Universidad Católica de Santa María

Facultad de Ciencias e Ingenierías Físicas y Formales

Escuela Profesional de Ingeniería Electrónica

Diseño e implementación de un prototipo de interfaz humano-computador para personas con movilidad reducida utilizando bioseñales, procesamiento digital de señales con DSPIC y el protocolo USB para que puedan acceder al mundo digital

Tesis presentada por el Bachiller:

Carreño Arone, Kevin Gonzalo

ORCID: 0009-0006-0965-6016

para optar el Título Profesional de Ingeniero Electrónico

Asesor:

Mg. Málaga Chávez, César Eduardo

ORCID: 0000-0002-5461-5380

Arequipa - Perú

2025

UCSM-ERP

UNIVERSIDAD CATÓLICA DE SANTA MARÍA

INGENIERIA ELECTRONICA

TITULACIÓN CON TESIS

DICTAMEN APROBACIÓN DE BORRADOR

Arequipa, 22 de Marzo del 2025

Dictamen: 009810-C-EPIE-2025

Visto el borrador del expediente 009810, presentado por:

2017150051 - CARREÑO ARONE KEVIN GONZALO

Titulado:

DISEÑO E IMPLEMENTACIÓN DE UN PROTOTIPO DE INTERFAZ HUMANO-COMPUTADOR PARA PERSONAS CON MOVILIDAD REDUCIDA UTILIZANDO BIOSEÑALES, PROCESAMIENTO DIGITAL DE SEÑALES CON DSPIC Y EL PROTOCOLO USB PARA QUE PUEDAN ACCEDER AL MUNDO DIGITAL

Nuestro dictamen es:

APROBADO

Titulo Profesional/Titulo de Segunda Especialidad/Grado Académico a optar:

INGENIERO ELECTRONICO

29267682 - RODRIGUEZ GONZALES PEDRO ALEX
DICTAMINADOR

29715414 - COAGUILA GOMEZ RONALD PERCING
DICTAMINADOR

29410027 - SULLA TORRES RAUL RICARDO
DICTAMINADOR

Diseño e implementación de un prototipo de interfaz humano-computador para personas con movilidad reducida utilizando bioseñales, procesamiento digital de señales con DSPIC y el protocolo USB para que

INFORME DE ORIGINALIDAD

11%	11%	2%	4%
INDICE DE SIMILITUD	FUENTES DE INTERNET	PUBLICACIONES	TRABAJOS DEL ESTUDIANTE

FUENTES PRIMARIAS

1	idoc.pub Fuente de Internet	1%
2	www.oftalmologia-online.es Fuente de Internet	1%
3	www.coursehero.com Fuente de Internet	<1%
4	mil.ufl.edu Fuente de Internet	<1%
5	hdl.handle.net Fuente de Internet	<1%
6	www.microchip.com Fuente de Internet	<1%
7	ru.djvu.online Fuente de Internet	<1%
8	Submitted to Universidad Católica de Santa María Trabajo del estudiante	<1%
9	github.com Fuente de Internet	<1%
10	dokumen.pub Fuente de Internet	<1%
11	es.scribd.com Fuente de Internet	<1%
12	repositorio.ucsm.edu.pe Fuente de Internet	<1%
13	doku.pub Fuente de Internet	<1%
	www.minsa.gob.pe	

DEDICATORIA

A Dios, por ser mi guía y fortaleza en cada paso de este camino.

A mis padres, por su amor incondicional, sacrificios y apoyo constante, quienes siempre han creído en mí y me han enseñado el valor del esfuerzo.

A quien, con su amor y paciencia, me ha dado la fuerza para superar cada obstáculo durante este proceso, y que me alentó a seguir avanzando.

Con todo mi corazón, a ustedes les dedico este logro.

RESUMEN

Esta tesis aborda la necesidad de inclusión digital para personas con movilidad reducida, destacando cómo la pandemia del COVID-19 ha llevado a que muchos servicios ahora se ofrezcan de forma digital, permitiendo el acceso desde el hogar. Esto representa una oportunidad significativa para la inclusión de estas personas en el mundo digital. La tesis se enfoca en el diseño e implementación de un prototipo de interfaz humano-computador utilizando bioseñales, procesamiento digital de señales con DSPIC y el protocolo USB, permitiendo a estas personas controlar un mouse de computadora. La propuesta busca no solo mejorar la calidad de vida y la independencia física y emocional de los usuarios, además, pretende fomentar su inclusión social y digital.

El estudio se centra en la adaptación del mouse, considerado el periférico más crucial para la interacción digital, utilizando señales eléctricas generadas por el movimiento ocular. Los objetivos específicos incluyen el estudio del protocolo USB en su configuración HID, la configuración del PIC18F2550 para controlar su módulo USB y la utilización del dsPIC33CH128MP506 para el procesamiento y caracterización de las bioseñales.

Palabras clave: Procesamiento de bioseñales, interfaz humano-computador y movilidad reducida.

ABSTRACT

This thesis addresses the need for digital inclusion for people with reduced mobility, highlighting how the COVID-19 pandemic has led to many services now being offered digitally, allowing access from home. This represents a significant opportunity for the inclusion of these people in the digital world. The thesis focuses on the design and implementation of a human-computer interface prototype using biosignals, digital signal processing with DSPIC and the USB protocol, allowing these people to control a computer mouse. The proposal seeks not only to improve the quality of life and the physical and emotional independence of users, but also to promote their social and digital inclusion.

The study focuses on the adaptation of the mouse, considered the most crucial peripheral for digital interaction, using electrical signals generated by eye movement. Specific objectives include the study of the USB protocol in its HID configuration, the configuration of the PIC18F2550 to control its USB module and the use of the dsPIC33CH128MP506 for the processing and characterization of biosignals.

Key words: Biosignal processing, human-computer interface, and reduced mobility.

ÍNDICE

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

ÍNDICE DE ANEXOS

INTRODUCCIÓN

En la era de la revolución digital, la conectividad y el acceso a plataformas digitales se han vuelto esenciales para la vida diaria. Sin embargo, las personas con movilidad reducida enfrentan una barrera significativa en el acceso y uso de dispositivos móviles y computadoras, lo que limita su inclusión en la sociedad digital. La pandemia de la COVID - 19 ha acelerado la digitalización de varios servicios, permitiendo acceder a ellos desde la comodidad del hogar a través de dispositivos multimedia, haciendo más necesario crear soluciones que permitan a estas personas acceder a la información, servicios y oportunidades disponibles desde el mundo digital. (Katz, 2022)

El problema radica en la incapacidad de estas personas para interactuar físicamente con los periféricos convencionales, como el mouse, lo cual las excluye de la revolución tecnológica actual. Al adaptar este periférico crítico mediante la interpretación de bioseñales, es posible devolverles la independencia y la capacidad de participar plenamente en la era digital.

Esta tesis se propone diseñar e implementar un prototipo de Interfaz Humano-Computador para personas con movilidad reducida, utilizando bioseñales y procesamiento digital de señales con DSPIC, así como el protocolo USB con el objetivo de permitir que estas personas controlen un mouse de computadora a través del movimiento de sus ojos, facilitando su interacción con el mundo digital y mejorando su calidad de vida.

La estructura de esta tesis incluye el estudio del protocolo USB en su configuración HID, la configuración del PIC18F2550 y del dsPIC33CH128MP506, y el desarrollo de un sistema que utilice bioseñales eléctricos del movimiento ocular para controlar el puntero del mouse.

La hipótesis central es que, al ser el mouse el periférico con el cual más se interactúa para utilizar los medios digitales en la mayoría de plataformas, el lograr adaptar el mouse USB para ser controlado mediante bioseñales, puede facilitar la integración digital de personas con movilidad reducida, permitiéndoles interactuar con diversas plataformas tecnológicas y mejorar su calidad de vida.

CAPÍTULO I

1. PLANTEAMIENTO TEÓRICO

1.1. Identificación del Problema

La conectividad es hoy más importante que nunca y, como resultado de la pandemia de la COVID-19, el mundo depende como nunca de las plataformas digitales para acceder a información, servicios y oportunidades esenciales. Sin embargo, la brecha en el acceso y el uso de dispositivos móviles para las personas con discapacidad impide la inclusión. (GSM Association, 2020)

Las personas que enfrentan limitaciones para utilizar dispositivos móviles o computadoras, debido a alguna discapacidad, se ven excluidas de la revolución digital que estamos experimentando en la actualidad. Con esta tesis, buscamos contribuir a la reducción de esta brecha digital, facilitando que aquellos que se hayan visto obligados a abandonar sus estudios o empleos debido a enfermedades o accidentes, puedan retomar estas actividades de manera remota de manera proporcional a sus capacidades, además que puedan continuar con sus terapias de rehabilitación.

El propósito central de este estudio es devolver independencia y mejorar la calidad de vida, tanto a nivel físico como emocional. Al brindar la posibilidad de participar en actividades a distancia, se busca que estas personas se sientan nuevamente útiles, contrarrestando la depresión derivada de la inactividad. Además, se busca fomentar una mayor inclusión en la sociedad al permitir su participación en la era digital en igualdad de condiciones.

1.2. Descripción del Problema

En la actualidad, estamos viviendo una revolución tecnológica tanto a nivel de software como de hardware. El uso de Internet en nuestros dispositivos nos permite aprender e interactuar de diferentes formas y desde cualquier parte del mundo. Sin embargo, más

importante que tener estos dispositivos a nuestra disposición, es la forma en la que realizamos esas interacciones. Por ejemplo, en una computadora interactuamos a través de ciertos periféricos usando nuestro cuerpo (ojos, dedos, voz y oídos), en un celular usamos directamente nuestros dedos sobre la pantalla, mientras en un asistente virtual usamos únicamente comandos de voz. Pero ¿qué sucedería si alguna parte de nuestro cuerpo dejara de funcionar y nos impidiera interactuar físicamente con el periférico que es la puerta a todas estas maravillas tecnológicas? En este caso, quedaríamos parcial o totalmente excluidos del desarrollo tecnológico actual (sin restar importancia a las dificultades físicas y emocionales que conlleva quedar discapacitado). Si bien es cierto que existen asistentes virtuales para personas invidentes o con algún defecto visual, sordas (auto titulado de videos), todavía queda mucho por hacer para mejorar la accesibilidad para aquellos que tienen discapacidades físicas que les impiden interactuar con los periféricos de manera convencional.

Esta Tesis se enfoca en resolver los problemas a los que se enfrentan aquellas personas que se ven imposibilitadas de interactuar con, lo que consideramos, el periférico más importante a la hora de usar una computadora, el mouse. El mouse (junto al protocolo USB) es el principal periférico que nos permite interactuar con el mundo digital, por lo que, si encontramos la forma de adecuar este periférico para este grupo de personas de movilidad reducida, de tal forma que puedan volver a utilizarlo; podríamos ayudarlos a integrarse al mundo digital a través de una computadora según sus necesidades y posibilidades.

1.3. Justificación

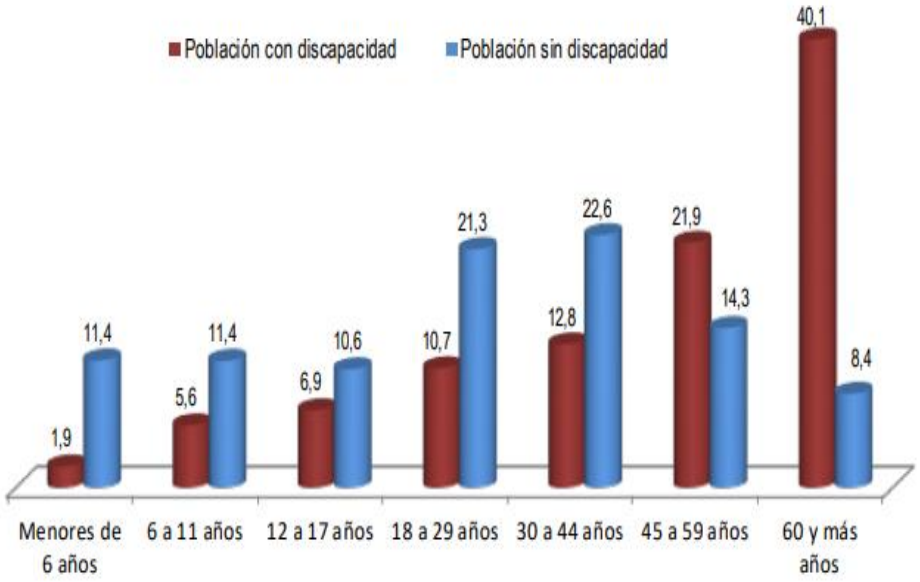
Según el artículo de la INEI actualizado el 2022, en nuestro país el 5,2% de la población (1 millón 575 mil personas) presenta algún tipo de discapacidad, además, indica que la discapacidad más frecuente son las que limitan el caminar, usar brazos y piernas, los cuales representan 59.2% (932 mil personas), este tipo de discapacidad se presenta en

personas de edad avanzada (32,5%), con enfermedad crónica (24,4%), genético/congénito o de nacimiento (9,8%), enfermedad común (6,8%), accidente común fuera del hogar (4,9%), accidente común en el hogar (4,6%),accidente de tránsito (4,0%), entre otros. (INEI, 2022)

La mayor incidencia de discapacidad se observó en los departamentos de Lima donde el 6,8% de su población presenta alguna discapacidad, seguido por Arequipa (6,7%) y Moquegua (6,6%). En cambio, los departamentos de Loreto (3,2%), Amazonas (3,3%) y Cusco, Junín y Lambayeque con 3,5% cada uno, registraron las menores tasas. (INEI, 2022).

Figura 1

Distribución porcentual de la población con y sin discapacidad



Nota: El grafico se agrupa según edades. Tomado de: INEI-Censos Nacionales 2017

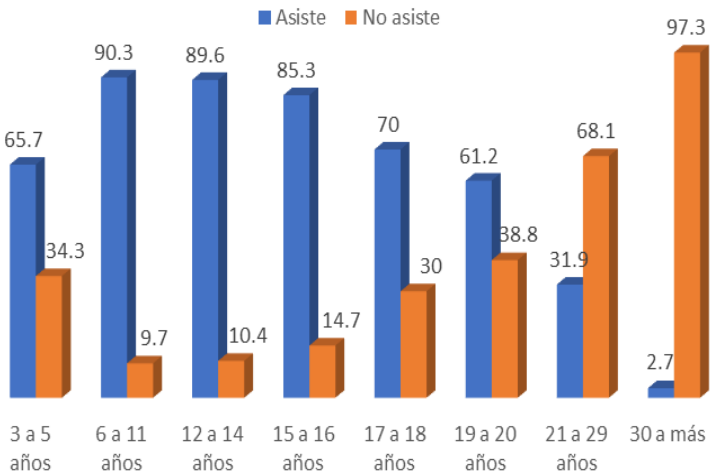
Este tipo de discapacidad física conlleva diferente tipo de problemas, alguno de ellos son perdida de estudio, trabajo y depresión por inactividad. Según se observa en las estadísticas de la INEI, existe un grupo de personas que no estudian en las edades que corresponden a la educación básica y superior, uno de los motivos para esto son las dificultades para acceder a estos servicios debido.

Las personas con discapacidades físicas pueden sentirse excluidas de la revolución digital actual debido a su incapacidad para interactuar con la tecnología. Aunque el trabajo

y el estudio remotos se han vuelto cada vez más comunes, estas personas a menudo no tienen acceso a estos recursos debido a la falta de adaptaciones para sus necesidades específicas. Para garantizar una inclusión digital completa, es fundamental encontrar soluciones que permitan a estas personas interactuar con la tecnología y aprovechar las oportunidades que ofrece la era digital.

Figura 2

Población con discapacidad que asiste o no, a algún colegio, instituto o universidad



Nota: El grafico se agrupa según edades, muestra que la asistencia educativa disminuye progresivamente con la edad, siendo alta en la infancia y adolescencia, pero baja drásticamente a partir de los 21 años. Tomado de: INEI-Censos 2017.

Las personas con discapacidades físicas pueden sentirse excluidas de la revolución digital actual debido a su incapacidad para interactuar con la tecnología. Aunque el trabajo y el estudio remotos se han vuelto cada vez más comunes, estas personas a menudo no tienen acceso a estos recursos debido a la falta de adaptaciones para sus necesidades específicas. Para garantizar una inclusión digital completa, es fundamental encontrar soluciones que permitan a estas personas interactuar con la tecnología y aprovechar las oportunidades que ofrece la era digital.

Existen varias formas de interactuar con el mundo digital a través de la computadora utilizando diferentes periféricos. A pesar de que existen periféricos modernos que compiten

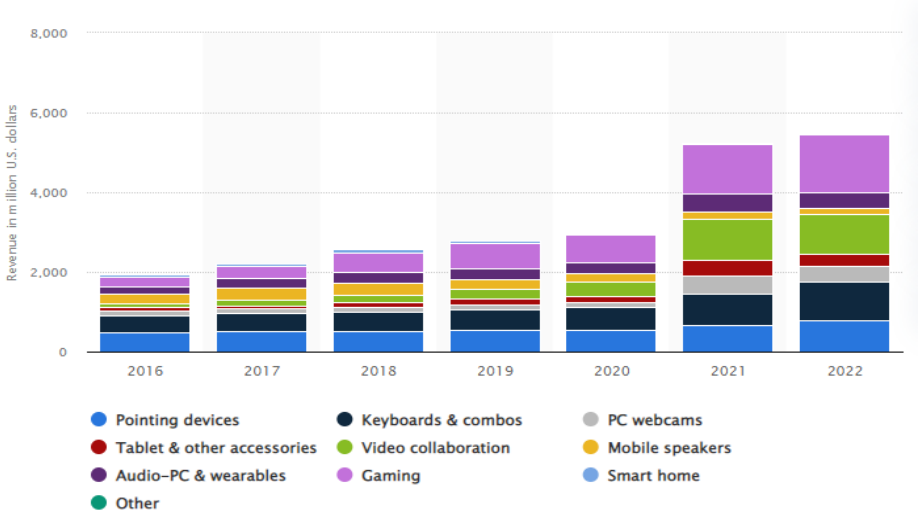
con el mouse, como el trackpad y las tabletas gráficas, el mouse sigue siendo el dispositivo de entrada más utilizado. Según una encuesta realizada por la revista The Atlantic, el 75% de las personas encuestadas todavía utilizan el mouse y lo consideran esencial para el uso de la computadora, a pesar de que este dispositivo fue creado hace muchos años (Madrigal, 2014).

Además del mouse, otro periférico importante que complementa su uso es el teclado, ya que permite ingresar texto y realizar diversas acciones en la computadora. Ambos periféricos son esenciales para una experiencia completa de navegación en el mundo digital.

La **Figura 3** muestra las ventas de la empresa Logitech en diferentes años, donde se puede observar que la venta de estos dos periféricos (el mouse en Pointing Devices y el teclado en Keyboards & Combos) representan la mayoría de sus ventas desde antes de la pandemia, y aún siguen siendo considerables después de ella.

Figura 3

Ingresos de Logitech International en todo el mundo desde el año 2016 hasta 2022.

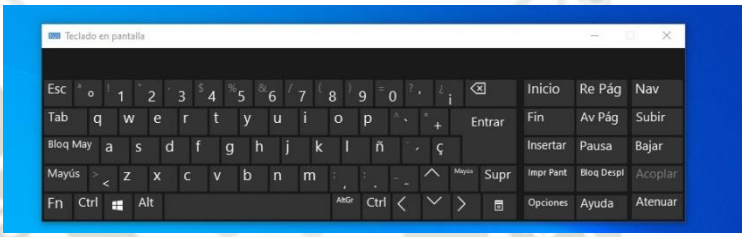


Nota: El gráfico refleja un crecimiento sostenido en los ingresos por accesorios tecnológicos de PC entre 2016 y 2022, destacando un aumento notable desde 2020 debido a la mayor demanda de productos para trabajo remoto y entretenimiento, coincidiendo con la pandemia del COVID-19. Tomado de: Statista.com

A pesar de la importancia que tiene el teclado físico como periférico de entrada, existen alternativas que permiten reemplazarlo. En la actualidad, muchos dispositivos incorporan teclados digitales que brindan una opción adicional para la entrada de texto, los cuales pueden ser manipulados a través de la pantalla táctil del dispositivo o con el uso del puntero del mouse.

Figura 4

Teclado virtual en Pantalla, Windows 10.

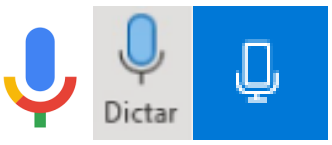


Nota: Se muestra la ventana con el teclado virtual disponible en el sistema operativo Windows. Fuente propia.

Otra alternativa disponible es el uso del reconocimiento de voz, una tecnología implementada en diversos programas como Google Chrome, Microsoft Word o Windows 10, que permite la transcripción del habla en texto escrito. Si bien esta opción requiere de una buena pronunciación y de la eliminación de ruidos ambientales, puede ser muy útil en situaciones en las que la entrada de texto manual es difícil o imposible.

Figura 5

Asistentes de voz para convertir voz en texto.



Nota: De Izquierda a derecha.1) Búsqueda por voz de Google. 2)Herramienta Dictar de M. Word. 3)Dictado por Voz de Windows 10. Fuente propia.

Por lo expuesto, podemos ver que es posible llevar a cabo las interacciones que tradicionalmente se realizan con el teclado de la computadora utilizando solamente el mouse, gracias a la disponibilidad de herramientas como el teclado virtual y los programas

de reconocimiento de voz, dejando al mouse como principal periférico de entrada para interactuar con el mundo digital.

Este enfoque se presenta como una solución efectiva para personas con discapacidades que les impiden utilizar una computadora de manera convencional, permitiéndoles hacer uso de la amplia gama de herramientas que ofrece la computadora para realizar tareas cotidianas. En este estudio, se explorará la posibilidad de utilizar bioseñales del cuerpo para interpretar los movimientos del puntero del mouse en la computadora, en un esfuerzo por mejorar la accesibilidad para personas con discapacidades y así fomentar la inclusión digital.

2. OBJETIVOS

2.1. Objetivo General

- Adaptar el uso del mouse de computadora mediante el procesamiento y caracterización de las bioseñales generados por el cuerpo humano, con el fin de lograr el control del puntero de una computadora a voluntad del usuario que tenga alguna discapacidad que lo inhabilite de usar uno convencionalmente.

2.2. Objetivos Específicos

- Estudiar del protocolo USB en su configuración HID, y del proceso que se realiza en la configuración interna del mouse en la computadora.
- Configurar el PIC18F2550 para controlar su modulo USB en la configuración HID y permitir que actúe como interfaz humano-computadora.
- Configurar y analizar del dsPIC33CH128MP506 y de su arquitectura de 16 bits para ser utilizado como centro de procesamiento digital y caracterización de señales obtenidas del cuerpo humano.

- Investigar el proceso de generación de las bioseñales de los ojos (EOG) y de los músculos (EMG) para conocer los mejores lugares donde colocar los electrodos superficiales y poder canalizarlas a las etapas de acondicionamiento y procesamiento.
- Acondicionar las bioseñales obtenidos por medios no invasivos (electrodos superficiales) para llevarlos a un nivel de voltaje que puedan ser utilizados por el microcontrolador encargado de su procesamiento.
- Construir un prototipo funcional que contenga todas las etapas de comunicación, procesamiento y adquisición de las bioseñales en un solo dispositivo.

3. MARCO TEÓRICO

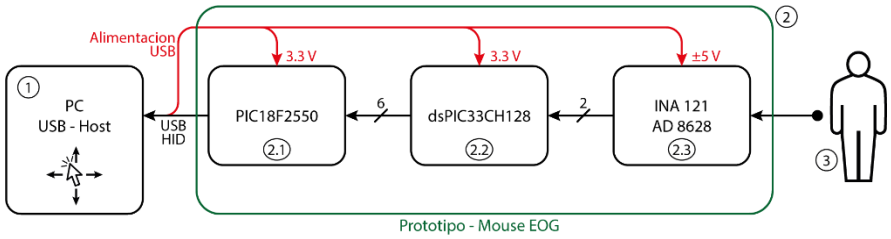
3.1. Principio de Funcionamiento

Este proyecto tiene como objetivo analizar las bioseñales del cuerpo humano para utilizarlas como señales en el movimiento del puntero de una computadora. Para lograr este propósito, se identificará y segmentará todo el sistema, tal como se muestra en la **Figura 6**. El enfoque de análisis del proyecto se divide en tres partes fundamentales:

I. La computadora.
II. El dispositivo USB.
III. El cuerpo humano.

Figura 6

Segmentación del sistema que es objeto de este proyecto de tesis.



Nota: La figura muestra los 3 componentes principales para el desarrollo del prototipo, 1) la computadora. 2) el prototipo funcionando como mouse. 3) las bioseñales del cuerpo humano con las que se controlara el puntero a través del prototipo. Fuente propia.

I. En la primera etapa del proceso de diseño de nuestro dispositivo USB, es necesario comprender el funcionamiento del protocolo USB en su configuración HID (Dispositivo de Interfaz Humana) en la computadora, es importante destacar que el protocolo USB trabaja de la misma manera en cualquier dispositivo que lo integre, lo que nos permite un funcionamiento multiplataforma. En esta etapa en particular, analizaremos en detalle el proceso de configuración y operación de un mouse estándar, ya que servirá como punto de referencia para el diseño de nuestro dispositivo.

   Al comprender el proceso mediante el cual, el sistema operativo identifica e instala de manera automática los controladores necesarios para el funcionamiento del mouse, y como es que se convierte y transmite el movimiento del mouse hacia la computadora, podremos aplicar dicho conocimiento en el diseño y desarrollo de nuestro propio dispositivo USB ajustado a nuestros requerimientos.

II. La segunda etapa del proceso aborda el diseño e implementación del dispositivo propuesto, el cual actuará como la interfaz entre las bioseñales humanas y el movimiento del puntero de la computadora. Para lograr esto, subdividiremos esta etapa en tres subetapas, cada una encargada de tareas críticas en el proceso de adaptar las bioseñales:

   II.i. En la primera subetapa, se empleará el integrado PIC18F2550, que cuenta con un módulo USB integrado, el cual se utilizará en la configuración HID, para permitir

que nuestro dispositivo actúe como un mouse estándar, sin necesidad de drivers adicionales, esto asegura la compatibilidad Plug and Play y Hot Swap, dos aspectos importantes y apreciados dentro del protocolo USB que generalmente damos por sentado (Tecnología invisible). Los movimientos esperados de un mouse estándar (arriba, abajo, derecha, izquierda, click derecho e izquierdo) serán accionado por la orden de la segunda subetapa, la cual está encargada del análisis y procesamiento digital de las bioseñales obtenidas del cuerpo. Además, en esta primera subetapa se configura la velocidad y la sensibilidad del movimiento del puntero.

II.ii. En la segunda subetapa, se realiza el procesamiento digital de las bioseñales, para lograr esto se utilizará el integrado dsPIC33CH128MP506, que cuenta con hardware embebido especializado que lo hace altamente eficiente para este propósito. Este integrado se encargará de realizar el filtrado digital (FIR) a todos los canales y mediante la caracterización de estas señales filtradas, se interpretará el comportamiento de las bioseñales con lo que se podrá enviar las ordenes al PIC18F2550 para mover el puntero del mouse de acuerdo con las intenciones del usuario.

II.iii. En la tercera subetapa, se realiza el proceso de captura y acondicionamiento de bioseñales obtenidos de los ojos (EOG), se utilizarán amplificadores operacionales de instrumentación, específicamente el INA121 y AD8628, con el fin de amplificar, acondicionar y preparar las bioseñales para que puedan ser procesadas en la segunda subetapa. Se implementarán al menos dos canales para los ejes X e Y, respectivamente, con el fin de analizar los movimientos oculares en diferentes direcciones. Además, se incorporará un canal adicional que acondicionará la bioseñal muscular (EMG) más apropiada, lo cual contribuirá a mejorar el control del movimiento del mouse.

Es importante destacar que, esta segunda subetapa junto a la primera subetapa, combinan y utilizan de manera embebida en nuestro dispositivo, el protocolo USB en su configuración HID y el procesamiento digital de las bioseñales, esto nos permite utilizar el dispositivo en una amplia variedad de dispositivos electrónicos como celulares, tabletas, Smart TVs, computadoras y cualquier otro dispositivo que actúe como Host de USB y utilice un mouse para la navegación, es decir, estas dos sub etapas permitirá que nuestro dispositivo sea multiplataforma.

Figura 7

Aparatos con soporte para usar un mouse para su control, potencial multiplataforma del prototipo.

Nota: La figura muestra una serie de aparatos multimedia, los cuales soportan el control de su interfaz por medio de un mouse estándar, siendo el prototipo de esta tesis un mouse multiplataforma, estos aparatos tienen el potencial de ser funcionales sin necesidad de hardware o software adicional. Fuente: infobae.com

III. La tercera etapa del sistema se enfoca en el cuerpo humano y su capacidad de generar bioseñales y biopotenciales. Se llevará a cabo un análisis exhaustivo de los patrones de señales de los ojos y músculos, identificando los puntos más adecuados para la toma de estas bioseñales mediante técnicas no invasivas. Además, se establecerá un punto de referencia de voltaje (tierra) para un mejor análisis y procesamiento de las señales obtenidas.

3.2. Antecedentes

En la convención sobre los derechos de las personas con discapacidad de las Naciones Unidas (ONU) se estableció diferentes medidas para proteger a personas con cualquier tipo de discapacidades, entre ellas están la no discriminación, la participación e inclusión plena y efectiva en la sociedad, igualdad de oportunidades y la accesibilidad (ONU, 2006) las cuales deben ser cumplidas y respetadas por todos los países miembros. Para el desarrollo de este proyecto de tesis tomamos como pilares la igualdad de oportunidades y la accesibilidad.

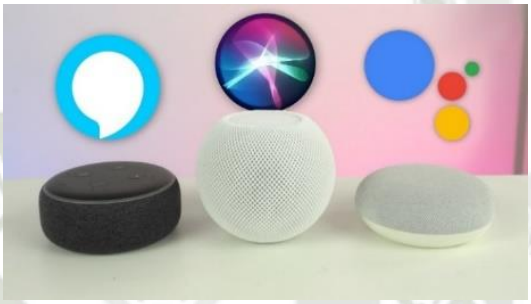
Siendo el Perú un miembro de la ONU desde el año 1945, el estado viene integrando medidas para cumplir con este convenio, el Decreto Supremo más reciente indica que el estado debe adoptar *"medidas pertinentes para asegurar el acceso de las personas con discapacidad, en igualdad de condiciones con las demás, al entorno físico, el transporte, la información y las comunicaciones, a fin de que puedan vivir en forma independiente y participen plenamente en todos los aspectos de la vida cotidiana"* (Decreto Supremo que aprueba el Plan Nacional de Accesibilidad 2018-2023, 2018), por esta razón, se permite que las personas con discapacidad accedan a licencias de conducir, siempre y cuando se realicen las adaptaciones necesarias para cubrir sus discapacidades y se supere el examen de manejo. Además, gracias a la implementación del CONADIS, se han establecido beneficios para este grupo de personas, lo que les ha permitido acceder a una mayor inclusión y participación en la sociedad.

La inclusión de personas con discapacidad se observa en diversas áreas del desarrollo humano a nivel global. Un ejemplo de ello es que empresas como Google, han incorporado herramientas como Google Talk Back y su teclado Braille para celulares, que permiten a personas invidentes utilizar un celular de manera más accesible haciendo uso del sonido. Además, los asistentes de voz, como Siri, Cortana, Alexa, Sam y Google Assistant, entre

otros, pueden ser utilizados por personas con discapacidad para realizar tareas digitales como abrir aplicaciones, escribir mensajes de texto, realizar llamadas, reproducir música, navegar por internet, entre otras acciones, lo que representa un avance significativo en la inclusión y accesibilidad para estas personas.

Figura 8

Dispositivos con asistentes de voz e IA integrados



Nota: De izquierda a derecha: Alexa, Siri, Google Assistant. Fuente: iccsi.com.ar , Asistente inteligente: revolución de la IA.

Con la implementación de la domótica se logra la automatización de actividades que se realizan en el hogar, en las que se puede incluir control centralizado del alumbrado, calefacción, ventilación y electrodomésticos inteligentes. La domótica y las consolas remotas virtuales tienen un interés especial para las personas mayores y las personas con discapacidad ya que pueden ofrecerles una mejor calidad de vida dentro de los confines de su propio hogar evitando al mismo tiempo la pérdida de independencia que suele conllevar el traslado a una residencia y problemas de depresión (ITU, 2012).

Otro sector en el que se está observando una inclusión cada vez mayor es en la adaptación y creación de vehículos que permiten a las personas discapacitadas movilizarse de manera más rápida y cómoda. En la Universidad de Piura, se ha estado trabajando en el desarrollo de un vehículo autónomo para discapacitados durante algún tiempo. Sin embargo, no se trata solo de construir el modelo, sino de asegurarse de cumplir con las normas de seguridad vial, ofrecer un óptimo estado de confort y accesibilidad, y todo ello a un precio

accesible. Por lo tanto, los proyectos de investigación en este ámbito tienen que realizar análisis minuciosos para cumplir con estos parámetros y lograr así una inclusión real y efectiva en el sector de la movilidad para las personas con discapacidad (Gonzales Cruz, 2019).

Las investigaciones que se centran en el análisis señales nerviosas o cerebrales están abriendo nuevas posibilidades en el desarrollo de prótesis y exoesqueletos para permitir un mayor grado de independencia a personas con discapacidades motrices. Gracias al procesamiento de estas señales, se están sentando las bases para un futuro en el que las limitaciones físicas no impidan la autonomía de las personas con discapacidad. Por lo tanto, el estudio de los biopotenciales del cuerpo humano es de vital importancia para seguir avanzando en esta dirección y lograr una mayor inclusión y calidad de vida para las personas con discapacidad.

Figura 9

Uso de bioseñales para exoesqueletos de uso médico.

Nota: La figura muestra una escenificación del modo de funcionamiento del exoesqueleto "HANK" de la empresa Gogoa Mobility Robots. Fuente: gogoa.eu

Según nuestra investigación, se tiene dos enfoques para controlar el movimiento del mouse mediante señales corporales, con los ojos o con la cabeza. El primero se basa en el uso de una cámara web junto al procesamiento de imagen, mientras que el segundo hace uso de electrodos y procesamiento de señales para utilizar los biopotenciales del usuario.

Una de las ventajas de usar procesamiento de imágenes es que solo se requiere una cámara web, iluminación y el software adecuado. Por ejemplo, Enable Viacam es un proyecto gratuito y de código abierto que permite reemplazar el mouse con el movimiento de la cabeza detectado por una webcam, sin necesidad de cables adicionales ni hardware dedicado (Mauri, 2018). Aunque este programa utiliza el movimiento de toda la cabeza en lugar de solo los ojos, ha demostrado ser efectivo en su propósito de mejorar la accesibilidad de las personas que lo necesiten.

Otro programa que utiliza procesamiento de imágenes para controlar el movimiento del mouse es Precision Gaze Mouse. Sin embargo, a diferencia de Enable Viacam, este programa requiere de dos periféricos específicos para su funcionamiento preciso y efectivo: Tobii Eye Tracker y Track IR. A pesar de este requisito, Precision Gaze Mouse sigue siendo un proyecto gratuito y de código abierto.

Figura 10

Software Precision Gaze Mouse y sus periféricos adicionales para su funcionamiento.

Nota: La figura muestra el hardware y software adicional para funcionar en una computadora. (1) Track IR, detecta el movimiento de la cabeza. (2)Tobii eye tracker, detecta el movimiento de los ojos (3) Precision Gaze Mouse, integra las señales del hardware adicional realiza el movimiento del puntero del mouse en la computadora. Fuente: youtube.com, Precision Gaze Mouse Demo

Gonzalo Salinas Zegarra de la Universidad Católica Santa María desarrolló una tesis en la que implementa el control de una silla de ruedas utilizando el movimiento del iris del ojo a través de una cámara y comunicación USB para controlar servomotores con la computadora. El procesamiento de imágenes se realiza en la computadora y se envía la orden

de movimiento de los servomotores por medio del protocolo USB en la configuración CDC con el PIC18F4550 (Zegarra, 2013).

Figura 11

Prototipo para el control de silla de ruedas mediante el movimiento de los ojos.

Nota: La figura muestra un prototipo para controlar el movimiento de una silla de ruedas utilizando el procesamiento de las imágenes de los ojos obtenidas por una cámara montada en unos ante ojos. Fuente: Tesis Salinas Zegarra, 2013

En un reportaje emitido por el canal de YouTube, Agencia EFE, el 7 de febrero de 2011, titulado "El ratón controlado por los ojos que aspira a acercar Internet a personas con discapacidad", se presenta un dispositivo desarrollado por estudiantes de la Universidad Politécnica de Palestina en Hebrón. Este dispositivo, permite a personas con discapacidades físicas totales acceder a Internet mediante el control ocular (Agencia EFE, 2011). Si bien esta investigación es de gran interés como antecedente para la presente Tesis, se han encontrado dificultades al tratar de obtener más detalles sobre la misma.

Figura 12

Entrevista sobre un prototipo similar al realizado en esta tesis

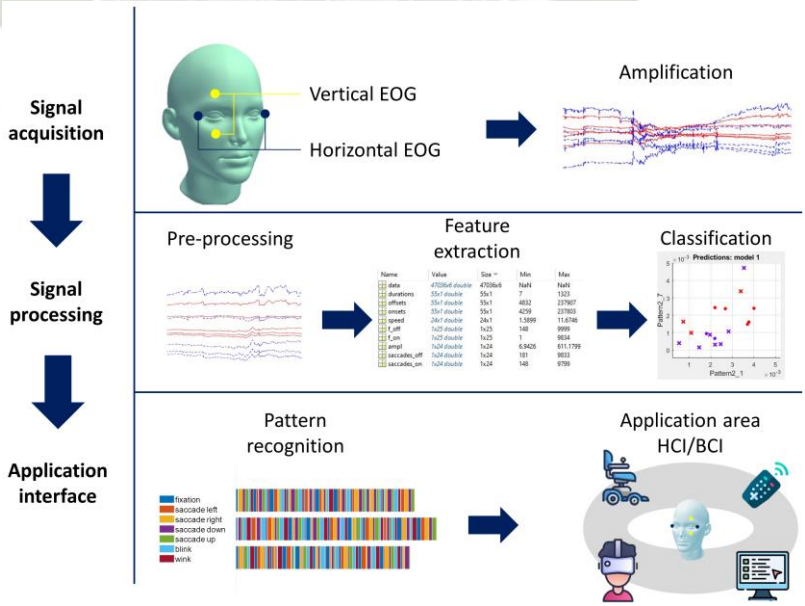
Nota: La figura muestra parte de una entrevista, donde muestran el funcionamiento de un mouse controlado por el movimiento de los ojos a través de electrodos. "El ratón controlado

por los ojos que aspira a acercar a internet a personas con discapacidad" (0:25). Fuente: Youtube.com, Canal: AGENCIA EFE

En la revista "Sensors", se hace una análisis general de lo que ellos denominan: Electro-oculography based brain-computer interface(EOG-BCI) (Interfaz cerebro-computadora basada en electro calografía). En este mencionan que *"con las señales EOG, se puede identificar y clasificar los movimientos oculares mediante interacciones activas o pasivas"* (C., A., C., & V., 2022) . Según esta revista, estas interacciones tienen el potencial de controlar la salida de dispositivos, permitiendo al usuario comunicarse e interactuar con su entorno. Un ejemplo que mencionan, es que, en el campo de la aeronáutica, están investigando sistemas basados en EOG-BCI, para su utilización como herramientas para reemplazar comandos manuales y como herramienta de comunicación para acelerar las interacciones de los usuarios.

Figura 13

Esquema de un sistema EOG-BCI (Electro-oculography based Brain-Computer Interface).



Nota: La figura muestra un flujograma general que utilizan en sus proyectos para controlar aparatos mediante el movimiento ocular capturado por electrodos. Fuente: (C., A., C., & V., 2022)

Recientemente, la industria de los videojuegos ha comenzado a implementar tecnologías de seguimiento ocular para mejorar la experiencia de los jugadores. Actualmente, existen portátiles y monitores con sistemas de seguimiento ocular integrados en el mercado. No obstante, la mayoría de estos sistemas dependen del procesamiento de video, la iluminación con luz infrarroja y algoritmos de visión artificial complejos. Este método, que demanda mucha energía, no es ideal para un uso prolongado debido al consumo de energía y el calor generado. Además, el seguimiento ocular basado en video requiere integrar una cámara en el entorno existente, lo cual puede ser complicado en determinadas circunstancias, o usar gafas que pueden ser incompatibles con quienes ya usan gafas correctivas. Además, las cámaras pueden obstruir la visión del usuario de su entorno.

Una alternativa más eficiente, en términos energéticos, al seguimiento ocular por procesamiento de imagen, es la electrooculografía (EOG), que utiliza electrodos colocados en la piel de la cara. La EOG fue una de las primeras técnicas para estudiar los movimientos oculares y se ha considerado como una herramienta valiosa para la comunicación entre humanos y computadoras. Adicionalmente, una de las ventajas de la EOG es que los electrodos no interfieren con el campo visual del usuario.

3.3. Protocolo USB

El Bus Universal en Serie, USB (en inglés: Universal Serial Bus), es estándar de comunicaciones que define los cables, conectores y protocolos usados en un bus para conectar, comunicar y proveer de alimentación eléctrica entre computadoras, periféricos y dispositivos electrónicos.

El USB fue diseñado para economizar y estandarizar la conexión de periféricos.

Fue desarrollado a mediados de los años 1990; desde 2004 aproximadamente 6 mil millones de dispositivos se encuentran actualmente en el mercado global, y alrededor de 2 mil millones se venden cada año.

Su desarrollo partió de un grupo de empresas del sector que buscaban unificar la forma de conectar periféricos a sus equipos, las cuales eran poco compatibles entre sí, entre las que estaban Intel, Microsoft, IBM, Compaq, DEC, NEC y Nortel.

La primera especificación completa, USB 1.0, se publicó en 1996, pero en 1998 con la especificación USB 1.1 comenzó a usarse de forma masiva. El objetivo principal de esta estandarización consistía en el desarrollo de una interfaz única capaz de funcionar en diversas plataformas, con la intención de eliminar la necesidad de emplear múltiples conectores preexistentes, además, se buscaba aumentar la velocidad de transmisión de datos entre dispositivos. (BasuMallick, , 2023)

Las principales características del protocolo USB son:

1. Plug and Play: Permite la conexión y desconexión de dispositivos sin necesidad de reiniciar la computadora ni usar software adicional.
2. Hot Swapping: Permite conectar y desconectar dispositivos sin apagar el sistema (en caliente).
3. Alimentación Eléctrica: Muchos dispositivos USB pueden recibir alimentación eléctrica a través del mismo cable que se utiliza para la transferencia de datos.
4. Versatilidad: Es utilizado para una amplia variedad de dispositivos, desde unidades de almacenamiento e impresoras hasta cámaras y dispositivos de entrada.
5. Compatibilidad: Es compatible con varios sistemas operativos, como Windows, macOS, Linux, entre otros.

6. Velocidades de Transferencia: Existen diferentes versiones de USB con distintas velocidades de transferencia de datos, como USB 2.0, USB 3.0, USB 3.1, y USB 3.2.
7. Conectividad en Cadena: Permite conectar varios dispositivos en cadena (daisy chain), extendiendo así las posibilidades de conexión, pudiendo conectar hasta 127 dispositivos.
8. Interfaz Estándar: Proporciona una interfaz estándar para la conexión de periféricos, facilitando la interoperabilidad.

3.3.1. Estándar Mecánico

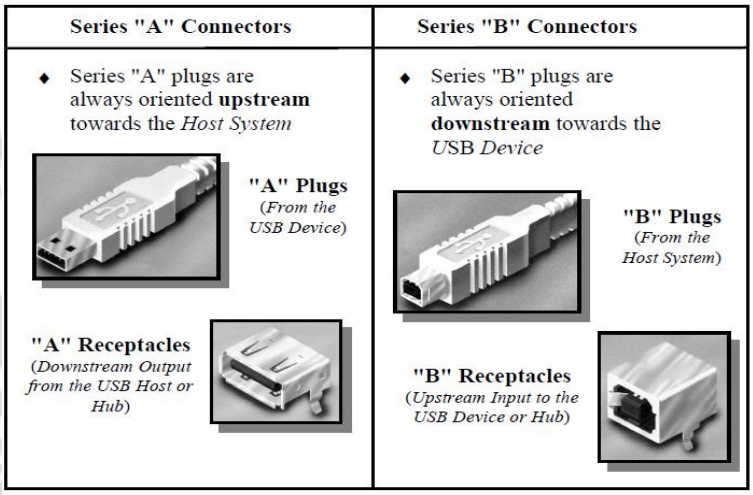
La topología física del USB implica la interconexión del puerto de un concentrador (hub) principal con el puerto de otro concentrador inferior o con un dispositivo. El estándar USB 2.0 tiene la capacidad de operar a tres velocidades distintas. Para la velocidad alta (High Speed - 480 Mbps) y la velocidad completa (Full Speed - 12 Mbps), se hace imprescindible la utilización de un cable blindado que conste de dos conductores de alimentación y conductores de señal dispuestos en par trenzado. En el caso de la baja velocidad (Low Speed 1.5 Mbps), se sugiere, aunque no se obliga, el empleo de un cable con conductores de señal en par trenzado.

Con el fin de mitigar posibles inconvenientes de conexión por parte del usuario final, el estándar USB implementa un protocolo conocido como *"keyed connector"*, este término alude al uso de conectores con perfiles mecánicos distintivos que solo pueden acoplarse de manera específica y que excluyen la posibilidad de conexión errónea o con conectores de diseño diferente. La diferencia física entre los conectores tipo "A" y "B" asegura una conectividad apropiada para el usuario final. El conector tipo "A" constituye el principal medio para vincular dispositivos USB directamente a un host o al puerto descendente de un concentrador. En contraste, el conector "B" permite a los fabricantes de dispositivos

suministrar un cable desmontable estandarizado, facilitando así la eventual sustitución del cable por parte del usuario final.

Figura 14

"Keyed Connector", Conectores Tipo A y B.



Nota: La imagen ilustra dos tipos principales de conectores USB: Serie "A" y Serie "B", los cuales tienen la característica de conectarse de una sola forma. Tomado de: USB.org

Conforme al estándar del USB 2.0, los dispositivos operando a velocidades High-speed y Full-speed, pueden utilizar cables desmontables, caracterizados por la presencia del conector tipo "A", el cual se encuentra permanentemente asociado al Host, y en el extremo opuesto, el conector tipo "B", vinculado al dispositivo. Esta disposición facilita el intercambio de cables entre diferentes dispositivos USB, eliminando la necesidad de acumular múltiples cables y simplificando la sustitución de aquellos que se encuentren defectuosos o dañados. Según estándar, los cables High-speed y Full-speed constan de un par trenzado de cables de señales D+ y D- (verde y blanco respectivamente), recubiertos por una capa de aluminio apantalladora, así como de VBUS y GND (rojo y negro respectivamente), todos de calibre 28 AWG (USB-IF, Abril del 2000).

3.3.2. Estándar Eléctrico

A. Alimentación

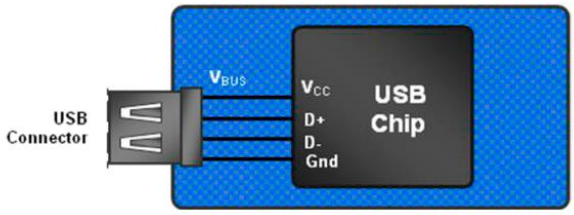
Una de las características del estándar USB es que puede alimentar al dispositivo USB a través del mismo bus, esto es gracias a que admite varias formas de suministro y consumo de energía, de los cuales los más importantes son Bus-Powered (alimentados a través del bus) y Self-Powered (Alimentación externa).

i. Dispositivo Bus-Powered

A través de la configuración Bus-powered, la energía para el dispositivo USB se obtiene directamente del bus, eliminando la necesidad de fuentes de alimentación internas o externas adicionales, esta energía es suministrada por el Host.

El dispositivo debe limitar su consumo de energía antes que complete su configuración, es decir, desde la conexión inicial al bus hasta la finalización del proceso de **Enumeración**. Durante este periodo, el dispositivo no debe exceder los 100 mA de consumo. Según la especificación USB, establece este límite como carga unitaria para dispositivos de baja, completa o alta velocidad (Universal Serial Bus Specification., Abril del 2000). En la fase configuración, el dispositivo solicita su energía nominal y se le es asignada al concluir el proceso de Enumeración (USB-IF, Abril del 2000).

Figura 15
Dispositivo Bus-Powered



Nota: La figura muestra la forma de alimentar el chip USB cuando la alimentación está configurada como Bus Powered. Fuente: Cypress Technologies

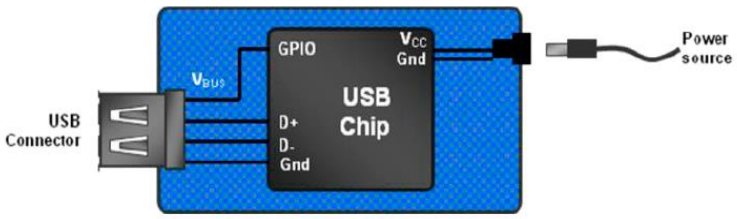
Los dispositivos alimentados a través del bus se clasifican en dos categorías: baja potencia (hasta 100 mA) y alta potencia (hasta 500 mA). Consumos superiores a 500 mA requieren que el dispositivo cuente con su propia fuente de alimentación.

En el apartado 3.3.1, se describe como a través del Descriptor de Configuración, en el campo “bmAttributes” y “bMaxPower”, se puede seleccionar el modo y la cantidad de energía que utilizara el dispositivo USB (Murphy, 2010).

ii. Dispositivo Self-Powered

Los dispositivos autoalimentados se abastecen de su propia energía mediante una fuente externa, como un adaptador de corriente o una batería. La especificación USB exige que estos dispositivos monitoreen continuamente su línea V_{BUS}, retirando la alimentación de su resistor pull-up D+/D- en un tiempo específico después de desconectar V_{BUS} para evitar retroceso de tensión. (Murphy, 2010)

Figura 16
Dispositivo Self-Powered



Nota: La figura muestra la forma de alimentar el chip USB cuando la alimentación está configurada como Self-Powered. Fuente: Cypress Technologies

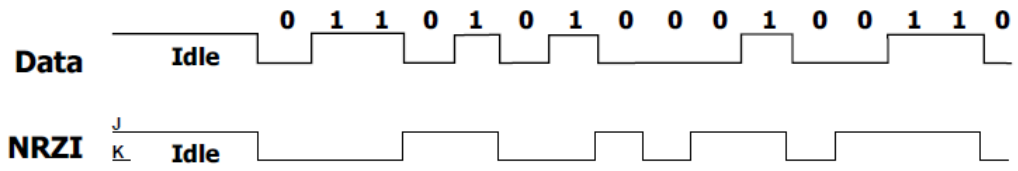
Un dispositivo autoalimentado puede extraer hasta 100 mA del bus (Hibrido), esto permite que el dispositivo pueda comunicarse con el Host aun si la alimentación externa del dispositivo no este prendido, esto permite que el Host pueda diferenciar si el dispositivo esta desconectado o desenrizado.

El dispositivo autoalimentado debe implementar protección de sobre corriente y reportar al Host sobre esta condición. (Murphy, 2010)

B. Codificación de Datos

El protocolo USB emplea la codificación de datos NRZI (Inversa No Retorno a Cero) durante la transmisión de paquetes. En esta codificación, la representación de un "1" se caracteriza por la ausencia de cambios en el nivel, mientras que un "0" se denota mediante un cambio en el nivel. La **Figura 17** ilustra una secuencia de datos y su equivalente en NRZI. El nivel alto representa el estado J en las líneas de datos mientras que el nivel najo representa el estado K. Una secuencia continua de ceros induce alternancias en los datos NRZI en cada intervalo de bits. (Murphy, 2010)

Figura 17
Codificación de datos NRZI (Non-Return-to-Zero)



Nota: La figura muestra una trama de datos en codificación NRZI. Fuente: www.usb.org/

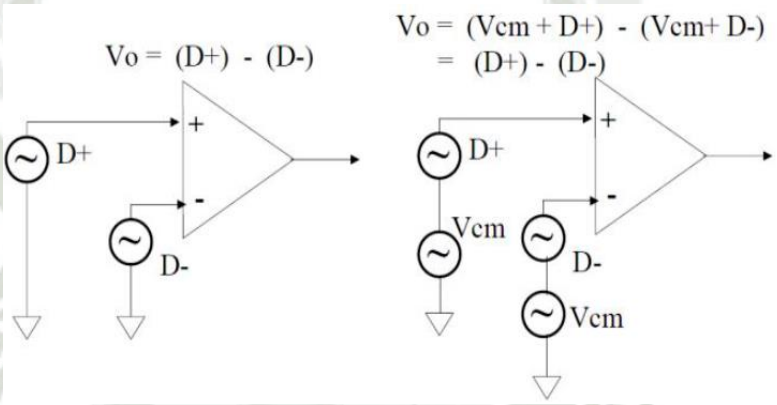
Dentro del protocolo USB, la elección de emplear dos líneas de comunicación inversas D+ y D-, se fundamenta en la técnica de codificación NRZI (Inversa No Retorno a Cero) utilizada para la transmisión de datos. La inversión de estas líneas se traduce en que cualquier cambio de estado en una de ellas se refleje inversamente en la otra. Este diseño tiene como propósito principal facilitar y asegurar la distinción entre los bits "1" y "0" durante el proceso de transmisión, ofreciendo así redundancia informativa.

Adicionalmente, en una configuración diferencial como la de D+ y D-, el ruido que afecta ambas líneas por igual tiende a cancelarse cuando se realiza la resta entre ellas en un amplificador diferencial. Esto mejora la inmunidad del sistema ante interferencias y ruido

externo, ya que cualquier interferencia que afecte ambas líneas de manera similar será atenuada durante el proceso de recepción. Este enfoque diferencial es comúnmente utilizado para mejorar la calidad de la señal y la resistencia a interferencias en sistemas de comunicación. (Murphy, 2010)

Figura 18

Comprobación del rechazo en modo común del ruido en líneas diferenciales



Nota: La figura muestra el comportamiento de los OPAMPs ante ruido que afecta a ambas entradas (rechazo en modo común). Fuente: Cypress Technologies

Por lo tanto, la utilización de señales D+ y D- como inversos en USB no solo se relaciona con la codificación NRZI sino también con la capacidad de rechazo al ruido en modo común, mejorando la robustez y la integridad de la transmisión de datos.

La comunicación USB se desarrolla a través de estados en la señalización en las líneas D+ y D-. Algunos de estos estados están destinados para la transmisión de datos, mientras que otros cumplen funcione definidas en el protocolo USB. (Murphy, 2010)

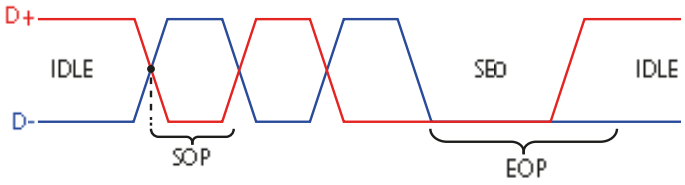
Tabla 1

.Estado del bus USB D+ y D-.

ESTADO DEL BUS	ESTADO DE LAS LINEAS	DESCRIPCION
Differential 1	D+ Alto D- Bajo	Estos estados son utilizados para la comunicación de datos general.
Differential 0	D+ Bajo D- Alto	
Single Ended 0 (SE0)	D+ y D- Bajo	Ambas líneas son puestas a nivel bajo. Indica reinicio, desconexión o Fin de un paquete (EOP).
Single Ended 1 (SE1)	D- y D+ Alto	Ambas líneas son puestas a nivel alto. No debe ocurrir en una transmisión USB.
Estado J		Representa un nivel lógico alto (1 lógico) en la transmisión de datos.
Low-Speed	Differential 0	
Full-Speed	Differential 1	El estado de las líneas D+/- varía dependiendo la velocidad de transmisión.
High-Speed	Differential 1	
Estado K		Representa un nivel lógico bajo (0 lógico) en la transmisión de datos.
Low-Speed	Differential 1	
Full-Speed	Differential 0	El estado de las líneas D+/- varía dependiendo la velocidad de transmisión.
High-Speed	Differential 0	
Estado Resumir	Estado K	Se usa para despertar el dispositivo luego de una suspensión.
Idle:		Ocurre antes y después de enviar un paquete de datos.
Low-Speed	Differential 0	
Full-Speed	Differential 1	
High-Speed	Differential 1	
Start of Packet (SOP)	Líneas de datos cambian de estado Idle a estado K.	
End of Packet (EOP)	Estado SE0 por 2 bit seguido de un estado J	

Figura 19

Estado de las líneas de comunicación D+ y D-.



Nota: La figura muestra el comportamiento de las señales USB al inicio de un intercambio de datos entre el Host y el dispositivo USB, empezando por un paquete SOP (Start Of Package) y finalizando en un EOP (End Of Package). Fuente propia.

3.3.3. Modelo de Flujo de Datos

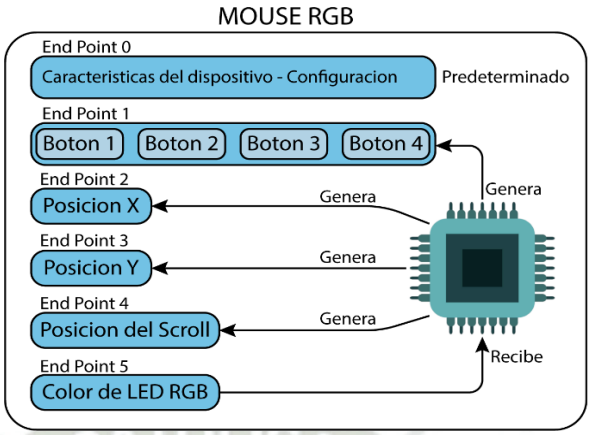
A. End Point

Un punto final (End Point) de un dispositivo es una porción única de la memoria de un dispositivo USB a la cual, el Host le asigna una dirección única. Un dispositivo USB puede tener más de un punto final. El punto final de un dispositivo es la fuente o destino de información en un flujo de comunicación entre el Host y el dispositivo. Cada uno de estos puntos finales tiene una dirección e identificador único (número de punto final) asignada por el Host, el sentido de flujo del punto final es determinado en el momento del diseño del dispositivo. La combinación de la dirección del dispositivo, el número de punto final y el sentido del flujo, permite referenciar de manera única a cada punto final.

Durante la secuencia de enumeración, se utiliza un conjunto especial de puntos finales para la comunicación y configuración inicial del dispositivo. Estos puntos finales especiales, son denominados como Punto Final de Control o Punto Final 0, se definen como Punto Final 0 IN y Punto Final 0 OUT. Aunque Punto Final 0 IN y Punto Final 0 OUT son dos puntos finales, aparentan y actúan como un solo punto final para el desarrollador. Cada dispositivo USB debe ser implementado por defecto con un Punto Final 0 IN y OUT. (Murphy, 2010)

Figura 20

Usos de los End Point, sentido del flujo de transmisión.



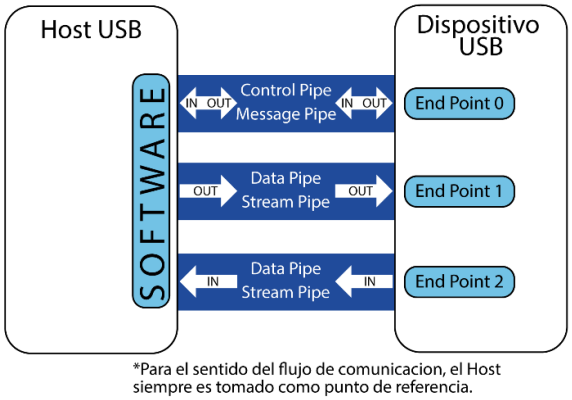
Nota: La figura muestra los espacios de memoria en el periférico USB (End Point) donde se almacena información sobre el comportamiento del mouse, y el sentido de transmisión de estos datos con el Host. Fuente propia.

B. Pipes

El USB posibilita el intercambio funcional de información de datos y control entre el Host USB y un dispositivo USB a través de una serie de canales designados como "pipes" (tuberías), que pueden ser unidireccionales o bidireccionales. Una tubería relaciona un End Point del dispositivo USB con el software del Host. El flujo de datos en una tubería es independiente del flujo de datos en cualquier otra tubería. Un dispositivo USB dado puede contar con múltiples tuberías. Por ejemplo, un dispositivo USB puede tener un punto final que admita una tubería para el transporte de datos hacia el dispositivo y otro punto final que admita una tubería para el transporte de datos desde el dispositivo hacia el Host.

Figura 21

Tipo y sentido de Tuberías de comunicación (Pipes)



Nota: La figura muestra las conexiones virtuales (Pipes) que se crean entre el Host y el dispositivo USB para la transmisión de datos almacenados en la memoria del dispositivo USB. Fuente: Modificado de Cypress Technologies.

Se definen dos tipos de tuberías de comunicación, Stream Pipe (Tubería de datos) los cuales no tienen una estructura definida, ya que estas se adaptan de acuerdo al diseño del dispositivo USB, estos siempre transmiten información desde un único End Point "X" de manera unidireccional. El otro tipo de tubería de comunicación es Message Pipe (Tubería de Control) estos tienen una estructura estandarizada para todos los dispositivos ya que son los que el Host utiliza este tipo para realizar la configuración inicial del dispositivo USB, este tipo de tubería está asociado únicamente con el Endpoint 0 y se realiza de manera bidireccional, ya que el Host envía, pide y recibe información para la configuración inicial del dispositivo USB.

C. Tipo de Transferencias

El protocolo USB transporta datos a través de una tubería (pipe) entre una porción de memoria asociado con un cliente de software en el Host y un punto final (endpoint) en el dispositivo USB, a este flujo de información a través de un canal especifico, se le denomina "Transferencia". En una comunicación USB, pueden existir varios tipos de transferencias, cada uno diseñado para satisfacer distintos requisitos de aplicaciones y servicios de cada

punto final del dispositivo. Los principales tipos de transferencias estandarizados en el protocolo USB son: (Pantoja & Montilla , 2007)

1. **Transferencia de Control:** Se utilizan para la configuración del dispositivo, el control y el estado. Son esenciales para el establecimiento inicial de la comunicación entre el host y el dispositivo USB.
2. **Transferencia de Masa o Bulk:** Ideales para transferir grandes cantidades de datos a velocidades elevadas, pero sin garantía de tiempo. Son comunes en dispositivos como discos duros USB y memorias flash.
3. **Transferencias Isocrónicas:** Proporcionan una transferencia de datos en tiempo real con un ancho de banda garantizado, pero sin garantía de entrega. Son esenciales para aplicaciones de audio y video, donde la continuidad de la transmisión es crítica.
4. **Transferencia de Interrupción:** Permiten la transferencia de pequeñas cantidades de datos con requisito de latencia baja. Son útiles para dispositivos como teclados y ratones, donde la rapidez de respuesta es fundamental

Estos tipos de transferencia delinean las propiedades fundamentales del flujo de comunicaciones entre el Host y el dispositivo, facilitando así la adaptación a las exigencias de rendimiento y funcionalidad en múltiples tipos de comunicación con distintos dispositivos. Las características primordiales de las transferencias incluyen: (Pantoja & Montilla , 2007)

- Establecen un formato de datos, impuesto por el estándar USB.
- Dirección del flujo de comunicación.
- Restricciones en el tamaño del paquete.
- Restricciones de acceso al bus.
- Restricciones de latencia.

- Secuencias de datos requeridas.
- Manejo y detección de errores.

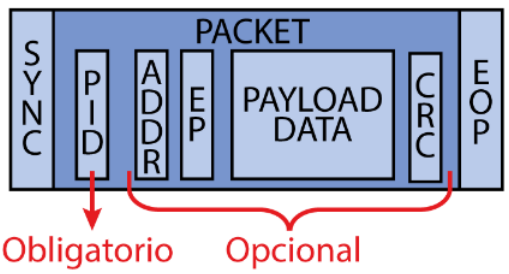
Los diseñadores de un dispositivo USB seleccionan el tipo de transferencia apropiado para los puntos finales (endpoints) del dispositivo. Una vez establecida una tubería (pipe) para un punto final, la mayoría de las características de transferencia de dicha tubería se determinan y permanecen inalterables durante toda la conexión de la tubería.

3.3.4. Capa de Protocolo / Protocolo de Comunicación

A. Estructura de Paquetes

Los paquetes son fundamentales para las diferentes fases de la comunicación USB, desde el establecimiento de la conexión y la configuración del dispositivo hasta la transferencia de datos y el control de errores. Dependiendo del tipo de transferencia (control, masa o bulk, isocrónica, e interrupción), el formato y el contenido de los paquetes pueden variar para adaptarse a los requerimientos específicos de cada tipo de comunicación. Sin embargo, todos comparten una estructura similar de campos: (Murphy, 2010)

Figura 22
Estructura general de los paquetes USB.

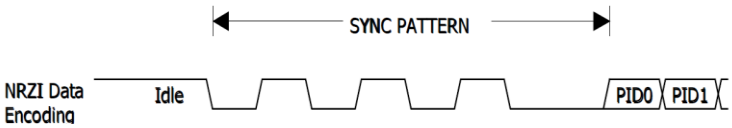


Nota: La figura muestra la estructura estándar de un paquete de datos en la comunicación USB. Fuente: Modificado de Cypress Technologiesa.

1. Patrón de Sincronización (Sync): Una secuencia de bits que marca el inicio del paquete y permite al receptor sincronizarse con el flujo de datos. Consiste de tres pares de estado K-J, seguido por dos estados K, haciendo un total de ocho bits.

Figura 23

Patrón de Sincronización

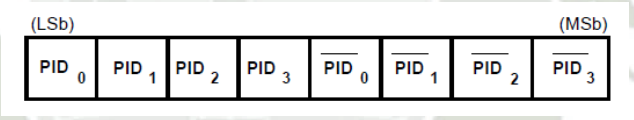


Nota: La figura muestra el comportamiento de las señales USB, los cuales marcan el inicio de la transmisión de un paquete de datos USB, adicionalmente, permite que tanto el Host como el dispositivo USB se sincronicen con el flujo de datos Fuente: Usb.org

2. Identificador de Paquete (PID): Un campo que indica el nombre del paquete, su función y, en consecuencia, su formato (por ejemplo, si es un paquete de datos, de control, de entrada, etc.). Este campo siempre debe ir en el paquete. Consiste de un paquete de 4 bits, seguido de 4 bits complementarios a los primeros 4 bits, esto permite detectar si ocurrió un error en la transmisión.

Figura 24

Formato del campo PID.

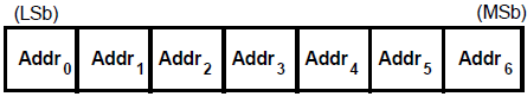


Nota: La figura muestra la estructura que tiene el Identificador del Paquete, se observa que los últimos bits transmitidos, son el complemento de los primeros, esto con el fin de identificar errores en la transmisión de datos. Fuente: Usb.org

3. Dirección (ADDR): Especifica el destino o el origen de los datos que serán transmitidos en una transmisión. En un entorno donde múltiples dispositivos USB están conectados al mismo Host, este campo es crucial para gestionar el tráfico de los datos. Consiste de un paquete de 7 bits, en teoría permitiría 128 direcciones, pero debido a que la dirección 0x0 está reservada para el proceso de configuración por lo que no se asigna, permite la conexión de un máximo de 127 dispositivos.

Figura 25

Formato del campo ADDRESS.

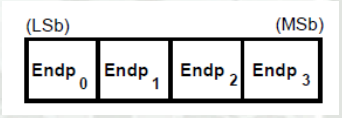


Nota: La figura muestra la estructura que tiene el campo de Dirección. Fuente : Usb.org

4. Numero de Punto Final (EP): Identifica uno de los Puntos Finales de un dispositivo USB. Cada dispositivo USB puede tener múltiples End Points y cada uno puede estar configurado para un tipo especifico de transferencia. Este campo consiste de 4 bits, lo que permite asignar 16 Puntos Finales, siendo el EndPoint 0 reservado para la configuración y los otros 15 para propósito general.

Figura 26.

Formato del campo END POINT.

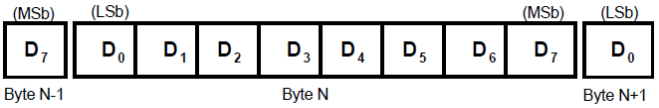


Nota: La figura muestra la estructura que tiene el campo de Punto Final. Fuente: Usb.org

5. Datos (Payload): El cuerpo del paquete, que contiene la información o los comandos que se están transmitiendo. No todos los paquetes contienen datos; algunos son solo para control o estado, puede tener hasta 1024 bytes.

Figura 27

Formato del campo PAYLOAD.



Nota: La figura muestra la estructura que tiene el campo de Datos, pudiendo tener hasta 1024 bytes. .Fuente: Usb.org

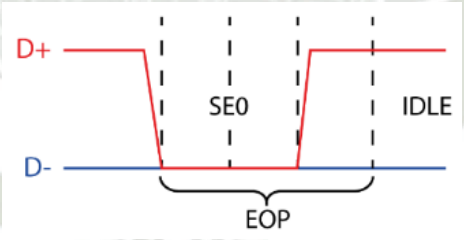
6. Chequeo de Redundancia Cíclica (CRC): Un código de error utilizado para verificar la integridad de los datos transmitidos en el paquete exceptuando el campo PID. Un CRC con error indica que uno o más de los campos protegidos están corrompidos, por lo que el paquete se descarta.

Tabla 2
Ecuación algebraica CRC5.

Token Packets CRC (CRC5)	Data Packets CRC (CRC16)
$G(X) = X^5 + X^2 + X^0$	$G(X) = X^{16} + X^{15} + X^2 + X^0$

7. Fin del Paquete (EOP): Es una secuencia de bits que marca el fin del paquete (End of Packet), este patrón se indica mediante una secuencia de dos bits del estado SE0 (Single Ended 0, Líneas de datos D+ y D- en estado bajo) seguido por un bit de estado J (D+ alto y D- bajo).

Figura 28
Patrón de Fin de Paquete (EOP).



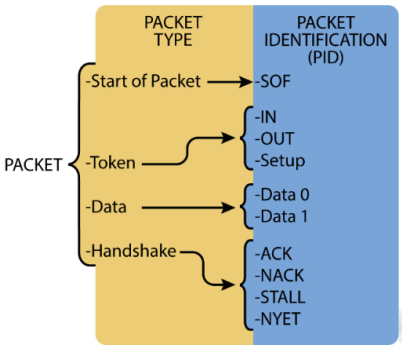
Nota: La figura muestra el comportamiento de las señales USB para marcar el fin de una transmisión de datos. Fuente propia.

B. Tipos y Nombres de Paquetes

Un paquete (Packet) se refiere a una secuencia de bits estructurada que se utiliza para la transmisión de datos entre el Host y el dispositivo USB conectado. Los paquetes son las unidades básicas de la comunicación en el USB. En el protocolo USB se diferencian tres tipos de paquetes, cada uno está diseñado para llevar a cabo una función específica dentro del proceso de comunicación, en la **Figura 29** se muestra un resumen global de los tipos de paquetes. (Thomas E. , 2002)

Figura 29

Tipos y nombres de Paquetes.



Nota: La figura muestra los tipos de paquetes y sus identificaciones (PID). Fuente propia.

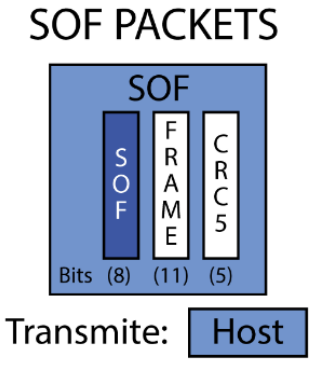
i. Start of Frame Packet

Los tokens SOF (Start of Frame) son emitidos por el Host, marcan el inicio de cada marco en la secuencia de comunicación y previenen la entrada de los dispositivos en modo de suspensión, ya que si en el bus no se envía tokens SOF en un periodo de 3 ms los dispositivos entran en suspensión. Los tokens SOF se transmiten a intervalos regulares de un milisegundo.

Los tokens SOF, se emiten únicamente en dispositivos de completa y alta velocidad. Estructuralmente incorporan un contador de “Frames” de 11 bits, que incrementa con cada Frame y se reinicia tras alcanzar su capacidad máxima (esto permite detectar la perdida de algún Frame) así como un código CRC de 5 bits.

Figura 30

Estructura del paquete SOF.



Nota: La figura muestra la estructura de los paquetes SOF, estos paquetes son enviados por el Host, ya que ellos son los que controlan la comunicación. Fuente: Modificado de Cypress Technologies.

Los paquetes SOF son esenciales en aplicaciones que requieren transferencias isocrónicas, como el audio y el video en tiempo real, donde la entrega de datos debe ocurrir en intervalos de tiempo precisos para evitar interrupciones o degradación de la calidad.

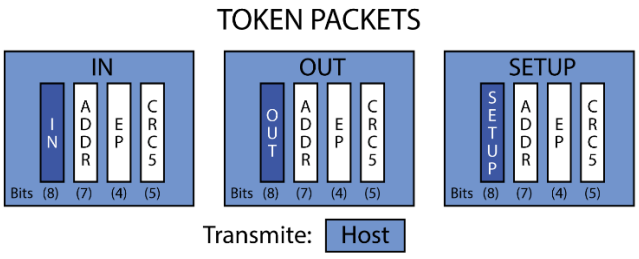
ii. Token Packets:

Los paquetes de tipo Token, son emitidos por el Host, desempeñan un papel esencial en la regulación de la dirección del flujo de datos dentro del bus USB (IN - OUT), configuración del dispositivo USB (SETUP) y el ritmo de la transmisión (SOF).

Los tokens de tipo IN se destinan a solicitar la transmisión de datos desde los dispositivos hacia el Host. En contraste, los tokens OUT se emiten para informar al dispositivo que el Host enviara datos.

Los tokens SETUP, indican al dispositivo que se enviara un comando de configuración por parte del Host.

Figura 31
Estructura de los paquetes Token.



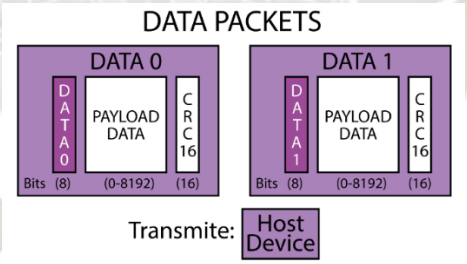
Nota: La figura muestra la estructura de los paquetes de tipo Token, estos paquetes son enviados por el Host, ya que ellos son los que controlan el flujo de la comunicación. Fuente: Modificado de Cypress Technologies.

La estructura de los paquetes Token IN, OUT y SETUP, se especifica una dirección de dispositivo de 7 bits, un identificador de End Point de 4 bits y un código de verificación de redundancia cíclica (CRC) de 5 bits.

iii. Data Packets:

Los paquetes de tipo dato, pueden ser emitidos por el Host y dispositivo USB, contienen información específica destinada a ser transferida entre el Host y el dispositivo USB, hay dos tipos de este paquete, “Data 0” y “Data 1”. Son la respuesta a los paquetes Token IN, OUT o SETUP. Estructuralmente, estos paquetes tienen una carga útil (Payload) de un tamaño que puede variar desde 0 hasta 1024 bytes, dependiendo del tipo de transferencia. El ID del paquete alterna entre DATA0 y DATA1 por cada transferencia exitosa de paquete de datos, y el paquete se cierra con un CRC de 16 bits.

