Proyecto Final Bioingeniería

Diseño de un prototipo de monitor cardíaco precordial monocanal portátil para evaluación electrocardiográfica básica en forma rápida.

Tutor: Ing. Axel Mancino

Alumno: Gaspar Rodriguez Palacios

Índice

Sección:	Página:
1. Glosario	Pg. 3
2. Introducción	Pg. 4-15
2.1. Anatomía y fisiología cardíaca	Pg. 4-6
2.2. El electrocardiograma (ECG)	Pg. 6-13
2.2.1. Derivaciones del ECG	Pg. 7-8
2.2.2. Parámetros del ECG	Pg. 8-10
2.2.3. Adaptación de la señal de ECG	Pg. 10-13
2.3. Propuesta de trabajo	Pg. 13-15
2.3.1. Problemática actual y propuesta	Pg. 13-14
2.3.2. Objetivo general	Pg. 14
2.3.3. Objetivos específicos	Pg. 14-15
3. Desarrollo	Pg. 16-67
3.1. Electrodos	Pg. 19-20
3.2. Amplificador de instrumentación	Pg. 21-22
3.3. Filtros analógicos	Pg. 22-37
3.3.1. Pasa bajos	Pg. 23-35
3.3.2. Pasa altos	Pg. 35-36
3.3.3. Implementación y medición en *protoboard*	Pg. 36-37
3.4. Etapa de amplificación	Pg. 37-38
3.5. Circuito de alimentación	Pg. 38-39
3.6. Conversión digital y visualización	Pg. 39-40
3.7. Integración de la placa final	Pg. 41-49
3.7.1. Diseño e impresión de placa PCB	Pg. 41-47
3.7.2. Pruebas de la placa integrada	Pg. 47-48
3.7.3. Pantalla LCD	Pg.48-49
3.8. Implementación en Arduino	Pg. 49-63
3.8.1. Sistema de adquisición	Pg. 50-51
3.8.2. Filtro digital en tiempo real	Pg. 51-63
3.9. Diseño e impresión 3D del gabinete	Pg. 64-65
3.10. Ensamblaje final	Pg. 66-67
4. Conclusión	Pg. 68-70
5. Bibliografía	Pg. 71-72
6. Anexos	Pg. 73-94

1. Glosario

- PCB: *Printed Circuit Board* o circuito impreso en una placa
- ECG: Electrocardiograma o electrocardiógrafo
- CMRR: *Common mode rejection ratio* o rechazo al modo común
- QUCS: Quite Universal Simulator, software dedicado a simulación de circuitos
- IDE: *Integrated development environment* o entorno integrado para el desarrollo de código
- ABS: Acrilonitrilo butadieno estireno
- Conversor A/D: Conversor analógico a digital
- SMD: *Surface-mount device* o dispositivo de montaje en superficie
- SAB: *Single amplifier biquad* o biquadrática con un único amplificador
- IIR: *Infinite impulse response*
- FIR: *Finite impulse response*

2. Introducción

2.1 Anatomía y fisiología cardíaca

El sistema cardiovascular está formado por el corazón, que bombea la sangre a todo el cuerpo, y los vasos sanguíneos, que forman una red cerrada de conductos que transportan la sangre [1]. El corazón se encuentra contenido en el mediastino torácico, un grueso y flexible tabique de partes blandas orientado en sentido longitudinal en posición mediosagital (figura 2.1.1).

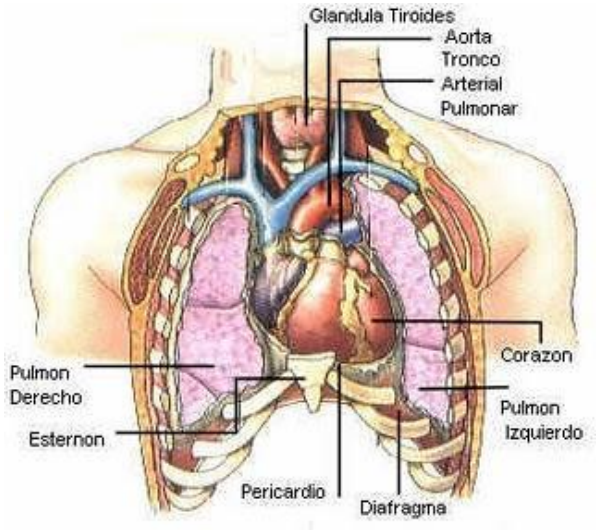


Figura 2.1.1: Posición normal del corazón en la cavidad torácica. [1]

En condiciones normales, las partes del corazón (figura 2.1.2) laten en una secuencia ordenada: la contracción de las aurículas (sístole auricular) va seguida de la contracción de los ventrículos (sístole ventricular) y, durante la diástole, las cuatro cavidades se relajan.

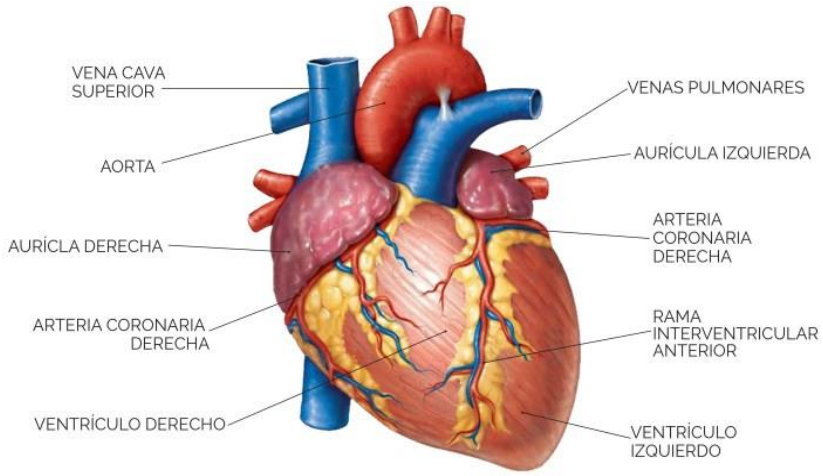


Figura 2.1.2: Partes del corazón [1]

El latido cardíaco se origina en un sistema de conducción cardiaca especializado y se extiende por este sistema a todas las partes de las paredes cardíacas, denominadas miocardio. Las estructuras que conforman el sistema de conducción (figura 2.1.3) son el nodo sinoauricular (nodo SA); las vías auriculares internodales; el nodo auriculoventricular (nodo AV), el haz de His y sus ramas, y el sistema de Purkinje.

Las diversas partes del sistema de conducción y, en condiciones normales, las partes del miocardio, son capaces de emitir una descarga espontánea. Sin embargo, el nodo sinoauricular descarga con más rapidez, con la despolarización que se extiende desde este a las otras regiones antes que éstas emitan descargas espontáneas. Por tanto, dicho nodo es el marcapaso cardíaco normal, su frecuencia de activación determina la frecuencia con la que late el corazón. Los impulsos generados en el nodo sinoauricular pasan por las vías auriculares hasta el nodo auriculoventricular; a través de este último, aquellos van al haz de His y, por las ramas de este, mediante el sistema de Purkinje, hacia el músculo ventricular. [2]

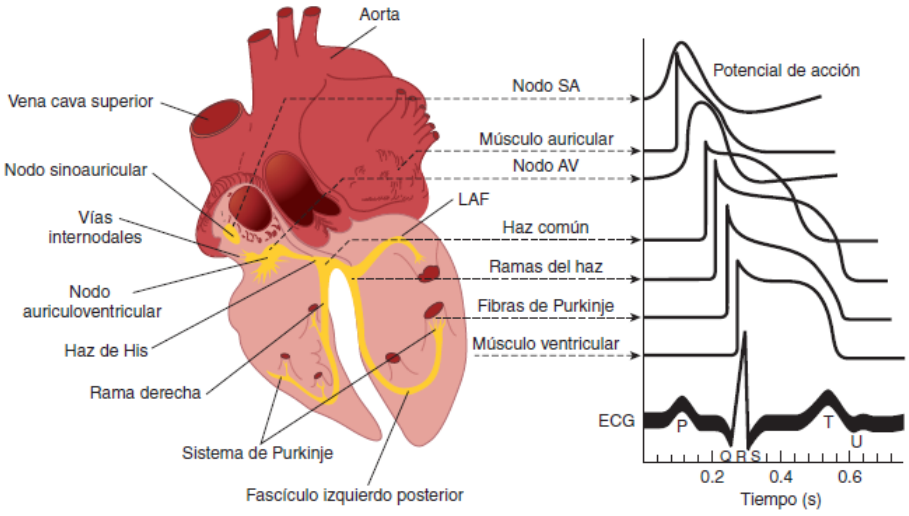


Figura 2.1.3: A la izquierda se representa anatómicamente el corazón humano con un enfoque especial en áreas del sistema de conducción. A la derecha se muestran los potenciales de acción típicos para los nodos sinoauricular y auriculoventricular; otras partes del sistema de conducción y para el músculo cardíaco junto con a relación que guardan con el potencial registrado fuera de la célula, o sea la lectura del electrocardiograma (ECG).[2]

La despolarización iniciada en el nodo sinoauricular se propaga en sentido radial a través de las aurículas y luego converge en el nodo auriculoventricular. La despolarización auricular se completa en 0.1 s. Como la conducción del nodo auriculoventricular es lenta (figura 2.1.4), hay un retraso de 0.1 s (retraso nodal auriculoventricular) antes que la excitación se extienda a los ventrículos. Desde la parte superior del tabique, la onda de despolarización se extiende en las fibras de Purkinje de conducción rápida a todas las regiones de los ventrículos en 0.08 a 0.1 s. En los seres humanos, la despolarización del musculo ventricular comienza del lado izquierdo del tabique interventricular y se desplaza primero a la derecha a través de la parte media del tabique;

luego, la onda de despolarización se disemina por el tabique hasta la punta del corazón. Regresa por las paredes ventriculares a la hendidura auriculoventricular, y se dirige de la superficie endocárdica a la epicárdica (figura 2.2.1). Las ultimas partes del corazón en despolarizarse son la parte posterobasal del ventrículo izquierdo, el cono pulmonar y la parte superior del tabique.

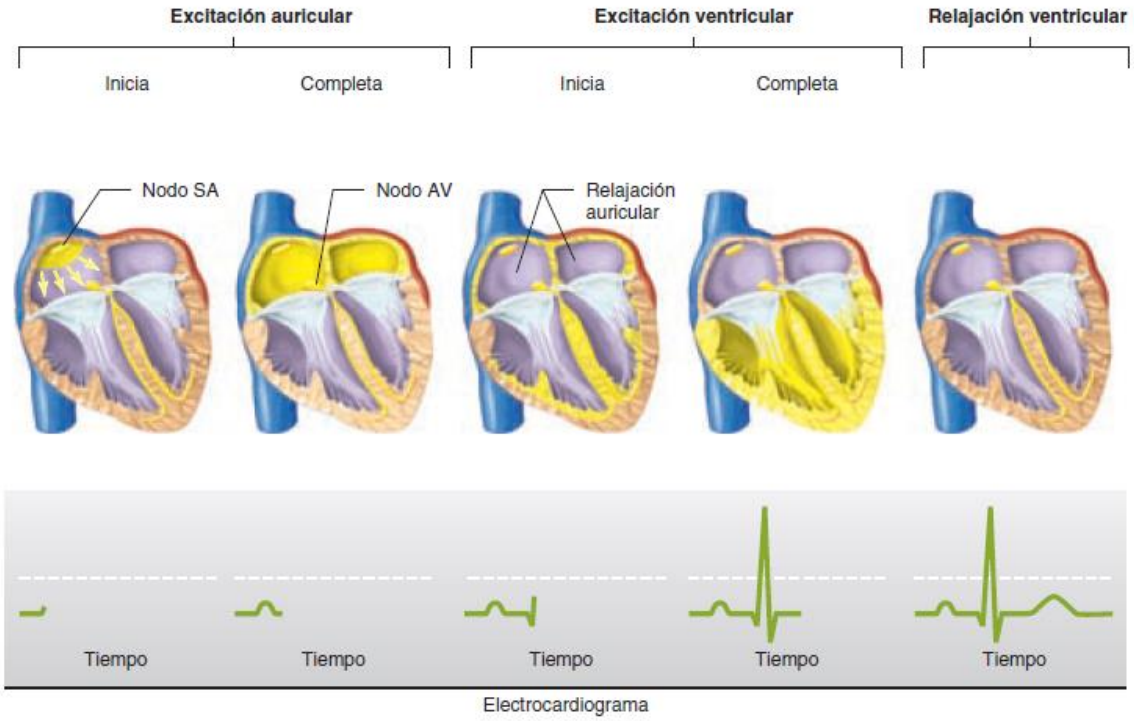
Tejido	Velocidad de conducción (m/s)
Nodo sinoauricular	0.05
Vías auriculares	1
Nodo auriculoventricular	0.05
Haz de His	1
Sistema de Purkinje	4
Músculo ventricular	1

Figura 2.1.4: Velocidad de conducción en el tejido cardíaco [2]

2.2 El electrocardiograma (ECG)

Al ser los líquidos corporales buenos conductores eléctricos, las fluctuaciones en el potencial eléctrico del corazón pueden registrarse fuera de las células como la suma algebraica de los potenciales de acción de las fibras del miocardio.

El registro de las fluctuaciones de este potencial eléctrico durante el ciclo cardíaco es el electrocardiograma (ECG). Este estudio, como plasma la figura 2.2.1, permite observar la evolución de la despolarización de las fibras del miocardio en el tiempo.



Electrocardiograma

Figura 2.2.1: Propagación normal de la señal eléctrica del corazón. El color amarillo en la figura del corazón indica la despolarización de esa zona. Abajo variación del electrocardiograma correspondiente a cada etapa del latido.[2]

La creación del ECG entre los años 1895 y 1903, es atribuida al Holandés Willem Einthoven, ganador del premio Nobel en 1924 por este hallazgo. En la actualidad este método de diagnóstico es el más utilizado en la medicina para la detección de condiciones cardíacas [3]. El ECG mide y grafica la actividad eléctrica del corazón, que se ve alterada por la morfología de cada individuo, e incluso por posibles patologías. Para medir esta actividad eléctrica, una señal de solo algunos *millivolts* y con muchas fuentes de ruido, se colocan electrodos en diversas configuraciones dependiendo de la complejidad del estudio y se representan gráficamente los resultados de las derivaciones medidas.

2.2.1 Derivaciones del ECG

Entre las distintas configuraciones en las que pueden colocarse los electrodos para realizar un ECG y así medir la variación del potencial cardíaco, se puede utilizar el registro unipolar que solo usa un electrodo activo, conectado a un electrodo de referencia en potencial cero o se puede obtener un registro bipolar, conectando 2 electrodos activos a uno de referencia. En un conductor de volumen como el cuerpo humano, si se disponen los electrodos en forma de triángulo equilátero con respecto a un centro que actúa como fuente de corriente, la suma de los potenciales en las puntas es cero en todo momento. Para obtener un triángulo equilátero con el corazón en el centro como fuente de corriente se utiliza el triángulo de Einthoven (figura 2.2.1.1), colocando electrodos en ambas extremidades superiores y en el miembro inferior izquierdo, obteniendo así las derivaciones estándar de las extremidades utilizadas en electrocardiografía (Derivaciones I, II y III). Al conectarse estos electrodos entre sí, se obtiene una derivación que permanece en un potencial cercano a cero mientras que la despolarización que se desplaza volumétricamente hacia un electrodo activo registra una desviación positiva y si se aleja volumétricamente registra una desviación negativa.

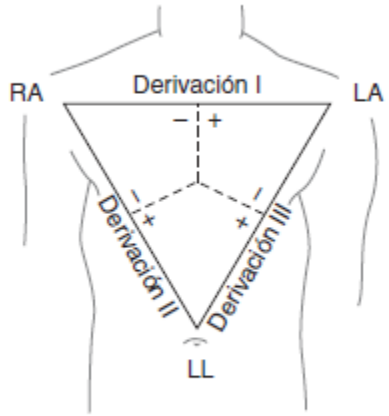


Figura 2.2.1.1: Triángulo de Einthoven con sus derivaciones correspondientes. Las líneas

perpendiculares que parten de cada derivación se intersecan en el centro de la actividad eléctrica en el corazón. LA, extremidad superior izquierda; RA, extremidad superior derecha; LL, extremidad inferior izquierda.[2]

Otra forma común pero más compleja de colocar los electrodos en electrocardiografía clínica para determinados estudios es utilizar registros de electrodos unipolares como muestra la figura 2.2.1.2. Esta configuración utiliza seis derivaciones unipolares torácicas (derivaciones precordiales) designadas V_1-V_6 y tres derivaciones unipolares de extremidades VR (brazo derecho), VL (brazo izquierdo) y VF (pie izquierdo).

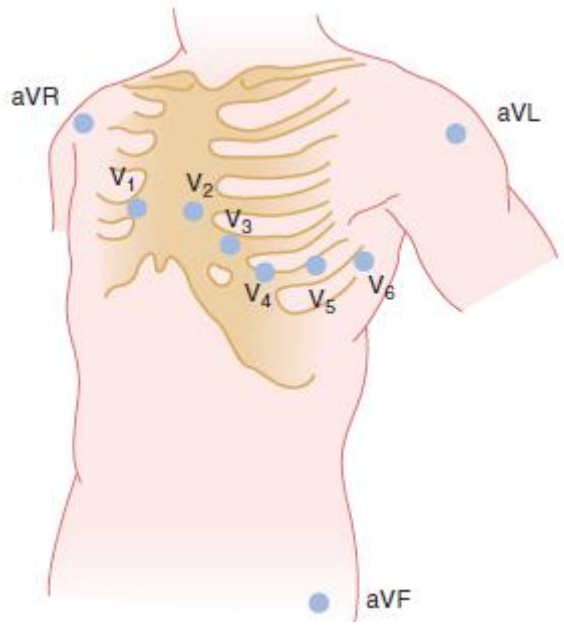


Figura 2.2.1.2: Derivaciones electrocardiográficas unipolares. [2]

2.2.2 Parámetros del ECG

Los parámetros principales que se observan en un ECG son la presencia, duración, y amplitud de los complejos registrados en un individuo normal, como muestra la figura 2.2.2.1. La onda P es producida por la despolarización auricular, el complejo QRS por la despolarización ventricular, la onda T por la repolarización ventricular y la onda U por la repolarización lenta de los músculos papilares. Como referencia de la duración normal y de qué fenómeno representa cada intervalo se incluyó la figura 2.2.2.2.

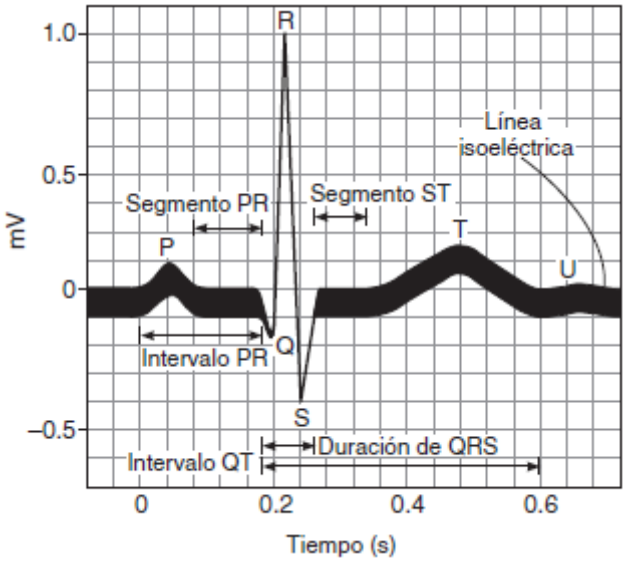


Figura 2.2.2.1: Nombres de los segmentos y complejos normales en un ECG humano.[2]

	Duraciones normales		
Intervalos	**Promedio**	**Intervalo**	**Fenómenos en el corazón durante el intervalo**
Intervalo PR[a]	0.18[b]	0.12-0.20	Despolarización auricular y conducción por el nodo AV
Duración de QRS	0.08	Hasta 0.10	Despolarización ventricular y repolarización auricular
Intervalo QT	0.40	Hasta 0.43	Despolarización ventricular más repolarización ventricular
Intervalo ST (QT menos QRS)	0.32	…	Repolarización ventricular (durante onda T)

[a]Medido desde el inicio de la onda P al inicio del complejo QRS.

[b]Se acorta conforme aumenta la frecuencia cardiaca, de un promedio de 0.18 s a 70 latidos por minuto hasta 0.14 s a 130 latidos por minuto.

AV, auriculoventricular.

Figura 2.2.2.2: Duraciones normales en un entre complejos del ECG.[2]

Observando la escala de la figura 2.2.2.1 y con la información que provee la figura 2.2.2.2, se evidencia que la resolución temporal es un factor a tener en cuenta a la hora del diagnóstico ya que muchos de los eventos importantes que se registran en el ECG suceden en la escala de los milisegundos. Por este motivo, es importante tener en cuenta que la "American Heart Association" [4] recomienda un mínimo de 500 muestras por segundo (una resolución de 0.002 segundos) para electrocardiogramas digitales aunque otros refutan esta recomendación alegando

que con 125 muestras por segundo (0.008 segundos) basta para análisis básicos de la señal cardíaca. [5]

2.2.3 Adaptación de la señal de ECG

Para poder comparar un ECG adquirido con uno normal, tal como el presentado en la figura 2.2.2.1, es necesario adaptar esta señal, amplificarla y filtrar ruidos indeseados.

Fuentes de ruido

En un ECG típico, hay dos grupos de ruidos principales a eliminar: los producidos por el cuerpo y los ambientales. El primer grupo de ruidos incluye interferencias que se dan por la contracción de los músculos cercanos a los electrodos (señales electromiográficas), ruidos de baja frecuencia debidos a la respiración, cambios en la impedancia de los electrodos por la transpiración y ruido de base contínuo debido al movimiento de los electrodos. El segundo grupo de señales de ruido está formado principalmente por el ruido de línea de la alimentación eléctrica, cuya frecuencia fundamental depende de cada país: en países como Argentina y Chile, Europa, Asia y Oceanía es de 50 Hz, mientras que en el resto de América y en Japón se emplean 60 Hz (figura 2.2.3.1). Además, también se deben considerar los armónicos y la interferencia electromagnética que puedan generar equipos eléctricos cercanos. [6]

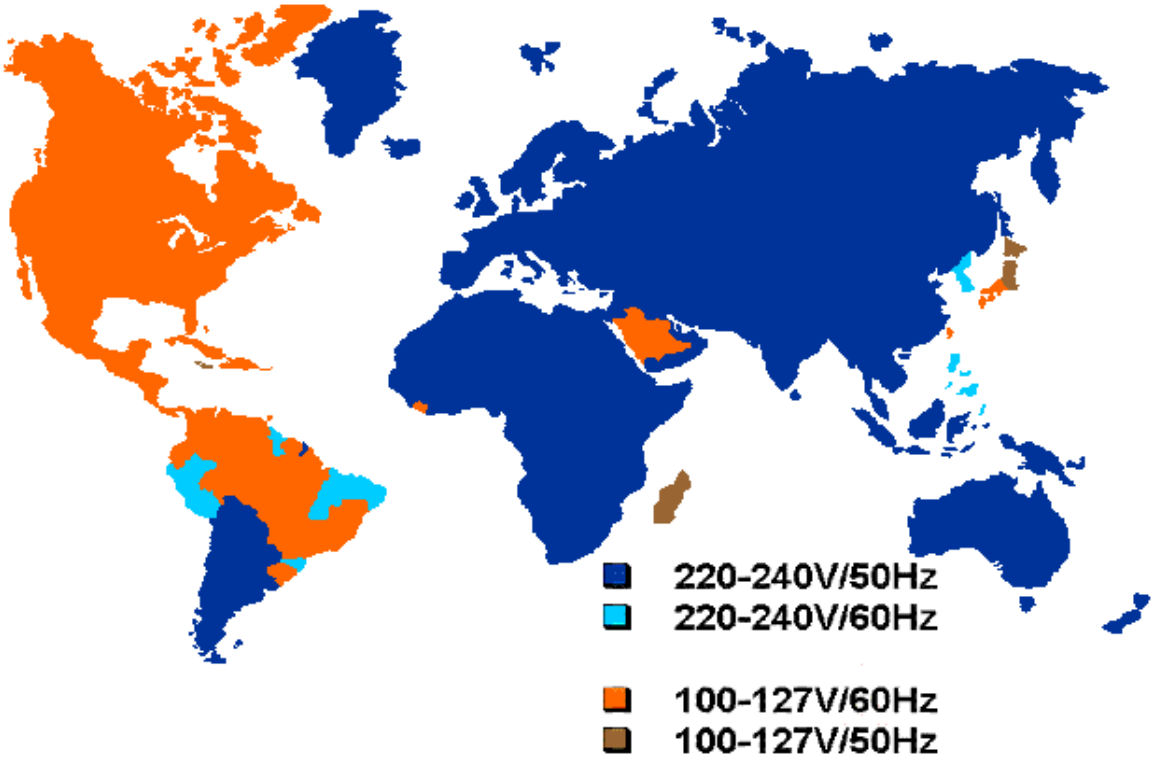


Figura 2.2.3.1: Detalle de los potenciales y las frecuencias de línea y que se utilizan en cada país.[2]

Filtrado

El proceso que permite minimizar el ruido de una señal, no dejando pasar ciertas frecuencias o amplificando otras, es denominado filtrado [7] y puede llevarse a cabo analógica o digitalmente a través de diversas implementaciones. El filtrado es un proceso que requiere caracterizar el ruido que se quiere eliminar de la señal, para ello, debe conocerse cuáles son las frecuencias relevantes de la señal deseada y cuáles sólo contienen ruido. De esta manera, es posible diseñar y aplicar filtros analógicos o digitales que pueden ser utilizados en conjunto, afectando distintas frecuencias, para obtener así una señal limpia de ruido.

Filtrado analógico

Para caracterizar las frecuencias de la señal se buscaron publicaciones académicas acerca de las frecuencias que es importante incluir para llevar a cabo estudios de ECG confiables. En los estudios se encontró información diversa acerca de cuál es la frecuencia de muestreo conveniente para utilizar, luego de analizarla, se concluyó que si se quieren utilizar los datos para estudios avanzados de diagnóstico es necesario utilizar una frecuencia de muestreo de por lo menos 1000Hz [9] pero que si solamente se quiere mostrar la curva de la actividad cardíaca, con muestrear a 125Hz es suficiente. Como situación de compromiso, la "American Heart Association" recomienda muestrear a 500 Hz, frecuencia suficiente para hacer estudios básicos sobre la onda del ECG [5].

Amplificador de instrumentación

Un amplificador de instrumentación es un circuito integrado construido a partir de amplificadores operacionales cuya función es adquirir una señal débil (unos pocos *millivolts)* y amplificarla según un factor variable que puede ser modificado cambiando una única resistencia en el circuito. Con este objetivo, un amplificador de instrumentación debe tener una alta impedancia de entrada para desacoplar la señal y un alto CMRR, ratio que cuantifica la capacidad de un amplificador operacional de rechazar señales que aparecen en ambas entradas con la misma magnitud y fase, para adquirir una señal con la menor cantidad de ruido posible.

En la figura 2.2.3.2 puede observarse el diagrama de un amplificador de instrumentación modelo con sus componentes, todos ellos internos al circuito integrado excepto la resistencia R_{gain} o R_g que es externa para poder modificar el factor de amplificación de la señal.

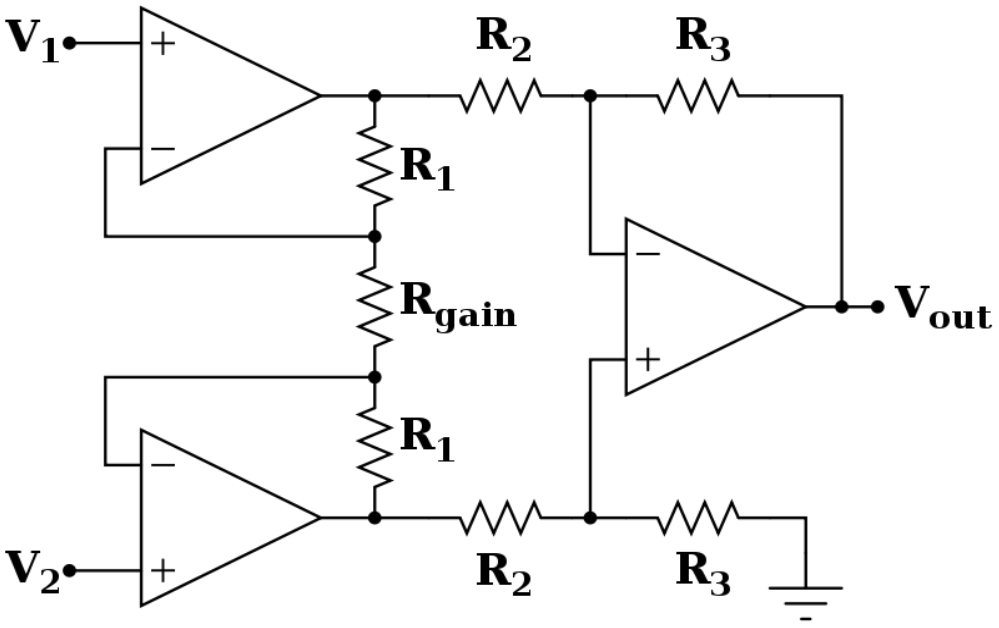


Figura 2.2.3.2: Diagrama del circuito modelo que forma un amplificador de instrumentación.

Al haber realimentación negativa en los amplificadores operacionales, puede considerarse que hay un corto circuito virtual entre las entradas inversoras (-) y no inversoras (+), por lo tanto las tensiones en dichas terminales y en los extremos de R_{gain} serán conocidas. Por lo tanto, a través de esta resistencia circulará una corriente

$$I_g = (V_2 - V_1)\left(\frac{1}{R_g}\right)$$

Debido a la alta impedancia de entrada de los amplificadores operacionales, esa corriente será la misma que atraviesa R_1, siendo así la tensión que cae en toda la rama R_g, R_1 y R_1. Esta será la diferencia de tensión entre la salida de los amplificadores operacionales.

$$V_{intermedia} = \frac{(V_2 - V_1)}{R_g}\left(R_g + 2R_1\right) = (V_2 - V_1)\left(1 + \frac{2R_1}{R_g}\right)$$

Ya que el resto del circuito es un restador de ganancia, la salida será exactamente la diferencia de tensión de sus entradas, definida como

$$V_{out} = (V_2 - V_1)\left(1 + \frac{2R_1}{R_g}\right)\frac{R_3}{R_2}$$

Dada la ecuación anterior, se evidencia el alto CMRR del amplificador ya que, al restar ambas señales en el restador de ganancia, se cancelan las señales que aparecen en ambas entradas. Por otro lado, la tensión de referencia que en la figura 2.2.3.2 se encuentra conectada a tierra puede conectarse a una tensión de referencia distinta.

