.2 y 4.6.2; el algoritmo es descrito en la Fig. 4.162 y a continuación se detallan los estados que tienen diferencias con respecto de lo expuesto en la sección donde se explican los estados del ADM:

Descubrimiento de nivel de pertenencia: En esta etapa se hará lo mismo que se explicó anteriormente solamente que el nodo descubre su nivel de pertenencia en la red tanto para ADM como para AMD; para encontrar su nivel de AMD simplemente se observa el nivel de ADM del nodo una vez que se obtiene, si el nivel es múltiplo de $max\_cluster\_level \times 2$ este se ubicará como cabeza de grupo, o sea en el nivel uno de AMD, en caso contrario el nodo calculará la diferencia entre nivel de AMD del nodo y el múltiplo de $max\_cluster\_level \times 2$ más cercano y se ubicará en el nivel $diferencia + 1$. Por ejemplo: si $max\_cluster\_level$ es igual a tres, y el nivel de ADM del nodo es igual a siete, los múltiplos de $max\_cluster\_level \times 2$ más cercanos son seis y doce, con una diferencia de uno y cinco, por tanto el nodo tomará un nivel de AMD igual a dos.

Primera fase de petición de trama: En este estado el nodo envía una trama preguntado a los nodos de nivel superior si tienen una trama para retransmitir (de aquí en adelante se le llamará a esta trama PT "petición de transmisión"). En caso de no recibir respuesta, el nodo verifica si hay alarmas en la pila de alarmas y/o tramas en la pila de tramas (la diferencia entre una trama y una alarma es que las tramas se re-transmiten usando ADM mientras que las alarmas se retransmiten usando AMD). En caso de existir tramas y alarmas, se procede a comprimir las alarmas y las tramas en un solo paquete hasta alcanzar el tamaño máximo permitido por el protocolo. En caso de que la trama resultante exceda dicho tamaño se crearán dos o más tramas para acomodar todos los datos en una cantidad mínima de tramas. Una vez que los paquetes resultantes están listos, se guardan en la pila de tramas y se pasa al primer estado de espera. En caso de que solo existan alarmas, estas se comprimirán y se enviarán usando AMD en el primer estado de espera. En caso de que no hayan ni tramas ni alarmas en la pila se retornará al estado de hibernación.

4.6.3 Diseño de las tramas usadas en el protocolo

Para el diseño de las tramas se procura que las mismas tengan una cantidad mínima de bits, para minimizar el tiempo que duran las transmisiones y reducir la probabilidad de colisión así como el consumo de energía.

Cada uno de los espacios de información dentro de las tramas tendrá una longitud predefinida, con excepción de las tramas de alarma, que tendrán una longitud variable.

Para efectos del presente ejemplo se supone un byte para representar la mayoría de los

campos en las tramas, pero estos tamaños pueden modificarse para mejorar las características de la red; por ejemplo, si se sabe que solamente se tendrán 8 tipos de tramas en la red entonces no necesario utilizar 8 bits para la cabecera de las tramas sino solamente 3 bits.

Las tramas requeridas para la implementación del protocolo anteriormente descrito son las siguientes:

Trama de PT: Esta trama contiene el nivel del nodo emisor con el objetivo de que los nodos que la escuchen puedan verificar que la misma corresponde a un nodo más cercano a la base que ellos mismos. Además la trama debe contener la dirección MAC del emisor para que el receptor sepa a quién debe hacer el RTS.

F1h Nivel_del_emisor_AMD Nivel_del emisor_ADM Dirección_MAC_del_emisor

Trama de RTS: La trama de RTS debe contener la dirección MAC del emisor para que el nodo que la escuche la incluya en la trama del CTS y este sepa que su solicitud fue aceptada. Además, debe contener el nivel del nodo emisor para facilitar el proceso de búsqueda de identidad de los nodos vecinos y el largo de la trama que se desea transmitir para que en caso de ser escuchado por algún nodo vecino, este sepa el periodo por el cual debe esperar antes de intentar enviar su trama.

F2h Nivel_del_emisor_AMD Nivel_del_emisor_ADM Tamaño_trama_por_enviar Dirección_MAC_del_emisor

Trama de CTS : La trama CTS es una trama de difusión y por esto debe contener la dirección MAC del nodo a la que va dirigida para que los demás nodos identifiquen a quien se dirige y guarden silencio en caso de que no sea para ellos

F3h Nivel_emisor_AMD Nivel_emisor_ADM Tamaño_trama_por_enviar Dirección_MAC_destinatario

Trama de alarma: Esta trama debe dejar claro el lugar de procedencia y su tipo. Es por esto que debe contener el código de la alarma así como la dirección MAC del nodo en el que se produjo. Además debe estar clasificada si debe ser enviada usando ADM o AMD. En caso de que el nodo tenga el reporte de cuatro alarmas tipo uno de los nodos del 01h, 03h, 05h y 07h, y además tenga en cola dos alarmas de tipo tres de los nodos 01h y 09h, los datos de la trama tendrían el siguiente formato:

F4h	01h	04h	01h	03h	05h	07h	03h	02h	01h	09h	F4h	CHECK_SUM

El campo que se asigna al código de la alarma es de un byte, con apacidad de comunicar 256 diferentes tipos de información. Se definen cinco tipos de alarmas por el momento, las cuales se muestran en la tabla 4.62.

Código del tipo de alarma	Alarma asignada
0x04	Salida de operaciones del nodo
0x03	Fuego
0x02	Motosierras
0x01	Disparos
0x00	Inicio de operaciones del nodo

Tabla 4.62: Descripción de los tipos de alarmas asignadas hasta el momento. Tomada de [71].

La salida de operación del nodo se define como una alarma generada por el módulo de alimentación e indica que el nodo se encuentra en un límite de carga que hace necesario apagar el nodo para que se cargue hasta un nivel que permita su operación normal o si no es posible recargar el nodo entonces para informar a la base que el nodo saldrá de operación debido a carga insuficiente. Para indicar si la alarma debe transmitirse usando ADM o AMD se utilizarán 2 cabeceras diferentes: F4 para ADM y F6 para AMD.

Trama de ACK: Esta trama debe dar la capacidad al nodo de saber si el dato fue recibido correctamente.

0xF5 CHECK_SUM Dirección_MAC_de_la_procedencia

Se añade un campo para un $CHECK\_SUM$ que para la implementación propuesta tendrá 8 bits y será simplemente los últimos 8 bits de la suma de los bits individuales del campo de la información de la alarma, o sea, desde el byte de cabecera hasta antes del byte de $CHECK\_SUM$. Se selecciona el anterior código de $CHECK\_SUM$ por simplicidad, no obstante queda al criterio del implementador del mismo el uso de algún otro método para validar que el dato se recibió correctamente.

4.6.4 Diseño de los tiempos del protocolo

Para definir los tiempos del protocolo es necesario tener alguna referencia en cuanto al consumo energético que un nodo de comunicación tendrá en escucha pasiva, transmisión y en estado de hibernación, además de otros parámetros básicos que serán dependientes de la capa física sobre la cual se implemente la red. Dado que al momento de realizar estas pruebas, no se habían definido los nodos físicos que se utilizarían para la implementación del presente proyecto en la red de sensores de monitorización del bosque, y que para efectos del diseño del protocolo se pretende obtener resultados independientes de la capa física, se tomó como referencia el consumo energético de los módulos TR1000 usados también como referencia en [79]. Los datos de consumo de potencia obtenidos de estos nodos se utilizarán solamente para obtener datos de referencia que posteriormente deberán ser repetidos usando datos que correspondan con la capa física que se quiera usar para la implementación de la red. Posteriormente, será cuestión de realizar estos cálculos sobre la red ya diseñada en la sección 4.6.1.

La base de tiempo B: Este es el tiempo que se utilizará como base para el cálculo

del resto de los tiempos y es igual a la mitad del tiempo necesario para que la trama más grande del protocolo sea transmitida a la máxima distancia de alcance de la red, que sea escuchada y retransmitida hacia la primera estación nuevamente. Es importante mencionar que este tiempo es directamente dependiente de la capa física que se utilice en la implementación del protocolo, por lo que será necesario medirlo cada vez que se cambie la tecnología utilizada en esta capa. Para el caso de los nodos de referencia descritos anteriormente este tiempo es de aproximadamente 58ms; esto sabiendo que la trama potencialmente más larga del protocolo es la trama de alarma. Suponiendo arbitrariamente que el parámetro de red "*Limite_de_mezcla*" es igual a 10, o sea que la cantidad máxima de alarmas que se pueden mezclar en una trama son 10, la trama más larga posible en el protocolo será de 19 bytes, o sea, 4 bytes del tipo de alarma, asumiendo que se presenten las 4 alarmas posibles, 4 bytes para el número de alarmas de cada tipo que hay representadas en la trama, 1 byte para el encabezado y 10 bytes para los identificadores de los nodos que emitieron cada una de las alarmas.

Tiempo de hibernación: Este tiempo influye de manera directa en la cantidad de energía consumida en la etapa de comunicación de los nodos. Entre más grande sea el tiempo destinado en este estado, mayor será el ahorro en la energía del protocolo, no obstante el atraso en el proceso de envío de las tramas aumentará.

Estado	Potencia promedio
Escucha	12,5 mW
Transmisión	14,8 mW
Sueño profundo	0,016 mW

Tabla 4.63: Potencia promedio del TR1000 en los diferentes estados de la etapa de comunicación. (Tomada de [79])

El consumo promedio de potencia de los módulos de comunicación se describe en la tabla 4.63. Tomando en cuenta que en un protocolo en el que mientras la red se encuentra en reposo el módulo de comunicación se encuentra en estado de escucha se tiene que el consumo de potencia en reposo es constante, y la potencia promedio será de 12,5 mW.

En el caso de la red en reposo con el protocolo implementado el consumo de potencia no será constante debido a que el módulo de comunicación pasará por diferentes estados a lo largo de cada uno de los ciclos de petición de trama e hibernación. Así, el consumo de energía en cada uno de los ciclos puede describirse aproximadamente como se muestra en la Fig. 4.166 donde la potencia promedio es:

$$P_{promedio}\,(mW) = \frac{0,016 \times T + 12,5 \times (P - 0,013) + 14.88 \times 0,013}{T + P} \tag{4.52}$$

Puede verse que la potencia consumida será dependiente de la proporción entre los tiempos P y T, si la red se encuentra en reposo y se gasta un tiempo máximo aproximado de P $(2B)$ en la primera fase de petición de trama, el porcentaje de ahorro de energía en el nodo con respecto de un protocolo en el que la etapa de radio (transmisión, recepción)

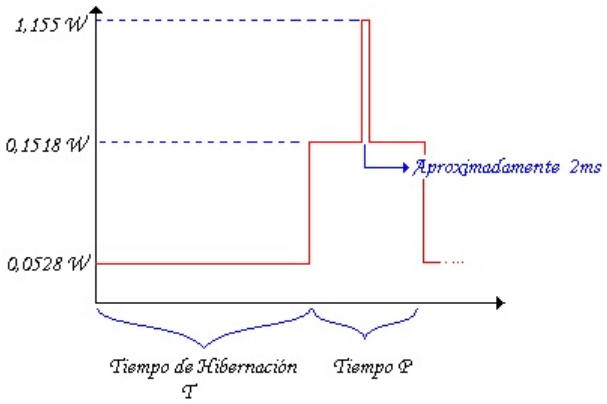


Figura 4.166: Gráfico aproximado del consumo de potencia esperado de los nodos con la red en reposo utilizando el protocolo propuesto. Tomada de [71].

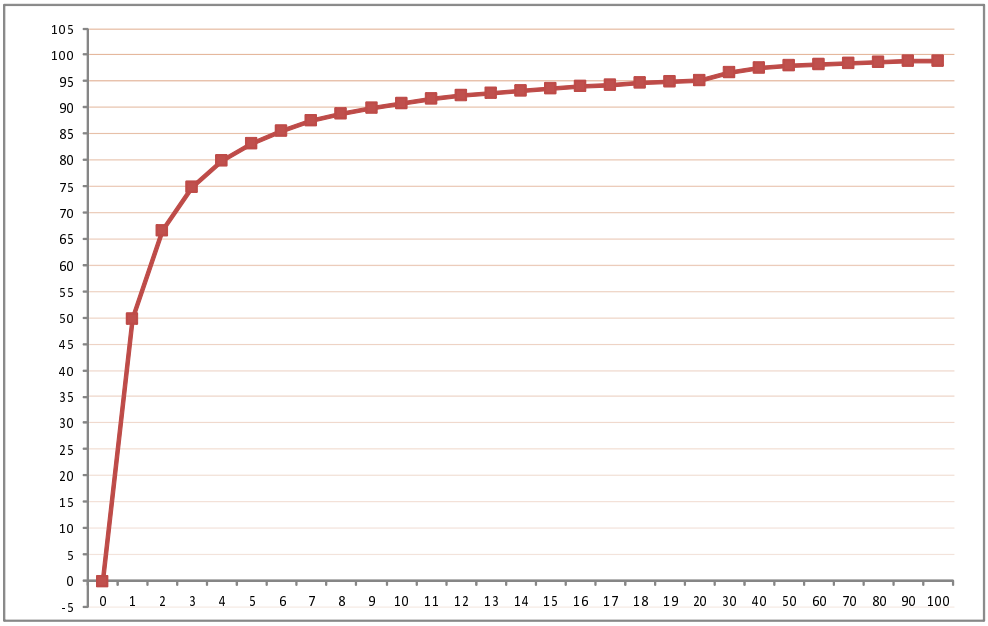


Figura 4.167: En el eje vertical se ve el porcentaje de ahorro en la energía para la red en reposo en la etapa de comunicación con respecto de un protocolo en el que esta etapa se mantiene constantemente encendida, mientras que en el eje horizontal se ve que tan grande es el tiempo T con respecto a P. Tomada de [71].

debe mantenerse permanentemente encendida, se comportará aproximadamente como se muestra en la Fig. 4.167 conforme aumenta el tiempo de hibernación

Se puede notar que el porcentaje de ahorro en la energía no cambia de manera drástica después de que el tiempo de hibernación alcanza un valor de 10P, donde se alcanza un ahorro energético de aproximadamente 90%.

