UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE INGENIERÍA

Sistema de Biorretroalimentación para Terapias de Rehabilitación Muscular

TESIS

Que para obtener el título de

Ingeniero en Computación

P R E S E N T A

Ortíz Díaz Abraham

DIRECTOR DE TESIS

M. I. Rafael Prieto Meléndez

Ciudad Universitaria, Cd. Mx., 2017

A mis padres los cuales me apoyaron en cada una de las decisiones que he tomado a lo largo de mi vida.

A mis hermanos, para los cuales me gustaría ser el mejor ejemplo en su vida.

A mi esposa e hijo los cuales desde el día que llegaron a mi vida me han iluminado a lo largo de este camino.

Al M.I. Rafael Prieto Meléndez por ser mi asesor y darme la oportunidad de trabajar con él en este proyecto.

Al M.I. Sergio Quintana Thierry por su revisión, asesoría y comentarios a esta tesis.

A la Dra. Karina Mendoza Ángeles por su revisión y comentarios a esta tesis.

Al Ing. Jaime Héctor Díaz Osornio por su revisión y comentarios a esta tesis.

Al Dr. Jesús Manuel Álvarez López por su revisión y comentarios a esta tesis.

Contenido

Listado de Imágenes

Listado de tablas

Introducción

En la actualidad existen diversos sistemas para auxiliar en la rehabilitación de pacientes que sufren de alguna atrofia o parálisis muscular, y dependiendo de la técnica de rehabilitación utilizada, estos pueden ser de costos elevados, así como de dimensiones muy grandes.

Sin embargo, existen pocos hospitales que ofrezcan terapias de rehabilitación mediante la técnica de biorretroalimentación (*biofeedback en inglés*), y la mayoría de ellos se encuentran en la Ciudad de México. Esto dificulta a pacientes que proceden de regiones fuera de la ciudad el poder llevar de manera continua sus terapias, ya que no siempre se encuentran en condiciones de dejar su hogar de procedencia y poder viajar para llevar a cabo sus sesiones de terapia. Esto provoca llevar un proceso de rehabilitación más lento y en ocasiones la suspensión del mismo debido a la distancia entre el hospital y el lugar de residencia del paciente, los gastos que requieren los traslados, el hospedaje y la comida.

A lo largo de esta tesis veremos paso a paso el diseño, desarrollo, construcción y evaluación de un prototipo de un sistema de biorretroalimentación, el cual nos proporcionará información para un análisis de los alcances que puede llegar a tener, si se comporta como lo deseamos, si es una propuesta factible y viable para nuestros objetivos.

Con el desarrollo de este sistema se pretende contar con una herramienta la cual pueda ser un sistema compacto, portátil, económico y de manejo muy sencillo para que doctores y pacientes puedan usarlo sin problemas y con ello evitar la interrupción de la rehabilitación en pacientes, haciendo posible la idea de poder llevar las terapias de rehabilitación hasta sus hogares.

Es importante mencionar que el sistema se usará para fines de investigación, por lo que el sistema puede mejorar en cada una de sus etapas, ya que es posible agregar o quitar componentes.

Objetivos

- Crear un prototipo de un sistema basado en las técnicas de biorretroalimentación para auxiliar en la rehabilitación de pacientes que sufren algún tipo de atrofia o parálisis muscular debido a alguna enfermedad, accidente o cirugía.
- Adquirir señales mioeléctricas mediante el uso de muy pocos componentes eléctricos, con una sencillez y facilidad en el ensamblaje; con el fin de crear un prototipo lo más económico y accesible posible, sin perder calidad en la obtención y tratamientos de señales, perdida de la información y, sobre todo; que los datos adquiridos sean de alta confiablidad y precisión.
- Utilizar herramientas que faciliten la implementación del prototipo tales como microcontroladores, tarjetas de desarrollo, computadoras, etc.
- Uso de software libre, para poder disminuir costos de implementación.
- Desarrollar un software dedicado a la aplicación, sin la dependencia del poder de cómputo y sobre todo con la posibilidad de crecimiento y mejora.
- Tener un prototipo compacto, seguro y fácil de transportar; el cual su construcción se base mediante secciones o bloques, con la finalidad de poder mejorar individual o grupalmente cada una de las secciones sin necesidad de interferir en cualquier otro proceso.

Marco teórico

Señales biológicas a través de un proceso de biorretroalimentación

Mayormente las aplicaciones de la biorretroalimentación están provistas de herramientas para la detección y medición de señales físico-psicológicas.

Pero, ¿Qué se entiende por biorretroalimentación?

"La biorretroalimentación, es un procedimiento que nos permite aprender a controlar las funciones fisiológicas de uno o varios organismos medidos y monitoreados a través de dispositivos".

Un instrumento de biorretroalimentación debe desempeñar tres tareas esenciales:

1. **Monitorear** (de alguna manera) un proceso fisiológico de interés.
2. **Medir** (de una forma objetiva) lo que se está monitoreando.
3. **Presentar o Representar** de alguna forma lo que se está monitoreando y/o midiendo como información significativa.

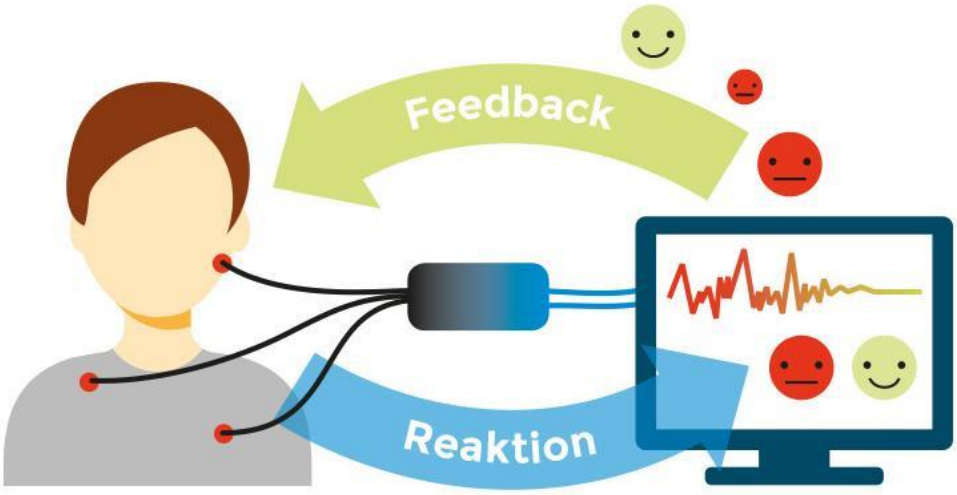


Imagen 1. Biorretroalimentación

Fuente: http://www.neurofeedback-fuchs.de/therapie/index.php

Sin embargo, la acción de medir una señal mioeléctrica producida por una contracción muscular no es para un dispositivo de biorretroalimentación algo simple y directo.

Pero, ¿cómo se producen las señales mioeléctricas y cómo es que las podemos captar?

Cada función que realiza el cuerpo humano es iniciada por impulsos eléctricos, estos impulsos eléctricos son generados por procesos electroquímicos de células específicas en una región del cuerpo, mismas que, generan una diferencia de potencial mediante el flujo de iones.

La mayoría de estos potenciales se originan y se llevan a cabo en el cerebro, el cual envía órdenes en forma de potenciales de acción a otras partes del cuerpo, estas señales eléctricas son transportadas a través del sistema nervioso.

El sistema nervioso es toda una red compleja de órganos especializados que incluye: el encéfalo, la medula espinal y diversas terminaciones nerviosas alrededor del organismo. El cual tiene como función, controlar y regular el funcionamiento de los sistemas restantes del cuerpo humano, así como coordinar la relación del organismo con el medio exterior.

Este se divide en: sistema nervioso central y sistema nervioso periférico.

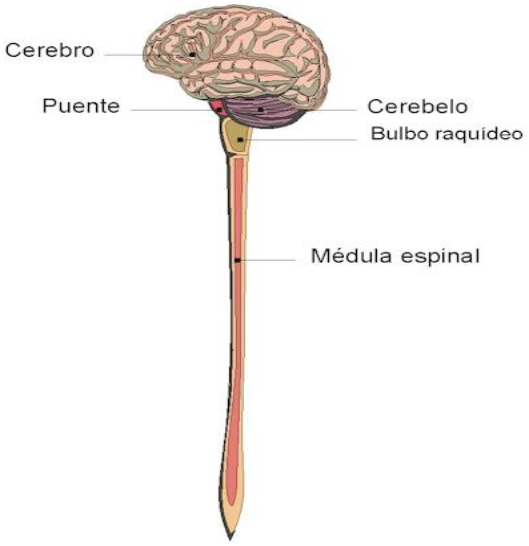


Imagen 2. Sistema Nervioso Central

Fuente: https://quintogradomav.wordpress.com/sistema-nervioso/

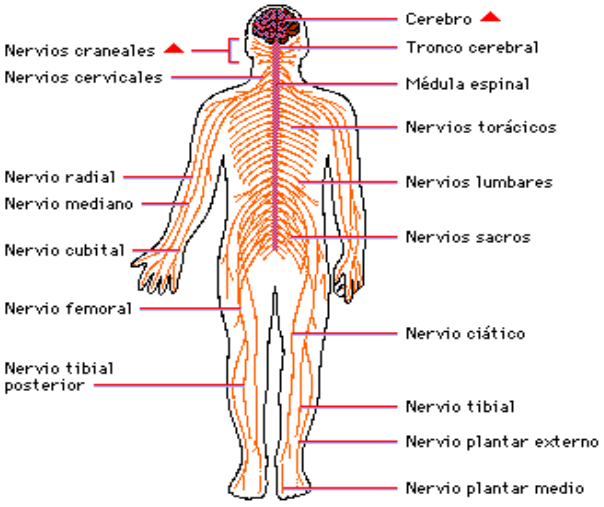


Imagen 3. Sistema Nervioso Periférico

Fuente: http://aduperierconotercerciclo.blogspot.mx/2015/02/el-sistema-nervioso.html

El *sistema nervioso central* está compuesto por el encéfalo y la medula espinal. Por otro lado, el *sistema nervioso periférico* consta de todos los tejidos nerviosos que no están dentro del sistema nervioso central.

El sistema nervioso central es también donde se originan nuestros pensamientos, recuerdos y emociones. Cuando esta información se integra, se genera una respuesta, la cual viaja a través de nervios que son parte del sistema nervioso periférico y son traducidas como funciones motoras.

Las *neuronas* son las unidades funcionales del sistema nervioso, ya que son los elementos excitables que generan los *potenciales de acción*.

Cabe mencionar que estas células representan la unidad básica funcional y estructural del sistema nervioso.

Una neurona está compuesta por:

- Un cuerpo o soma neuronal
- El axón
- Las dendritas

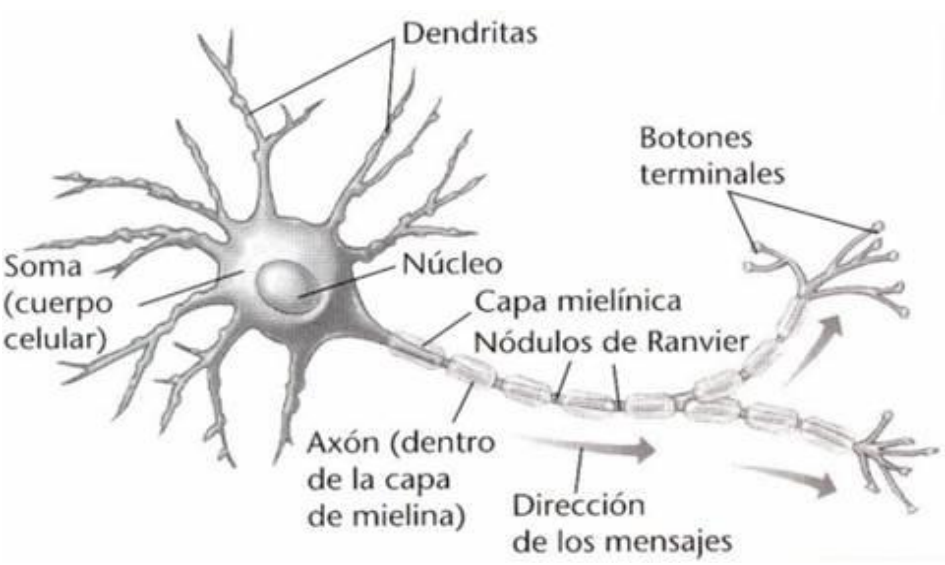


Imagen 4. Composición de una neurona

Fuente: http://www.monografias.com/trabajos75/sistema-nervioso/sistema-nervioso.shtml

EL *cuerpo* de una neurona contiene al núcleo y citoplasma el cual se encuentra rodeado por una membrana plasmática.

Las *dendritas* son ramificaciones por las cuales una neurona recibe estímulos procedentes de neuronas vecinas, así es como se establece una comunicación entre neuronas. Este proceso se conoce como sinapsis.

Finalmente, el *axón* es una extensión única de longitud variable y es el medio por el cual los impulsos eléctricos son transmitidos desde el cuerpo celular a otras células o a otros órganos del cuerpo.

Una *señal mioeléctrica* es un *potencial de acción,* el cual viaja a través de neuronas dedicadas a la comunicación entre el sistema nervioso y el sistema muscular, estas neuronas son denominadas motoneuronas o neuronas motoras.

Un potencial de acción es el cambio de estado de una neurona, el cual pasa de un estado de reposo a un estado activo o en acción. El objetivo de un potencial de acción es la comunicación entre neuronas, las cuales se rigen por el principio de todo o nada. Si una neurona no alcanza el umbral de activación, esta se encontrará en reposo siempre y cuando el potencial de acción no llegue a superar dicho umbral de activación.

Las fases de un potencial de acción son:

- Fase de reposo
- Fase de despolarización
- Fase de repolarización

En la *fase de reposo* la célula se encuentra en la posibilidad de responder a un estímulo, el cual debe de tener la intensidad suficiente para superar el umbral de acción, una vez superado este umbral comienza a haber dentro de la célula actividad electroquímica la cual *despolariza* la membrana celular llegando a un pico máximo. Una vez alcanzado el pico máximo la membrana se *repolariza* disminuyendo la actividad eléctrica, hasta que se llega a un estado de reposos.

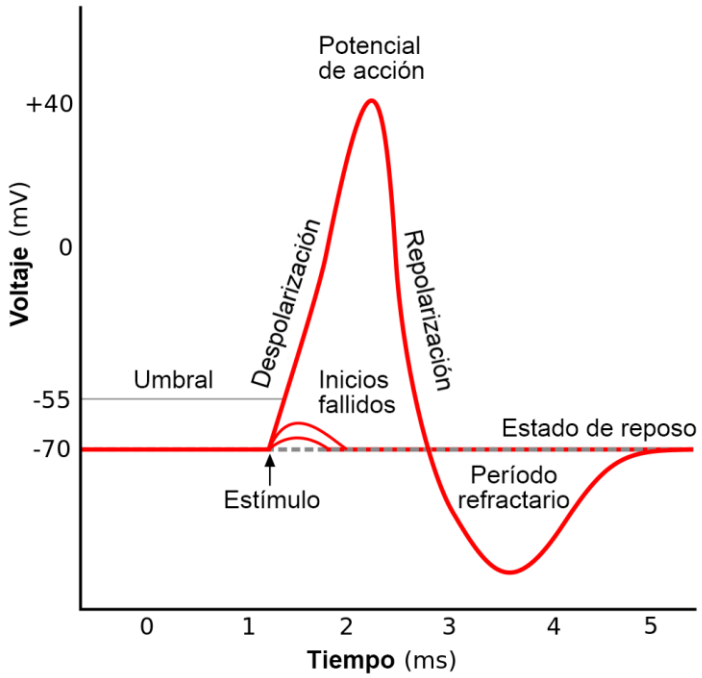


Imagen 5. Potencial de acción

Fuente:https://curiosoando.com/cual-es-la-diferencia-entre-potencial-de-accion-y-potencial-graduado

Como sabemos, el músculo esquelético es aquel que junto con los tendones mantienen unido al esqueleto humano. Este constituye gran parte de la masa muscular, la cual nos permite llevar a cabo diversos movimientos como la contracción y relajación.

La acción de contraer y relajar un músculo se lleva a cabo mediante un ciclo, el cual requiere energía en forma de ATP (trifosfato de adenosina) para poder realizar sus funciones, las cuales son:

- Adhesión
- Desplazamiento
- Separación

Este ciclo inicia cuando los iones de calcio (Ca^{2+}) se unen a la troponina C, los cuales son liberados por el retículo sarcoplasmico. Al unirse a la troponina C, esta cambia de forma y provoca el desplazamiento de la tropomiosina, este desplazamiento deja libre al sitio activo de la actina para que la cabeza de miosina se pueda unir.

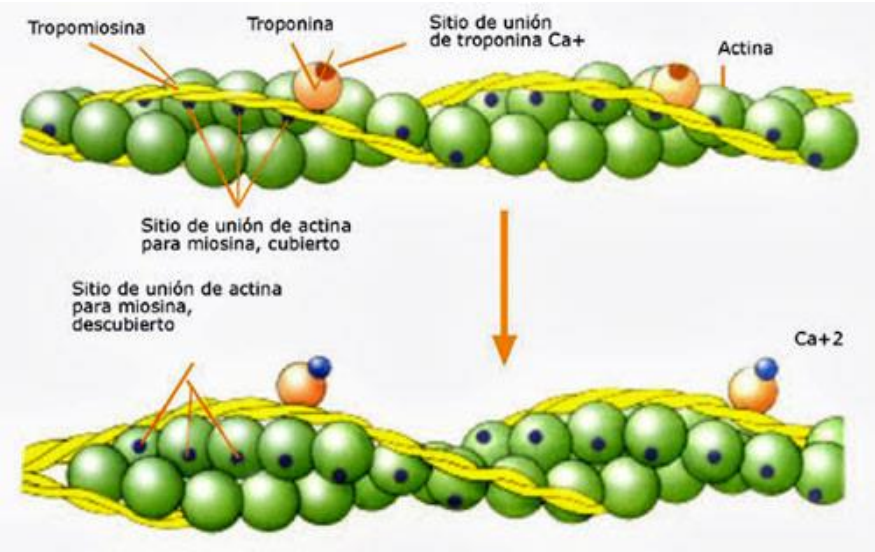


Imagen 6. Inicio del ciclo del ATP

Fuente: https://www.asturnatura.com/articulos/citosol-citoesqueleto/filamentos-de-actina.php

La *adhesión* inicia cuando se une el ATP (trifosfato de adenosina) a las cabezas de miosina y se produce la hidrólisis del ATP (trifosfato de adenosina) descomponiéndolo en ADP (adenosín difosfato) + P (fosfato) inorgánico.

La energía liberada de la hidrólisis del ATP (trifosfato de adenosina), activa la cabeza de miosina la cual se une a la actina. Ya unida se libera P (fosfato) inorgánico, y esto produce que la unión de la cabeza de miosina con la actina se fortalezca.

El ADP (adenosín difosfato) es liberado por la cabeza de miosina y aquí es cuando se desplaza. El *desplazamiento* consta de una rotación producida por ángulos diferentes entre la cabeza de miosina y los filamentos delgados de la actina. Este movimiento es ejercido en sentido contrario en los dos extremos del filamento de miosina.

Por último, se une otra molécula de ATP (trifosfato de adenosina) a la cabeza de miosina. La unión entre la cabeza de miosina produce que la actina se debilite, esto provoca que se *separe* la cabeza de miosina, sin embargo, aquí es cuando vuelve a iniciar el ciclo, ya que ese ATP (trifosfato de adenosina) activa la cabeza de miosina de nuevo.

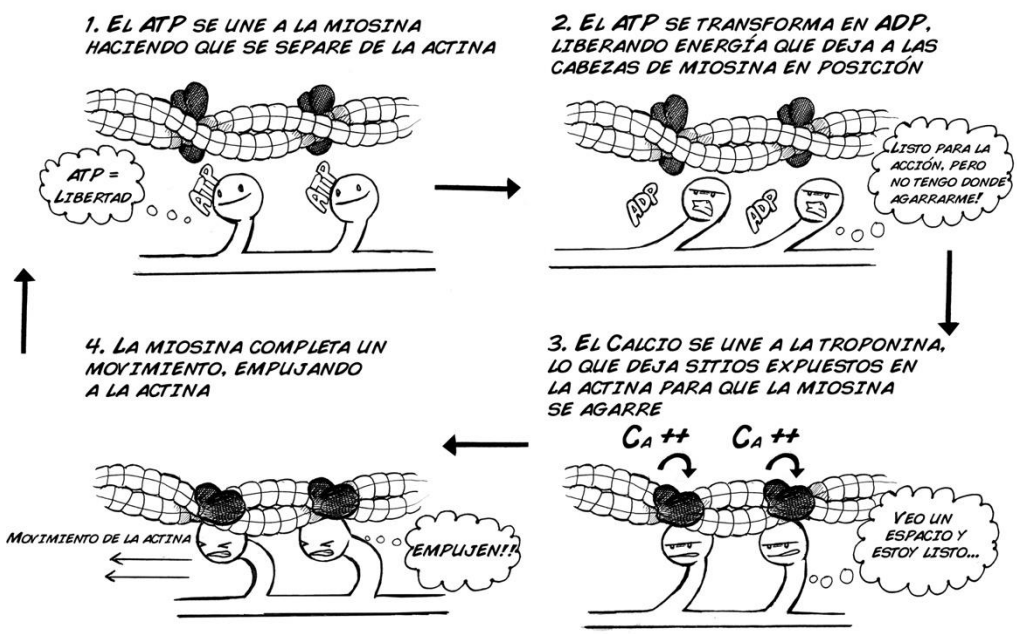


Imagen 7. Ciclo completo del ATP

Fuente: http://www.backyardbrains.cl/experiments/fatigue

Medición de señales mioeléctricas

El proceso descrito anteriormente es el causante de generar las señales mioeléctricas, esto debido a que en él se involucran reacciones electroquímicas las cuales pueden ser captadas mediante electrodos conectados a instrumentos de medición.

Un electrodo se define como un transductor que convierte potenciales iónicos generados en el interior del cuerpo por corrientes iónicas, en potenciales eléctricos.

La medición de esta señal se puede llevar a cabo de dos formas:

- Medida invasiva
- Medida no invasiva

La forma *invasiva* es mediante electrodos de aguja. Este proceso llega a ser doloroso ya que se debe insertar una pequeña aguja directamente en el nervio o músculo de interés. Por otro lado, la medida de las señales de forma *no invasiva* se lleva a cabo mediante la colocación de electrodos de superficie, este proceso no produce dolor alguno y es usado con mayor frecuencia, siempre y cuando el tipo de medición lo permita.

En el momento que tenemos una señal eléctrica, la podemos tratar de diferentes formas, se puede amplificar, reducir, filtrar, rectificar, etc. Todos estos tratamientos a la señal los podemos llevar a cabo mediante el uso de amplificadores operacionales.

Un amplificador operacional es un circuito analógico que sirve para amplificar señales eléctricas, consta de:

- Una entrada inversora
- Una entrada no inversora
- Una salida

Idealmente un amplificador operacional se rige por las siguientes características:

- ❖ La impedancia de entrada es infinita
- ❖ La impedancia de salida es cero
- ❖ La ganancia en lazo abierto es infinita
- ❖ Su ancho de banda es infinito
- ❖ La ganancia en modo común es cero

Sin embargo, realmente un amplificador operacional tiene las siguientes características:

- ❖ La impedancia de entrada es mayor a 1 MΩ
- ❖ La impedancia de salida es menor a 100 Ω
- ❖ La ganancia en lazo abierto es alrededor de 100 dB
- ❖ Su ancho de banda es de 1 MHz

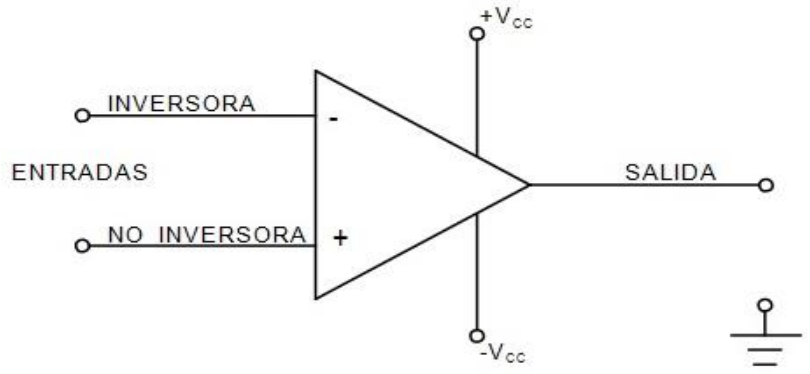


Imagen 8. Amplificador Operacional

Fuente: http://www.areatecnologia.com/amplificadores-operacionales/amplificador-operacional-introduccion.htm

Los amplificadores operacionales tienen diferentes arreglos los cuales se usan en diferentes aplicaciones, una de ellas son los *amplificadores de instrumentación*.

Un amplificador de instrumentación se emplea en el sensado de señales muy pequeñas que no proporcionan suficiente corriente. Esto se logra gracias a que las señales entran directamente por las terminales de alta impedancia de los amplificadores operacionales. Además, su ganancia en modo común es muy baja respecto a su ganancia diferencial, esto nos ayuda a tener un rechazo en modo común alto y una disminución del ruido.

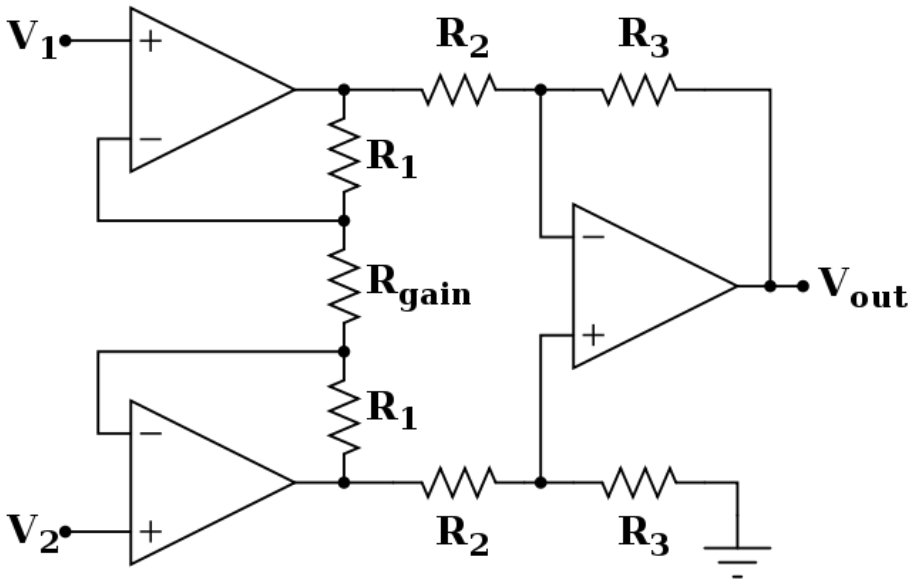


Imagen 9. Amplificador de instrumentación

Fuente: https://es.wikipedia.org/wiki/Amplificador_de_instrumentaci%C3%B3n

Sin embargo, no sólo basta con obtener la señal y amplificarla, recordemos que las señales mioeléctricas tienen valores de amplitud y frecuencia muy bajos y estas se encuentran inmersas en señales que no son de nuestro interés y que no necesariamente sea ruido, sino que sean otro tipo de señales bioeléctricas, por lo que debemos delimitar la frecuencia de operación de nuestra señal.

A continuación presento un cuadro con algunos valores de amplitud y frecuencia de diferentes señales bioeléctricas que pueden estar presentes en el cuerpo humano.

SEÑAL	MAGNITUD	ANCHO DE BANDA (Hz)
ECG (electrocardiograma)	0'5 - 4 mV	0'01 - 250
EEG (electroencefalograma)	5 - 300 μV	DC - 150
EGG (electrogastrograma)	10 - 1000 μV	DC - 1
EMG (electromiograma)	0.1 - 5 mV	DC - 10.000
EOG (electrooculograma)	50 - 3500 μV	DC - 50
ERG (electroretinograma)	0 - 900 μV	DC - 50

Tabla 1. Magnitud y ancho de banda de señales bioeléctricas

Como podemos observar, las señales mioeléctricas van de un rango de voltaje de 0.1 a 5 mV y con una frecuencia de hasta 10,000 Hz.

No solo hay que tomar en cuenta los rangos de voltaje y frecuencia en los que opera nuestra señal de interés, sino también se debe de considerar la existencia de ruido. Este ruido puede provenir de cualquier parte del medio que nos rodea, incluso el mismo paciente es un emisor de ruido, por ello, es indispensable la implementación de filtros en nuestro sistema de adquisición de señales mioeléctricas.

Un filtro proporciona una salida constante, desde DC hasta la frecuencia de corte y una vez que se llega a dicha frecuencia de corte, permite o no, según sean las características del filtro el paso de la señal.

Los filtros se clasifican en:

- Filtros Activos
- Filtros Pasivos

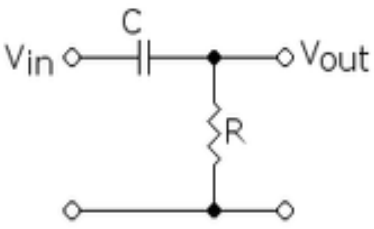


Imagen 10. Ejemplo de un filtro pasivo

Fuente: http://lab51g7.blogspot.mx/2010/11/trabajo-practico-n-11-filtros-activos.html

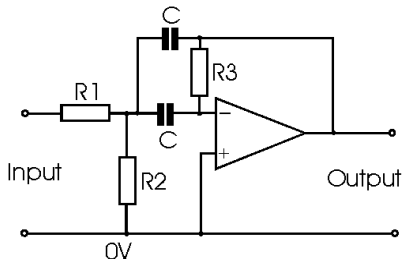


Imagen 11. Ejemplo de un filtro activo

Fuente: http://lab51g7.blogspot.mx/2010/11/trabajo-practico-n-11-filtros-activos.html

Un filtro pasivo se caracteriza por tener elementos pasivos, esto quiere decir que no proporcionan alguna ganancia a tu señal. Cuenta con elementos como resistencias, capacitores e inductores. En su mayoría se utilizan en altas frecuencias y aplicaciones de potencia.

Una aplicación muy común de los amplificadores operacionales es su uso para la construcción de filtros activos, el cual combina elementos pasivos y activos, proporcionando alguna ganancia a tu señal.

Los filtros activos complejos pueden ser divididos en filtros más simples, permitiendo que cada sección sea diseñada por separado y luego puesta en cascada.

Algo a tomar en cuenta al momento de diseñar nuestros filtros es la respuesta en la frecuencia de los filtros, ya que depende de los factores f_0 (frecuencia de corte) y Q (factor de calidad). La forma y comportamiento que puede presentar un filtro se conoce como aproximación, y son tres las aproximaciones más conocidas de la respuesta de un filtro con orden mayor o igual a dos, las cuales son: Butterworth, Bessel y Chebyshev.

La aproximación Butterworth se caracteriza por:

- Mantener su respuesta plana durante la banda de paso; esto quiere decir que mantiene su ganancia constante durante este intervalo.
- Dada la característica anterior, este tipo de aproximación es muy usada en aplicaciones en donde se desea atenuar cierto intervalo de frecuencias sin modificar otras.
- Después de la frecuencia de corte, la magnitud decrece a un ritmo de n (20dB/década), donde n es el orden del filtro. Por ejemplo, la pendiente de un filtro de 2º orden, decrece con una pendiente de 40dB/década.

La aproximación Chebyshev:

- Tiene un rizo en su banda de paso, es decir, la aproximación Chebyshev no presenta una respuesta plana.
- La ventaja del filtro radica en la velocidad con la que se transita de la banda de paso a la banda de rechazo. Por tanto, dado que la banda de transición se reduce, el filtro

rechaza más rápidamente las frecuencias que se encuentran después de la frecuencia de corte.

La aproximación Bessel se caracteriza por:

- Presentar siempre atenuación en todas las bandas, siendo moderada en la banda de paso y mayor en las bandas de transición y rechazo.
- La ventaja del filtro radica en la aproximación lineal de su respuesta en fase. Esta ventaja que se aprovecha en líneas de retardo y en el área de procesamiento digital de señales.

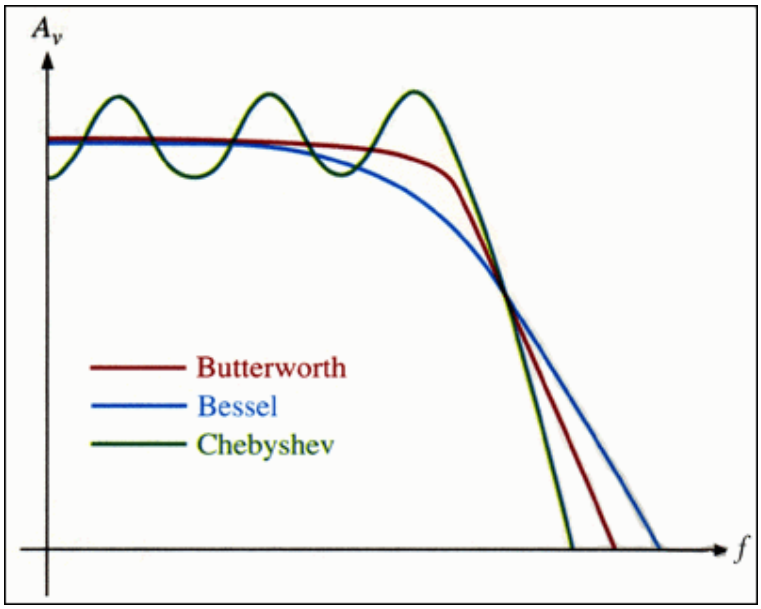


Imagen 12. Comportamiento de las diferentes aproximaciones

Fuente: http://www8.tfe.umu.se/courses/elektro/anakrets/TDV00/html/grupp9/

Funcionalidad del sistema de biorretroalimentación

El siguiente diagrama de bloques generaliza la idea de cómo funcionara a lo largo del proyecto el sistema de biorretroalimentación.

