

UNIVERSIDAD DE MÁLAGA

ESCUELA DE INGENIERÍAS INDUSTRIALES

TRABAJO FIN DE MÁSTER

Diseño e implementación de la arquitectura de un procesador para la detección del deslizamiento de objetos sobre una matriz de sensores táctiles controlado por una FPGA

Tutor:

Dr. Manuel Jesús Martín

Vázquez

Autor:

Daniel Rodríguez Criado

Co-tutor:

Dr. Óscar Oballe Peinado



26 de junio de 2019

Declaración de originalidad

Yo, Daniel Rodríguez Criado, por la presente **declaro** que excepto las referencias específicas al trabajo de otros, el contenido de este trabajo titulado “ DISEÑO E IMPLEMENTACIÓN DE LA ARQUITECTURA DE UN PROCESADOR PARA LA DETECCIÓN DEL DESLIZAMIENTO DE OBJETOS SOBRE UNA MATRIZ DE SENSORES TÁCTILES CONTROLADO POR UNA FPGA”, es original y no ha sido plagiado ni total ni parcialmente. No ha sido autoplagiado, es decir, no ha sido publicada ni presentada anteriormente para obtener algún grado académico previo o título profesional. Los datos presentados en los resultados son reales, no han sido falseados, ni duplicados, ni copiados y por lo tanto los resultados que se presenten en este trabajo fin de máster se constituirán en aportes de la realidad investigada.

Firmado:

Fecha 26 de junio de 2019

Agradecimientos

En primer lugar, quiero agradecer a mis dos tutores Dr. Manuel Jesús Martín Vázquez y Dr. Óscar Oballe Peinado del departamento de electrónica de la UMA por toda la ayuda prestada durante el desarrollo del trabajo, su guía en los momentos en los que andaba perdido y por todo lo que me han enseñado. En general agradecer al departamento por darme la oportunidad de realizar este proyecto.

También a los alumnos de laboratorio del departamento que siempre se han mostrado dispuestos a resolverme todas las dudas que me han surgido en el momento y me han aportado ideas muy útiles.

Por último, me gustaría también agradecer a mi familia y amigos por todo el apoyo recibido durante mi etapa académica y por animarme a seguir adelante.

Resumen

UNIVERSIDAD DE MÁLAGA

ESCUELA DE INGENIERÍAS INDUSTRIALES

Departamento de Electrónica

Master en Ingeniería Industrial

**Diseño e implementación de la arquitectura de un procesador para la
detección del deslizamiento de objetos sobre una matriz de sensores
táctiles controlado por una FPGA**

por Daniel Rodríguez Criado

Este trabajo describe el desarrollo del diseño y la implementación de un algoritmo capaz de detectar el deslizamiento con los datos recogidos por una matriz de sensores táctiles. El principal objetivo es el de identificar las etapas tempranas de este deslizamiento para que una vez instalado en un mano robótica funcional, se pueda reaccionar con antelación a la caída del objeto sujetado. Todo esto usando MATLAB para el diseño del algoritmo y el análisis de los datos; el sistema de FPGA y matriz de sensores táctiles para implementar la lógica; y una aplicación en LabVIEW para la representación en tiempo real de los datos arrojados por la FPGA.

Nomenclatura

| | |
|---------------|-----------------------------------------------------|
| TFM | Trabajo Fin de Máster |
| FPGA | Field-Programmable Gate Array |
| VHDL | VHSIC Hardware Description Language |
| VHSIC | Very High Speed Integrated Circuit |
| MIS | Minimally Invasive Surgery |
| EIS | Electrónica para la instrumentación y sistemas |
| PCB | Printed Circuit Board |
| CAD | Computer-Aided Design |
| CAM | Computer-Aided Manufacturing |
| UMA | Universidad de Málaga |
| LED | Light Emitting Diode |
| PROM | Programmable Read-Only Memory |
| EEPROM | Electrically Erasable Programmable Read-Only Memory |
| USB | Universal Serial Bus |
| DSP | Digital Signal Processor |
| FIR | Finite Impulse Response |
| IIR | Infinite Impulse Response |
| LP | Low Pass (Filter) |
| HP | High Pass (Filter) |

FFT Fast Fourier Transform
SPI Serial Peripheral Interface

Índice

| | |
|----------------------------------------------|------------|
| Declaración de originalidad | III |
| Agradecimientos | V |
| Resumen | VII |
| Nomenclatura | IX |
| | |
| I Introducción | 1 |
| Introducción y visión general | 3 |
| Objetivos | 4 |
| Estado del arte | 5 |
| Metodología y directrices seguidas | 8 |
| Estructura del documento | 9 |
| | |
| II Desarrollo del proyecto | 11 |
| 1 Descripción del equipo de trabajo | 13 |
| 1.1 Software utilizado | 14 |
| 1.2 Equipo electrónico | 15 |
| 1.2.1 Módulo de control | 16 |
| 1.2.2 Sensor del dedo | 18 |

| | | |
|----------|---------------------------------------------------------------|-----------|
| 1.3 | Materiales adicionales y conectores | 20 |
| 1.3.1 | Láminas de silicona y sus moldes | 20 |
| 1.3.2 | Lámina piezoresistiva | 21 |
| 1.3.3 | Cables y conectores | 22 |
| 1.3.4 | Objetos a deslizar | 23 |
| 1.4 | Banco de pruebas | 25 |
| 2 | Análisis de los algoritmos utilizados | 27 |
| 2.1 | Algoritmo en MATLAB | 28 |
| 2.1.1 | Análisis de los datos con MATLAB | 29 |
| 2.2 | Algoritmo en ISE Design Suite | 30 |
| 2.2.1 | Implementación del filtro en la FPGA | 32 |
| 2.2.2 | Círculo de umbral de detección | 33 |
| 2.3 | Algoritmo en LabVIEW | 36 |
| 2.3.1 | Modificaciones al programa | 37 |
| 3 | Teoría de filtros digitales | 41 |
| 3.1 | Filtros digitales | 42 |
| 3.1.1 | Rango de frecuencias y teorema de muestreo de Nyquist-Shannon | 42 |
| 3.1.2 | Clasificación en función de la frecuencia | 44 |
| 3.1.3 | Filtros FIR e IIR | 47 |
| 3.2 | Diseño e implementación digital de un filtro FIR | 50 |
| 3.2.1 | Definición y funcionamiento | 50 |
| 3.2.2 | Obtención de los coeficientes | 51 |
| 3.2.3 | Métodos de diseño | 55 |
| 3.3 | Asistente de diseño de filtros en MATLAB | 60 |
| 3.3.1 | Método de diseño de filtros FIR | 61 |
| 3.3.2 | Resultado final, coeficientes | 62 |
| 4 | Experimentación | 65 |

| | | |
|---------------------|-------------------------------------------------------------------|------------|
| 4.1 | Resumen del proceso | 66 |
| 4.2 | Análisis de datos | 67 |
| 4.2.1 | Respuesta en el tiempo del sensor ante el deslizamiento | 67 |
| 4.2.2 | Espectro en frecuencia | 71 |
| 4.3 | Diseño del filtro | 75 |
| 4.3.1 | Diferentes aproximaciones de diseño | 75 |
| 4.3.2 | Filtros usados | 77 |
| 4.4 | Ajuste experimental del filtro | 81 |
| 5 | Análisis de los resultados y validación del algoritmo | 85 |
| 5.1 | Filtro utilizado | 86 |
| 5.2 | Experimentos con el proyecto final | 87 |
| 5.2.1 | Test de deslizamiento | 87 |
| 5.2.2 | Test de fuerza | 90 |
| 5.3 | Conclusiones | 93 |
| 5.4 | Trabajo futuro | 94 |
| III | Apéndices | 95 |
| A | Código MATLAB | 97 |
| B | Código VHDL | 105 |
| B.1 | Modificaciones en el módulo sensor | 106 |
| B.2 | Modificaciones en el módulo TOP | 109 |
| Bibliografía | | 112 |

Índice de figuras

| | | |
|------|----------------------------------------------------------------------------|----|
| 1 | Mano de Barrett | 5 |
| 2 | Modelo de FPGA usada en el proyecto. Xiling Spartan 6. | 6 |
| 1.1 | Logos de los diferentes programas utilizados. | 14 |
| 1.2 | Mano de Barrett del trabajo de Óscar Oballe. <i>Fuente: [11]</i> | 15 |
| 1.3 | Módulo de control del sistema. <i>Fuente: [11]</i> | 16 |
| 1.4 | Esquema de los puertos de expansión SPI. <i>Fuente: [11]</i> | 17 |
| 1.5 | PCB del sensor de presión. <i>Fuente: [11]</i> | 18 |
| 1.6 | Aerosol de silicona usado para los experimentos. | 20 |
| 1.7 | Proceso de fabricación de los moldes de silicona. | 21 |
| 1.8 | Lámina piezoresistiva aplicada sobre el sensor. | 21 |
| 1.9 | Conectores para la experimentación. | 22 |
| 1.10 | Programador convertidor de USB a JTAG. | 22 |
| 1.11 | Carrito y contrapeso unidos por un hilo. | 23 |
| 1.12 | Pesa usada para rodadura. | 24 |
| 1.13 | Dedo deslizando sobre el sensor. | 24 |
| 1.14 | Banco de pruebas completo. | 25 |
| 2.1 | Diagrama del algoritmo implementado en MATLAB. | 28 |
| 2.2 | Asistente de análisis de señales en MATLAB. | 29 |
| 2.3 | Esquemático de más alto nivel. | 30 |
| 2.4 | Esquemático del módulo sensor. | 31 |
| 2.5 | Compilador de un bloque de filtro FIR del ISE Design Suite. | 32 |

| | | |
|------|----------------------------------------------------------------------------------------------------------|----|
| 2.6 | Comunicación entre módulos del circuito. | 33 |
| 2.7 | Generación de la señal de detección de deslizamiento. | 34 |
| 2.8 | Lógica de generación de la imagen que se envía al ordenador. | 35 |
| 2.9 | Diagrama de bloques de la aplicación en LabVIEW. | 36 |
| 2.10 | Panel gráfico original de la interfaz en LabVIEW. | 37 |
| 2.11 | Interfaz de LabVIEW modificada. | 38 |
| 2.12 | Diagrama de bloques original y modificaciones. | 39 |
| 2.13 | Nuevo módulo creado para la representación de la matriz de LEDs. . | 40 |
| 3.1 | Representación ideal de los diferentes tipos de filtro básicos. | 44 |
| 3.2 | Diagrama Atenuación-Frecuencia de un filtro paso de baja. Fuente [5] | 45 |
| 3.3 | Diagrama que muestra el uso de un filtro paso de bajo para eliminar la componente DC. | 46 |
| 3.4 | Esquemas de implementación de filtro FIR e IIR | 48 |
| 3.5 | Diagrama de bloques de la implementación de un filtro FIR. | 51 |
| 3.6 | Respuesta en frecuencia (a) y en el tiempo (b) de un filtro LP ideal y normalizado. Fuente [21]. | 52 |
| 3.7 | Función del seno cardinal $\text{sinc}(x)$ discretizada. | 52 |
| 3.8 | Diferentes tipos de filtros ideales y normalizados en frecuencia. | 54 |
| 3.9 | Efecto del truncamiento de la señal respuesta al impulso. | 56 |
| 3.10 | Proceso de aplicación de una ventana tipo Blackman. | 57 |
| 3.11 | Respuestas en frecuencia para diferentes tipos de ventana. | 58 |
| 3.12 | Respuestas en frecuencia para diferentes anchos de ventana. | 58 |
| 3.13 | Asistente de diseño de filtros en MATLAB. | 60 |
| 3.14 | Métodos de diseño y tipos de ventanas implementadas por el asistente de MATLAB. | 61 |
| 3.15 | Menú de opciones del método de ventanado. | 61 |
| 3.16 | Menú de opciones del método de equiripple. | 62 |
| 3.17 | Coeficientes generados por el asistente de MATLAB. | 62 |
| 4.1 | Esquema del flujo del proceso de experimentación seguido. | 66 |

| | | |
|------|-----------------------------------------------------------------------------------------------------------------|----|
| 4.2 | Fases del deslizamiento en un solo táctel. | 67 |
| 4.3 | Otro ejemplo de deslizamiento, esta vez con pesa de $1Kg$ | 68 |
| 4.4 | Otro ejemplo de deslizamiento, esta vez usando el dedo. | 69 |
| 4.5 | Lectura de un táctel que se muestra en una zona intermedia de la matriz. | 69 |
| 4.6 | Representación de las lecturas de todos los sensores durante el deslizamiento. | 70 |
| 4.7 | Análisis en frecuencia de la lectura de un táctel. Deslizamiento del carrito. | 71 |
| 4.8 | Análisis en frecuencia de la lectura de un táctel. Deslizamiento del rodillo en vertical sobre su base. | 72 |
| 4.9 | Análisis en frecuencia de la lectura de un táctel. Deslizamiento de un dedo. | 73 |
| 4.10 | Elementos principales en frecuencia de un filtro LP. Fuente [7]. | 75 |
| 4.11 | Especificaciones de un filtro FIR representadas como un triángulo. Fuente [7]. | 76 |
| 4.12 | Especificaciones de filtro paso de alta usando una ventana Kaiser. . . | 77 |
| 4.13 | Especificaciones de filtro paso de alta usando una ventana Hamming. . | 78 |
| 4.14 | Método de equiripple para diseño de filtro paso de alta fijando el orden. . | 78 |
| 4.15 | Método de equiripple para diseño de filtro paso de alta marcando la opción de orden mínimo. | 79 |
| 4.16 | Especificaciones de filtro paso de alta usando el método de equiripple con constricciones. | 80 |
| 4.17 | Ejemplo de señal de fuerza y salida del filtro paso de alta. | 81 |
| 4.18 | Progresión de valores máximos de amplitud con la frecuencia de corte. . | 82 |
| 4.19 | Progresión de componente en continua con la frecuencia de corte. . | 83 |
| 5.1 | Representación y especificaciones del filtro final elegido para el proyecto. . | 86 |
| 5.2 | Momentos antes y al inicio del deslizamiento. | 88 |
| 5.3 | Momentos durante y al final del deslizamiento. | 89 |
| 5.4 | Experimentación de fuerza con la pesa de 1 Kg. | 91 |

| | | |
|-----|-------------------------------------------------------------------------------------------------------------|-----|
| 5.5 | 5 Kg apoyados sobre el sensor usando la cara de menor superficie de la pesa. | 92 |
| B.1 | Puertos de entrada y salida del módulo sensor. | 106 |
| B.2 | Nuevas señales creadas y multiplexor. | 106 |
| B.3 | Modificaciones en el contador de columna y contador de fila. | 107 |
| B.4 | Modificación en la máquina de estados que hace la lectura y escritura de datos. | 107 |
| B.5 | Comparadores con los umbrales y la salida del filtro y definición de la parte alta de los umbrales. | 108 |
| B.6 | Conexiones entre el módulo sensor y el módulo SPI. | 109 |

Índice de Tablas

| | | |
|-----|-------------------------------------------------------------------------|----|
| 3.1 | Tabla de tres mecanoreceptores que intervienen en el sentido del tacto. | 42 |
| 3.2 | Comparativa entre filtros FIR e IIR. | 47 |
| 3.3 | Propiedades de las diferentes ventanas. | 55 |
| 4.1 | Resultado de pruebas de frecuencia para deslizamiento con carrito. . . | 82 |
| 5.1 | Coeficientes del filtro final. | 87 |

Parte I

Introducción

Introducción y visión general

Contenido

| | |
|----------------------------------------------|---|
| Objetivos | 4 |
| Estado del arte | 5 |
| Metodología y directrices seguidas | 8 |
| Estructura del documento | 9 |

Sinopsis

Éste es el capítulo de introducción, donde se explica todo lo que un lector externo necesita para entender el resto de la documentación.

Objetivos

El objetivo final de este proyecto será el de detectar el deslizamiento de un objeto en tiempo real sobre el hardware existente con el fin de poder incorporar este dispositivo a una mano robótica funcional.

A continuación se detallan los objetivos secundarios de una forma más específica y ordenada para alcanzar el objetivo principal:

- Realizar experimentos en un banco de pruebas consistente en un plano inclinado en el que deslizara un objeto sobre la matriz de sensores con el objetivo de obtener datos para su posterior análisis.
- Utilizar los datos obtenidos en el objetivo anterior para realizar un programa en MATLAB que simule el hardware que se implementaría posteriormente.
- Diseñar un procesador específico para analizar las señales originadas en una matriz de sensores de presión.
- Implementar el procesador sobre el hardware existente el cual consiste en una matriz de sensores y su lógica de control implementada sobre una FPGA usando VHDL.

Estado del arte

Los sensores táctiles están presentes en la vida diaria de cualquier persona como por ejemplo en botones de ascensores, en el smartphone o incluso en televisores. Además de muchas otras aplicaciones de las que la gente no es consciente. En este proyecto se hará uso de estos sensores con la intención de instalarlos en una mano robótica tipo Barrett similar a la que se muestra en la figura 1.

Los mencionados sensores táctiles, miden la presión o la fuerza a la que son sometidos basándose en diferentes principios, como el capacitivo, resistivo o métodos ópticos [3]. Además, se disponen en matrices para abarcar una mayor área y suele ser necesario implementar una electrónica local para reducir los errores e interferencias en la medición, realizando una función de acondicionamiento de la señal de referencia.

En lo que se refiere a los robots diseñados para interactuar con objetos con precisión o destreza, necesitan de estos transductores para dotar a la máquina del sentido del tacto. Dependiendo de la aplicación, estos pueden ser más o menos sensibles para captar pequeñas variaciones, así como, variar en tamaño y forma para adaptarse a las diferentes superficies donde serán colocados. En esta aplicación se usarán matrices de sensores de presión conectados en una lámina flexible la cual puede adaptarse a la superficie de la mano de un robot o a unas pinzas como los mostrados en [12].

Conferir a un robot el sentido del tacto puede tener el cometido de complementar otros sistemas de percepción en los autómatas como los de detección visual. En este sentido, cuando el robot agarra un objeto, los sensores de presión pueden detectar características que la visión no puede como el peso, la textura o el momento en el



Figura 1: Mano de Barrett¹

¹Fuente de la imagen: <https://robots.ros.org/barrett-hand/>

que va a empezar a deslizar. Esta última función es en la que se enfoca el presente proyecto.

En el caso del deslizamiento, cuando este comienza, aparecen unas pequeñas vibraciones de alta frecuencia. Este principio de partida en el que se fundamentan los estudios desarrollados en este trabajo, ha sido demostrado con anterioridad en múltiples trabajos realizados en diferentes universidades. Sin embargo, se ha querido destacar los llevados a cabo en el grupo de investigación en el que se desarrolla este proyecto, como [9] o [12]. Para detectar estas micro vibraciones, será necesario emplear un sistema que filtre la señal que proviene de la lectura de los sensores y reconozca cuando las señales de alta frecuencia son importantes.



Figura 2: Modelo de FPGA usada en el proyecto. Xiling Spartan 6.²

La implementación del algoritmo de detección de deslizamiento se hará sobre una FPGA (Field Programmable Gate Array). Estos dispositivos son muy flexibles ya que permiten reconfigurar su hardware. La mayor ventaja de esta estrategia es la posibilidad de realizar tareas bastante complejas en tiempo real gracias a la capacidad que tienen las FPGAs para el tratamiento de datos en paralelo. Debido a estas características se convierte en una buena opción para implementar los filtros necesarios en la detección del deslizamiento. El procesamiento en tiempo real de las señales será un objetivo crucial en este diseño.

Uno de los propósitos de la comunidad científica, desde el comienzo de la investigación con robots, ha sido dotar a estos de los diferentes sentidos que tiene el ser humano. El interés por los sensores táctiles ha ido incrementándose en los últimos años debido a la demanda en diferentes entornos como la cirugía mínimamente invasiva o MIS (Minimally Invasive Surgery), la robótica, telepresencia, etc. Por otro lado, aunque se han fabricado muchas unidades enfocadas a este campo, el desarrollo de esta tecnología es lento. Por esta razón, este proyecto podría tener un componente innovador.

²Fuente de la imagen: <https://www.xilinx.com>

Los sensores actuales son básicamente matrices pasivas que no implementan ningún procesamiento on chip (es decir integrado en el mismo silicio del sensor). Cuando se necesita que la aplicación sea en tiempo real y la matriz tiene una gran densidad de sensores, se hace indispensable que exista algún tipo de preprocesado electrónico al mismo nivel que el sensor, consiguiendo transmitir menos datos innecesarios a la unidad de control [16].

Si hablamos del control del deslizamiento, hay antecedentes de estudios que utilizan una estimación del coeficiente de fricción para calcular cuando va a comenzar a resbalar el objeto [14]. Sin embargo, esta solución requiere de sensores más complejos y es difícil calcular este coeficiente si el robot tiene que agarrar diferentes objetos con diferente tipo de superficies. Otra solución es la implementación explícita del coeficiente de fricción. No obstante, sus resultados y aplicaciones están limitadas debido al gran número de variables involucradas. Por último, el método que da mejores resultados es el mencionado anteriormente, que detecta las microvibraciones que se producen justo antes del comienzo del deslizamiento [9]. Este sería el enfoque del presente trabajo. En el equipo de investigación EIS (Electrónica para la Instrumentación y Sistemas) de la universidad de Málaga se han venido realizando estudios en este sentido. Como fruto de este trabajo se han publicado diversos artículos entre los cuales se encuentran los siguientes:

- En [16] se analiza cual es la mejor electrónica para el manejo y la toma de datos de una matriz de sensores de fuerza. Se estudia el rendimiento de varios procesadores para este tipo de aplicación: ASICs (Integrado en el mismo chip), Microprocesadores, PSoCs y FPGAs. Se concluye que, para este tipo de aplicación, las FPGAs dan el mejor resultado debido a su capacidad para el procesamiento en paralelo y su hardware reconfigurable.
- Por otro lado, en [9] se estudian las condiciones que se dan justo antes del comienzo del deslizamiento: Vibraciones producidas por secuencias de agarre y resbalamiento.

Por último, mencionar que este proyecto será continuación de los diferentes trabajos realizados por el equipo de investigación EIS con el objetivo de mejorar la detección del deslizamiento.

Metodología y directrices seguidas

Se organiza un plan de trabajo basado en los siguientes hitos o procesos que conllevarán el cumplimiento de los objetivos mencionados en el apartado anterior:

- 1 Recopilación de información y conocimientos previos como punto de partida del proyecto:
 - Estudio de la arquitectura de la FPGA a utilizar y el entorno de programación ISE de Xilinx.
 - Aprendizaje del lenguaje de programación VHDL.
 - Recopilación de información sobre los algoritmos de detección del deslizamiento.
- 2 Documentación para la comprensión del hardware que proporciona el departamento (conjunto de FPGA y sensores):
 - Comportamiento y composición de la matriz de sensores de presión.
 - Comprender los algoritmos de obtención de datos ya implementados en la FPGA.
 - Estudiar artículos del equipo de investigación EIS donde se detalla el hardware de partida.
- 3 Toma de datos experimentales. Generación de archivos de datos a partir de diversos experimentos en los que se desplaza un objeto sobre la matriz de sensores en un plano inclinado.
- 4 Usar los datos obtenidos para diseñar el algoritmo usando MATLAB.
- 5 Implementación del algoritmo seleccionado en VHDL sobre el hardware existente.
- 6 Pruebas de evaluación en tiempo real.

Estructura del documento

Este documento se divide en cuatro partes para una mejor comprensión del mismo. Cada una de estas partes se listan a continuación:

- **Parte I Introducción:** Esta primera parte introduce los objetivos que se quieren conseguir con la ejecución de este proyecto. Así mismo, se introduce un marco tecnológico y temporal que ayuda a situar esta investigación en el ámbito científico actual.
- **Parte II Desarrollo del proyecto:** esta es la parte donde se desarrolla el grueso del trabajo. Está dividida en una serie de capítulos que se enumeran seguidamente:
 - *Capítulo 1:* En este capítulo se describirán todos los elementos, tanto hardware como software, utilizados para llevar a cabo el proyecto.
 - *Capítulo 2:* Todos los algoritmos implementados en los diferentes programas usados en este proyecto serán descritos en este capítulo.
 - *Capítulo 3:* Aquí se hará una breve clasificación de los diferentes filtros y las técnicas de diseño para posteriormente indicar el desarrollo seguido para la creación del filtro en MATLAB.
 - *Capítulo 4:* Una vez que se ha analizado la teoría de los filtros digitales, en este capítulo, se explica como se pasa a la práctica para el diseño y la implementación física de un filtro FIR en un dispositivo real a través de todo el proceso de experimentación realizado en el presente trabajo.
 - *Capítulo 5:* En este último capítulo se muestra la solución final del proyecto: filtro utilizado, programa para la visualización de los datos y detección del deslizamiento. Se hará un análisis del funcionamiento del sistema y el porqué de la solución escogida. Así mismo, se incluye una parte final que describe las conclusiones que se pueden sacar de las soluciones encontradas en el desarrollo de este trabajo fin de máster. Además, se proponen diferentes tareas como líneas futuras de investigación para mejorar o ampliar ciertos aspectos que no han podido ser completamente estudiados.
- **Parte IV Apéndices:** Es una parte adicional donde se encuentran todos los apéndices del trabajo.

Al final de este documento, también se encuentran las referencias a las diferentes citas realizadas en la memoria.

Parte II

Desarrollo del proyecto

Capítulo 1

Descripción del equipo de trabajo

Contenido

| | | |
|------------|--------------------------------------------|-----------|
| 1.1 | Software utilizado | 14 |
| 1.2 | Equipo electrónico | 15 |
| 1.2.1 | Módulo de control | 16 |
| 1.2.2 | Sensor del dedo | 18 |
| 1.3 | Materiales adicionales y conectores | 20 |
| 1.3.1 | Láminas de silicona y sus moldes | 20 |
| 1.3.2 | Lámina piezoresistiva | 21 |
| 1.3.3 | Cables y conectores | 22 |
| 1.3.4 | Objetos a deslizar | 23 |
| 1.4 | Banco de pruebas | 25 |

Sinopsis

En este capítulo se describirán todos los elementos utilizados para llevar a cabo el proyecto. La mayoría de ellos prestados por el departamento de electrónica de la universidad de Málaga y por el equipo de investigación EIS.

Constará de cuatro partes: el software utilizado, el equipo electrónico relativo al sensor táctil, el banco de pruebas y descripción de materiales adicionales que han sido necesarios para los experimentos realizados.

1.1. Software utilizado

Para toda la labor experimental que este proyecto acarrea, ha sido necesario la utilización de diferentes programas comerciales además de aplicaciones desarrolladas, bien por el departamento o bien durante el desarrollo de este trabajo fin de máster.



Figura 1.1: Logos de los diferentes programas utilizados.

Son tres las principales aplicaciones informáticas utilizadas: MATLAB, ISE Design Suite y LabVIEW. Todas ellas permiten la programación de los diferentes elementos de control que serán usados en este trabajo. A continuación se explica la funcionalidad de cada uno de ellos:

- **MATLAB:** Este entorno de trabajo es ampliamente conocido en el ámbito de la ingeniería debido a su versatilidad y potencia. Para el propósito que atañe a este proyecto, se desarrolla una aplicación que analiza los datos tomados por el sensor y con las herramientas integradas en MATLAB se diseña el filtro apropiado que será la base de la detección del deslizamiento. Este será el filtro que es programado posteriormente en la FPGA que controla el sensor.
- **LabVIEW:** Este es otro programa bastante usado en el mundo de la electrónica ya que permite crear interfaces gráficas entre un PC y otros dispositivos electrónicos. En este caso, se ha desarrollado una aplicación por parte del equipo de investigación EIS que es capaz de comunicarse desde un ordenador con la FPGA. De este modo, se pueden leer los datos leídos por el sensor y guardarlos en un archivo.
- **ISE Design Suite:** Por último, este es un software de la empresa Xilinx que permite el diseño, simulación y posterior programación de sus FPGAs, en concreto la familia Spartan que será la usada en este proyecto.

Las aplicaciones desarrolladas serán vistas con más detalle en el capítulo 2, donde se describen los algoritmos y la funcionalidad de cada programa parte a parte.

1.2. Equipo electrónico

Tanto el equipo electrónico de acondicionamiento de la señal, como el cableado del equipo y el ordenador para la visualización de los datos, han sido cedido por el equipo de investigación EIS de un proyecto anterior [11]. Éste está diseñado para dotar de sensores táctiles a una mano robótica tipo Barrett con tres dedos y una palma. Como se puede ver en la imagen 1.2, hay una placa a la que se conectan todas las partes de la mano. Los sensores de los dedos llevan incorporados, en la misma PCB, una FPGA para hacer un preprocesado de la señal mientras que el de la palma se realiza en la placa principal.

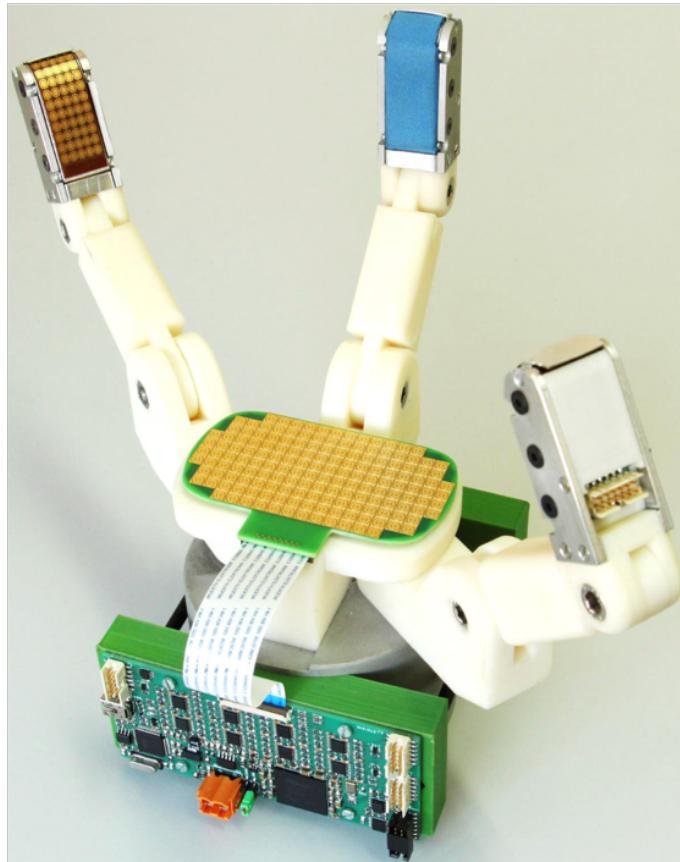


Figura 1.2: Mano de Barrett del trabajo de Óscar Oballe. *Fuente: [11].*

El equipo utilizado para realizar los experimentos para la detección del deslizamiento consta de los sensores de uno de los dedos conectado por un bus tipo SPI a la placa principal, la cual se comunica con el ordenador mediante el estándar USB. A continuación, se procede a la descripción de estas dos unidades.

1.2.1. Módulo de control

El módulo de control se encarga de reunir los datos de cada uno de los sensores instalados en la mano y enviarlos al ordenador mediante el puerto USB 2.0 que lleva incorporado. En la siguiente imagen se pueden ver las diferentes partes de la placa. A continuación, se explican los componentes que serán de utilidad para este proyecto, ya que solo se usará el dedo número dos y esta placa está pensada para tres dedos y la palma. Todos los datos técnicos de este modelo han sido obtenidos de [11].

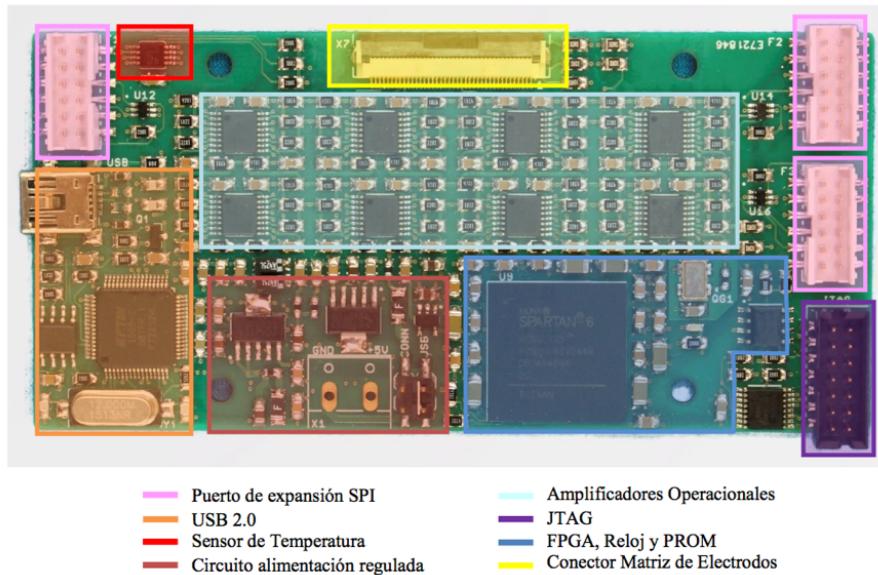


Figura 1.3: Módulo de control del sistema. *Fuente: [11]*.