Figura 32
Estructura de los paquetes Data.



Nota: La figura muestra la estructura de los paquetes de Datos, estos paquetes son enviados por el Host y por el dispositivo USB, es en este tipo de paquetes en los que se transmite la información útil de la aplicación relacionada al dispositivo USB. Fuente: Modificado de Cypress Technologies.

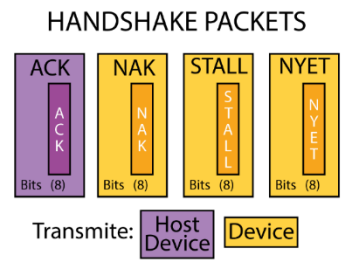
La alternancia del PID del paquete de Dato, ayuda a asegurar la integridad de la transmisión, permitiendo tanto al host como al dispositivo detectar si un paquete ha sido perdido o si ha ocurrido un error de transmisión. La utilización de DATA0 y DATA1 facilita también la implementación del mecanismo de control de flujo y la detección de errores.

Un ejemplo donde se utiliza el cambio de datos es si se envía un ACK pero no se recibe, en este caso, el emisor actualiza el cambio de datos de '1' a '0', pero el receptor no, el receptor permanece en '1'. Esto causa que el host y el dispositivo estén desincronizados en la siguiente etapa de datos, lo que indica un error.

iv. Handshake Packets:

Los paquetes de tipo Handshake finalizan cada transacción exitosa o errónea, pueden ser transmitidas generalmente por el dispositivo USB, sin embargo, el Host también las emite en ciertas circunstancias. Estructuralmente solo se componen de los bits del paquete que lo identifica (8 bits).

Figura 33
Estructura de los paquetes Handshake.



Nota: La figura muestra la estructura de los paquetes de tipo Handshake, estos paquetes son enviados por el Host y por el dispositivo USB, estos paquetes se envían como respuesta ante un envió de datos previo. Fuente: Modificado de Cypress Technologies.

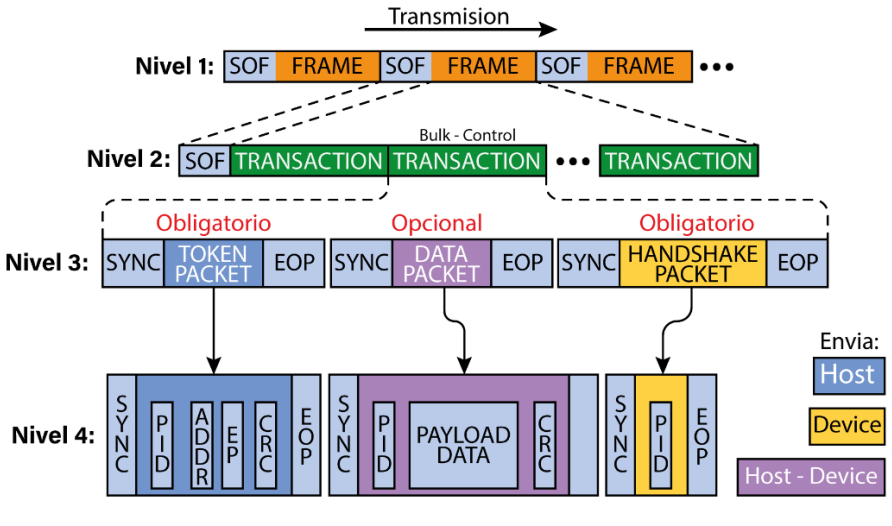
- ACK: Reconocimiento de finalización exitosa. (Emitida por el Host y dispositivo USB)
- NAK: Reconocimiento negativo. (Emitida por el dispositivo USB)
- STALL: Indicación de error enviada por un dispositivo. (Emitida por el dispositivo USB)
- NYET: Indica que el dispositivo no está listo para recibir otro paquete de datos. (Emitida por el dispositivo USB en una configuración de High Speed)

C. Flujo de una Transmisión USB

Al examinar la comunicación USB en la forma en cómo se realiza una transmisión, se observa que esta transmisión se compone de 4 niveles estructurales que están estandarizados en el protocolo USB, esto se puede observar en la **Figura 34**.

Figura 34

Estructura de la comunicación USB



Nota: La figura muestra la estructura general de la comunicación USB, siendo la transmisión de los paquetes, la unidad mínima en este tipo de comunicación. Fuente: Modificado de Cypress Technologies.

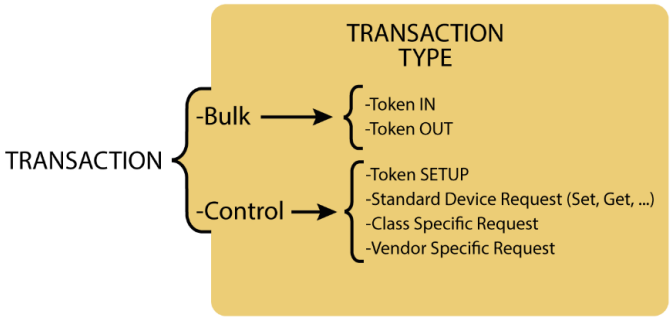
Observando la **Figura 34**, se puede identificar que en el nivel 1 hay una estructura global denominada “Frame” (Marco), la importancia de esta estructura es que, en su inicio, lleva un patrón de señal denominado “Start of Frame-SOF” (Inicio de Marco), este patrón cumple dos funciones importantes, en primer lugar, como su nombre lo indica, marca el inicio de un nuevo “frame”, en segundo lugar, sirve para sincronizar los relojes de comunicación del Host y del dispositivo, permitiendo que se ajuste la velocidad de transmisión.

En el nivel 2 se observa que cada “Frame” está compuesto por una o varias estructuras denominadas “Transactions” (Transacciones), existen dos tipos de transacciones, de tipo Bulk y Control. Las transacciones de tipo Bulk se usan para intercambiar datos

propios del funcionamiento del dispositivo, mientras las transacciones de tipo Control, se usan para solicitar o establecer datos o configuraciones específicas del dispositivo.

Figura 35

Tipos de transacciones en el protocolo USB.

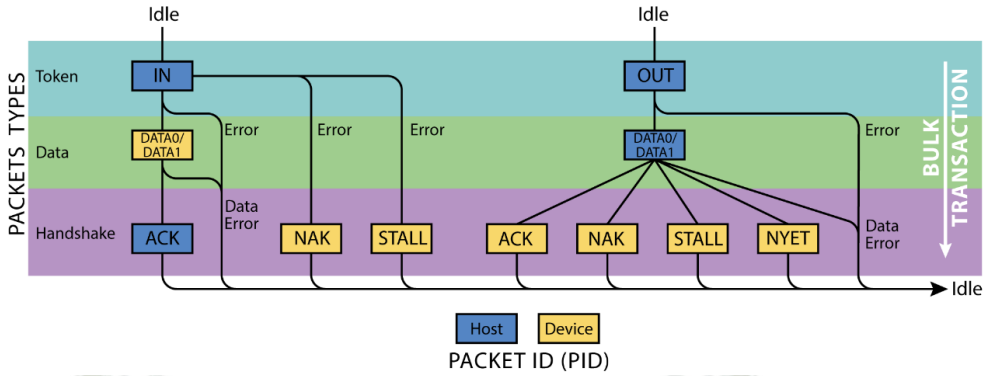


Nota: La figura muestra los tipos de Transacciones establecidas en la comunicación USB, los de tipo Bulk son las relacionadas directamente con el intercambio de datos para la aplicación del dispositivo USB, las de tipo Control, se usan para las configuraciones iniciales del dispositivo USB realizadas por el Host. Fuente propia.

En el nivel 3 se ven los paquetes (los cuales son la estructura básica del protocolo USB) que tiene la transacción, el tipo y la cantidad de paquetes dependerá del tipo y propósito de la transacción, los paquetes tipo Token y Handshake son siempre obligatorios y únicos, los paquetes tipo Data son opcionales, pudiendo tener desde ninguno hasta múltiples paquetes Data. En la **Figura 36** se muestra el flujo de los paquetes involucrados en una transacción de tipo Bulk, los paquetes Data, intercambian entre DATA0 y DATA1 a medida que se envían estos paquetes consecutivamente, esto con el fin de detectar alguna perdida de paquetes, sobre dicha Figura, se debe recordar que cada paquete se precede de un patrón de sincronización (SYNC) y concluye con un patrón de Fin de Paquete (End of Packet - EOP).

Figura 36

Flujo de paquetes en una Transacción tipo Bulk.

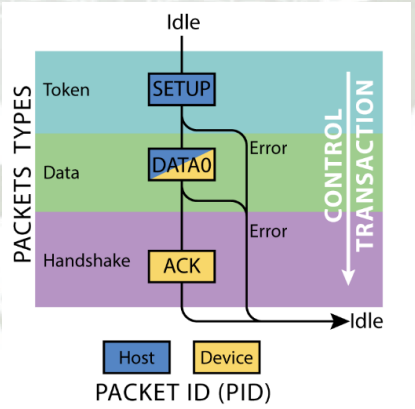


Nota: La figura muestra el flujograma para las transacciones de datos de la aplicación USB (Bulk), se puede observar que es el Host que controla el sentido de la comunicación, pidiendo o enviando información . Fuente: Modificado de Usb.org.

En la **Figura 37** se muestra el flujo de los paquetes involucrados en una transacción de tipo Control, los datos que se envían en este tipo de transacción, generalmente tienen espacio suficiente para ingresar en un solo paquete tipo Data,

Figura 37

Flujo de paquetes en una Transacción tipo Control



Nota: La figura muestra el flujograma para las transacciones de configuración inicial (Control), el Host pide información del estado del dispositivo y envía configuraciones al dispositivo USB. Fuente: Modificado de Usb.org.

Un ejemplo de Transacción de tipo Control, es él envió de una solicitud SET CONFIGURATION, el cual siempre se envía al iniciar el proceso de enumeración, cuando el dispositivo se conecta por primera vez. El Host envía un paquete Token SETUP, indicando la dirección del dispositivo (0 si es recién conectado, diferente de 0 si el

dispositivo ya tiene dirección única) y el End Point al que va dirigido (generalmente es el End Point 0 para solicitudes de control), luego el Host envía la configuración que desea activar (al iniciar la enumeración indica que activa la configuración 1, ya que es la primera y muchas veces la única) dentro de un paquete Data, luego, el dispositivo responde un ACK y ejecuta la orden que le dio el Host, el flujo es similar para para solicitudes SET ADDRESS, GET DESCRIPTOR, etc.

En el nivel 4 se observa su estructura interna, de todos los campos que integran un Paquete, únicamente el "Packet Identifier Name" es obligatorio, es decir, en función de la naturaleza de un Paquete, es posible que algunos paquetes incluyan o prescindan de dirección (ADDR), numero de "End Point" (EP), datos útiles (Payload) y bits de verificación (CRC).

3.3.1. USB Device Framework

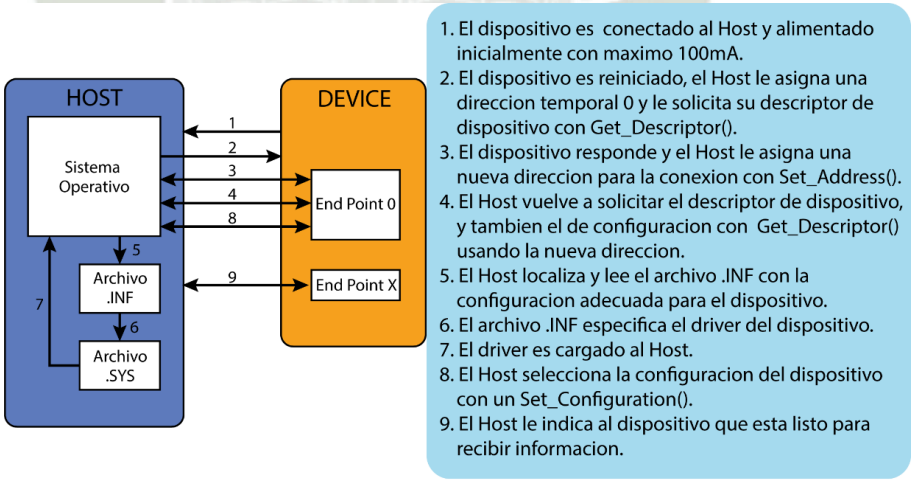
En el ámbito de los dispositivos USB, por ejemplo, el "USB Device Framework" es parte de la especificación USB que define cómo los dispositivos USB deben ser detectados, configurados y gestionados por un host. Incluye descripciones de cómo se deben estructurar los dispositivos USB, cómo deben comunicarse y cómo el sistema operativo y los controladores deben interactuar con ellos. Este marco establece los requisitos para los perfiles de dispositivos, las clases de dispositivos, y la forma en que se deben manejar las transferencias de datos y los comandos de control.

En resumen, un Device Framework proporciona las reglas y procedimientos necesarios para asegurar que los dispositivos puedan comunicarse efectivamente con otros componentes de un sistema, ofreciendo una arquitectura común que facilita su desarrollo, implementación y gestión.

A. Enumeración

Cuando un dispositivo USB se conecta por primera vez a un host, se inicia el proceso de enumeración USB. La enumeración es el proceso de intercambio de información entre el dispositivo y el host, que incluye aprender acerca del dispositivo. Además, la enumeración incluye asignar una dirección al dispositivo, leer descriptores (que son estructuras de datos que proporcionan información sobre el dispositivo) y asignar y cargar un controlador de dispositivo. Este proceso completo puede ocurrir en cuestión de segundos. Para obtener más información, consulte la sección de Enumeración y Configuración USB. Cuando este proceso está completo, el dispositivo está listo para transferir datos al host. El diagrama de flujo del proceso general de enumeración se muestra en la Figura 2. Dos archivos están asociados con la enumeración y la carga de un controlador. Estos archivos existen en el lado del host (Murphy, 2010).

Figura 38
Proceso de Enumeración.



Nota: La figura muestra el proceso de configuración inicial (Enumeración) que realiza el Host al dispositivo USB . Fuente: Modificado de Cypress Technologies.

B. Solicitudes Estándar de Dispositivo

Los "Standard Device Requests" son un conjunto de solicitudes estandarizadas definidas en la especificación USB que el host puede enviar a cualquier dispositivo USB

para realizar operaciones básicas de configuración, control y estado. Estas solicitudes son fundamentales para la comunicación inicial entre el host (como una computadora) y el dispositivo USB, permitiendo al host identificar, configurar y gestionar dispositivos USB de manera uniforme.

Las solicitudes estándar del dispositivo incluyen operaciones como obtener la descripción del dispositivo, seleccionar una configuración, configurar el estado de suspensión y despertar del dispositivo, y otras tareas de gestión de dispositivos. Se comunican a través de la estructura de paquete de control USB, que tiene campos específicos para el tipo de solicitud, el receptor de la solicitud, el valor y el índice de la solicitud, y la longitud de los datos a transferir. (Thomas E. , 2002)

Tabla 3
Solicitudes estándar de configuración de dispositivo USB.

Solicitudes	**Valor**
GET_STATUS	0
CLEAR_FEATURE	1
SET_FEATURE	3
SET_ADDRESS	5
GET_DESCRIPTOR	6
SET_ DESCRIPTOR	7
GET_CONFIGURATION	8
SET_ CONFIGURATION	9
GET_INTERFACE	10
SET_ INTERFACE	11
SYNCH_FRAME	12

i. Set Configuration

La solicitud SET CONFIGURATION le dice al dispositivo USB que active una de sus configuraciones soportadas. Cada configuración puede asociar el dispositivo con un conjunto de interfaces y capacidades de energía específicas.

Al inicio del proceso de Enumeración de un dispositivo USB, como primer paso el Host le indica al dispositivo que active su primera configuración (generalmente es la única),

de esta manera el dispositivo se prepara para enviar toda su información con la que está configurada su función de operación.

ii. Get Descriptor

Se utiliza para que el host solicite varios tipos de descriptores del dispositivo USB, empezando por el Device y Configuration, se envía una solicitud para cada tipo de Descriptor.

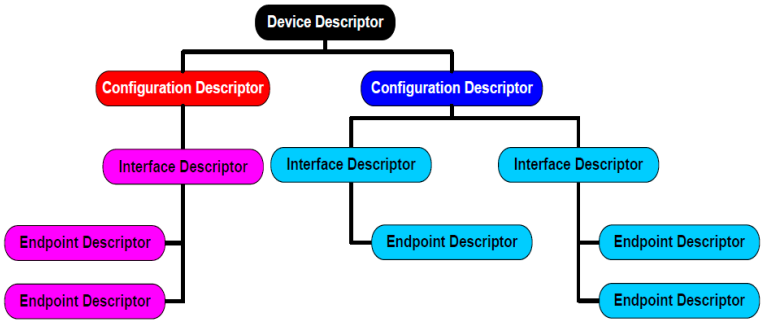
El dispositivo debe de responder cada solicitud, al responder la solicitud de descriptor de configuración, el dispositivo envía todos los descriptores que están asociados al descriptor de configuración, es decir, el descriptor de Interface y de Punto Final. Adicionalmente, si el dispositivo está configurado como clase HID, el dispositivo, luego de enviar el descriptor de Interface, enviara descriptor adicional especifico de la clase HID, el cual se llama HID Report Descriptor.

C. Descriptores

Un descriptor es una estructura de datos que proporciona información sobre el dispositivo USB y sus características a un host. Los descriptores se utilizan durante el proceso de enumeración, cuando un dispositivo USB se conecta a un host por primera vez, y sirven para identificar y configurar el dispositivo correctamente para la comunicación. Cada descriptor consta de varios campos que describen atributos específicos del dispositivo o de sus componentes.

Figura 39

Árbol de descriptores de un dispositivo que tiene dos configuraciones posibles.



Nota: La figura muestra que un dispositivo USB puede tener asociado varias formar de funcionamiento (configuraciones) dependiendo de la aplicación del mismo, en el proceso de enumeración se selección una predeterminada, pero en el transcurso del funcionamiento se puede activar las demás configuraciones según la aplicación . Fuente: Cypress Technologies.

i. Device Descriptor

Este descriptor proporciona información global y básica sobre el dispositivo al host y es único en un dispositivo USB. Incluye detalles como el ID del fabricante, el ID del producto, la versión del dispositivo, el número de posibles configuraciones que el dispositivo soporta, y otros datos esenciales que identifican al dispositivo de manera única.

Tabla 4

Descriptor de dispositivo.

Campo	Tamaño (Bytes)	Descripción
bDescriptorType	1	Indica el tipo de descriptor que se está proporcionando. Device (01h)
bcdUSB	2	Muestra la versión de USB del dispositivo. USB 2.0 (0200h)
bDeviceClass	1	Indica la clase del dispositivo. Generalmente se indica la clase en la Interface Descriptor, por lo que este campo se deja en (00h). Ver ANEXO A.
bDeviceSubClass	1	Indica la subclase del dispositivo. Determina el controlador de software que el Host debe utilizar. Ver ANEXO A
idVendor	2	Proporciona el ID del fabricante del dispositivo, este ID es asignado por el USB-IF.

Campo	Tamaño (Bytes)	Descripción
idProduct	2	Indica el ID del producto, este es asignado por el fabricante
bNumConfigurations	1	Indica el número de configuraciones que tiene el dispositivo.

i.i. HID Report Descriptor

Cuando se indica que el dispositivo es de Clase HID, es obligatorio generar un reporte específico para esta clase de dispositivo, el cual se llama HID Report Descriptor. Este descriptor entrega información más detallada de cómo funciona el dispositivo, en términos más didácticos se puede asumir que el HID Report Descriptor, es una plantilla que describe la estructura de los datos que se intercambian entre el Host y el dispositivo.

ii. Configuration Descriptor

Cada dispositivo USB puede tener varias configuraciones posibles, y para cada configuración, hay un Configuration Descriptor Un dispositivo puede cambiar entre configuraciones bajo el comando del host, pero solo puede operar en una configuración a la vez.

Tabla 5
Ejemplos de dispositivos con varios descriptores de configuración.

Dispositivos con un solo Configuration Descriptor	Dispositivos con múltiples Configuration Descriptors
1. **Memorias USB (Flash Drives):** Estos dispositivos de almacenamiento masivo son simples en términos de configuración USB y generalmente solo necesitan una configuración para operar. 2. **Ratones y Teclados USB:** Los dispositivos de entrada básicos como ratones y teclados suelen tener una única configuración que describe su funcionalidad específica de entrada.	1. **Dispositivos USB Compuestos:** Como un dispositivo que combina un teclado y un ratón en una sola unidad USB. Puede tener diferentes configuraciones para optimizar el consumo de energía o habilitar/deshabilitar conjuntos de funciones específicas. 2. **Teléfonos Móviles y Tablets:** Estos dispositivos pueden ofrecer múltiples configuraciones para diferentes modos de operación, como montaje de medios, modo de

Dispositivos con un solo Configuration Descriptor	Dispositivos con múltiples Configuration Descriptors
3. **Cámaras Web USB:** A menudo vienen con una configuración estándar para la captura de video y audio, simplificando su uso y compatibilidad. 4. **Auriculares USB y Micrófonos:** Estos dispositivos de audio generalmente ofrecen una configuración única para su operación, facilitando la gestión de audio.	depuración USB, o carga únicamente. 3. **Impresoras Multifunción:** Dispositivos que ofrecen funciones de impresión, escaneo y fax. Pueden tener configuraciones para diferentes modos de operación o para optimizar el consumo de energía cuando ciertas funciones no son necesarias.

El descriptor de configuración describe una configuración específica del dispositivo, incluyendo información sobre el consumo de energía de esa configuración, si el dispositivo es autoalimentado o alimentado por el bus, el número de interfaces que contiene esa configuración, y otros detalles relevantes.

Tabla 6
Descriptor de configuración.

Campo	Tamaño (Bytes)	Descripción
bDescriptorType	1	Indica el tipo de descriptor que se está proporcionando. Configuration (02h)
bNumInterfaces	1	Indica el número de interfaces que tiene la confirmación.
bmAttributes	1	Este campo especifica los atributos de la configuración. Bit 6: Self Powered. Bit 5: Remote Wakeup
bMaxPower	1	Indica el consumo máximo de energía del dispositivo USB en esta configuración. El valor es dado en unidades de 2mA. (50=100mA)

iii. Interface Descriptor

El Descriptor de Interfaz proporciona información sobre una interfaz específica dentro de una configuración de dispositivo USB. Cada interfaz puede ser una tarea específica

(por ejemplo, una interfaz de teclado en un dispositivo compuesto que también incluye un ratón). Aquí están los campos de un Descriptor de Interfaz y sus propósitos:

Tabla 7.
Descriptor de interface.

Campo	Tamaño (Bytes)	Descripción
bDescriptorType	1	Indica el tipo de descriptor que se está proporcionando. Interface (04h)
bInterfaceNumber	1	En este campo se asigna un numero único la interfaz dentro de la configuración, para poder identificarla en caso existan dos o más interfaces.
bNumEndpoints	1	Indica el número de EndPoints que usa esta interface, sin contar el EndPoint0.
bInterfaceClass	1	Define la clase de la Interfaz. Indica al Host que tipo de funcionalidad proporciona, así, el Host puede elegir entre un driver estándar o una especifica. Ver ANEXO A.

iv. End Point Descriptor

El Descriptor de Endpoint en el protocolo USB especifica las características de un endpoint particular de una interfaz de dispositivo USB.

A continuación, se detallan los campos de un Descriptor de Endpoint y sus funciones:

Tabla 8
Descriptor de punto final.

Campo	Tamaño (Bytes)	Descripción
bDescriptorType	1	Indica el tipo de descriptor que se está proporcionando. Endpoint (05h)
bEndpointAddress	1	Indica la dirección del EndPoint y la dirección del flujo de datos. Bit<3:0>: Numero de EndPoint. Bit<7>:Dirección del flujo.
bmAttributes	1	Describe el tipo transferencia en la que se usara el EndPoint (control, isocrónico, bulk, interrupción).

wMaxPacketSize	2	Indica el tamaño máximo de paquete que el EndPoint puede manejar. Esto es importante para planificar el ancho de banda del bus USB

v. String Descriptor

El Descriptor de Cadena (String Descriptor) en el protocolo USB proporciona información legible por humanos en forma de cadenas de texto. Estas cadenas se utilizan para describir información sobre el dispositivo, como el fabricante, el producto, y el número de serie. Los Descriptores de Cadena son opcionales, excepto en ciertos casos donde se requiere el número de serie. Aquí están los campos de un Descriptor de Cadena y sus propósitos:

Tabla 9
Descriptor de texto.

Campo	Tamaño (Bytes)	Descripción
bDescriptorType	1	Indica el tipo de descriptor que se está proporcionando. String (03h)
bString	Varia	Es una cadena de caracteres Unicode. Cada carácter se codifica en UTL-16LE, lo que significa que cada carácter utiliza 2 bytes. Suele llevar el nombre del fabricante, el nombre del producto, número de serie, etc.

3.4. Filtros Digitales

3.4.1. Sistemas Lineales Invariantes en el Tiempo

Un sistema se puede definir como un "grupo de elementos que interactúan regularmente o son interdependientes, y forman un todo unificado". En el contexto de procesamiento de señales, un sistema se define como un proceso en el que una señal llamada entrada se transforma en otra señal llamada salida.

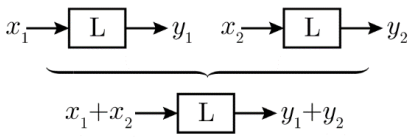
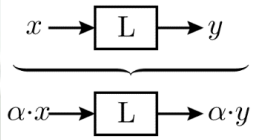
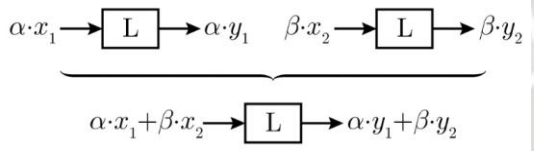
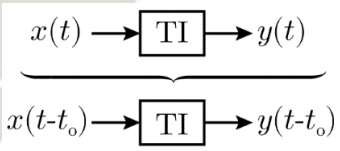
Los sistemas Lineales Invariantes en el Tiempo (LTI) son fundamentales en el análisis y diseño de sistemas de señales y control. La teoría de sistemas LTI se aplica en una

variedad de campos, incluyendo la ingeniería eléctrica, la física, el procesamiento de señales y la teoría de control (Hsu, 2013).

Un Sistema Lineal Invariante en el Tiempo (LTI) es un sistema en el que la salida es una respuesta lineal a la entrada y cuyas propiedades no cambian con el tiempo. Esto implica dos características clave: linealidad e invariancia en el tiempo. (Aparicio, 2007)

Tabla 10

Características principales de un sistema LTI.

Sistema Lineal (L)	
Superposición	**Homogeneidad**
Si $x_1(t)$ produce una salida $y_1(t)$ y $x_2(t)$ produce una salida $y_2(t)$; entonces: $x_1(t) + x_2(t)$ producirá $y_1(t) + y_2(t)$ 	Si $x(t)$ produce una salida $y(t)$, entonces, una entrada escalada $\alpha \cdot x(t)$ producirá una salida escalada $\alpha \cdot y(t)$ 
	
Sistema Invariante en el Tiempo (TI) Si un entrada $x(t)$ produce una salida $y(t)$ entonces, una entradsa desplazada en el tiempo $x(t - t_o)$ producirá una salida desplazada en el tiempo $y(t - t_o)$.	

Otra de las propiedades importantes de los sistemas LTI son la causalidad y la estabilidad:

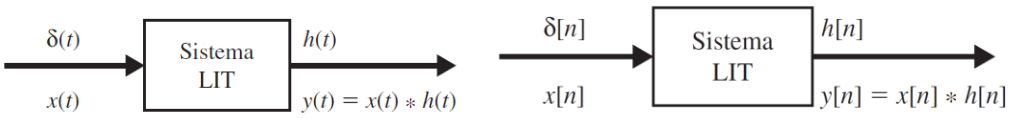
- Causalidad: Un sistema es causal si la salida en cualquier tiempo depende solo de valores presentes y pasados de la entrada, pero no de valores futuros.
- Estabilidad: Un sistema es BIBO (Bounded Input, Bounded Output) estable si, para cualquier entrada acotada, la salida también es acotada.

La relación entre la entrada y la salida en un sistema LTI se describe en términos de una operación de convolución. La operación de convolución es crucial porque, conocer la

respuesta de un sistema LTI a la entrada de impulso unitario (*h(t)*), nos permite determinar su salida (*y(t)*) para cualquier señal de entrada (*x(t)*) (Hsu, 2013), en términos más simples, la convolución nos permite analizar como una señal de entrada se modifica al pasar a través de un sistema representado por su respuesta al impulso.

Figura 40

Sistema LTI en tiempo continuo y discreto.



Nota: La figura muestra el comportamiento más importante de los sistemas LTI, la respuesta al impulso unitario del sistema, la cual es fundamental para describir completamente el comportamiento del sistema. Fuente propia.

El operador convolución (*) es muy importante para el procesamiento de señales, este operador cumple las propiedades básicas de conmutación, asociación y distribución. Sin embargo, la propiedad más importante de esta operación, relacionada a la aplicación de filtros digitales, se da en conjunto a la Transformada de Fourier, esta propiedad relaciona el comportamiento de un sistema LTI en el campo temporal con su comportamiento en el campo frecuencial, estas propiedades se muestran en la Tabla 11

Tabla 11

Propiedades a destacar de la convolución.

Propiedad	Continuo	Discreto
Definición	$y(t) = x(t) * h(t) = \int x(\tau)h(t-\tau)\,d\tau$	$y[n] = x[n] * h[n] = \sum x[k]h[n-k]$
Convolución en la Transformada de Fourier (filtros digitales)	$x_1(t) * x_2(t) \leftrightarrow X_1(\omega)X_2(\omega)$	$x_1[n] * x_2[n] \leftrightarrow X_1[\Omega]X_2[\Omega]$
Multiplicación (modulación) en la Transformada de Fourier	$x_1(t)x_2(t) \leftrightarrow \frac{1}{2\pi}X_1(\omega) * X_2(\omega)$	$x_1[n]x_2[n] \leftrightarrow \frac{1}{2\pi}X_1[\Omega] * X_2[\Omega]$

Nota: Fuente (Hsu, 2013)

A. Filtros Digitales

Los filtros digitales son implementaciones específicas de sistemas LTI en el procesamiento digital de señales, cada uno con características, ventajas y aplicaciones particulares que derivan directamente de las propiedades de los sistemas LTI. (Manolakis & Ingle, 2011)

- Convolución: Ambos utilizan la convolución para determinar la salida del sistema. En FIR, la convolución es directa con la respuesta finita al impulso. En IIR, la convolución incluye términos de retroalimentación.
- Superposición: La salida de ambos filtros para una combinación lineal de entradas es la combinación lineal de las salidas respectivas.
- Invariancia en el Tiempo: La respuesta de los filtros FIR e IIR no cambia con el tiempo, manteniendo sus características independientemente del momento en que se aplique la entrada.

Para determinar el comportamiento de las funciones de transferencia de un filtro, existen dos tipos principales de filtros digitales: FIR e IIR, cada uno con características específicas.

- Filtros FIR (Finite Impulse Response): Dependiendo únicamente de los valores pasados de la entrada, tienen una respuesta finita al impulso y son inherentemente estables. Además, pueden ser diseñados para tener una fase lineal exacta. Sin embargo, suelen requerir más coeficientes para lograr la misma selectividad que un filtro IIR.
- Filtros IIR (Infinite Impulse Response): Utilizan valores pasados tanto de la entrada como de la salida, lo que permite una alta selectividad con un número reducido de

coeficientes. No obstante, estos filtros pueden no tener una fase lineal y requieren un diseño cuidadoso para evitar inestabilidades.

Existen tres autores principales que definen los coeficientes para lograr comportamientos específicos en los filtros IIR:

- Filtros Butterworth: Caracterizados por una respuesta de magnitud máximamente plana en la banda de paso, sin ondulaciones. Son conocidos por su suavidad en la transición y la ausencia de ripples en la respuesta de magnitud.
- Filtros Chebyshev: Ofrecen una transición más abrupta entre la banda de paso y la banda de parada, con ondulaciones controladas en la banda de paso (Chebyshev Tipo I) o en la banda de parada (Chebyshev Tipo II). Estos filtros no tienen fase lineal y se utilizan cuando se necesita una alta selectividad.
- Filtros de Bessel: Son conocidos por su control de fase y minimización de distorsiones de fase, manteniendo un retraso de grupo constante. Estos filtros son ideales para aplicaciones donde es crucial preservar la forma de la señal en el tiempo, aunque no son tan selectivos en términos de respuesta en frecuencia.

Como filtro IIR, adicionalmente, se tienen a los IIR Bilineales, estos son una categoría que se obtienen mediante la transformación bilineal, un método que convierte filtros analógicos en filtros digitales. Esta transformación preserva la estabilidad y mapea el eje jω del plano s en el círculo unitario del plano z, evitando efectos de aliasing. Se utilizan ampliamente debido a su capacidad para mantener las características de los filtros analógicos originales.

Tabla 12
Comparación entre filtros FIR e IIR.

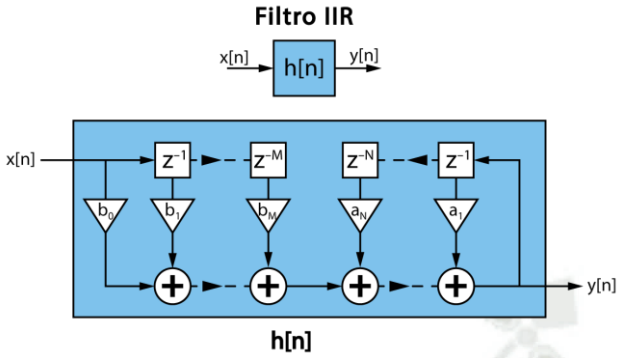
Característica	FIR	IIR
Respuesta al Impulso	Finita	Infinita
Estabilidad	Siempre estables	Pueden ser inestables
Linealidad de Fase	Logran la linealidad de fase	Difícil de lograr la linealidad de fase
Memoria y requerimiento computacional	Requieren más memoria y recursos de computación.	Menos memoria y computación para una misma especificación.
Causalidad	Siempre causales	Pueden ser causales y no causales
Diseño	Mas sencillo y directo	Mas complejo debido a los polos en el plano z
Coeficientes	Términos de numerador	Términos de numerador y denominador
Representación analógica	Son únicamente digitales	Pueden representar filtros analógicos
Categorías	Media móvil	Butterworth Chebyshev Bessel Bilineal

3.4.2. Filtro IIR

Los filtros IIR (Respuesta Infinita al Impulso) son un tipo de filtro digital que se caracterizan por tener una respuesta al impulso que teóricamente se extiende hasta el infinito. Estos filtros pueden responder indefinidamente a una entrada de impulso debido a la presencia de realimentación en su estructura. Esto significa que los filtros IIR utilizan valores de salida anteriores además de las entradas actuales y anteriores para calcular su salida, siendo la retroalimentación una característica importante de este tipo de filtros. Generalmente no tienen fase lineal. (John G. Proakis, 2007)

Figura 41

Diagrama de bloques de un filtro IIR.



Nota: La figura muestra el diagrama de bloques de un filtro IIR, en esta se puede observar que la salida del sistema depende de las salidas anteriores, es decir, tiene retro alimentación . Fuente propia.

La estructura de la función de transferencia de los filtros IIR cuenta con ceros y polos, a diferencia de los filtros FIR que únicamente cuenta con ceros, se puede observar que la forma directa de los filtros IIR incluyen a los sistemas FIR en su numerador.

$$H(z) = \frac{\sum_{k=0}^{M} b_k z^{-k}}{1 - \sum_{k=1}^{N} a_k z^{-k}}$$

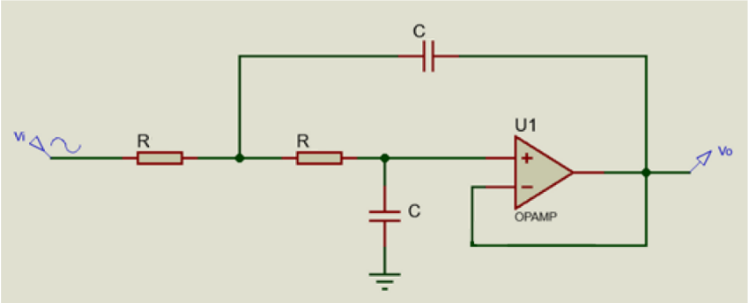
Debido a la realimentación, la estabilidad de los filtros IIR es una consideración crucial al momento de diseñarlos. Un filtro IIR es estable si todos los polos de su función de transferencia se encuentran dentro del círculo unitario en el plano z.

A. Representación de Filtros Analógicos

Una de las características principales de los filtros IIR, es su capacidad de poder implementar filtros analógicos, aplicando una conversión del plano S al plano Z.

Figura 42

Filtro analógico pasa bajo Sallen Key.



Nota: Diagrama eléctrico de un filtro pasa bajos de segundo orden, Sallen Key. Fuente propia

De la ecuación (6) que está en el 0 se obtiene la función de transferencia del filtro analógico Sallen Key de segundo orden

$$H(s) = \frac{\left(\frac{1}{C.R}\right)^2}{s^2 + 2\left(\frac{1}{C.R}\right).s + \left(\frac{1}{C.R}\right)^2} = \frac{a^2}{s^2 + 2 \cdot a \cdot s + a^2}$$

Como se mencionó párrafos arriba, un método para convertir del plano S al plano Z, es aplicar la transformada bilineal:

$$s \approx \frac{2}{T_s} \cdot \frac{1 - z^{-1}}{1 + z^{-1}}$$

$$H(z) = H(s)|_{s=\frac{2}{T_s}\cdot\frac{1-z^{-1}}{1+z^{-1}}}$$

$$H(z) = \frac{a^2 \cdot {T_s}^2}{(a \cdot T_s + 2)^2} \cdot \frac{1 + 2 \cdot z^{-1} + z^{-2}}{1 + 2\left(\frac{a \cdot T_s - 2}{a \cdot T_s + 2}\right) z^{-1} + \left(\frac{a \cdot T_s - 2}{a \cdot T_s + 2}\right)^2 z^{-2}}$$

$$H(z) = k \cdot \frac{1 + 2 \cdot z^{-1} + z^{-2}}{1 + 2\left(\frac{a \cdot T_s - 2}{a \cdot T_s + 2}\right) z^{-1} + \left(\frac{a \cdot T_s - 2}{a \cdot T_s + 2}\right)^2 z^{-2}}$$

Para comparar la respuesta en frecuencia del filtro Sallen Key analógico y digital (IIR), llevaremos la función de transferencia que calculamos a la forma básica de los filtros IIR:

$$H(z) = \frac{\sum_{k=0}^{M} b_k z^{-k}}{1 - \sum_{k=1}^{N} a_k z^{-k}} = \frac{b_0 + b_1 z^{-1} + b_2 z^{-2}}{1 + a_1 z^{-1} + a_2 z^{-2}}$$

$$H(z) = \frac{k + 2kz^{-1} + kz^{-2}}{1 + 2\left(\frac{a \cdot T_s - 2}{a \cdot T_s + 2}\right) z^{-1} + \left(\frac{a \cdot T_s - 2}{a \cdot T_s + 2}\right)^2 z^{-2}} = \frac{b_0 + b_1 z^{-1} + b_2 z^{-2}}{1 + a_1 z^{-1} + a_2 z^{-2}}$$

Definiéremos la frecuencia de muestreo f_s a 2000 Hz, esto nos permite trabajar en el dominio digital hasta una frecuencia de Nyquist f_{NYQ} igual a 1000 Hz y evaluaremos las funciones de transferencia para diferentes frecuencias de corte.

Tabla 13

Parámetros del filtro pasa bajos Sallen Key (f_s=2kHz).

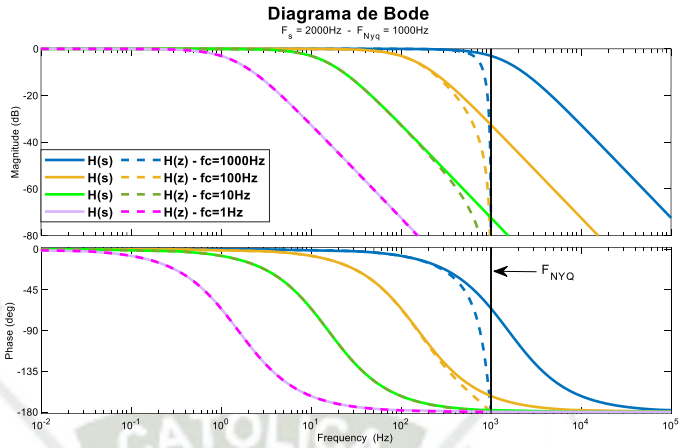
		H(s)		**H(z) - Orden 2**				
f_c (-3dB) (Hz)	f_c/f_{NYQ} (%)	C (µF)	R (Ω)	b_0	b_1	b_2	a_1	a_2
1000	100,00%	4,7	21,7939	0,50319	1,0064	0,5032	0,8374	0,1753
100	10,00%	4,7	217,9387	0,03849	0,0770	0,0385	-1,2153	0,3692
10	1,00%	4,7	2179,3874	0,00057	0,0011	0,0006	-1,9047	0,9070
1	0,10%	4,7	21793,8738	5,93E-06	1,19E-05	5,93E-06	-1,9903	0,9903

En la **Figura 43** se puede observar que la frecuencia de Nyquist limita el comportamiento del filtro Sallen Key IIR, mientras menor sea la frecuencia de corte respecto a la frecuencia de Nyquist, el filtro digital responderá casi de forma al filtro analógico. Para este caso particular, el filtro analógico se desempeña mejor que el digital ya que no está limitado a una frecuencia en particular, por lo que puede operar para cualquier frecuencia.

Una desventaja que se puede resaltar sobre el filtro analógico es que, los componentes físicos no son exactos ya que tienen un valor de tolerancia, además, que el resultado que se obtuvo para las resistencias, son valores que difícilmente se podrá conseguir en el mercado, esto no es un problema para su contraparte digital, ya que los valores de los coeficientes que determinan al filtro, son escritos directamente por software, por lo que son mucho más exactos que los analógicos.

Figura 43

Respuesta en frecuencia de filtro Sallen Key en H(s) y H(z).

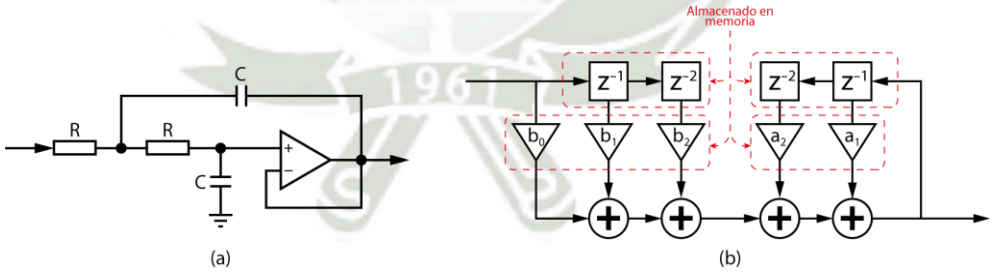


Nota: La figura muestra la comparación en magnitud y fase entre un filtro analógico de segundo orden y un filtro digital IIR de transformación bilineal, ambos en diferentes frecuencias de corte pero con una misma frecuencia de muestreo para el filtro digital, se observa que el filtro digital está limitado por la frecuencia de Nyquist. Fuente propia.

Por ese motivo, para este caso en particular, la elección entre un filtro analógico o digital dependerá directamente si la aplicación que se vaya a implementar, requiera un nivel de exactitud elevado.

Figura 44

Comparación contractiva de un filtro analógico y digital IIR.



Nota: La figura muestra constructivamente dos filtros equivalentes (a)un filtro analógico Sallen Key con componentes electrónicos. (b) un filtro digital IIR Sallen Key (bilineal) con su algoritmo computacional. Fuente propia.

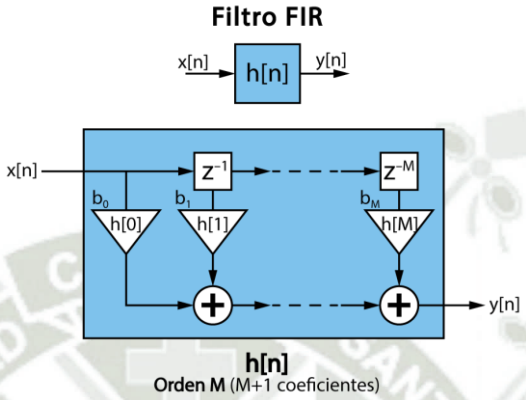
3.4.3. Filtro FIR

Los filtros FIR (Respuesta Finita al Impulso) son un tipo de filtro digital que se caracterizan por tener una respuesta al impulso de duración finita. Esto significa que, para

una entrada de impulso, el filtro responde con una salida que se anula tras el paso de un cierto número de muestras. (Sciencedirect, 2012)

Figura 45

Diagrama de Bloques de un filtro FIR



Nota: La figura muestra el diagrama de bloques de un filtro FIR, en esta se puede observar que la salida del sistema depende únicamente de las salidas pasadas, es decir, no tiene retro alimentación, por lo que se puede representar como una suma algebraica de términos . Fuente propia.

A diferencia de los filtros IIR, los filtros FIR se caracterizan por ser estable ya que únicamente cuentan con ceros, como se observa en su función de transferencia.

$$H(z) = \sum_{k=0}^{M} b_k z^{-k}$$

Otra característica importante de los filtros FIR es su respuesta frecuencial, el cual tiene un comportamiento lineal, esto es beneficioso para el filtrado de señales, a diferencia de los filtros IIR los cuales no cuentan con esta característica.

Para el diseño de filtros FIR se tienen diferentes técnicas, en esta investigación se estudiará y utilizará únicamente le método del Ventaneo.

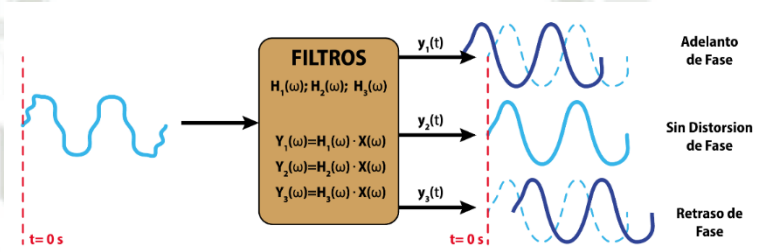
A. Fase Lineal y Retardo de Grupo

Cuando una señal pasa por un filtro, sin importar su tipo, la señal de salida sufre distorsiones en la amplitud, las cuales son deseadas, existiendo también las no deseadas

como la respuesta no ideal y las ondulaciones. Otra distorsión que sufre la señal son las distorsiones de fase, las cuales se les conoce como adelanto o retraso. Un filtro ideal tiene en retraso de cero radianes, sin embargo, un filtro real, por procesamiento computacional o limitaciones físicas, siempre genera un retraso en la señal de salida. (Sciencedirect, 2012)

Figura 46

Disposición en la respuesta de fase de los filtros

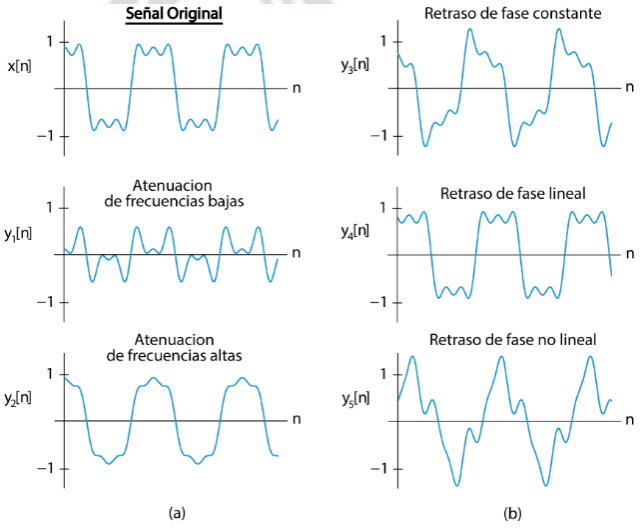


Nota: La figura muestra la distorsión de fase que sufren las señales al pasar por un filtro, pudiendo idealmente sufrir adelanto de fase o tener cero distorsión, y comúnmente sufrir retraso de fase. Fuente propia.

Una distorsión adicional que genera el retraso de fase, es la distorsión temporal que sufre la señal de salida. Debido a que el desface es diferente para cada frecuencia, todas las amplitudes para cada frecuencia se desplazan desordena mente, por lo que la morfología temporal de señal de salida distorsionada.

Figura 47

Efecto de las distorsiones de magnitud y fase en los filtros.



Nota: Se observa (a) la distorsión en magnitud, la cual esta relaciona al comportamiento del filtro, y (b) la distorsión por retraso de fase, esta última es una consecuencia del filtrado, por lo que la naturaleza del retraso de fase, si bien no tiene efecto en la magnitud de la señal, tiene un gran impacto en la forma de la misma. Fuente: (Manolakis & Ingle, 2011)

Lo más cercano que podemos tener a un filtro ideal de retraso constante de cero radianes en todas las frecuencias, son los filtros de fase lineal, los cuales tienen un retraso constante en todas las frecuencias, es decir, un filtro tiene fase lineal cuando su respuesta frecuencial $\varphi(\omega)$ de su función de transferencia $H(\omega)$, varia linealmente en proporción a la frecuencia angular ω, matemáticamente se puede representar la respuesta frecuencial con la ecuación de la recta: (Sciencedirect, 2012)

$$\varphi_L(\omega) = \alpha + \beta \cdot \omega$$

Donde:

$\varphi_L(\omega)$	:	Fase de la función de transferencia del filtro $H(\omega)$ con fase lineal.
α	:	Constante de fase, representa el desplazamiento inicial de fase en $\omega = 0$.
β	:	Pendiente de la variación de fase con respecto a la frecuencia. Si β es positivo, indica un retraso en el tiempo; si β es negativo, indica un adelanto en el tiempo.

La propiedad de tener una fase lineal en un filtro es importante ya que permite preservar la forma temporal de las señales de entrada al pasar a través del sistema, aunque agrega una distorsión de fase denominada Retardo de Grupo el cual es el más próximo a considerarse ideal.

El Retardo de Grupo de un filtro se refiere a la tasa de cambio de la fase del filtro con respecto a la frecuencia angular, el Retarde Grupo se representa como el negativo de la derivada de la fase del filtro con respecto a la frecuencia angular.

$$\tau_g(\omega) = -\frac{d\varphi(\omega)}{d\omega}$$

Si tomamos en cuenta que la fase de un filtro lineal tiene la ecuación de una recta, y que la derivada de una recta es su pendiente, podemos calcular que el Retraso de Grupo de un filtro de fase lineal es constante para toda frecuencia.

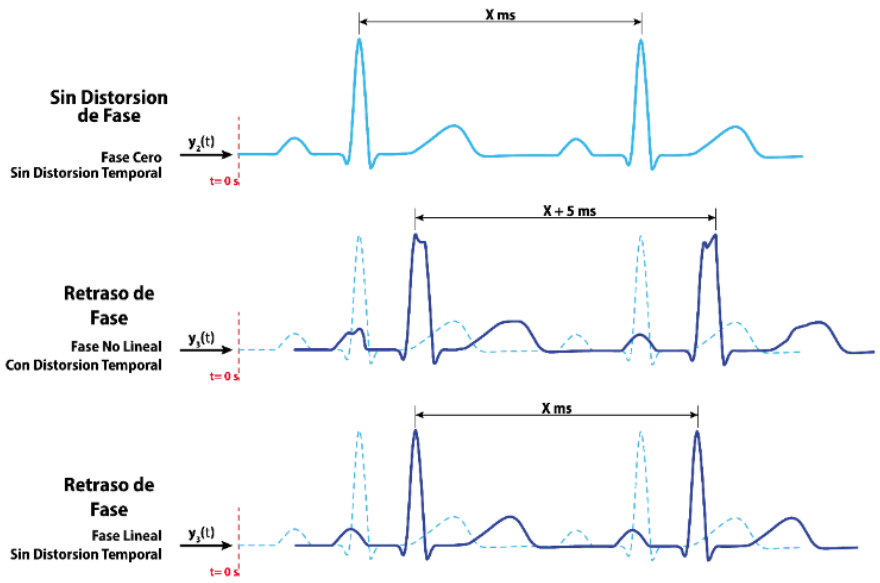
$$\tau_g(\omega) = -\frac{d\varphi_L(\omega)}{d\omega}$$

$$\tau_g(\omega) = -\beta$$

El concepto de retardo de grupo es especialmente relevante en sistemas de comunicaciones y procesamiento de señales, donde se desea que todas las componentes frecuenciales de una señal de entrada se retrasen por la misma cantidad de tiempo para evitar distorsiones temporales. Esto es crucial en el diseño de filtros y sistemas de comunicación, donde se busca preservar la integridad de la señal original tanto en amplitud como en fase. Un ejemplo ilustrativo de este caso se presenta en la **Figura 48**, en la cual se muestra una señal ECG, donde la distancia temporal entre los picos P, Q, R, S y T tienen un significado médico único, si no se toma en cuenta la distorsión de fase se induce a un diagnóstico erróneo en caso que la señal analizada este distorsionada temporalmente por efecto de un retardo de fase que no sea constante.

Figura 48

Distorsión de fase no lineal y lineal en una señal ECG.



Nota: Se muestra los efectos de la distorsión por retraso de fase de los filtros en una señal ECG, se puede observar que dependiendo del tipo de retraso de fase, la señal filtrada modifica su forma debido a una distorsión temporal, esto tiene como consecuencia una interpretación errónea de las señales ECG. Fuente propia.

Según el libro Tratamiento de señales en tiempo discreto (Alan V. Oppenheim, 2009), para que un filtro tenga fase lineal es necesario que sea causal y que la respuesta al impulso cumpla la condición de simetría, es decir, si se tiene una secuencia $h[n]$, que va desde $n: 0,1, \dots, M$, y la secuencia es simétrica con respecto al termino $M/2$, se puede afirmar que la respuesta del filtro ante cualquier entrada tendrá una fase lineal.

Tabla 14

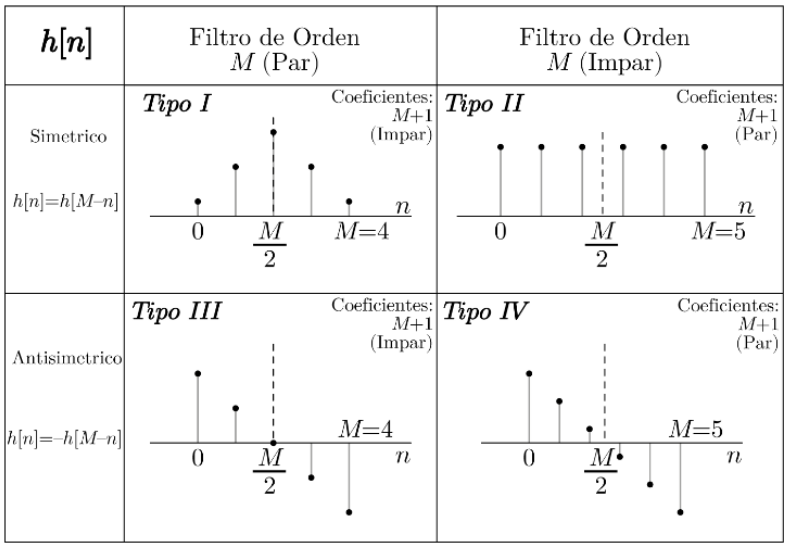
Condición de señales para la linealidad de fase de un filtro digital.

Respuesta Simétrica	**Respuesta Anti simétrica**
$h[n] = \begin{cases} h[M-n], & 0 \le n < M \\ 0, & en\ el\ resto \end{cases}$	$h[n] = \begin{cases} -h[M-n], & 0 \le n < M \\ 0, & en\ el\ resto \end{cases}$

En la **Figura 49** se muestra de forma más grafica la representación de las respuestas al impulso, $h[n]$, simétricas y antisimétricas, se debe observar que el efecto del orden de $h[n]$ no se ve afectado si este es par o impar. Se puede mencionar también que los filtros pasa-bajos son siempre simétricos, y que los filtros pasa-altos son filtros antisimétricos.

Figura 49

Respuesta al impulso con fase lineal., h[n]



Nota: La figura muestra la clasificación de los filtros FIR según el número de coeficientes y la simetría de sus coeficientes, el filtro tendrá retraso de fase lineal solo si se ubica en alguno de estos 4 tipos . Fuente propia.

Otra forma de comprender como un filtro FIR tiene una fase lineal, es analizando el lugar de sus ceros en el plano Z, iniciamos el análisis con el análisis de la función de transferencia del filtro y la condición de simetría

$$H(z) = \sum_{n=0}^{M} h[n]z^{-n} \;;\; h[n] = h[M-n]$$

$$H(z) = \sum_{n=0}^{M} h[M-n]z^{-n}$$

Aplicando las propiedades de la transformada z, desplazamiento en el tiempo y la inversión temporal, obtenemos la transformada z de un sistema que tiene fase lineal.

$$H(z) = z^{-M}H(z^{-1})$$

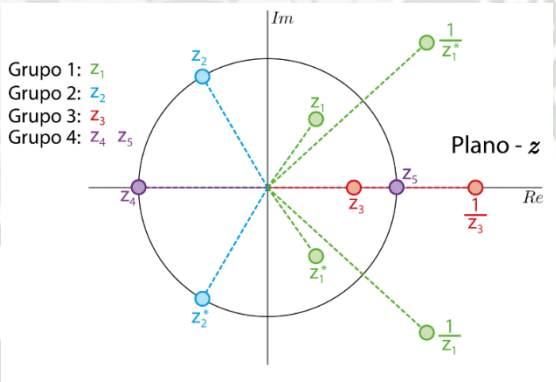
Si evaluamos la función de transferencia en cualquier cero $z_0 = re^{j\theta}$, evaluamos la función y empezamos el análisis

$$H(z_0) = {z_0}^{-M} H({z_0}^{-1}) = 0$$

Esto implica que si $z_0 = re^{j\theta}$ es un cero de $H(z)$, también lo será su inverso ${z_0}^{-1} = r^{-1}e^{-j\theta}$. Debido a que el filtro FIR es un sistema real, es decir todos sus coeficientes son números reales, todos los ceros del filtro deben de estar acompañados de su complejo conjugado, entonces ${z_0}^* = re^{-j\theta}$ también será un cero de la función de transferencia, evaluando este nuevo cero en $H(z)$, obtenemos que su inverso $({z_0}^*)^{-1} = r^{-1}e^{j\theta}$ también será un cero la función. Debemos acotar la ubicación del cero $z_0 = re^{j\theta}$ para poder entender el comportamiento de esta relación por lo que dividiremos todos los posibles casos en 4 grupos de detallaremos a continuación:

Figura 50

Simetría de los ceros con respecto a la circunferencia unitaria.



Nota: La figura muestra la distribución de los Ceros del filtro FIR con fase lineal alrededor del círculo unitario. Para lograr la fase lineal, el filtro debe tener sus ceros agrupados en uno o más de los grupos que se muestra . Fuente: modificado de (Alan V. Oppenheim, 2009).

- **Grupo 1:** $z_0 = re^{j\theta}$, es un cero que tiene parte real e imaginaria, y su modulo es diferente a 1, entonces su $H(z)$ seria:

$$\underset{z_0}{} \qquad \underset{{z_0}^{-1}}{} \qquad \underset{{z_0}^*}{} \qquad \underset{({z_0}^*)^{-1}}{}$$

$$H(z) = \left(1 - re^{j\theta}z^{-1}\right)\left(1 - r^{-1}e^{-j\theta}z^{-1}\right)\left(1 - re^{-j\theta}z^{-1}\right)\left(1 - r^{-1}e^{j\theta}z^{-1}\right)$$

- **Grupo 2:** $z_0 = 1e^{j\theta} = e^{j\theta}$, es un cero en la circunferencia unidad, su modulo es 1, por lo que los ceros inversos serian iguales, entonces su $H(z)$ seria:

$$H(z) = \underbrace{\left(1 - re^{j\theta}z^{-1}\right)}_{z_0}\underbrace{\left(1 - re^{-j\theta}z^{-1}\right)}_{z_0^{*}}$$

- **Grupo 3:** $z_0 = re^{j0} = r$, es un cero real, no tiene parte imaginaria, por lo que tampoco tiene una conjugada, entonces su $H(z)$ seria:

$$H(z) = \underbrace{\left(1 - re^{j\theta}z^{-1}\right)}_{z_0}\underbrace{\left(1 - r^{-1}e^{-j\theta}z^{-1}\right)}_{z_0^{-1}}$$

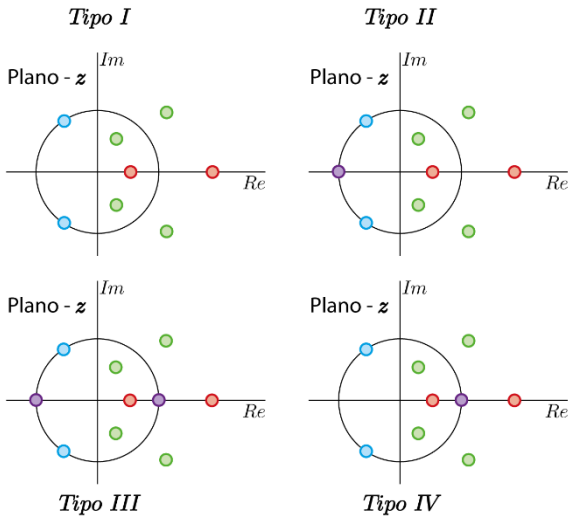
- **Grupo 4:** $z_0 = \pm 1$, es un cero de modulo 1 que no tiene parte imaginaria, por lo que no necesita inverso ni conjugada, por consiguiente, puede aparecer solo, entonces su $H(z)$ seria:

$$H(z) = \underbrace{(1 - 1z^{-1})}_{z_0}$$

Comprobamos que la ubicación de los ceros es importante para determinar que los filtros tengan fase lineal, como se mostró en la **Figura 49**, también se puede determinar si el filtro tiene fase lineal a partir de la secuencia de coeficientes de la respuesta al impulso del filtro, homólogamente, se puede replantear esa Figura desde una perspectiva de análisis de la ubicación de sus ceros, esta se muestra en la **Figura 51**.

Figura 51

Respuesta al impulso con fase lineal, H(z).



Nota: La figura muestra la relación entre el tipo del filtro FIR y la distribución de sus Ceros en el circulo unitario para alcanzar la fase lineal. Fuente: modificado de (Alan V. Oppenheim, 2009).

B. Cálculo de Coeficientes por el método del enventanado

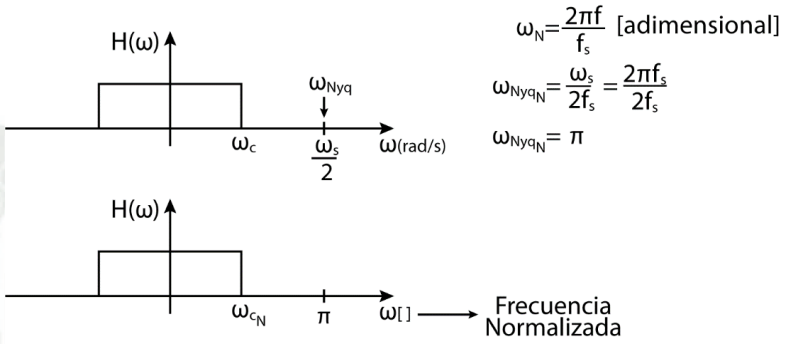
El siguiente punto en el análisis de filtros FIR, es el cálculo de los coeficientes de la respuesta ante al impulso y como estos afectan a la amplitud de frecuencias deseadas, así como la atenuación de frecuencias no deseadas. Teniendo en cuenta que los filtros FIR se aplican sobre señales digitales, es decir, una señal analógica pasa por un proceso de muestro, los coeficientes dependen directamente de la frecuencia de muestreo y de la frecuencia de Nyquist. Para poder diseñar filtros que puedan ser aplicados sin importar la frecuencia de muestreo que tenga el sistema, se debe de normalizar las frecuencias con las que se diseñan los filtros FIR.

El proceso de normalización de la frecuencia consiste en dividir todas las frecuencias entre la frecuencia de muestreo, adicionalmente, se debe de tener presente la frecuencia de Nyquist, el cual limita las frecuencias a las que el sistema puede responder correctamente. Luego de realizar la normalización de la frecuencia, como se realizó en la **Figura 52**, se

observa que el filtro se diseñará teniendo como limite el valor de π, por lo que la frecuencia de corte solo podrá ser seleccionada hasta un valor de π. (Alan V. Oppenheim, 2009)

Figura 52

Normalización de la frecuencia.



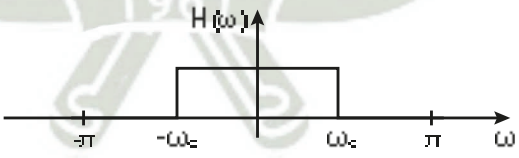
Nota: La figura muestra el proceso de normalización de la frecuencia. Se debe notar que la escala de la frecuencia normalizada es adimensional. Fuente propia.

El primer paso en el diseño y cálculo de los filtros es determinar la respuesta en frecuencia ideal (h_d, H_d) que deseamos, para nuestro caso, buscamos aplicar un filtro pasa bajos, por lo que empezaremos a modelar

Figura 53

Respuesta en frecuencia ideal de un filtro pasa bajos.

$$H_d(e^{j\omega}) = \begin{cases} 1; & |\omega| \leq \omega_c \\ 0; & en\ otro\ caso \end{cases}$$



Nota: La figura muestra la respuesta, en el dominio de frecuencias, que tiene un filtro pasa bajos ideal, se observa que al ser ideal, corta de manera abrupta todas las frecuencias mayores a la frecuencia de corte y no altera las frecuencias dentro de su banda de paso. Fuente propia.

El siguiente paso es aplicar la transformada inversa de Fourier en $H_d(\omega)$.

$$h_d(n) = \frac{1}{2\pi}\int_{-\pi}^{\pi} H_d(\omega)\cdot e^{j\omega n} d\omega$$

$$h_d(n) = \frac{1}{2\pi}\int_{-\omega_c}^{\omega_c} 1 \cdot e^{j\omega n} d\omega$$

$$h_d(n) = \frac{1}{2\pi jn} e^{j\omega n}\Big|_{-\omega_c}^{\omega_c} = \frac{1}{2\pi jn}\left[e^{j\omega_c n} - e^{-j\omega_c n}\right]$$

$$h_d(n) = \frac{1}{\pi n} \cdot \frac{e^{j\omega_c n} - e^{-j\omega_c n}}{2j}$$

$$h_d(n) = \frac{1}{\pi n} \cdot \sin(\omega_c n)$$

$$h_d(n) = \frac{\sin(\omega_c n)}{\pi n}$$

Se puede observar que la respuesta al impulso temporal tiene una discontinuidad cuando $n = 0$, volveremos a realizar la transformada Inversa de Fourier en $h_d(0)$.

$$h_d(0) = \frac{1}{2\pi}\int_{-\omega_c}^{\omega_c} 1 \cdot e^{j\omega 0} d\omega$$

$$h_d(0) = \frac{\omega_c}{\pi}$$

Una vez evaluado la discontinuidad en $h_d(0)$, podemos escribir la respuesta al impulso en el tiempo.

$$h_d(n) = \begin{cases} \dfrac{\sin(\omega_c n)}{\pi n} & ; \quad n \neq 0 \\ \dfrac{\omega_c}{\pi} & ; \quad n = 0 \end{cases}$$

i. Enventanado de una serie infinita

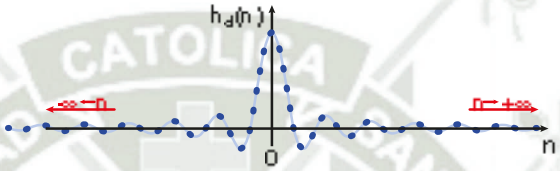
Los coeficientes del filtro son todos los valores de la serie deseada $h_d(n)$, sin embargo, esta serie es infinita ya que el valor de "*n*" va desde +∞ hasta -∞, por lo que no puede utilizarse para implementarlo físicamente, para poder implementarse físicamente, esta

serie debe ser truncado mediante la multiplicación punto a punto con una ventana $w(n)$ para obtener un numero finito de coeficientes con los que puedan trabajarse, mientras más valores se tomen al momento de hacer este truncamiento, la respuesta frecuencial se aproximara más a lo ideal, sin embargo, esto lograra a costa de un mayor costo computacional.

Figura 54

Respuesta al impulso deseada hd(n) como serie infinita no causal.

$$h(n) = h_d(n) \cdot w(n) \quad \leftrightarrow \quad H(\omega) = H_d(\omega) * W(\omega)$$

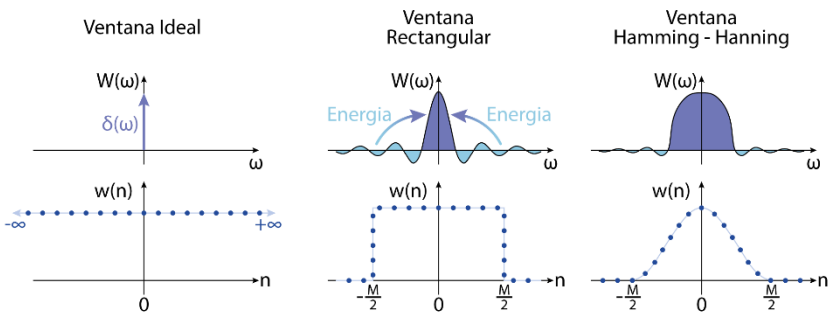


Nota: La figura muestra la respuesta (en dominio temporal discreto) al impulso de un filtro ideal, se observa que esta respuesta es infinita, positiva y negativamente, esto incumple las características de un sistema LTI. Fuente propia.

El truncar la secuencia infinita trae como efecto que la respuesta en frecuencia ideal de nuestro filtro deje de serlo y presentara lóbulos como consecuencia de la convolución entre la respuesta ideal con la frecuencia de la ventana. La ventana que permite mantener la respuesta al impulso deseada $h_d(n)$ como ideal, es la función Dirac, sin embargo, esta ventana en el dominio de tiempo es una serie constante infinita, lo que vuelve a traer el problema de que no es posible implementarlo físicamente.

Figura 55

Espectro en frecuencia y serie temporal de las ventanas.



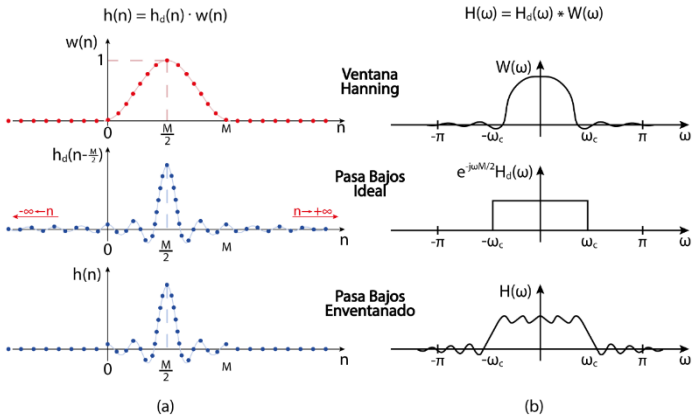
Nota: La figura muestra ventanas para truncar la respuesta de un filtro ideal, se muestra el comportamiento de las ventanas en el dominio frecuencial (arriba) y temporal (abajo). Debido a que no es posible realizar una ventana Delta Dirac, lo que se procura hacer es enviar toda la energía posible al primer lóbulo del dominio frecuencial. Fuente propia.

Una opción para acercarnos a lo ideal, es tratar de simular la función Dirac, si observamos el espectro en frecuencia de las ventanas en la **Figura 55** vemos que tienen un lóbulo principal junto a otros lóbulos más pequeños. La característica principal de la función Dirac es que tiene concentrada toda su energía en un único punto, en contraste las demás ventanas tienen su energía distribuida en todo el eje de frecuencias, entonces, para poder aproximarnos a la función Dirac debemos concentrar la mayor cantidad de energía de los lóbulos secundarios hacia el lóbulo principal. Entonces, una ventana tiene mejor comportamiento mientras más energía acumule en el lóbulo principal, sin embargo, en consecuencia, a esto, serán más complejos y tendrán más coeficientes.

Otro problema que podemos observar de la secuencia de la Respuesta al Impulso Deseado $h_d(n)$, es que, incluso a pesar de haber sido truncada, sigue siendo una señal no causal, es decir la serie abarca valores negativos de n, por lo que la seria deseada serie $h_d(n)$ debe ser desplazada temporalmente de tal manera que el valor central coincida con la mitad de la ventana, ya que la ventana es en ancho M, la serie deseada $h_d(n)$ debe ser desplazada temporalmente $M/2$.

Figura 56

Enventanado de Hd(n).



Nota: La figura muestra el proceso de enventanado (truncamiento) de la respuesta temporal infinita de un filtro pasa bajos ideal. (a)Multiplicación punto a punto de las respuesta en el dominio temporal. (b)Convolución de las respuestas en el dominio frecuencial. Fuente propia.

Una vez que la serie de la respuesta al impulso deseada, $h_d(n)$, ha sido truncada y desplazada con el propósito de hacerlo una serie finita y causal, el último paso para calcular los coeficientes del filtro que pueda ser implementado físicamente, $h(n)$, en una computadora o, en nuestro caso, en un microcontrolador, es evaluar la multiplicación punto a punto $h(n) = h_d(n) \cdot w(n)$

Tabla 15

Respuesta al impulso y ventana para filtrado pasa bajos.

Respuesta al impulso deseada	Ventana Hamming
$h_d(n)$ $= \begin{cases} \dfrac{\sin\left(\omega_c\left(n-\frac{M}{2}\right)\right)}{\pi\left(n-\frac{M}{2}\right)} & ;\ 0 \le n \le M\ ;\ n \neq \cdot \\ \dfrac{\omega_c}{\pi} & ;\ n = \dfrac{M}{2} \end{cases}$	$w(n)$ $= \begin{cases} 0.54 - 0.46 \cdot \cos\left(\dfrac{2\pi n}{M}\right) & 0 \le n \le M \\ 0 & en\ el\ rest \end{cases}$

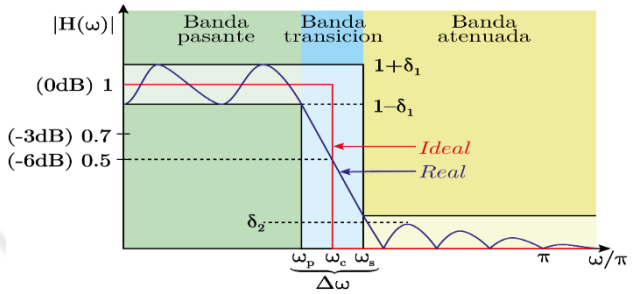
ii. Enventanado paramétrico

Al realizar los calculo para obtener los coeficientes de un filtro, es mejor manejar el eje de las frecuencias en su forma normalizada, esto nos permite evitar que la frecuencia de corte sobre pase la frecuencia Nyquist, con lo que evitamos el efecto “Aliasing”. Otra ventaja

de operar con la frecuencia normalizada, es que podemos realizar los cálculos de los coeficientes del filtro de una forma parametrizada.

Figura 57

Filtro digital pasa bajos parametrizado.



Nota: Plantilla para diseño paramétrico de un filtro normalizado. Fuente: Modificado de (Alan V. Oppenheim, 2009)

Tabla 16

Parámetros de la respuesta frecuencial de un filtro.