Este ahorro energético se ve acotado en aproximadamente 99,87%, ya que para este caso la diferencia entre la potencia consumida en la etapa de sueño profundo y la potencia consumida en escucha pasiva es de 12,48mW, de forma que conforme aumente el tiempo de hibernación la potencia consumida en el nodo tenderá a la potencia consumida con la

etapa de transmisión apagada.

Para una red con N niveles en los que se programa un tiempo de hibernación T y la suma de todos los demás tiempos de espera es K, sin retransmisiones se tendrá que pasar como máximo por un proceso de agrupamiento de datos hacia una cabeza de conglomerado. En el peor de los casos se debe sumar al número de saltos $Max\_cluster\_level - 1$, tomando en cuenta que cada uno de los nodos opera sin sincronización. El tiempo máximo (en el peor de los casos) entre la detección de una alarma y su recepción en la estación base será:

$$T_{max} = ((N-2) + (Max\_Cluster\_level - 1)) \times (T + K) + T + T_{base} \tag{4.53}$$

donde T_{max} es el tiempo de latencia máximo y T_{base} es el tiempo de ciclo base. Dado que el tiempo más grande contemplado en la red es el de hibernación, se tiene que el tiempo entre la emisión de una alarma y su recepción en la estación base será claramente afectado por el mismo.

Descubrimiento de nivel de pertenencia: El tiempo que se escucha en este estado debe ser lo suficientemente largo como para que el nodo escuche las tramas PT de todos sus vecinos y al mismo tiempo no debe ser más largo de lo necesario para que no se desperdicie energía.

Si se supone que la red se encuentra en reposo y ya ha sido configurada, entonces cada uno de los nodos transmitirá una trama PT aproximadamente cada vez que se termina el tiempo de hibernación; para asegurar que los nodos tengan el tiempo suficiente para escuchar las tramas PT de todos sus vecinos se asigna en este estado un tiempo de espera máximo del doble del tiempo de hibernación, o sea $2T$, en caso que después de $2T$ el nodo no escuche ninguna trama de PT entonces el nodo pasará al estado de hibernación y volverá al estado descubrimiento de nivel de pertenencia al terminar la hibernación.

Primer estado de espera: En este estado, al igual que en el estado de descubrimiento de nivel es necesario esperar tiempo máximo suficiente para escuchar las PT de sus vecinos o sea $2T$ si después de este tiempo no se escucha ninguna trama de PT válida, entonces se pasará al estado de descubrimiento del nivel de pertenencia.

Segundo estado de espera: En este estado se espera un tiempo aleatorio con el propósito de evitar colisiones después de haber escuchado una trama de PT de un nodo de nivel inferior. El tiempo aleatorio que se espera antes de enviar la trama debe ser lo más corto posible para que no se gaste mucha energía, no obstante entre más sean los posibles tiempos de transmisión en este estado, menor será la probabilidad de colisión .

Este estado influye directamente en el tiempo que debe ser asignado en la espera entre cada una de las tramas de PT de la primera fase de petición de trama y con esto influye directamente en el tiempo que debe ser asignado al estado de hibernación y al tiempo de retraso total de las tramas transmitidas.

Tomando en cuenta todo lo anterior este tiempo se define entre cero y $8B$, de esta forma

el nodo tiene cinco tiempos posibles en los cuales puede intentar transmitir su trama de RTS; en $t = 0$, $t = 2B$, $t = 4B$, $t = 6B$ y $t = 8B$ esto con el objetivo de que si otro nodo transmite en un tiempo menor el nodo que se encuentra esperando tenga tiempo suficiente para escuchar la trama de CTS y evite la colisión.

Primera fase de petición de trama: En este estado es necesario implementar un tiempo de espera antes de trasmitir la trama de PT para evitar que la trama corrompa alguna transmisión en proceso por no haber escuchado otras tramas de PT, CTS o RTS. Este tiempo debe ser aproximadamente $2B$. Una vez que se transmite la trama de PT la estación debe esperar por $9B$ en el peor de los casos para dar tiempo a que la estación que desea reenviar su alarma transmita su RTS. Se puede notar que el tiempo P anteriormente usado para modelar el ahorro energético utilizando estados de hibernación será de aproximadamente $11B$ más el tiempo requerido para transmitir la trama de PT. En el caso de los nodos de referencia, el tiempo requerido para la transmisión de la trama será de aproximadamente 4,5ms, lo cual es despreciable si se compara los 58ms de la base de tiempo B por lo que de aquí en adelante se dirá que el tiempo P es de $11B$.

Estado de escucha: En este estado se espera un tiempo máximo de $2B$ después de haber enviado la trama de CTS para recibir el mensaje del nodo al que se le notificó, o para volver al estado de hibernación.

Verificación: En este estado se espera un tiempo máximo de $2B$ antes de dar por aceptado como correcto el mensaje recibido después de enviar la trama de ACK

Tercer estado de espera: Al igual que en el caso anterior el nodo esperará $2B$ para recibir un CTS.

Cuarto estado de espera: En este estado el nodo espera un tiempo $2B$ para recibir la trama de ACK.

En [71] se ofrece un análisis teórico del algoritmo propuesto que no se detalla en este informe. En ese análisis se profundiza sobre las probabilidades de retardo y falla del protocolo propuesto.

4.6.5 Desarrollo del ambiente de simulación del protocolo

Desarrollo de la Simulación del medio

Uno de los retos que es necesario resolver cuando se trabaja con un simulador basado en eventos como Omnet++ al simular redes inalámbricas de sensores, es el modelado del medio inalámbrico en sí mismo. En un medio inalámbrico se debe tomar en cuenta que la conexión entre los sensores va a depender de la potencia de transmisión del nodo que emite el mensaje, de la posición de los nodos, tanto los que escuchan como los que transmiten, y de la geografía en la que se encuentran ubicados, ya que es posible que existan obstáculos que no permitan la transmisión entre nodos colocados a una distancia pequeña. Además es necesario tomar en cuenta que es posible que exista colisión de

```
package org.mixim.examples.baseNetwork;

import org.mixim.base.connectionManager.ConnectionManager;
import org.mixim.base.modules.BaseWorldUtility;

module BaseNetwork
{
    parameters:
        double playgroundSizeX  @unit(m); // x size of the area the nodes are in  (in meters)
        double playgroundSizeY  @unit(m); // y size of the area the nodes are in  (in meters)
        double playgroundSizeZ  @unit(m); // z size of the area the nodes are in  (in meters)
        double numNodes; // total number of hosts in the network

submodules:
        connectionManager : ConnectionManager;
        world: BaseWorldUtility {
            parameters:
                playgroundSizeX = playgroundSizeX;
                playgroundSizeY = playgroundSizeY;
                playgroundSizeZ = playgroundSizeZ;
        }
        node[numNodes]: BaseNode;
```

Figura 4.168: Fragmento de código de un módulo de definición de red en Mixim (Tomado de [74]). Tomada de [71].

tramas cuando alguno de los nodos que reciba una transmisión, se encuentre en el rango de escucha de otra transmisión en el mismo momento. El nodo emisor no necesariamente será capaz de reconocer que ocurrió una colisión.

Se propone para la simulación del medio de transmisión en el presente trabajo el uso de Mixim[74] Mixim es una biblioteca de modelado para Omnet++, creada para redes inalámbricas fijas y móviles (redes inalámbricas de sensores, redes vehiculares, etc). Ofrece modelos detallados de propagación de ondas, estimación de interferencias, consumo del tranceptor, y protocolos de acceso al medio. Una red MiXiM básica se compone de los siguientes elementos:

- Módulo de definición de la red
- Módulo de definición del nodo
- Módulo de interfaz de red, que define la capa de red del nodo
- La configuración de la capa física, modelo analógico (cálculo de atenuación) y *decider* (clasificación de ruido y cálculo de errores de bit).
- La configuración de la simulación

En la Fig. 4.168 se muestra un fragmento de código para la definición de un módulo de red en mixim. En la definición de red se incluyen parámetros de definición de las dimensiones del terreno, los submódulos de utilidad global y de administración de conexiones, y el tipo de nodo que compone la red. El administrador de conexiones verifica si dos nodos pueden oírse uno al otro y actualiza sus conexiones de acuerdo a eso. Si dos nodos están conectados significa que pueden recibir algo de cada uno, pero no significa que pueden entenderse. Si dos nodos no están conectados, están fuera del rango de alcance y no recibirán ninguna señal uno del otro. Se debe diferenciar el rango de alcance del modelo

```
package org.mixim.examples.baseNetwork;

import org.mixim.base.connectionManager.ConnectionManager;
import org.mixim.base.modules.BaseWorldUtility;

module BaseNetwork
{
    parameters:
        double playgroundSizeX  @unit(m); // x size of the area the nodes are in  (in meters)
        double playgroundSizeY  @unit(m); // y size of the area the nodes are in  (in meters)
        double playgroundSizeZ  @unit(m); // z size of the area the nodes are in  (in meters)
        double numNodes; // total number of hosts in the network

submodules:
        connectionManager: ConnectionManager;
        world: BaseWorldUtility {
            parameters:
                playgroundSizeX = playgroundSizeX;
                playgroundSizeY = playgroundSizeY;
                playgroundSizeZ = playgroundSizeZ;
        }
        node[numNodes]: BaseNode;
```

Figura 4.169: Fragmento de código del módulo *decider* en Mixim (Tomado de [74]).

de comportamiento de la radio. El rango de alcance define la distancia a partir de la cual un nodo alejado deja de existir para el tranceptor. El modelo analógico y el *decider*, junto con los parámetros del módulo de interfaz de red, determinan la intensidad de señal combinada y si una trama es recibido o no.

El *decider* y los modelos analógicos se definen con un archivo xml que configura parámetros propios. En el módulo de interfaz se debe especificar qué archivo contiene esta configuración.

En el fragmento de código en la Fig. 4.169 es un ejemplo del contenido del archivo xml. En este ejemplo, se declara un modelo analógico de tipo $Break-pointPathlossModel$, que representa el modelo de atenuación definido en el estándar IEEE 802.15.4. Como puede observarse, es posible declarar más de un modelo analógico. Luego se declara el tipo de *decider*, Decider802154Narrow, que clasifica la señal en bits o ruido, según el modelo BER definido también en el estándar IEEE 802.15.4.

En la tabla 4.64 se enumeran algunos de los módulos de MiXiM utilizados para modelar la red de referencia.

Modelado de los Nodos

Los nodos de la red se modelan usando los siguientes submódulos:

battery: Permite definir una capacidad inicial de batería, y que la radio pueda consumir energía descontando de esa capacidad. **nic:** Define el comportamiento de la interfaz de red. Es a su vez un módulo compuesto por el módulo de capa de enlace y el módulo de capa física. El modelo de interfaz utilizado es IEEE 802.15.4-2006 y está implementado

Módulo	Propósito
WSNRouting	Definición de red para simulación de redes inalámbricas de sensores
host802154_2400MHz	Definición de nodo que utiliza un tranceptor 802.15.4 a 2,4 GHz.
BatteryStats	Módulo de recolección de estadísticas sobre la batería.
SimpleBattery	Modelo simple de batería.
StationaryMobility	Administrador de información básica de movilidad y posición.
Nic802154_TI_CC2420	Modelo de interfaz de red Texas Instruments CC 2420 IEEE 892.15.4 CSMA.
CSMA802154	Enlace IEEE 802.15.4-2006 CSMA no ranurado
PhyLayerBattery	Capa física que consume batería

Tabla 4.64: Algunos de los módulos de Mixim usados en la simulación de la red

por el módulo ic802154 TI CC2420, descrito más adelante. **net:** Define el protocolo de red que utiliza el dispositivo. En este trabajo se desarrolló un nuevo módulo que incluye la capa de red, la capa de control de acceso al medio para el protocolo propuesto, mientras que para el nodo de referencia se utiliza el protocolo de red wiseroute. **app:** Define el comportamiento de la aplicación que utiliza los servicios de red. Para el caso de las pruebas realizadas en el presente proyecto solamente se usa esta capa para simular el tráfico que circula por la red.

Interfaz de red IEEE 802.15.4: El módulo Nic802154 TI CC2420 implementa una interfaz de red Texas Instruments CC 2420 802.15.4 usando el protocolo de enlace CSMA especificado en el estándar IEEE 802.15.4-2006. El modelo CSMA802154 fue validado independientemente en una red inalámbrica de sensores de prueba[75], aunque la validación fue realizada con una cantidad de nodos muy pequeña.

Capa física: El módulo de capa física permite configurar la sensibilidad y potencia de transmisión. Para el tranceptor cc2024 la máxima potencia de transmisión es 1 mW [42] y esta es la que se utilizó en las simulaciones. No se ha podido hallar el alcance de la radio en la hoja de datos del tranceptor[42]. En la documentación del nodo MICAz de la compañía MEMSIC[53], que está basado en el mismo estándar y tiene características técnicas muy similares, se especifica que el alcance del tranceptor es de 75 a 100 m en exteriores, y de 20 a 30 m en interiores. Se configuró un valor de sensibilidad de -95 dBm [42]. El módulo de capa física utilizado calcula la atenuación utilizando el modelo establecido en la sección E.5.3 del estándar IEEE 802.15.4-2006[35]. Esta capa utiliza el módulo *decider* Decider802154Narrow para filtrar las señales recibidas según su intensidad, calcular los errores de bits y clasificar las señales en ruido o paquetes. Para las simulaciones realizadas

```
class PhyLayerBattery : public PhyLayer {

enum Activities {
SLEEP_ACCT=0,
RX_ACCT,
TX_ACCT,
SWITCHING_ACCT,
DECIDER_ACCT,
};
...
```

Figura 4.170: Fragmento de código del módulo de batería en Mixim (Tomado de [74]). Tomada de [71].

se deshabilitó el ruido térmico.

Modelo de batería: La batería es un módulo independiente, y define cuál es la capacidad inicial y el voltaje. La capa física de la interfaz Nic802154 TI CC2420, PhyLayerBattery, descuenta energía de la batería según tipos de actividades definidas: dormir, recepción, transmisión, cambio de modo, y actividad del *decider*. El fragmento de código de la Fig. 4.170 muestra la definición de las posibles actividades en el módulo.