- **Controlador FPGA:** El modelo incorporado en este módulo es el 6SLX25F-TG256 de la familia Spartan 6 de Xilinx. Esta FPGA realiza la recopilación de los datos de todos los dedos y el control de la palma, así como la comunicación con el ordenador por el puerto USB.
- **Memoria PROM y reloj:** Para el correcto uso de la FPGA es necesario añadir varios circuitos externos como son una memoria adicional y un circuito de reloj. Como la FPGA tiene una memoria limitada de 6,4 Mb para su programación, esta placa lleva incorporada una memoria EEPROM con protocolo serie SPI de 8 Mb de capacidad. Para el circuito de reloj, la placa lleva incorporada un resonador con una frecuencia de trabajo de 50 MHz.
- **Circuito de alimentación:** El voltaje al que trabajan las I/O de la FPGA es de 3,3 V, mientras que el núcleo necesita de 1,2 V. Como la placa es alimentada

por el puerto USB y este proporciona una tensión de 5 V es necesario utilizar un circuito que regule este voltaje.

- **Sensor de temperatura:** La matriz de sensores de presión lleva una fila de resistencias de calibración para estimar el valor real de medición independientemente de la temperatura. Esta estimación es mejor cuanto más parecido sea el coeficiente de temperatura tanto del sensor como de la resistencia de calibración. Aun así, se ha añadido un sensor de temperatura externo en la placa para tener la posibilidad de realizar diferentes estudios sobre la influencia de las variaciones de la temperatura.
- **Puertos de expansión SPI:** Estos conectores son usados para la unión del módulo de control con los tres dedos utilizando el protocolo de comunicaciones SPI. Además, estos puertos llevan también las señales del protocolo JTAG para la configuración/depuración de las FPGAs y para la programación de las memorias EEPROM, una para la alimentación de los sensores de dedo (5 V), una para la salida del circuito de alimentación de los sensores de dedo (3,3 V) y dos para la tensión de tierra (GND). De esta forma se usa un solo bus físico para los dos protocolos. En la imagen 1.4 tomada de [11] se puede ver el esquema del sistema de comunicaciones entre todos los componentes. No se entrará en más profundidad en el sistema de comunicaciones de los dispositivos ya que se escapa de los objetivos del proyecto.

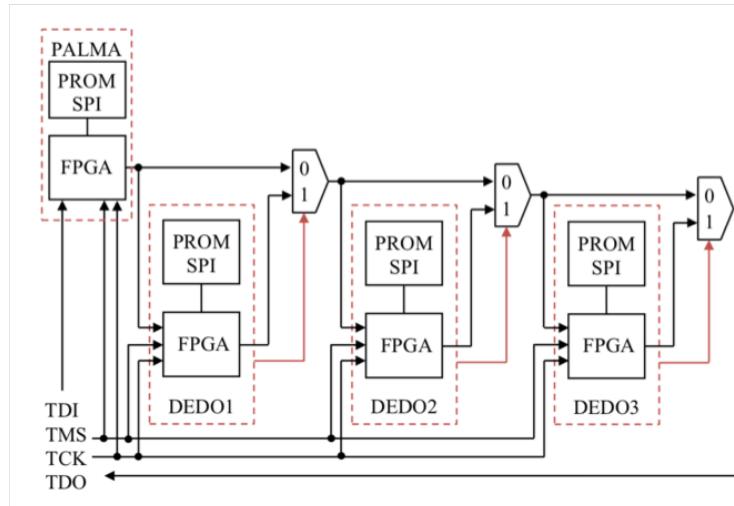


Figura 1.4: Esquema de los puertos de expansión SPI. *Fuente: [11].*

- **Conector JTAG:** Este es el conector que se ha utilizado para programar la FPGA incorporada en el dedo 2 haciendo un puente a la de la presente

placa. Esta segunda FPGA es la única que se modificará en este trabajo y está explicada con más detalle en el siguiente apartado.

- **Módulo de comunicaciones:** En este módulo están incorporado todos los componentes necesarios para implementar el protocolo de comunicaciones de alta velocidad USB 2.0. Su versatilidad y velocidad permiten la transmisión eficaz de la imagen táctil de los sensores.

1.2.2. Sensor del dedo

Este es el elemento que irá acoplado al dedo de la mano de Barrett y por lo tanto, deberá estar fabricado con un material flexible que permita adaptarse a la forma de la falange. Para ello, se ha usado la tecnología “Rigiflex” que sirve para crear PCBs que combinan una parte flexible con la ya conocida parte rígida.

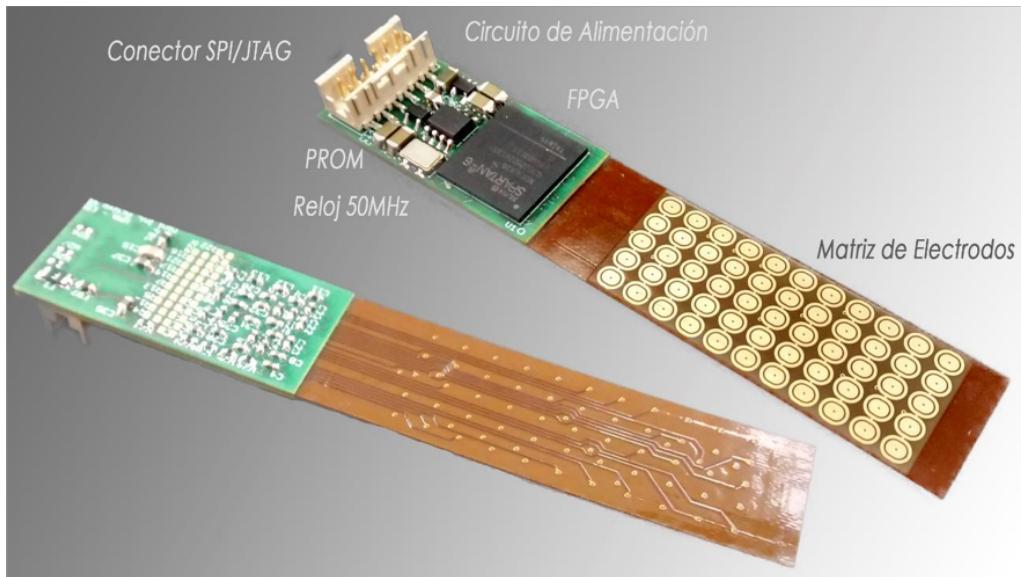


Figura 1.5: PCB del sensor de presión. *Fuente: [11].*

De esta forma, las capas interiores o centrales están hechas de polimida con cobre laminado y las capas externas se componen de fibra de vidrio (FR-4). De las seis capas que posee la PCB del sensor, la matriz de electrodos está impresa sobre las internas (flexibles), mientras que para la electrónica de acondicionamiento se han usado, además, las capas exteriores rígidas.

A continuación, se listan los diferentes elementos que componen el sensor y su acondicionamiento:

- **FPGA:** En esta placa, como en el módulo de control, se usa otra FPGA. La función de esta, es la de gestionar los datos medidos por la matriz de electrodos y la comunicación con la placa central. Además, en este dispositivo se implementa el filtro para la detección del deslizamiento que es la base de este proyecto. Por consiguiente, esta será la FPGA en la que se basa la programación de este trabajo. El modelo usado es 6SLX16CSG225 de Xilinx que tiene un empaquetado más pequeño y se adapta a los requerimientos de tamaño del sensor.
- **PROM:** Como en el caso del módulo de control, la FPGA necesita de una memoria permanente en la que se guarda el fichero que describe el circuito que será creado cada vez que se inicie el dispositivo. En concreto es una EEPROM que se conecta a través de SPI con la FPGA.
- **Circuito de alimentación:** Estos son reguladores de tensión que generan 3,3 V para los pines digitales y 1,2 V para el núcleo de la FPGA. Al igual que en el módulo de control.
- **Circuito de reloj:** De la misma forma que en el módulo de control este circuito posee un cristal de reloj que trabaja a 50 MHz. Se pueden conseguir diferentes frecuencias dividiendo y multiplicando la del reloj principal.
- **Puerto de comunicaciones/programación:** Como se indicó anteriormente, por el mismo conector se envían los pines necesarios para los buses JTAG y SPI, así como para la alimentación y tierra.
- **Matriz de electrodos:** Este es el sensor propiamente dicho. Está compuesto por 55 electrodos dispuestos en 11 columnas y 5 filas. Se leen las 5 filas de forma secuencial y las 11 columnas (de cada una de las filas) en paralelo para hacer la lectura más eficiente.

1.3. Materiales adicionales y conectores

En esta sección se describe el uso de otros materiales necesarios durante el proceso de experimentación y obtención de los datos del proyecto.

1.3.1. Láminas de silicona y sus moldes

Con el objetivo de aumentar las vibraciones mecánicas producidas en el deslizamiento sobre el sensor, se han usado láminas de silicona entre el objeto deslizante y la superficie del transductor. Se han realizado pruebas tanto con esta lámina como sin ella para comprobar los resultados, estos se verán en capítulos posteriores.

Esta silicona fue elegida por el compañero en el departamento Jose Miguel Ríos Pérez, así mismo, él diseñó los moldes necesarios para la creación de las láminas. En concreto, la solución ha sido un aerosol de silicona de la marca *fischer* que se puede ver en la figura 1.6.



(a) Marca y características.



(b) Modo de empleo y aplicaciones

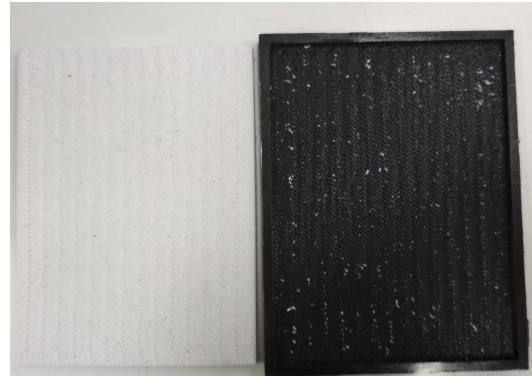
Figura 1.6: Aerosol de silicona usado para los experimentos.

Como es en modo aerosol, es necesario utilizar un molde para crear la lámina final. Según las características, es anti-adherente y fácilmente desmoldeable. Por este

motivo, se ha realizado el molde en la impresora 3D con una superficie rugosa para favorecer las vibraciones.



(a) Creación de las láminas.



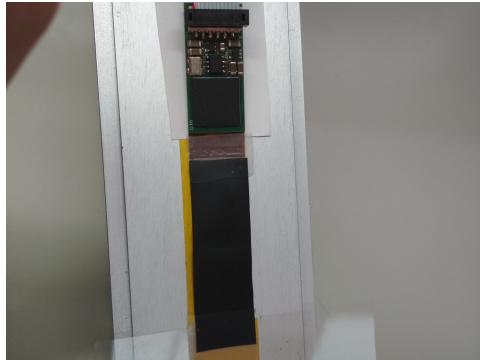
(b) Rugosidad del molde aplicada en la lámina.

Figura 1.7: Proceso de fabricación de los moldes de silicona.

Como se puede apreciar en la imagen 1.7(b), grabado rugoso en el molde es aplicado a la lámina de silicona.

1.3.2. Lámina piezoresistiva

Para el correcto funcionamiento del sensor, tiene que haber una resistencia entre los electrodos que cambie su valor en función de la presión que se ejerza sobre el material. Para este propósito, se ha usado una lámina piezoresistiva isotrópica que se coloca uniformemente sobre la matriz de electrodos.



(a) Lámina sobre el sensor.

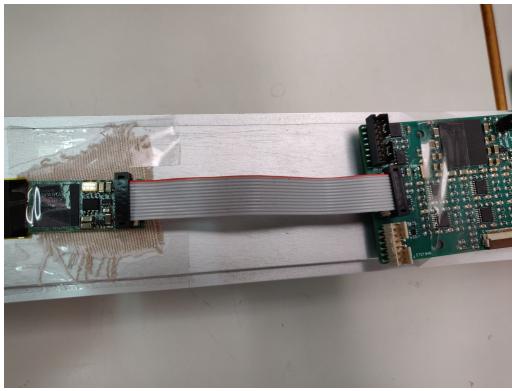


(b) Rollo de material.

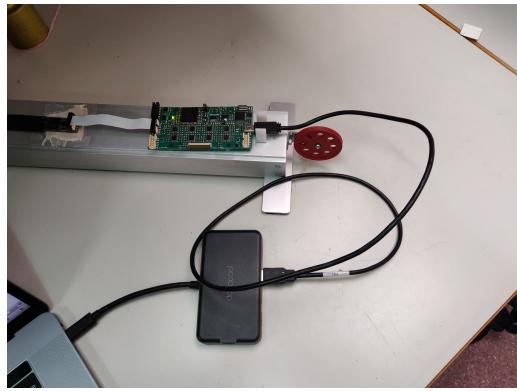
Figura 1.8: Lámina piezoresistiva aplicada sobre el sensor.

1.3.3. Cables y conectores

En la sección anterior de equipo electrónico, se comentó en detalle los puertos y conectores que poseen cada una de las placas. Aquí simplemente se mencionan los usados en este trabajo, ya que no se utiliza el equipo al completo. En primer lugar, para la realización de los experimentos se tienen los conectores que se muestran en la figura 1.9.



(a) Conector al dedo 2.



(b) Conector USB al ordenador.

Figura 1.9: Conectores para la experimentación.

En la figura 1.9a, se muestra el bus que conecta el módulo de control con la placa del sensor del dedo 2. De esta forma, se alimenta el sensor desde la placa principal, se envían los datos medidos por el sensor y se permite la programación de la FPGA que controla la matriz de electrodos gracias al protocolo JTAG.

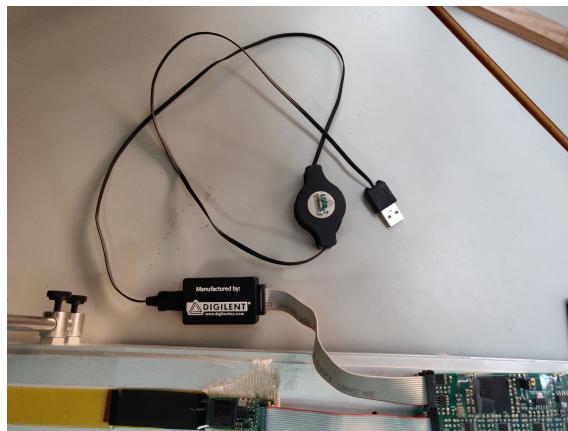


Figura 1.10: Programador convertidor de USB a JTAG.

Por otro lado, en la figura 1.9b se muestra el conector USB que va desde el módulo de control al ordenador. Por este bus serie, el módulo de control envía los datos recogidos por todos los dedos y la palma (en este caso un solo dedo) al ordenador. Además, se da alimentación a todo el sistema.

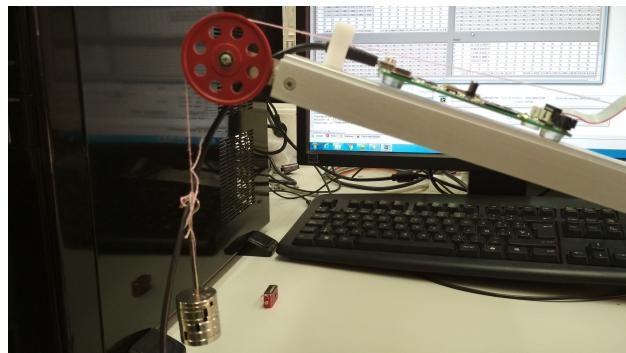
El último conector que se utiliza es el programador, éste se puede ver en la figura 1.10. El cable incorpora un módulo que hace la conversión de USB al protocolo JTAG que es el que se usa para la programación de las FPGAs. Este módulo es del mismo fabricante (Xilinx).

1.3.4. Objetos a deslizar

Para la experimentación realizada en este proyecto, se han usado tres objetos diferentes para deslizar sobre el sensor. El primero de ellos es un carrito y un contrapeso que se pueden ver en la figura 1.11.



(a) Carrito con goma para mayor fricción.



(b) Contrapeso unido al carrito.

Figura 1.11: Carrito y contrapeso unidos por un hilo.

Como se muestra en la figura 1.11a rodeado en rojo, se ha añadido un trozo de goma de borrar para reducir la superficie y aumentar la presión sobre el sensor. Además, la rugosidad de la goma produce más vibraciones mecánicas lo que permite obtener resultados más claros como se verá posteriormente.

El segundo elemento utilizado ha sido una pesa de 1 Kg de metal de forma cilíndrica que se puede ver en la figura 1.12. Esta se ha usado tanto como para experimentos de rodadura (sobre el lado), como para experimentos de deslizamiento (sobre su base).



Figura 1.12: Pesa usada para rodadura.

Por último, también se han realizado experimentos simplemente deslizando un dedo sobre el sensor como se puede apreciar en la figura 1.13.

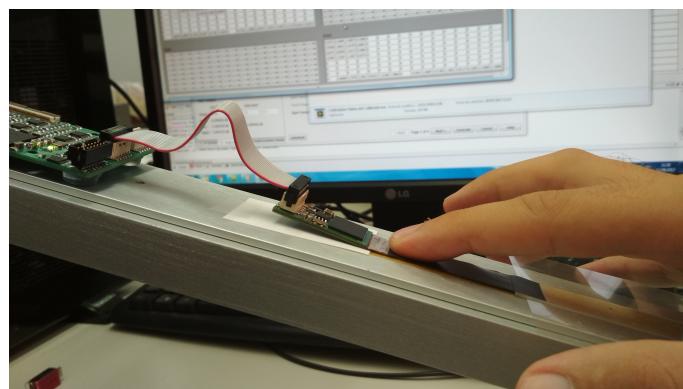


Figura 1.13: Dedo deslizando sobre el sensor.

1.4. Banco de pruebas

Finalmente, se muestra el banco de pruebas al completo, que está compuesto de todos los elementos descritos hasta ahora. Se trata de un plano inclinado en el que se encuentra adherido el sensor y por el que se hacen deslizar o rodar los objetos mencionados anteriormente. Además, se hace uso de un ordenador para obtener los datos enviados por el sensor en tiempo real.



Figura 1.14: Banco de pruebas completo.

En la figura 1.14 se puede ver el banco de pruebas funcionando, usando en este caso el carrito como elemento deslizante.

El software que se ve en el ordenador de la imagen, es el programa desarrollado en LabVIEW que toma y guarda los datos del experimento en un fichero de texto. Todo estos elementos se verán con más detalle en capítulos posteriores.

Capítulo 2

Análisis de los algoritmos utilizados

Contenido

| | |
|------------------------------------------------------|-----------|
| 2.1 Algoritmo en MATLAB | 28 |
| 2.1.1 Análisis de los datos con MATLAB | 29 |
| 2.2 Algoritmo en ISE Design Suite | 30 |
| 2.2.1 Implementación del filtro en la FPGA | 32 |
| 2.2.2 Circuito de umbral de detección | 33 |
| 2.3 Algoritmo en LabVIEW | 36 |
| 2.3.1 Modificaciones al programa | 37 |

Sinopsis

En el presente capítulo se describirán con detalle los algoritmos implementados en cada uno de los programas que se han utilizado en este proyecto y que fueron mencionados en el capítulo anterior.

Se hará más hincapié en la aplicación desarrollada en MATLAB ya que es totalmente genuina de este proyecto y es la que realiza todo el análisis de datos, para obtener el filtro adecuado, que es el objetivo primordial del presente trabajo. El código de este programa se encuentra en los apéndices.

2.1. Algoritmo en MATLAB

La aplicación desarrollada en MATLAB toma los datos que han sido almacenados en un fichero de texto por el programa hecho en LabVIEW. Posteriormente, analiza los datos, los representa y diseña un filtro para detectar el deslizamiento. Una vez obtenidos los parámetros del filtro, estos serán usados para su diseño hardware en la FPGA.

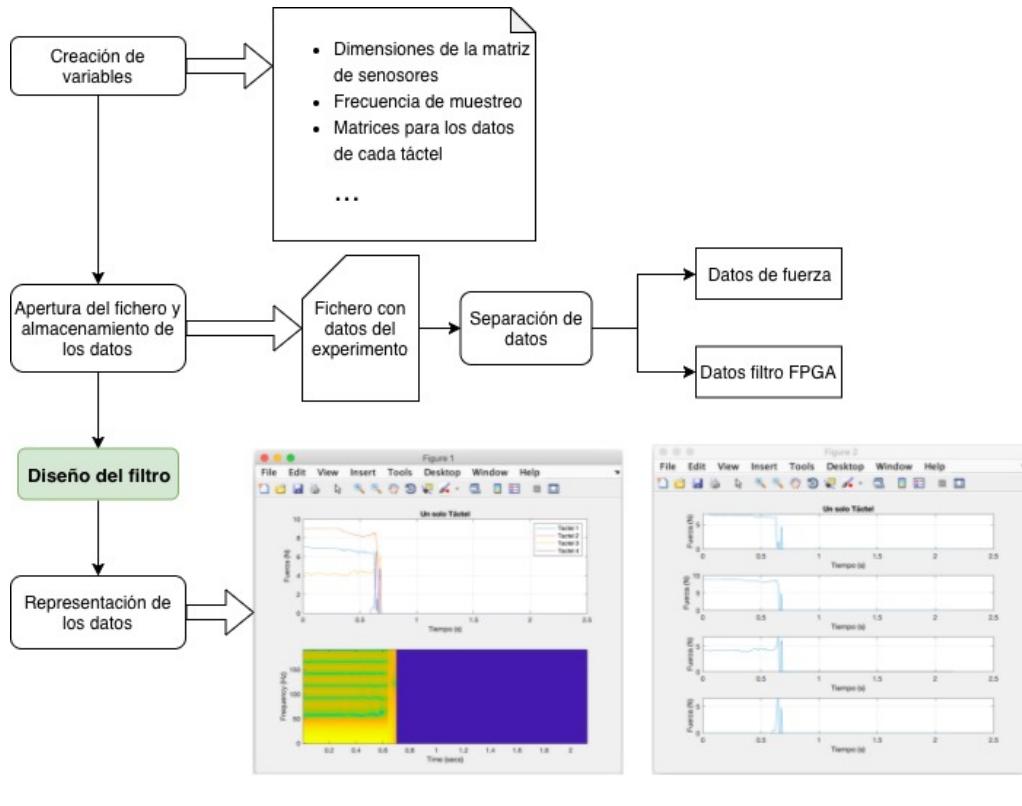


Figura 2.1: Diagrama del algoritmo implementado en MATLAB.

Analizando los datos en las diferentes gráficas, se puede estudiar cómo capta el sensor los diferentes tipos de deslizamiento y si funcionan correctamente los filtros diseñados. Todo esto a través de un proceso de experimentación que se explicará en detalle en el capítulo 4.

El paso más importante de la ejecución de este algoritmo es la de la creación del filtro. Por lo tanto, esta parte se explicará en más profundidad en un capítulo a

parte, en concreto el capítulo 3 en el que también se hará una breve introducción a la teoría de diseño de filtros digitales. En la siguiente sección, se habla de un asistente que ayuda al análisis de las señales de una forma sencilla y que será muy usado en la etapa de experimentación del proyecto.

2.1.1. Análisis de los datos con MATLAB

MATLAB proporciona un asistente para el análisis en frecuencia de señales. Su interfaz principal se puede ver en la figura 2.2. En ella se puede ver la lectura de un táctel mientras se producía el deslizamiento de un dedo sobre el sensor. La lectura de presión sin más, se muestra en la primera gráfica, en la segunda la potencia de la señal para cada frecuencia y en la última el espectrograma de la señal.

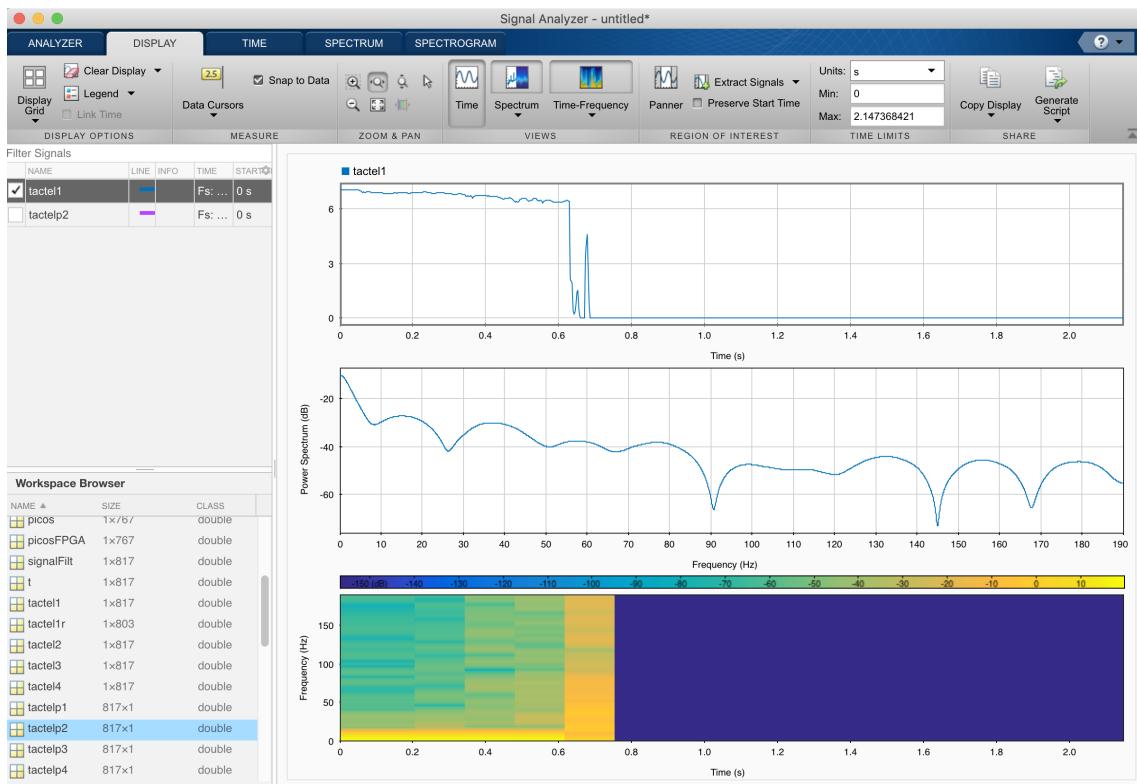


Figura 2.2: Asistente de análisis de señales en MATLAB.

Para introducir las señales en este asistente, primero hay que hacer un script que lea el archivo donde están guardadas, las cargue en el espacio de trabajo de MATLAB y les de un formato adecuado. En esto se entrará en más detalle en el capítulo 4 y un ejemplo de este código se puede ver en el apéndice A.

2.2. Algoritmo en ISE Design Suite

Este algoritmo define la electrónica que será creada en la FPGA de la placa del sensor (matriz de electrodos). Esta se encarga de acondicionar los datos capturados y enviarlos a la placa de control para posteriormente ser recibidos por el ordenador. El programa de partida ha sido diseñado en [11] con anterioridad y solo se han hecho pequeñas modificaciones que se explicaran en este mismo capítulo.

En este apartado se describen los elementos principales del sistema para tener un mapa conceptual del algoritmo creado dentro de este dispositivo.

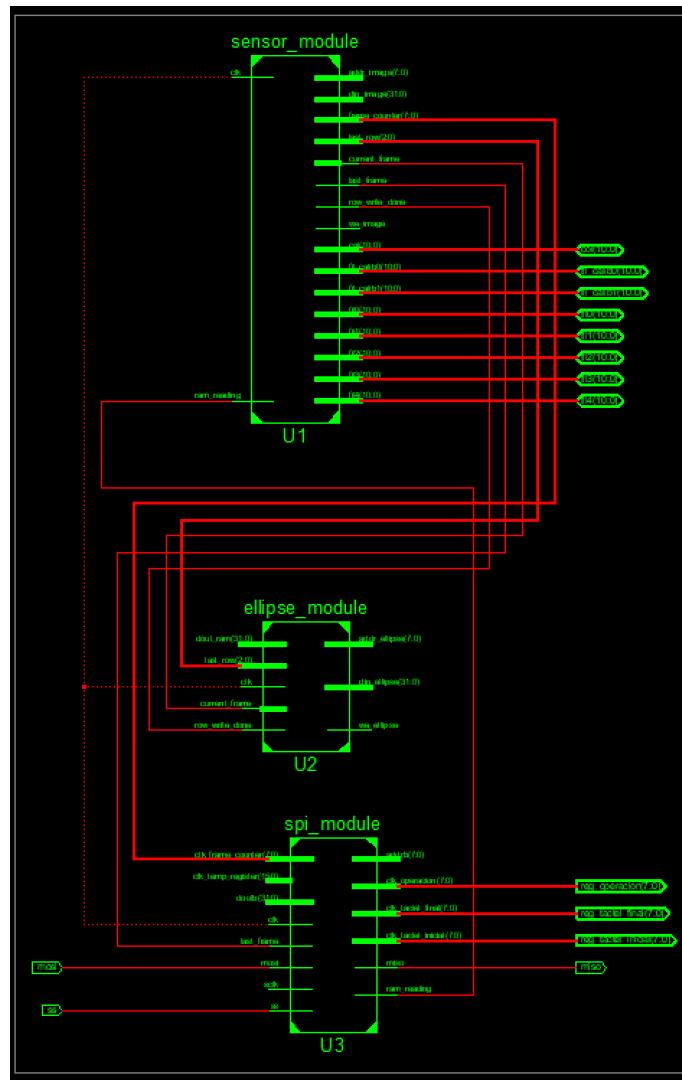
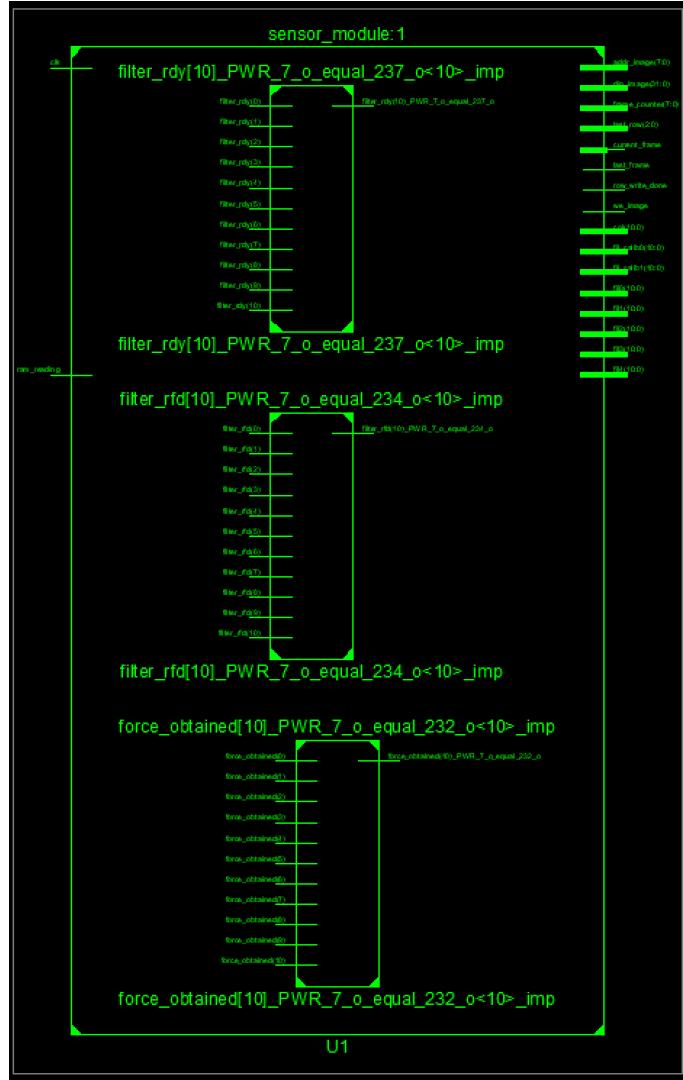


Figura 2.3: Esquemático de más alto nivel.

Como se puede ver en la figura 2.3, el circuito se compone de tres elementos principales. El módulo SPI se encarga de la comunicación con el circuito de control; el módulo de la elipse hace el cálculo de los momentos y la elipse de inercia de la imagen táctil; y el que concierne a este proyecto es el módulo sensor que se comenta a continuación.



Los módulos ya diseñados para el funcionamiento correcto de la FPGA son los siguientes:

- **Módulo de obtención de fuerza:** Se divide en 11 módulos que calculan la fuerza de cada táctel por columna.
- **Módulos de filtros:** Son dos módulos creados por el “Core Generator” del ISE studio. Físicamente también se crean 11 filtros, uno por cada columna de la matriz de electrodos.

2.2.1. Implementación del filtro en la FPGA

El ISE Design Suit incorpora un asistente de creación de bloques de lógica electrónica sin necesidad de describirlos en VHDL o cualquier otro lenguaje de descripción *hardware*. Usan un interfaz amigable para especificar todos los parámetros de estos bloques. En concreto, se ha usado el que se muestra en la figura 2.5 para la compilación de un filtro FIR.