Parámetro	Simbología
Rizado en la banda pasante	$\delta_1 \; ; \; r_p(dB) = 20 \log \frac{1+\delta_1}{1-\delta_1} > 0$
Rizado en la banda de rechazo	$\delta_2 \; ; \; r_s(dB) \approx -20 \log \delta_2 > 0$
Frecuencia límite de la banda de paso	ω_p
Frecuencia límite de la banda de rechazo	ω_s
Banda de transición	$\Delta\omega = \omega_s - \omega_p$
Frecuencia de corte	$\omega_c \; ; \; \omega_c = \omega_p + \frac{\Delta\omega}{2}$ $\lvert H(\omega_c) \rvert = 0.5 = -6dB$

Tabla 17

Comparación de características entre ventanas.

Ventana	Anchura del lóbulo principal de la ventana $\sim\omega_\delta$	Anchura de la banda de transición del filtro diseñado $\Delta\omega$	Pico de la banda atenuada $H(\omega_s)$ - dB	Atenuación máxima de la banda de corte del filtro diseñado $20\log\delta_2$ - dB
Rectangular	$\frac{4\pi}{M}$	$\frac{1.8\pi}{M}$	-13	-21
Barlett (Triangular)	$\frac{8\pi}{M}$	$\frac{6.1\pi}{M}$	-25	-25

Ventana	Anchura del lóbulo principal de la ventana $\sim\omega_\delta$	Anchura de la banda de transición del filtro diseñado $\Delta\omega$	Pico de la banda atenuada $H(\omega_s)$ - dB	Atenuación máxima de la banda de corte del filtro diseñado $20\log\delta_2$ - dB
Hanning	$\frac{8\pi}{M}$	$\frac{6.2\pi}{M}$	-31	-44
Hamming	$\frac{8\pi}{M}$	$\frac{6.6\pi}{M}$	-41	-53
Blackman	$\frac{12\pi}{M}$	$\frac{11\pi}{M}$	-57	-74

Nota: Fuente (Alan V. Oppenheim, 2009)

Como ejemplo, diseñaremos un filtro pasa bajos con el método del enventanado, damos los parámetros necesarios para poder diseñarlo

$$f_{Samp} = 2000\ Hz$$

$$f_c = 100\ Hz$$

$$\Delta f = 80\ Hz$$

$$|H(\omega_s)| = -41\ dB$$

Como primer paso, normalizamos los parámetros frecuenciales que nos dan.

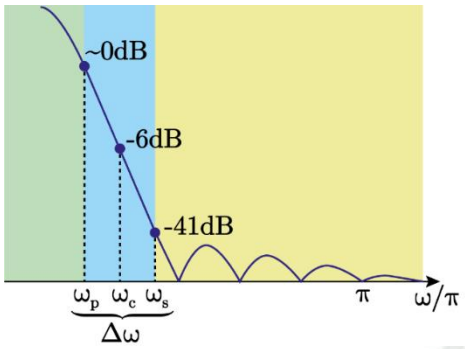
Tabla 18

Parámetros iniciales para el diseño de filtro pasa bajos.

Parámetros iniciales	
$f_{Samp} = 2000\ Hz$	$\omega_{Samp/2} = \pi$
$f_c = 100\ Hz$	$\omega_c = 0.1\pi$
$\Delta f = 80\ Hz$	$\Delta\omega = 0.08\pi$
$f_p = 60Hz \quad ; \quad f_s = 140Hz$	

Figura 58

Pendiente característica de la banda de paso de un filtro con ventana Hamming.



Nota: La figura muestra la banda de transición característica de un filtra FIR que utilizo una ventana Hamming. Fuente propia.

Como segundo lugar, se debe aproximar el orden del filtro seleccionando la ventana que se va a utilizar, viendo los valores de la **Tabla 17**, la ventana de Hamming se ajusta a los parámetros que se especificó ya que esta ventana admite hasta -41dB en su banda de transición, entonces.

$$\Delta\omega = \frac{6.6\pi}{M} \quad \rightarrow \quad M = \frac{6.6\pi}{\Delta\omega}$$

$$M = \frac{6.6\pi}{0.08\pi}$$

$$M \approx 83$$

Si revisamos la **Figura 49**, observamos que un filtro de Tipo es aquel que tiene orden par, esto hace que tenga un numero impar de coeficientes, con lo que el punto central de la secuencia recae en la mitad del orden del filtro, por lo que llevaremos el orden de nuestro filtro calculado al par inmediato superior, por lo que el orden final de nuestro filtro seria:

$$M = 84$$

En la **Tabla 19** se muestra los coeficientes necesarios para mostrar que la serie cumple con la condición de simetría expuesta al inicio de este apartado, el eje de simetría de la secuencia de coeficientes se observa en M/2. El hecho que los coeficientes del filtro

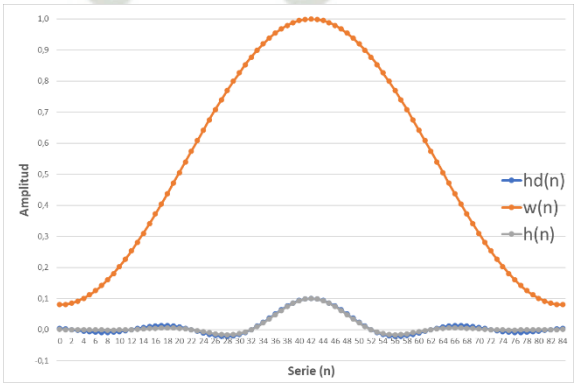
diseñado cumplan la simetría, nos permite asegurar que la respuesta frecuencial del filtro, tendrá una respuesta de fase lineal.

Tabla 19
Coeficientes críticos del filtro pasa bajos.

n	$h_d(n)$	w(n)	$h(n)= h_d(n)\cdot w(n)$	Coeficiente (b_k)
0	0,0045	0,0800	0,000356	b_0
1	0,0024	0,0813	0,000195	b_1
2	0,0000	0,0851	0,000000	b_2
…	…	…	…	…
41	0,0984	0,9987	0,098237	b_{41}
42 (M/2)	0,1000	1,0000	0,100000	b_{42}
43	0,0984	0,9987	0,098237	b_{43}
…	…	…	…	…
82	0,0000	0,0851	0,000000	b_{82}
83	0,0024	0,0813	0,000195	b_{83}
84 (M)	0,0045	0,0800	0,000356	b_{84}

En la **Figura 59** se observa la distribución de las señales involucradas en el diseño de la respuesta la impulso final que implementaremos, se puede observar que aparentemente la señal $h_d(n)$ y h(n) son iguales, sin embargo, las pequeñas diferencias debido a la multiplicación punto a punto con la ventana hacen que el comportamiento del filtro sea mejor, si se usara una ventana rectangular estas dos señales serian iguales, siendo comportamiento inferior, esto se puede comprobar viendo la **Tabla 17**.

Figura 59
Multiplicación punto a punto de $h_d(n)$ y w(n).



Nota: La figura muestra una comparación de la respuesta (temporal) al impulso del filtro ideal "$h_d(n)$", de la ventana Hamming "w(n)", y el resultado de la multiplicación punto a punto "h(n)", siendo este ultimo los valores de los coeficientes del filtro FIR.

C. Convolución de la entrada con los coeficientes – Filtrado digital

Una vez calculados los coeficientes del filtro, es fundamental comprender cómo se lleva a cabo el proceso de filtrado digital dentro de un microcontrolador. La operación de convolución de tiempo discreto, análogamente en el dominio de tiempo continuo, se basa en la superposición de dos señales: una señal móvil que recorre otra señal estática, realizando una multiplicación punto a punto seguida de una suma acumulativa en cada punto de la superposición.

$$y[n] = x[n] * h[n]$$

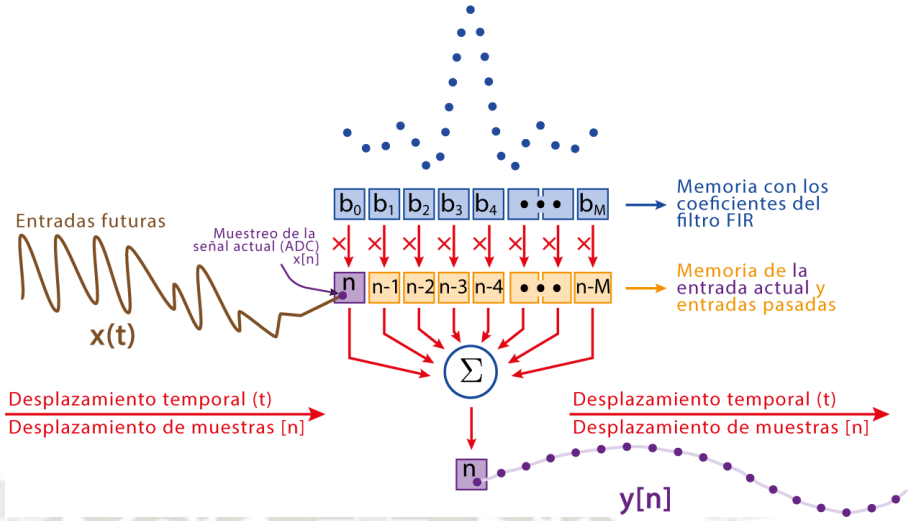
$$y[n] = b_0 x[n] + b_1 x[n-1] + b_2 x[n-2] + \cdots + b_M x[n-M]$$

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

Desde el enfoque del microcontrolador, este proceso es similar. La señal estática representa los coeficientes de la respuesta al impulso del filtro ideal, que se almacenan en un espacio de memoria del microcontrolador, mientras que la señal móvil corresponde a la señal analógica muestreada, que se desplaza y superpone con cada nueva muestra. En cada ciclo de muestreo, el microcontrolador realiza una multiplicación punto a punto entre los coeficientes del filtro y los valores de la señal muestreada superpuesta; posteriormente, suma los productos resultantes, obteniendo así el valor filtrado para una única muestra. Este proceso se repite continuamente, generando una señal filtrada digitalmente.

Figura 60

Proceso de convolución entre coeficientes del filtro y la señal de entrada muestreada.



Nota: La figura muestra el proceso que realiza el algoritmo FIR en un microcontrolador. Los coeficientes (Azul) se guardan en una porción de la memoria estática, la señal analógica (Marrón) se muestrea con un ADC (Morado), y se desplaza una posición todos los valores de la señal muestreadas (naranja) con anterioridad, estos se guardan en otra porción de la memoria, por último, se realiza la convolución (multiplicación punto a punto y posterior suma general) entre los coeficientes del filtro (Azul) y todos los valores de la señal muestreada (Naranja), como resultado se tiene el valor instantáneo de la señal filtrada este proceso se repite indefinidamente. Fuente propia.

Por lo tanto, el filtrado digital en un microcontrolador se basa en la convolución de la señal muestreada con los coeficientes del filtro en tiempo real. La implementación específica de la convolución puede variar según el microcontrolador o el algoritmo utilizado, pero en esencia, todos los métodos dependen de operaciones de multiplicación y suma entre dos vectores, además que la calidad el filtrado dependerá de las capacidades computacionales del microcontrolador.

3.4.4. Muestreo de Señales Analógicas

Los filtros digitales están estrechamente relacionados con la forma y la frecuencia con la que se ingresan las muestras de la señal a ser procesada, independientemente de la naturaleza de la fuente. Es crucial asegurar que el filtro reciba el valor de la señal muestreada

a una frecuencia igual a aquella para la cual fue diseñada; de lo contrario, se producirá un desplazamiento en la frecuencia de corte.

La señal procesada por este prototipo es de naturaleza analógica. Por lo tanto, para alimentar la etapa de filtrado, primero es necesario convertir esta señal analógica en una señal digital. Es importante destacar que el proceso de filtrado es completamente diferente del proceso de conversión de analógico a digital. Estos dos procesos no deben solaparse ni interrumpirse, ya que esto podría provocar errores. (UNR-Rosario, 2024)

A. Conversión analógica a digital

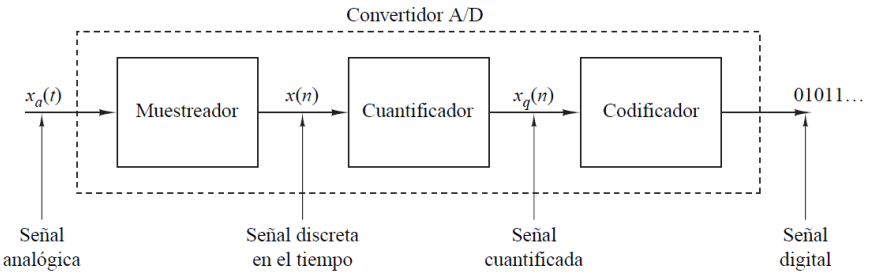
En general, la mayoría de señales de interés practico, como las señales de voz, biológicas, sísmicas, radar, etc., son de tipo analógicas, y para poder procesar estas señales de forma digital, primero es necesario convertirlas a una secuencia de números de precisión finita, este proceso se denomina Conversión analógica-digital (A/D).

El proceso de conversión A/D, tiene tres pasos

1. Muestreo: En este paso, una señal continua en el tiempo se vuelve una señal discreta en tiempo, mediante la toma "muestras" de la señal continua en el tiempo en instantes discretos de tiempo. Si $x_a(t)$ es la entrada del muestreador, la salida será $x_a(nT) \equiv n(n)$, donde T es el Tiempo de Muestreo.
2. Cuantificación: En este paso se realiza la conversión de una señal de valores continuos tomados en instantes discretos de tiempo en una señal de valores discretos en instantes de tiempo discretos
3. Codificación: En este paso, cada valor discreto se representa mediante una secuencia binaria de bits, la cantidad de estos bits depende de la resolución del convertidor.

Figura 61

Proceso de conversión Analógico-Digital.



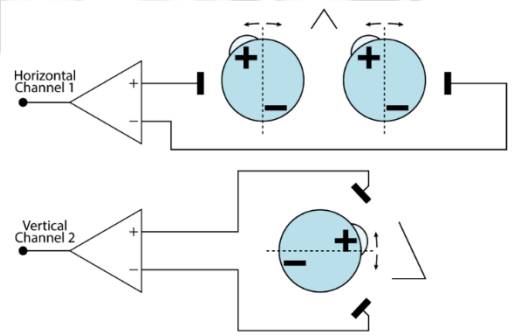
Nota: Diagrama de bloques del proceso de digitalización de una señal analógica. Fuente: (Alan V. Oppenheim, 2009)

3.5. Electro Óculo Grafía

El ojo es una fuente importante de información relevante asociada con las actividades y funciones cognitivas del usuario. El ojo puede modelarse como un dipolo eléctrico, entre la parte frontal (cornea) y la trasera (retina), que sigue los movimientos oculares. Si colocamos dos electrodos cerca de las esquinas de los ojos y dos electrodos en la parte superior e inferior de un ojo, habrá una diferencia de voltaje entre los electrodos vertical y horizontal, lo que se conoce como señal EOG. (Alanko, 1984)

Figura 62

Captura del Potencial de Reposo Corneoretinal en dos ejes.



Nota: La figura muestra el comportamiento dipolar del ojo, en base a esto, se puede obtener una señal diferencial que varía en función de la posición de los ojos, horizontal y verticalmente. Fuente propia.

La electrooculografía (EOG) es una técnica electrofisiológica desarrollada y difundida en el diagnóstico médico contemporáneo. Se basa en las propiedades bioeléctricas

del ojo y aprovecha la diferencia de potencial que oscila aproximadamente entre 15 y 200 µV, entre la córnea (positiva) en el polo anterior y el epitelio pigmentario de la retina (negativa) en el polo posterior. Esta diferencia de potencial, conocida como Potencial de Reposo Corneoretinal, permite considerar al ojo como un dipolo con extremos en su eje anteroposterior.

Cuando el globo ocular se mueve en la dirección del electrodo, el potencial eléctrico aumenta y si el ojo se mueve en la otra dirección, el potencial eléctrico disminuye. La señal del EOG es una medida de la diferencia de potencial corneal-retiniana con una amplitud de señal que oscila. Utilizando un procedimiento de calibración adecuado, la señal se puede asignar a la posición angular del globo ocular con un rango de 2 grados en la dirección vertical y un rango de 1,5 grados en la horizontal (C., A., C., & V., 2022).

El campo eléctrico generado por este dipolo, que se activa durante los movimientos oculares provocados por el reflejo vestíbulo-ocular, puede registrarse mediante electrodos de superficie colocados alrededor de la órbita ocular, el nivel de voltaje de esta señal está en el orden de los microvoltios (µV).

Tabla 20
Parámetros médicos y fisiológicos

Técnica de Medición	Rango del Parámetro	Rango de Frecuencia de la señal (Hz)	Sensor Estándar o Método
Electro cardiografía (ECG)	0.5-4 mV	0.01-250	Electrodos de piel
Electro encefalografía (EEG)	5-300 µV	dc-150	Electrodos de cuero cabelludo
	10-1000 µV	dc-1	Electrodos de piel superficial
Electro gastrografía (EGG)	0.5-80 mV	dc-1	Electrodos de superficie estomacal
Electro miografía (EMG)	0.1-5 mV	dc-10,000	Electrodos de aguja
Electro oculografía (EOG)	50-3500 µV	dc-50	Electrodos de contacto

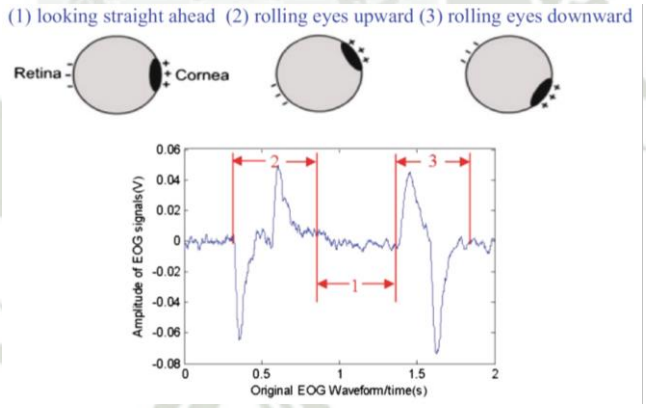
Técnica de Medición	Rango del Parámetro	Rango de Frecuencia de la señal (Hz)	Sensor Estándar o Método
Electro retinograma (ERG)	0-900 µV	dc-50	Electrodos de contacto

Nota: Fuente (Webster, 2009).

Como se muestra en la **Figura 63**, se generarán pulsos positivos o negativos cuando los ojos giren hacia arriba o hacia abajo. La amplitud del pulso aumentará con el incremento del ángulo de giro, y el ancho del pulso positivo/negativo es proporcional a la duración del proceso de giro del globo ocular. Simultáneamente, se registra el EOG en las dos direcciones horizontal y vertical utilizando cinco electrodos (dos verticales, dos horizontales y uno para la referencia) colocados alrededor de los ojos.

Figura 63

Movimiento vertical del ojo y su correspondiente onda



Nota: La figura muestra la señal eléctrica del movimiento ocular capturada en un laboratorio, .(1) Vista hacia el frente. (2)Vista hacia arriba. (3)Vista hacia abajo. Fuente: (Reda, Tantawi, shedeed, & Tolba, 2019).

La detección del parpadeo de los ojos también es posible debido a que el ojo reacciona de una manera similar a cuando se mira hacia arriba, esto se denomina reflejo o fenómeno de Bell. El fenómeno de Bell es un reflejo ocular involuntario y protector que ocurre cuando una persona cierra los ojos, especialmente durante el parpadeo o cuando intenta cerrar los ojos con fuerza. Consiste en un movimiento hacia arriba y hacia afuera de

los globos oculares. Este reflejo lleva el nombre de Sir Charles Bell, un neurofisiólogo y anatomista escocés que lo describió (Reda, Tantawi, shedeed, & Tolba, 2019).

3.5.1. Variación de la Intensidad de la Señal

Los cambios en la iluminación del ojo pueden cambiar la respuesta del potencial eléctrico del ojo siendo este directamente proporcional a la intensidad de la luz, sin embargo, se ha demostrado que estimulantes comunes como el tabaco, café y alcohol, también tienen un efecto en estas respuestas ya que pueden inducir oscilaciones rítmicas en el potencial corneo retinal.

Un estímulo luminoso intenso, provoca un incremento inicial en el potencial corneo retinal, seguido de una caída abrupta. Posteriormente, el potencial aumenta lentamente hasta alcanzar el nivel máximo en aproximadamente 10 minutos. Una vez que el ojo se adapta a este estimulo, el nivel del potencial corneo-retinal empieza a regresar a su nivel inicial, describiendo oscilaciones alrededor de este, con una frecuencia de dos ciclos por hora.

Esta reacción es exactamente opuesta en el caso de un estímulo de oscuridad, donde el potencial alcanza su nivel mínimo en 10 minutos, para luego regresar al nivel inicial con oscilaciones de dos ciclos por hora. La magnitud del aumento del potencial inducido por la luz depende del estado de adaptación previo de la retina, así como de la intensidad, duración y longitud de onda del estímulo (Ophtal, 1966).

En las enfermedades oculares que afectan solo a uno de los ojos, el aumento de potencial eléctrico debido a un incremento de luz, puede inducir el efecto contrario en el ojo afectado, provocando curvas EOG anormales.

Una respuesta patológica y plana de la señal EOG, es característica de enfermedades degenerativas tapetorretinales (degeneraciones progresivas de la retina) como la retinitis

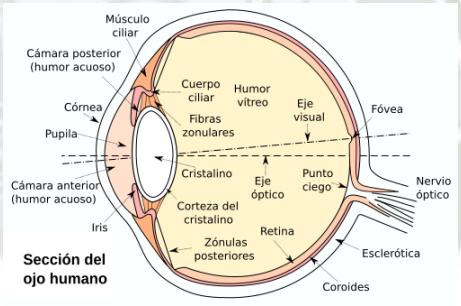
pigmentosa y la coroideremia, lo cual indica lesiones generalizadas de la retina y en el epitelio pigmentario (Alanko, 1984).

Todos estos factores deben de ser tomados en cuenta al momento de realizar la captura y medida de las señales EOG.

3.5.2. Fisiología del Ojo

El ojo es el órgano sensorial responsable de la visión, uno de los sentidos más importantes para los seres humanos y muchos animales, permitiéndonos percibir el entorno en términos de forma, color, tamaño y movimiento. Funciona como una cámara biológica, capturando y enfocando la luz para formar imágenes. (Garrity, 2022)

Figura 64
Estructura interna del ojo humano.



Nota: La figura muestra la estructura interna del ojo humano. Fuente: Tomado de wikibooks.org

Las estructura especializadas más importante del ojo son las siguientes:

- Córnea: La cubierta transparente en la parte frontal del ojo que ayuda a enfocar la luz que entra.
- Pupila: La abertura central del iris que regula la cantidad de luz que entra al ojo.
- Iris: La parte coloreada del ojo que controla el tamaño de la pupila.
- Cristalino: Una lente flexible que ajusta su forma para enfocar la luz en la retina.

- Retina: La capa de tejido sensible a la luz en la parte posterior del ojo que contiene células fotorreceptoras (conos y bastones).
- Nervio Óptico: Transmite las señales visuales desde la retina al cerebro.

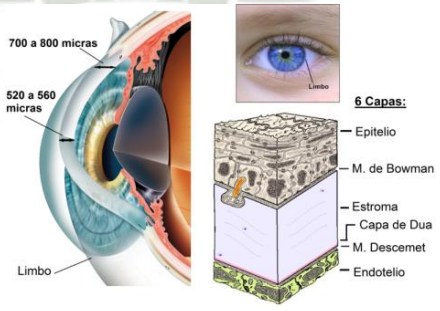
A. Cornea

Sus principales funciones son la de permitir la transmisión y refracción de la luz, contener la presión intraocular y proporcionar una interfaz de protección con el medio ambiente.

Es la estructura más externa y frontal del globo ocular, y junto con los párpados y el sistema lagrimal, la conjuntiva bulbar y tarsal, los fórnix conjuntivales, el limbo, y película lagrimal conforma la superficie ocular (SO). Todas estas estructuras permiten proteger al ojo de agresiones externas y mantener la transparencia e integridad de la superficie corneal y del medio intraocular.

La córnea tiene una superficie de 123 milímetros cuadros y posee un espesor de 520 - 560 micras a nivel central y de 700 a 800 en la periferia. (García, s.f.)

Figura 65
Estructura de la córnea



Nota: La figura muestra la estructura de la córnea: Fuente: Tomado de (García, s.f.).

Una de las principales características de la córnea es su transparencia, condición indispensable para permitir el paso de la luz. Para mantener esta transparencia la córnea es avascular, es decir, carece de vasos sanguíneos por lo que la mayor parte de los nutrientes

(carbohidratos, vitaminas, aminoácidos) y otros sustratos son entregados a través del endotelio (capa más interna), de las arcadas vasculares que se ubican alrededor de ella en el limbo o disueltos en la película lagrimal (García, s.f.).

Desde su superficie externa a la interna la córnea consta de 6 capas:

- Epitelio (con su membrana basal)
- Membrana de Bowman
- Estroma
- Capa de Dua
- Membrana de Descemet
- Endotelio.

La córnea tiene una relativa abundancia de cationes como el sodio (Na^+) en comparación con su concentración en la retina. Aunque la córnea es principalmente avascular (sin vasos sanguíneos) y tiene menos actividad metabólica que la retina, los iones Na^+ están presentes en el fluido que baña la córnea, como el humor acuoso.

La concentración de aniones como el cloruro (Cl^-) también es equilibrada, pero la carga neta de la córnea se mantiene relativamente más positiva en comparación con la retina debido a la menor actividad iónica y metabólica.

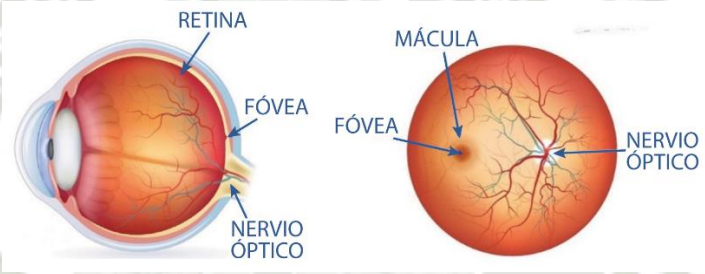
B. Retina

La retina constituye la capa fotosensible del globo ocular encargada de recoger, elaborar y transmitir los estímulos ópticos provenientes del medio exterior. Su adecuado funcionamiento es indispensable para la discriminación del detalle, la percepción del color, la sensibilidad al contraste, y la visión periférica.

A pesar de su localización periférica, debido a su estirpe neurosensorial, la retina se considera como una parte especializada del sistema nervioso central.

En su sector posterior, se encuentra firmemente adherida y se continua con la cabeza del nervio óptico, mientras que en el sector anterior, se integra con el cuerpo ciliar (pars plana) a nivel de la ora serrata. En esta zona, la retina neurosensorial y el epitelio pigmentario retiniano se continúan respectivamente, con el epitelio no pigmentario y con el epitelio pigmentario del cuerpo ciliar (García, s.f.).

Figura 66
Estructura de la retina.



Nota: La figura muestra la estructura de la retina. Fuente: Modificado de vedicmedglobal.com

La retina está compuesta por varias capas de células especializadas, sin embargo, debido al enfoque de esta tesis, no se detallarán. No obstante, se describirán las estructuras especializadas presentes en la retina:

- **Mácula:** Es una pequeña área central de la retina responsable de la visión central detallada. La mácula es crucial para actividades que requieren precisión visual, como leer y reconocer caras.
- **Fóvea:** Ubicada en el centro de la mácula, es la región con la mayor densidad de conos y proporciona la visión más aguda y detallada. No contiene bastones.
- **Disco Óptico (Punto Ciego):** Es el punto donde los axones de las células ganglionares se agrupan para formar el nervio óptico. No contiene fotorreceptores, por lo que es insensible a la luz, creando un punto ciego en el campo visual.

La retina, especialmente en la capa de las células fotorreceptoras y el epitelio pigmentario, tiene una alta actividad metabólica, lo que implica un flujo constante de iones. Las células fotorreceptoras utilizan bombas de sodio-potasio (Na^+/K^+-ATPasa) que transportan tres iones de sodio fuera de la célula y dos iones de potasio dentro de la célula, contribuyendo a una mayor concentración de potasio (K^+) en el interior de las células y de sodio (Na^+) en el exterior (García, s.f.).

Además, el epitelio pigmentario transporta calcio (Ca^{2+}) y otros cationes hacia la capa intermedia, mientras que las células generan aniones como fosfatos y bicarbonato debido a los procesos metabólicos, contribuyendo a un ambiente interno más negativo.

En el entorno extracelular de la retina, también hay una alta concentración de aniones como el cloruro (Cl^-), que ayuda a mantener la negatividad relativa en esta área.

C. El Ojo como dipolo eléctrico

La diferencia de potencial entre la córnea y la retina, conocida como el potencial de reposo corneo-retinal, se debe a la distribución de iones en estas partes del ojo. Este fenómeno genera un dipolo eléctrico natural, donde la córnea es relativamente positiva en comparación con la retina, que es relativamente negativa. Se considera que el origen del potencial corneo – retinal, se encuentra principalmente en el epitelio pigmentario de la retina (Alanko, 1984).

La polaridad de la córnea y la retina se debe a diferencias en la composición celular, la actividad metabólica y el manejo de iones en estas partes del ojo. La córnea es relativamente más positiva y la retina más negativa, lo que crea el dipolo eléctrico que puede ser detectado y medido mediante técnicas como la electro oculografía (EOG) (Villamil, 2004).

i. Transparencia y Absorción de la luz

La córnea es una estructura transparente que permite el paso de la luz hacia el interior del ojo. Está compuesta principalmente por agua y colágeno, y tiene relativamente pocos iones y células pigmentadas. Esto hace que tenga un potencial menos negativo o relativamente más positivo.

La retina, en cambio, contiene una gran cantidad de células fotorreceptoras (conos y bastones) que son esenciales para la detección de la luz. Estas células son metabólicamente activas y, en el proceso de foto transducción, generan una corriente iónica que contribuye a un potencial eléctrico negativo en la retina.

ii. Flujo Iónico

La actividad eléctrica del ojo también es influenciada por la presencia de bombas de iones y canales en las membranas celulares de la retina. Estas bombas y canales ayudan a mantener una concentración específica de iones, como el sodio y el potasio, dentro y fuera de las células, lo que contribuye al potencial de reposo. La retina, al estar más activa en términos de procesamiento visual, presenta un ambiente intracelular más negativo.

iii. Función de las células pigmentadas

La retina contiene el epitelio pigmentario, esta es una capa que ayuda a nutrir las células fotorreceptoras y absorbe el exceso de luz para mejorar la calidad visual. Esta capa también participa en el transporte de iones, lo que puede contribuir a la generación de un potencial negativo.

3.6. El Parpadeo y el Fenómeno de Bell

El parpadeo es el cierre y apertura rápida y automática de los párpados. Es un movimiento involuntario que ocurre de manera regular, aunque también puede ser realizado

voluntariamente. Durante el parpadeo, los párpados cubren la superficie del ojo, protegiéndolo y distribuyendo una capa uniforme de lágrimas.

Un parpadeo tiene una duración promedio de 100 a 150 milisegundos (0.1 a 0.15 segundos), lo que lo convierte en una de las respuestas reflejas más rápidas del cuerpo humano. La frecuencia de parpadeo varía, pero en condiciones normales, las personas suelen parpadear de 15 a 20 veces por minuto, aunque este ritmo puede cambiar dependiendo de factores como la actividad visual, el nivel de concentración o el estado emocional. En situaciones que requieren atención prolongada, como trabajar frente a una pantalla, la tasa de parpadeo puede disminuir significativamente, lo que puede causar sequedad ocular (Lusby, 2023).

El parpadeo tiene varias funciones esenciales:

- **Humedecimiento del ojo:** El parpadeo distribuye una fina capa de lágrimas sobre la superficie ocular, lo que ayuda a mantener la córnea y la conjuntiva húmedas. Esto previene la sequedad y la irritación ocular.
- **Protección:** Sirve como un reflejo protector frente a partículas, cuerpos extraños, luz intensa o estímulos potencialmente dañinos. Cuando algo amenaza los ojos, el parpadeo es una respuesta rápida que ayuda a protegerlos.
- **Limpieza:** El parpadeo también ayuda a eliminar pequeñas partículas o polvo que pueden entrar en contacto con los ojos. El movimiento de los párpados arrastra estas partículas hacia los conductos lagrimales para ser eliminadas.
- **Descanso visual:** El parpadeo da breves momentos de descanso a los músculos oculares, especialmente durante actividades que demandan mucha concentración visual, como leer o usar pantallas.

El fenómeno de Bell, o reflejo ocurógiro palpebral, es un reflejo de protección ocular que se manifiesta mediante un desplazamiento ascendente de los ojos cuando se intenta cerrar los párpados de manera forzada, o cuando el glóbulo ocular es interceptado por un objeto externo, como al intentar tocar la córnea de un paciente. Este movimiento ascendente ocurre en la mayoría de las personas pero es mucho más notorio en personas que han sufrido parálisis facial.

Durante el parpadeo normal, que es rápido y automático, este movimiento de elevación de los globos oculares (fenómeno de Bell) generalmente no ocurre de manera significativa. Sin embargo, cuando el cierre de los párpados es más prolongado o forzado, es más probable que el fenómeno de Bell se active. Aproximadamente el 75-85% de las personas presentan este reflejo en mayor o menor grado, pero no está presente en todos los individuos (Thomas L. , 2019).

3.7. Electrodos Biomédicos

Para medir potenciales eléctricos en el cuerpo, es necesario utilizar una interfaz adecuada entre el cuerpo humano y el dispositivo de medición electrónica. En este contexto, los electrodos juegan un papel fundamental. Los electrodos deben de ser capaces de facilitar la conducción de la corriente a través de la interfaz entre el cuerpo y el sistema electrónico de medición.

Es importante destacar que los electrodos actúan como transductores, ya que convierten la corriente bioeléctrica del cuerpo, que se transmite mediante iones, en corriente eléctrica que se transmite mediante electrones en el electrodo y los cables conectores.

Esta conversión es crucial para garantizar que los dispositivos electrónicos puedan procesar las señales biológicas. Por lo tanto, los electrodos deben ser diseñados para realizar

esta conversión, asegurando una adecuada transferencia de información bioeléctrica al equipo de medición.

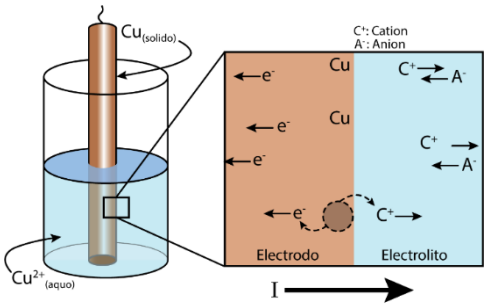
3.7.1. Electrode Half-Cell Potential

El "electrode half-cell potential" (potencial de media celda de un electrodo) es el potencial eléctrico que se desarrolla entre un electrodo y una solución electrolítica debido a las reacciones electroquímicas que ocurren en la interfaz entre el electrodo y la solución. Este potencial es una medida de la tendencia de los iones a ganar o perder electrones en la superficie del electrodo.

El potencial de media celda es específico para cada tipo de electrodo y depende del material del electrodo, la composición de la solución electrolítica, la temperatura, y otras condiciones ambientales. Es una propiedad intrínseca de la combinación de estos factores y no se puede medir directamente; en su lugar, se mide en relación con un electrodo de referencia conocido.

Para comprender el paso de la corriente del cuerpo, hacia el electrodo, se puede observar la **Figura 67**, donde se muestra una interfaz entre un electrolito, que contiene la forma ionizada del electrodo presente. La corriente que cruza por la interfaz Electrodo-Electrolito, inicia con el flujo de electrones hacia la izquierda, opuesto al sentido de la corriente del electrodo (sentido convencional de la corriente eléctrica). (Neuman, 2009)

Figura 67
Interface Electrodo-Electrolito



Nota: La figura muestra el intercambio de electrones que ocurre entre un electrodo y un electrolito, este flujo de electrones es el tipo de corriente eléctrica que se puede sensar con los electrodos EOG. Fuente: Modificado de (Neuman, 2009)

Producto de la oxidación del electrodo, los átomos neutros del electrodo se oxidan liberando electrones, volviéndose cationes, estos se mueven en el mismo sentido que la corriente, ingresando al electrolito y los aniones se mueven hacia el electrodo, en dirección opuesta a la corriente, para oxidarse y volverse neutral, entregando sus electrones libres al electrodo (Webster, 2009), todo esto se puede representar por la siguiente reacción:

$$C \leftrightarrows C^{n+} + ne^{-} \quad ; \quad A^{m-} \leftrightarrows A + me^{-}$$

Donde:

C	:	Catión neutro del material del electrodo
n	:	Electrón de valencia del material del electrodo
A	:	Anión neutro disuelto en el electrolito
m	:	Electrón de valencia del anión del electrolito
e^{-}	:	Electrón

Al iniciar el proceso electro-químico, la concentración local de cationes en la solución cambia, producto de la oxidación del electrodo, lo que afecta la concentración de aniones en el mismo punto, esto produce que la carga neutra inicial cambie. Por este motivo, el electrolito que esta alrededor del electrodo tiene un potencial eléctrico diferente al resto de la solución. A este diferencial de potencial se le conoce como Potencial de Media Celda de un Electrodo (Electrode Half-Cell Potential - EHC) (Neuman, 2009).

3.7.2. Modelo Eléctrico

Las características eléctricas de corriente-voltaje de un electrodo han sido sujetos de muchos estudios, por lo que se presentara un resumen de estos. Se ha determinado que estas características son no lineales, por lo que es necesario utilizar componentes no lineales para

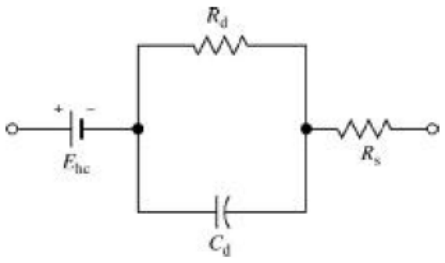
modelar el comportamiento del electrodo. Adicionalmente, las características del electrodo, dependen de la forma de onda de la corriente que atraviesa por él. Cuando se aplica una corriente senoidal para medir el comportamiento del circuito del electrodo, se observó que las características también son dependientes de la frecuencia.

Para entradas senoidales, las características de un electrodo tienen un componente tano resistivo como reactivo en todas las frecuencias, excepto en las más bajas. Esto se puede modelar como una resistencia y capacitancia en serie. Este circuito equivalente de una resistencia y un condensador en serie, tiene el inconveniente de que en las frecuencias bajas, la impedancia del circuito llegaría al infinito a medida que la frecuencia de acerca a 0 Hz (DC). Para evitar este problema, se puede convertir este circuito RC en serie, en un circuito paralelo que tenga una impedancia puramente resistiva a frecuencias muy bajas.

Si se combina este circuito con una fuente de voltaje que representa el potencial de media celda (Half-Cell Potential), y una resistencia en serie que representa los efectos de la interfaz y la resistencia del electrolito, podemos llegar al modelo del circuito equivalente de electrodo biopotencial (Neuman, 2009) que se muestra en la **Figura 68**.

Figura 68

Circuito equivalente de un electrodo biopotencial, en contacto con un electrolito.



Nota: La figura muestra el circuito equivalente de un electrodo junto a su electrolito. Fuente: Tomado de (Neuman, 2009).

En este circuito equivalente se observa que el voltaje producto del contacto entre el electrodo y el electrolito, es el que determina el valor de la señal (E_{hc}), adicionalmente, los

componentes R_d y C_d marcan la impedancia asociada con la interface electrodo-electrolito y los efectos de la polarización, R_s es la resistencia asociada al electrolito.

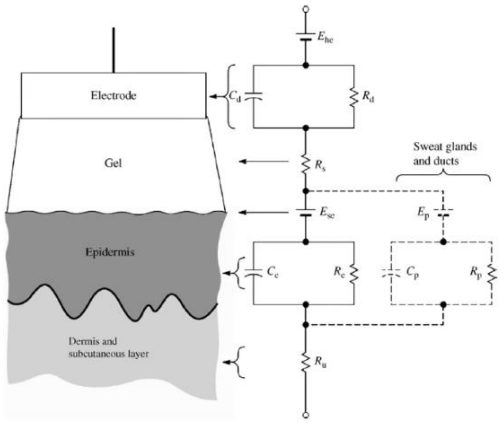
Cuando la bioseñal es medida desde la superficie de la piel, es necesario considerar una interface adicional, la interface entre el electrodo-electrolito y la piel.

Generalmente se usa como electrolito un gel de cloruro (Cl^-) como principal anión para mantener un buen contacto con la piel. La piel consiste de tres capas principales que cubren todo el cuerpo para protegerlo del ambiente. La capa más exterior, la epidermis, cumple la función más importante en la interface electrodo-piel. Esta capa consiste de tres capas adicionales y se renuevan constantemente, naciendo en la capa más inferior (capa germinativa), se mueve gradualmente hacia afuera hasta ser expulsada al ambiente en la capa "cornea".

Las capas de piel más internas, contienen más componentes vasculares y nervioso, así como glándulas y conductos sudorales. Estas capas son similares a los tejidos del resto del cuerpo, por lo que no tienen una característica eléctrica especial a diferencia de la capa "Cornea" (Neuman, 2009).

Figura 69

Circuito equivalente del electrodo sobre la piel.



Nota: La figura muestra el equivalente eléctrico del contacto entre la piel humana y un electrodo junto a su electrolito. Fuente: Tomado de (Neuman, 2009).

El circuito equivalente de la piel, se muestra en la **Figura 69**, y parte justo después de la resistencia del electrolito R_s, en la capa "cornea", este es la parte más externa de la epidermis. Debido a que esta capa es una membrana semipermeable a los iones, hay una diferencia en la concentración iónica a través de ella, por lo que se genera un diferencial de potencial E_{se}. También se determina que la capa epidérmica tiene una impedancia eléctrica que se comporta como un circuito RC en paralelo, con la resistencia R_e y el capacitor C_e. Las capas más internas, como la dermis y las subcutáneas debajo de ella, se comportan generalmente como resistencias puras (Neuman, 2009).

Si fuese posible, es mejor reducir el efecto de la capa "Cornea", para obtener un electrodo más estable, esto se consigue retirando esta capa alrededor del electrodo. Se puede lograr de diferentes manera, desde frotar vigorosamente con una almohadilla empapada de alcohol o acetona, hasta raspar la capa "cornea" con papel lija para perforarlo, en todos los casos, se tiene a eliminar o reducir los efectos de R_e y C_e. Sin embargo, esto puede provocar irritaciones en las zonas tratadas, y se debe considerar que esta capa se regenera muy rápidamente, en aproximadamente 24 horas, por lo que esto debe ser considerado en casos crónicos (Neuman, 2009).

3.7.3. Desviación de Voltaje del Electrodo

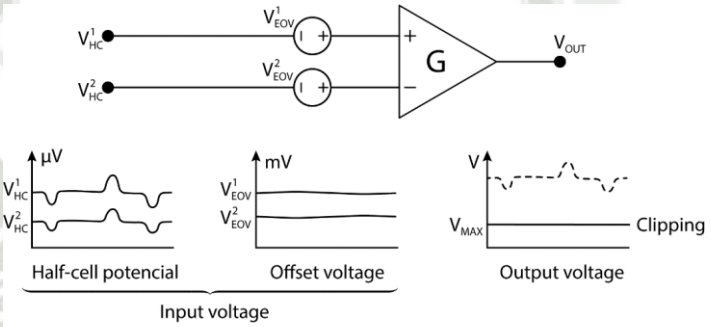
El desplazamiento de voltaje en un electrodo (electrode offset voltage), es un fenómeno que puede ocurrir en los sistemas de medición de biopotenciales en general, incluyendo el caso de este estudio, el electrooculograma (EOG).

Este fenómeno se manifiesta cuando los electrodos de medición desarrollan, aleatoriamente, un voltaje DC (DC offset) en serie a cada uno de ellos. Si el electrodo derecho e izquierdo presentan diferentes valores de estos voltajes, en la señal de entrada que se recibe de los electrodos, se reflejará un diferencial de voltaje, que puede estar en el rango

de cientos de milivoltios. A este diferencial de voltaje se conoce como desplazamiento de voltaje de un electrodo (Electrode offset voltaje - EOV) y puede ser causado por diversas razones, incluyendo la composición de los materiales de los electrodos, las condiciones de la interfaz piel-electrodo, y reacciones electroquímicas en dicha interfaz.

Figura 70

"Clipping" de la señal, por consecuencia del desplazamiento de voltaje del electrodo.



Nota: La figura muestra los efectos perjudiciales del voltaje de deriva del electrodo. El valor de este voltaje al ser mayor que la señal útil, hace que al pasar por la amplificación, este voltaje sature al amplificador operacional, por tal motivos, este defecto del electrodo debe de ser tomado en cuenta. Fuente: Modificado de youtube.com, canal "Patrick Mercier", lecturas: 8, 9.

Los efectos del desplazamiento de voltaje en un electrodo pueden ser significativos, incluyendo distorsión de la señal biopotencial real, saturación de los amplificadores operacionales (OPAMPs) utilizados en la medición, y errores en la línea base de la señal. En particular, un voltaje de desplazamiento grande puede reducir la ganancia utilizable de los amplificadores e incluso, llevarlos a saturación, provocando un corte (clipping) de la señal a la salida del amplificador.

Las causas comunes de este fenómeno incluyen:

- Material de los Electrodos: Diferentes materiales pueden tener diferentes potenciales electroquímicos, lo que contribuye al voltaje de desplazamiento.
- Interfaz Piel-Electrodo: La condición de la piel y el gel conductor utilizado pueden influir en el potencial de contacto.

- Reacciones Electroquímicas: Estas reacciones en la interfaz pueden generar un voltaje de desplazamiento.
- Diferencias de Temperatura: Las variaciones de temperatura pueden afectar el voltaje de desplazamiento.
- Contaminación de los Electrodos: La acumulación de contaminantes en los electrodos puede incrementar el voltaje de desplazamiento.
- Degradación del contacto de los Electrodos: El uso prolongado o inadecuado de estos, puede degradar el contacto con la piel, incrementando el voltaje de desplazamiento.

Para mitigar los efectos de este fenómeno, se pueden tomar varias medidas:

- Uso de Electrodos de Alta Calidad: La utilización de electrodos con materiales homogéneos y estables puede minimizar el voltaje de desplazamiento.
- Preparación Adecuada de la Piel: Limpiar y preparar adecuadamente la piel reduce la resistencia de contacto y el voltaje de desplazamiento.
- Aplicación de Filtros: Los filtros de alta pasada pueden eliminar los componentes de DC y las frecuencias bajas de la señal.
- Uso de pre amplificadores: Limitar la ganancia del primer amplificador puede permitir aplicar a la señal desplazada, métodos para bloquear la componente DC.
- Compensación Electrónica: Algunos dispositivos incluyen circuitos que compensan automáticamente el voltaje de desplazamiento, ajustando la señal medida.

Estas estrategias ayudan a garantizar que las señales biopotenciales sean precisas y libres de distorsiones, facilitando una interpretación clínica correcta. Para una revisión mucho más detallada de este fenómeno, se aconseja al lector visitar el canal “Patrick

Mercier" de YouTube, las "lecturas" 8 al 9, donde se realiza un análisis sobre los amplificadores de instrumentación.

3.7.4. Electrodo de Plata/Plata-Cloruro (Ag/AgCl)

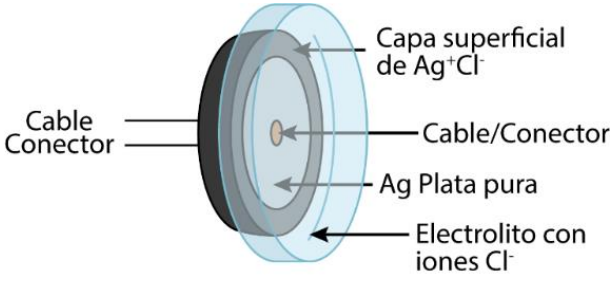
Los electrodos utilizados para medir los biopotenciales, se clasifican en dos categorías: electrodos Polarizables y no Polarizables. Esta clasificación se basa en la respuesta electroquímica del electrodo cuando una corriente pasa a través de él hacia el electrolito.

Los electrodos Polarizables presentan una resistencia significativa al paso de corriente debido a la acumulación de cargas en la interfaz electrodo-electrolito, lo que resulta en un cambio en el potencial del electrodo. Por otro lado, los electrodos no Polarizables permiten un flujo de corriente más libre sin un cambio notable en el potencial, debido a la transferencia equilibrada de cargas entre el electrodo y el electrolito.

Los electrodos de plata/plata-cloruro (Ag/AgCl), tienen una respuesta electroquímica que se aproxima a las características de un Electrodo no Polarizable Perfecto, además, son prácticos, ya que pueden ser fácilmente fabricados en un laboratorio. Consisten de un metal recubierto con una capa de un compuesto iónico ligeramente soluble de ese metal, con un anión adecuado. Toda la estructura está sumergida en un electrolito que contiene el anión en una concentración relativamente alta. (Angulo, 2025)

Figura 71

Vista transversal de un electrodo Ag/AgCl



Nota: La figura muestra la estructura de un electrodo Ag/AgCl. Fuente: Propia.

Los electros de este tipo de aleación vienen en diversas presentación, según sea la aplicación, sin embargo, los tipos más ampliamente usados son

- Electrodos de placa de metal
- Electrodos de succión
- Electrodos flotantes

CAPÍTULO II

1. INGENIERÍA DEL PROTOCOLO USB

1.1. Análisis de un Mouse Estándar

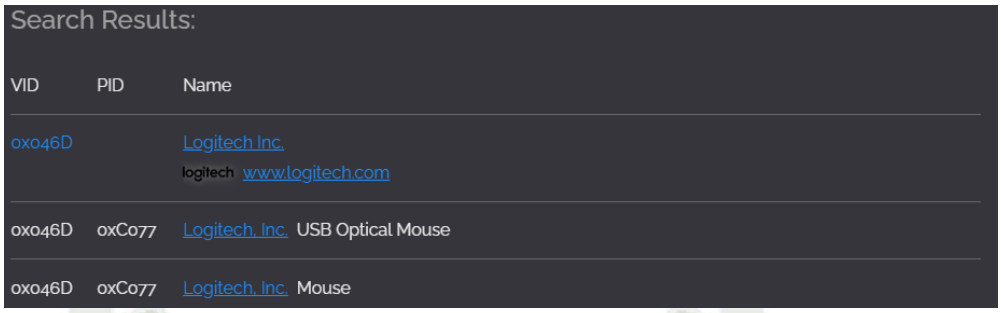
Para hacer el estudio del protocolo USB, utilizaremos un mouse estándar Logitech M90, el cual es de uso personal, una vez conectado, utilizando el programa USBDeview, localizamos nuestros dispositivos y extraemos toda la información sobre el dispositivo.

Tabla 21
Datos USB del periférico de prueba.

	MOUSE LOGITECH
Device Name	USB Optical Mouse
Description	USB Input Device
Device Type	HID (Human Interface Device)
Connected	Yes
Safe To Unplug	Yes
Disabled	No
USB Hub	No
VendorID	046d
ProductID	c077
Firmware Revision	72.00
USB Class	03
USB SubClass	01
USB Protocol	02
Hub / Port	Hub 1, Port 1
Computer Name	KEVCAR-LAPTOP
ParentId Prefix	7&31166be&1
Service Name	HidUsb
Service Description	@input.inf,%HID.SvcDesc%;Microsoft HID Class Driver
Driver Filename	hidusb.sys
Device Class	
Device Mfg	(Standard system devices)
Friendly Name	
Power	100 mA
USB Version	2.00
Driver Description	USB Input Device
Driver Version	10.0.19041.2486
Driver InfSection	HID_Inst.NT
Driver InfPath	input.inf
Instance ID	USB\VID_046D&PID_C077\6&1d7e6c28&0&1
Capabilities	Removable, SurpriseRemovalOK

Figura 72

Resultado de búsqueda de código Vendor ID



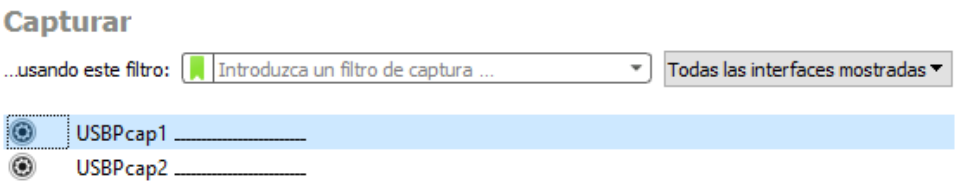
Nota: La figura muestra el resultado de la búsqueda del código de Vendor ID obtenido del periférico de prueba que se utiliza. El número "046D" es asignado únicamente al fabricante Logitech Inc. Fuente: Búsqueda de vendor ID en: the-sz.com

Una ventaja del programa USBDeview es que nos permite ver información general y del sistema, que permite el funcionamiento entre el mouse y el Host, así como la dirección física. Se debe resaltar la diferencia entre dirección física y lógica, el programa USBDeview únicamente nos proporciona la dirección física, con la cual podemos ubicar en que puerto de la computadora se conectó el dispositivo.

Para ver todo el proceso de comunicación de un puerto de un Host y un dispositivo es necesario utilizar dispositivos intermediarios como el Beagle USB 480 Analizador de Protocolo de la marca Total Phase, sin embargo, debido al costo elevado, utilizaremos WireShark con el Plugin USBPCap, el cual captura parcialmente el tráfico del puerto USB, en el cual conectaremos el mismo dispositivo.

Figura 73

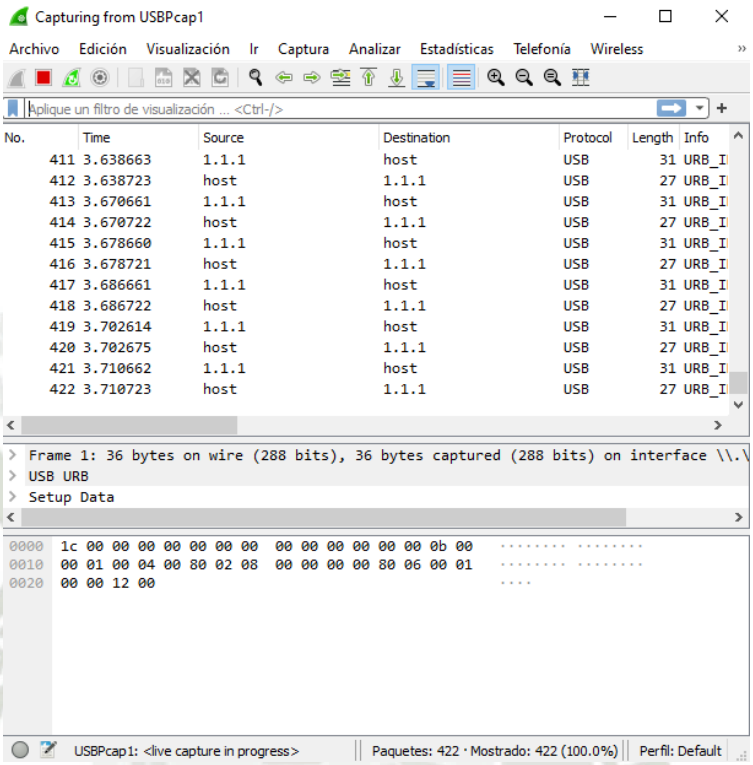
Inicio de captura de tráfico. USBPcap detecto 2 Hubs en el Host.



Nota: La figura muestra la interfaz inicial del software Wireshark, utilizando al extensión USBPcap, con el que se analizara el tráfico USB de los dispositivos a probar. Fuente propia.

Figura 74

Proceso de captura de tráfico USB con WireShark.



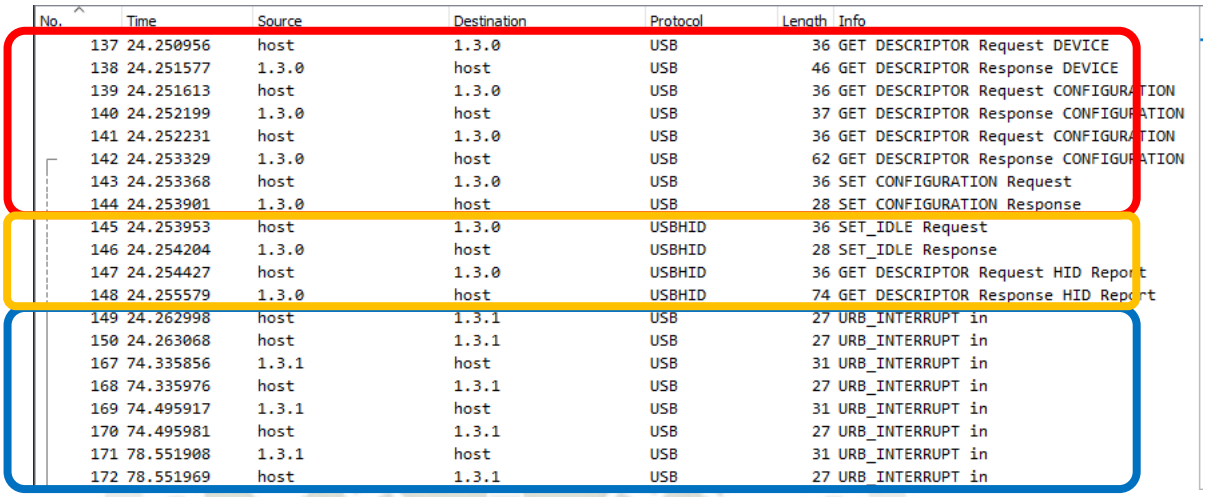
Nota: La figura muestra una porción del proceso de captura de tráfico USB entre un dispositivo USB (mouse) y la computadora usando Wireshark y USBPcap, se observa que al mover el mouse, se realiza una comunicación constante entre el dispositivo (dirección 1.1.1) y la computadora (host). Fuente propia.

Al iniciar la captura de datos, conectamos el mouse, para ver el proceso de Enumeración, luego presionamos lo botones y movemos el mouse hacia arriba y abajo, luego derecha e izquierda, una vez hecho esto, guardamos la captura para un análisis detallado.

La captura de tráfico USB con Wireshark se realiza a partir del ítem 4 que se muestra en la **Figura 74**. En la captura de datos USB que podemos ver en la **Figura 75**, se observa que el Host asigno al dispositivo una dirección lógica de 3, y la comunicación de control se dirige al End Point 0 ya que se está realizando las configuraciones iniciales.

Figura 75

Análisis de tráfico de datos USB con Wireshark.



No.	Time	Source	Destination	Protocol	Length	Info
137	24.250956	host	1.3.0	USB	36	GET DESCRIPTOR Request DEVICE
138	24.251577	1.3.0	host	USB	46	GET DESCRIPTOR Response DEVICE
139	24.251613	host	1.3.0	USB	36	GET DESCRIPTOR Request CONFIGURATION
140	24.252199	1.3.0	host	USB	37	GET DESCRIPTOR Response CONFIGURATION
141	24.252231	host	1.3.0	USB	36	GET DESCRIPTOR Request CONFIGURATION
142	24.253329	1.3.0	host	USB	62	GET DESCRIPTOR Response CONFIGURATION
143	24.253368	host	1.3.0	USB	36	SET CONFIGURATION Request
144	24.253901	1.3.0	host	USB	28	SET CONFIGURATION Response
145	24.253953	host	1.3.0	USBHID	36	SET_IDLE Request
146	24.254204	1.3.0	host	USBHID	28	SET_IDLE Response
147	24.254427	host	1.3.0	USBHID	36	GET DESCRIPTOR Request HID Report
148	24.255579	1.3.0	host	USBHID	74	GET DESCRIPTOR Response HID Report
149	24.262998	host	1.3.1	USB	27	URB_INTERRUPT in
150	24.263068	host	1.3.1	USB	27	URB_INTERRUPT in
167	74.335856	1.3.1	host	USB	31	URB_INTERRUPT in
168	74.335976	host	1.3.1	USB	27	URB_INTERRUPT in
169	74.495917	1.3.1	host	USB	31	URB_INTERRUPT in
170	74.495981	host	1.3.1	USB	27	URB_INTERRUPT in
171	78.551908	1.3.1	host	USB	31	URB_INTERRUPT in
172	78.551969	host	1.3.1	USB	27	URB_INTERRUPT in

Nota: La figura muestra el tráfico de datos inicial al conectar un mouse a la computadora. En rojo se muestra el proceso de Enumeración, en el cual la computadora le solicita al dispositivo USB información sobre su configuración, sus necesidades eléctricas, su forma general de funcionar, además le asigna una dirección genérica (1.3.0). En naranja se muestra como la computadora le pide al dispositivo USB información a detalle de su comportamiento como dispositivo de clase HID. Una vez se ha terminado de configurar todo, el dispositivo le asigna una dirección definitiva (1.3.1), con esto, el dispositivo y la computadora están listos para enviar y recibir información de la aplicación de funcionamiento del dispositivo USB. Fuente propia.

Al realizar un análisis más profundo de este tráfico, el cual se realiza en el ANEXO B, podemos obtener la información más importante que nos servirá como punto de partida.

En primer lugar, revisamos los descriptores principales del dispositivo. Podemos ver que el dispositivo, efectivamente, está configurado como un mouse, el cual consume un máximo de 100mA, además de solo contar con una configuración, interfaz y Endpoint. La información más importante de estos descriptores es la clase de dispositivo, clase de interface, el tipo de transferencia del Endpoint, el tipo de la transferencia que usa el Endpoint y la longitud del Endopoint.

Tabla 22
Información del mouse de prueba obtenida mediante WireShark - USB Cap

Descriptor de Dispositivo	
Clase del Dispositivo	Especificado en el descriptor de Interface
ID de Fabricante	0x046D (Logitech)
Numero de Configuraciones	1
Descriptor de Configuración	
Numero de Interfaces	1
Modo de Funcionamiento	No Auto Alimentado
Consumo Máximo	100mA
Descriptor de Interface	
Número de Puntos Finales	1
Clase de Interface	HID
Protocolo de Interface	Mouse
Descriptor de Punto Final	
Flujo de EndPoint	Entrada
Numero de EndPoint	1
Tipo de Transferencia	Transferencia de Interrupción
Tamaño Máximo	4 bytes

En segundo lugar, observamos el descriptor especifico de los dispositivos que son configurados con clase HID. Este descriptor de reporte HID, le entrega al Host una información más detallada de la estructura que tiene el Endpoint 1, por lo que el Host sabrá el orden y la longitud de los datos que le llegaran desde el dispositivo ya que es esencialmente una plantilla que describe la estructura de datos que se envía y/o recibe entre el Host y el dispositivo, es importante indicar que este descriptor se transmite siguiendo una sintaxis completamente estandarizada entre mnemónicos y hexadecimal, es decir, que es similar al lenguaje Assembler y el lenguaje maquina en hexadecimal.

Tabla 23
Descriptor HID del mouse de prueba obtenida mediante WireShark - USB Cap

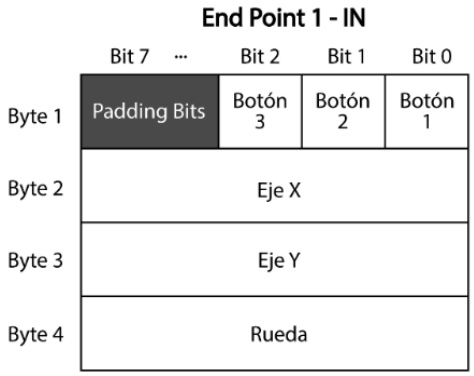
Descriptor de Reporte HID	
Código	Hexadecimal
USAGE_PAGE (Generic Desktop)	05 01
USAGE (Mouse)	09 02
COLLECTION (Application)	A1 01
USAGE (Pointer)	09 01
COLLECTION (Physical)	A1 00

Descriptor de Reporte HID	
USAGE_PAGE (Button)	05 09
USAGE_MINIMUM (Button 1)	19 01
USAGE_MAXIMUM (Button 3)	29 03
LOGICAL_MINIMUM (0)	15 00
LOGICAL_MAXIMUM (1)	25 01
REPORT_COUNT (3)	95 03
REPORT_SIZE (1)	75 01
INPUT (Data,Var,Abs)	81 02
REPORT_COUNT (1)	95 01
REPORT_SIZE (5)	75 05
INPUT (Cnst,Var,Abs)	81 03
USAGE_PAGE (Generic Desktop)	05 01
USAGE (X)	09 30
USAGE (Y)	09 31
USAGE (Wheel)	09 38
LOGICAL_MINIMUM (-127)	15 81
LOGICAL_MAXIMUM (127)	25 7F
REPORT_SIZE (8)	75 08
REPORT_COUNT (3)	95 03
INPUT (Data,Var,Rel)	81 06
END_COLLECTION	C0
END_COLLECTION	C0

La información más importante que se puede extraer de este descriptor, la podemos observar en la **Figura 76**, como habíamos indicado antes, este descriptor es una plantilla de la estructura de los datos que serán enviados al Host, esta información es muy útil para nuestro propósito ya que nos permitirá conseguir el objetivo de replicar el comportamiento de un mouse en nuestro dispositivo.

Figura 76

Descriptor de Reporte HID indicando la estructura del Endpoint 1.



Nota: La figura muestra la estructura que describe el reporte HID, esto también se puede definir como una plantilla que indica la manera en la que el Host debe leer los datos que se generan con el funcionamiento del mouse, asignando un significado a cada porción de la memoria del dispositivo USB. Fuente propia.

1.1.1. Botones

Cada botón ocupa un bit, debido a que cada botón solo tiene dos estados, 1 o 0, se debe observar que quedan 5 bits sin utilizar, esto se debe a que el mouse utilizado para las pruebas solo cuenta con funciones básicas, los dispositivos con más botones, utilizaran estos bits al momento de enviar información al Host. La computadora se encarga de interpretar estos estados y decide como realizar la acción de respuesta, pudiendo configurarse si la acción se realiza en el flanco de bajada o de subida, siendo este ultimo la que está establecido por defecto.

Figura 77

Captura de tráfico de datos USB con Wireshark. Activación de botones.

```
HID Data: 01000000
    .... ...1 = Button: 1 (primary/trigger): DOWN
    .... ..0. = Button: 2 (secondary): UP
    .... .0.. = Button: 3 (tertiary): UP
    Padding: 00
    0000 0000 = X Axis: 0
    0000 0000 = Y Axis: 0
    0000 0000 = Usage: Wheel: 0
```

```
HID Data: 02000000
    .... ...0 = Button: 1 (primary/trigger): UP
    .... ..1. = Button: 2 (secondary): DOWN
    .... .0.. = Button: 3 (tertiary): UP
    Padding: 00
    0000 0000 = X Axis: 0
    0000 0000 = Y Axis: 0
    0000 0000 = Usage: Wheel: 0
```

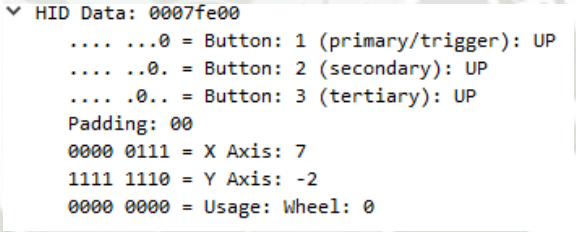
```
HID Data: 04000000
    .... ...0 = Button: 1 (primary/trigger): UP
    .... ..0. = Button: 2 (secondary): UP
    .... .1.. = Button: 3 (tertiary): DOWN
    Padding: 00
    0000 0000 = X Axis: 0
    0000 0000 = Y Axis: 0
    0000 0000 = Usage: Wheel: 0
```

Nota: La figura muestra el intercambio de paquetes USB entre el dispositivo USB y el Host, de izquierda a derecha y de abajo hacia arriba, se muestra en la primera imagen, un paquete correspondiente a una presión del botón click izquierdo, en la segunda una del botón click derecho y la tercera, del botón de la rueda central del mouse. Fuente propia.

1.1.2. Ejes X y Y

Los bytes que representan el movimiento en los ejes X o Y, indican el movimiento relativo, esto significa que los datos transmitidos indican cuanto se ha movido el mouse desde la última vez que se enviaron datos, no la posición exacta del cursor en la pantalla. Se puede hacer una analogía con la velocidad, sin embargo, se debe tener cuidado al interpretar esto, ya que los valores de los bytes de los ejes no están expresados en unidades de distancia y tiempo, están asociados a la unidad DPI.

Figura 78
Captura de tráfico de datos USB con Wireshak. Movimiento de mouse.



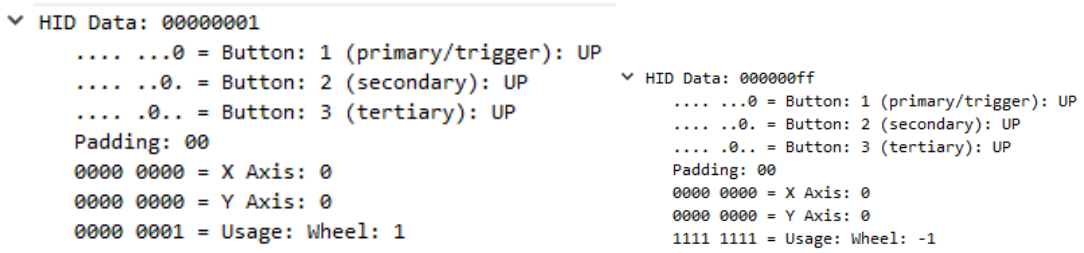
Nota: La figura muestra el intercambio de paquetes USB entre el dispositivo USB y el Host cuando hay un cambio en la posición del mouse, en cualquier eje. Fuente propia.

1.1.3. Rueda

El movimiento de la rueda se interpreta igual que en los ejes X y Y, sin embargo, la diferencia radica en cómo se generan estos datos, para los ejes, el mouse tiene un chip que se encarga de realizar un procesamiento de imagen para determinar el desplazamiento del mouse, debido a que este chip puede realizar este procesamiento a un ritmo superior al que se envían estos datos, es posible que los valores de los ejes sean diferentes de la unidad. En contraste, para generar un cambio en el movimiento de la rueda se usa un enconder, por lo que el byte asignado a la rueda solo varia de -1 hasta 1.

Figura 79

Captura de tráfico de datos USB. Giro de rueda.



Nota: La figura muestra el intercambio de paquetes USB entre el dispositivo USB y el Host al girar la rueda del mouse, en la izquierda se muestra el paquete que se envía cuando la rueda gira hacia adelante, en la derecha se muestra cuando gira hacia atrás. Fuente propia.

1.2. PIC 18F2550

Este microcontrolador se destaca por ser parte de la gama alta dentro de la serie de los PIC, proporcionando un conjunto de características avanzadas que facilitan la elaboración de aplicaciones complejas.

Pertenece a la familia 18FX550/X455, los cuales tienen como principal atributo contar con un Motor Serial de Interface USB (USB SIE) y un transceptor interno, los cuales permiten una comunicación rápida con cualquier Host USB, esta familia es compatible con la versión de USB 2.0, soporta las velocidades baja (1.5Mb/s) y completa (12Mb/s), puede operar con transferencias de tipo Control, Interrupción, Isocrónicas y de Masa, otra característica importante es que cuenta con 32 Puntos Finales (16 bidireccionales).

El PIC18F2550 es el microcontrolador más pequeño de esta familia, siendo la principal diferencia de contar con menor número de pines de GPIO, sin embargo, esto no afecta el propósito de nuestra aplicación, al contrario, nos permite miniaturizar nuestro prototipo.

Tabla 24
Descripción general del PIC18F2550.

PIC 18F2550	
Pines	28
Pines GPIO	24
ADC	10 canales – 10 bits
Soporte USB Embebido	USB 2.0
Memoria Flash	32KB
Memoria SRAM	2048 Bytes

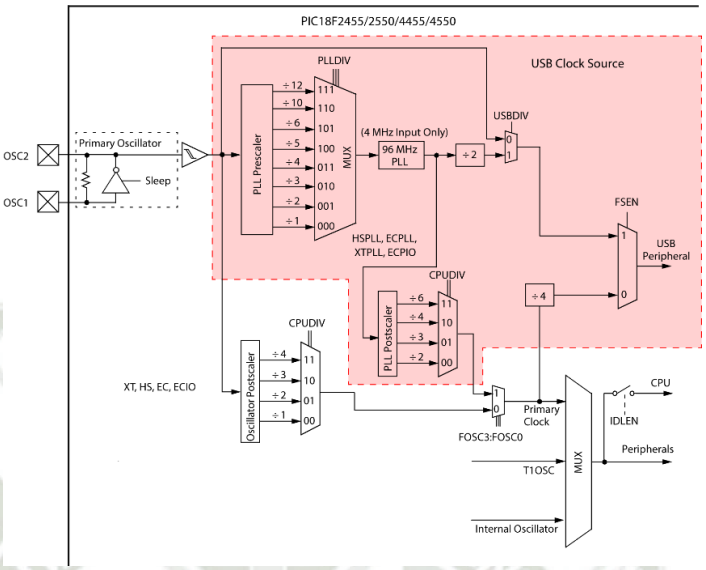
El apartado que analizaremos más a profundidad es el módulo USB, en la **Figura 81** podemos observar que el módulo tiene embebido todo lo necesario para funcionar sin necesidad de circuitos externos adicionales, aprovecharemos esta característica para miniaturizar lo más posible el prototipo.

1.2.1. Motor USB Integrado

Para el correcto funcionamiento del módulo USB, es crucial seleccionar adecuadamente la frecuencia del oscilador primario y el valor del pre escalador del PLL. En la sección 3.3 del capítulo previo, se estableció que la frecuencia de operación estándar para USB 2.0 es de 48 MHz. El PIC cuenta con un arreglo de hardware que, al recibir una entrada de 4 MHz, garantiza una salida de 48 MHz. Por razones de practicidad, se optó por emplear un oscilador externo de 20 MHz, lo que requiere el ajuste del pre escalador para obtener una división por cinco. Además, según se observa en la **Figura 80**, el ajuste del reloj del módulo USB sólo puede realizarse mediante el uso del oscilador externo; cualquier intento de emplear el oscilador interno del PIC resultará en errores en el funcionamiento del módulo USB.

Figura 80

Diagrama de bloques del oscilador del PIC.

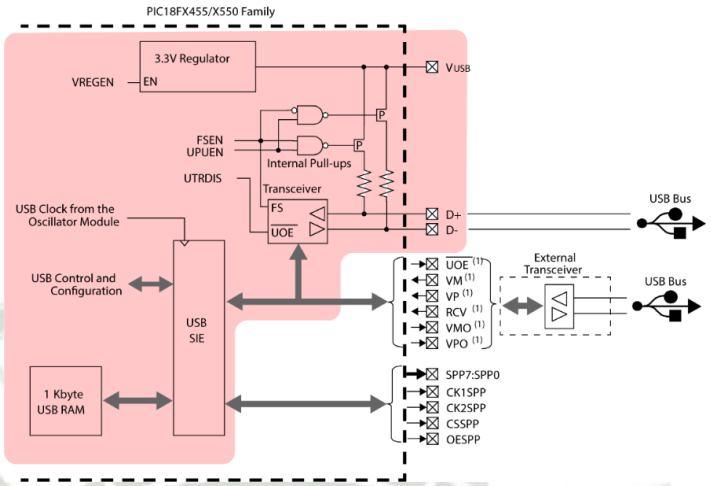


Nota: La figura muestra la estructura interna del reloj del PIC18F2550 necesarios para llegar a la frecuencia con la que opera el protocolo USB, el cual es de 48MHz. Fuente: Hoja de datos PIC18F2550 - Microchip.