En general el modelo usado para las implementación de los nodos fue como se muestra en la Fig. 4.171.

4.6.6 Resultados y análisis de las pruebas del protocolo

Para el diseño de las pruebas que se utilizaron en la caracterización del protocolo propuesto en el presente trabajo, se debe tomar en cuenta que los parámetros de latencia, consumo de potencia, tiempo de configuración y colisiones van a depender no solamente de los factores de configuración del protocolo sino también de factores externos al mismo como lo son: la topología física de la red, el tiempo de ocurrencia de las alarmas, los nodos que detecten las tramas en un área geográfica determinada, y los posibles fallos en los nodos.

Dado lo anterior, si se decide por ejemplo probar la latencia las tramas usando mil escenarios diferentes en los que se usan parámetros aleatorios de frecuencia de generación de las alarmas, topología, nodos de ocurrencia de las alarmas y cantidad de nodos en la red; se van a tener diferencias en los resultados de un escenario al siguiente, debido a que el cambio de alguno de los parámetros anteriormente descritos puede causar cambios drásticos en las condiciones de la red, por ejemplo crear cuellos de botella para el envío de mensajes por algún nodo en particular entre muchos otros casos posibles, y la cantidad de configuraciones posibles para probar es infinita. Es por esto que dependiendo del parámetro de la red que se desee caracterizar se van a seleccionar parámetros específicos que se consideren más relevantes para modificar en cada uno de los escenarios, y no todos al mismo tiempo, de manera que se pueda obtener información relevante para cada uno

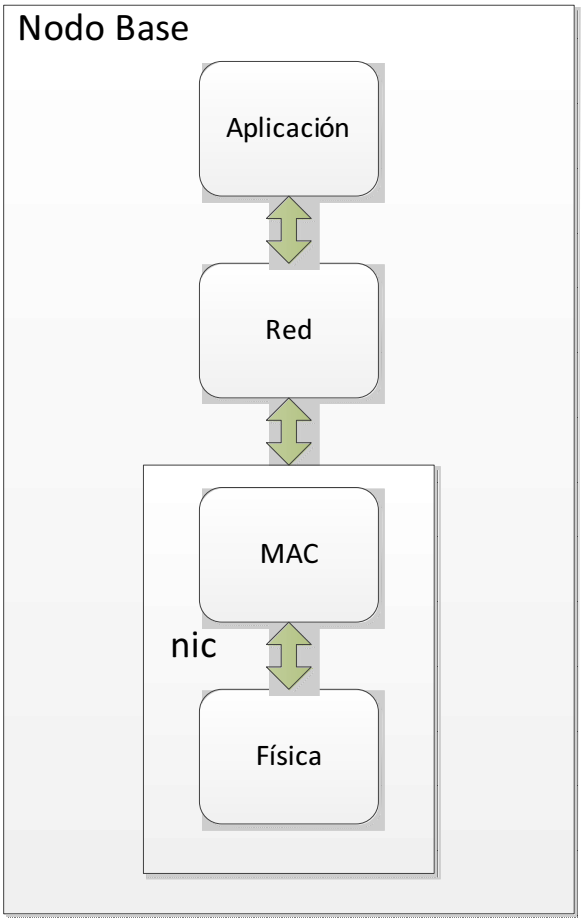


Figura 4.171: Módulos usados en la implementación de los nodos en Omnet++. Tomada de [71].

de los casos.

Los principales parámetros que se desean caracterizar basados en los objetivos específicos que se plantearon fueron:

- Consumo energético de los nodos de la red con respecto de los nodo en una red en la que se implemente el estándar IEEE802.15.4.
- Tiempo de latencia de las tramas
- Comparar los resultados de la simulación del consumo energético de los nodos y de la latencia de las tramas con respecto del modelado teórico expuesto anteriormente para estos parámetros
- Capacidad de autoconfiguración de la red ante fallos en los nodos.

Parámetro de Mixim	Descripción	Valor
sleepCurrent	Corriente promedio del nodo en estado de hibernación	0,021 uA
rxCurrent	Corriente promedio del nodo en estado de escucha	17,4 mA
txCurrent	Corriente promedio del nodo en estado de transmisión	18,8 mA
setupRxCurrent	Corriente promedio del nodo mientras se pasa al estado de escucha	0,6391mA
setupTxCurrent	Corriente promedio del nodo mientras se pasa al estado de	0,6845mA
phy.headerLength	Tamaño del encabezado de la capa física	48 bit
phy.timeSleepToRX	Tiempo necesario para pasar de hibernación a escucha	0,001792 s
phy.timeSleepToTX	Tiempo necesario para pasar de hibernación a transmisión	0,001792 s
phy.timeRXToTX	Tiempo necesario para pasar de escucha a transmisión	0,000192 s
phy.timeTXToRX	Tiempo necesario para pasar de transmisión a escucha	0,000192 s
phy.timeRXToSleep	Tiempo necesario para pasar de escucha a hibernación	0 s
phy.timeTXToSleep	Tiempo necesario para pasar de escucha a hibernación	0 s

Tabla 4.65: Resumen de los parámetros de la etapa de radiofrecuencia usados por defecto en las pruebas. Tomada de [71].

Los parámetros utilizados para cada una de las pruebas excepto en los casos en los que así se especifique se muestran en las tablas 4.65 y 4.66.

A continuación se describe el procedimiento utilizado para cada una de las pruebas reali-

Parámetro de la Red	valor usado
limite_de_mezcla	100 alarmas
B	58ms
T	$44B = 4P$
X	10
$Max\_cluster\_level$	3

Tabla 4.66: Resumen de las parámetros del protocolo usados para las pruebas

zadas, así como los resultados obtenidos.

Capacidad de autoconfiguración de la red ante fallos en los nodos

En la primera prueba para comprobar la capacidad de autoconfiguración de la red, se crearon cien escenarios en los que se ponen sesenta nodos de forma aleatoria en un área de simulación de $25000000m^2$ (5000m en X y 5000m en Y) como se muestra por ejemplo en la figura4.172. Para todos los casos la base es el nodo[0].

Para la realización de esta prueba se configuró la red como se muestra en la tabla 4.66.

Tiempo (Minutos)	Evento
0	Inicio de la red
5	se apagan nodos 1 y 59
7	se apagan nodos 2 y 58.
9	se apagan nodos 3 y 57.
11	se apagan nodos 4 y 56.
13	se apagan nodos 5 y 55.
15	se apagan nodos 6 y 54.
17	se apagan nodos 7 y 53.
19	se apagan nodos 8 y 52.
21	se apagan nodos 9 y 51.
23	se apagan nodos 10 y 50.
25	se encienden todos los nodos de nuevo.
100	finaliza la prueba y se cuentan las alarmas que llegan al sumidero

Tabla 4.67: Tabla de resumen de la prueba de capacidad de autoconfiguracion y efectividad en la entrega de tramas. Tomada de [71].

En la prueba todos los nodos se encienden en tiempos aleatorios desde cero hasta dos minutos de la simulación, esto con el objetivo de aleatorizar el estado de los nodos de una forma similar a como sería en ambientes reales, y se empiezan a generar alarmas en todos

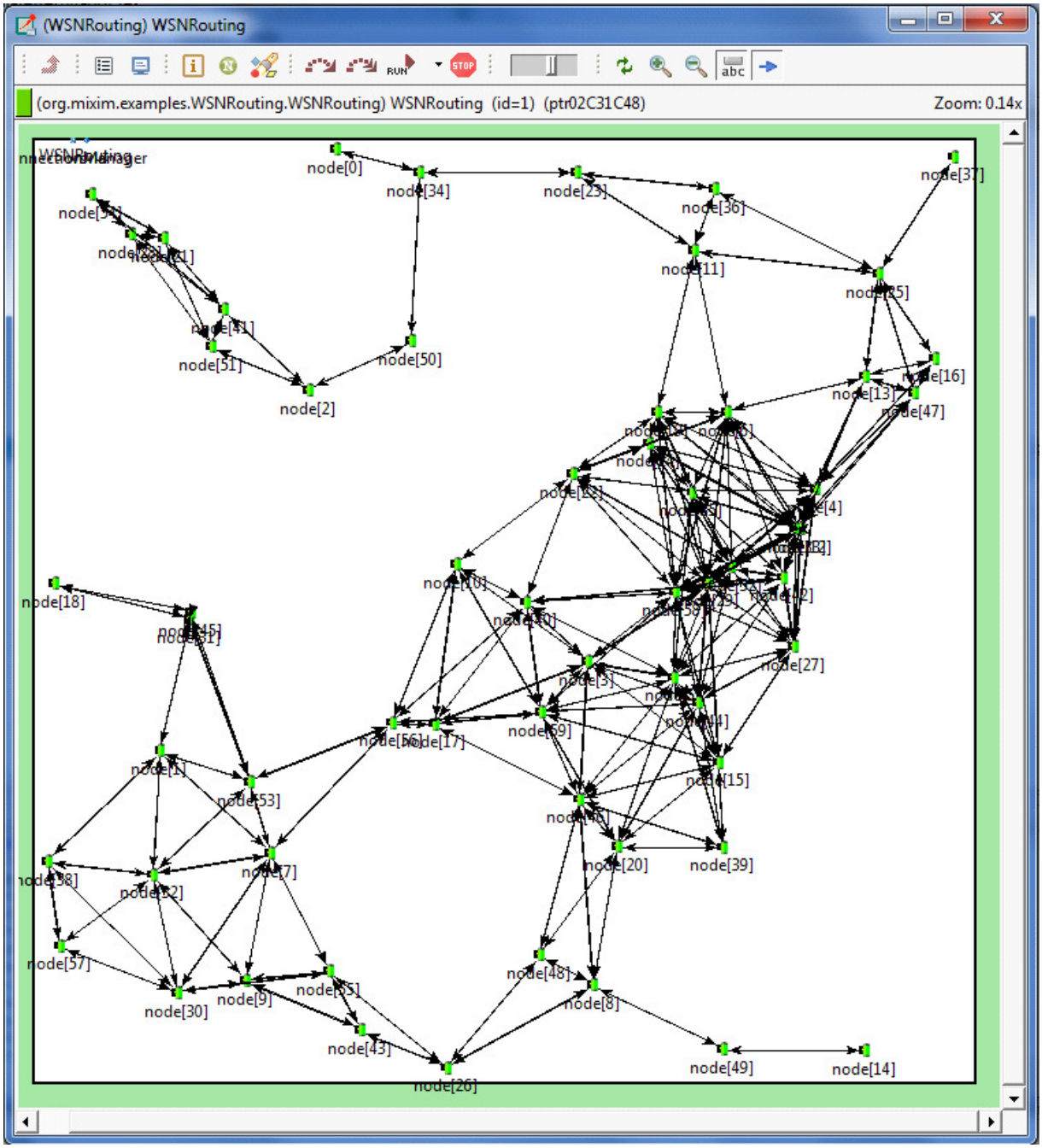


Figura 4.172: Configuración física de red para una de las pruebas de autoconfiguración realizadas. Tomada de [71].

los nodos cada minuto después de cinco minutos de encendida la red para dar tiempo a que los nodos lleguen a una configuración estable. En ese momento se empiezan a apagar las etapas de radio de algunos nodos como se muestra en la tabla 4.67; cuando se llega a veinticinco minutos de simulación se dejan de generar alarmas y se encienden todos los nodos de nuevo, la prueba finaliza a los cien minutos de simulación. Aun cuando los nodos se encuentran con la etapa de radio apagada, estos siguen generando y guardando alarmas, las cuales deberán ser enviadas cuando los nodos vuelvan a encender sus etapas de radio. Una vez que se termina la prueba, se cuentan las alarmas recibidas en el nodo base para corroborar que la red fue capaz de reconfigurarse para enviar las alarmas al sumidero.

En esta prueba específica se obtiene que llegaron al nodo base un total de mil docientas tramas de alarma de tipo nueve para cada uno de los escenarios, lo cual indica que los nodos fueron capaces de reconfigurarse para soportar los cambios de topología causados por los nodos que salieron de la red y después regresaron.

Además de los cien casos anteriormente mencionados se realizan cien pruebas más usando configuraciones como la que se muestra en la Fig. 4.172, en la que se repite la misma prueba anteriormente descrita, solo que esta vez no se enciende todos los nodos apagados en el minuto veinticinco sino que estos permanecen apagados y simplemente se cuentan las alarmas que llegan a la base para verificar que la red fue capaz de reconfigurarse para tomar en cuenta los fallos en los nodos. La selección de estos nuevos cien casos se debió hacer muy cuidadosamente debido a que si se apaga un nodo que deje aislado un segmento de la red, dicho segmento no será capaz de enviar los datos a la base debido a que no existirán caminos posibles para hacerlo. Es por esta razón que se escogen topologías, en donde se sabe que los nodos que se apagan si bien es cierto requerirán de los nodos de la red cambien sus datos de nivel de ADM y AMD, se sabe que estos dejan la posibilidad de encontrar otro camino a la base, para estos casos se generan nodos en posiciones aleatorias pero luego se utiliza un programa para validar que las posiciones seleccionadas cumplen con la restricción anteriormente expuesta, si cumple se mantiene el escenario y si no se genera una nueva topología hasta encontrar los cien casos que funcionarán para la prueba.

En este caso se tiene que la cantidad de alarmas reportadas a la base fueron novecientas ochenta, lo cual era lo esperado para esta prueba, e indica que los nodos fueron capaces de reconfigurarse para soportar los cambios de topología causados por los fallos en los nodos.