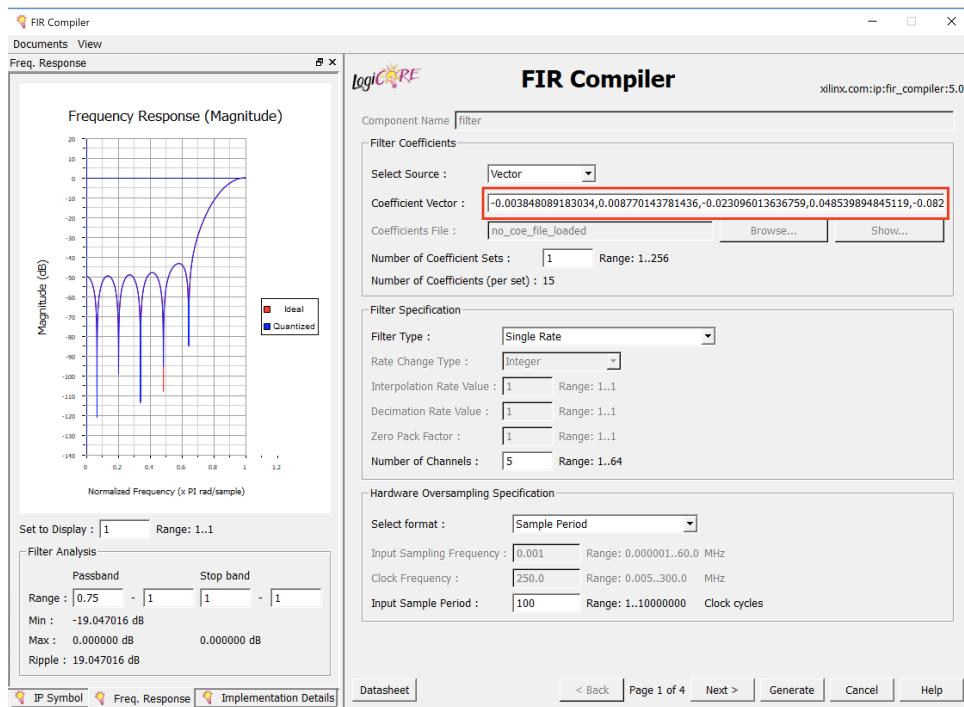


Figura 2.5: Compilador de un bloque de filtro FIR del ISE Design Suite.

Hay 11 de estos módulos ya creados en la lógica de partida de este proyecto. Lo único que se ha hecho es modificarlos para el propósito que se quiere conseguir.

Por lo tanto, solo se introducen los coeficientes del filtro obtenidos del proceso de experimentación en MATLAB en la casilla que está enmarcada en rojo en la imagen.

Una vez que se introducen estos coeficientes, el interfaz en la parte de la izquierda, muestra una pre-visualización de la respuesta en frecuencia del filtro. Además, se hace una comparación entre la respuesta ideal del filtro si se aplican esos coeficientes y la respuesta real una vez implementado en la FPGA.

2.2.2. Circuito de umbral de detección

Para la detección real del deslizamiento sobre la matriz de sensores se ha implementado una lógica que compara la salida del filtro con dos umbrales (uno positivo y uno negativo) por cada uno de los táctiles. A continuación, se explicará de forma esquemática en que consiste este circuito y las partes del código VHDL que lo genera se encuentra comentado y explicado en el apéndice B.

Para no tener que reprogramar la FPGA cada vez que se quiere cambiar el valor de los umbrales, se puede pasar la parte baja de estos valores desde la aplicación de LabVIEW del ordenador que se verá en la próxima sección. De esta forma, en la figura 2.6 se muestra un esquema de la interconexión de los diferentes módulos ya citados.

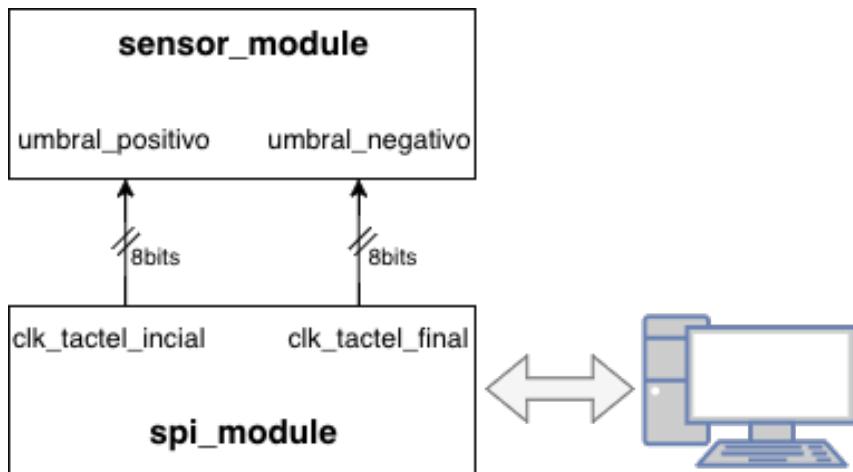


Figura 2.6: Comunicación entre módulos del circuito.

Como se puede ver, el módulo SPI es el que se ocupa de las comunicaciones con el ordenador a través de la FPGA del controlador de la palma. Este módulo usa dos registros de un byte cada uno (*clk_tactel_inicial* y *clk_tactel_final*) en los que guarda dos valores que se pueden mandar desde el ordenador y que en principio no

se usan. En este nuevo circuito se utilizan estos valores para los umbrales. Como se ve en el esquema, una vez que el módulo SPI recibe estos valores desde el ordenador, los envía al módulo sensor donde se usarán como entrada del circuito comparador que se puede ver en la figura 2.7.

Este circuito combinacional se encuentra en el módulo sensor. Como la salida del filtro (*filter_data*) mide 16bits, hay que concatenar la parte alta a los dos umbrales para que tengan la misma longitud cuando se haga la comparación. Como esta comparación se hace en complemento a 2, primero se pone el bit de signo y después se rellena con bits de forma que se obtenga el máximo valor de la parte alta, serían unos para el umbral positivo y ceros para el umbral negativo. Esto se hace porque la salida del filtro normalmente satura o tiene un gran valor cuando se produce el deslizamiento. El ajuste fino se hará con la parte baja del umbral que viene desde la aplicación del ordenador. La comparación de la salida del filtro con los dos umbrales se hacen al mismo tiempo y posteriormente se realiza la operación OR de la salida de los dos bits de los comparadores para generar la señal *slip_detectado_columna*.

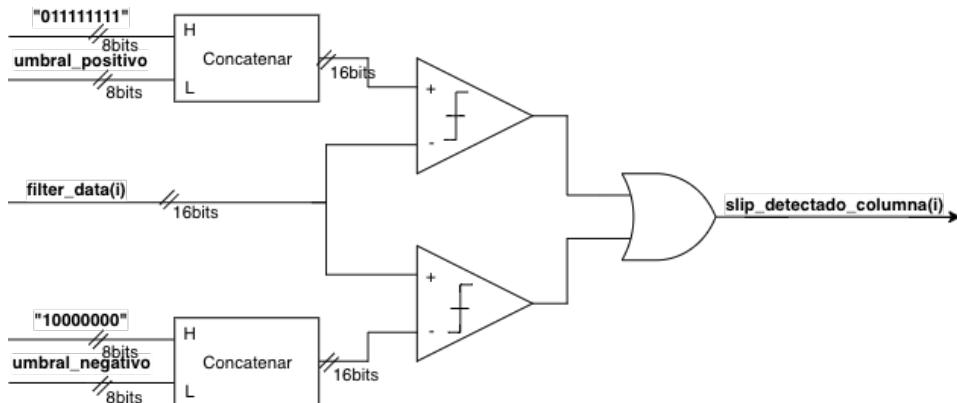


Figura 2.7: Generación de la señal de detección de deslizamiento.

En el módulo sensor está implementada una máquina de estados en la que en el primer estado se lee la fuerza de todas las columnas, en el siguiente se lee la salida del filtro para todas las columnas y en el último se guardan todos estos datos para mandarlos al módulo SPI enviándolos posteriormente al ordenador. Este proceso de tres estados se hace por cada una de las cinco filas en un bucle infinito. Por consiguiente, la comparación se hace por cada columna siendo las señales *filter_data* y *slip_detectado_columna* vectores de 11 valores de longitud (uno por cada columna). Guardando uno valores numéricos y el otro valores booleanos respectivamente.

Una vez obtenido el vector de 11 bits donde cada uno representa si ha habido deslizamiento en cada uno de los tácteles de la presente columna (un uno lógico si ha habido un deslizamiento y un cero si no) es momento para enviarlos al ordenador.

Para ello se ha diseñado la lógica que se muestra en la figura 2.8.

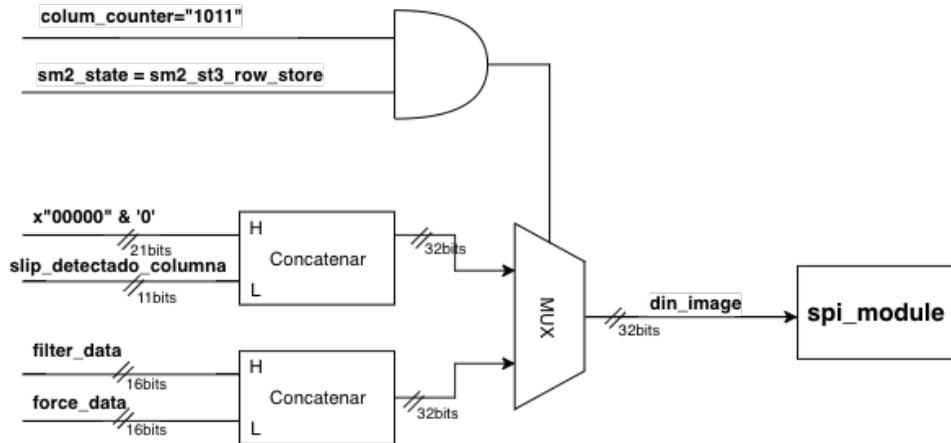


Figura 2.8: Lógica de generación de la imagen que se envía al ordenador.

Esta lógica consiste en un multiplexor que forma la imagen (*din_image*) que será enviada al módulo SPI para posteriormente mandarla al ordenador. Para las 11 primeras columnas (hasta el valor 10 en el contador de columna) se envían los datos de fuerza y filtro, que era lo que el programa original hacía hasta el momento. Sin embargo, se ha modificado el circuito para que se cuente una columna más (contador de columna hasta 11) y en esta se envía el vector que contiene los bits de detección del deslizamiento por cada columna (*slip_detectado_columna*). También es importante mencionar que este envío de la imagen a escribir se hace en el último estado de la máquina de estados como se puede ver en la lógica de selección del multiplexor.

2.3. Algoritmo en LabVIEW

El propósito principal de la aplicación desarrollada en LabVIEW es la de recoger los datos enviados desde la FPGA y almacenarlos en un fichero de texto.

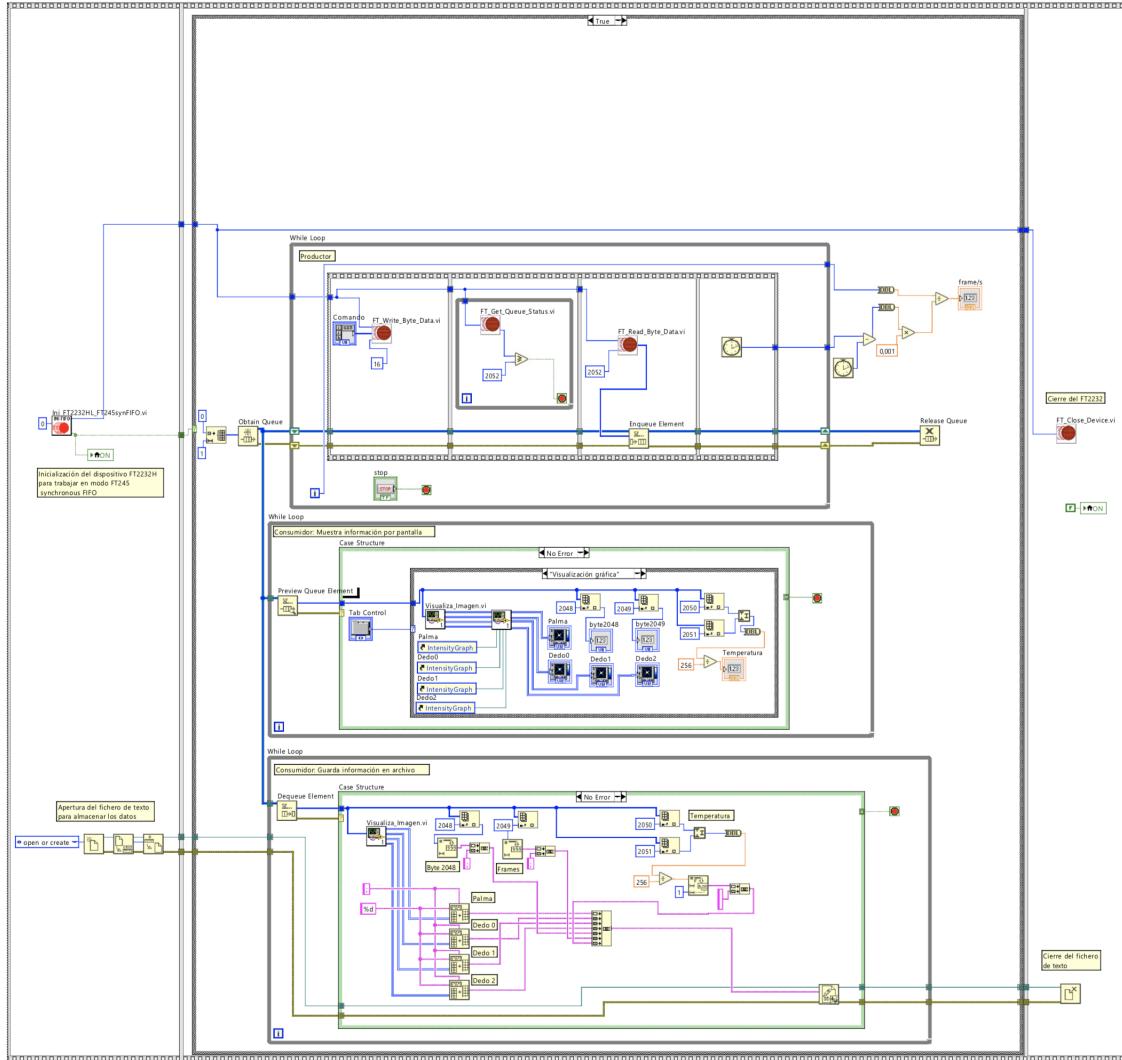


Figura 2.9: Diagrama de bloques de la aplicación en LabVIEW.

Como se puede observar en la figura 2.9, el programa completo está compuesto por tres módulos que realizan tareas diferenciadas. Empezando por el de arriba estos módulos se encargan de: gestión de la comunicación del puerto serie USB, visualización de los datos en una interfaz gráfica y escritura de los datos leídos en un archivo de texto.

Esta aplicación fue diseñada para la mano de Barrett completa, es decir, toma los datos de todos los sensores al mismo tiempo (palma y los tres dedos). Como en este proyecto solo se usa el dedo 2, no se prestará atención a los demás. En la figura 2.10, se puede ver la interfaz gráfica de la aplicación. Esta está compuesta de cuatro cuadros de visualización donde se muestran las matrices de datos enviadas por cada sensor a una frecuencia de 380 muestras por segundo.

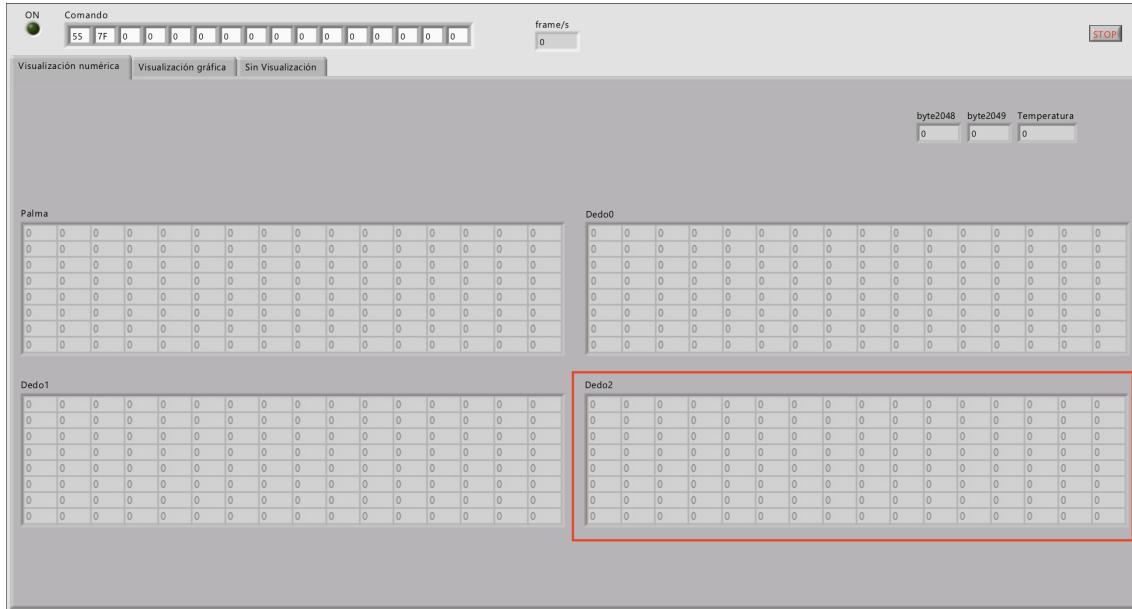


Figura 2.10: Panel gráfico original de la interfaz en LabVIEW.

Solamente es de interés el cuadro de visualización enmarcado en rojo en la figura, que es el correspondiente al dedo 2. Ya que como se dijo anteriormente, éste es el único sensor conectado al módulo de control.

2.3.1. Modificaciones al programa

Debido a que este programa se hizo originalmente para medir la fuerza de todos los sensores de la mano, se han realizado algunas modificaciones para la adaptación a este proyecto y para la comunicación con las modificaciones realizadas en la nueva lógica de la FPGA.

Básicamente, se ha modificado el bucle en el que se muestran los datos visualmente en la aplicación. Se han eliminado los indicadores de la palma, el dedo 0 y el dedo 1 dejando simplemente el indicador del dedo 2 que es el que se ha usado en el presente trabajo. Además se ha añadido una matriz de indicadores LED, uno por

cada táctel que se encienden cada vez que alguno de ellos detecta un deslizamiento. Estos indicadores representan la variable *slip_detectado_columna* enviada desde la FPGA del sensor y cuya generación se explicó en la sección anterior. Después de estas modificaciones la interfaz del programa quedaría como se muestra en la figura 2.11.

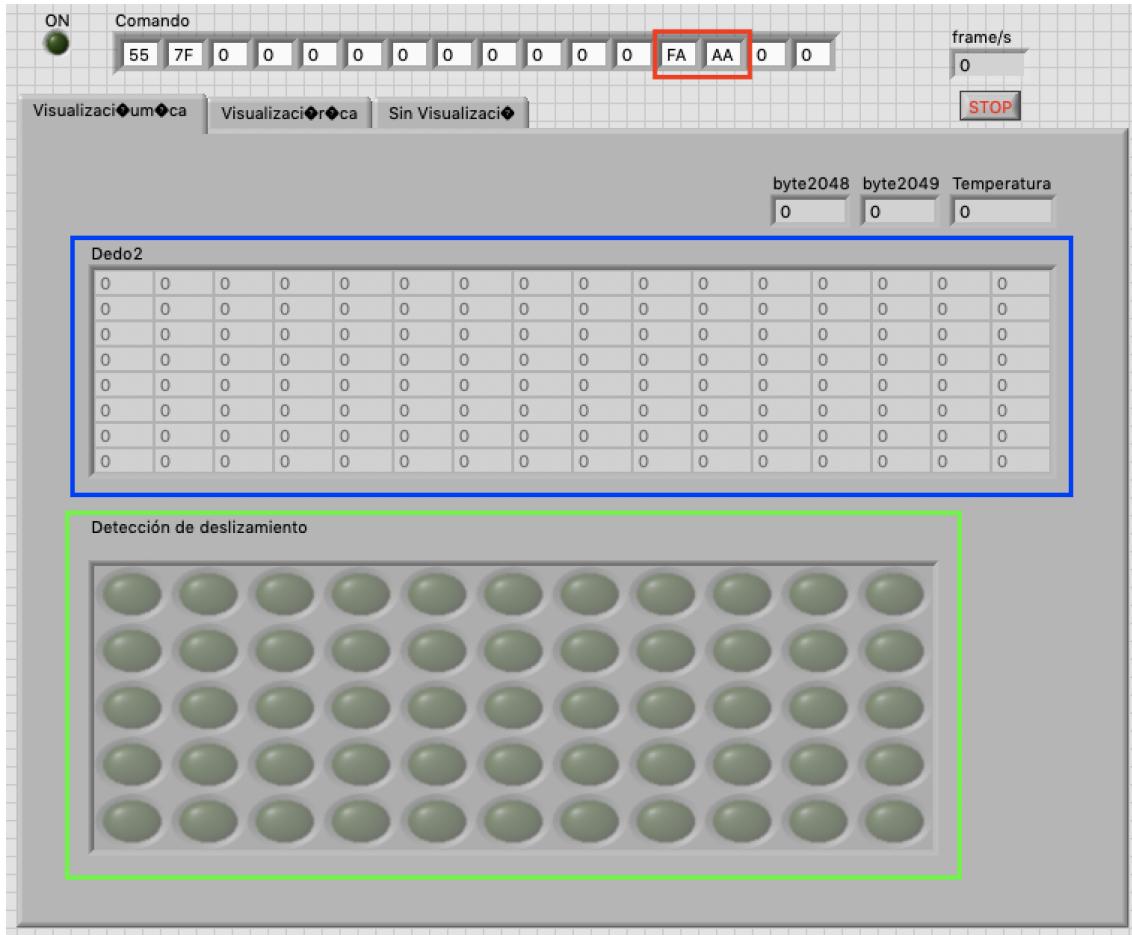


Figura 2.11: Interfaz de LabVIEW modificada.

Como se puede apreciar, se mantiene el indicador de la fuerza medida por cada uno de los táctiles del sensor del dedo 2 (recuadro azul en la imagen) pero se han eliminado los indicadores de los demás sensores. En el recuadro verde se ve la nueva modificación, consistente en una matriz de LEDs, uno por cada táctel del sensor de presión. Por último, en el recuadro rojo, se ven dos campos para la introducción de texto en los que se puede especificar la parte baja del umbral positivo (campo de la izquierda) y del umbral negativo (campo de la derecha) que serán recibidos por el módulo SPI de la FPGA. Hay que tener en cuenta que estos valores serán

interpretados en complemento a dos y que la parte alta es la que da el valor máximo posible ya que viene fijada en el código VHDL (ver apéndice B).

En la figura 2.12 se pueden ver las modificaciones realizadas en el diagrama de bloques. Como se ha mencionado, solo se ha modificado el bucle de la representación de los datos.

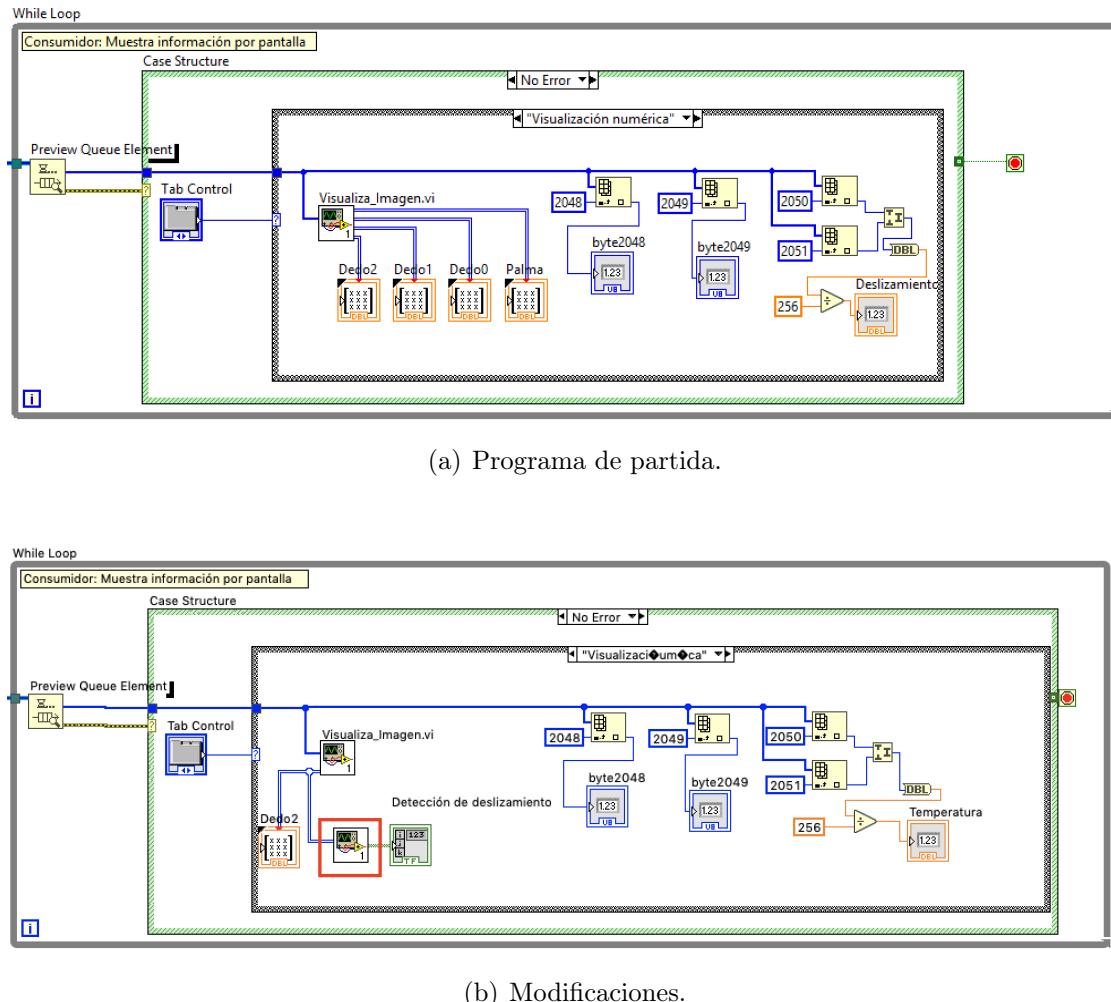


Figura 2.12: Diagrama de bloques original y modificaciones.

Simplemente se han eliminado los indicadores sobrantes y se ha creado y añadido un nuevo bloque para la representación de la matriz de LEDs que se encuentra recuadrado en rojo en la figura 2.12b. Por otro lado, en la figura 2.13, se muestra el interior de este nuevo bloque en el que simplemente se extraen los datos de la columna adicional que se envió desde la nueva modificación de la lógica de la FPGA. En estos datos se encuentran los bits que representan la detección del deslizamiento

por cada táctel.

Si se analizan los bloques de este nuevo módulo, se ve como se seleccionan los bytes que se quieren usar de la matriz de datos del dedo 2, se transforman a valores booleanos y se representan en la matriz de LEDs.

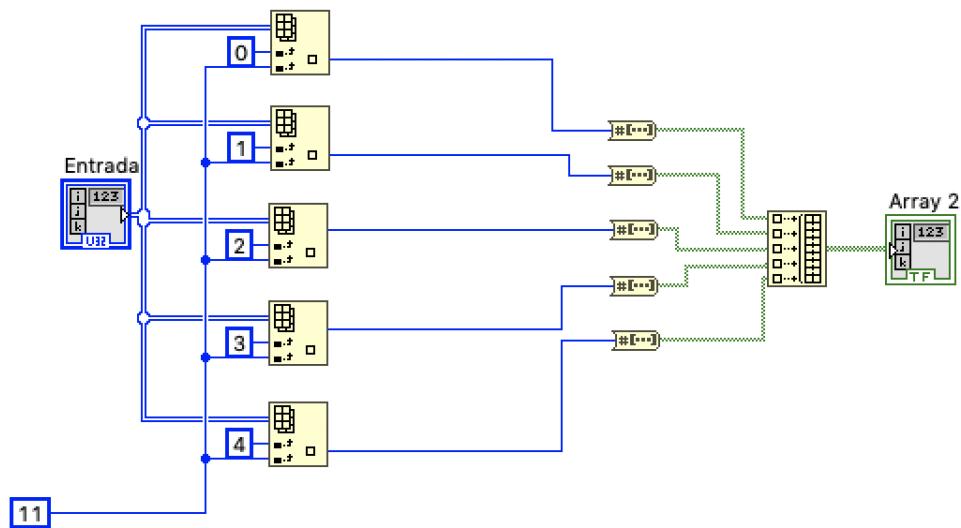


Figura 2.13: Nuevo módulo creado para la representación de la matriz de LEDs.

Capítulo 3

Teoría de filtros digitales

Contenido

| | |
|-------------------------------------------------------------------------------|-----------|
| 3.1 Filtros digitales | 42 |
| 3.1.1 Rango de frecuencias y teorema de muestreo de Nyquist-Shannon | 42 |
| 3.1.2 Clasificación en función de la frecuencia | 44 |
| 3.1.3 Filtros FIR e IIR | 47 |
| 3.2 Diseño e implementación digital de un filtro FIR | 50 |
| 3.2.1 Definición y funcionamiento | 50 |
| 3.2.2 Obtención de los coeficientes | 51 |
| 3.2.3 Métodos de diseño | 55 |
| 3.3 Asistente de diseño de filtros en MATLAB | 60 |
| 3.3.1 Método de diseño de filtros FIR | 61 |
| 3.3.2 Resultado final, coeficientes | 62 |

Sinopsis

Como se ha mencionado en capítulos anteriores, el estudio de los filtros y su diseño es tan complejo que requiere un capítulo aparte. Aquí se hará una breve introducción a la teoría de filtros digitales y al asistente de diseño que incorpora MATLAB.

Así, este capítulo sirve como una introducción a los conocimientos previos que se usarán para la creación del filtro en el capítulo 4.

3.1. Filtros digitales

Se estudiaron los filtros digitales ya que son los que se pueden implementar en un sistema digital como es la FPGA. Además se discutirá sobre los filtros FIR e IIR para discernir la mejor opción y el problema que supone el teorema de muestreo de Nyquist-Shannon en este proyecto.

3.1.1. Rango de frecuencias y teorema de muestreo de Nyquist-Shannon

En primer lugar, es importante comentar y acotar el rango de frecuencias a las que se ha trabajado en este proyecto. Para ello, se empezará hablando de la frecuencia a las que se producen las micro-vibraciones en las etapas tempranas del deslizamiento, fundamental para este trabajo.

Según el artículo del grupo de investigación EIS [9], estas micro-vibraciones se producen entre los 20 Hz y los 220 Hz. Esto ha sido comprobado usando varios experimentos y un osciloscopio como se explica en el texto. Además, diversos estudios desde el punto de vista biológico, como [2] y [13], corroboran estos datos.

Tabla 3.1: Tabla de tres mecanorreceptores que intervienen en el sentido del tacto.

| Mechanoreceptor | Detection | Adaptation rate | Specialization | Location | Encapsulation | Responsiveness to continuous deformation (property related to encapsulation) |
|-----------------------|--------------------------------------------------------------------------------------------------------------|-----------------|----------------|------------------|---------------|---------------------------------------------------------------------------------|
| Pacinian corpuscles | - rapid vibratory pressure and touch (max sensitivity at about 250 Hz) | rapid | yes | deep skin | yes | no |
| Meissner's corpuscles | - light touch - changes in texture - relatively slow vibrations (up to 50 Hz) | rapid | yes | superficial skin | yes | no |
| Merkel's discs | - touch - pressure - changes in texture. Respond from steady state to low frequencies (up to 15 Hz) | slow | yes | superficial skin | no | yes |

En la tabla 3.1 obtenida de [13], se listan tres de los mecanoreceptores que intervienen en el sentido del tacto en los seres humanos. Estos son los corpúsculos de Pacinian, los corpúsculos de Meissner y los discos de Merkel. En la segunda columna de la tabla se muestran las frecuencias máximas a las que son sensibles cada uno de los receptores. Si se presta atención, el rango de frecuencias es desde 15 Hz hasta 250 Hz, muy parecido al medido por los experimentos realizados en [9].

Ya se ha definido el rango de frecuencia en el que se produce el deslizamiento y que es que se quiere aislar con el filtrado. Por otro lado, el muestreo de la señal leída por el sensor se realiza a 380 Hz por el hardware implementado en la FPGA. Esta frecuencia de muestreo es invariable en este proyecto ya que depende de las características físicas del sistema. En este momento, es imprescindible introducir el teorema de muestreo de Nyquist-Shannon. Este teorema, como su propio nombre indica, trata sobre la etapa de muestreo en el proceso de conversión analógica-digital. En concreto, el teorema establece:

“Si una función $x(t)$ no contiene componentes en frecuencias mayores de B Hercios, esta queda completamente definida por sus puntos discretos ordenados ordenados espaciados por $1/(2B)$ segundos”

En otras palabras, demuestra que la reconstrucción exacta de una señal periódica continua en banda base a partir de sus muestras, es matemáticamente posible si la señal está limitada en banda y la tasa de muestreo es superior al doble de su ancho de banda. Dado que el proceso de conversión de una señal analógica a digital no es perfecto (presenta diferentes errores en sus etapas) siempre se recomienda que la tasa de muestreo sea algo superior al doble de la frecuencia más alta de la señal.

Dicho todo lo anterior, sabiendo que la tasa de muestreo es 380, toda componente con frecuencia mayor de $380/2 = 190\text{Hz}$ no podrá ser reconstruida adecuadamente o presentará problemas de distorsión o solapamiento (aliasing). Esta frecuencia está en medio del rango en el que se producen las vibraciones previas al deslizamiento. Por consiguiente, algunas componentes se podrán recuperar perfectamente y otras no. En capítulos posteriores se estudia y se comprueba si esta velocidad de toma de muestras es suficiente para desarrollar un algoritmo que detecte el deslizamiento eficazmente.

El efecto de solapamiento o aliasing se puede explicar fácilmente con un ejemplo obtenido de [21]. Supongamos que una señal telefónica, que no debe tener frecuencias mayores de 4 KHz, no es filtrada adecuadamente y deja pasar componentes en frecuencia mayores que la mitad de la frecuencia de muestreo. Considerando una señal no deseada de 6 KHz con una tasa de muestreo de 8 KHz, la señal de salida tendrá un espectro en frecuencia que incluye $(8 + 6 = 14\text{KHz})$ y $(8 - 6 = 2\text{KHz})$.