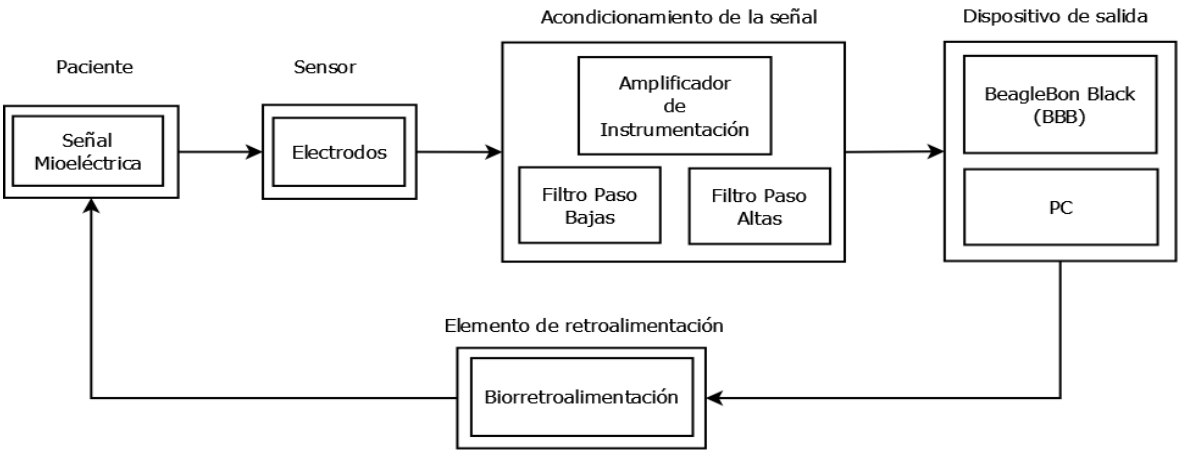


Imagen 13. Diagrama a bloques del sistema de biorretroalimentación

Fuente: Autor

Este diagrama tiene la finalidad de presentar de forma más detallada cada uno de los componentes que integra el sistema de biorretroalimentación.

Paciente

Nombre con el cual nos referiremos a la persona, quien con su actividad muscular; es el encargado de generar las *señales mioeléctricas*, las cuales serán la materia prima con la cual trabajaremos.

Sensor

Convierte una medida física en una señal eléctrica. El elemento en el cual nos auxiliaremos para la conversión de nuestra señal serán los *electrodos*.

Acondicionamiento de la señal

Nos permite adecuar nuestra señal al dispositivo de salida (computadoras o sistemas basados en microcontroladores). Un acondicionamiento simple puede amplificar, reducir y filtrar de acuerdo a la impedancia del dispositivo a utilizar. Aquí entran en juego el *amplificador de instrumentación, el filtro paso bajas y paso altas*.

Dispositivo de salida

El objetivo principal de nuestro dispositivo de salida, es el de mostrar los resultados de manera simple, ya sea por medio de sonidos, dígitos o graficas las cuales sean de ayuda al paciente y al médico especialista para obtener los datos de su interés, posteriormente se procede a dar una valoración según los datos que se obtuvieron en el sistema.

Como se puede observar en la imagen 13, los dispositivos que se usarán para mostrar los resultados serán dos: una tarjeta de desarrollo llamada *BeagleBone Black* y una *computadora (PC)*.

La función que llevará a cabo la tarjeta de desarrollo BeagleBone Black, es la de convertir niveles de voltaje a señales de bits (1´s y 0´s); esto mediante la conversión analógico digital de la señal mioeléctrica, cabe destacar que la información más importante de esta es la *magnitud*. Además, la tarjeta de desarrollo solo soporta voltajes que van de 0 a 1.8 Volts, por lo que en el acondicionamiento de la señal debemos de alguna forma poder limitar los voltajes de entrada al proceso de conversión analógico digital y así poder evitar sobre cargas de voltajes y no llegar a dañar algún componente de la tarjeta.

Otra de sus funciones es comunicarse con el ordenador mediante un cable de datos con una conexión USB, esto con la finalidad de compartir información entre la tarjeta y el ordenador. En su conjunto estas dos características son las que estaremos explotando en mayor medida en la tarjeta de desarrollo.

Por otro lado, una vez que los datos lleguen al ordenador estos serán procesados mediante un *software* desarrollado específicamente para los fines que requiera el sistema.

Elemento de retroalimentación

Es el elemento en el sistema que, una vez que se tenga un resultado final, la información que proporcione ayudará al paciente con su rehabilitación.

En nuestro caso, el sistema contará con la *biorretroalimentación*. La biorretroalimentación es controlada por el ordenador, ya que, al ser nuestro dispositivo de salida, debe de proporcionar la información necesaria para tener la retroalimentación correcta para el funcionamiento del sistema y ayuda al paciente.

Obtención y adecuación de la señal mioeléctrica

La obtención de la señal mioeléctrica será a través de electrodos de superficie conectados a un amplificador de instrumentación. Posteriormente la señal será filtrada mediante un filtro Sallen-Key y adecuada para que el voltaje de salida oscile entre 0 y 1.8 volts.

Nota: Para la alimentación del amplificador de instrumentación y los amplificadores operacionales se utilizará un voltaje simétrico de -12 y +12 Volts.

Amplificación de la señal mioeléctrica

Primeramente, la señal será adquirida mediante electrodos de superficie conectados a un amplificador de instrumentación, para ello se utilizó el integrado de Texas Instruments INA114 con la siguiente configuración.

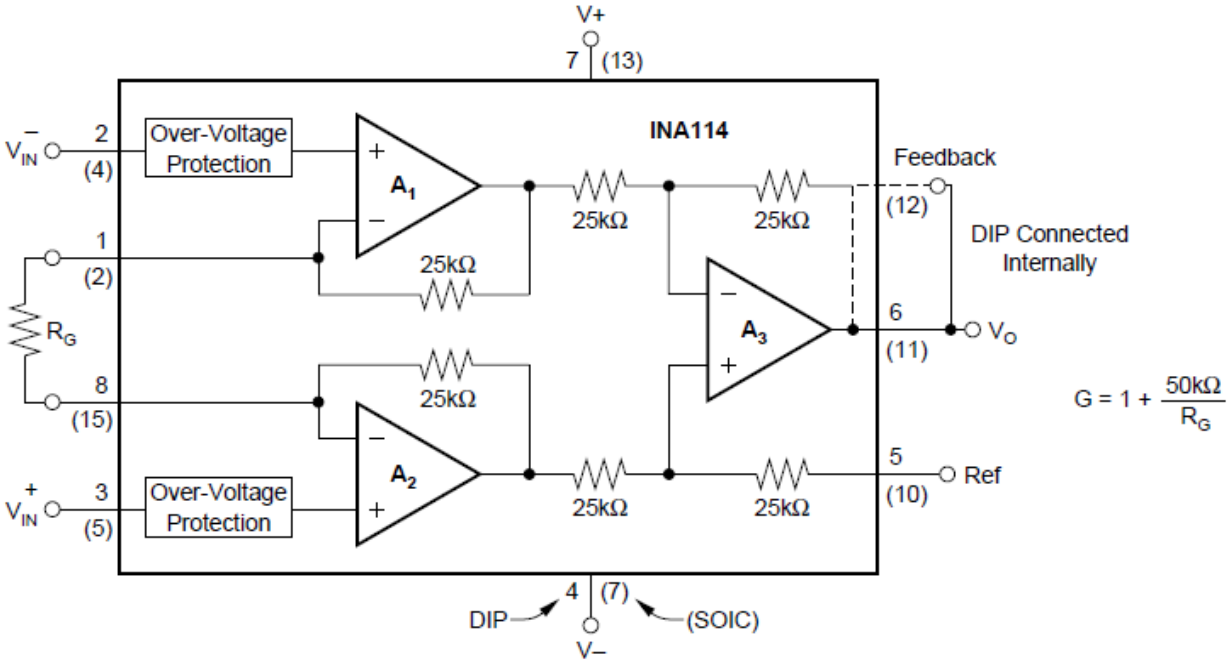


Imagen 14. Integrado INA114

Fuente: www.ti.com/lit/ds/symlink/ina114.pdf

El fabricante nos proporciona la fórmula para calcular la ganancia total la cual está dada por:

$$G = 1 + \frac{50k\Omega}{R_G}$$

Ya que se desea obtener la mayor ganancia posible sin llegar a saturar el integrado se optó por tener una $R_G = 39\Omega$. Así que la ganancia sería de:

$$G = 1 + \frac{50k\Omega}{39\Omega} = 1283$$

Se llegó a esta ganancia mediante la variación de resistencia, con el objetivo de obtener una ganancia real máxima, sin llegar a la saturación del circuito.

La forma en la cual se obtuvo esta resistencia, fue mediante el uso de un potenciómetro, el cual fue variando el valor R_G para encontrar el valor óptimo y dejarlo fijo.

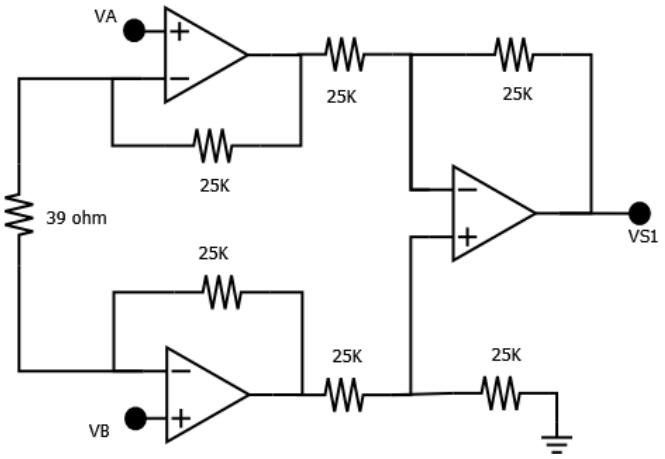


Imagen 15. Diagrama amplificador de instrumentación

Fuente: Autor

Pruebas y simulación del amplificador de instrumentación

Las pruebas y simulaciones del amplificador de instrumentación se llevaron a cabo en al programa NI Multisim 14.0. En el cual se simula el funcionamiento y comportamiento del amplificador de instrumentación con la R_G que se eligió.

Esto es de gran ayuda, ya que si vemos que el comportamiento no es el que nosotros esperamos o deseamos podemos realizar las modificaciones pertinentes antes de tener un diseño final y evitar que nuestro trabajo presente fallas a la hora de implementarlo con los demás componentes que constituyen el sistema.

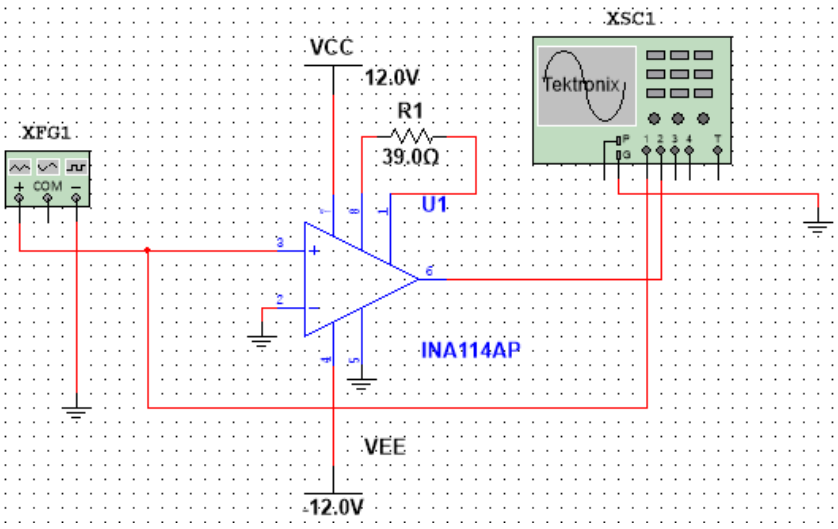


Imagen 16. Simulación en Multisim

Fuente: Autor

Como podemos observar en la simulación de la imagen 17, la señal que se encuentra en color amarillo es nuestra señal de entrada, la cual tiene un voltaje pico a pico de 4 mV (en escala de 2 mV por división). Por otro lado, la señal en azul es nuestra señal de salida que tiene un voltaje pico a pico de 5 Volts (en una escala de 1 Volt por división).

Esto nos habla de que existe una ganancia aproximadamente de 1250 veces la señal de salida con respecto a la de entrada.

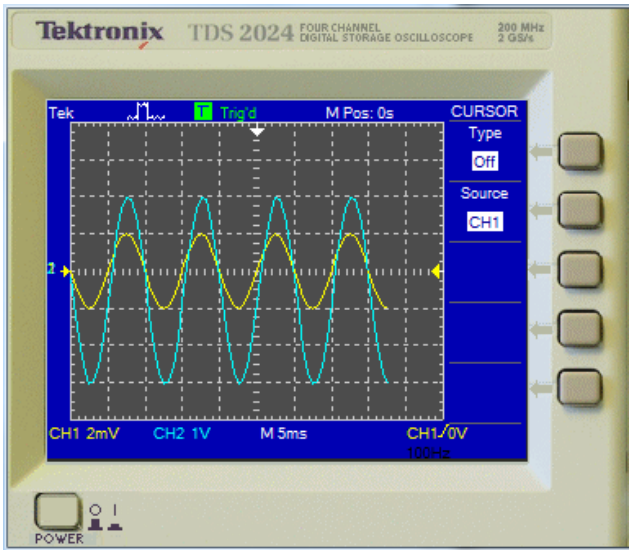


Imagen 17. Simulación del amplificador de instrumentación

Fuente: Autor

Filtrado

Antes de empezar con nuestro diseño de filtros debemos saber en qué frecuencias oscila nuestra señal de trabajo. La señal mioeléctrica oscila entre una frecuencia desde DC llegando inclusive hasta los 10 kHz, este valor va variando dependiendo del músculo en el cual nos estemos enfocando, ya que no es la misma frecuencia que puede generar el dedo meñique de cualquiera de nuestras dos manos comparado con la frecuencia generada en el músculo gemelo interno ubicado en la parte inferior trasera de cualquiera de nuestras piernas. Los músculos faciales por ejemplo, tienen movimientos tenues y no oscilan en frecuencias más allá de 1 kHz.

Cabe señalar que nuestro sistema debe tener una frecuencia de trabajo considerable, ya que en un futuro se desea implementar no solamente en músculos faciales, sino inclusive en los músculos de las extremidades del cuerpo (brazos y piernas).

Por ello, se estableció una frecuencia de trabajo que va desde los 10 Hz hasta 1 KHz. Entonces, nuestro filtro paso bajas y paso altas trabajaran con dichas frecuencias de corte.

Para el diseño de nuestros filtros, se decidió utilizar filtros de segundo orden configuración tipo Sallen-Key con ganancia variable y unitaria. La configuración utilizada se obtuvo del libro "*Operational Amplifiers Design and Applications* (Jerald G. Graeme y Gene E. Tobey)".

- Amplificador Operacional No Inversor

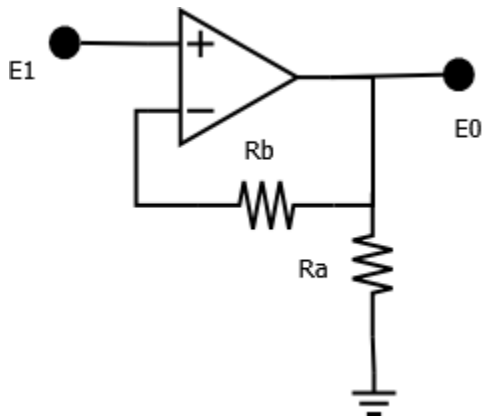


Imagen 18. Amplificador operacional no inversor

Fuente: Autor

Del amplificador sabemos que:

$$\frac{E_0}{E_1} = 1 + \frac{R_b}{R_a} = K$$

- Configuración Filtro de Fuente de Voltaje Controlada por Voltaje de segundo orden

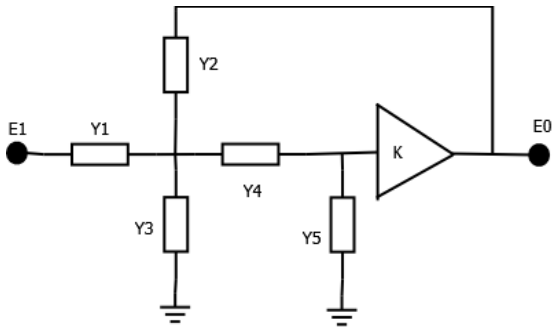


Imagen 19. Filtro de fuente de voltaje controlado por voltaje de segundo orden

Fuente: Autor

Mediante la función de transferencia:

$$\frac{E_0}{E_1} = \frac{KY_1Y_4}{Y_5(Y_1 + Y_2 + Y_3 + Y_4) + [Y_1 + Y_2(1 - K) + Y_3]}$$

Podemos obtener los diferentes tipos de filtros:

-Paso Bajas

-Paso Altas

-Paso Bandas

-Supresor de Bandas

Filtro paso bajas

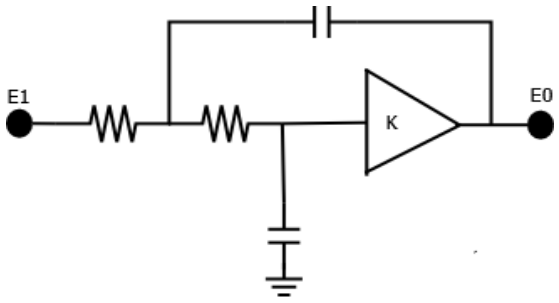


Imagen 20. Filtro paso bajas

Fuente: Autor

Para el caso particular de un filtro paso bajas tenemos que, sustituyendo en la función de transferencia los valores de la configuración de la imagen 20, obtenemos:

$$\frac{E_0}{E_1} = \frac{\frac{K}{R_1 R_2 C_1 C_2}}{s^2 + s\left[\frac{1}{R_1 C_1} + \frac{1}{R_2 C_1} + \frac{1-K}{R_2 C_1}\right] + \frac{1}{R_1 R_2 C_1 C_2}}$$

Los parámetros con los que trabajarán serán:

$$H_o = K$$

$$\omega_o = \sqrt{\frac{1}{R_1 R_2 C_1 C_2}}$$

$$\alpha = \left(\frac{R_2 C_2}{R_1 C_1}\right)^{\frac{1}{2}} + \left(\frac{R_1 C_2}{R_2 C_1}\right)^{\frac{1}{2}} + \left(\frac{R_1 C_1}{R_2 C_2}\right)^{\frac{1}{2}} - K\left(\frac{R_1 C_1}{R_2 C_2}\right)^{\frac{1}{2}}$$

El proceso de diseño según él libro nos dice que debemos tener:

$$H_o\,, \alpha\,, \omega_o = 2\pi f_o$$

Escogemos:

$$C_1 = C_2 = C$$

Y calculamos:

$$K = H_o > 2$$

$$R_2 = \frac{\alpha}{2\omega_o C}\left[1 + \sqrt{1 + \frac{4(H_o - 2)}{\alpha^2}}\right]$$

$$R_1 = \frac{1}{\omega_o^2 C^2 R_2}$$

Para nuestro filtro paso bajas con una frecuencia de corte de 1 kHz tenemos una K variable, la cual va desde 4.3 hasta 5.3 como máximo, esto es, dado que se conectó una resistencia fija con un valor de 33 KΩ en serie con un potenciómetro de 10 KΩ, esto se hizo para poder tener una ganancia ajustable en caso de que llegue a ser necesaria.

Así que tendríamos una K mínima de 4.3:

$$H_{o\,min} = K_{min} = \left(\frac{33K\Omega + 0K\Omega}{10K\Omega} + 1\right) = 4.3$$

Y una K máxima de 5.3:

$$H_{o\,max} = K_{max} = \left(\frac{33K\Omega + 10K\Omega}{10K\Omega} + 1\right) = 5.3$$

Por lo que se proponen los siguientes valores:

$$Ra = 10K\Omega$$
$$Rb = 33K\Omega + Potenciometro = 10K\Omega$$
$$fo = 1KHz$$
$$\alpha = 0.5$$
$$C = C1 = C2 = 120nf$$

Para $R_{2\,min}$ se tiene que:

$$R_{2\,min} = \frac{0.5}{2(2\pi)(1KHz)(120nf)}\left[1 + \sqrt{1 + \frac{4(4.3-2)}{0.5^2}}\right]$$

$$R_{2\,min} = 2.37K\Omega \approx Valor\ comercial\ 2.3K\Omega$$

Para $R_{2\,max}$ se tiene que:

$$R_{2\,max} = \frac{0.5}{2(2\pi)(1KHz)(120nf)}\left[1 + \sqrt{1 + \frac{4(5.3-2)}{0.5^2}}\right]$$

$$R_{2\,max} = 2.76K\Omega \approx Valor\ comercial\ más\ 2.7K\Omega$$

Para $R_{1\,min}$ se tiene que:

$$R_{1\,min} = \frac{1}{[(2\pi)(1KHz)(120nf)]^2\ (120nf)^2(2)(2.37K\Omega)}$$

$$R_{1\,min} = 742\Omega \approx Valor\ comercial\ más\ cercano\ 680\Omega$$

Para $R_{1\,max}$ se tiene que:

$$R_{1\,max} = \frac{1}{[(2\pi)(1KHz)(120nf)]^2\ (120nf)^2(2)(2.76K\Omega)}$$

$$R_{1\,max} = 637\Omega \approx Valor\ comercial\ más\ cercano\ 680\Omega$$

Tenemos que para R1 el valor de la resistencia es de 680 **Ω**, sin embargo, para R2 tenemos dos posibles valores, para saber cuál utilizar demostraremos a que frecuencia se da el corte para cada valor.

Para $R_{2\,min}$ se tiene que:

$$\omega_o = \frac{\sqrt{\frac{1}{(680\Omega)(2.3K\Omega)(120nf)(120nf)}}}{2\pi}$$

$$\omega_o = 1060\ Hz$$

Para $R_{2\,max}$ se tiene que:

$$\omega_o = \frac{\sqrt{\frac{1}{(680\Omega)(2.7K\Omega)(120nf)(120nf)}}}{2\pi}$$

$$\omega_o = 978\ Hz$$

Por decisión de diseño dada la frecuencia de corte por cada resistencia, se opta por utilizar el valor de $R_{2\,max}$, para que finalmente tengamos los siguientes valores de resistencias:

$$R_1 = 680\Omega$$

$$R_2 = 2.7K\Omega$$

Quedando nuestro diagrama final de la siguiente manera:

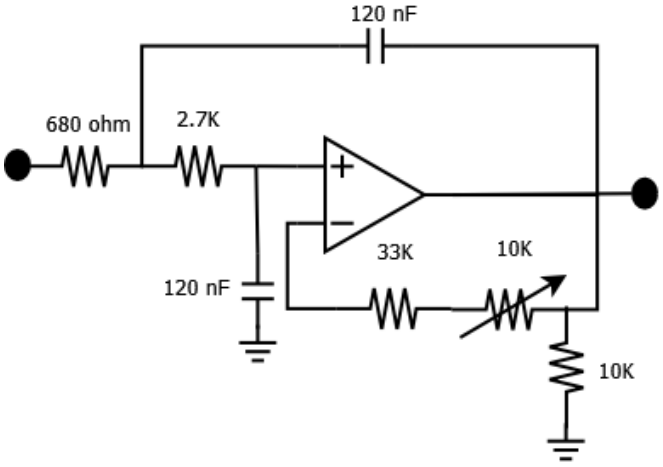


Imagen 21. Filtro paso bajas final

Fuente: Autor

Pruebas de simulación de la respuesta en frecuencia del filtro paso bajas

Una manera de tener una idea más visible de si un filtro llegará a cortar en la frecuencia que nosotros deseamos según nuestro diseño y cálculos, es mediante un diagrama de Bode, el cual es una representación de la respuesta en frecuencia de un filtro y está definido como:

$$Lm = 20 \log |G(j\omega)|$$

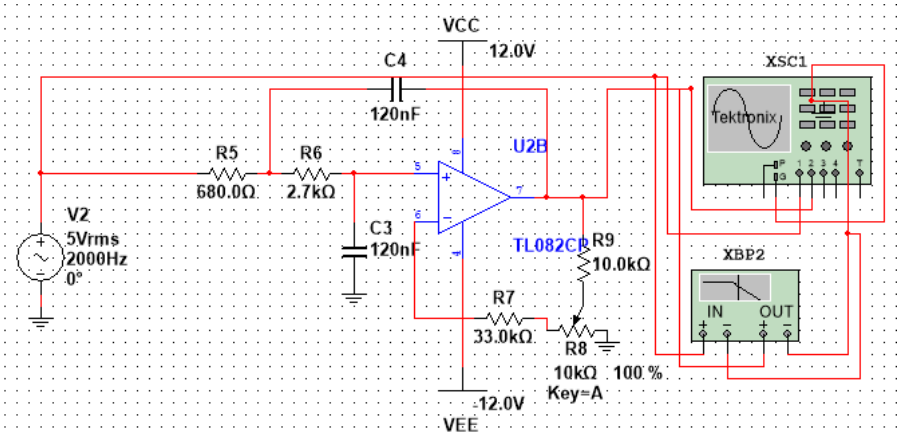


Imagen 22. Simulación del filtro paso bajas en Multisim

Fuente: Autor

Como podemos observar en la imagen 23 a 8dB (decibeles) tenemos una frecuencia de aproximadamente 370 Hz y conforme disminuimos en dB esta frecuencia va aumentando tal como se muestra en la imagen 24, en la cual su frecuencia es de aproximadamente 870 Hz.

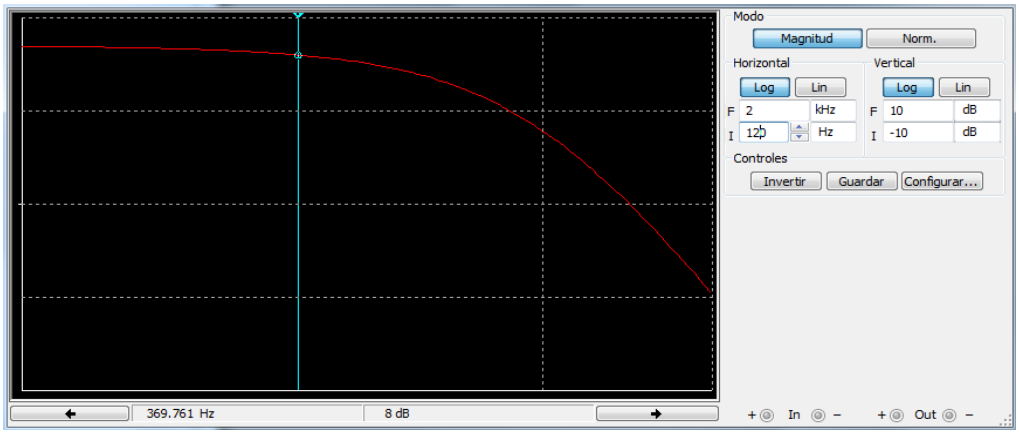


Imagen 23. Diagrama de Bode del filtro paso bajas a 8 dB

Fuente: Autor

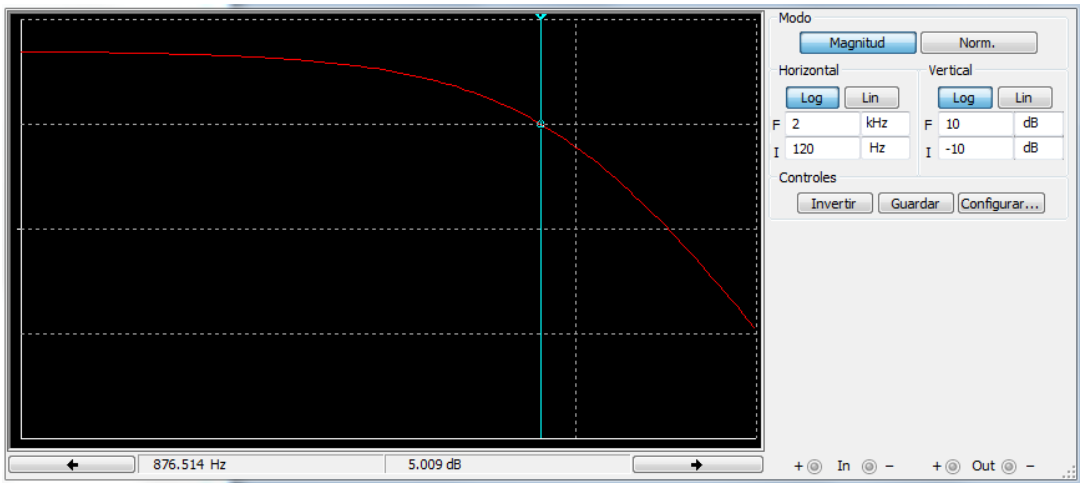


Imagen 24. Diagrama de Bode del filtro paso bajas a 5 dB

Fuente: Autor

Filtro paso altas

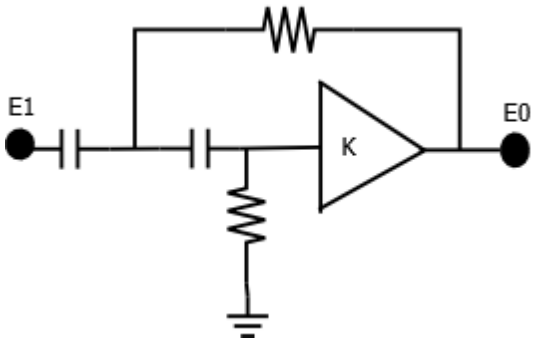


Imagen 25. Filtro paso altas

Fuente: Autor

Por otro lado, para el caso particular de un filtro paso altas tenemos que, sustituyendo en la función de transferencia las resistencias y capacitores de la imagen 25 tendremos que:

$$\frac{E_0}{E_1} = \frac{Ks^2}{s^2 + s\left[\frac{1}{R_2C_1} + \frac{1}{R_2C_2} + \frac{1-K}{R_1C_1}\right] + \frac{1}{R_1R_2C_1C_2}}$$

Los parámetros con los que trabajaremos serán:

$$H_o = K$$

$$\omega_o = \sqrt{\frac{1}{R_1R_2C_1C_2}}$$

$$\alpha = \left(\frac{R_1 C_1}{R_2 C_2}\right)^{\frac{1}{2}} + \left(\frac{R_1 C_2}{R_2 C_1}\right)^{\frac{1}{2}} + \left(\frac{R_2 C_2}{C_1 C_1}\right)^{\frac{1}{2}} - K\left(\frac{R_2 C_2}{R_1 C_1}\right)^{\frac{1}{2}}$$

El proceso de diseño según el libro nos dice que debemos tener:

$$H_o\ , \alpha\ , \omega_o = 2\pi f_o$$

Escogemos:

$$C_1 = C_2 = C$$

Y calculamos:

$$R_1 = \frac{\alpha + \sqrt{\alpha^2 + 8\ (H_o - 1)}}{4\omega_o C}$$

$$R_2 = \left(\frac{4}{\omega_o C}\right)\left(\frac{1}{\sqrt{\alpha^2 + 8\ (H_o - 1)}}\right)$$

Se proponen los siguientes valores:

$$Ra = 10K\Omega$$
$$Rb = 10K\Omega$$
$$fo = 10Hz$$
$$\alpha = 0.5$$
$$C = C1 = C2 = 120nf$$

Sustituyendo nuestros valores en Ho, tenemos:

$$H_o = K = \left(\frac{10K\Omega}{10K\Omega} + 1\right) = 2$$

Para R_1 tenemos que:

$$R_1 = \frac{0.5 + \sqrt{0.5^2 + 8\ (2 - 1)}}{(4)(2\pi)(10Hz)(120nf)}$$

$$R_1 = 111.81K\Omega\ \approx\ Valor\ comercial\ más\ cercano\ 110K\Omega$$

Para R_2 se tiene:

$$R_2 = \left(\frac{4}{(2\pi)(10Hz)(120nf)}\right)\left(\frac{1}{\sqrt{0.5^2 + 8\ (2 - 1)}}\right)$$

$$R_2 = 184.7K\Omega \approx\ Valor\ comercial\ más\ cercano\ 180K\Omega$$

Finalmente tenemos los valores correspondientes de:

$$R_1 = 110K\Omega$$

$$R_2 = 180K\Omega$$

Quedando nuestro diagrama final de la siguiente manera:

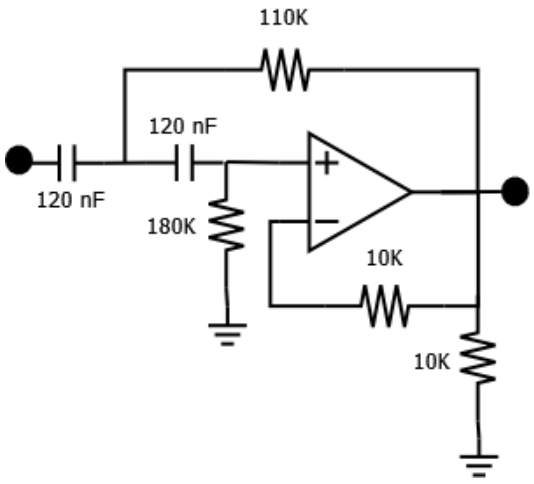


Imagen 26. Filtro paso altas final

Fuente: Autor

Pruebas de simulación de la respuesta en frecuencia del filtro paso altas

De igual forma que en los casos anteriores (amplificador de instrumentación y filtro paso bajas), se realizó una simulación en el programa Multisim para tener una idea del comportamiento de nuestra configuración y valores calculados para el filtro paso altas.