Otro parámetro que se considera relevante tomar en cuenta con respecto a la capacidad de autoconfiguración de la red, es el tiempo que tardan los nodos de diferentes niveles en obtener su nivel de ADM. Para medir esto se utilizarán dos topologías base; en una se ubican los nodos en una cuadricula como se muestra en la figura4.173 solamente que en lugar de hacer la cuadricula de 7X7 se hará de 51X51, la segunda topología será como la que se muestra en la Fig. 4.174 solo que en lugar de tener cinco niveles se tendrán cincuenta niveles. Para ambas topologías se harán dos experimentos los cuales se repetirán cien veces; en el primero se encienden todos lo nodos en tiempos aleatorios del tiempo cero

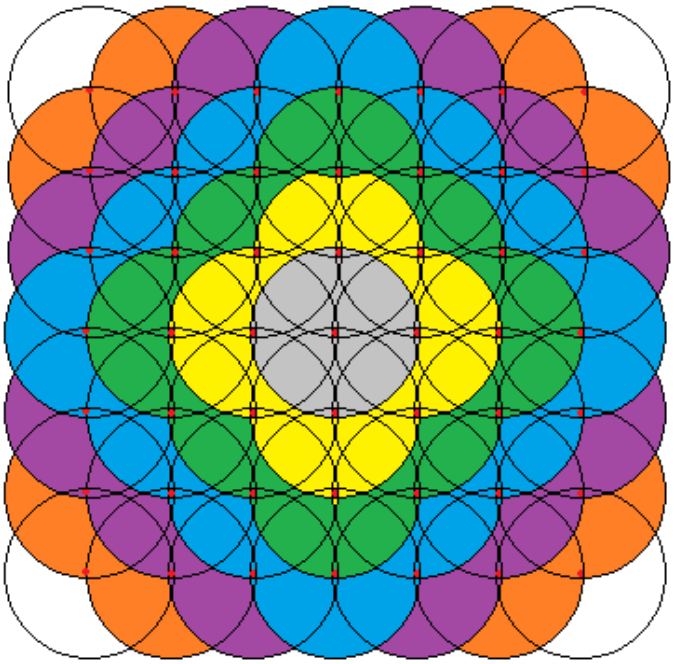


Figura 4.173: Topología en cuadricula para una red de sensores en la que los nodos se encuentran al límite del alcance de radio de sus vecinos. Tomada de [71].

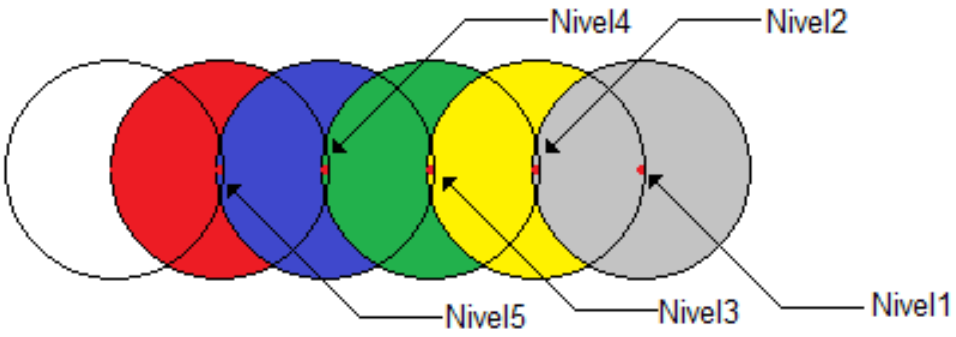


Figura 4.174: Topología en línea para una red de sensores en la que los nodos se encuentran alineados al límite del alcance de radio de sus vecinos. Tomada de [71].

hasta dos minutos de la simulación, luego en el tercer minuto de la simulación se enciende la base y desde ahí se mide el tiempo que tarda cada uno de los nodos de la red en los diferentes niveles en obtener su nivel de ADM y AMD,en el segundo experimento una vez que todos lo nodos tienen sus niveles configurados se cambia el nivel de la base para que en lugar de ser uno sea cinco y se mide el tiempo que tardan los nodos en reconocer el cambio con y sin trafico en la red para diferentes valores el parámetro X, el cual se define como el número de veces que cada nodo pasa por el estado de hibernación antes de pasar al estado de descubrimiento de nivel para verificar su nivel. Este experimento se hace usando la topología lineal anteriormente descrita. Para la generación del tráfico se insertan alarmas en los nodos de nivel múltiplo de cinco cada minuto.

Nivel	Retardo promedio en topología lineal	σ^2 en topología lineal	Retardo promedio en topología malla	σ^2 en topología malla
5	272,68	681,41	254,30	398,07
10	599,56	1753,54	566,59	1133,88
15	935,32	2162,42	872,0947	1374,30
20	1268,60	3402,07	1181,67	1828,41
25	1602,87	3578,15	1490,07	2194,07
30	1935,67	4384,13	1798,99	2544,20
35	2269,48	5916,29	2105,93	3217,46
40	2604,41	6821,45	2419,39	3702,63
45	2931,20	7403,77	2726,38,	4009,99
50	3259,73	7925,22	3034,99	4691,99

Tabla 4.68: Datos de tiempo para la obtención de nivel de ADM y AMD en las topologías lineal y en malla para la red en reposo usando valores normalizados en términos de B. Tomada de [71].

En la tabla 4.68 se muestran los resultados obtenidos para los tiempos promedio de adquisición del nivel de red normalizado en función de B cuando la red está en reposo. Se puede notar que los tiempos para la topología de malla son un poco menores a los tiempos para la topología lineal, este resultado puede deberse a que en la topología de malla los nodos pueden tener más de un vecino de nivel inferior,razón por la que hay mayor probabilidad de escuchar tramas de PT de algún camino que se configuró más rápido que otro, además se puede notar que la varianza de los datos aumenta con el nivel de los nodos de la red de manera a mayor nivel del nodo mayor es la variabilidad del tiempo necesario para que el nodo identifique su nivel de red. Para un nodo de nivel 50 el tiempo promedio que se necesitó para obtener su nivel de red en la topología de malla, tomando en cuenta un valor de $B = 0,058s$ fue de: $3035 * 0,058/60 = 2,93 minutos$ en la topología de malla

En la Fig. 4.175 se observan los resultados de retardo para la obtención del nivel de red en los nodos de nivel 50 cuando la red se encuentra en reposo, en esta figura se pueden notar las diferencias de tiempo entre el retardo de adquisición de nivel entre diferentes

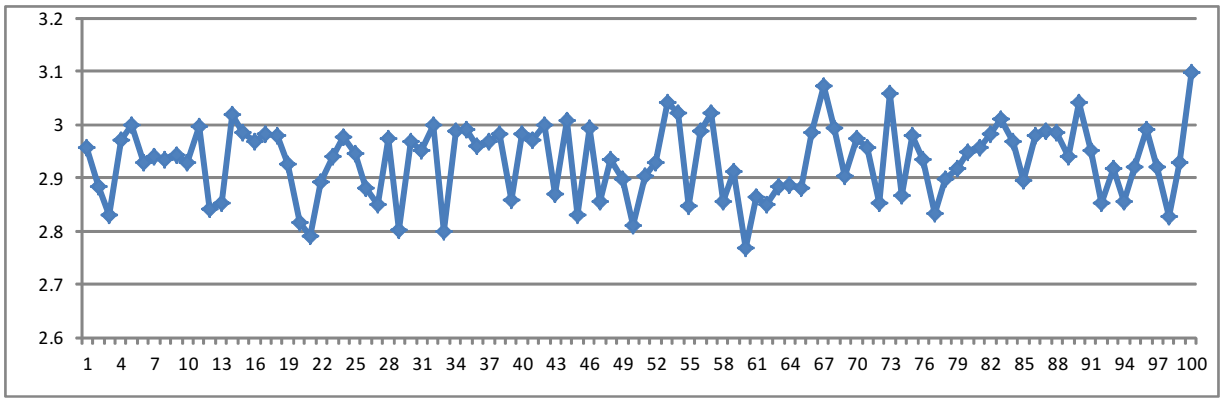


Figura 4.175: Retardo de la primera configuración de los nodos de nivel 50 para la topología de malla, en el eje Y se observa el retardo en minutos y en el eje X se observa el número de experimento. Tomada de [71].

experimentos.

Nivel	Retardo promedio en topología lineal	σ^2 en topología lineal	Retardo promedio en topología malla	σ^2 en topología malla
5	289,38	1178,54	279,98	783,34
10	652,00	2124,38	617,15	1454,53
15	1006,13	4011,51	963,51	2488,13
20	1363,39	5405,66	1308,34	3170,50
25	1724,28	5633,33	1642,56	4438,31
30	2085,34	7049,61	1984,34	5405,02
35	2438,74	7925,81	2325,75	6272,56
40	2794,64	8209,45	2669,61	7068,37
45	3152,01	8869,53	3014,3	7739,93
50	3507,52	10784,89	3352,60	8750,16

Tabla 4.69: Datos de tiempo para la obtención de nivel de ADM y AMD en las topologías lineal y en malla para la red con trafico usando valores normalizados en términos de B. Tomada de [71].

En la tabla 4.69 se muestran los resultados obtenidos para los tiempos promedio de adquisición del nivel de red normalizado en función de B cuando se le añade tráfico a la red. Se puede notar que al igual que en el caso anterior el retardo para la configuración lineal es un poco mayor que para la de malla, dicho comportamiento se puede atribuir a que para transmitir las alarmas en la topología lineal todos los mensajes deben usar el mismo camino de forma que cada nodo tiene un solo vecino; no obstante, en la topología de malla cada nodo tiene más de un vecino por lo que puede escuchar más ramas de PT por lo que es posible que el nodo se configure un poco más rápido.

En la tabla 4.70 se muestran los resultados de tiempo promedio requerido para detectar un cambio en la topología para nodos detectados a diferentes niveles de distancia del cambio

nivel	Retardo $X = 1$	Retardo $X = 5$	Retardo $X = 10$	σ^2 con $X = 10$
5	8,50	41,20	78,33	381,87
10	14,69	71,83	138,13	674,031
15	21,01	101,87	203,03	1070,37
20	27,66	134,53	266,70	1578,50
25	33,80	164,80	328,73	1598,52
30	40,36	196,96	392,00	1988,26
35	45,12	219,37	442,11	2218,49
40	53,70	257,51	520,99	2740,80
45	60,25	288,77	585,41	3517,39
50	66,35	319,43	650,699	3867,51

Tabla 4.70: Datos de tiempo promedio en minutos requerido por nodos de diferentes niveles para la detección de cambios en la red, con la red en reposo y usando topología lineal. Tomada de [71].

nivel	Retardo $X = 1$	Retardo $X = 5$	Retardo $X = 10$	σ^2 con $X = 10$
5	8,26	19,78	24,44	38,77
10	14,86	35,76	42,19	53,93
15	20,91	51,43	59,78	89,56
20	27,36	67,44	78,38	113,66
25	33,38	83,19	96,89	136,29
30	40,22	99,29	115,89	147,40
35	44,96	111,61	130,05	159,99
40	53,12	132,38	154,07	213,35
45	59,20	147,04	173,40	265,50
50	65,73	162,64	191,65	317,24

Tabla 4.71: Datos de tiempo promedio en minutos requerido por nodos de diferentes niveles para la detección de cambios en la red, con la red con tráfico y usando topología lineal. Tomada de [71].

con la red en reposo, en la tabla 4.71 se muestran los mismos resultados pero con tráfico en la red, se puede notar que cuando se usan los parámetros de la red como se muestra en la tabla 4.66, un nodo a 50 niveles de distancia de un cambio importante en su topología puede tardar en promedio 11 horas en darse cuenta, si la red está en reposo, mientras que durará solamente 1 hora cuando hay trafico en la red. Este resultado refleja el hecho de que cuando la red está en reposo depende del parámetro X para refrescar su información de nivel, mientras que cuando hay trafico el nodo pasará al estado de descubrimiento de nivel si no escucha una trama de PT válida sin importar el valor en el parámetro X, esto también explica en hecho de que el parámetro X no tiene un efecto tan grande cuando hay tráfico en la red, en el retardo de configuración del nivel de red de los nodos como lo tiene cuando la red se encuentra en reposo.

Latencia de las tramas

Para medir la latencia de las tramas se realizaron dos experimentos. En el primero se insertarán mil alarmas con una distancia de un minuto entre cada una, para los nodos de niveles múltiplos de cinco y se medirá el retraso de las mismas en función de B, utilizando la topología lineal para diferentes valores del tiempo de hibernación T en el segundo experimento se utilizará la topología en malla de 51X51 usada en el experimento anterior y se insertarán alarmas en quinientos nodos aleatorios a diferentes frecuencias para medir el efecto que esto tiene en el retardo de las tramas a diferentes niveles usando un $T = 4$.

Resultados del primer experimento:

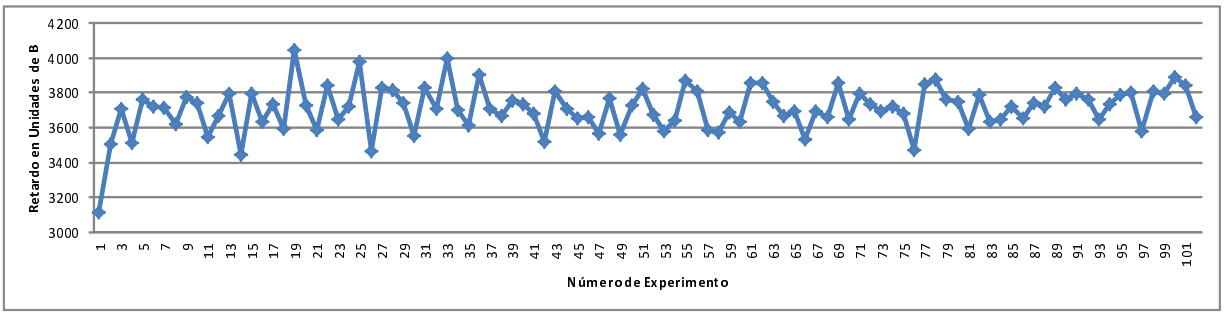


Figura 4.176: Resultado de latencia para las tramas de nivel cincuenta en algunos de los mil experimentos realizados cuando $T = 4P$

En la tabla 4.72 se resumen los resultados obtenidos para el experimento cuando se usa $T = 4P$, se puede ver que para todos los casos los retardos promedio en función de B se encuentran dentro de los rangos esperados del análisis teórico, además se puede ver que conforme aumenta el nivel del nodo emisor de la alarma la varianza σ^2 del retardo aumenta, en la figura4.176 se puede ver las diferencias en el retardo para algunos de las alarmas detectadas en nodos de nivel 50 usando $T = 4P$.