El problema ocurre con la señal de 2 KHz ya que está en medio del ancho de banda donde se encuentra la información útil.

Este problema aplicado al presente proyecto, supondría que las frecuencias mayores a la mitad de la tasa de muestreo se superpondrían en la banda de frecuencias donde se producen las microvibraciones. Sin embargo, mientras este solapamiento no elimine la señal no habría ningún problema, de hecho, si se suma a la señal original provocaría un efecto favorable para su detección.

3.1.2. Clasificación en función de la frecuencia

Como se mencionó en la introducción, cuando se produce deslizamiento sobre el sensor táctil, se originan unas señales de altas frecuencias. Por lo tanto, el objetivo es diseñar un filtro que aísle solo esta señal.

Atendiendo a las frecuencias que el filtro deja pasar, existen diferentes tipos de filtros y la clasificación básica que se puede ver en la figura 3.1 es la siguiente: filtro paso-bajo, filtro paso-alto, filtro paso-banda y filtro rechazo banda. Se ha dicho clasificación básica ya que existen otros tipos como el filtro peine que deja pasar solo ciertas frecuencias, por ejemplo.

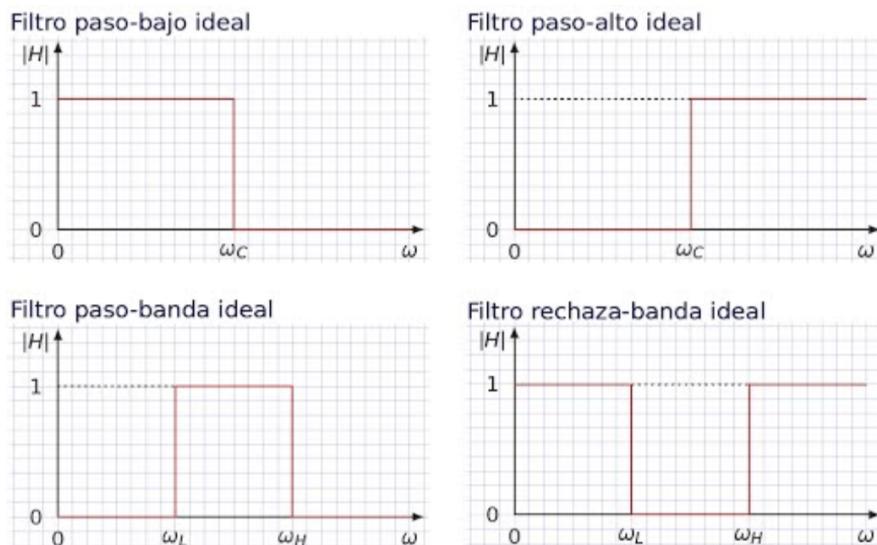


Figura 3.1: Representación ideal de los diferentes tipos de filtro básicos.¹

¹Fuente: <https://sites.google.com/a/goumh.umh.es/circuitos-electronicos-analogicos/transparencias/tema-2>

Como se ve en la imagen, en el eje horizontal está representada la frecuencia y en el eje vertical la ganancia. Esta es la representación ideal ya que la atenuación es completa en la banda de rechazo, es decir, a partir de la frecuencia de corte (w_c) en el caso del filtro paso-baja por ejemplo.

En el caso real hay una banda de transición y la atenuación en la banda de rechazo no es completa. Esto se puede ver en la figura 3.2. En este caso se representa un filtro paso de baja que será representativo para los demás tipos.

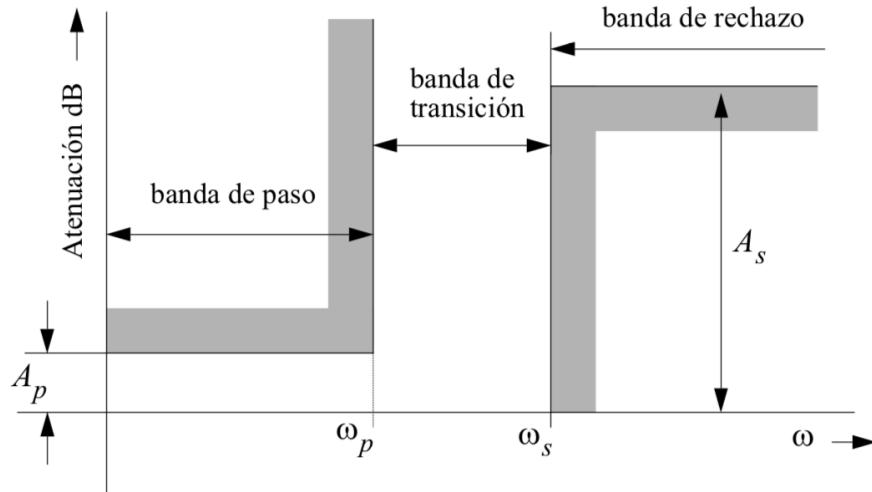


Figura 3.2: Diagrama Atenuación-Frecuencia de un filtro paso de baja. Fuente [5]

En este caso, en el eje vertical se representa la atenuación, en dB, en lugar de la ganancia (una es la opuesta de la otra). Además, aparecen tres bandas que se explicarán a continuación:

- **Banda de paso:** En esta banda la señal tiene que tener una atenuación máxima A_p . En el caso ideal la ganancia de esta banda es la unidad, es decir 0 dB. En el caso de un filtro paso-bajo, esta banda va desde DC hasta la frecuencia de corte w_p .
- **Banda de transición:** Esta banda comprende un estado de cambio entre las banda de paso y la de rechazo. En el caso ideal, esta transición se produce inmediatamente a una frecuencia determinada. Sin embargo, en el caso real, para un filtro LP (Low Pass) comprende el rango de frecuencias de w_p hasta w_s denominada frecuencia límite de la banda de rechazo.
- **Banda de rechazo:** Esta es la banda de frecuencias donde se atenúa la señal. En el caso de un filtro LP las frecuencias por encima de w_s deben tener al menos

A_s dB de atenuación. La ganancia en esta banda se suele establecer a menos de 3 dB normalmente para considerar que el filtro funciona correctamente. Este valor es la mitad de la potencia de la señal original pasada a decibelios.

Todos estos parámetros que se muestran en la imagen y que se han explicado anteriormente definen completamente un filtro paso de baja. El valor de todos ellos dependerán, principalmente, del método que se use para diseñarlo así como del orden de éste.

Se ha puesto como ejemplo el filtro LP para explicar las variables que definen un filtro, sin embargo, estas son aplicables también a un filtro paso de alta ya que tiene la misma morfología, simplemente las bandas de rechazo y de paso están intercambiadas. Como en este proyecto lo que interesa es eliminar las bajas frecuencias y maximizar las frecuencias donde se produce el deslizamiento (altas frecuencias), lo que a priori puede funcionar mejor es un filtro HP (High Pass). Por consiguiente, éste es el tipo de filtro que se ha usado como se verá posteriormente, no obstante, también se han probado otras combinaciones como el uso de un filtro LP siguiendo el esquema de la figura 3.3:

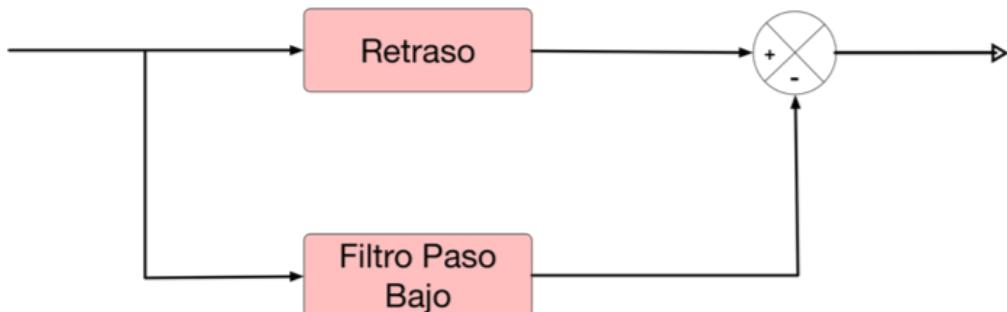


Figura 3.3: Diagrama que muestra el uso de un filtro paso de bajo para eliminar la componente DC.

Como se puede observar, debido a que el filtro supone un retraso de muestras igual a su orden, a la señal se le aplica el mismo retraso y el filtro LP al mismo tiempo y posteriormente se restan. Con esto se consigue el mismo efecto que el de un filtro HP ya que se le resta la componente continua a la señal original obteniendo solo las altas frecuencias. El objetivo final de esta técnica es la de intentar eliminar completamente la componente DC (direct current) de la señal final. El resultado se discutirá en capítulos posteriores.

3.1.3. Filtros FIR e IIR

Otra clasificación importante a la hora de hablar de filtros es la diferencia entre FIR (Finite Impulse Response) e IIR (Infinite Impulse Response). Como su propio nombre indica, el filtro FIR tiene una respuesta finita al estímulo impulso, es decir, pasado un cierto tiempo la salida del filtro es nula. Por otro lado, los filtros IIR debido a que tienen algún grado de realimentación, su respuesta no llega a ser cero en ningún momento.

En la figura 3.4 se ven los diagramas de bloques para la implementación digital de los dos tipos. La señal de entrada se escribe como $x(n)$ mientras que la de salida como $y(n)$. Como se puede observar, los filtros IIR tienen un bucle de realimentación cogiendo la señal de salida y restándola o sumándola a la entrada después de multiplicarla por los coeficientes de realimentación.

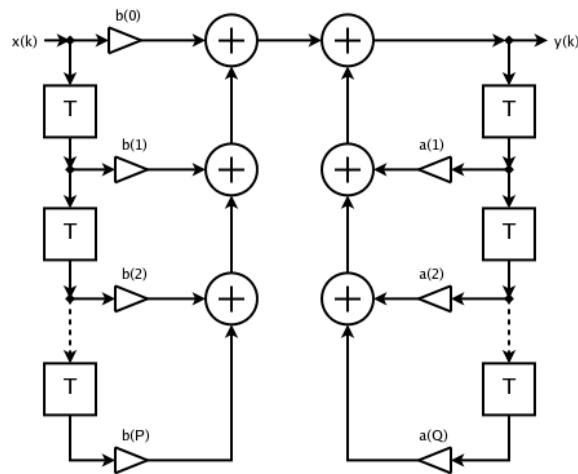
A continuación, se muestra una tabla con las características comparadas de cada uno de estos dos tipos de filtros:

Tabla 3.2: Comparativa entre filtros FIR e IIR.

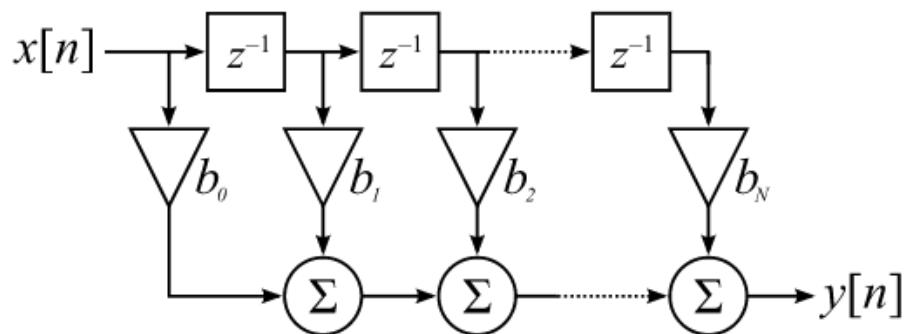
| Filtros FIR | Filtros IIR |
|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| No usa realimentación. Esto significa que si existe un error este no se reducirá automáticamente por el sistema, sin embargo, reduce la complejidad de la implementación del filtro. | Debido a la realimentación, necesitan menos recursos <i>hardware</i> para su ejecución digital. Sin embargo, su diseño y creación son más complejos y laboriosos. |
| Son estables de forma inherente ya que la salida es la suma de un número finito de entradas multiplicadas por coeficientes que son números reales. | Pueden ser inestables, por lo que necesitan un análisis de estabilidad en su fase de diseño, requiriendo un esfuerzo mayor. |
| Se pueden diseñar fácilmente para que tengan una fase lineal con simplemente hacer que los coeficientes del filtro sean simétricos. | No tienen fase lineal, lo que puede ser inapropiado para cierto tipo de aplicaciones. |

A parte de lo mencionado en la tabla 3.2, hay que decir que los filtros FIR solo se pueden implementar de forma digital mientras que los IIR pueden hacerlo tanto analógica como digitalmente.

Por otro lado, como se explicó en el capítulo 2, el entorno de programación de la FPGA solo permite crear módulos de filtros FIR. La cantidad de recursos



(a) Diagrama de bloques filtro IIR. Fuente [19].



(b) Diagrama de bloques filtro FIR. Fuente [18].

Figura 3.4: Esquemas de implementación de filtro FIR e IIR

necesario para este tipo de filtros no es una preocupación ya que la FPGA dispone de suficientes elementos hardware. De hecho, como se vio anteriormente, se crean 11 filtros en paralelo, uno por cada columna de la matriz de tácteles y aún así no se consume una gran parte de sus recursos totales.

Por lo mencionado anteriormente, solamente se realizarán diseños de filtros tipo FIR que además posee ciertas ventajas respecto a los tipo IIR como se mencionó en la tabla 3.2. En particular, interesa que la respuesta sea finita ya que cuando termina el deslizamiento, es deseable que la salida del filtro sea nula. De esta forma, no se generaría una falsa detección del deslizamiento. En la siguiente sección se explica el proceso de diseño de un filtro FIR y como está implementado en un procesador digital.

3.2. Diseño e implementación digital de un filtro FIR

El filtrado digital es un concepto totalmente diferente al filtrado analógico. Los filtros digitales procesan las señales en el dominio del tiempo al contrario que los filtros analógicos que lo hacen en el dominio de la frecuencia. Por lo tanto, después de hacer el estudio en frecuencia para ver las características que debe tener el filtro, habrá que convertir la respuesta en su equivalente en el dominio del tiempo. La forma de la señal en el tiempo no es obvia, por lo que en un principio puede resultar difícil de entender.

En esta sección, se verá este cambio del dominio de la frecuencia al dominio del tiempo, para así poder obtener los coeficientes del filtro y como se implementa este en un entorno digital. Además, se verán las técnicas de diseño de filtros FIR más comúnmente usadas.

3.2.1. Definición y funcionamiento

En primer lugar, ¿cómo es físicamente un filtro FIR? Simplemente es la suma de un número de entradas multiplicadas por diferentes coeficientes. Esto queda expresado en la ecuación 3.1 que se muestra a continuación:

$$y[n] = \sum_{k=0}^M b_k x[n - k] \quad (3.1)$$

En esta expresión, $y[n]$ y $x[n]$ representan la salida y las entradas respectivamente. M es el orden del filtro y b_k son los diferentes coeficientes que multiplican a cada una de las entradas. Esto mismo, expresado con un diagrama de bloques se puede ver en la figura 3.5. Donde los bloques Z representan retrasos, los triángulos multiplicaciones y los círculos sumas.

Como se observa en la imagen, es una lógica bastante sencilla de implementar. Se necesitan registros para los retardos, multiplicadores y sumadores. Tantos como orden tenga el filtro, ya que dependiendo del orden se usarán más o menos muestras retardadas de la entrada.

En este punto, es importante destacar que los retardos pueden ser un problema para esta aplicación en concreto. Se desea que la detección de los momentos previos del deslizamiento sea prácticamente instantánea para que el supuesto robot pueda reaccionar a tiempo. Por consiguiente, hay una limitación en el orden del filtro a

usar. En la siguiente subsección se analiza entre otras cosas como afecta el orden del filtro a su respuesta en frecuencia.

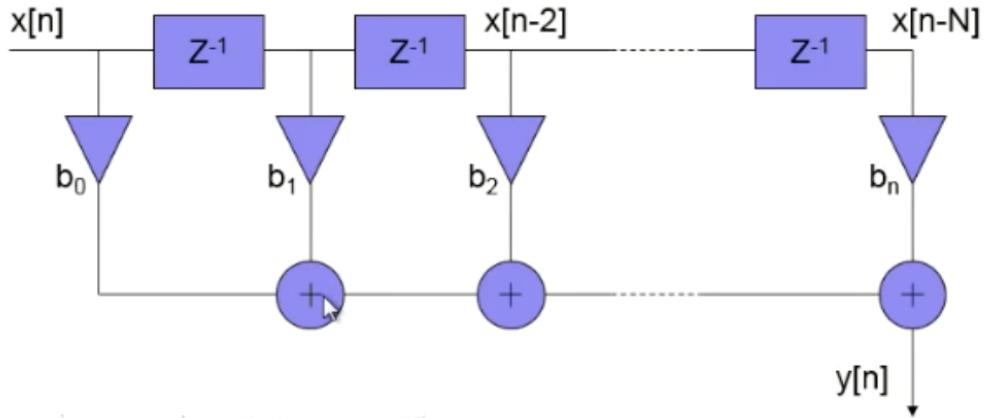


Figura 3.5: Diagrama de bloques de la implementación de un filtro FIR.

La flexibilidad que ofrece la FPGA permite la creación de varios de estos filtros en paralelo sin consumir demasiados recursos como se vio en el capítulo anterior.

3.2.2. Obtención de los coeficientes

Una vez que se sabe como funcionan este tipo de filtros, ya solo queda saber como se obtienen los coeficientes. Para ello, se analiza el filtro paso de baja, ya que es el más sencillo, y posteriormente se verá como se hace con los diferentes tipos de filtros.

Si tenemos el filtro paso de baja ideal y normalizado (“Brick Wall” en inglés), teniendo una ganancia unitaria en el paso de banda, una frecuencia de corte abrupta en 1 rad/s y una ganancia nula en la banda de rechazo, entonces la respuesta al impulso en el dominio del tiempo será la función cardinal del seno o $\text{sinc}(t)$. Esta conversión se hace usando la transformada inversa de Fourier² en la que no se entrará en detalle. La función en el tiempo tiene la siguiente expresión:

$$\text{sinc}(x) = \frac{\sin(x)}{x} \quad (3.2)$$

Tanto la función en frecuencia del filtro paso de baja ideal como su respuesta

²Más información en el siguiente enlace: https://es.wikipedia.org/wiki/Transformada_de_Fourier

al impulso unitario (o función delta de Dirac³) en el tiempo se pueden ver en la figura 3.6. Como se puede observar ambas funciones, idealmente se extienden hacia el infinito en ambas direcciones.

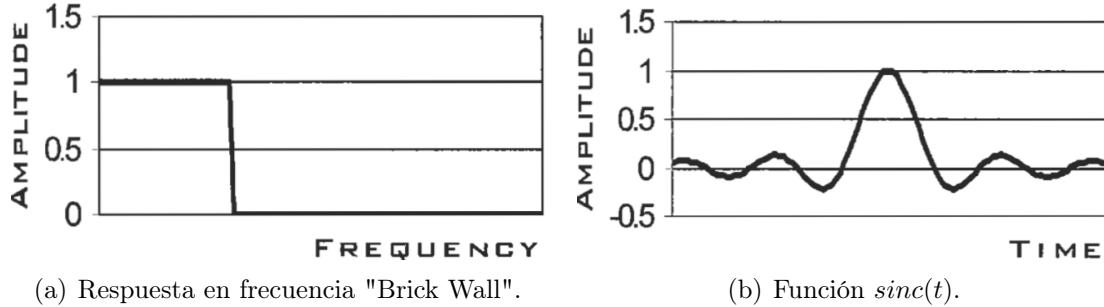


Figura 3.6: Respuesta en frecuencia (a) y en el tiempo (b) de un filtro LP ideal y normalizado. Fuente [21].

Dicho todo esto, multiplicando una señal digitalizada por la función $\text{sinc}(t)$ se obtiene un efecto de filtrado. En esto es lo que se basa básicamente el filtro FIR. Lo que se ha explicado hasta ahora es el caso ideal y normalizado en un entorno continuo, sin embargo, lo que interesa es el caso real en tiempo discreto ya que el filtro será usado en un dispositivo físico digital.

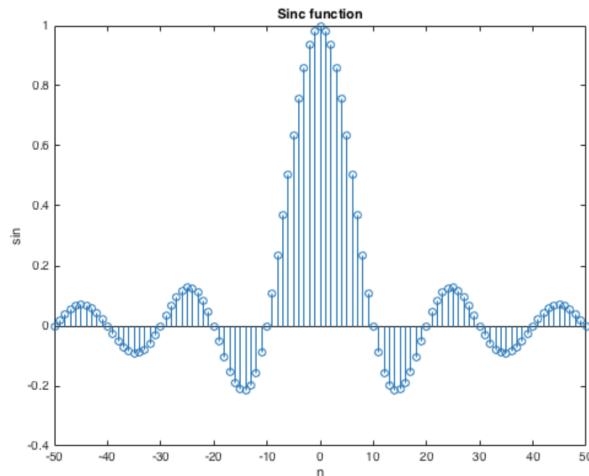


Figura 3.7: Función del seno cardinal $\text{sinc}(x)$ discretizada.⁴

³Más información en el siguiente enlace: https://en.wikipedia.org/wiki/Dirac_delta_function

⁴Fuente: <https://es.mathworks.com/matlabcentral/fileexchange/54530-lab-1-digital-signal-processing-sampling-and-quantization>

En el caso real, aparecen ciertas limitaciones. Para empezar, la función en frecuencia no tiene una banda de paso y una banda de rechazo totalmente plana y la frecuencia de corte no es totalmente abrupta. Además, la respuesta en el tiempo no puede ser infinita ya que no se puede multiplicar la señal de entrada por algo infinito. Por consiguiente, hay que truncar la señal por los dos extremos y además, como la señal es simétrica se puede “doblar” por la mitad para ahorrar recursos en su implementación práctica. Los coeficientes del filtro son simplemente los valores de la función $\text{sinc}(t)$ digitalizada. Esto se puede ver en la figura 3.7. Haciendo la transformada de Fourier inversa de un filtro real se obtiene la siguiente ecuación discreta como respuesta al impulso:

$$h[n] = \begin{cases} \frac{w_c}{\pi} & , n = 0 \\ \frac{w_c}{\pi} \left(\frac{\sin(w_c n)}{w_c n} \right) & , n \neq 0 \end{cases} \quad (3.3)$$

Donde w_c es la frecuencia normalizada en rad/s que se obtiene de la ecuación 3.4 y n tiene un rango de valores enteros de $0 \leq n \leq M$, siendo $M + 1$ el orden del filtro.

$$w_c = \frac{f_c}{f_m} \cdot 2\pi \quad (3.4)$$

Donde f_c es la frecuencia de corte y f_m es la frecuencia de muestreo ambas medidas en Hz. Hay que tener en cuenta, que la frecuencia de corte en un filtro FIR es directamente proporcional a la frecuencia de muestreo. Por ejemplo, usando el mismo set de coeficientes, la frecuencia de corte será el doble si se duplica la frecuencia de muestreo. En este proyecto, como la frecuencia de muestreo es un valor fijo, la frecuencia de corte se obtendrá en función de los parámetros del filtro.

En conclusión, la definición del filtro, si se sabe el orden que va a tener, se obtiene con las ecuaciones 3.3 y 3.4. Esta es la forma más burda de crear un filtro FIR, en la siguiente sección se hablará de métodos que refinan el diseño para la obtención de un resultado con mejores propiedades.

Por último mencionar que el truncamiento de la función $\text{sinc}(t)$ y su discretización en función del orden, son los dos factores que separan al filtro ideal del real. Por lo tanto, cuanto más grande sea el rango usado de la función $\text{sinc}(t)$ y cuanto más grande sea el orden del filtro, más se acercará la respuesta real a la ideal. Es decir, un menor rizado en las bandas de paso y rechazo y un corte en frecuencia más abrupto. Sin embargo, esto provocaría un incremento en el retraso de la entrada a la salida, una mayor potencia de computación y más recursos *hardware*. Hay que llegar a un equilibrio entre la respuesta que se quiere y los medios físicos disponibles.

Hasta ahora se ha explicado el procedimiento de obtención de los coeficientes para un filtro paso de baja. ¿Cómo se aplica esta técnica a los diferentes tipos de filtros en frecuencia? La respuesta a esta pregunta es bastante sencilla. Simplemente hay que desplazar a la frecuencia deseada la banda de paso. Esto queda ilustrado en la figura 3.8 obtenida de [21].

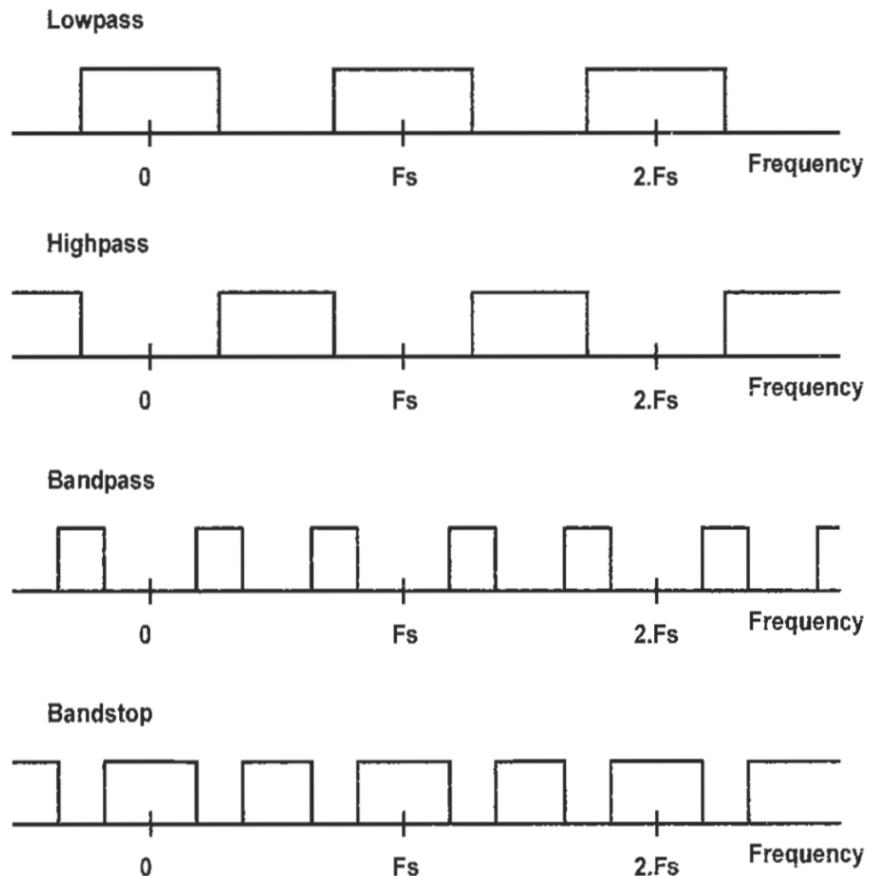


Figura 3.8: Diferentes tipos de filtros ideales y normalizados en frecuencia.

En la figura, F_s es la frecuencia de muestreo en Hz. Haciendo la transformada inversa de Fourier de cada una de las funciones representadas en la imagen, se obtienen sus correspondientes funciones en el tiempo. Ante una entrada impulso, todas se comportan como la función $\text{sinc}(t)$ pero desplazada en el tiempo o invertida. Por lo tanto, el desarrollo matemático para obtener los coeficientes de filtros de paso alta, paso de banda o rechazo de banda, es muy similar al de paso baja que se ha explicado con anterioridad.

3.2.3. Métodos de diseño

En esta sección se habla de los métodos de diseño más utilizados para la creación de filtros FIR. No se entrará en detalles matemáticos de los métodos ya que en este proyecto se ha usado el asistente de filtros de MATLAB como se explicará posteriormente. Simplemente se explica las características más importantes de cada método.

Lo explicado en la sección anterior es la forma más simple de obtener los coeficientes del filtro. El objetivo de esa explicación era entender el funcionamiento y la estructura de filtros FIR. Sin embargo, existen métodos más sofisticados y complejos para obtener unos coeficientes que generan una mejor respuesta. En este proyecto se explican las propiedades del método usando ventanas y los métodos algorítmicos ya que son los utilizados para esta aplicación.

Método empleando ventanas

Este método es bastante sencillo de entender. Simplemente los coeficientes obtenidos en la sección anterior se multiplican por una ventana. Esta ventana, no es más que una función discreta que trunca y modifica en alguna medida la función de respuesta al impulso de un filtro ideal. En la tabla 3.3, se pueden ver las ecuaciones de las ventanas más famosas y sus características.

Tabla 3.3: Propiedades de las diferentes ventanas.

| Nombre función ventana | Ancho de Transición (Hz) (normalizado) | Rizo (dB) Pasabanda | Relación (dB) lóbulo principal lóbulos laterales | Atenuación (dB) Máxima Parabanda | función ventana $w(n), n \leq (N-1)/2$ |
|------------------------|----------------------------------------|---------------------|--------------------------------------------------|----------------------------------|---------------------------------------------------------------------------------------------------|
| Rectangular | $0.9/N$ | 0.7416 | 13 | 21 | 1 |
| Hanning | $3.1/N$ | 0.0546 | 31 | 44 | $0.5 + 0.5\cos\left(\frac{2\pi n}{N}\right)$ |
| Hamming | $3.3/N$ | 0.0194 | 41 | 53 | $0.54 - 0.46\cos\left(\frac{2\pi n}{N}\right)$ |
| Blackman | $5.5/N$ | 0.0017 | 57 | 75 | $0.42 - 0.50\cos\left(\frac{2\pi n}{N-1}\right) + 0.08\cos\left(\frac{4\pi n}{N-1}\right)$ |
| Kaiser | $5.71/N(\beta = 8.96)$ | 0.000275 | | 90 | $\frac{I_0\left[\beta\sqrt{\left(1 - \left(\frac{2n}{(N-1)}\right)^2\right)}\right]}{I_0(\beta)}$ |

En la tabla, se pude apreciar con claridad que cuanto más se elimina el rizado más se alarga la banda de transición. Así, se necesitará identificar que característica del filtro es más crítica para, de este modo, elegir la ventana adecuada.

El método que se explicó en la sección anterior para obtener los coeficientes del filtro es como si se aplicara la ventana rectangular. Es decir, se trunca la señal ideal

sinc y se multiplican los coeficientes por la unidad. Para aplicar el resto de ventanas, simplemente se sigue el procedimiento explicado hasta ahora y una vez obtenidos los coeficientes, estos se multiplican por la función de la ventana elegida. Esto queda mejor ilustrado en las siguientes figuras obtenidas de [4].

En la gráfica de la figura 3.9, se representa el truncamiento y desplazamiento explicado anteriormente, o lo que es lo mismo, el uso de una ventana rectangular. Como se puede observar, la respuesta en frecuencia dista de la ideal. Aparece un rizado en las bandas de paso y de rechazo y la banda de transición no es instantánea. En la gráfica, el término M hace referencia a la longitud de la ventana. Idealmente, M sería un valor infinito.

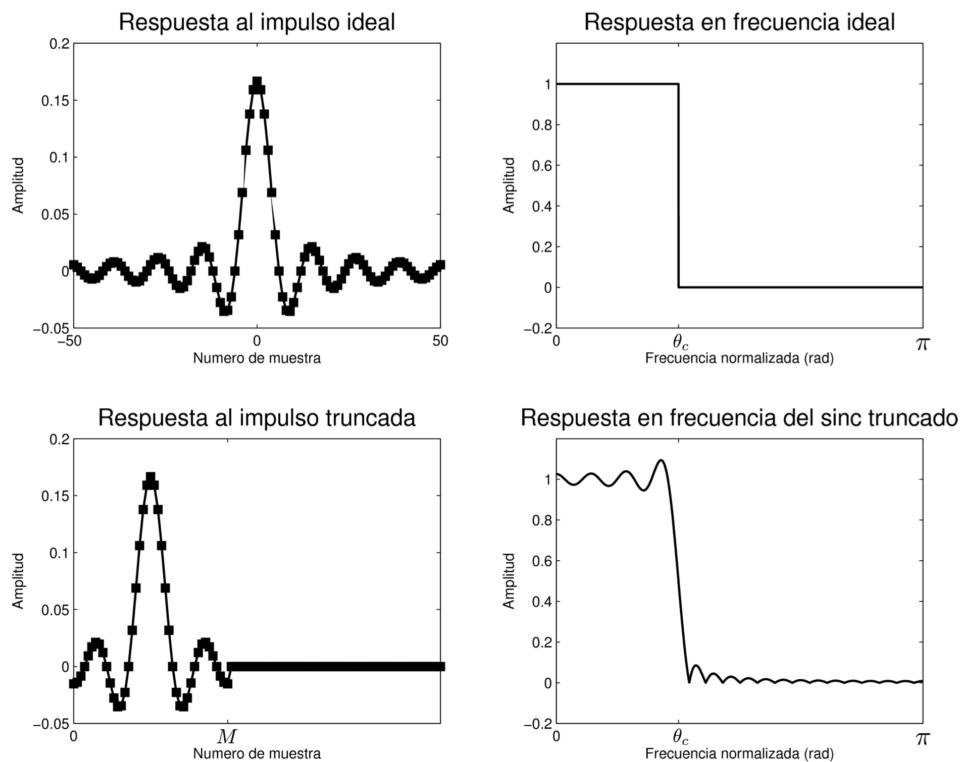


Figura 3.9: Efecto del truncamiento de la señal respuesta al impulso.