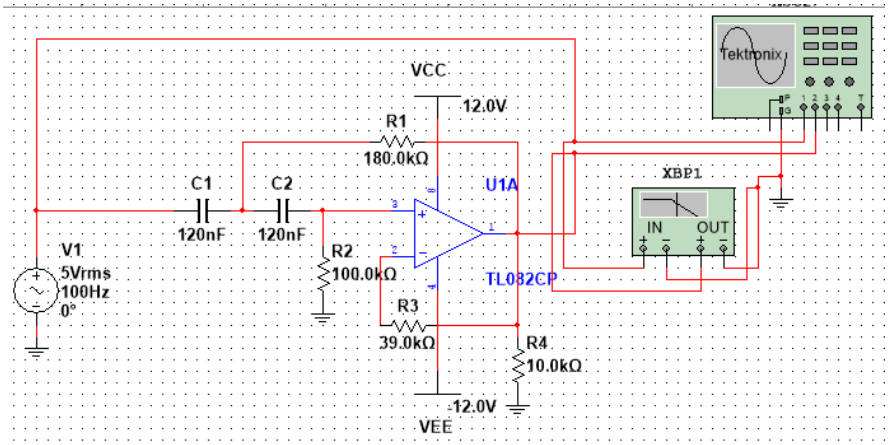


Imagen 27. Simulación del filtro paso bajas en Multisim

Fuente: Autor

De igual manera se presenta su respuesta en frecuencia mediante la representación de un diagrama de bode. En la cual, se pude observar que en la imagen 28 tenemos aproximadamente 10 Hz a 8 dB y conforme vamos aumentando en frecuencia los dB disminuyen, tal y como se muestra en la imagen 29.

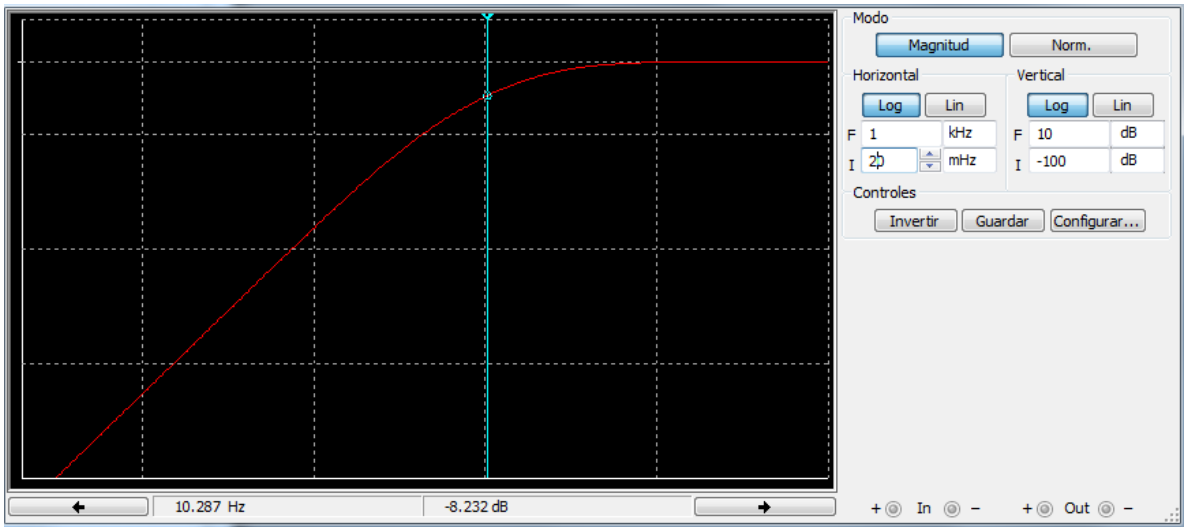


Imagen 28. Diagrama de bode del filtro paso altas a 8dB

Fuente: Autor

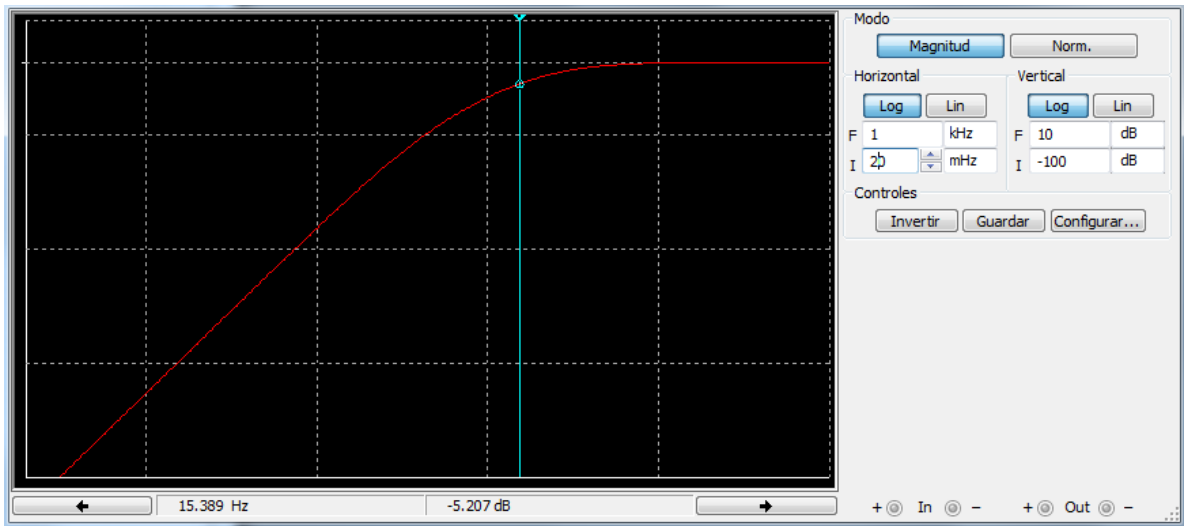


Imagen 29. Diagrama de bode de filtro paso altas a 5dB

Fuente: Autor

Protección del equipo digital (BeagleBone Black)

Para la protección de nuestra tarjeta de trabajo BeagleBone Black, se implementó un rectificador de onda completa de precisión, con el cual garantizamos que nuestra señal solo oscilará entre valores de voltaje positivos (de 0 a +12 Volts y no de -12 a +12 Volts); al igual que un divisor de voltaje, ya que como recordamos solo podemos trabajar con voltajes que van de 0 a 1.8 Volts.

Rectificador de onda completa de precisión

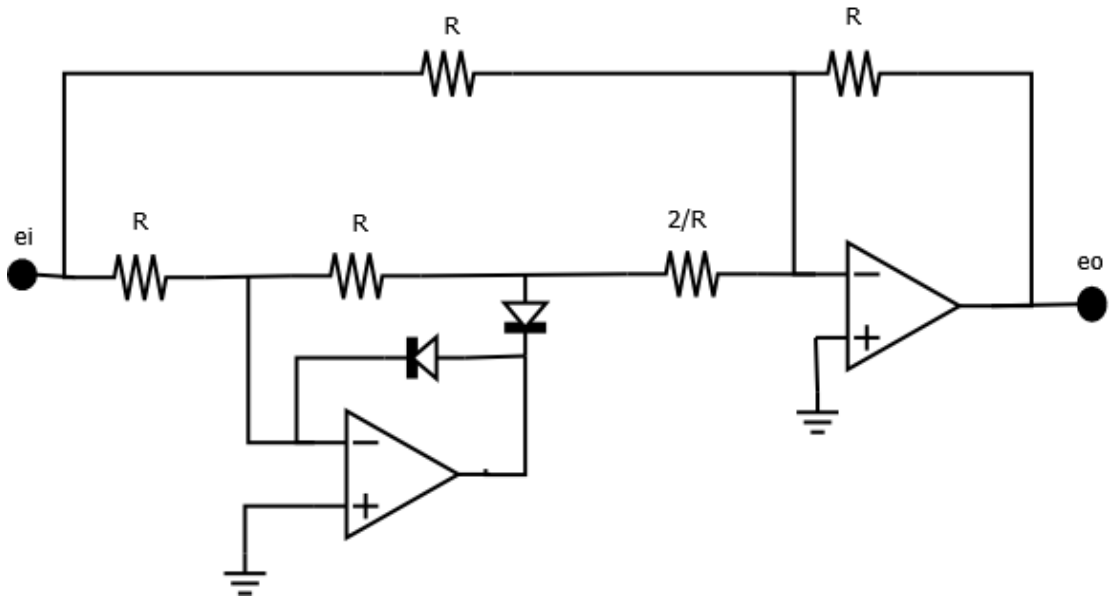


Imagen 30. Rectificador de onda completa de precisión

Fuente: Autor

En la imagen 30 podemos observar los diferentes elementos que componen un rectificador de onda completa de precisión, así como la conexión entre cada uno de ellos. Ya que los valores en las resistencias y en cualquier otro componente eléctrico no son totalmente exactos, se optó por utilizar un trimpot, con la finalidad de compensar la diferencia de valores entre las resistencias, y así poder tener valores entre las resistencias lo más exactos posibles.

El trimpot se coloca en la entrada no inversora del primer amplificador operacional, tal y como se muestra en la imagen 31, el valor del trimpot debe de ser lo bastante grande para poder tener un amplio rango de valores diferentes; además el diseño del rectificador nos indica que podemos proponer un valor de R cualquiera.

Los valores que se eligieron fueron R = 10 KΩ y el valor del trimpot es de 5 KΩ.

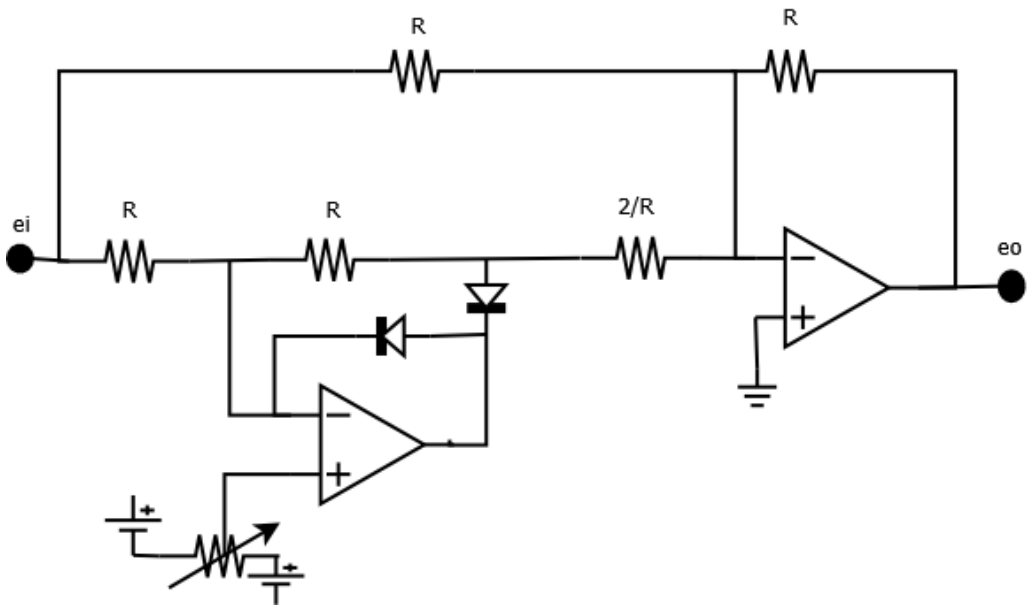


Imagen 31. Rectificador de onda completa de precisión con diferenciador de resistencias

Fuente: Autor

Adicional al rectificador de onda completa, se implementa un divisor de voltaje, ya que nuestro voltaje de salida en cada rectificador será en un rango de 0 a 12 Volts y la tarjeta BeagleBone Black solo soporta voltajes en un rango de 0 a 1.8 Volts. Es por ello que debemos ser muy precisos y rigurosos en los límites que nos proporciona el fabricante, para así evitar daños.

Aplicando la fórmula del divisor de voltaje tenemos que:

$$V_2 = \frac{R_2}{R_1 + R_2} V$$

Donde

$$V_2 = 1.8\, V$$
$$V = 12\, V$$

Sustituyendo valores, tenemos una relación entre resistencias de:

$$\frac{V_2}{V} = \frac{R_2}{R_1 + R_2}$$
$$\frac{R_2}{R_1 + R_2} = \frac{1.8\, V}{12\, V}$$

Proponiendo R_2 = 120KΩ y sustituyendo, tenemos:

$$\frac{120K\Omega}{R_1 + 120K\Omega} = \frac{1.8}{12}$$

$$(12)(120K\Omega) = (1.8)(R_1 + 120K\Omega)$$

$$1.44M\Omega = 1.8R_1 + 216K\Omega$$

$$1.44M\Omega - 216K\Omega = 1.8R_1$$

$$1.224M\Omega = 1.8R_1$$

$$R_1 = 680K\Omega$$

Por lo que nuestro sistema de protección para la tarjeta de trabajo quedaría finalmente de la siguiente manera:

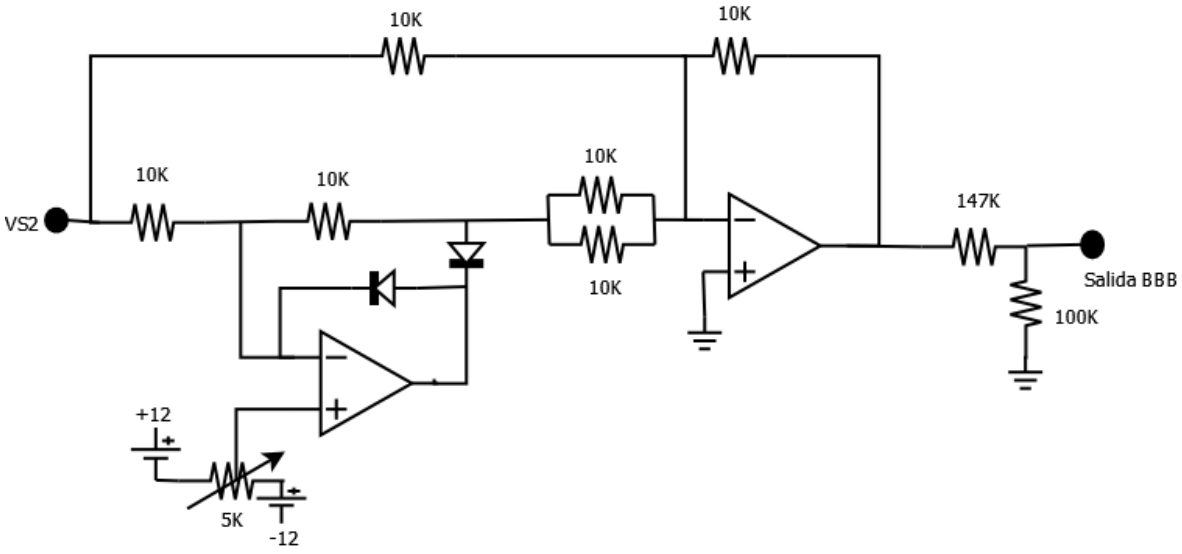


Imagen 32. Rectificador de onda completa de precisión con valores finales

Fuente: Autor

Pruebas del rectificador de onda completa sin divisor de voltaje

De igual forma ayudándonos del software Multisim se simuló el comportamiento del rectificador de onda completa. Como primer caso, la simulación se basó en el comportamiento enteramente del rectificador.

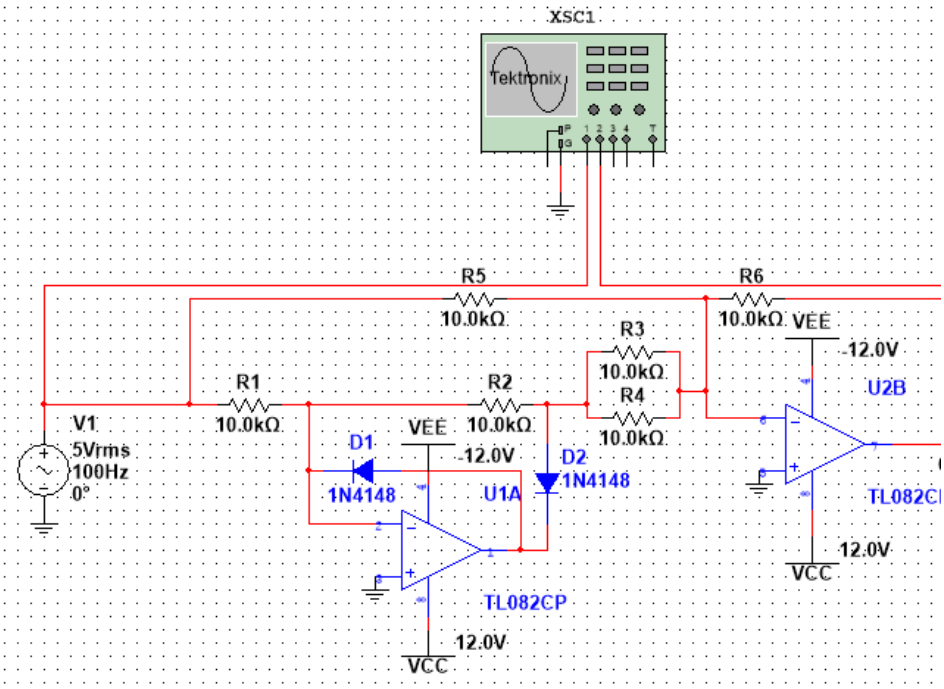


Imagen 33. Simulación del rectificador de onda completa en Multisim

Fuente: Autor

Como podemos observar en la imagen 34, tenemos nuestra entrada con un voltaje pico a pico de 12 Volts (en una escala de 5 Volts por división). Por otro lado, en la señal de salida la cual está representada de color azul, se puede observar que solo se muestra durante el ciclo de la señal la parte positiva de la misma, a diferencia de la entrada que posee tanto la parte positiva como negativa, además de no aportar ganancia o pérdida alguna a la señal.

Esto nos indica que los valores proporcionados y el diseño son los correctos y podemos proceder añadiendo el divisor de voltaje.

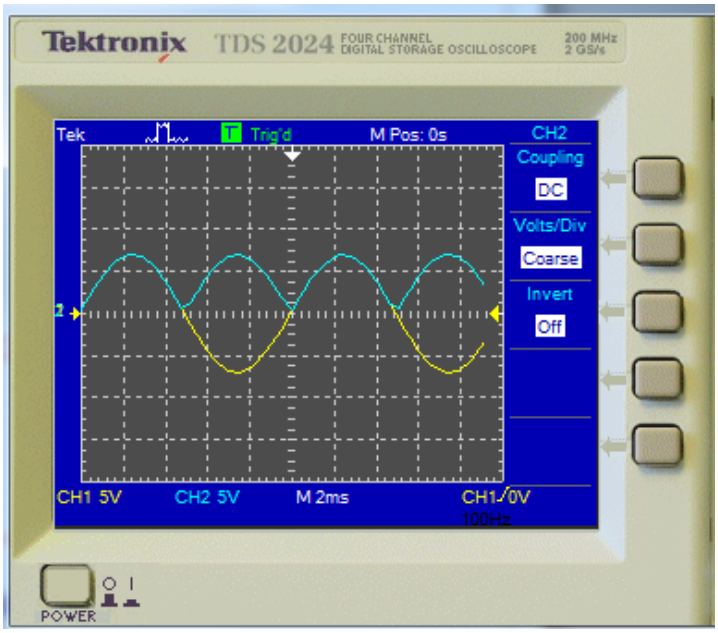


Imagen 34. Pruebas del rectificador de onda completa sin divisor de voltaje

Fuente: Autor

Pruebas del rectificador de onda completa con divisor de voltaje

Como podemos observar en la imagen 35, a nuestro diseño le hemos agregado un divisor de voltaje, esto con el fin de proteger nuestra tarjeta de trabajo BeagleBone Black evitando superar el voltaje soportado en los puertos ADC.

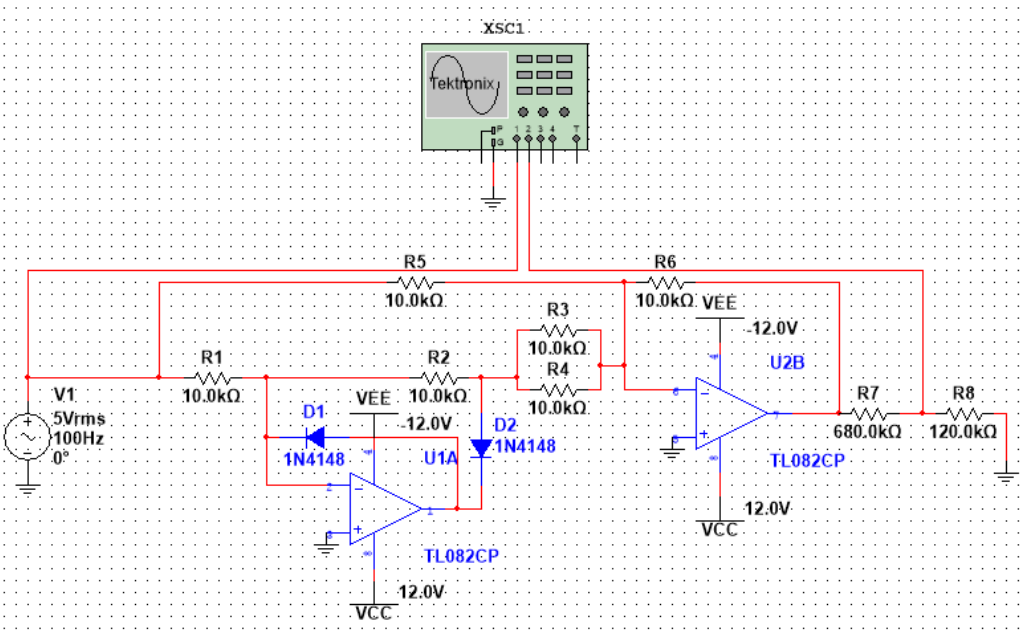


Imagen 35. Simulación del rectificador de onda completa con divisor de voltaje en Multisim

Fuente: Autor

En la imagen 36 se puede observar una reducción considerable de voltaje en nuestra señal de entrada, la cual es de color amarillo y tiene un voltaje pico a pico de 12 Volts (en una escala de 2 Volts por división). La señal en azul representa nuestra señal de salida rectificada y reducida, la cual tiene un voltaje de 1.8 Volts (en una escala de 2 Volts por división) y sólo posee la parte positiva.

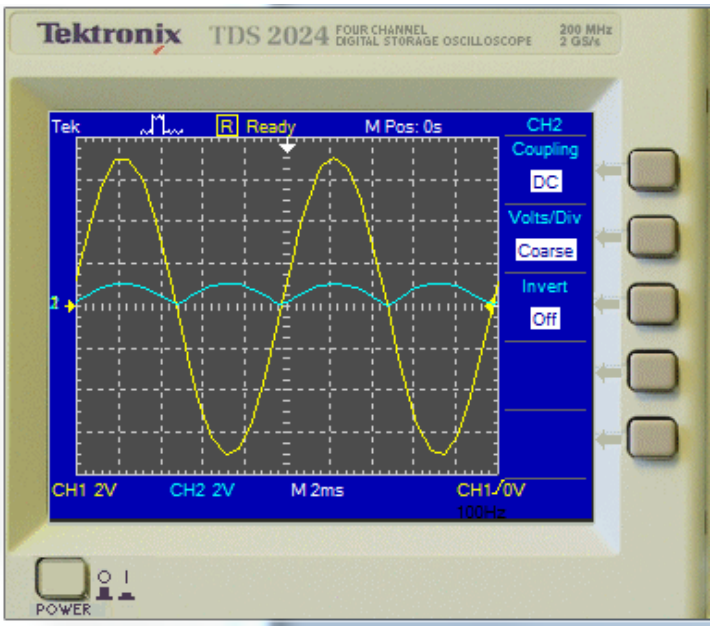


Imagen 36. Pruebas del rectificador de onda completa con divisor de voltaje

Fuente: Autor

Diagramas y tarjeta final

A continuación, se muestra el diseño en PCB para el planchado y ensamblado del sistema.

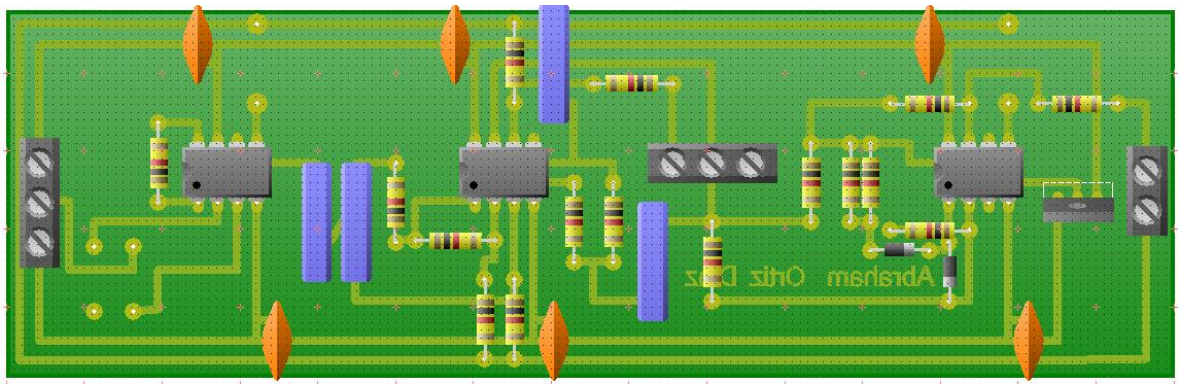


Imagen 37. Diseño de la tarjeta de adquisición, filtrado y rectificación en PCB con componentes

Fuente: Autor

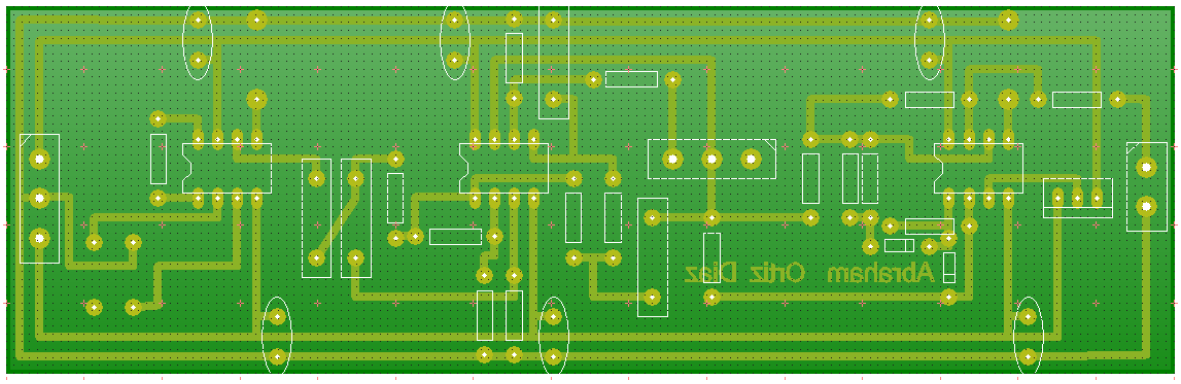


Imagen 38. Diseño de la tarjeta de adquisición, filtrado y rectificación en PCB sin componentes

Fuente: Autor

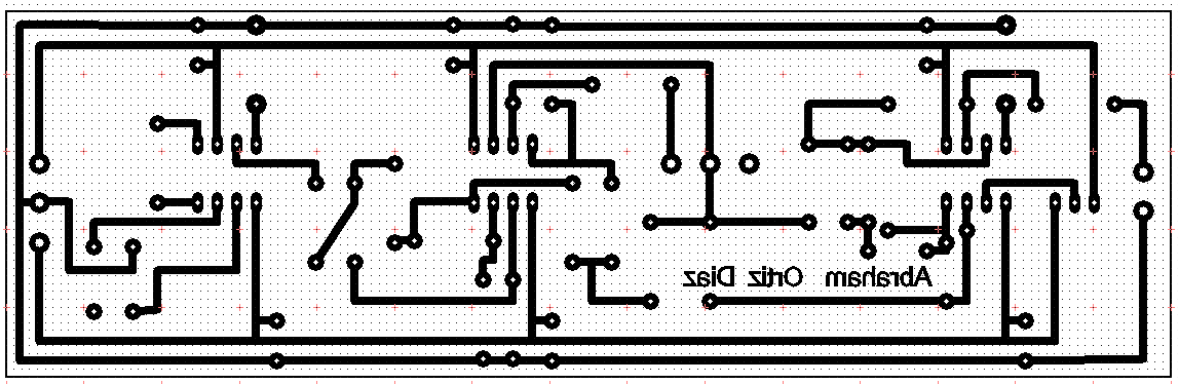


Imagen 39. Pistas de la tarjeta de adquisición, filtrado y rectificación en PCB

Fuente: Autor

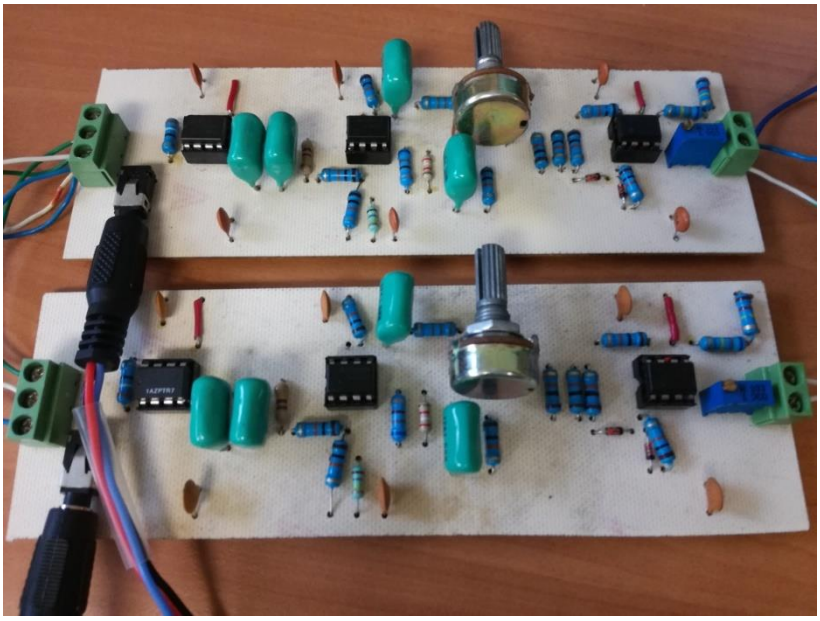


Imagen 40. Tarjeta de adquisición, filtrado y rectificación física

Fuente: Autor

Digitalización y procesamiento de las señales

Digitalizar es un proceso mediante el cual se tiene una señal en el campo de lo analógico (voltaje o corriente) y se desea pasar al campo de lo digital (binario 1's y 0's).

Existen varios dispositivos electrónicos capaces de llevar a cabo la digitalización de señales analógicas a señales digitales, estos se conocen como convertidores Analógico Digitales o ADC (por sus siglas en inglés de Analog-to-Digital Converter) entre los cuales están los ADC flash o paralelo, ADC de aproximaciones sucesivas, ADC de contador de rampas en escalera; por mencionar sólo algunos.

Existen varias etapas por las cuales debe de pasar una señal analógica para poder convertirse en digital. Las cuales consiste en:

- ✓ Muestreo: Consiste en tomar muestras periódicas de una señal.
- ✓ Retención: El dispositivo debe de ser capaz de retener las muestras para su evaluación.
- ✓ Cuantificación: Es el proceso mediante el cual se le asigna un margen de valor de una señal a un único valor de salida.
- ✓ Codificación: Es traducir a un valor binario los valores obtenidos en el proceso de cuantificación

Esta es la base de los diferentes convertidores analógicos digitales, los cuales con las mejoras de sus etapas de una forma individual o conjunta ha sido posible el desarrollo de variantes.

ADC en la BeagleBone Black

El procesador AM335x de la BeagleBone Black es el encargado de realizar la conversión analógico-digital mediante el método de aproximaciones sucesiva.