Al observar los complementos que acompañan al motor USB del PIC, observamos que, si se configura adecuadamente, no es necesario acoplar hardware adicional para el funcionamiento del USB 2.0. En el estándar USB que se analizó en la sección 3.3 del capítulo previo, indica que para determinar si la comunicación del USB se realizara a baja o alta velocidad, se debe colocar una resistencia Pull Up en los pines D+ o D- dependiendo de la velocidad que queremos seleccionar, gracias a que el módulo USB ya tiene todo esto integrado, podemos elegir la velocidad adecuada mediante software.

Adicionalmente, el PIC integra un regulador de voltaje a 3.3V el cual se usa para la transmisión de las señales USB a través de los pines D+ y D-, al igual que con las resistencias Pull Up, no es necesario agregar regulador externo adicional.

Figura 81
Modulo USB embebido



Nota: La figura muestra la estructura interna del módulo USB embebido del PIC18F2550, este integra un regulador de voltaje resistencias Pull-up internas necesarias para estabilizar las señales en las líneas de datos. Fuente: Hoja de datos PIC18F2550 - Microchip.

Para manejar adecuadamente el módulo USB, es crucial entender y configurar correctamente los registros que regulan su funcionamiento. La **Tabla 25** detalla estos registros importantes y proporciona los valores óptimos para su configuración. Conocer la función y la interacción de estos registros es vital para garantizar un rendimiento eficiente del módulo USB, permitiendo ajustes precisos que mejoran la compatibilidad y la funcionalidad del sistema en diferentes escenarios operativos.

Tabla 25
Registros del Módulo USB.

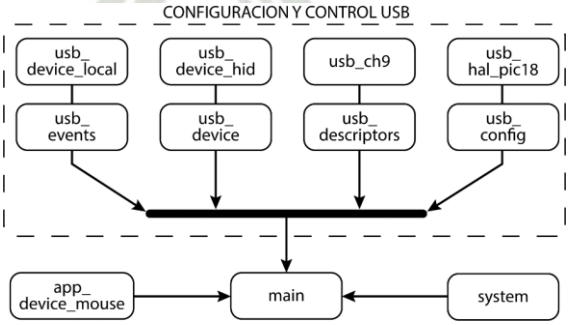
	Bits de Configuración	
CONFIG1L	USBDIV=1	La fuente de reloj viene del PLL de 48MHz.
	CPUDIV=0	El reloj del sistema se ajusta a 48MHz.
	PLLDIV=4	El oscilador externo se divide entre 5 para alimentar el PLL.
CONFIG1L	FOSC=16	Se elige oscilador externo HS (High Speed), habilitando el PLL.
CONFIG2L	VREGEN=1	Habilita el regulador de voltaje interno para las líneas del USB.
Registros USB UCFG	UPUEN=1	Habilita las resistencias Pull Up en las líneas USB.
	FSEN=1	Selecciona la velocidad Full Speed.

Bits de Configuración	
	UTRDIS=0 Habilita el transpondedor del USB.
USTAT	Registro de estado USB. Indica el ultimo EP que tuvo actividad.
UEPn	Registro de control de los EPn (EP0 al EP15). Configura el comportamiento de los EP.
UIR	Registro de estado de interrupciones. Indica información como el estado del SOF, detección de actividad en las líneas, indicación de transacción completa, etc.

1.3. Librería de Microchip para Aplicaciones (MLA)

La biblioteca para aplicaciones de Microchip (MLA, por sus siglas en inglés) proporciona una colección integral de software y documentación diseñada para facilitar la implementación de aplicaciones específicas como USB, TCP/IP, y sistemas gráficos, entre otras. Esta compilación está disponible en el sitio web oficial de Microchip (microchip.com) y tiene como objetivo demostrar las capacidades de sus microcontroladores PIC en contextos aplicados específicos. Para nuestro estudio, nos centraremos exclusivamente en la aplicación USB proporcionada por la MLA, analizando detalladamente su estructura y funcionalidad para evaluar su eficacia y aplicabilidad en escenarios prácticos. Esto nos permitirá aprovechar la librería de MLA para optimizar el rendimiento y la implementación de tecnologías USB nuestro prototipo.

Figura 82
Archivos relevantes del proyecto de MLA.



Nota: La figura muestra un diagrama de resumen de todos los archivos relevantes involucrados en el proyecto de Demo USB de la librería MLA de microchip, para tener un mejor orden en el proyecto USB, se eliminará cualquier archivo que no esté en este diagrama. Fuente propia.

El proyecto que desarrollamos, hace uso del módulo USB proporcionado por la biblioteca de Microchip Application Libraries (MLA), este repositorio está diseñado para adaptarse a la serie de microcontroladores PIC 18FX455/X550, además de otros integrados. Dado que la MLA debe ser compatible con una amplia gama de dispositivos, el código suministrado por la biblioteca es notablemente extenso, lo que puede complicar su implementación debido a la necesidad de ajustarse a diferentes hardware. Esta amplia compatibilidad conlleva una complejidad tanto en el código como en la estructura del proyecto, manifestada en un número significativo de archivos que pueden dificultar la gestión y el entendimiento del mismo.

Tras una evaluación del módulo USB de la librería MLA, se han identificado los archivos más importantes para la configuración inicial y el adecuado funcionamiento de la comunicación USB. Estos archivos esenciales se detallan en la **Figura 82**. Adicionalmente, el archivo “app_device_mouse”, tendrá el código que determine comportamiento de nuestro prototipo, que se muestra en la **Figura 84**. Para proporcionar una claridad adicional, la **Tabla 26** ofrece una descripción de los archivos involucrados en el proyecto, indicando su propósito y relevancia.

Tabla 26
Descripción de archivos que integran el proyecto.

Archivo	Descripción
usb_ch9.h	Define macros para configurar los descriptores. Define estructuras para crear los descriptores. Define estructuras para el control de envío los descriptores.
usb_descriptors.c	Define los descriptores y sus valores respecto del dispositivo.
usb_device_hid.c/.h	Controla y responde la petición del descriptor HID.
usb_device.c/.h	Crea variable en dirección fija para posicionar el EP0 de configuración. Crea define funciones necesarias para el inicio del módulo USB.
usb_config.h	Define macros para configurar velocidad del USB, código VIP y PID, tiempo de espera, cantidad de descriptores String.

Archivo	Descripción
	Permite deshabilitar eventos.
usb_hal_pic18.h	Configuración de los registros USB del PIC18F
usb__device_local.h	Define macros para manejar los EP disponibles en el PIC18F.
	Define macros para ejecutar eventos específicos.
usb_events.c	Recepciona los eventos y llama a la función que se encarga de responder al evento.
app_device_mouse.c/.h	Define el reporte HID.
	Se encarga de la aplicación física del dispositivo USB.
system.c/.h	Define los bits de configuración.

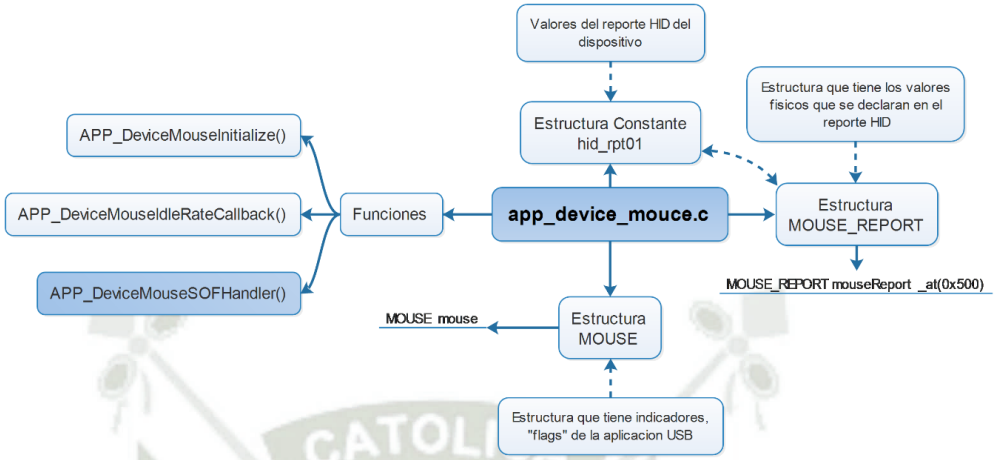
Como se mencionó previamente, el archivo “app_device_mouse.c” desempeña un papel importante, ya que alberga la programación para el funcionamiento de la aplicación del proyecto. Es en este archivo donde se implementará el código que define el comportamiento específico de nuestro prototipo. Adicionalmente, este archivo contiene el descriptor HID, el cual define la estructura o plantilla de datos del dispositivo, también estructura y almacena un informe que detalla los valores que serán transmitidos a través del USB.

Específicamente, este archivo incorpora funciones que permiten la operación del dispositivo USB:

1. APP_DeviceMouseInitialize(): Esta función es responsable de inicializar el dispositivo mouse, configurando el entorno necesario para su funcionamiento dentro del sistema.
2. APP_DeviceMouseIdleRateCallback(): Función que gestiona las tasas de inactividad del dispositivo, permitiendo ajustes que pueden optimizar el rendimiento del sistema basado en las condiciones operativas.
3. APP_DeviceMouseSOFHandler(): Encargada de manejar eventos específicos del USB, esta función es crítica para la correcta respuesta del dispositivo a las señales de inicio de trama (SOF, Start of Frame), asegurando una sincronización precisa y eficiente en la transferencia de datos.

Figura 83

Estructura del contenido del archivo "app_device_mouce.c"



Nota: La figura muestra un diagrama de resumen de las funciones relevantes del archivo "app_device_mouce.c", este archivo tiene las funciones y variables que definen directamente el comportamiento del PIC en modo mouse USB. Fuente propia.

Entre las funciones que se definen en el archivo "app_device_mouse.c", se considera que "APP_DeviceMouseSOFHandler()" es la más importante para nuestro propósito, ya que transmite los valores de la estructura MOUSE_REPORT por USB. Esta estructura almacena el estado actual del dispositivo, y se envían para ejecutar acciones en el Host. Es aquí donde implementaremos el código de nuestro prototipo, asegurando que las interacciones del usuario se procesen y envíen eficientemente al Host. En el siguiente apartado se analizará como el código que se introducirá en esta función.

1.4. Interfaz USB Propia

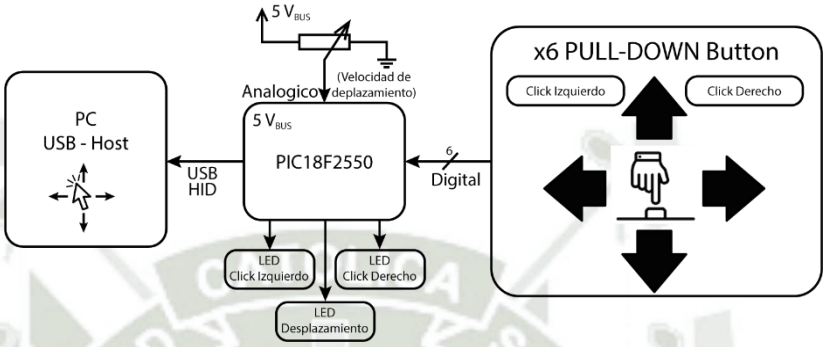
Para la realización de nuestra Interfaz Humano-Computador, simularemos el comportamiento esencial de un mouse, teniendo únicamente 2 botones, el eje X y el eje Y.

La sensibilidad de desplazamiento del USB se ajustará mediante un potenciómetro, este tendrá que ser ajustado por otra persona a pedido del usuario del prototipo. Para ver el estado de lo que está realizando el prototipo, se establecen tres indicadores LED, que

reflejara la interacción con el click izquierdo, derecho y el movimiento del puntero en cualquier dirección.

Figura 84

Arquitectura de la Interfaz USB del Prototipo

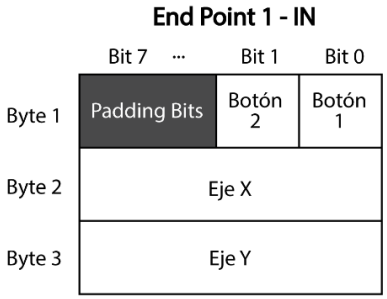


Nota: La figura muestra la estructura general que tendrá el módulo USB de nuestro prototipo, esta estructura está diseñada de tal manera que pueda conectarse y ser controlada de manera fácil por el módulo de procesamiento de señales EOG, el control del movimiento y los botones del mouse se realizará mediante 6 entradas digitales, el control de velocidad de desplazamiento del puntero se definiera mediante un potenciómetro. Fuente propia.

De la Figura superior se observa que el prototipo recibe y envía varios parámetros, sin embargo, no todos se enviaran hacia el Host. Es importante identificar que únicamente los parámetros de dirección y estado de los botones son los que se enviaran, por lo tanto, estos se deben estructurar para poder ser transmitidos con el Reporte HID y ubicarlos en el End Point 1.

Figura 85

Estructura del End Point 1



Nota: La figura muestra la estructura del reporte HID que debe tener el módulo USB del prototipo, en esta se indicara al Host que el prototipo tiene 2 botones y 2 datos de posición, una para cada eje. Fuente propia.

En la **Tabla 27** se resalta de color rojo el código en hexadecimal que representa la estructura que tendrá los datos del End Point 1, esta misma información se muestra de forma gráfica en la **Figura 85**, como mencionamos antes, el valor del potenciómetro y el estado de los indicadores LED no serán enviados al Host, si no, serán manejados internamente en el PIC

Tabla 27
Descriptor de Reporte HID del prototipo.

Descriptor de Reporte HID	
USAGE_PAGE (Generic Desktop)	05 01
USAGE (Mouse)	09 02
COLLECTION (Application)	A1 01
USAGE (Pointer)	09 01
COLLECTION (Physical)	A1 00
USAGE_PAGE (Button)	05 09
USAGE_MINIMUM (Button 1)	19 01
USAGE_MAXIMUM (Button 2)	29 02
LOGICAL_MINIMUM (0)	15 00
LOGICAL_MAXIMUM (1)	25 01
REPORT_COUNT (2)	95 02
REPORT_SIZE (1)	75 01
INPUT (Data,Var,Abs)	81 02
REPORT_COUNT (1)	95 01
REPORT_SIZE (6)	75 06
INPUT (Cnst,Var,Abs)	81 03
USAGE_PAGE (Generic Desktop)	05 01
USAGE (X)	09 30
USAGE (Y)	09 31
LOGICAL_MINIMUM (-127)	15 81
LOGICAL_MAXIMUM (127)	25 7F
REPORT_SIZE (8)	75 08
REPORT_COUNT (2)	95 02
INPUT (Data,Var,Rel)	81 06
END_COLLECTION	C0
END_COLLECTION	C0

Tabla 28
Descriptores de configuración del prototipo.

Descriptor de Dispositivo	
Clase del Dispositivo	Especificado en el descriptor de Interface
ID de Fabricante	0x04D8 (Microchip)
Numero de Configuraciones	1
Descriptor de Configuración	
Numero de Interfaces	1
Modo de Funcionamiento	No Auto Alimentado
Consumo Máximo	100mA
Descriptor de Interface	
Número de Puntos Finales	1
Clase de Interface	HID
Protocolo de Interface	Mouse
Descriptor de Punto Final	
Flujo de EndPoint	Entrada
Numero de EndPoint	1
Tipo de Transferencia	Transferencia de Interrupción
Tamaño Máximo	3 bytes

El siguiente paso en el desarrollo de nuestro prototipo es la asignación de valores para los descriptores de configuración. Dado que nuestra aplicación es semejante a la descrita en la sección 1.1 de este capítulo, es posible adoptar valores similares para los descriptores de nuestro prototipo. Esto es posible debido a que las diferencias entre nuestro dispositivo y el previamente estudiado, radican principalmente en una reducción del volumen de datos transmitidos, así como el cambio del código de identificación del fabricante (VID) y el código de producto (PID), entre otros menos significativos.

1.4.1. Programación en MPLAB X IDE

Para integrar eficiente el software y el hardware de nuestro prototipo, debemos asignar los pines físicos a sus respectivas funciones, esta tarea se llevará a cabo teniendo en cuenta la **Figura 84**. Además, otro criterio importante en la asignación de pines es la proximidad entre los demás componentes del circuito impreso (PCB), esto permite optimizar las conexiones y minimizar interferencias.

El diseño del PCB ha pasado por múltiples iteraciones para ajustar la disposición de los componentes y las pistas, estos ajustes se reflejan en los valores finales que se muestran en la **Tabla 29**.

Tabla 29
Asignación de pines para el PIC 18F2550.

DESCRIPCION	PUERTO	Valor	DIRECCION
X+	RB4	1	
X-	RB5	-1	
Y+	RB3	1	
Y-	RB2	-1	ENTRADAS
Click Izquierdo	RB1	-	
Click Derecho	RB0	-	
Velocidad (Potenciómetro)	RA0	1 - 127	
LED Click Izquierdo	RC0	-	
LED Movimiento	RC1	-	SALIDA
LED Click Derecho	RC2	-	

Para mejorar la interpretación y facilitar el mantenimiento del código, se usan macros que definen la configuración inicial. De esta manera no solo se mejora la comprensión del código, sino que también permite ajustes flexibles de los valores, facilitando cambios rápidos en el software sin alteraciones extensivas del código fuente reduciendo la posibilidad de generar errores.

Tabla 30
Configuración de pines físicos.

Configuración de pines físico en XC16	
#define X_Right_TRIS	TRISBbits.RB4=1
#define X_Left_TRIS	TRISBbits.RB5=1
#define Y_Up_TRIS	TRISBbits.RB3=1
#define Y_Down_TRIS	TRISBbits.RB2=1
#define Click_Left_TRIS	TRISBbits.RB1=1
#define Click_Right_TRIS	TRISBbits.RB0=1
#define Pot_Speed_TRIS	TRISAbits.RA0=1
#define LED_Left_TRIS	TRISCbits.TRISC0=0
#define LED_Move_TRIS	TRISCbits.TRISC1=0
#define LED_Right_TRIS	TRISCbits.TRISC2=0

Nota: Ignorar ausencia de tildes en la tabla, el IDE de programación no reconoce símbolos con tildes.

Una vez que se asignado las funciones a sus pines correspondientes, procedemos a implementar el código que determine el comportamiento que nuestro prototipo debe tener, en la **Figura 86** se muestra un diagrama de flujo simplificado de cómo debe realizarse el código.

Tabla 31
Detección de movimiento y dirección de ejes.

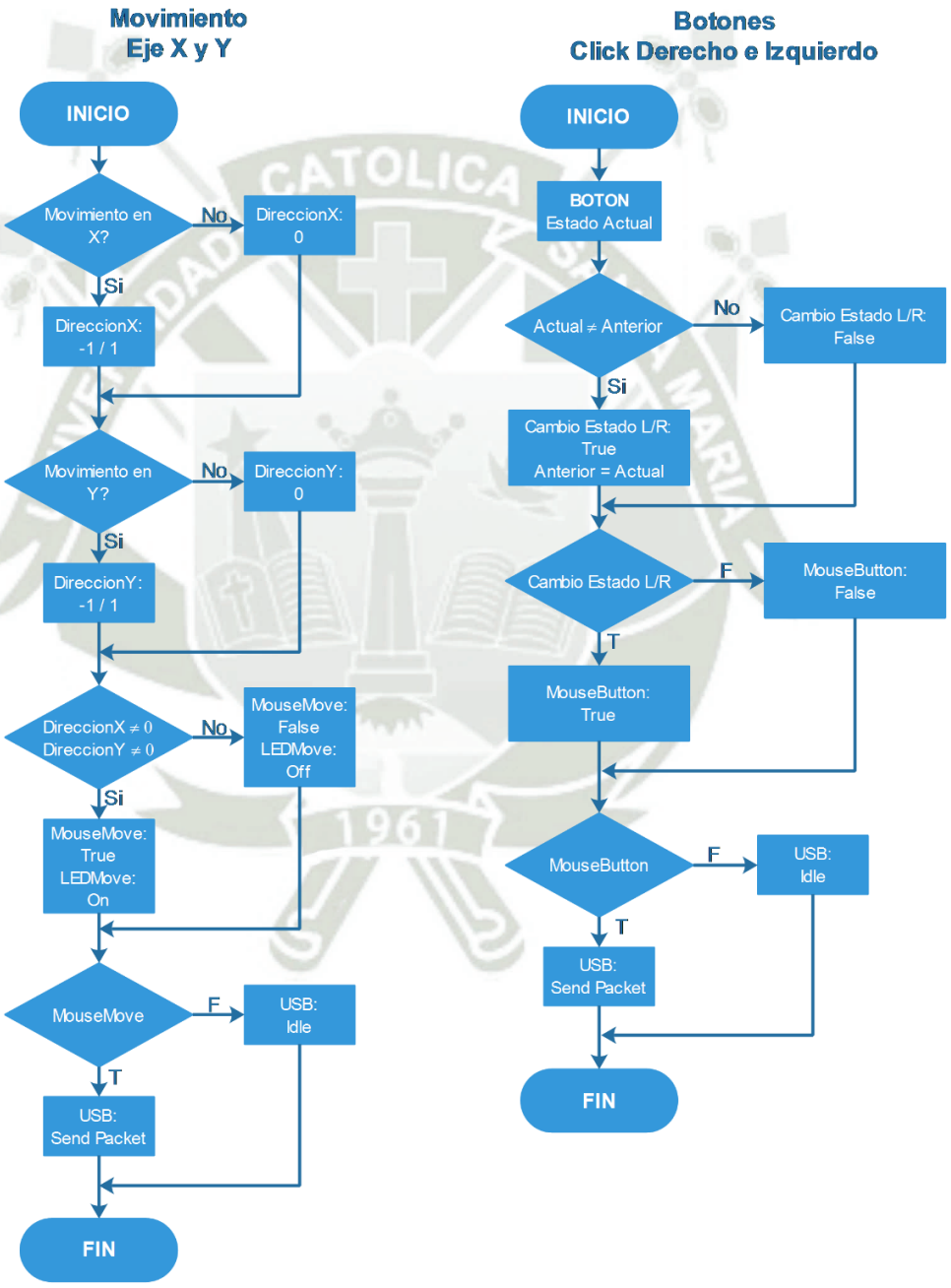
Detección de movimiento y dirección de ejes en XC16

```
uint8_t x_axis=0;
uint8_t y_axis=0;

if(Is_BUTTON_Pressed(Boton_X_Left) && !Is_BUTTON_Pressed (Boton_X_Right))
    {x_axis=-1; }//x-
if(!Is_BUTTON_Pressed(Boton_X_Left) &&  Is_BUTTON_Pressed(Boton_X_Right))
    {x_axis=1; }//x+

if(Is_BUTTON_Pressed(Boton_Y_Down) && !Is_BUTTON_Pressed(Boton_Y_Up))
    {y_axis=-1;} //y-
if(!Is_BUTTON_Pressed(Boton_Y_Down) && Is_BUTTON_Pressed(Boton_Y_Up))
    {y_axis=1;}//y+

if(Is_BUTTON_Pressed(Boton_X_Left) == Is_BUTTON_Pressed(Boton_X_Right))
    {x_axis=0;}//x0
if(Is_BUTTON_Pressed(Boton_Y_Down) == Is_BUTTON_Pressed(Boton_Y_Up) )
    {y_axis=0;} //y0

if(x_axis!=0||y_axis!=0) {
mouse.movementMode = true;
LED_On(LED_Move);}
if(x_axis==0&&y_axis==0) {
mouse.movementMode = false;
LED_Off(LED_Move);}
```

Nota: Ignorar ausencia de tildes en la tabla, el IDE de programación no reconoce símbolos con tildes.

La orden de movimiento de los ejes se recibirá del dsPIC a través de los pines de entrada del PIC18F, por este motivo, se puede trabajar como si las entradas fuesen botones con resistencia Pull Up, haciendo uso de esta lógica se realiza el código de la **Tabla 31**.

Figura 86
Flujograma del código para detectar movimiento y uso de botones.



Nota: La figura muestra el flujograma del algoritmo que detecta el movimiento del mouse y la activación de los botones del mouse. Fuente propia.

Para la detección de las pulsaciones de los botones, es importante enviar al Host un único pulso representativo del estado de los botones, evitando así la saturación del bus USB con datos repetitivos. Esto implica emitir un "Click" en el flanco de bajada y finalizarlo en el flanco de subida. De esta forma, solo se transmiten cambios significativos en el estado de los botones, optimizando el uso del bus USB y permitiendo que el Host interprete la ejecución del "Click". Para conseguir enviar el pulso solo en los flancos, se debe usar una variable que guarde el estado anterior y actual del botón, la comparación de estas dos variables dará como resultado la detección de los flancos, esto se puede observar en la **Figura 86**.

Tabla 32
Detección de flancos de los botones.

Detección de flancos de los botones

```
// Guardar el estado actual del botón
bool estado_actual = Is_BUTTON_Pressed(Boton_Click_Left);

// Verificar si ha habido un cambio en el estado del botón
if (estado_actual != mouse.Click_Left_Anterior) {
   mouse.Click_Left_Cambio = true;
   mouse.Click_Left_Anterior = estado_actual;

   // Establecer el estado del botón en el informe del mouse
   mouseReport.buttons.button1 = estado_actual ? 1 : 0;
} else {
   mouse.Click_Left_Cambio = false;}

// Encender o apagar el LED según el estado del botón
if(estado_actual)  LED_On(LED_Left); else LED_Off(LED_Left); //Luces Click izquierdo

SE REPITE LO MISMO PARA EL BOTON DERECHO Y SE TERMINA CON LAS SIGUIENTES LINEAS:

if(mouse.Click_Right_Cambio||mouse.Click_Left_Cambio)
    // Establecer el modo de clic
   mouse.clicktMode = true; else mouse.clicktMode = false;
```

Nota: Ignorar ausencia de tildes en la tabla, el IDE de programación no reconoce símbolos con tildes.

Igual que el caso anterior, para evitar saturar el bus USB con información redundante o sin importancia (como cuando no hay actividad en el mouse), solo se debe de mandar paquetes por el bus USB cuando haya actividad en el mouse, esto se logra con los flags "mouse.clicktMode" y "mouse.movementMode", estos solo se activan cuando hay movimiento o se activó algún botón, estos flags nos permite condicionar el envío de los paquetes.

Tabla 33
Condicionamiento de la transmisión de paquetes al bus USB.

Condicionamiento de la transmisión de paquetes al bus USB

if(mouse.movementMode || mouse.clicktMode)
{
mouseReport.x = x_axis;
mouseReport.y = y_axis;
 	
mouse.inputReport[0].handle = HIDTxPacket(
HID_EP,
(uint8_t*)&mouseReport,
sizeof(mouseReport)
);
}

Nota: Ignorar ausencia de tildes en la tabla, el IDE de programación no reconoce símbolos con tildes.

2. PROCESAMIENTO DIGITAL DE SEÑALES CON DSPIC

2.1. DSPIC 33CH128MP506

El dsPIC33CH128MP506 es parte de la familia dsPIC33CH de Microchip Technology. Esta familia de microcontroladores fue introducida al mercado alrededor del año 2018. Los microcontroladores dsPIC33CH están diseñados específicamente para aplicaciones embebidas que requieren un alto rendimiento y capacidad de procesamiento digital de señales, tanto en aplicaciones industriales como controladores de motores,

sistemas de control de potencia, aplicaciones automotrices y aplicaciones biomédicas (Mitchell, 2018).

Microchip Technology ha continuado desarrollando y mejorando la familia de los dsPIC desde su lanzamiento inicial, incorporando características avanzadas y mejorando la eficiencia de los microcontroladores para satisfacer las crecientes demandas de las aplicaciones modernas. El dsPIC33CH128MP506, como uno de los modelos dentro de esta familia, se beneficia de estos avances continuos en tecnología y diseño de microcontroladores.

Tabla 34
Descripción general del dsPIC33CH128MP506

DSPIC 33CH128MP506	
Pines	64
Pines GPIO	24
ADC	16 canales – 12 bits – 4 módulos
DAC	1 canal – 10 bits
Motor DSP	Si Operaciones implementadas por Hardware: Suma de 40 bits Multiplicación de 32 bits División de 32 bits
Memoria Flash	32KB
Memoria SRAM	2048 Bytes

Debido a que en el PIC18F2550 se usa un oscilador a 20MHz para lograr que este alcance los 48MHz requeridos para el protocolo USB, en el microcontrolador que se usa en este apartado se utilizara también un oscilador de 20MHz, por lo que, los valores con los que operara el dsPIC son:

Oscilador Primario (P_{OSC}): 20 MHz
Ciclo de instrucción (F_{cy}): 10 MHz

Tabla 35

Distribución de entradas y salidas.

DESCRIPCION	PUERTO	DIRECCION
X+ (Derecha)	RB15	ENTRADA
X- (Izquierda)	RB14	
Y+ (Arriba)	RB13	
Y- (Abajo)	RB12	
Click Izquierdo	RB9	
Click Derecho	RB8	
Channel 1	RC7 – AN15	
Channel 2	RC1 – AN13	
DAC Out H	RB10	SELECCIONADOR DE SALIDA DEL DAC
DAC Out V	RC11	
LED Funcionamiento	RA2	SALIDA

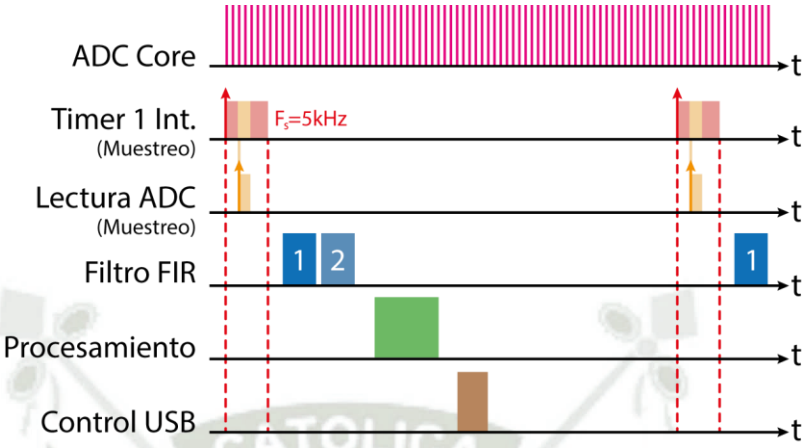
Dentro del microcontrolador dsPIC, además del código de programación, se integrarán varios módulos independientes que trabajarán de manera conjunta para alcanzar los objetivos del prototipo. A diferencia del módulo ADC, que opera de manera continua para digitalizar señales analógicas, cada uno de los otros módulos involucrados deberá activarse en un tiempo y orden específicos para asegurar un funcionamiento coordinado y eficiente.

Para asegurar que todos estos módulos trabajen de manera eficiente y coordinada, es crucial establecer una estrategia de gestión de tiempo y orden. El controlador de interrupciones, junto con el temporizador, desempeñan un papel central en esta coordinación. Por ejemplo, al completar la lectura del ADC, se puede generar una interrupción que desencadene el procesamiento inmediato de los datos convertidos, evitando retrasos y asegurando una respuesta rápida del sistema.

La integración efectiva de estos módulos, junto con una programación cuidadosa, permite que el dsPIC maneje tareas complejas y tiempo-críticas de manera eficiente. Esta capacidad es especialmente útil en aplicaciones bio médicas industriales y de control donde la precisión y la respuesta rápida son esenciales.

Figura 87

Cronología de eventos en el dsPIC para el filtrado FIR y el control USB



Nota: La figura muestra la cronología de eventos que suceden para el funcionamiento del prototipo, El ADC del dsPIC funciona independientemente a una velocidad elevada, se determina que la lectura del valor analógico (tiempo de muestreo) sea a 5kHz, en el tiempo restante entre lectura y lectura, se debe realizar la aplicación de los filtros FIR en cada canal, además de actualizar la memoria de los valores anteriores de la señal, también debe evaluar el algoritmo que relaciona la señal EOG con alguna acción de movimiento del mouse y por último, mandar la orden del movimiento previa evaluación al módulo USB, todo este proceso se repita indefinidamente a 5kHz. Fuente propia.

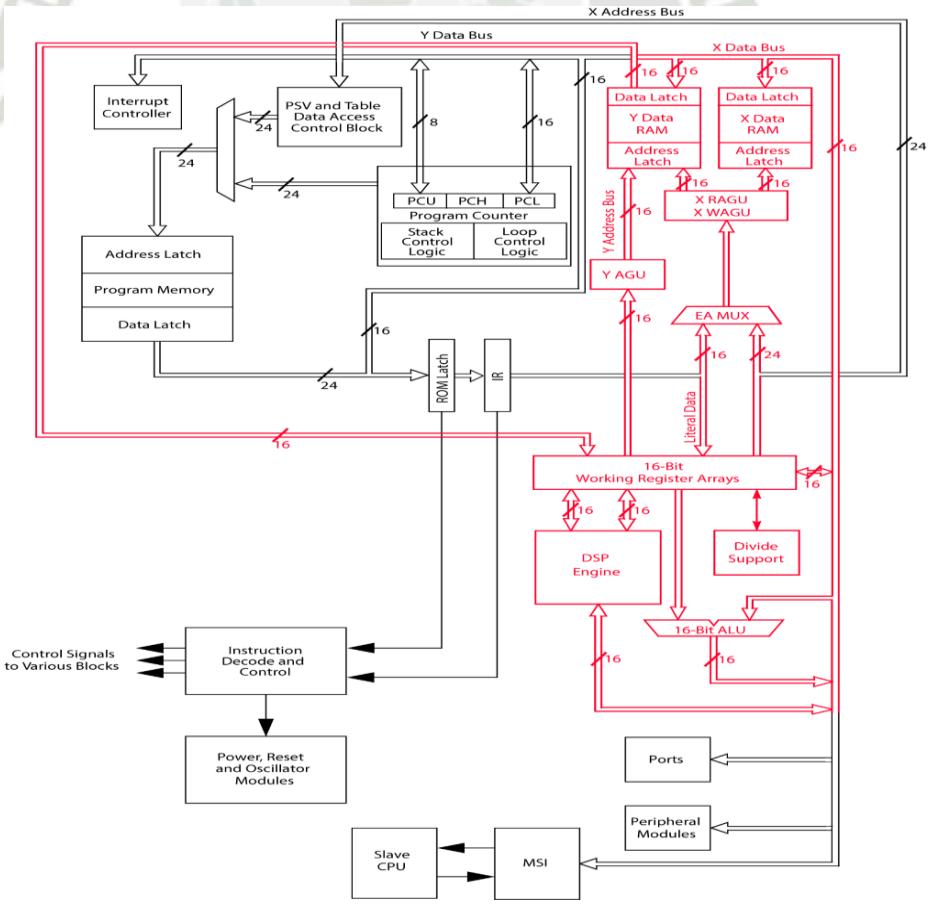
Como se ilustra en la **Figura 87**, la secuencia de eventos necesaria para procesar señales bioeléctricas del movimiento ocular y transformarlas en movimientos del cursor de un mouse no es compleja, y podría ejecutarse con un microcontrolador de menor capacidad. Sin embargo, esto conllevaría la desventaja de obtener un rendimiento significativamente inferior. Dado que la captura de datos se realizará a una frecuencia de 5 kHz, el microcontrolador dispone de un máximo de 200 µs para completar el filtrado de tres canales con filtros de orden 250 y procesar cada uno de los resultados para interpretar la señal y convertirla en una acción correspondiente para el movimiento del mouse. Cabe destacar que, como se demostrará posteriormente, el dsPIC es capaz de completar todas estas tareas con suficiente margen de tiempo, garantizando un procesamiento eficiente y una respuesta rápida del sistema.

2.1.1. Motor DSP

Los microcontroladores optimizados para el procesamiento de señales, dsPIC, cuentan con una arquitectura de 16 bits y están diseñados para aplicaciones que necesiten tanto el procesamiento típico de un microcontrolador como capacidades avanzadas de procesamiento digital de señales (DSP). Todos los dsPIC de microchip cuentan con un motor DSP embebido y hardware dedicado para realizar operaciones aritméticas, como memoria dedicada para las variables a ser operadas, buses exclusivos que conectan estas etapas y varios registros de trabajo, permitiendo ejecutar operaciones matemáticas complejas y algoritmos de procesamiento de señales de manera eficiente y rápida, lo que les hace ideales para una amplia gama de aplicaciones de control y procesamiento de señales.

Figura 88

Diagrama de bloques del bloque del CPU del dsPIC33CH series.

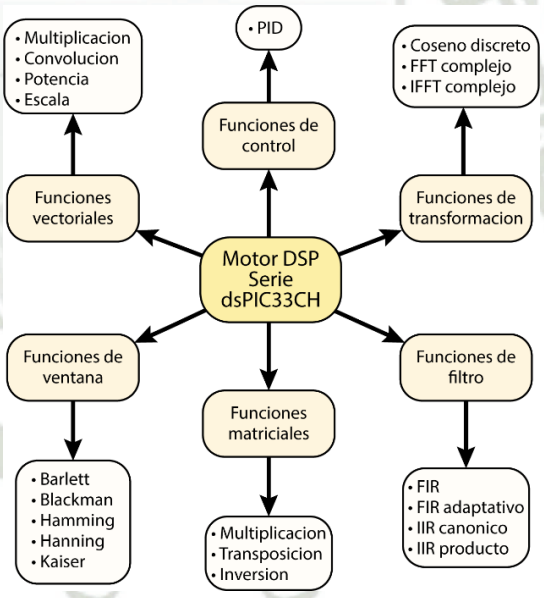


Nota: La figura muestra la estructura interna del módulo dedicado al procesamiento de señales embebido en el dsPIC33CH128MP506, se puede observar que los registros de la ALU son de 16 bits, además de tener un motor DSP dedicado. Fuente: Modificado de Hoja de datos dsPIC33CH series – Microchip.

La ventaja principal de utilizar hardware dedicado para resolver operaciones aritméticas sobre la resolución de las mismas mediante software, es la menor cantidad de ciclos de operación para obtener el resultado, permitiendo realizar aplicaciones en tiempo real. Este aspecto se comprueba en el canal de YouTube con nombre Fun Factory, en la serie de videos dedicado al análisis del comportamiento y uso de otro modelo de dsPIC, pero con el mismo motor DSP (FunFactory, 2023).

Figura 89

Funciones implementadas a nivel de hardware en el dsPIC33CH128MP506



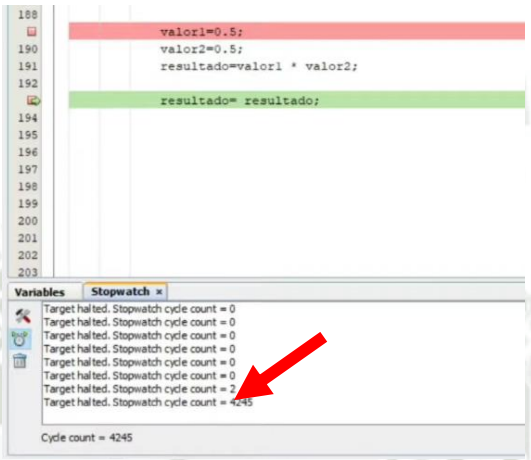
Nota: La figura muestra un diagrama resumen donde se muestra todas las funciones integradas por hardware disponibles en el dsPIC33CH series gracias al motor DSP dedicado para el procesamiento de señales en tiempo real. Fuente propia.

En este video, usando el programa MPLAB X IDE, se simula la ejecución en un dsPIC. la multiplicación de dos números decimales de coma flotante obteniendo que, para realizar esta operación, aparentemente, sencilla, el dsPIC necesita en total 4245 ciclos de instrucción. Para aplicaciones que no necesitan la operar en tiempo real, o donde no se necesiten realizar múltiples operaciones para obtener un único resultado, este tiempo de

ejecución no representa problema, sin embargo, la aplicación que estamos implementado en el prototipo si necesita realizar múltiples operaciones y debe poder hacerlo en tiempo real.

Figura 90

Tiempo de ejecución de una multiplicación de dos números decimales con un dsPIC sin usar hardware dedicado.



Nota: La figura muestra la simulación del tiempo de ejecución de una multiplicación de 2 números decimales, según el software MPLab X, esta multiplicación le tomaría al dsPIC en total 4245 ciclos de instrucción, tomando en cuenta que para realizar un filtro FIR se debe realizar cientos de estas multiplicaciones, se puede inferir que realizar un filtro FIR en base de operaciones de software sería demasiado lento e ineficiente. Fuente: (FunFactory, 2023).

El caso que se presenta en la **Figura 90** es únicamente para una operación de multiplicación, teniendo en cuenta que, si usamos un filtro FIR de orden 100, en total utilizaríamos 4245*100 ciclos de instrucción sin contar los que se necesitan para el proceso de sumar todos los resultados para completar la convolución del vector de coeficientes del filtro con el vector de memoria de las entradas de la señal. Como mencionamos antes, esto es inaceptable para una aplicación que se deben ejecutar en tiempo real, por lo que el uso de operaciones aritméticas resueltas mediante algoritmos de software queda completamente descartado.

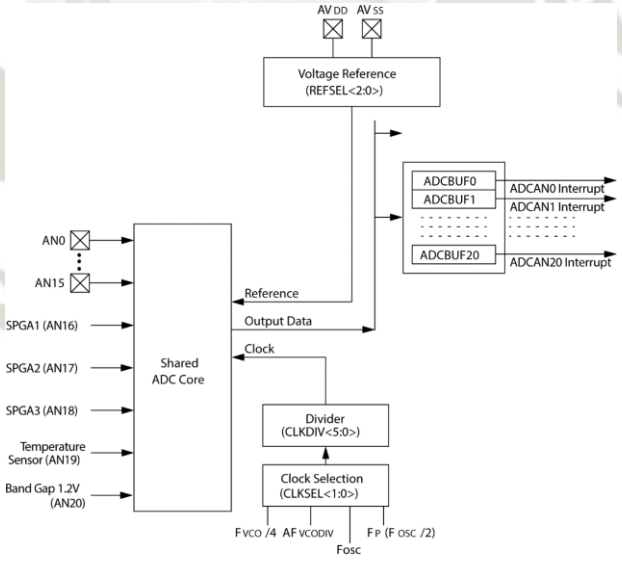
En contraste, según la hoja de datos, DSP tools library, para que el dsPIC complete toda la operación del filtrado (convolución - multiplicaciones y sumas) de una muestra ocupa 53+N(4+M) ciclos de instrucción, siendo N el número de muestras filtradas por operación

(1) y M el número de coeficientes del filtro FIR, por lo que podemos simplificar la ecuación previa a (57 +M) ciclos, teniendo en cuenta un filtro FIR de orden 100, para completar toda la operación de convolución, solo se necesitaría 157 ciclos de instrucción, pudiendo considerarse con esto que el dsPIC efectivamente puede trabajar en tiempo real.

2.1.2. ADC Dedicado a DSP

Una de las características importantes el módulo ADC del dsPIC, es que una vez inicializado, siempre está realizando la conversión A/D, por lo que el tiempo de muestro del filtro FIR no se verá afectado por las demoras del software, esto permite calcular un tiempo de muestreo más exacto, con lo que se puede utilizar este dato para los cálculos en la transformada Z, y una conversión mucho más rápida.

Figura 91
Diagrama de bloques del Módulo ADC



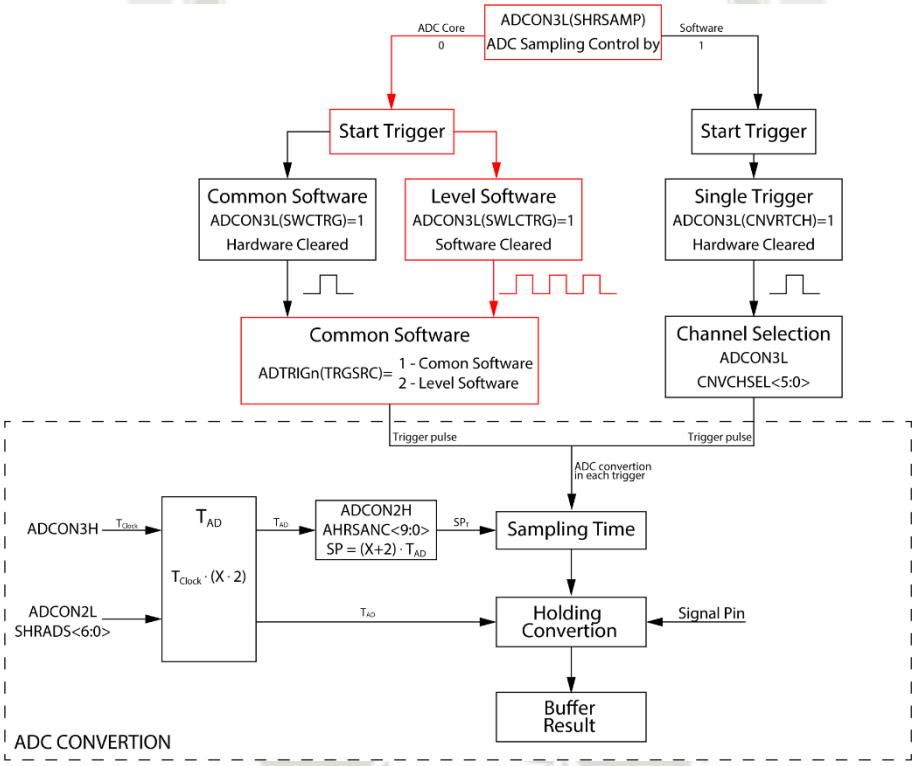
Nota: La figura muestra la estructura interna del módulo ADC del dsPIC33CH128MP506, este módulo funciona de manera independiente, una vez iniciada la conversión en modo continuo, los datos de la conversión se almacenan en los registros ADCBUFXX, hay un buffer dedicado para cada entrada analógica. Fuente: Hoja de datos dsPIC33CH series - Microchip.

El módulo ADC puede operar en dos formas, en modo continuo y en modo “disparo”. En modo “disparo”, se envía un “disparo” mediante Software, esto permite que se realice

una única conversión de todos los canales activados. En el modo continuo, el módulo ADC genera sus propios "disparos" independientemente del software, esto permite que el ADC siempre este convirtiendo sin necesidad de cargarle tiempos de espera al programa. Utilizaremos esto para mejorar el desempeño del prototipo.

Figura 92

Diagrama de flujo de configuración del ADC



Nota: La figura muestra un esquema de los modos de configuración de los registros del módulo ADC del dsPIC33CH128, se debe iniciar el inicio de la conversión mandando un Trigger, de pendiendo de la configuración de la limpieza de este Trigger, el ADC funcionara en modo continuo o en modo de una sola lectura. Fuente propia.

Como se indicó en el apartado 3.4.4 del capítulo anterior, la etapa de filtrado requiere que se cargue el valor de la nueva muestra a una frecuencia constante, para la cual el filtro fue diseñado. En este contexto, la frecuencia controlada por el Timer 1 se convierte en un parámetro crítico para el funcionamiento del filtro FIR. El ADC debe asegurar que la señal digitalizada de los canales esté disponible cuando ocurra la interrupción del Timer 1 (5 kHz). Por lo tanto, la velocidad de conversión del ADC debe ser superior a la frecuencia del

Timer 1. Como referencia, la frecuencia de conversión predeterminada en la placa de desarrollo Arduino Uno es de 125 kHz, pudiendo llegar hasta 1 MHz modificando el valor del registro del “prescaler”. En comparación, el módulo ADC del dsPIC33CH128MP506, que utilizamos con un oscilador de 20 MHz, configurado a una velocidad "lenta", alcanza los 500 kHz. Esta velocidad es más que suficiente para la aplicación que estamos desarrollando, por lo que la configuración del módulo ADC, se limitara a conseguir a funciones, sin importar el tiempo de conversión.

A. Configuración del Módulo ADC

i. Configuración Inicial

Tabla 36
Configuración inicial del módulo ADC.

Función ADC_Init()	Descripción
ANSELAbits.ANSELA0=1; TRISAbits.TRISA0=1; ANSELAbits.ANSELA1=1; TRISAbits.TRISA1=1; ANSELAbits.ANSELA2=1; TRISAbits.TRISA2=1;	Se configura los pines que serán las entradas analógicas, activando los canales analógicos correspondientes y estableciendo los pines GPIO como entradas.

//Seleccion de trigger
ADTRIG0Lbits.TRGSRC0=1;//AN0 Common Software Trigger
ADTRIG0Lbits.TRGSRC1=1;//AN1 Common Software Trigger
// ADTRIG0Hbits.TRGSRC2=1;//AN2 Common Software Trigger
	El segundo paso es elegir la forma en la que se iniciara la conversión, como referencia se tiene la **Figura *92***. En este caso, se configura los canales analógicos para que sean activados por hardware.

//Formato de dato
ADCON1Hbits.FORM=0; //Format Integer
 	
ADCON1Hbits.SHRRES=3;//Resolution 12 bits
	Se selecciona el formato del resultado de la conversión, en este caso como Entero y con una resolución de 12 bits.

//Voltaje de Referencia
ADCON3Lbits.REFSEL=0;
	Se selecciona el voltaje de referencia, los cuales se tomarán en los pines de voltaje de referencia.

Nota: Ignorar ausencia de tildes en la tabla, el IDE de programación no reconoce símbolos con tildes.

ii. Tiempo de muestreo ADC

La selección del reloj es importante para determinar el tiempo que el módulo ADC empleara para realizar las conversiones de todos los canales activados, sin embargo, como se detalló antes, este tiempo de muestreo no es fundamental para el comportamiento del filtro, solo debe poder garantizar que el tiempo de muestreo del ADC es superior al tiempo de muestreo del filtro, debido a que

Tabla 37
Configuración del tiempo de muestreo del módulo ADC.

Función ADC_Init()	Descripción

//Seleccion de reloj
ADCON3Hbits.CLKSEL=0;
ADCON3Hbits.CLKDIV=0;
//Dentro del Módulo ADC
ADCON2Lbits.SHRADCS=1;
//TAD=(1 *2))(Tfosc) -> TAD = 100ns
//Sampling Time
ADCON2Hbits.SHRSAMC=8;
//SP=(8 +2)*TAD -> SP=10*(100ns))
//Conversion Time = 8*Tcoresrc + (Bits_resolution + 2.5)*TADcore
//Conv Time = 8*50ns + (12 + 2.5)*100ns = 1.85us
//Prendemos el ADC
ADCON1Lbits.ADON=1;
ADCON5Lbits.SHRPWR=1;
while(!ADCON5Lbits.SHRRDY){}
ADCON3Hbits.SHREN=1;
	Se selecciona como reloj del ADC, la Frecuencia del Oscilador (FOSC)
	Para calcular el tiempo de conversión del ADC, se debe calcular el periodo del ADC (TAD), el periodo de muestreo (SP), las ecuaciones que condicionan estos parámetros están definidos en las hojas de datos del dsPIC.
	Se configura el ADC con un tiempo de conversión de 1.8 µs, equivalente a 540.5 kHz.
	Al finalizar la configuración, podemos prender y habilitar el módulo ADC. A partir de este punto, las conversiones se realizarán constan mente en todos los canales que se hayan habilitado.

Nota: Ignorar ausencia de tildes en la tabla, el IDE de programación no reconoce símbolos con tildes.

iii. Lectura del resultado

Tabla 38
Lectura del resultado del módulo ADC.

Función AnalogRead_All ()	Descripción
ADCON3Lbits.SWCTRG = 1;	Para leer el resultado, solo se debe copiar el valor del resultado de la conversión almacenado en el buffer dedicado al canal de interés.
while(!(ADSTATLbits.AN14RDY)) { }	En caso que la consulta se realice en plena conversión, se espera hasta que este termine.
*chan_0 = ADCBUF13; *chan_1 = ADCBUF14;	Por último, se copia el valor del buffer en la variable con la que estamos trabajando.

Nota: Ignorar ausencia de tildes en la tabla, el IDE de programación no reconoce símbolos con tildes.

B. Interrupción por Timer 1

La selección del Timer 1 como árbitro para la toma de nuevas muestras del filtro FIR se debe principalmente a que, en todos los microcontroladores de Microchip, este módulo genera la interrupción de mayor prioridad. Esto garantiza que, independientemente de lo que esté ocurriendo dentro del dsPIC, la carga de un nuevo dato hacia el filtro FIR este asegurado y, además, siempre se realice en un tiempo constante.

La configuración del Timer 1 es crucial, ya que debe operar a la misma frecuencia para la cual se diseñó el filtro FIR. Si esta frecuencia varía, se experimentará un desplazamiento en la frecuencia de corte del filtro FIR.

i. Configuración Inicial del Módulo del Timer 1

Tabla 39
Configuración Inicial del Módulo del Timer 1

Función Timer1_Init()	Descripción
void Timer1_Init(void){ T1CONbits.TON = 0; T1CONbits.TCS = 0; T1CONbits.TGATE=0; T1CONbits.TCKPS = 0x1;	La primera parte se encarga de seleccionar el reloj que alimentara al Timer 1, siendo TCS=0, se selección la frecuencia de instrucción (F_{CY}) que es de 10 MHz.

Función Timer1_Init()	Descripción
	Adicionalmente, se selecciona el valor del pre escalador, siendo TCKPS=1, el pre escalador se ajusta en 8.
PR1 = 249;	Con PR1 se ajusta el registro del periodo, este sigue
TMR1 = 0;	la siguiente ecuación: $$PR1 = \frac{F_{CY} * t}{Pre} - 1$$ Si deseamos una interrupción de 5kHz, el valor a cargar en este registro es 249.
IEC0bits.T1IE = 1;	Con el registro T1IE se habilita la interrupción del
IFS0bits.T1IF = 0;	Timer 1.
IPC0bits.T1IP = 7;	Con el registro T1IF se desactiva el flag del Timer 1 para asegurar un inicio en cero.
T1CONbits.TON = 1;	Con el registro T1IP se asigna una prioridad a la
}	interrupción, siendo el valor de 7 como la prioridad máxima.
	El registro TON inicia el módulo del Timer 1.

Nota: Ignorar ausencia de tildes en la tabla, el IDE de programación no reconoce símbolos con tildes.

ii. Función de Lectura ADC por Timer 1

Tabla 40

Lectura del resultado del ADC mediante interrupción del Timer 1.

Función _T1Interrupt()	Descripción
void __attribute__((interrupt,no_auto_psv)) _T1Interrupt(void){	"__attribute__" es una especificación de atributo utilizada por el compilador para indicar propiedades especiales de la función, en este caso, una interrupción. El nombre de la función _T1Interrupt() está reservada por el compilador, y le asocia todas las configuraciones previas realizadas al módulo del Timer 1.
AnalogRead_All(&RecepADC);	Únicamente cuando ocurra la interrupción del Timer 1, se llamará a la función que lee los valores que tiene el módulo ADC en ese instante.
TMR1_Done=true;	Se activa un "flag" para uso personal, el cual
IFS0bits.T1IF = 0;	dará pase libre para que se ejecute el código
};	general.

Función _T1Interrupt()	Descripción
	Se desactiva el "flag" del sistema que está asociado a la activación del Timer 1, reiniciando las condiciones para una nueva interrupción.

Nota: Ignorar ausencia de tildes en la tabla, el IDE de programación no reconoce símbolos con tildes.

iii. Ejecución de código por Timer 1

Tabla 41

Ejecución de código principal por activación de flag de la Interrupción del Timer 1

Función main()	Descripción
void main(void) { // CONFIGURACION INICIAL//	La función principal "main", contiene y enlaza todo el comportamiento del dsPIC.
while(1) {	

while(TMR1_Done){
// CODIGO GENERAL//
 	
TMR1_Done=false;
}
	El código del filtro FIR se ejecutará únicamente cuando ocurra la interrupción del Timer 1, como línea final, se desactiva el "flag" de uso personal, garantizando que el código dentro de este bucle se ejecute una sola vez en cada interrupción.

} }	

Nota: Ignorar ausencia de tildes en la tabla, el IDE de programación no reconoce símbolos con tildes.

2.2. Filtro FIR Optimizado por Hardware

La mayoría de los microcontroladores están equipados para realizar las operaciones matemáticas esenciales requeridas para el funcionamiento de filtros de respuesta finita al impulso (FIR). Sin embargo, se observa una distinción crucial en la implementación de dichas operaciones. Mientras que las operaciones básicas de suma y resta se efectúan directamente a través de la unidad de lógica aritmética (ALU) del microcontrolador, operaciones más complejas como la multiplicación y la división suelen ser manejadas mediante técnicas de software que emplean múltiples ciclos de instrucción. Este método,

aunque funcional, resulta poco óptimo para aplicaciones que demandan alta velocidad de procesamiento en tiempo real.

En contraste, el microcontrolador dsPIC33CH128MP506 de Microchip se destaca por tener capacidades de multiplicación y división integradas directamente en su hardware, además de otras funciones especializadas para el procesamiento digital de señales. Esto permite que los resultados de dichas operaciones se obtengan en uno o dos ciclos de operación, ofreciendo una ventaja significativa en términos de eficiencia y velocidad, elementos críticos en aplicaciones de tiempo real. La inclusión de estas capacidades en el hardware facilita un desempeño superior en tareas de procesamiento intensivo, como es el caso del filtrado FIR, comparado con microcontroladores que dependen exclusivamente de soluciones de software para estas funciones. (Microchip, DS51456B, 2004).

2.2.1. Formato de Coma Flotante y Coma Fija

A. Coma flotante (Float32)

En el ámbito de la computación y las ciencias de la información, el estándar IEEE 754-2008 y complementado por el estándar ISO/IEC/IEEE 60559:2011, define las especificaciones para la representación numérica en formato binario de 32 bits, comúnmente conocido como float32 o binary32. Este formato permite la codificación de números extremadamente grandes, aproximadamente hasta 3.4028235×10^{38}, y extremadamente pequeños, alrededor de 1.17549435×10^{-38}. Adicionalmente, el formato float32 facilita la representación de valores especiales tales como infinito positivo y negativo, y NaN (Not a Number), este último utilizado para denotar los resultados de operaciones matemáticas indefinidas.

La estructura de un número en este formato incluye 32 bits divididos en tres secciones: 1 bit se asigna al signo del número, 8 bits destinados al exponente, y los 23 bits

restantes a la mantisa. Este diseño permite una amplia gama de expresión numérica, crucial para aplicaciones que requieren alta precisión y rango extenso en cálculos científicos y técnicos.

$$\begin{matrix} 1\ bit & 8\ bits & 23\ bits \\ s & e & f \end{matrix}$$
$$a = (-1)^s \cdot 2^{e-127} \cdot 1.f$$

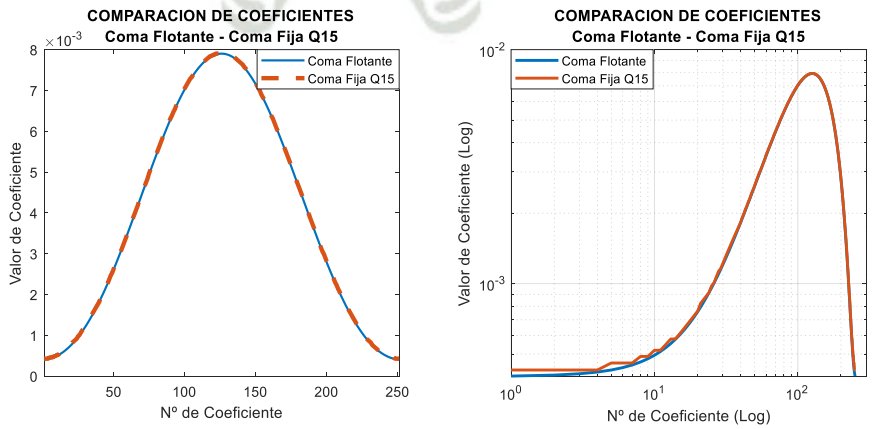
Debido a que en el desarrollo del filtro FIR, se utilizara únicamente el formato de punto fijo Q15, ya que se está elije velocidad sobre exactitud; aprovechando el Hardware del motor DSP; solo se detallara la estructura básica, hasta este punto, del formato de Coma Flotante en este apartado.

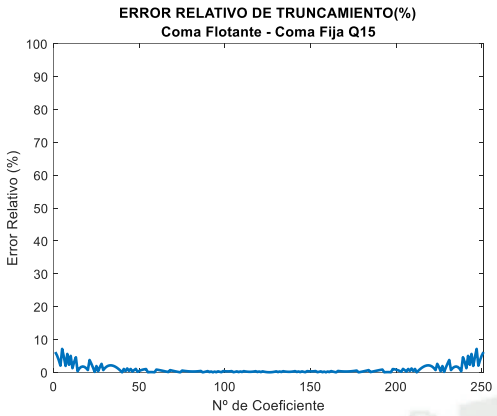
B. Coma fija (fractional Q15)

A diferencia del formato de punto flotante de 32 bits, el tipo de variable en coma fija denominado fractional Q15 de 16 bits no está regulado por un estándar internacional de IEEE. Sin embargo, este formato se ha establecido como un estándar de facto ampliamente aceptado en la industria del procesamiento digital de señales. Se emplea predominantemente en aplicaciones embebidas que requieren un control exhaustivo sobre el rendimiento del sistema y la eficiencia en el uso del hardware.

Figura 93

Comparación de coeficientes de filtro en coma flotante y coma fija.





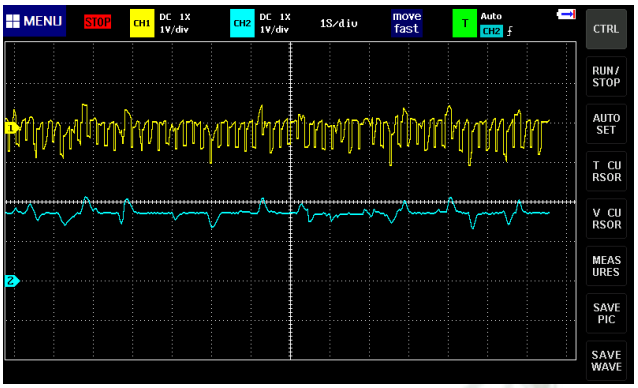
Nota: La figura muestra una comparación de los coeficientes de un filtro FIR de orden 250, Fs=5kHz, Fc=10Hz - en coma flotante y coma fija, en una escala lineal ambos se ven iguales, pero en una escala logarítmica, se puede observar una desviación de ambos valores esto se debe a que el formato de coma fija no tiene tanta resolución como en como flotante. Fuente propia.

Microchip implementa este tipo de variable para facilitar operaciones con números decimales optimizadas a nivel de hardware. Por lo tanto, es necesario convertir los coeficientes al tipo de variable fractional Q15. No obstante, esta representación posee la limitación de abarcar un rango más reducido de números, debido a que la variable fractional Q15 asigna un bit para el signo y los 15 bits restantes para la fracción decimal del número. El valor positivo máximo representable es aproximadamente 1 ($1 - 2^{-15}$), mientras que el mínimo negativo es -1. El valor más cercano a cero que puede representarse es 2^{-15} (aproximadamente 3.051758×10^{-5}).

Esta limitación implica que los coeficientes menores a este valor deben ser truncados, lo cual podría parecer una desventaja significativa, esto se puede apreciar en la Figura 93. Sin embargo, en la práctica, este truncamiento generalmente no incide de manera considerable en los resultados del proceso de filtrado. En la Figura 94 se muestra el filtrado FIR con coeficientes truncados fractional Q15, se observa que, a pesar del truncamiento aplicado a los coeficientes, el resultado es muy bueno, por lo que esto no afecta a nuestro objetivo.

Figura 94.

Filtrado de la señal EOG con movimientos oculares.



Nota: La figura muestra una señal EOG, en el canal 1 (amarillo) se observa la señal EOG luego de ser amplificada, esta tiene bastante ruido eléctrico, en el canal 2 (cian) se observa la señal EOG amplificada luego de pasar por el filtro FIR, se observa una señal mucho más limpia a pesar que los coeficientes del filtro FIR implementado en el dsPIC128MP506 fueron escritos en formato de coma fija. Fuente propia.

Para convertir un numero decimal a fractional Q15, se debe seguir los siguientes pasos:

1) Escalar el decimal: el número debe decimal que esta entre 1 a -1 debe multiplicarse por 2^{15}.
2) Redondeo: El resultado obtenido será un numero entero con parte decimal, se debe redondear al entero más cercano.
3) Representación: En este punto el resultado ya está en format Q15, este paso es opcional dependiendo como se guardará el valor, se convierte al sistema hexadecimal, binario o se deja en decimal.

Tabla 42

Pasos para convertir número a coma fija Q15.

Paso	Procedimiento
Escalamiento	$0.2065 \times 2^{15} = 6766.592$
Redondeo	$6766.592 \rightarrow 6767$
Representación	$6767_{10} \rightarrow 0x1A6F_{16}$

Para convertir un valor en formato de coma fija Q15 a su equivalente decimal, se requiere realizar el proceso inverso al escalado inicial. Este proceso consiste en multiplicar

el número en coma fija Q15 por 2^{-15}. Este proceso será empleado para evaluar y comparar el error inherente que surge al redondear o truncar un número. Tomaremos como caso de análisis el ejemplo presentado en la **Tabla 42**, donde el valor decimal original es 0.2065.

Tabla 43
Error relativo por el redonde del número escalado, con referencia a 0.2065.

Sist. Decimal	Decimal	Error Relativo	Truncamiento a 4 decimales
6768_{10}	0,2065430	2,08%	0.2065
6767_{10}	0,2065125	0,60%	0.2065
6766_{10}	0,2064819	0,87%	0.2065
6765_{10}	0,2064514	2,35%	0.2065

En el análisis presentado en la **Tabla 43**, se evidencia el error generado por el proceso de redondeo asociado con la conversión de un número decimal a su equivalente en el formato de coma fija Q15. El valor de referencia utilizado en este estudio es 0.2065000..., y se observa que las representaciones numéricas obtenidas mediante este método no coinciden exactamente con el valor original. Esto es atribuible a las limitaciones inherentes de la representación en coma fija, donde la precisión está restringida por el número de bits disponibles para la fracción del número.

Aunque es posible mitigar errores aplicando técnicas de redondeo o truncamiento al resultado final, ajustando así el número representado en formato de punto fijo para que coincida con el número de decimales del valor original, este método enfrenta limitaciones cuando el número es menor que el valor más cercano a cero permitido por este formato. En particular, en el contexto de los filtros FIR, a medida que se incrementa el orden del filtro, los coeficientes tienden a disminuir en magnitud. Esto conlleva a un incremento en el error de truncamiento en los extremos de los coeficientes, esto se puede observar en la Figura 93.

2.2.2. Filtrado por Hardware (DSP)

Las funcionalidades de procesamiento digital de señales que están integradas por hardware en el microcontrolador dsPIC33CH128MP506 no están incorporadas de manera

predeterminada en la biblioteca estándar del lenguaje C del compilador XC16 del entorno MPLAB X IDE. Para acceder y utilizar estas capacidades especializadas, es necesario emplear las funciones embebidas (__builtin()) provistas por el compilador XC16. Estas funciones permiten a los desarrolladores aprovechar directamente las características de hardware del microcontrolador, optimizando así el rendimiento de las aplicaciones.

Además, para asegurar una gestión eficiente y una ejecución rápida, los coeficientes de los filtros y otras constantes críticas deben ser almacenados directamente utilizando código ensamblador. Este enfoque facilita un control más preciso sobre la ubicación y el manejo de la memoria, crucial para el rendimiento de sistemas embebidos que realizan tareas complejas de procesamiento de señales en tiempo real.

A. Asignación de Coeficientes

En la Sección 3.4 del capítulo anterior, se examinó el método para calcular los coeficientes de un filtro FIR. En el presente apartado, se discutirá exclusivamente el procedimiento para cargar dichos coeficientes en la memoria del dsPIC. Es posible declarar los coeficientes directamente en el código del programa principal para su almacenamiento en la memoria RAM. Sin embargo, esto consume una cantidad significativa de espacio en dicha memoria de forma innecesaria. Adicionalmente, el motor DSP no accede a la memoria RAM para ejecutar operaciones que están optimizadas por hardware, por consiguiente, si los coeficientes se almacenan en la memoria de programa, no solo se utilizará ineficientemente el espacio de la memoria RAM, sino que también nos veremos obligados a utilizar el filtrado por software, teniendo como resulta un procesamiento más lento y de menor calidad, siendo esto un problema ya que este prototipo maneja múltiples canales simultáneamente.

La solución a esto, consiste en declarar los coeficientes en las memorias específicamente diseñadas para interactuar con el motor DSP, denominadas memoria X y memoria Y. La memoria X se utiliza para almacenar las variables que son necesarias para el funcionamiento del motor DSP, mientras que la memoria Y se emplea para reservar espacio destinado a los cálculos realizados por el mismo motor. Como se ha mencionado anteriormente, las funcionalidades del motor DSP no se integran de manera directa en el compilador XC16, por lo tanto, la declaración de los coeficientes y la estructura del filtro FIR deben realizase en un lenguaje de bajo nivel, para esto emplearemos un archivo escrito en lenguaje ensamblador, adicionalmente, se debe tener en cuenta que los coeficientes deben ser declaradas en formato coma fija Q15, esto se mostró en el apartado 2.2.1 de este capítulo.

Tabla 44

Declaración de coeficientes para el filtro FIR optimizado por hardware.

Macro para declaraciones futuras del orden del filtro	
.equ FIR_Orden, 84	Se crea una macro FIR_Orden equivalente a 84 (Orden del filtro). Esta manera de declarar el orden del filtro, nos permite modificar todos los campos que hagan uso de este parámetro modificando una sola línea de código.
Declaración de los coeficientes del filtro FIR	
.section .xdata, data, xmemory .align 256	Se indica que se usara la memoria X, alineando todos los datos con un valor de 256.
FIR_Coeffs: .hword 0XA3E5, 0xA2F1, … hword … … … … …	Se declaran los 85 coeficientes (Orden +1) del filtro FIR. La palabra reservada “hword.” guarda los valores en la memoria de forma consecutiva. Coeficientes del filtro la Figura 95 en ANEXO F.
Reserva de memoria para los resultados del filtrado	
.section .ydata, data, ymemory .align 256	Se indica que se usara la memoria Y, alineando todos los datos con un valor de 256.

FIR_Delay: .space FIR_Orden*2	Se reserva un espacio de memoria de tamaño del doble del orden del filtro, aquí guardara los resultados históricos del filtro.
Creación de estructura del FIR	
.section .data .global _FIR_Struct	Se define una porción de memoria con el nombre _FIR_Struct, y se determina que será accesible de manera global. El uso del guion bajo "_" como prefijo del nombre de la estructura, es una convencionalidad que indica que la variable es global, de esta forma el compilador XC16 puede reconocer esta estructura si es que se le referencia con el lenguaje C.
_FIR1_Struct: .hword FIR_Orden .hword FIR1_Coeffs .hword FIR1_Coeffs+FIR_Orden*2-1 .hword 0xff00 .hword FIR1_Delay .hword FIR1_Delay+FIR_Orden*2-1 .hword FIR1_Delay	Se crea y define los datos de la estructura de que se usará para el filtro FIR, el cual será referenciado desde el compilador XC16 para llamar y ejecutar otras funciones del ensamblador del motor DSP.

El procedimiento para cargar los coeficientes de un filtro FIR debe realizarse de manera independiente para cada filtro que se esté implementado. Es importante que los nombres de cada parte relacionado a la carga de coeficientes sean claramente distinguibles dentro del código, para evitar ambigüedades y prevenir errores durante la compilación.

Para este prototipo en particular, se utilizarán tres canales con filtros independientes cada uno. A pesar que estos filtros son exactamente iguales, no hay manera de multiplexar un mismo filtro para los tres canales, teniendo esto en cuenta, se determinad tres filtros, _FIR1, _FIR2 y _FIR3.

B. Estructura del filtrado por Hardware

Si bien los parámetros esenciales para el funcionamiento del filtrado por hardware requieren ser configurados en un lenguaje de bajo nivel, como el Ensamblador, las demás funciones del programa que gestionan periféricos adicionales, como el ADC, el Timer 1, etc., se programan utilizando el lenguaje C mediante el compilador XC16. Los dsPIC

facilitan la integración de ambos lenguajes en distintos archivos simultáneamente. Por esta razón, nos centraremos principalmente en el manejo del filtro utilizando el lenguaje C, desde el cual se realizarán llamadas a los archivos en lenguaje Ensamblador para ejecutar el filtrado optimizado por hardware. Para ello, es imprescindible desarrollar en XC16 ciertas funciones que interactúen con las implementadas en Ensamblador.

i. FIRStruct

Describe la estructura del filtro para cualquiera de los filtros FIR, esta estructura tiene los mismos campos que la estructura creada en el archivo Ensamblador.

Tabla 45
Declaración de la estructura que se usara para el filtro FIR.

FIR Struct	Argumentos
typedef struct {	***numCoeffs:*** Número de coeficientes en el filtro.
int numCoeffs;	***coeffsBase:*** Dirección base para los coeficientes del filtro.
fractional* coeffsBase;	***coeffsEnd:*** Dirección final para los coeficientes del filtro
fractional* coeffsEnd;	***coeffsPage:*** Número de página del búfer de coeficientes
int coeffsPage;	***delayBase:*** Dirección base para el búfer de retardo
fractional* delayBase;	***delayEnd:*** Dirección final para el búfer de retardo
fractional* delayEnd;	***delay:*** Valor actual del puntero de retardo.
fractional* delay;	
} FIRStruct;	

Nota: Ignorar ausencia de tildes en la tabla, el IDE de programación no reconoce símbolos con tildes.

ii. FIR

La librera dsp.h declara una función que devuelve un puntero hacia una variable tipo fractional, esa función denominada FIR() esta defina en otro archivo denominado FIR.asm, el cual se encarga de hacer funcionar el filtro FIR optimizado por Hardware.

Tabla 46
Descripción de los argumentos de la función FIR.

Función FIR()	Argumentos
extern fractional* FIR (	***numSamps:*** Numero de muestras de entrada a filtrar.
int numSamps,	***dstSamps:*** Puntero a las muestras de destino.

fractional* `dstSamps`, fractional* `srcSamps`, FIRStruct* `filter`);	***srcSamps:*** Puntero a las muestras de origen. ***filter:*** Puntero a la estructura del filtro FIRStruct.

Nota: Ignorar ausencia de tildes en la tabla, el IDE de programación no reconoce símbolos con tildes.

Esta función desempeña un papel importante, dado que transfiere los datos y hace que se ejecute el código ensamblador responsable del filtrado por hardware. El algoritmo implementado por el motor DSP del dsPIC para esta operación es denominado un Filtro Fraccional, ya que esta procesa una secuencia de datos de entrada representada como un vector de tipo fraccional (Microchip, DS51456B, 2004). La realización de este proceso implica la resolución de la siguiente ecuación:

$$y[n] = \sum_{m=0}^{M-1} (b[m])(x[n-m])$$

Esta ecuación en diferencias que se proporciona en el manual de las librerías del módulo DSP proporcionado por Microchip, es el mismo que se estudió en el apartado 3.4.3 del capítulo anterior, y su representación gráfica es exactamente igual al mostrado en la **Figura 60**,

iii. FIRDelayInit

Inicializa en cero los valores de retardo en una estructura de filtro FIRStruct, esto permite que la estructura esta lista para poder inicializar el filtrado por hardware.

Tabla 47
Argumento de la función FIRDelayInit.

Función FIRDelayInit()	**Argumentos**
extern void FIRDelayInit (FIRStruct* `filter`);	**filter**: Puntero que devuelve la dirección de la estructura del filtro que se quiere inicializar los valores iniciales.

Nota: Ignorar ausencia de tildes en la tabla, el IDE de programación no reconoce símbolos con tildes.