A continuación, en la figura 3.10, se muestra gráficamente el proceso de aplicación de una ventana *Blackman*. La operación se hace en el dominio del tiempo. La respuesta al impulso original es multiplicada por la función de la ventana de *Blackman* y el resultado se puede ver en la gráfica de abajo a la izquierda. Si se presta atención, se ve que además de modificar, la ventana trunca la señal original. En este caso se trunca entre -20 y 20 , es decir, una longitud de ventana de $M = 40$.

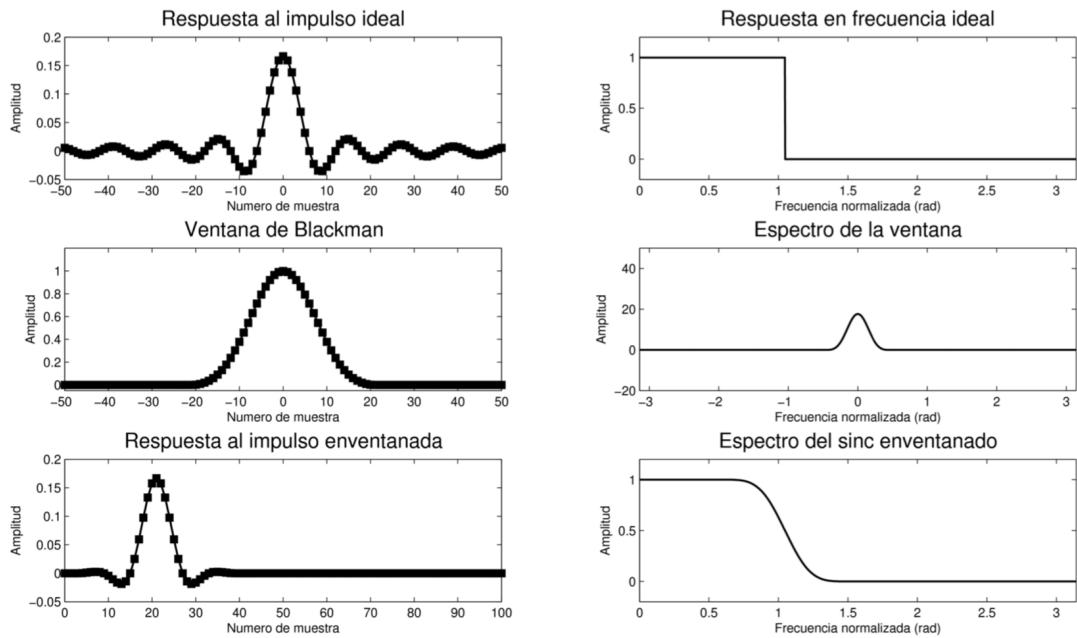


Figura 3.10: Proceso de aplicación de una ventana tipo Blackman.

La gráfica del espectro de la función *sinc* enventanada no tiene el rizado que tenía la ventana rectangular. No obstante, la banda de transición es más amplia. En la figura 3.11, se muestra una gráfica del espectro en frecuencia para la aplicación de diferentes ventanas para el mismo tamaño de enventanado, la misma frecuencia de corte y la misma frecuencia de muestreo. Esta, es una representación visual de lo mostrado en la tabla 3.3. Cuanto más se reduce el rizado, más aumenta la banda de transición.

Por otro lado, no es nada despreciable el efecto que tiene en la respuesta del filtro el tamaño de la ventana M . En la gráfica de la figura 3.12, están representadas las respuestas de un filtro con ventana rectangular para distintos tamaños de ésta. Como se puede ver, cuanto mayor es el tamaño de la ventana, más se acercaría la respuesta al caso ideal. En otras palabras, el rizado es menor y la banda de transición también.

Tanto el tamaño de la ventana, el orden del filtro, el tipo de ventana e incluso la densidad de ésta, son parámetros que se pueden introducir en el asistente de creación de filtros del *software* MATLAB que se verá en la próxima sección. También añadir, que cada ventana tiene una ecuación que determina su orden mínimo para un rizado y una frecuencia de corte dada. Esta opción de orden mínimo también aparece en el interfaz de diseño de MATLAB.

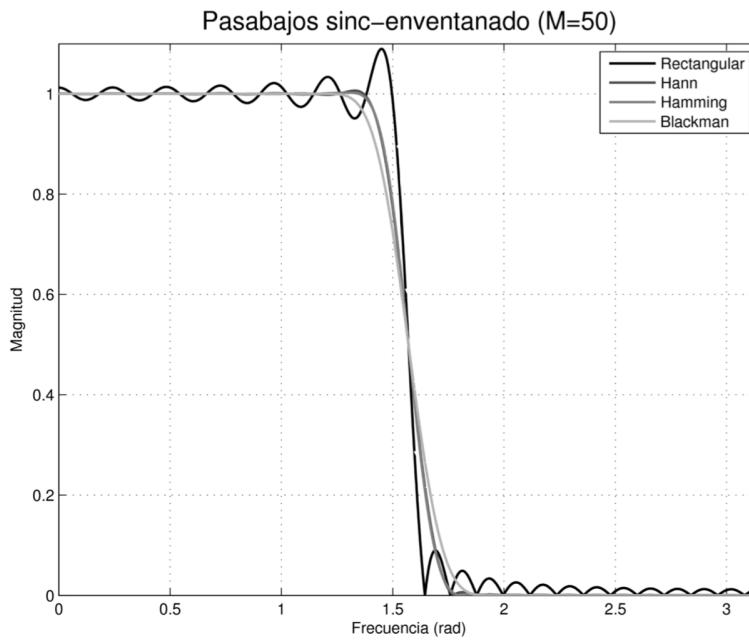


Figura 3.11: Respuestas en frecuencia para diferentes tipos de ventana.

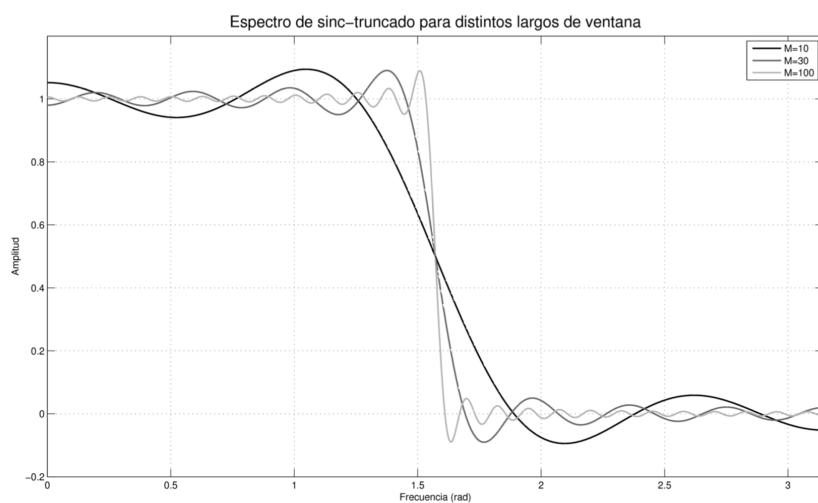


Figura 3.12: Respuestas en frecuencia para diferentes anchos de ventana.

Métodos algorítmicos

Matemáticamente, el concepto de filtro FIR se puede ver como un problema de minimización para reducir los errores que más interese atenuar según la aplicación. Lo difícil es encontrar la función que el algoritmo debe optimizar. No se entrará en detalle de como encontrar estas funciones ya que se escapa al ámbito del proyecto. Se confiará en los algoritmos que implementa internamente MATLAB. Sin embargo, en [15] hay un ejemplo de unos algoritmos basados en métodos de inteligencia artificial como los algoritmos genéticos por si se quiere ahondar en el tema.

A continuación, se hace una introducción básica a algunos de los algoritmos que implementa MATLAB para saber cual es la característica que intenta optimizar cada uno. Así, a la hora de diseñar el filtro se podrá elegir el que más se ajuste a las necesidades del proyecto. Destacar, que en el proceso de diseño está definido de manera más amplia en el capítulo 4.

- **Equiripple:** Como su propio nombre indica, intenta que el rizado tanto en la banda de paso como la banda de rechazo tenga la misma altura. Esto hace que se minimice el máximo error entre la respuesta del filtro deseada y la aproximación diseñada. Usa el algoritmo de Parks-McClellan para conseguirlo.
- **Mínimos cuadrados:** Este algoritmo busca minimizar el error total cuadrático entre la respuesta deseada y la aproximación diseñada. Este error está relacionado con la energía de la señal. Por consiguiente, si se desea que la señal filtrada tenga la mínima energía posible, es recomendable usar este método de diseño.
- **Mínimos cuadrados con restricciones:** Los métodos de equiripple y mínimos cuadrados intentan optimizar errores opuestos. Es decir, si se consigue minimizar el error global con el método de equiripple, se obtiene un error de la energía relativamente alto y viceversa. Una forma de solucionar esto es el uso del método de mínimos cuadrados con restricciones. Es decir, se busca optimizar el error energético pero poniendo restricciones al error global. También existe el método de equiripple con restricciones que hace lo opuesto.
- **Maximally flat:** Este algoritmo solo es aplicable a una respuesta paso de baja. Busca minimizar lo máximo posible el rizado de la banda de paso.
- **Interpolated FIR:** Este algoritmo intenta reducir al mínimo posible la carga computacional para el sistema digital donde es implementado el filtro.

3.3. Asistente de diseño de filtros en MATLAB

El software MATLAB ofrece una herramienta que asiste en el diseño y la generación de los parámetros del filtro con una interfaz gráfica intuitiva y amigable. Esta interfaz se puede ver en la figura 3.13. En esta sección, se explicarán los elementos de este asistente que han sido usadas en el presente trabajo fin de máster.

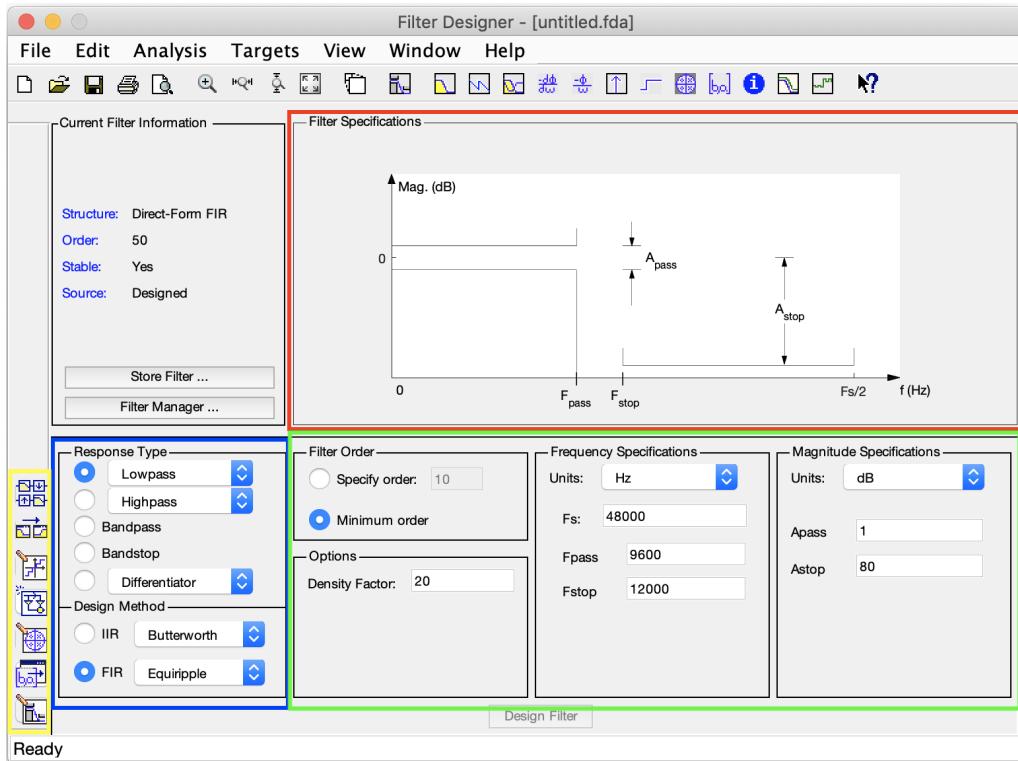


Figura 3.13: Asistente de diseño de filtros en MATLAB.

En la figura 3.13 están resaltadas las zonas de interés. En el recuadro amarillo se encuentra el acceso a diferentes menús, de los cuales, algunos serán de especial interés como se verá en el capítulo 4. Con estas herramientas se pueden obtener los coeficientes del filtro, hacer análisis del lugar de las raíces o hacer una transformación en frecuencia entre otras cosas.

En el recuadro azul se puede elegir la respuesta del filtro (en este proyecto se usará el HP o el LP) y el método de diseño, que como ya se ha explicado en el presente capítulo, será siempre de tipo FIR. Por otro lado, en el recuadro verde se muestran todas las opciones y variables que el usuario puede modificar en función del método escogido.

Por último, el recuadro rojo, muestra gráficamente las variables y especificaciones del filtro que pueden ser modificadas en el menú del recuadro verde.

3.3.1. Método de diseño de filtros FIR

Una vez vistas las partes del interfaz principal se va a explicar los diferentes métodos de diseño de filtros FIR que se pueden usar con este asistente. Simplemente con desplegar el menú de FIR se muestran todos estos métodos como se puede ver en la figura 3.14a. Básicamente, aparecen las mismas opciones ya presentadas en la sección anterior, además de otros algoritmos que no serán usados en este TFM.

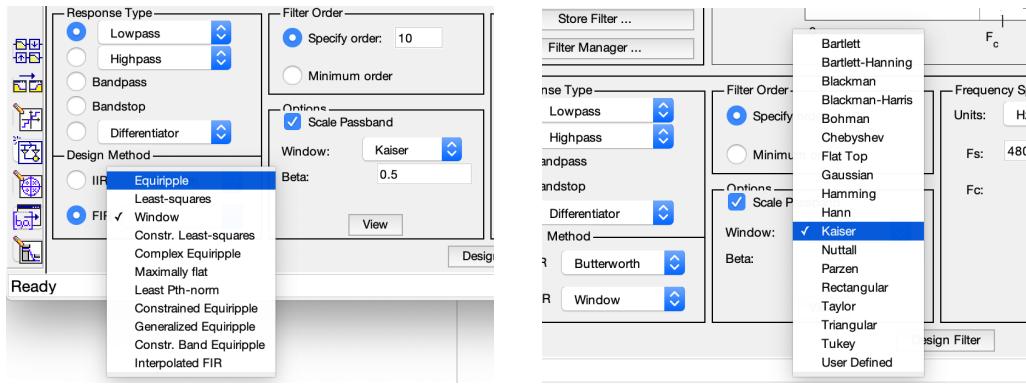


Figura 3.14: Métodos de diseño y tipos de ventanas implementadas por el asistente de MATLAB.

Si se escoge el método de ventanado, aparecen las opciones que se pueden ver en la figura 3.15. Si se despliega el menú de las ventanas aparecen todas las mostradas en la figura 3.14.

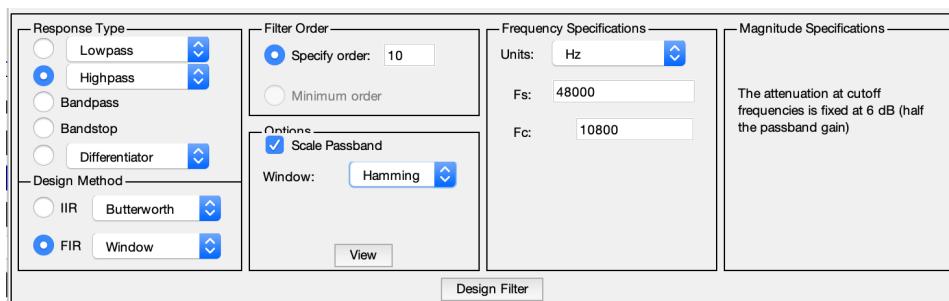


Figura 3.15: Menú de opciones del método de ventanado.

En este menú se puede especificar tanto el orden del filtro, como la frecuencia de muestreo (F_s) y la frecuencia de corte (F_c) en Hercios. Además la interfaz permite visualizar gráficamente la ecuación de la ventana si se pulsa el botón “View” e incluso diseñar una ventana personalizada.

En cuanto a los métodos algorítmicos, cada uno tiene sus propias especificaciones. Se pone como ejemplo el método de equiripple cuyos parámetros se muestran en la figura 3.16.

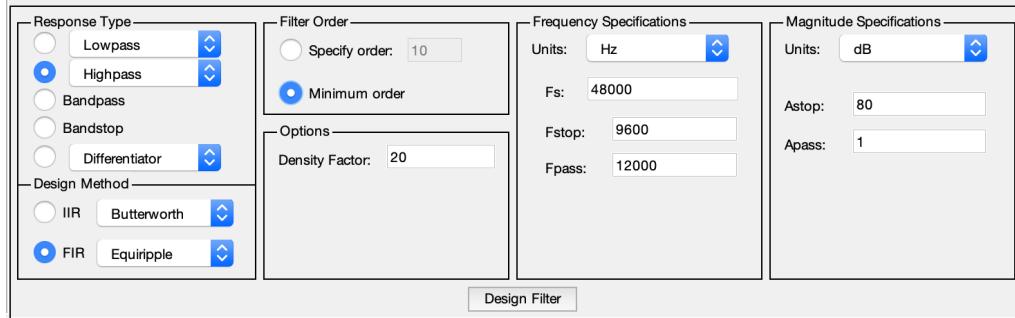


Figura 3.16: Menú de opciones del método de equiripple.

Además de las frecuencias de corte y de muestreo, hay que especificar el rizado de la banda de paso y la banda de rechazo (A_{pass} y A_{stop} respectivamente).

3.3.2. Resultado final, coeficientes

Una vez que se han rellenado todas las especificaciones se pulsa el botón “Design Filter” y el filtro se crea. Para obtener los coeficientes de este, se pulsa el menú que aparece recuadrado en azul en la figura 3.17. Estos coeficientes aparecen en el recuadro enmarcado en rojo en esta misma figura.

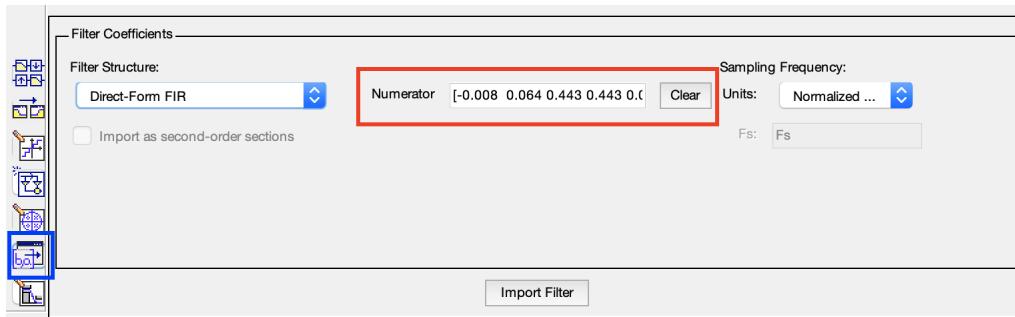


Figura 3.17: Coeficientes generados por el asistente de MATLAB.

Estos coeficientes son los que se usarán para la creación del filtro en la FPGA como se explicó en el capítulo 2.

Capítulo 4

Experimentación

Contenido

| | | |
|------------|---------------------------------------------------------|-----------|
| 4.1 | Resumen del proceso | 66 |
| 4.2 | Análisis de datos | 67 |
| 4.2.1 | Respuesta en el tiempo del sensor ante el deslizamiento | 67 |
| 4.2.2 | Espectro en frecuencia | 71 |
| 4.3 | Diseño del filtro | 75 |
| 4.3.1 | Diferentes aproximaciones de diseño | 75 |
| 4.3.2 | Filtros usados | 77 |
| 4.4 | Ajuste experimental del filtro | 81 |

Sinopsis

Una vez que se ha analizado la teoría de los filtros digitales, en este capítulo, se explica como se pasa a la práctica para el diseño y la implementación física de un filtro FIR en un dispositivo real.

Se hace un recorrido a lo largo de todo el proceso de experimentación llevado a cabo en este proyecto para la consecución del objetivo final.

4.1. Resumen del proceso

En esta primera sección se introduce el proceso de experimentación. Básicamente consiste en un flujo de trabajo que queda representado en el esquema de la figura 4.1.

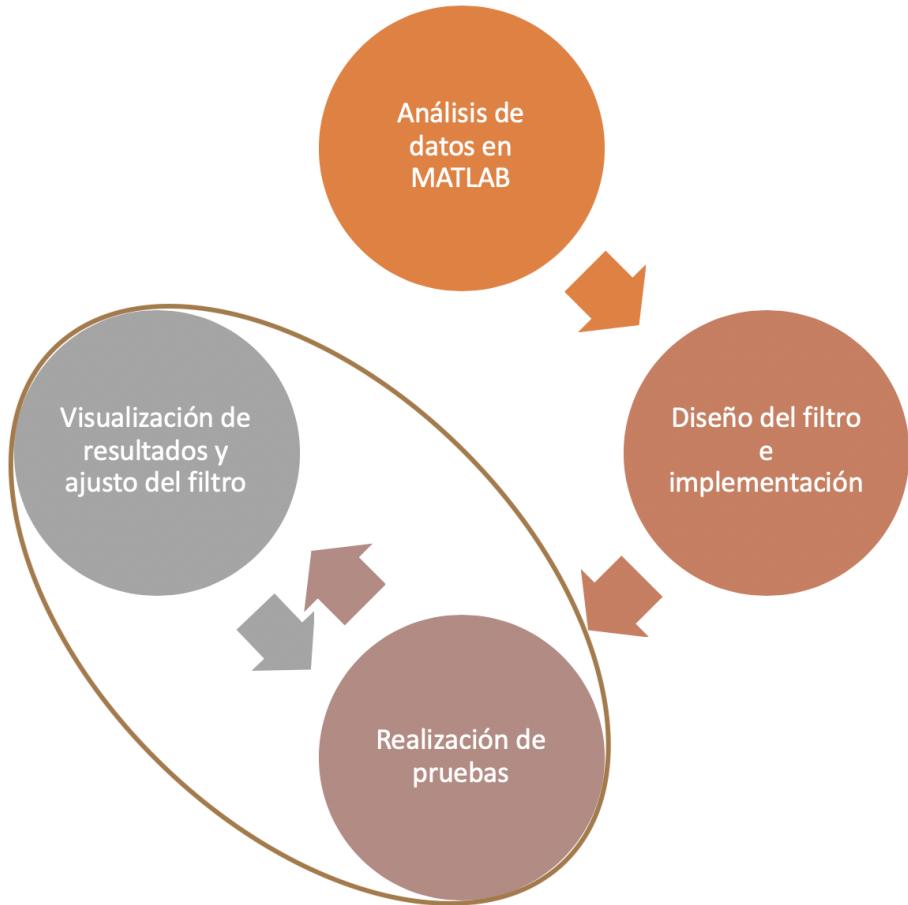


Figura 4.1: Esquema del flujo del proceso de experimentación seguido.

El primer paso es realizar una batería de pruebas de deslizamiento y analizar como son las señales medidas por los sensores en estos experimentos. A continuación, basándose en el análisis anterior, se diseña un filtro FIR adecuado para la detección de este deslizamiento en la señal. Una vez hecho esto, se implementa el filtro en la FPGA y se entra en un bucle de prueba y error para perfeccionar la solución. Es decir, se realizan nuevas pruebas de deslizamiento con el filtro de la FPGA, se analizan los resultados y se reajusta el filtro para una mejor respuesta.

4.2. Análisis de datos

Este es el primer paso que se llevó a cabo. Ya se disponían de algunas medidas de deslizamiento realizadas con el sensor por parte del equipo de trabajo EIS y además se hicieron nuevas medidas de prueba para tener datos de partida.

4.2.1. Respuesta en el tiempo del sensor ante el deslizamiento

En primer lugar, Se representan con MATLAB la respuesta en el tiempo de un táctel de la primera columna donde está apoyado el objeto antes de deslizar. El resultado se puede ver en la figura 4.2.

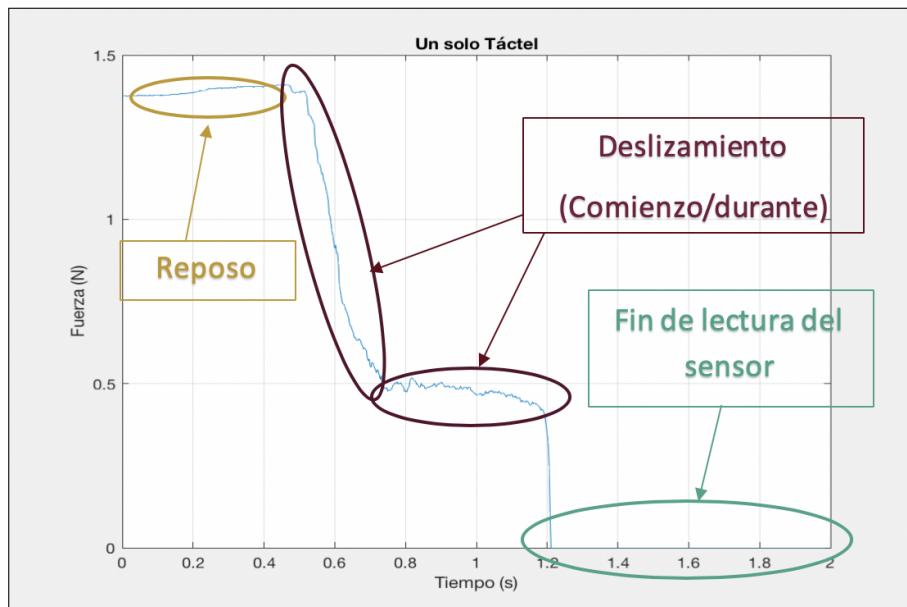


Figura 4.2: Fases del deslizamiento en un solo táctel.

Se ha elegido esta gráfica porque se pueden apreciar claramente las diferentes fases del deslizamiento. En un primer instante, el objeto (en este caso un carrito con una goma en la base para mayor adherencia) está en reposo y haciendo presión sobre el sensor. A continuación, hay una etapa de comienzo del deslizamiento en la que se reduce la presión sobre el sensor y continúa deslizando hasta que pasa la superficie del táctel y este ya no lee ninguna señal.

Aunque este gráfico es el más representativo del problema a tratar, las muestras no son siempre tan perfectas, ya que algunas veces el deslizamiento puede producirse de manera más rápida o de diferente forma y la gráfica queda algo distinta. No

obstante, en todas se pueden apreciar las tres fases que se muestran en la figura 4.2. En la gráfica 4.3 se muestra el resultado de otra prueba pero esta vez usando la pesa de $1Kg$ como objeto deslizante.

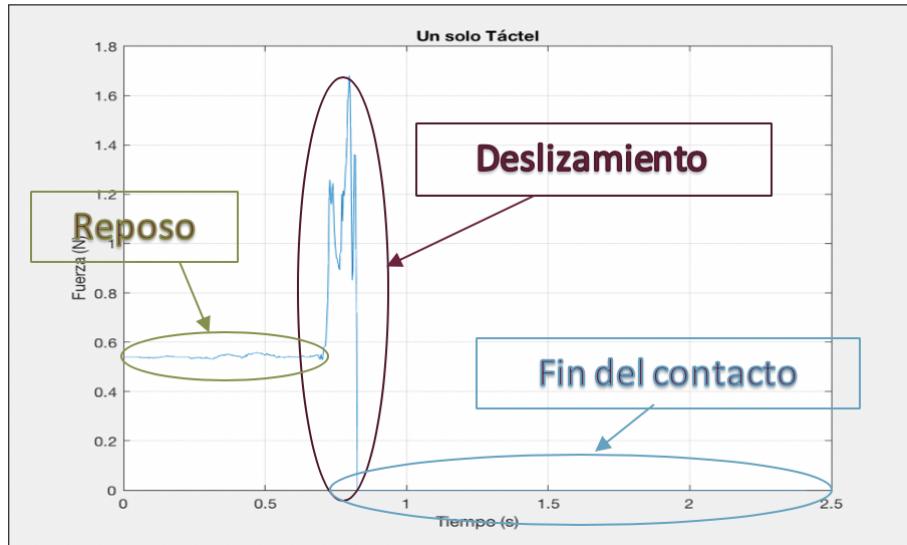


Figura 4.3: Otro ejemplo de deslizamiento, esta vez con pesa de $1Kg$.

La forma de la gráfica no es la misma que se vio en el primer experimento. En este caso al comenzar el deslizamiento, la presión sobre el sensor aumenta. Esto puede deberse a que cuando este objeto comienza a moverse sobre el sensor se genera un momento que lo hace inclinarse. Esto disminuye la superficie sobre la que desliza y aumenta la presión. Este fenómeno habrá que tenerlo en cuenta para este tipo de objetos, sin embargo, no se cree que esto afecte a la velocidad de las microvibraciones en las que se basa este estudio como se verá en el posterior análisis de la respuesta en frecuencia.

En el ultimo ejemplo mostrado en la figura 4.4 se ha usado el dedo como elemento deslizante sobre el sensor. En este caso, la acción la realiza un ser humano por lo que cuando está en reposo sobre el sensor no se ejerce una presión constante. Como se ve en la gráfica, desde que el sensor empieza a medir hay variaciones en la presión sobre el táctel. Por consiguiente, en esta gráfica no está claro a simple vista cuando el objeto empieza a deslizar y habrá que utilizar el análisis en frecuencia para ver la diferencia. Esto se verá posteriormente durante la experimentación. Cabe mencionar que también se han realizado experimentos usando el dedo como objeto deslizante pero con la capa de silicona introducida en el capítulo 1 y los resultados obtenidos en la representación de la presión respecto al tiempo es muy parecida a la mostrada.

Todos los ejemplos vistos hasta ahora muestran la lectura del deslizamiento de

un objeto pero con una medición inicial debido al elemento en reposo sobre el sensor. Estas son las mediciones de los táctiles que se encuentran más arriba en el plano inclinado, es decir, los de la primera fila. Sin embargo, es también importante ver que ocurre en un táctel donde el objeto está siempre deslizando, es decir uno de los táctiles intermedios de la matriz.

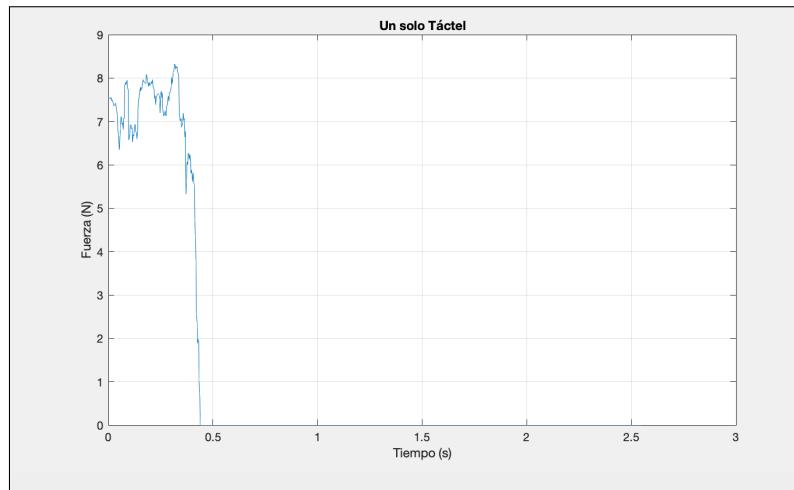


Figura 4.4: Otro ejemplo de deslizamiento, esta vez usando el dedo.

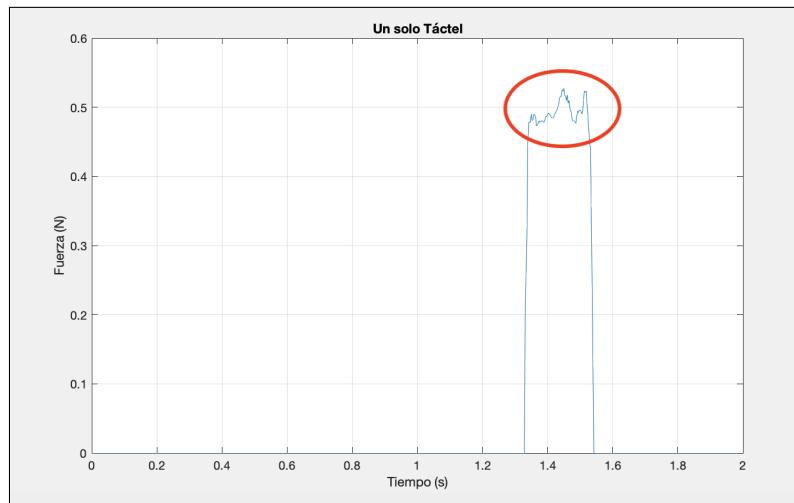


Figura 4.5: Lectura de un táctel que se muestra en una zona intermedia de la matriz.

En el caso de la figura 4.5 se puede ver la lectura de un táctel intermedio de la matriz. Este no detecta nada hasta que el objeto se desliza por encima y, cuando termina el deslizamiento, vuelve a no detectarse ninguna medición. Durante este

fenómeno se producen las mismas microvibraciones vistas en ejemplos anteriores, señaladas con un círculo rojo en la imagen.

En la siguiente figura 4.6 se muestra una representación de todos los táctiles de la matriz y la presión que detectan durante el tiempo. Como se puede observar, entre la medición de cada fila hay un pequeño espacio temporal en la que el objeto deslizante no activa ningún sensor. En este caso, la primera fila es la única que detecta al objeto en reposo. Además, como el objeto va aumentando su velocidad mientras transcurre el deslizamiento las mediciones de las últimas filas son más rápidas y contienen menos puntos para su análisis.