Como ejemplo de la aplicación de un amplificador de instrumentación para la medición del ECG se proporciona la figura 2.2.3.3 que ilustra una posible configuración del amplificador de

instrumentación AD620 conectado por un lado a los electrodos que obtienen la señal del cuerpo del paciente y por el otro a ciertos filtros que acondicionan la señal obtenida. En la figura se utiliza la pierna derecha del paciente como referencia en vez de la tierra. El circuito de la figura fue utilizado para probar en primera instancia el funcionamiento de los filtros digitales como se comenta más adelante.

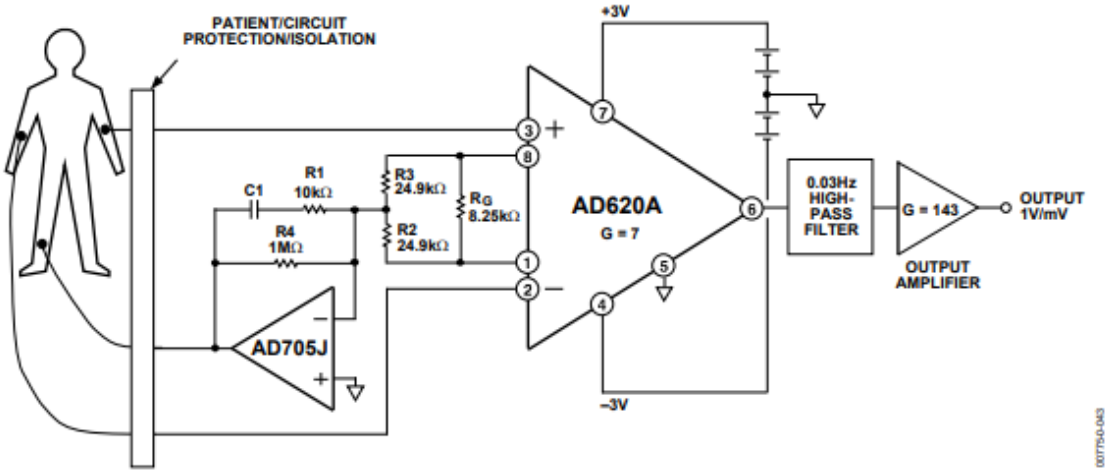


Figure 39. A Medical ECG Monitor Circuit

Figura 2.2.3.3: Posible configuración de un amplificador de instrumentación aplicado a la medición de un ECG.

2.3 Propuesta de trabajo

2.3.1 Problemática actual y propuesta

Para realizar un ECG son necesarios aparatos costosos que por sus dimensiones (alrededor de 500 x 300 x 100 mm) y características solo pueden ser utilizados en ambientes médicos como clínicas y hospitales. Esta limitante hace que mucha gente solo pueda visualizar su actividad cardíaca en un centro de asistencia o que los médicos tengan que desplazarse hasta algún lugar que cuente con electrocardiógrafos para poder realizar un primer diagnóstico.

Al ser el ECG un método de diagnóstico útil para detectar problemas cardíacos y considerando la cantidad de personas que mueren año a año por patologías cardíacas (alrededor de 17,7 millones en el mundo [8]), el diseño y desarrollo de un prototipo de monitor cardíaco portátil con los requerimientos necesarios para hacer un diagnóstico rápido de patologías cardíacas es un desarrollo necesario para que podría ayudar a aquellas personas que no pueden acercarse a un centro de salud con la frecuencia requerida o para los médicos que trabajan fuera de instituciones médicas.

Con el prototipo se busca hacer del monitoreo básico de la actividad electrocardiográfica un

procedimiento rápido y fácil para que cualquier médico, ya sea en un establecimiento médico o ambulatorio, pueda tener un pantallazo del estado del corazón del paciente.

Además, se puede incursionar en el ámbito de la telemedicina si los datos son almacenados y transmitidos por internet a teléfonos móviles o computadoras para que un especialista los coteje remotamente.

2.3.2 Objetivo general

Diseñar e implementar un monitor cardíaco precordial monocanal portátil que cumpla con los requisitos necesarios de diseño y funcionamiento para asistir al médico en diagnósticos rápidos de problemas asociados a la actividad cardíaca.

El mismo debe garantizar el cumplimiento de las prestaciones mínimas necesarias para una buena adquisición y representación de la señal, portabilidad, bajo costo; así como también estar preparado para el rápido acoplamiento de un módulo de comunicación inalámbrico y soportar desarrollos futuros orientados a la transmisión de datos a otros dispositivos.

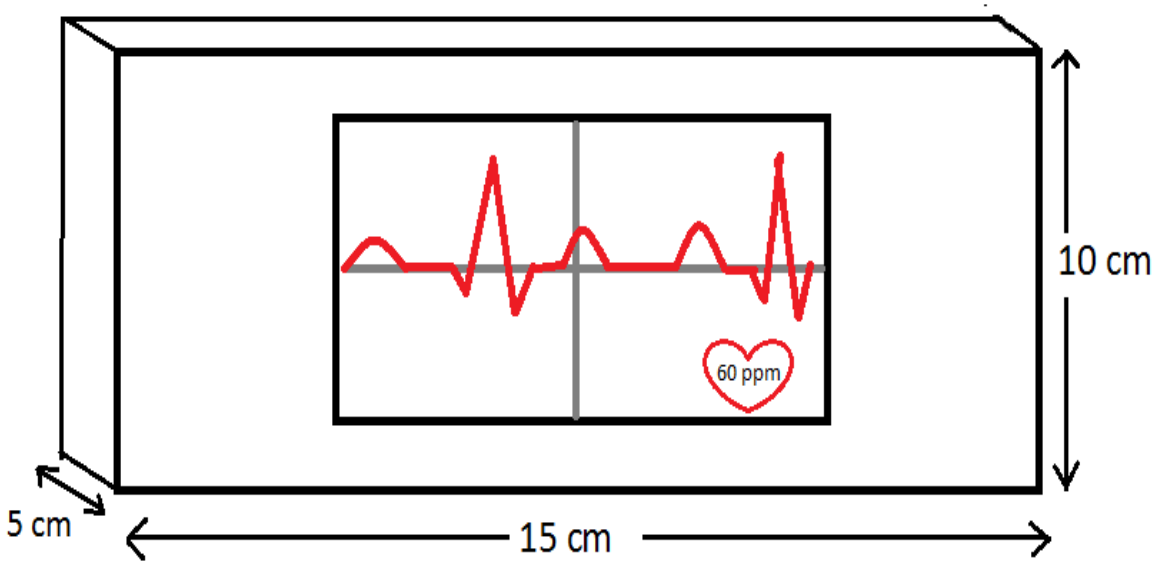


Figura 2.3.2: Diagrama de la idea inicial del monitor a diseñar.

2.3.3 Objetivos específicos

Los requisitos técnicos planteados en la concepción del proyecto son los que se presentan a continuación:

- Resolución mínima de la pantalla de 200x300 pixeles para poder visualizar 2 ondas completas de ECG con una frecuencia impresión en pantalla de 150Hz
- Conversor A/D de por lo menos 8 bits para la señal de ECG pico a pico

- Rechazo de modo común
- Detección y segmentación en tiempo real de latidos, frecuencia cardíaca, etc.
- Peso máximo 250 gramos
- Dimensiones máximas, rectangular de 5x10x15 centímetros
- Baterías de 4-5.5 V (DC), alrededor de 320 mA
- Filtrado analógico (Pasa altos + Pasa bajos)
- Filtrado Digital (Notch 50/100 y 60/120 Hz) con la opción de poder configurarlo dependiendo del lugar
- Amplificación de la señal por lo menos 1000 veces para la digitalización y el procesamiento
- Frecuencia de muestreo mínima de 200 Hz
- Gabinete impreso en 3D en acrilonitrilo butadieno estireno (ABS)
- Fuente de alimentación adecuada para un dispositivo portátil

3. Desarrollo

Para comenzar con el diseño del monitor cardíaco, se discutió e investigó acerca de las tecnologías existentes para diseñar y construir el dispositivo imaginado. En este punto se analizó desde una perspectiva macro, antes de llevar a cabo una investigación exhaustiva, si el proyecto pudiese ser realizado con la tecnología actual y un presupuesto limitado y bien establecido.

Desde el principio se decidió utilizar una placa de circuito impreso (PCB) que sería la que contendría los filtros analógicos, las etapas de amplificación, las baterías, el conversor de voltaje y las conexiones a la pantalla. Esta decisión fue basada en la practicidad de tener todos los componentes en una placa de un tamaño lógico para un dispositivo portátil y el bajo costo de fabricación que supondría hacerlas en grandes cantidades con un diseño preestablecido.

Por otro lado, la pantalla que se eligió, por su tamaño, costo y facilidad de utilización (principalmente la de su interfaz gráfica), fue 4Duino. Ésta integra un micro controlador Arduino ATMega a una pantalla a color táctil de 2.4'' de 320x240 pixeles, WiFi integrado, un conversor A/D y puertos para alimentación de la pantalla y Arduino además de ofrecer una comunicación simple entre el Microprocesador y la pantalla. Ese módulo con su procesador de 8 bits cumple con las especificaciones propuestas para hacer el procesamiento de datos necesario para mostrar en pantalla la señal adquirida por los electrodos y permitirle al usuario visualizarlos nítidamente.

El gabinete a utilizar sería uno de diseño propio impreso en 3D en el cuál entraran la placa, la pantalla y las pilas (medio de alimentación elegido por practicidad y costo).

Se dividió el trabajo en etapas de diseño que permitieron trabajar simultáneamente en más de una fase para agilizar el proyecto.

Las etapas elegidas fueron:

1. Filtros analógicos
    a. Investigación y diseño de los filtros analógicos
    b. Prueba digital en QUCS de los filtros analógicos
    c. Mediciones en laboratorio de los filtros analógicos
2. Diseño etapas de amplificación
3. Diseño del circuito de alimentación
4. Diseño e impresión de placa PCB
5. Diseño de electrodos de contacto
6. Programación de interfaz gráfica 4Duino
7. Diseño de filtros digitales e implementación en Arduino
    a. Filtro Notch configurable
    b. Muestreo de la señal para ser visualizada en pantalla
8. Diseño e impresión del gabinete en 3D
9. Etapa de prueba
    a. Medición completa de la placa final
    b. Mediciones de electrodos con PCB en laboratorio
    c. Prueba electrodos y PCB con pantalla 4Duino
10. Ensamblaje final y puesta en marcha

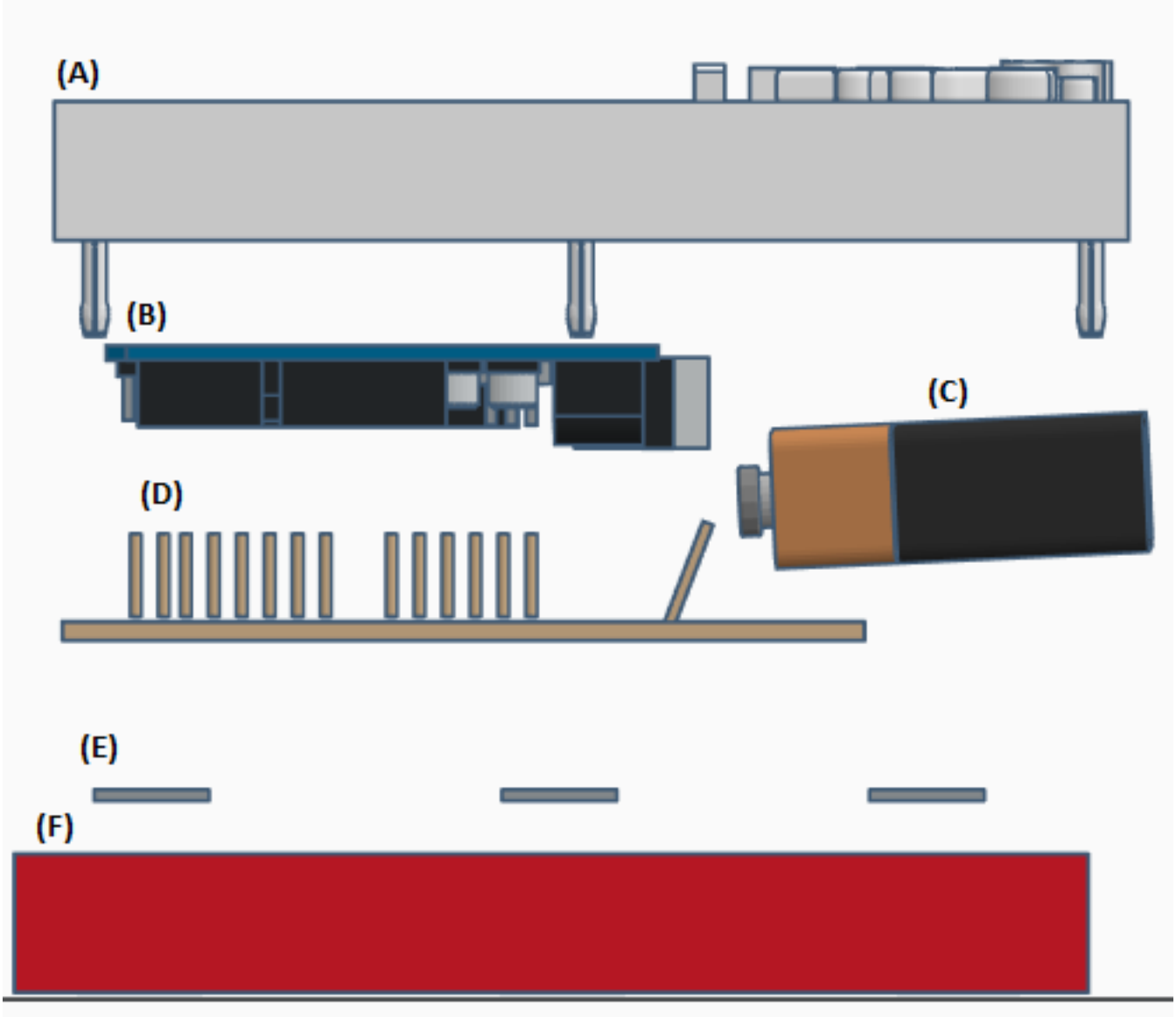


Figura 3.1: Componentes que conformarán el monitor. (A) Parte superior del gabinete; (B) Módulo con *display* y Arduino; (C) Batería de 9V para alimentar el monitor; (D) Placa PCB que contiene los circuitos; (E) Electrodos de acero inoxidable; (F) Parte inferior del gabinete.

En las figuras a continuación se detallan los circuitos más importantes contenidos en la placa PCB:

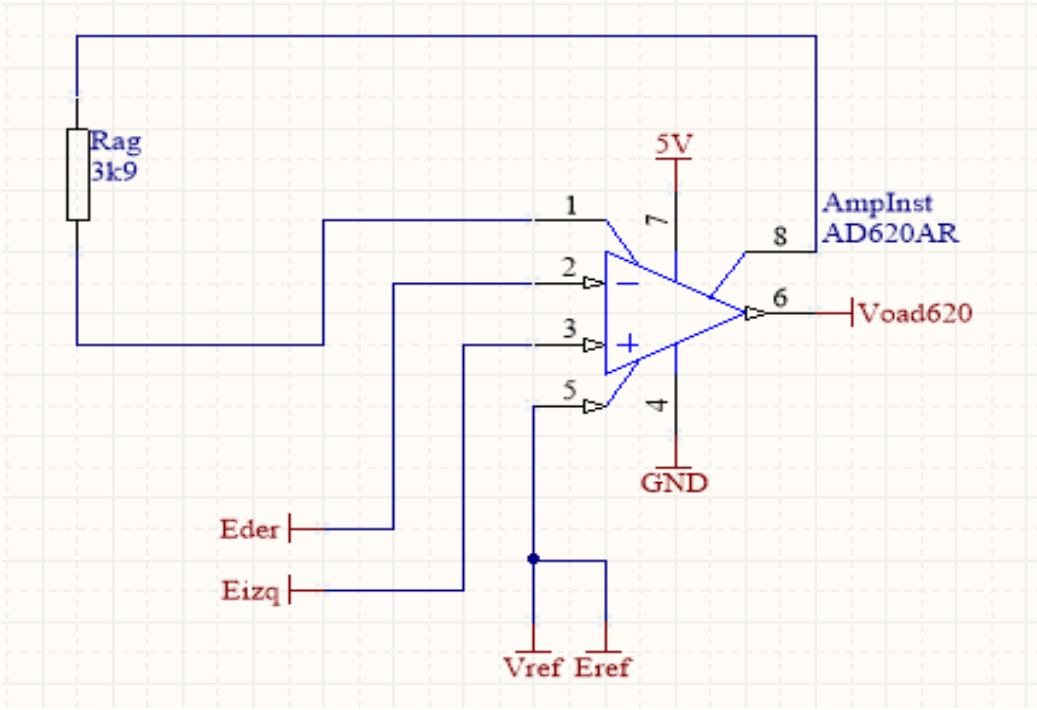


Figura 3.2: Esquemático del amplificador de instrumentación.

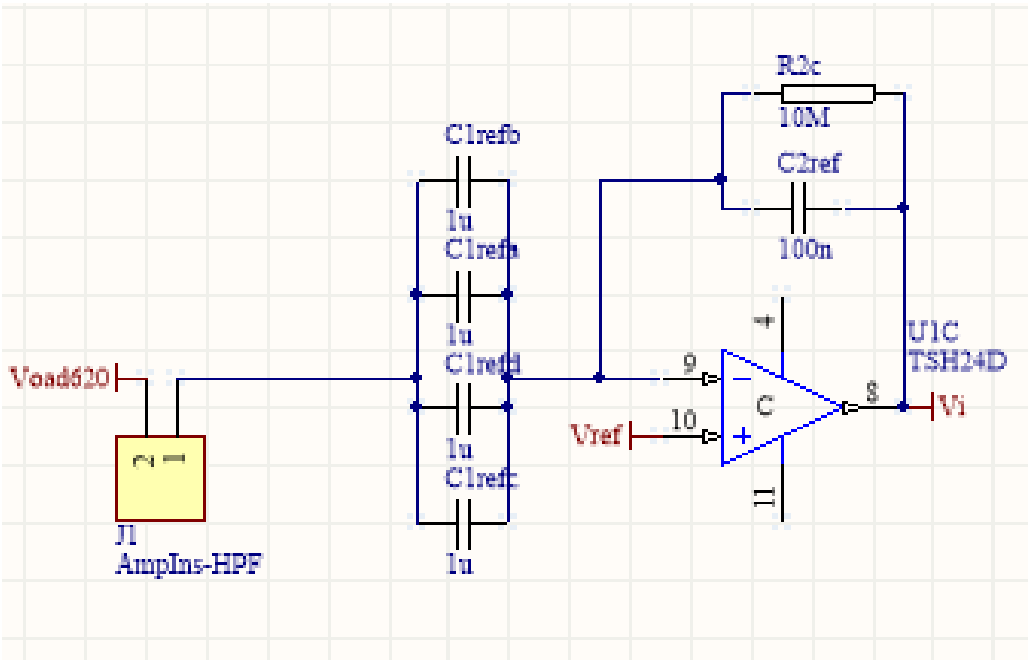


Figura 3.3: Esquemático del filtro pasa altos.

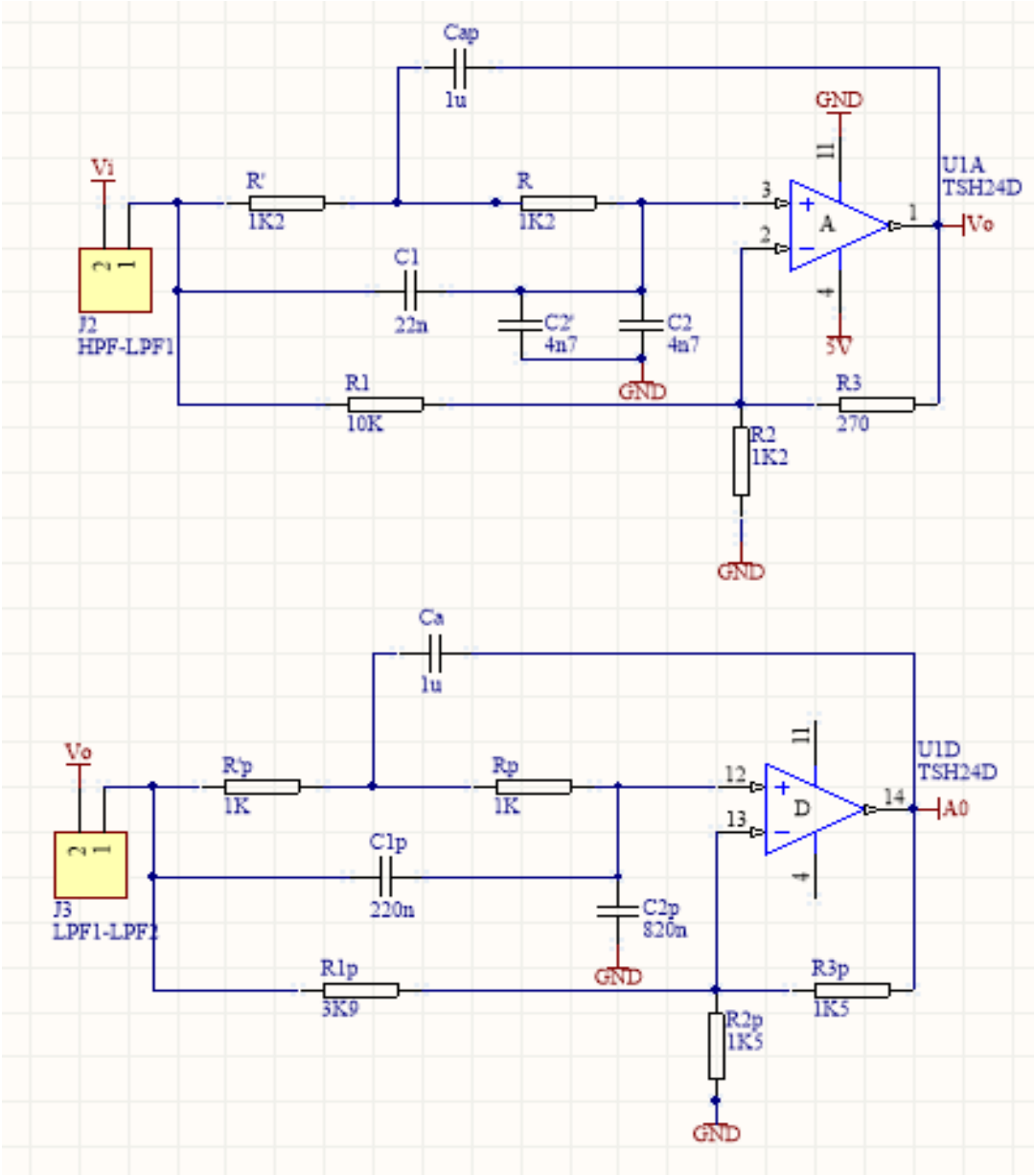


Figura 3.4: Esquemático del filtro pasa bajos implementado en 2 celdas.

3.1 Electrodos

Para la construcción de los electrodos de adquisición de la señal de ECG, se buscó un material que fuese resistente a la corrosión generada por el contacto con las sales de la transpiración presentes en el cuerpo del paciente, buen conductor de la electricidad y fácil de encontrar. El material que se entendió cumple con estos requerimientos fue el acero inoxidable.

Con esta decisión en mente, se utilizó una pequeña plancha de acero inoxidable de 0.5mm de espesor y se cortaron tres piezas rectangulares de 2 cm x 4 cm que serían los electrodos del monitor (figura 3.1.1). Las tres piezas se colocaron luego en los nichos diseñados para alojar a los

electrodos en la parte inferior del gabinete (figura 33) tal como se muestra en la figura 3.1.2. Una vez fijados los electrodos, se los unió a los cables que, del lado interior del gabinete conectarían al conversor A/D con los electrodos que a su vez transmitirían la señal adquirida del pecho del paciente.

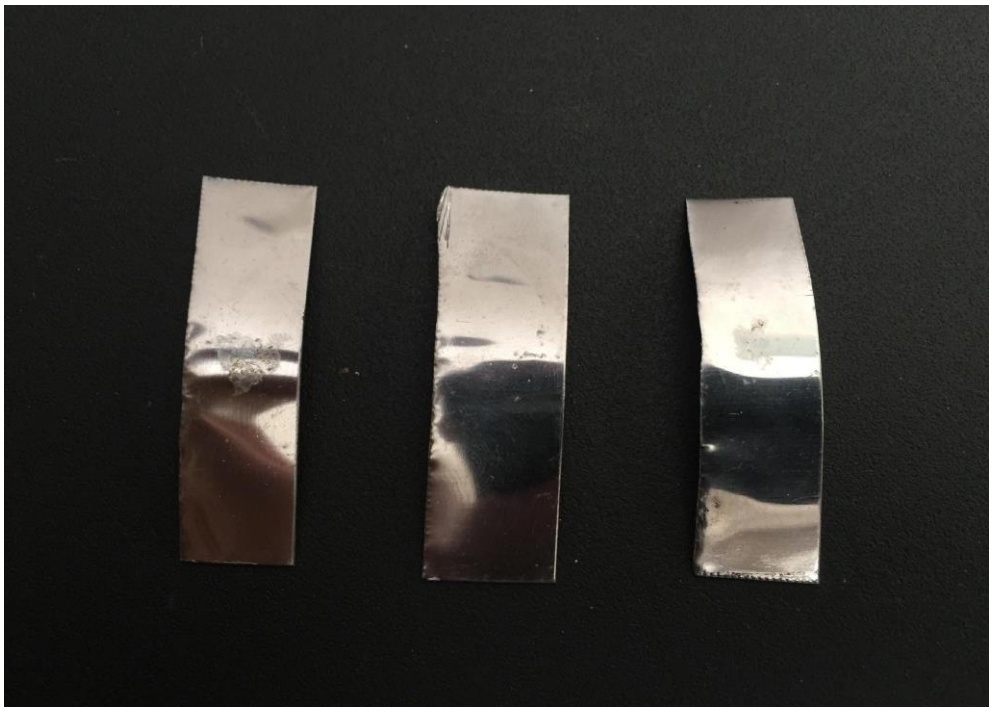


Figura 3.1.1: Foto de los electrodos de acero inoxidable.

Figura 3.1.2: Foto de los electrodos de acero inoxidable colocados en el monitor.

3.2 Amplificador de instrumentación

Para el diseño de la amplificación de la señal adquirida del cuerpo, se decidió utilizar un amplificador de instrumentación comercial que cumpliera con las características de poder amplificar la señal introduciendo la menor cantidad de ruido posible para así hacer la tarea de filtrado más efectiva. El amplificador de instrumentación elegido fue un INA333 en formato SMD fabricado por Texas Instruments que cumple con los requerimientos de introducir poco ruido (50 nV/$\sqrt{Hz}$), poder ser alimentado con 5V, ser *rail to rail* (trabaja con tensiones en el rango ±5.5 V) y permitir la regulación de la ganancia de la señal tomando como la ganancia que multiplica a la señal $G = \left(1 + \frac{100k\Omega}{R_g}\right)$, siendo R_g la resistencia que se coloca entre los pines 1 y 8 de la figura 3.2.1.

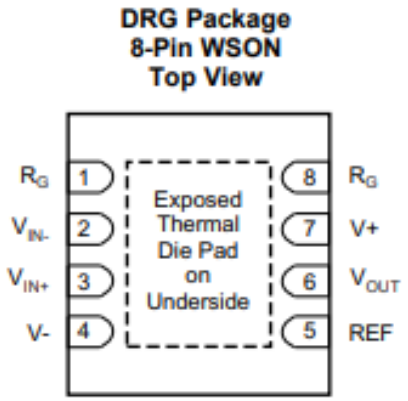


Figura 3.2.1: Representación del amplificador de instrumentación INA333.

Las características principales del INA333, según su hoja de datos, se listan en la figurasiguiente figura:

- Low Offset Voltage: 25 µV (Maximum), G ≥ 100
- Low Drift: 0.1 µV/°C, G ≥ 100
- Low Noise: 50 nV/√Hz, G ≥ 100
- High CMRR: 100 dB (Minimum), G ≥ 10
- Low Input Bias Current: 200 pA (Maximum)
- Supply Range: 1.8 V to 5.5 V
- Input Voltage: (V–) +0.1 V to (V+) –0.1 V
- Output Range: (V–) +0.05 V to (V+) –0.05 V
- Low Quiescent Current: 50 µA
- Operating Temperature: –40°C to +125°C
- RFI Filtered Inputs
- 8-Pin VSSOP and 8-Pin WSON Packages

Figura 3.2.2: Características principales del amplificador de instrumentación *SMD* elegido .

Entre las características del INA333, es de gran importancia el alto CMRR, ya que, como se

explicó en la introducción, logra cancelar (en este caso por lo menos 100 dB) las señales que estén presentes tanto en la entrada positiva como en la entrada negativa del amplificador. Esta cancelación es necesaria para la medición de señales médicas ya que la señal a medir suele ser pequeña en comparación a las señales ruidosas en el contorno de la medición, por ello, se deben filtrar las señales de ruido antes de ser amplificadas para no hacer más complejo el filtrado de la señal al amplificar el ruido.

3.3 Filtros analógicos

Para poder visualizar correctamente la señal eléctrica que el corazón produce con cada latido se necesitan filtros analógicos para muestrear la señal de tiempo contínuo que transmiten los electrodos, evitar el *alliasing* y filtrar en primera instancia el ruido que generan los músculos, el movimiento de los electrodos, la transpiración, la línea eléctrica y poder tener en última instancia la señal deseada con el espectro que brinda la información cardíaca.

Como se discutió en la introducción, existen diversas versiones acerca de cuál es la frecuencia necesaria para obtener una señal de ECG que incluya las frecuencias importantes para los estudios que se desee realizar. Dado que uno de los objetivos buscados fue obtener una señal que pudiera ser utilizada para llevar a cabo estudios de detección de latidos y frecuencia cardíaca, se decidió utilizar la frecuencia de muestreo de 1000 Hz, que como se mencionó anteriormente es superior a la recomendada por la “American Heart Association” para estudios de ECG y permite realizar estudios avanzados sobre la actividad cardíaca.

Como fuentes de información para el diseño y la implementación de los filtros se utilizaron libros de procesamiento de señales que ayudaron a definir qué tipo de filtro y qué implementación sería conveniente utilizar [10,11, 12]. La conclusión a la que se llegó fue que se debería trabajar con dos filtros, un pasa bajos y un pasa altos. El Pasa bajos cumpliría con la función de eliminar las componentes ruidosas de alta frecuencia que distorsionan la señal, representado gráficamente en la figura 3.3.1. Por otro lado, el pasa altos debería filtrar la componente contínua de la señal, dejando pasar las frecuencias mayores a un valor a determinar. Lo esperado luego de estas etapas de filtrado sería una señal de ECG sin ruidos de alta frecuencia, atenuados por el pasa bajos, ni error de *offset* por la componente contínua, atenuado por el pasa altos.