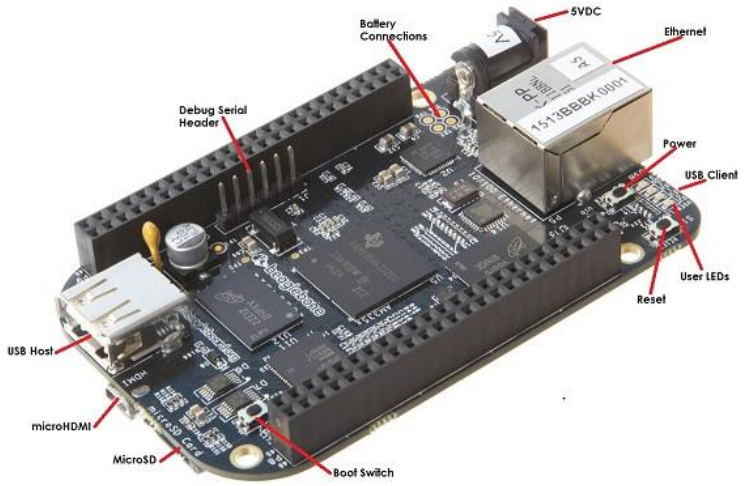


Imagen 41. BeagleBone Black

Fuente: http://elinux.org/Beagleboard:BeagleBoneBlack

Los puertos digitales se encuentran identificados en la imagen 42.

P9				P8			
DGND	1	2	DGND	DGND	1	2	DGND
VDD_3V3	3	4	VDD_3V3	GPIO_38	3	4	GPIO_39
VDD_5V	5	6	VDD_5V	GPIO_34	5	6	GPIO_35
SYS_5V	7	8	SYS_5V	GPIO_66	7	8	GPIO_67
PWR_BUT	9	10	SYS_RESETn	GPIO_69	9	10	GPIO_68
GPIO_30	11	12	GPIO_60	GPIO_45	11	12	GPIO_44
GPIO_31	13	14	GPIO_50	GPIO_23	13	14	GPIO_26
GPIO_48	15	16	GPIO_51	GPIO_47	15	16	GPIO_46
GPIO_5	17	18	GPIO_4	GPIO_27	17	18	GPIO_65
I2C2_SCL	19	20	I2C2_SDA	GPIO_22	19	20	GPIO_63
GPIO_3	21	22	GPIO_2	GPIO_62	21	22	GPIO_37
GPIO_49	23	24	GPIO_15	GPIO_36	23	24	GPIO_33
GPIO_117	25	26	GPIO_14	GPIO_32	25	26	GPIO_61
GPIO_115	27	28	GPIO_113	GPIO_86	27	28	GPIO_88
GPIO_111	29	30	GPIO_112	GPIO_87	29	30	GPIO_89
GPIO_110	31	32	VDD_ADC	GPIO_10	31	32	GPIO_11
AIN4	33	34	GNDA_ADC	GPIO_9	33	34	GPIO_81
AIN6	35	36	AIN5	GPIO_8	35	36	GPIO_80
AIN2	37	38	AIN3	GPIO_78	37	38	GPIO_79
AIN0	39	40	AIN1	GPIO_76	39	40	GPIO_77
GPIO_20	41	42	GPIO_7	GPIO_74	41	42	GPIO_75
DGND	43	44	DGND	GPIO_72	43	44	GPIO_73
DGND	45	46	DGND	GPIO_70	45	46	GPIO_71

Imagen 42. Puertos ADC

Fuente: Autor

El ADC de la BeagleBone Black cuenta con una resolución de 12 Bits (0 - 4095) con un tiempo de muestreo de 125 ns y un rango de voltaje de entrada de 0 – 1.8 Volts.

Funcion del ADC en nuestro sistema

Una vez nuestra señal mioeléctrica es adquirida, filtrada y libre de ruidos, entrará en uno de los seis canales disponibles para la conversión analógico digital.

Seleccionando el canal de nuestro agrado, este muestreará cada uno de los valores de voltaje que a este lleguen, en una frecuencia de muestreo de 100 Hz; referenciado al reloj de la BeagleBone Black, y por cada 10 muestras se estará calculando el voltaje RMS. Esto equivale a un dato disponible para poder enviar al ordenador por cada voltaje RMS que se calcule, lo que se convierte en una frecuencia de 10 Hz en el envío de datos de la BeagleBone Black a la PC.

El termino RMS se refiere a tensiones sinusoidales variables en el tiempo o formas de onda compleja. Cuando se utiliza para comparar el valor equivalente entre una señal continua y una señal alterna en el tiempo se conoce como valor eficaz. Es por eso ello que utilizamos el valor RMS, ya que independientemente de las frecuencias generadas por la señal mioeléctrica, estas serán representadas continuamente en el tiempo.

Por otro lado, el voltaje RMS está definido como el promedio cuadrático de los valores instantáneos durante un periodo de tiempo, lo cual se traduce matemáticamente como:

$$Y_{ef} = \sqrt{\frac{1}{T}\int_0^T [y(t)]^2 dt}$$

Donde:

- T = 10
- y(t) = Valores en el instante 0 a T de tiempo

La expresión matemática anterior es posible programarla mediante las siguientes lineas de código:

```
if activaCH_1:
    canal1 =  float(ADC.read("P9_39"))
        voltaje_canal1 = canal1 ** 2 #Elevamos al cuadrado para sacar voltaje RMS
        sumaRms_CH1 = sumaRms_CH1 + voltaje_canal1 #Sumamos los valores
        numMuestras_CH1 = numMuestras_CH1 + 1
        if (numMuestras_CH1 == 10):
            divideRms_CH1 = sumaRms_CH1 / numMuestras_CH1
            rmsVoltaje_CH1 = divideRms_CH1 ** 0.5 #Sacamos raiz cuadrada
            rmsVoltaje_CH1 = format(rmsVoltaje_CH1, '.2f')
            canal1_char = str(rmsVoltaje_CH1)
            numMuestras_CH1 = 0
            sumaRms_CH1 = 0.0
            divideRms_CH1 = 0.0
            print "U" + canal1_char
            ser.write('U')
            ser.write(canal1_char)
            ser.write("\n")
```

Imagen 43. Código en python para sacar valor RMS

Fuente: Autor

El código anterior lo que realiza es: primeramente leemos un dato de nuestro canal activo, que va de 0 a 1.8 Volts, internamente la BeagleBone Black hace la conversión de bits y lo traduce a voltaje. Cabe aclarar que el código es similar para los cinco canales restantes.

Continuamos elevando al cuadrado el valor del voltaje leído y vamos realizando la sumatoria de todos los valores elevados al cuadrado hasta que lleguemos a 10 valores. Sacamos un promedio de los valores y realizamos la raíz cuadrada (Voltaje RMS). Este valor es el que enviaremos como dato a la PC para poder graficar.

Puerto UART a USB

La trasferencia de datos entre la BeagleBone Black y el programa en la PC se hace mediante el integrado FT232R, el cual es una interface de puerto serie UART a USB compatible con los pines UART de la BeagleBone Black.

Los pines de conexión de la BeagleBone Black se muestran a continuación en la imagen 44.

P9

DGND	1	2	DGND
VDD_3V3	3	4	VDD_3V3
VDD_5V	5	6	VDD_5V
SYS_5V	7	8	SYS_5V
PWR_BUT	9	10	SYS_RESETn
UART4_RXD	11	12	GPIO_60
UART4_TXD	13	14	GPIO_50
GPIO_48	15	16	GPIO_51
GPIO_5	17	18	GPIO_4
UART1_RTSn	19	20	UART1_CTSn
UART2_TXD	21	22	UART2_RXD
GPIO_49	23	24	UART1_TXD
GPIO_117	25	26	UART1_RXD
GPIO_115	27	28	GPIO_113
GPIO_111	29	30	GPIO_112
GPIO_110	31	32	VDD_ADC
AIN4	33	34	GNDA_ADC
AIN6	35	36	AIN5
AIN2	37	38	AIN3
AIN0	39	40	AIN1
GPIO_20	41	42	UART3_TXD
DGND	43	44	DGND
DGND	45	46	DGND

P8

DGND	1	2	DGND
GPIO_38	3	4	GPIO_39
GPIO_34	5	6	GPIO_35
GPIO_66	7	8	GPIO_67
GPIO_69	9	10	GPIO_68
GPIO_45	11	12	GPIO_44
GPIO_23	13	14	GPIO_26
GPIO_47	15	16	GPIO_46
GPIO_27	17	18	GPIO_65
GPIO_22	19	20	GPIO_63
GPIO_62	21	22	GPIO_37
GPIO_36	23	24	GPIO_33
GPIO_32	25	26	GPIO_61
GPIO_86	27	28	GPIO_88
GPIO_87	29	30	GPIO_89
UART5_CTSn+	31	32	UART5_RTSn
UART4_RTSn	33	34	UART3_RTSn
UART4_CTSn	35	36	UART3_CTSn
UARR5_TXD+	37	38	UART5_RXD+
GPIO_76	39	40	GPIO_77
GPIO_74	41	42	GPIO_75
GPIO_72	43	44	GPIO_73
GPIO_70	45	46	GPIO_71

Imagen 44. Puertos UART

Fuente: Autor

La configuración de la interface UART, así como de la lectura de los pines ADC en la BeagleBone Black está dada por la siguiente línea de código:

```
#!/usr/bin/python
import Adafruit_BBIO.ADC as ADC
import Adafruit_BBIO.UART as UART
import serial
import time

ADC.setup()
UART.setup("UART1")

ser = serial.Serial(port = "/dev/ttyO1", baudrate = 115200, timeout = None)
ser.close()
ser.open()
```

Imagen 45. Código en Python para definir el uso de puertos UART

Fuente: Autor

Conexión fisica

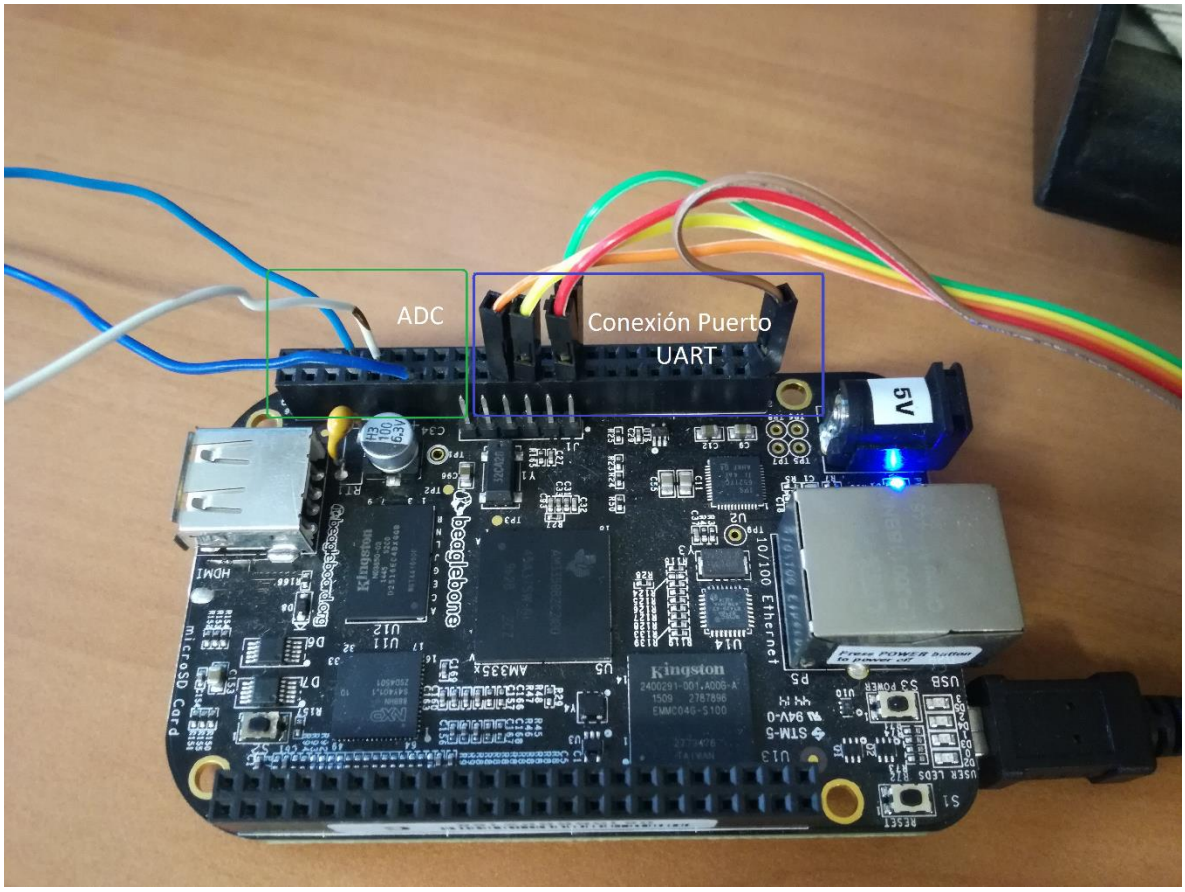


Imagen 46. Conexión de los puertos ADC y UART

Fuente: Autor

En la imagen 46 podemos observar las conexiones correspondientes a la entrada del convertidor analógico digital y su correspondiente referencia a tierra, también se observa la conexión referente a los pines que utiliza el puerto UART.

Biorretroalimentación

En este caso el proceso de biorretroalimentación se realiza mediante el uso de dos programas, uno desarrollado en lenguaje C++ con ayuda de la IDE Visual Studio en su versión 2013, y otro más en lenguaje Python. El software desarrollado en lenguaje Python lleva por nombre *enviaDatos*; por otro lado, el software en lenguaje C++ lleva por nombre *BiorretAOD*.

El uso de *enviaDatos* tiene la finalidad de, una vez que llega nuestra señal analógica, convertirla a digital y colocarla dentro de nuestros rangos de operación y con ello proceder a realizar el uso de los datos enviándolos a la PC mediante un cable de datos.

BiorretAOD lo que hará con los datos enviados por *enviaDatos*, es utilizarlos para poder mostrar al paciente y al médico gráficamente los resultados de la terapia.

La comunicación mediante el puerto serie que va desde el trasmisor (BeagleBone Black y enviaDatos) y el receptor (PC y BiorretAOD) tiene las siguientes especificaciones:

- Baudrate = 115200
- Timeout = None

Uso del software enviaDatos y BiorretAOD

Una vez que tenemos conectados los puertos USB de la BeagleBone como del integrado FT232R, en nuestro navegador de internet, tecleamos la dirección http://192.168.7.2, nos deberá de aparecer algo similar a la imagen 47.

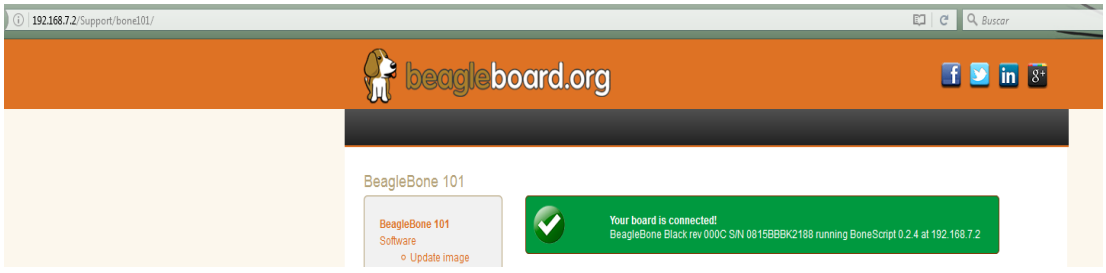


Imagen 47. Check de conexión correcta BBB - PC

Fuente: Autor

Lo cual nos indica que nuestro BeagleBone Black está bien conectada y podemos seguir.

Posteriormente debemos ingresar por SSH a la BeagleBone Black para poder ejecutar el programa que estará leyendo los puertos ADC. Para ello abrimos el software *Putty*.

Imagen 48. Icono del software putty

Fuente: Autor

Ya abierto, configuramos. Nuestro *Host Name* es 192.168.7.2, el puerto es el 22 y la conexión es SSH. Una vez terminado damos en Open. Todos estos valores son los de default de la tarjeta BeagleBone Black.

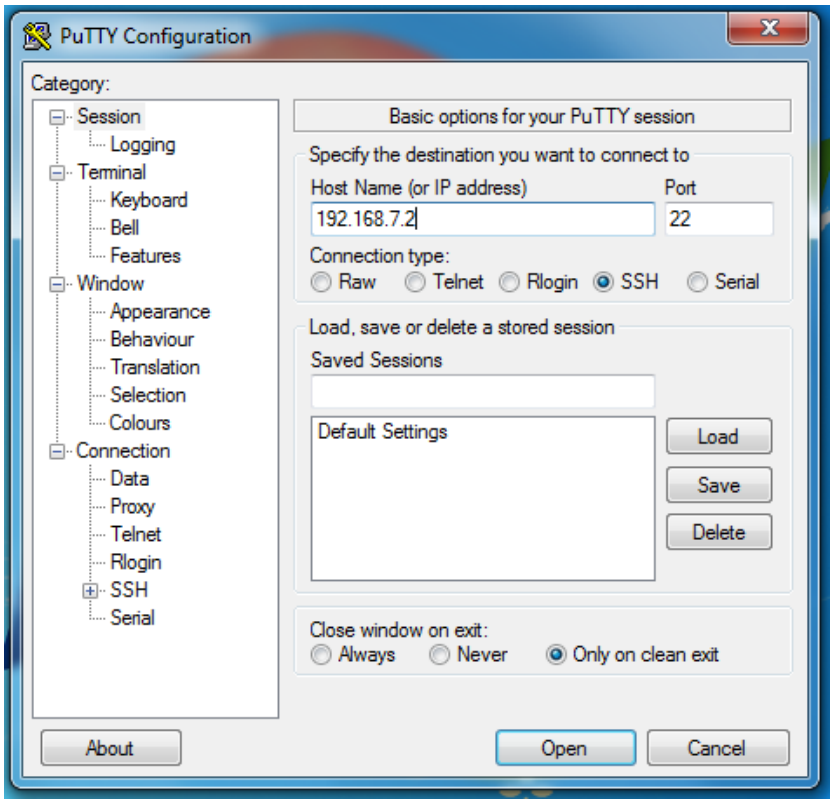


Imagen 49. Configuración para la conexión por SSH

Fuente: Autor

Cuando se haya establecido la conexión nos aparecerá una terminal con el texto *login as:*

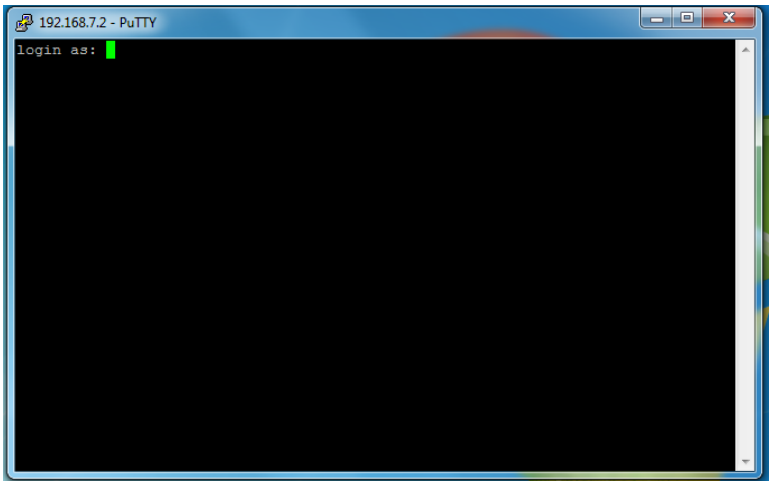


Imagen 50. Login BBB

Fuente: Autor

Escribimos *root* y presionamos enter. Cuando estemos dentro de la BeagleBone Black el promp debería ser similar a *root@beaglebone:~#*

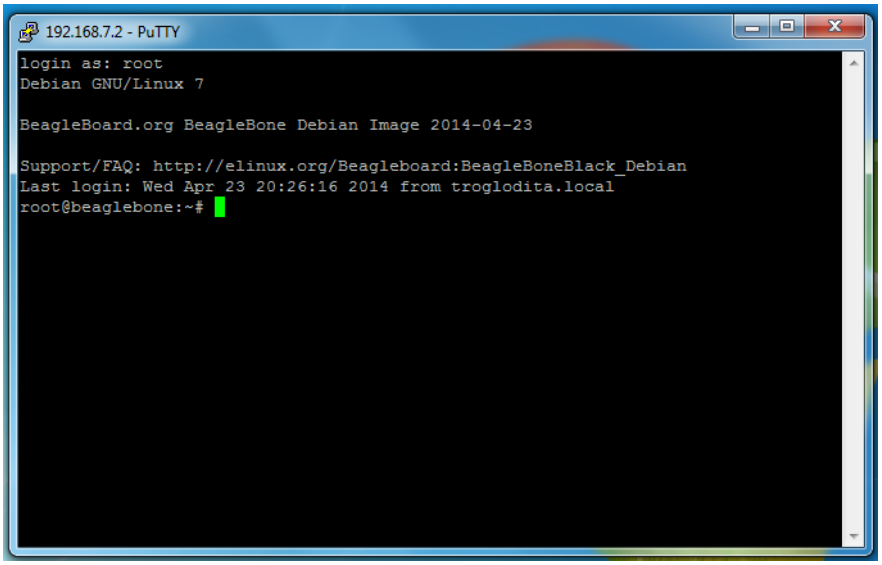


Imagen 51. Login root BBB

Fuente: Autor

Ahora nos dirigimos a la carpeta donde se encuentra almacenado el programa llamado *enviaDatos.py*. La ruta de la carpeta es */home/Debian/my_python*

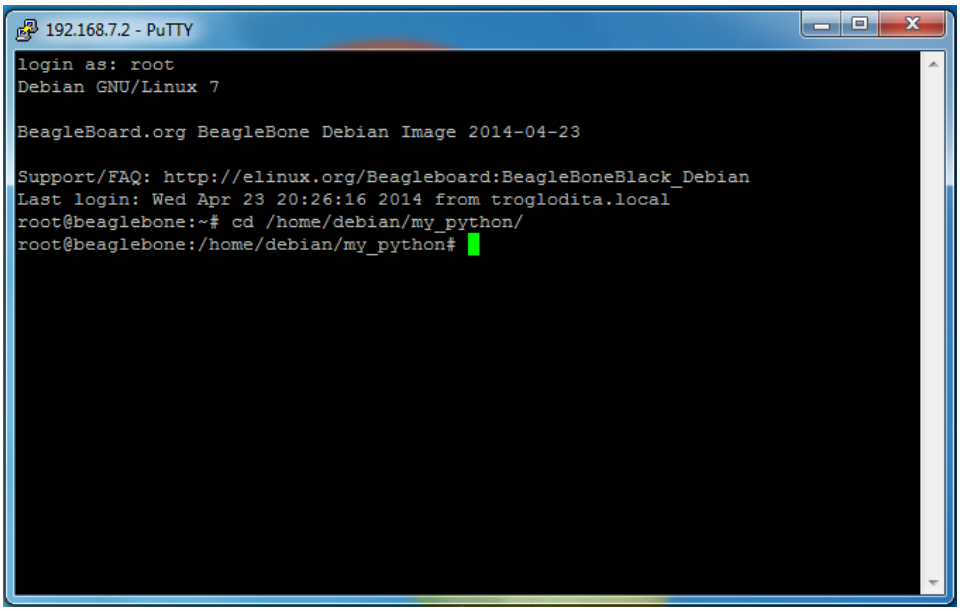


Imagen 52. Ruta de programas

Fuente: Autor

Una vez dentro de la carpeta *my_python* ejecutamos el programa llamado *enviaDatos.py* el cual está escrito en lenguaje de programación Python. Para ejecutar el programa escribimos *Python enviaDatos.py* en la terminal.

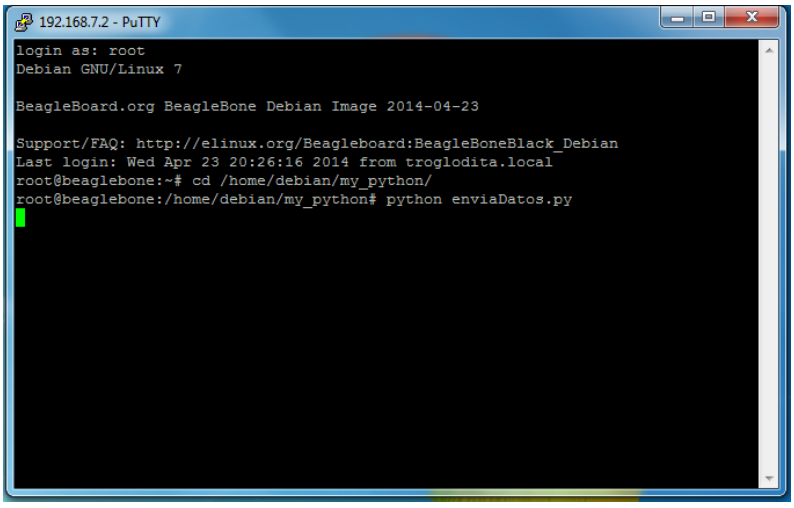


Imagen 53. Running de programa en la BBB

Fuente: Autor

Ahora el programa *enviaDatos.py* está ejecutándose, el programa estará esperando una señal de la PC para empezar a leer datos analógicos.

Por otro lado, en la PC, se encuentra ejecutando el programa *BiorretAOD*, procedemos a configurar el puerto, los canales, la escala y la ruta donde se guardará nuestra base de datos.

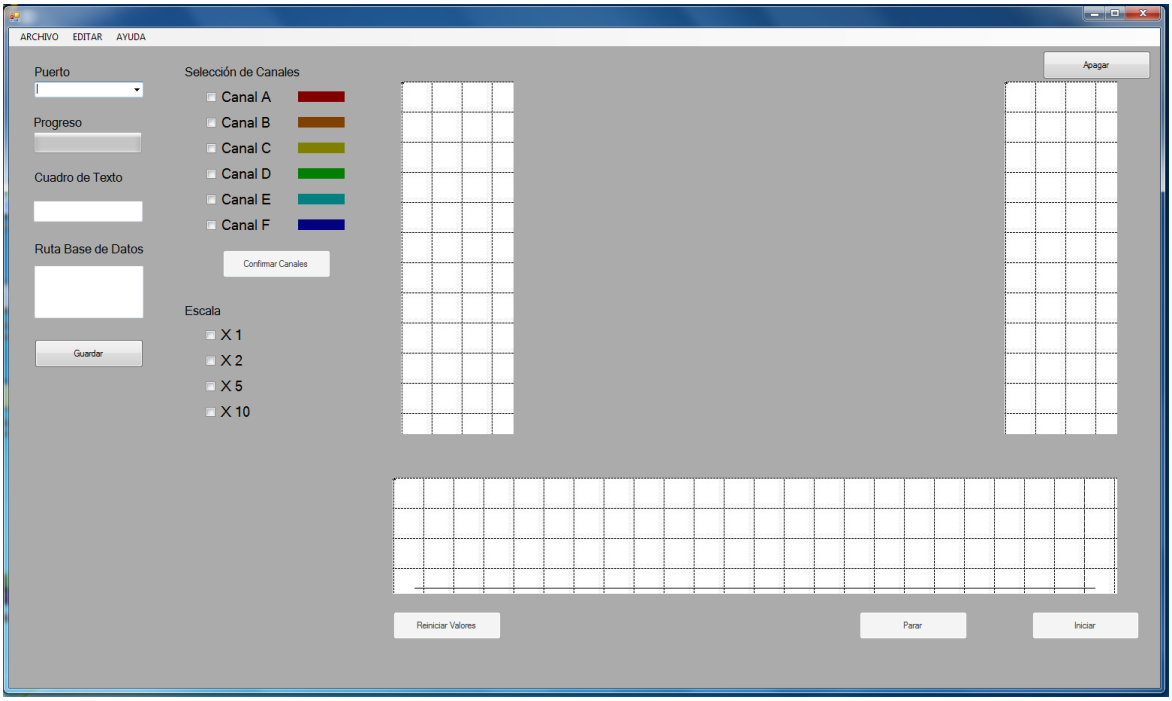


Imagen 54. Software de biofeedback

Fuente: Autor

Para seleccionar el puerto de comunicación, en *Puerto* desplegamos y nos aparecerá el puerto que tenemos conectado.

Si sólo tenemos el puerto UART y la BegleBone Black en el ordenador, regularmente serán identificados como COM3 y COM4. COM4 es el que corresponde al integrado FT232R.

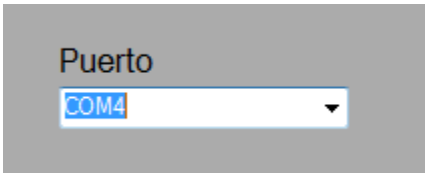


Imagen 55. Selección de puerto

Fuente: Autor

Para continuar, seleccionamos la ruta donde se guardará la base de datos, dando clic en *Guardar*. Esta acción nos desplegará un cuadro de archivos y seleccionamos donde queremos guardar nuestra base de datos.

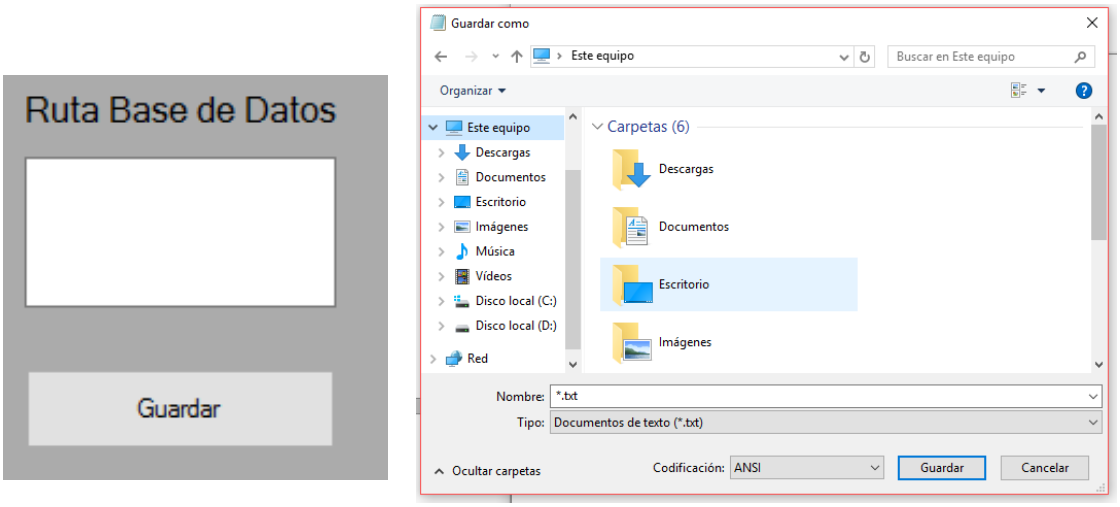


Imagen 56. Elección de ruta de la base de datos

Fuente:

Cuando tengamos una ruta de destino aparecerá el cuadro de texto con la ruta del archivo

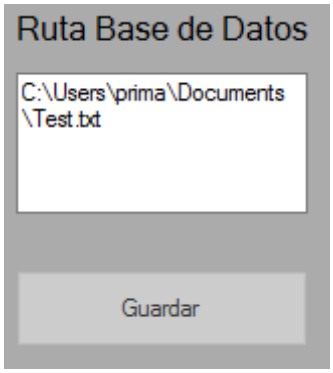


Imagen 57. Ruta donde se guardará la base de datos

Fuente: Autor

Continuamos seleccionando la escala de la señal, la cual puede variar entre 1, 2, 5 y 10 veces la magnitud de la señal.

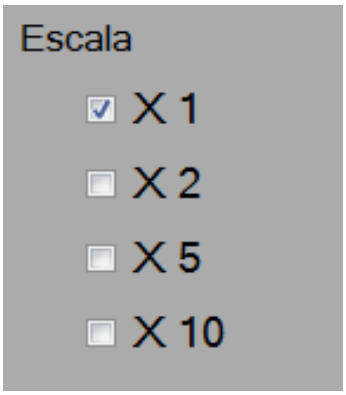


Imagen 58. Elección de escala

Fuente: Autor

Seleccionamos los canales que utilizaremos.

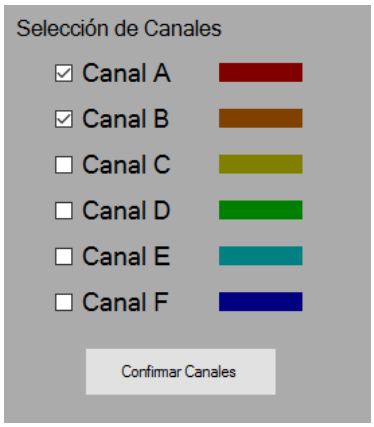


Imagen 59. Elección de canales

Fuente: Autor

Una vez seleccionados los canales que deseamos utilizar, damos clic en *Confirmar Canales*.

Nota: Aparecerá un mensaje que nos indica que podemos comenzar con el programa. Finalmente damos clic en *Iniciar*.

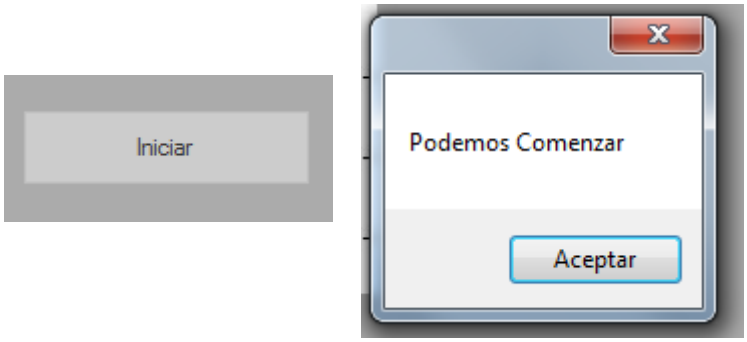


Imagen 60. Botón de inicio

Fuente: Autor

Existen tres botones adicionales que son *Parar*, *Reiniciar Valores* y *Apagar*.