C. Ejecución del filtro por hardware

Teniendo claro la configuración inicial, los parámetros y funciones asociadas a la implementación del filtro optimizado por hardware,

Tabla 48

Estructura del código necesario para ejecutar el filtrado optimizado por hardware.

Referencia a librerías externas	
#include <dsp.h>	La librería dsp.h contine las definiciones y funciones necesarias para realizar las operaciones de procesamiento digital de señales en general.
Declaraciones Globales	
extern FIRStruct FIR1_Struct; #define Block 1 fractional FilterOut[Block]; uint16_t Salida;	Se referencia a la estructura definida en el código en ensamblador "_FIR1_Struct". "FilterOut" almacenara el resultado del filtro FIR. "Salida" almacenara los datos que serán procesados para el propósito del prototipo.
Declaración de variables iniciales dentro del Main	

char *apuntador = NULL;
uint8_t valor_h,valor_l;
apuntador = &FilterOut;
FIRDelayInit(&FIR1_Struct);
	Las variables "apuntador, valor_h y valor_l" se usan para acceder los datos que guarda la variable FilterOut.
	La función "FIRDelayInit()" que tiene como argumento la dirección de la estructura del filtro FIR1, inicializa en cero el campo de memoria que almacena los resultados del filtro, es decir, declara las condiciones iniciales del filtro.

Bucle Principal	

FIR(1,&FilterOut,&RecepADC,&FIR1_Struct);
valor_h = apuntador[1];
valor_l = apuntador[0];
Salida=(valor_h<<8)|valor_l;
 	
DAC_Write(Salida);
	Se pasa los argumentos de la función FIR, y esta llama a la función FIR definida en el archivo FIR.asm, el cual realiza la suma de diferencia implementado el filtro FIR.
	Las últimas cuatro líneas del código, envían los datos del resultado, almacenado en la variable FilterOut a

	la variable Salida, con el cual trabajaremos para cumplir los objetivos de nuestro prototipo.

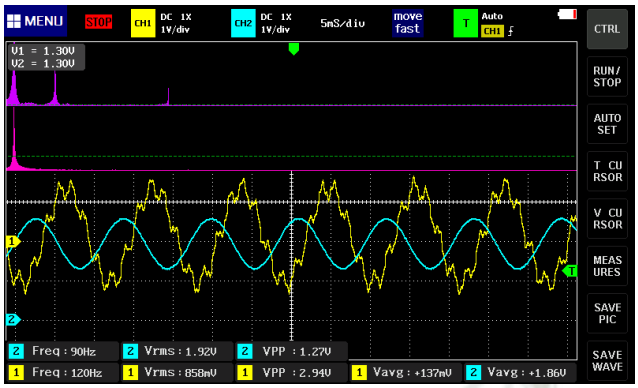
Nota: Ignorar ausencia de tildes en la tabla, el IDE de programación no reconoce símbolos con tildes.

Otra característica importante del dsPIC33CH128MP506 es su integración de un módulo convertidor de digital a analógico (DAC), el cual permite comparar el resultado digital del filtrado digital realizado por el dsPIC con una señal analógica. En la Figura 95 se muestra el resultado de una prueba del filtrado FIR utilizando exclusivamente el dsPIC. En el canal 1 se introduce una señal senoidal con una componente principal de 90 Hz, una secundaria de 600 Hz, y una componente adicional de ruido aproximadamente a 2100 Hz, como se observa en la primera gráfica FFT y en la señal amarilla a lo largo del tiempo.

Dentro del dsPIC se aplica un filtro FIR pasa bajos con una frecuencia de corte de 100 Hz (el valor de los coeficientes se muestra en el ANEXO D), y el resultado digital se convierte en una señal analógica utilizando el módulo DAC del dsPIC, visible en el canal 2. Observando el dominio frecuencial en la segunda gráfica FFT correspondiente a la señal de salida, se distingue que solo se conservaron las frecuencias inferiores a 100 Hz. En el plano temporal, se observa una señal senoidal de 90 Hz, ligeramente atenuada debido a la proximidad de dicha frecuencia a la banda de transición del filtro que se ubica alrededor de la frecuencia de corte. Además, se aprecia que la señal de salida está retrasada con respecto a la señal de entrada, esto es producto de los fenómenos analizados en el apartado 3.4.3 del Capítulo 1.

Figura 95

Filtro FIR pasa bajos a 100Hz



Nota: La figura muestra la captura de señal en un osciloscopio, en el canal amarillo se muestra una señal senoidal 90 Hz con ruido de 600Hz de un generador de funciones, en el canal cian se muestra la señal luego de pasar por un filtro FIR pasa bajos con corte a 100Hz, la señal filtrada se ve limpia en la escala temporal como frecuencial. Fuente propia.

Adicionalmente, se puede observar que la señal de entrada del canal 1, oscila con respecto a 0V y que la señal del canal 2, correspondiente a la salida del dsPIC, tiene un desplazamiento positivo de 2 V, esto es un problema que se analiza y explica en el apartado 3.2 del presente capitulo.

3. PROCESO DE ADQUISICIÓN DE SEÑALES EOG

3.1. Aislación de Ruido

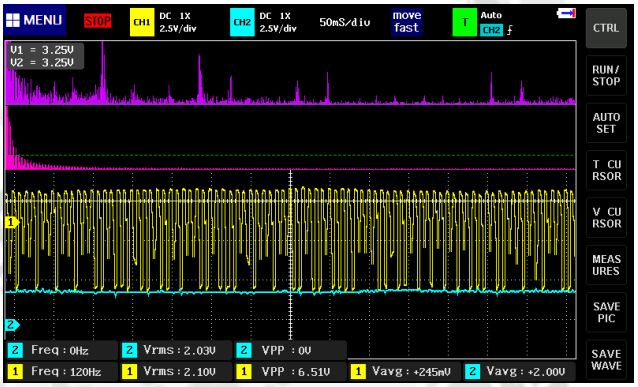
Uno de los objetivos principales en el desarrollo de este prototipo es aprovechar la alimentación suministrada por el bus USB. Si bien esta estrategia ofrece beneficios significativos, también conlleva ciertas desventajas que deben ser cuidadosamente evaluadas. La principal preocupación radica en la posibilidad de que el usuario sufra una descarga eléctrica a causa de una fuente de alimentación defectuosa o un inadecuado aterrizaje del chasis del host. Para mitigar este riesgo, se adoptará la Norma IEC 60601-1, que regula la seguridad de los equipos médicos eléctricos.

Además, el prototipo se enfrenta al problema de exposición a ruido de armónicos, generado por las fuentes de alimentación conmutadas presentes en todos los ordenadores.

Este ruido puede distorsionar significativamente las señales analógicas captadas del cuerpo humano, resultando en mediciones erráticas y complicando la identificación de patrones consistentes para evaluación. Este aspecto es crucial, dado que la integridad de los datos obtenidos es fundamental para la aplicación clínica efectiva del prototipo.

Figura 96

Efecto del ruido eléctrico en la entrada de señales EOG.

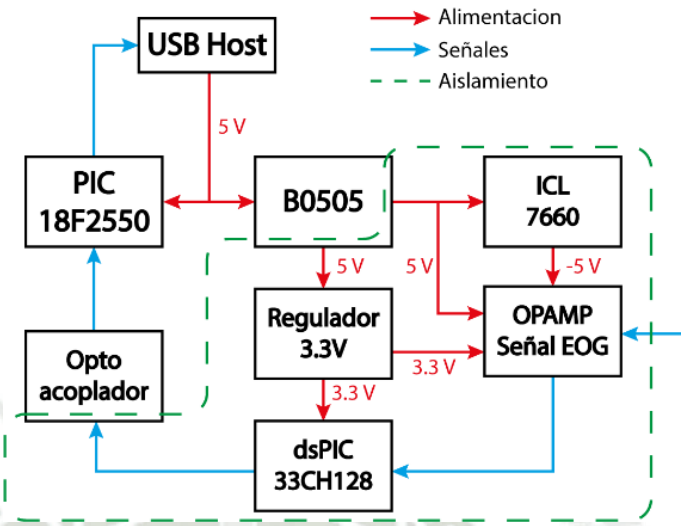


Nota: En la figura se observa la captura de señal con un osciloscopio, en el canal amarillo se muestra la señal de salida del amplificador previo al filtrado , este circuito está expuesto al ruido producto de una fuente swiching conectada a la computadora, en el canal cian se muestra la señal filtrada, se observa que el filtro elimino el ruido, pero aun logra pasar parte del ruido de la red eléctrica. Fuente propia.

Una solución que aborda ambas problemáticas consiste en implementar una fuente de alimentación que aísle el bus USB del host de todas las etapas implicadas en el manejo de señales analógicas. Este enfoque no solo mejora la integridad de la señal, sino que también protege todos los circuitos del ruido eléctrico proveniente del bus USB. Adicionalmente, una característica destacable del prototipo es su bajo consumo eléctrico, lo cual permite la utilización del componente B0505S, el cual trabaja a 1W de potencia máxima, optimizando así el rendimiento energético del sistema.

Figura 97

Diagrama de bloques de las señales de alimentación y comunicación.



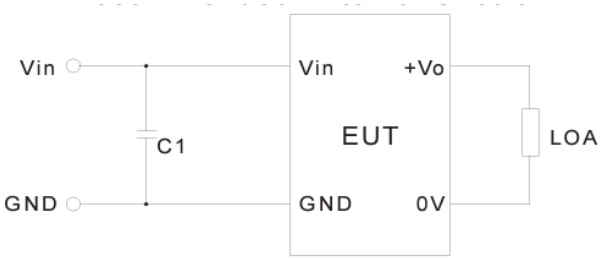
Nota: La figura muestra un diagrama de bloques que se utilizara para aislar galvánicamente el ruido eléctrico de los componentes que se encargan de la lectura y filtrado de las señales analógicas. Fuente propia.

El componente B0505S es un aislador de voltaje altamente eficiente, diseñado para ofrecer una salida estabilizada de 5V con una corriente máxima de 200mA. Este dispositivo es esencial en aplicaciones donde la separación eléctrica y la protección contra interferencias de voltaje son críticas. Además de su función principal de aislamiento, el B0505S contribuye a la mejora de la seguridad y la integridad de los sistemas electrónicos al minimizar los riesgos asociados con sobretensiones y picos de voltaje.

El B0505S se destaca por su compactibilidad y facilidad de integración en una variedad de circuitos electrónicos, incluyendo, pero no limitándose a dispositivos médicos, instrumentación industrial y sistemas de comunicación. Su capacidad para soportar condiciones de operación extremas y su alta fiabilidad lo hacen adecuado para aplicaciones que requieren un aislamiento robusto y una regulación precisa del voltaje.

Figura 98

Diagrama de componentes para el B0505S.



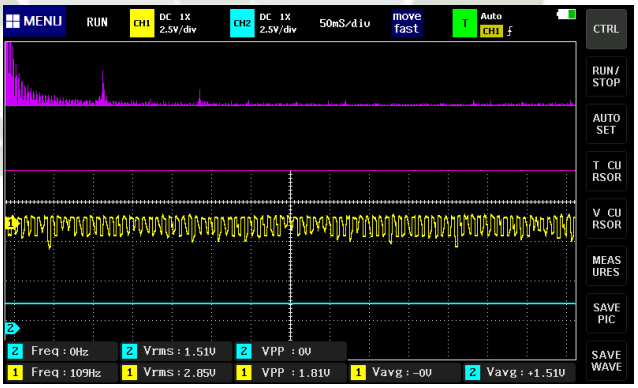
Nota: La figura muestra, la recomendación de conexión del componente B0505S para un correcto funcionamiento. Fuente: Hoja de datos B0505S.

De acuerdo con la ficha técnica del componente, es imperativo mantener un consumo mínimo de 6mA. Por consiguiente, se procederá al cálculo de una resistencia que se incorporará en paralelo al circuito completo para asegurar que se cumpla esta especificación.

Procedemos a probar el circuito con el B0505S de por medio, observamos el resultado

Figura 99

Señales aisladas al ruido eléctrico.



Nota: En la figura se observa la captura de señal con un osciloscopio, en el canal amarillo se muestra la señal de salida del amplificador previo al filtrado , este circuito está aislado del ruido por el componente B0505S, se observa que el ruido que logra pasar, es mucho menor que antes, en el canal cian se muestra la señal filtrada, se observa que, el filtro elimino el ruido por completo, en el dominio de la frecuencia no se observa ni un solo pico, ya que la señal que se inyecta para la prueba es una señal continua. Fuente propia.

La Figura 96 representa la condición donde el prototipo opera sin una fuente de alimentación aislada; en esta configuración, la señal EOG amplificada muestra un nivel de ruido excesivo. A pesar de que el dsPIC logra filtrar eficazmente esta señal ruidosa, se detectan comportamientos anómalos en otras etapas del prototipo, los cuales están asociadas con la magnitud del ruido introducido.

Por otro lado, la Figura 99 muestra las señales del prototipo que está equipado con la fuente de alimentación de aislamiento. En esta configuración, se observa una mejora notable en la calidad de la señal EOG amplificada, la cual aún no fue filtrada. El ruido presente en la señal es inherente al tipo de señal y, tras el procesamiento mediante un filtro FIR, se obtiene una señal óptima para los propósitos específicos del prototipo. Esta comparativa permite ver la importancia de la implementación de medidas de aislamiento en la fuente de alimentación para mejorar la integridad de las señales en dispositivos electrónicos que procesan información analógica.

3.2. Acondicionamiento de Señal

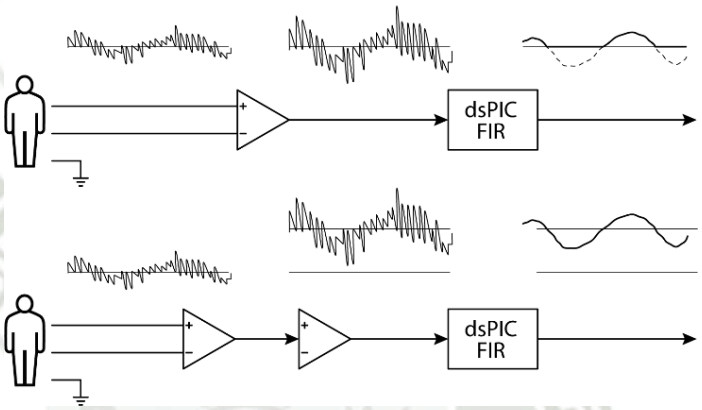
El proceso de acondicionamiento de señales es fundamental para asegurar que las señales que se están analizando permanezcan dentro de los límites aceptables de voltaje y frecuencia que el hardware, como el dsPIC, puede manejar adecuadamente. Este procedimiento es crucial para prevenir daños en el microcontrolador y evitar errores en la interpretación de datos. La ausencia de un adecuado acondicionamiento puede llevar a que las señales superen los límites permitidos del convertidor analógico a digital (ADC), resultando en saturación o recorte (clipping), lo que a su vez puede distorsionar los datos y llevar a interpretaciones erróneas.

Las señales electrooculográficas (EOG), que suelen presentarse en niveles de baja intensidad, desde microvoltios hasta milivoltios, necesitan ser amplificadas para que el

módulo ADC y el procesador digital de señales (DSP) del dsPIC puedan detectarlas adecuadamente. El proceso de amplificación incrementa la amplitud de la señal sin alterar su forma original, permitiendo así que el dsPIC detecte y procese los cambios más sutiles en la señal.

Figura 100

Efectos de la amplificación y desplazamiento positivo de la señal EOG.



Nota: La figura muestra la necesidad de desplazar positivamente la señal EOG, debido a que los módulos ADC de los microcontroladores, en general, no tienen la capacidad de muestrear señales con voltaje negativo, por este motivo se debe montar la señal EOG en un peldaño positivo para permitir que el ADC muestreo toda la señal EOG. Fuente propia.

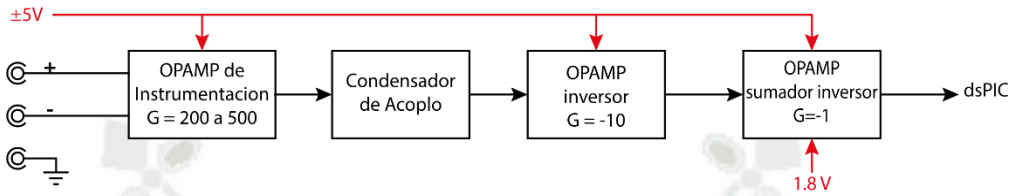
En la mayoría de los microcontroladores, como el dsPIC, los ADC no pueden procesar señales negativas. Introducir un desplazamiento positivo, u offset, eleva toda la señal a un rango de voltaje que el ADC puede manejar con normalidad. Esta adaptación es esencial para garantizar que todas las variaciones de la señal EOG, ya sean positivas o negativas, sean capturadas y digitalizadas correctamente, mejorando así la fidelidad de las mediciones y optimizando el uso del rango dinámico disponible en el ADC.

Para efectuar la amplificación y el desplazamiento positivo de la señal electro oculográfica (EOG), el proceso se estructura en dos fases diferenciadas, como se muestra en la Figura 101. Inicialmente, la señal es procesada mediante un amplificador operacional de instrumentación (OPAMP), que pre amplifica la señal en un factor que varía entre mil y

cinco mil veces. Esta amplificación permite que las variaciones sutiles en la señal sean más discernibles y manejables para etapas posteriores de procesamiento.

Figura 101

Acondicionamiento de la señal EOG.



Nota: La figura muestra el diagrama de bloques de la etapa de acoplamiento de la señal EOG hacia el ADC del dsPIC, en esta etapa se pre amplifica, se desacopla la señal de componentes DC, se vuelve a amplificar, y por último se suma 1.8V a la señal para subirle a un escalón positivo. Fuente propia.

Posteriormente, la señal amplificada pasa por un condensador de desacoplo DC, con el fin de eliminar el desplazamiento de voltaje de los electrodos (Capitulo 1, apartado 3.7), con la señal siempre oscilando respecto a GND, se vuelve a aplicar una amplificación final, en este punto, obtenemos una señal en aproximadamente 1.5 V de amplitud pico, pero aún conserva su lado negativo. Para hacer que la señal siempre sea positiva, a la señal se le suma un desplazamiento positivo de voltaje, este desplazamiento corresponde aproximadamente a la mitad del voltaje de referencia de 3.3V, es decir 1.8V, facilitando así que la señal modificada permanezca dentro del rango operativo del convertidor analógico a digital del sistema.

Para concluir el proceso de acondicionamiento de la señal y asegurar la compatibilidad de las impedancias, se incorpora un seguidor de voltaje. Este componente es fundamental para mantener la continuidad y estabilidad de la señal ajustada, asegurando que la transferencia de la señal hacia el microcontrolador dsPIC se realice sin degradaciones significativas de la señal o desajustes en la respuesta de frecuencia.

3.3. Filtrado

Para determinar el punto de corte que tendrá el filtro FIR que usaremos para nuestro propósito, debemos tener claro el rango de frecuencias en las que se ubican las señales útiles para nuestro propósito. Según la **Tabla 20**, la técnica de Electro óculo grafía (EOG) tiene un rango de frecuencias desde 0 Hz, hasta 50 Hz, sin embargo, en las diferentes pruebas que se realizó con el hardware del prototipo, fijaremos esta frecuencia de corte a 10 Hz y mantendremos la banda de transición hasta, aproximadamente, los 50 Hz.

Para realizar los cálculos de los coeficientes del filtro, utilizaremos el método del Enventanado del filtro paramétrico que se mostró en el apartado 3.4.3 del capítulo anterior. Para esto, utilizaremos la ventana Hamming y calcularemos 251 coeficientes, haciendo que el filtro sea de orden 250, esto con el propósito que el filtro tenga una banda de transición pequeña, el cual permita ser más selectivo.

Tabla 49
Diseño de filtro FIR mediante parámetros iniciales.

Parámetro	Valor
Frecuencia de muestre (f_s)	5 kHz
Frecuencia de corte (f_c)	10 Hz
Orden del Filtro FIR	250
Tipo de Ventana	Hamming
Atenuación Máxima de la Banda de Transición	-41 dB

$$\Delta\omega_N = \frac{6.6\pi}{M} \quad \rightarrow \quad \Delta\omega_N = 0.0829$$

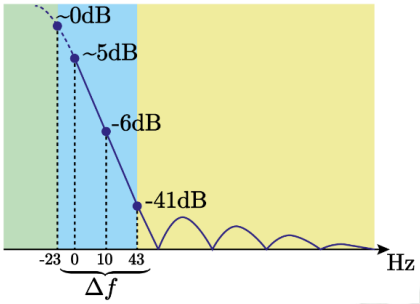
$$\Delta f = 66\ Hz$$

$$\Rightarrow f_p = -23\ Hz \ ; \ f_s = 43\ Hz$$

Según los parámetros obtenidos de la **Tabla 17**, calculamos los límites teóricos de la banda de transición del filtro a diseñar, podemos observar que tiene un componente negativo, sin embargo, en la práctica, este límite se corta a los 0 Hz, y va hasta los 43 Hz.

Figura 102

Banda de transición del filtro FIR EOG paramétrico.



Nota: La figura muestra la banda de transición de un filtro FIR enventanado con una ventana Hamming. Fuente propia.

3.3.1. Cálculo de Coeficientes del Filtro FIR

Para calcular los coeficientes, se realiza una multiplicación punto a punto entre la función que representa la Respuesta al Impulso Deseado "h_d" (Filtro pasa bajos ideal) con la Ventana "w", el cual enmascara la Respuesta al Impulso, "cortando" todos los puntos que estén por fuera de la ventana de tamaño igual al orden del filtro. Ambas funciones, Respuesta al Impulso y la Ventana, son desplazadas horizontalmente de manera simétricamente para encajar con el orden del filtro (M).

Tabla 50

Respuesta al impulso y ventana para filtrado pasa bajos.

Respuesta al impulso deseada	Ventana Hamming
$h_d(n) = \begin{cases} \dfrac{\sin\left(\omega_c\left(n-\frac{M}{2}\right)\right)}{\pi\left(n-\frac{M}{2}\right)} & ;\ 0 \le n \le M\ ;\ n \neq \dfrac{M}{2} \\ \dfrac{\omega_c}{\pi} & ;\ n = \dfrac{M}{2} \end{cases}$	$w(n) = \begin{cases} 0.54 - 0.46 \cdot \cos\left(\dfrac{2\pi n}{M}\right) & 0 \le n \le M \\ 0 & en\ el\ resto \end{cases}$

Debemos realizar una evaluación de las funciones presentadas en la Tabla 50 (estas funciones ya fueron analizadas en el apartado 3.4.3 del capítulo anterior), el resultado de cada función evaluada, es multiplicada punto a punto, obteniendo como resultado, los coeficientes del filtro FIR que hemos diseñado de forma paramétrica.

Tabla 51

Evaluación y multiplicación punto a punto de funciones para cálculo de coeficientes.

n	hd(n)	w(n)	h(n)= hd(n)·w(n)	Coeficiente (bk)
0	0.002546479	0.08	0.000203718	b_0
1	0.002566813	0.080145273	0.000205718	b_1
2	0.002587068	0.080581	0.000208469	b_2
…	…	…	…	…
124	0.003999895	0.999854727	0.003999314	b_{124}
125(M/2)	0.004	1	0.004	b_{125}
126	0.003999895	0.999854727	0.003999314	b_{126}
…	…	…	…	…
248	0.002587068	0.080581	0.000208469	b_{248}
249	0.002566813	0.080145273	0.000205718	b_{249}
250 (M)	0.002546479	0.08	0.000203718	b_{250}

En la Tabla 51 se realiza la evaluación y se muestra los puntos que muestran que los coeficientes cumplen la propiedad de simetría alrededor de la mitad del orden M, para ver todos los coeficientes, se puede consultar el ANEXO F.

3.3.2. Corrección de Ganancia del filtro FIR

Como se observa en la Figura 102 , parte de la banda de transición del filtro que estamos evaluando se encuentra en el eje negativo de las frecuencias, y ya que esta porción de la banda de transición no puede existir, la respuesta frecuencial del filtro atenúa desde el inicio de la respuesta frecuencial (0 Hz), es decir, nuestro filtro tiene el comportamiento de un Atenuador.

Esto se comprueba revisando si los coeficientes cumplan la condición de ganancia unitaria del filtro, si sumamos los valores de los 251 coeficientes que se calculan en la Tabla **51**, obtendremos que el resultado no llega a la unidad, por lo que tenemos que multiplicar a todos los coeficientes con un valor, el cual, haga que la nueva suma sea la unidad, es decir, este factor será el inverso de la primera suma.

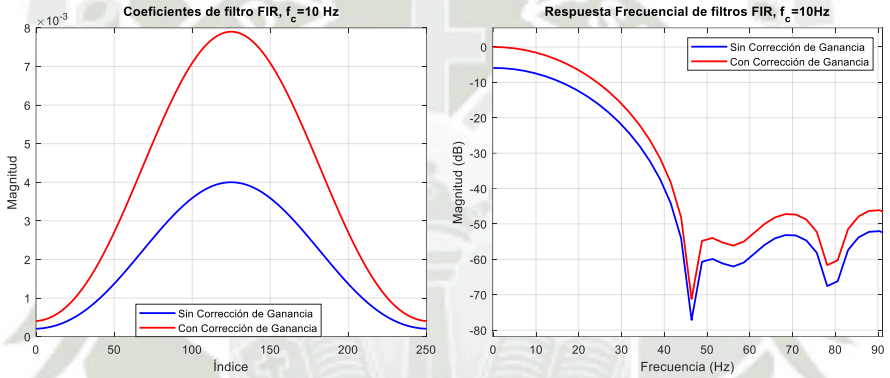
Tabla 52

Cálculo del factor de corrección de los coeficientes.

Operación	Valor
Suma de los 126 coeficientes del ANEXO F (Mitad simétrica)	0.25511383
Suma de todos los 251 coeficientes	0.50622766
Factor de corrección para lograr la ganancia unitaria del filtro FIR.	$^{1}/_{0.50622766}$ 1.975395813

Figura 103

Comparación de los valores de los coeficientes, y la respuesta frecuencial, entre un filtro sin corrección y uno corregido.



Nota: La figura muestra la comparación de la respuesta frecuencia de un filtro con coeficiente sin ganancia unitaria, con un filtro con coeficientes corregidos. En la curva azul se puede observar que el filtro atenúa considerablemente desde los 0Hz, para corregir esto, se multiplica a los coeficientes por su factor de corrección. Fuente propia.

En la Tabla 53 se realiza la corrección de los coeficientes, y se muestra únicamente los puntos que muestran que los coeficientes cumplen la propiedad de simetría alrededor de la mitad del orden M, para ver todos los coeficientes corregidos, se puede consultar el ANEXO G.

Tabla 53

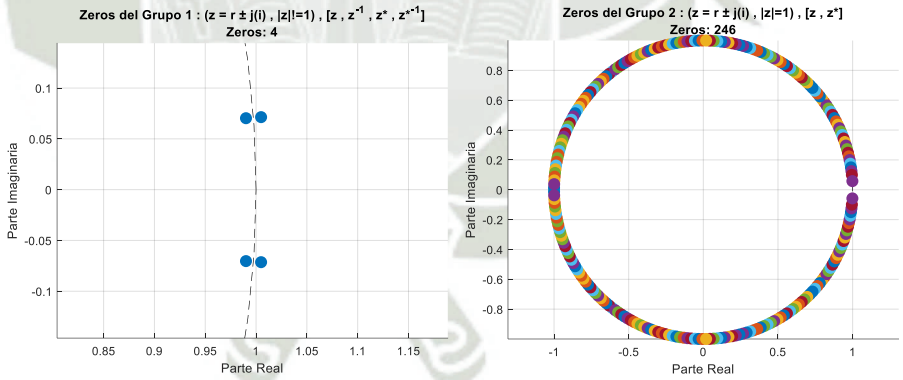
Corrección de los coeficientes del filtro FIR para obtener una ganancia unitaria (Factor: 1.975…).

n	Coeficientes Sin Corrección de Ganancia	Coeficientes Con Corrección de Ganancia	Coeficiente (bk)
0	0.000203718	0.000402424	b_0

n	Coeficientes Sin Corrección de Ganancia	Coeficientes Con Corrección de Ganancia	Coeficiente (bk)
1	0.000205718	0.000406374	b_1
2	0.000208469	0.000411807	b_2
…	…	…	…
124	0.003999314	0.007900217	b_{124}
125(M/2)	0.004	0.007901573	b_{125}
126	0.003999314	0.007900217	b_{126}
…	…	…	…
248	0.000208469	0.000411807	b_{248}
249	0.000205718	0.000406374	b_{249}
250 (M)	0.000203718	0.000402424	b_{250}

Por último, para comprobar que nuestro filtro corregido es simétrico y tiene fase lineal, además de observar los 251 coeficientes del filtro, podemos comprobar la ubicación geométrica de los ceros del filtro, los 250 ceros que este debe tener, deben estar agrupados de acuerdo a los grupos que se muestran en la **Figura 50**.

Figura 104.
Comprobación del balanceo de los ceros del filtro FIR corregido.



Nota: La figura muestra la ubicación y agrupación de los 250 zeros del filtro FIR de orden 250 que se diseñó para el prototipo, se observa que hay 4 zeros pertenecientes al grupo 1, y 246 zeros pertenecientes al grupo 2. Fuente propia.

Luego de analizar y agrupar los 250 ceros que tiene el filtro FIR corregido, haciendo uso del software Matlab, se define que estos conforman el Grupo 1 y el Grupo 2. Como se observa en la Figura 104, en el Grupo 1 se tiene un cuarteto de ceros (inverso, complejo conjugado e inverso complejo conjugado), en el Grupo 2 se tiene 123 pares de ceros

(complejo conjugado), haciendo en total 250 ceros agrupados. Sin sobrar ni faltar un cero, podemos asegurar que el filtro es simétrico, tiene fase lineal y tiene ganancia unitaria, por tal motivo podemos proceder a capturar las señales EOG para empezar a procesar la información.

3.3.3. Captura y Filtrado de Señales EOG con Filtro FIR

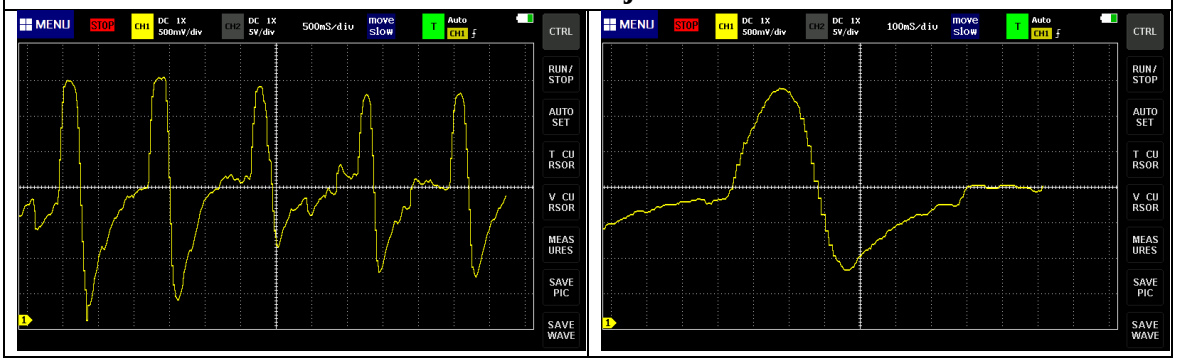
El movimiento de los ojos genera una señal eléctrica característica, es necesario conocer el comportamiento de esta señal para cada movimiento y como esta varia por factores exteriores.

En la Tabla 54 se observa que el movimiento ocular en una dirección, como se mencionó previamente, genera una señal que, tras pasar por la etapa de filtrado digital, adopta la forma de un ciclo senoidal. Esto se debe principalmente al uso de un condensador de acoplamiento, empleado para eliminar la desviación de voltaje del electrodo. Sin este condensador, la señal resultaría en un único pico por cada movimiento.

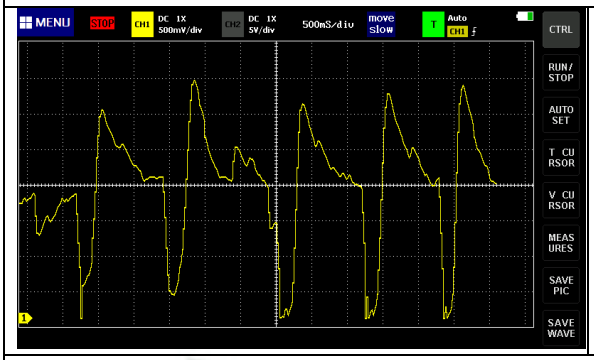
Asimismo, en dicha tabla se ve que, aunque las señales de un mismo tipo comparten una forma similar, no son siempre idénticas, ya que pueden presentar variaciones en la amplitud o la duración. Además, existe ruido ocasionado por factores externos que no pueden ser filtrados electrónicamente, como el movimiento de los electrodos o su desgaste.

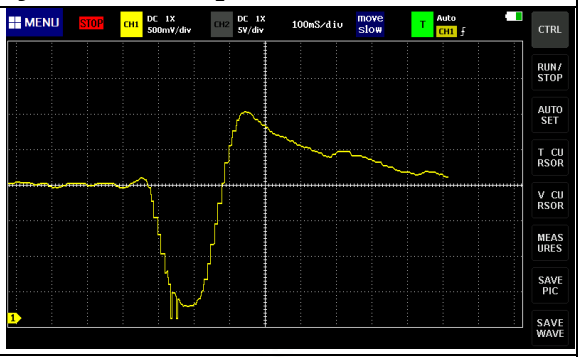
Tabla 54

Señal filtrada del movimiento de los ojos, salida DAC dsPIC

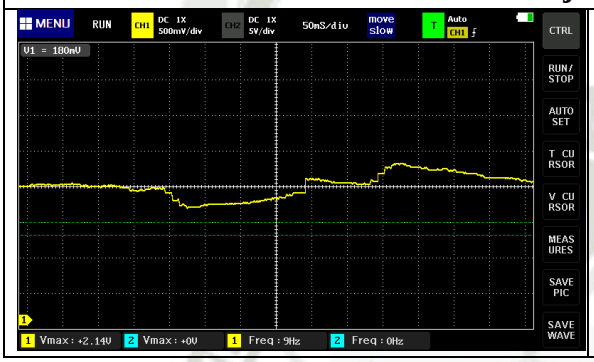
Señal de movimiento de ojos hacia la Derecha


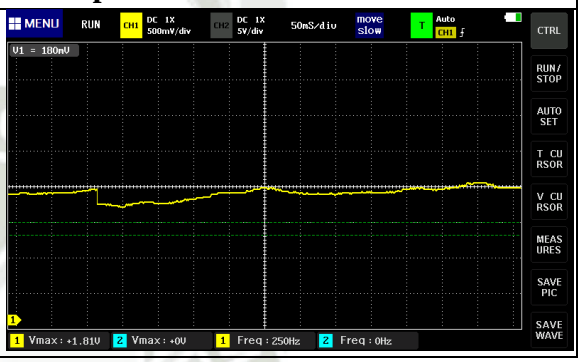
Señal de movimiento de ojos hacia la Izquierda





Señal de ojos de Reposo



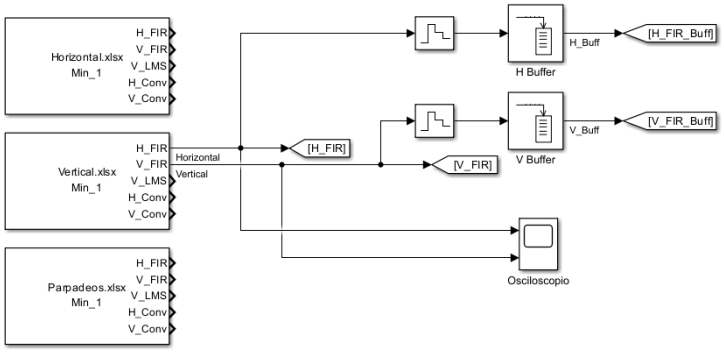


Cuando los ojos están en reposo, se esperaría que la señal sea constante en el nivel de referencia central, sin embargo, como se muestra en la tabla, esta presenta variaciones leves alrededor de la referencia central. Por esta razón, es necesario desarrollar un algoritmo capaz de ignorar estas fluctuaciones y centrarse exclusivamente en las características más esenciales de la señal de movimiento de los ojos, para lograr esto nos apoyaremos de Matlab y Simulink, para realizar un análisis minucioso de las señales.

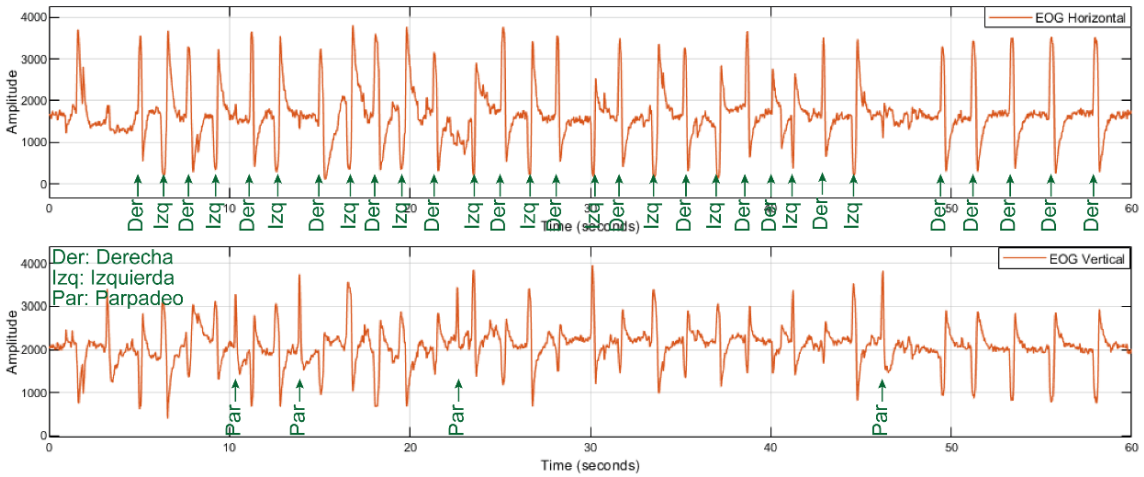
Haciendo uso del módulo UART del dsPIC, capturamos las señales que serán utilizadas para la digitalización de la señal y reconocimiento de patrones, para este propósito, capturamos 4 minutos de movimientos para cada eje y para los parpadeos, teniendo en total 12 minutos de señal a analizar.

Figura 105

Ingreso de señales EOG capturadas hacia Simulink



a)



b)

Nota: a) Introducción y manejo de señales EOG capturadas por el dsPIC a Simulink. b) Validación de la señal EOG en Simulink mediante correlación con movimientos oculares voluntarios. Fuente propia.

Una vez capturadas las señales EOG para cada movimiento, y habiendo guardados estos datos en un archivo de Excel, acotamos el tiempo a analizar a 1 minuto por prueba, esto nos permitirá ver la señal de una forma más entendible. Haciendo uso del bloque “From Spreadsheet” de Simulink, introducimos la señal EOG capturada al entorno de Simulink, como primer paso, comprobamos que la señal que se muestra en el osciloscopio es idéntica a la que capturamos.

Tabla 55
Lista de movimientos oculares horizontales en el primer minuto de la muestra.

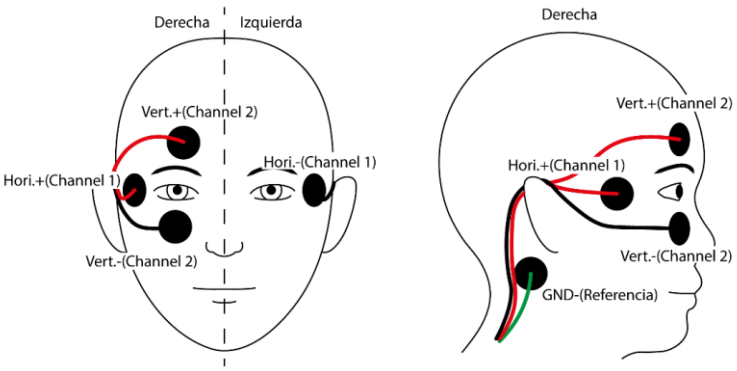
Movimiento Ocular	Cantidad
Derecha	18
Izquierda	12
Parpadeo	4

Identificamos y marcamos los instantes en los que se realizaron los movimientos oculares, esto nos servirá para diseñar, probar y medir la eficacia del algoritmo que diseñemos.

3.4. Ubicación de Electrodos para la Adquisición de Señal EOG

Hasta este punto, para el proceso de captura de las señales EOG, no se le ha dado importancia a la polarización de los electrodos, alternando la polaridad de los ejes entre pruebas para observar los cambios y la respuesta del sistema de adquisición de las señales. Sin embargo, para la etapa de control del mouse mediante la señal obtenida a partir del movimiento de los ojos, es esencial establecer un sentido único para poder implementar un algoritmo que trabaje con señales estandarizadas. El sentido de las señales EOG dependerá de la polaridad aplicada a los electrodos en los ejes horizontal y vertical, como se muestra en la Figura 106.

Figura 106
Ubicación y polarización fijada de los electrodos, para la captura de señales EOG.

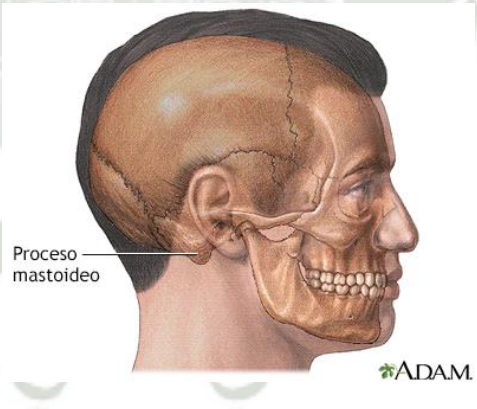


Nota: La figura muestra la ubicación y polarización de los electrodos que se utilizara para la adquisición de señales EOG. Fuente propia.

La correcta configuración y polarización de los electrodos es fundamental para mejorar significativamente la precisión del control de dispositivos basados en señales EOG. Una adecuada configuración asegura que las señales eléctricas generadas por el movimiento ocular se interpreten correctamente, lo que resulta en un control más preciso y fiable del cursor del mouse.

Además, al ubicar y polarizar los electrodos de manera que las señales EOG describan una polarización equivalente a la de un plano cartesiano (valores positivos hacia la derecha y negativos hacia la izquierda en el eje horizontal, y valores positivos hacia arriba y negativos hacia abajo en el eje vertical), facilitan una interpretación consistente de las señales de movimiento ocular. Esta configuración es crucial para el procesamiento y utilización eficaz de las señales en el diseño del algoritmo de control del mouse.

Figura 107
Ubicación generalizada del electrodo de referencia.



Nota: La figura muestra el lugar donde se colocará el electrodo de referencia a tierra, siendo el hueso Mastoideo el lugar seleccionado. Fuente: Tomado de SmartEngage.

La ubicación del electrodo de referencia, que representa el punto equipotencial entre el cuerpo y el prototipo, se coloca sobre la protuberancia del cráneo detrás del pabellón auricular conocida como “Apófisis mastoides” o “Proceso mastoideo”. Esta elección se debe a que los huesos tienen un potencial eléctrico bajo o nulo, proporcionando un punto de referencia estable para la adquisición de señales biológicas.

El uso de un electrodo de referencia en la Apófisis mastoides es común en diversas técnicas de bioseñales, incluida la electroencefalografía (EEG) y la EOG. Esto se debe a la estabilidad eléctrica de esta ubicación, lo cual minimiza las interferencias y proporciona un punto de referencia consistente (Nunez & Srinivasan, 2006).

CAPÍTULO III

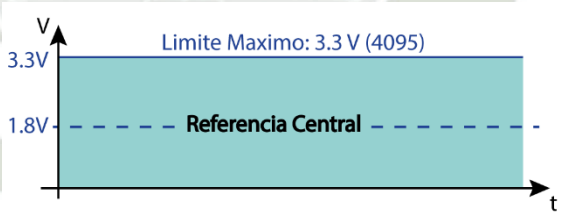
1. DIGITALIZACIÓN Y CLASIFICACIÓN DEL MOVIMIENTO OCULAR

El objetivo de este apartado es desarrollar un algoritmo para la digitalización de las señales EOG filtradas, consiguiendo de manera aproximada una señal que refleje las acciones oculares de dicha señal analógica que únicamente tenga como máximo 3 valores posible. Esto permitirá analizar la señal digital obtenida y aplicar un algoritmo para identificar y clasificar los movimientos oculares, con el propósito de controlar el desplazamiento del puntero de un mouse USB.

ara poder analizar y caracterizar las señales EOG, utilizaremos las señales obtenidas en el apartado 3.3.3 del Capítulo anterior. Adicionalmente, debemos recordar que el nivel de voltaje máximo con el que trabajaremos en esta sección es de 3.3V, y de forma digital, el valor máximo del ADC es 4095 (2^{12}-1), valores superiores a estos, sufren un corte por parte de los Amplificadores Operacionales, este corte también es conocido como “Clipping”.

Figura 108

Delimitación de las áreas de variación de las señales EOG.



Nota: La figura muestra la delimitación del área de voltaje que se usara en el prototipo. Fuente propia.

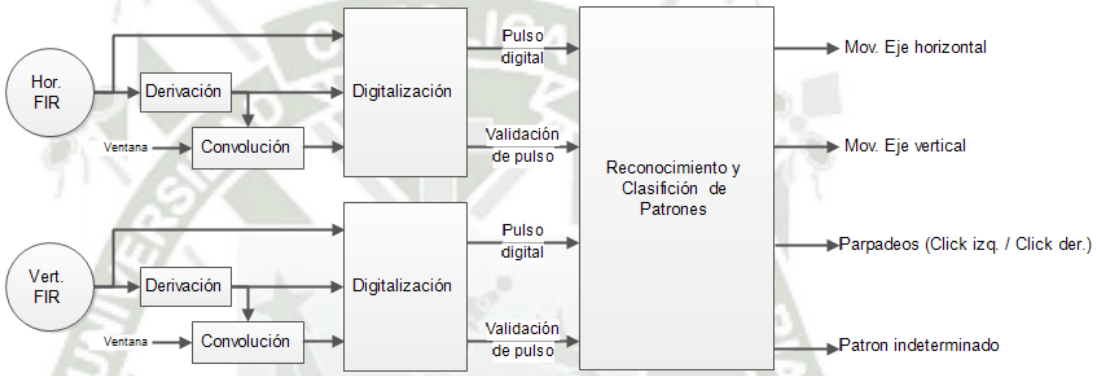
Las ondas que se generan al mover los ojos, si bien no siempre son iguales en amplitud o duración; debido a variaciones como intensidad del movimiento, distancia del desplazamiento o variaciones propias del contacto entre el electrodo, la piel o el cable; la forma y orden de los valles de la onda están asociadas de forma directa a cada movimiento de los ojos.

Luego de un análisis extenso, se diferencia una secuencia de tres etapas definidas para lograr el objetivo:

1. Preparación de la señal filtrada,
2. Digitalización de la señal EOG por reconocimiento de la onda EOG.
3. Reconocimiento de patrones y clasificación de movimientos oculares

Figura 109

Flujograma general del algoritmo para el procesamiento de señales EOG.



Nota: La figura muestra el diagrama de bloques que seguirá el algoritmo para la digitalización, caracterización y clasificación de señales EOG para el control de movimiento de un mouse USB. Fuente propia.

La etapa de preparación y digitalización de las señales se lleva a cabo de manera independiente para cada uno de los canales correspondientes a las señales EOG. Por otro lado, la etapa de reconocimiento y clasificación de las señales digitales se realiza considerando el comportamiento conjunto de ambas señales, ya que los movimientos oculares generan una interrelación entre las señales EOG horizontales y verticales debido a la proximidad física de los electrodos.

1.1. Preparación de la Señal Filtrada

La señal EOG obtenida, luego de pasar por el sistema de adquisición y acondicionamiento de señal por hardware, se caracteriza por estar en lo posible al nivel de la referencia central, existen pequeñas variaciones positivas o negativas que se asocian a

cambios físicos, como el movimiento del contacto entre el electrodo y el cable, sin embargo, estas variaciones son de corta duración y amplitud baja, generalmente no salen de la referencia central. Para distinguir los movimientos oculares entre derecha, izquierda, arriba, abajo y el parpadeo, se establece la regla de que, si la señal EOG se desvía lo suficiente de la "referencia central", se está produciendo un movimiento ocular.

El primer problema que enfrentamos es que esta regla a veces puede inducir a falsos positivos, en estos casos, esta “referencia central” sufre alteraciones debido a factores externos al prototipo, modificando el valor de esta. Por este motivo, un análisis basado en la medición de la amplitud queda descartado, por lo contrario, nos basaremos en el análisis del cambio de la señal mediante la derivada de la misma, de esta manera no afectara cuando una voltaje DC se acople a la señal, siempre que esta no induzca al corte superior o inferior de la señal. Por este motivo, lo primero que se realizara es realizar la derivada discreta de la señal EOG.

La deriva discreta se define como la diferencia entre dos muestras consecutivas de una señal discreta, dividida por el período de muestreo de la señal. Sin embargo, en muchos casos, podemos omitir la división por el tiempo de muestreo, lo que nos permite obtener una derivada normalizada que mantiene la información relativa sobre la variación de la señal sin la dependencia directa del intervalo de muestreo, esto nos permite tener una señal independiente a la resolución temporal.

Adicionalmente, en lugar de emplear la derivada discreta estándar, optamos por una derivada discreta normalizada con una técnica de realce de extremos. Esta variante tiene la particularidad de amplificar los cambios más significativos en la señal, favoreciendo la detección de variaciones abruptas. Este enfoque es útil en situaciones donde se requiere una

mayor sensibilidad a los picos y valles de la señal, mejorando la identificación de eventos transitorios, como los movimientos oculares rápidos o los parpadeos.

Tabla 56
Ecuación en diferencia de la derivada discreta

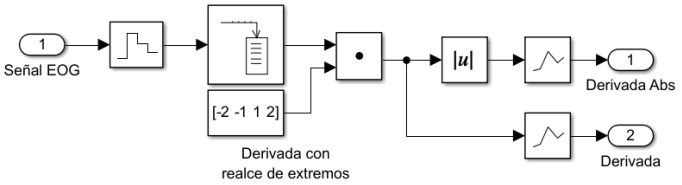
Derivada de x[n]	Ecuación en diferencias
Derivada Discreta	$\delta[n] = \frac{x[n] - x[n-1]}{T_s}$
Derivada Discreta Normalizada	$\delta[n] = x[n] - x[n-1]$
Derivada con realce de extremos	$\delta[n] = 2 \cdot x[n] + x[n-1] - x[n-2] - 2 \cdot x[n-3]$

La derivada absoluta de la señal se calcula con el fin de eliminar la dependencia del signo de la variación de la señal (ya sea ascendente o descendente), permitiendo enfocar el análisis únicamente en los cambios de magnitud. Al aplicar este enfoque, no se distingue si la señal está aumentando o disminuyendo, sino que se destacan los puntos en los que ocurre un cambio sustancial en la pendiente, lo cual es relevante para detectar eventos como picos o transitorios en la señal.

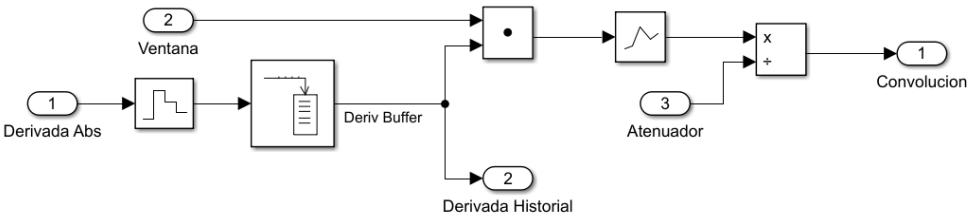
La segunda señal necesaria para la digitalización se obtiene mediante la convolución de la derivada absoluta de la señal con una ventana rectangular. Este proceso tiene como objetivo identificar los momentos en los que el paso de un pulso deja de tener influencia significativa en el canal, marcando el fin de la reacción de la señal EOG ante un movimiento ocular.

Figura 110

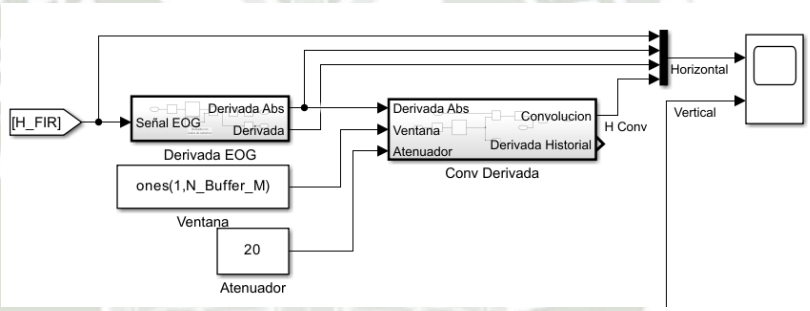
Bloques de Simulink para las etapas iniciales del procesamiento de señales EOG.



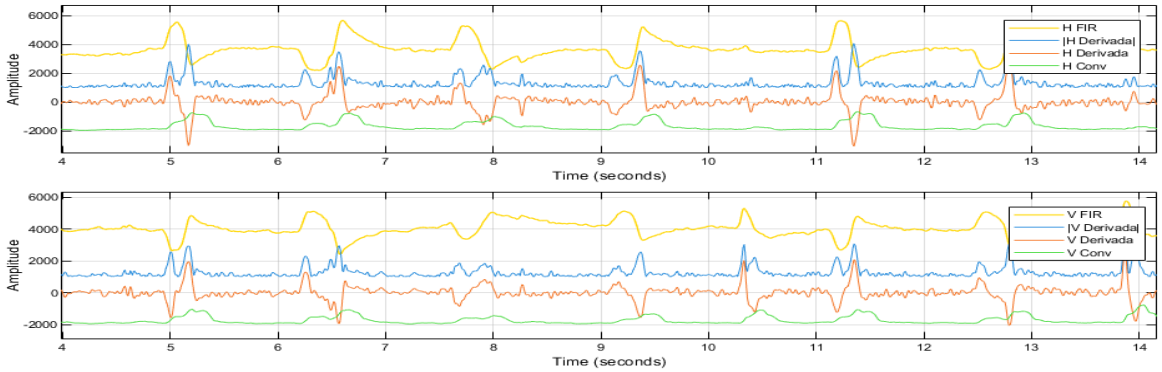
a)



b)



c)



d)

Nota: a)Bloque de derivación de la señal EOG b)Bloque de convolución de la derivada absoluta c)Proceso de preparación de señales para la digitalización. d)Señales resultantes de la preparación de señales para la digitalización. Fuente propia.

1.2. Digitalización de la Señal

Podemos proceder a buscar una relación entre las variaciones de la señal EOG frente a los movimientos oculares, , y las señales calculadas en el paso anterior (derivada y convolución). Esto nos permitirá identificar patrones repetitivos y diseñar un algoritmo que contemple estos cambios para su detección y modelado.

Como se mencionó anteriormente, un pulso EOG, generado por el movimiento ocular, se considera que ha iniciado cuando la derivada de la señal presenta un cambio que supera un umbral preestablecido. Este pulso, al ser limitado, alcanzará un valor máximo (o mínimo) y, por un breve periodo, dejará de variar. En este punto, la derivada de la señal cae por debajo del umbral, aproximándose a cero, lo que indica que podemos comenzar a buscar el pico de la señal.

Dado que la señal ocular siempre regresa a la referencia central tras el final de un estímulo, el siguiente evento que experimenta la señal EOG es un cambio inverso al inicial, el cual marca el fin del pulso EOG asociado al movimiento ocular.

En la Tabla 57 y la Figura 111 se presenta una secuencia de eventos repetitivos para todas las señales EOG. A partir de esta secuencia, se calcula la altura del pico, y dicho valor se asigna a la altura que tomará la señal digital durante toda la duración del pulso. Este procedimiento permite comparar la altura de un canal con respecto al otro, lo que facilita la identificación del movimiento ocular correspondiente, basándose en cuál de las señales digitales es mayor o menor.

Tabla 57
Secuencia de eventos para la digitalización de un pulso EOG

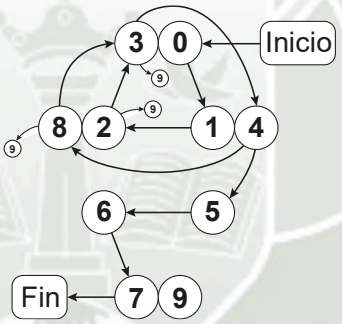
Secuencia	Etapa	Descripción
0	Búsqueda de inicio de impulso	Se busca un cambio significativo en la derivada de la señal, esto nos indica que la señal EOG tuvo un cambio brusco el cual podría ser el inicio de un pulso EOG.
1	Búsqueda de final de impulso	Cuando la derivada de la señal empieza a regresar a cero, significa que la señal EOG dejo de cambiar, por lo que en ese momento, la señal EOG presenta un pico máximo positivo o negativo
2	Búsqueda de pico	Se verifica el signo que tuvo la derivada de la señal, si esta fue positiva, se busca el valor máximo de la señal EOG. Si la derivada fue negativa, se busca el valor mínimo de la señal EOG.
3	Búsqueda de inicio de impulso	En el canal de la derivada, se espera el inicio de un cambio significativo de signo opuesto al primero, esto significa que la señal EOG ha empezado a cambiar opuestamente a como lo hizo al inicio
4	Búsqueda de final de impulso	Cuando la derivada de la señal empieza a regresar a cero, significa que la señal EOG dejo de cambiar, por lo que en ese momento, la señal EOG presenta un pico máximo opuesto al primero, marcando que el final del pulso EOG.
5	Búsqueda de pico Opuesto	Se busca el pico opuesto de la señal EOG, que debe encontrarse muy próximo a los datos de la señal más reciente.
6	Transición final	Con el pulso EOG finalizado, se espera que la señal se normalice, para esto, se espera a que la convolución de la señal, este por debajo de un umbral predeterminado.
7	Reinicio	En este punto, el análisis de la señal EOG se completó sin tener casos anormales, por lo que todas las variables deben reiniciarse para dejar al algoritmo, a la espera de detectar el siguiente pulso.
8	Búsqueda de pico, segunda oportunidad	Al terminar lo que se considera como un posible pulso EOG, la señal EOG no se ha alejado lo suficiente del pico del pulso, por lo que se considera 2 posibilidades. La señal no termino por que presentara un reimpulso. La señal no fue un pulso EOG, sino, la señal EOG se acomodó a un nuevo nivel de referencia. La segunda posibilidad marca al pulso como invalido, por lo que debe anularse de inmediato.
9	Reinicio y anulación	En este punto, el análisis de la señal EOG fue detenido en pleno proceso, debido a irregularidades en la señal EOG, estas

Secuencia	Etapa	Descripción
		irregularidades descartan la fiabilidad del análisis de la señal, por lo que se marca el pulso como anulado, posteriormente se reinicia el algoritmo, dejándolo a la espera de detectar el siguiente pulso.

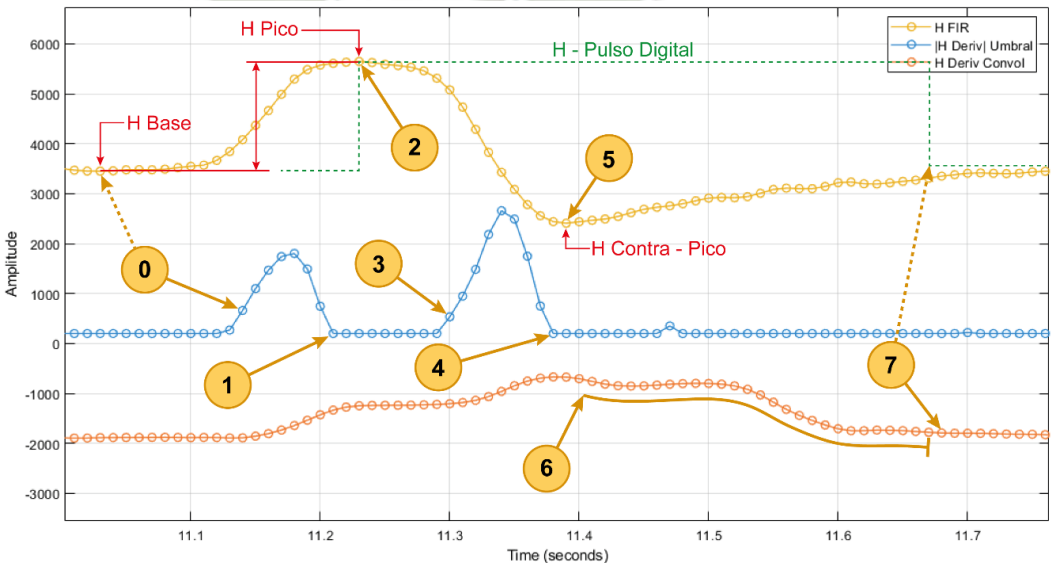
Debido a que la señal EOG es muy sensible a los movimientos de los ojos, el algoritmo detecta múltiples movimientos que no son los que deseamos identificar. Es necesario establecer restricciones y límites de altura y duración de pulso, para evitar falsos positivos. Además, se debe marcar estos pulsos indeseados para que no sean tomadas en cuenta por la etapa posterior, permitiendo que el algoritmo de identificación de movimiento, se encargue únicamente de los movimientos que esta dirigidos para mover el mouse USB.

Figura 111

Secuencia de pasos para la digitalización de las señales EOG.



a)



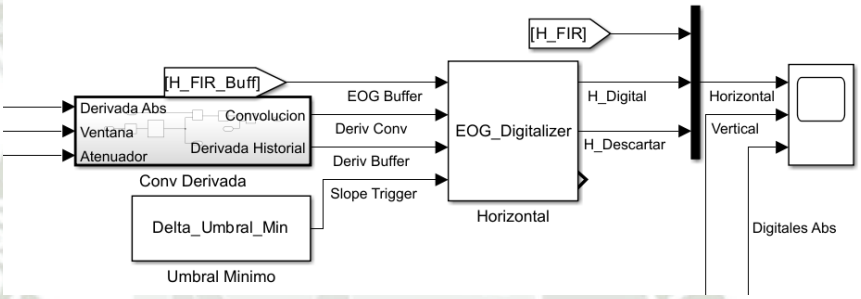
b)

Nota: a) Flujograma de la secuencia de eventos para la digitalización EOG b) Secuencia de eventos en un pulso EOG. Fuente propia.

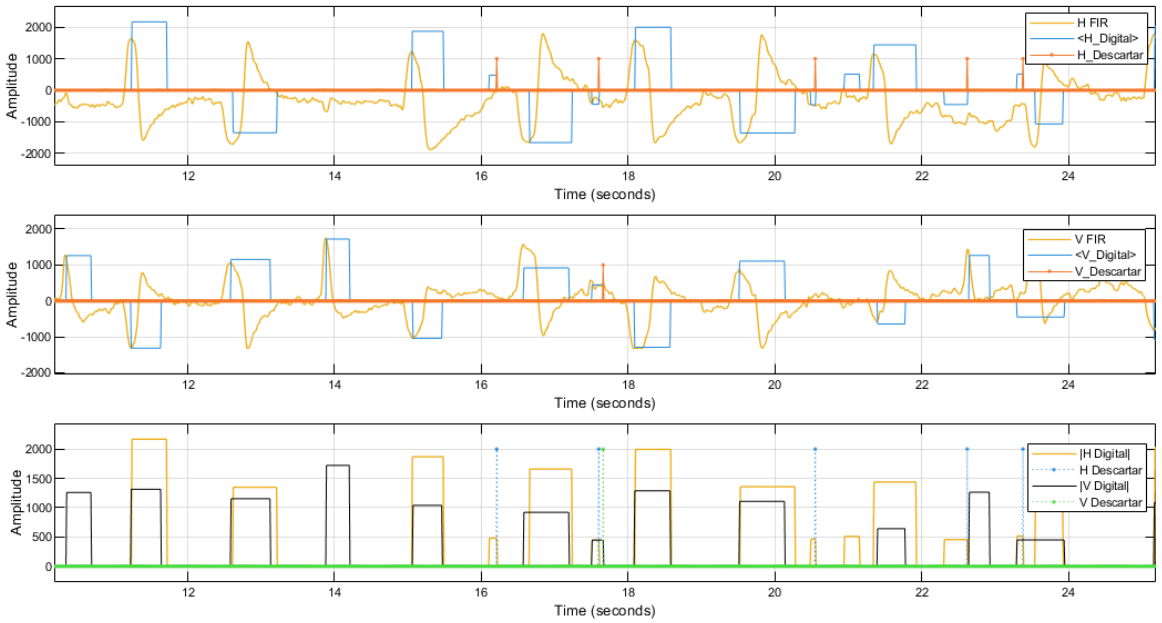
Tomando en cuenta esta secuencia de eventos establecida, restringiendo los pulsos que no deben ser considerados, y tomando precaución de los pulsos que se digitalizaron cuando no debieron, se diseña el algoritmo encargado de digitalizar las señales EOG, haciendo uso de Simulink, ponemos este algoritmo en el bloque *EOG_Digitalizer.m* y procedemos a mostrar los resultados del algoritmo en la Figura 112.

Figura 112

Bloques de Simulink para digitalizar las señales EOG.



a)



b)

Nota: a) Diagrama de bloques de la digitalización EOG en Simulink. b) Resultados de la digitalización EOG del algoritmo diseñado.

Al igual que con la derivada en la etapa anterior, para comparar los pulsos en la siguiente etapa, se toma el valor absoluto de los pulsos digitales, esto se muestra en la Figura 112b.

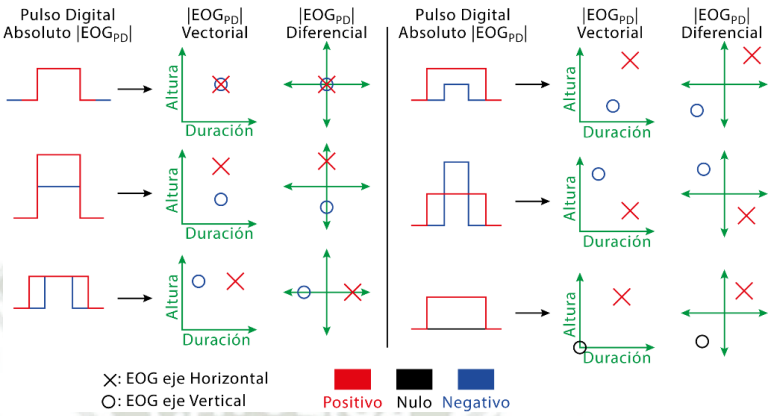
Es importante considerar que el comportamiento de la señal EOG analógica, antes de ingresar a la etapa de filtrado digital, está directamente influenciado por el circuito de acondicionamiento previo. Por este motivo, el algoritmo diseñado en este apartado ha sido diseñado específicamente para trabajar con el circuito de acondicionamiento utilizado en el prototipo desarrollado en esta tesis. Si por alguna razón dicho circuito es modificado o reemplazado, debe ser necesario analizar cómo varía la señal analógica y evaluar el impacto de estos cambios en el desempeño del algoritmo de digitalización. Esto podría implicar la necesidad de ajustar, parcial o totalmente, el algoritmo descrito en este apartado.

1.3. Reconocimiento y Clasificación de Patrones

Esta etapa del algoritmo, se encarga de clasificar los pulsos obtenidos con su movimiento correspondiente, para hacer esto, se debe saber el patrón que hay en la comparación de ambas señales digitales para cada movimiento. Por esta razón, como primer paso, se analizará las relaciones que pueden tener los pulsos digitales de ambos canales, comparando la altura en magnitud y duración de ambos pulsos.

Figura 113

Relación entre los pulsos digitales de ambos canales con referencia en el eje horizontal



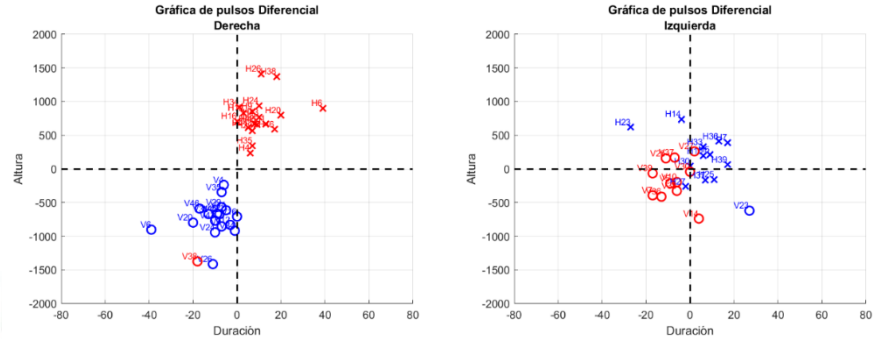
Nota: La figura muestra todos los posibles casos que se puede obtener al relacionar las señales EOG horizontal y vertical digitalizadas, de esta manera se identifica un comportamiento característico y repetitivo. Fuente propia.

Independientemente del movimiento ocular, las señales digitales pueden relacionarse de seis maneras básicas. Estas relaciones, inicialmente descritas en dos dimensiones (duración y altura), se amplían al incorporar una tercera dimensión: la polaridad del pulso. De esta forma, se obtiene un espectro completo que describe las interacciones entre ambos canales.

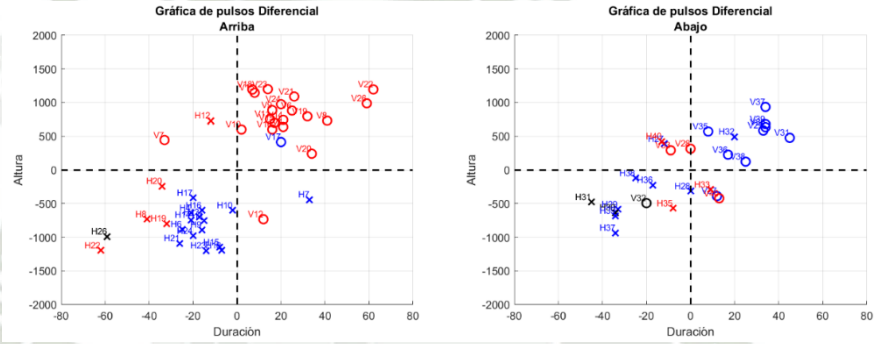
Se ejecuta el algoritmo de digitalización de la señal EOG en los archivos de muestras correspondientes a cada eje, y se almacena todos los pulsos digitales resultantes. Posteriormente, se enumera manualmente los pulsos obtenidos de la digitalización de los tres archivos de muestras y se asocia con los movimientos oculares correspondientes. Esto permite agrupar con exactitud todos los pulsos según el tipo de movimiento ocular que representan. Para cada grupo de pulsos, se realiza una comparación de la duración, altura y polaridad del pulsos, esto se muestra en la Figura 114.

Figura 114

Agrupación manual de pulsos EOG para búsqueda de patrones.



a)



b)

c)

Nota: La figura muestra los patrones obtenidos luego de relacionar el eje horizontal con el vertical, se puede identificar una tendencia única para cada acción con los ojos. a) Eje Horizontal b) Eje Vertical c) Parpadeos Fuente propia.

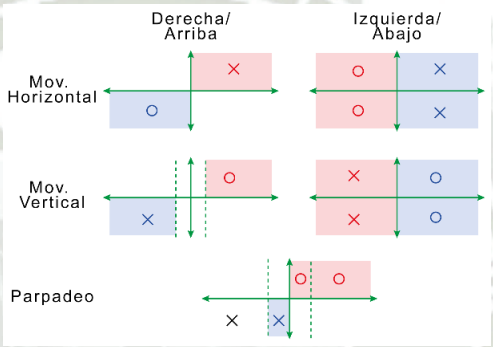
Al observar la distribución de la diferencia de los pulsos para cada movimiento ocular, se puede identificar que estos pulsos siguen un patrón único para cada tipo de

movimiento de los ojos, se procede a generalizar el comportamiento de las señales ante cada movimiento ocular con el patrón.

En la Figura 115 se muestra las reglas que se aplicaran a cada par de pulsos (horizontal y vertical) que se halla digitalizado y que no estén descartadas, con el fin de clasificarlas y asignarles un movimiento ocular. Cualquier otro caso se descartará y se lo considerará como indeterminado.

Figura 115

Identificación de patrones de pulsos para cada tipo de movimiento ocular

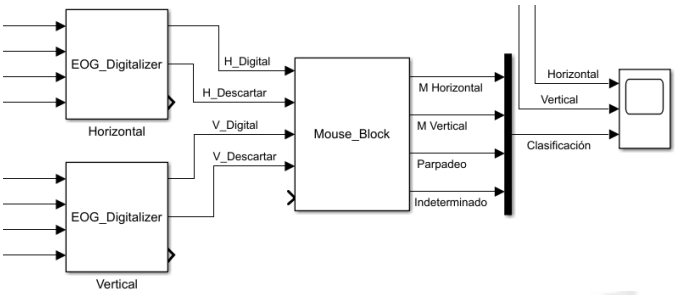


Notal: Luego de identificar los patrones para cada movimiento ocular, se clasifica y asigna una sección para cada patrón de los movimientos oculares, de esta manera se pretende generalizar todos los movimientos oculares. Fuente propia.

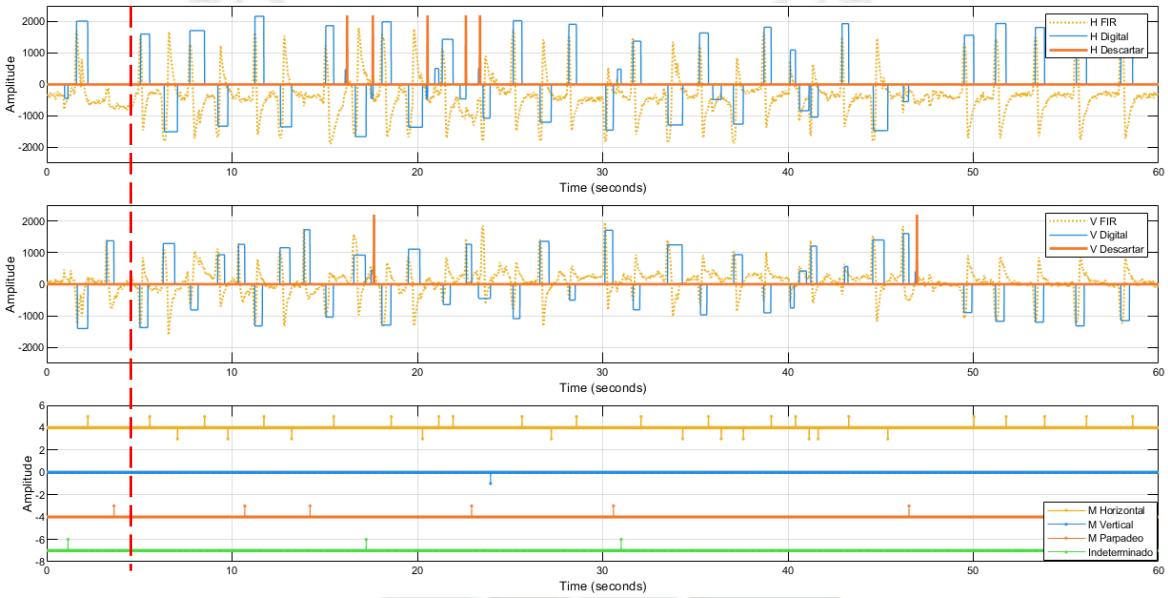
Con el patrón de pulsos del movimiento ocular identificado, se procede a implementarlo en el algoritmo y se guarda en el bloque *Mouse_Block.m*, y para medir su efectividad, la probaremos introduciéndole la señal mostrada en la Figura 105 y se evaluara los resultados.