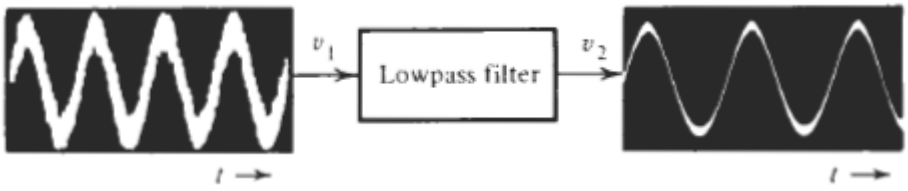


Figura 3.3.1: Representación gráfica de la función de un filtro pasa bajos en el espectro temporal.

Estas fuentes permitieron concluir que los filtros necesarios para obtener una señal que podría ser graficada y procesada correctamente serían:

- Pasa bajos que atenúe 60dB en $\frac{F_S}{2}$, siendo F_s la frecuencia de muestreo elegida
- Pasa altos que amplifique la señal, filtre la componente contínua y con un tiempo de establecimiento lógico para su aplicación

Para que el pasa bajos atenuase correctamente a partir de la frecuencia elegida, se tomó una frecuencia de paso de 100H z donde se tiene un *ripple* muy bajo para no distorsionar la señal en la banda de paso mientras que luego de $\frac{F_S}{2}$ la señal se destruye, evitando así el *alliasing*. La magnitud en que la señal disminuye luego de la frecuencia de corte fue otra variable que debió ser tenida en cuenta, se consideró que 60dB de atenuación sería suficiente ya que al cuantizar con 10 bits, aunque luego se utilizan 8 bits efectivos en el microprocesador, la señal queda por debajo del piso del ruido de cuantización.

3.3.1 Pasa bajos

Para el diseño del filtro pasa bajos a utilizar, se usó Matlab [13]. Se simularon distintos tipos de filtros para evaluar cuál devolvería la señal con la atenuación en las frecuencias especificadas con el filtro de menor orden posible. Se estudiaron las aproximaciones de Butterworth y la elíptica, o Cauer [10].

Butterworth

La respuesta de Butterworth suele caracterizarse como

$$|T_n(j\omega)|^2 = \frac{1}{1+\omega^{2n}}$$

Analizando la función, se encuentra que el filtro de Butterworth tiene únicamente polos con la misma frecuencia de corte ya que ω se encuentra únicamente en el denominador de la función de transferencia. También puede observarse que si ω es grande, aparece un *roll-off* de n-polos en el cuál la atenuación se incrementa 20n dB por década.

Para lograr implementar un pasa bajos Butterworth se utiliza la aproximación de Butterworth con diversos órdenes según cuánto se quiera acercar el filtro al funcionamiento ideal de un pasa bajos. Debido a que las N-ésimas derivadas de la función son nulas, la aproximación cumple con la condición de considerarse *maximally-flat* ya que, para un orden dado, tienen el mayor *roll-off* posible sin introducir sobrepicos al diagrama de bode, es decir, acerca lo máximo posible el comportamiento del filtro al de una *brick-wall* de un pasa bajos ideal. En las figuras 3.3.1.1 y 3.3.1.2 puede apreciarse el comportamiento del filtro de Butterworth en los diagramas de Bode de amplitud y fase y en los diagramas de polos y ceros para distintos órdenes.

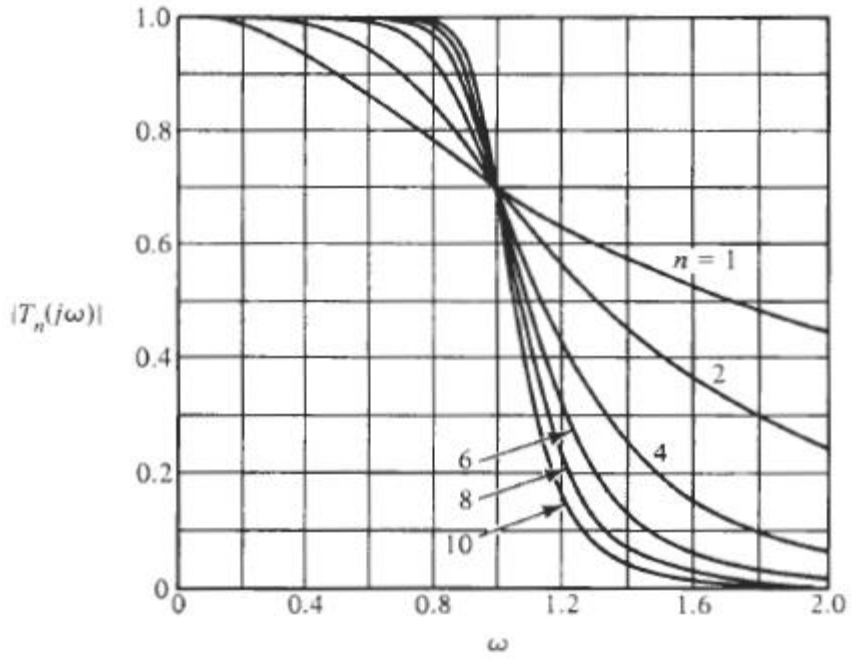


Figura 3.3.1.1: Diagrama Bode de magnitud de un filtro de Butterworth de orden n=1 a n=10.

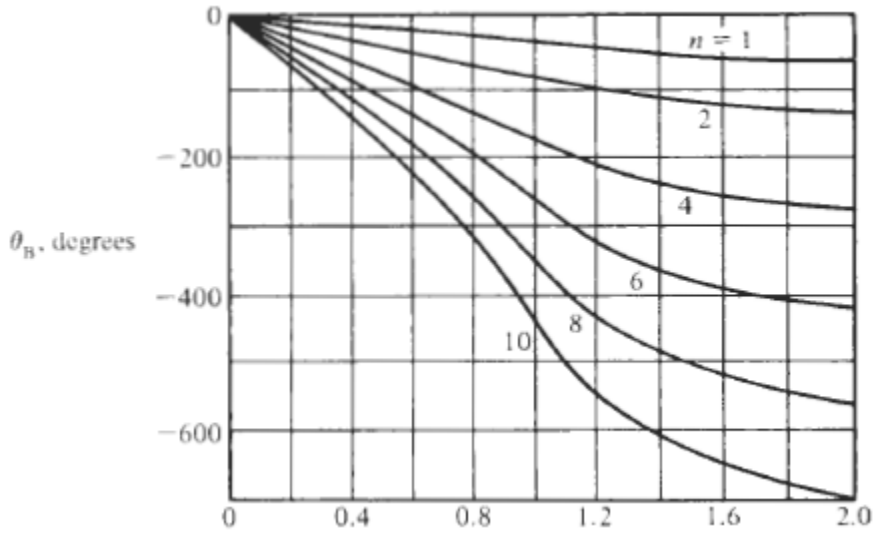


Figura 3.3.1.2: Diagrama Bode de la fase de un filtro de Butterworth de orden n=1 a n=10.

En la figura 3.3.1.3 (a continuación), dónde se muestran los polos para filtros de orden n=4 hasta n=7, solo se representó la parte izquierda del diagrama ya que solo estos corresponden a circuitos estables.

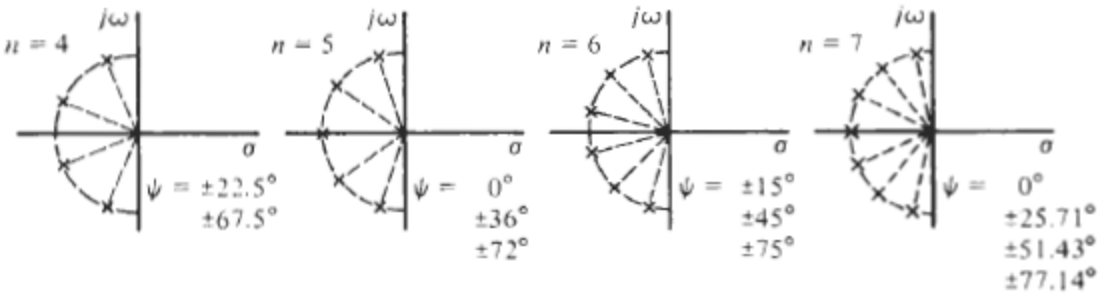


Figura 3.3.1.3: Diagrama de polos para el filtro de Butterworth de orden n=4 a n=7. Se muestra solo el lado izquierdo de los diagramas.

Elíptica (Cauer)

Como se mencionó anteriormente, el filtro de Butterworth no tiene ceros finitos de transmisión, en cambio, el filtro elíptico o de Cauer introduce ceros de transmisión en su función de transferencia para lograr un *roll-off* de la ganancia a través de la banda de transición más rápido. El costo de insertar ceros de transmisión finitos en la función de transferencia es la reducción en la rapidez de la atenuación a altas frecuencias, en este caso, la atenuación se incrementa $20(n-m)$ dB por década en vez de $20n$ dB por década, siendo n el orden de los polos y m el orden de los ceros de transmisión de la función de transferencia. Para ciertas aplicaciones es más deseable una banda de transición corta que una rápida atenuación de las altas frecuencias, razón por la cual se trabajó con filtros elípticos en el diseño del monitor.

Los filtros elípticos cumplen con la importante característica de garantizar el menor orden para una frecuencia y atenuación determinadas en comparación con otro tipo de filtros pero relegan la característica de ser *maximally-flat* ya que introducen oscilaciones tanto en la banda de paso como en la banda de atenuación**.** En la figura 3.3.1.4 (a) puede observarse la respuesta de un filtro *maximally-flat* como el filtro de Butterworth mientras que en (b) se observan oscilaciones o *ripple* hasta la frecuencia de paso. En el caso del filtro de Cauer el *ripple* se extiende también más allá de la frecuencia de paso para distribuir este error en todo el espectro, minimizando así la magnitud de las oscilaciones.

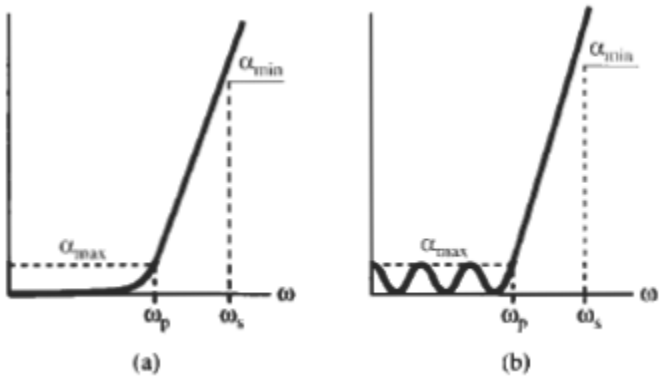


Figura 3.3.1.4: Representación de la respuesta en frecuencia de una filtro (a) *maximally-flat* (b) con oscilaciones o *ripple.*

La respuesta en frecuencia de un filtro de Cauer con *ripple* tanto en la banda de paso como en la de rechazo puede observarse en el diagrama de Bode de la figura 3.3.1.5. Para entender el detalle de las diversas notaciones utilizadas en el diagrama, se provee la figura 3.3.1.6 con la explicación de las notaciones utilizadas.

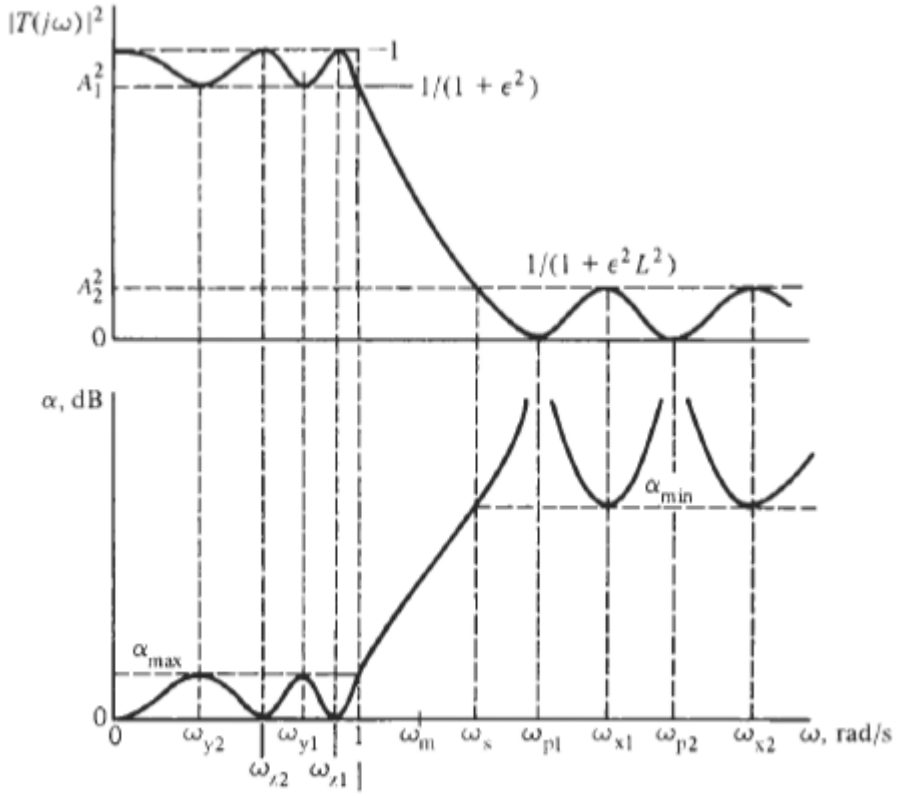


Figura 3.3.1.5: Diagrama de Bode de la respuesta en frecuencia de un filtro de Cauer.

Band	$\lvert T(j\omega)\rvert^2$	Attenuation α	$R_n(\omega, L)$	Frequency Notation	Frequency Name
Pass	1	$\alpha = 0$	0	ω_{zv}	Attenuation zeros, zeros of R_n
	$\frac{1}{1+\varepsilon^2}$	$\alpha = \alpha_{max}$	± 1	ω_{yv}	
Stop	0	$\alpha = \infty$	∞	ω_{pv}	Attenuation poles, transmission zeros
	$\frac{1}{1+\varepsilon^2 L^2}$	$\alpha = \alpha_{min}$	$\pm L$	ω_{xv}	

Figura 3.3.1.6: Características de la magnitud de la respuesta de Cauer.

La representación de la magnitud de la respuesta de un filtro Cauer de orden n=5 puede observarse en la figura 3.3.1.7 para visualizar gráficamente la respuesta del filtro con un orden determinado.

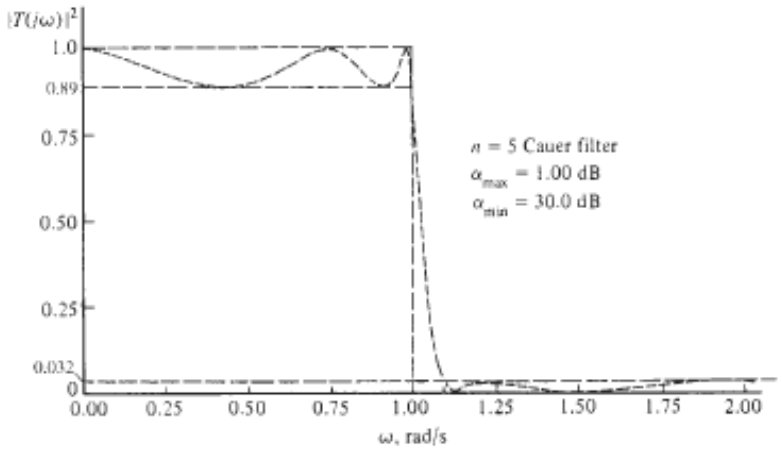


Figura 3.3.1.7: Diagrama de Bode de la respuesta en frecuencia de un filtro de Cauer.

Como se mencionó anteriormente, los filtros Cauer cumplen con la característica de ser los de menor orden de implementación para una plantilla determinada, haciendo de este tipo de filtro el más eficiente en este sentido. En la figura 3.3.1.8 a continuación se comparan los diagramas de Bode de la implementación de una misma plantilla con distintos filtros en la cuál se evidencia la mayor eficiencia de los filtros elípticos en cuanto al orden de su aproximación.

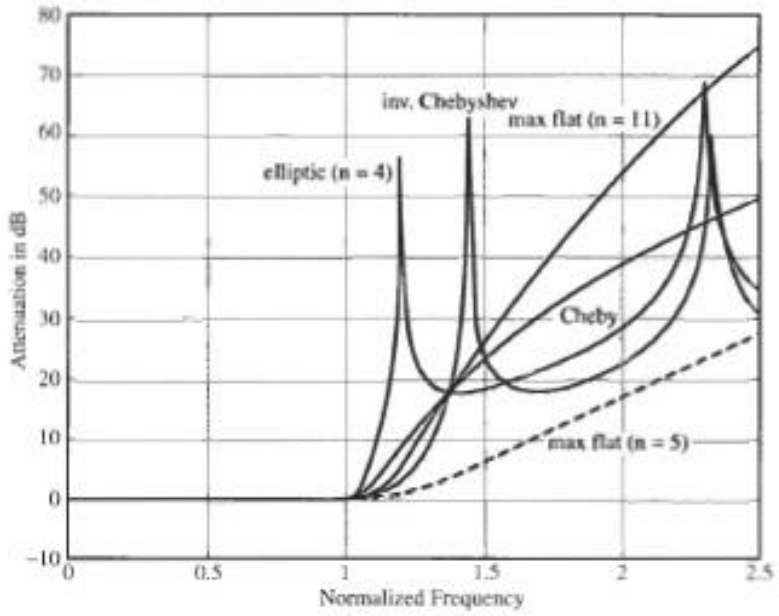


Figura 3.3.1.8: (a) Respuestas comparadas de filtros *maximally-flat* (n=11 y n=5), Chebyshev (n=5), Chebyshev inverso (n=5), y Cauer (n=4) para obtener la misma atenuación.

Elección del tipo de filtro a utilizar

En el caso del pasa bajos Cauer, el orden mínimo del filtro necesario para obtener la atenuación en las frecuencias requeridas fue de orden 4 mientras que con el pasa bajos Butterworth el orden necesario fue 7. En las figuras NN y MM a continuación se encuentran respectivamente el diagrama de bode de los filtros pasa bajos Butterworth y Cauer diseñados (Anexos 1 y 2).

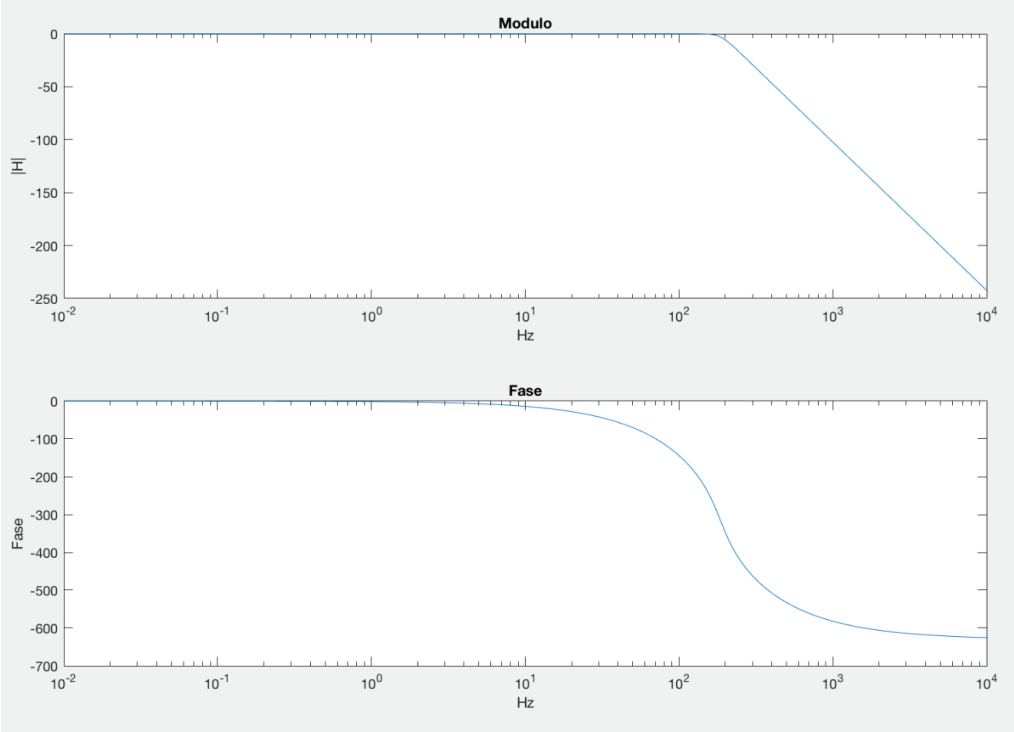


Figura 3.3.1.9: Diagrama de bode de magnitud y fase del filtro de Butterworth de orden 7 diseñado.

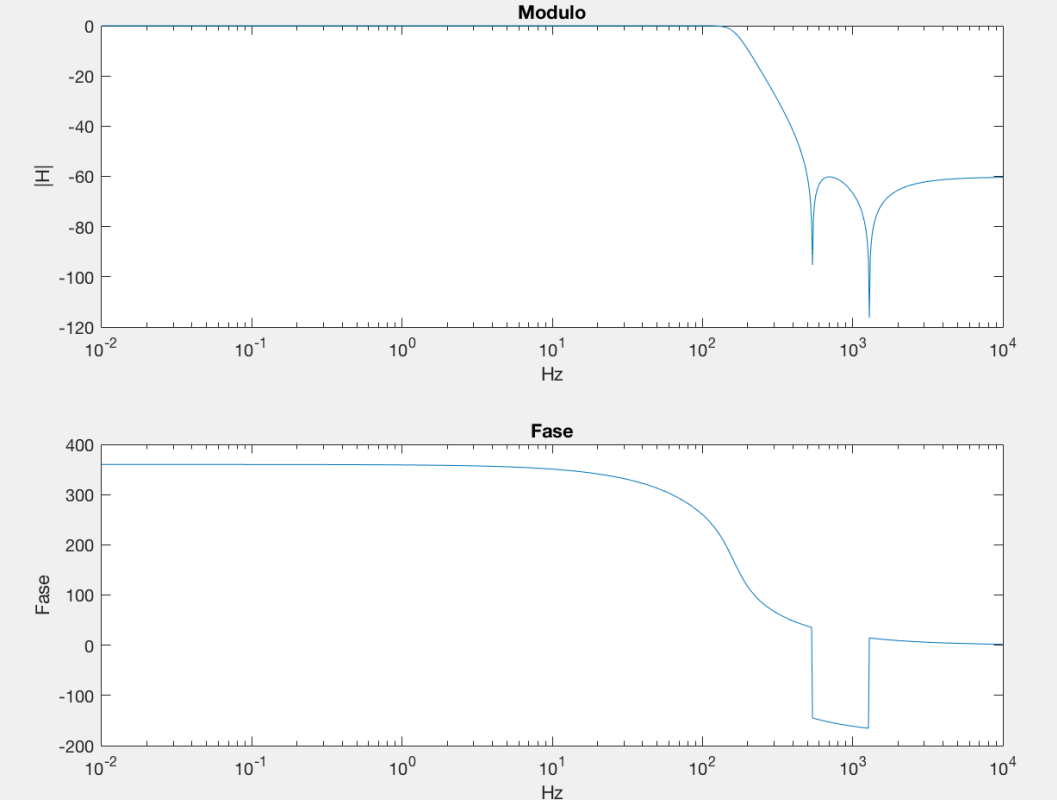


Figura 3.3.1.10: Diagrama de bode de magnitud y fase del filtro de Cauer de orden 4 diseñado.

Como se observa en los diagramas anteriores, ambos filtros cumplen con los requisitos previamente expuestos y aunque el filtro Cauer resultó ser tres órdenes menores al Butterworth, se siguió trabajando con ambos ya que la implementación del Cauer resulta más compleja que la del Butterworth por tener ceros de trasmisión. Por este motivo, se investigó para ambos qué tipo de celdas eran necesarias para implementar cada uno de ellos y cuánto espacio ocuparía cada uno en la placa.

La implementación del filtro de Cauer requirió una celda que agrupara simultáneamente un par de polos complejos conjugados con un par de ceros de transmisión, mientras que el filtro de Butterworth como se explicó anteriormente, solo necesita polos, haciendo la implementación del Cauer más compleja que la del Butterworth al tener que generar esos ceros.

La investigación que continuó dictaminó que las celdas podrían ser las siguientes, dependiendo de la aproximación elegida:

Celda Sallen-Key

La configuración de la celda Sallen-Key fue introducida por primer a vez en 1955 por R.P. Sallen y E.L. Key de los laboratorios del M.I.T. Este tipo de configuración es una de las más utilizadas dada la facilidad de su armado y la poca dependencia de la performance del filtro con la del amplificador operacional. La fase de la señal a través del filtro resulta igual a la salida que a la entrada ya que el filtro tiene una configuración *non-inverting.* La utilización de esta implementación tiene también el beneficio de ser de fácil manufactura ya que el ratio entre las resistencias y capacitores

a utilizar suele ser chico, permitiendo utilizar componentes de escalas comparables y fáciles de conseguir.

Por otro lado, el filtro pasa bajos implementado con celdas de Sallen-Key permite crear polos reales o complejos conjugados, pero no puede implementar ceros de transmisión, haciendo esta celda ideal para implementar filtros pasa bajos del tipo Butterworth pero no para la implementación de un filtro del tipo Cauer. La función de transferencia que puede implementar este tipo de celdas se detalla a continuación.

En la figura 3.3.1.11 puede observarse el diseño de la celda Sallen-Key para implementar un filtro pasa bajos de orden 2 y en las ecuaciones a continuación se observan las fórmulas para elegir los parámetros fundamentales del filtro y cómo implementarlos con los componentes del circuito.

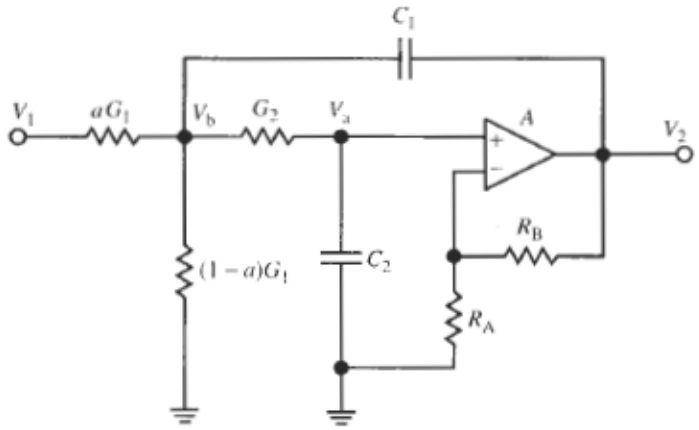


Figura 3.3.1.11: Celda Sallen-Key de orden 2 para implementar Butterworth.[10]

$$\omega_0 = \frac{G_1 G_2}{C^2} \; ; Q = \frac{\sqrt{G_1 G_2}}{G_1 + G_2(2-K)} \; ; H = K > 1$$

Para elegir los cuatro elementos $C, G_1, G_2 \; y \; K$ asumimos que $R_1 = R_2 = R$ para tener mayor libertad de elección en el valor del capacitor C ya que suelen encontrarse valores más variados de resistencias que de capacitores.

En conclusión, los componentes a elegir deberán seguir los valores dados por:

$$R = \frac{1}{w_0 C} \; y \; K = 1 + \frac{R_B}{R_A}, siendo \; R_A \; un \; valor \; arbitrario$$

Celda Biquad General

También se consideró la celda *general biquad*, nombrada en referencia a que su función de transferencia tiene una función cuadrática tanto en su numerador como en su denominador. Para facilitar la elección de los componentes que forman el circuito, se elige arbitrariamente C_1=C_2=C y R_1=R_2=R. Este tipo de celda es implementada con circuitos de Sallen-Key como se observa en la figura 3.3.1.12 y su función de transferencia se detalla a continuación.

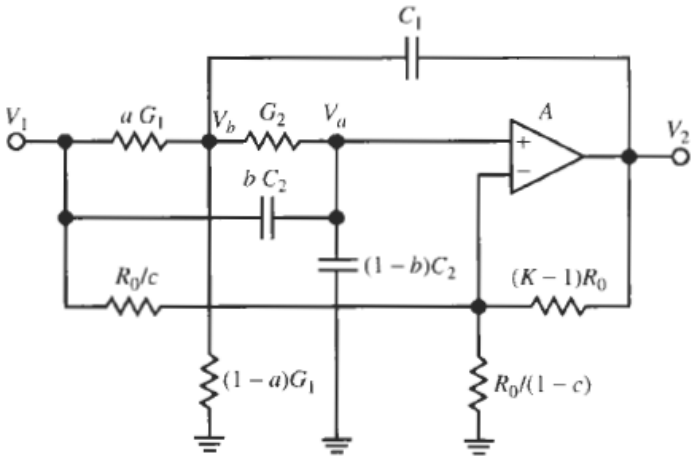


Figura 3.3.1.12: Celda Biquad general basada en Sallen-Key *low pass* de orden 2 para implementar Cauer.[10]

$$T(s) = K\frac{c_k}{2}\frac{s^2 + 2\frac{a-c_k}{c_k}{\omega_0}^2}{s^2 + s\,\omega_0/Q + {\omega_0}^2}$$

$$con\ \omega_0 = 1/(CR)\ ;\ Q = 1/(3-K)\ y\ c_k = c(K-1)/K$$

Celda SAB (*single amplifier biquad*)

La celda SAB, también conocida como celda de Delyiannis-Friend en honor a sus inventores en 1970 [10], se utiliza para implementar filtros pasa bajos o pasa bandas cuyas funciones de transferencia contengan ceros de transmisión como es el caso con los filtros tipo Cauer. El circuito que permite implementar este tipo de filtros se muestra en la figura 3.3.1.13.