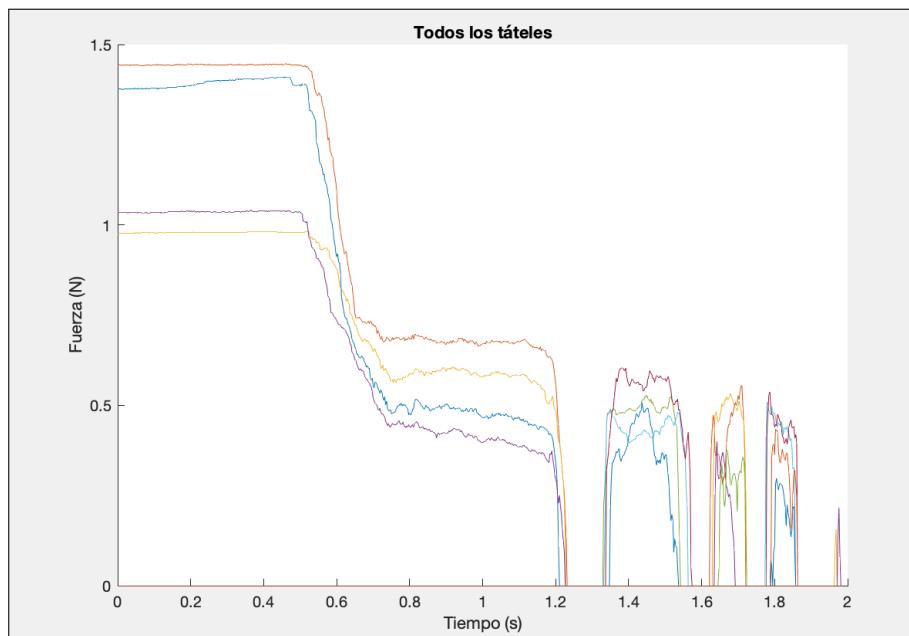


Figura 4.6: Representación de las lecturas de todos los sensores durante el deslizamiento.

Es importante aclarar que en este proyecto se analizará cada táctel individualmente y no se tiene en cuenta si hay alguna relación entre las mediciones de sensores diferentes pero cercanos en el espacio. Así, la señal de cada sensor es filtrada por separado por el mismo filtro digital y cada uno es capaz de detectar si un objeto está deslizando por sí mismo.

En posteriores análisis de este proceso de experimentación se verá el comportamiento tanto de los sensores en los que hay una medición inicial como los intermedios en los que el objeto siempre está deslizando sobre el sensor. Se comprobará si ambos son válidos para la detección del deslizamiento. Sin embargo, debido a que el objetivo es detectar las etapas tempranas de éste, los táctiles que leen al objeto en reposo

son los que detectarán el desplazamiento antes y por lo tanto los más interesantes para la consecución del propósito de este proyecto.

Por todo lo mencionado anteriormente, el análisis y el diseño de los filtros se enfocará sobre los sensores en los que repose el objeto a deslizar antes de que el deslizamiento se produzca.

4.2.2. Espectro en frecuencia

Debido a que la detección del deslizamiento se basa en la velocidad de las vibraciones leídas por el sensor, es fundamental hacer un análisis en frecuencia de la señal de presión respecto al tiempo. Para llevar a cabo esta tarea se ha usado principalmente el asistente de MATLAB descrito en el capítulo 2 además de diferentes gráficas y simulaciones realizadas con el código que se puede ver en el apéndice A.

Lo primero ha sido usar la transformada rápida de Fourier (FFT) y la representación del espectrograma de la señal para ver cuales son las frecuencias o rango de frecuencias predominantes en la señal. Como se hizo en la sección anterior, se mostrará una gráfica por cada objeto deslizante que se ha usado para la experimentación. La primera figura 4.7, muestra el espectrograma y la potencia en decibelios de cada componente en frecuencia.

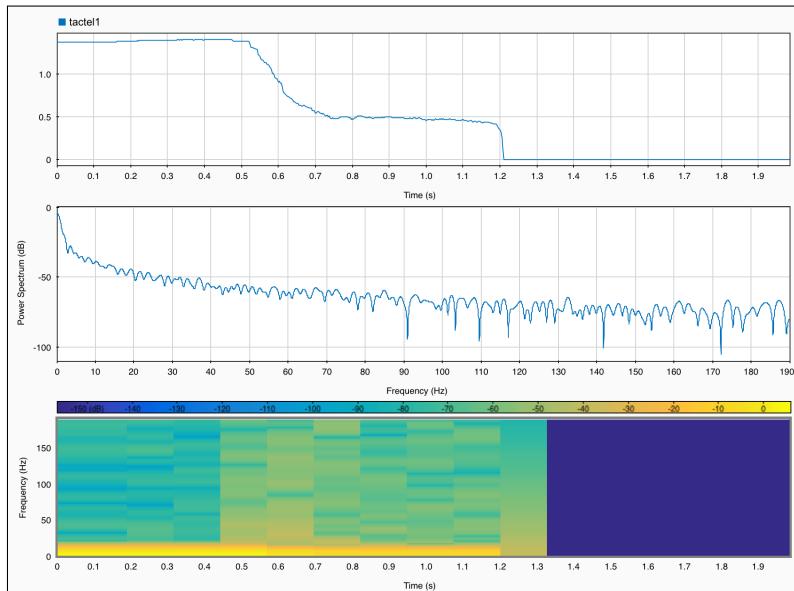


Figura 4.7: Análisis en frecuencia de la lectura de un táctel. Deslizamiento del carrito.

Esta primera gráfica representa la fuerza leída respecto al tiempo que ejerce un carrito deslizando (ya se vio en la sección anterior), después aparece representada la

potencia de la señal por frecuencias y la última gráfica muestra el espectrograma de la señal. Se puede ver claramente en el espectrograma que en la parte en la que se produce el deslizamiento hay más componentes en frecuencias altas, esto corrobora la hipótesis inicial del proyecto.

Por otro lado, si se atiende a la gráfica de las potencias de cada componente en función de la frecuencia, vemos que predominan las frecuencias bajas. Esto nos indica que la mayor parte de la potencia de la señal probablemente venga de la alimentación en continua de la FPGA que queda representada en la señal que es transmitida al ordenador y también, que cuando el objeto está en reposo la señal es prácticamente continua. También se ve que las frecuencias de esta gráfica solo llegan a 190Hz que es la mitad de la frecuencia de muestreo, debido al teorema de Nyquist-Shannon que se mencionó en el capítulo 3. Esto limita el estudio de las frecuencias más altas y no se puede obtener demasiada información de esta gráfica. No obstante, existen picos de frecuencias altas que hace ver que también existen estas componentes y que por lo tanto podrán ser filtradas.

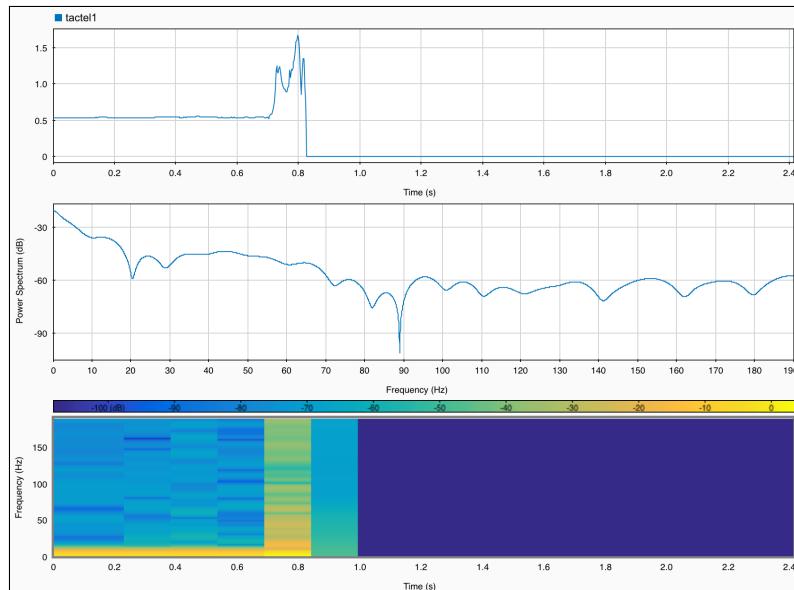
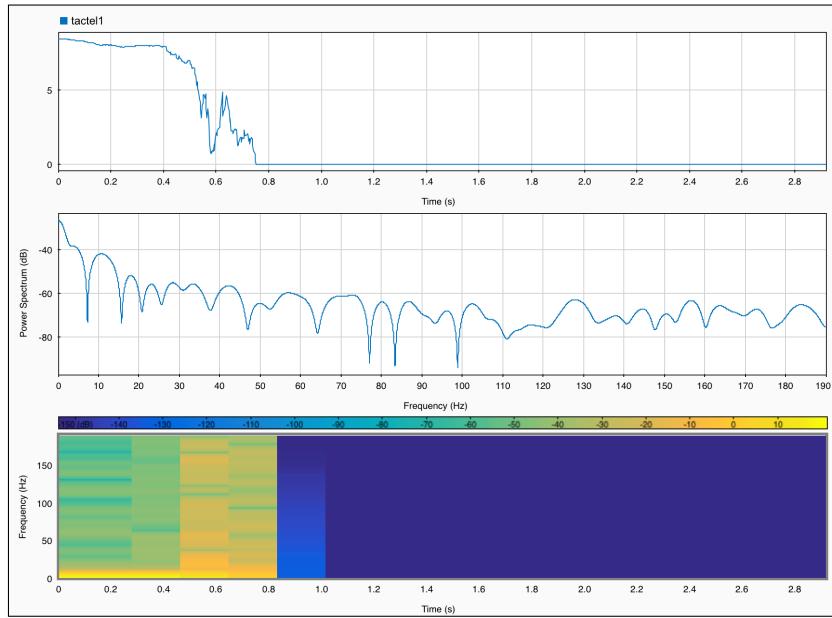
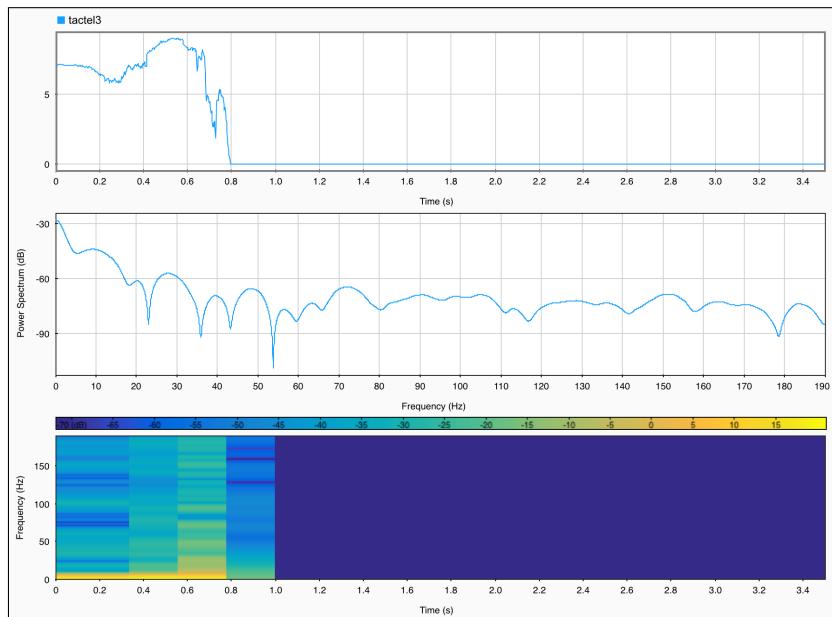


Figura 4.8: Análisis en frecuencia de la lectura de un táctel. Deslizamiento del rodillo en vertical sobre su base.

En la figura 4.8 se muestra, en este caso, el análisis en frecuencia del deslizamiento de la pesa de 1 Kg en vertical. Como en el caso anterior, durante el deslizamiento se puede apreciar claramente en el espectrograma un aumento de las componentes de frecuencias más altas. Además, para este experimento se puede ver en la gráfica del espectro en potencia que las frecuencias más altas tienen mayor potencia.



(a) Deslizamiento del dedo.



(b) Deslizamiento del dedo usando lámina de silicona.

Figura 4.9: Análisis en frecuencia de la lectura de un táctel. Deslizamiento de un dedo.

En las gráficas de la figura 4.9 se ve el análisis en frecuencia del deslizamiento de un dedo con lámina de silicona y sin ella. En la primera (4.9a) se desliza el dedo directamente sobre el sensor. Como se mencionó anteriormente, existen vibraciones en el estado de reposo sobre el sensor ya que es muy difícil que un ser humano ejerza una fuerza constante. Sin embargo, si se observa el espectrograma las componentes de estas vibraciones son menores que las que se producen mientras el dedo desliza. Esto hace posible que se pueda separar fácilmente esta señal con el filtro.

Por otro lado se tiene la gráfica 4.9b en la que se ha usado la lámina de silicona entre el dedo y el sensor. No se aprecian diferencias significativas respecto a los otros casos de deslizamiento. El espectrograma de esta señal es similar al de los demás.

Como conclusión del análisis previo de frecuencia, se puede decir que para todos los objetos deslizados sobre el sensor se puede ver que aparecen componentes de altas frecuencias durante el deslizamiento lo que confirma la hipótesis inicial de este proyecto. Por otro lado, atendiendo al espectro en potencia en todas las gráficas se aprecia una disminución de la potencia a partir de los 30-60 Hz. Esta es la componente en continua y de alimentación de la red (50 Hz en España) que se quiere eliminar. Por consiguiente, esto da una idea de cuál será la frecuencia de corte a la hora de diseñar el filtro.

4.3. Diseño del filtro

Una vez analizado los datos experimentales obtenidos desde el sensor es la hora de diseñar un filtro que aísle las frecuencias altas que se producen durante el deslizamiento. Para ello, se ha seguido principalmente la guía que se propone en la documentación de MATLAB para el diseño de un filtro FIR con este software [7].

4.3.1. Diferentes aproximaciones de diseño

En primer lugar, es importante recordar los diferentes parámetros que definen un filtro. En la figura 4.10, se muestran gráficamente los elementos en frecuencia de un filtro paso de baja que podrán ser optimizados. Esto es análogo para un filtro paso de alta.

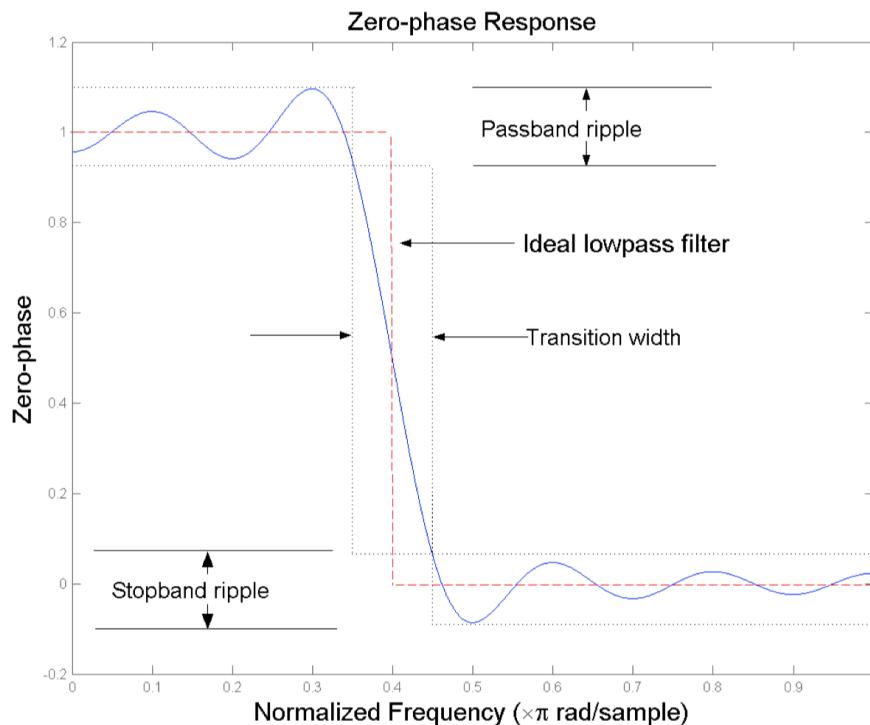


Figura 4.10: Elementos principales en frecuencia de un filtro LP. Fuente [7].

Para el proceso de diseño, normalmente se fijan unos valores y se van ajustando otros en función del resultado que se quiera obtener. En general, existen tres parámetros fundamentales interrelacionados entre sí que son: el orden del filtro, el ancho

de la banda de transición y el rizado de las bandas de paso y rechazo. Se pueden representar como una metáfora en la que cada especificación es un ángulo de un triángulo como se puede ver en la figura 4.11.

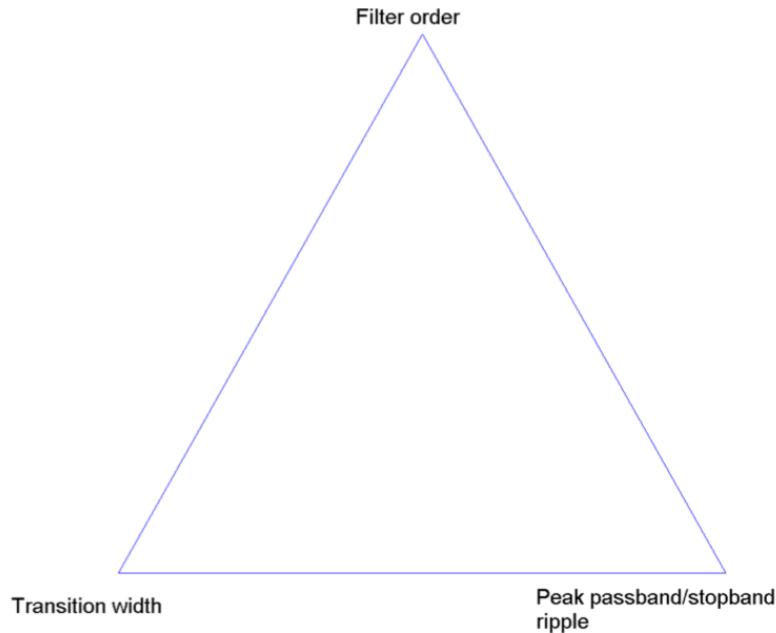


Figura 4.11: Especificaciones de un filtro FIR representadas como un triángulo. Fuente [7].

De esta forma se entienden los grados de libertad que quedan disponibles cuando se van estableciendo las especificaciones de diseño. De la misma forma que la suma de los ángulos de un triángulo es siempre la misma, cuando se fijan dos de las especificaciones mostradas la otra es dada por el algoritmo de diseño utilizado. Además, como los ángulos de un triángulo, si una de las especificaciones se incrementa o decremente tendrá un impacto en el resto de las especificaciones.

Para este caso interesa que los coeficientes del filtro sean simétricos, lo que se consigue usando una ventana simétrica. Esto permite que una vez implementado en la FPGA, el filtro use la mitad de multiplicadores. Por consiguiente, el primer método de diseño que se ha probado ha sido el de ventanado con una ventana tipo Kaiser. Con este método se fija el orden del filtro y el ancho de la banda de transición que se quiere conseguir y el algoritmo da el rizado de las bandas de paso y rechazo. Esto obviamente provoca que la respuesta del rizado y del error medio no sea óptima como lo sería si se usara el método de equiripple. También, como se dijo anteriormente, si se quiere que la señal de salida tenga la menor energía posible se

usará el método de mínimos cuadrados. Sin embargo, eso no es una prioridad para esta aplicación.

4.3.2. Filtros usados

Una vez que se sabe lo mencionado en la sección anterior, aquí se mostraran diferentes ejemplos de los filtros que se han usado para realizar las pruebas. En primer lugar se ha usado un filtro paso de alta con una ventana Kaiser cuyos parámetros se pueden ver en la figura 4.12.

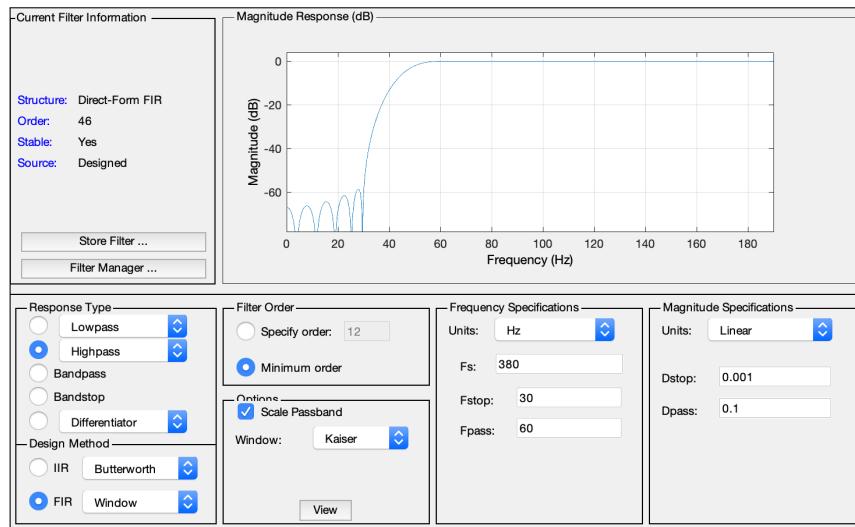


Figura 4.12: Especificaciones de filtro paso de alta usando una ventana Kaiser.

En este caso se ha fijado la anchura de la banda de transición (entre los 30 y los 60 hercios) y la amplitud del rizado en las bandas de paso y de rechazo. Esto deja como grado de libertad el orden del filtro, por lo que el algoritmo nos da el orden menor que puede obtener para conseguir los parámetros ya fijados.

En este caso, para las restricciones que se ven en la figura se obtiene un filtro con un orden de 46. Este valor se considera bastante alto porque significa que la salida del filtro tendrá un retraso de 46 imágenes táctiles respecto a la señal de entrada. En este proyecto, donde se quiere una respuesta casi inmediata, este valor se considera fuera de lo aceptable.

Por consiguiente, el siguiente método de diseño que se ha probado es un ventanado Hamming en el que se puede especificar el orden del filtro que se quiere. Se ha escogido este tipo de ventana porque como se dijo en capítulos anteriores tiene una banda de transición menor en comparación con las demás. Esto implica un mayor

rizado pero este parámetro es el menos crítico para el proyecto. Los resultados y parámetros de este método se pueden ver en la figura 4.13.

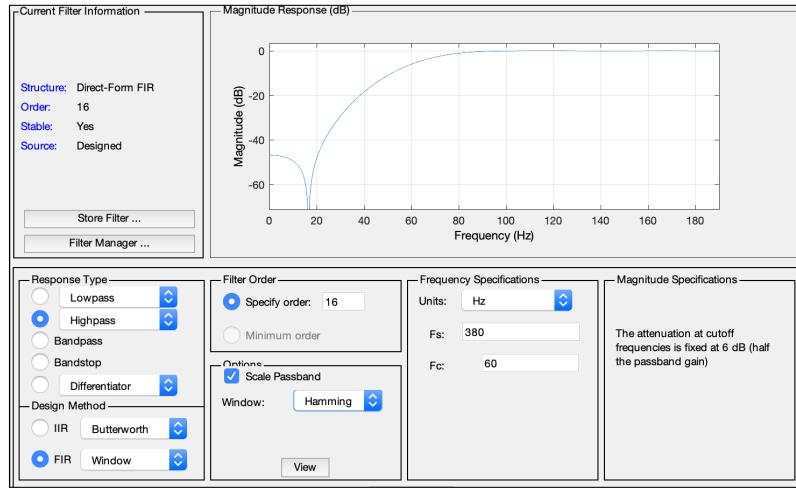


Figura 4.13: Especificaciones de filtro paso de alta usando una ventana Hamming.

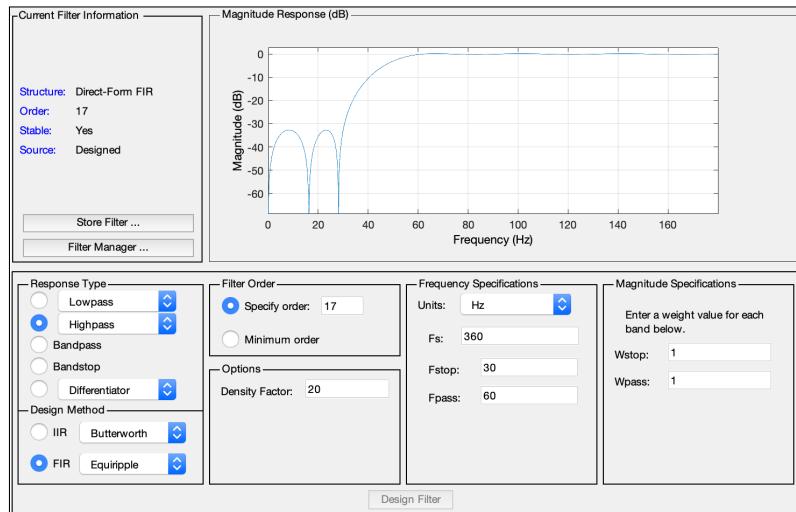


Figura 4.14: Método de equiripple para diseño de filtro paso de alta fijando el orden.

En este caso, se obtiene un orden de filtro bastante menor pero tanto el ancho de la banda de transición como el rizado del filtro no son buenos.

Pasamos ahora al método equiripple que trata de minimizar el error total de la señal de salida respecto a la salida del filtro ideal. Este método nos permite tanto fijar el orden que se quiere (figura 4.14) como dejar que lo establezca el propio

algoritmo en función de los valores que se le de a los otros dos parámetros (figura 4.15).

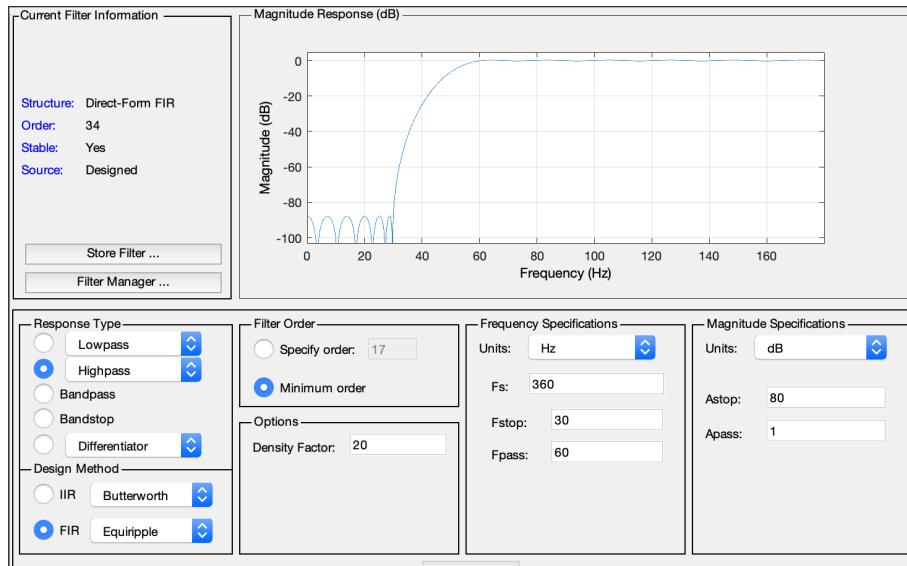


Figura 4.15: Método de equiripple para diseño de filtro paso de alta marcando la opción de orden mínimo.

Obviamente si el orden es mayor, la respuesta del filtro será mejor. Sin embargo, podemos ver que para el mismo orden, el método de equiripple da mejor resultado en cuanto al ancho de la banda de transición que los métodos de ventana.

Por último, se ha probado con el algoritmo de equiripple con restricciones que trata de llegar a un compromiso entre el error total y el error cuadrático medio para conseguir un resultado más equilibrado. En este método también se puede ajustar el orden del filtro así como la frecuencia de corte (sin especificar el ancho de la banda de transición) y el rizado de las bandas de rechazo y transición. Las especificaciones y diagrama en frecuencia del filtro creado con este método se puede ver en la figura 4.16.

Todos estos filtros, además de aplicarlos a los datos de partida en el propio MATLAB, también se han implementado en la FPGA para ver la diferencia con los resultados del ordenador. Se ha demostrado que la salida de la simulación por ordenador y la salida del mismo filtro en la FPGA son prácticamente idénticas. Por lo tanto, una vez comprobado esto y para ahorrar tiempo, el resto de las pruebas se han hecho directamente simulando los filtros en MATLAB. Esto se debe a que el proceso de programación e implementación del filtro en la FPGA es un proceso largo y poco eficiente.

Dicho lo anterior, todo el proceso iterativo que se explica en la siguiente sección

se ha realizado por simulaciones en ordenador de los datos de fuerza obtenidos de los experimentos.

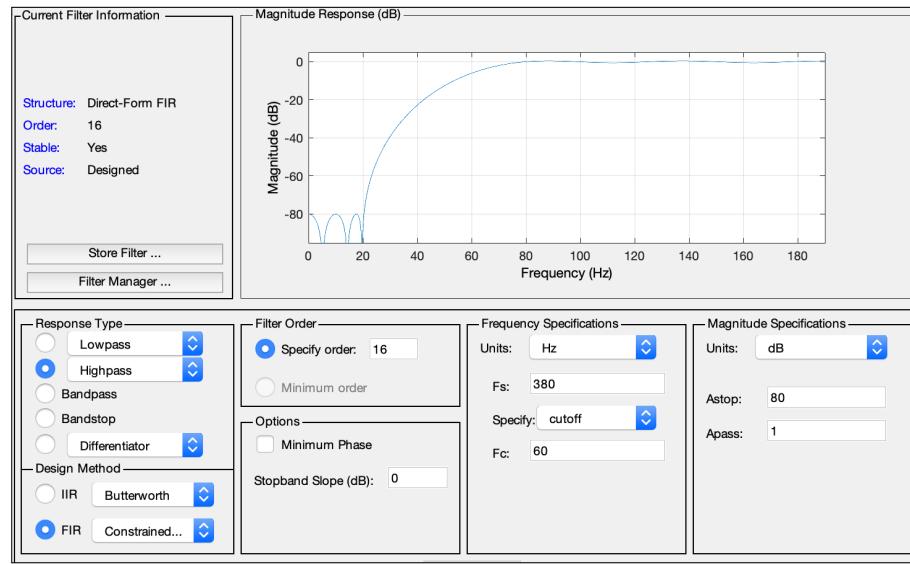


Figura 4.16: Especificaciones de filtro paso de alta usando el método de equiripple con constricciones.

4.4. Ajuste experimental del filtro

Este proceso consiste en la prueba de diferentes filtros aplicados a los datos obtenidos con los experimentos de deslizamientos en el banco de pruebas. Se analizan con MATLAB las salidas del filtro basándose principalmente en dos parámetros de la señal: el offset en continua y la amplitud máxima. En la figura 4.17 se muestra abajo la gráfica de la señal de salida de un filtro para la señal de entrada de fuerza de la gráfica de arriba.

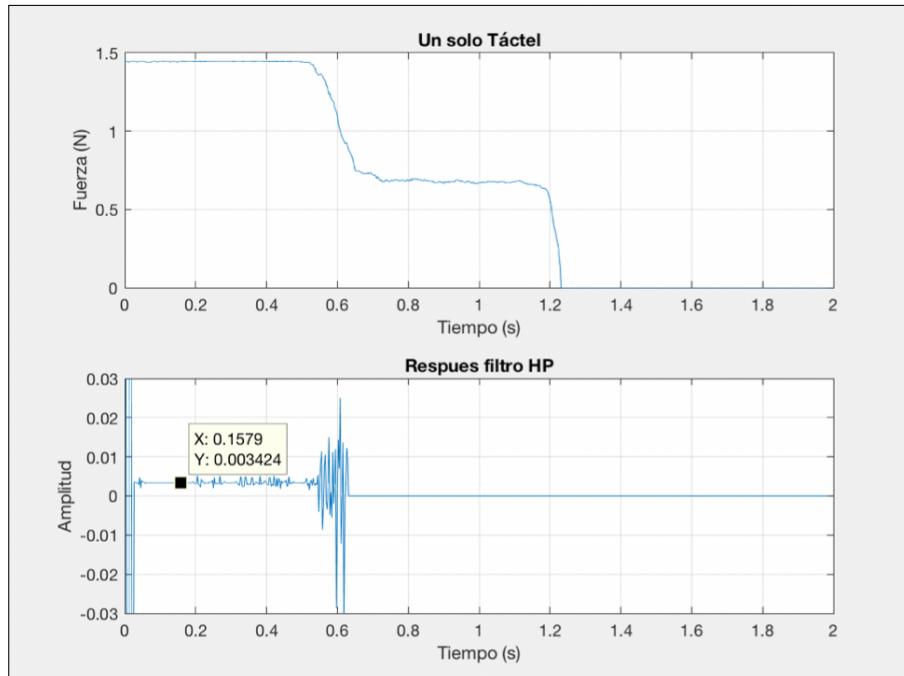


Figura 4.17: Ejemplo de señal de fuerza y salida del filtro paso de alta.