Figura 116

Clasificación de pulsos EOG con simulación por Simulink.



a)



b)

Nota: La figura muestra el diagrama de bloques necesario para la clasificación de los pulsos EOG

a) Diagrama de bloques de la clasificación de pulsos EOG, b)Resultados de la clasificación de pulsos EOG. Fuente propia.

Los resultados de la clasificación de los pulsos digitales, se puede observar en la Figura 116b, en esta lamina, en el tercera vista, se observa 4 señales que representan a los movimientos oculares, siendo la cuarta, aquellos pulsos que no encajaban con los patrones predefinidos. En la Tabla 58 y

Tabla 59, se muestra una comparación de los resultados del algoritmo de clasificación de movimientos mediante pulsos EOG digitalizados.

Tabla 58

Comparación de los movimientos intencionales con los clasificados con el algoritmo.

Clasificación	Movimientos Intencionales	Movimientos Clasificados
M. Derecha	18	19
M. Izquierda	12	11
M. Vertical	0	1
Parpadeos	4	5
Indeterminado	-	2
TOTAL	34	38

Tabla 59

Medición de la efectividad del algoritmo

Comparación	Conteo	Porcentual
M. Acertados	31	91.2%
M. Mal Interpretados (Error)	3	8.8%
M. Digitalización insuficiente	4	-

Se identificaron tres errores de interpretación en los pulsos asociados a los movimientos hacia la izquierda, los cuales fueron clasificados incorrectamente como "Indeterminado", "Movimiento Abajo" y "Parpadeo". Esto evidencia la necesidad de ajustar y refinar el patrón de clasificación para mejorar la precisión para la clasificación de movimientos hacia la izquierda. Además, se detectaron cuatro pulsos adicionales que, pese a no estar asociados a un movimiento ocular, fueron erróneamente interpretados como tales, los cuales lograron pasar por las restricciones y limites que se pusieron en la etapa de digitalización. No obstante, dado que la cantidad de estos errores es mínima en comparación con el total de pulsos clasificados, se puede asumir que la etapa de digitalización de la señal EOG opera con un nivel aceptable de precisión.

1.4. Generación de Orden de Movimiento del Mouse

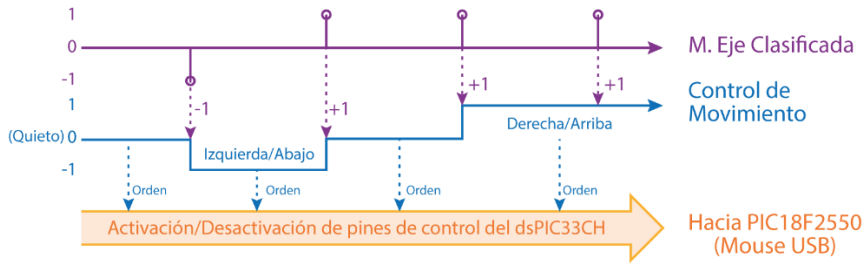
Como último paso para el control de movimiento del mouse mediante uso de las señales EOG, se debe generar una orden para realizar los movimientos voluntarios del mouse.

Estas órdenes de movimientos, se realizará a partir de las señales obtenidas en la etapa de Reconocimiento y Clasificación de patrones. Si bien estas señales representan el movimiento que se hizo con los ojos, no pueden ser utilizadas directamente para controlar el mouse, esto es así, principalmente, por que desde el inicio se planteó que el control del mouse iba a ser mediante la acumulación de movimientos oculares, implementándose un enclavamiento en el movimiento del mouse en función a la acumulación de estos movimientos.

En cada eje se puede realizar 2 movimientos, cada uno de estos movimientos agrega o quita valor a la variable encargada de dirigir el movimiento en un eje, además, esta variable solo puede ir desde 1 hasta -1 en números enteros. Si la señal de movimiento clasificado le suma 1 (derecha o arriba) o le resta -1 (izquierda o abajo) a la variable de su eje correspondiente, el movimiento en dicho eje se realizará dependiendo del valor de dicha variable y se mantendrá hasta que este valor cambie.

Figura 117

Generación de orden de movimiento del puntero del mouse hacia el PIC18F2550



Nota: Cuando el algoritmo de procesamiento de señales EOG determina que un movimiento ocular es producto de una orden para el mouse, el siguiente paso es mandar dicha orden hacia el módulo USB. Fuente propia.

Por ejemplo, si el Control de Movimiento tiene una orden de movimiento hacia la izquierda (valor de -1) y se desea mover el mouse hacia la derecha, será necesario realizar dos pulsos hacia la derecha. El primer pulso hacia la derecha sumará un valor de +1 al acumulador, neutralizando el valor acumulado de -1, lo que llevará al estado de reposo (valor

de 0) y detendrá el movimiento hacia la izquierda. Un segundo pulso hacia la derecha sumará nuevamente +1 al acumulador, llevando el valor del acumulador de 0 a +1, activando así el movimiento hacia la derecha.

Para cada eje (vertical y horizontal), se asigna un par de pines físicos de control en el dsPIC33CH, lo que implica un total de cuatro pines para gestionar el movimiento en las cuatro direcciones posibles. Cada pin, asociado a su respectivo eje, se activará o desactivará en función del sentido determinado por la variable Control de Movimiento. Cuando el estado es de reposo, ambos pines del par correspondiente a un eje permanecerán desactivados simultáneamente.

Estos pines están conectados físicamente al microcontrolador PIC18F2550, que es el encargado de recibir las señales provenientes del dsPIC33CH. A partir de esta información, el PIC18F2550 realiza las operaciones necesarias para traducir dichas órdenes en el movimiento del puntero del mouse en el dispositivo al que esté conectado el prototipo.

2. DISGREGACIÓN DEL PARPADEO, CLIC IZQUIERDO Y DERECHO

Para tener un control completo de un mouse, no solo basta con poder mover el puntero del mouse, sino, realizar acciones mediante los botones click izquierdo y click derecho. Para poder mover el mouse, se usa el movimiento horizontal y vertical de los ojos, dejando a los parpadeos como última acción voluntaria que se puede realizar con los ojos, por lo que usaremos estas para disgregar entre los parpadeos normales (que se realizan de forma involuntaria para descansar los ojos) de los parpadeos con la intención de realizar un click izquierdo o derecho.

2.1. Efecto de los Parpadeos en la Señal EOG

En la Figura 115 Se puede observar que el patrón de movimiento hacia Arriba y el del Parpadeo son similares, esto se debe principalmente al fenómeno de Bell, esto se

describió en el apartado 3.6 del Capítulo 1, si bien en la teoría se menciona que el efecto de este fenómeno en los parpadeos rápidos y automáticos no genera el movimiento ascendente ocular y por consecuencia no debería generar alguna señal bioeléctrica, en la práctica se vio que estos parpadeos si generan señales bioeléctricas, los cuales son detectados y clasificados por el algoritmo como tales.

2.2. Separación por Parpadeos Consecutivos

El acto de parpadear, por sí solo, no proporciona información suficiente para diferenciar entre un parpadeo normal y aquellos destinados a ejecutar acciones específicas, como un clic izquierdo o derecho. Sin embargo, es posible definir que una cantidad determinada de parpadeos consecutivos represente una acción concreta.

Con este propósito, se establece que dos parpadeos consecutivos se interpreten como la acción de realizar un clic izquierdo, mientras que tres parpadeos consecutivos correspondan a la acción de realizar un clic derecho.

Si bien es posible asignar acciones adicionales a diferentes cantidades de parpadeos consecutivos (arrastrar, scroll, etc), debe considerarse que esto podría afectar el patrón asociado a un parpadeo ya establecido. Este aspecto no ha sido abordado en la presente tesis, por lo que se deja abierta la posibilidad de explorarlo en futuros estudios.

2.2.1. Conteo de Picos

Para determinar si un pulso es producto de un parpadeo, es necesario esperar a que sea procesado por el módulo *Mouse_Block.m*. Al finalizar este procesamiento y establecerse el equivalente del pulso, la señal habrá concluido. Aunque se almacena información de los últimos 20 datos del pulso, esto podría no ser suficiente para contabilizar los picos asociados a la señal de parpadeo. Por ello, el conteo de pulsos en la señal debe realizarse

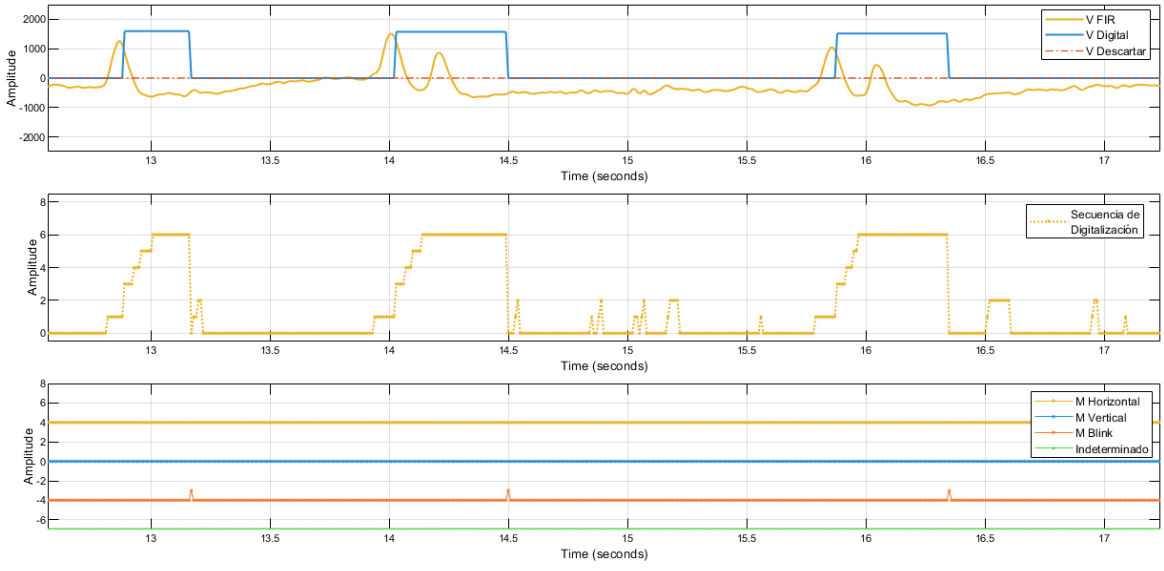
constantemente en el eje vertical, para que el módulo *Mouse_Block.m* use esta información y determine si el pulso corresponde a un parpadeo simple o a una acción del mouse.

Según los datos analizados, puede generarse más de un pulso significativo únicamente durante los parpadeos, no se encontró una forma de anticipar esta situación. Por ello, independientemente de si la señal representa un movimiento ocular ascendente o descendente, el conteo de pulsos de parpadeos debe ejecutarse en todos los casos.

Como primer paso para desarrollar un algoritmo capaz de contar los pulsos en la señal vertical, es necesario definir las condiciones bajo las cuales debe iniciarse el conteo de los pulsos presentes mientras la señal digitalizada este en alto positivo. En la Figura 118 se presenta la señal EOG correspondiente al eje vertical, en la cual se observan un total de cinco parpadeos, siendo los últimos cuatro ejecutados de forma rápida y consecutiva de dos en dos.

Figura 118

Digitalización de los parpadeos de la señal EOG, eje vertical.



Nota: La figura muestra la señal del movimiento ocular en el eje vertical relacionados a los parpadeos, en la gráfica superior se puede observar se presenta un parpadeo único, luego hay 2 pares de parpadeos consecutivos. La grafica del medio muestra los pasos secuenciales que

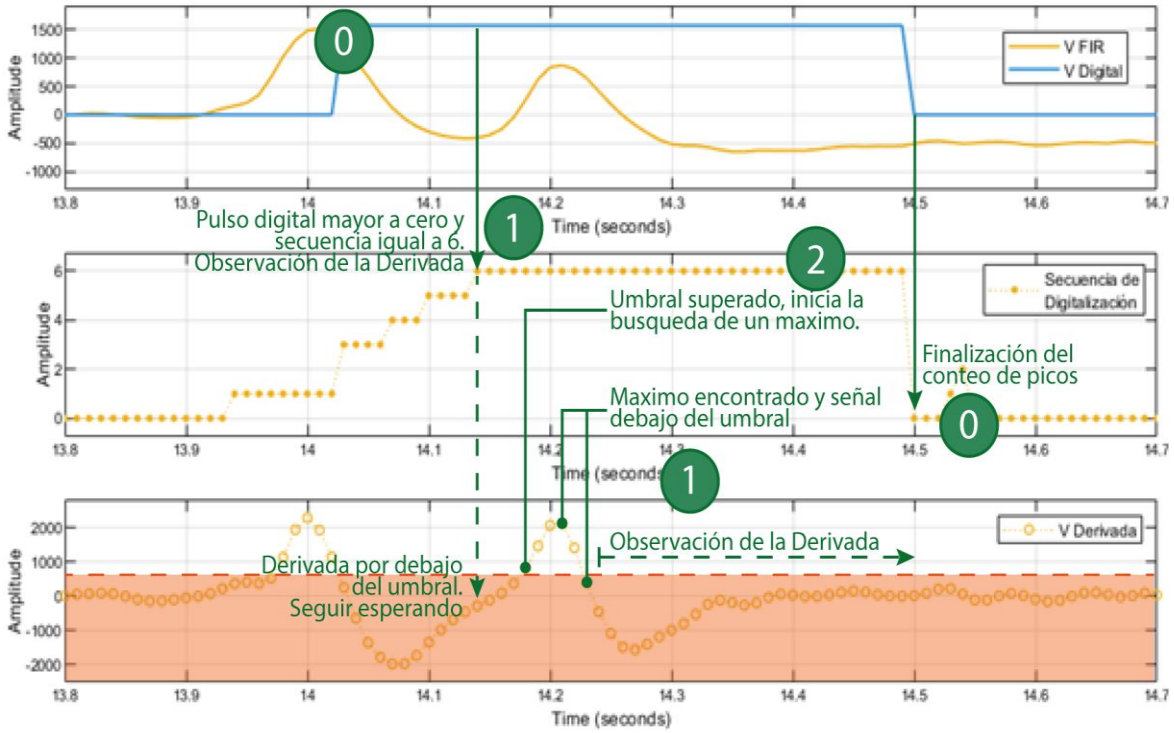
realiza el algoritmo en el proceso de digitalización. La última grafica muestra la interpretación del algoritmo ante la señal del movimiento ocular. Fuente propia.

Se observa que el proceso de digitalización de la señal entra en la secuencia 6 justo cuando el primer pico de la señal alcanza su punto opuesto más bajo. En este punto, para finalizar el pulso digital, únicamente se espera que la convolución de la derivada de la señal descienda por debajo del umbral establecido, como se explicó en la etapa anterior.

Durante este intervalo de espera, el algoritmo de digitalización ignora cualquier otra variación que pueda manifestarse en la señal, incluidos picos que podrían ser significativos. Por consiguiente, es en este período que debe implementarse un algoritmo dedicado al conteo de dichos picos significativos, el cual permita identificar y cuantificar de manera precisa los picos presentes en la señal.

Figura 119

Secuencia de funcionamiento del algoritmo de conteo de picos.

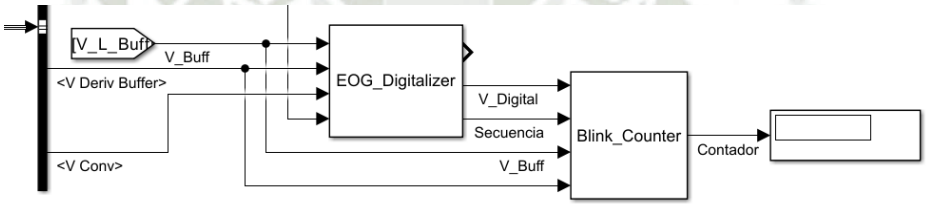


Nota: La figura muestra una secuencia de pasos que sigue el algoritmo para contabilizar la cantidad de parpadeos consecutivos que se realiza, esto permitirá asignar una acción a una cantidad de parpadeos específicos. Fuente propia.

Cuando el algoritmo de conteo comienza a ejecutarse, este debe esperar a que la derivada de la señal supere un umbral predefinido. Esto garantiza que el pico en formación tiene un tamaño significativo y que es positivo. Una vez superado el umbral, se establece una ventana de 4 muestras en la que se buscará un valor máximo. Si el pico continúa creciendo, el máximo se ubicará en la muestra más reciente dentro de la ventana. Sin embargo, si la posición del máximo comienza a desplazarse hacia muestras más antiguas de la ventana, se interpreta que se ha identificado un máximo en la derivada.

Figura 120

Ubicación del bloque del algoritmo de conteo de parpadeos.



Nota: La figura muestra los bloques utilizados en Simulink para el conteo de parpadeos, se muestra todas las entradas y salidas que requiere el algoritmo. Fuente propia.

En este punto, se incrementa en una unidad el contador de picos y se espera a que la señal descienda por debajo del umbral, lo que indica que el pico ha finalizado. El algoritmo entonces reinicia el proceso, esperando nuevamente que la derivada supere el umbral para buscar otro máximo. Este procedimiento se repite hasta que el pulso digital de la señal finaliza y la secuencia de digitalización regresa a 0.

Finalmente, el resultado del conteo de picos se reinicia junto con la señal digital, ya que el módulo *Mouse_Block.m* almacena de manera continua los valores de entrada para su procesamiento.

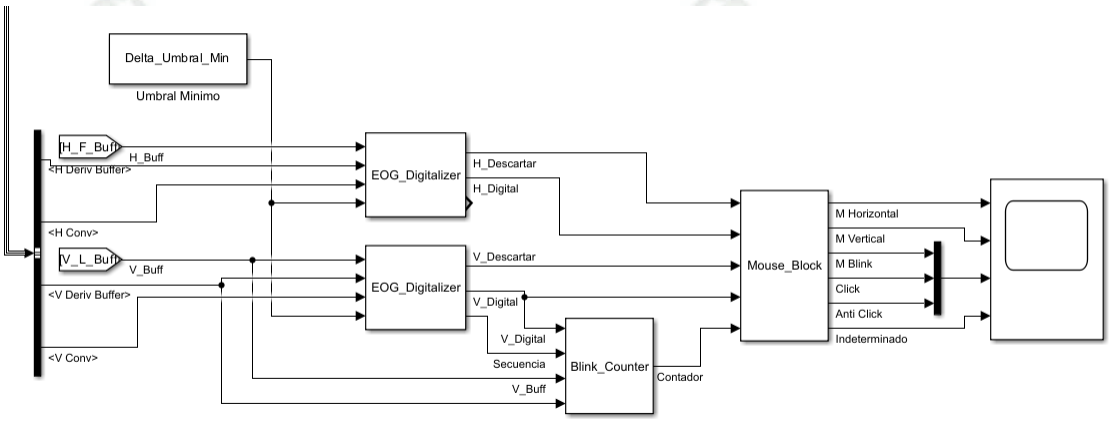
2.2.2. ***Actualización del Bloque*** *Mouse_Block.m*

En los apartados 1.2 y 1.3 de este capítulo, se definieron los algoritmos correspondientes a la digitalización de las señales y la clasificación de patrones. Sin

embargo, en ese momento no se abordó cómo se realizaría la diferenciación entre el clic izquierdo y el clic derecho respecto a otras acciones. Por esta razón, fue imprescindible que la implementación del contador de pulsos se diseñara de manera que no alterara los códigos previamente establecidos.

Figura 121

Etapa de digitalización de señales, conteo de picos y segregación de acciones del mouse EOG.



Nota: La figura muestra el diagrama de bloques de Simulink de las etapas de digitalización de la señal EOG por canal, conteo de parpadeos en el eje vertical y el de control de movimiento del mouse en base a la señal EOG digital, todos interconectados a la vez. Fuente propia.

Bajo esta premisa, el bloque que contiene el algoritmo encargado de realizar el conteo de picos en la señal vertical fue implementado de manera que no modificara la estructura previamente establecida. En lugar de alterar el flujo existente, este bloque actúa como un *"sniffer"* de las señales procesadas a partir de la señal vertical. Gracias a este enfoque, se logró integrar el nuevo bloque sin comprometer el correcto funcionamiento de la estructura original.

Para procesar adecuadamente la información que el nuevo bloque proporciona al bloque *Mouse_Block.m*, es necesario realizar ciertas modificaciones a este último. En el apartado 1.3 de este capítulo se describe que el bloque *Mouse_Block.m* entra en acción

cuando ambas señales EOG digitales regresan a cero, momento en el cual todas las variables son reiniciadas para que el proceso pueda repetirse con nuevas señales.

Dado este funcionamiento, el bloque *Mouse_Block.m* debe almacenar de forma continua la información que recibe. La primera modificación consiste precisamente en aumentar el mecanismo que usa este bloque, para conservar de manera constante los datos proporcionados por el nuevo bloque *Blink_Counter.m* , asegurando que dicha información esté disponible para su posterior procesamiento sin interferir con el reinicio de variables al final de cada ciclo.

Tabla 60
Modificación del archivo "Mouse_Block.m" para considerar el conteo de picos en el parpadeo

Sin Conteo de Picos	Con Conteo de Picos

...
...
%%%%% Pulso Parpadeo %%%%%%%%%%%%%
 	
if V_Height> 200 && V_Duration > 20 && V_Anuladado==0 && Triggers == 2
  if V_dif(1)<=10 && V_dif(2)>10
  Blink=1;
  Triggers=3;
  elseif V_dif(1)>10 && V_dif(2)>10 && H_Height==0
  Blink=1;
  Triggers=3;
  End
 	
end
...
...
	...
	...
	%%%%% Pulso Parpadeo %%%%%%%%%%%%%
	
	if V_Height> 200 && V_Duration > 20 && V_Anuladado==0 && Triggers == 2
	  if V_dif(1)<=10 && V_dif(2)>10
	  Blink=1;
	  Triggers=3;
	  elseif V_dif(1)>10 && V_dif(2)>10 && H_Height==0
	  Blink=1;
	  Triggers=3;
	  end
	
	  if Blink==1
	  if N_Parpadeos==1
	  Blink=1;
	  Click=0;
	  Anti_Click=0;
	  end
	  if N_Parpadeos==2
	  Blink=0;

Sin Conteo de Picos	Con Conteo de Picos

```
        Click=1;
        Anti_Click=0;
    end
    if N_Parpadeos==3
        Blink=0;
        Click=0;
        Anti_Click=1;
    end

  end
end
...
...
```

Nota: Ignorar ausencia de tildes en la tabla, el IDE de programación no reconoce símbolos con tildes.

Cuando llega el momento de reconocer el patrón de los pulsos digitales EOG recibidos y que acaban de finalizar, el bloque *Mouse_Block.m* utiliza la información previamente almacenada en memoria para determinar a qué acción corresponden dichos pulsos. Es en este punto donde se utiliza la información del número de picos detectados en la señal.

Si la acción es interpretada como un parpadeo, el bloque verifica la cantidad de picos contados en el eje vertical. Según esta cantidad, determina si el parpadeo corresponde a una acción de clic izquierdo, clic derecho o si simplemente fue un parpadeo destinado al descanso ocular.

3. TRADUCCIÓN DE CÓDIGO DE MATLAB-SIMULINK A XC16

Para el desarrollo del algoritmo presentado en esta tesis, se optó por capturar las señales filtradas mediante el dsPIC y enviarlas a una computadora, donde fueron almacenadas en archivos de Excel para su posterior procesamiento con MATLAB y Simulink. Aunque estos programas son herramientas extremadamente potentes para el análisis de señales y la simulación de sistemas, hay un inconveniente significativo en su uso:

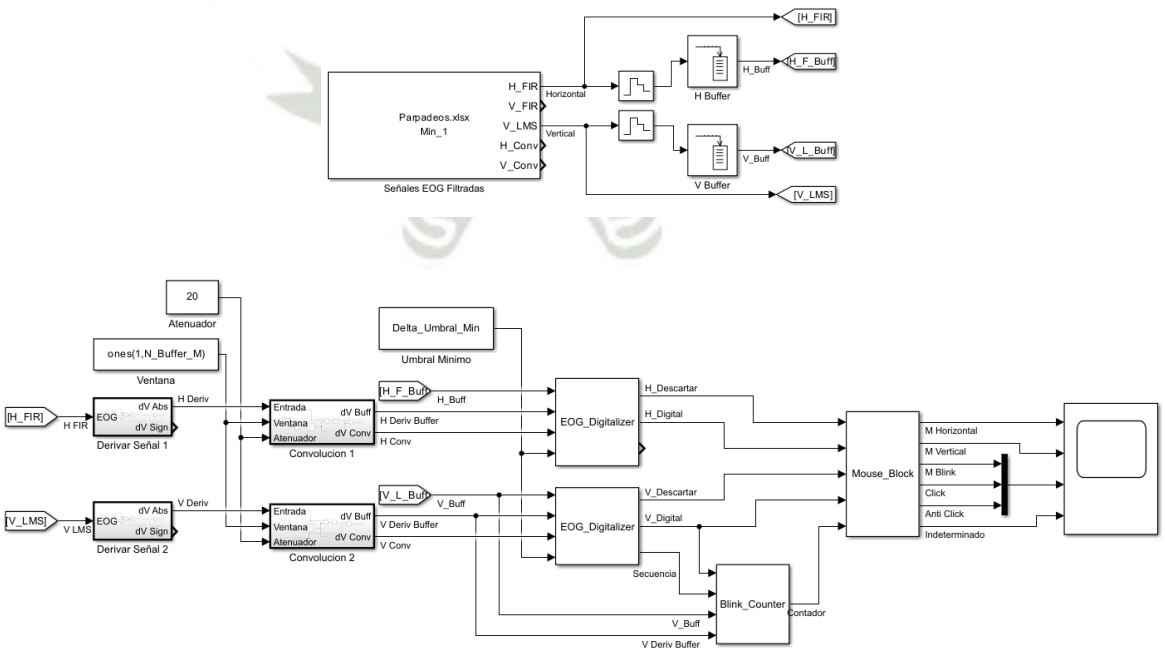
muchas de sus funciones y características son complejas de traducir e implementar en un programa para el lenguaje XC16 para el dsPIC.

Por esta razón, toda la simulación en MATLAB-Simulink se realizó utilizando únicamente funciones que pudieran ser implementadas en el dsPIC con XC16. En el ámbito de la programación, se limitaron las operaciones a condicionales como IF, bloques de desplazamiento de vectores (Buffer Block), manejo de vectores y funciones de búsqueda de máximos y mínimos.

El principal beneficio obtenido con MATLAB fue el aprovechamiento de sus capacidades de análisis de señales mediante el osciloscopio (Scope) de Simulink. Esta herramienta permitió visualizar y evaluar cambios en la señal con una duración del orden de pocos milisegundos, un tipo de análisis que sería extremadamente difícil de realizar en tiempo real directamente en el dsPIC.

Figura 122

Estructura general de bloques para la digitalización de señales y segregación de acciones del mouse EOG.



Nota: La figura muestra el diagrama de bloques de Simulink del algoritmo de procesamiento de señal EOG completo, empezando por la adquisición de señales, preparación de señales y digitalización de la señal EOG, todo interconectado de forma sistemática. Fuente propia.

En la Tabla 61 se presenta un extracto del archivo *EOG_Mouse_Control.h*, correspondiente al proyecto desarrollado en MPLAB X IDE para el compilador XC16 que se usa en el dsPIC33CH128MP506. En este archivo se definen los recursos necesarios para replicar el comportamiento observado durante la simulación en Simulink.

Se declaran dos estructuras principales, “Slope_Control” y “Channel_Comp”, que almacenan en memoria toda la información y cálculos requeridos para el procesamiento de las señales EOG. Además, se especifican todas las funciones necesarias para realizar las mismas operaciones implementadas en la simulación de Simulink.

En el archivo *Main.c*, también desarrollado en MPLAB X IDE, se detalla la disposición y secuencia en la que se utilizan las funciones declaradas en *EOG_Mouse_Control.h*, garantizando un funcionamiento equivalente al logrado en Simulink. Asimismo, se definió una función adicional denominada “Mouse_Move_Order()”, cuya finalidad es activar o desactivar los pines digitales que indican al microcontrolador PIC18F2550 la acción que debe ejecutar el puntero del mouse EOG. De esta manera, se completa el proceso de convertir las señales oculares EOG en una acción concreta de movimiento del puntero del mouse en la computadora.

Tabla 61
Declaraciones del código para igualar el funcionamiento de Simulink

EOG_Mouse_Control.h	
typedef struct {	typedef struct{
int Eje; //1:Horizontal. 2:Vertical.	
	int Triggers; //Secuencia
int Valor_FIR_PB;	
int Valor_FIR_LMS;	int H_Duration;
//	int H_Height;
int Discret_Buffer[20];	int H_Anulado;

EOG_Mouse_Control.h

```
    int Deriv_Buffer[20];                          int H_History[3];
    int Actual_Slope_Sign;
    int32_t Deriv_Conv;                            int V_Duration;
    ////////////////////////////////////////////////// int V_Height;
                                                   int V_Anulado;
    int Triggers; //Secuencia                      int V_History[4];
    int Slope_Sign_Init;
    int Slope_Sign_Final;                          int H_Diferencia[2];
    int Idle;                                      int V_Diferencia[2];
    int Offset;
    int Remontada;                                 int N_Parpadeos;
    int Anular;
                                                   int Mov_Horizontal;
    ////////////////////////////////////////////////// int Mov_Vertical;
    int Polarizacion;
                                                   int Blink;
    int EOG_Base;                                  int Click_Izq;
    int EOG_n_Base;                                int Click_Der;

    int EOG_Peak;                                  int Indeterminado;
    int EOG_n_Peak;
                                                   int Orden_Eje_H;
    int EOG_Fin;                                   int Orden_Eje_V;
    int EOG_n_Fin;                                 int Orden_Click_L;
    ////////////////////////////////////////////////// int Orden_Click_R;
    int Pulso_Dig;
                                                 }Channel_Comp;
    //////////////////////////////////////////////////
    int Blink_Contador;
    int Blink_Secuencia;
} Slope_Control;
//////////
void Historial_Discreto_Update(Slope_Control*,int);
void Delta_Signal(Slope_Control*);
void Convolucion_Derivada(Slope_Control*);

void Signal_Digitalitation(Slope_Control*);
void Blink_Counter(Slope_Control*);
void          Signal_Comparation(Slope_Control*,Slope_Control*,Channel_Comp*);
//Mouse_Block
```

EOG_Mouse_Control.h

```
void Mouse_Move_Order  (Channel_Comp*);

///Funciones Optimizadas con DPS Engine para la operacion de señales
extern int Convolution_MOD (int numElems,int* srcV1,int* srcV2, int32_t* Acumulador);

extern int Vector_Max_n(int Index_Limite, int* Vector, int Vector_Length, int* Indice_Resultado);
extern int Vector_Min_n(int Index_Limite, int* Vector, int Vector_Length, int* Indice_Resultado);
Main.c
//Definicion de las estructuras de control:

Slope_Control Horizontal;
Slope_Control Vertical;
Channel_Comp Comparacion_H_V;
…
…

//Recreacion el modelo de bloques realizado en Simulink:

    Historial_Discreto_Update(&Horizontal,Horizontal.Valor_FIR_PB);
    Delta_Signal(&Horizontal);
    Convolucion_Derivada(&Horizontal);
    Signal_Digitalitation(&Horizontal);

    Historial_Discreto_Update(&Vertical,Vertical.Valor_FIR_LMS);
    Delta_Signal(&Vertical);
    Convolucion_Derivada(&Vertical);
    Signal_Digitalitation(&Vertical);

    Signal_Comparation(&Horizontal,&Vertical,&Comparacion_H_V);
//Mouse_Block
    Mouse_Move_Order(&Comparacion_H_V);
…
…
```

Nota: Ignorar ausencia de tildes en la tabla, el IDE de programación no reconoce símbolos con tildes.

Para traducir el código de MATLAB-Simulink a XC16, se consideraron las siguientes diferencias y adaptaciones:

3.1. Indexación de Vectores

En MATLAB, los vectores de longitud N tienen su primer y último término indexados como 1 y N, respectivamente. En XC16, la indexación comienza en 0, por lo que el primer y último término se ubican en las posiciones 0 y N−1.

3.2. Estructura del Condicional IF:

En XC16, los condicionales están definidos de la siguiente forma: if( ){ }, lo que difiere del formato en MATLAB.

3.3. Cálculo de Mínimos y Máximos:

Para encontrar los valores mínimo y máximo de un vector, se implementó un código en lenguaje ensamblador. Este código permite realizar búsquedas tanto en todo el vector como en una porción específica, replicando el comportamiento de las funciones “min( )” y “max( )” de MATLAB.

3.4. Corrimiento de Vectores hacia la Izquierda:

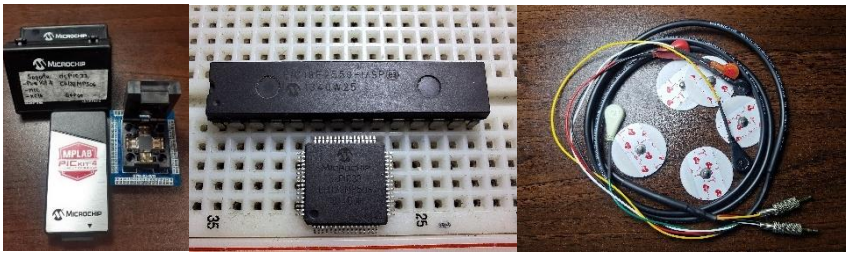
Se utilizó el periférico DMA (Direct Memory Access - Acceso Directo a Memoria) para optimizar el proceso de desplazamiento de vectores. Dado que se procesan múltiples vectores, realizar este corrimiento de manera ordinaria podría generar retrasos significativos en el procesamiento, lo que afectaría el funcionamiento del sistema en el dsPIC.

4. DISEÑO Y FABRICACIÓN DE PROTOTIPOS

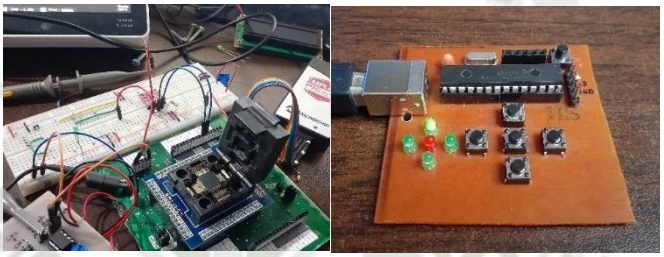
Una vez se ha logrado el funcionamiento independiente de cada etapa, el siguiente paso consiste en integrar estas etapas para que interactúen entre sí de manera coordinada. La interacción entre los dos microcontroladores se complementa con las etapas de regulación de voltaje, aislamiento de ruido y adquisición de señales EOG, por lo que es importante tener cuidado al empezar el proceso. De no prestarse la atención adecuada, podrían ocurrir errores o fallos en el funcionamiento debido a problemas en la interacción entre todas las etapas.

Figura 123

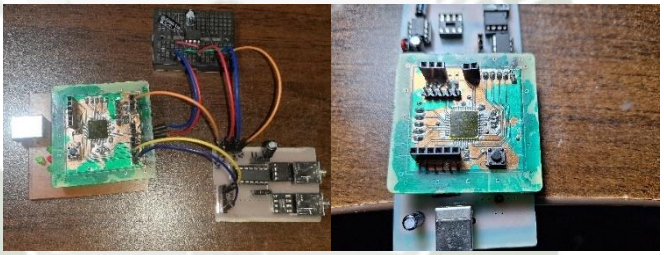
Evolución de las pruebas preliminares de manera independiente



a)



b)



c

Nota: a)Vista de los microcontroladores y del cable de conexión de los electrodos. b)Bancos de prueba independientes del dsPIC33CH y del PIC18F2550. c)Pruebas entre las etapas conectadas parcialmente. Fuente propia.

Cada etapa del prototipo fue probada de forma independiente utilizando bancos de prueba diseñados específicamente para cada fase de evaluación. La configuración del PIC18F2550 como dispositivo de control de un mouse USB fue relativamente rápida, gracias a que Microchip proporciona una librería con un ejemplo funcional de esta aplicación. Sin embargo, la dificultad radicó en adaptar este ejemplo a los requerimientos específicos del prototipo, eliminando partes del código que no eran funcionales para nuestra implementación (originalmente incluidas para garantizar la compatibilidad en diversos modelos de microcontroladores).

Adicionalmente, se identificaron y corrigieron problemas como la carga excesiva de datos en el canal USB conectado, presente en el código de ejemplo, que afectaba la estabilidad de la conexión. Para acercar el rendimiento del sistema al de un mouse comercial estándar, fue necesario realizar un análisis detallado del protocolo USB, lo que permitió optimizar el flujo de datos y adaptar el funcionamiento a los requerimientos del prototipo.

Lograr que el dsPIC33CH funcione correctamente requirió una considerable labor de investigación, principalmente debido a la poca información disponible en su documentación, ya que esta familia de microcontroladores pertenece a una generación reciente. Para configurar el dsPIC33CH como filtro FIR, se llevaron a cabo múltiples pruebas previas, logrando activar y evaluando cada módulo del microcontrolador de manera independiente para asegurar su correcto funcionamiento. Una vez logrado el control individual de cada módulo, se procedió con pruebas de funcionamiento en conjunto entre todos sus módulos, hasta llegar a incluir al motor DSP en su función de filtro digital.

Otro desafío al trabajar con este microcontrolador fue su tamaño reducido: la distancia entre sus pines es de apenas medio milímetro. Esto hizo necesario diseñar un hardware específico para cada prueba, con el fin de asegurar conexiones físicas precisas y confiables, requiriendo tiempo de planificación y fabricación para realizar cada prueba con este microcontrolador.

La prueba individual de la etapa de adquisición de bioseñales resultó relativamente rápida y sencilla, principalmente debido a la amplia disponibilidad de información sobre el tratamiento de señales mioeléctricas (musculares), por ser un tema ampliamente puesto en práctica en el área bio electrónica. Se decidió iniciar las pruebas con este tipo de señales; una vez lograda su adquisición y amplificación adecuada, se procedió con las bioseñales

oculares, comenzando con el eje horizontal (utilizando un electrodo en cada sien y un electrodo de referencia en el centro de la frente).

A pesar de la limitada información disponible sobre la electrooculografía (EOG), se logró capturar satisfactoriamente las señales oculares. Estas señales, por ser significativamente más débiles que las señales mioeléctricas, requirieron una amplificación considerablemente mayor. Sin embargo, esta amplificación adicional presentó un desafío, ya que aumentó también el nivel de ruido, complicando la obtención de una señal limpia. La señal EOG mostró ser altamente susceptible a variaciones en el estado de los electrodos; factores como electrodos levemente secos, una adherencia insuficiente a la piel, una iluminación insuficiente en los ojos o incluso el movimiento excesivo de los cables que conectan los electrodos y el prototipo, resultaban en fallas funcionales de la etapa de adquisición.

Tras implementar un balance adecuado en el tratamiento de la señal y aplicar cuidados necesarios en la manipulación de los electrodos, se finalizó con éxito esta etapa. No obstante, al integrar esta sección con las demás etapas del sistema, se observó que su funcionamiento era notoriamente afectado por factores como caídas de voltaje y diferencias de impedancia entre los componentes, lo cual requirió ajustes y pruebas posteriores adicionales para estabilizar su funcionamiento con el sistema en conjunto.

Una vez logrado el funcionamiento independiente de cada una de las etapas, el siguiente paso fue su integración para evaluar el desempeño del sistema completo. Este proceso implicó interconectar los bancos de prueba de cada etapa con el objetivo de iniciar una serie de pruebas conjuntas. Las pruebas en conjunto permiten observar las interacciones entre cada módulo y verificar su comportamiento dentro del sistema final, identificando y resolviendo posibles interferencias o incompatibilidades entre las etapas. En la Figura

123 c), se presenta una de estas pruebas, donde se analizaron las primeras señales en conjunto. Estos ensayos fueron repetidos y ajustados conforme a las observaciones y datos obtenidos, permitiendo la refinación de cada etapa hasta lograr un funcionamiento óptimo en el sistema integrado.

Uno de los principales problemas al integrar todas las etapas del prototipo en un solo sistema fue la alimentación del sistema. Inicialmente, se había planificado que el prototipo se alimentara de la energía proporcionada por el host USB (computadora); sin embargo, se observó que esta introducía un nivel de ruido significativo al sistema debido a la naturaleza de su fuente de alimentación, afectando principalmente la etapa de adquisición de señal EOG. Este problema no interfería con el rendimiento del PIC18F2550, por lo que pasó desapercibido en la prueba individual de esta etapa, pero se volvió evidente al conectar todo el sistema.

Se evaluaron varias soluciones, siendo la más óptima implementar un aislamiento entre la computadora y el prototipo. Sin embargo, este aislamiento planteaba nuevos desafíos, ya que impedía una transferencia de datos USB adecuada debido a la falta de un voltaje de referencia común entre ambos sistemas. La solución final fue aplicar el aislamiento después de la etapa de comunicación USB entre la computadora y el prototipo, es decir, luego del PIC18F2550. Para mantener la comunicación entre el PIC18F2550 y el resto del prototipo, se decidió simplificar la interacción entre estos a pulsos digitales que se transmitirían de manera aislada mediante optoacopladores, asegurando así un nivel de voltaje independiente en cada lado del sistema.

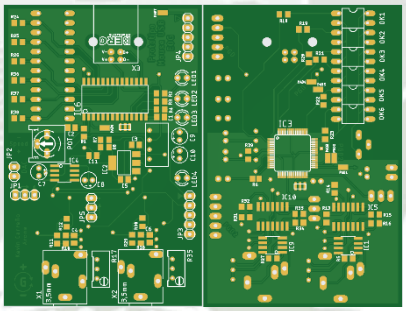
Superado el problema del ruido, surgió un nuevo desafío: la etapa de adquisición de señal EOG requería obligatoriamente trabajar con un nivel de voltaje simétrico (±Vcc). Este nivel también era esencial para mantener un punto de referencia de voltaje compartido entre

el dsPIC y la etapa de adquisición de señal, lo que permitía que el dsPIC leyera correctamente la señal. La solución fue introducir un componente capaz de invertir un voltaje positivo a su correspondiente voltaje negativo, lo cual resolvió este problema de manera rápida y eficaz.

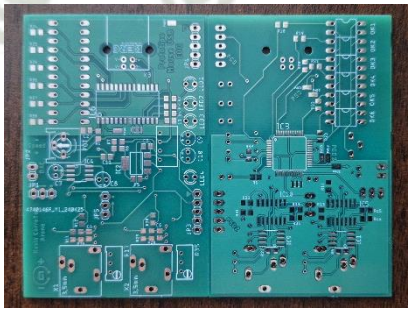
Con todo el hardware del prototipo ya en funcionamiento y la interacción entre cada etapa individual satisfactoriamente verificada, el paso final consistía en implementar el software. Este debía analizar la señal EOG procesada por el motor DSP, interpretando los patrones del movimiento ocular y transformándolos en acciones del puntero del mouse USB. Para ello, se decidió fabricar una tarjeta PCB que constituiría la versión final del prototipo, permitiendo afinar el software a través de pruebas más específicas.

Figura 124

Layout de la primera versión del prototipo.



a)



b)

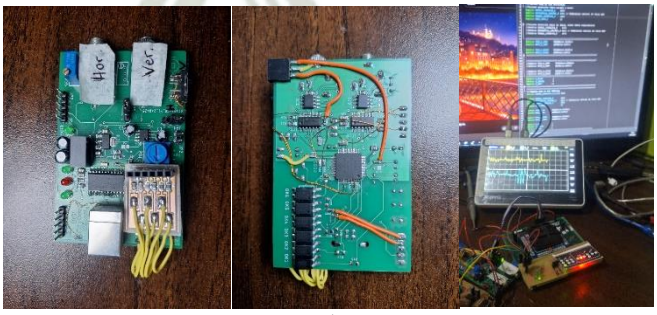
Nota: a)Layout de la PCB del primer prototipo diseñado que se mandó a fabricar con JLCPCB.com
b)PCB fabricada, listo para la soldadura de los componentes. Fuente propia.

Una vez se logró armar y probar la PCB del primer prototipo, se procedió a realizar las pruebas entre todas las etapas en conjunto, utilizando las señales EOG. Debido a que estas señales tienen una duración corta, las variaciones en esta señal ocurren muy rápido, y la velocidad de procesamiento del dsPIC es aún más rápido, se vio que iba a ser complicado crear un algoritmo únicamente guiándonos en la respuesta del mouse que íbamos observado en la pantalla, por tal motivo, se tuvo que hacer un banco de pruebas para la depuración del programa que estábamos diseñando, para este motivo se tuvo que hacer nuevas modificaciones

Una vez ensamblada la PCB del primer prototipo, se procedió a realizar pruebas integradas de todas las etapas utilizando las señales EOG. Dado que estas señales son de corta duración y sus variaciones ocurren rápidamente, además que la velocidad de procesamiento del dsPIC es aún mayor, resultó complejo diseñar un algoritmo guiándose únicamente por la respuesta visual del mouse en pantalla. Por ello, fue necesario implementar un banco de pruebas para depurar el programa en desarrollo, lo cual requirió realizar nuevas modificaciones.

Figura 125

Pruebas iniciales y modificaciones en el primer prototipo funcional



a)

b)

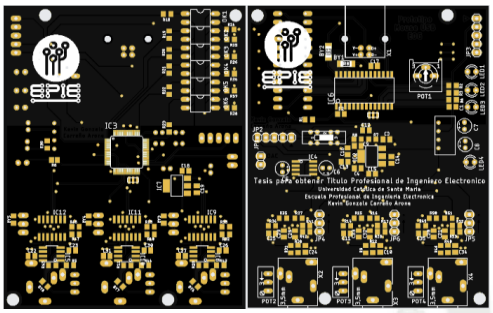
Nota: a) Modificaciones para realizar una depuración del programa para el control de movimiento del mouse a partir de las señales EOG. b) Rediseño de la etapa de adquisición y acondicionamiento de la señal EOG. Fuente propia.

Gracias al uso del banco de pruebas para la depuración del programa, se logró examinar todas las variaciones de la señal, permitiendo el desarrollo de un algoritmo funcional. Sin embargo, surgió un problema en esta etapa: los electrodos introducían alteraciones que amplificaban la señal hasta exceder los valores máximos permitidos por los amplificadores operacionales, causando una saturación que eliminaba la información de la señal. Este fenómeno, detallado en los apartados 3.7.3 del Capítulo 1 y 3.2 del Capítulo 2, evidenció que la etapa de adquisición de señal EOG del primer prototipo no era adecuada para el reconocimiento de patrones, ya que era susceptible a la pérdida de información. Ante esto, fue necesario rediseñar dicha etapa.

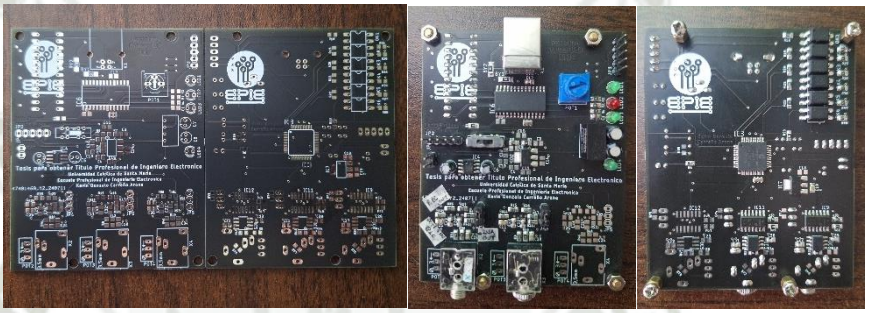
Con un nuevo diseño implementado de manera provisional, se continuaron las pruebas del programa, logrando que esta etapa sea inmune el error asociado a la deriva de voltaje de los electrodos. Posteriormente, se actualiza el diseño del prototipo con las correcciones nuevas, incorporando componentes que faciliten las pruebas del software y la verificación de la señal EOG amplificada. Adicionalmente, se implementa un nuevo canal de adquisición de señales EOG con propósito general, ya que permite tener un pin del dsPIC, libre para ser usado como entrada analógica o digital, salida analógica o para comunicación UART.

Figura 126

Layout de la primera versión del prototipo.



a)



b)

Nota: a)Layout de la PCB del segundo prototipo deseñado que se mandó a fabricar con JLCPCB.com
b)PCB fabricada, y con todos los componentes ensamblados. Fuente propia.

Con el hardware del prototipo terminado, se procede a realizar pruebas del software para el movimiento del mouse. Haciendo uso el hardware, se captura la señal EOG, y se transmite el resultado de la filtración FIR a una computadora, con el fin de usar programas para analizar señales temporales como Matlab/Simulink.

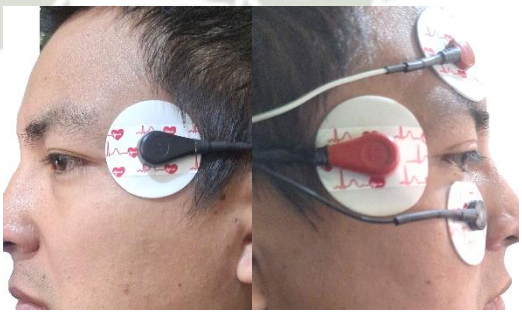
5. RESULTADOS

Como etapa final de esta tesis, es necesario presentar los resultados obtenidos del prototipo. Para ello, es preciso especificar las condiciones en las que se llevaron a cabo las pruebas.

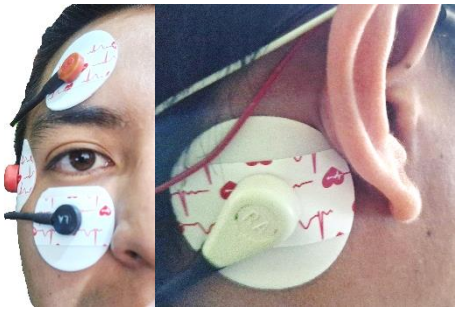
5.1. Conexión de Electrodos

Para las pruebas, se utilizaron cinco electrodos de plata/cloruro de plata (Ag/AgCl) con adhesivo de espuma micro poroso. Además, se aplicó una pequeña cantidad de gel conductor para optimizar el contacto entre los electrodos y la piel. Como se mencionó previamente, en cada eje se colocó un par de electrodos, representando la referencia positiva y negativa. En el eje horizontal, la referencia positiva se ubicó en la sien derecha y la negativa en la sien izquierda. Para el eje vertical, la referencia positiva se colocó sobre el ojo y la negativa debajo de este. Se debe procurar que cada par de electrodos queden alineados en el mismo eje, teniendo como referencia central la ubicación del ojo mirando de frente.

Figura 127
Colocación de los electrodos para adquisición de señales EOG.



a)



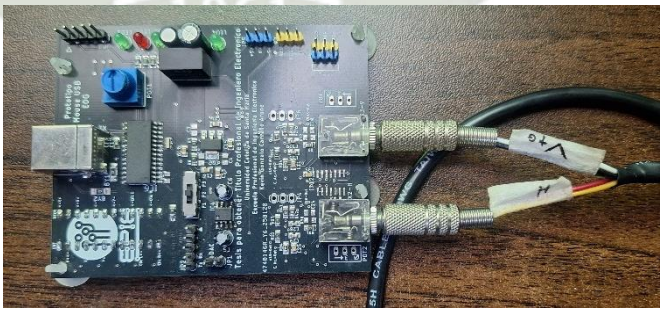
b) *c)*

Nota: La figura muestra la colocación de los electrodos para la fase de pruebas. a)Eje Horizontal
b)Eje Vertical

c)Referencia a tierra. Fuente propia.

Una vez acoplados los broches de los cables a los electrodos, se procede a conectar los cables al prototipo. Este proceso se lleva a cabo con el dispositivo apagado para evitar que el ruido generado durante la conexión sature los circuitos electrónicos. Una vez realizada la conexión, es fundamental acomodar los cables de manera que su peso no ejerza tensión sobre los electrodos, ya que esto podría afectar la calidad del contacto entre el electrodo y la piel.

Figura 128
Conexionado entre el prototipo y los electrodos



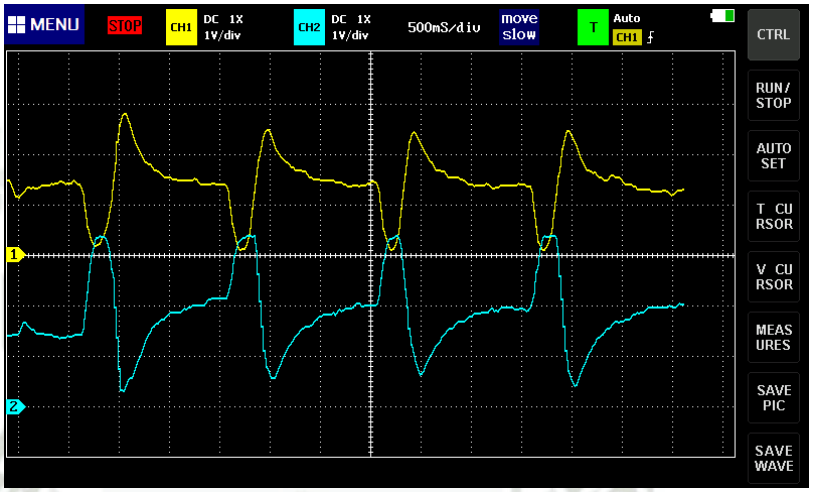
Nota: La figura muestra el conexionado entre el prototipo y el cable que conecta los electrodos. Fuente propia.

En este punto, el prototipo está listo para su uso. Una vez conectado a la alimentación USB, es posible medir simultáneamente las señales EOG verticales y horizontales para verificar que presenten una amplitud adecuada. Esto permite garantizar que los mecanismos

de reconocimiento de patrones, implementados en los algoritmos, puedan activarse correctamente.

Figura 129

Señales EOG con filtrado FIR de las señales Horizontales y Verticales.



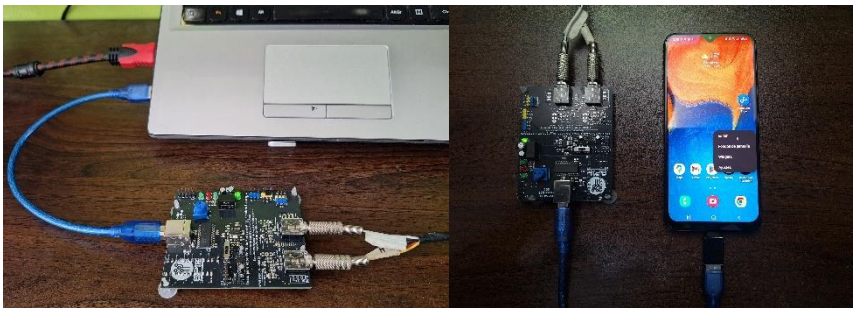
Nota: La figura muestra las señales EOG capturadas por el osciloscopio, representando el movimiento ocular de los canales horizontal (Amarillo) y vertical (Cian). Fuente propia.

5.2. Conexión USB multiplataforma

Gracias a los diversos modos de operación del PIC18F2550, se logró que el prototipo sea funcional en cualquier plataforma sin requerir hardware o software adicional. Esto brinda una mayor flexibilidad en su uso, permitiendo adaptarse a diferentes entornos y a los recursos disponibles, según las necesidades del usuario.

Figura 130

Prototipo conectado a diferentes plataformas.



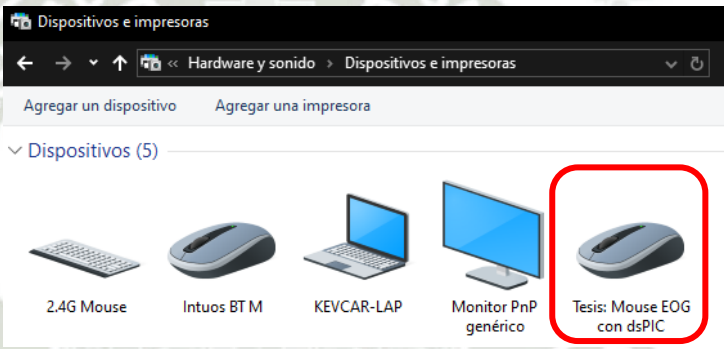
a) *b)*

Nota: a) Computadora. b) Teléfono Android con capacidad OTG. Fuente propia.

Al conectar el prototipo a la computadora mediante un cable USB, el sistema operativo lo reconoce de inmediato e inicia automáticamente el proceso de enumeración, sin necesidad de instalar software adicional. Este procedimiento es igual en otros dispositivos capaces de trabajar con un mouse USB estándar. Como se observa en la Figura 130, el prototipo puede conectarse tanto a una computadora como a un teléfono Android mediante un conector OTG compatible con el dispositivo móvil.

Figura 131

Reconocimiento del dispositivo en diferentes plataformas.



a)

b)

Nota: La figura muestra como el prototipo es reconocido en diferentes plataformas. a) Computadora b) Teléfono Android con capacidad OTG. Fuente propia.

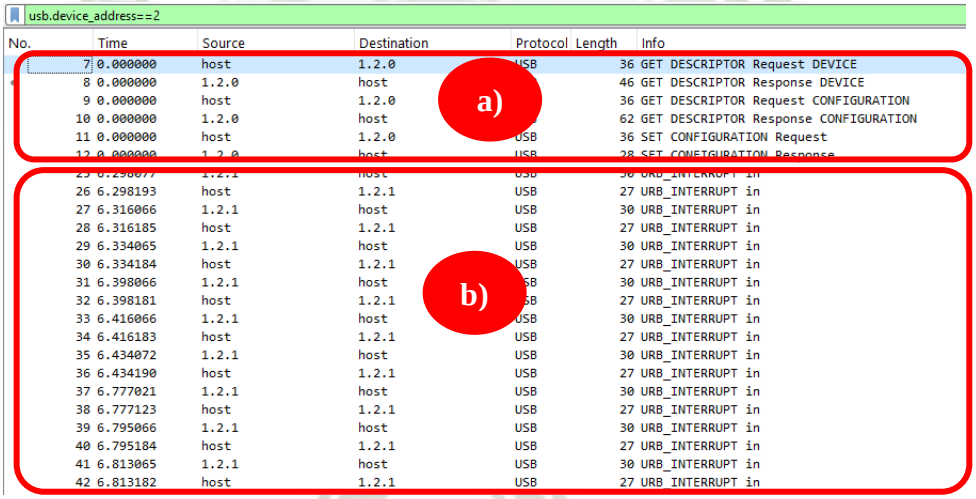
El nombre del dispositivo se asigna durante el envío de los descriptores de configuración. Esto se puede verificar en la sección USB_DESCRIPTOR_STRING de la

tabla *"Archivo usb_descriptors.c"* en el ANEXO H. En una computadora, el nombre asignado al prototipo puede visualizarse en el apartado Dispositivos e impresoras.

Al emplear el software Wireshark junto con el complemento USBP-Cap para analizar el tráfico de paquetes USB entre la computadora y el prototipo, específicamente con el PIC18F2550, se observa que los primeros paquetes intercambiados corresponden a la configuración inicial del dispositivo. Una vez completado el proceso de enumeración, se transmiten los paquetes destinados al control del movimiento del puntero del mouse en la computadora gracias a la orden que genero el dsPIC producto del procesamiento de las señales EOG horizontales y verticales.

Figura 132

Trafico de paquetes de datos entre la computadora y el prototipo.



Nota: La figura muestra el tráfico USB capturado con Wireshark entre el Host y nuestro prototipo. a)Configuración inicial en la conexión (Enumeración).b)Movimiento del puntero del mouse. Fuente propia.

5.3. Movimiento del Puntero del Mouse

Dado que la única respuesta visible del prototipo es el movimiento del puntero del mouse, resulta necesario visualizar información adicional que permita evaluar su efectividad además de mostrar el funcionamiento del mismo. Para ello, se exportarán desde el dsPIC todos los datos requeridos para este propósito. Este proceso se llevará a cabo mediante el

uso del módulo UART del dsPIC, un convertidor UART a USB Serial y el código implementado en ambos extremos de la comunicación, garantizando así la correcta transmisión y recepción de los datos.

Figura 133

Uso de UART para transmisión de datos desde el dsPIC hacia la computadora.

a)

```
Datos[0]= Comprimir_Channel_Comp(&Comparacion_H_V); //Flags de movim.

Datos[1]=(int16_t)Comparacion_H_V.H_Diferencia[0]; //Duracion Diff
Datos[2]=(int16_t)Comparacion_H_V.V_Diferencia[0]; //Duracion Diff
Datos[3]=(int16_t)Comparacion_H_V.H_Diferencia[1]; //Altura Diff
Datos[4]=(int16_t)Comparacion_H_V.V_Diferencia[1]; //Altura Diff

Datos[5]=(int16_t)Comparacion_H_V.H_Duration; //Duracion
Datos[6]=(int16_t)Comparacion_H_V.H_Height;   //Altura
Datos[7]=(int16_t)Comparacion_H_V.V_Duration; //Duracion
Datos[8]=(int16_t)Comparacion_H_V.V_Height;   //Altura

UART_DMA_Send(Datos); //Enviar datos por UART
```

b)

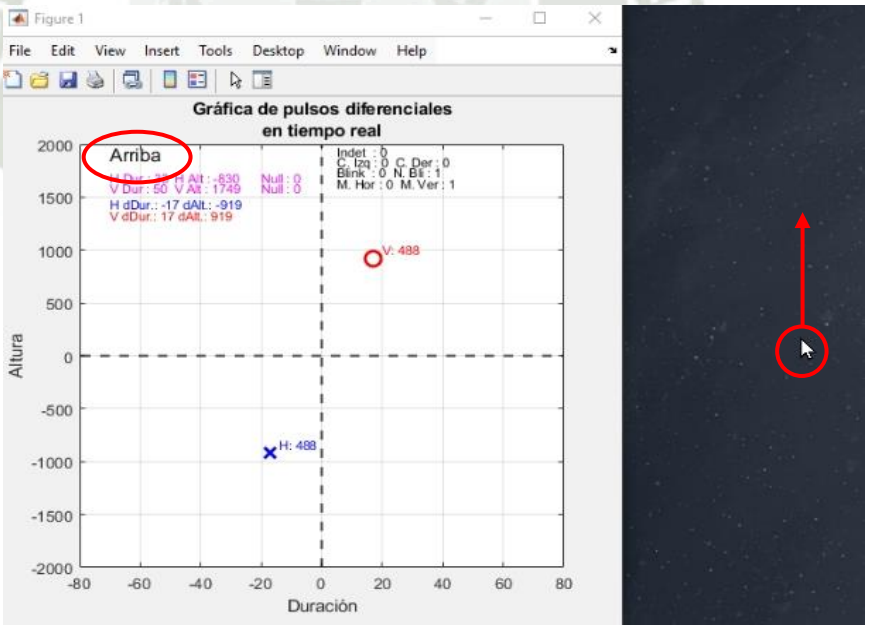
```
% Configuración del puerto serial
serialPort = 'COM9';
baudRate = 115200;
s = serialport(serialPort, baudRate);
s.DataBits = 8;
s.StopBits = 1;
s.ByteOrder = 'big-endian'; % Orden de recepcion

% Esperar a que el puerto esté listo
pause(2);

disp("Esperando datos...");

% Leer datos en tiempo real
while true
    if s.NumBytesAvailable >= 20  % 9 + 1 valores de 16 bits (2 bytes) = 20 bytes

        rawData = typecast(uint16(read(s, 10, "uint16")), "int16");

        fila = fila + 1; % Incrementar el número de fila

        % Verificar si el último valor es el delimitador 0A0D
        if rawData(end) == hex2dec('0A0D')
```

c)

Nota: a) Conexión entre el prototipo y la computadora, mediante un convertidor, b)Empaquetamiento de datos a enviar desde el dsPIC, c)Escucha y recepción de datos desde la computadora. Fuente propia.

A partir de la información recibida desde el dsPIC, se desarrolla una interfaz gráfica que permite visualizar en tiempo real, de manera clara y accesible, las acciones ejecutadas por el prototipo. Como referencia, se emplean las gráficas presentadas en el apartado 1.3 de este capítulo, por lo que los puntos representados en la interfaz seguirán las mismas reglas establecidas en dicho apartado. Además, se incorpora una representación textual de las órdenes enviadas por el dsPIC al módulo USB, junto con información numérica y el estado de los flags de control utilizados el procesamiento de la señal EOG, lo que facilita la depuración de errores en caso de ser necesario.

Figura 134
Interfaz gráfica para evaluar el prototipo en tiempo real.



Nota: La figura muestra la interfaz creada en Matlab para evaluar en tiempo real la información que envía el dsPIC a la computadora al mover el puntero por medio del prototipo. Fuente propia.

La Figura 134 muestra la interfaz gráfica desarrollada en MATLAB, en la cual se visualiza la información recibida desde el dsPIC mediante un gráfico (plot). Esta interfaz

opera de manera independiente al funcionamiento del mouse, por lo que su existencia no influye en el movimiento del cursor. Su propósito principal es facilitar la validación del movimiento del puntero del mouse y como consecuencia, la medición de resultados.

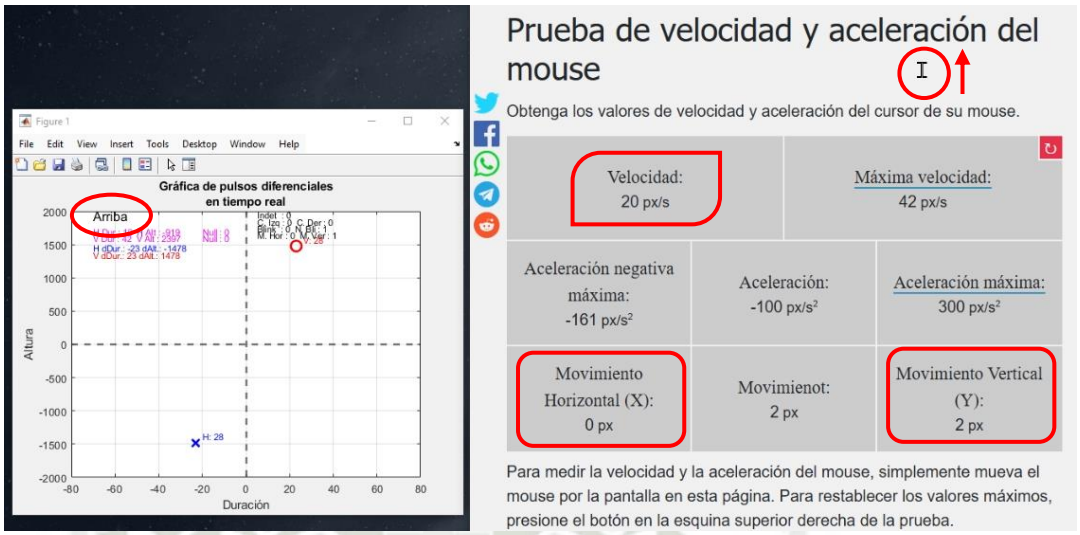
Hasta este punto, se ha logrado demostrar, mediante WireShark y la interfaz gráfica desarrollada, que el puntero del mouse efectivamente se desplaza. Ahora es necesario determinar la velocidad a la que se produce dicho desplazamiento. Para ello, se utilizará una aplicación web, lo que permitirá contrastar la información visualizada en la interfaz gráfica con las mediciones proporcionadas por una fuente externa.

La Figura 135 muestra la interfaz gráfica junto con una aplicación web capaz de medir la velocidad de desplazamiento del mouse. En nuestra interfaz gráfica, se observa que los puntos “O” y “X” están posicionados de manera que pueden clasificarse como un movimiento en dirección “Arriba”, según los criterios definidos en la sección 1.3 de este capítulo. En consecuencia, el dsPIC envía la orden Mover Arriba al módulo USB, lo que se traduce en un desplazamiento vertical positivo del puntero del mouse.

Si comparamos las mediciones realizadas por la aplicación web, se observa que el desplazamiento instantáneo en el eje X es de 0 px, mientras que en el eje Y es de 2 px, lo que confirma que el puntero se está moviendo hacia arriba. Estos resultados permiten validar la información recibida por el dsPIC y constituyen evidencia de que el desplazamiento del mouse se está efectuando correctamente.

Figura 135

Medición de velocidad del puntero del mouse EOG.



Nota: La figura muestra la comparación de resultados de la medición del movimiento del puntero, a la izquierda se muestra la interfaz creada, a la derecha una página web que evalúa el movimiento del mouse. Se puede observar que ambos dan información coherente del movimiento del puntero. Fuente propia, página web: cps-check.com

El valor de 2 px en el parámetro de movimiento vertical no representa la velocidad promedio de desplazamiento, sino el movimiento instantáneo del puntero. Esto es coherente con el funcionamiento del módulo USB de nuestro prototipo, el cual desplaza el puntero en pequeños pasos de 1 a 2 píxeles por vez. La velocidad promedio del movimiento se ajusta modificando el intervalo de tiempo entre el envío de estos pasos, lo que permite variar la velocidad de desplazamiento del puntero del mouse.

El prototipo permite ajustar la velocidad de desplazamiento del puntero desde 1 px/s, un valor extremadamente lento, hasta más de 400 px/s, una velocidad excesiva que dificulta su control. Es importante destacar que, a diferencia de un mouse convencional, el prototipo está programado para mover el puntero a una velocidad constante.

La aplicación web utilizada para medir la velocidad del cursor, indica que se está desplazando a 20 px/s. Aunque esta velocidad es baja en comparación con la de un mouse estándar, se considera moderada para la fase inicial de familiarización con el prototipo. Para

optimizar su manejo, se utilizará un rango de velocidad entre 20 y 40 px/s. Si bien el puntero sigue siendo controlable hasta velocidades de 60 a 100 px/s, se descartan estos valores, ya que una interpretación incorrecta de las señales EOG por parte del prototipo, podría provocar que el puntero sobrepase el punto de destino de manera abrupta, dificultando así la operación del sistema.

5.3.1. Prueba de Desplazamiento

En la sección 1.3 de este capítulo se presentó una evaluación del error del algoritmo. Si bien los resultados obtenidos en dicha evaluación fueron prometedores, estos derivan de una simulación basada en la grabación de señales EOG obtenidas bajo condiciones cuidadosamente controladas. No obstante, en un entorno de uso real, la calidad de la señal puede verse afectada debido a diversos factores ambientales, químicos y eléctricos inherentes a los dispositivos utilizados en la etapa de adquisición de señales EOG. Esto se debe a que dichos dispositivos no han sido diseñados específicamente para el procesamiento de señales bioeléctricas, lo que puede comprometer la precisión del algoritmo en condiciones no controladas.

Para evaluar el desempeño del prototipo, es necesario realizar una prueba que permita obtener resultados repetibles y comparables a lo largo de múltiples mediciones. Para ello, se ha diseñado un "juego" simple en el que el usuario debe mover el puntero del mouse sobre una serie de círculos enumerados, distribuidos en distintas posiciones en la pantalla, en el menor tiempo posible. El tiempo requerido para completar la prueba será registrado, permitiendo así una comparación objetiva de los resultados de múltiples mediciones.

Adicionalmente, se realizará un conteo manual de los movimientos oculares interpretados correctamente por el prototipo y se comparará con la cantidad total de

movimientos realizados. Esto permitirá evaluar la precisión del sistema, determinando la exactitud con la que el prototipo reconoce e interpreta los movimientos del usuario.

Figura 136

"Juego" para medir el tiempo de movimiento del puntero del mouse.

Nota: Para evaluar el funcionamiento del prototipo, se creó un prototipo que mide el tiempo para completar un recorrido con el puntero controlado por señales EOG. Fuente propia.

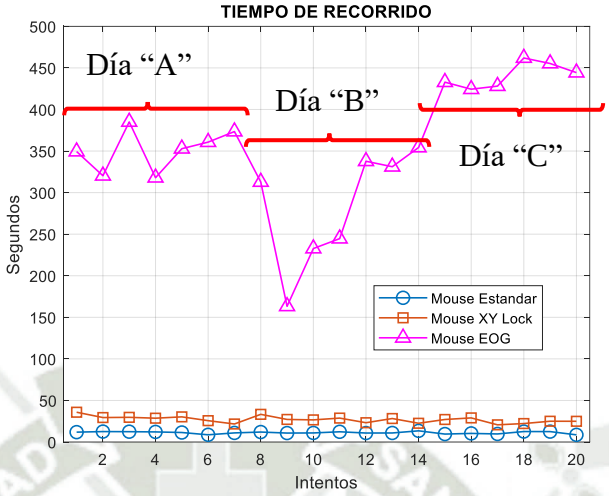
Para establecer un punto de referencia en la velocidad con la que se completa el "juego" diseñado, se comparará el desempeño del mouse EOG con el de un mouse estándar en dos configuraciones: una que permite movimientos en todas las direcciones y otra restringida a desplazamientos ortogonales en los ejes X e Y. Esta comparación permitirá tener una idea del rendimiento relativo del prototipo en relación con dispositivos de control convencionales.

5.3.2. Medición de Resultados

Dado que no se logró identificar completamente la causa de las variaciones abruptas en la señal adquirida por el hardware del prototipo EOG encargado de esto, se realizaron pruebas en días distintos, utilizando electrodos nuevos en cada sesión. Esto con la intención de que las variaciones en las condiciones eléctricas, corporales y ambientales contribuyeran a mejorar la adquisición de la señal.

Figura 137

Comparación de tiempos de finalización de la prueba entre un mouse convencional, un mouse con movimiento ortogonal y el mouse EOG.



Nota: La figura muestra una comparación del tiempo necesario para completar el circuito de prueba, se puede apreciar que en general, el tiempo de respuesta del prototipo es inferior al de un mouse estándar, sin embargo esto puede llegar a mejorarse. Fuente propia.

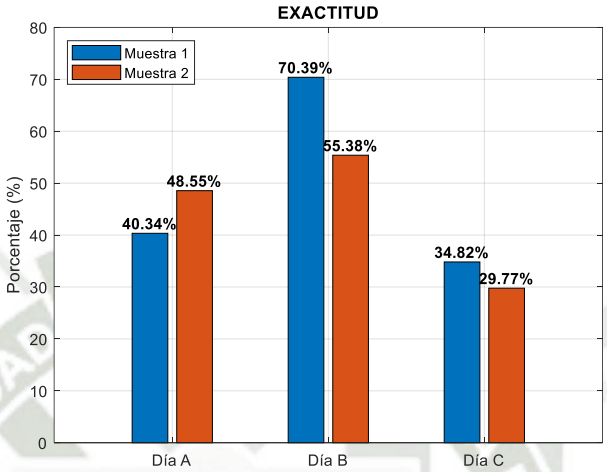
En la Figura 137, se observa que el uso de un mouse convencional, incluso cuando está restringido a movimientos ortogonales en los ejes X e Y, permite completar la prueba en menor tiempo en comparación con el prototipo de mouse EOG. El tiempo elevado registrado con el prototipo se debe principalmente a la falta de precisión en el control del puntero, lo que genera movimientos erráticos y obliga al usuario a realizar múltiples intentos para corregir su trayectoria. Esta imprecisión está vinculada a las limitaciones del hardware empleado en la adquisición de la señal EOG. Sin embargo, si se optimizara esta etapa con componentes más precisos, el tiempo necesario para ejecutar cada acción con el mouse EOG se reduciría significativamente.

En la Figura 138, se presenta el porcentaje de aciertos del prototipo de mouse EOG, medido en seis pruebas. Se observa que la precisión es inferior al 50 % en la mayoría de los casos, lo que coincide con los tiempos elevados registrados en la Figura 137. La baja exactitud implica que, en muchas ocasiones, las órdenes enviadas no corresponden al movimiento esperado, lo que obliga a realizar intentos adicionales para recuperar el control

del puntero. A pesar de estas dificultades, el desempeño obtenido no descarta la posibilidad de mejorar y optimizar el sistema para aplicaciones futuras.