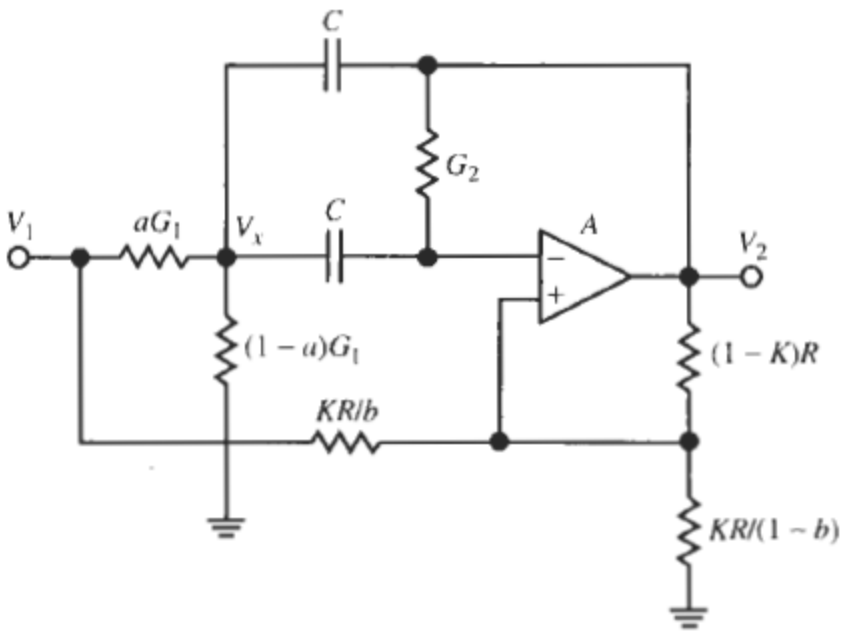


Figura 3.3.1.13: Celda Delyiannis-Friend de orden 2 para implementar Cauer. [10]

La función de transferencia que caracteriza al circuito, en la cual pueden observarse los polos y ceros que introduce se presenta a continuación en función de sus parámetros de diseño. Esta celda permite implementar una función de transferencia con dos polos y dos ceros de transmisión al igual que la celda *general biquad* descripta previamente.

$$T(s) = b\frac{s^2 + s\frac{\omega_0}{Q_0}\left[1 + 2{Q_0}^2\left(1 - \frac{a/b}{1-K}\right)\right] + {\omega_0}^2}{s^2 + s\frac{\omega_0}{Q_0}\left(1 - 2{Q_0}^2\frac{K}{1-K}\right) + {\omega_0}^2}$$

$$con\ \omega_0 = 1/\left(C\sqrt{R_1R_2}\right)\ y\ Q_0 = 0.5\sqrt{R_2/R_1}$$

Implementación del filtro

Luego de evaluar las tres celdas y el orden del filtro diseñado en cada caso, se concluyó que lo mejor sería utilizar el filtro de Cauer implementado con las celdas Biquad basada en Sallen-Key que permite crear los polos y ceros de trasmisión necesarios para el filtro. La elección tuvo que ver con el poco espacio que ocupaba en la placa este filtro comparado con un filtro de orden 7 implementado con celdas Sallen-Key standard. Por otro lado, se descartó el uso de la celda Delyiannis-Friend ya que su implementación resultó ser más compleja de lo esperado [11].

Dado que el filtro que se pretendió implementar era de orden 4, se tuvo que separar en dos filtros de orden 2 colocados en cascada para poder armarlo con la celda elegida (Código

Matlab Anexo 1). Para armar esta cascada, se buscaron los 4 polos y ceros del filtro diseñado y se crearon dos funciones de transferencia, utilizando en cada una 2 polos y 2 ceros de la transferencia original como se muestra a continuación en la funcion de transferencia original 'H' (figura 3.3.1.14), sus partes descompuestas 'H1' y 'H2" (figura 3.3.1.15, 3.3.1.16) y sus respectivos diagramas de Bode (figuras 3.3.1.17 y 3.3.1.18) que muestran como cada celda atenúa la señal en la vecindad de los 1000Hz pretendidos.

$$H = \frac{s^4 + 7.183^{-14}s^3 + 7.722^7 s^2 - 2.258^{-6}s + 7.551^{14}}{s^4 + 2034s^3 + 2.575^6 s^2 + 1.912^9 s + 7.382^{11}}$$

Figura 3.3.1.14: Función de transferencia 'H(s)' del filtro Cauer de orden 4.

$$H_1 = \frac{s^2 + 1.287^{-13}s + 6.573^7}{s^2 + 1468s + 7.199^5}$$

Figura 3.3.1.15: Función de transferencia 'H1', primer parte del filtro Cauer de orden 4.

$$H_2 = \frac{s^2 - 5.684^{-14}s + 1.149^7}{s^2 + 565.3s + 1.025^6}$$

Figura 3.3.1.16: Función de transferencia 'H2', segunda parte del filtro Cauer de orden 4.

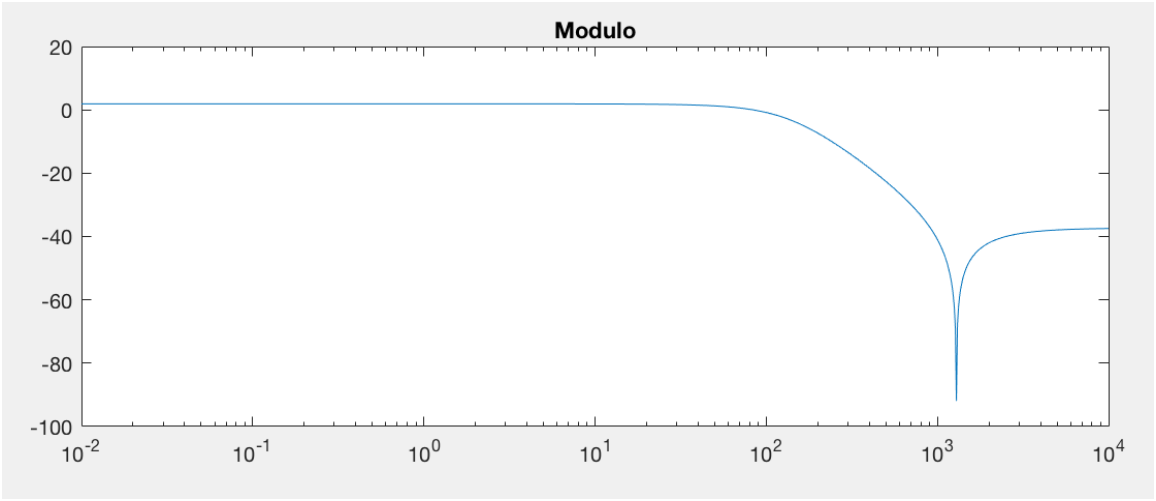


Figura 3.3.1.17: Diagrama de Bode de la atenuación de 'H1'.

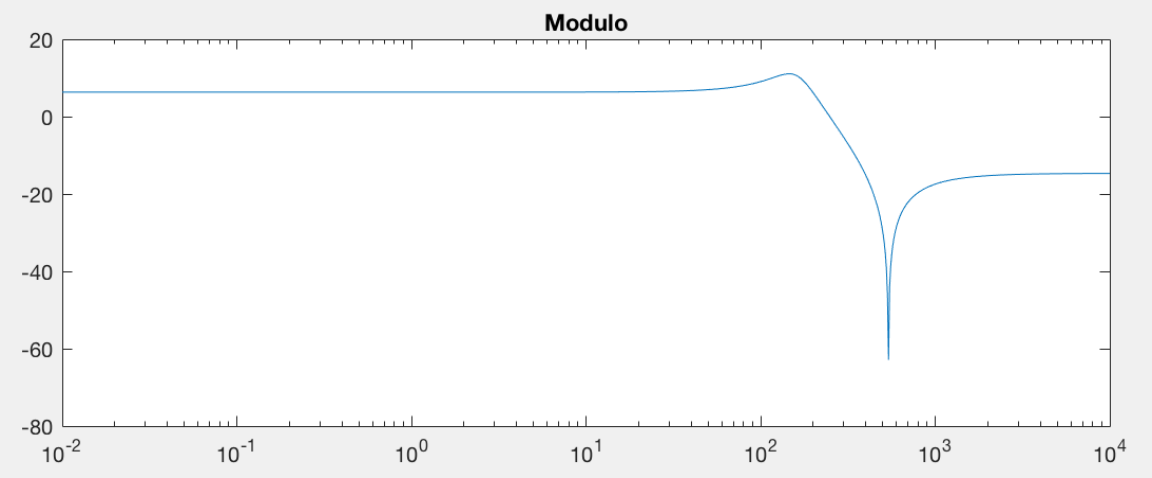


Figura 3.3.1.18: Diagrama de Bode de la atenuación de 'H2'.

Para decidir qué celda colocar primero, se tuvo en cuenta que al haber un sobrepico alrededor de los 150 Hz en la respuesta en frecuencia H_2, el mismo no debería filtrar la señal primero ya que atenuaría la señal por demás en una primera etapa y se correría el riesgo de saturar la señal. Para asegurarse de que estos factores fueran tomados en cuenta a la hora de decidir el orden en que se colocarían las celdas, se utilizó el criterio de que la celda con menor factor de calidad Q ($Q = \frac{1}{\omega_0}$, $\omega_0 = 2\pi F_0$ con F_0 representando la frecuencia de corte del filtro) debe colocarse antes que la de mayor Q [10].

Con el objetivo de probar el filtro antes de armarlo en *protoboard* e imprimir su diseño en PCB, se procedió a simular el funcionamiento del diseño en el programa QUCS que devolvió el diagrama de Bode representado en la figura 3.3.1.19 para el diseño que se muestra en la figura 3.3.1.20. Del diagrama se pudo concluir que los dos filtros en cascada que se diseñaron cumplieron con las especificaciones buscadas según la simulación y por consiguiente habría que armar las celdas requeridas en un *protoboard* para luego imprimir el diseño en un PCB.

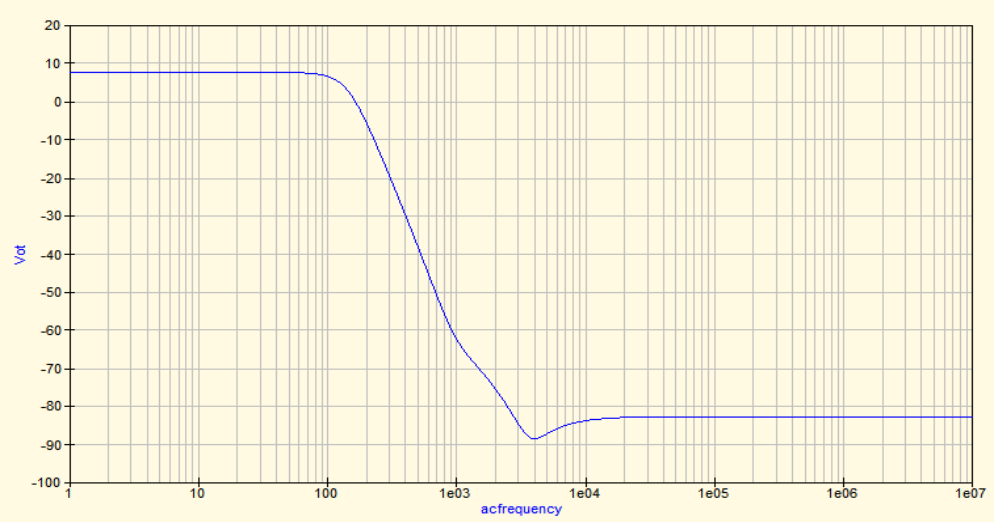


Figura 3.3.1.19: Diagrama de Bode de la implementación del filtro pasa bajos.

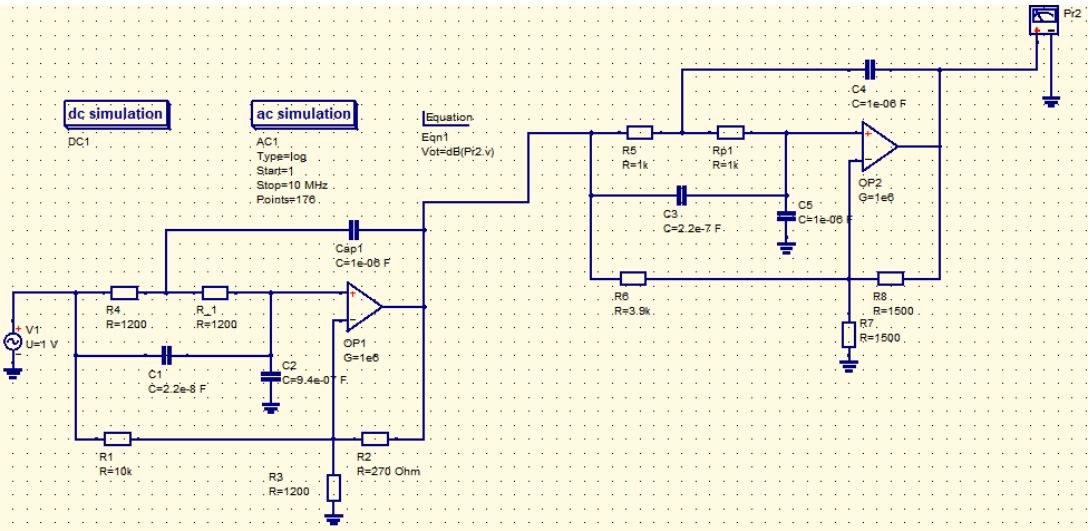


Figura 3.3.1.20: Implementación de las celdas elegidas para el pasa bajos.

3.3.2 Pasa Altos

Como se mencionó previamente, es necesario incluir un filtro pasa altos para cancelar ruido de baja frecuencia que no brindan información relevante, así como también cualquier componente de tensión contínua (error de *offset*) debida a los electrodos o introducidas por los amplificadores operacionales de etapas anteriores. La eliminación de este nivel de contínua es mandatorio, ya que la señal debe ser amplificada alrededor de 40 veces para poder muestrearla correctamente. El diseño del pasa altos elegido puede verse en la figura 3.3.2.1.

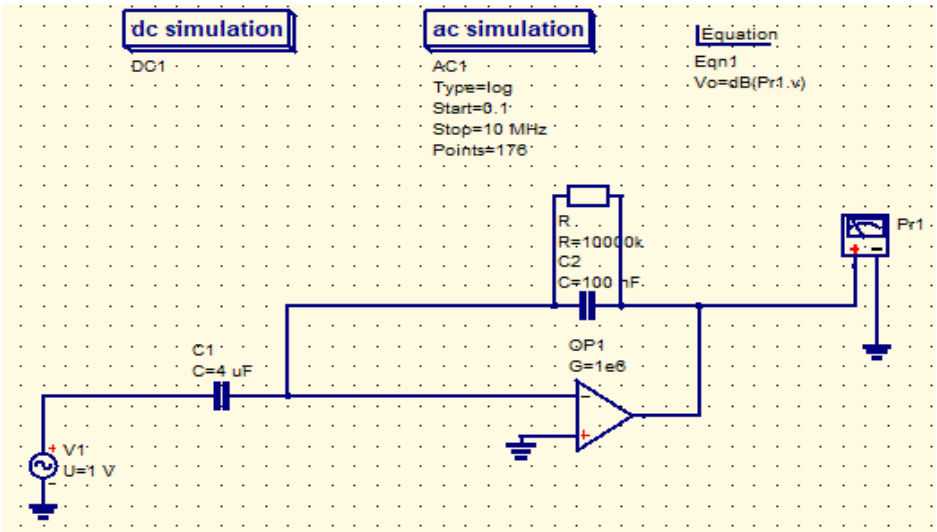


Figura 3.3.2.1: Filtro pasa altos utilizado.

Se decidió la utilización de un filtro pasa altos activo, en el cuál la amplificación se

determina por el cociente entre sus capacitores C_1 y C_2, su frecuencia de paso es elegida utilizando el cociente entre C_2 y R y el tiempo de establecimiento del filtro, tiempo que tarda el filtro en llegar a un equilibrio de carga de sus capacitores, depende de la constante de tiempo del circuito τ, dada por R y C.

En este caso, se eligió una frecuencia de paso de 0.15Hz. Con la amplificación propuesta, resultó C_1=4µF, C_2=100nF y R=10kΩ. Con estas especificaciones, el tiempo de establecimiento del pasa altos resultó ser de 7 segundos, aceptable teniendo en cuenta la situación de compromiso entre la frecuencia de paso del filtro y el tiempo que tarda en estabilizarse ya que son inversamente proporcionales.

Al igual que con el filtro pasa bajos, se procedió a probar si el filtro diseñado efectivamente cumpliría el efecto esperado con el simulador QUCS que brindó el resultado en forma del diagrama de magnitud de Bode que puede apreciarse en la figura 3.3.2.2. Dicha figura muestra que la implementación del filtro diseñado debería cumplir con las condiciones de diseño previamente expuestas.

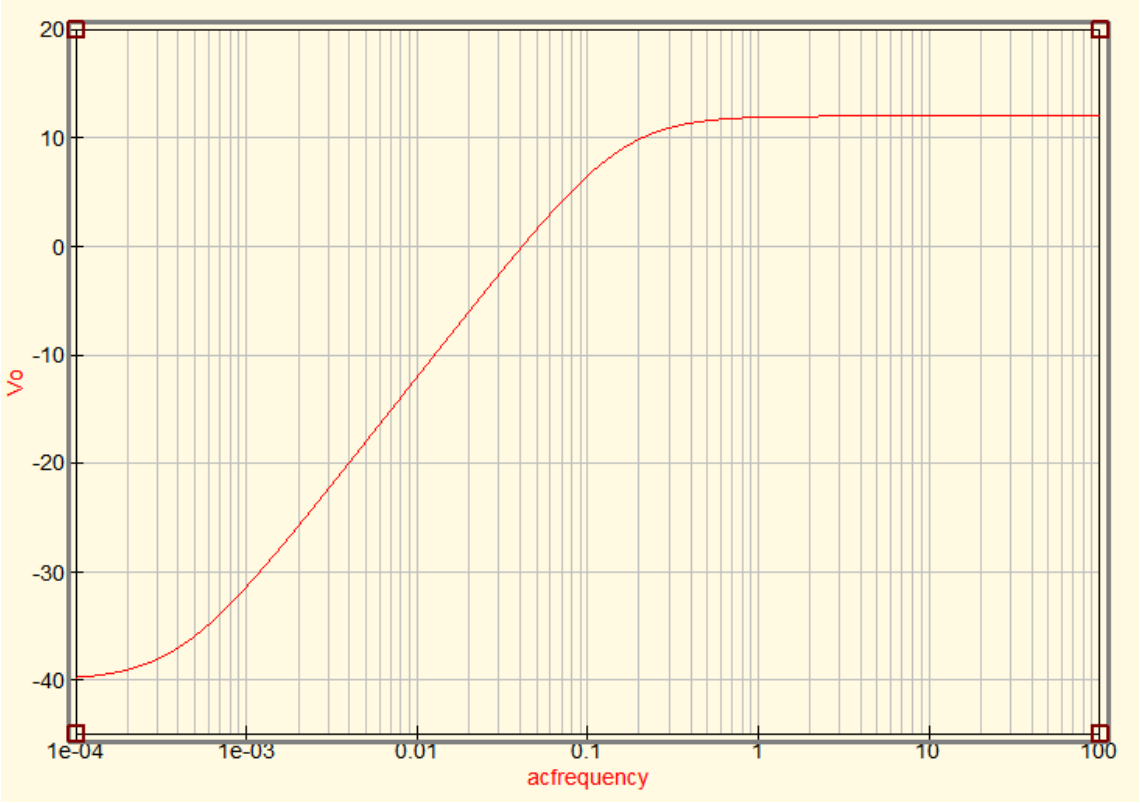


Figura 3.3.2.2: Diagrama de Bode de magnitud del filtro pasa altos utilizado.

3.3.3 Implementación y medición en Protoboard

Luego de comprobar que el diseño de los filtros elegidos cumplía efectivamente con su función en el simulador, se pasó a llevar los filtros a un circuito electrónico real usando un *protoboard* y los componentes electrónicos previamente dimensionados. De esta forma, el correcto funcionamiento de los filtros en la realidad podría ser medido en el laboratorio para

comprobar que el diseño era correcto más allá de las pruebas digitales antes de pasar a un diseño definitivo en una placa PCB.

Luego de reiterados intentos, por la complejidad del filtro para ser armado en un *protoboard* (que puede tener juntas defectuosas o componentes que pierden el contacto con el circuito), se logró que las celdas cumplieran con las características de diseño que se pautaron. Con estas pruebas, se contó con la evidencia suficiente para afirmar que las celdas construidas realizaban el filtrado que se pretendía y podrían ser utilizadas para la función de filtrar el ruido en la señal de ECG a ser adquirida.

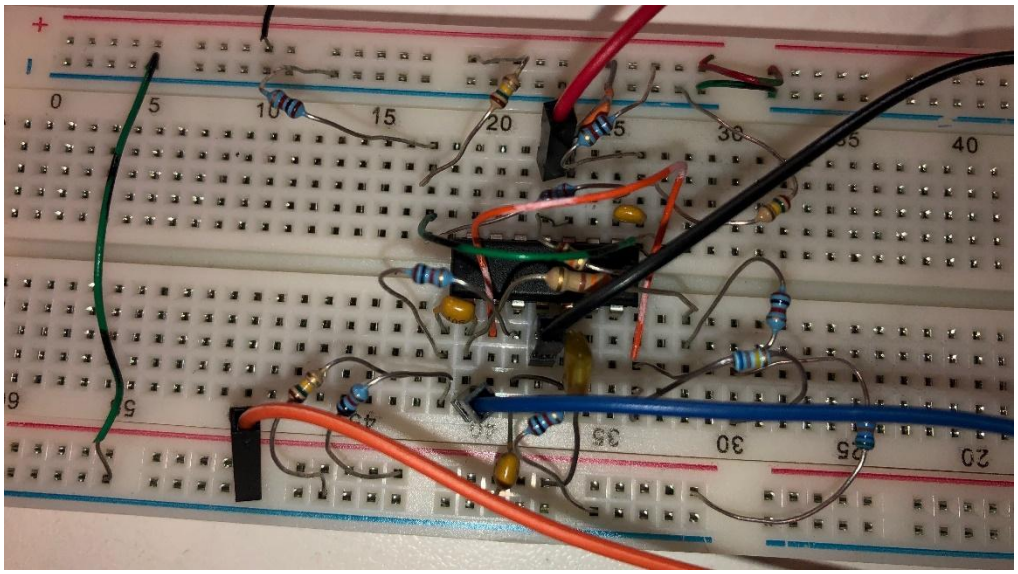


Figura 3.3.3.1: Filtros pasa bajos implementados en un *protoboard*.

3.4 Etapa de amplificación

Para lograr la representación y evaluación de la señal de ECG, es necesario amplificar la señal ya que la medición que adquieren los electrodos es de tan solo algunos *millivolts*, haciendo necesaria una amplificación de alrededor de mil veces la amplitud de la señal original para lograr una buena cantidad de niveles en el conversor A/D para representar una señal suave (en la que no se notaran los niveles de la conversión) en la pantalla.

Como se discutió previamente, es posible definir una cierta amplificación de la señal a través de la combinación de diferentes valores de resistencias y capacitores con los circuitos integrados utilizados para adquirir y filtrar la señal, como es el caso con el amplificador de instrumentación y los filtros pasa bajos y pasa altos. Se definió que la amplificación a utilizar, con una ganancia a determinar con mediciones experimentales de alrededor de 1000 veces, sería implementada en tres fases, de este modo, no se saturaría la señal con una amplificación demasiado grande en una única etapa. Con el mismo propósito de evitar que la señal saturase en sus valores positivos y negativos con respecto a la referencia, se utilizaron amplificadores operacionales *rail to rail* que amplifican señales de ambas polaridades.

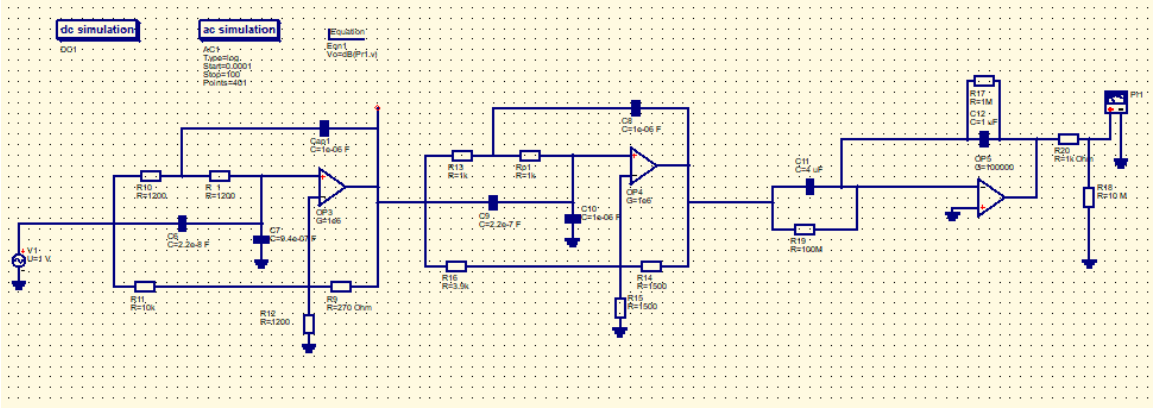


Figura 3.4.1: Representación en Qucs, de izquierda a derecha, las dos celdas del pasa bajos y la celda del pasa altos.

3.5 Circuito de alimentación

Para cumplir con los requisitos de alimentación de la pantalla que debe ser alimentada con 5V y al mismo tiempo tener un monitor portátil, se utilizó una batería de Litio de 9 Volts que entrega la corriente necesaria para que la pantalla y el resto de los componentes funcionen a la vez, la prueba con baterías normales no fue exitosa ya que no entregaban la corriente necesaria para mantener el sistema funcionando.

Para convertir la tensión de 9 Volts que entrega la batería a los 5 Volts con los que debe alimentarse el Arduino y el resto del circuito, se diseñó un conversor de voltaje utilizando un circuito integrado LM7805 (figura 3.5.1) conectado a dos capacitores como se muestra en el esquemático provisto en la figura 3.5.2 e implementado en la figura 3.5.3. Este circuito permitió cambiar efectivamente el voltaje que entrega la batería de 9V a 5V sin afectar la capacidad de la batería de proveer la corriente necesaria para que el sistema funcionara ya que el LM7805 puede entregar hasta 1A de corriente de salida.

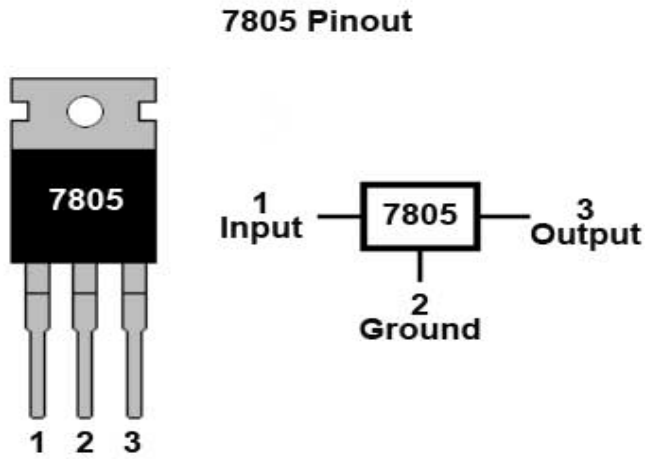


Figura 3.5.1: Esquemático de un conversor de voltaje 7805.

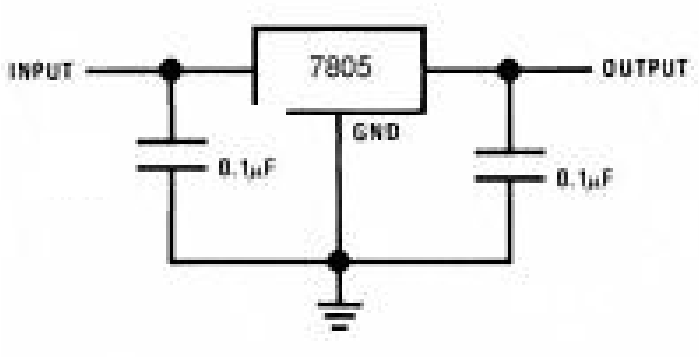


Figura 3.5.2: Circuito a utilizar con el conversor de voltaje 7805 para conseguir una salida de 5V con el detalle de los valores de los capacitores necesarios.

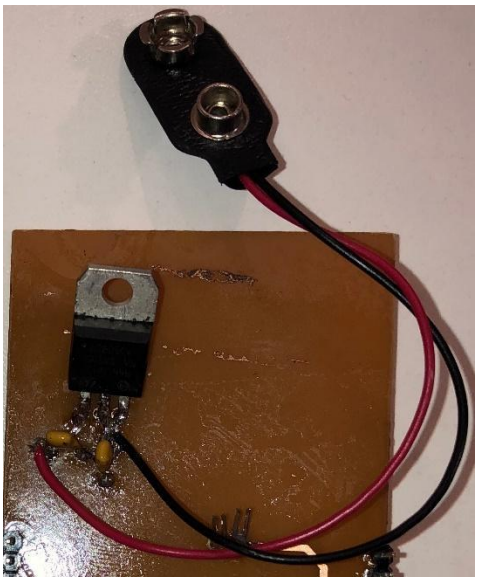
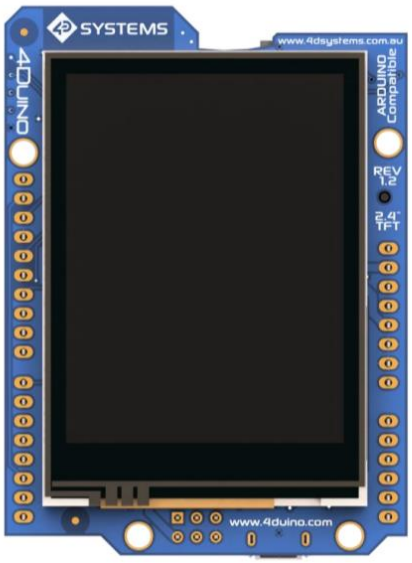


Figura 3.5.3: Circuito integrado LM7805 conectado a los dos capacitores de 0.1µF y al conector de las baterías.

3.6 Conversión digital y visualización

Para tomar la señal analógica devuelta por las etapas previas de acondicionamiento y convertirla en una señal digital, se debe utilizar un conversor A/D. Este circuito permite tomar una señal contínua y muestrearla en una cantidad de escalones discretos que permiten el procesamiento digital de la misma. Se decidió que mínimamente este conversor deberá tener 8bits de resolución para tener suficiente resolución para filtrar la señal digitalmente. Con este fin, se consideró la utilización de un microprocesador que llevase a cabo la tarea de muestrear la señal analógica y que pudiera luego comunicarse con un *display* para visualizar la señal. Se decidió que lo óptimo sería encontrar una plataforma que integrase el microprocesador con el conversor A/D, un procesador gráfico para crear la imagen que se mostraría en el *display* y el la pantalla en sí.

Dadas las condiciones expuestas, se consideró la utilización de un módulo 4Duino [14], una placa que incluye todos los requisitos que se plantearon en la concepción del Proyecto para representar la señal del corazón obtenida por los electrodos en la pantalla LCD táctil incluida con el Arduino como puede verse en la figura 3.6.1. El módulo 4Duino incluye un conversor A/D de 10 bits que sería suficiente para muestrear la señal, con un microcontrolador ATmega2560 de 8 bits y la integración con un módulo de procesamiento gráfico interno.



2.4" colour TFT LCD display, 240 x 320 Resolution, RGB 65K true to life colours, with integrated 4-wire Resistive Touch Panel.
Main processor is the Atmel ATmega32U4, and Graphics are powered by the feature-rich 4D Systems PICASO Graphics Processor.