En la figuras 4.177 se puede notar que para diferentes niveles de los nodos emisores de

nivel	Retardo mínimo esperado	Retardo máximo esperado	Retardo promedio	σ^2
5	75	625	306,44	1623,31
10	150	1195	602,13	2616,98
15	225	1765	902,77	4127,24
20	300	2335	1206,05	6679,56
25	375	2905	1508,77	7834,38
30	450	3475	1808,75	10121,96
35	525	4045	2111,81	11284,34
40	600	4615	2413,07	14278,23
45	675	5185	2716,42	16011,45
50	750	5755	3698,78	18709,39

Tabla 4.72: Datos de tiempo promedio normalizado en B requerido por las tramas enviadas desde diferentes niveles para llegar a la base usando topología lineal, insertando una alarma por minuto en los nodos de nivel múltiplo de cinco con $T = 4P$. Dado que los datos son normalizados no tiene unidades

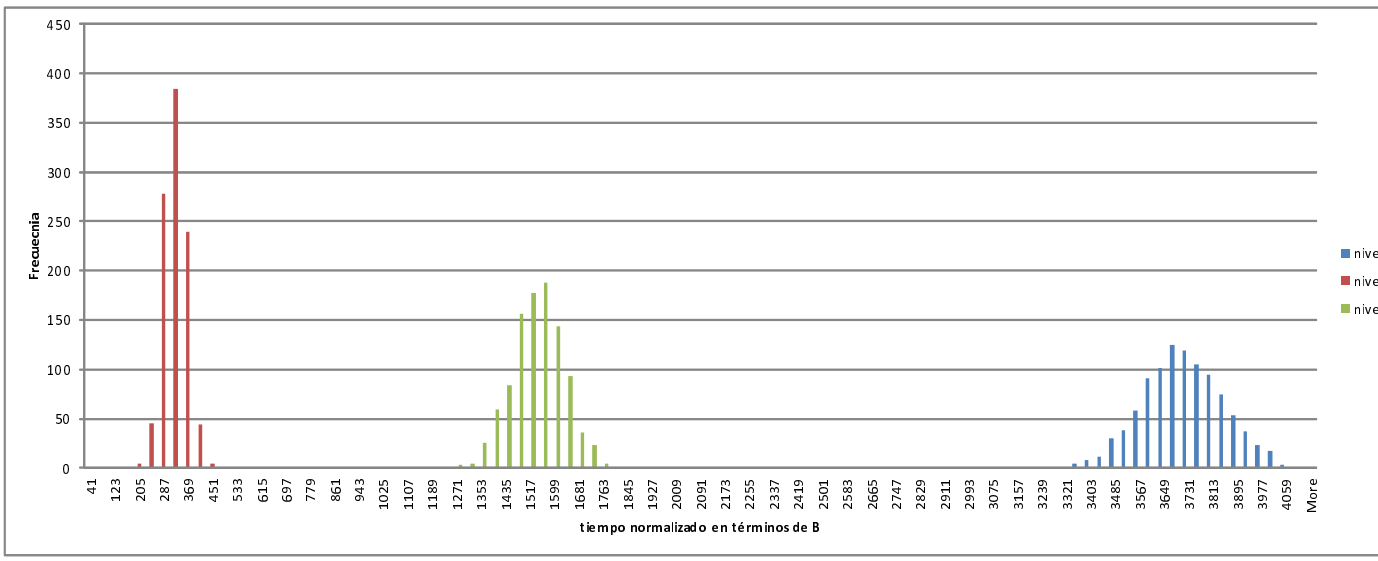


Figura 4.177: distribución de frecuencia del retardo de las alarmas generadas en nodos de niveles 5, 25 y 50 con $T = 4P$. Tomada de [71].

las alarmas la función de probabilidad en el retraso de las tramas se ve muy similar a una curva normal, en donde la varianza aumenta conforme se aumenta el nivel del nodo emisor de la alarma, lo cual concuerda con el análisis teórico de la latencia que se expuso en la sección de diseño del protocolo en la que se modela el retardo de las tramas.

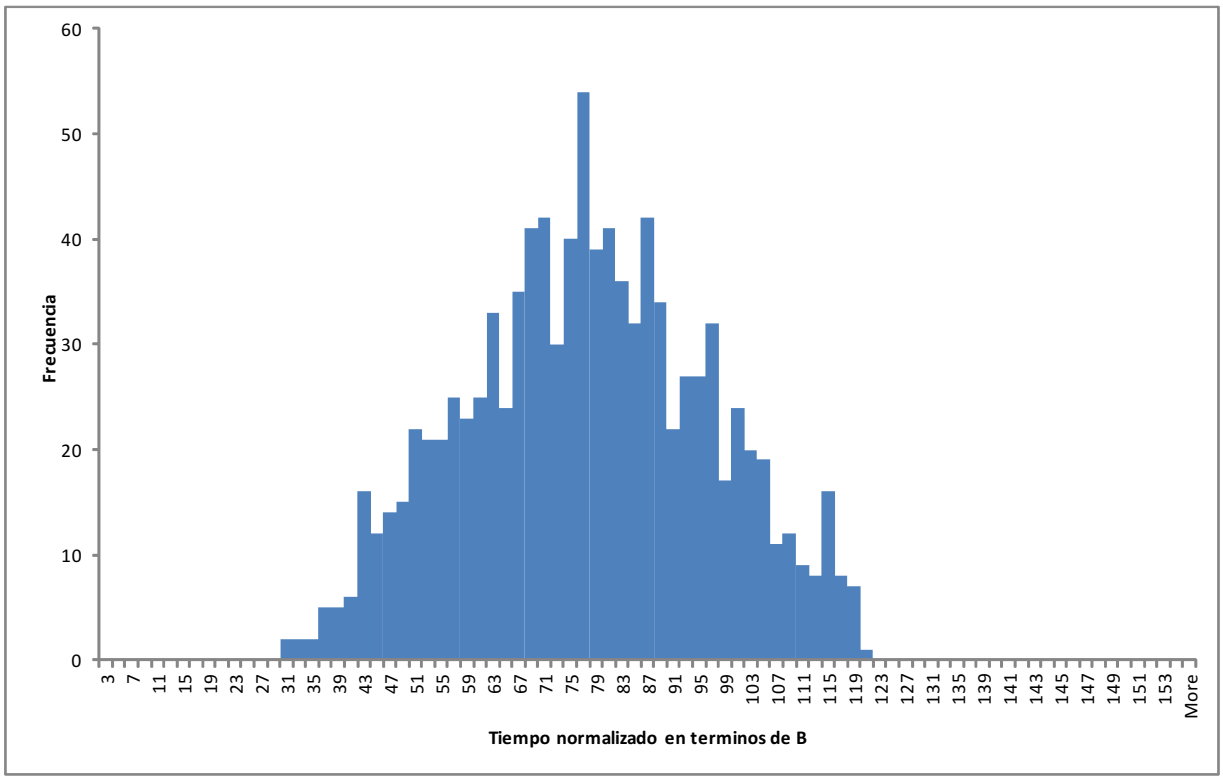


Figura 4.178: Distribución de frecuencia del retardo de las alarmas para saltar de un nodo al siguiente con $T = 4P$. Tomada de [71].

En la figura4.178 se observa la distribución de frecuencia para la latencia al enviar una trama de un nivel al siguiente usando la topología lineal como referencia, dicho gráfico se puede comparar con la Fig. 4.179, que viene del análisis teórico de la distribución de probabilidad teórica del retardo en el envío de una alarma de un nivel al siguiente. Se puede notar que en comparación la curva obtenida en simulación se encuentra corrida hacia la derecha aproximadamente 25 unidades de B y tiene una varianza mayor que la estimada teóricamente. Esta diferencia puede deberse a simplificaciones hechas en el modelo teórico, como por ejemplo que se asume que P va a ser exactamente $11B$ lo cual no será cierto en la simulación ya que hay que tomar en cuenta el tiempo de transmisión de las tramas o por ejemplo que alguno de los nodos tuvieran que esperar por haber escuchado una trama RTS o PT de alguno de sus nodos vecinos.

En la Fig. 4.180 se muestran los resultados obtenidos al medir la latencia promedio para diferentes valores T desde nodos en diferentes niveles de ADM. Se observa que el retardo crece de manera aproximadamente lineal con diferentes pendientes para los diferentes niveles conforme se aumenta T.

Resultados del segundo experimento:

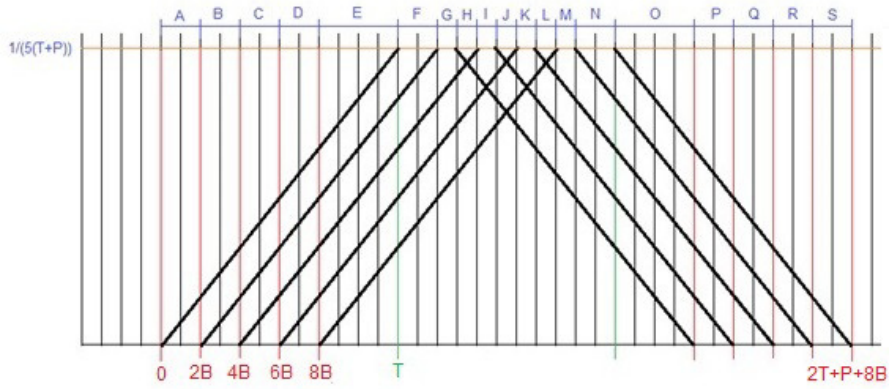


Figura 4.179: En (A) se muestra un ejemplo de las funciones de probabilidad independientes del tiempo C y en (B) se muestra la función de probabilidad que se obtiene al sumar el área debajo de las curvas. Para calcular la función de densidad de probabilidad para el tiempo de latencia de las tramas asumiendo que no hubo colisiones en la transmisión entre 2 nodos es necesario sumar las áreas bajo las curvas de las diferentes funciones de densidad de probabilidad, en donde se usa como ejemplo B=1, $P = 11B$ y $T = 2P = 22B$¿ el análisis teórico de esta probabilidad puede verse de forma completa en [71], de donde se toma esta imagen.

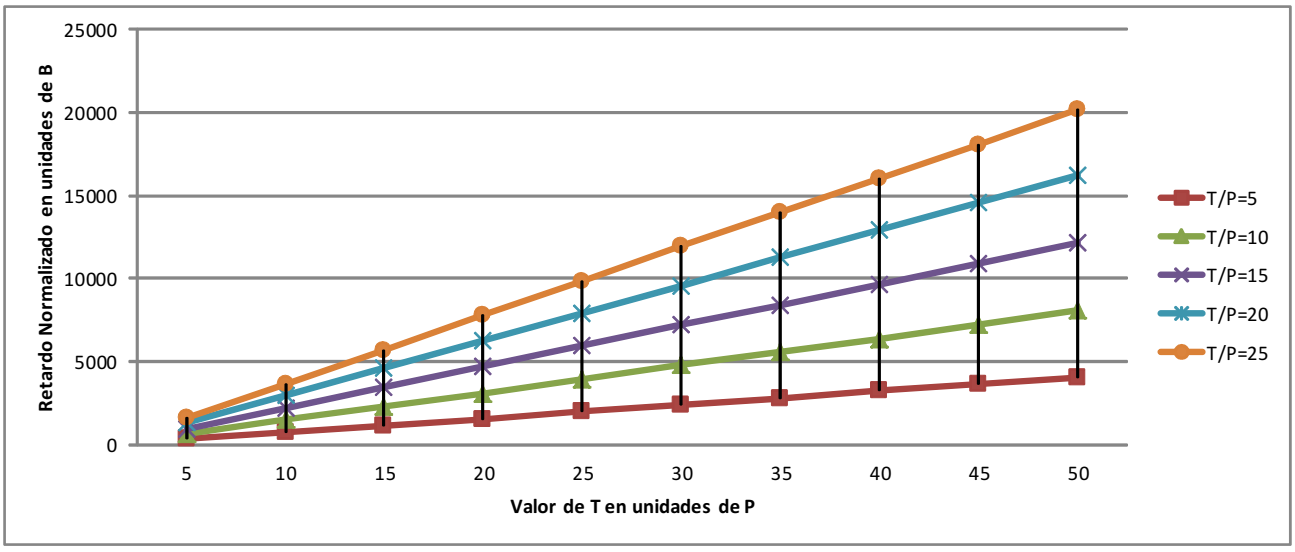


Figura 4.180: Latencia promedio usando diferentes valores de T para nodos que se encuentran en diferentes niveles de ADM. Tomada de [71].

nivel	Periodo $5000B$	Periodo $4000B$	Periodo $3000B$	Periodo $2000B$	Periodo $1000B$	Periodo $500B$
5	282,85	325,06	333,82	396,84	490,99	673,83
10	551,27	641,81	660,41	784,89	968,41	1330,32
15	824,52	961,06	988,52	1183,20	1453,86	1990,16
20	1098,50	1280,38	1315,02	1574,07	1937,28	2663,61
25	1372,27	1599,16	1644,67	1969,54	2420,83	3329,16
30	1647,34	1918,41	1969,94	2361,63	2904,68	3996,33
35	1922,16	2238,97	2300,48	2751,85	3386,00	4655,59
40	2195,74	2558,47	2626,16	3141,38	3867,33	5324,00
45	2471,46	2878,76	2955,85	3534,90	4350,52	5993,74
50	3370,97	3924,19	4032,10	4814,06	5930,31	8165,10

Tabla 4.73: Datos de tiempo promedio normalizado en B requerido por las tramas enviadas desde diferentes niveles para llegar a la base usando una topología en malla y diferentes frecuencias para la generación de alarmas en 500 nodos aleatorios a la vez. Dado que los datos son normalizados corresponden a unidades de B y no tienen unidades. Tomada de [71].