Figura 138

Porcentaje de aciertos en la detección de movimientos del mouse EOG.



Nota: La figura muestra una comparación de la efectividad del prototipo en 2 pruebas por día durante 3 días aleatorios. Fuente propia.

Los resultados obtenidos en la etapa experimental han permitido analizar el desempeño del prototipo de mouse EOG en comparación con dispositivos convencionales. Si bien el sistema ha demostrado ser funcional, su precisión y velocidad aún presentan limitaciones significativas debido a la variabilidad de la señal EOG y a los desafíos en su procesamiento en tiempo real. Estas dificultades afectan la estabilidad del puntero y la capacidad del usuario para realizar movimientos precisos de manera eficiente.

No obstante, el desarrollo de este prototipo representa un avance en la integración de señales biológicas en interfaces de control, sentando las bases para futuras mejoras en la adquisición y procesamiento de la señal EOG. Con optimización en la captura de las señales EOG y refinamiento en los algoritmos de interpretación, es posible incrementar la precisión y reducir la latencia del sistema, lo que facilitaría su aplicación en entornos reales y ampliaría su potencial como una alternativa de accesibilidad para personas con movilidad reducida.

CONCLUSIONES

Primera: Se demostró que el control del puntero de una computadora mediante bioseñales es viable y funcional. Sin embargo, el uso prolongado de este sistema podría generar fatiga ocular o condiciones médicas relacionadas con los músculos de los ojos, por lo que es necesario evaluar sus beneficios frente a posibles efectos adversos.

Segunda: Se observó que los movimientos involuntarios de los ojos pueden ser interpretados erróneamente por el sistema, generando acciones no deseadas del puntero. Para mitigar este problema, es necesario establecer límites dinámicos en la interpretación de señales, ya que la intensidad de la señal EOG varía en función de factores ambientales, químicos y eléctricos, como la iluminación, la degradación del gel conductor de los electrodos y el ruido electromagnético emitido por los equipos eléctricos cercanos.

Tercera: Se comprobó que el protocolo USB en su configuración HID permite la comunicación fluida entre el microcontrolador y la computadora, logrando que el sistema sea reconocido como un dispositivo de entrada estándar sin necesidad de controladores adicionales en múltiples plataformas.

Cuarta: La configuración del PIC18F2550 en modo HID fue exitosa, permitiendo su uso como interfaz humano-computadora, adicionalmente, se identificó defectos en el código fuente de la libre MLA de Microchip, el cual hacia deficiente la comunicación USB.

Quinto: el dsPIC33CH128MP506 demostró ser eficaz para el procesamiento digital de las bioseñales, proporcionando la capacidad de aplicar filtros digitales de un orden mayor a 200 y caracterizar la señal en tiempo real, sin embargo, el estudio realizado evidencia la necesidad de seguir explorando métodos para mejorar el procesamiento de las bioseñales de manera adaptativa, dado el potencial que aún tiene el microcontrolador mencionado como la posibilidad de usar la transformada rápida de Fourier FFT, manejo de matrices, control

PID, cálculo de números complejos, multiplicación y división de números de 32bits, etc., a nivel de hardware.

Sexto : La investigación sobre la generación de bioseñales confirmó que la Electrooculografía (EOG) y la Electromiografía (EMG) pueden ser utilizadas para el control del puntero. Sin embargo, se determinó que la EOG es más susceptible a interferencias externas, lo que puede afectar la estabilidad del sistema.

Séptimo: Se identifico una degradación recurrente en la calidad de la señal EOG causado por el desplazamiento de voltaje de los electrodos en la etapa del EOG, debido principalmente al deterioro del gel conductor del electrodo y a los cambios en el área de contacto con la piel. Si bien existen componentes electrónicos diseñados para la captura de señales biomédicas (ECG, EEG, EMG), se requiere evaluar su desempeño en Electrooculografía, ya que este último trabaja con niveles de voltaje menores.

Octava: El acondicionamiento de bioseñales mediante etapas de filtrado y amplificación permitió obtener señales con una relación señal/ruido adecuada para su procesamiento. A pesar del tratamiento que se dio a las señales EOG, se observó que es necesario aplicar más técnicas de acondicionamiento se señales, las cuales permitan mejorar la estabilidad de las señales EOG, adaptándose a diferentes usuarios y ambientes de manera automática.

Novena : Se logró la integración de todas las etapas del sistema en un solo dispositivo funcional, alineando factores como la adquisición, procesamiento y comunicación de bioseñales para el cumplimiento del objetivo principal de este estudio. A pesar de ello, es necesario realizar pruebas con una población más amplia para evaluar su usabilidad y mejorar la ergonomía del sistema.

Decima: Se utilizo Matlab y Simulink como herramientas principales para la realización del algoritmo de procesamiento de las señales biológicas, permitiendo observar detallada y repetitivamente los cambios de las señales biológicas en una escala temporal de milisegundos, consiguiendo depurar errores en la programación del algoritmo.

RECOMENDACIONES

Primera: Dado que los electrodos comerciales con autoadhesivo pueden causar irritación en la piel alrededor de los ojos, se recomienda utilizar electrodos sin adhesivo, fijados mediante una estructura externa como una vincha ajustable o un soporte impreso en 3D. Además, se sugiere el uso de gel conductivo especializado para electrocardiografía, lo que podría mejorar la calidad de la señal y reducir la incomodidad del usuario.

Segunda: Aunque el dsPIC de 16 bits con formato de coma fija permite obtener un filtrado FIR de buena calidad, se podría lograr una mayor precisión utilizando microcontroladores con soporte de coma flotante en hardware, como los ARM Cortex-M4/M7 o el Microchip SAM4E. Sin embargo, estos procesadores implican una mayor complejidad en la implementación y programación, por lo que su elección dependerá de la necesidad de precisión en el procesamiento de señales que se desea alcanza.

Tercero: Si bien el amplificador operacional de instrumentación INA121, empleado en este prototipo para la etapa de adquisición de señales EOG, cumplió su función de manera aceptable, este dispositivo no está diseñado específicamente para aplicaciones biomédicas. Por ello, se recomienda el uso de amplificadores desarrollados exclusivamente para el procesamiento de bioseñales, como el AD8232, con el propósito de eliminar o minimizar el ruido y las interferencias que no pudieron ser atenuadas en la etapa de amplificación de la señal EOG con el INA121.

Cuarto: Se recomienda seguir investigando algoritmos de procesamiento más avanzados para mejorar la fiabilidad del sistema. En particular, la implementación de inteligencia artificial o métodos adaptativos, con los cuales lograr una adaptación automática del sistema a las características individuales de cada usuario, optimizando la precisión y reduciendo los falsos positivos generados por movimientos involuntarios de los ojos.

REFERENCIAS

BasuMallick, , C. (10 de enero de 2023). *What Is USB (Universal Serial Bus)? Meaning, Types, and Importance*. Obtenido de https://www-spiceworks-com.translate.goog/tech/tech-general/articles/universal-serial-bus/?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es&_x_tr_pto=tc

Agencia EFE. (07 de Febrero de 2011). El raton controlado por los ojos que aspira a acercar a internet a personas con discapacidad. Obtenido de https://www.youtube.com/watch?v=aaHw9vHNl5w&ab_channel=AGENCIAEFE

Alan V. Oppenheim, R. W. (2009). *Tratamiento de señales en tiempo discreto* (3ra. Edicion ed.). Pearson.

Alanko, H. I. (1984). Clinical Electro-Oculography. En D. o. Forsius), *Ergophthalmology Symposium* (págs. 139-148).

Angulo, J. (6 de mayo de 2025). *Instrumentacion Biomedica*. Obtenido de https://es.scribd.com/document/221805050/ELECTRODOS-BIOPOTENCIALES

Aparicio, M. (2007). *Sistemas Lineales Invariantes en el Tiempo (SLTI).* España: Sistemas Lineales y convolución. Obtenido de https://www.academia.edu/10779402/Sistemas_Lineales_Invariantes_en_el_Tiempo_SLTI

C., B., A., B., C., H., & V., P. (2022). EOG-Based Human-Computer Interface: 2000-2020 Review. *Sensors*.

Diario El Peruano. (2018). Recuperado el 20 de 04 de 2023, de https://busquedas.elperuano.pe

FunFactory. (2023). *YouTube*. Obtenido de https://www.youtube.com/watch?v=RdPUh15UTZo&list=PLDO7aLOu7FANWJZ3z1oSfHQfQ9QwLkVMr&index=12&ab_channel=FunFactory

García, D. A. (s.f.). *oftalmologia-online.es*. Obtenido de https://www.oftalmologia-online.es/anatom%C3%ADa-del-globo-ocular/

Garrity, J. (2022). *Estructura y función de los ojos.* España: Mayo Clinic College of Medicine and Science. Obtenido de https://www.msdmanuals.com/es/hogar/trastornos-oft%C3%A1lmicos/biolog%C3%ADa-de-los-ojos/estructura-y-funci%C3%B3n-de-los-ojos?ruleredirectid=758

Gonzales Cruz, F. (2019). *Propuesta de vehiculo autonomo para discapacitados en la region Piura (Tesis de Pregrado).* Universidad de Piura.

GSM Association. (2020). *Principios para impulsar la inclusion digital de las personas con discapacidad.*

Hsu, H. P. (2013). *Señales y sistemas.* Mc Graw Hill.

INEI. (2022). *Instituto Nacional de Estadistica e Informatica*. Obtenido de https://m.inei.gob.pe/prensa/noticias/en-el-peru-1-millon-575-mil-personas-presentan-alg/

ITU. (Agosto de 2012). *Accesibilidad de los telefonos y servicios movilies para personas con discapacidad.* G3ict.

John G. Proakis, D. G. (2007). *Tratamiento digital de señales.*

Katz, J. (2022). *Impacto del COVID-19 en la digitalización de América Latina.* Lima: CEPAL. Obtenido de https://www.cepal.org/es/publicaciones/48486-impacto-covid-19-la-digitalizacion-america-latina

Lusby, F. W. (11 de Agosto de 2023). *MedLine Plus*. Obtenido de medlineplus.gov/spanish/ency/anatomyvideos/000010.htm

Madrigal, A. C. (06 de mayo de 2014). *The Atlantic*. Obtenido de https://www.theatlantic.com/technology/archive/2014/05/computer-mice-still-a-thing/361741/

Manolakis, D., & Ingle, V. (2011). *Applied Digital Signal Processing.* Cambridge.

Mauri, C. (2018). *Enable Viacam*. Obtenido de https://eviacam.crea-si.com/index_es.php

Microchip. (2004). *dsPIC Language Tools Libraries.* Obtenido de http://ww1.microchip.com/downloads/en/devicedoc/51456b.pdf

Microchip. (2009). *PIC18F2455/2550/4455/4550 Data Sheet.* Obtenido de https://ww1.microchip.com/downloads/en/devicedoc/39632c.pdf

Microchip. (2018). *dsPIC33CH128MP508 Family.* Obtenido de http://ww1.microchip.com/downloads/en/DeviceDoc/dsPIC33CH128MP508-Family-Datasheet-70005319b.pdf

Microchip. (2020). *MPLAB XC16 C Compiler User's Guide.* Obtenido de https://ww1.microchip.com/downloads/en/DeviceDoc/MPLAB%20XC16%20C%20Compiler%20Users%20Guide%20DS50002071H.pdf

Mitchell, R. (25 de Junio de 2018). *All About Circuits*. Obtenido de https://www.allaboutcircuits.com/news/microchip-introduces-its-new-line-of-microcontrollers-dspic33ch-family/

Murphy, R. (2010). *USB 101: An Introduction to Universal Serial Bus 2.0* (Vol. Rev. H). Infineon.com - Cypress.com. Obtenido de cypress.com: https://www.infineon.com/dgdl/Infineon-AN57294_USB_101_An_Introduction_to_Universal_Serial_Bus_2.0-ApplicationNotes-v09_00-EN.pdf?fileId=8ac78c8c7cdc391c017d072d8e8e5256

Neuman, M. R. (2009). *Medical Instrumentation: Biopontential Electrodes.* (J. G. Webster, Ed.) John Wiley&Sons.

Nunez, P. L., & Srinivasan, R. (2006). *Electric Fields of the Brain: The Neurophysics of EEG.* Oxford Univerity Press.

ONU. (2006). *Convencion sobre los derechos de las personas con discapacidad.*

Ophtal, B. J. (1966). Clinical Electro-Oculography. *Ophtalmological Synopses*, 438-439.

Pantoja, J., & Montilla , J. (2007). *Diseño e implementacion de una tarjeta de adquisicion y control para puerto USB 2.0 utilizando un FPGA.* Popoyan: Universidad del Cauca. Obtenido de http://repositorio.unicauca.edu.co:8080/bitstream/handle/123456789/2187/Dise%C3%B1o%20e%20implementaci%C3%B3n%20de%20una%20tarjeta%20de%20adquisici%C3%B3n%20y%20control%20para%20puerto%20USB%202.0.pdf?sequence=1&isAllowed=y

Reda, R., Tantawi, M., shedeed, H., & Tolba, M. F. (2019). Analyzing Electrooculography (EOG) for Eye Movement Detection. En *The International Coference on Advance Machine Learning Technologies and Applications* (págs. 179-189). Springer.

Sciencedirect. (2012). *Filtro de respuesta de impulso finito.* Mexico: Ciencias de la Computación. Obtenido de https://www.sciencedirect.com/topics/computer-science/finite-impulse-response-filter

Thomas , E. (2002). *Estructura del paquete.* España: Sciencedirect.com. Obtenido de https://www.sciencedirect.com/topics/computer-science/packet-structure

Thomas, L. (27 de marzo de 2019). *News Medical*. Obtenido de https://www.news-medical.net/health/What-is-Bells-Phenomenon.aspx

UNR-Rosario. (6 de mayo de 2024). *Teorema del Muestreo Muestreo en el dominio Frecuencial*. Obtenido de https://www.fceia.unr.edu.ar/tesys/html/Muestreo_Analogico.pdf

USB-IF. (Abril del 2000). *Universal Serial Bus Specification.* Compaq, HP, Intel, Lucent, Microsoft, NEC, Philips.

Villamil, A. E. (2004). *Construcción de un prototipo de sistema de adquisición de la señal Electro Oculográfica (EOG).* Bogota: Universidad de los Andes.

Ware, O. C. (s.f.). *Filtros Digitales*. Obtenido de ocw.uv.es/ingenieria-y-arquitectura/filtros-digitales/temario/

Webster, J. G. (2009). *Medical Instrumentation: Amplifiers and signal processing.* (J. G. Webster, Ed.) John Wiley&Sons.

Zegarra, G. S. (2013). *Diseño e implementacion de un prototipo a esacala de una silla de ruedas controlada por el iris del ojo, para personas discapacitadas, utilizando procesamiento de imagenes y tecnologia USB (Tesis de pregrado).* Universidad Catolica Santa Maria.

ANEXOS

ANEXO A Tabla de clases y subclases dispositivo USB

Se muestra las clases y subclases de los dispositivos USB según el protocolo establecido por USB.org

Descriptor de Dispositivo / Descriptor de Interface			
Tabla de Clases		**Tabla de Subclases**	
Código de Clase	**Descripción de la Clase**	**Código de Sub Clase**	**Descripción de la Sub Clase**
0x00	Dispositivo definido por el fabricante	-	-
0x01	Dispositivo de Audio	0x00	Subclase no definida
		0x01	Subclase de Control de Audio
		0x02	Subclase de Streaming de Audio
		0x03	Subclase de Procesador de Audio MIDI
0x02	Dispositivo de Comunicaciones y CDC Control	0x00	Subclase no especificada
		0x01	Subclase de Control de Línea Directa (Direct Line Control Model)
		0x02	Subclase de Control de Línea de Teléfono (Abstract Control Model)
		0x03	Subclase de Control Telefónico (Telephone Control Model)
		0x04	Subclase de Multicanal (Multi-Channel Control Model)
		0x05	Subclase CAPI (CAPI Control Model)
		0x06	Subclase Ethernet Networking (Ethernet Emulation Model)
		0x07	Subclase ATM Networking (ATM Networking Control Model)
		0xFF	Subclase definida por el vendedor
0x03	Dispositivo de Interfaz Humana (HID)	0x00	Sin subclase específica (o No Subclass)
0x05	Dispositivo de Física	-	-
0x06	Cámara de Imagen	-	-
0x07	Impresora		
0x08		0x01	Subclase de Control SCSI (RBC)

Descriptor de Dispositivo / Descriptor de Interface			
Tabla de Clases		**Tabla de Subclases**	
Código de Clase	**Descripción de la Clase**	**Código de Sub Clase**	**Descripción de la Sub Clase**
	Dispositivo de Almacenamiento Masivo	0x02	Subclase de Almacenamiento ATA (ATAPI)
		0x03	Subclase de QIC-157
		0x04	Subclase de UFI
		0x05	Subclase de Control SCSI (SFF-8070i)
		0x06	Subclase de Control SCSI (Transparente)
		0xFF	Subclase de Almacenamiento Masivo definida por el vendedor
0x09	Hub USB	-	-
0x0A	Dispositivo de CDC-Data	0x00	Subclase no especificada (Usada para transferencias de datos sin control específico)
		0xFF	Subclase definida por el vendedor
0x0B	Tarjeta Inteligente	-	-
0x0D	Dispositivo de Contenido Seguro	-	-
0x0E	Dispositivo de Video	-	-
0x0F	Dispositivo de Salud Personal	-	-
0x10	Interfaz de Dispositivos de Audio/Video (AV)	-	-
0xDC	Dispositivo de Diagnóstico	-	-
0xE0	Dispositivo de Red Inalámbrica	-	-
0xEF	Dispositivo de Interfaz Miscelánea	-	-
0xFE	Aplicación Específica	-	-
0xFF	Dispositivo definido por el vendedor	-	-

ANEXO B Detalles del proceso de enumeración del mouse Logitech M90

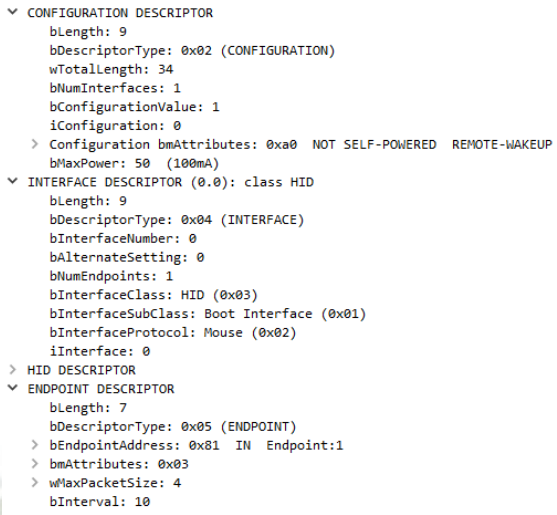
Utilizando el programa Wireshark junto con el complemento USBPcap, es posible capturar el tráfico de paquetes USB para su análisis detallado. En la **Figura 75** se muestra el tráfico de datos capturados durante el análisis de un mouse comercial. El primer frame, que corresponde al proceso de enumeración del mouse, aparece en el paquete capturado número 137. Este frame es enviado desde el host (la computadora) hacia el dispositivo, solicitándole el descriptor general. La respuesta a esta solicitud se encuentra en el frame 138, el cual se presenta en la figura.

```
> Frame 138: 46 bytes on wire (368 bits), 46 bytes captured (368 bits) on interface \\.\USBPcap1, id 0
> USB URB
v DEVICE DESCRIPTOR
    bLength: 18
    bDescriptorType: 0x01 (DEVICE)
    bcdUSB: 0x0200
    bDeviceClass: Device (0x00)
    bDeviceSubClass: 0
    bDeviceProtocol: 0 (Use class code info from Interface Descriptors)
    bMaxPacketSize0: 8
    idVendor: Logitech, Inc. (0x046d)
    idProduct: Mouse (0xc077)
    bcdDevice: 0x7200
    iManufacturer: 1
    iProduct: 2
    iSerialNumber: 0
    bNumConfigurations: 1
```

En este frame de respuesta se incluye información general del dispositivo, como la versión de USB utilizada, la clase y subclase del dispositivo, el tamaño de los paquetes de datos, el nombre del fabricante, el código del producto, entre otros datos relevantes para establecer la comunicación entre el host y el dispositivo.

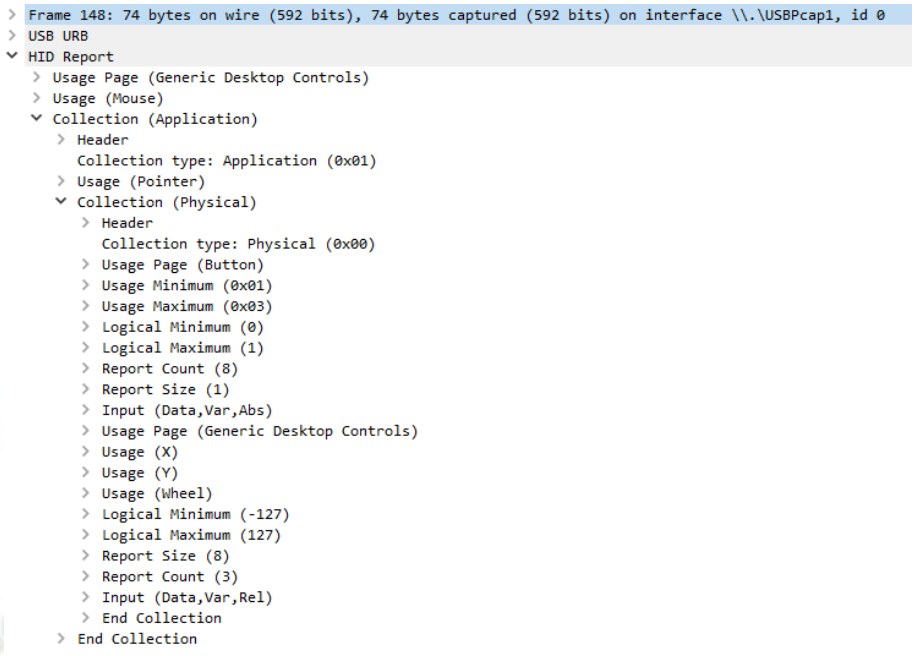
Una vez que el host ha obtenido los datos generales del dispositivo conectado, procede a solicitar información detallada sobre su configuración. Esta información se proporciona en el frame número 142, que se ilustra en la figura inferior. En la respuesta, el dispositivo incluye los descriptores de configuración, interfaz y End Point.

```
> Frame 142: 62 bytes on wire (496 bits), 62 bytes captured (496 bits) on interface \\.\USBPcap1, id 0
> USB URB
> CONFIGURATION DESCRIPTOR
> INTERFACE DESCRIPTOR (0.0): class HID
> HID DESCRIPTOR
> ENDPOINT DESCRIPTOR
```



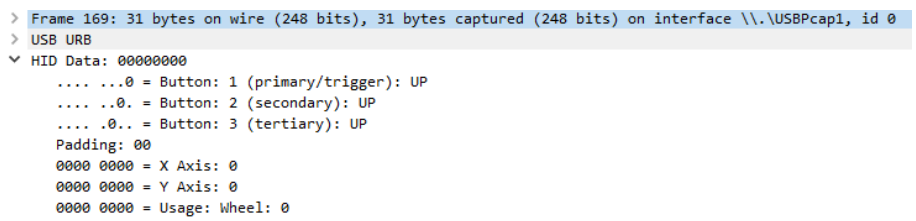
En este frame, el dispositivo informa al host sobre sus requerimientos eléctricos, el tipo de interfaz utilizada, así como la dirección y configuración de su End Point. Es en el descriptor de interfaz donde se especifica al host la naturaleza exacta del dispositivo: se indica que la clase de la interfaz es HID, que empleará el protocolo de un mouse y el número de End Points necesarios para la comunicación.

En la etapa final de configuración, dado que el dispositivo ha informado al host que utilizará una interfaz HID, el host solicita el envío del descriptor de reporte HID. Este descriptor define el formato para la transmisión de datos entre el End Point y el host.



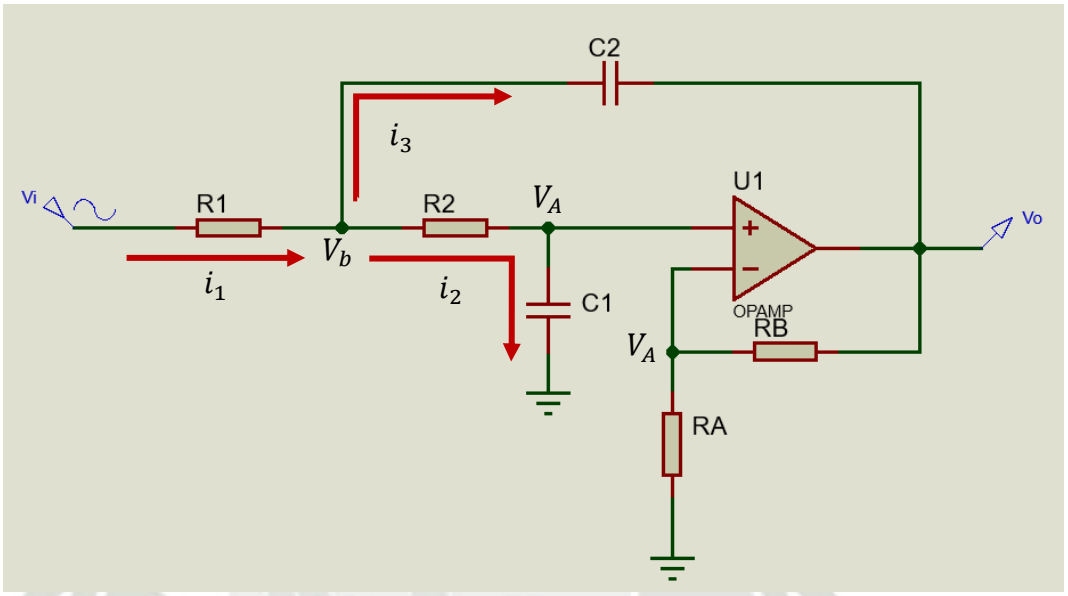
En la figura, se puede observar el contenido del reporte HID. Este indica que el dispositivo posee tres botones que tomarán valores binarios de 1 y 0, para los cuales reserva 8 espacios de un bit cada uno. Para los ejes X, Y y la rueda de desplazamiento, con valores que oscilan entre -127 y 127, se destinan tres espacios de 8 bits, uno para cada uno de estos datos.

Cuando el mouse está correctamente configurado y en funcionamiento, envía datos HID al Host, transmitiendo la información que el dispositivo almacena en su endpoint. Esta transmisión se realiza utilizando el formato especificado en el reporte HID, lo cual permite una comunicación estandarizada y eficiente entre el mouse y el Host. Este proceso se ilustra en la figura inferior, donde se muestran los datos enviados, reflejando los valores actual de los botones y los movimientos registrados en los ejes X, Y y la rueda de desplazamiento.



ANEXO C Diseño de Filtro Analógico Pasa Bajos Sallen Key Pasa Bajos

Para el modelado y diseño de un filtro Pasa Bajos Sallen Key, debemos de partir de la segunda ley de Kirchoff, el cual indica que la suma de corrientes que entran y salen de un nodo es igual a cero.



Determinamos el valor de V_A por un divisor de tensión:

$$V_A = \frac{R_a}{R_a + R_b} V_o$$

La corriente que pasa por la resistencia R2, es la misma que pasa por el condensador C1, por lo que igualamos estos valores:

$$i_2 = \frac{V_b - V_A}{R_2} = \frac{V_b}{R_2 + \frac{1}{S.C_1}}$$

$$\frac{V_b - V_A}{R_2} = \frac{V_b}{R_2 + \frac{1}{S.C_1}}$$

$$V_b = V_o \left(\frac{\boldsymbol{R_a}}{\boldsymbol{R_a + R_b}}\right)(C_1.R_2.S + 1)$$

$$V_b = V_o \boldsymbol{G}(C_1.R_2.S + 1) \cdots (1)$$

Analizamos las corrientes que entran y salen del nodo V_b:

$$i_1 = i_2 + i_3$$

$$\frac{V_{in} - V_b}{R_1} = \frac{V_b}{R_2 + \frac{1}{S.C_1}} + \frac{V_b - V_o}{\frac{1}{S.C_2}}$$

$$\frac{V_{in}}{R_1} + \frac{V_o}{\frac{1}{S.C_2}} = \frac{V_b}{R_2 + \frac{1}{S.C_1}} + \frac{V_b}{\frac{1}{S.C_2}} + \frac{V_b}{R_1}$$

$$\frac{V_{in}}{R_1} + \frac{V_o}{\frac{1}{S.C_2}} = V_b \left(\frac{(\boldsymbol{C_1.C_2.R_1.R_2}).\boldsymbol{S^2} + (\boldsymbol{C_1.R_1} + \boldsymbol{C_1.R_2} + \boldsymbol{C_2.R_1}).\boldsymbol{S} + \boldsymbol{1}}{(\boldsymbol{R_1} - \boldsymbol{R_2.C_2})\boldsymbol{S} + \boldsymbol{R_1}} \right) \cdots (2)$$

Reemplazamos (1) en (2):

$$\frac{V_{in}}{R_1} = V_o.G.(C_1.R_2.S + 1).\boldsymbol{A} - \frac{V_o}{\frac{1}{S.C_2}}$$

$$\frac{V_o}{V_{in}} = \frac{1}{R_1.(G.(C_1.R_2.S + 1).A - S.C_2)} \cdots (3)$$

Operamos, simplificamos y damos forma un sistema de segundo orden, a la ecuación (3):

$$\frac{V_o}{V_i}(s) = \left(\boldsymbol{1} + \frac{\boldsymbol{R_b}}{\boldsymbol{R_a}}\right).\frac{\left(\frac{\boldsymbol{1}}{\boldsymbol{C_1.C_2.R_1.R_2}}\right)}{S^2 + \left(\frac{\boldsymbol{1}}{\boldsymbol{C_2.R_2}} + \frac{\boldsymbol{1}}{\boldsymbol{C_2.R_1}} - \frac{\boldsymbol{R_b}}{\boldsymbol{C_1.R_2.R_a}}\right).S + \left(\frac{\boldsymbol{1}}{\boldsymbol{C_1.C_2.R_1.R_2}}\right)} \cdots (4)$$

La ecuación obtenida (4) es la de un sistema de segundo orden, por lo que procedemos a comparar los términos de estos para calcular la frecuencia de natural del filtro y la ganancia que tendrá:

$$\frac{V_o}{V_i}(s) = \boldsymbol{A}.\frac{\boldsymbol{\omega_n^2}}{S^2 + \boldsymbol{2\sigma}.S + \boldsymbol{\omega_n^2}} \cdots (5)$$

Si deseamos hacer un filtro solo, eliminamos la ganancia de la función de transferencia con $R_b = 0$ y $R_a = inf$, esta acción hace que el filtro siempre sea estable, ya que se elimina el termino negativo que hace que aparezcan polos inestables, además, podemos modelar el filtro para que sus componentes sean los mismos, $C_1 = C_2$ y $R_1 = R_2$, entonces la función de transferencia quedaría de la siguiente forma:

$$H(S) = \frac{V_o}{V_i}(s) = \frac{\left(\frac{\mathbf{1}}{\boldsymbol{C^2.R^2}}\right)}{S^2 + \left(\frac{\mathbf{2}}{\boldsymbol{C.R}}\right).S + \left(\frac{\mathbf{1}}{\boldsymbol{C^2.R^2}}\right)} \cdots (6)$$

Para calcular exactamente la frecuencia de corte en 50Hz, debemos de pasar el sistema al plano frecuencial e igualar la magnitud de la expresión a -3dB ($1/\sqrt{2}$):

$$\left|\frac{V_o}{V_i}\right|(j\omega) = \frac{\left(\frac{1}{C^2.R^2}\right)}{(j\omega)^2 + \left(\frac{2}{C.R}\right).(j\omega) + \left(\frac{1}{C^2.R^2}\right)} = \frac{1}{\sqrt{2}}$$

$$\left|\frac{V_o}{V_i}\right|(j\omega) = \frac{\left(\frac{1}{C^2.R^2}\right)}{\sqrt{\left(\frac{1}{C^2.R^2} - \omega^2\right)^2 + \left(\frac{2.\omega}{C.R}\right)^2}} = \frac{1}{\sqrt{2}}$$

Asignamos una frecuencia de corte de 50Hz y el valor del condensador en 4.7uF, entonces, solo queda despejar el valor de la resistencia:

$$\frac{\left(\frac{1}{4.7uF^2.R^2}\right)}{\sqrt{\left(\frac{1}{4.7uF^2.R^2} - (2\pi 50)^2\right)^2 + \left(\frac{2.2\pi 50}{4.7uF.R}\right)^2}} = \frac{1}{\sqrt{2}}$$

$$R = 435.877475\Omega$$

La ganancia del filtro se estableció para que sea de 1, es decir, el filtro no tendrá parte de amplificación:

$$R_b = 0\Omega$$

$$R_a = \infty\Omega$$

Con estos valores en el filtro Sallen Key, podemos ver el comportamiento frecuencial del filtro y comprobar los cálculos realizados:

$$H(S) = \frac{V_o}{V_{in}} = \frac{957.2k}{s^2 + 1957\ s + 957.2k} \cdots (7)$$

ANEXO D Coeficientes de filtro FIR para comparación

Frecuencia de muestreo (f_s)	2000 Hz
Frecuencia de corte -6dB (f_c)	100 Hz
Tipo	Pasa Bajos
Ventana	Hamming
Orden	84
Nº de coeficientes	85

n	h(n)	n	h(n)	n	h(n)	n	h(n)
0	0.000357	22	0.000000	44	0.093227	66	0.005530
1	0.000195	23	-0.003156	45	0.084994	67	0.005158
2	0.000000	24	-0.006688	46	0.074262	68	0.004338
3	-0.000231	25	-0.010251	47	0.061743	69	0.003252
4	-0.000495	26	-0.013420	48	0.048239	70	0.002075
5	-0.000779	27	-0.015721	49	0.034580	71	0.000954
6	-0.001058	28	-0.016679	50	0.021555	72	0.000000
7	-0.001290	29	-0.015857	51	0.009849	73	-0.000722
8	-0.001426	30	-0.012913	52	0.000000	74	-0.001188
9	-0.001410	31	-0.007640	53	-0.007640	75	-0.001410
10	-0.001188	32	0.000000	54	-0.012913	76	-0.001426
11	-0.000722	33	0.009849	55	-0.015857	77	-0.001290
12	0.000000	34	0.021555	56	-0.016679	78	-0.001058
13	0.000954	35	0.034580	57	-0.015721	79	-0.000779
14	0.002075	36	0.048239	58	-0.013420	80	-0.000495
15	0.003252	37	0.061743	59	-0.010251	81	-0.000231
16	0.004338	38	0.074262	60	-0.006688	82	0.000000
17	0.005158	39	0.084994	61	-0.003156	83	0.000195
18	0.005530	40	0.093227	62	0.000000	84	0.000357
19	0.005287	41	0.098404	63	0.002534		
20	0.004307	42	0.100170	64	0.004307		
21	0.002534	43	0.098404	65	0.005287		

ANEXO E Comparación de respuesta frecuencia de filtros digitales y analógicos.

Para tener una referencia más consistente sobre el comportamiento de los filtros analógicos y digitales (FIR, IIR), realizaremos una comparación de la respuesta de los 3 tipos de filtros vistos en los Capítulos 1 y 2, adicionalmente, diseñaremos un filtro IIR con la herramienta de Matlab, "filterDesigner", para obtener los coeficientes de la respuesta al impulso. Teniendo un total de 4 filtros pasa bajos.

Tipo de filtro	Orden	Magnitud de corte	Frec. de corte	Muestreo
Sallen Key	2	-3 dB	100 Hz	- Analógico
IIR Bilineal (Sallen Key)	2	-3 dB		2 kHz
FIR	84	-6 dB		
IIR Butterworth	10	-6 dB		

Las funciones de transferencia del filtro analógico Sallen Key y del IIR bilineal, se analizan en el Capítulo 1, en este apartado solo se hará la referencia.

$$H_{Sallen}(s) = \frac{\left(\frac{1}{C.R}\right)^2}{s^2 + 2\left(\frac{1}{C.R}\right).s + \left(\frac{1}{C.R}\right)^2} \;;\; H_{Bilineal}(z) = \frac{b_0 + b_1 z^{-1} + b_2 z^{-2}}{1 + a_1 z^{-1} + a_2 z^{-2}}$$

	H(s)		H(z)				
fc (-3dB) (Hz)	**C (µF)**	**R (Ω)**	**b0**	**b1**	**b2**	**a1**	**a2**
100	4,7	217,9387	0,03849	0,0770	0,0385	-1,2153	0,3692

La función de transferencia de un filtro FIR, también se describe en el Capítulo 1, esta sigue la función:

$$H_{FIR}(z) = \sum_{k=0}^{M} b_k z^{-k}$$

Los coeficientes de este filtro FIR, son los que se muestra en el ANEXO D.

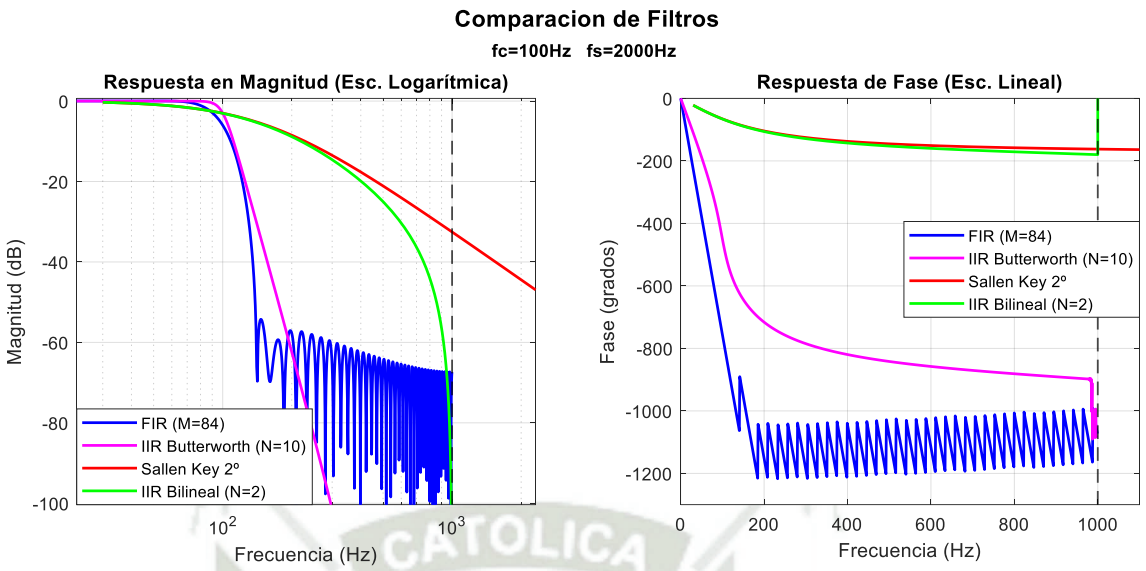
La función de transferencia de un filtro IIR Butterworth no fue vista en esta tesis, debido a que no es el objeto principal del estudio, sin embargo, la herramienta "filterDesigner" nos facilita los coeficientes de la función de transferencia en su forma de Secciones en cascada, esta herramienta exporta los coeficientes en un formato llamado Secciones de Segundo Orden "Second-Order Sections" (SOS) y ganancia de escala (G). La función de transferencia y los coeficientes se muestran a continuación.

$$H_{Butter}(z) = \prod_{k=1}^{N/2} \left(\frac{b_0 + b_1 z^{-1} + b_2 z^{-2}}{a_0 + a_1 z^{-1} + a_2 z^{-2}} \right)_k \cdot G_k$$

	Coeficientes de Sección k de filtro IIR Butterworth de orden 10 (N=10)						**Gk**
Sección *k*	b_0	b_1	b_2	a_0	a_1	a_2	
Sección 1	1	2	1	1	-1.8144	0.9078	0.0233
Sección 2	1	2	1	1	-1.6681	0.7539	0.0215
Sección 3	1	2	1	1	-1.561	0.6414	0.0201
Sección 4	1	2	1	1	-1.4915	0.5682	0.0192
Sección 5	1	2	1	1	-1.4573	0.5323	0.0187

Se debe resaltar que los Coeficientes de Sección (SOS), no son lo coeficientes con los que se multiplica a la entrada de señal para obtener la salida. Para calcular los coeficientes finales, se debe de llevar la función de transferencia a su forma directa y poner toda la función de transferencia en términos algebraicos, y despejar la salida, no se mostrará el procedimiento ya que el fin de este apartado es comparar las respuestas frecuenciales de los filtros.

La comparación de la respuesta frecuencial de estos 4 filtros, se muestra en la Figura inferior.



Si bien el filtro analógico no está limitado por la frecuencia de Nyquist, y en teoría puede atenuar por siempre en todo el espectro frecuencial, tiene la desventaja de ser lento, implementación física, y no ser lineal. El filtro IIR bilineal es una representación digital idéntica al del analógico, pero está limitado por la frecuencia de Nyquist.

El filtro FIR tiene una respuesta de fase lineal, lo cual es muy importante para conserva la forma de las señales, sin embargo para alcanzar una selectividad alta en la respuesta de magnitud, requiere de muchos coeficientes, esto trae el problema de necesitar bastante carga computacional, además, como consecuencia de la multiplicación temporal punto a punto entre la ventana (w[n]) y la repuesta frecuencial ideal deseada ($h_d[n]$) del filtro; frecuencial mente, la convolución de estas dos funciones, dejan una respuesta con múltiples lóbulos laterales.

La respuesta de magnitud del filtro IIR Butterworth, es mucho más rápido que todos, teniendo pocos coeficientes, por lo que la carga computacional es mucho menor, otra ventaja es que, no cuenta con lóbulos laterales. a pesar de esto, no cuenta con una respuesta de fase línea.

La teoría de Fourier establece que no es posible tener una respuesta de magnitud con una transición infinitamente abrupta (altamente selectiva) y sin lóbulos laterales. Reducir los lóbulos laterales generalmente implica suavizar la transición de la respuesta en frecuencia, lo cual reduce la selectividad. Para lograr una fase lineal, un filtro FIR puede ser diseñado con coeficientes simétricos, pero esto no elimina los lóbulos laterales. Además, si se busca minimizar los lóbulos laterales, se compromete la linealidad de fase y la selectividad.

Por tal motivo, no existe un filtro que sea altamente selectivo, fase línea (o sin retraso) y sin lóbulos laterales. La selección del filtro, depende más de la aplicación en la que se le vaya a utilizar.

ANEXO F Coeficientes de filtro FIR para EOG en fractional Q15 Sin Corrección de Ganancia

Se exhibe la primera mitad de los coeficientes correspondientes al filtro FIR que será implementado en el prototipo final. Para obtener la segunda mitad de los coeficientes, es necesario realizar una operación de reflexión en torno al coeficiente número 125, que actúa como pivote. A continuación, se detallan las características específicas de esta configuración:

Frecuencia de muestreo (f_s)	5000 Hz
Frecuencia de corte -6dB (f_c)	10 Hz
Tipo	Pasa Bajos
Ventana	Hamming
Orden	250
Nº de coeficientes	251
Corrección de Ganancia	No

Nº	**Coma Flotante**	**Coef. Coma Fija Q15**	**Nº**	**Coma Flotante**	**Coef. Coma Fija Q15**	**Nº**	**Coma Flotante**	**Coef. Coma Fija Q15**
0	0.00020372	0x0007	42	0.0010382	0x0023	84	0.0029709	0x0062
1	0.00020572	0x0007	43	0.0010768	0x0024	85	0.0030151	0x0063
2	0.00020847	0x0007	44	0.0011161	0x0025	86	0.0030587	0x0065
3	0.00021199	0x0007	45	0.0011561	0x0026	87	0.0031016	0x0066
4	0.00021629	0x0008	46	0.0011968	0x0028	88	0.0031439	0x0068
5	0.00022139	0x0008	47	0.0012382	0x0029	89	0.0031856	0x0069
6	0.00022731	0x0008	48	0.0012802	0x002a	90	0.0032264	0x006a
7	0.00023405	0x0008	49	0.0013228	0x002c	91	0.0032666	0x006c
8	0.00024163	0x0008	50	0.0013661	0x002d	92	0.0033059	0x006d
9	0.00025007	0x0009	51	0.0014098	0x002f	93	0.0033445	0x006e
10	0.00025937	0x0009	52	0.0014541	0x0030	94	0.0033821	0x006f
11	0.00026955	0x0009	53	0.001499	0x0032	95	0.003419	0x0071
12	0.00028062	0x000a	54	0.0015443	0x0033	96	0.0034549	0x0072
13	0.00029258	0x000a	55	0.00159	0x0035	97	0.0034898	0x0073
14	0.00030544	0x000b	56	0.0016362	0x0036	98	0.0035238	0x0074
15	0.00031922	0x000b	57	0.0016828	0x0038	99	0.0035569	0x0075
16	0.00033391	0x000b	58	0.0017297	0x0039	100	0.0035889	0x0076

Nº	Coma Flotante	Coef. Coma Fija Q15	Nº	Coma Flotante	Coef. Coma Fija Q15	Nº	Coma Flotante	Coef. Coma Fija Q15
17	0.00034953	0x000c	59	0.001777	0x003b	101	0.0036198	0x0077
18	0.00036608	0x000c	60	0.0018246	0x003c	102	0.0036497	0x0078
19	0.00038356	0x000d	61	0.0018725	0x003e	103	0.0036785	0x0079
20	0.00040197	0x000e	62	0.0019206	0x003f	104	0.0037062	0x007a
21	0.00042133	0x000e	63	0.0019689	0x0041	105	0.0037328	0x007b
22	0.00044162	0x000f	64	0.0020173	0x0043	106	0.0037582	0x007c
23	0.00046286	0x0010	65	0.002066	0x0044	107	0.0037825	0x007c
24	0.00048503	0x0010	66	0.0021147	0x0046	108	0.0038055	0x007d
25	0.00050814	0x0011	67	0.0021635	0x0047	109	0.0038273	0x007e
26	0.00053219	0x0012	68	0.0022124	0x0049	110	0.0038479	0x007f
27	0.00055716	0x0013	69	0.0022612	0x004b	111	0.0038672	0x007f
28	0.00058307	0x0014	70	0.0023101	0x004c	112	0.0038853	0x0080
29	0.00060989	0x0014	71	0.0023588	0x004e	113	0.0039021	0x0080
30	0.00063763	0x0015	72	0.0024075	0x004f	114	0.0039176	0x0081
31	0.00066628	0x0016	73	0.002456	0x0051	115	0.0039318	0x0081
32	0.00069582	0x0017	74	0.0025044	0x0053	116	0.0039447	0x0082
33	0.00072626	0x0018	75	0.0025526	0x0054	117	0.0039563	0x0082
34	0.00075757	0x0019	76	0.0026005	0x0056	118	0.0039665	0x0082
35	0.00078975	0x001a	77	0.0026481	0x0057	119	0.0039754	0x0083
36	0.00082279	0x001b	78	0.0026955	0x0059	120	0.0039829	0x0083
37	0.00085667	0x001d	79	0.0027424	0x005a	121	0.003989	0x0083
38	0.00089138	0x001e	80	0.0027891	0x005c	122	0.0039938	0x0083
39	0.0009269	0x001f	81	0.0028352	0x005d	123	0.0039973	0x0083
40	0.00096322	0x0020	82	0.002881	0x005f	124	0.0039993	0x0084
41	0.0010003	0x0021	83	0.0029262	0x0060	125	0.004	0x0084

ANEXO G Coeficientes de filtro FIR para EOG en fractional Q15 Con Corrección de Ganancia

Se exhibe la primera mitad de los coeficientes correspondientes al filtro FIR que será implementado en el prototipo final. Para obtener la segunda mitad de los coeficientes, es necesario realizar una operación de reflexión en torno al coeficiente número 125, que actúa como pivote. A continuación, se detallan las características específicas de esta configuración:

Frecuencia de muestreo (f_s)	5000 Hz
Frecuencia de corte -6dB (f_c)	10 Hz
Tipo	Pasa Bajos
Ventana	Hamming
Orden	250
Nº de coeficientes	251
Corrección de Ganancia	Si

Nº	**Coma Flotante**	**Coef. Coma Fija Q15**	**Nº**	**Coma Flotante**	**Coef. Coma Fija Q15**	**Nº**	**Coma Flotante**	**Coef. Coma Fija Q15**
0	0.00040242	0x000e	42	0.0020508	0x0044	84	0.0058688	0x00c1
1	0.00040637	0x000e	43	0.0021271	0x0046	85	0.005956	0x00c4
2	0.00041181	0x000e	44	0.0022047	0x0049	86	0.0060421	0x00c6
3	0.00041876	0x000e	45	0.0022838	0x004b	87	0.006127	0x00c9
4	0.00042726	0x000f	46	0.0023642	0x004e	88	0.0062105	0x00cc
5	0.00043733	0x000f	47	0.002446	0x0051	89	0.0062927	0x00cf
6	0.00044902	0x000f	48	0.0025289	0x0053	90	0.0063735	0x00d1
7	0.00046234	0x0010	49	0.0026131	0x0056	91	0.0064528	0x00d4
8	0.00047732	0x0010	50	0.0026985	0x0059	92	0.0065305	0x00d6
9	0.00049399	0x0011	51	0.002785	0x005c	93	0.0066066	0x00d9
10	0.00051236	0x0011	52	0.0028725	0x005f	94	0.0066811	0x00db
11	0.00053247	0x0012	53	0.002961	0x0062	95	0.0067538	0x00de
12	0.00055433	0x0013	54	0.0030505	0x0064	96	0.0068247	0x0000
13	0.00057795	0x0013	55	0.0031409	0x0067	97	0.0068938	0x0000
14	0.00060337	0x0014	56	0.0032321	0x006a	98	0.006961	0x0000
15	0.00063058	0x0015	57	0.0033241	0x006d	99	0.0070262	0x0000
16	0.00065961	0x0016	58	0.0034169	0x0070	100	0.0070894	0x0000

Nº	Coma Flotante	Coef. Coma Fija Q15	Nº	Coma Flotante	Coef. Coma Fija Q15	Nº	Coma Flotante	Coef. Coma Fija Q15
17	0.00069046	0x0017	59	0.0035103	0x0074	101	0.0071506	0x00eb
18	0.00072315	0x0018	60	0.0036043	0x0077	102	0.0072097	0x00ed
19	0.00075768	0x0019	61	0.0036988	0x007a	103	0.0072666	0x00ef
20	0.00079406	0x001b	62	0.0037938	0x007d	104	0.0073213	0x00f0
21	0.00083229	0x001c	63	0.0038893	0x0080	105	0.0073738	0x00f2
22	0.00087238	0x001d	64	0.0039851	0x0083	106	0.007424	0x00f4
23	0.00091432	0x001e	65	0.0040811	0x0086	107	0.0074718	0x00f5
24	0.00095812	0x0020	66	0.0041774	0x0089	108	0.0075173	0x00f7
25	0.0010038	0x0021	67	0.0042738	0x008d	109	0.0075605	0x00f8
26	0.0010513	0x0023	68	0.0043703	0x0090	110	0.0076011	0x00fa
27	0.0011006	0x0025	69	0.0044668	0x0093	111	0.0076393	0x00fb
28	0.0011518	0x0026	70	0.0045633	0x0096	112	0.007675	0x00fc
29	0.0012048	0x0028	71	0.0046596	0x0099	113	0.0077082	0x00fd
30	0.0012596	0x002a	72	0.0047558	0x009c	114	0.0077389	0x00fe
31	0.0013162	0x002c	73	0.0048517	0x009f	115	0.0077669	0x00ff
32	0.0013745	0x002e	74	0.0049472	0x00a3	116	0.0077923	0x0100
33	0.0014346	0x0030	75	0.0050423	0x00a6	117	0.0078152	0x0101
34	0.0014965	0x0032	76	0.005137	0x00a9	118	0.0078354	0x0101
35	0.0015601	0x0034	77	0.0052311	0x00ac	119	0.0078529	0x0102
36	0.0016253	0x0036	78	0.0053246	0x00af	120	0.0078677	0x0102
37	0.0016923	0x0038	79	0.0054174	0x00b2	121	0.0078799	0x0103
38	0.0017608	0x003a	80	0.0055095	0x00b5	122	0.0078894	0x0103
39	0.001831	0x003c	81	0.0056007	0x00b8	123	0.0078962	0x0103
40	0.0019027	0x003f	82	0.005691	0x00bb	124	0.0079002	0x0103
41	0.001976	0x0041	83	0.0057804	0x00be	125	0.0079016	0x0103

ANEXO H Programación del PIC18F2550

Como se indicó en el Capítulo 2, para hacer uso del módulo USB del PIC18F2550, se usó como base el ejemplo de la librería MLA que proporciona Microchip, en el cual se realizó las modificaciones, adaptaciones correspondientes y eliminación de código redundante, teniendo como resultado, un menora cantidad de archivos utilizados, los cuales se muestran en la **Figura 82**, de estos archivos, en los que se introduce la programación necesaria para el funcionamiento del prototipo, son "app_device_mouse.c" y "usb_descriptors.c", los cuales son se mostraran a continuación.

Es importante hacer notar la ausencia de tildes en el código, esto se debe a que el IDE de programación no reconoce símbolos con tilde.

Archivo "app_device_mouse.c"

```
#include "system.h"
#include <stdint.h>
#include "usb.h"
#include "usb_device.h"
#include "usb_device_hid.h"

#include "app_led_usb_status.h"
#include "app_device_mouse.h"
#include "usb_config.h"

#if(__XC8)
   #define PACKED
#else
   #define PACKED __attribute__((packed))
#endif

const struct{uint8_t report[HID_RPT01_SIZE];}
hid_rpt01=
{
   {
      0x05, 0x01, /* Usage Page (Generic Desktop)          */
      0x09, 0x02, /* Usage (Mouse)                        */
      0xA1, 0x01, /* Collection (Application)              */
```

Archivo "app_device_mouse.c"

```
      0x09, 0x01, /*  Usage (Pointer)                    */
      0xA1, 0x00, /*  Collection (Physical)              */
      0x05, 0x09, /*     Usage Page (Buttons)            */
      0x19, 0x01, /*     Usage Minimum (01)              */
      0x29, 0x02, /*     Usage Maximum (02)              */
      0x15, 0x00, /*     Logical Minimum (0)             */
      0x25, 0x01, /*     Logical Maximum (1)             */
      0x95, 0x02, /*     Report Count (2)                */
      0x75, 0x01, /*     Report Size (1)                 */
      0x81, 0x02, /*     Input (Data, Variable, Absolute)    */
      0x95, 0x01, /*     Report Count (1)                */
      0x75, 0x06, /*     Report Size (6)                 */
      0x81, 0x01, /*     Input (Constant)    ;6 bit padding  */
      0x05, 0x01, /*     Usage Page (Generic Desktop)        */
      0x09, 0x30, /*     Usage (X)                       */
      0x09, 0x31, /*     Usage (Y)                       */
      0x15, 0x81, /*     Logical Minimum (-127)          */
      0x25, 0x7F, /*     Logical Maximum (127)           */
      0x75, 0x08, /*     Report Size (8)                 */
      0x95, 0x02, /*     Report Count (2)                */
      0x81, 0x06, /*     Input (Data, Variable, Relative)    */
      0xC0, 0xC0  /* End Collection,End Collection          */
   }
};

typedef struct PACKED
{
   union PACKED
   {
      struct PACKED
      {
         unsigned button1   :1;
         unsigned button2   :1;
         unsigned :6;
      };
      uint8_t value;
   } buttons;
   uint8_t x;
   uint8_t y;
} MOUSE_REPORT;
```

Archivo "app_device_mouse.c"

```
//Variable mouseReport fijada en una dirección de memoria determinada, aquí se pone la acción que describirá el puntero
static MOUSE_REPORT mouseReport MOUSE_REPORT_DATA_BUFFER_ADDRESS;

typedef struct
{
   bool sentStop;

   bool movementMode;
   bool clicktMode;

   bool Click_Left_Anterior;
   bool Click_Left_Cambio;
   bool Click_Right_Anterior;
   bool Click_Right_Cambio;

   struct
   {
      USB_HANDLE handle;
      uint8_t idleRate;
      uint8_t idleRateSofCount;
   } inputReport[1];

} MOUSE;
//Variable "mouse", para controlar que el pulso de los botones del mouse, solo se ejecute en el flanco de subida
static MOUSE;

void APP_DeviceMouseInitialize(void)
{
   mouseReport.buttons.value = 0;
   mouseReport.x = 0;
   mouseReport.y = 0;
   mouse.movementMode = true;

   mouse.inputReport[0].handle = NULL;
   mouse.sentStop = true;


USBEnableEndpoint(HID_EP,USB_IN_ENABLED|USB_HANDSHAKE_ENABLED|USB_DISALLOW_SETUP);
}
```

Archivo "app_device_mouse.c"

```
void APP_DeviceMouseIdleRateCallback(uint8_t reportId, uint8_t idleRate)
{
   if(reportId == 0)
   {
      mouse.inputReport[reportId].idleRate = idleRate;
   }
}

void APP_DeviceMouseSOFHandler(void)
{

   if(USBGetDeviceState() != CONFIGURED_STATE)
   {
      return;
   }

   if(HIDTxHandleBusy(mouse.inputReport[0].handle) == false)
   {


////////////////////////////////////////////////////////////////////////////////////////////////

      uint8_t x_axis=0;
      uint8_t y_axis=0;

      if(BUTTON_IsPressed(Boton_X_Left)  &&  !BUTTON_IsPressed(Boton_X_Right))  //Izq precionada y derecha al aire =-1
          {x_axis=-1; LED_On(LED_Left);}//x-
      if(!BUTTON_IsPressed(Boton_X_Left) &&  BUTTON_IsPressed(Boton_X_Right)) //Izq al aire y derecha precionada =1
          {x_axis=1;LED_On(LED_Right);}//x+


      if(BUTTON_IsPressed(Boton_Y_Down) && !BUTTON_IsPressed(Boton_Y_Up))
          {y_axis=1;} //y- abajo
      if(!BUTTON_IsPressed(Boton_Y_Down) && BUTTON_IsPressed(Boton_Y_Up))
          {y_axis=-1;}//y+ arriba

      if(BUTTON_IsPressed(Boton_X_Left) == BUTTON_IsPressed(Boton_X_Right))
          {x_axis=0;LED_Off(LED_Left);LED_Off(LED_Right);}//x0
      if(BUTTON_IsPressed(Boton_Y_Down) == BUTTON_IsPressed(Boton_Y_Up) )
```

Archivo "app_device_mouse.c"

```
        {y_axis=0;} //y0


      if(x_axis!=0||y_axis!=0) {
      mouse.movementMode = true;
      LED_On(LED_Move);
      }
      if(x_axis==0&&y_axis==0) {
      mouse.movementMode = false;
      LED_Off(LED_Move);
      }


////////////////////////////////////////////////////////////////
// Guardar el estado actual del botón
bool estado_actual = BUTTON_IsPressed(Boton_Click_Left);

// Verificar si ha habido un cambio en el estado del botón
if (estado_actual != mouse.Click_Left_Anterior) {
   mouse.Click_Left_Cambio = true;
   mouse.Click_Left_Anterior = estado_actual;

   // Establecer el estado del botón en el informe del mouse
   mouseReport.buttons.button1 = estado_actual ? 1 : 0;


} else {
   mouse.Click_Left_Cambio = false;
}

////////////////////////////////////////////////////////////////////////////
// Guardar el estado actual del botón
estado_actual = BUTTON_IsPressed(Boton_Click_Right);

// Verificar si ha habido un cambio en el estado del botón
if (estado_actual != mouse.Click_Right_Anterior) {
   mouse.Click_Right_Cambio = true;
   mouse.Click_Right_Anterior = estado_actual;

   // Establecer el estado del botón en el informe del mouse
   mouseReport.buttons.button2 = estado_actual ? 1 : 0;
```

Archivo "app_device_mouse.c"

```
} else {
   mouse.Click_Right_Cambio = false;
}


if(mouse.Click_Right_Cambio||mouse.Click_Left_Cambio)
    // Establecer el modo de clic
   mouse.clicktMode = true; else mouse.clicktMode = false;

 ////////////////////////////////////////////////////////////////////////////////////////////////////

       if(mouse.movementMode ||  mouse.clicktMode)
       {

          mouseReport.x = x_axis; //Se indica la direccion del /////
          mouseReport.y = y_axis;       //movimiento        /////

          //Envio de datos al EP
          mouse.inputReport[0].handle = HIDTxPacket(
                                        HID_EP,
                                       (uint8_t*)&mouseReport,
                                        sizeof(mouseReport)
                                         );
//////////////////////////////////////////////////////////////////////////
          int valor_ch0 = (int) ADC_Read(0); // Lectura del canal 0
          long valor_ch0_long = 10 * valor_ch0 + 1;
          long cuenta = 0;
          for (cuenta = 0; cuenta < valor_ch0_long; cuenta++) {
             //for (cuenta = 0; cuenta < 10; cuenta++) {
             __delay_us(10);
          }
//////////////////////////////////////////////////////////////////////////

       }
       else
       {



          if(mouse.inputReport[0].idleRate != 0)
          {
```

Archivo "app_device_mouse.c"

```
            mouse.inputReport[0].idleRateSofCount++;
        }

        if(mouse.sentStop == true)
        {

            if(mouse.inputReport[0].idleRateSofCount <= mouse.inputReport[0].idleRate)
            {
                return;
            }
        }
    }

  }
}
```

Archivo "usb_descriptors.c"

```
#ifndef __USB_DESCRIPTORS_C
#define __USB_DESCRIPTORS_C

/** INCLUDES *******************************************************/
#include "usb.h"
#include "usb_device_hid.h"

/** CONSTANTS ******************************************************/
#if defined(__18CXX)
#pragma romdata
#endif

/* Device Descriptor */
const USB_DEVICE_DESCRIPTOR device_dsc=
{
   0x12,    // Size of this descriptor in bytes
   USB_DESCRIPTOR_DEVICE,              // DEVICE descriptor type
   0x0200,              // USB Spec Release Number in BCD format
   0x00,               // Class Code
   0x00,               // Subclass code
   0x00,               // Protocol code
   USB_EP0_BUFF_SIZE,         // Max packet size for EP0, see usb_config.h
   MY_VID,              // Vendor ID
```

Archivo "usb_descriptors.c"

```
    MY_PID,             // Product ID: Mouse in a circle fw demo
    0x0003,             // Device release number in BCD format
    0x01,               // Manufacturer string index
    0x02,               // Product string index
    0x00,               // Device serial number string index
    0x01                // Number of possible configurations
};

/* Configuration 1 Descriptor */
const uint8_t configDescriptor1[]={
    /* Configuration Descriptor */
    0x09,//sizeof(USB_CFG_DSC),    // Size of this descriptor in bytes
    USB_DESCRIPTOR_CONFIGURATION,               // CONFIGURATION descriptor type
    DESC_CONFIG_WORD(0x0022),   // Total length of data for this cfg
    1,                  // Number of interfaces in this cfg
    1,                  // Index value of this configuration
    0,                  // Configuration string index
    _DEFAULT | _SELF,           // Attributes, see usb_device.h
    50,                 // Max power consumption (2X mA)

    /* Interface Descriptor */
    0x09,//sizeof(USB_INTF_DSC),   // Size of this descriptor in bytes
    USB_DESCRIPTOR_INTERFACE,           // INTERFACE descriptor type
    0,                  // Interface Number
    0,                  // Alternate Setting Number
    1,                  // Number of endpoints in this intf
    HID_INTF,           // Class code
    BOOT_INTF_SUBCLASS,     // Subclass code
    HID_PROTOCOL_MOUSE,     // Protocol code
    0,                  // Interface string index

    /* HID Class-Specific Descriptor */
    0x09,//sizeof(USB_HID_DSC)+3,    // Size of this descriptor in bytes RRoj hack
    DSC_HID,            // HID descriptor type
    DESC_CONFIG_WORD(0x0111),           // HID Spec Release Number in BCD format (1.11)
    0x00,               // Country Code (0x00 for Not supported)
    HID_NUM_OF_DSC,         // Number of class descriptors, see usbcfg.h
    DSC_RPT,            // Report descriptor type
    DESC_CONFIG_WORD(50),   //sizeof(hid_rpt01),      // Size of the report descriptor

    /* Endpoint Descriptor */
    0x07,/*sizeof(USB_EP_DSC)*/
```

Archivo "usb_descriptors.c"

```
    USB_DESCRIPTOR_ENDPOINT,    //Endpoint Descriptor
    HID_EP | _EP_IN,            //EndpointAddress
    _INTERRUPT,                 //Attributes
    DESC_CONFIG_WORD(3),        //size
    0x01                        //Interval
};


//Language code string descriptor
const struct{uint8_t bLength;uint8_t bDscType;uint16_t string[1];}sd000={
sizeof(sd000),USB_DESCRIPTOR_STRING,{0x0409
}};

//Manufacturer string descriptor                25 -> 27
const struct{uint8_t bLength;uint8_t bDscType;uint16_t string[27];}sd001={
sizeof(sd001),USB_DESCRIPTOR_STRING,
    {'K','e','v','i','n',' ','C','a','r','r','e','N','o','-',
     'U','C','S','M',' ','A','r','e','q','u','i','p','a'
    }
};

//Product string descriptor                     22 -> 26
const struct{uint8_t bLength;uint8_t bDscType;uint16_t string[26];}sd002={
sizeof(sd002),USB_DESCRIPTOR_STRING,
    {'T', 'e', 's', 'i', 's', ':', ' ', 'M', 'o', 'u', 's', 'e', ' ',
     'E', 'O', 'G', ' ', 'c', 'o', 'n', ' ', 'd', 's', 'P', 'I', 'C'
    }
};

//Array of configuration descriptors
const uint8_t *const USB_CD_Ptr[]=
{
    (const uint8_t *const)&configDescriptor1
};

//Array of string descriptors
const uint8_t *const USB_SD_Ptr[]=
{
    (const uint8_t *const)&sd000,
    (const uint8_t *const)&sd001,
    (const uint8_t *const)&sd002
};
```

Archivo "usb_descriptors.c"

```
/** EOF usb_descriptors.c ***************************************************/

#endif
```

Adicionalmente, para definir y utilizar los pines con los que se comunica de manera digital con el dsPIC, se utiliza los archivos "buttons.h" y "buttons.c".

Archivo "buttons.h"

```
#include <stdbool.h>



/*** Button Definitions ****************************************************/
typedef enum
{
   BUTTON_NONE,
   Boton_X_Right,
   Boton_Y_Up,
   Boton_Y_Down,
   Boton_X_Left,
   Boton_Click_Left,
   Boton_Click_Right

} BUTTON;

/** LED defintions **********************************/
typedef enum
{
   LED_NONE,
   LED_Move,
   LED_Left,
   LED_Right
} LED;


bool BUTTON_IsPressed(BUTTON button);

void LED_On(LED led);
void LED_Off(LED led);

void BUTTON_Enable();// Se activan todos los pines
```

Archivo "buttons.h"

```
void LED_Enable();



void ADC_Init();
uint16_t ADC_Read(unsigned char ch);
```

Archivo "buttons.c"

```
#include <stdbool.h>
#include <xc.h>
#include <buttons.h>

#define X_Right_PORT  PORTBbits.RB4
#define X_Left_PORT   PORTBbits.RB5
#define Y_Up_PORT     PORTBbits.RB3
#define Y_Down_PORT   PORTBbits.RB2

#define Click_Left_PORT   PORTBbits.RB1
#define Click_Right_PORT  PORTBbits.RB0

#define X_Right_TRIS  TRISBbits.RB4
#define X_Left_TRIS   TRISBbits.RB3
#define Y_Up_TRIS     TRISBbits.RB3
#define Y_Down_TRIS   TRISBbits.RB2

#define Click_Left_TRIS   TRISBbits.RB1
#define Click_Right_TRIS  TRISBbits.RB0

#define BUTTON_PRESSED      1
#define BUTTON_NOT_PRESSED  0

#define PIN_INPUT          1
#define PIN_OUTPUT         0
//*****************************************************************************

#define LED_Move_LAT  LATCbits.LATC1   //Movimiento
#define LED_Left_LAT  LATCbits.LATC0   //Izquierda
#define LED_Right_LAT LATCbits.LATC2  //Derecha

#define LED_Move_TRIS  TRISCbits.TRISC1 //Movimiento
#define LED_Left_TRIS  TRISCbits.TRISC0 //Izquierda
```

Archivo "buttons.c"

```
#define LED_Right_TRIS TRISCbits.TRISC2 //Derecha

#define LED_ON  1
#define LED_OFF 0

#define INPUT  1
#define OUTPUT 0

bool BUTTON_IsPressed(BUTTON button)
{
   switch(button)
   {
      case Boton_X_Right:
         return ( (X_Right_PORT == BUTTON_PRESSED) ? true : false);

      case Boton_Y_Up:
         return ( (Y_Up_PORT == BUTTON_PRESSED) ? true : false);

      case Boton_Y_Down:
         return ( (Y_Down_PORT == BUTTON_PRESSED) ? true : false);

      case Boton_X_Left:
         return ( (X_Left_PORT == BUTTON_PRESSED) ? true : false);

      case Boton_Click_Left:
         return ( (Click_Left_PORT == BUTTON_PRESSED) ? true : false);

      case Boton_Click_Right:
         return ( (Click_Right_PORT == BUTTON_PRESSED) ? true : false);

      case BUTTON_NONE:
         return false;
   }

   return false;
}

/*****************************************************************************
******************************************************************************/
void BUTTON_Enable()// Se activan todos los pines
{
   X_Right_TRIS = PIN_INPUT;
```

Archivo "buttons.c"

```
    Y_Up_TRIS    = PIN_INPUT;
    Y_Down_TRIS  = PIN_INPUT;
    X_Left_TRIS  = PIN_INPUT;

    Click_Left_TRIS  = PIN_INPUT;
    Click_Right_TRIS = PIN_INPUT;
//    INTCON2bits.RBPU=0; // Se activan los pull Ups del puerto B
//                        //Si se quita el comentario a esta linea, negar (!) los return de la funcion BUTTON_IsPressed()
//                        //Si se pone el comentario a esta linea, quitar la negacion (!) a los return de la funcion BUTTON_IsPressed()
 }


/*****************************************************************
*****************************************************************/
void LED_On(LED led)
{
   switch(led)
   {
      case LED_Move:
         LED_Move_LAT = LED_ON;
         break;

      case LED_Left:
         LED_Left_LAT = LED_ON;
         break;

      case LED_Right:
         LED_Right_LAT = LED_ON;
         break;

       case LED_NONE:
         break;
   }
}

/*****************************************************************
*****************************************************************/

void LED_Off(LED led)
{
```

Archivo "buttons.c"

```
    switch(led)
    {
        case LED_Move:
            LED_Move_LAT = LED_OFF;
            break;

        case LED_Left:
            LED_Left_LAT = LED_OFF;
            break;

        case LED_Right:
            LED_Right_LAT = LED_OFF;
            break;

         case LED_NONE:
            break;
    }
}

void LED_Enable()
{
            LED_Move_TRIS  = OUTPUT;
            LED_Left_TRIS  = OUTPUT;
            LED_Right_TRIS = OUTPUT;

            LED_Move_LAT = LED_OFF;
            LED_Left_LAT = LED_OFF;
            LED_Right_LAT = LED_OFF;
}

////////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////////

void ADC_Init() {
    //ADCON1bits.PCFG=0b1110;  //AN0 analogico RA0
    ADCON0 = 0x00;
    ADCON1 = 0x0E; //AN0 analogico RA0 (Unicamente)

    ADCON2 = 0x8F;//Seleccion de reloj, tiempo de adquisicion
}
```

Archivo "buttons.c"

```
uint16_t ADC_Read(unsigned char ch) {
   if (ch > 13) {
      return 0;
   } else {
      ADCON0 = 0x00;
      ADCON0 = (ch << 2);//lo mismo que escribir: ADCON0bits.CHS=ch; (Seleccion de channel)
      ADCON0bits.ADON = 1;
      ADCON0bits.GO_DONE = 1;
      while (ADCON0bits.GO_DONE == 1);
      return ((uint16_t)ADRESH << 8) | ADRESL; //Devuelve el resultado de 10bits 0-1023
   }
}
```

ANEXO I Programación del dsPIC33CH128MP506

Para la programación del dsPIC, primero se crea y definen los archivos con las variables y funciones de todos sus módulos a utilizar, y luego, se crea los archivos correspondientes para el procesamiento de la señal EOG.

Es importante hacer notar la ausencia de tildes en el código, esto se debe a que el IDE de programación no reconoce símbolos con tilde.