Parar nos permite poner en pausa el programa.

Imagen 61. Botón de parar

Fuente: Autor

Estando en pausa el programa podemos realizar tres acciones:

- *Reanudar* el programa dando clic de nuevo en Iniciar.

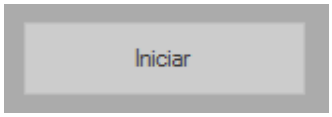


Imagen 62. Reanudar programa

Fuente: Autor

- *Reiniciar los Valores* para cambiar algún valor como es la Ruta de la Base de datos y los canales.

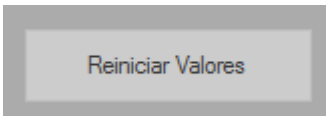


Imagen 63. Botón de reinicio de valores

Fuente: Autor

- *Apagar* la tarjeta BeagleBone Black de forma segura y así terminar el programa.

Imagen 64. Botón de apagado

Fuente: Autor

Lógica de BiorretAOD

Internamente *BiorretAOD* opera de la siguiente manera:

Una vez que nuestra señal analógica es convertida a digital, este dato va acompañado de una etiqueta con la inicial del canal que proviene.

Nombre del Canal	Etiqueta Identificadora
Canal A	U
Canal B	D
Canal C	T
Canal D	C
Canal E	I
Canal F	S

Tabla 2. Relación entre etiqueta y canal

BiorretAOD identifica la etiqueta y sabe a qué canal pertenece. Además, los datos deben de cumplir con la siguiente estructura:

[Etiqueta][Decimal 0 - 1][Punto .][Decimal 0 - 9][Decimal 0 - 9]

Si en algún momento el dato no cumple con esta estructura en automático es descartado y continúa leyendo el siguiente dato en fila.

Cada uno de estos datos es discriminado conforme a un rango de valores de 0 a 1.8 Volts que van:

Rango	Rango de Voltaje
Media Baja	0 - 0.6 Volts
Media	0.61 - 1.2 Volts
Media Alto	1.21 - 1.8 Volts

Tabla 3. Rango de voltaje

Para cada uno de los rangos, existe una animación equivalente, que se basa en los brazos estirados en diferentes ángulos de una persona.

Para un rango *Media Baja* la persona tendrá los brazos a 225^{o} y 315^{o}

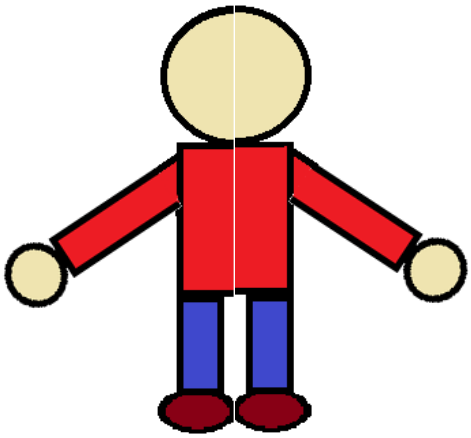


Imagen 65. Animación para un rango media baja

Fuente: Autor

Para un rango *Media* la persona tendrá los brazos a 0º y 180º

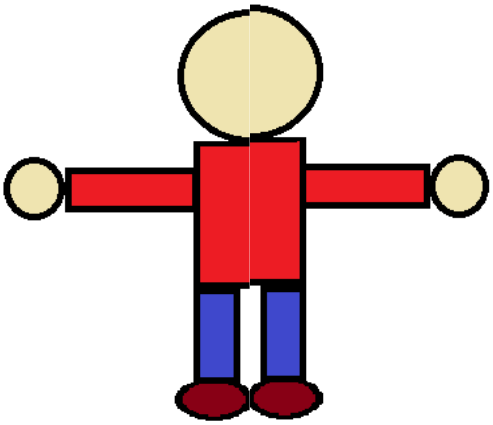


Imagen 66. Animación para un rango media

Fuente: Autor

Finalmente, si la persona está en un rango *Media Alta* tendrá los brazos a 45º y 135º

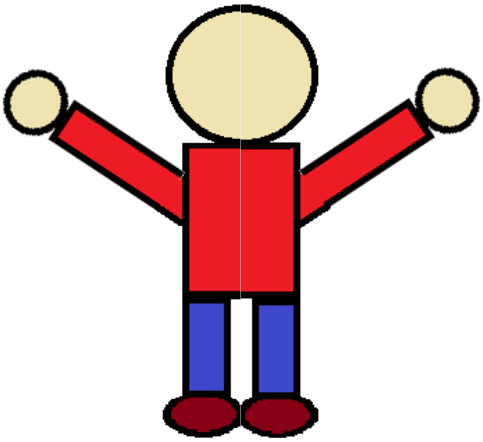


Imagen 67. Animación para un rango media alta

Fuente: Autor

Además, por cada dato recibido *BiorretAOD* incrementa el número de datos recibidos por canal, así hasta llegar a 100. Cada 100 datos se refresca la pantalla con la señal de dicho canal. Cabe señalar que el conteo de los datos es individual para cada canal y no un conjunto

de datos por canales activos. Como apoyo tenemos una gráfica de barras, la cual de igual manera nos indica como sube o baja el voltaje en cada canal.

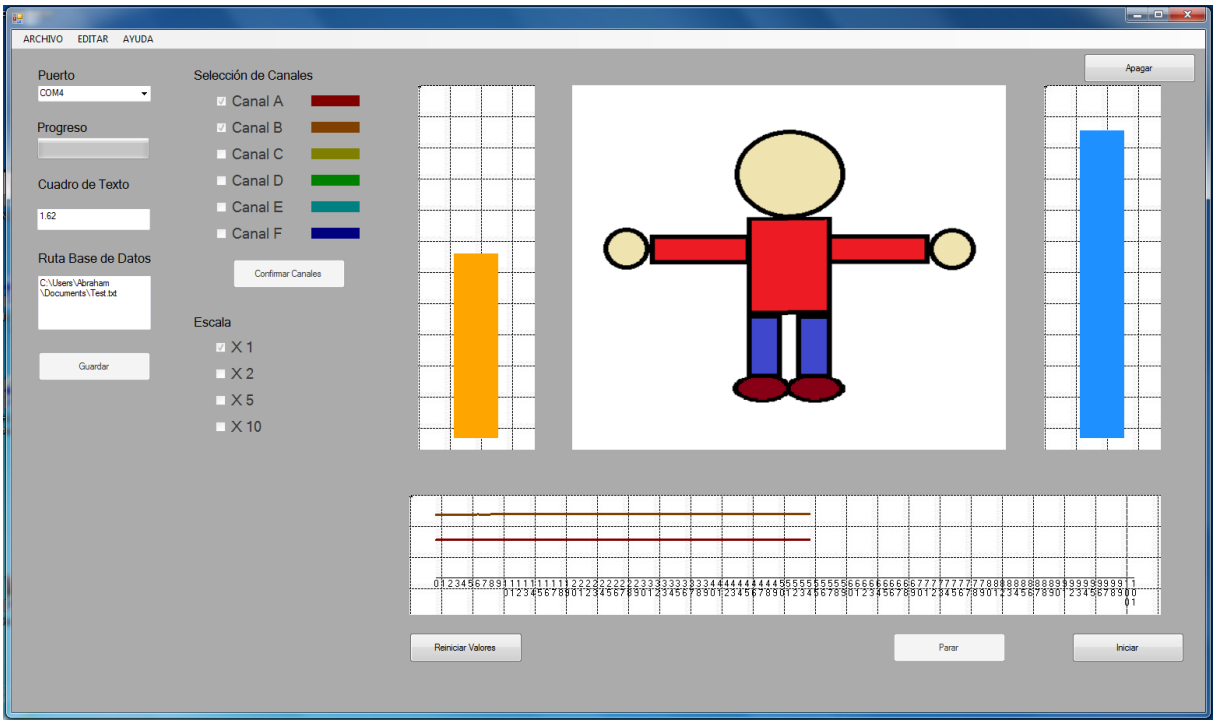


Imagen 68. Prueba del software de Biofeedback

Fuente: Autor

Alimentación del circuito

Para la alimentación del circuito se implementó una sencilla fuente lineal fija que tiene salidas de +12 Volts y -12 Volts. Dado que el consumo de los integrados es de muy poca corriente con un trasformador de 115 Volts a 24 Volts, y 1 Amper será más que suficiente para alimentar los circuitos.

El diseño de la fuente fue basado en el libro "*Electrónica: Teórica de Circuitos y Dispositivos Electrónicos* (Robert L. Bolestad y Louis Nashelsky)".

Este diseño aplica tanto para la línea de voltaje positivo como para la línea de voltaje negativo.

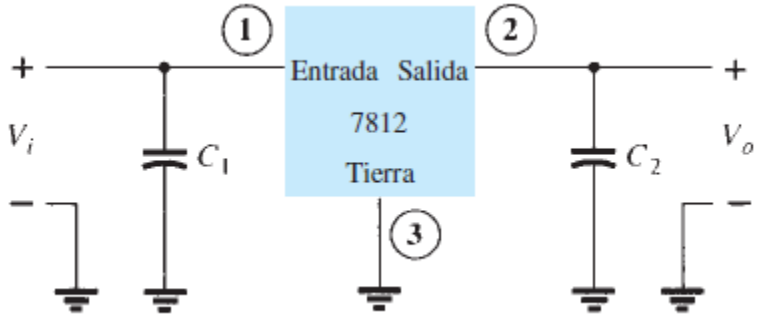


Imagen 69. Diseño fuente de alimentación

Fuente: Robert L. Bolestad y Louis Nashelsky

Los autores nos mencionan que C_1 filtra un voltaje de entrada no regulado V_i y este va conectado a la terminal IN del regulador. La terminal OUT del regulador proporciona un voltaje regulado filtrado por un capacitor C_2. Y, por último, la tercera terminal del regulador se conecta a tierra.

Para calcular el voltaje de rizo en el capacitor tenemos que:

$$V_r(rms) = \frac{2.4I_{cd}}{C}$$

Donde:

I_{cd} = 1 A = 1000 mA

C = 2200 µF

En este caso dado que se usará el mismo circuito rectificador para el voltaje positivo y negativo, el amperaje que es de 1 A, se divide en partes iguales, quedando 500 mA para el voltaje positivo y 500 mA para el voltaje negativo.

Finalmente tenemos un V_r de:

$$V_r(rms) = \frac{2.4 * 500}{2200} = 0.54\, Vrms$$

Además, tenemos que el rizo está definido como:

$$r = \frac{voltaje\ de\ rizo\ (rms)}{voltaje\ de\ cd} = \frac{V_r(rms)}{V_{cd}}\, x\ 100\%$$

$$r = \frac{0.54}{12\ V}\, x\ 100\% = 4.5\%$$

Adicionalmente se agregaron un capacitor de 100 nF para filtro de ruido proveniente de la señal no regulada de entrada y un capacitor de 220 µF en la señal regulada de salida, principalmente para ruido de alta frecuencia.

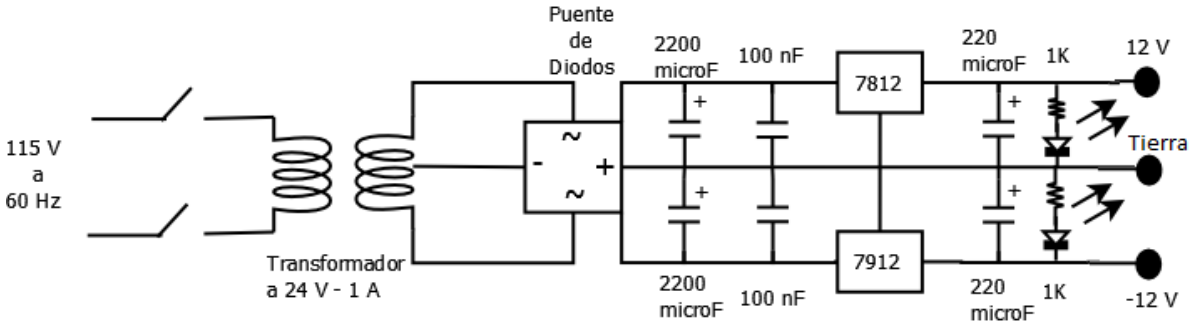


Imagen 70. Diseño de la fuente de alimentación

Fuente: Autor

Filtro paso bajas de 60 Hz

Dado que el ruido de 60 Hz proviene principalmente de la fuente de alimentación se propone la implementación de un filtro LC de primer orden.

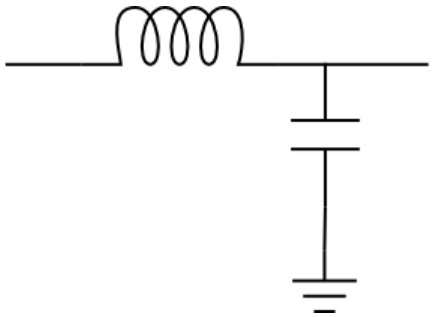


Imagen 71. Filtro LC

Fuente: Autor

Para el cálculo del inductor tenemos que

$$n = 6$$
$$a = 0.3\ cm$$
$$b = 0.9\ cm$$

Para el cálculo de un inductor tenemos que:

$$L(uH) = \frac{(0.393)(a)^2(n)^2}{9a + 10b}$$

Sustituyendo valores

$$L(uH) = \frac{(0.393)(0.3)^2(6)^2}{9(0.3) + 10(0.9)}$$

$$L(uH) = 108.8\ mL$$

Para un filtro paso Bajas LC, tenemos que la frecuencia de corte se calcula como:

$$fc = \frac{1}{(2\pi)\left(\sqrt{LC}\right)}$$

Despejando C tenemos que

$$C = \frac{1}{(2\pi)^2 (fc)^2 L}$$

Sustituyendo valores para una frecuencia de corte de 60 Hz tenemos que:

$$C = \frac{1}{(2\pi)^2 (60)^2 108.8mL}$$

$$C = 64.67\ \mu F\ \approx 68\ \mu F \rightarrow Valor\ comercial$$

Para un capacitor de 68 µF tenemos una frecuencia de corte de

$$fc = \frac{1}{(2\pi)\left(\sqrt{(108mL)(68\mu F)}\right)}$$

$$fc = 58.5\ Hz$$

Esta es una frecuencia de corte aceptable dado que lo que se busca es que no haya señales ruidosas con una frecuencia más allá de 60 Hz.

Quedando finalmente la fuente de alimentación de la siguiente manera:

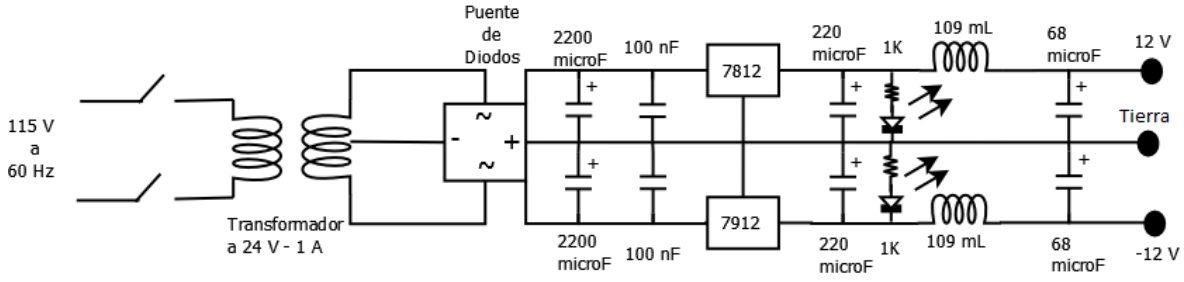


Imagen 72. Diseño de la fuente de alimentación con filtro de 60 Hz

Fuente: Autor

Diagramas y tarjeta final

A continuación, se muestra el diseño en PCB para el planchado y ensamblado del sistema.

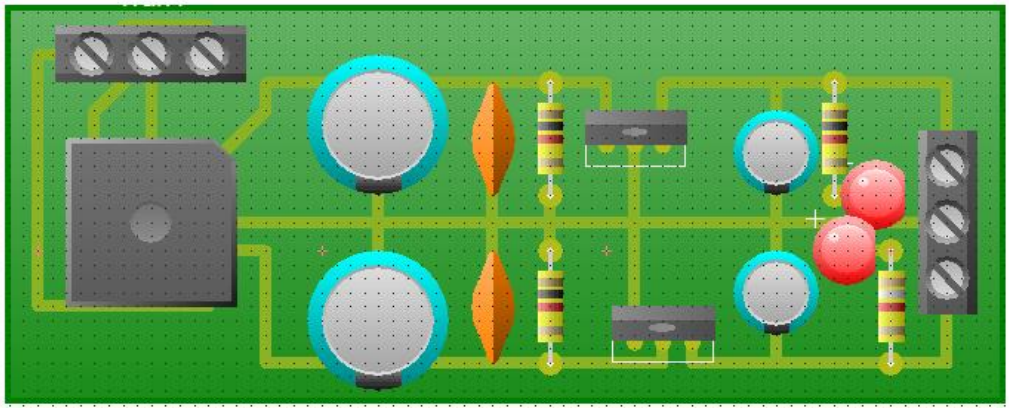


Imagen 73. Diseño de la tarjeta la fuente de alimentación en PCB con componentes

Fuente: Autor

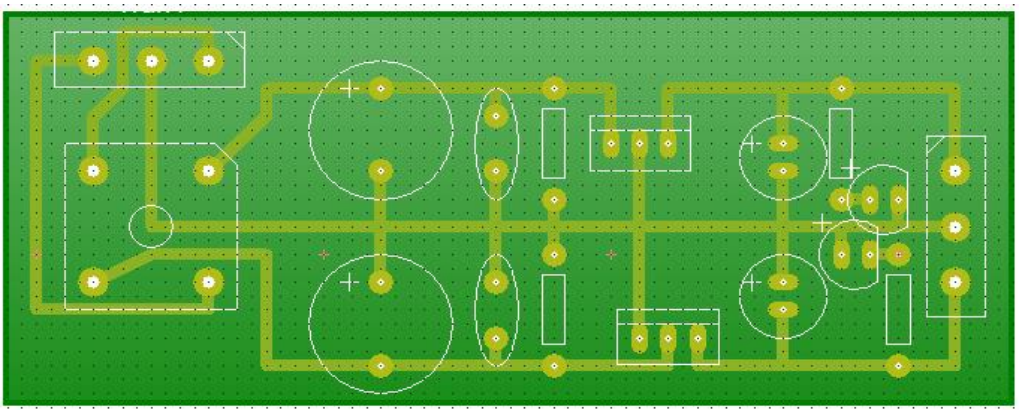


Imagen 74. Diseño de la tarjeta la fuente de alimentación en PCB sin componentes

Fuente: Autor

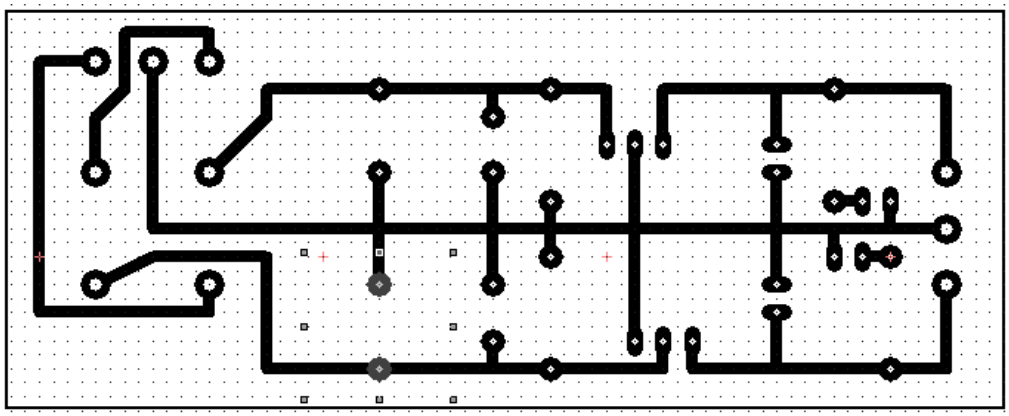


Imagen 75. Pistas la fuente de alimentación en PCB

Fuente: Autor

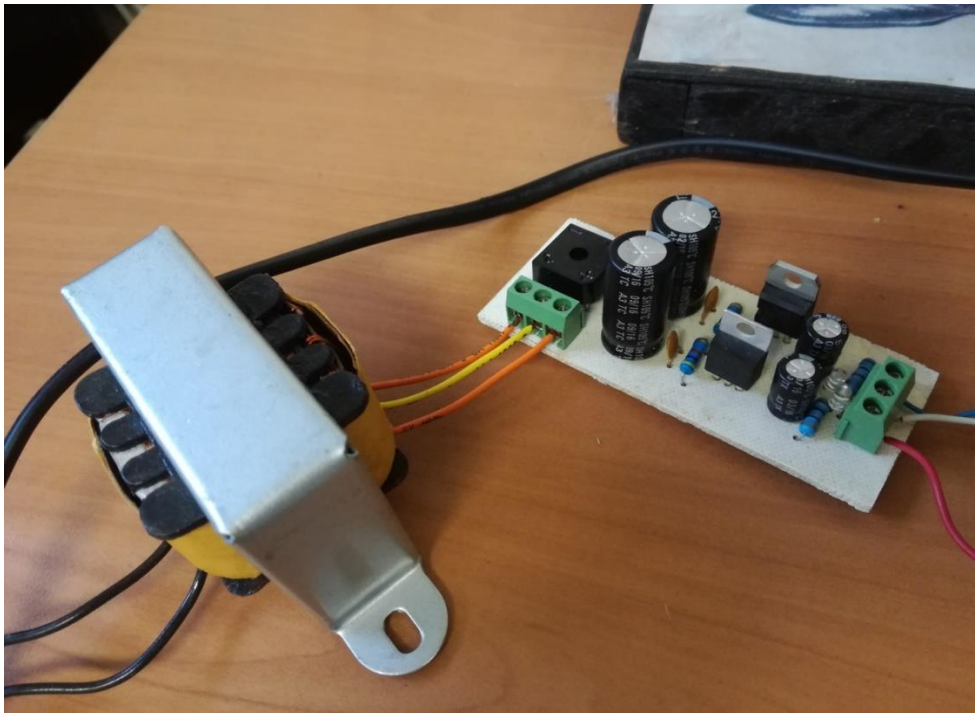


Imagen 76. Fuente de alimentación

Fuente: Autor

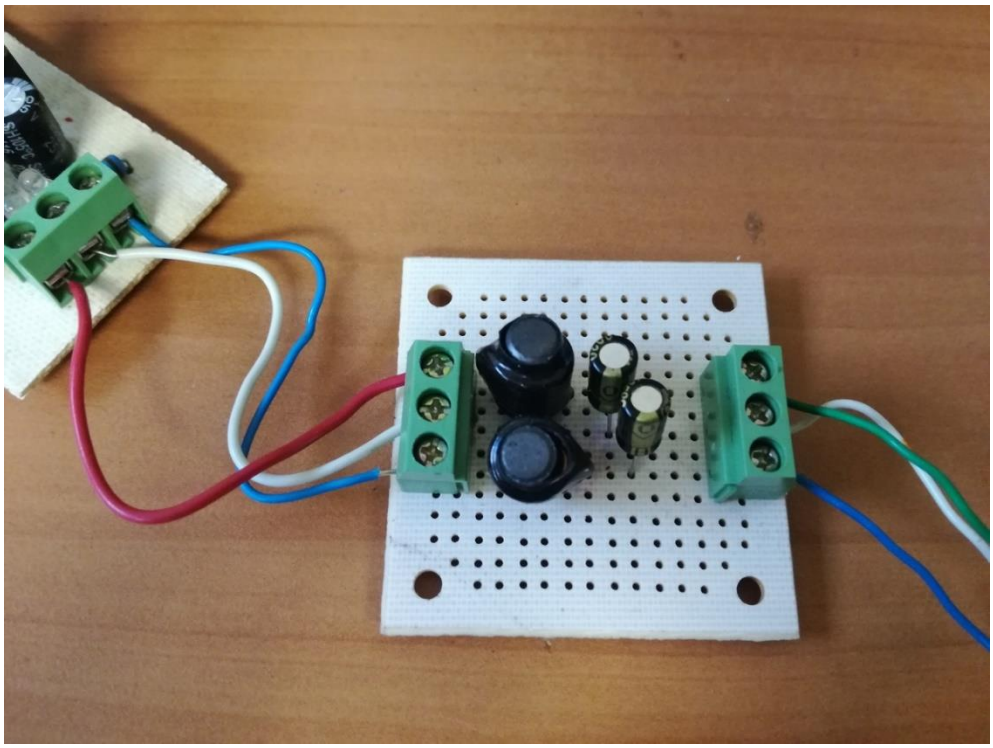


Imagen 77. Filtro de 60Hz LC

Fuente: Autor

Pruebas y resultados

Las pruebas de funcionamiento del sistema se hicieron a través de señales mioeléctricas del antebrazo. Esto para verificar la funcionalidad y seguridad de los filtros y de la fuente de alimentación.

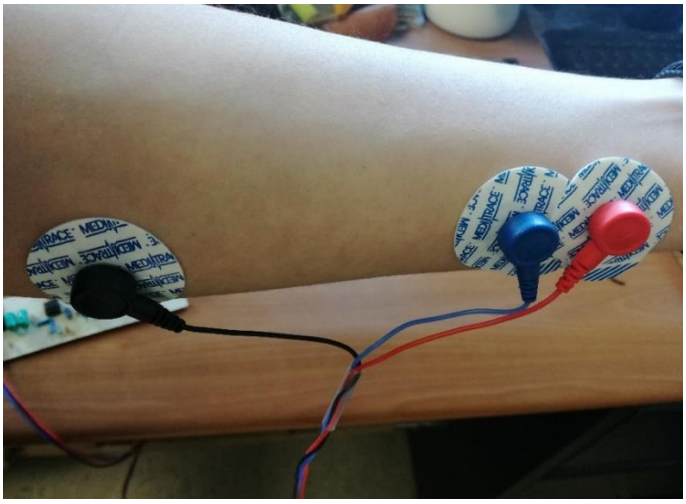


Imagen 78. Colocación de electrodos

Fuente: Autor

Esta es la señal mioeléctrica en la etapa del amplificador de instrumentación, como se puede observar la señal es muy pequeña y contiene demasiado ruido.

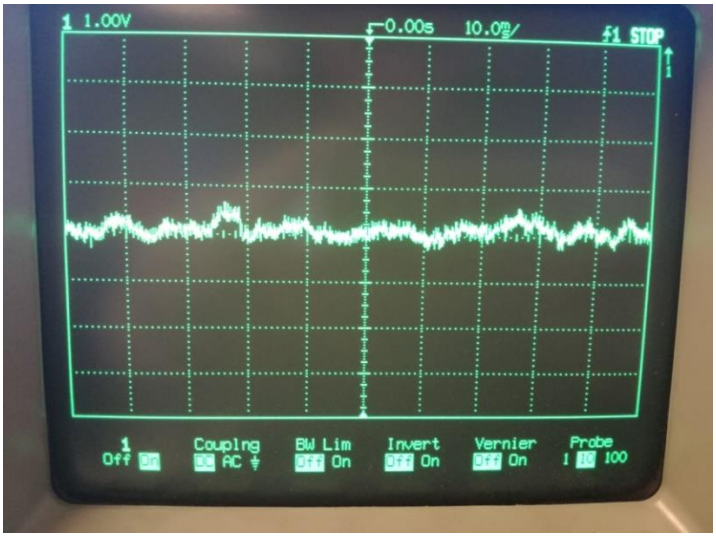


Imagen 79. Señal mioeléctrica a la salida del amplificador de instrumentación

Fuente: Autor

Posteriormente en la etapa de filtrado y ganancias, obtenemos una señal más limpia y con magnitud mucho mayor. Además, la señal fue rectificada, lo cual ya no presenta voltajes negativos.

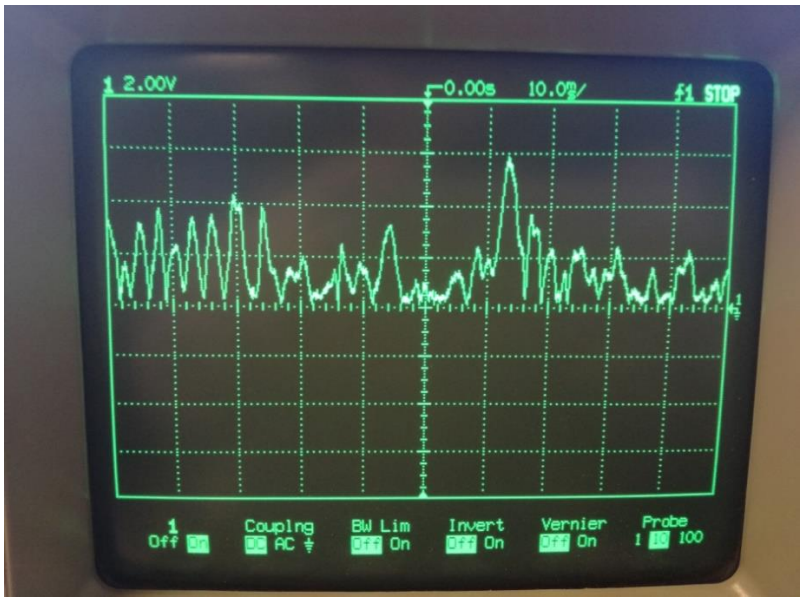


Imagen 80. Señal mioeléctrica, amplificada, filtrada y rectificada

Fuente: Autor

Finalmente, nuestra señal pasa por el divisor de voltaje, lo cual nos reduce el voltaje de 12 Volts a un máximo de 1.8 Volts.

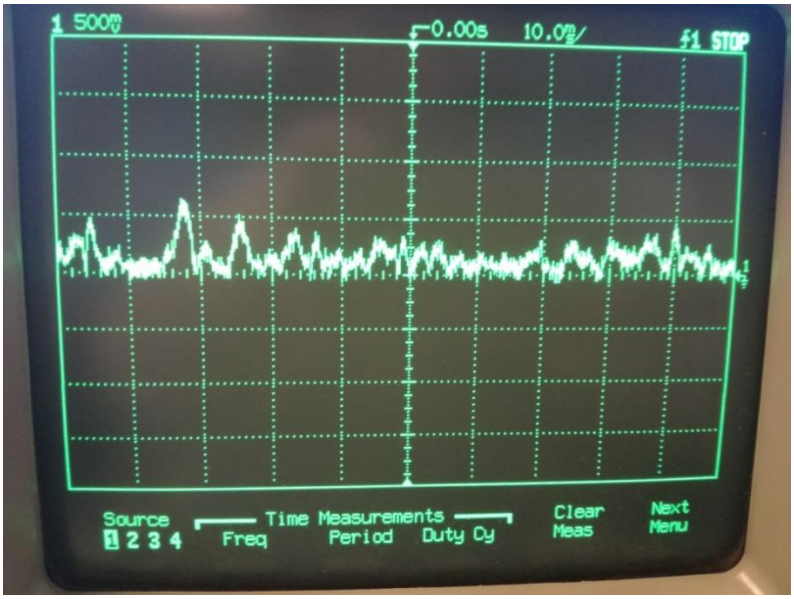


Imagen 81. Señal mioeléctrica a la entrada de la BBB

Fuente: Autor

Para verificar que el software está funcionando correctamente, se dejan los electrodos al aire y vemos que no registra nada.

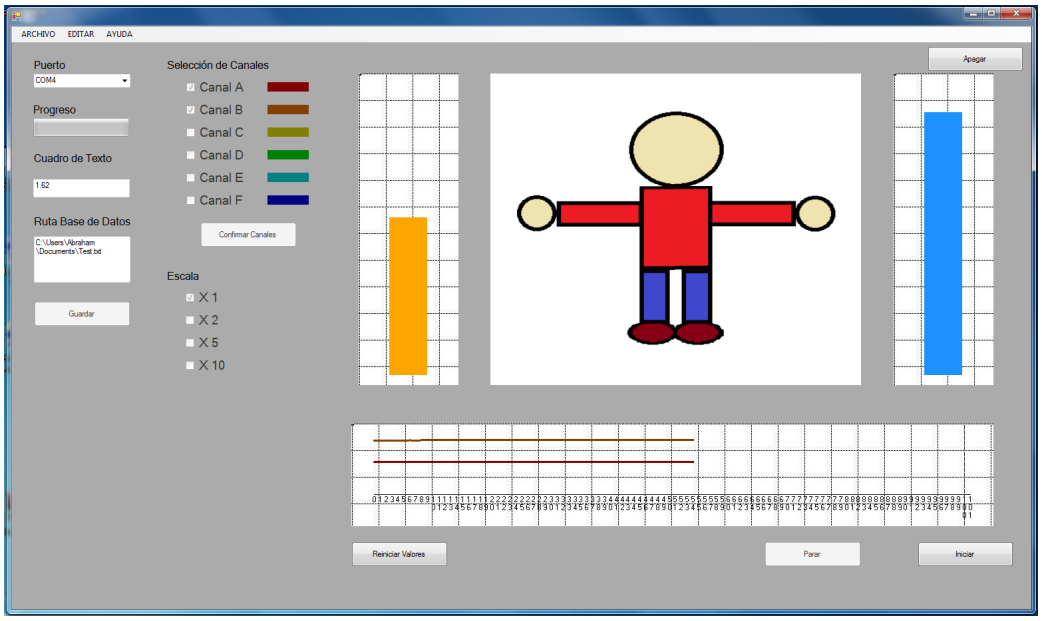


Imagen 82. Pruebas software de biofeedback con electrodos al aire

Fuente: Autor

Una vez conectados los electrodos en posición, empezamos a recibir lecturas, se registran y se grafican, dándonos una biorretroalimentación.