Figura 3.6.1: Pantalla LCD táctil (con procesador gráfico dedicado) incluida en el 4Duino. [14]

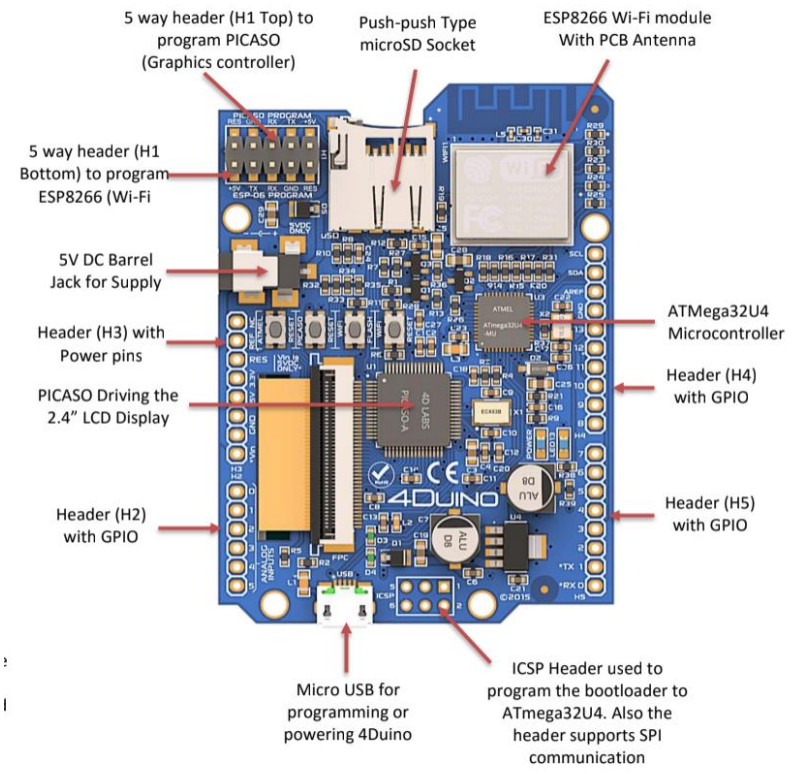


Figura 3.6.2: Componentes montados en la parte posterior de la placa 4Duino.[14]

3.7 Integración de la placa final

3.7.1 Diseño e impresión de placa PCB

Para el diseño de la placa PCB que se utilizó como parte del hardware para alojar los filtros analógicos, las etapas de amplificación y la alimentación, se recurrió al programa Altium Designer (v16.0) [15]. Este software permitió probar distintos diseños para optimizar el espacio en la placa y poder armarla virtualmente en distintos formatos mientras se iban incorporando y cambiando componentes sin tener que imprimir la placa cada vez que se cambiaba el *layout*. Durante este proceso se tomó la decisión de que la placa a diseñar sería de doble faz (contendría el circuito y sus componentes en ambas caras de la placa) para lograr circuitos eficientes en cuanto al espacio que ocuparían en el monitor.

En las figuras 3.7.1.1 y 3.7.1.2 puede verse el diseño final de la placa hecho en Altium en el cual se pudieron incluir cómodamente todos los circuitos propuestos previamente. La figura 26 muestra la parte superior de la placa, en la cual se soldaron los circuitos correspondientes a las etapas de amplificación y filtrado. En la contracara de la placa, representada en la figura 3.7.1.2, se trazaron algunas conexiones entre los filtros que no lograron hacerse en la cara superior ya que se superponían con otras conexiones y se incluyó el circuito del conversor de voltaje previamente expuesto y la batería. En el prototipo final no se utilizaron los porta pilas representados en la figura 3.7.1.2 ya que las pilas tipo “botón” resultaron no proveer la corriente

adecuada para la correcta alimentación del circuito, motivo por se colocó una batería en su lugar.

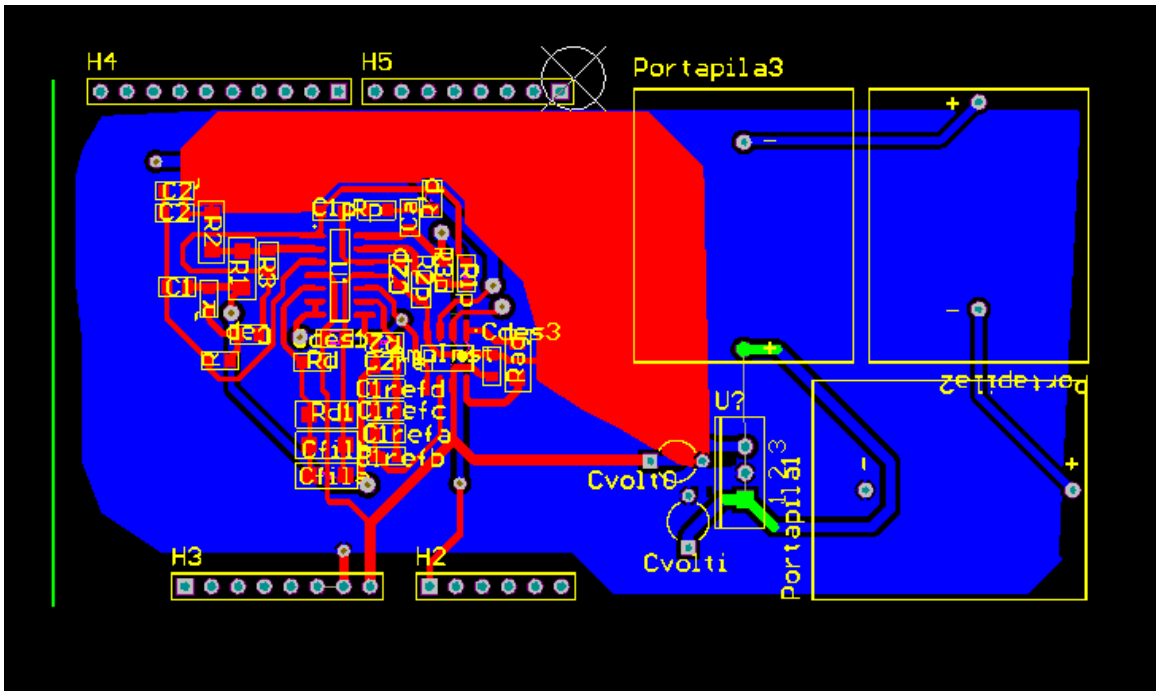


Figura 3.7.1.1: Vista superior del diseño final de la placa en Altium Designer.

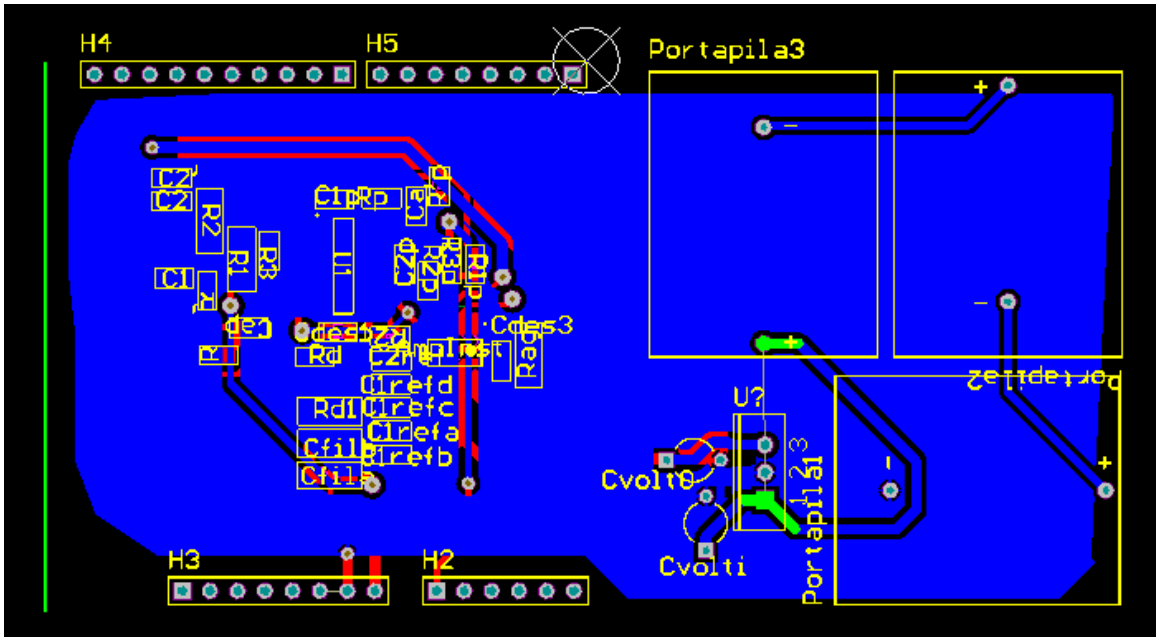


Figura 3.7.1.2: Vista inferior del diseño final de la placa en Altium Designer.

Para llevar la placa del entorno virtual a la realidad se recurre a la “impresión” de la placa en

el laboratorio. La técnica más común de prototipado para este propósito es, con una impresora láser imprimir sobre papel plástico el diseño obtenido del Altium. Luego, utilizando el calor de una plancha, se transfiere la tinta del papel plástico a una placa de cobre de la dimensión requerida y así se dibujan sobre la misma las conexiones diseñadas e impresas. Cumplidos esos pasos y verificando que las conexiones en la placa sean las deseadas, se la sumerge en ácido clorhídrico que desprenderá todo el cobre de la placa excepto el que se encuentra protegido por la tinta, dejando únicamente las conexiones requeridas.

Con la placa ya impresa, se pasó a realizar los agujeros de distintos tamaños requeridos para colocar los pines macho de conexión entre el circuito y el Arduino y los que se utilizarían únicamente para hacer mediciones sobre el funcionamiento del circuito. Para conectar la pantalla con la placa a las entradas correspondientes, se soldaron a la placa dos filas de pines macho tal como se muestra en la figura 3.7.1.3. Por otro lado, en la figura 3.7.1.4 se pueden observar los pines y *jumpers* que se utilizaron para poder extraer la señal del sistema en distintos puntos, como por ejemplo, podría extraerse la señal luego del pasa altos para probar su funcionamiento aislado del resto del sistema introduciéndole una señal de prueba por un pin antes del mismo extrayéndola por otro pin luego.

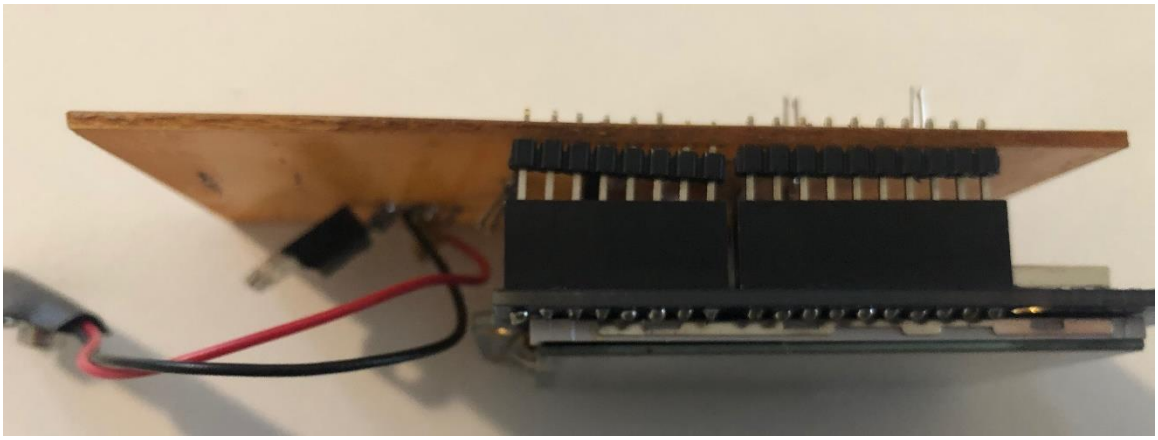


Figura 3.7.1.3: Foto de los pines macho soldados a la placa conectándose a los pines hembra del 4Duino.

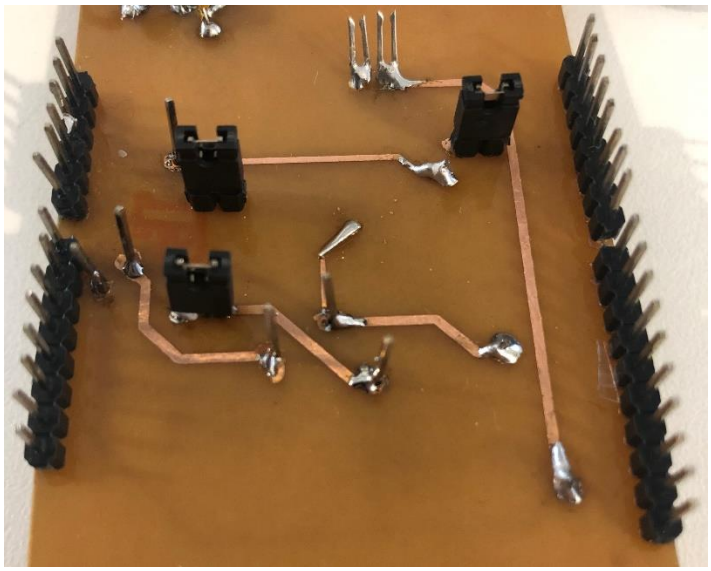


Figura 3.7.1.4: Detalle de los pines que se introdujeron para poder separar las etapas de procesamiento.

Como último paso, se soldaron los componentes electrónicos previamente definidos. En la figura 3.7.1.5 pueden observarse las primeras pruebas de diseño, impresión y soldadura junto a la placa final. En las figuras 3.7.1.6 y 3.7.1.7 se muestran los dos lados de la placa final con mayor detalle.

Figura 3.7.1.5: Abajo, placas de prueba de los filtros. Medio, placas de prueba. Arriba, placa final.

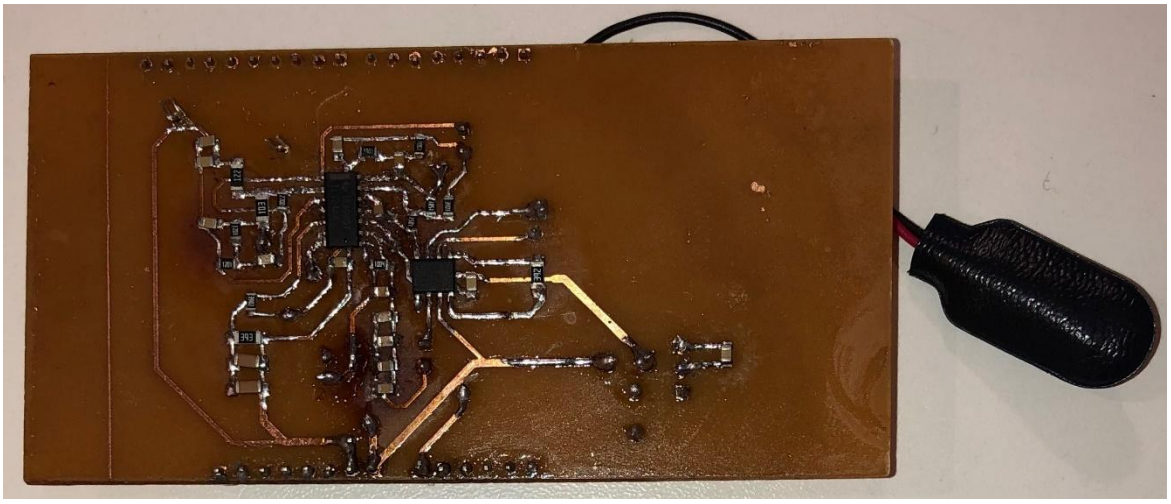


Figura 3.7.1.6: Vista superior de la placa terminada.

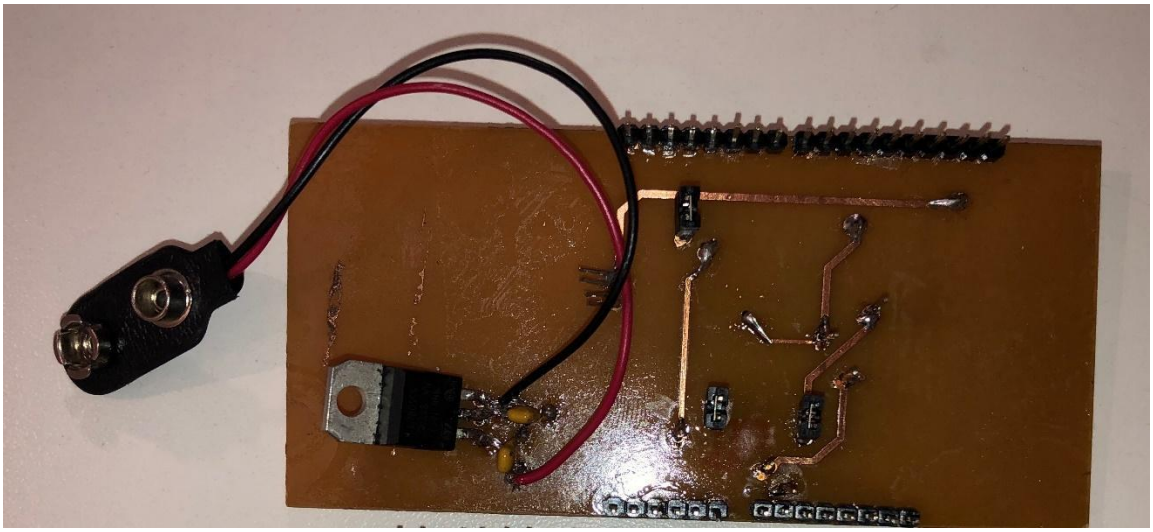


Figura 3.7.1.7: Vista inferior de la placa terminada.

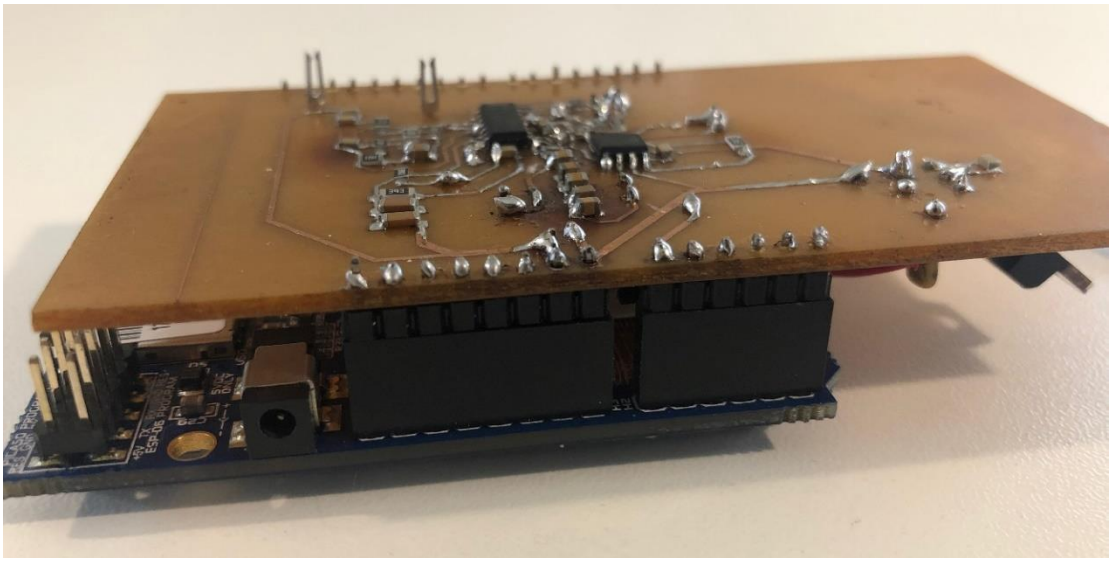


Figura 3.7.1.8: Foto de la parte superior placa montada al 4Duino por sus pines.

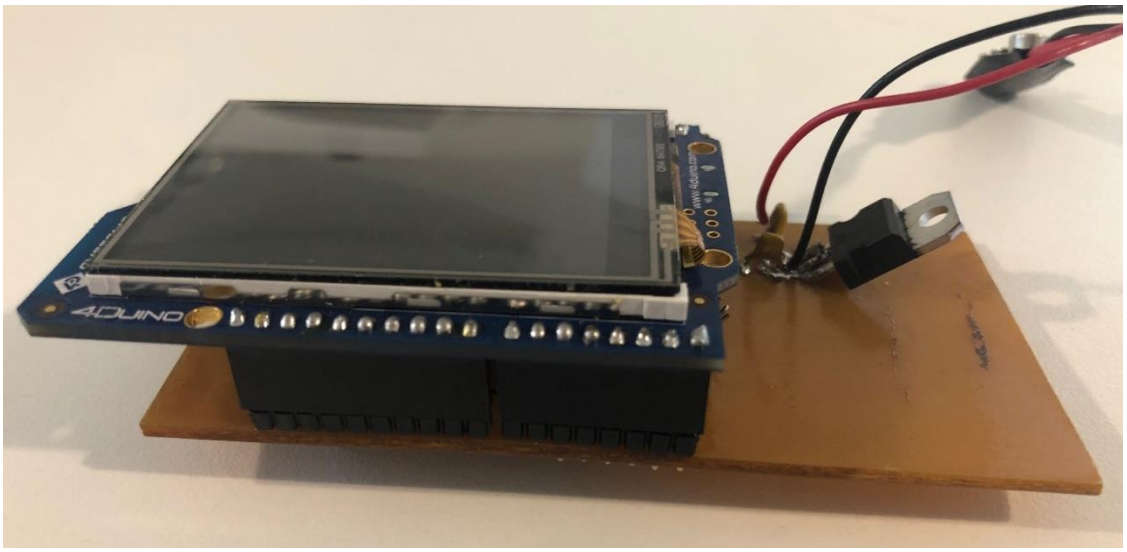


Figura 3.7.1.9: Foto de la parte superior placa montada al 4Duino por sus pines.

3.7.2 Pruebas de la placa integrada

Luego de haber conseguido integrar las etapas previamente diseñados en una placa PCB y de haber logrado generar la conexión entre la placa y el 4Duino, se pasó a realizar pruebas al conjunto para verificar el funcionamiento de las etapas por separado y en conjunto.

Las pruebas fueron llevadas a cabo a través de los pines y *jumpers* que se introdujeron a la placa con este fin (figura 3.7.1.4). A través de los *jumpers* se logra desacoplar una etapa de la próxima y utilizando los pines para medir con el osciloscopio o introducir señales con un generador de ondas.

Las pruebas del circuito de alimentación resultaron exitosas, midiéndose la tensión requerida de 5V a la salida del conversor de tensión. Las pruebas por separado del amplificador de instrumentación y de las etapas de filtrado también resultaron exitosas cuando se las midió por separado introduciendo una señal en la entrada del amplificado con un generador de onda y midiendo la salida de cada etapa con el osciloscopio.

3.7.3 Pantalla LCD

Al ser 4Duino un módulo que integra una pantalla con un procesador gráfico y un Arduino, fue necesario aprender a usar el programa que hace de interfaz entre las dos plataformas, tarea que se llevó a cabo con los manuales de usuario del fabricante. El programa se llama "WORKSHOP4" de 4D Systems [16] y facilitó la tarea de crear una interfaz gráfica de una manera intuitiva para cargarla a la memoria interna de la pantalla para que el Arduino desde su IDE pudiera comunicarse con la misma.

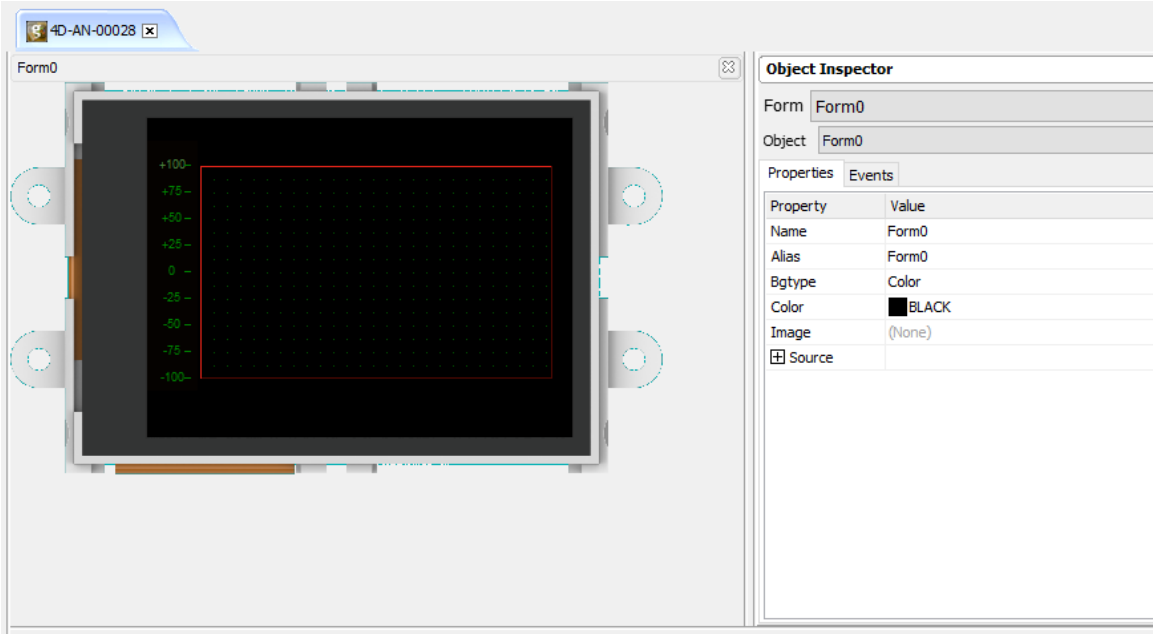


Figura 3.7.3.1: Entorno de programación de la pantalla que se cargó a la pantalla del 4Duino [17].

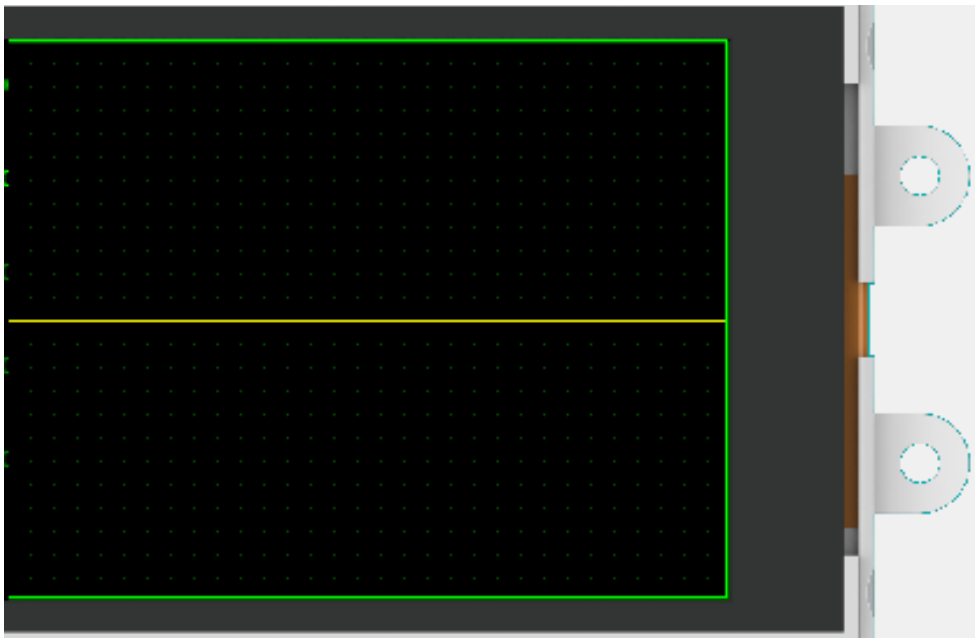


Figura 3.7.3.2: Captura de pantalla en el entorno del Workshop 4 del diseño de la interfaz gráfica que fue cargada al *display.* La línea central amarilla marca la referencia del ECG.

Se estableció una conexión por un puerto serie interno del módulo entre la pantalla y el Arduino para que el Arduino pudiera adquirir los datos analógicos. La conexión se llevó a cabo utilizando el paquete "genieArduino.h" [18] que se integra a la IDE de Arduino y permite enviar los datos requeridos del conversor A/D del Arduino al procesador AtMega, procesar la información y enviarla a la pantalla. Para ver la implementación del código que envía los datos obtenidos a la pantalla del Arduino ver Anexo 9.

3.8 Implementación en Arduino

Arduino es una placa que resulta principalmente de la combinación entre un microcontrolador AtMega 2560, una serie de puertos para la entrada y salida que permiten la conversión de señales analógicas a digitales y un circuito que permite alimentar la placa a través de USB. Para programar el microcontrolador, Arduino cuenta con una IDE dedicada en la cual se programa en lenguaje C y tiene integrados o pueden instalarse fácilmente paquetes desarrollados por usuario del entorno que permiten llevar a cabo tareas de procesamiento y adquisición de la señal de una forma efectiva y relativamente simple a través de funciones que cumplen ciertas tareas predeterminadas. Para utilizar las funciones, se deben introducir los parámetros que solicita para el procesamiento de la señal y sin tener que programar el funcionamiento básico de la tarea, la función lleva a cabo el procesamiento buscado con los parámetros diseñados.

En el monitor, se utilizó Arduino para convertir la señal cardíaca de analógica a digital a través de su conversor de 10 bits y procesar los datos adquiridos a través de los electrodos con su microcontrolador AtMega 2560 para luego enviar los paquetes de datos generados al procesador

gráfico de 4Duino para visualizarlos en el *display.*

Para la programación del Arduino se utilizan dos funciones principales, “setup()” y “loop()”. La función “setup()” es ejecutada una única vez por el microcontrolador al iniciar su funcionamiento, o luego de ser reiniciado, con el objetivo de inicializar las variables, pines y librerías que serán utilizadas para realizar la tarea indicada. Luego de haber inicializado los objetos necesarios, a través de la función “loop()” se introduce, como su nombre lo indica, un *loop* con un conjunto de acciones que serán repetidas una y otra vez como se definieron en el programa.

3.8.1 Sistema de adquisición

Para el sistema de adquisición de los datos en Arduino se utilizó la librería “TimerThree” [25], que permite medir repetitivamente un período de tiempo en microsegundos y llevar a cabo alguna acción deseada, en este caso, tomar datos del puerto de entrada A0 del Arduino y guardarlos en un vector. Como se discutió previamente, los datos fueron muestreados con una frecuencia de 1000Hz, razón por la cual la función para inicializar “TimerThree” se inicializa en el código del Anexo 7 con un tiempo de 1000 microsegundos, para de estar forma, tomar datos del puerto con la frecuencia de muestreo requerida. El código a continuación muestra el uso de la librería para la adquisición de los datos del puerto analógico.

```
Void setup() {
Serial.begin(115200);
Serial1.begin(115200) ;
Timer3.initialize(1000);
Timer3.attachInterrupt(rts); }
```

La primer línea del código indica que se está en la función “setup()” con la función expuesta al inicio de la sección 3.8. La función Serial.begin() inicializa la conexión interna entre los puertos del Arduino y el microcontrolador y Serial1.begin() inicializa la conexión del puerto serie 1 entre Arduino y el procesador gráfico del *display*, en este caso con un *baud rate* (número de datos enviados por segundo) de 115200 datos/segundo. Luego se inicializa la librería “TimerThree” con un tiempo entre que se repite la función de 1000 microsegundos= 0.001 segundos lo que equivale a una frecuencia de 1000Hz. Por último, se utiliza la función “Timer3.attachinterrupt(rts)” para introducir la función “rts” cada vez que se reinicie el *loop* introducido por el *timer* y que se ejecute el código a continuación, guardando de este modo los datos adquiridos en un *buffer* y enviándolos luego al puerto serie en comunicación con el AtMega o con la pantalla según corresponda.

```
void rts(){
  static unsigned int pos=0;
  static double t =0;
  buffer1[pos++]=analogRead(A0);
  Serial.println(buffer1[pos-1]);
```

```
t = t+0.001;
if(pos==N)
{
  for(unsigned int i=0 ; i<L;i++)
    x[i]=x[N+i];
  for(unsigned int i=0 ; i<N;i++)
    x[L+i]=buffer1[i];
    pos=0;
    bufferFlag=true; } }
```

Luego de haber adquirido la señal, se pasa a filtrar los datos digitalizados, con la frecuencia de muestreo diseñada, para luego poder armar paquetes de datos y a través del puerto serie que se inicializó, poder enviar los datos ya procesados a donde corresponda. Si no se buscara más procesamiento que la adquisición de los datos luego de pasar por los filtros analógicos, la señal podría enviarse directamente a la pantalla sin pasar por el procesamiento de la sección 3.8.2 a través de código expuesto en el Anexo 9.