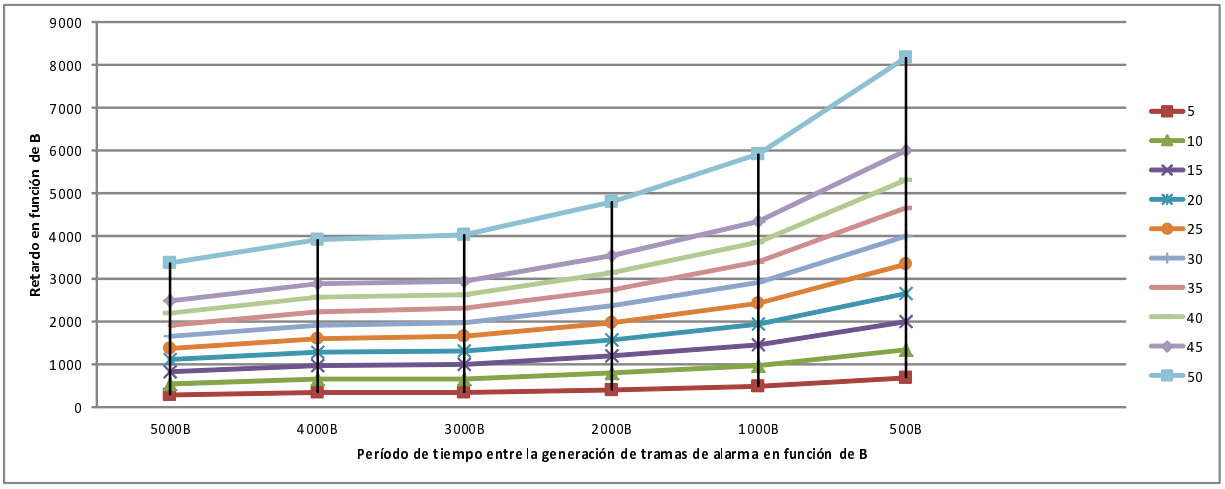


Figura 4.181: Latencia promedio normalizada en B de las alarmas en llegar a la base para nodos en diferentes niveles de ADM de la red para diferentes niveles de tráfico. Tomada de [71].

En la tabla 4.73 se resumen los resultados obtenidos en el segundo experimento, se puede notar que la latencia de las alarmas no solamente aumenta conforme se aumenta el tráfico en la red, sino que aumenta en proporciones diferentes para los diferentes niveles de los nodos de la red, a mayor nivel mayor aumento en la latencia conforme se aumenta el tráfico, esto se ve con claridad en la Fig. 4.181.

Si se asume $B = 0,058s$ se tiene que para una alarma que se genere en una red con topología de malla en la que se generen alarmas en quinientos nodos aleatorios cada veintinueve segundos, lo cual para la aplicación que estamos evaluando sería un tráfico muy alto, tendrá una latencia promedio de aproximadamente ocho minutos en llegar a la base mientras que si la alarma se genera cuando el trafico que se esta dando en la red es de quinientas alarmas en nodos aleatorios cada 290s, entonces la alarma tendrá una latencia promedio de aproximadamente tres minutos.

Consumo de potencia

Para medir el consumo de potencia se utilizan dos pruebas. En la primera se mide el consumo de potencia promedio de la red (entiéndase la suma del consumo de potencia promedio de todos los nodos en la red dividido entre el número de nodos) en reposo para dos redes con la misma configuración física, en esta prueba se usa la topología de malla de 51X51 pero usando en un caso el protocolo propuesto y en el otro los nodos de referencia conforme se aumenta el tiempo de hibernación, y en la segunda se hace manteniendo el tiempo de hibernación e insertando alarmas en 500 nodos aleatorios a diferentes frecuencias de la misma manera que se hizo en la prueba de retardo, para ver el efecto que esto tiene en el consumo de potencia. Para cada uno de los casos se repite el experimento cien veces y se reportan los resultados promediados después de cien minutos de simulación.

Para la simulación que utiliza los nodos de referencia se usó la interfaz Nic802154 TICC2420 de Texas Instruments implementada en la biblioteca de simulación Mixim 2.1, la cual implementa CSMA no ranurado (nonbeacon-enabled unslotted CSMA-CA) en modo competitivo.

Los parámetros utilizados en el simulador para describir el consumo de potencia de la capa física se describen en la tabla4.65 y son los parámetros por defecto que usa Mixim para la configuración de la capa física del nodo de referencia.

Además de los parámetros anteriormente mencionados se define la base de tiempo B en 58ms. Esto basado en los parámetros del módulo TR1000 usados también como referencia en [79]. La razón por la que se usa como referencia el transceptor TR1000 en lugar del CC2024 es porque no fué posible encontrar el dato en la hoja de datos de este último dispositivo. El resto de los parámetros de la red se describen en la tabla 4.66 con la diferencia de que para esta prueba se utilizará $X = 50$..

Suponiendo que el tiempo P es aproximadamente igual a $11B$, se tiene que la potencia

T/P	Potencia referencia	Potencia del protocolo propuesto	%Ahorro	%Ahorro modelado	Differencia
5	0,0574	0,0109	81,0432	83,1854	2,1422
10	0,0574	0,0074	87,1840	90,7702	3,5862
15	0,0574	0,0059	89,6499	93,6145	3,9646
20	0,0574	0,0049	91,4135	95,1044	3,6909
30	0,0574	0,0038	93,4570	96,6423	3,1853
40	0,0574	0,0037	93,5906	97,4301	3,8395

Tabla 4.74: Tabla de resumen de los valores de potencia promedio de la red en reposo simulados usando diferentes valores de tiempo de hibernación. Tomada de [71].

promedio simulada será cómo se muestra en la tabla 4.74. Además del valor de potencia promedio, se muestran en la tabla 4.74, los valores esperados con el modelo mostrado en el apartado de diseño del protocolo.

Se puede ver una diferencia que fluctúa entre 2% y 3,5%. Esta diferencia puede deberse a que en el modelo no se tomaron en cuenta corrientes diferentes en los periodos de transición entre los estados de escucha, transmisión e hibernación, además de que no se tomó en cuenta el tiempo necesario para el descubrimiento de nivel que se hará periódicamente cada cincuenta veces que se pase por el estado de hibernación.

Al igual que en el experimento anterior se procede a inyectar alarmas en grupos aleatorios de quinientos nodos en la red con topología de malla

T/P	Periodo ∞	Periodo 120s	Periodo 60s	Periodo 30s	Periodo 15s	Periodo 10s
10	0,0109	0,01412	0,01963	0,0315	0,0546	0,0590
20	0,0049	0,01054	0,01620	0,0280	0,0498	0,0573
30	0,0038	0,00888	0,01464	0,0264	0,0487	0,0573
40	0,0037	0,00810	0,01373	0,0249	0,0475	0,0578

Tabla 4.75: Datos de potencia promedio en Watts para proporciones de T/P simulados al generar alarmas a diferentes frecuencias. Tomada de [71].

En la tabla 4.75 se muestra el resultado obtenido. Se puede notar cómo el consumo de potencia crece de forma rápida conforme aumenta el tráfico en la red, este comportamiento es de esperarse debido a que conforme aumenta el tráfico en la red es necesario que los nodos mantengan la etapa de radio encendida por más tiempo para transmitir la información. Es importante apreciar que dado que la aplicación para la que pensó el protocolo propuesto asume que se va a encontrar en reposo la mayoría del tiempo el ahorro energético de la red se asume que será alto.

En la tabla 4.76 se muestran los resultados de potencia promedio consumida para los

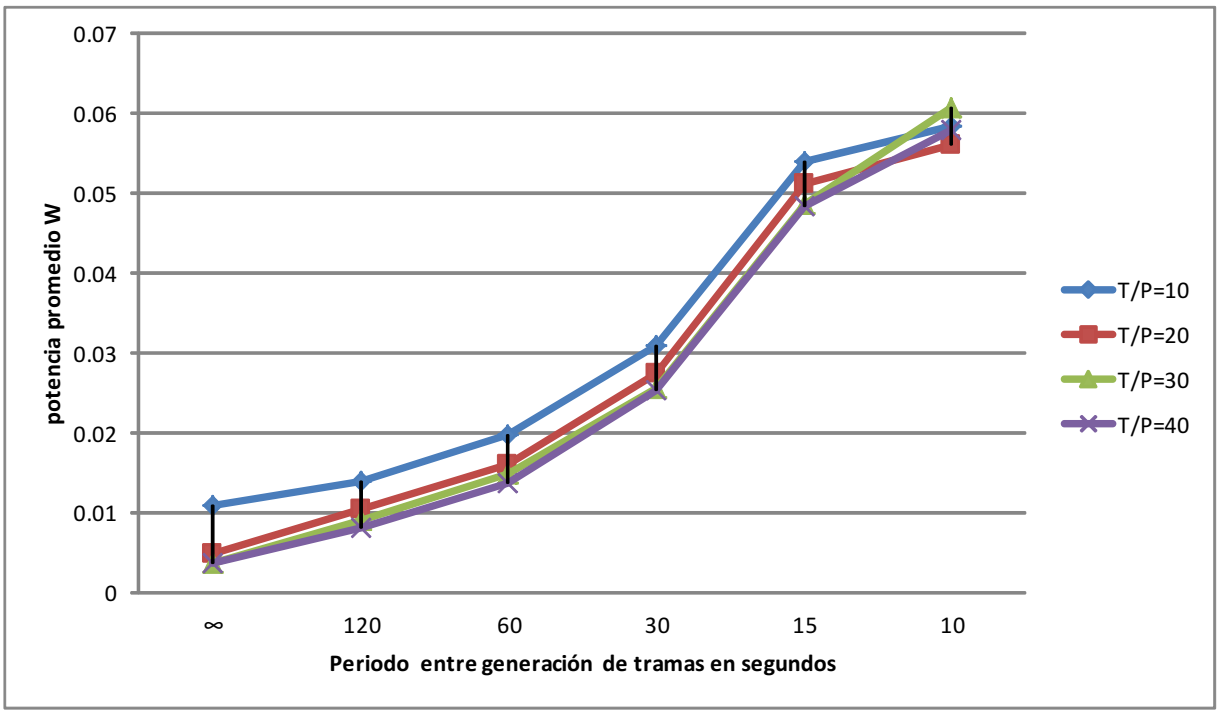


Figura 4.182: Latencia promedio normalizada en B de las alarmas en llegar a la base para nodos en diferentes niveles de ADM de la red para diferentes niveles de tráfico. Tomada de [71].

nivel	Potencia promedio(W) con la red en reposo	Potencia promedio(W) generando 500 alarmas cada 10 segundos
2	0,0037	0,0602
5	0,0037	0,0591
20	0,0037	0,0585
30	0,0037	0,0576
40	0,0037	0,0572
50	0,0037	0,0569

Tabla 4.76: Datos de potencia promedio en Watts para los nodos de diferentes niveles de ADM usando $T = 40$. Tomada de [71].

nodos de diferentes niveles de la red, se puede notar que la potencia consumida en los niveles bajos de la red cuando se generan quinientas alarmas en nodos aleatorios cada diez segundos es mayor en los nodos de niveles bajos, este resultado puede deberse a que para la topología seleccionada todos los paquetes fluyen hacia un solo sumidero de forma que todos los paquetes deben pasar por lo nodos de nivel bajo, los cuales son menos.

4.6.7 Conclusiones parciales sobre la propuesta a nivel de bloques de la arquitectura de comunicación del SoC con los demás nodos de la red

Se ha logrado implementar una red funcional de sensores (ver [56]), que puede funcionar como base de comunicaciones para los sensores de detección de disparos y motosierras. Además, se ha producido como aporte de una tesis de maestría [71], un protocolo funcional exhaustivamente verificado y que ha sido comprobado ya sobre una plataforma de red específica. Este protocolo aprovecha la ventaja de las características de la aplicación para apagar la etapa de radio de los nodos que forman la red por periodos de tiempo que es posible configurar para mejorar el consumo energético.

Se ha modelado la relación entre el tiempo de hibernación de los nodos y el ahorro energético de los mismos con respecto de una red en la que no se apaga la etapa de transmisión de los nodos, y se ha modelado la latencia en la entrega de las alarmas al nodo base dependiendo del tiempo de hibernación y la cantidad de saltos necesarios para que las alarmas alcancen la base. Dicho modelado provee al diseñador de la red de herramientas para configurarla de manera que cumpla con los requerimientos específicos de la aplicación se busque soportar. Con respecto del consumo de potencia en los nodos se comprueba que será inversamente proporcional al tiempo asignado en el estado hibernación cuando la red se encuentra en reposo y de relación directamente proporcional a la base de tiempo B. Además se observa que consumo de potencia en la red aumenta de manera rápida conforme aumenta el tráfico en la red para aproximarse al consumo de un protocolo que mantiene su etapa de radio encendida de manera constante, por lo que se obtiene un mayor ahorro energético en aplicaciones en las que se sabe que existirá un trafico de información bajo. Además, se ha comprobado por medio de simulación que el protocolo propuesto es capaz de autoconfigurarse para adecuarse a posibles fallas de los nodos y entregar las alarmas detectadas siempre y cuando no queden aislados de manera permanente (sin conexión a la base) o sufran un daño antes de poder entregar las tramas al siguiente nodo. Para facilitar el proceso de configuración de la red todos los tiempos del protocolo se basan en el parámetro B, el cual se define como el tiempo requerido para que el paquete más largo de la red se transmita a la distancia máxima de alcance del nodo transmisor y sea recibida por el nodo receptor, dicho parámetro depende de la capa física de la Red y deberá ser determinado para la implementación final de la misma. Con respecto al nodo de referencia Nic802154 TICC2420 de Texas Instruments el cual implementa el estándar IEEE802.15.4 con CSMA no ranurado (nonbeacon-enabled unslotted CSMA-CA) en modo competitivo, y usando wiseroute como protocolo de red, el protocolo propuesto es capaz de ahorrar aproximadamente un 93% de energía si se usa una proporción entre el periodo

de hibernación y el periodo P de 40:1 con la red en reposo.

Con respecto a la latencia de las tramas, esta va a depender de factores de configuración del protocolo como la base de tiempo B y el tiempo de hibernación seleccionado. Además dependerá de factores externos al protocolo como lo son la topología de la red y la carga de la misma. Si se supone $B = 0,058s$ se tiene que para una alarma que se genere en una red con topología de malla en la que se generen alarmas en quinientos nodos aleatorios cada 29 segundos, lo cual para la aplicación que estamos evaluando sería un tráfico muy alto, tendrá una latencia promedio de dieciséis minutos en llegar a la base mientras que si la alarma se genera cuando el trafico es 500 alarmas en nodos aleatorios cada 290s, entonces la alarma tendrá una latencia promedio de 3 minutos.