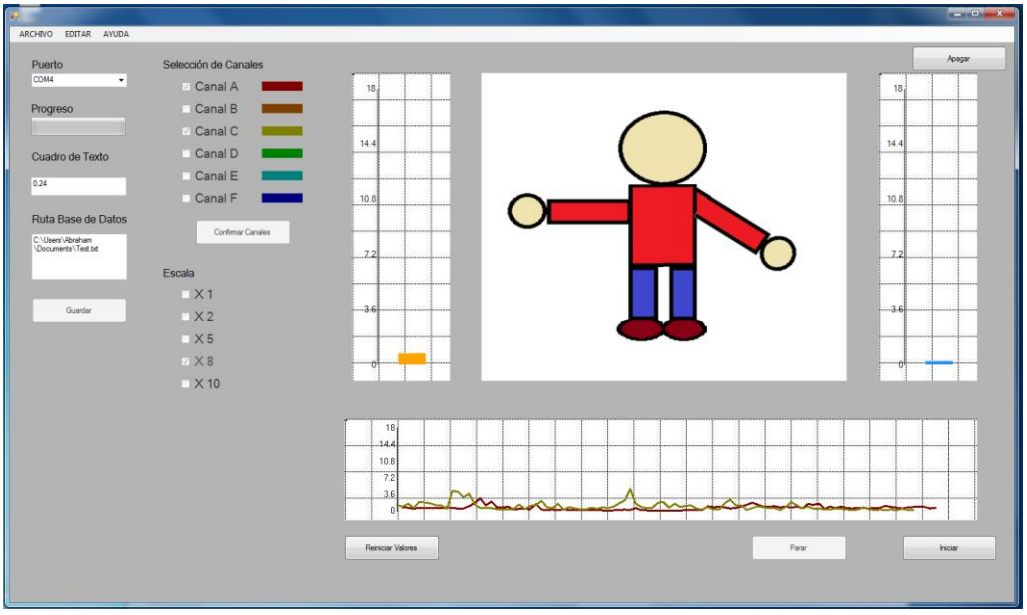


Imagen 83. Software de biofeedback con datos reales

Fuente: Autor

Conclusiones

De acuerdo a lo realizado puedo decir que los sistemas biomédicos, sean de la índole que sea, con objetivos diferentes y empleados en diferentes áreas en la medicina fueron, son y serán cada vez más importantes para la humanidad. Ya que conforme se han descubriendo nuevas formas de implementación y tecnologías, estos son cada vez más sofisticados y complejos.

Puede que su precio sea una desventaja para la gente de escasos recursos, ya sea que vivan en un país denominado de primer o tercer mundo. Al ser el costo muy elevado, las personas que tienen la posibilidad de recuperarse simplemente adquiriendo un sistema biomédico o terapias que se apoyen de estos sistemas, no lo pueden hacer por no tener los recursos para pagarlos.

Es comprensible que estos sistemas sean caros, ya que el tiempo de investigación no es tan corto y puede incluso alargarse por varios años. Pero, existe la posibilidad de replicar estos sistemas de una forma económica y sencilla.

Sin embargo, a pesar de saber esto cumplí con mis objetivos establecidos en un principio, el sistema biomédico enfocado para terapias de rehabilitación es funcional. Este contiene pocos componentes eléctricos y es de un ensamblaje sencillo pero con una calidad y costo aceptable.

Se encuentra compuesto por varias herramientas, las cuales a lo largo del desarrollo fueron de gran ayuda, me refiero a la tarjeta BeagleBone Black y el uso de una computadora.

Al realizar el sistema mediante secciones o bloques, facilita la sustitución de los componentes que lo conforman, ya que en caso de que lleguen a existir fallas en el sistema estas serán más fáciles de detectar y sobre todo de resolver y mejorar. Sin embargo, existen varias modificaciones para un mejor rendimiento.

Una de ellas es buscar otro lenguaje de programación para el software *BiorretAOD*, ya que Visual C++ depende mucho del hardware de cómputo que se tenga, y uno de nuestros objetivos es tener software que no dependa del poder de cómputo.

Otra manera en que se podría mejorar *BiorretAOD* es realizando plantillas (temas), las cuales podrían intercambiarse y ser más ameno para el paciente.

A lo largo de este trabajo me encontré con varias dificultades, la más grande fue la del desarrollo del software dedicado *BiorretAOD*. El lenguaje de programación con el cual se pensaba realizar en un principio se tuvo que cambiar una vez, ya que no nos daba el rendimiento apropiado; estuvo a punto de cambiarse por segunda ocasión, pero esto ya no se llevó a cabo debido a que ya se llevaba un gran avance en el tema y se contaba con poco tiempo para poder realizarlo nuevamente.

Bibliografía

BANTA, H.D.& B.R. Luce, Health Care Techonology and its Assessment Cambridge University Press Publications Oxford Medical Serial Communication

BRISEBIOS, W. The Biomed's Handbook Ontario 1994

COUGHLIN Robert F. y Frederick F. Driscoll, Amplificadores Operacionales y Circuitos Integrados Lineales 4a. Edición Naucalpan, México Prentice Hall, 1993

D. BRONZINO Joseph, The Biomedical Engineering Handbook CRC Press, IEEE Press, 1997

DAILEY, Denton J., Operacional Amplifiers and Linear Integrated Circuits Theory and Applications Singapur Mc. Graw Hill,1989

FOLEY, James D.; DAM VAN, Andries; FEINER, Steven K.; HUGHES, John F, Computer Graphics: Principles and Practice in C (2nd Edition) 2a. Edición USA Portland Addison-Wesley Pub Co, 1995

G, E. Tobey, J. G, Graeme, and L. P. Huelsman, Operational Amplifiers: Design and Applications, McGraw-Hill Book Company, New York, 1971.

G. WEBSTER John, Houghton Mifflin, Medical Instrumentation Application and Design 1996

HEARN, Donald; BAKER, M. Pauline Computer Graphics, C Version 2a. Edición Upper Saddle River, N.J. Prentice Hall, 1997

LANGLEY, Leroy Lester, Anatomía y Fisiología México Interamericana, 1979

MARK S. Schwartz and Frank Andrasik, Biofeedback Fourth Edition A Practitioner's Guide 3a Edición Guildford Press, 2003

R. ATLES Leslie, BA, Cbet, Scott Segalewitz Reference, Guide for Biomedical Technicians Kendall/Hunt Publishing Company

ROBERT L. Boylest, Louis Nashelsku, Electrónica: Teoría de Circuitos y Dispositivos Electrónicos 10ª. Edición México Pearson Educación, 2009

ROSEN, A.&Rosen, H.D., New Frontiers in Medical Device Technology New York Wiley Interscience

SCHOTTELIUS, Byron A., Textbook of Physiology Saint Louis C.V. Mosby, 1978

STANLEY William D., Operacional Amplifiers with Linear Integrated Circuits 3a. Edición New Jersey Prentice Hall, 1994

TORTORA, Gerard J., Principios de Anatomía y Fisiología México Harla, 1977

VANDER, Arthur J., Human Physiology: The Mechanims of Body Function New York McGraw-Hill, 1970

WAIT John V., Huelsman Lawrence P. and Korn Granino A., Introduction to Operational Amplifiers Theory and Applications 2ª. Edición New York Mc. Graw Hill,1992

WEBSTER G., John, Encyclopedia of Medical Devices and Instrumentation Wiley, 1998

Referencias

Microsoft. (1 de Enero de 2016). *Microsoft Developer Network*. Recuperado el 17 de Octubre de 2016, de Microsoft Developer Network: https://msdn.microsoft.com/es-es

ANEXOS

BURR • BROWN®

INA114

Precision
INSTRUMENTATION AMPLIFIER

FEATURES

- LOW OFFSET VOLTAGE: 50µV max
- LOW DRIFT: 0.25µV/°C max
- LOW INPUT BIAS CURRENT: 2nA max
- HIGH COMMON-MODE REJECTION: 115dB min
- INPUT OVER-VOLTAGE PROTECTION: ±40V
- WIDE SUPPLY RANGE: ±2.25 to ±18V
- LOW QUIESCENT CURRENT: 3mA max
- 8-PIN PLASTIC AND SOL-16

APPLICATIONS

- BRIDGE AMPLIFIER
- THERMOCOUPLE AMPLIFIER
- RTD SENSOR AMPLIFIER
- MEDICAL INSTRUMENTATION
- DATA ACQUISITION

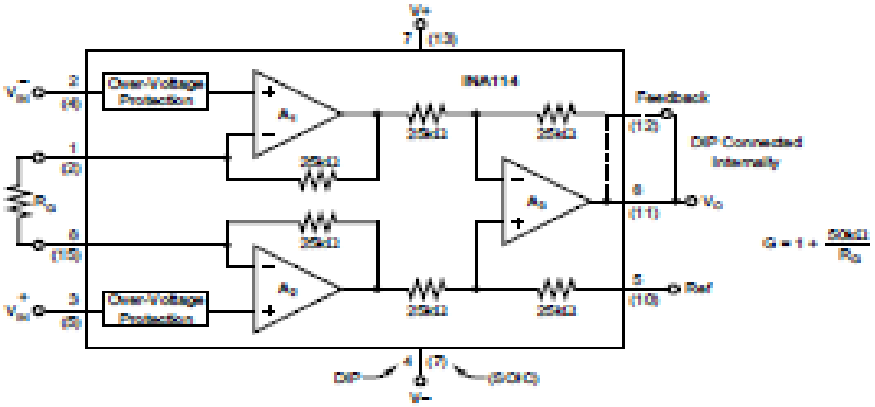
DESCRIPTION

The INA114 is a low cost, general purpose instrumentation amplifier offering excellent accuracy. Its versatile 3-op amp design and small size make it ideal for a wide range of applications.

A single external resistor sets any gain from 1 to 10,000. Internal input protection can withstand up to ±40V without damage.

The INA114 is laser trimmed for very low offset voltage (50µV), drift (0.25µV/°C) and high common-mode rejection (115dB at G = 1000). It operates with power supplies as low as ±2.25V, allowing use in battery operated and single 5V supply systems. Quiescent current is 3mA maximum.

The INA114 is available in 8-pin plastic and SOL-16 surface-mount packages. Both are specified for the -40°C to +85°C temperature range.



International Airport Industrial Park • Mailing Address: PO Box 11400, Tucson, AZ 85734 • Street Address: 6730 S. Tucson Blvd., Tucson, AZ 85706 • Tel: (520) 746-1111 • Twx: 910-952-1111
Internet: http://www.burr-brown.com/ • FAXLine: (800) 548-6133 (USA/Canada Only) • Cable: BBRCORP • Telex: 066-6491 • FAX: (520) 889-1510 • Immediate Product Info: (800) 548-6132

PDS-1142D

Printed in U.S.A. March, 1993

FTDI Chip

FT232R USB UART IC Datasheet
Version 2.13

Document No.: FT 000053 Clearance No.: FTDI# 38

Future Technology Devices International Ltd.

FT232R USB UART IC

The FT232R is a USB to serial UART interface with the following advanced features:

- Single chip USB to asynchronous serial data transfer interface.
- Entire USB protocol handled on the chip. No USB specific firmware programming required.
- Fully integrated 1024 bit EEPROM storing device descriptors and CBUS I/O configuration.
- Fully integrated USB termination resistors.
- Fully integrated clock generation with no external crystal required plus optional clock output selection enabling a glue-less interface to external MCU or FPGA.
- Data transfer rates from 300 baud to 3 Mbaud (RS422, RS485, RS232) at TTL levels.
- 128 byte receive buffer and 256 byte transmit buffer utilising buffer smoothing technology to allow for high data throughput.
- FTDI's royalty-free Virtual Com Port (VCP) and Direct (D2XX) drivers eliminate the requirement for USB driver development in most cases.
- Unique USB FTDIChip-ID™ feature.
- Configurable CBUS I/O pins.
- Transmit and receive LED drive signals.
- UART interface support for 7 or 8 data bits, 1 or 2 stop bits and odd / even / mark / space / no parity
- FIFO receives and transmits buffers for high data throughput.
- Synchronous and asynchronous bit bang interface options with RD# and WR# strobes.
- Device supplied pre-programmed with unique USB serial number.
- Supports bus powered, self-powered and high-power bus powered USB configurations.
- Integrated +3.3V level converter for USB I/O.
- Integrated level converter on UART and CBUS for interfacing to between +1.8V and +5V logic.
- True 5V/3.3V/2.8V/1.8V CMOS drive output and TTL input.
- Configurable I/O pin output drive strength.
- Integrated power-on-reset circuit.
- Fully integrated AVCC supply filtering - no external filtering required.
- UART signal inversion option.
- +3.3V (using external oscillator) to +5.25V (internal oscillator) Single Supply Operation.
- Low operating and USB suspend current.
- Low USB bandwidth consumption.
- UHCI/OHCI/EHCI host controller compatible.
- USB 2.0 Full Speed compatible.
- -40°C to 85°C extended operating temperature range.
- Available in compact Pb-free 28 Pin SSOP and QFN-32 packages (both RoHS compliant).

TEXAS INSTRUMENTS

TL081, TL081A, TL081B, TL082, TL082A
TL082B, TL084, TL084A, TL084B

SLOS081I – FEBRUARY 1977 – REVISED MAY 2015

TL08xx JFET-Input Operational Amplifiers

1 Features

- Low Power Consumption: 1.4 mA/ch Typical
- Wide Common-Mode and Differential Voltage Ranges
- Low Input Bias Current: 30 pA Typical
- Low Input Offset Current: 5 pA Typical
- Output Short-Circuit Protection
- Low Total Harmonic Distortion: 0.003% Typical
- High Input Impedance: JFET Input Stage
- Latch-Up-Free Operation
- High Slew Rate: 13 V/μs Typical
- Common-Mode Input Voltage Range Includes V_{CC+}

2 Applications

- Tablets
- White goods
- Personal electronics
- Computers

3 Description

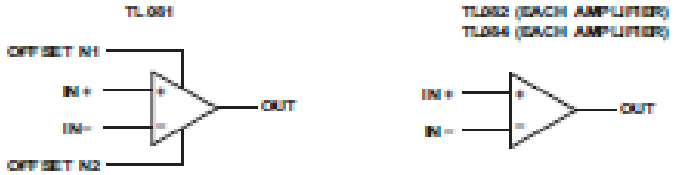
The TL08xx JFET-input operational amplifier family is designed to offer a wider selection than any previously developed operational amplifier family. Each of these JFET-input operational amplifiers incorporates well-matched, high-voltage JFET and bipolar transistors in a monolithic integrated circuit. The devices feature high slew rates, low input bias and offset currents, and low offset-voltage temperature coefficient.

Device Information(1)

PART NUMBER	PACKAGE	BODY SIZE (NOM)
TL084xD	SOIC (14)	8.65 mm × 3.91 mm
TL08xxFK	LCCC (20)	8.89 mm × 8.89 mm
TL084xJ	CDIP (14)	19.56 mm × 6.92 mm
TL084xN	PDIP (14)	19.3 mm × 6.35 mm
TL084xNS	SO (14)	10.3 mm × 5.3 mm
TL084xPW	TSSOP (14)	5.0 mm × 4.4 mm

(1) For all available packages, see the orderable addendum at the end of the data sheet.

Schematic Symbol



Programa enviaDatos.py

```
#!/usr/bin/python
import Adafruit_BBIO.ADC as ADC
import Adafruit_BBIO.UART as UART
import serial
import time

ADC.setup()
UART.setup("UART1")

ser = serial.Serial(port = "/dev/ttyO1", baudrate = 115200, timeout = None)
ser.close()
ser.open()
#####Inicio#####
try:
	while True:
		###Config Inicial###
		activaCH_1 = False
		activaCH_2 = False
		activaCH_3 = False
		activaCH_4 = False
		activaCH_5 = False
		activaCH_6 = False
		canalesActivos = 0
		numCanales = ser.read()

		if numCanales == 'A':
			ser.close()
			print "Puerto Cerrado"
			exit()
			UART.cleanup()
		elif numCanales == 'R':
			numCanales = ser.read()
		elif numCanales == 'I':
			  numCanales = ser.read()

		numCanales = int(numCanales)
		##### CH1 ########
		muestras_CH1 = 0
		numMuestras_CH1 = 0
		sumaRms_CH1 = 0.0
		divideRms_CH1 = 0.0
		##### CH2 ########
		muestras_CH2 = 0
		numMuestras_CH2 = 0
		sumaRms_CH2 = 0.0
		divideRms_CH2 = 0.0
		##### CH3 ########
		muestras_CH3 = 0
		numMuestras_CH3 = 0
		sumaRms_CH3 = 0.0
		divideRms_CH3 = 0.0
		##### CH4 ########
		muestras_CH4 = 0
		numMuestras_CH4 = 0
		sumaRms_CH4 = 0.0
		divideRms_CH4 = 0.0
		##### CH5 ########
		muestras_CH5 = 0
		numMuestras_CH5 = 0
```

```
sumaRms_CH5 = 0.0
divideRms_CH5 = 0.0
##### CH6 ########
muestras_CH6 = 0
numMuestras_CH6 = 0
sumaRms_CH6 = 0.0
divideRms_CH6 = 0.0
######-----#######
while numCanales != canalesActivos:
	activaCanal = ser.read()
	if activaCanal == 'A':
		activaCH_1 = True
		canalesActivos = canalesActivos + 1
	elif activaCanal == 'B':
		activaCH_2 = True
		canalesActivos = canalesActivos + 1
	elif activaCanal == 'C':
		activaCH_3 = True
		canalesActivos = canalesActivos + 1
	elif activaCanal == 'D':
		activaCH_4 = True
		canalesActivos = canalesActivos + 1
	elif activaCanal == 'E':
		activaCH_5 = True
		canalesActivos = canalesActivos + 1
	elif activaCanal == 'F':
		activaCH_6 = True
		canalesActivos = canalesActivos + 1

Inicia = ser.read()
Pausa = 'I'
timer = 0
tiempo = 0

 while Inicia == 'I':
	while Pausa == 'P':
		Condicion = ser.read()
		if Condicion == 'A':
			ser.close()
			    print "Puerto Cerrado"
			    exit()
			    UART.cleanup()
		   elif Condicion == 'R':
			    Inicia = 'R'
			Pausa = 'I'
		elif Condicion == 'I':
			Pausa = 'I'

	if (int(time.time()*100000)%1300) == 0:
		timer = timer + 0.1
		if timer >= 0.9:
			if tiempo <= 100:
				tiempo_char = str(tiempo)
				print tiempo_char
				tiempo = tiempo + 1
			else:
				tiempo = 0
				tiempo_char = str(tiempo)
				print tiempo_char
				    tiempo = tiempo + 1
```

```
                                timer = 0

                        if activaCH_1:
                                canal1 =  float(ADC.read("P9_39"))
                                voltaje_canal1 = canal1 ** 2 #Elevamos al cuadrado
para sacar voltaje RMS
                                sumaRms_CH1 = sumaRms_CH1 + voltaje_canal1 #Sumamos
los valores
                                numMuestras_CH1 = numMuestras_CH1 + 1
                                if (numMuestras_CH1 == 10):
                                        divideRms_CH1 = sumaRms_CH1 / numMuestras_CH1
                                        rmsVoltaje_CH1 = divideRms_CH1 ** 0.5
#Sacamos raiz cuadrada
                                        rmsVoltaje_CH1 = format(rmsVoltaje_CH1,
'.2f')
                                        canal1_char = str(rmsVoltaje_CH1)
                                        numMuestras_CH1 = 0
                                        sumaRms_CH1 = 0.0
                                        divideRms_CH1 = 0.0
                                        print "U" + canal1_char
                                        ser.write('U')
                                        ser.write(canal1_char)
                                        ser.write("\n")
                        if activaCH_2:
                                canal2 =  float(ADC.read("P9_40"))
                                voltaje_canal2 = canal2 ** 2 #Elevamos al cuadrado
para sacar voltaje RMS
                                sumaRms_CH2 = sumaRms_CH2 + voltaje_canal2 #Sumamos
los valores
                                numMuestras_CH2 = numMuestras_CH2 + 1
                                        if (numMuestras_CH2 == 10):
                                        divideRms_CH2 = sumaRms_CH2 / numMuestras_CH2
                                        rmsVoltaje_CH2 = divideRms_CH2 ** 0.5
#Sacamos raiz cuadrada
                                        rmsVoltaje_CH2 = format(rmsVoltaje_CH2,
'.2f')
                                        canal2_char = str(rmsVoltaje_CH2)
                                        numMuestras_CH2 = 0
                                        sumaRms_CH2 = 0.0
                                        divideRms_CH2 = 0.0
                                        print "D" + canal2_char
                                        ser.write('D')
                                        ser.write(canal2_char)
                                        ser.write("\n")
                        if activaCH_3:
                                canal3 =  float(ADC.read("P9_37"))
                                voltaje_canal3 = canal3 ** 2 #Elevamos al cuadrado
para sacar voltaje RMS
                                sumaRms_CH3 = sumaRms_CH3 + voltaje_canal3 #Sumamos
los valores
                                numMuestras_CH3 = numMuestras_CH3 + 1
                                if (numMuestras_CH3 == 10):
                                        divideRms_CH3 = sumaRms_CH3 / numMuestras_CH3
                                        rmsVoltaje_CH3 = divideRms_CH3 ** 0.5
#Sacamos raiz cuadrada
                                        rmsVoltaje_CH3 = format(rmsVoltaje_CH3,
'.2f')
                                        canal3_char = str(rmsVoltaje_CH3)
                                        numMuestras_CH3 = 0
                                        sumaRms_CH3 = 0.0
                                        divideRms_CH3 = 0.0
```

```
                                        print "T"+canal3_char
                                        ser.write('T')
                                        ser.write(canal3_char)
                                        ser.write("\n")
                                if activaCH_4:
                                        canal4 =  float(ADC.read("P9_38"))
                                        voltaje_canal4 = canal4 ** 2 #Elevamos al cuadrado
para sacar voltaje RMS
                                        sumaRms_CH4 = sumaRms_CH4 + voltaje_canal4 #Sumamos
los valores
                                        numMuestras_CH4 = numMuestras_CH4 + 1
                                                if (numMuestras_CH4 == 10):
                                                        divideRms_CH4 = sumaRms_CH4 /
numMuestras_CH4
                                                rmsVoltaje_CH4 = divideRms_CH4 ** 0.5
#Sacamos raiz cuadrada
                                                rmsVoltaje_CH4 = format(rmsVoltaje_CH4,
'.2f')
                                                canal4_char = str(rmsVoltaje_CH4)
                                                numMuestras_CH4 = 0
                                                sumaRms_CH4 = 0.0
                                                divideRms_CH4 = 0.0
                                                print "C"+canal4_char
                                                ser.write('C')
                                                ser.write(canal4_char)
                                                ser.write("\n")
                                if activaCH_5:
                                        canal5 =  float(ADC.read("P9_33"))
                                        voltaje_canal5 = canal5 ** 2 #Elevamos al cuadrado
para sacar voltaje RMS
                                        sumaRms_CH5 = sumaRms_CH5 + voltaje_canal5 #Sumamos
los valores
                                        numMuestras_CH5 = numMuestras_CH5 + 1
                                        if (numMuestras_CH5 == 10):
                                                divideRms_CH5 = sumaRms_CH5 / numMuestras_CH5
                                                rmsVoltaje_CH5 = divideRms_CH5 ** 0.5
#Sacamos raiz cuadrada
                                                rmsVoltaje_CH5 = format(rmsVoltaje_CH5,
'.2f')
                                                canal5_char = str(rmsVoltaje_CH5)
                                                numMuestras_CH5 = 0
                                                sumaRms_CH5 = 0.0
                                                divideRms_CH5 = 0.0
                                                print "I"+canal5_char
                                                ser.write('I')
                                                ser.write(canal5_char)
                                                ser.write("\n")
                                if activaCH_6:
                                        canal6 =  float(ADC.read("P9_36"))
                                        voltaje_canal6 = canal6 ** 2 #Elevamos al cuadrado
para sacar voltaje RMS
                                        sumaRms_CH6 = sumaRms_CH6 + voltaje_canal6 #Sumamos
los valores
                                        numMuestras_CH6 = numMuestras_CH6 + 1
                                        if (numMuestras_CH6 == 10):
                                                divideRms_CH6 = sumaRms_CH6 / numMuestras_CH6
                                                rmsVoltaje_CH6 = divideRms_CH6 ** 0.5
#Sacamos raiz cuadrada
                                                rmsVoltaje_CH6 = format(rmsVoltaje_CH6,
'.2f')
                                                canal6_char = str(rmsVoltaje_CH6)
```

```
                                        numMuestras_CH6 = 0
                                        sumaRms_CH6 = 0.0
                                        divideRms_CH6 = 0.0
                                        print "S"+canal6_char
                                        ser.write('S')
                                        ser.write(canal6_char)
                                        ser.write("\n")

                                if (ser.inWaiting() > 0):
                                        Condicion = ser.read()
                                        if Condicion == 'A':
                                                ser.close()
                                                print "Puerto Cerrado"
                                                exit()
                                                UART.cleanup()
                                        elif Condicion == 'R':
                                                Inicia = 'R'
                                        elif Condicion == 'P':
                                                Pausa = 'P'
                                                Inicia = 'I'
                                        elif Condicion == 'I':
                                                Inicia = 'I'
                        #sleep(.01)

except KeyboardInterrupt:
    ser.close()
    print "Puerto Cerrado"
    exit()
    UART.cleanup()
```