3.8.2 Filtro digital en tiempo real

Para el diseño de los filtros digitales, se utilizó la herramienta *Filter Design Tool* de Matlab. Este paquete permite diseñar filtros con las especificaciones de tipo, orden y frecuencias a filtrar y devuelve una respuesta en frecuencia que cumple con las especificaciones lista para implementar en un código de Arduino.

En este caso, se buscó crear dos filtros, un pasa bajos que filtrara las frecuencias altas que no contribuyen al análisis del ECG y dos filtros Notch para filtrar el ruido de línea de 50/60 Hz dependiendo del país en donde se encuentre el monitor.

3.8.2.1 Filtros Notch IIR

En las figuras a continuación se observan los diagramas de bode y de polos y ceros de los filtros Notch que atenúan la señal en 50Hz y 60Hz respectivamente. Notar en los diagramas de Bode de las figuras 3.8.2.1.1 y 3.8.2.1.4 que la fase de los filtros no es lineal ya que se utilizó un filtro IIR que desfasa la señal. Se eligió un filtro IIR aunque afecte la fase de la señal ya que cumple con las especificaciones impuestas con un orden mucho menor, y por ende con mayor velocidad, que un filtro del tipo FIR que no afecta la fase de la señal de entrada pero en este caso hubiera requerido un orden de alrededor de 140 para cumplir con los requisitos buscados.

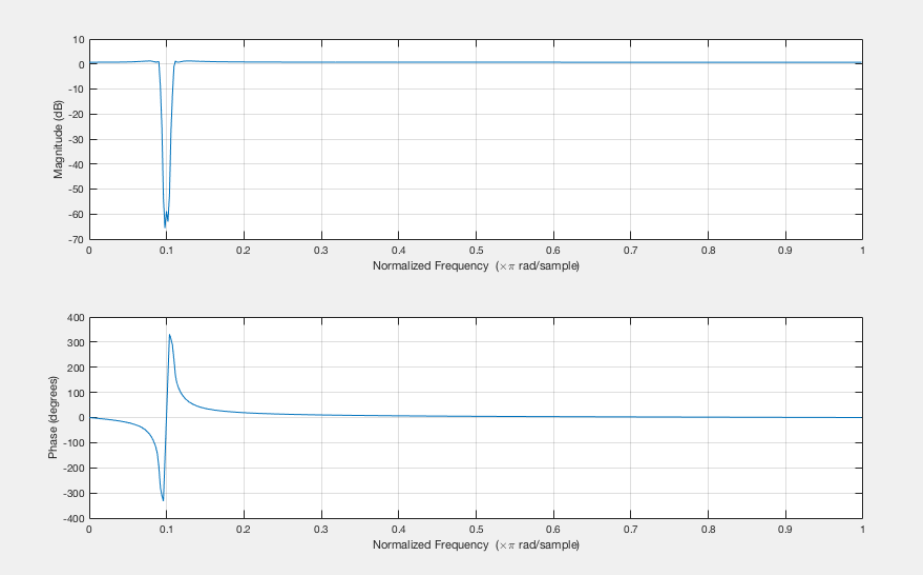


Figura 3.8.2.1.1: Diagrama de Bode de magnitud y fase del filtro Notch en 50Hz.

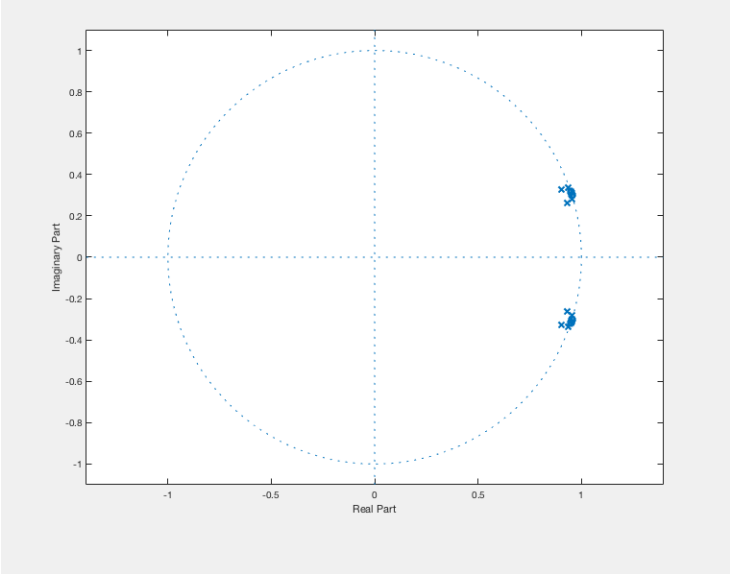


Figura 3.8.2.1.2: Diagrama de polos y ceros del filtro Notch en 50Hz.

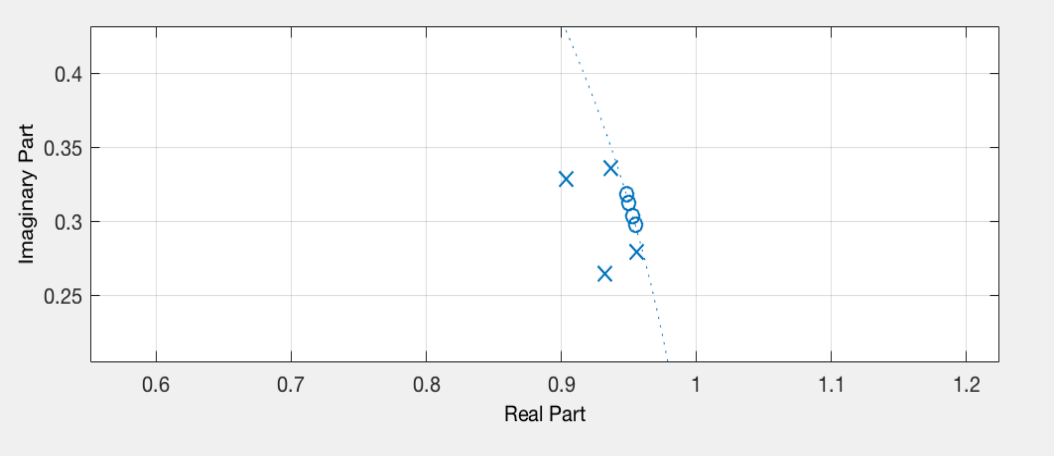


Figura 3.8.2.1.3: Detalle del diagrama de polos y ceros del filtro Notch en 50Hz.

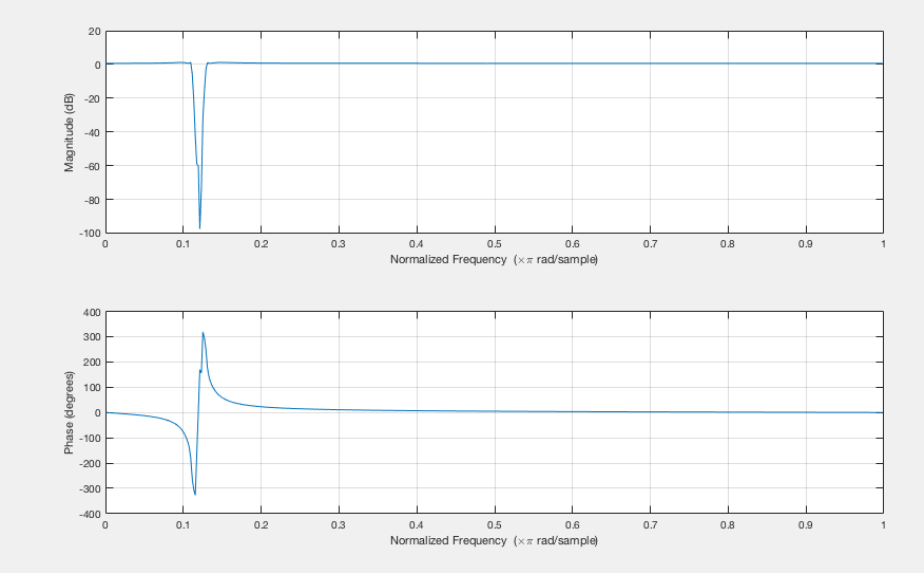


Figura 3.8.2.1.4: Diagrama de Bode de magnitud y fase del filtro Notch en 60Hz.

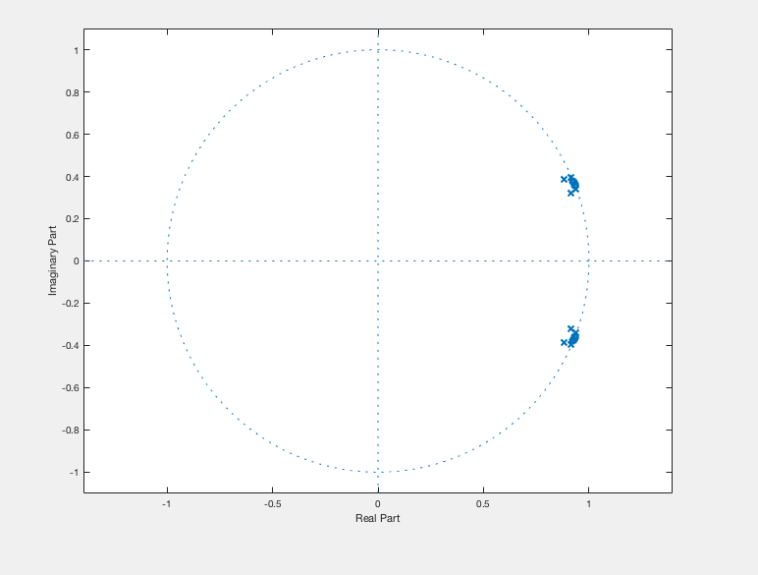


Figura 3.8.2.1.5: Diagrama de polos y ceros del filtro Notch en 60Hz.

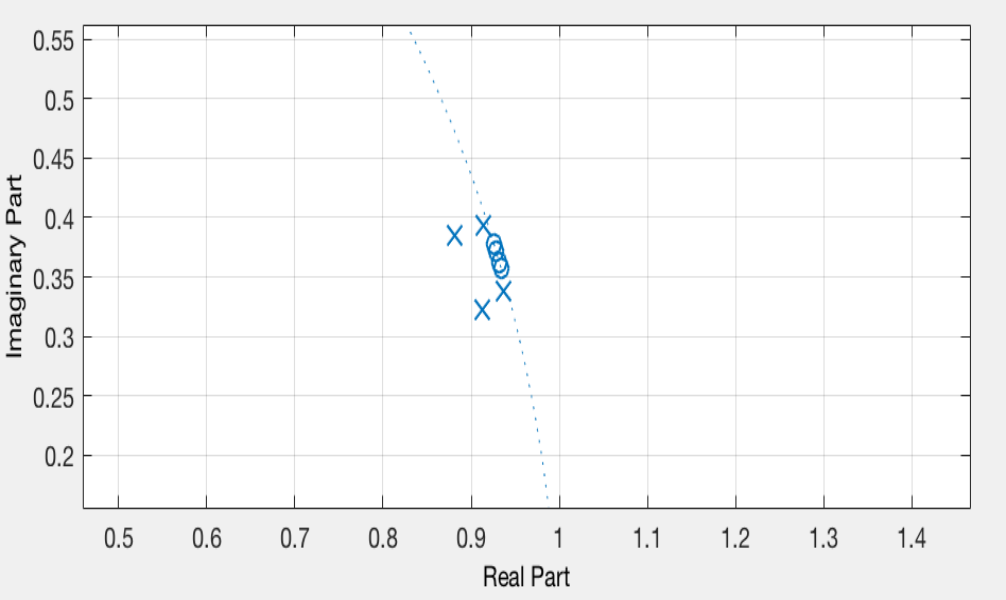


Figura 3.8.2.1.6: Detalle del diagrama de polos y ceros del filtro Notch en 60Hz.

Para el diseño del filtro se tuvo en cuenta que el conversor A/D del Arduino convierte la señal con un escalonamiento de 10 bits y que utiliza un *clock* interno con una frecuencia de 16MHz, equivalente a un tiempo de 62.5 ns, por lo que se tiene un microcontrolador capaz de llevar a cabo la adquisición de los datos con un *baud rate* de 1000000bps y al mismo tiempo poder procesar la señal y enviarla al procesador gráfico para ser impresa en la pantalla en tiempo real.

Para la implementación de los filtros Notch en tiempo real, se decidió utilizar un filtro del tipo IIR implementado por el método de ventana ya que de esta forma se logra obtener una señal filtrada en tiempo real dado que las operaciones que utiliza este tipo de filtro son

computacionalmente simples y por ende se llevan a cabo en poco tiempo. Los filtros IIR funcionan realizando un promedio ponderado de la señal que se desea filtrar, esto se lleva a cabo utilizando un vector con los coeficientes del filtro que multiplica al vector de entrada que se busca modificar.

Para diseñar los filtros Notch que deberían filtrar las frecuencias que introduce el ruido de línea a la medición, se ingresaron en la interfaz de *Filter Design Tool* los siguientes parámetros de frecuencia de muestreo: las frecuencias en el entorno de la que se busca atenuar, la atenuación que se busca en la banda de rechazo, el tipo de filtro y el orden requerido. Los parámetros utilizados para generar estos filtros se encuentran en el código expuesto en los anexos 3 y 4 para el filtro que rechaza 50Hz y el que rechaza 60Hz respectivamente.

Utilizando la *Filter Design Tool* de Matlab, se obtuvieron los parámetros necesarios para implementar los filtros Notch en la Arduino IDE. Ambos filtros IIR diseñados para cumplir con las especificaciones impuestas resultaron ser de orden 8 y su implementación en una etapa como se muestra en el código a continuación resultó ser inestable por lo que debió buscarse una solución para implementar con un algoritmo estable.

```
void loop() {
while (i<20000)
{
datain[i]=analogRead(0);
x=datain[i];
y=b0*x + b1*x1 + b2*x2+b3*x3+b4*x4+b5*x5+b6*x6+b7*x7+b8*x8 - a1*yl - a2*y2 - a3*y3-
a4*y4- a5*y5- a6*y6- a7*y- a8*y8- a9*y9;
y9=y8;y8=y7;y7=y6;y6=y5;y5=y4;y4=y3;y3=y2;y2=yl;yl=y;
x8=x7;x7=x6;x6=x5;x5=x4;x4=x3;x3=x2;x2=x1;x1=x;
dataout[i] = y;
Serial.println(dataout[i]);
i=i+1;
}
```

Habiendo comprobado que la implementación del filtro en una sola etapa de orden 8 resulta inestable, se buscó implementar el filtro en 4 etapas de orden 2 y para ello, se investigó acerca de la implementación con el tipo de filtro “Direct Form II” que permite implementar filtros de orden 2 en serie y de este modo obtener una salida de orden 2n siendo n la cantidad de etapas requeridas. El diagrama que representa el filtro que se buscó aplicar puede ser apreciado en la siguiente figura.

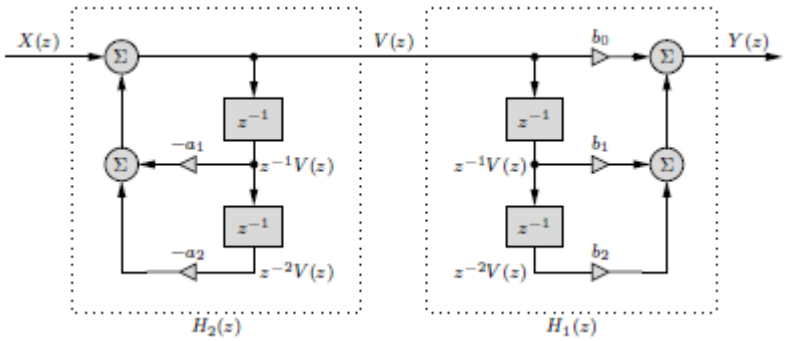


Figura 3.8.2.1.5: Diagrama que representa el filtro *Direct Form II* con los retrasos y coeficientes correspondientes. [26]

El diagrama anterior puede ser expresado como una función de transferencia H(z) que modifica a la señal de entrada X(z) convirtiéndose en la señal de salida Y(z). La función H(z) es la siguiente:

$$H(z) = \frac{Y(z)}{X(z)} = \frac{b_0 + b_1 z^{-1} + b_2 z^{-2}}{1 + a_1 z^{-1} + a_2 z^{-2}}$$

En la formula precedente, b_0, b_1, b_2, a_1 y a_2 representan los coeficientes del filtro y z^{-1} y z^{-2} representan el retraso temporal que se le aplica a la señal. A partir de esta función de transferencia se llega a la ecuación que representa la salida de un filtro IIR de orden 2 como puede observarse en la siguiente ecuación:

$$Y(z) = X(z)b_0 + X(z)b_1 z^{-1} + X(z)b_2 z^{-2} 1 - Y(z)a_1 z^{-1} - Y(z)a_2 z^{-2}$$

Si se aplica la misma lógica a la salida Y(z) nuevamente y se la multiplica por la misma función de transferencia, se crea una serie de los filtros, que serán la base de la lógica que luego se intenta replicar en el código del Arduino, para así obtener un filtro de orden 8 aplicando 4 veces un filtro de orden 2 con diferentes coeficientes. El siguiente diagrama ilustra la iteración de multiplicar a Y(z) por funciones de transferencia como la H(z) expuesta anteriormente repetidas veces, generando un filtro de orden mayor a 2k.

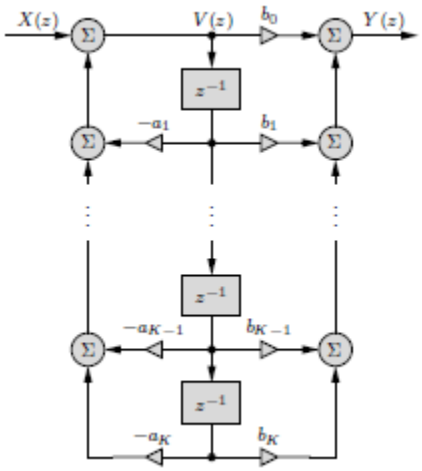


Figura 3.8.2.1.6: Diagrama que representa el filtro *Direct Form II* aplicado k veces a una misma entrada X(z). [26]

En la aplicación, los coeficientes del filtro a implementar fueron obtenidos exportando la matriz con los coeficientes del filtro desde Matlab en la forma "Direct Form II" para poder implementarlo en las 4 etapas. A continuación, se detallan 2 de las 4 etapas del filtro Notch de 50Hz aplicado en Arduino, las otras dos etapas siguen la misma lógica y junto con el código que lleva a cabo toda la operación de filtrado digital en Arduino y permite imprimir los valores en pantalla para corroborar su funcionamiento pueden ser observados en el Anexo 7.

```
//Stage 1
 y0 = 1 * x - 1.9002 * x1 + 1 * x2 + 1.8067 * y1 - 0.9242 * y2;
 y2 = y1;
 y1 = y0;
 //Stage 2
y00 = 1 * x - 1.9059 * x1 + 1 * x2 + 1.8644 * y1 - 0.9390 * y2;
 y2 = y1;
 y1 = y00;
```

Para comprobar que la implementación del filtro a través de la fórmula utilizada fuera consistente con el filtro buscado, se crearon en Matlab funciones senoidales para representar el ruido de línea y se lo filtró comparando la salida del filtro en cada etapa de la serie con la función "filter()" de Matlab utilizando los coeficientes del IIR diseñado. El código utilizado para generar las figuras a continuación y probar el comportamiento del filtro en distintos escenarios se encuentra en los anexos 10 y 11. Como puede observarse en las figuras a continuación, la respuesta del filtro con los mismos coeficientes aplicados por la función de Matlab o por la fórmula, tienen la misma respuesta y logran atenuar el ruido de línea cada vez más según se incrementa el orden del filtro hasta casi eliminar la frecuencia no deseada.

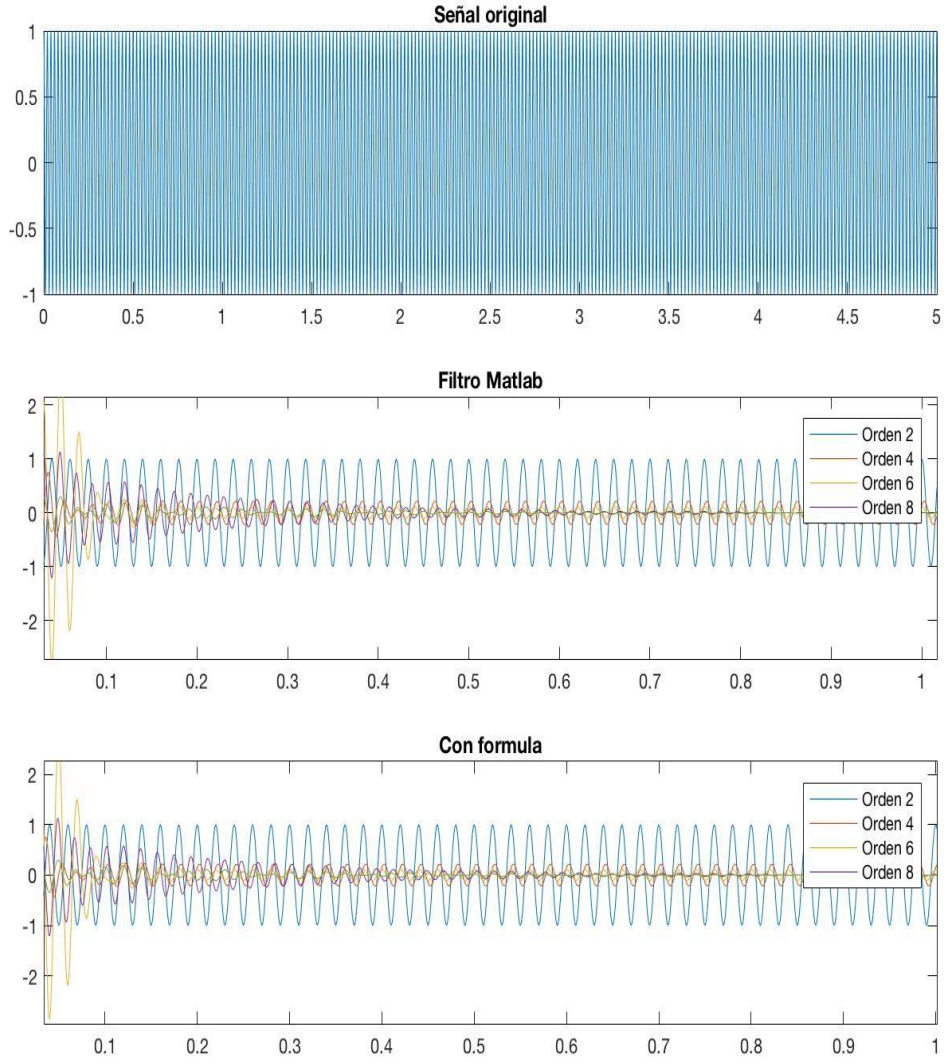


Figura 3.8.2.1.7: Respuesta del filtro IIR de orden creciente (según se van cumpliendo iteraciones de los bloques de orden 2) a un coseno con frecuencia 50Hz implementado con la fórmula del filtro o con la función “Filter()”.

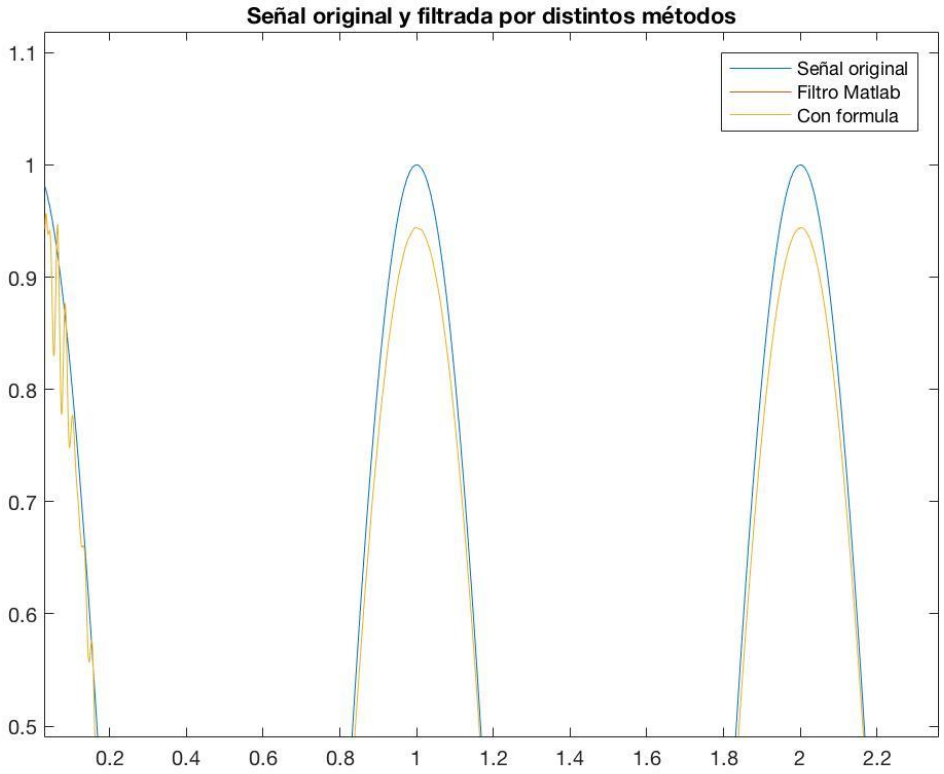


Figura 3.8.2.1.8: Comparación del filtro de orden 8 aplicado a la señal de entrada cosenoidal de 1Hz filtrada con la función de Matlab y con la fórmula del filtro.

Para simular el comportamiento del filtro como realmente se implementaría en el Arduino, se utilizó el código contenido en el anexo 12 que implementa el filtro sobre la señal de entrada tomando un bloque de muestras, filtrándolo este número de muestras, graficándolo y repitiendo el proceso continuamente. El filtro resultó ser, como se muestra en las figuras a continuación, adecuado y logra filtrar casi completamente el ruido de línea de 50Hz mientras que deja pasar el resto de las señales casi sin cambios.

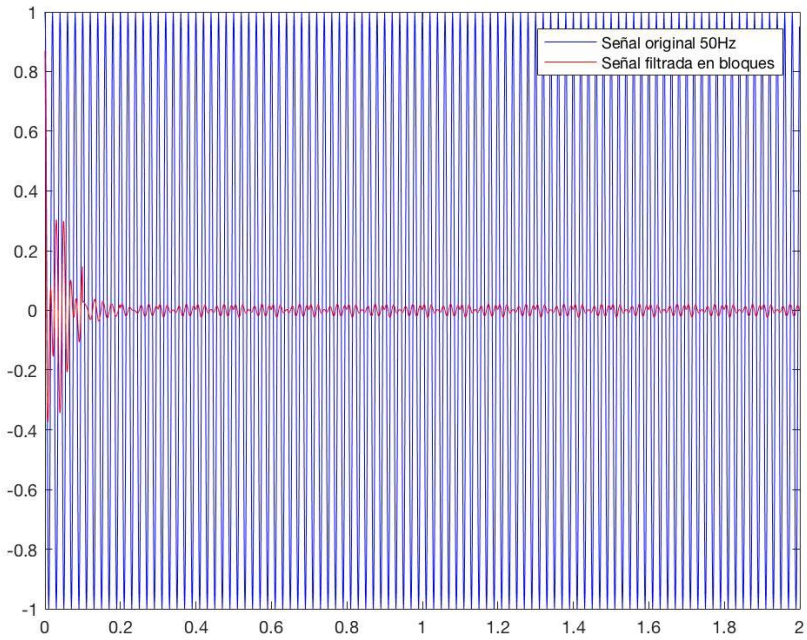


Figura 3.8.2.1.10: Filtro IIR de orden 8 aplicado a una señal cosenoidal de 50Hz de a 100 muestras.

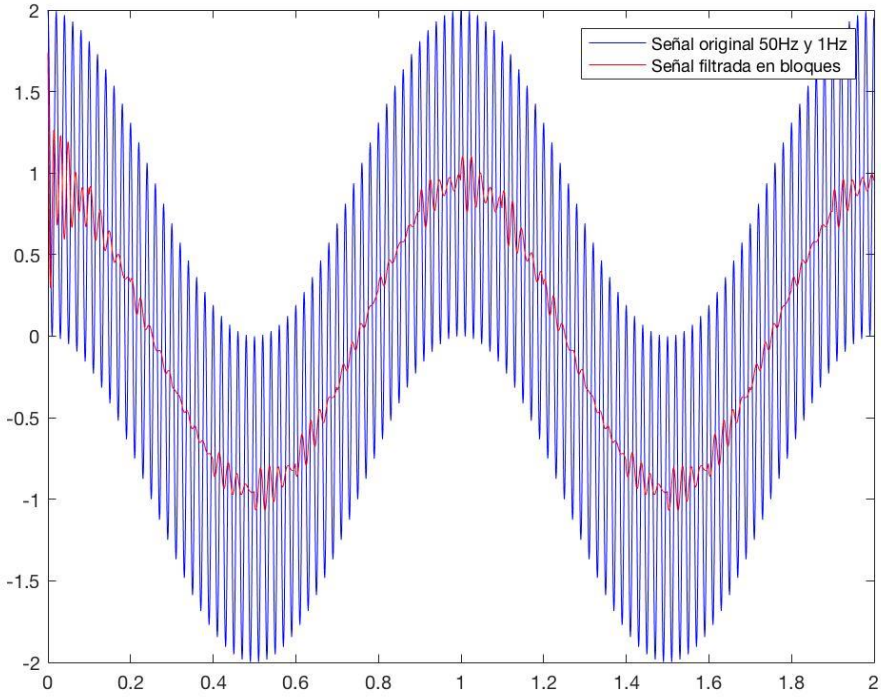


Figura 3.8.2.1.11: Filtro IIR de orden 8 aplicado a una señal cosenoidal de 50Hz sumado a una señal cosenoidal de 1HZ de a 100 muestras.