Queda pendiente nada más para este objetivo el portar el protocolo a la red desarrollada en [56].

4.7 Publicaciones y divulgación

Se produjo una publicación en Tecnología en marcha.

Se produjo una publicación de congreso.

Se presentó el proyecto en la CASS en Buenos Aires, en 2014. El avance de este proyecto nos ayudó a ser invitados a participar en el proyecto XXXX, actualmente en proceso e inscrito ante la VIE como proyecto de investigación internacional.

El atraso en el desarrollo de varios de los objetivos atentó contra una mayor divulgación de los resultados, que se espera sea solventado con los resultados presentados en este proyecto, y los pendientes de obtener al comprobar tanto el circuito digital que se enviará a fabricar en agosto 2014, como la evaluación estadística exhaustiva del filtro analógico de onditas, que se planea llevar a cabo en el segundo semestre del 2014.

5. Conclusiones

Se han generado conclusiones técnicas parciales para cada objetivo realizado, que pueden hallarse en las secciones 4.2.2, 4.2.4, 4.3.3, 4.4.4, 4.5.5 y 4.6.7. No se repiten para no extender más el informe. No obstante, de estas conclusiones parciales, y el fallo de cumplimiento parcial o total de los objetivos 1, 3, 4, 5 y, por ende, 7, pueden generarse las siguientes conclusiones generales del proyecto:

a Es posible desarrollar circuitos funcionales para la detección y clasificación de patrones acústicos, pero debe trabajarse más en la capacitación de los investigadores para asegurar su correcta implementación.

b Debe trabajarse más en transformar la teoría de procesamiento de señales en aplicaciones prácticas que consideren las limitaciones computacionales que tendrá un sistema embebido o de hardware integrado, sea en FPGA o en un ASIC digital o analógico.

c Deben integrarse más los esfuerzos entre los distintos equipos de investigación, para aprovechar los resultados de estos proyectos y establecer así una curva creciente de desarrollo técnico e investigativo en estos equipos.

Por otra parte, conviene plantear varias recomendaciones para futuros proyectos de índole similar, no solo para los investigadores involucrados en este proyecto, sino para la misma Escuela de Ingeniería Electrónica y la institución:

a El análisis de propuestas proyectos de desarrollo debe proveer muy bien las capacidades administrativas de la institución, y el tiempo real que dedicarán los investigadores en desarrollar los productos, contrastando tanto objetivos como recursos contra proyectos similares ya sea en la industria o en instituciones académicas de prestigio. En el caso de este proyecto, se propuso la construcción de tres circuitos integrados en un periodo de dos años, junto con la implementación de una red de sensores para integrar los dispositivos fabricados. En la industria, solo la fabricación de un ASIC es un proyecto que lleva de dos a tres años para un equipo de decenas de ingenieros capacitados con todas las herramientas y recursos necesarios. Esto sin considerar que ninguno de los investigadores involucrados era un experto *per se* en redes de sensores. Ser claros y humildes con los objetivos esperados ha de producir mejores resultados que propuestas mal calibradas y demasiado ambiciosas.

b Como especial cuidado, debería evitarse que los investigadores tengan responsabilidades administrativas más allá de las comunes que debe cumplir un profesor de la institución. En este caso, en paralelo con el proyecto, el investigador principal fungió durante casi dos años como coordinador de la Unidad de Posgrados de la Institución (actualmente Dirección de Posgrado), mientras que el segundo investigador fungió como coordinador del área académica del Doctorado en Ciencias Naturales. Si bien en el segundo caso, el tiempo formal dedicado a esta coordinación era de medio tiempo,

debe tenerse claro que la investigación es un proceso que requiere claridad mental y esfuerzo técnico concentrado. Es conocido que las labores administrativas de puestos de mucha responsabilidad no solo tienden a sobrepasar el tiempo asignado a ellas, sino que desligan a los responsables de sus actividades académicas para abocarse a tediosos procesos burocráticos, algo que aparte de robar tiempo, suele disminuir la capacidad técnica del investigador al alejarlo de su campo de especialidad. Esta situación es aún más grave en el caso del investigador principal, que asumió un puesto administrativo de tiempo completo, con grandes responsabilidades y retos que incluyeron una gran dosis de actividades de negociación política intra e interinstitucional, por tratarse del primer coordinador/director de posgrado que tuvo la institución.

c Desarrollar proyectos de investigación de alta complejidad con estudiantes asistentes de grado es contraproducente, pues la curva de aprendizaje de las herramientas y procedimientos necesarios consume buena parte del tiempo en que estos estudiantes cooperarán con el proyecto. Es conveniente que para un futuro, estos proyectos incluyan a estudiantes de posgrado, que posean ya no solo la experticia básica que disminuya su curva de aprendizaje, sino que además permanezcan en el proyecto durante toda su duración. Como comentario, puede decirse que el proyecto solo avanzó a grandes tractos cuando se incorporaron al mismo varios estudiantes de maestría y un doctorando, dos de los cuales finalizaron su tesis precisamente con este proyecto (ver [16, 71], tesis que dieron algunos de los mayores aportes al proyecto. Además, la única publicación indexada del proyecto vino del trabajo del doctorando que participó parcialmente y de manera *ad-honorem* en el proyecto (ver [64]).

6. Aportes

Se han desarrollado varios prototipos de circuitos integrados detectores analógicos de disparos que son eficientes energéticamente y que prometen ser eficientes algorítmicamente (se encuentran en su etapa de evaluación como detectores). Además, se ha encontrado un algoritmo de detección de motosierras altamente eficiente, que puede ser fácilmente implementado en una FPGA. Estos circuitos pretenden ser la etapa inicial de detección de eventos acústicos no deseados en un ambiente selvático.

Se cuenta con un prototipo funcional embebido de clasificación de patrones acústicos, con un alto grado de eficiencia y apto para usarse como un nodo de mediano consumo de potencia en una red de sensores de monitoreo ambiental. El prototipo además provee del patrón de comparación para desarrollar unidades con mejores prestaciones energéticas, y se cuenta con las herramientas de software para hacer estas unidades totalmente programables.

Además, se ha desarrollado el código RTL de una unidad de preprocesamiento y extracción de características de patrones acústicos, la cual está a punto de enviarse a fabricar en un circuito digital CMOS (esta unidad ya ha sido verificada sobre una plataforma FPGA). Además, se cuenta con el desarrollo inicial de un clasificador eficiente que se encuentra en pruebas sobre FPGA en la actualidad y que podrá integrarse a futuro a la unidad anterior para tener así un clasificador completo de patrones acústicos.

Se ha desarrollado un protocolo eficiente para una red inalámbrica de sensores ad-hoc, y se tiene ya la plataforma sobre la que puede montarse este protocolo.

Bibliografía

[1] Ian F. Akyildiz, Weilian Su, Yogesh Sankarasubramaniam, y Erdal Cayirci. A survey on sensor networks. *IEEE Communications Magazine*, págs. 102–114, August 2002.

[2] Phillip E. Allen y Douglas R. Holberg. *CMOS Analog Circuit Design, second edition.* Oxford University Press, 2002.

[3] D. Porras Alvarado. Diseño de una unidad de cálculo utilizada en la detección de disparos de armas de fuego. Tesis de Licenciatura, Escuela de Ingeniería Electrónica, ITCR, Cartago, Costa Rica, 2011.

[4] Pablo Alvarado, Apolinar González, y Luis Villaseñor. Propuesta de aplicación de redes de sensores en el modelado de cultivos protegidos y en campo. In *Memorias. "Workshop on Sensor Networks and Applications"*, Gramado, Brasil, Setiembre 2008.

[5] Pablo Alvarado y Marvin Hernández. D2ars: Diseño y desarrollo de aplicaciones basadas en redes de sensores. Informe Final de Proyecto, Vicerrectoría de Investigación, ITCR, Mayo 2010.

[6] Pablo Alvarado, Néstor Hernández, y Marvin Hernández. D2ARS: Diseño y desarrollo de aplicaciones basadas en redes de sensores. Propuesta de Proyecto, Vicerrectoría de Investigación, ITCR, Octubre 2007.

[7] A. Arnaud. *Very Large Time Constant Gm-C Filters.* PhD thesis, Universidad de la República, Uruguay, Montevideo, Uruguay, 2002.

[8] Esteban Baradín. Diseño de una unidad de preprocesamiento de señales impulsivas basada en un filtro haar. Tesis de Licenciatura, Escuela de Ingeniería Electrónica, ITCR, Cartago, Costa Rica, 2011.

[9] beagleboard.org. BeagleBoard-xM system reference manual [online]. 2010 [visitado el 3 de abril de 2013]. URL http://beagleboard.org/static/BBxMSRM_latest.pdf.

[10] M. Carvajal. Banco de pruebas del Sistema de Reconocimiento de Patrones Acústicos de motosierras y disparos. Tesis de Licenciatura, Escuela de Ingeniería Electrónica, ITCR, Cartago, Costa Rica, 2013.

[11] A. Chacon-Rodriguez, P. Julián, L. Castro, P. Alvarado, y N. Hernández. Evaluation of gunshots detection algorithms. *Circuits and Systems I: Regular Papers, IEEE Transactions on*, págs. 363–373, 2011.

[12] A. Chacon-Rodriguez, Shuo Li, M. Stanacevic, L. Rivas, E. Baradin, y P. Julian. Low power switched capacitor implementation of discrete haar wavelet transform. In *Circuits and Systems (LASCAS), 2012 IEEE Third Latin American Symposium on*, págs. 1–4, Feb 2012.

[13] A. Chacón-Rodriguez. *Circuitos Integrados de Bajo Consumo de Potencia para Detección y Localización de Disparos de Armas de Fuego*. PhD thesis, Universidad Nacional de Mar del Plata Argentina, Argentina, 2009.

[14] A. Chacón Rodríguez y P. Julian. Evaluation of gunshot detection algorithms. In *Micro-Nanoelectronics, Technology and Applications, 2008. EAMTA 2008. Argentine School of*, págs. 49–54, 2008.

[15] Minor Coto. Desarrollo de una plataforma de prueba para un circuito integrado de uso específico de ultra baja potencia mediante instrumentación definida por software. Tesis de Licenciatura, Escuela de Ingeniería Electrónica, ITCR, 2013.

[16] Jorge Cárdenas. Estrategias de entrenamiento de modelos ocultos de markov para reconocimiento de patrones acústicos. Tesis de maestría, Escuela de Ingeniería en Computación, ITCR, 2012.

[17] Analog Devices. Ad7476 7477 7478: Datasheet [online, visitado el 13 junio de 2013]. URL http://www.analog.com/static/imported-files/data_sheets/AD7476_7477_7478.pdf.

[18] Digilent. Digilent PmodAD1 Analog to Digital Module Converter Board. Reference Manual [online, visitado el 10 de agosto de 2013]. URL http://www.digilentinc.com/Data/Products/PMOD-AD1/Pmod%20AD1_rm.pdf.

[19] Digilent. Nexys 3 Spartan-6 FPGA Board [online, visitado el 10 de agosto de 2013]. URL http://www.digilentinc.com/Products/Detail.cfm?Prod=NEXYS3.

[20] Berny Dinarte. Diseño de una fuente de corriente auto-polarizada de ultra-baja potencia, independiente de la tensión de alimentación y la temperatura para circuitos integrados analógicos. Tesis de Licenciatura, Escuela de Ingeniería Electrónica, ITCR, Cartago, Costa Rica, 2011.

[21] G.L. Duckworth, J.E. Barger, S.H. Carlson, D.C. Gilbert, M.L. Knack, J. Korn, y R.J. Mullen. Fixed and wearable acoustic counter-sniper systems for law enforcement. In *SPIE International Symposium on Enabling Technologies for Law Enforcement and Security Sensors, C3I, Information, and Training Technologies for Law Enforcement, Proceedings of the*, págs. 3575–3577. Society of Photo-Optical Instrumentation Engineers, 1998.

[22] A. Dufaux, L. Besacier, M. Ansorge, y F. Pellandini. Automatic sound detection and recognitions for noisy environment. In *X European Signal Processing Conference, EUSIPCO 2000, Proceedings of the*, pág. 1889. TTKK-Paino, Tampere, Finlande, 1997. URL http://www-clips.imag.fr/geod/User/laurent.besacier/Publis/eusipco2000-2.pdf.

[23] A. Eleyan y H. Demirel. PCA and LDA based neural networks for human face recognition. *InTech*, 2007.

[24] A. Schliep et. al. GHMM library [online]. 2013 [visitado el 3 de abril de 2013]. URL http://ghmm.org/.

[25] C. Cannam et. al. QM-DSP [online]. 2013 [visitado el 3 de abril de 2013]. URL http://code.soundsoftware.ac.uk/projects/qm-dsp/repository/entry/hmm/hmm.c.

[26] P. Alvarado et. al. LTI-Lib [online]. 2012 [visitado el 3 de abril de 2013]. URL http://ltilib.sourceforge.net/doc/homepage/index.shtml.

[27] P. Davis et. al. JACK documentation [online]. 2013 [visitado el 3 de abril de 2013]. URL http://jackaudio.org/documentation.

[28] B.G. Ferguson, L.G. Criswick, y K.W. Lo. Locating far-field impulsive sound sources in air by triangulation. *The Journal of the Acoustical Society of America*, 111(1 Pt 1):104–16, 2002. URL http://www.ncbi.nlm.nih.gov/pubmed/11831786. PMID: 11831786.

[29] The Eclipse Foundation. Eclipse IDE [online]. 2013 [visitado el 3 de abril de 2013]. URL http://www.eclipse.org/.

[30] J. Ibarra García. Diseño de un filtro analógico para la detección de disparos de armas de fuego usando amplificadores operacionales de transconductancia. Tesis de Licenciatura, Escuela de Ingeniería Electrónica, ITCR, Cartago, Costa Rica, Junio 2011.