Programa BiorretAOD.h

```
#pragma once
#include <iostream>
#include <math.h>
#include <string>
#include <fstream>

namespace Proyecto_Alfa {

	using namespace System;
	using namespace System::ComponentModel;
	using namespace System::Collections;
	using namespace System::Windows::Forms;
	using namespace System::Data;
	using namespace System::Drawing;
	using namespace System::IO;
	using namespace System::IO::Ports;
	using namespace System::Text;

	/// <summary>
	/// Resumen de MyForm
	/// </summary>
	public ref class MyForm : public System::Windows::Forms::Form
	{
	public:
		MyForm(void)
		{
			InitializeComponent();
			//
			//TODO: agregar cÛdigo de constructor aquÌ
			//
			Control::CheckForIllegalCrossThreadCalls = false;
			for each (String ^ s in SerialPort::GetPortNames())
			{
				comboBox1->Items->Add(s);
			}

			//DoubleBuffered = true;
		}

	protected:
		/// <summary>
		/// Limpiar los recursos que se estÈn utilizando.
		/// </summary>
		~MyForm()
		{
			if (components)
			{
				delete components;
			}
		}
	private: System::IO::Ports::SerialPort^  serialPort1;
	protected:
	private: System::Windows::Forms::Button^  buttonIniciar;
	private: System::Windows::Forms::TextBox^  textBox1;
	private: System::Windows::Forms::DataVisualization::Charting::Chart^  chartSenal;
```

```
        private: System::Windows::Forms::Button^  buttonParar;
        private: System::Windows::Forms::MenuStrip^  menuStrip1;
        private: System::Windows::Forms::ToolStripMenuItem^  aRCHIVOToolStripMenuItem;
        private: System::Windows::Forms::ToolStripMenuItem^  inicioToolStripMenuItem;
        private: System::Windows::Forms::ToolStripMenuItem^  salirToolStripMenuItem;
        private: System::Windows::Forms::ToolStripMenuItem^  eDITARToolStripMenuItem;
        private: System::Windows::Forms::ToolStripMenuItem^  estiloToolStripMenuItem;
        private: System::Windows::Forms::ToolStripMenuItem^  tema1ToolStripMenuItem;
        private: System::Windows::Forms::ToolStripMenuItem^  tema2ToolStripMenuItem;
        private: System::Windows::Forms::ToolStripMenuItem^  tema3ToolStripMenuItem;
        private: System::Windows::Forms::ToolStripMenuItem^  tema4ToolStripMenuItem;
        private: System::Windows::Forms::ToolStripMenuItem^  tema5ToolStripMenuItem;
        private: System::Windows::Forms::ToolStripMenuItem^  aYUDAToolStripMenuItem;
        private: System::Windows::Forms::ToolStripMenuItem^  verAyudaToolStripMenuItem;
        private: System::Windows::Forms::ToolStripMenuItem^
acercaDelProgramaToolStripMenuItem;
        private: System::Windows::Forms::ComboBox^  comboBox1;
        private: System::Windows::Forms::ProgressBar^  progressBar1;
        private: System::Windows::Forms::Label^  label1;
        private: System::Windows::Forms::Label^  label2;
        private: System::Windows::Forms::Label^  label3;
        private: System::Windows::Forms::Label^  label4;
        private: System::Windows::Forms::PictureBox^  pictureBoxAnimaciones;
        private: System::Windows::Forms::Button^  buttonReset;
        private: System::Windows::Forms::Button^  buttonApagar;
        private: System::Windows::Forms::CheckBox^  checkBoxCanal1;
        private: System::Windows::Forms::CheckBox^  checkBoxCanal2;
        private: System::Windows::Forms::CheckBox^  checkBoxCanal3;
        private: System::Windows::Forms::CheckBox^  checkBoxCanal4;
        private: System::Windows::Forms::CheckBox^  checkBoxCanal5;
        private: System::Windows::Forms::CheckBox^  checkBoxCanal6;

        private: System::Windows::Forms::Button^  buttonCanales;
        private: System::Windows::Forms::TextBox^  textBox2;
        private: System::Windows::Forms::TextBox^  textBox3;
        private: System::Windows::Forms::DataVisualization::Charting::Chart^  chartBarras;
        private: System::Windows::Forms::TextBox^  textBox4;
        private: System::Windows::Forms::TextBox^  textBox5;
        private: System::Windows::Forms::TextBox^  textBox6;
        private: System::Windows::Forms::TextBox^  textBox7;
        private: System::Windows::Forms::TextBox^  textBox8;
        private: System::Windows::Forms::TextBox^  textBox9;
        private: System::Windows::Forms::SaveFileDialog^  saveFileDialog1;
        private: System::Windows::Forms::TextBox^  textBoxRuta;

        private: System::Windows::Forms::Label^  label5;
        private: System::Windows::Forms::Button^  buttonGuardar;

        private: System::ComponentModel::IContainer^  components;

        private:
               /// <summary>
               /// Variable del diseÒador requerida.
               /// </summary>


#pragma region Windows Form Designer generated code
               /// <summary>
               /// MÈtodo necesario para admitir el DiseÒador. No se puede modificar
               /// el contenido del mÈtodo con el editor de cÛdigo.
               /// </summary>
```

```
		void InitializeComponent(void)
		{
			this->SetStyle(ControlStyles::DoubleBuffer, true);//Mejorar Paint
			this->SetStyle(ControlStyles::OptimizedDoubleBuffer, true);
			this->components = (gcnew System::ComponentModel::Container());
			System::Windows::Forms::DataVisualization::Charting::ChartArea^
chartArea1 = (gcnew System::Windows::Forms::DataVisualization::Charting::ChartArea());
			System::Windows::Forms::DataVisualization::Charting::Series^
series1 = (gcnew System::Windows::Forms::DataVisualization::Charting::Series());
			System::Windows::Forms::DataVisualization::Charting::Series^
series2 = (gcnew System::Windows::Forms::DataVisualization::Charting::Series());
			System::Windows::Forms::DataVisualization::Charting::Series^
series3 = (gcnew System::Windows::Forms::DataVisualization::Charting::Series());
			System::Windows::Forms::DataVisualization::Charting::Series^
series4 = (gcnew System::Windows::Forms::DataVisualization::Charting::Series());
			System::Windows::Forms::DataVisualization::Charting::Series^
series5 = (gcnew System::Windows::Forms::DataVisualization::Charting::Series());
			System::Windows::Forms::DataVisualization::Charting::Series^
series6 = (gcnew System::Windows::Forms::DataVisualization::Charting::Series());
			System::ComponentModel::ComponentResourceManager^  resources =
(gcnew System::ComponentModel::ComponentResourceManager(MyForm::typeid));
			System::Windows::Forms::DataVisualization::Charting::ChartArea^
chartArea2 = (gcnew System::Windows::Forms::DataVisualization::Charting::ChartArea());
			System::Windows::Forms::DataVisualization::Charting::Series^
series7 = (gcnew System::Windows::Forms::DataVisualization::Charting::Series());
			System::Windows::Forms::DataVisualization::Charting::Series^
series8 = (gcnew System::Windows::Forms::DataVisualization::Charting::Series());
			this->serialPort1 = (gcnew System::IO::Ports::SerialPort(this-
>components));
			this->buttonIniciar = (gcnew System::Windows::Forms::Button());
			this->textBox1 = (gcnew System::Windows::Forms::TextBox());
			this->chartSenal = (gcnew
System::Windows::Forms::DataVisualization::Charting::Chart());
			this->buttonParar = (gcnew System::Windows::Forms::Button());
			this->menuStrip1 = (gcnew System::Windows::Forms::MenuStrip());
			this->aRCHIVOToolStripMenuItem = (gcnew
System::Windows::Forms::ToolStripMenuItem());
			this->inicioToolStripMenuItem = (gcnew
System::Windows::Forms::ToolStripMenuItem());
			this->salirToolStripMenuItem = (gcnew
System::Windows::Forms::ToolStripMenuItem());
			this->eDITARToolStripMenuItem = (gcnew
System::Windows::Forms::ToolStripMenuItem());
			this->estiloToolStripMenuItem = (gcnew
System::Windows::Forms::ToolStripMenuItem());
			this->tema1ToolStripMenuItem = (gcnew
System::Windows::Forms::ToolStripMenuItem());
			this->tema2ToolStripMenuItem = (gcnew
System::Windows::Forms::ToolStripMenuItem());
			this->tema3ToolStripMenuItem = (gcnew
System::Windows::Forms::ToolStripMenuItem());
			this->tema4ToolStripMenuItem = (gcnew
System::Windows::Forms::ToolStripMenuItem());
			this->tema5ToolStripMenuItem = (gcnew
System::Windows::Forms::ToolStripMenuItem());
			this->aYUDAToolStripMenuItem = (gcnew
System::Windows::Forms::ToolStripMenuItem());
			this->verAyudaToolStripMenuItem = (gcnew
System::Windows::Forms::ToolStripMenuItem());
			this->acercaDelProgramaToolStripMenuItem = (gcnew
System::Windows::Forms::ToolStripMenuItem());
```

```
                    this->comboBox1 = (gcnew System::Windows::Forms::ComboBox());
                    this->progressBar1 = (gcnew System::Windows::Forms::ProgressBar());
                    this->label1 = (gcnew System::Windows::Forms::Label());
                    this->label2 = (gcnew System::Windows::Forms::Label());
                    this->label3 = (gcnew System::Windows::Forms::Label());
                    this->label4 = (gcnew System::Windows::Forms::Label());
                    this->pictureBoxAnimaciones = (gcnew
System::Windows::Forms::PictureBox());
                    this->buttonReset = (gcnew System::Windows::Forms::Button());
                    this->buttonApagar = (gcnew System::Windows::Forms::Button());
                    this->checkBoxCanal1 = (gcnew System::Windows::Forms::CheckBox());
                    this->checkBoxCanal2 = (gcnew System::Windows::Forms::CheckBox());
                    this->checkBoxCanal3 = (gcnew System::Windows::Forms::CheckBox());
                    this->checkBoxCanal4 = (gcnew System::Windows::Forms::CheckBox());
                    this->checkBoxCanal5 = (gcnew System::Windows::Forms::CheckBox());
                    this->checkBoxCanal6 = (gcnew System::Windows::Forms::CheckBox());
                    this->buttonCanales = (gcnew System::Windows::Forms::Button());
                    this->textBox2 = (gcnew System::Windows::Forms::TextBox());
                    this->textBox3 = (gcnew System::Windows::Forms::TextBox());
                    this->chartBarras = (gcnew
System::Windows::Forms::DataVisualization::Charting::Chart());
                    this->textBox4 = (gcnew System::Windows::Forms::TextBox());
                    this->textBox5 = (gcnew System::Windows::Forms::TextBox());
                    this->textBox6 = (gcnew System::Windows::Forms::TextBox());
                    this->textBox7 = (gcnew System::Windows::Forms::TextBox());
                    this->textBox8 = (gcnew System::Windows::Forms::TextBox());
                    this->textBox9 = (gcnew System::Windows::Forms::TextBox());
                    this->saveFileDialog1 = (gcnew
System::Windows::Forms::SaveFileDialog());
                    this->textBoxRuta = (gcnew System::Windows::Forms::TextBox());
                    this->label5 = (gcnew System::Windows::Forms::Label());
                    this->buttonGuardar = (gcnew System::Windows::Forms::Button());
                    (cli::safe_cast<System::ComponentModel::ISupportInitialize^>(this-
>chartSenal))->BeginInit();
                    this->menuStrip1->SuspendLayout();
                    (cli::safe_cast<System::ComponentModel::ISupportInitialize^>(this-
>pictureBoxAnimaciones))->BeginInit();
                    (cli::safe_cast<System::ComponentModel::ISupportInitialize^>(this-
>chartBarras))->BeginInit();
                    this->SuspendLayout();
                    //
                    // serialPort1
                    //
                    this->serialPort1->BaudRate = 115200;
                    this->serialPort1->DataReceived += gcnew
System::IO::Ports::SerialDataReceivedEventHandler(this,
&MyForm::serialPort1_DataReceived);
                    //
                    // buttonIniciar
                    //
                    this->buttonIniciar->Enabled = false;
                    this->buttonIniciar->Location = System::Drawing::Point(1362, 774);
                    this->buttonIniciar->Name = L"buttonIniciar";
                    this->buttonIniciar->Size = System::Drawing::Size(144, 38);
                    this->buttonIniciar->TabIndex = 1;
                    this->buttonIniciar->Text = L"Iniciar";
                    this->buttonIniciar->UseVisualStyleBackColor = true;
                    this->buttonIniciar->Click += gcnew System::EventHandler(this,
&MyForm::buttonIniciar_Click);
                    //
                    // textBox1
```

```
                    //
                    this->textBox1->Location = System::Drawing::Point(33, 229);
                    this->textBox1->Multiline = true;
                    this->textBox1->Name = L"textBox1";
                    this->textBox1->Size = System::Drawing::Size(145, 28);
                    this->textBox1->TabIndex = 6;
                    //
                    // chartSenal
                    //
                    this->chartSenal->AccessibleRole =
System::Windows::Forms::AccessibleRole::None;
                    this->chartSenal->AntiAliasing =
System::Windows::Forms::DataVisualization::Charting::AntiAliasingStyles::None;
                    this->chartSenal->BackColor = System::Drawing::Color::Transparent;
                    this->chartSenal->BackgroundImageLayout =
System::Windows::Forms::ImageLayout::Stretch;
                    this->chartSenal->BackImage =
L"C:\\Users\\Abraham\\Documents\\Visual Studio
2013\\Projects\\Proyecto_Alfa_1\\Proyecto_A"
                           L"lfa\\Imagenes\\Cuadricula.jpg";
                    this->chartSenal->BackImageAlignment =
System::Windows::Forms::DataVisualization::Charting::ChartImageAlignmentStyle::Center;
                    this->chartSenal->BorderlineColor =
System::Drawing::Color::Transparent;
                    this->chartSenal->CausesValidation = false;
                    chartArea1->AlignmentStyle =
System::Windows::Forms::DataVisualization::Charting::AreaAlignmentStyles::None;
                    chartArea1->AxisX->Enabled =
System::Windows::Forms::DataVisualization::Charting::AxisEnabled::False;
                    chartArea1->AxisX->Interval = 1;
                    chartArea1->AxisX->IsLabelAutoFit = false;
                    chartArea1->AxisX->MajorGrid->Enabled = false;
                    chartArea1->AxisX->MajorTickMark->Enabled = false;
                    chartArea1->AxisX->Maximum = 101;
                    chartArea1->AxisX->Minimum = 0;
                    chartArea1->AxisX->ScaleView->Zoomable = false;
                    chartArea1->AxisX->ScrollBar->Enabled = false;
                    chartArea1->AxisX->ScrollBar->IsPositionedInside = false;
                    chartArea1->AxisY->Enabled =
System::Windows::Forms::DataVisualization::Charting::AxisEnabled::False;
                    chartArea1->AxisY->Interval = 0.2;
                    chartArea1->AxisY->MajorGrid->Enabled = false;
                    chartArea1->AxisY->MajorTickMark->Enabled = false;
                    chartArea1->AxisY->Maximum = 2.3;
                    chartArea1->AxisY->Minimum = 0;
                    chartArea1->AxisY->ScaleView->Zoomable = false;
                    chartArea1->AxisY->ScrollBar->Enabled = false;
                    chartArea1->AxisY->ScrollBar->IsPositionedInside = false;
                    chartArea1->BackColor = System::Drawing::Color::Transparent;
                    chartArea1->CursorX->AutoScroll = false;
                    chartArea1->CursorY->AutoScroll = false;
                    chartArea1->Name = L"ChartArea1";
                    this->chartSenal->ChartAreas->Add(chartArea1);
                    this->chartSenal->Enabled = false;
                    this->chartSenal->ImeMode =
System::Windows::Forms::ImeMode::NoControl;
                    this->chartSenal->IsSoftShadows = false;
                    this->chartSenal->Location = System::Drawing::Point(511, 597);
                    this->chartSenal->Name = L"chartSenal";
                    this->chartSenal->Palette =
System::Windows::Forms::DataVisualization::Charting::ChartColorPalette::None;
```

```
                    series1->BackImageAlignment =
System::Windows::Forms::DataVisualization::Charting::ChartImageAlignmentStyle::Top;
                    series1->BorderWidth = 3;
                    series1->ChartArea = L"ChartArea1";
                    series1->ChartType =
System::Windows::Forms::DataVisualization::Charting::SeriesChartType::FastLine;
                    series1->Color = System::Drawing::Color::Maroon;
                    series1->IsVisibleInLegend = false;
                    series1->Name = L"Series1";
                    series2->BackImageAlignment =
System::Windows::Forms::DataVisualization::Charting::ChartImageAlignmentStyle::Top;
                    series2->BorderWidth = 3;
                    series2->ChartArea = L"ChartArea1";
                    series2->ChartType =
System::Windows::Forms::DataVisualization::Charting::SeriesChartType::FastLine;
                    series2->Color =
System::Drawing::Color::FromArgb(static_cast<System::Int32>(static_cast<System::Byte>(128
)), static_cast<System::Int32>(static_cast<System::Byte>(64)),
                            static_cast<System::Int32>(static_cast<System::Byte>(0)));
                    series2->IsVisibleInLegend = false;
                    series2->Name = L"Series2";
                    series3->BackImageAlignment =
System::Windows::Forms::DataVisualization::Charting::ChartImageAlignmentStyle::Top;
                    series3->BorderWidth = 3;
                    series3->ChartArea = L"ChartArea1";
                    series3->ChartType =
System::Windows::Forms::DataVisualization::Charting::SeriesChartType::FastLine;
                    series3->Color = System::Drawing::Color::Olive;
                    series3->IsVisibleInLegend = false;
                    series3->Name = L"Series3";
                    series4->BackImageAlignment =
System::Windows::Forms::DataVisualization::Charting::ChartImageAlignmentStyle::Top;
                    series4->BorderWidth = 3;
                    series4->ChartArea = L"ChartArea1";
                    series4->ChartType =
System::Windows::Forms::DataVisualization::Charting::SeriesChartType::FastLine;
                    series4->Color = System::Drawing::Color::Green;
                    series4->IsVisibleInLegend = false;
                    series4->Name = L"Series4";
                    series5->BackImageAlignment =
System::Windows::Forms::DataVisualization::Charting::ChartImageAlignmentStyle::Top;
                    series5->BorderWidth = 3;
                    series5->ChartArea = L"ChartArea1";
                    series5->ChartType =
System::Windows::Forms::DataVisualization::Charting::SeriesChartType::FastLine;
                    series5->Color = System::Drawing::Color::Teal;
                    series5->IsVisibleInLegend = false;
                    series5->Name = L"Series5";
                    series6->BackImageAlignment =
System::Windows::Forms::DataVisualization::Charting::ChartImageAlignmentStyle::Top;
                    series6->BorderWidth = 3;
                    series6->ChartArea = L"ChartArea1";
                    series6->ChartType =
System::Windows::Forms::DataVisualization::Charting::SeriesChartType::FastLine;
                    series6->Color = System::Drawing::Color::Navy;
                    series6->IsVisibleInLegend = false;
                    series6->Name = L"Series6";
                    this->chartSenal->Series->Add(series1);
                    this->chartSenal->Series->Add(series2);
                    this->chartSenal->Series->Add(series3);
                    this->chartSenal->Series->Add(series4);
```

```
                    this->chartSenal->Series->Add(series5);
                    this->chartSenal->Series->Add(series6);
                    this->chartSenal->Size = System::Drawing::Size(965, 154);
                    this->chartSenal->SuppressExceptions = true;
                    this->chartSenal->TabIndex = 2;
                    this->chartSenal->TabStop = false;
                    this->chartSenal->Text = L"chart1";
                    this->chartSenal->TextAntiAliasingQuality =
System::Windows::Forms::DataVisualization::Charting::TextAntiAliasingQuality::Normal;
                    this->chartSenal->Click += gcnew System::EventHandler(this,
&MyForm::chartSenal_Click_1);
                    //
                    // buttonParar
                    //
                    this->buttonParar->Enabled = false;
                    this->buttonParar->Location = System::Drawing::Point(1132, 774);
                    this->buttonParar->Name = L"buttonParar";
                    this->buttonParar->Size = System::Drawing::Size(145, 38);
                    this->buttonParar->TabIndex = 7;
                    this->buttonParar->Text = L"Parar";
                    this->buttonParar->UseVisualStyleBackColor = true;
                    this->buttonParar->Click += gcnew System::EventHandler(this,
&MyForm::buttonParar_Click);
                    //
                    // menuStrip1
                    //
                    this->menuStrip1->Items->AddRange(gcnew cli::array<
System::Windows::Forms::ToolStripItem^  >(3) {
                            this->aRCHIVOToolStripMenuItem,
                                    this->eDITARToolStripMenuItem, this-
>aYUDAToolStripMenuItem
                    });
                    this->menuStrip1->Location = System::Drawing::Point(0, 0);
                    this->menuStrip1->Name = L"menuStrip1";
                    this->menuStrip1->Size = System::Drawing::Size(1533, 24);
                    this->menuStrip1->TabIndex = 13;
                    this->menuStrip1->Text = L"menuStrip1";
                    //
                    // aRCHIVOToolStripMenuItem
                    //
                    this->aRCHIVOToolStripMenuItem->DropDownItems->AddRange(gcnew
cli::array< System::Windows::Forms::ToolStripItem^  >(2) {
                            this->inicioToolStripMenuItem,
                                    this->salirToolStripMenuItem
                    });
                    this->aRCHIVOToolStripMenuItem->Name = L"aRCHIVOToolStripMenuItem";
                    this->aRCHIVOToolStripMenuItem->Size = System::Drawing::Size(70,
20);
                    this->aRCHIVOToolStripMenuItem->Text = L"ARCHIVO";
                    //
                    // inicioToolStripMenuItem
                    //
                    this->inicioToolStripMenuItem->Name = L"inicioToolStripMenuItem";
                    this->inicioToolStripMenuItem->Size = System::Drawing::Size(103,
22);
                    this->inicioToolStripMenuItem->Text = L"Inicio";
                    //
                    // salirToolStripMenuItem
                    //
                    this->salirToolStripMenuItem->Name = L"salirToolStripMenuItem";
```

```
                    this->salirToolStripMenuItem->Size = System::Drawing::Size(103,
22);
                    this->salirToolStripMenuItem->Text = L"Salir";
                    //
                    // eDITARToolStripMenuItem
                    //
                    this->eDITARToolStripMenuItem->DropDownItems->AddRange(gcnew
cli::array< System::Windows::Forms::ToolStripItem^  >(1) { this->estiloToolStripMenuItem
});
                    this->eDITARToolStripMenuItem->Name = L"eDITARToolStripMenuItem";
                    this->eDITARToolStripMenuItem->Size = System::Drawing::Size(58,
20);
                    this->eDITARToolStripMenuItem->Text = L"EDITAR";
                    //
                    // estiloToolStripMenuItem
                    //
                    this->estiloToolStripMenuItem->DropDownItems->AddRange(gcnew
cli::array< System::Windows::Forms::ToolStripItem^  >(5) {
                            this->tema1ToolStripMenuItem,
                                   this->tema2ToolStripMenuItem, this-
>tema3ToolStripMenuItem, this->tema4ToolStripMenuItem, this->tema5ToolStripMenuItem
                    });
                    this->estiloToolStripMenuItem->Name = L"estiloToolStripMenuItem";
                    this->estiloToolStripMenuItem->Size = System::Drawing::Size(104,
22);
                    this->estiloToolStripMenuItem->Text = L"Tema";
                    //
                    // tema1ToolStripMenuItem
                    //
                    this->tema1ToolStripMenuItem->Name = L"tema1ToolStripMenuItem";
                    this->tema1ToolStripMenuItem->Size = System::Drawing::Size(113,
22);
                    this->tema1ToolStripMenuItem->Text = L"Tema 1";
                    //
                    // tema2ToolStripMenuItem
                    //
                    this->tema2ToolStripMenuItem->Name = L"tema2ToolStripMenuItem";
                    this->tema2ToolStripMenuItem->Size = System::Drawing::Size(113,
22);
                    this->tema2ToolStripMenuItem->Text = L"Tema 2";
                    //
                    // tema3ToolStripMenuItem
                    //
                    this->tema3ToolStripMenuItem->Name = L"tema3ToolStripMenuItem";
                    this->tema3ToolStripMenuItem->Size = System::Drawing::Size(113,
22);
                    this->tema3ToolStripMenuItem->Text = L"Tema 3";
                    //
                    // tema4ToolStripMenuItem
                    //
                    this->tema4ToolStripMenuItem->Name = L"tema4ToolStripMenuItem";
                    this->tema4ToolStripMenuItem->Size = System::Drawing::Size(113,
22);
                    this->tema4ToolStripMenuItem->Text = L"Tema 4";
                    //
                    // tema5ToolStripMenuItem
                    //
                    this->tema5ToolStripMenuItem->Name = L"tema5ToolStripMenuItem";
                    this->tema5ToolStripMenuItem->Size = System::Drawing::Size(113,
22);
                    this->tema5ToolStripMenuItem->Text = L"Tema 5";
```

```
                    //
                    // aYUDAToolStripMenuItem
                    //
                    this->aYUDAToolStripMenuItem->DropDownItems->AddRange(gcnew
cli::array< System::Windows::Forms::ToolStripItem^  >(2) {
                            this->verAyudaToolStripMenuItem,
                                    this->acercaDelProgramaToolStripMenuItem
                    });
                    this->aYUDAToolStripMenuItem->Name = L"aYUDAToolStripMenuItem";
                    this->aYUDAToolStripMenuItem->Size = System::Drawing::Size(58, 20);
                    this->aYUDAToolStripMenuItem->Text = L"AYUDA";
                    //
                    // verAyudaToolStripMenuItem
                    //
                    this->verAyudaToolStripMenuItem->Name =
L"verAyudaToolStripMenuItem";
                    this->verAyudaToolStripMenuItem->Size = System::Drawing::Size(184,
22);
                    this->verAyudaToolStripMenuItem->Text = L"Ver Ayuda";
                    //
                    // acercaDelProgramaToolStripMenuItem
                    //
                    this->acercaDelProgramaToolStripMenuItem->Name =
L"acercaDelProgramaToolStripMenuItem";
                    this->acercaDelProgramaToolStripMenuItem->Size =
System::Drawing::Size(184, 22);
                    this->acercaDelProgramaToolStripMenuItem->Text = L"Acerca del
Programa";
                    //
                    // comboBox1
                    //
                    this->comboBox1->FormattingEnabled = true;
                    this->comboBox1->Location = System::Drawing::Point(34, 71);
                    this->comboBox1->Name = L"comboBox1";
                    this->comboBox1->Size = System::Drawing::Size(145, 21);
                    this->comboBox1->TabIndex = 0;
                    this->comboBox1->SelectedIndexChanged += gcnew
System::EventHandler(this, &MyForm::comboBox1_SelectedIndexChanged);
                    //
                    // progressBar1
                    //
                    this->progressBar1->Location = System::Drawing::Point(33, 138);
                    this->progressBar1->Name = L"progressBar1";
                    this->progressBar1->Size = System::Drawing::Size(144, 28);
                    this->progressBar1->TabIndex = 4;
                    //
                    // label1
                    //
                    this->label1->AutoSize = true;
                    this->label1->Font = (gcnew System::Drawing::Font(L"Microsoft Sans
Serif", 12, System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,
                            static_cast<System::Byte>(0)));
                    this->label1->Location = System::Drawing::Point(30, 48);
                    this->label1->Name = L"label1";
                    this->label1->Size = System::Drawing::Size(56, 20);
                    this->label1->TabIndex = 8;
                    this->label1->Text = L"Puerto";
                    //
                    // label2
                    //
                    this->label2->AutoSize = true;
```

```
                    this->label2->Font = (gcnew System::Drawing::Font(L"Microsoft Sans
Serif", 12, System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,
                            static_cast<System::Byte>(0)));
                    this->label2->Location = System::Drawing::Point(29, 115);
                    this->label2->Name = L"label2";
                    this->label2->Size = System::Drawing::Size(73, 20);
                    this->label2->TabIndex = 9;
                    this->label2->Text = L"Progreso";
                    //
                    // label3
                    //
                    this->label3->AutoSize = true;
                    this->label3->Font = (gcnew System::Drawing::Font(L"Microsoft Sans
Serif", 12, System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,
                            static_cast<System::Byte>(0)));
                    this->label3->Location = System::Drawing::Point(230, 48);
                    this->label3->Name = L"label3";
                    this->label3->Size = System::Drawing::Size(162, 20);
                    this->label3->TabIndex = 11;
                    this->label3->Text = L"SelecciÛn de Canales";
                    //
                    // label4
                    //
                    this->label4->AutoSize = true;
                    this->label4->Font = (gcnew System::Drawing::Font(L"Microsoft Sans
Serif", 12, System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,
                            static_cast<System::Byte>(0)));
                    this->label4->Location = System::Drawing::Point(30, 188);
                    this->label4->Name = L"label4";
                    this->label4->Size = System::Drawing::Size(126, 20);
                    this->label4->TabIndex = 12;
                    this->label4->Text = L"Cuadro de Texto";
                    //
                    // pictureBoxAnimaciones
                    //
                    this->pictureBoxAnimaciones->BackgroundImage =
(cli::safe_cast<System::Drawing::Image^>(resources-
>GetObject(L"pictureBoxAnimaciones.BackgroundImage")));
                    this->pictureBoxAnimaciones->BackgroundImageLayout =
System::Windows::Forms::ImageLayout::Center;
                    this->pictureBoxAnimaciones->Location = System::Drawing::Point(511,
100);
                    this->pictureBoxAnimaciones->Name = L"pictureBoxAnimaciones";
                    this->pictureBoxAnimaciones->Size = System::Drawing::Size(601,
440);
                    this->pictureBoxAnimaciones->TabIndex = 14;
                    this->pictureBoxAnimaciones->TabStop = false;
                    this->pictureBoxAnimaciones->Click += gcnew
System::EventHandler(this, &MyForm::pictureBoxAnimaciones_Click);
                    //
                    // buttonReset
                    //
                    this->buttonReset->Enabled = false;
                    this->buttonReset->Location = System::Drawing::Point(511, 774);
                    this->buttonReset->Name = L"buttonReset";
                    this->buttonReset->Size = System::Drawing::Size(145, 38);
                    this->buttonReset->TabIndex = 23;
                    this->buttonReset->Text = L"Reiniciar Valores";
                    this->buttonReset->UseVisualStyleBackColor = true;
                    this->buttonReset->Click += gcnew System::EventHandler(this,
&MyForm::buttonReset_Click);
```

```
                    //
                    // buttonApagar
                    //
                    this->buttonApagar->Location = System::Drawing::Point(1377, 30);
                    this->buttonApagar->Name = L"buttonApagar";
                    this->buttonApagar->Size = System::Drawing::Size(144, 38);
                    this->buttonApagar->TabIndex = 24;
                    this->buttonApagar->Text = L"Apagar";
                    this->buttonApagar->UseVisualStyleBackColor = true;
                    this->buttonApagar->Click += gcnew System::EventHandler(this,
&MyForm::buttonApagar_Click);
                    //
                    // checkBoxCanal1
                    //
                    this->checkBoxCanal1->AutoSize = true;
                    this->checkBoxCanal1->Font = (gcnew
System::Drawing::Font(L"Microsoft Sans Serif", 14.25F,
System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,
                            static_cast<System::Byte>(0)));
                    this->checkBoxCanal1->Location = System::Drawing::Point(263, 78);
                    this->checkBoxCanal1->Name = L"checkBoxCanal1";
                    this->checkBoxCanal1->Size = System::Drawing::Size(95, 28);
                    this->checkBoxCanal1->TabIndex = 25;
                    this->checkBoxCanal1->Text = L"Canal A";
                    this->checkBoxCanal1->UseVisualStyleBackColor = true;
                    this->checkBoxCanal1->CheckedChanged += gcnew
System::EventHandler(this, &MyForm::checkBoxCanal1_CheckedChanged);
                    //
                    // checkBoxCanal2
                    //
                    this->checkBoxCanal2->AutoSize = true;
                    this->checkBoxCanal2->Font = (gcnew
System::Drawing::Font(L"Microsoft Sans Serif", 14.25F,
System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,
                            static_cast<System::Byte>(0)));
                    this->checkBoxCanal2->Location = System::Drawing::Point(263, 112);
                    this->checkBoxCanal2->Name = L"checkBoxCanal2";
                    this->checkBoxCanal2->Size = System::Drawing::Size(94, 28);
                    this->checkBoxCanal2->TabIndex = 26;
                    this->checkBoxCanal2->Text = L"Canal B";
                    this->checkBoxCanal2->UseVisualStyleBackColor = true;
                    this->checkBoxCanal2->CheckedChanged += gcnew
System::EventHandler(this, &MyForm::checkBoxCanal2_CheckedChanged);
                    //
                    // checkBoxCanal3
                    //
                    this->checkBoxCanal3->AutoSize = true;
                    this->checkBoxCanal3->Font = (gcnew
System::Drawing::Font(L"Microsoft Sans Serif", 14.25F,
System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,
                            static_cast<System::Byte>(0)));
                    this->checkBoxCanal3->Location = System::Drawing::Point(263, 146);
                    this->checkBoxCanal3->Name = L"checkBoxCanal3";
                    this->checkBoxCanal3->Size = System::Drawing::Size(95, 28);
                    this->checkBoxCanal3->TabIndex = 27;
                    this->checkBoxCanal3->Text = L"Canal C";
                    this->checkBoxCanal3->UseVisualStyleBackColor = true;
                    this->checkBoxCanal3->CheckedChanged += gcnew
System::EventHandler(this, &MyForm::checkBoxCanal3_CheckedChanged);
                    //
                    // checkBoxCanal4
```

```
                    //
                    this->checkBoxCanal4->AutoSize = true;
                    this->checkBoxCanal4->Font = (gcnew
System::Drawing::Font(L"Microsoft Sans Serif", 14.25F,
System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,
                            static_cast<System::Byte>(0)));
                    this->checkBoxCanal4->Location = System::Drawing::Point(263, 180);
                    this->checkBoxCanal4->Name = L"checkBoxCanal4";
                    this->checkBoxCanal4->Size = System::Drawing::Size(95, 28);
                    this->checkBoxCanal4->TabIndex = 28;
                    this->checkBoxCanal4->Text = L"Canal D";
                    this->checkBoxCanal4->UseVisualStyleBackColor = true;
                    this->checkBoxCanal4->CheckedChanged += gcnew
System::EventHandler(this, &MyForm::checkBoxCanal4_CheckedChanged);
                    //
                    // checkBoxCanal5
                    //
                    this->checkBoxCanal5->AutoSize = true;
                    this->checkBoxCanal5->Font = (gcnew
System::Drawing::Font(L"Microsoft Sans Serif", 14.25F,
System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,
                            static_cast<System::Byte>(0)));
                    this->checkBoxCanal5->Location = System::Drawing::Point(263, 214);
                    this->checkBoxCanal5->Name = L"checkBoxCanal5";
                    this->checkBoxCanal5->Size = System::Drawing::Size(95, 28);
                    this->checkBoxCanal5->TabIndex = 29;
                    this->checkBoxCanal5->Text = L"Canal E";
                    this->checkBoxCanal5->UseVisualStyleBackColor = true;
                    this->checkBoxCanal5->CheckedChanged += gcnew
System::EventHandler(this, &MyForm::checkBoxCanal5_CheckedChanged);
                    //
                    // checkBoxCanal6
                    //
                    this->checkBoxCanal6->AutoSize = true;
                    this->checkBoxCanal6->Font = (gcnew
System::Drawing::Font(L"Microsoft Sans Serif", 14.25F,
System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,
                            static_cast<System::Byte>(0)));
                    this->checkBoxCanal6->Location = System::Drawing::Point(263, 248);
                    this->checkBoxCanal6->Name = L"checkBoxCanal6";
                    this->checkBoxCanal6->Size = System::Drawing::Size(94, 28);
                    this->checkBoxCanal6->TabIndex = 30;
                    this->checkBoxCanal6->Text = L"Canal F";
                    this->checkBoxCanal6->UseVisualStyleBackColor = true;
                    this->checkBoxCanal6->CheckedChanged += gcnew
System::EventHandler(this, &MyForm::checkBoxCanal6_CheckedChanged);
                    //
                    // buttonCanales
                    //
                    this->buttonCanales->Enabled = false;
                    this->buttonCanales->Location = System::Drawing::Point(284, 294);
                    this->buttonCanales->Name = L"buttonCanales";
                    this->buttonCanales->Size = System::Drawing::Size(145, 38);
                    this->buttonCanales->TabIndex = 32;
                    this->buttonCanales->Text = L"Confirmar Canales";
                    this->buttonCanales->UseVisualStyleBackColor = true;
                    this->buttonCanales->Click += gcnew System::EventHandler(this,
&MyForm::buttonCanales_Click);
                    //
                    // textBox2
                    //
```

```
                    this->textBox2->Location = System::Drawing::Point(385, 371);
                    this->textBox2->Multiline = true;
                    this->textBox2->Name = L"textBox2";
                    this->textBox2->Size = System::Drawing::Size(60, 38);
                    this->textBox2->TabIndex = 34;
                    //
                    // textBox3
                    //
                    this->textBox3->Location = System::Drawing::Point(291, 371);
                    this->textBox3->Multiline = true;
                    this->textBox3->Name = L"textBox3";
                    this->textBox3->Size = System::Drawing::Size(70, 38);
                    this->textBox3->TabIndex = 35;
                    //
                    // chartBarras
                    //
                    this->chartBarras->AccessibleRole =
System::Windows::Forms::AccessibleRole::Window;
                    this->chartBarras->BackColor = System::Drawing::Color::Transparent;
                    this->chartBarras->BackImage =
L"C:\\Users\\Abraham\\Documents\\Visual Studio
2013\\Projects\\Proyecto_Alfa_1\\Proyecto_A"
                           L"lfa\\Imagenes\\Cuadricula.jpg";
                    this->chartBarras->BorderlineColor =
System::Drawing::Color::Transparent;
                    chartArea2->AxisX->Enabled =
System::Windows::Forms::DataVisualization::Charting::AxisEnabled::False;
                    chartArea2->AxisX->MajorGrid->Enabled = false;
                    chartArea2->AxisX->MajorTickMark->Enabled = false;
                    chartArea2->AxisY->Enabled =
System::Windows::Forms::DataVisualization::Charting::AxisEnabled::False;
                    chartArea2->AxisY->MajorGrid->Enabled = false;
                    chartArea2->AxisY->MajorTickMark->Enabled = false;
                    chartArea2->AxisY->Maximum = 1.8;
                    chartArea2->AxisY->Minimum = 0;
                    chartArea2->BackColor = System::Drawing::Color::Transparent;
                    chartArea2->Name = L"ChartArea1";
                    this->chartBarras->ChartAreas->Add(chartArea2);
                    this->chartBarras->Location = System::Drawing::Point(1160, 180);
                    this->chartBarras->Name = L"chartBarras";
                    series7->ChartArea = L"ChartArea1";
                    series7->Name = L"Series1";
                    series8->ChartArea = L"ChartArea1";
                    series8->Name = L"Series2";
                    this->chartBarras->Series->Add(series7);
                    this->chartBarras->Series->Add(series8);
                    this->chartBarras->Size = System::Drawing::Size(316, 359);
                    this->chartBarras->TabIndex = 36;
                    this->chartBarras->Text = L"chart1";
                    //
                    // textBox4
                    //
                    this->textBox4->BackColor = System::Drawing::Color::Maroon;
                    this->textBox4->BorderStyle =
System::Windows::Forms::BorderStyle::None;
                    this->textBox4->Location = System::Drawing::Point(385, 84);
                    this->textBox4->Multiline = true;
                    this->textBox4->Name = L"textBox4";
                    this->textBox4->Size = System::Drawing::Size(62, 14);
                    this->textBox4->TabIndex = 37;
                    //
```

```
                    // textBox5
                    //
                    this->textBox5->BackColor =
System::Drawing::Color::FromArgb(static_cast<System::Int32>(static_cast<System::Byte>(128
)), static_cast<System::Int32>(static_cast<System::Byte>(64)),
                            static_cast<System::Int32>(static_cast<System::Byte>(0)));
                    this->textBox5->BorderStyle =
System::Windows::Forms::BorderStyle::None;
                    this->textBox5->Location = System::Drawing::Point(385, 118);
                    this->textBox5->Multiline = true;
                    this->textBox5->Name = L"textBox5";
                    this->textBox5->Size = System::Drawing::Size(62, 14);
                    this->textBox5->TabIndex = 38;
                    //
                    // textBox6
                    //
                    this->textBox6->BackColor = System::Drawing::Color::Olive;
                    this->textBox6->BorderStyle =
System::Windows::Forms::BorderStyle::None;
                    this->textBox6->Location = System::Drawing::Point(385, 152);
                    this->textBox6->Multiline = true;
                    this->textBox6->Name = L"textBox6";
                    this->textBox6->Size = System::Drawing::Size(62, 14);
                    this->textBox6->TabIndex = 39;
                    //
                    // textBox7
                    //
                    this->textBox7->BackColor = System::Drawing::Color::Green;
                    this->textBox7->BorderStyle =
System::Windows::Forms::BorderStyle::None;
                    this->textBox7->Location = System::Drawing::Point(385, 186);
                    this->textBox7->Multiline = true;
                    this->textBox7->Name = L"textBox7";
                    this->textBox7->Size = System::Drawing::Size(62, 14);
                    this->textBox7->TabIndex = 40;
                    //
                    // textBox8
                    //
                    this->textBox8->BackColor = System::Drawing::Color::Teal;
                    this->textBox8->BorderStyle =