3.8.2.2 Filtros Decimador IIR

Luego se implementó, también como un IIR, un filtro pasa bajos decimador para muestrear la señal de manera tal que la cantidad de puntos graficados en la pantalla permitiesen mostrar una imagen contínua y representativa de la señal obtenida. Dado que la pantalla tiene una cantidad finita de píxeles para mostrar, no tendría sentido que se procese y grafique toda la información de la señal sino que basta con mostrar 2 segundos, tiempo que le permite al procesador procesar los datos nuevos que le entran mientras que grafica los ya procesados. Con la implementación de este filtro, se evita el *alliasing* y se logra mostrar una señal representativa de la señal digitalizada.

El filtro decimador elegido fue un pasa bajos de Chebyschev de tipo 2 ya que fue el que menor orden requirió según su diseño con el *Filter design Tool* en Matlab (código en Anexo 6) este filtro también fue aplicado utilizando filtros del tipo *Direct Form II* de orden 2 en serie para conservar su estabilidad. La implementación del filtro, que toma una muestra de cada aproximadamente 3 puntos que digitaliza el Arduino puede verse en el Anexo 7.

El diagrama de polos y ceros que representa al filtro diseñado con su diagrama de magnitud correspondiente se muestra a continuación. Notar que no se representa el diagrama de la fase ya que ésta ya fue desfasada por las etapas anteriores de filtrado y por ende no contiene información valiosa.

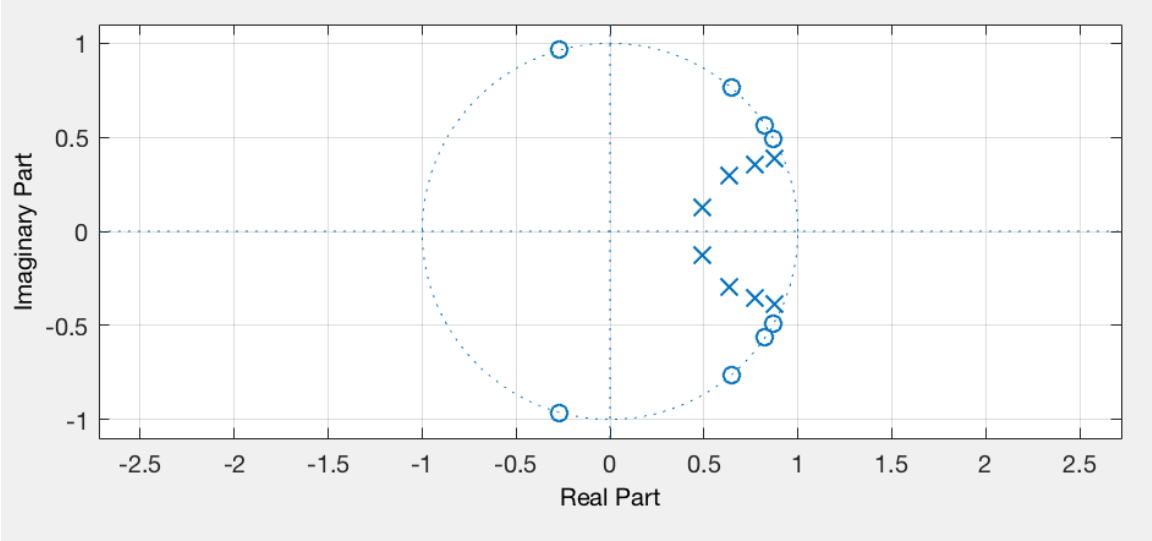


Figura 3.8.2.2.1: Diagrama de polos y ceros del filtro decimador de orden 8.

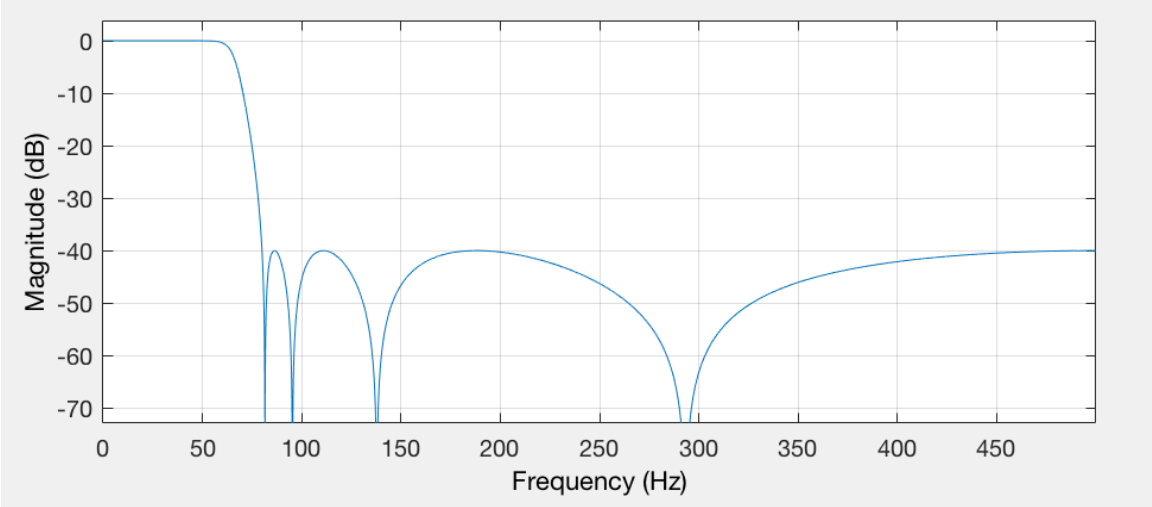


Figura 3.8.2.2.2: Diagrama de magnitud del filtro decimador de orden 8.

La respuesta de este filtro a distintas frecuencias puede apreciarse en las figuras siguientes, las mismas muestran la respuesta del filtro a una señal de entrada cosenoidal de 1Hz, 10Hz y 100Hz respectivamente para mostrar como efectivamente las bajas frecuencias pasan sin problema mientras que las altas frecuencias son filtradas.

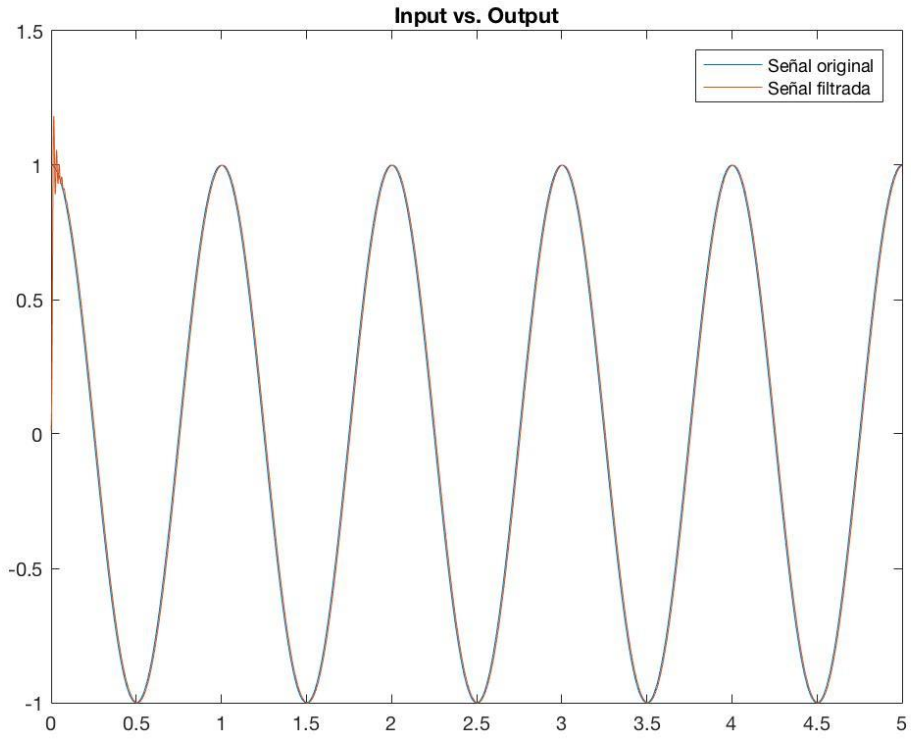


Figura 3.8.2.2.3: Comparación entre la señal original, un coseno de 1Hz, y la filtrada.

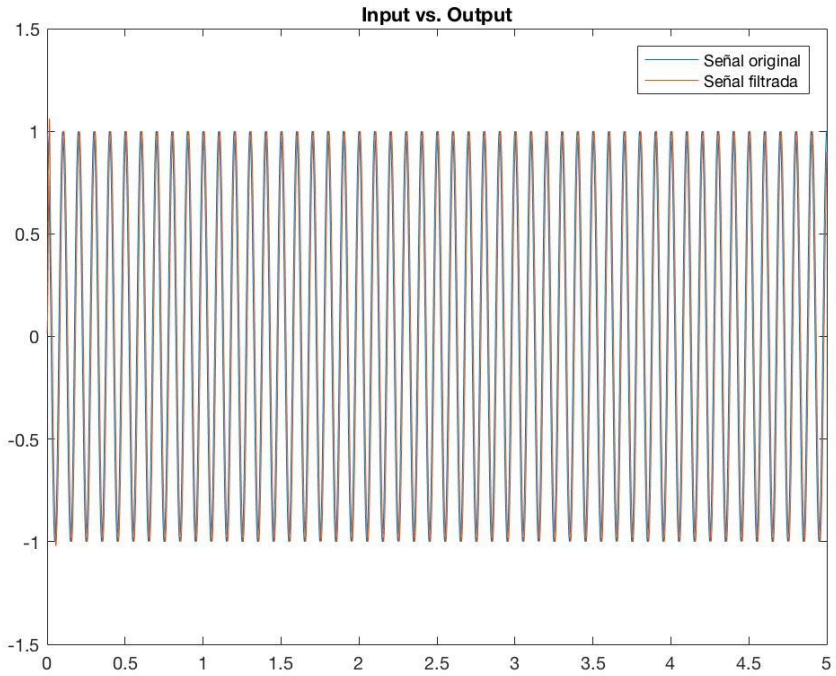


Figura 3.8.2.2.4: Comparación entre la señal original, un coseno de 10Hz, y la filtrada.

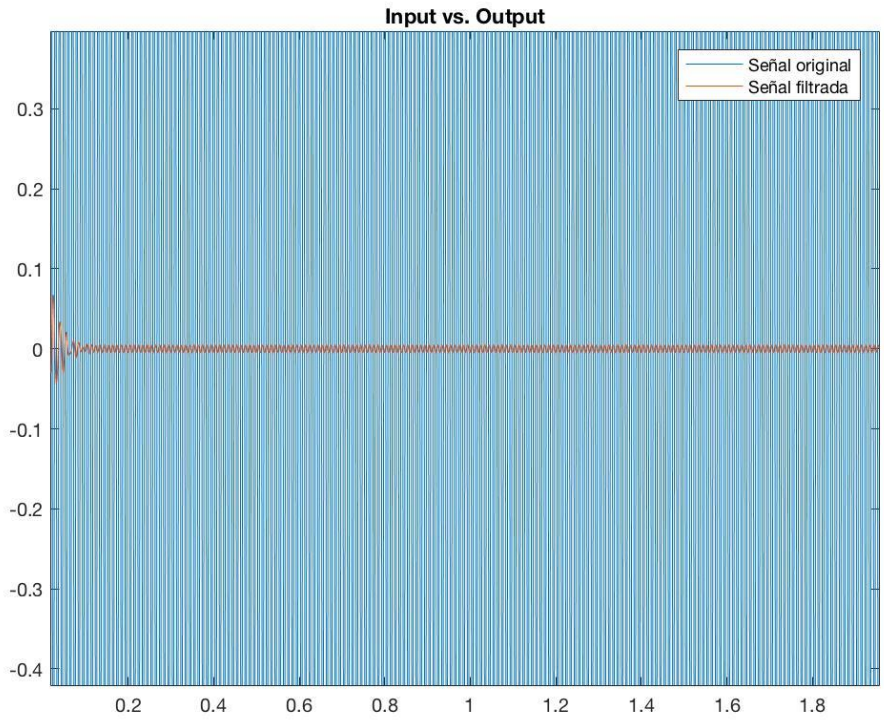


Figura 3.8.2.2.5: Comparación entre la señal original, un coseno de 100Hz, y la filtrada.

3.9 Diseño e impresión 3D del gabinete

Al ser la tecnología de impresión 3D tan accesible y práctica en cuanto a disponibilidad, diseño y costo, se decidió utilizar este método para crear el gabinete que contendría la placa PCB, el 4Duino, la batería y los electrodos del ECG. Esta decisión se basó en que se crearía un gabinete a medida y con los requerimientos necesarios y específicos para el monitor, El gabinete fue diseñado utilizando TinkerCad [19], un programa gratuito que ofrece una plataforma *online* amigable para realizar diseños en formato .stf, la extensión necesaria para la impresión 3D. El resultado del diseño se observa en las figuras 3.9.1 y 3.9.2.

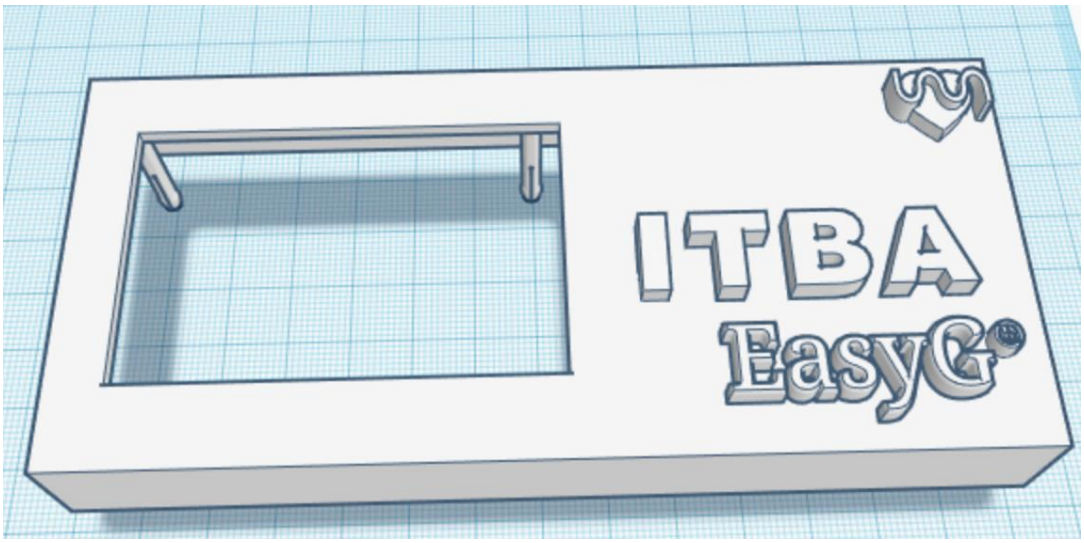


Figura 3.9.1: Vista superior del diseño del gabinete en TinkerCad.[19]

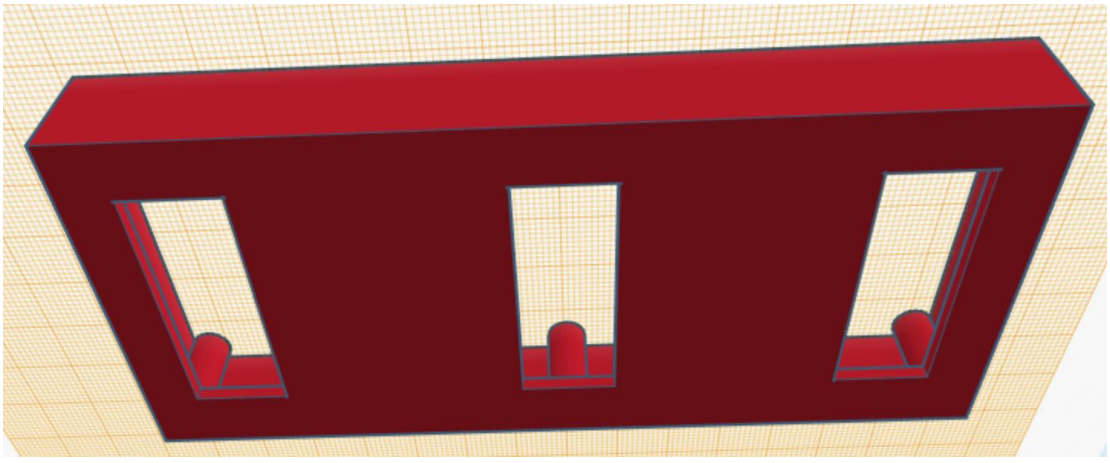


Figura 3.9.2: Vista inferior del diseño del gabinete en TinkerCad.[19]

El prototipo se imprimió en las impresoras del ITBA en material plástico ABS que es el material más común y de menor costo para prototipado. La resolución de la impresión resultante no fue de la mejor calidad en sus detalles pero cumplió con las especificaciones básicas requeridas,

contener los distintos componentes del monitor cardíaco, ser de un tamaño portátil e ilustrar como podría ser el gabinete si se fuese a utilizar una técnica industrial para fabricarlo. Las figuras 3.9.3 y 3.9.4 muestran el gabinete impreso con la resolución resultante.

Figura 3.9.3: Vista superior del gabinete impreso en el ITBA.

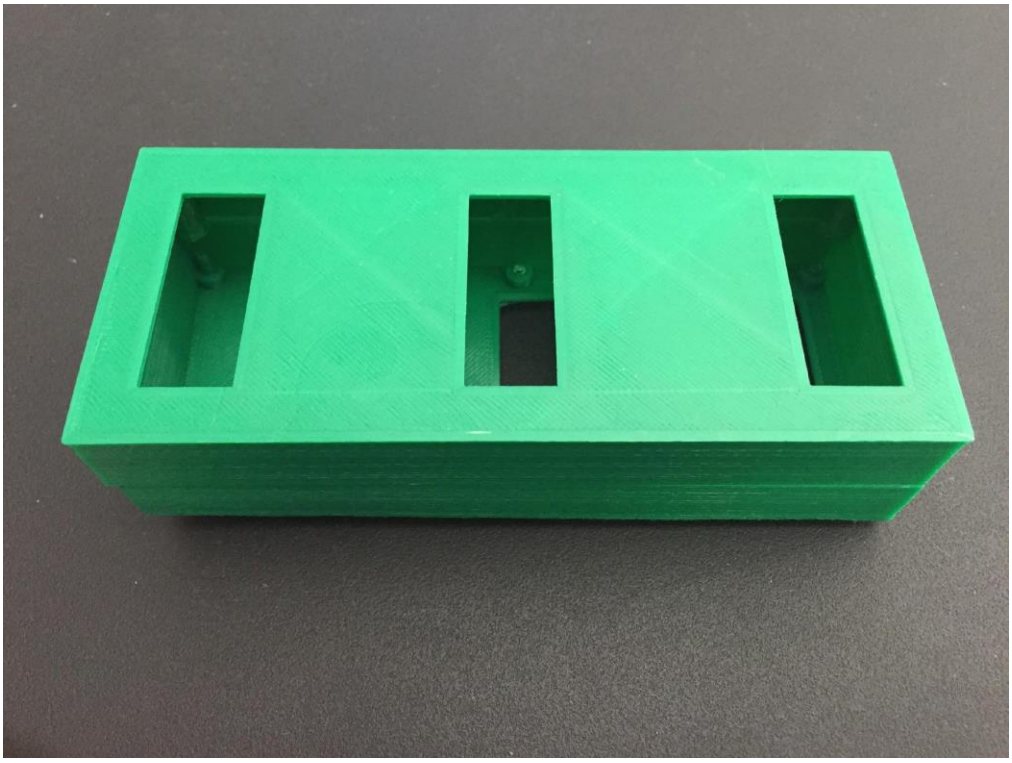


Figura 3.9.4: Vista inferior del gabinete impreso en el ITBA.

3.10 Ensamblaje final

En la última etapa del armado del monitor se pasó a conectar la pantalla y a la placa PCB completa a través de los pines macho de la placa y hembra del 4Duino (figura 3.10.1). Luego se conectó la batería (figura 3.10.2), dándole así la alimentación requerida a la pantalla y a la placa para iniciar su funcionamiento. Por último, se colocaron los electrodos en la parte inferior del gabinete y se colocó el conjunto previamente ensamblado por encima de los mismos, permitiendo así cerrar el gabinete con todos los componentes contenidos en el (figura 3.10.3)

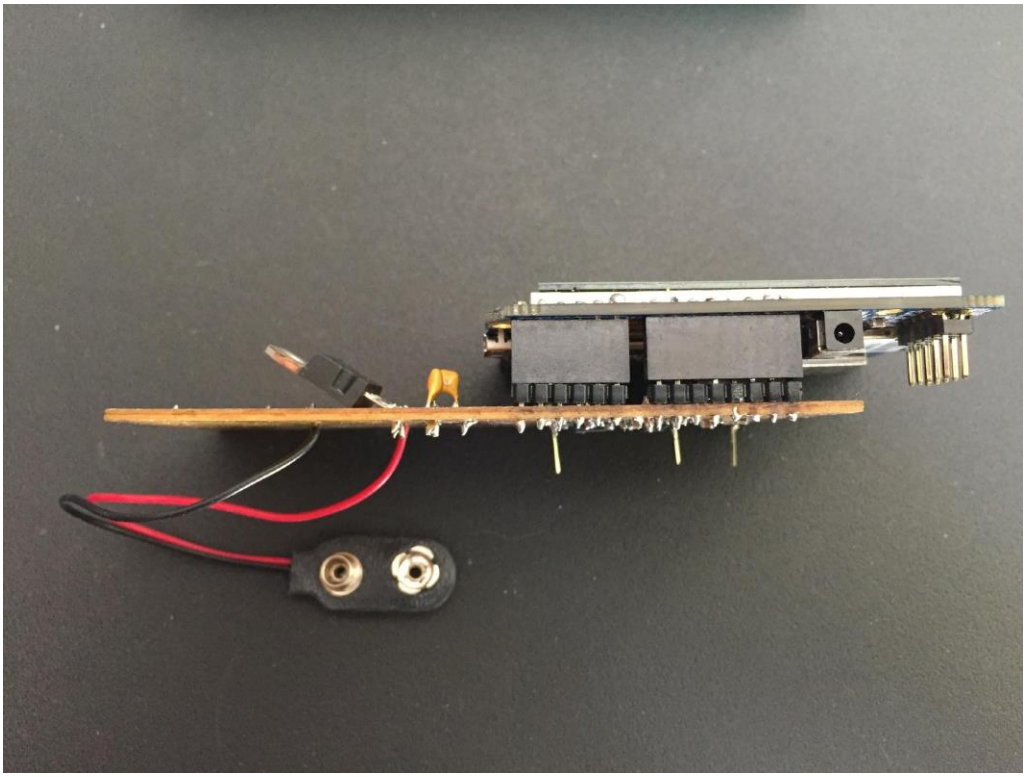


Figura 3.10.1: La placa conectada por sus pines macho a los pines hembra de la Pantalla.

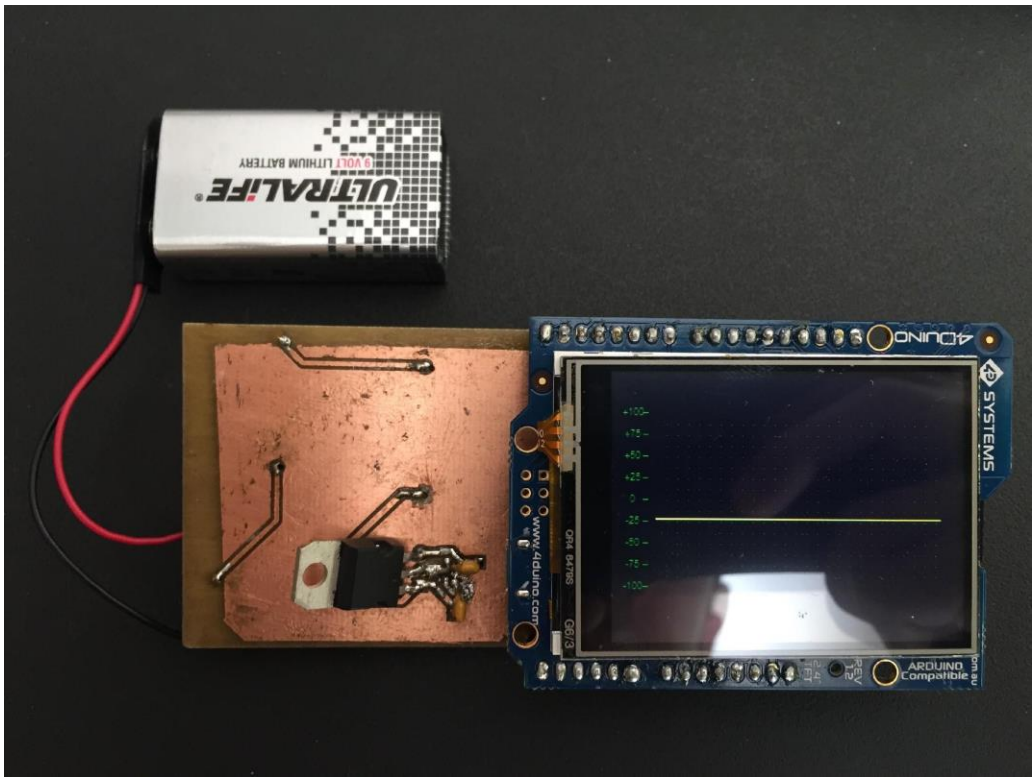


Figura 3.10.2: La placa y el 4Duino funcionando conectadas a la batería con el *scope* diseñado para representar la señal de ECG.

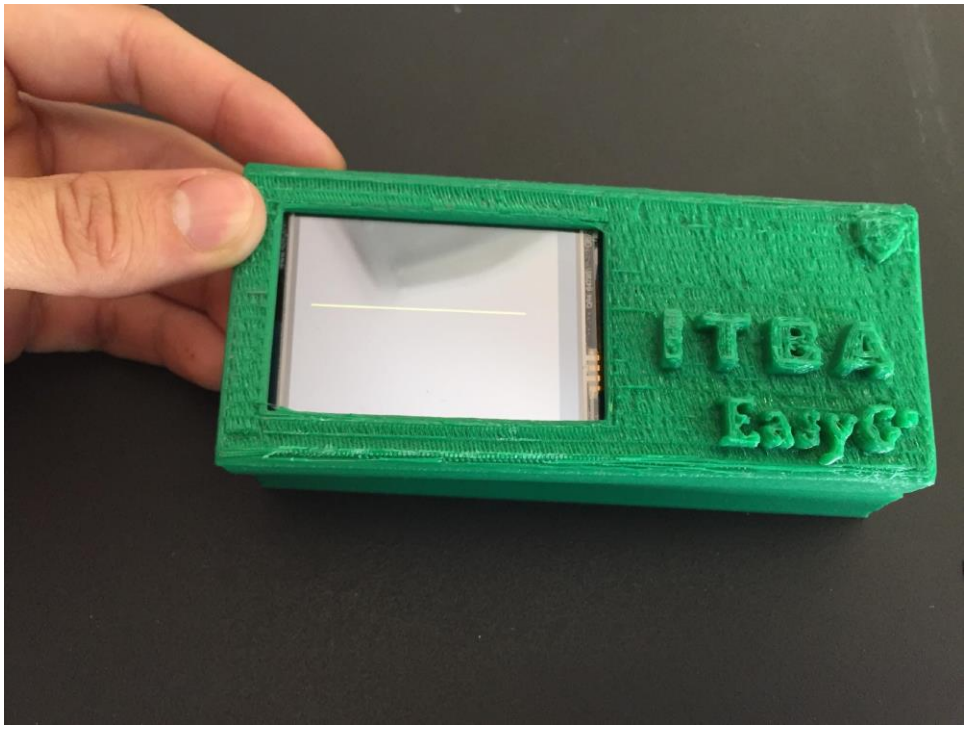


Figura 3.10.3: El monitor alimentado dentro del gabinete.

4. Conclusión

Durante el transcurso del trabajo se lograron construir todas las etapas individuales que conforman el monitor de manera exitosa. Se logró adquirir la señal de ECG a través del amplificador de instrumentación AD620 (figura 4.1), rechazando el ruido de modo común. Dicha señal fue adaptada mediante un filtrado digital pasa-altos de orden 1 para eliminación de continua y ruido de baja frecuencia, fue amplificada para lograr una amplitud pico a pico de alrededor de 3V, y luego acotada en frecuencia mediante el empleo de un filtro antialias pasa bajos analógico de orden 4 de aproximación elíptica (Cauer) e implementado con celdas biquad. Todos los circuitos implementados fueron simulados y probados en el software QUCS, ensamblados en *protoboard* y medidos en el laboratorio. Se implementó un circuito de alimentación con una fuente acorde a los requerimientos del monitor, para minimizar la inducción de ruido de línea y garantizar la portabilidad del prototipo. Se logró diseñar, imprimir y soldar un circuito sobre una placa PCB del tamaño requerido, que alojara todas las etapas analógicas del filtrado y de la adquisición de la señal. Dicho circuito se centró en garantizar un tamaño chico, por lo que se trabajó con componentes de montaje superficial (SMD). También se consiguió programar exitosamente la interfaz gráfica que permitiría mostrar en el *display* los datos procesados, así como también se permitió la transferencia de datos por comunicación serie (USART, a través de USB) hacia Matlab para el análisis de las señales. Se diseñaron en Matlab y se implementaron en la placa de Arduino los filtros digitales necesarios para filtrar el ruido de línea de 50 o 60Hz (figura 4.2 y 4.3) y el filtro decimador para mostrar una cantidad adecuada de puntos en la pantalla. Además, se diseñó e imprimió un gabinete con las medidas necesarias para alojar el monitor con sus componentes y batería (figura 3.10.3).

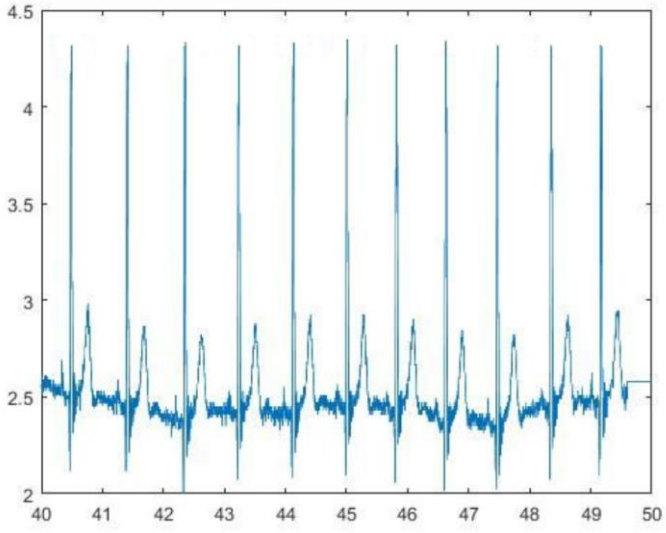


Figura 4.1: Señal cardíaca con ruido de línea adquirida por el Arduino y mostrada en Matlab.