Como se puede ver en la imagen, con el cursor se mide la componente de continua que para este caso de ejemplo es de 0,003424. El valor máximo del valor absoluto de la señal (para tener en cuenta también la parte negativa) se mide con la función `max(abs(tactFilH1))` de MATLAB pero despreciando los primeros valores del filtro que no son significativos. De esta forma siguiendo con el ejemplo de la figura, se han usado los datos de un experimento usando el carrito como objeto deslizante y filtros con frecuencias de corte de: 30, 60, 80, 120 y 160 Hz. El resto de los parámetros del filtro se mantienen constantes, el orden, la atenuación en la banda de rechazo, el tipo de filtro, etc. La mayoría de estos parámetros se han dejado en los valores por defecto y se ha establecido el orden en 16 ya que se considera que es un retraso aceptable. Teniendo en cuenta que la frecuencia de muestreo es de 380 Hz el retraso

en segundos sería aproximadamente:

$$\text{retraso} = \frac{16}{380} = 42ms \quad (4.1)$$

Dicho esto se obtiene la siguiente tabla con los datos de los resultados del experimento anterior:

Tabla 4.1: Resultado de pruebas de frecuencia para deslizamiento con carrito.

| Prueba para filtro orden 16 | Offset | Amplitud máx. |
|---------------------------------------|-----------|---------------|
| Filtro con frecuencia de corte 30 Hz | 0.003424 | 0.2665 |
| Filtro con frecuencia de corte 60 Hz. | 0.0006484 | 0.2224 |
| Filtro con frecuencia de corte 80 Hz | 0.0006484 | 0.1897 |
| Filtro con frecuencia de corte 120 Hz | 0.0006484 | 0.1216 |
| Filtro con frecuencia de corte 160 Hz | 0.0006484 | 0.0541 |

El siguiente paso es representar estos datos en gráficas para ver con claridad la tendencia de estos parámetros cuando se aumenta la frecuencia de corte. En la figura 4.18 se ve la gráfica de la amplitud:

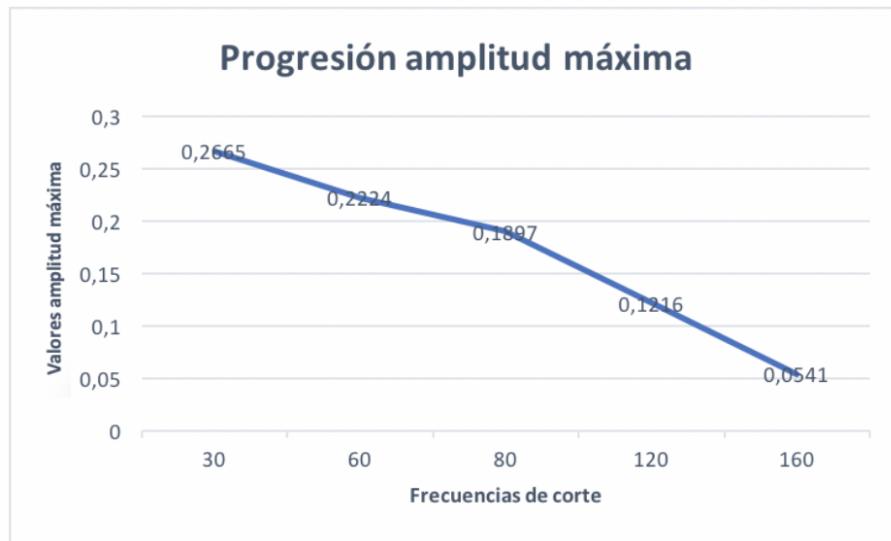


Figura 4.18: Progresión de valores máximos de amplitud con la frecuencia de corte.

Como es obvio, la relación entre la amplitud y la frecuencia de corte es inversamente proporcional. Cuando se aumenta la frecuencia de corte se atenúa un mayor rango de componentes de la señal resultando en amplitudes menores. Por el otro

lado, tenemos la progresión del valor de la señal en continua que se puede ver en la imagen

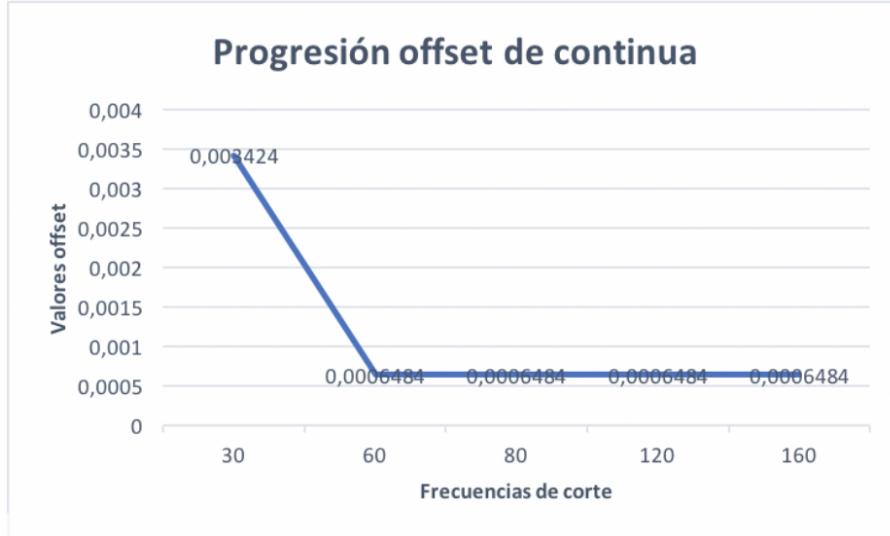


Figura 4.19: Progresión de componente en continua con la frecuencia de corte.

El resultado de esta progresión es muy interesante. Hay un corte claro en el que el offset de continua se reduce mucho y a partir del cual se estabiliza por mucho que se aumente la frecuencia de corte. Este valor de corte es de 60 Hz y aparece en todos los tipos de filtros con los que se ha hecho este experimento.

Por lo tanto, después de las pruebas realizadas en este bucle iterativo podemos sacar en claro que 60Hz es la frecuencia de corte más adecuada para reducir la componente de continua al mínimo y aun así obtener buenos valores de amplitud. Además, se considera que un orden entre 10 y 20 tiene un retraso adecuado para la aplicación y se obtienen buenas respuestas para detectar el deslizamiento. Con todo esto, y después de haber probado muchas combinaciones, en el siguiente y último capítulo se mostrará la solución final adoptada.

Capítulo 5

Análisis de los resultados y validación del algoritmo

Contenido

| | | |
|------------|-------------------------------------------|-----------|
| 5.1 | Filtro utilizado | 86 |
| 5.2 | Experimentos con el proyecto final | 87 |
| 5.2.1 | Test de deslizamiento | 87 |
| 5.2.2 | Test de fuerza | 90 |
| 5.3 | Conclusiones | 93 |
| 5.4 | Trabajo futuro | 94 |

Sinopsis

En este último capítulo se muestra la solución final del proyecto: filtro utilizado, programa para la visualización de los datos y detección del deslizamiento. Se hará un análisis del funcionamiento del sistema y el porqué de la solución escogida.

En este mismo capítulo se habla de las conclusiones finales del proyecto, destacando lo que se ha conseguido y proponiendo como trabajo futuro algunas posibles líneas de investigación adicionales.

5.1. Filtro utilizado

Después del todo el proceso de experimentación explicado en el capítulo anterior, el método de diseño que mejores resultados ha dado (mayor amplitud, y menor offset para el menor orden posible) ha sido el de equiripple con constricciones. En concreto, las características del filtro final se puede ver en la figura 5.1. El orden se ha establecido en 15 que da un retraso de algo más de 40 ms teniendo en cuenta el retraso de la electrónica. Esto se puede considerar prácticamente tiempo real. Por otro lado, la frecuencia de corte se ha establecido en 60Hz por el motivo explicado en el capítulo anterior. Por último, una atenuación en la banda de rechazo de 60dB se ha considerado suficiente para esta aplicación a través de la experimentación.

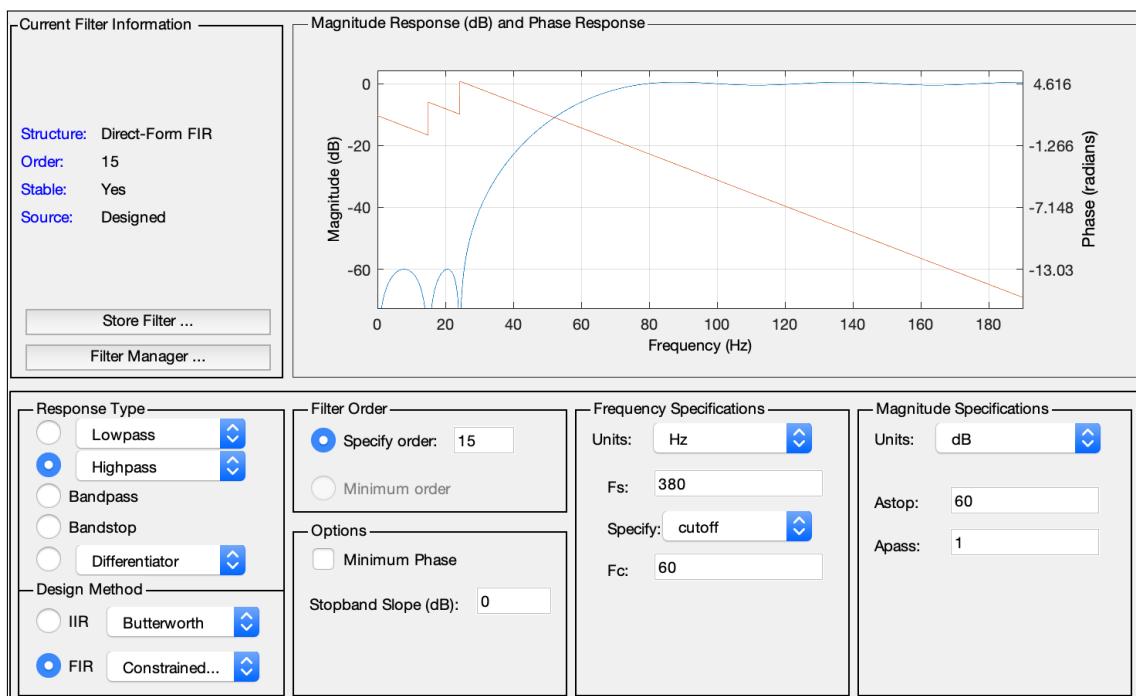


Figura 5.1: Representación y especificaciones del filtro final elegido para el proyecto.

Los coeficientes de este filtro se ven en la tabla 5.1. Como es un filtro FIR de fase lineal, los coeficientes son simétricos y por lo tanto a la hora de implementarlos en la FPGA se puede ahorrar la mitad de la lógica. En la tabla están representados los ocho primeros coeficientes y la otra mitad son los mismos pero de signo cambiado. Además se han representado todos los decimales que ofrece MATLAB, sin embargo hay que decir que la FPGA tiene una menor precisión y por lo tanto la respuesta final varía un poco. Pero de una forma no significativa.

Tabla 5.1: Coeficientes del filtro final.

| | |
|---------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Mitad de los coeficientes del filtro | -0.01370699983018093710829266029804784921 0.030768577691379774691204929126797651406 0.022208086452711734565257373219537839759 -0.014794862919806431311720018584310309961 -0.070180421761295325167040459746203850955 -0.086425462317581511206832090010721003637 0.023054476833018568149302751635332242586 0.56057745313253082297677565293270163238 |
|---------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|

Una vez se tienen estos coeficientes se implementa el filtro como se explicó en el capítulo 2. Posteriormente se procede a realizar experimentos con el montaje completo para verificar la validez de la solución adoptada.

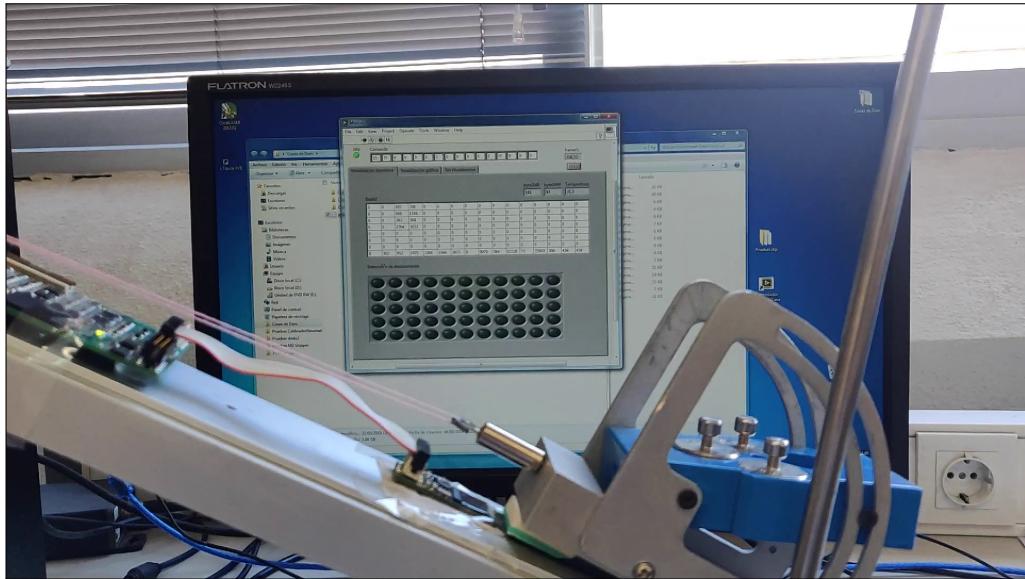
5.2. Experimentos con el proyecto final

Usando el montaje especificado en el capítulo 1 e implementado todos los algoritmos y el filtro final, se procede a realizar experimentos para ver como se comporta el sistema ante diferentes estímulos.

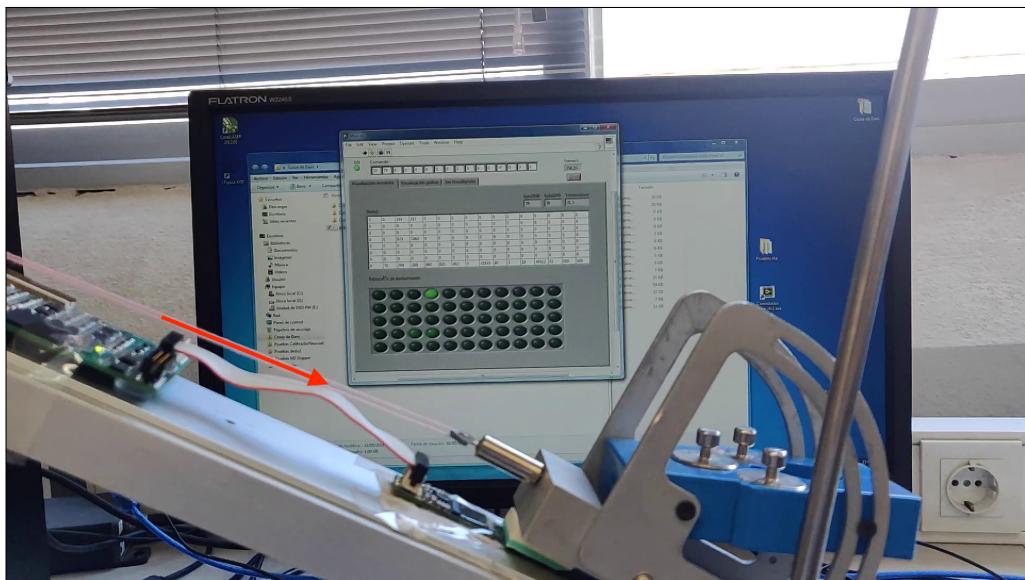
5.2.1. Test de deslizamiento

El primer elemento deslizante usado ha sido el carrito ya que es el caso más parecido al de una mano robótica que sujet a un objeto ofreciendo fuerza constante y es el que se usará como ejemplo. Se muestran fotos del banco de pruebas junto con la lectura por ordenador en tiempo real durante las diferentes fases del experimento.

Mientras el carro está apoyado sobre el sensor, este lee un valor prácticamente constante que corresponde a la fuerza de la gravedad sobre el plano inclinado. Esta sería la situación inicial del experimento (figura 5.2a). A continuación, se va reduciendo la fuerza que se ejerce en el hilo que sujeta el carrito con el objetivo de que este empiece a deslizar sobre el plano. En este momento, cuando se empieza a superar el coeficiente de rozamiento estático, ya se detecta deslizamiento en los tácteles sobre los que está apoyada la goma del carrito (figura 5.2b). Esto se puede ver porque se encienden los LEDs de los tácteles correspondientes en la aplicación de LabVIEW.

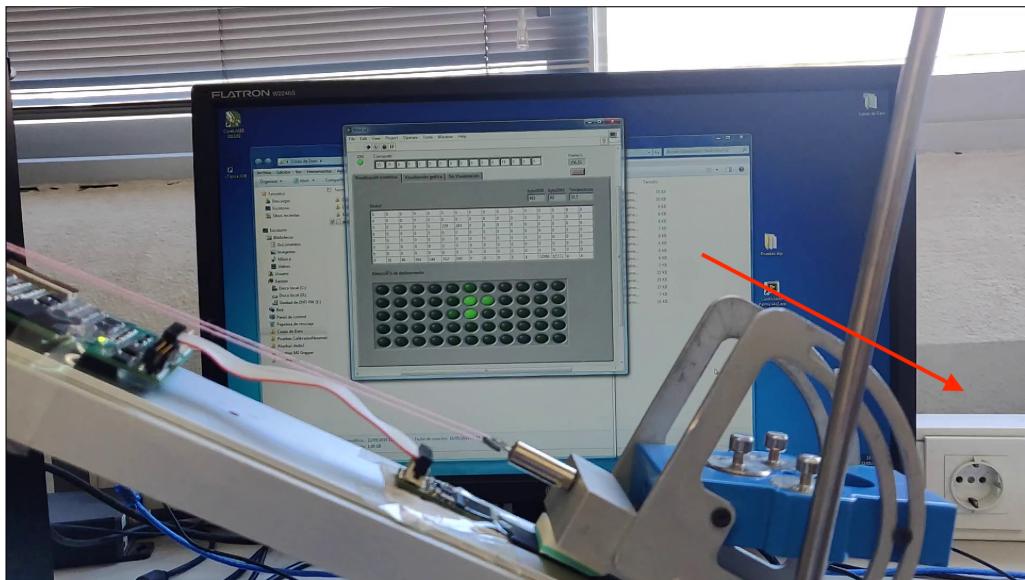


(a) Carrito apollado en reposo.

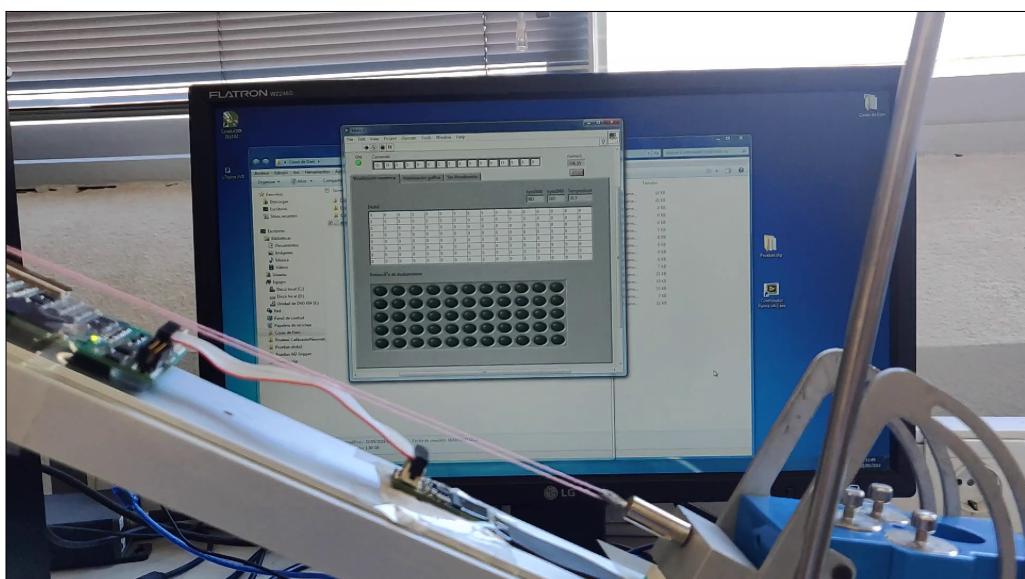


(b) Momentos previos al deslizamiento.

Figura 5.2: Momentos antes y al inicio del deslizamiento.



(a) Instante durante el deslizamiento del carrito.



(b) Final del deslizamiento, carrito fuera del sensor.

Figura 5.3: Momentos durante y al final del deslizamiento.

A continuación, el carrito empieza a moverse sobre el sensor (figura 5.3a) y se van activando los LEDs de los tácteles sobre los que va deslizando la goma. Esto, como ya se sabe, significa que estos sensores van detectando deslizamiento a medida que el carro desliza sobre ellos.

Por último, se muestra el momento en el que el carrito sobrepasa el sensor y deja de deslizar (figura 5.3b). Aquí, obviamente ni se detecta fuerza alguna ni deslizamiento ya que el sensor no está expuesto a ningún estímulo.

También se ha probado a usar la pesa de 1 Kg como objeto deslizante apoyada sobre su base de mayor superficie. Los resultados obtenidos son prácticamente idénticos a los mostrados con el carrito. En este caso, también se ha comprobado si los valores del offset en continua a la salida del filtro aumentan debido a que se está ejerciendo una mayor fuerza sobre el sensor. Estos valores son prácticamente iguales a los del experimento anterior. También hay que mencionar que se ha hecho una prueba deslizando el dedo sobre el sensor. Como el ser humano es incapaz de ejercer una fuerza constante, con el dedo en reposo había veces que se detectaba el deslizamiento. Sin embargo, ajustando un poco los umbrales se puede hacer que funcione si las variaciones de fuerza no son demasiado grandes.

5.2.2. Test de fuerza

Con el objetivo de ver cual es la máxima fuerza con la que funciona este algoritmo, se han apoyado diferentes pesos sobre el sensor. Esta prueba tiene sentido ya que cuando aumenta la presión ejercida sobre el sensor, también aumenta la componente en continua a la salida del filtro. Además, cuando la fuerza ejercida es excesiva, el sensor no lee bien el estímulo y se generan altas frecuencias que no son debidas al deslizamiento.

De esta forma, se ha apoyado primero la pesa de 1 *Kg* sobre su cara de menor superficie para ejercer más presión, a continuación se ha realizado la misma prueba con la pesa de 1 *Kg* pero añadiendo encima una pesa de 4 *Kg* haciendo un total de 5 *Kg* sobre el sensor. Teniendo en cuenta que la superficie sobre la que se apoya todo el peso es un círculo de radio 1,5 cm la presión para los dos casos se calcula como sigue:

$$\text{Presión } 1\text{Kg} = \frac{1 \cdot 9,8}{2 \cdot \pi \cdot (1,5 \cdot 10^{-2})^2} = 6932\text{Pa} \quad (5.1)$$

$$\text{Presión } 5\text{Kg} = \frac{5 \cdot 9,8}{2 \cdot \pi \cdot (1,5 \cdot 10^{-2})^2} = 34600\text{Pa} \quad (5.2)$$

A continuación, se puede ver dos imágenes para la pesa de 1 Kg en la figura 5.4. No se han modificado los valores de los umbrales respecto al experimento anterior. Como se puede apreciar, esta vez el plano no está inclinado. Cuando la pesa está en reposo (figura 5.4a) se mide una fuerza constante pero no se detecta deslizamiento. Al desplazarla un poco con la mano (figura 5.4b) ya se encienden los LEDs correspondientes a los táctiles que han detectado deslizamiento.

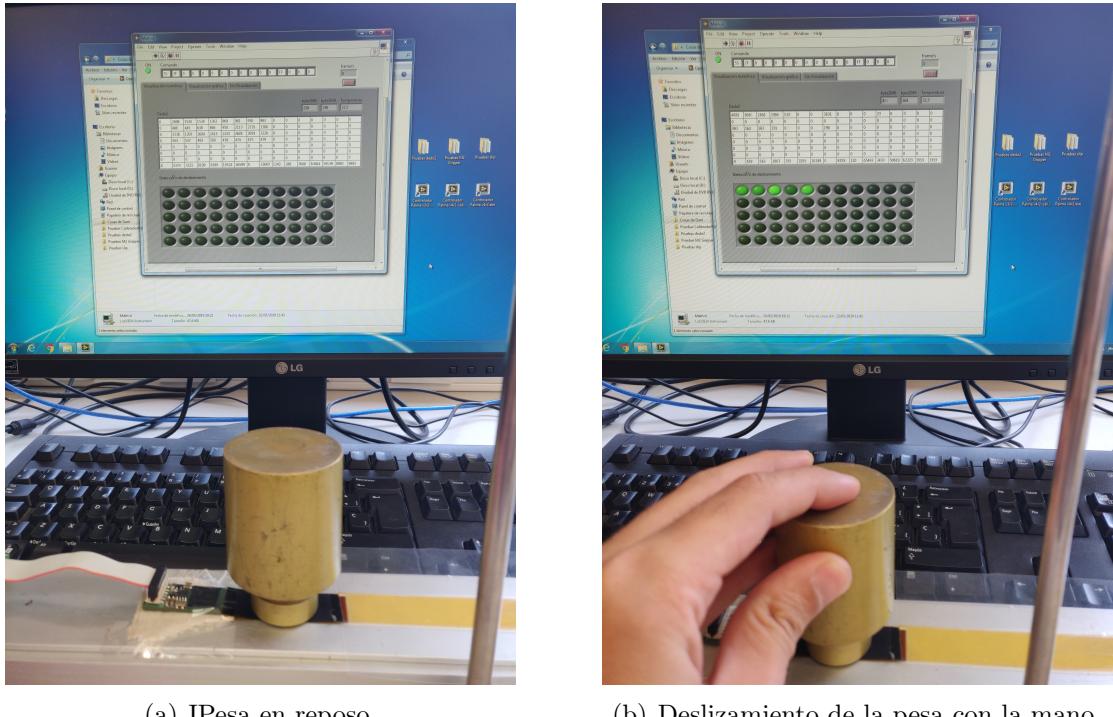


Figura 5.4: Experimentación de fuerza con la pesa de 1 Kg.

Para el caso de los 5 Kg se han tenido que ajustar los umbrales de detección al máximo de su valor. Aún así, había algunas posiciones de la pesa sobre el sensor en la que se detectaba deslizamiento sin haberlo. En la figura 5.5 se muestra la colocación de las pesas sobre el sensor totalmente paralelo al eje horizontal del espacio y se ve como la matriz de LEDs está apagada indicando que no se detecta deslizamiento.

Como se puede deducir de este último experimento, el límite de presión para este algoritmo de detección de deslizamiento es de $34\,600\text{Pa}$ aproximadamente. Es importante resaltar que este valor también depende del hardware del sensor. Si se toma como referencia las mediciones de fuerza realizadas en el artículo [12], esta presión está al límite de lo que mide el sensor de forma correcta. Sin embargo, la curva de caracterización del sensor cambia según la lámina piezorresistiva usada y la cobertura de este (no se ha usado ninguna cobertura en este experimento). Todo

esto hace necesario un análisis más profundo de la fuerza que es posible aplicar pero se sale de los objetivos marcados en este proyecto.

La única conclusión que se puede obtener de esta medición de fuerza es que el algoritmo y el hardware utilizado en este proyecto es válido para esta presión, que se considera que es lo suficientemente alta para el uso habitual de una mano robótica que no tenga que levantar pesos excesivos.



Figura 5.5: 5 Kg apoyados sobre el sensor usando la cara de menor superficie de la pesa.

5.3. Conclusiones

Después de todo el desarrollo del proyecto, es pertinente hacer una valoración final del mismo, respecto a los resultados obtenidos, las expectativas o el resultado de la experiencia acumulada.

Como se puede ver a lo largo del desarrollo del proyecto, la mayoría de los objetivos propuestos se han cumplido. A continuación, se muestra una lista que muestra todo lo que se ha conseguido en este proyecto:

- Se han realizado experimentos en el banco de pruebas consiguiendo acotar con éxito el problema a tratar. Estos datos han permitido definir la estructura del algoritmo posterior.
- Se han llevado a cabo diferentes simulaciones en MATLAB previas a la implementación de la lógica en el hardware. Esto ha agilizado en gran medida el desarrollo del algoritmo.
- Con las simulaciones mencionadas, se ha diseñado un procesador específico para la detección del deslizamiento en el sensor formado por una matriz de tácteles.
- Se ha implementado satisfactoriamente el procesador sobre el hardware existente (FPGA) usando VHDL. Realizando diferentes pruebas para corroborar el correcto funcionamiento de la lógica de control.
- Se ha conseguido detectar el deslizamiento en sus etapas más tempranas, justo cuando se está venciendo la fuerza de rozamiento estática.

Uno de los retos más difíciles de este proyecto ha sido el diseño del filtro. Esto es debido a la gran variedad de métodos de diseño y las restricciones hardware de la FPGA donde se iba a implementar. Además, como ya se comentó en el capítulo 3 el límite impuesto por el teorema del muestreo ha dificultado el análisis en frecuencia de la señal.

Por otro lado, el logro más importante de este proyecto es la detección del deslizamiento en sus etapas tempranas. Esto permitiría que una mano robótica detectara si el objeto que tiene agarrado va a deslizar, pudiendo reaccionar a tiempo y agarrarlo con más fuerza.

Esta versión del algoritmo es el primer paso para un posterior desarrollo que permita su utilización en robots funcionales. En el siguiente apartado se proponen líneas de trabajo en este sentido y en la mejora del algoritmo.

5.4. Trabajo futuro

Considerando el estado de la técnica, se pueden deducir líneas futuras de trabajo, proponer otros puntos de vista o cualquier otra sugerencia como postámbulo del presente trabajo, para ser considerada por el lector. De esta forma se sugieren los siguientes puntos de trabajo futuro:

- Lo primero sería solucionar el problema de la frecuencia de muestreo mencionada anteriormente. Sería adecuado duplicarla o incluso triplicarla para obtener un mejor resultado en frecuencia pudiendo afinar los parámetros del filtro.
- Analizar en mayor profundidad la fuerza máxima que cada táctel puede soportar para el correcto funcionamiento del detector de deslizamiento.
- En este proyecto se ha usado cada táctel por separado para el algoritmo de detección de deslizamiento. Estudiar si existe alguna relación entre las mediciones de los táctiles para diseñar un algoritmo que use una correlación entre ellos de forma que se consiga una mayor eficiencia en la detección.
- Se puede mejorar la interfaz de visualización de los datos en el ordenador para tener más control sobre la FPGA y hacerla más agradable a la vista.
- También se podría intentar aplicar esta lógica a otros sensores táctiles con un hardware diferente. Con el objetivo de hacerlo universal y fácilmente utilizable.
- Probar la solución desarrollada en este proyecto en una mano robótica funcional para comprobar que efectivamente funciona para lo que ha sido diseñado.

Cada uno de estos puntos pueden ser el comienzo de otros trabajos posteriores que contribuyan al perfeccionamiento de esta tecnología.

Parte III

Apéndices

Apéndice A

Código MATLAB

En este apéndice se muestra un script completo de MATLAB que representa un ejemplo de como se leen y analizan los datos, se diseña y se aplica el filtro y se muestran los diferentes resultados gráficamente.

Como se han usado muchos códigos diferentes a lo largo de la experimentación, el programa mostrado aquí es solo una muestra representativa de las secciones en las que se estructura, e intenta englobar todas las funciones y análisis que se han realizado en algún momento del desarrollo del proyecto.