System::Windows::Forms::BorderStyle::None;
                    this->textBox8->Location = System::Drawing::Point(385, 220);
                    this->textBox8->Multiline = true;
                    this->textBox8->Name = L"textBox8";
                    this->textBox8->Size = System::Drawing::Size(62, 14);
                    this->textBox8->TabIndex = 41;
                    //
                    // textBox9
                    //
                    this->textBox9->BackColor = System::Drawing::Color::Navy;
                    this->textBox9->BorderStyle =
System::Windows::Forms::BorderStyle::None;
                    this->textBox9->Location = System::Drawing::Point(385, 254);
                    this->textBox9->Multiline = true;
                    this->textBox9->Name = L"textBox9";
                    this->textBox9->Size = System::Drawing::Size(62, 14);
                    this->textBox9->TabIndex = 42;
                    //
                    // saveFileDialog1
                    //
                    this->saveFileDialog1->DefaultExt = L"txt";
```

```
                    this->saveFileDialog1->Filter = L"|*.txt";
                    //
                    // textBoxRuta
                    //
                    this->textBoxRuta->Location = System::Drawing::Point(34, 315);
                    this->textBoxRuta->Multiline = true;
                    this->textBoxRuta->Name = L"textBoxRuta";
                    this->textBoxRuta->Size = System::Drawing::Size(145, 70);
                    this->textBoxRuta->TabIndex = 43;
                    //
                    // label5
                    //
                    this->label5->AutoSize = true;
                    this->label5->Font = (gcnew System::Drawing::Font(L"Microsoft Sans
Serif", 12, System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,
                           static_cast<System::Byte>(0)));
                    this->label5->Location = System::Drawing::Point(30, 283);
                    this->label5->Name = L"label5";
                    this->label5->Size = System::Drawing::Size(154, 20);
                    this->label5->TabIndex = 44;
                    this->label5->Text = L"Ruta Base de Datos";
                    //
                    // buttonGuardar
                    //
                    this->buttonGuardar->Location = System::Drawing::Point(34, 413);
                    this->buttonGuardar->Name = L"buttonGuardar";
                    this->buttonGuardar->Size = System::Drawing::Size(145, 38);
                    this->buttonGuardar->TabIndex = 45;
                    this->buttonGuardar->Text = L"Guardar";
                    this->buttonGuardar->UseVisualStyleBackColor = true;
                    this->buttonGuardar->Click += gcnew System::EventHandler(this,
&MyForm::buttonGuardar_Click);
                    //
                    // MyForm
                    //
                    this->AutoScaleDimensions = System::Drawing::SizeF(6, 13);
                    this->AutoScaleMode = System::Windows::Forms::AutoScaleMode::Font;
                    this->BackColor = System::Drawing::Color::SteelBlue;
                    this->BackgroundImageLayout =
System::Windows::Forms::ImageLayout::Center;
                    this->ClientSize = System::Drawing::Size(1533, 876);
                    this->Controls->Add(this->buttonGuardar);
                    this->Controls->Add(this->label5);
                    this->Controls->Add(this->textBoxRuta);
                    this->Controls->Add(this->textBox9);
                    this->Controls->Add(this->textBox8);
                    this->Controls->Add(this->textBox7);
                    this->Controls->Add(this->textBox6);
                    this->Controls->Add(this->textBox5);
                    this->Controls->Add(this->textBox4);
                    this->Controls->Add(this->chartBarras);
                    this->Controls->Add(this->textBox3);
                    this->Controls->Add(this->textBox2);
                    this->Controls->Add(this->buttonCanales);
                    this->Controls->Add(this->checkBoxCanal6);
                    this->Controls->Add(this->checkBoxCanal5);
                    this->Controls->Add(this->checkBoxCanal4);
                    this->Controls->Add(this->checkBoxCanal3);
                    this->Controls->Add(this->checkBoxCanal2);
                    this->Controls->Add(this->checkBoxCanal1);
                    this->Controls->Add(this->buttonApagar);
```

```
                    this->Controls->Add(this->buttonReset);
                    this->Controls->Add(this->pictureBoxAnimaciones);
                    this->Controls->Add(this->label4);
                    this->Controls->Add(this->label3);
                    this->Controls->Add(this->buttonParar);
                    this->Controls->Add(this->label2);
                    this->Controls->Add(this->chartSenal);
                    this->Controls->Add(this->label1);
                    this->Controls->Add(this->textBox1);
                    this->Controls->Add(this->progressBar1);
                    this->Controls->Add(this->buttonIniciar);
                    this->Controls->Add(this->comboBox1);
                    this->Controls->Add(this->menuStrip1);
                    this->MainMenuStrip = this->menuStrip1;
                    this->Name = L"MyForm";
                    this->StartPosition =
System::Windows::Forms::FormStartPosition::CenterParent;
                    this->Text = L"  ";
                    this->Load += gcnew System::EventHandler(this,
&MyForm::MyForm_Load);
                    (cli::safe_cast<System::ComponentModel::ISupportInitialize^>(this-
>chartSenal))->EndInit();
                    this->menuStrip1->ResumeLayout(false);
                    this->menuStrip1->PerformLayout();
                    (cli::safe_cast<System::ComponentModel::ISupportInitialize^>(this-
>pictureBoxAnimaciones))->EndInit();
                    (cli::safe_cast<System::ComponentModel::ISupportInitialize^>(this-
>chartBarras))->EndInit();
                    this->ResumeLayout(false);
                    this->PerformLayout();

             }

             int i1 = 0, i2 = 0, i3 = 0, i4 = 0, i5 = 0, i6 = 0, id = 0, numCanales =
0, cuentaBotones = 0, baseDatos = 0;
             //int timerA = 0, timerB = 0, timerC = 0, timerD = 0, timerE = 0, timerF =
0;
             double timer = 0 ;
             double promedioCH1 = 0.0, datosCH1 = 0.0;
             double promedioCH2 = 0.0, datosCH2 = 0.0;
             double promedioCH3 = 0.0, datosCH3 = 0.0;
             double promedioCH4 = 0.0, datosCH4 = 0.0;
             double promedioCH5 = 0.0, datosCH5 = 0.0;
             double promedioCH6 = 0.0, datosCH6 = 0.0;
             bool flagActiva = false, escribeA = true, escribeB = true, escribeC =
true, escribeD = true, escribeE = true, escribeF = true;

             StreamWriter^ sw;



       #pragma endregion
             private: System::Void MyForm_Load(System::Object^  sender,
System::EventArgs^  e) {

                    this->chartSenal->Series["Series1"]->Enabled = false;
                    this->chartSenal->Series["Series2"]->Enabled = false;
                    this->chartSenal->Series["Series3"]->Enabled = false;
                    this->chartSenal->Series["Series4"]->Enabled = false;
                    this->chartSenal->Series["Series5"]->Enabled = false;
                    this->chartSenal->Series["Series6"]->Enabled = false;
```

```
		}

		private: System::Void buttonGuardar_Click(System::Object^  sender,
System::EventArgs^  e) {
			saveFileDialog1->ShowDialog();
			textBoxRuta->Text = Convert::ToString(saveFileDialog1->FileName);
			sw = gcnew StreamWriter(saveFileDialog1->FileName);
			buttonIniciar->Enabled = false;
			buttonParar->Enabled = false;
			buttonApagar->Enabled = true;
			buttonReset->Enabled = false;
			buttonCanales->Enabled = true;
			buttonGuardar->Enabled = false;

		}


		void leeDatos(double datoEntrante){
					try{
						if (datoEntrante >= 90.0 && datoEntrante <
99.0){
							timer = datoEntrante - 90.0;
						}
						else if (datoEntrante >= 910.0 &&
datoEntrante < 999.0){
							timer = datoEntrante - 900.0;
						}
						else if (datoEntrante >= 9100.0){
							timer = datoEntrante - 9000.0;
						}

						/*Time A*/
						if (datoEntrante >= 10.00 && datoEntrante <=
19.99){
							if (timer < 99){
								datoEntrante =
Math::Round((datoEntrante - 10.0), 2);
								this->chartSenal-
>Series["Series1"]->Points->AddXY(timer, datoEntrante);
								textBox1->Text =
Convert::ToString(datoEntrante);
								if (baseDatos != numCanales)
								{
									sw->Write("Canal A ->
");
									sw-
>Write(Convert::ToString(datoEntrante));
									sw->Write("|");
									baseDatos++;
								}
								else
								{
									baseDatos = 0;
									sw->WriteLine(" ");
									sw->Write("Canal A ->
");
									sw-
>Write(Convert::ToString(datoEntrante));
									sw->Write("|");
									baseDatos++;
								}
```

```
                                                        datosCH1 = datosCH1 +
datoEntrante;
                                                        i1 = i1 + 1;
                                                }
                                                else{
                                                        chartBarras-
>Series["Series1"]->Points->Clear();
                                                        promedioCH1 =
Math::Round((datosCH1 / i1), 2);
                                                        chartBarras-
>Series["Series1"]->Points->AddXY(1, promedioCH1);
                                                        textBox2->Text =
Convert::ToString(promedioCH1);

                                                        if (checkBoxCanal6->Checked){
                                                                return;
                                                        }
                                                        else if (checkBoxCanal5-
>Checked){
                                                                return;
                                                        }
                                                        else if (checkBoxCanal4-
>Checked){
                                                                return;
                                                        }
                                                        else if (checkBoxCanal3-
>Checked){
                                                                return;
                                                        }
                                                        else if (checkBoxCanal2-
>Checked){
                                                                return;
                                                        }
                                                        else{
                                                                this->chartSenal-
>Series["Series1"]->Points->Clear();
                                                                promedioCH1 = 0.0;
datosCH1 = 0.0; i1 = 0; baseDatos = numCanales;
                                                                this->serialPort1-
>DiscardInBuffer(); //Descarta datos del buffer
                                                        }

                                                        if (promedioCH1 >= 0.9){
                                                                pictureBoxAnimaciones-
>BackgroundImage = Image::FromFile("Pelota_75.jpg");
                                                        }
                                                        else if (promedioCH1 <= 0.9){
                                                                pictureBoxAnimaciones-
>BackgroundImage = Image::FromFile("Pelota_10.jpg");
                                                        }
                                                }
                                        }

                                        /*Time B*/
                                        if (datoEntrante >= 20.00 && datoEntrante <=
29.99){
                                                if (timer < 99){
                                                        datoEntrante =
Math::Round((datoEntrante - 20.0), 2);
                                                        this->chartSenal-
>Series["Series2"]->Points->AddXY(timer, datoEntrante);
```

```
Convert::ToString(datoEntrante);


");
>Write(Convert::ToString(datoEntrante));




");
>Write(Convert::ToString(datoEntrante));



datoEntrante;


>Series["Series2"]->Points->Clear();

Math::Round((datosCH2 / i2), 2);
>Series["Series2"]->Points->AddXY(5, promedioCH2);
Convert::ToString(promedioCH2);




>Checked){


>Checked){


>Checked){


>Checked){
>Series["Series1"]->Points->Clear();
datosCH1 = 0.0; i1 = 0; baseDatos = numCanales;
>Series["Series2"]->Points->Clear();
                                                        textBox1->Text =

                                                        if (baseDatos != numCanales)
                                                        {
                                                                sw->Write("Canal B ->

                                                                sw-

                                                                sw->Write("|");
                                                                baseDatos++;
                                                        }
                                                        else{
                                                                baseDatos = 0;
                                                                sw->WriteLine(" ");
                                                                sw->Write("Canal B ->

                                                                sw-

                                                                sw->Write("|");
                                                                baseDatos++;
                                                        }
                                                        datosCH2 = datosCH2 +

                                                        i2 = i2 + 1;
                                                }
                                                else{
                                                        chartBarras-


                                                        promedioCH2 =

                                                        chartBarras-

                                                        textBox3->Text =


                                                        if (checkBoxCanal6->Checked){
                                                                return;
                                                        }
                                                        else if (checkBoxCanal5-

                                                                return;
                                                        }
                                                        else if (checkBoxCanal4-

                                                                return;
                                                        }
                                                        else if (checkBoxCanal3-

                                                                return;
                                                        }
                                                        else if (checkBoxCanal1-

                                                                this->chartSenal-

                                                                promedioCH1 = 0.0;

                                                                this->chartSenal-
```

```
                                                                promedioCH2 = 0.0;
datosCH2 = 0.0; i2 = 0;
                                                                sw->WriteLine(" ");
                                                                this->serialPort1-
>DiscardInBuffer(); //Descarta datos del buffer
                                                        }
                                                        else{
                                                                this->chartSenal-
>Series["Series2"]->Points->Clear();
                                                                promedioCH2 = 0.0;
datosCH2 = 0.0; i2 = 0;  baseDatos = numCanales;
                                                                this->serialPort1-
>DiscardInBuffer(); //Descarta datos del buffer
                                                        }

                                                }
                                        }

                                        /*Time C*/
                                        if (datoEntrante >= 30.00 && datoEntrante <=
39.99){
                                                if (timer < 99){
                                                        datoEntrante =
Math::Round((datoEntrante - 30.0), 2);
                                                        this->chartSenal-
>Series["Series3"]->Points->AddXY(timer, datoEntrante);
                                                        textBox1->Text =
Convert::ToString(datoEntrante);
                                                        if (baseDatos != numCanales)
                                                        {
                                                                sw->Write("Canal C ->
");
                                                                sw-
>Write(Convert::ToString(datoEntrante));
                                                                sw->Write("|");
                                                                baseDatos++;
                                                        }
                                                        else{
                                                                baseDatos = 0;
                                                                sw->WriteLine(" ");
                                                                sw->Write("Canal C ->
");
                                                                sw-
>Write(Convert::ToString(datoEntrante));
                                                                sw->Write("|");
                                                                baseDatos++;
                                                        }
                                                        datosCH3 = datosCH3 +
datoEntrante;
                                                        i3 = i3 + 1;
                                                }
                                                else{
                                                        promedioCH3 =
Math::Round((datosCH3 / i3), 2);

                                                        if (checkBoxCanal6->Checked){
                                                                return;
                                                        }
                                                        else if (checkBoxCanal5-
>Checked){
                                                                return;
```

```
                                                        }
                                                        else if (checkBoxCanal4-
>Checked){
                                                               return;
                                                        }
                                                        else if (checkBoxCanal1-
>Checked || checkBoxCanal2->Checked){
                                                               this->chartSenal-
>Series["Series1"]->Points->Clear();
                                                               promedioCH1 = 0.0;
datosCH1 = 0.0; i1 = 0; baseDatos = numCanales;
                                                               this->chartSenal-
>Series["Series2"]->Points->Clear();
                                                               promedioCH2 = 0.0;
datosCH2 = 0.0; i2 = 0;
                                                               this->chartSenal-
>Series["Series3"]->Points->Clear();
                                                               promedioCH3 = 0.0;
datosCH3 = 0.0; i3 = 0;
                                                               escribeA = true;
escribeB = true; escribeC = true;
                                                               sw->WriteLine(" ");
                                                               this->serialPort1-
>DiscardInBuffer(); //Descarta datos del buffer
                                                        }
                                                        else{
                                                               this->chartSenal-
>Series["Series3"]->Points->Clear();
                                                               promedioCH3 = 0.0;
datosCH3 = 0.0; i3 = 0; baseDatos = numCanales;
                                                               this->serialPort1-
>DiscardInBuffer(); //Descarta datos del buffer
                                                        }
                                                 }
                                          }

                                          /*Time D*/
                                          if (datoEntrante >= 40.00 && datoEntrante <=
49.99){
                                                 if (timer < 99){
                                                        datoEntrante =
Math::Round((datoEntrante - 40.0), 2);
                                                        this->chartSenal-
>Series["Series4"]->Points->AddXY(timer, datoEntrante);
                                                        textBox1->Text =
Convert::ToString(datoEntrante);
                                                        if (baseDatos != numCanales)
                                                        {
                                                               sw->Write("Canal D ->
");
                                                               sw-
>Write(Convert::ToString(datoEntrante));
                                                               sw->Write("|");
                                                               baseDatos++;
                                                        }
                                                        else{
                                                               baseDatos = 0;
                                                               sw->WriteLine(" ");
                                                               sw->Write("Canal D ->
");
```

```
                                                                sw-
>Write(Convert::ToString(datoEntrante));
                                                                sw->Write("|");
                                                                baseDatos++;
                                                        }
                                                        datosCH4 = datosCH4 +
datoEntrante;
                                                        i4 = i4 + 1;
                                                }
                                                else{
                                                        promedioCH4 =
Math::Round((datosCH4 / i4), 2);
                                                        if (checkBoxCanal6->Checked){
                                                                return;
                                                        }
                                                        else if (checkBoxCanal5-
>Checked){
                                                                return;
                                                        }
                                                        else if (checkBoxCanal1-
>Checked || checkBoxCanal2->Checked || checkBoxCanal3->Checked){
                                                                this->chartSenal-
>Series["Series1"]->Points->Clear();
                                                                promedioCH1 = 0.0;
datosCH1 = 0.0; i1 = 0; baseDatos = numCanales;
                                                                this->chartSenal-
>Series["Series2"]->Points->Clear();
                                                                promedioCH2 = 0.0;
datosCH2 = 0.0; i2 = 0;
                                                                this->chartSenal-
>Series["Series3"]->Points->Clear();
                                                                promedioCH3 = 0.0;
datosCH3 = 0.0; i3 = 0;
                                                                this->chartSenal-
>Series["Series4"]->Points->Clear();
                                                                promedioCH4 = 0.0;
datosCH4 = 0.0; i4 = 0;
                                                                sw->WriteLine(" ");
                                                                this->serialPort1-
>DiscardInBuffer(); //Descarta datos del buffer
                                                        }
                                                        else{
                                                                this->chartSenal-
>Series["Series4"]->Points->Clear();
                                                                promedioCH4 = 0.0;
datosCH4 = 0.0; i4 = 0; baseDatos = numCanales;
                                                                this->serialPort1-
>DiscardInBuffer(); //Descarta datos del buffer
                                                        }
                                                }
                                        }

                                        /*Time E*/
                                        if (datoEntrante >= 50.00 && datoEntrante <=
59.99){
                                                if (timer < 100){
                                                        datoEntrante =
Math::Round((datoEntrante - 50.0), 2);
                                                        this->chartSenal-
>Series["Series5"]->Points->AddXY(timer, datoEntrante);
```

```
                                                        textBox1->Text =
Convert::ToString(datoEntrante);
                                                        if (baseDatos != numCanales)
                                                        {
                                                                sw->Write("Canal E ->
");
                                                                sw-
>Write(Convert::ToString(datoEntrante));
                                                                sw->Write("|");
                                                                baseDatos++;
                                                        }
                                                        else{
                                                                baseDatos = 0;
                                                                sw->WriteLine(" ");
                                                                sw->Write("Canal E ->
");
                                                                sw-
>Write(Convert::ToString(datoEntrante));
                                                                sw->Write("|");
                                                                baseDatos++;
                                                        }
                                                        datosCH5 = datosCH5 +
datoEntrante;
                                                        i5 = i5 + 1;
                                                }
                                                else{
                                                        promedioCH5 =
Math::Round((datosCH5 / i5), 2);
                                                        if (checkBoxCanal6->Checked){
                                                                return;
                                                        }
                                                        else if (checkBoxCanal1-
>Checked || checkBoxCanal2->Checked || checkBoxCanal3->Checked || checkBoxCanal4-
>Checked){
                                                                this->serialPort1-
>DiscardInBuffer(); //Descarta datos del buffer
                                                                this->chartSenal-
>Series["Series1"]->Points->Clear();
                                                                promedioCH1 = 0.0;
datosCH1 = 0.0; i1 = 0; baseDatos = numCanales;
                                                                this->chartSenal-
>Series["Series2"]->Points->Clear();
                                                                promedioCH2 = 0.0;
datosCH2 = 0.0; i2 = 0;
                                                                this->chartSenal-
>Series["Series3"]->Points->Clear();
                                                                promedioCH3 = 0.0;
datosCH3 = 0.0; i3 = 0;
                                                                this->chartSenal-
>Series["Series4"]->Points->Clear();
                                                                promedioCH4 = 0.0;
datosCH4 = 0.0; i4 = 0;
                                                                this->chartSenal-
>Series["Series5"]->Points->Clear();
                                                                promedioCH5 = 0.0;
datosCH5 = 0.0; i5 = 0;
                                                                sw->WriteLine(" ");
                                                                this->serialPort1-
>DiscardInBuffer(); //Descarta datos del buffer
                                                        }
                                                        else{
```

```
                                                                this->chartSenal-
>Series["Series5"]->Points->Clear();
                                                                promedioCH5 = 0.0;
datosCH5 = 0.0; i5 = 0; baseDatos = numCanales;
                                                                this->serialPort1-
>DiscardInBuffer(); //Descarta datos del buffer
                                                        }
                                                }
                                        }


                                        /*Time F*/
                                        if (datoEntrante >= 60.00 && datoEntrante <=
69.99){
                                                if (timer < 100){
                                                        datoEntrante =
Math::Round((datoEntrante - 60.0), 2);
                                                        this->chartSenal-
>Series["Series6"]->Points->AddXY(timer, datoEntrante);
                                                        textBox1->Text =
Convert::ToString(datoEntrante);
                                                        if (baseDatos != numCanales)
                                                        {
                                                                sw->Write("Canal F ->
");
                                                                sw-
>Write(Convert::ToString(datoEntrante));
                                                                sw->Write("|");
                                                                baseDatos++;
                                                        }
                                                       else{
                                                                baseDatos = 0;
                                                                sw->WriteLine(" ");
                                                                sw->Write("Canal F ->
");
                                                                sw-
>Write(Convert::ToString(datoEntrante));
                                                                sw->Write("|");
                                                                baseDatos++;
                                                       }
                                                        datosCH6 = datosCH6 +
datoEntrante;
                                                        i6 = i6 + 1;

                                                }
                                                else {
                                                        promedioCH6 =
Math::Round((datosCH6 / i6), 2);
                                                        if (checkBoxCanal1->Checked ||
checkBoxCanal2->Checked || checkBoxCanal3->Checked || checkBoxCanal4->Checked ||
checkBoxCanal5->Checked){
                                                                this->chartSenal-
>Series["Series1"]->Points->Clear();
                                                                promedioCH1 = 0.0;
datosCH1 = 0.0; i1 = 0; baseDatos = numCanales;
                                                                this->chartSenal-
>Series["Series2"]->Points->Clear();
                                                                promedioCH2 = 0.0;
datosCH2 = 0.0; i2 = 0;
```

```
                                                                this->chartSenal-
>Series["Series3"]->Points->Clear();
                                                                promedioCH3 = 0.0;
datosCH3 = 0.0; i3 = 0;
                                                                this->chartSenal-
>Series["Series4"]->Points->Clear();
                                                                promedioCH4 = 0.0;
datosCH4 = 0.0; i4 = 0;
                                                                this->chartSenal-
>Series["Series5"]->Points->Clear();
                                                                promedioCH5 = 0.0;
datosCH5 = 0.0; i5 = 0;
                                                                this->chartSenal-
>Series["Series6"]->Points->Clear();
                                                                promedioCH6 = 0.0;
datosCH6 = 0.0; i6 = 0;
                                                                sw->WriteLine(" ");
                                                                this->serialPort1-
>DiscardInBuffer(); //Descarta datos del buffer
                                                        }
                                                        else{
                                                                this->chartSenal-
>Series["Series6"]->Points->Clear();
                                                                promedioCH6 = 0.0;
datosCH6 = 0.0; i6 = 0; baseDatos = numCanales;
                                                                this->serialPort1-
>DiscardInBuffer(); //Descarta datos del buffer
                                                        }
                                                }
                                        }
                                }
                                catch (...){
                                        return;
                                }
                        }


                private: System::Void serialPort1_DataReceived(System::Object^  sender,
System::IO::Ports::SerialDataReceivedEventArgs^  e){

                        double arrayDatos[8];

                        try{
                                if (flagActiva){
                                        for (int  x = 0; x <= numCanales + 1; x++){

                                                arrayDatos[x] = Convert::ToDouble(this-
>serialPort1->ReadLine());
                                        }
                                        for (int y = 0; y <= numCanales + 1; y++){

                                                leeDatos(arrayDatos[y]);

                                        }
                                }
                        }
                        catch (...){
                                return;
                        }

                }
```

```
            private: System::Void comboBox1_SelectedIndexChanged(System::Object^
sender, System::EventArgs^  e) {

                    serialPort1->PortName = comboBox1->Text;

                    try{
                           serialPort1->Open();
                    }
                    catch (...){
                           MessageBox::Show("Puerto no v·lido");
                           return;
                    }
            }

      private: System::Void buttonCanales_Click(System::Object^  sender,
System::EventArgs^  e) {

                    checkBoxCanal1->Enabled = false;
                    checkBoxCanal2->Enabled = false;
                    checkBoxCanal3->Enabled = false;
                    checkBoxCanal4->Enabled = false;
                    checkBoxCanal5->Enabled = false;
                    checkBoxCanal6->Enabled = false;
                    buttonGuardar->Enabled = false;

                    //Cuenta el numero de canales
                    numCanales = 0;

                    if (checkBoxCanal1->Checked){
                           numCanales++;
                    }
                    if (checkBoxCanal2->Checked){
                           numCanales++;
                    }
                    if (checkBoxCanal3->Checked){
                           numCanales++;
                    }
                    if (checkBoxCanal4->Checked){
                           numCanales++;
                    }
                    if (checkBoxCanal5->Checked){
                           numCanales++;
                    }
                    if (checkBoxCanal6->Checked){
                           numCanales++;
                    }

                    switch (numCanales){
                    case 1:
                           this->serialPort1->Write("1");
                           break;
                    case 2:
                           this->serialPort1->Write("2");
                           break;
                    case 3:
                           this->serialPort1->Write("3");
                           break;
                    case 4:
                           this->serialPort1->Write("4");
                           break;
```

```
case 5:
	this->serialPort1->Write("5");
	break;
case 6:
	this->serialPort1->Write("6");
	break;
default:
	MessageBox::Show("Escoge un n˙mero de canales valido");
	buttonIniciar->Enabled = false;
	buttonParar->Enabled = false;
	buttonApagar->Enabled = true;
	buttonReset->Enabled = false;
	buttonCanales->Enabled = true;
	buttonGuardar->Enabled = false;
	checkBoxCanal1->Enabled = true;
	checkBoxCanal2->Enabled = true;
	checkBoxCanal3->Enabled = true;
	checkBoxCanal4->Enabled = true;
	checkBoxCanal5->Enabled = true;
	checkBoxCanal6->Enabled = true;
	break;
}

//Habilita los canales
if (checkBoxCanal1->Checked){
	this->chartSenal->Series["Series1"]->Enabled = true;
	escribeA = true;
	this->serialPort1->Write("A");
}
if (checkBoxCanal2->Checked){
	this->chartSenal->Series["Series2"]->Enabled = true;
	escribeB = true;
	this->serialPort1->Write("B");
}
if (checkBoxCanal3->Checked){
	this->chartSenal->Series["Series3"]->Enabled = true;
	escribeC = true;
	this->serialPort1->Write("C");
}
if (checkBoxCanal4->Checked){
	this->chartSenal->Series["Series4"]->Enabled = true;
	escribeD = true,
	this->serialPort1->Write("D");
}
if (checkBoxCanal5->Checked){
	this->chartSenal->Series["Series5"]->Enabled = true;
	escribeE = true;
	this->serialPort1->Write("E");
}
if (checkBoxCanal6->Checked){
	this->chartSenal->Series["Series6"]->Enabled = true;
	escribeF = true;
	this->serialPort1->Write("F");
}
if (numCanales != 0){
	MessageBox::Show("Podemos Comenzar");
	buttonCanales->Enabled = false;
	buttonIniciar->Enabled = true;
	buttonReset->Enabled = true;
	buttonGuardar->Enabled = false;
}
```

```
            }

            private: System::Void buttonIniciar_Click(System::Object^  sender,
System::EventArgs^  e) {

                    baseDatos = 0;
                    timer = 0;
                    sw->WriteLine("");
                    sw->WriteLine("#-----------------------------------------------------
-----------------------------------------#");
                    sw->WriteLine("");

                    this->serialPort1->DiscardInBuffer(); //Descarta datos del buffer

                    chartBarras->Series["Series1"]->Points->Clear();
                    chartBarras->Series["Series2"]->Points->Clear();


                    this->chartSenal->Series["Series1"]->Points->Clear();
                    this->chartSenal->Series["Series2"]->Points->Clear();
                    this->chartSenal->Series["Series3"]->Points->Clear();
                    this->chartSenal->Series["Series4"]->Points->Clear();
                    this->chartSenal->Series["Series5"]->Points->Clear();
                    this->chartSenal->Series["Series6"]->Points->Clear();

                    flagActiva = true;

                    promedioCH1 = 0.0; promedioCH2 = 0.0; promedioCH3 = 0.0;
promedioCH4 = 0.0; promedioCH5 = 0.0; promedioCH6 = 0.0;
                    i1 = 0; i2 = 0; i3 = 0; i4 = 0; i5 = 0; i6 = 0;

                    progressBar1->Value = 100;
                    comboBox1->Enabled = false;
                    buttonCanales->Enabled = false;
                    buttonIniciar->Enabled = false;
                    buttonParar->Enabled = true;
                    buttonReset->Enabled = false;
                    buttonApagar->Enabled = false;
                    buttonGuardar->Enabled = false;
                    this->serialPort1->Write("I");
            }
            private: System::Void buttonParar_Click(System::Object^  sender,
System::EventArgs^  e) {

                    flagActiva = false;
                    progressBar1->Value = 0;
                    comboBox1->Enabled = true;
                    buttonIniciar->Enabled = true;
                    buttonParar->Enabled = false;
                    buttonReset->Enabled = true;
                    buttonApagar->Enabled = true;
                    buttonGuardar->Enabled = false;
                    this->serialPort1->Write("P");
            }

            private: System::Void buttonReset_Click(System::Object^  sender,
System::EventArgs^  e) {
                    checkBoxCanal1->Enabled = true;
                    checkBoxCanal2->Enabled = true;
                    checkBoxCanal3->Enabled = true;
                    checkBoxCanal4->Enabled = true;
```

```
                    checkBoxCanal5->Enabled = true;
                    checkBoxCanal6->Enabled = true;

                    this->serialPort1->Write("R");
                    buttonCanales->Enabled = true;
                    buttonIniciar->Enabled = false;
                    buttonReset->Enabled = false;
                    buttonGuardar->Enabled = true;
             }

             private: System::Void buttonApagar_Click(System::Object^  sender,
System::EventArgs^  e) {
                    this->serialPort1->Write("A");
                    MessageBox::Show("La aplicaciÛn se cerrar·");
                    sw->Close();
                    Application::Exit();
             }

             private: System::Void pictureBoxAnimaciones_Click(System::Object^  sender,
System::EventArgs^  e) {
             }
             private: System::Void chartSenal_Click_1(System::Object^  sender,
System::EventArgs^  e) {
             }
             private: System::Void checkBoxCanal1_CheckedChanged(System::Object^
sender, System::EventArgs^  e) {
             }
             private: System::Void checkBoxCanal2_CheckedChanged(System::Object^
sender, System::EventArgs^  e) {
             }
             private: System::Void checkBoxCanal3_CheckedChanged(System::Object^
sender, System::EventArgs^  e) {
             }
             private: System::Void checkBoxCanal4_CheckedChanged(System::Object^
sender, System::EventArgs^  e) {
             }
             private: System::Void checkBoxCanal5_CheckedChanged(System::Object^
sender, System::EventArgs^  e) {
             }
             private: System::Void checkBoxCanal6_CheckedChanged(System::Object^
sender, System::EventArgs^  e) {
             }


};

}
```

Programa BiorretAOD.cpp

```
#include "MyForm.h"

using namespace System;
using namespace System::Windows::Forms;
[STAThread]
int main(array<String^>^ arg) {
	Application::EnableVisualStyles();
	Application::SetCompatibleTextRenderingDefault(false);
	Proyecto_Alfa::MyForm form;
	Application::Run(%form);
}
```