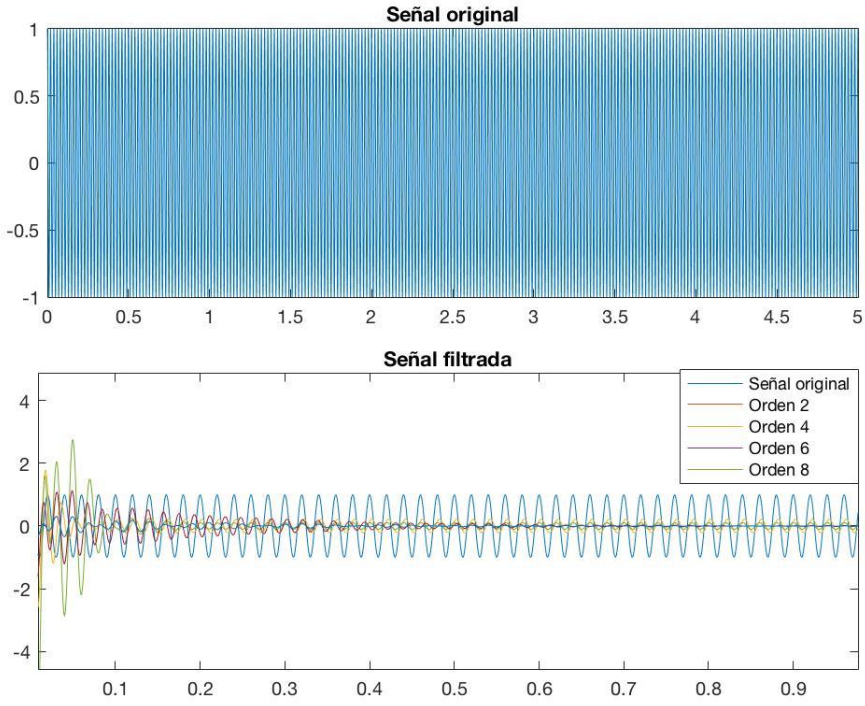


Figura 4.2: Filtro digital IIR de orden 8 implementado en etapas de orden 2 como "*Direct-Form II*" filtrando una señal cosenoidal de 50Hz.

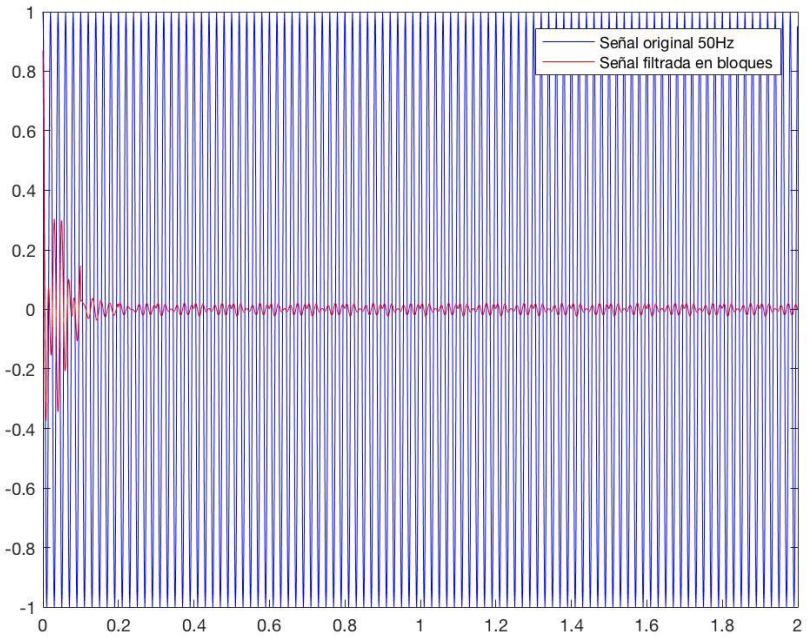


Figura 4.3: Filtro digital IIR de orden 8 implementado en etapas como "*Direct-Form II*" filtrando una señal cosenoidal de 50Hz cada 100 muestras.

Sin embargo, la integración de todas estas etapas fue menos que ideal. Se encontraron muchas dificultades técnicas a la hora de construir y soldar la placa impresa. La decisión de trabajar con integrados de montaje superficial complejizó la etapa de soldado de la placa así como también la disponibilidad de éstos en el mercado argentino. En la búsqueda de miniaturizar el modelo se obviaron medidas de separación físicas de etapas circuitales, como podría ser el uso de jumpers. Esto dificultó enormemente la detección de problemas en la placa, que para ser resueltos requerían desoldar componentes. En más de una ocasión se debió reemplazar el integrado quad-opamp, tarea que implicaba remover el estaño y retirar el componente, en un espacio muy reducido y sin la suficiente experiencia técnica para hacerlo sin levantar alguna pista de la placa. Esto desencadenó la necesidad de re-hacer la placa (en tres oportunidades), y el reemplazo de los componentes supuso además la espera de más de tres semanas por parte del importador.

Como aprendizaje del trabajo realizado, y observando las dificultades en las que se incurrió retrospectivamente, se destacan los siguientes posibles cambios que deberían realizarse para una mejor integración del dispositivo en el futuro:

- El uso de integrados *through-hole*: éstos tienen mejor disponibilidad en el mercado local, son más fáciles de reemplazar, pueden ser montados sobre zócalos para el reemplazo directo en caso de quemarse. Como desventaja son más grandes que los de montaje superficial, sin embargo, manteniendo los demás componentes (resistencias y capacitores) como SMD se logra una placa de tamaño acorde a las especificaciones planteadas.

- La separación de los circuitos de las distintas etapas. Si bien esto impide la optimización en la colocación de los integrados en la placa, resulta imprescindible a la hora de probar la placa, puesto que permite aislar fácilmente las fallas electrónicas.

- Si bien se decidió construir el prototipo con los materiales disponibles en el laboratorio de electrónica, se podría contemplar centrarse únicamente en las etapas de diseño y ruteo para luego tercerizar la construcción de la placa, lo que implicaría calidad comercial.

Como conclusión final, se logró armar y probar todas las etapas del dispositivo por separado, por lo que se concluye que es factible la construcción de éste. Sin embargo, se sugiere tomar medidas más conservadoras a la hora de su diseño. Se sugiere encontrar una relación de compromiso entre miniaturización y comodidad, para mejorar la integración de todas las etapas y poder obtener un prototipo portátil funcional.

5. Bibliografía

1. Gray Anatomía para Estudiantes, Vogl, A.Wayne, Drake, Richard L., Mitchell, Adam W.M. Barcelona, España: Elservier Inc. 2015. 3er edición
2. Ganong Fisiología Médica, Barret, Kim E., Barman, Susan M. México D.F.: McGrawHill Interamericana Editores,2010, 23avaedición
3. https://link.springer.com/article/10.1023/A:1023667812925
4. American heart association: recommendation for standarization of instruments in electrocardiography and vectorcardiography
5. Digital Sampling Rate and ECG Analysis , G.P. Pizzuti, S. Cifaldi and G. Noife*, Journal of Biomedical Engineering, volume 7, issue 3. Elservier Inc. 1987
6. ECG Noise Cancellation Using Digital Filters, Gholam-Hosseini, H., Nazeran, H., Reynolds, K.J. Proceedings of the 2nd International Conference on Bioelectromagnetism. Melbourne, Victoria, Australia, 1998
7. https://www.allaboutcircuits.com/technical-articles/an-introduction-to-filters/
8. https://www.who.int/es/news-room/fact-sheets/detail/cardiovascular-diseases-(cvds)
9. What is the adequate sampling interval of the ECG signal for heart rate variability analysis in the time domain?, Laszlo Hejjel & Elizabeth Roth. Physiological Measurement, Volume 25, Number 6, IOPscience, 2004.
10. Design of Analog Filters, Rolf Schaumann & Mac E. Van Valkenburg, The Oxford Series in Electrical and Computer Engineering, 2009, 2nd edition.
11. Basic Linear Design, Zumbahlen, Hank, Analog Devices, Inc. 2007.
12. Design and Analysis of Analog Filters: A Signal Processing Perspective, Paarmann, Larry D. Boston (USA),Dordrecht (Netherlands),London (UK). Kluwer Academic Publishers, 2003, 1st edition.
13. MATLAB and Statistics Toolbox Release 2016b, The MathWorks, Inc., Natick, Massachusetts, United States.
14. www.4dsystems.com.au
15. https://www.altium.com/altium-designer/
16. https://www.4dsystems.com.au/product/4D_Workshop_4_IDE/
17. https://4dsystems.com.au/4duino-24#software-tab
18. https://www.arduinolibraries.info/libraries/genie-arduino
19. www.tinkercad.com
20. Handbook of Biomedical Instrumentation, R.S. Khandpur, New Delhi, India: Tata McGraw-Hill Publishing Company Limited, 2012, 1st edition.
21. www.arduino.cc
22. www.mouserelectronics.com
23. https://www.cdc.gov/heartdisease/facts.htm
24. http://www.analog.com/media/en/training-seminars/design-handbooks/Basic-Linear-Design/Introduction.pdf
25. https://www.arduinolibraries.info/libraries/timer-three
26. Essentials of Digital Signal Processing, Lathi, B.P. and Roger A. Green. New York, NY: Cambridge University Press, 2014, 1st edition.

27. Modern Digital and Analog Communication Systems, Lathi, B.P. and Ding, Zhi. New York, NY: Oxford University Press, 2010, 4th edition.

6. Anexos

Anexo 1: Código Matlab Bode Cauer orden 4 con el Bode de sus celdas por separado

```
Fs = 1e3;
Ts = 1/Fs;
B = 10;          % bits de resolucion
Vref = 5;        % tension de referencia
Q = Vref/2^B;

dP = Q/Vref;     % delta P (ver Lathi)  dP = 1/2^B
Ap = 20*log10((1+dP/2)/(1-dP/2));
Aa = -20*log10(dP);

fp = 113;
fa = Fs/2;

% [N, Wn] = buttord(2*pi*fp, 2*pi*fa, Ap, Aa, 's');
% [N, Wn] = cheb1ord(2*pi*fp, 2*pi*fa, Ap, Aa, 's');

%% AproximaciÛn Cauer (mÌnimo orden: 4)
[N, Wn] = ellipord(2*pi*fp, 2*pi*fa, Ap, Aa, 's');
[z,p,k] = ellip(N,Ap,Aa,Wn,'s');
[b,a] = ellip(N,Ap,Aa,Wn,'s');
sys = zpk(z,p,k);

s = tf('s');
H1 =  ((s-z(1))*(s-z(2))) / ((s-p(1))*(s-p(2)));
H2 =  ((s-z(3))*(s-z(4))) / ((s-p(3))*(s-p(4)));

f = logspace(-2,4,1000);
[mag,fase] = bode(H1*H2*k,2*pi*f);
```

```
 figure(1);
 subplot(2,1,1);semilogx(f,20*log10(mag(:)));title('Modulo');
xlabel('Hz');ylabel('|H|');
 subplot(2,1,2);semilogx(f,fase(:));title('Fase');
xlabel('Hz');ylabel('Fase');
%estos graficos toman la info de arriba y me da los diagramas en
%frecuencia!


 %% celda 1


Wo = sqrt(real(p(1)*p(2)));
Wz = sqrt(real(z(1)*z(2)));
Q = Wo/1468; % /segundo termino del denom de la tf


[R,c1,c2,r1,r2,r3,cap] = celda(Wo,Wz,Q)


%% celda 2


Wop = sqrt(real(p(3)*p(4)));
Wzp = sqrt(real(z(3)*z(4)));
Qp = Wop/565.3; % /segundo termino del denom de la tf


[Rp,c1p,c2p,r1p,r2p,r3p,capp] = celda(Wop,Wzp,Qp)
```

Anexo 2: Código Matlab Bode Butterworth orden 7

```
%% Aproximacion Butterworth (minimo orden 7)
[Nb, Wnb] = buttord(2*pi*fp, 2*pi*fa, Ap, Aa, 's');
[z,p,k] = butter(Nb,Wnb,'s');
[b,a] = butter(Nb,Wnb,'s');
sys = zpk(z,p,k);


s = tf('s');
H1b =  (1) / ((s-p(1))*(s-p(2)));
H2b =  (1) / ((s-p(3))*(s-p(4)));
H3b =  (1) / ((s-p(5))*(s-p(6)));
H4b =  (1) / ((s-p(7)));


f = logspace(-2,4,1000);
[magb,faseb] = bode(H1b*H2b*H3b*H4b*k,2*pi*f);


 figure(1);
 subplot(2,1,1);semilogx(f,20*log10(magb(:)));title('Modulo');
xlabel('Hz');ylabel('|H|');
 subplot(2,1,2);semilogx(f,faseb(:));title('Fase');
xlabel('Hz');ylabel('Fase');
```

Codigo Matlab calculo de componentes para implementación de Pasa bajos Elíptico

```
function [R,c1,c2,r1,r2,r3,cap] = celda(Wo,Wz,Q)


    %% celda 1
s=tf('s');
a = 1; %Lo fijo yo, me hace mas facil el circuito


K = 3-(1/Q);
% K2 = 10^(-Aa/20); %NO LO USE en el caso de poner el h1 o h2
```

```
a0 = ((2+((Wz^2)/(Wo^2)))*(1/K));
h1 = 1/a0;

% %Pre distorsiono porque el opamp no es ideal
% error = K/(1500/Wo);
% Womod = Wo*(1+error);
% Qmod = Q*(1-error);
% Kmod= 3-(1/Qmod);

% C = 2*K2/(K-1);
C = 2*h1/(K-1);
Ck = C*(K-1)/K;
% b = 3*K2/K;
b = 3*h1/K;

N = -K * (Ck/2) * (s^2 + (Wo^2)*(2*((a-Ck)/Ck)));
D = s^2 + s*Wo/Q + Wo^2;


Hc = N/D;
f = logspace(-2,4,1000);
[mag,fase] = bode(Hc,2*pi*f);
figure;
subplot(2,1,1);semilogx(f,20*log10(mag(:)));title('Modulo');
subplot(2,1,2);semilogx(f,fase(:));title('Fase');

%% Componentes

cap = (1e-6); %0.01micro
R0 = 1000; %Elegidas segun conveniencia
```

```
R = 1/(Wo*cap);


c1 = b*cap;
c2 = (1-b)*cap;
r1 = R0/C;
r2 = R0/(1-C);
r3 = (K-1)*R0;


disp('fin')


end
```

<u>Anexo 3:</u> Código de Matlab de la función que devuelve las especificaciones del filtro Notch en 50Hz a utilizar

```
function Hd = FiltroElliptic50Hz
%FILTROELLIPTIC50HZ Returns a discrete-time filter object.

% MATLAB Code
% Generated by MATLAB(R) 9.0 and the DSP System Toolbox 9.2.
% Generated on: 19-Dec-2017 14:53:52

% Elliptic Bandstop filter designed using FDESIGN.BANDSTOP.

% All frequency values are in Hz.
Fs = 1000;  % Sampling Frequency

Fpass1 = 45;      % First Passband Frequency
Fstop1 = 49;      % First Stopband Frequency
Fstop2 = 51;      % Second Stopband Frequency
Fpass2 = 55;      % Second Passband Frequency
Apass1 = 0.5;     % First Passband Ripple (dB)
Astop  = 60;      % Stopband Attenuation (dB)
Apass2 = 1;       % Second Passband Ripple (dB)
match  = 'both';  % Band to match exactly

% Construct an FDESIGN object and call its ELLIP method.
h  = fdesign.bandstop(Fpass1, Fstop1, Fstop2, Fpass2, Apass1, Astop,
...
                      Apass2, Fs);
Hd = design(h, 'ellip', 'MatchExactly', match);

% [EOF]
```

<u>Anexo 4:</u> Código de Matlab de la función que devuelve las especificaciones del filtro Notch en 60Hz a utilizar

```
function Hd = FiltroElliptic60Hz
%FILTROELLIPTIC50HZ Returns a discrete-time filter object.

% MATLAB Code
% Generated by MATLAB(R) 9.0 and the DSP System Toolbox 9.2.
% Generated on: 19-Dec-2017 14:53:52

% Elliptic Bandstop filter designed using FDESIGN.BANDSTOP.

% All frequency values are in Hz.
Fs = 1000;  % Sampling Frequency

Fpass1 = 55;      % First Passband Frequency
Fstop1 = 59;      % First Stopband Frequency
Fstop2 = 61;      % Second Stopband Frequency
Fpass2 = 65;      % Second Passband Frequency
Apass1 = 0.5;     % First Passband Ripple (dB)
Astop  = 60;      % Stopband Attenuation (dB)
Apass2 = 1;       % Second Passband Ripple (dB)
match  = 'both';  % Band to match exactly

% Construct an FDESIGN object and call its ELLIP method.
h  = fdesign.bandstop(Fpass1, Fstop1, Fstop2, Fpass2, Apass1, Astop,
...
                      Apass2, Fs);
Hd = design(h, 'ellip', 'MatchExactly', match);

% [EOF]
```

Anexo 5: Código de Matlab para la implementación de la plantilla del filtro Notch de 50/60Hz

```
Hd=FiltroElliptic50Hz;

% Hd=FiltroElliptic60Hz;

h=Hd.sosMatrix;

% freqz(h(1,1:3),h(1,4:6)); %Me da la rta en frecuencia del numerador de
a fila 1 a la 3 por el numerador de la fila 3 a la 6 de la matriz que me
da el filtro

%Puedo usar zplane de lo mismo para ver los polos y ceros

% figure;

% zplane(h(1,1:3),h(1,4:6));

%mi numerador es de grado 8, para que me quede de este grado hago la conv
de todos los numeradores. 4 convoluciones solo para sacar el numerador y
4 para el denominador:

num0=conv(h(1,1:3),h(2,1:3));

num1=conv(h(3,1:3),h(4,1:3));

num=conv(num0,num1);


den0=conv(h(1,4:6),h(2,4:6));

den1=conv(h(3,4:6),h(4,4:6));

den=conv(den0,den1);


figure;

zplane(num,den);

figure;

freqz(num,den);
```

<u>Anexo 6:</u> Código de Matlab para el diseño del Decimador de Chebyshev tipo 2.

```
function Hd = DECIMADORChebyshev
%DECIMADORCHEBYSHEV Returns a discrete-time filter object.

% MATLAB Code
% Generated by MATLAB(R) 9.0 and the Signal Processing Toolbox 7.2.
% Generated on: 09-Mar-2018 13:05:18

% Chebyshev Type II Lowpass filter designed using FDESIGN.LOWPASS.

% All frequency values are in Hz.
Fs = 1000;  % Sampling Frequency

Fpass = 60;          % Passband Frequency
Fstop = 80;          % Stopband Frequency
Apass = 1;           % Passband Ripple (dB)
Astop = 40;          % Stopband Attenuation (dB)
match = 'stopband';  % Band to match exactly

% Construct an FDESIGN object and call its CHEBY2 method.
h  = fdesign.lowpass(Fpass, Fstop, Apass, Astop, Fs);
Hd = design(h, 'cheby2', 'MatchExactly', match);

% [EOF]
```

<u>Anexo 7:</u> Código de Arduino para la implementación del filtro Notch y el decimador.

```
#include <TimerThree.h>

// defines for setting and clearing register bits
#ifndef cbi
#define cbi(sfr, bit) (_SFR_BYTE(sfr) &= ~_BV(bit))
#endif
#ifndef sbi
#define sbi(sfr, bit) (_SFR_BYTE(sfr) |= _BV(bit))
#endif

// Analog pin that the ECG singal is connected to
const int analogInPin = A0;

int k = 0;

// Originally collected signal (x, x1 and x2), notch filtered (y, y1, y2 and y3)
// x = x(t); x1 = x(t-1); x1 = x(t-2). The same applies for y

volatile double x = 0.0;
volatile double x1 = 0.0;
volatile double x2 = 0.0;

volatile double y0 = 0.0; // 1st Notch stage output
volatile double y00 = 0.0;  // 2nd Notch stage output
volatile double y000 = 0.0;  // 3rd Notch stage output
volatile double y0000 = 0.0;  // 4th Notch stage output
```

```
volatile double y1 = 0.0; // y(t-1) notch output

volatile double y2 = 0.0; // y(t-2) notch output


double timeVecto2 [1024]; // Array which keeps 1024 samples from the ECG signal (more or less, one cycle: sampling frequency = 1000 Hz)


void setup() {


  // Se aumenta la velocidad del clock de conversión de datos

  // Set prescale to 16

  sbi(ADCSRA,ADPS2) ;

  cbi(ADCSRA,ADPS1) ;

  cbi(ADCSRA,ADPS0) ;


  typedef unsigned int uint;

double t = 0.0;

const uint N = 100;

const uint L = 8;

double buffer1[N] = {0.0};

double x[N+L] = {0.0};

double y[N+L] = {0.0};

bool BufferLleno = false;


void rts();

void SendData();

   Timer3.initialize(1000); //Inicializo timer3 para que corte cada 0.001s = 1000Hz de Fs

   Timer3.attachInterrupt(rts); //readAnalog will run every 0.001s

   Serial.begin(1000000);

}
```

```
void loop() {
  if (BufferLleno)
 {
   FilterData();
   SendData();
   BufferLleno = false;
 }
}

// Mando los datos
void SendData(){
 for (uint n=0; n<N; n++)
 {
   Serial.print(x[n+L]);
   Serial.print('\t');
   Serial.println(y[n+L]);
 }
}

void FilterData(){  // commands executed at each time interrupt
  for (int n=0; n<L; n++)
   y[n] = y[n+N];

 for (uint n=L; n<N+L; n++)
 {
   y[n] = x[n]*b[0] + x[n-1]*b[1] + x[n-2]*b[2];
   y[n] -= y[n-1]*a[1] + y[n-2]*a[2];
```

```
 }
y[n] *= G;
}
}
 // Filtering around 50Hz with order 8 IIR Notch Filter implemented in 4 stages
 //Stage 1
  y0 = 1 * x - 1.9002 * x1 + 1 * x2 + 1.8067 * y1 - 0.9242 * y2;
 y2 = y1;
 y1 = y0;

 //Stage 2
  y00 = 1 * x - 1.9059 * x1 + 1 * x2 + 1.8644 * y1 - 0.9390 * y2;
 y2 = y1;
 y1 = y00;

 //Stage 3
  y000 = 1 * x - 1.8961 * x1 + 1 * x2 + 1.8729 * y1 - 0.9896 * y2;
 y2 = y1;
 y1 = y000;

 //Stage 4
  y0000 = 1 * x - 1.9095 * x1 + 1 * x2 + 1.9116 * y1 - 0.9913 * y2;
 y2 = y1;
 y1 = y0000;

 // Saving x(t) as x(t-1) and x(t-1) as x(t-2) values of x
 x2 = x1;
 x1 = x;
```

```
}

void rts()
{
  static uint pos = 0;
    // Para probar!
 // t += 0.002; // 2ms
 // buffer1[pos++] = cos(2.0*PI*t*50);

  buffer1[pos++] = analogRead(analogInPin);

  if (pos == N)
  {
    // guardamos las entradas anteriores
    for (uint n=0; n<L; n++)
      x[n] = x[n+N];
        // guardamos las nuevas entradas
    for (uint n=0; n<N; n++)
      x[n+L] = buffer1[n];

    BufferLleno = true;
    pos = 0;
  }
}
```

<u>Anexo 8:</u> Código de Arduino para implementar el filtro decimador IIR de orden 8.

```
// Low pass filtering to eliminate aliasing and filter necessary pixels with order 8 IIR Notch Filter implemented in 4 stages

 //Stage 1

y0 = 1 * x - 1.7435 * x1 + 1 * x2 - 1.7470 * y1 + 0.9112 * y2;

 y2 = y1;

  y1 = y0;

  //Stage 2

y00 = 1 * x - 1.6518 * x1 + 1 * x2 - 1.5434 * y1 + 0.0.9112 * y2;

  y2 = y1;

  y1 = y00;

//Stage 3

  y000 = 1 * x - 1.2960 * x1 + 1 * x2 - 1.2646 * y1 + 0.4871 * y2;

  y2 = y1;

  y1 = y000;

//Stage 4

  y0000 = 1 * x + 0.5359 * x1 + 1 * x2 - 0.9881 * y1 + 0.2602 * y2;

  y2 = y1;

  y1 = y0000;

  // Saving x(t) as x(t-1) and x(t-1) as x(t-2) values of x

  x2 = x1;

  x1 = x;

 // Saving z value in a vector

  timeVecto2[k] = z;

  k = (k + 1) % 1024;
```

<u>Anexo 8:</u> Código de Arduino para mostrar en pantalla los datos obtenidos.

```
#include <genieArduino.h>

#define PICASSO_RESET_LINE     30 //defino desde donde se resetea el display

#define GENIE_OBJ_SCOPE        25 //ID del objeto scope

Genie genie;


void setup() {

  Serial.begin(200000); //Inicializo los puertos que quiero con su respectivo baud rate, fijarme que sea el mismo que en visi-genie

  Serial1.begin(200000);

  genie.Begin(Serial1); // Me dice que desde este puerto le voy a hablar a la pantalla

  //genie.AttachEventHandler(myGenieEventHandler); //name of the function used in the user's code space, so it knows what to call when an event is received and it needs processing

  pinMode(PICASSO_RESET_LINE, OUTPUT); digitalWrite(PICASSO_RESET_LINE, 1); delay(100); digitalWrite(PICASSO_RESET_LINE, 0); // Reseteo y espero a q se configure la pantalla

  delay(3000);

  genie.WriteContrast(15); //Display con maximo brillo(15) , si quiero apagarlo hago genie.WriteContrast(0);

}


void loop()

{   genie.WriteObject(GENIE_OBJ_SCOPE , 0 , 124); }
```

<u>Anexo 9:</u> Código de Arduino para la implementación del filtro.

```
#include <TimerThree.h>

#include <genieArduino.h>

void rts();

void sendData();


void setup() {

Serial.begin(1000000);

Timer3.initialize(1000);

Timer3.attachInterrupt(rts);

  Serial.begin(1000000);

  Serial1.begin(1000000);

  genie.Begin(Serial1);

  pinMode(PICASSO_RESET_LINE, OUTPUT); digitalWrite(PICASSO_RESET_LINE, 1); delay(100); digitalWrite(PICASSO_RESET_LINE, 0); // Reseteo y espero a q se configure la pantalla

  delay(3000);

  genie.WriteContrast(15); //Display con maximo brillo(15) , si quiero apagarlo hago genie.WriteContrast(0);

}
```

<u>Anexo 10:</u> Código de Matlab que utiliza los coeficientes del filtro digital IIR diseñado y la función Filter() para implementarlo y devolver sus respuestas en frecuencia.

```
close all;

t = 0:1e-3:5;
x = cos(2*50*pi*t); % cos(2*pi*t);

figure(1);
subplot(2,1,1);plot(t,x);title('Señal original')


%% using script
xx = x;
Zf = [0 0];

for etapa = 1:4

  b = mat(etapa,1:3);
  a = mat(etapa,4:6);
  G = g(etapa);

  [y,zf] = filter(b,a,x,Zf);
  y = G*y;

  subplot(2,1,2);title('Señal filtrada');
  plot(t,x);hold all;plot(t,y);
  legend([{'Señal original','Orden 2','Orden 4','Orden 6','Orden
8'}]);

  x = y;

%   for i=3:length(x)
%     y(i) = (x(i) + mat(etapa,2) * x(i-1) + x(i-2) - mat(etapa,5) *
y(i-1) - mat(etapa,6) * y(i-2))*g(etapa);
%   end

end

figure(2);
plot(t,xx,t,y);title('Input vs. Output');
legend([{'Señal original','Señal filtrada'}]);
```

<u>Anexo 11:</u> Código de Matlab que utiliza los coeficientes del filtro digital IIR diseñado con la función Filter() o con la función implementada por fórmula para visualizar la respuesta en frecuencia de cada del filtro según se acrecienta el orden del mismo.

```
close all;

t = 0:1e-3:5;
xo = cos(2*50*pi*t); % + cos(2*pi*t);

figure(1);
subplot(3,1,1);plot(t,xo);title('Señal original');



%% using script
xx = xo;
Zf = [0 0];

% agregamos dos datos basura
y = zeros(1,length(xo)+2);
x = [0 0 xo];
subplot(3,1,2);plot(t,xo);
subplot(3,1,3);plot(t,xo);

for etapa = 1:4

  %% using filter
  b = mat(etapa,1:3);
  a = mat(etapa,4:6);
  G = g(etapa);

  [yy,zf] = filter(b,a,xx,Zf);
  yy = G*yy;

  subplot(3,1,2);
  hold all;plot(t,yy);title('Filtro Matlab') %%se usa la funcion
filter con los coef. de mi matriz del IRR diseñado
  legend([{'Orden 2','Orden 4','Orden 6','Orden 8'}]);

  xx = yy;

  %% using script

  for i=3:length(x)
%     disp(i)
    y(i) = x(i) + b(2) * x(i-1) + x(i-2) - a(2) * y(i-1) - a(3) *
y(i-2);
  end

  y = y*g(etapa);

  subplot(3,1,3);hold all;plot(t,y(3:end));title('Con formula');
  legend([{'Orden 2','Orden 4','Orden 6','Orden 8'}]);
```

```
  x = y;

end

% eliminamos los dos datos basura
y = y(3:end);

figure(2);
plot(t,xo);
hold all;
plot(t,yy);
plot(t,y);title('Señal original y filtrada por distintos métodos');
legend([{'Señal original','Filtro Matlab','Con formula'}]);
```

<u>Anexo 12:</u> Código de Matlab que utiliza los coeficientes del filtro digital IIR diseñado con la función Filter() o con la función implementada por fórmula para visualizar la respuesta en frecuencia del filtro según la cantidad de muestras que se toman.

```
%En un script aparte colocal la función "filtrar"
function [y] = filtrar(mat,g,xo,xi,yi)

  L = length(xi);

  % agregamos dos datos basura
  y = [yi zeros(1,length(xo))];
  x = [xi xo];

  for etapa = 1:4

    b = mat(etapa,1:3);
    a = mat(etapa,4:6);
    G = g(etapa);

    for i=3:length(x)
      y(i) = x(i) + b(2) * x(i-1) + x(i-2) - a(2) * y(i-1) - a(3) *
y(i-2);
    end

    y = y*G;
    x = y;

  end

  % eliminamos los dos datos basura
  y = y(L+1:end);

end
```

%Llamar a la función en otro script

```
close all;

to = 0:1e-3:5;
xo = cos(2*50*pi*to) + cos(2*pi*to);

xf=[];
tf=[];
yf = [];

figure(1);
N = 100; %Muestras tomadas
L = 8;%Cantidad de datos que necesito del vector anterior
xi = zeros(1,L);
yi = zeros(1,L);
z = [];
for bloque=1:20 %tiempo de la simulacion
```

```
  x = xo((bloque-1)*N+1:bloque*N);
  t = to((bloque-1)*N+1:bloque*N);


  y = filtrar(mat,g,x,xi,yi);
%   y = filter(Hd,x);


  yi = y(end-L-1:end);
  xi = x(end-L-1:end);

  tf = [tf t];
  xf = [xf x];
  yf = [yf y];
  plot(tf,xf,'b');hold on;
  plot(tf,yf,'r');legend([{'Señal original 50Hz y 1Hz','Señal
filtrada en bloques'}]);
  pause(0.1);
end
```