[31] D. H. Goldberg, A. G. Andreou, P. Julián, P.O. Pouliquen, L. Riddle, y R. Rosasco. VLSI implementation of an energy-aware wake-up detector for an acoustic surveillance sensor network. *ACM Trans. Sen. Netw.*, 2(4):594–611, 2006. URL http://portal.acm.org/citation.cfm?id=1218556.1218562.

[32] R. Gregorian y G. Temes. *Analog MOS Integrated Circuits for Signal Processing*. Wiley Interscience, 1ra edición, 1986.

[33] Néstor Hernández. Diseño de una red inalámbrica de telecomunicaciones para la protección ambiental en el bosque. Informe Final. Vicerrectoría de Investigación, ITCR, Julio 2007.

[34] Luis Adolfo Alfaro Hidalgo. Implementación en hardware del sistema de reconocimiento de patrones acústicos (sirpa). Tesis de Licenciatura, Escuela de Ingeniería Electrónica, ITCR, 2013.

[35] IEEE. IEEE Standard for Information Technology– Local and metropolitan area networks– Specific requirements– part 15.4: Wireless medium access control (MAC) and Physical layer (PHY) Specifications for Low Rate Wireless Personal Area Networks (WPANs). *IEEE Std 802.15.4-2006 (Revision of IEEE Std 802.15.4-2003)*, págs. 1–320, 2006.

[36] Texas Instruments. TMS320C64x Technical Overview [online]. 2001 [visitado el 3 de abril de 2013]. URL http://www.ti.com/lit/ug/spru395b/spru395b.pdf.

[37] Texas Instruments. TMS320C64x/c64x+ DSP CPU and Instruction Set [online]. 2008 [visitado el 3 de abril de 2013]. URL http://www.ti.com/lit/ug/spru198k/spru198k.pdf.

[38] Texas Instruments. C6RunLib documentation [online]. 2011 [visitado el 3 de abril de 2013]. URL http://processors.wiki.ti.com/index.php?title=C6RunLib_Documentation.

[39] Texas Instruments. DM3730, DM3725 digital media processors [online]. 2011 [visitado el 3 de abril de 2013]. URL http://www.ti.com/lit/ds/symlink/dm3730.pdf.

[40] Texas Instruments. TMS320C6000 programmer's guide [online]. 2011 [visitado el 3 de abril de 2013]. URL http://www.ti.com/lit/ug/spru198k/spru198k.pdf.

[41] Maxim Integrated. Microphone Amplifier with AGC and Low-Noise Microphone Bias [online, visitado el 13 de junio de 2013]. URL http://www.maximintegrated.com/datasheet/index.mvp/id/5433.

[42] Texas Intruments. cc2420 2.4 ghz ieee 802.15.4 / zigbee-ready rf transceiver [online, visitado el 12 de junio de 2014]. URL http://www.ti.com/lit/ds/symlink/cc2420.pdf.

[43] D. Istrate, E. Castelli, M. Vacher, L. Besacier, y J. F. Serignat. Information extraction from sound for medical telemonitoring. *IEEE Transactions on Information Technology in Biomedicine*, 10(2):264–274, April 2006.

[44] P. Julian, F.N.M. Pirchio, y A.G. Andreou. Experimental results for cascadable micropower time delay estimator. *Electronics Letters*, 42(21):1218–1219, 2006.

[45] T. Kaneko, Y. Akazawa, y M. Nagatani. Switched capacitor and active-RC filter layout using a parameterizable generator. In *Circuits and Systems, 1991., IEEE International Sympoisum on*, págs. 2012–2015 vol.4, 1991.

[46] Á. Lédeczi, P. Völgyesi, M. Maróti, G. Simon, G. Balogh, A. Nádas, B. Kusy, S. Dóra, y G. Pap. Multiple simultaneous acoustic source localization in urban terrain. In *Proceedings of the 4th international symposium on Information processing in sensor networks*, pág. 69, Los Angeles, California, 2005. IEEE Press. URL http://portal.acm.org/citation.cfm?id=1147771.

[47] C. Galup-Montoro M. Camacho-Galeano y M. Cherem Schneider. A 2-nW 1.1-V Self-Biased Current Reference in CMOS Technology. *IEEE Transactions on Circuits and Systems II: Express Briefs*, 52:61–65, feb 2005.

[48] C. Galup-Montoro M. Camacho-Galeano y M. Cherem Schneider. Temperature performance of sub-1V ultra-low power current sources. *IEEE Transcations on Circuits and Systems II: Express Briefs*, 52:2230–2233, may 2008.

[49] D. MacKay. *Information Theory, Inference and Learning Algorithms*. Cambridge, 2003.

[50] R. Maher. Summary of gun shot acoustics. *Montana State University*, Abril 2006.

[51] H. Mamitsuka. Supervised learning of hidden markov models for sequence discrimination. *Proceedings of the first annual international conference on Computational molecular biology*, 1997.

[52] T. P. Mann. Numerically stable Hidden Markov Model implementation. 2006.

[53] memsic. Micaz wireless mesurement system data sheet [online, visitado el 12 de junio de 2014]. URL http://www.memsic.com/userfiles/files/DataSheets/WSN/micaz_datasheet-t.pdf.

[54] K. Molnár, Á. Lédeczi, L. Sujbert, y G. Péceli. Muzzle blast detection via short time Fourier transform. Online. http://home.mit.bme.hu/~kmolnar/index.html, 2005. URL http://home.mit.bme.hu/~kmolnar/index.html.

[55] J. Montero. Implementación de un sistema de reconocimiento de patrones acústicos de disparos y motosierras en un sistema en un sistema embebido. Tesis de Licenciatura, Escuela de Ingeniería Electrónica, ITCR, Abril 2013.

[56] Jordan Montero, Luis A. Alfaro, y José A. Zúñiga. Proyecto de red inalámbrica de sensores ad hoc (risah). Technical report, Escuela de Ingeniería Electrónica, ITCR, Cartago, Costa Rica, Noviembre 2012.

[57] A. Moore. An intoductory tutorial on kd-trees. Tesis de Doctorado, Computer Laboratory, University of Cambridge, 1991.

[58] F. Nicaragua. Diseño de un amplificador operacional de transconductancia para la implementación de filtros analógicos utilizados en la detección de disparos de armas de fuego. Tesis de Licenciatura, Escuela de Ingeniería Electrónica, ITCR, Cartago, Costa Rica, Noviembre 2010.

[59] Nokia. Qt development frameworks [online]. 2013 [visitado el 3 de abril de 2013]. URL http://qt.digia.com/Product/Developer-Tools/.

[60] National Institute of Justice. Random gunfire problems and gunshot detection systems. Online. http://www.ojp.usdoj.gov/nij/pubs-sum/179274.htm. URL http://www.ojp.usdoj.gov/nij/pubs-sum/179274.htm.

[61] S. M. Omohundro. Efficient algorithms with neural network behavior. *University of Illinois at Urbana-Champaign*, 1987.

[62] A. Oppenheim, R. Schafer, y J. Buck. *Discrete-Time Signal Processing*. Prentice Hall, 1999.

[63] C. Loría Padilla. Sistema para el acondicionamiento de señal para reconocimiento de patrones acústicos de motosierras y disparos en el bosque. Tesis de Licenciatura, Escuela de Ingeniería Electrónica, ITCR, Cartago, Costa Rica, 2013.

[64] R. Pereira-Arroyo. *Optimización de*. PhD thesis, Universidad Nacional Estatal a Distancia, Instituto Tecnológico de Costa Rica, 2014.

[65] Roberto Pereira-Arroyo, Pablo Alvarado-Moya, y Wolfgang H. Krautschneider. Design of a mcml gate library applying multiobjective optimization. In *Proceedings of the IEEE Computer Society Annual Symposium on VLSI*, págs. 81–85, Washington, DC, USA, 2007. IEEE Computer Society.

[66] Allan Zúñiga Porras. Diseño de un circuito microelectrónico para la detección de disparos mediante filtros digitales. Tesis de Licenciatura, Escuela de Ingeniería Electrónica, ITCR, Cartago, Costa Rica, 2012.

[67] GNU Project. GCC, the GNU compiler collection [online]. 2013 [visitado el 3 de abril de 2013]. URL http://gcc.gnu.org/.

[68] M. Puckette. Envelope following [online]. 2006 [visitado el 3 de abril de 2013]. URL http://crca.ucsd.edu/~msp/techniques/v0.11/book-html/node153.html.

[69] L.R. Rabiner. A tutorial on Hidden Markov Models and selected applications in speech recognition. In *Proceedings of the IEEE*, volumen 77(2), págs. 257–286, February 1989.

[70] Vijay Raghunathan, Curt Schurgers, Sung Park, y Mani B. Srivastava. Energy-aware wireless microsensor networks. *IEEE Signal Processing Magazine*, págs. 40–50, Marzo 2002.

[71] Ronny García Ramírez. Diseño y evaluación de un protocolo de comunicación de bajo consumo energético para una red inalámbrica de sensores para la monitorización ambiental. Tesis de maestría, Escuela de Ingeniería en Computación, ITCR, 2013.

[72] B. Razavi. *Design of Analog CMOS Integrated Circuits*. McGraw-Hill Science/Engineering/Math, 1 edición, August 2000.

[73] A. Chacón Rodríguez, F. Martin-Pirchio, S. Sañudo, y P. Julián. A low power integrated circuit for interaural time delay estimation without delay lines. *Circuits and Systems II: Brief Papers, IEEE Transactions on*, Aceptado para publicación.

[74] Jerome Rousselot. Wiseroute [online, visitado el 12 de junio de 2014]. URL http://mixim.sourceforge.net/doc/MiXiM/doc/doxy/a00227.html.

[75] Jerome Rousselot, Jean-Dominique Decotignie, Marc Aoun, Peter van der Stok, Ramon Serna Oliver, y Gerhard Fohler. Accurate timeliness simulations for real-time wireless sensor networks. págs. 476–481, 2009. URL http://dx.doi.org/10.1109/EMS.2009.34.

[76] B. Sadler, L. Sadler, y T. Pham. Optimal and robust shockwave detection and estimation. In *Proceedings of the 1997 IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP '97)-Volume 3 - Volume 3*, pág. 1889. IEEE Computer Society, 1997. URL http://portal.acm.org/citation.cfm?id=839593.

[77] Erick Salas. Reconocimiento en tiempo real de patrones acústicos de motosierras y disparos por medio de una implementación en FPGA de Modelos Ocultos de Markov. Tesis de Licenciatura, Escuela de Ingeniería Electrónica, ITCR, Cartago, Costa Rica, Abril 2010.

[78] T. Saramäki y R. Bregović. *Multirate Systems and Filter Banks*. Idea Group Publishing, 2010.

[79] C. Schurgers, V. Tsiatsis, S. Ganeriwal, y M. Srivastava. Optimizing sensor networks in the energy-latency-density design space. *Mobile Computing, IEEE Transactions on*, 1(1):70–80, 2002.

[80] Mario Sequeira. Módulo de reducción de dimensiones espectrales en un sistema de reconocimiento de patrones acústicos de motosierras y disparos por medio de una implementación en FPGA. Tesis de Licenciatura, Escuela de Ingeniería Electrónica, ITCR, Cartago, Costa Rica, Junio 2011.

[81] D. Shen. Some mathematics for HMM. *Massachusetts Institute of Technology*, 2008.

[82] Esteban Smith. Reconocimiento digital en línea de patrones acústicos para la protección del ambiente por medio de HMM. Tesis de Licenciatura, Escuela de Ingeniería Electrónica, ITCR, Cartago, Costa Rica, Enero 2008.

[83] M. Stanacevic y G. Cauwenberghs. Mixed-signal gradient flow bearing estimation. In *Circuits and Systems, 2003. ISCAS '03. Proceedings of the 2003 International Symposium on*, volumen 1, págs. I–777–I–780 vol.1, 2003.

[84] R. Stoughton. Measurements of small-caliber ballistic shock waves in air. *The Journal of the Acoustic Society of America*, 102(2):781–787, August 1997.

[85] F. G. Stremler. *Introducción a los sistemas de comunicación*. Addison Wesley Longman, 1993.

[86] Gabriela Sáenz. Reconocimiento de patrones acústicos para la protección del ambiente utilizando wavelets y modelos ocultos de Markov. Tesis de Licenciatura, Escuela de Ingeniería Electrónica, ITCR, Cartago, Costa Rica, Noviembre 2006.

[87] Andrew S. Tanenbaum. *Redes de computadoras 4ta edicion*. Prentice Hall, 2003.

[88] S. Theodoridis y K. Koutroumbas. *Pattern Recognition*. Elsevier, 2009.

[89] R. F. Tobler. The rkd-tree. *Computer Graphics International*, 2011.

[90] J. Tsiombikas. kdtree [online]. 2012 [visitado el 3 de abril de 2013]. URL http://code.google.com/p/kdtree/.

[91] Y. Tsividis. *Mixed Analog-Digital VLSI Devices and Technology*. World Scientific Publishing Co. Pte. Ltd, 2002.

[92] J. Valverde. Implementación de una metodología para lograr la integración física " correcta por construcción (CBS)" de un banco de filtros digitales descrito en alto nivel. Tesis de Licenciatura, Escuela de Ingeniería Electrónica, ITCR, Junio 2011.

[93] A. Villegas. Diseño de un detector de sonidos de motosierra con bajo consumo energético. Tesis de Licenciatura, Escuela de Ingeniería Electrónica, ITCR, Cartago, Costa Rica, 2013.

[94] P.G. Weissler y M. T. Kobal. Noise of police firearms. *The Journal of the Acoustical Society of America*, 56(5):1515–1522, November 1974. URL http://link.aip.org/link/?JAS/56/1515/1.

[95] Q. Wu, F. Merchant, y K. Castleman. *Microscope Image Processing*. Elsevier, 2008.

[96] Jennifer Yick, Biswanath Mukherjee, y Dipak Ghosal. Wireless sensor network survey. *Computer Networks*, 52:2292–2330, 2008.

[97] M. Zu, P. Su, R. Shi, W. Wang, y J. Yu. Intelligent tracer of hunting activities. Lily Studio, University of Nanjing, June 2006.