Archivo "Pins.h"

```
#include <stdbool.h>
//////////////////DIGIT_INPUT PINS/////////////////////////////////////////////////////////
#define SET_PIN_Select_DACOut_Function_1    TRISCbits.TRISC5=1; \
                                    CNPDCbits.CNPDC5=1;
//                          Function_2:  Es la cuando  funcion 1 y 3 esta n en bajo
#define SET_PIN_Select_DACOut_Function_3    TRISCbits.TRISC4= 1; \
                                    CNPDCbits.CNPDC4= 1;

#define PIN_STATUS_Select_DACOut_Function_1 PORTCbits.RC5==1
//                            Function_2:  Es la cuando  funcion 1 y 3 esta n en bajo
#define PIN_STATUS_Select_DACOut_Function_3 PORTCbits.RC4==1
////////////////////////////////////////////////////////////

#define CONFIG_Input_Pin_1    TRISDbits.TRISD13= 1; \
                        CNPDDbits.CNPDD13= 1;
#define CONFIG_Input_Pin_2    TRISDbits.TRISD14= 1; \
                        CNPDDbits.CNPDD14= 1;
#define CONFIG_Input_Pin_3    TRISDbits.TRISD15= 1; \
                        CNPDDbits.CNPDD15= 1;

#define IS_Input_Pin_1_SET_   PORTDbits.RD13==1
#define IS_Input_Pin_2_SET_   PORTDbits.RD14==1
#define IS_Input_Pin_3_SET_   PORTDbits.RD15==1


//#define DACOut_Channel_1_Selected_H true
//#define DACOut_Channel_2_Selected_V false


//////////////////DIGIT_OUTPUT PINS/////////////////////////////////////////////////////////
```

Archivo "Pins.h"

```
#define DIGITAL_OUTPUTS 0
#define DIGITAL_INPUTS  1
#define ANALOGICOS      2
#define PIN_USES DIGITAL_INPUTS   //Seleccinar que uso se le dara a los pines


#define DIGIT_ANALOG_PINS_in_Out_AS_  PIN_USES
#if DIGIT_ANALOG_PINS_in_Out_AS_ == DIGITAL_OUTPUTS

   #define CONFIG_Output_Pin_1   TRISAbits.TRISA4 =0;ANSELAbits.ANSELA4 =0;
   #define CONFIG_Output_Pin_2   TRISDbits.TRISD12=0;
   #define CONFIG_Output_Pin_3   TRISCbits.TRISC1 =0;ANSELCbits.ANSELC1 =0;

   #define PRENDER_Output_Pin_1       LATAbits.LATA4 =1;
   #define PRENDER_Output_Pin_2       LATDbits.LATD12=1;
   #define PRENDER_Output_Pin_3       LATCbits.LATC1 =1;

   #define APAGAR_Output_Pin_1       LATAbits.LATA4 =0;
   #define APAGAR_Output_Pin_2       LATDbits.LATD12=0;
   #define APAGAR_Output_Pin_3       LATCbits.LATC1 =0;

   #define TOGGLE_Output_Pin_1       LATAbits.LATA4  ^= 1
   #define TOGGLE_Output_Pin_2       LATDbits.LATD12 ^= 1
   #define TOGGLE_Output_Pin_3       LATCbits.LATC1  ^= 1

#elif DIGIT_ANALOG_PINS_in_Out_AS_ == DIGITAL_INPUTS

   #define CONFIG_Input_Pin_4    ANSELAbits.ANSELA4 = 0;  \
                         TRISAbits.TRISA4   = 1; \
                         CNPDAbits.CNPDA4   = 1;
   #define CONFIG_Input_Pin_5    TRISDbits.TRISD12  = 1; \
                         CNPDDbits.CNPDD12  = 1;
   #define CONFIG_Input_Pin_6    ANSELCbits.ANSELC1 = 0; \
                         TRISCbits.TRISC1   = 1; \
                         CNPDCbits.CNPDC1   = 1;

   #define IS_Input_Pin_4_SET_   PORTAbits.RA4  ==1
   #define IS_Input_Pin_5_SET_   PORTDbits.RD12 ==1
   #define IS_Input_Pin_6_SET_   PORTCbits.RC1  ==1
```

Archivo "Pins.h"

```
#endif

//////////////////////////////////////////////////////////////

#define CONFIG_LED_Ok_Indicator  TRISAbits.TRISA2=0;

#define PRENDER_LED_Ok_Indicator     LATAbits.LATA2 =1;
#define APAGAR_LED_Ok_Indicator      LATAbits.LATA2 =0;
#define TOGGLE_LED_Ok_Indicator      LATAbits.LATA2  ^= 1
//////////////USB CONTROL PINS/////////////////////////////////////////////////////////////////

                    // TRIS_H_LEFT       TRIS_H_RIGHT
#define SET_PIN_H_AXIS  TRISBbits.TRISB14=0; TRISBbits.TRISB15=0;
                    // TRIS_V_DOWN       TRIS_V_UP
#define SET_PIN_V_AXIS  TRISBbits.TRISB12=0; TRISBbits.TRISB13=0;
                    // Click_L           Click_R
#define SET_PIN_CLICKs  TRISBbits.TRISB9 =0; TRISBbits.TRISB8 =0;


// Pines para control USB
#define MOVE_H_LEFT     LATBbits.LATB14
#define MOVE_H_RIGHT    LATBbits.LATB15

#define MOVE_V_UP       LATBbits.LATB13
#define MOVE_V_DOWN     LATBbits.LATB12

#define PRESS_CLICK_L   LATBbits.LATB9
#define PRESS_CLICK_R   LATBbits.LATB8
//////////////////////////////////////////////////////////////////////////////////////////////

typedef enum{
   Function_1, //DAC Source Channel Function - RC5 - Pendiente       - IZQUIERDA
   Function_2, //DAC Source Channel Function -     - Pulso Analogico - CENTRO
   Function_3, //DAC Source Channel Function - RC4 - Pulso Digital   - DERECHA

   Input_Pin_1,  //DAC Source Channel Select ON-OFF - RC1 - ON -> Channel 1
       //                                OFF-> Channel 2
   Input_Pin_2,
   Input_Pin_3,
   Input_Pin_4,
   Input_Pin_5,
   Input_Pin_6
```

Archivo "Pins.h"

```
}dsPIC_Pin;


void digital_init(void);

bool get_pin(dsPIC_Pin Pin);
```

Archivo "Pins.c"

```
#include " Pins.h"

void digital_init(void)
{  //Conf de pines para el selector de salida del DAC
   SET_PIN_Select_DACOut_Function_1;    //Select DAC Out - Function 1
   SET_PIN_Select_DACOut_Function_3;    //Select DAC Out - Function 2

   CONFIG_Input_Pin_1;//Select DAC Out - Channel
   CONFIG_Input_Pin_2;
   CONFIG_Input_Pin_3;

#if DIGIT_ANALOG_PINS_in_Out_AS_ == DIGITAL_OUTPUTS
   CONFIG_Output_Pin_1;
   CONFIG_Output_Pin_2;
   CONFIG_Output_Pin_3;
   APAGAR_Output_Pin_1;
   APAGAR_Output_Pin_2;
   APAGAR_Output_Pin_3 ;
#elif DIGIT_ANALOG_PINS_in_Out_AS_ == DIGITAL_INPUTS

   CONFIG_Input_Pin_4;
   CONFIG_Input_Pin_5;
   CONFIG_Input_Pin_6;

#endif


////////////////////////////////////////////////
   SET_PIN_H_AXIS;
   SET_PIN_V_AXIS;
   SET_PIN_CLICKs;
```

Archivo "Pins.c"

```
////// LED de Funcionamiento
   CONFIG_LED_Ok_Indicator;
  //  LATAbits.LATA2=1;

}




bool get_pin(dsPIC_Pin Pin)
{
   switch(Pin)
   {

       case Function_1:
          return ( (PIN_STATUS_Select_DACOut_Function_1) ? true : false);
          break;
       case Function_2:
          return                                                                    (
!(PIN_STATUS_Select_DACOut_Function_1||PIN_STATUS_Select_DACOut_Function_3)     ?
true : false);
          break;
       case Function_3:
          return ( (PIN_STATUS_Select_DACOut_Function_3) ? true : false);
          break;

       case Input_Pin_1:
          return ( (IS_Input_Pin_1_SET_) ? true : false);
          break;
       case Input_Pin_2:
          return ( (IS_Input_Pin_2_SET_) ? true : false);
          break;
       case Input_Pin_3:
          return ( (IS_Input_Pin_3_SET_) ? true : false);
          break;

#if DIGIT_ANALOG_PINS_in_Out_AS_ == DIGITAL_INPUTS
       case Input_Pin_4:
          return ( (IS_Input_Pin_4_SET_) ? true : false);
          break;
       case Input_Pin_5:
          return ( (IS_Input_Pin_5_SET_) ? true : false);
```

Archivo "Pins.c"

```
            break;
        case Input_Pin_6:
            return ( (IS_Input_Pin_6_SET_) ? true : false);
            break;
#endif



        default:
            break;


    }

//    return false;
}
```

Archivo "ADC_Lib.h"

```
#ifndef ADC_LIB_H

#define          ADC_LIB_H
#include <stdint.h>

#define SET_ADC_PIN_Signal_Channel_1    ANSELCbits.ANSELC7=1;  \
                                TRISCbits.TRISC7=1;    \
                                ADTRIG3Hbits.TRGSRC15=1;//AN15 Common Software Trigger - Ver
Reg. 3-178
#define CONVERTING_Signal_Channel_1     ADSTATLbits.AN15RDY
#define BUFFER_RESULT_Signal_Channel_1  ADCBUF15;


#define SET_ADC_PIN_Signal_Channel_2    ANSELCbits.ANSELC2=1;  \
                                TRISCbits.TRISC2=1;    \
                                ADTRIG3Hbits.TRGSRC14=1;//AN14 Common Software Trigger - Ver
Reg. 3-178
#define CONVERTING_Signal_Channel_2     ADSTATLbits.AN14RDY
#define BUFFER_RESULT_Signal_Channel_2  ADCBUF14;


typedef enum{
   CHAN_AN0
```

Archivo "ADC_Lib.h"

```
}Canal_ADC;

void ADC_Iniciar(void);

void AnalogRead_All(int* chan_0,int* chan_1);//(Canal_ADC ANx)
bool Rellenar_Buffers_EOG_START_PROGRAM(int, int);

#endif  /* ADC_LIB_H */
```

Archivo "ADC_Lib.c"

```
#include "Config.h"
#include <libpic30.h>

//extern volatile uint16_t RecepADC_1;
//extern volatile uint16_t RecepADC_2;

void ADC_Iniciar(void){
////////////////////////////////////
   SET_ADC_PIN_Signal_Channel_l;
   SET_ADC_PIN_Signal_Channel_2;

 ////////////////////////////////////////////////////////////
   //Apagamos el ADC
   ADCON1Lbits.ADON=0;
   ADCON3Hbits.SHEN=0;

   //Formato de dato
   ADCON1Hbits.FORM=2; //Format Integer
   ADCON1Hbits.SHRES=0;//Resolution 12 bits

   ADCON3Lbits.SHSAMP=3;

   //Voltaje de Referencia
   ADCON3Lbits.REFSEL=1;

   //Tiempo de Muestreo  FOSC-> POSC=20MHz
      //Fuera del Modulo ADC
   ADCON3Hbits.CLKSEL=0; //ADC Module Source -> 1=FOSC ; 0 =FOSC/2
   ADCON3Hbits.CLKDIV=0; //ADC Module Source div -> FOSC/1

      //Dentro del Modulo ADC
```

Archivo "ADC_Lib.c"

```
    ADCON2Lbits.SHRADCS=1;//TAD=( 1 *2))(Tfosc) -> TAD = 100ns
       //Sampling Time
    ADCON2Hbits.SHRSMC=8; //SP=(8 +2)*TAD -> SP=10*(100ns))
       //Conversion Time = 8*Tcoresrc + (Bits_resolution + 2.5)*TADcore
       //Conv Time = 8*50ns + (12 + 2.5)*100ns  =  1.85us


    //Prendemos el ADC
    ADCON1Lbits.ADON=1;
    ADCON5Lbits.SHRPWR=1;
    while(!ADCON5Lbits.SHRRDY){}
    ADCON3Hbits.SHEN=1;


}


void AnalogRead_All(int* chan_0,int* chan_1){

    ADCON3Lbits.SWCTRG = 1;

    while(!(CONVERTING_Signal_Channel_1)) { }
    *chan_0 = BUFFER_RESULT_Signal_Channel_1;

    while(!(CONVERTING_Signal_Channel_2)) { }
    *chan_1 = BUFFER_RESULT_Signal_Channel_2;
}

uint32_t Esperando=10;
bool Rellenar_Buffers_EOG_START_PROGRAM(int Analog_H, int Analog_V){
    bool Start_Program=false;


//    Esperando=0x3FFFF;
    //rango inicial aceptado : 1.21V a 2.74V
    if((Analog_H<1500||Analog_H>3400) || (Analog_V<1500||Analog_V>3400) ){
       if (--Esperando == 0) {
          Esperando=0x5FFF;TOGGLE_LED_Ok_Indicator;}
    }else
    {
       Start_Program=true;
```

Archivo "ADC_Lib.c"

```
        PRENDER_LED_Ok_Indicator;

        for(int i=0;i<=N_;i++){
        Horizontal.Discret_Buffer[i] = Analog_H;
        Horizontal.Deriv_Buffer[i]   = 100;

        Vertical.Discret_Buffer[i]   = Analog_V;
        Vertical.Deriv_Buffer[i]     = 100;
        }
        Vertical.Eje=2; //Para que solo esta variable ejecute Blink Counter.

    };


        return Start_Program;
}
```

Archivo "Timers.h"

```
extern bool TMR1_Done;
extern bool TMR2_Done;


//void Timer1_Init(void);
void Timer1_CCP1_Init(void);

//void __attribute__ ((interrupt,no_auto_pv)) _T1Interrupt(void);
void __attribute__((interrupt, no_auto_pv)) _CCT1Interrupt(void);
```

Archivo "Timers.c"

```
void Timer1_CCP1_Init(void) {
    // Configurar CCP1 para funcionar como un temporizador de 16 bits
    CCP1CON1Lbits.CCPON = 0;      // Apagar el módulo CCP1 para configuración
    CCP1CON1Lbits.CCSEL = 0;      // Seleccionar modo de comparador/PWM/Timer
    CCP1CON1Lbits.MOD = 0b0010;   // Seleccionar modo de temporizador de 16 bits //OJO CON
ESTE
    CCP1CON1Lbits.TMRPS = 0b11;   // Prescaler 1:64

    // pr ->   fcy*t/pre -1
    CCP1PRL = 1562;               // Período del temporizador (ajustar según la frecuencia deseada)
312 ->500 Hz 2ms
```

Archivo "Timers.c"

```
                        // 1562 -> 100Hz 10ms
    CCP1TMRL = 0;


    // Inicializar el contador del temporizador
    IEC0bits.CCT1IE = 1;          // Habilitar la interrupción del CCP1
    IFS0bits.CCT1IF = 0;          // Limpiar la bandera de interrupción del CCP1
    IPC1bits.CCT1IP = 5;          // Prioridad de la interrupción del CCP1


    CCP1CON1Lbits.CCPON = 1;      // Encender el módulo CCP1
}

void __attribute__((interrupt, no_auto_psv)) _CCT1Interrupt(void) {

//    dsPIC_Init();
    TMR2_Done=true;
//    LATDbits.LATD1=!LATDbits.LATD1;
    IFS0bits.CCT1IF = 0; // Limpiar la bandera de interrupción del CCP1
    // Código de manejo de la interrupción del CCP1
}
```

Archivo "DAC_Lib.h"

```
#ifndef DAC_LIB_H
#define          DAC_LIB_H
#include <stdint.h>

void DAC_APLL_Init(void);
void DAC_write(int16_t ,uint16_t );

#endif  /* DAC_LIB_H */
```

Archivo "DAC_Lib.c"

```
#include "DAC_Lib.h"
#include <xc.h>

void DAC_APLL_Init(void){

    ////////Auxiliary PLL
    ACLKCON1bits.PLLEN=1; // Salida del PLL es: La salida del PLL
```

Archivo "DAC_Lib.c"

```
    ACLKCON1bits.FRSEL=0; //POSCLK de 20MHz

    ACLKCON1bits.APLLPRE=1; //PRE 1/1
    APLLFBD1bits.APLLFBDIV=20; // Feedback Divider de 1/40 Fvco=400MHz
    APLLDIV1bits.APOST1DIV=2;  //Post Div de 1/2    F=200MHz
    APLLDIV1bits.APOST2DIV=2;  //Post Div de 1/2    AFPLLO=100MHz

    APLLDIV1bits.AVCODIV=2; //AFvcodiv = AFvco/2

    //while(!ACLKCON1bits.APLLCK) //Hacer nada mientras el APLL no haga LOCK

    ////////DAC Confg

    DACCTRL1Lbits.CLKSEL=2; //ADC Clock= AFPLLO 100MHZ //Al trabajar frecuencias mas
bajas, se genera ruido en la salida del DAC
    DACCTRL1Lbits.CKDIV=0; //ADC Clock Div= 1/1
    DAC1CONLbits.DACOEN=1; //Connect DAC1 a pin DACOUT1

    DACCTRL1Lbits.DACON=1; //Prender los modulos DAC
    DAC1DATHbits.DACDATH=0; //Carga 0 para sacar por el DAC
    DAC1CONLbits.DACEN=1;  //Habilita el modulo DAC1

}

void DAC_write(int16_t Input,uint16_t Offset){
    if(Offset>4095||Input>4095){
            DAC1DATHbits.DACDATH=4000;
            DAC1CONLbits.DACEN=1;
            return;
    }
    int16_t DAC_Out=Input+Offset;
    if(DAC_Out>=4050) DAC_Out=4000;
    if(DAC_Out<=40) DAC_Out=0;
    DAC1DATHbits.DACDATH=DAC_Out;
    DAC1CONLbits.DACEN=1;
}
```

A continuación, se muestra los archivos utilizados para el procesamiento de las señales EOG.

Archivo "Slope_Cont.h"

```
#define DigitalPulse_History_Length  20//30 x 10ms = 0.3s // prueba con 40->400ms
```

Archivo "Slope_Cont.h"

```
#define DigitalPulse_Derivada_Length DigitalPulse_History_Length
#define N_                           DigitalPulse_History_Length-1  // Para simular a Matlab

#define Conv_Atenuador   20

#define Slope_Umbral 200
#define Height_Umbral    400
#define Height_Escalon_Umbral Height_Umbral+100
#define Height_Remonte_Umbral Height_Umbral+200

#define Blink_Counter_Umbral        700

//Determina la inclinacion vertical minima

#define Dir_Positivo  1
#define Dir_Central   0
#define Dir_Negativo -1



#define DigitalPulse_Blink_H         4 //4  x 10ms = 0.04s
#define DigitalPulse_Blink_V         11 //9  x 10ms = 0.11s
#define Digital_Pulse_Actual        DigitalPulseHistory[0]

//////////////////////////////////////////////////////////////

#define Pos_Slope    Dir_Positivo
#define Flat_Slope   Dir_Central
#define Neg_Slope    Dir_Negativo

#define Umbral_Enmascardo_Horizontal 3000
#define Umbral_Enmascardo_Vertical   3000



#define H_LEFT    -1
#define H_RIGHT    1
#define H_CENTER   0

#define V_DOWN    -1
#define V_UP       1
#define V_CENTER   0
```

Archivo "Slope_Cont.h"

```
typedef struct {
   int Eje; //1:Horizontal.  2:Vertical.

   int Valor_FIR_PB;
   int Valor_FIR_LMS;
   ////////////////////////////////////////////////////
   int Discret_Buffer[DigitalPulse_History_Length];
   int Deriv_Buffer[DigitalPulse_Derivada_Length];
   int Actual_Slope_Sign;
   int32_t Deriv_Conv;
   ////////////////////////////////////////////////////

   int Triggers;
   int Slope_Sign_Init;
   int Slope_Sign_Final;
   int Idle;
   int Offset;
   int Remontada;
   int Anular;

   ////////////////////////////////////////////////////
   int Polarizacion;

   int EOG_Base;
   int EOG_n_Base;

   int EOG_Peak;
   int EOG_n_Peak;

   int EOG_Fin;
   int EOG_n_Fin;
   ////////////////////////////////////////////////////
   int Pulso_Dig;

   ////////////////////////////////////////////////////
   int Blink_Contador;
   int Blink_Secuencia;
} Slope_Control;
```

Archivo "Slope_Cont.h"

```
typedef struct{

   int Triggers;

   int H_Duration;
   int H_Height;
   int H_Anulado;
   int H_History[3];

   int V_Duration;
   int V_Height;
   int V_Anulado;
   int V_History[4];

   int H_Diferencia[2];
   int V_Diferencia[2];

   int N_Parpadeos;

   int Mov_Horizontal;
   int Mov_Vertical;

   int Blink;
   int Click_Izq;
   int Click_Der;

   int Indeterminado;

   int Orden_Eje_H;
   int Orden_Eje_V;
   int Orden_Click_L;
   int Orden_Click_R;

}Channel_Comp;

void Historial_Discreto_Update(Slope_Control*,int);
void Delta_Signal(Slope_Control*);
void Convolucion_Derivada(Slope_Control*);

void Signal_Digitalitation(Slope_Control*);
void Blink_Counter(Slope_Control*);
void Signal_Comparation(Slope_Control*,Slope_Control*,Channel_Comp*);
```

Archivo "Slope_Cont.h"

```
void Mouse_Move_Order  (Channel_Comp*);
//Funciones Optimizadas con DPS Engine

extern int VectorDotProduct_MOD (int numElems,int* srcV1,int* srcV2, int32_t* Acumulador);
//~115 ciclo de ejecucion

extern int Vector_Max_n(int Index_Limite, int* Vector, int Vector_Length, int* Indice_Resultado);
extern int Vector_Min_n(int Index_Limite, int* Vector, int Vector_Length, int* Indice_Resultado);
```

Archivo "Slope_Cont.c"

```
#include "Config.h"

int unidad [DigitalPulse_History_Length]=   {1 ,1 ,1 ,1 ,1 ,1 ,1 ,1 ,1 ,1 ,
                                    1 ,1 ,1 ,1 ,1 ,1 ,1 ,1 ,1 ,1 };
//                                   1 ,1 ,1 ,1 ,1 ,1 ,1 ,1 ,1 ,1 };

void Delta_Signal(Slope_Control* Signal){
   int Delta_derivada_Signed;
   Delta_derivada_Signed=2*Signal->Discret_Buffer[N_]+Signal->Discret_Buffer[N_-1]-Signal->Discret_Buffer[N_-2]-2*Signal->Discret_Buffer[N_-3];
   Signal->Actual_Slope_Sign=Delta_derivada_Signed;
   DMA_LeftShift(Signal->Deriv_Buffer, abs(Delta_derivada_Signed));
//    ShiftRightPointer(,DigitalPulse_Derivada_Length,Delta_derivada);


}

void Historial_Discreto_Update(Slope_Control* Signal,int Nuevo){
   DMA_LeftShift(&Signal->Discret_Buffer, Nuevo);
//        ShiftRightPointer(Signal->Historial_Discreto,DigitalPulse_History_Length,Nuevo);  //150 ciclos

}

void Convolucion_Derivada(Slope_Control* Signal_1){
   uint32_t  Enmascarado_1=0;
```

Archivo "Slope_Cont.c"

```
   VectorDotProduct_MOD (DigitalPulse_History_Length,Signal_1->Deriv_Buffer,unidad
,&Enmascarado_1);

   signed int *remainder;
   Signal_1->Deriv_Conv =__builtin_divmodsd( Enmascarado_1, Conv_Atenuador,
&remainder)+10;



}

void Signal_Digitalitation(Slope_Control* Signal){
////////////////////////////////////////////////////////////////////////////
   if(Signal->Triggers==0 || Signal->Triggers==3){
      int Actual_Slope=Signal->Deriv_Buffer[N_]-Signal->Deriv_Buffer[N_ - 5];
      if(Actual_Slope>Slope_Umbral){
         Signal->Slope_Sign_Init = Signal->Actual_Slope_Sign;

         if(Signal->Triggers==0){
            if (Signal->Idle<10) Signal->Offset=1;
            else          Signal->Offset=7;

            if(Signal->Slope_Sign_Init>0){
              Signal->EOG_Base = Vector_Min_n(
                    N_-Signal->Offset     , Signal->Discret_Buffer,
                    DigitalPulse_History_Length, &(Signal->EOG_n_Base)   );
            }
            else{
              Signal->EOG_Base = Vector_Max_n(
                    N_-Signal->Offset     , Signal->Discret_Buffer,
                    DigitalPulse_History_Length, &(Signal->EOG_n_Base)   );
            }
             Signal->EOG_n_Base++;
             Signal->Triggers=1;
         } else if(Signal->Triggers==3) Signal->Triggers=4;
      }
      else
      {
         if(Signal->Triggers==0){
            if (Signal->Idle < 1000) Signal->Idle++;
         }
         if(Signal->Triggers==3){
```

Archivo "Slope_Cont.c"

```
            if(Signal->EOG_n_Peak <=12 && abs(Signal->EOG_Peak - Signal->Discret_Buffer[N_]) < 100 && abs(Signal->EOG_Peak) < Height_Escalon_Umbral ){
                Signal->Triggers=9;
            }else if (Signal->EOG_n_Peak <=6 && abs(Signal->EOG_Peak - Signal->Discret_Buffer[N_]) < 100 && abs(Signal->EOG_Peak) >= Height_Escalon_Umbral ){
                Signal->Triggers=9;
            }
        }
    }
  }

////////////////////////////////////////////////////////////////////////////
  if(Signal->Triggers==1 || Signal->Triggers==4){
    int Actual_Slope=Signal->Deriv_Buffer[N_]-Signal->Deriv_Buffer[N_ - 5];
    Signal->Slope_Sign_Final = Signal->Actual_Slope_Sign;

    int Cambio_Signo = (Signal->Slope_Sign_Init ^ Signal->Slope_Sign_Final);

    if(Actual_Slope <= Slope_Umbral){
      if(Signal->Triggers==1){
        Signal->Polarizacion = Signal ->Discret_Buffer[N_]- Signal ->EOG_Base;
        Signal->Triggers=2;
      }
      else if(Signal->Triggers==4){

        if(Signal->Polarizacion > 0){
          if(Signal->Discret_Buffer[N_] > (Signal->EOG_Base + abs(Signal->Pulso_Dig)*0.8)){
            Signal->Triggers=8;
            Signal->Remontada++;
          }else  Signal->Triggers=5;
        } else {
          if(Signal->Discret_Buffer[N_] < (Signal->EOG_Base - abs(Signal->Pulso_Dig)*0.8)){
            Signal->Triggers=8;
            Signal->Remontada++;
          }else  Signal->Triggers=5;
        }
      }
    }
    else if (Actual_Slope > Slope_Umbral && Cambio_Signo < 0 ){
      Signal->Polarizacion = Signal->Discret_Buffer[N_] - Signal->EOG_Base;
```

Archivo "Slope_Cont.c"

```
        Signal->Triggers=2;
      }
      else{
        if (Signal->Triggers==1){}
      }


  }
////////////////////////////////////////////////////////////////////////////
  if(Signal->Triggers==2 || Signal->Triggers==8){

    int V_Peak,n_Peak;
    ////////////////////////////////////////////////////////////////////
    if(Signal->Polarizacion > 0){
      if(Signal->EOG_n_Base >= 0){
          V_Peak = Vector_Max_n(
              Signal->EOG_n_Base      , Signal->Discret_Buffer,
              DigitalPulse_History_Length , &n_Peak      );
        }
        else{
          V_Peak = Vector_Max_n(
              10                      , Signal->Discret_Buffer,
              DigitalPulse_History_Length , &n_Peak      );
        }
      }
    else{
      if(Signal->EOG_n_Base >= 0){
          V_Peak = Vector_Min_n(
              Signal->EOG_n_Base         , Signal->Discret_Buffer,
              DigitalPulse_History_Length , &n_Peak      );
        }
        else{
          V_Peak = Vector_Min_n(
              10                      , Signal->Discret_Buffer,
              DigitalPulse_History_Length , &n_Peak      );
        }
    }
    ////////////////////////////////////////////////////////////////////
    if     (n_Peak == N_ && Signal->Triggers == 2)              {}
    else if(n_Peak <  Signal->EOG_n_Base && Signal->Triggers == 2){Signal->Triggers =7;}
    else if(n_Peak >= Signal->EOG_n_Base && n_Peak < N_  && Signal->Triggers == 2){
```

Archivo "Slope_Cont.c"

```
        if(abs(V_Peak - Signal->EOG_Base) >= Height_Umbral){
          Signal->EOG_Peak=V_Peak;
          Signal->Pulso_Dig = Signal->EOG_Peak - Signal->EOG_Base;
          Signal->EOG_n_Peak= n_Peak+1;
          Signal->Triggers=3;
        }
        else Signal->Triggers=7;
      }
      ////////////////////////////////////////////////////////////////////////
      if(Signal->Triggers==8 && Signal->Remontada==1){
        if(Signal->EOG_n_Peak<=10 && (abs(Signal->EOG_Peak-V_Peak)<300) &&
abs(Signal->EOG_Peak)< Height_Remonte_Umbral) Signal->Triggers=9;
        else{
          if(n_Peak == N_){}
          else if (n_Peak == N_-1){
            if(abs(Signal->EOG_Peak - V_Peak)>300){
              Signal->EOG_Peak = V_Peak;
              Signal->Pulso_Dig = Signal->EOG_Peak - Signal->EOG_Base;
              Signal->EOG_n_Base= n_Peak+1;
              Signal->Triggers=3;
            }
            else if(abs(Signal->EOG_Peak - V_Peak)<300 && abs(Signal-
>EOG_Peak)>Height_Remonte_Umbral){
              if(Signal->EOG_n_Peak <= 5) Signal->Triggers =9;
            }
            else Signal->Triggers =9;
          }
          else{
            Signal->Triggers =3;
            Signal->Remontada=0;
          }
        }
      }
      else if (Signal->Triggers==8 && Signal->Remontada>1) Signal->Triggers =6;

    }
////////////////////////////////////////////////////////////////////////////
    if(Signal->Triggers==5){

      int V_Peak,n_Peak;

      if(Signal->Polarizacion > 0){
```

Archivo "Slope_Cont.c"

```
        if(Signal->EOG_n_Peak >= 0){
            V_Peak = Vector_Min_n(
                Signal->EOG_n_Peak        , Signal->Discret_Buffer,
                DigitalPulse_History_Length , &n_Peak     );
          }
          else{
            V_Peak = Vector_Min_n(
                10                        , Signal->Discret_Buffer,
                DigitalPulse_History_Length , &n_Peak     );
          }
        }
      else{
        if(Signal->EOG_n_Peak >= 0){
            V_Peak = Vector_Max_n(
                Signal->EOG_n_Peak        , Signal->Discret_Buffer,
                DigitalPulse_History_Length , &n_Peak     );
          }
          else{
            V_Peak = Vector_Max_n(
                10                        , Signal->Discret_Buffer,
                DigitalPulse_History_Length , &n_Peak     );
          }
      }

      if (n_Peak == N_){}
      else if(n_Peak > Signal->EOG_n_Peak){
        Signal->EOG_Fin=V_Peak;
        Signal->EOG_n_Fin=n_Peak;
        Signal->Triggers=6;
      }
      else Signal->Triggers=7;
    }
////////////////////////////////////////////////////////////////////////////////
    if(Signal->Triggers==6){
      if (Signal->Deriv_Conv<200) Signal->Triggers=7;
    }
////////////////////////////////////////////////////////////////////////////////
    if(Signal->Triggers==7 || Signal->Triggers==9){
      if(Signal->Triggers==9){
        if(Signal->Anular==0)Signal->Anular=1;
        else                 Signal->Anular=0;
      }
```

Archivo "Slope_Cont.c"

```
        if(Signal->Anular==0){
          Signal->Triggers       =0;
          Signal->Slope_Sign_Init =0;
          Signal->Slope_Sign_Final=0;
          Signal->Idle           =0;
          Signal->Offset         =0;
          Signal->Remontada      =0;
          Signal->Anular         =0;

          Signal->Polarizacion    =0;
          Signal->EOG_Base        =0;
          Signal->EOG_n_Base      =0;
          Signal->EOG_Peak        =0;
          Signal->EOG_n_Peak      =0;
          Signal->EOG_Fin         =0;
          Signal->EOG_n_Fin       =0;

          Signal->Pulso_Dig       =0;
        }
    }
////////////DESPLAZAMIENTO DE INDICEL DEL PULSO//////////////////////////////////
    if(Signal->Triggers==1){
       Signal->EOG_n_Base--;
    }
    else if(Signal->Triggers==2){
       Signal->EOG_n_Base--;
    }
    else if(Signal->Triggers==3){
       Signal->EOG_n_Base--;
       Signal->EOG_n_Peak--;
    }
    else if(Signal->Triggers==4){
       Signal->EOG_n_Base--;
       Signal->EOG_n_Peak--;
    }
     else if(Signal->Triggers==5){
       Signal->EOG_n_Base--;
       Signal->EOG_n_Peak--;
    }
    else if(Signal->Triggers==8){
       Signal->EOG_n_Base--;
       Signal->EOG_n_Peak--;
```

Archivo "Slope_Cont.c"

```
////////////////////////////////////////////////////////////////////////////
   if(Signal->Eje==2){
      Blink_Counter(Signal);
   }//Para Mandar al contador de picos para el click cuando sea Vertical
}

void Blink_Counter(Slope_Control* Vertical_EOG){
   // Longitud del buffer de derivadas
//    int N_ = DigitalPulse_History_Length; // Equivalente a `length(Deriv_Buffer)`

   // Condición inicial para Blink_Secuencia
   if (Vertical_EOG->Triggers == 6 && Vertical_EOG->Pulso_Dig > 0 && Vertical_EOG->Blink_Secuencia == 0) {
      Vertical_EOG->Blink_Secuencia = 1;
   } else if (Vertical_EOG->Triggers != 6 && Vertical_EOG->Pulso_Dig == 0 && Vertical_EOG->Blink_Secuencia != 0) {
      Vertical_EOG->Blink_Secuencia = 3;
   }

   // Secuencia 1
   if (Vertical_EOG->Blink_Secuencia == 1) {
      if (Vertical_EOG->Blink_Contador == 0) {
         Vertical_EOG->Blink_Contador = 1;
      }

      if (Vertical_EOG->Deriv_Buffer[N_] > Blink_Counter_Umbral && Vertical_EOG->Actual_Slope_Sign > 0) {
         Vertical_EOG->Blink_Secuencia = 2;
      }
   }

   // Secuencia 2
   if (Vertical_EOG->Blink_Secuencia == 2) {
      int V_peak, n_peak;
      V_peak = Vector_Max_n(
           N_-4         , Vertical_EOG->Deriv_Buffer,
           DigitalPulse_History_Length , &n_peak      );


   }
```

Archivo "Slope_Cont.c"

```
   // Secuencia 3 (Reset)
   if (Vertical_EOG->Blink_Secuencia == 3) {
      Vertical_EOG->Blink_Contador = 0;
      Vertical_EOG->Blink_Secuencia = 0;
   }
}



void Signal_Comparation(Slope_Control* Chan_H,Slope_Control* Chan_V,Channel_Comp*
Result){
   if(Chan_H -> Pulso_Dig == 0 && Chan_V -> Pulso_Dig == 0){
      if (Result->Triggers==1){

         if(Result->H_Duration>0){}
         else{
            Result->H_Duration = Result->H_History[0];
            Result->H_Height   = Result->H_History[1];
            Result->H_Anulado  = Result->H_History[2];
         }

         if(Result->V_Duration>0){}
         else{
            Result->V_Duration  = Result->V_History[0];
            Result->V_Height    = Result->V_History[1];
            Result->V_Anulado   = Result->V_History[2];
            Result->N_Parpadeos = Result->V_History[3];
         }

         if(Result->H_Duration>0||Result->V_Duration>0){
            Result->Triggers=2;
         }

      }
      else if(Result->Triggers==0){
         Result->H_Duration=0;
         Result->V_Duration=0;

         Result->H_Height=0;
         Result->V_Height=0;

         Result->H_Anulado=0;
```

Archivo "Slope_Cont.c"

```
        Result->V_Anulado=0;

        Result->N_Parpadeos=0;

        Result->H_History[0]=0;
        Result->H_History[1]=0;
        Result->H_History[2]=0;

        Result->V_History[0]=0;
        Result->V_History[1]=0;
        Result->V_History[2]=0;
        Result->V_History[3]=0;

    }

}
else{
    Result->Triggers=1;
    ////////////Horizontal/////////////////////////////////////////////////
    if(abs(Chan_H->Pulso_Dig)>0 && Chan_H->Anular == 0 ){
        Result->H_Duration++;
        Result->H_Height=Chan_H->Pulso_Dig;
    }
    else if(abs(Chan_H->Pulso_Dig)>0 && Chan_H->Anular >=1){
        Result->H_Duration=0;
        Result->H_Height=0;
        Result->H_Anulado=1;
    }
        Result->H_Duration=0;
        Result->H_Height  =0;
        Result->H_Anulado =0;
    }
    ////////////Vertical/////////////////////////////////////////////////
    if(abs(Chan_V->Pulso_Dig)>0 && Chan_V->Anular == 0 ){
        Result->V_Duration++;
        Result->V_Height=Chan_V->Pulso_Dig;
        Result->N_Parpadeos=Chan_V->Blink_Contador;
    }
    else if(abs(Chan_V->Pulso_Dig)>0 && Chan_V->Anular >=1){
        Result->V_Duration=0;
        Result->V_Height=0;
        Result->V_Anulado=1;
```

Archivo "Slope_Cont.c"

```
            Result->N_Parpadeos=0;
        }
        else{
            if(Result->V_Duration>0){
                Result->V_History[0]=Result->V_Duration;
                Result->V_History[1]=Result->V_Height;
                Result->V_History[2]=Result->V_Anulado;
                Result->V_History[3]=Result->N_Parpadeos;
            }
            Result->V_Duration  =0;
            Result->V_Height    =0;
            Result->V_Anulado   =0;
            Result->N_Parpadeos =0;
        }

    }

    if (Result->Triggers == 3) Result->Triggers = 4; // Para Delay de 1 ciclo, antes de reiniciar todo.

    if(Result->Triggers == 2){
        Result->H_Diferencia[0]=Result->H_Duration - Result->V_Duration;
        Result->H_Diferencia[1]=abs(Result->H_Height) - abs(Result->V_Height);

        Result->V_Diferencia[0]=Result->V_Duration - Result->H_Duration;
        Result->V_Diferencia[1]=abs(Result->V_Height) - abs(Result->H_Height);

        // Pulso Derecha
        if (Result->H_Height > 200 && Result->H_Duration > 20 && Result->H_Anulado == 0 &&
Result->Triggers == 2) {
            if (Result->H_Diferencia[0] >= 0 && Result->H_Diferencia[1] > 0) {
                Result->Mov_Horizontal = 1;
                Result->Triggers = 3;
            }
        }

        // Pulso Izquierda
        if (Result->H_Height < -200 && Result->H_Duration > 20 && Result->H_Anulado == 0
&& Result->Triggers == 2) {
            if (Result->H_Diferencia[0] >= 0) { // && Result->H_Diferencia[1] > 0
                Result->Mov_Horizontal = -1;
                Result->Triggers = 3;
            }
```

Archivo "Slope_Cont.c"

```
    }

    // Pulso Arriba
    if (Result->V_Height > 200 && Result->V_Duration > 20 && Result->V_Anulado == 0 &&
Result->Triggers == 2) {
        if (Result->V_Diferencia[0] > 10 && Result->V_Diferencia[1] > 0 && Result-
>H_Duration > 20) {
            Result->Mov_Vertical = 1;
            Result->Triggers = 3;
        }
    }

    // Pulso Abajo
    if (Result->V_Height < -200 && Result->V_Duration > 20 && Result->V_Anulado == 0
&& Result->Triggers == 2) {
        if (Result->V_Diferencia[0] > 0) { // && Result->V_Diferencia[1] > 0
            Result->Mov_Vertical = -1;
            Result->Triggers = 3;
        }
    }

    // Pulso Parpadeo
    if (Result->V_Height > 200 && Result->V_Duration > 20 && Result->V_Anulado == 0 &&
Result->Triggers == 2) {
        if (Result->V_Diferencia[0] <= 10 && Result->V_Diferencia[1] > 10) {
            Result->Blink = 1;
            Result->Triggers = 3;
        } else if (Result->V_Diferencia[0] > 10 && Result->V_Diferencia[1] > 10 && Result-
>H_Height == 0) {
            Result->Blink = 1;
            Result->Triggers = 3;
        }

        if (Result->Blink == 1){
            if(Result->N_Parpadeos==1){
                Result->Blink     = 1;
                Result->Click_Izq = 0;
                Result->Click_Der = 0;
            }
            if(Result->N_Parpadeos==2){
                Result->Blink     = 0;
                Result->Click_Izq = 3;
```

Archivo "Slope_Cont.c"

```
                Result->Click_Der = 0;
            }
            if(Result->N_Parpadeos==3){
                Result->Blink     = 0;
                Result->Click_Izq = 0;
                Result->Click_Der = 3;
            }
        }


    }

    // Pulso Indeterminado
    if (Result->Triggers == 2) {
        Result->Indeterminado=1;
        Result->Triggers = 3; // Marcar como evaluado
    }
}

    // Reiniciar Triggers si fue evaluado Reiniciar Todo?
if (Result->Triggers == 4) {
    Result->Mov_Horizontal = 0;
    Result->Mov_Vertical   = 0;

    Result->Blink         = 0;
    Result->Click_Izq     = 0;
    Result->Click_Der     = 0;

    Result->Indeterminado  = 0;

    Result->Triggers      = 0;
}

}


void Mouse_Move_Order  (Channel_Comp* Comparacion){

// Actualiza y trunca Orden_Eje_H
Comparacion->Orden_Eje_H += Comparacion->Mov_Horizontal;
if (Comparacion->Orden_Eje_H >  1) Comparacion->Orden_Eje_H = 1;
if (Comparacion->Orden_Eje_H < -1) Comparacion->Orden_Eje_H = -1;
```

Archivo "Slope_Cont.c"

```
// Actualiza y trunca Orden_Eje_V
Comparacion->Orden_Eje_V += Comparacion->Mov_Vertical;
if (Comparacion->Orden_Eje_V >  1) Comparacion->Orden_Eje_V = 1;
if (Comparacion->Orden_Eje_V < -1) Comparacion->Orden_Eje_V = -1;


Comparacion->Orden_Click_L += Comparacion->Click_Izq;
if(Comparacion->Orden_Click_L > 0) Comparacion->Orden_Click_L--;
Comparacion->Orden_Click_R += Comparacion->Click_Der;
if(Comparacion->Orden_Click_R > 0) Comparacion->Orden_Click_R--;



switch (Comparacion->Orden_Eje_H){
   case H_LEFT:
      MOVE_H_LEFT=1;
      MOVE_H_RIGHT=0;
      break;
   case H_RIGHT:
      MOVE_H_LEFT=0;
      MOVE_H_RIGHT=1;
      break;

   case H_CENTER:
      MOVE_H_LEFT=0;
      MOVE_H_RIGHT=0;
      break;
   default:
      MOVE_H_LEFT=0;
      MOVE_H_RIGHT=0;
      break;
}

switch (Comparacion->Orden_Eje_V){
   case V_DOWN:
      MOVE_V_DOWN=1;
      MOVE_V_UP=0;
      break;
   case V_UP:
      MOVE_V_DOWN=0;
      MOVE_V_UP=1;
```

Archivo "Slope_Cont.c"

```
            break;

        case V_CENTER:
            MOVE_V_DOWN=0;
            MOVE_V_UP=0;
            break;
        default:
            MOVE_V_DOWN=0;
            MOVE_V_UP=0;
            break;
    }

    if(Comparacion->Orden_Click_L>0)          PRESS_CLICK_L=1;
    else if (Comparacion->Orden_Click_L==0)     PRESS_CLICK_L=0;

    if(Comparacion->Orden_Click_R>0)          PRESS_CLICK_R=1;
    else if (Comparacion->Orden_Click_R==0)     PRESS_CLICK_R=0;
}
```

Con los archivos para controlar los modulos del dsPIC y los que definen el algoritmo para procesar la señal EOG, se procede a utilizar todo en el archivo "main.c"

Archivo "main.c"

```
#include "Config.h"
#include <dsp.h> //para llamar funciones del DSPIC
            //agregar --library dsp en:Project Properties/XC16/xc16-ld/Additional options:--library dsp


#include <libpic30.h> //Siempre agregar esta lib, creo que para el delay

#include <stdio.h>
#include <stdlib.h>
#include <stdint.h>

////////////////////////////////////////////////////////////////////////////////////////////////

bool Signal_Processing_Init=false;
//Se inicializa valores iniciales del control del Mouse
                    //{actual,anterior,direccion,flag_1,flag_2,flag_3,flag_4}
Slope_Control Horizontal;
Slope_Control Vertical;
```

Archivo "main.c"

```
Channel_Comp Comparacion_H_V;

//     int32_t resultado;
//////////////////FILTRO FIR/////////////////////////////////////////////////

extern FIRStruct FIR1_Struct;
extern FIRStruct FIR2_Struct;



volatile fractional FilterOut;
volatile fractional FilterOut2;


fractional  MU_VALUE= Q15(0.0055);
extern   FIRStruct  FIR_1_LMS;           // Declara la estructura FIR externa
volatile fractional FIR_1_LMS_Output;     // Variable de salida para la señal filtrada

//////////////////////////////////////////////////////////////
volatile uint16_t RecepADC_1=0;
volatile uint16_t RecepADC_2=0;
 bool TMR1_Done=false;
 bool TMR2_Done=false;
void __attribute__ ((interrupt,no_auto_psv)) _T1Interrupt(void){

   AnalogRead_All(&RecepADC_1,&RecepADC_2);//AN13-AN15

   TMR1_Done=true;
   IFS0bits.T1IF = 0;
}
//////////////////////////////////////////////////////////////



void main(void) {

   dsPIC_Init();


   //Inicializamos la memoria del filtro con puros ceros
   FIRDelayInit(&FIR1_Struct);
```

Archivo "main.c"

```
   FIRDelayInit(&FIR2_Struct);

   FIRDelayInit(&FIR_1_LMS);

   while(1){

    while(TMR1_Done){
//     PRENDER_Output_Pin_2;////////////////////////////////////////////////////////////////////////////////////////
       FIR(1,&FilterOut ,&RecepADC_1,&FIR1_Struct);
       FIR(1,&FilterOut2,&RecepADC_2,&FIR2_Struct);
   ////////////////////////////

       FilterOut  = (FilterOut  <= 0) ? 10 : (FilterOut  > 4090 ? 4085 : FilterOut );
       FilterOut2 = (FilterOut2 <= 0) ? 10 : (FilterOut2 > 4090 ? 4085 : FilterOut2);

       Horizontal.Valor_FIR_PB=FilterOut;
       Vertical.Valor_FIR_PB  =FilterOut2;

    ////////////////////////////

       FIRLMS(1,  &FIR_1_LMS_Output,  &Horizontal.Valor_FIR_PB,  &FIR_1_LMS,
&Vertical.Valor_FIR_PB,  MU_VALUE);

       FIR_1_LMS_Output  = (FIR_1_LMS_Output  <= 0) ? 10 : (FIR_1_LMS_Output  > 4090 ?
4085 : FIR_1_LMS_Output );

       Vertical.Valor_FIR_LMS  =Vertical.Valor_FIR_PB  -((int)(FIR_1_LMS_Output))+1000;
   ////////////////////////////

//   APAGAR_Output_Pin_2;////////////////////////////////////////////////////////////////////////////////////////



      TMR1_Done=false;
    }



    if(Signal_Processing_Init==false)
    {
```

Archivo "main.c"

```
Signal_Processing_Init=Rellenar_Buffers_EOG_START_PROGRAM(Horizontal.Valor_FIR_P
B,Vertical.Valor_FIR_LMS);
    }


   while(TMR2_Done && Signal_Processing_Init){
//       while(1){
//      TOGGLE_Output_Pin_1;////////////////////////////////////////////////////////////////////////////////

       Historial_Discreto_Update(&Horizontal,Horizontal.Valor_FIR_PB);
       Delta_Signal(&Horizontal);
       Convolucion_Derivada(&Horizontal);
       Signal_Digitalitation(&Horizontal);

       Historial_Discreto_Update(&Vertical,Vertical.Valor_FIR_LMS);
       Delta_Signal(&Vertical);
       Convolucion_Derivada(&Vertical);
       Signal_Digitalitation(&Vertical);

       Signal_Comparation(&Horizontal,&Vertical,&Comparacion_H_V);
 //
Mouse_Move_Order(&Comparacion_H_V);////////////////////////////////////////////////////////////////////////
/////



       if (get_pin(Input_Pin_1)) //Pin 3 RC1 -> Seleccion de Canal para el DAC
       {   //Channel 1 Seleccionado  (PIN 3 "prendido")
//              PRENDER_LED_DAC_CHAN_1;
              if (get_pin(Function_1)){  // Posicion Pin 1 RC5 -> Pendiente Actual
                 DAC_write(Horizontal.Deriv_Buffer[0],0);
              }
              else if (get_pin(Function_3))// Posicion Pin 2 RC4 -> Pulso Digital
                 DAC_write((int16_t)Horizontal.Pulso_Dig,2000);
              else                  // Posicion CENTREL -> Pulso Analogico
                 DAC_write(Horizontal.Discret_Buffer[0],0);
       }
              else{
            //Channel 2 Seleccionado  (PIN 3 "Apagado")
//              PRENDER_LED_DAC_CHAN_2;
              if (get_pin(Function_1)){  // Posicion Pin 1 RC5 -> Pendiente Actual
```

Archivo "main.c"

```
            DAC_write(Vertical.Deriv_Buffer[0],0);
          }
          else if (get_pin(Function_3))// Posicion Pin 2 RC4 -> Pulso Digital
            DAC_write((int16_t)Vertical.Pulso_Dig,2000);
          else                  // Posicion CENTREL -> Pulso Analogico
            DAC_write(Vertical.Discret_Buffer[0],0);
      }

////////////////////////////////////////////////////////////////

      TMR2_Done=false;
    }

   }


}
```

ANEXO J Hoja de datos del prototipo

PROTOTIPO - MOUSE EOG		
PARAMETRO	**DETALLES**	
Alimentación de voltaje	Cable USB - VBUS	5Vbus
Consumo de corriente máximo	Cable USB	100mA
Regulación de Voltaje por etapas		
	Etapa USB (PIC18F2550)	5Vbus
	Etapa D.P.S. (dsPIC33CH)	3.3V (aislado)
	Etapa adquisición y acondicionamiento de señales EOG	±5V (aislado) (alimentación OPAMP) 1.8V(aislado)(pedestal de señal EOG)
Baterías		No requiere
Comunicación con el Host		Protocolo USB 2.0
Canales de entrada EOG		2
		Horizontal
		Vertical
Transductores/electrodos necesarios		5 und. total
	Canal Horizontal	2 und.
	Canal Vertical	2 und.
	Referencia	1 und.
Eficiencia al reconocer patrones EOG		46.45%
Protección contra choques eléctricos		
	Separación de alimentación	Aislamiento Galvánico B0505
	Separación de comunicación	Optoacoplador PC817
Tipo de procesamiento de señales		Filtrado FIR y reconocimiento de patrones EOG, embebido en chip
Requisitos del Host USB		
Tipo de plataforma compatible		Multiplataforma
		Android (Teléfono, tablet, TV, etc.)
		Windows (PC, laptop)
		macOS (PC, laptop)
		etc.
Software de operación adicional		No requiere (Plug and play)
Hardware adicional necesario		Cable USB Tipo A a Tipo B
		Conector para electrodos, con 5 terminales.
Hardware mínimo del Host		Soporte de USB 2.0
	Conector USB tipo A	1 puerto

ETAPA ADQUISICION Y ACONDICIONAMIENTO DE SEÑALES		
Etapa de adquisición (1ra. Etapa)	Por canal	INA121
Etapa de acondicionamiento (2da. Etapa)	Por canal	AD8628
Impedancia de entrada	Canal 1 (Horizontal)	1TOhm
	Canal 2 (Vertical)	1TOhm
Ganancia	Canal 1 (Horizontal)	
	1ra Etapa	500
	2da Etapa	10
	TOTAL	5000
	Canal 2 (Vertical)	
	1ra Etapa	500
	2da Etapa	10
	TOTAL	5000
Ancho de banda de la señal EOG		0-50Hz

ETAPA PROCESAMIENTO DE SEÑALES		
Tiempo de muestreo		200us, f_s=5kHz
Frecuencia de Nyquist		f_{nyq}=2.5kHz
Orden de filtro FIR		250
	Canal 1 (Horizontal)	250 (251 coeficientes-taps)
	Canal 2 (Vertical)	250 (251 coeficientes-taps)
Retardo de grupo	Canal 1,2	125 muestras 25ms
Respuesta de fase	Canal 1,2	Fase lineal
Tipo de ventana	Canal 1,2	Hamming
Tipo de filtro FIR	Canal 1,2	Pasa bajos
Frecuencia de corte	Canal 1,2	10 Hz
Banda de transición	Canal 1,2	40 Hz
Atenuación en banda de rechazo	Canal 1,2	60 dB
Precision de coeficientes	Canal 1,2	16 bits
Resolución de la señal	Canal 1,2	12 bits

ANEXO K 86Selección de OPAM para la etapa de adquisición y acondicionamiento de señales

Como parte del diseño del prototipo, es importante seleccionar los Amplificadores Operacionales que tengan las características que mejor se adapten a las señales EOG, para tal propósito se compara las características eléctricas más importantes de diferentes modelos de OPAMP.

Se debe tener en cuenta que la señal EOG tiene un rango de frecuencias de 0 a 50 HZ, por lo que la propiedad de ancho de banda y slew rate, no deben ser grandes necesariamente, se tomara en consideración parámetros que permitan proteger a la señal contra ruido eléctrico.

ETAPA DE ADQUISICION DE SEÑALES EOG (1ra. Etapa)

PARAMETRO	AD620 (Analog Devices)	INA121 (Texas Instruments)	INA128 (Texas Instruments)	AD8429 (Analog Devices)
Tecnología de entrada	BJT (bipolar)	FET (JFET)	FET (JFET)	CMOS (chopper, rail-to-rail)
Rango de alimentación	±2.3 V a ±18 V	±2.25 V a ±18 V	±2.25 V a ±18 V	±1.35 V a ±18 V
Consumo típico (mA)	~1.3 mA	~375 µA	~750 µA	~3.3 mA
Impedancia de entrada	~10^9 Ω || 2 pF	~10^{12} Ω (1 TΩ)	~10^{10} Ω || 2 pF	~10^9 Ω || 2 pF
Input Offset Voltage (µV típ.)	típ. 50 µV	típ. 250 µV	típ. 50 µV	típ. 50 µV
Deriva Offset (µV/°C)	~0.6 µV/°C	~2.5 µV/°C	~0.5 µV/°C	~0.02 µV/°C
Ruido de voltaje (nV/√Hz @1kHz)	~9 nV/√Hz	~35 nV/√Hz	~8 nV/√Hz	~1 nV/√Hz
CMRR (dB @ Gain ≥ 10)	~100 dB	~106 dB	~120 dB	~140 dB
Slew Rate (V/µs)	0.3 V/µs	0.1 V/µs	0.2 V/µs	55 V/µs
BW @ Ganancia baja (kHz)	120 kHz	560 kHz	700 kHz	15 MHz
Input Bias Current	típ. 1 nA	típ. 4 pA	típ. 2 nA	típ. 200 pA
Ganancia (con R externo)	1 – 1000	5 – 10,000	1 – 10,000	1 – 1000

El OPAMP INA121, es el mas adecuado por las siguientes razones. En primer lugar, tiene el parámetro “Input Bias Current” más bajo de todos, esto es ideal para el manejo de bioseñales como las de EOG. en segundo lugar, los parámetros como CMRR, si bien no es

de los mejores, es lo suficiente como para proteger las señales EOG, el ruido que pueda pasar será eliminado con el filtro FIR, por ultimo se elije este OPAMP ya que su stock en tiendas virtuales es amplio, y además, es económico.

ETAPA DE ACONDICIONAMIENTO DE SEÑALES EOG (2da. Etapa)

PARAMETRO	LM324	LT6018 (Analog Devices)	AS2333 (Diodes Inc.)	AD8628 (Analog Devices)
Tecnología	Bipolar	Precision Op Amp	Zero-drift, Chopper-stabilized	Zero-drift CMOS (chopper)
# de Op-Amps por chip	4	1	2	1
Rango de alimentación	3V – 32V (o ±1.5V – ±16V)	3.5V – 60V (o ±1.75V – ±30V)	1.8V – 5.5V	2.7 – 5.5 V
Consumo típico (Iq)	0.7 mA por amp	~1.5 mA	~110 µA	~1 mA
Impedancia de entrada	~2 MΩ	~10 GΩ	~100 MΩ	~10 GΩ
Offset de entrada (Vos)	típ. 2 mV (máx. 7 mV)	típ. 50 µV (máx. 250 µV)	típ. 8 µV (máx. 30 µV)	típ. 1 µV (máx. 10 µV)
Deriva de offset (dVos/dT)	~7 µV/°C	~0.3 µV/°C	~0.02 µV/°C	~0.002 µV/°C
Ruido de entrada (en @1 kHz)	~40 nV/√Hz	~14 nV/√Hz	~22 nV/√Hz	~22 nV/√Hz
CMRR (Rechazo modo común)	~70 dB	126 dB	120 dB	130 dB
Slew Rate	0.5 V/µs	13 V/µs	0.1 V/µs	1 V/µs
Ancho de banda (GBW)	~1 MHz	15 MHz	400 kHz	2.5 MHz
Input Bias Current (Ib)	típ. 20 – 100 nA	típ. 7 nA	típ. 1 pA	típ. 1 pA
Aplicaciones típicas	Filtros activos, comparadores de uso general	Instrumentación, adquisición de datos, sensores de precisión	Equipos médicos portátiles, ECG/EEG, sensores de muy baja señal	Instrumentación biomédica, sensores de precisión

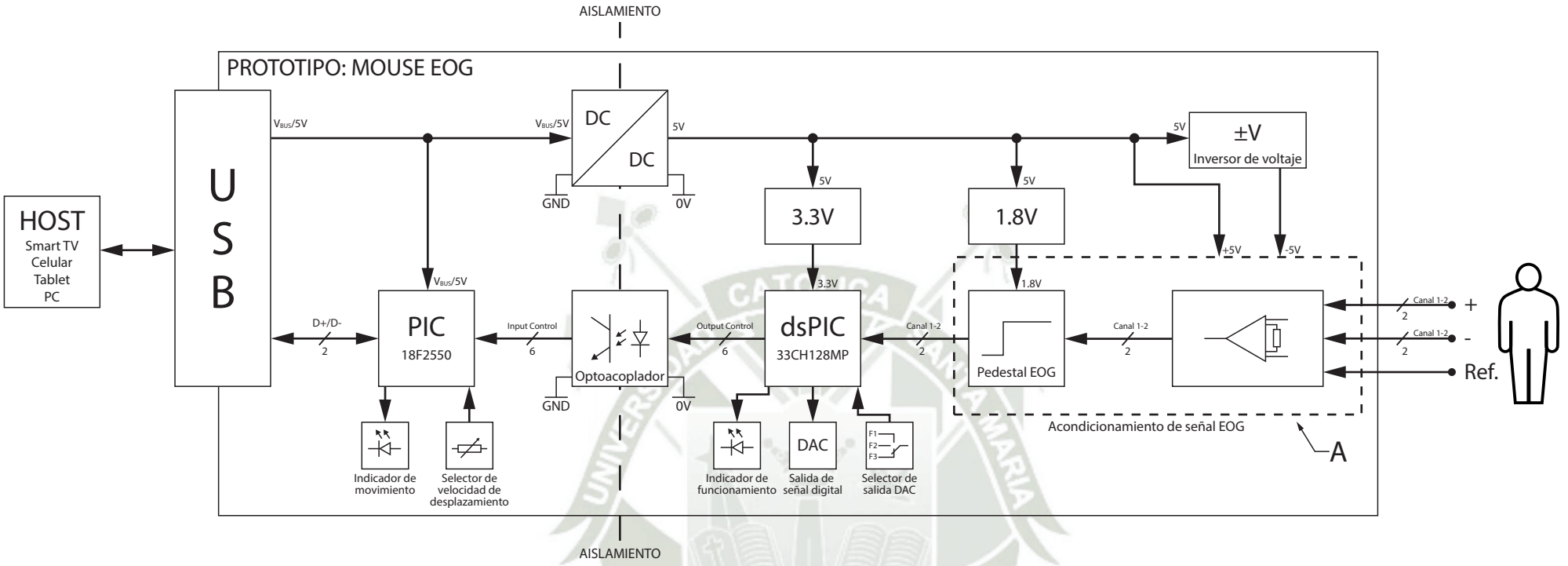
Para la etapa de acondicionamiento de señales, elegimos el OPAMP AD8628 debido a que, si bien es un OPAMP de propósito general, sus características están mas perfiladas a aplicaciones como el manejo de señales biomédica, adicionalmente, las tiendas virtuales tienen stock constante de este amplificador.

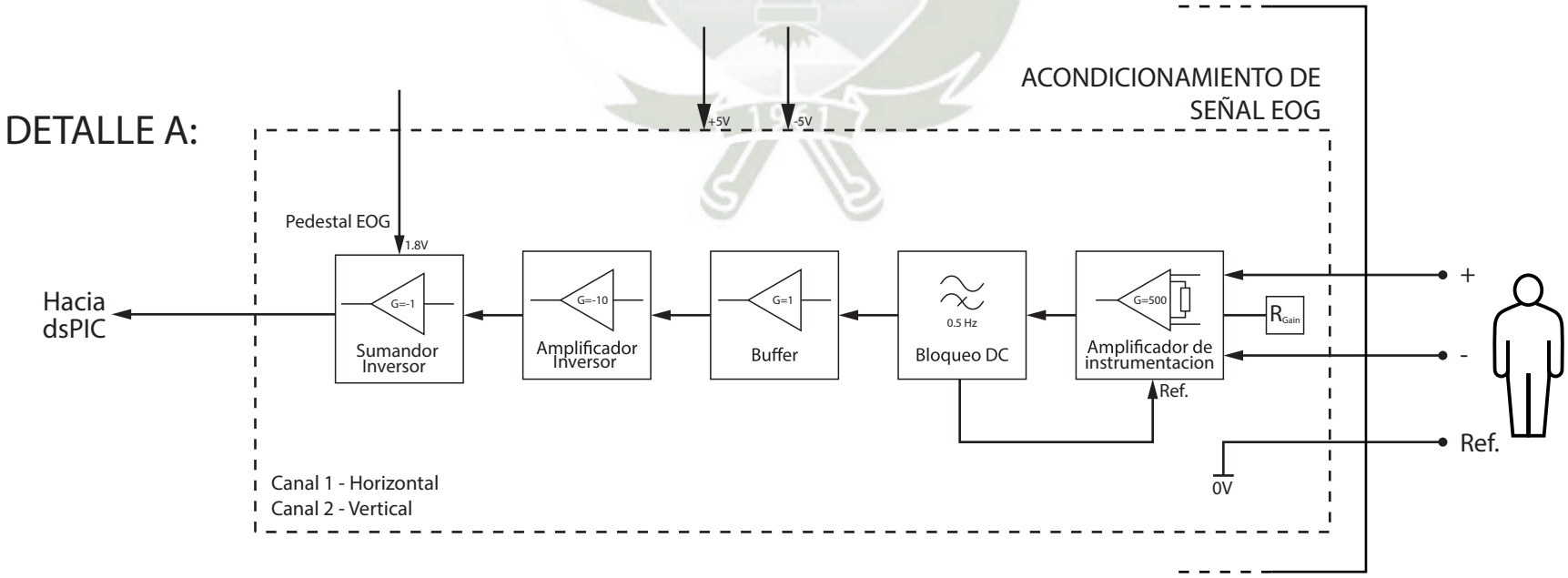
Como resumen, se adjunta una tabla de parámetros eléctricos de los dos amplificadores operacionales seleccionados.

	1RA. ETAPA	2DA. ETAPA
PARAMETRO	**INA121 (Texas Instruments)**	**AD8628 (Analog Devices)**
Tecnología	FET (JFET)	Zero-drift CMOS (chopper)
# de Op-Amps por chip	1	1
Rango de alimentación	±2.25 V a ±18 V	2.7 – 5.5 V
Consumo típico (Iq)	~375 µA	~1 mA
Impedancia de entrada	~10^{12} Ω (1 TΩ)	~10 GΩ
Offset de entrada (Vos)	típ. 250 µV	típ. 1 µV (máx. 10 µV)
Deriva de offset	~2.5 µV/°C	~0.002 µV/°C
Ruido de entrada (en @1 kHz)	~35 nV/√Hz	~22 nV/√Hz
CMRR (Rechazo modo común)	~106 dB	130 dB
Slew Rate	0.1 V/µs	1 V/µs
Ancho de banda (GBW)	560 kHz	2.5 MHz
Input Bias Current (Ib)	típ. 4 pA	típ. 1 pA
Ganancia (con R externo)	5 – 10,000	-

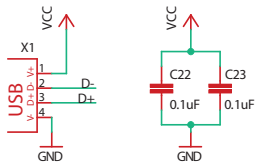
ANEXO L Diseño de PCB

El diseño del circuito empleado en este prototipo pasó por múltiples iteraciones, culminando en la versión final que se expone en las siguientes páginas.

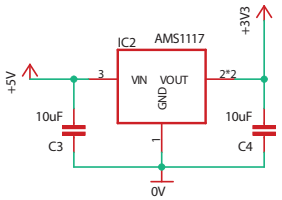

AISLAMIENTO
PROTOTIPO: MOUSE EOG
HOST
Smart TV
Celular
Tablet
PC
USB
VBUS/5V
DC
DC
5V
GND
0V
3.3V
1.8V
±V
Inversor de voltaje
+5V
-5V
PIC
18F2550
D+/D-
2
Input Control
6
Optoacoplador
Output Control
6
dsPIC
33CH128MP
Canal 1-2
2
Pedestal EOG
Canal 1-2
2
Canal 1-2
2
+
Canal 1-2
2
-
Ref.
Indicador de movimiento
Selector de velocidad de desplazamiento
Indicador de funcionamiento
DAC
Salida de señal digital
F1
F2
F3
Selector de salida DAC
Acondicionamiento de señal EOG
A
AISLAMIENTO


DETALLE A:
ACONDICIONAMIENTO DE SEÑAL EOG
+5V
-5V
Pedestal EOG
1.8V
Hacia dsPIC
G=-1
Sumandor Inversor
G=-10
Amplificador Inversor
G=1
Buffer
0.5 Hz
Bloqueo DC
G=500
Amplificador de instrumentacion
RGain
Ref.
0V
+
-
Ref.
Canal 1 - Horizontal
Canal 2 - Vertical

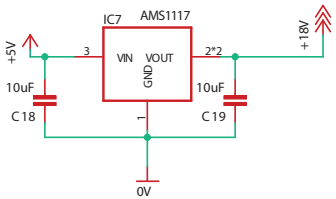
ALIMENTACIÓN Y COMUNICACIÓN USB



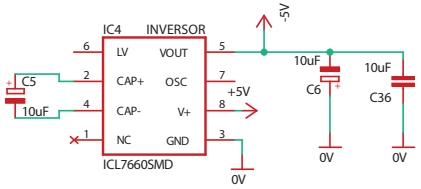
ALIMENTACIÓN dsPIC 3.3V



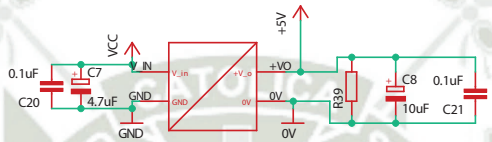
PEDESTAL EOG 1.8V



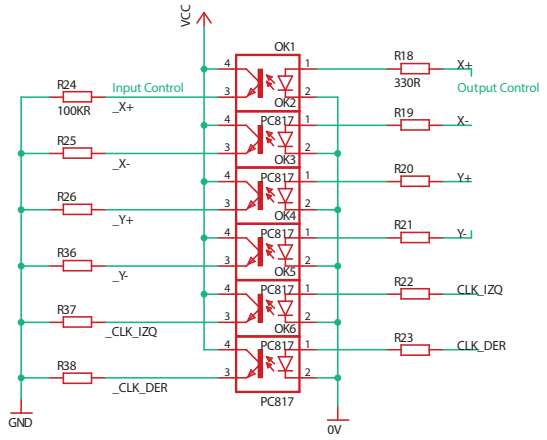
ALIMENTACIÓN SIMÉTRICA OPAMPS ±5V



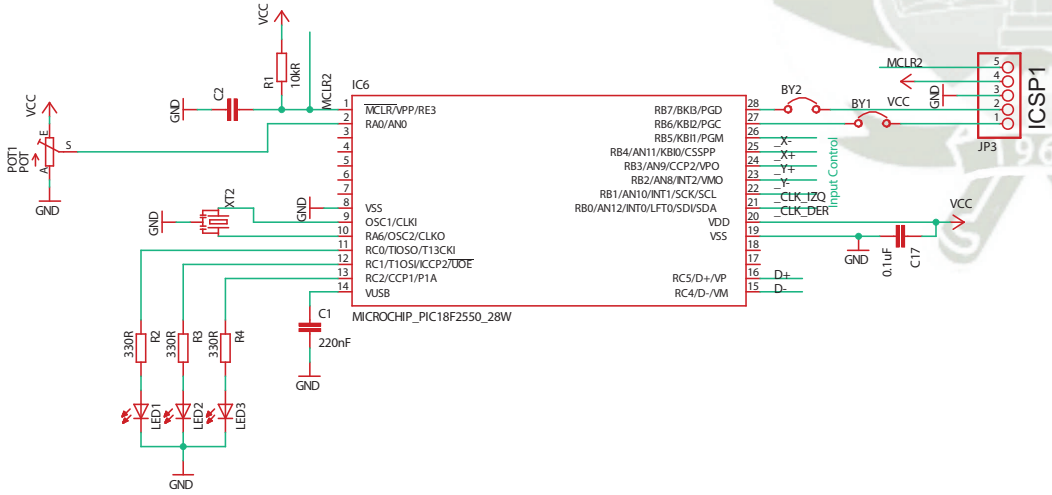
AISLAMIENTO DE ALIMENTACIÓN USB PIC - dsPIC EOG



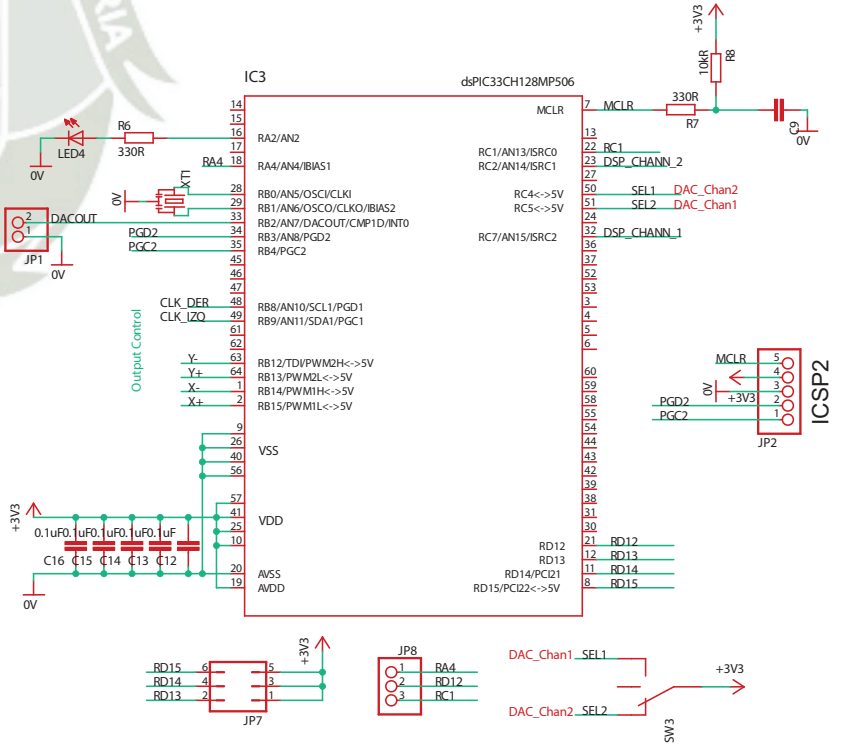
AISLAMIENTO DE COMUNICACIÓN PIC - dsPIC



PIC18F2550

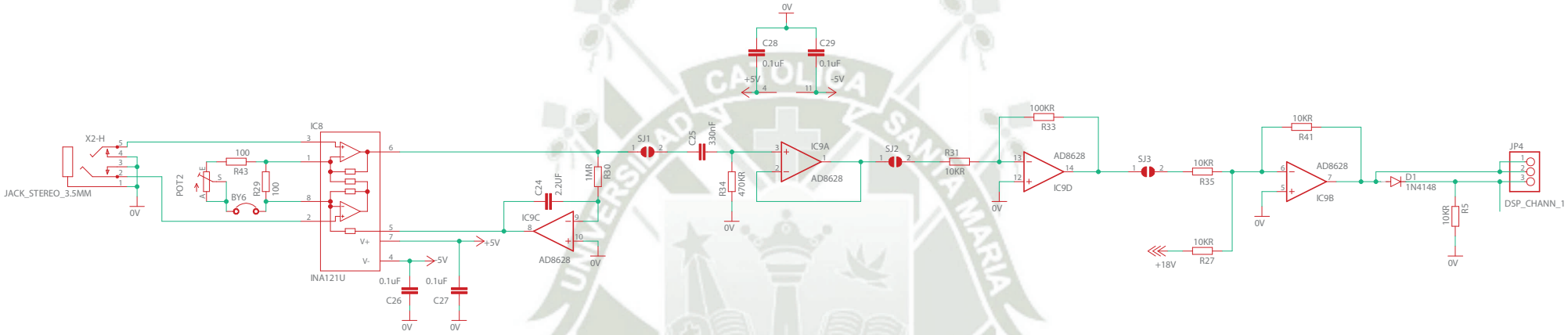


dsPIC33CH128MP506

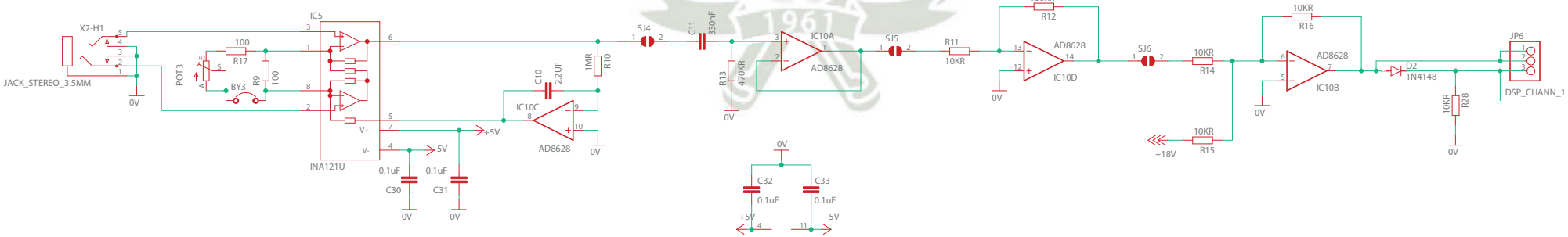


ACONDICIONAMIENTO DE SEÑAL EOG

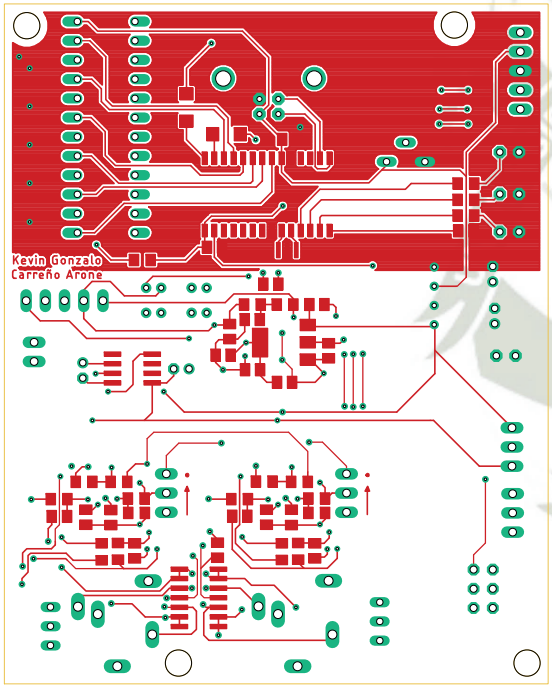
CANAL EOG 1 - HORIZONTAL



CANAL EOG 2 - VERTICAL

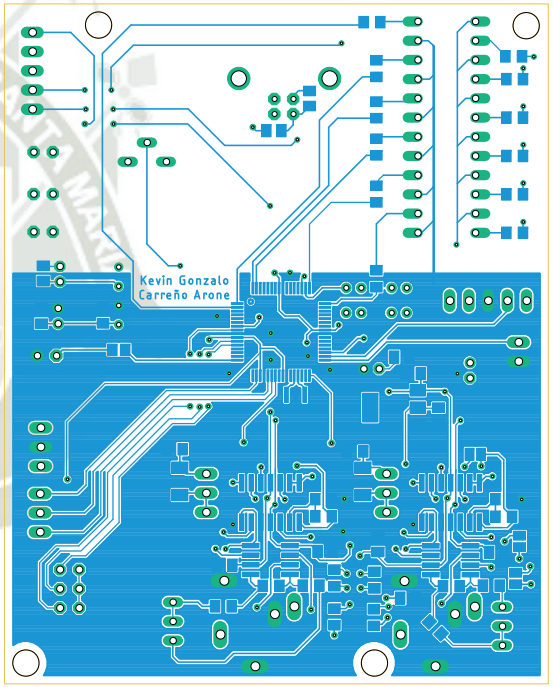


VISTA SUPERIOR



ESCALA 1:1

VISTA INFERIOR



ESCALA 1:1