Contents

- Obtención de datos y variables
- Diseño del filtro
- Ventanado para cuenta de pasos por cero
- Representación

```
%Creador: Daniel Rodríguez Criado
%Fecha de creación:11-09-2017
%Ultima modificación: 11-09-2017

%Este archivo analiza los datos para el sensor en vertical.
%Obtiene los datos del filtro de la FPGA y los compara con el
%filtro implementado en MATLAB con los mismos coeficientes.

clear all
close all
clc
```

Obtención de datos y variables

```
% Dimensiones de la matriz de sensores

xS = 11;
yS = 5;

muestras = 380; %Frecuencia
T= 1/muestras; %Periodo

% Zoom

tzmin=0.4; %En segundos
tzmax=0.8;

% Carga de los datos experimentales y estracción de lecturas del dedo2

filename = fullfile('..../dataV/filtroFPGA60','slip-dedo-nFiltro2-0.txt');

fileID=fopen(filename);
[data,pos]=textscan(fileID,repmat('%f',1,16), 'CollectOutput',true);
fclose(fileID);

data=data{1};
data=data(:,1:xS);
corte=33;
numMuestras = floor(length(data)/corte);
dataDedo2 = zeros(yS,xS,numMuestras);
metadatos = zeros(numMuestras,3);

for i=1:1:(numMuestras)
    dataDedo2(:,:,:,i)=data((corte*i-8):(corte*i-8)+(yS-1),:);
    metadatos(i,:)=data(corte*i,1:3);
end

t = 0:T:((numMuestras*T)-T);
```

```

zoom=find(t>tzmin & t<tzmax);
zoomt=1:length(zoom);

tz= t(zoomt);

%Convertir los datos de los t cteles, pero no los de la elipse
filtroFPGA=(dataDedo2/(2^16));
%handles.dedo2_filtrado_etapa1=dedo2_filtrado; %Dato Provisional
for i=1:size(filtroFPGA,1)
    for j=1:size(filtroFPGA,2)
        valor(1,:)=filtroFPGA(i,j,:);
        filtroFPGA(i,j,:)=typecast(uint16(valor), 'int16');
    end
end
%handles.dedo2_filtrado_etapa2=dedo2_filtrado; %Dato Provisional
filtroFPGA=(filtroFPGA/(2^13));

dataDedo2=((bitand(dataDedo2,65535))/(2^9)); %Toma los bits donde se guardan los datos de fuerza

filtroFPGAT1(1,:)=filtroFPGA(2,4,:);
filtroFPGAT2(1,:)=filtroFPGA(3,4,:);
filtroFPGAT3(1,:)=filtroFPGA(4,4,:);
filtroFPGAT4(1,:)=filtroFPGA(5,4,:);

tactel1(1,:)=dataDedo2(2,4,:);
tactel2(1,:)=dataDedo2(3,4,:);
tactel3(1,:)=dataDedo2(4,4,:);
tactel4(1,:)=dataDedo2(5,4,:);

tactz1=tactel1(zoom);
tactz2=tactel2(zoom);
tactz3=tactel3(zoom);
tactz4=tactel4(zoom);

```

Dise o del filtro

```

orden=15;
coefLP=[0.00131361611258411 0.00686111023670250 0.0209395633520919 0.0467828581528936 0.083
2780749835650 0.123107338975716 0.154504080553793 0.166468055273443 0.154504080553793 0.123
107338975716 0.0832780749835650 0.0467828581528936 0.0209395633520919 0.00686111023670250 0
.00131361611258411];
coefHP=[0.113662606758408 -0.0268684513917467 -0.142399120066621 -0.122714749037241 -0.2286
04102215777 0.819128434022424 -0.228604102215777 -0.122714749037241 -0.142399120066621 -0.0
268684513917467 0.113662606758408];

filtLP=dfilt.dffir(coefLP);
filtHP=dfilt.dffir(coefHP);
%filtLP=designfilt('lowpassfir','PassbandFrequency',30,'StopbandFrequency',50,'PassbandRipp
le',1,'StopbandAttenuation',60,'SampleRate',muestras);

tactFil1=filter(filtLP,tactel1);
tactFil2=filter(filtLP,tactel2);
tactFil3=filter(filtLP,tactel3);
tactFil4=filter(filtLP,tactel4);

tactFilH1=filter(filtHP,tactel1);
tactFilH2=filter(filtHP,tactel2);

```

```

tactFilH3=filter(filtHP,tactel3);
tactFilH4=filter(filtHP,tactel4);

tactel1r=tactel1(orden:end);
tactFillr=tactFill(50:end);

signalFilt=tactFill-tactel1;

```

Ventanado para cuenta de pasos por cero

```

ini=30;
ventana=50;
saltoV=1;

% Analisis de la señal del filtro de MATLAB

x=tactFilH1; %Señal a analizar
picos=zeros(1,length(x)-ventana);
pcero=zeros(1,length(x)-ventana);

for i=ini:saltoV:(length(x)-ventana)
    picos(i)=cpicos(x(i:i+ventana));
end

tp=t(1:length(picots));

for i=ini:saltoV:(length(x)-ventana)
    pcero(i)=ppcero(x(i:i+ventana));
end

tc=t(1:length(pcero));

% Analisis de la señal del filtro de la FPGA

x=filtroFPGAT1; %Señal a analizar
picosFPGA=zeros(1,length(x)-ventana);
pceroFPGA=zeros(1,length(x)-ventana);

for i=ini:saltoV:(length(x)-ventana)
    picosFPGA(i)=cpicos(x(i:i+ventana));
end

tpFPGA=t(1:length(picotsFPGA));

for i=ini:saltoV:(length(x)-ventana)
    pceroFPGA(i)=ppcero(x(i:i+ventana));
end

tcFPGA=t(1:length(pceroFPGA));

```

Representación

```

% Representación de cuatro tácteles y espectrograma del primero

figure(1)

subplot(2,1,1)

```

```

%plot(tz,tactz1,tz,tactz2,tz,tactz3,tz,tactz4)
plot(t,tactel1,t,tactel2,t,tactel3,t,tactel4)
title('Una sola fila de tácteles')
ylabel('Fuerza (N)');
xlabel('Tiempo (s)');
legend('Tactel 1','Tactel 2','Tactel 3','Tactel 4')
grid on

subplot(2,1,2)
spectrogram(tactel2,16,4,256,muestras,'yaxis')
%view(-77,72)
shading interp
colorbar off

% Representación primera fila de tacteles por separado

figure(2)
subplot(4,1,1)
plot(t,tactel1)
title('Un solo Táctel')
ylabel('Fuerza (N)');
xlabel('Tiempo (s)');
%xlim([1,1.5])
%axis([0,3.5,2,3])
grid on

subplot(4,1,2)
plot(t,tactel2)
ylabel('Fuerza (N)');
xlabel('Tiempo (s)');
%xlim([1,1.5])
grid on

subplot(4,1,3)
plot(t,tactel3)
ylabel('Fuerza (N)');
xlabel('Tiempo (s)');
%xlim([1,1.5])
grid on

subplot(4,1,4)
plot(t,tactel4)
ylabel('Fuerza (N)');
xlabel('Tiempo (s)');
%xlim([1,1.5])
grid on

%Representación evolución de todos los tácteles

figure(3)
title('Todos los tátteles')
hold on
for i=1:1:11
    tactelp1(:,1)=dataDedo2(2,i,:);
    tactelp2(:,1)=dataDedo2(3,i,:);
    tactelp3(:,1)=dataDedo2(4,i,:);
    tactelp4(:,1)=dataDedo2(5,i,:);
    plot(t,tactelp1,t,tactelp2,t,tactelp3,t,tactelp4)
end
hold off
ylabel('Fuerza (N)');

```

```

xlabel('Tiempo (s)');

%Representación filtros

figure(4)

subplot(5,1,1)
plot(t,tactel1)
title('Tactel')
ylabel('Fuerza (N)');
xlabel('Tiempo (s)');
%xlim([0,1.5])
grid on

% subplot(4,1,2)
% plot(t,signalFilt)
% title('Filtrado por resta de LP')
% ylabel('Amplitud');
% xlabel('Tiempo (s)');
% xlim([0,1.5])

subplot(5,1,2)
plot(t,tactFilH1)
title('Filtrado por HP')
ylabel('Amplitud');
xlabel('Tiempo (s)');
%xlim([0,1.5])
grid on

subplot(5,1,3)
plot(tp,picos)
title('Numero de picos a lo largo del tiempo')
ylabel('Numero de picos');
xlabel('Tiempo (s)');
%xlim([0,1.5])
grid on

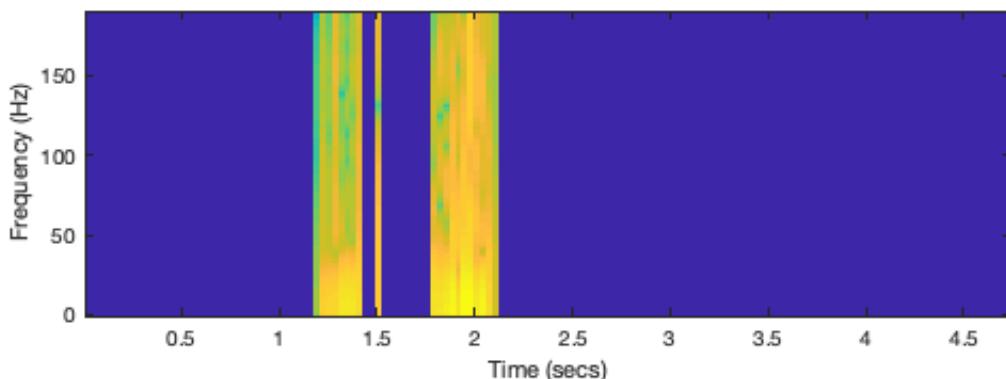
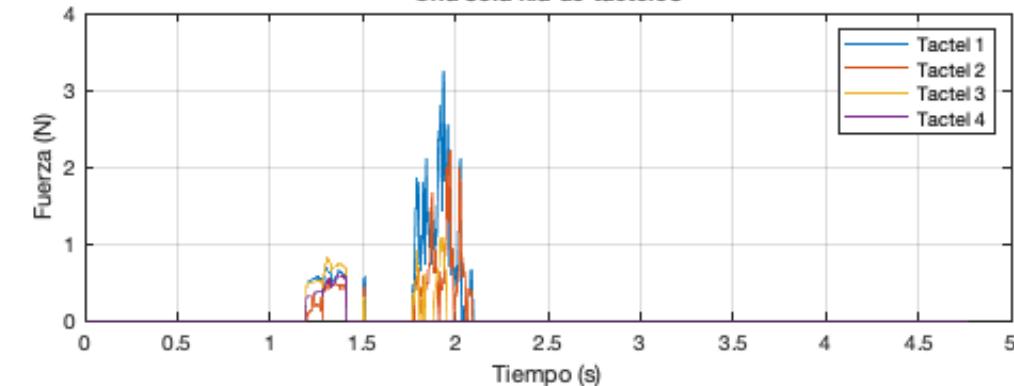
% subplot(4,1,4)
% plot(tc,pcero)
% title('Numero de pasos por cero a lo largo del tiempo')
% ylabel('Numero de pasos por cero');
% xlabel('Tiempo (s)');
% xlim([0,1.5])

subplot(5,1,4)
plot(t,filtroFPGAT1)
title('Respuesta filtro FPGA')
ylabel('Amplitud');
xlabel('Tiempo (s)');
%xlim([0,1.5])
grid on

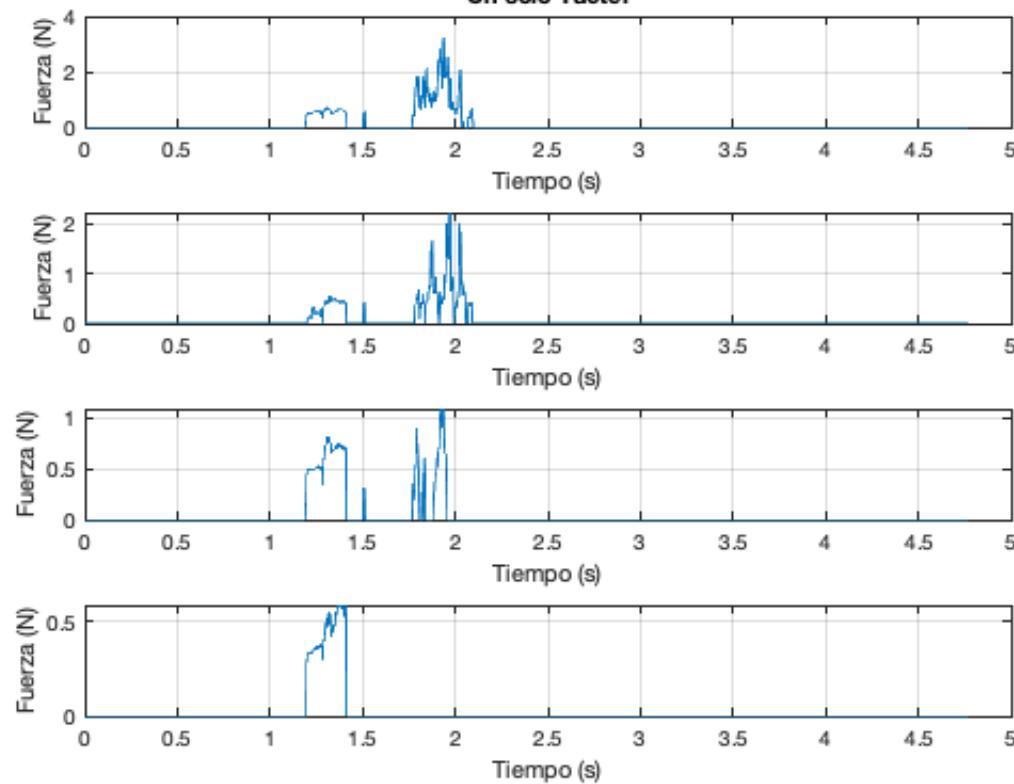
subplot(5,1,5)
plot(tpFPGA,picosFPGA)
title('Conteo de picos filtro FPGA')
ylabel('Picos y valles');
xlabel('Tiempo (s)');
%xlim([0,1.5])
grid on

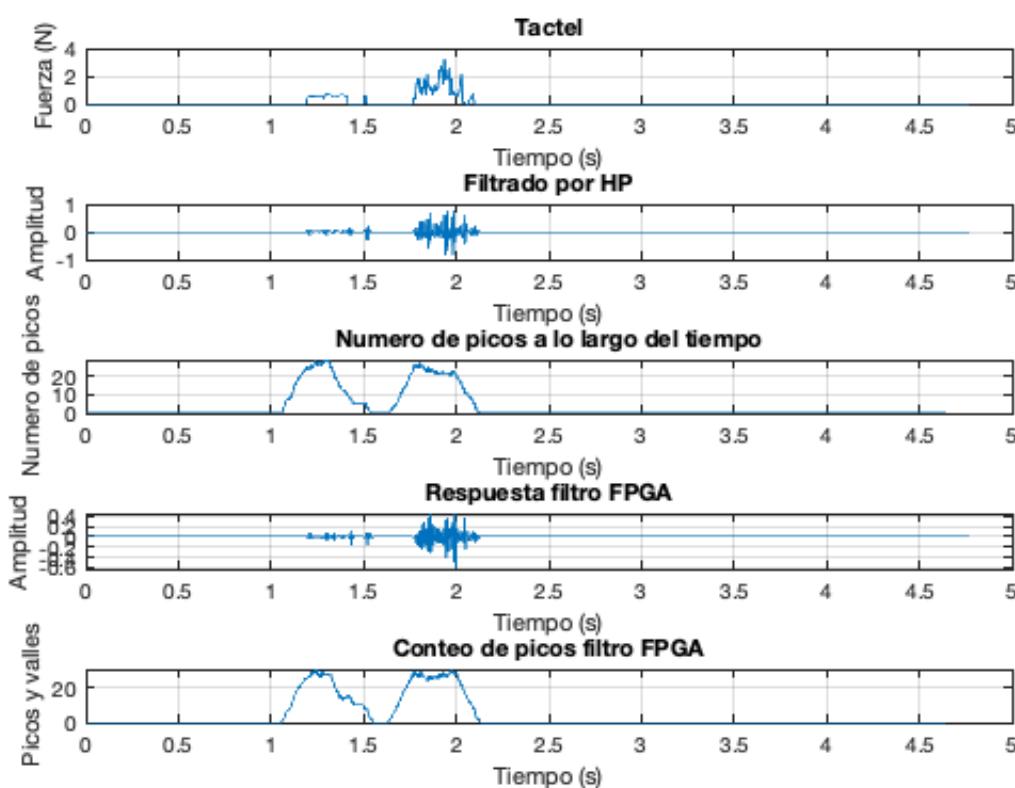
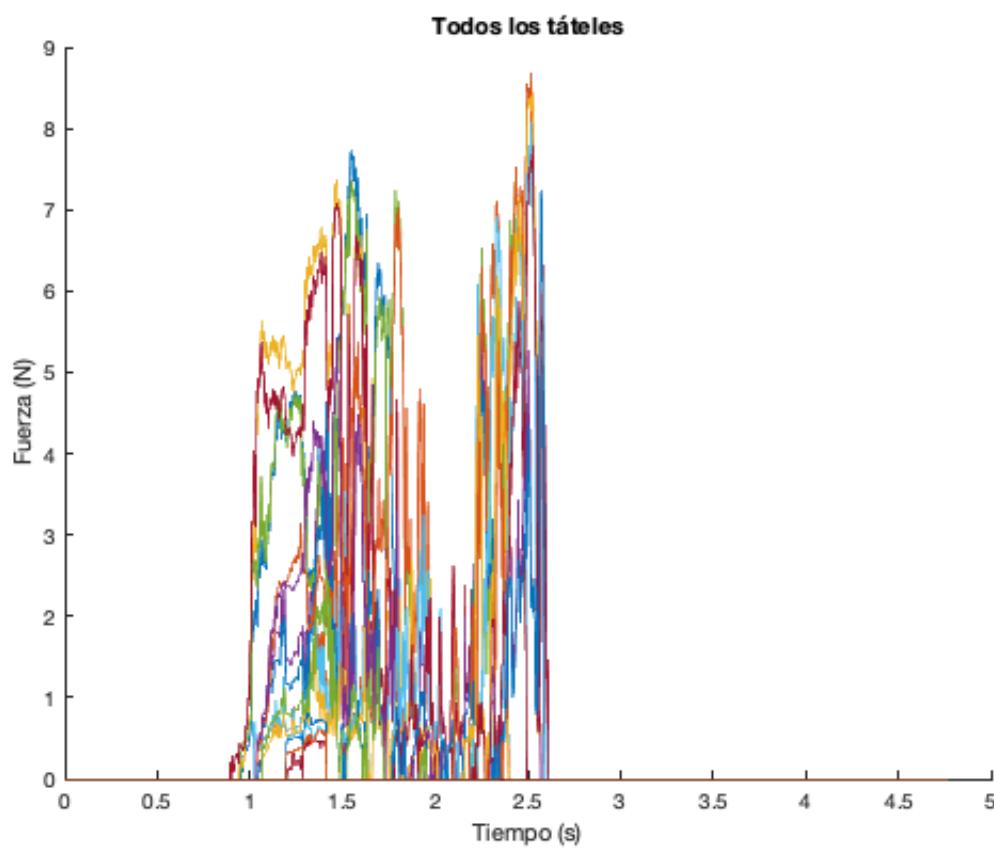
```

Una sola fila de táctiles



Un solo Táctel





Apéndice B

Código VHDL

Este apéndice muestra todas las modificaciones que se le han hecho al programa original que estaba implementado en la FPGA. Todas las capturas muestran el número de línea en las que se ha introducido el código.

Como solo se han hecho cambios en dos de los módulos del programa en VHDL, la explicación de estas modificaciones se separaran en dos secciones. Todas las imágenes están brevemente explicadas para indicar en que consiste cada cambio introducido.

B.1. Modificaciones en el módulo sensor

En primer lugar se han añadido los dos umbrales como puertos de entrada al módulo sensor que es donde se utilizan para realizar la comparación. Estos puertos son "umbral_positivo" y "umbral_negativo".

```

30 entity sensor_module is
31   port (
32     clk           : in std_logic;                      -- Reloj 50MHz
33     fil0          : inout std_logic_vector(10 downto 0); -- Fila 0 matriz sensores
34     fil1          : inout std_logic_vector(10 downto 0); -- Fila 1 matriz sensores
35     fil2          : inout std_logic_vector(10 downto 0); -- Fila 2 matriz sensores
36     fil3          : inout std_logic_vector(10 downto 0); -- Fila 3 matriz sensores
37     fil4          : inout std_logic_vector(10 downto 0); -- Fila 4 matriz sensores
38     fil_calib0   : inout std_logic_vector(10 downto 0); -- Fila 0 resistencias calibracion
39     fil_calib1   : inout std_logic_vector(10 downto 0); -- Fila 1 resistencias calibracion
40     col           : inout std_logic_vector(10 downto 0); -- Columnas matriz sensores
41     we_image      : out std_logic;
42     addr_image   : out std_logic_vector(7 downto 0);
43     din_image    : out std_logic_vector(31 downto 0);
44     row_write_done: out std_logic;
45     current_frame: out std_logic;
46     last_row     : out std_logic_vector(2 downto 0);
47     ram_reading  : in std_logic;
48     last_frame   : out std_logic;
49     umbral_positivo: in std_logic_vector(7 downto 0);
50     umbral_negativo: in std_logic_vector(7 downto 0);
51     frame_counter: out std_logic_vector(7 downto 0)
52   );
53 end sensor_module;
54 
```

Figura B.1: Puertos de entrada y salida del módulo sensor.

De la línea 149 a la línea 153 se muestran las señales que ha sido necesario crear para implementar la nueva lógica. La línea 170 crea el multiplexor que selecciona los datos que se van a escribir en la RAM para posteriormente ser enviado al ordenador por el puerto serie USB.

```

147 -- signal detecta_slip : std_logic; --Guarda si se ha detectado algún deslizamiento
148 -- signal slip_detectado_fila : std_logic; --Guarda si en la fila correspondiente ha habido un deslizamiento
149 signal slip_detectado_columna : std_logic_vector(10 downto 0); --Guarda si en cada tactel de la fila ha habido algún deslizamiento
150 signal slip_detectado_columna_reg :std_logic_vector(11 downto 0):= (others => '0');
151 signal umbral_positivo_amp : signed(15 downto 0); --Se amplia el umbral positivo y el negativo para hacer los cálculos
152 signal umbral_negativo_amp : signed(15 downto 0);
153 signal din_image_dat : std_logic_vector(31 downto 0);
154
155 begin
156
157 -- ;;; Estudiar los efectos de una posible inanición !!!! 
158
159 -----
160 -- Señales de escritura de imágenes en RAM -----
161 -----
162 sensor_writing <= '1' when sm2_state = sm2_st3_row_store else '0';
163 we_image <= sensor_writing;
164 addr_image <= int_current_frame & row_counter & column_counter;
165 op <= rg_med_columna(conv_integer(column_counter)) & rg_low_columna(conv_integer(column_counter));
166 resta <= x"FFFF" - op;
167 din_image <= rg_med_columna(conv_integer(column_counter)) & rg_low_columna(conv_integer(column_counter)) & resta;
168 din_image <= force_data(conv_integer(column_counter)) & resta;
169 din_image <= resta & force_data(conv_integer(column_counter));
170 din_image <= x"0000" & slip_detectado_columna_reg when (sm2_state = sm2_st3_row_store and column_counter = "1011") else din_image_dat;
171 din_image <= x"0000" & umbral_positivo & umbral_positivo when (sm2_state = sm2_st3_row_store and column_counter = "1011") else din_image_dat;
172 din_image_dat <= filter_data (conv_integer(column_counter)) & force_data(conv_integer(column_counter)); 
```

Figura B.2: Nuevas señales creadas y multiplexor.

En la siguiente imagen se muestra como se han modificado los contadores de filas y columnas para que la cuenta llegue a una unidad más. Esto se ve en las líneas 499 y 515 del código.

```

493  -----
494  -- Contador de Columnas
495  -----
496 process (clk)
497 begin
498   if clk'event and clk = '1' then
499     if sm2_state = sm2_st3_row_store and column_counter= "1011" then
500       column_counter <= (others => '0');
501     elsif sm2_state = sm2_st3_row_store then
502       column_counter <= column_counter + 1;
503     end if;
504   end if;
505 end process;
506
507  -----
508  -- Contador de Filas
509  -----
510 process (clk)
511 begin
512   if clk'event and clk = '1' then
513     if sm2_state = sm2_st1_idle and row_counter = "111" then
514       row_counter <= (others => '0');
515     elsif sm2_state = sm2_st3_row_store and column_counter = "1011" then
516       last_row <= row_counter;
517       row_counter <= row_counter + 1;
518     end if;
519   end if;
520 end process;

```

Figura B.3: Modificaciones en el contador de columna y contador de fila.

En la siguiente figura se muestra la máquina de estados que realiza la lectura de los datos de fuerza y los datos del filtro e indica cuando estos datos están listos para ser escritos en la RAM. La única modificación ha sido incrementar en una unidad el valor al que tiene que llegar el contador de columnas. En las lineas 560 y 561 se puede ver el código original comentado y el nuevo.

```

536  -----
537  -- Proximos estados de la máquina de estados SM2
538  -----
539 SM2_NEXT_STATE_DECODE: process (sm2_state, sml_state, force_obtained, filter_rfd, filter_rdy, row_counter, column_counter)
540 begin
541   next_sm2_state <= sm2_state;
542   case (sm2_state) is
543     when sm2_st1_idle =>
544       if (sml_state = sml_st4_store) then
545         next_sm2_state <= sm2_st2_obtain_force;
546       end if;
547     when sm2_st2_obtain_force =>
548       if force_obtained = "1111111111" then
549         if row_counter < "101" and filter_rfd = "1111111111" then
550           next_sm2_state <= sm2_st2bis_obtain_filter;
551         else
552           next_sm2_state <= sm2_st3_row_store;
553         end if;
554       end if;
555     when sm2_st2bis_obtain_filter =>
556       if filter_rdy = "1111111111" then
557         next_sm2_state <= sm2_st3_row_store;
558       end if;
559     when sm2_st3_row_store =>
560       --
561       if column_counter = "1010" then
562         if column_counter = "1011" then
563           next_sm2_state <= sm2_st1_idle;
564         end if;
565       end case;
566   end process;

```

Figura B.4: Modificación en la máquina de estados que hace la lectura y escritura de datos.

Por último, en la siguiente imagen se muestran todos los comparadores, uno por cada filtro. Si se supera alguno de los umbrales, se guarda un '1' en la posición correspondiente de la señal "slip_detectado_columna". En las líneas 738 y 739 también se definen la parte alta de los umbrales como su valor máximo (en complemento a 2).

```

737 --Se amplian los umbrales para que coincidan en tamaño con filter_data
738 umbral_positivo_amp <= signed(std_logic_vector'("00000000" & umbral_positivo));
739 umbral_negativo_amp <= signed(std_logic_vector'("11111111" & umbral_negativo));
740
741
742 filter_data(0) <= filter_tmp0(27 downto 12);
743 slip_detectado_columna(0) <= '1' when (signed(filter_data(0)) < umbral_negativo_amp or signed(filter_data(0)) > umbral_positivo_amp) else '0';
744 filter_data(1) <= filter_tmp1(27 downto 12);
745 slip_detectado_columna(1) <= '1' when (signed(filter_data(1)) < umbral_negativo_amp or signed(filter_data(1)) > umbral_positivo_amp) else '0';
746 filter_data(2) <= filter_tmp2(27 downto 12);
747 slip_detectado_columna(2) <= '1' when (signed(filter_data(2)) < umbral_negativo_amp or signed(filter_data(2)) > umbral_positivo_amp) else '0';
748 filter_data(3) <= filter_tmp3(27 downto 12);
749 slip_detectado_columna(3) <= '1' when (signed(filter_data(3)) < umbral_negativo_amp or signed(filter_data(3)) > umbral_positivo_amp) else '0';
750 filter_data(4) <= filter_tmp4(27 downto 12);
751 slip_detectado_columna(4) <= '1' when (signed(filter_data(4)) < umbral_negativo_amp or signed(filter_data(4)) > umbral_positivo_amp) else '0';
752 filter_data(5) <= filter_tmp5(27 downto 12);
753 slip_detectado_columna(5) <= '1' when (signed(filter_data(5)) < umbral_negativo_amp or signed(filter_data(5)) > umbral_positivo_amp) else '0';
754 filter_data(6) <= filter_tmp6(27 downto 12);
755 slip_detectado_columna(6) <= '1' when (signed(filter_data(6)) < umbral_negativo_amp or signed(filter_data(6)) > umbral_positivo_amp) else '0';
756 filter_data(7) <= filter_tmp7(27 downto 12);
757 slip_detectado_columna(7) <= '1' when (signed(filter_data(7)) < umbral_negativo_amp or signed(filter_data(7)) > umbral_positivo_amp) else '0';
758 filter_data(8) <= filter_tmp8(27 downto 12);
759 slip_detectado_columna(8) <= '1' when (signed(filter_data(8)) < umbral_negativo_amp or signed(filter_data(8)) > umbral_positivo_amp) else '0';
760 filter_data(9) <= filter_tmp9(27 downto 12);
761 slip_detectado_columna(9) <= '1' when (signed(filter_data(9)) < umbral_negativo_amp or signed(filter_data(9)) > umbral_positivo_amp) else '0';
762 filter_data(10) <= filter_tmp10(27 downto 12);
763 slip_detectado_columna(10) <= '1' when (signed(filter_data(10)) < umbral_negativo_amp or signed(filter_data(10)) > umbral_positivo_amp) else

```

Figura B.5: Comparadores con los umbrales y la salida del filtro y definición de la parte alta de los umbrales.

B.2. Modificaciones en el módulo TOP

En las siguientes dos imágenes se muestran las conexiones entre el módulos sensor y el módulo de comunicaciones SPI. Estas conexiones permiten enviar la parte baja de los umbrales desde la aplicación de LabVIEW en el ordenador.

```

217  U1 : sensor_module
218    port map(
219      clk          => clk,
220      fil0         => fil0,
221      fill         => fill,
222      fil2         => fil2,
223      fil3         => fil3,
224      fil4         => fil4,
225      fil_calib0   => fil_calib0,
226      fil_calib1   => fil_calib1,
227      col          => col,
228      we_image     => we_image,
229      addr_image   => addr_image,
230      din_image    => din_image,
231      row_write_done => row_write_done,
232      current_frame  => current_frame,
233      last_row       => last_row,
234      ram_reading   => ram_reading,
235      last_frame     => last_frame,
236      umbral_positivo => umbral_positivo,
237      umbral_negativo => umbral_negativo,
238      frame_counter  => frame_counter
239    );
240
258  U3 : spi_module
259    port map(
260      clk          => clk,
261      ss           => ss,
262      sclk         => sclk,
263      mosi         => mosi,
264      miso         => miso,
265      clk_temp_register  => clk_temp_register,
266      ram_reading   => ram_reading,
267      last_frame     => last_frame,
268      addrb        => addrb,
269      doutb        => doutb,
270      clk_frame_counter  => frame_counter,
271      clk_operacion  => reg_operacion,
272      clk_tactel_inicial  => umbral_positivo,
273      clk_tactel_final   => umbral_negativo
274    );
275  reg_tactel_inicial <= umbral_positivo;
276  reg_tactel_final   <= umbral_negativo;
277
278  clk_temp_register <= x"CAFE";
279
280

```

(a) Conexiones módulo sensor.

(b) Conexiones módulo SPI.

Figura B.6: Conexiones entre el módulo sensor y el módulo SPI.

Bibliografía

- [1] Xilinx tutorials and how to use ise design suite. <https://www.xilinx.com/support.html#knowledgebase>. Accessed: 2017-03-22.
- [2] Purves D, Augustine GJ, and Fitzpatrick D. *Neuroscience. 2nd edition.* Sunderland (MA): Sinauer Associates, 2001.
- [3] R. S. Dahiya, G. Metta, M. Valle, and G. Sandini. Tactile sensing 2014;from humans to humanoids. *IEEE Transactions on Robotics*, 26(1):1–20, Feb 2010.
- [4] Escuela Universitaria de Música EUM. Filtros senoc-enventanado y filtros personalizados. clase 11. <http://www.eumus.edu.uy/eme/ensenanza/electivas/dsp/presentaciones/clase11.pdf>. Accessed: 2019-03-22.
- [5] ESI Dpto. de Electrónica y Electromagnetismo. Conceptos fundamentales en el diseño del filtros. http://www2.imse-cnm.csic.es/elec_esi/asignat/ASC/pdf/tema2.pdf. 2004/05.
- [6] Gaganpreet Kaur. *VHDL: Basics to Programming.* Pearson, 2011.
- [7] Ricardo A. Losada. Practical fir filter design in matlab. *The MathWorks, Inc.*, page 32, January 12, 2004.
- [8] Richard G. Lyons. *Understanding Digital Signal Processing.* Prentice Hall, 2010.
- [9] Rocío Maldonado-López, Fernando Vidal-Verdú, Gustavo Liñán, Elisenda Ro- ca, and Ángel Rodríguez-Vázquez. Early slip detection with a tactile sensor based on retina. *Analog Integrated Circuits and Signal Processing*, 53(2):97–108, 2007.
- [10] miniDSP. Fir vs iir filtering. <https://www.minidsp.com/applications/dsp-basics/fir-vs-iir-filtering>. Accessed: 2019-03-05.

- [11] Óscar Oballe Peinado. Aportaciones al hardware para sensores táctiles inteligentes basados en fpgas.
- [12] Óscar Oballe-Peinado, José Antonio Hidalgo-López, Julián Castellanos-Ramos, José Antonio Sánchez-Durán, Rafael Navas-González, Jaime Herrán, and Fernando Vidal-Verdú. Fpga-based tactile sensor suite electronics for real-time embedded processing. *IEEE Transactions on Industrial Electronics*, 64(12):9657–9665, 2017.
- [13] Reserach Center “E. Piaggio” University of Pisa. Cutaneous mechanoreceptors. http://www.centropiaggio.unipi.it/sites/default/files/course/material/Cutaneous_mechanoreceptors.pdf. Accessed: 2018-03-22.
- [14] Z. Su, K. Hausman, Y. Chebotar, A. Molchanov, G. E. Loeb, G. S. Sukhatme, and S. Schaal. Force estimation and slip detection/classification for grip control using a biomimetic tactile sensor. pages 297–303, Nov 2015.
- [15] Daniel Mayor Tomillo. Diseño de filtros digitales fir mediante técnicas de computación evolutiva y estudio de su aplicación al procesado de señales biomédicas. Master’s thesis, Universidad de Valladolid, Octubre de 2016.
- [16] Fernando Vidal-Verdú, Óscar Oballe-Peinado, José A. Sánchez-Durán, Julián Castellanos-Ramos, and Rafael Navas-González. Three realizations and comparison of hardware for piezoresistive tactile sensors. *Sensors*, 11(3):3249–3266, 2011.
- [17] Wikipedia. Diseño de filtros de respuesta finita al impulso. https://es.wikipedia.org/wiki/Dise%F1o_de_Filtros_de_Respuesta_Finita_al_Impulso. Accessed: 2019-03-05.
- [18] Wikipedia. Finite impulse response. https://en.wikipedia.org/wiki/Finite_impulse_response. Accessed: 2019-03-05.
- [19] Wikipedia. Infinite impulse response. https://en.wikipedia.org/wiki/Infinite_impulse_response. Accessed: 2019-03-05.
- [20] Wikipedia. Teorema de muestreo de nyquist-shannon. https://es.wikipedia.org/wiki/Teorema_de_muestreo_de_Nyquist-Shannon. Accessed: 2018-03-22.
- [21] Steve Winder. *Analog and Digital Filter Design (EDN Series for Design Engineers)*. Newnes, 2002.