UNIVERSIDAD NACIONAL de MAR DEL PLATA

FACULTAD DE INGENIERIA

Sistema Medidor de Conductividad en Soluciones Acuosas

Ana Paula Pucheu

20 de Diciembre de 2019

INDICE

Introducción

El proyecto final surgió a partir de una selección entre varias propuestas ofrecidas por el Laboratorio de Instrumentación y Control (LIC). La idea fue desarrollar un dispositivo que permita colaborar con la UNMdP, a modo de devolución, luego de haber transitado la formación en la institución; permitiendo volcar mis conocimientos en el desarrollo de un instrumento de medición para la Planta Piloto de Ingeniería Química de la Universidad. Un conductímetro que, además de habilitar la integración de los conocimientos teóricos adquiridos durante la carrera, permite reforzar el manejo de hardware y software con la supervisión de un docente calificado.

En este trabajo se diseñó e implementó un dispositivo para la medición de conductividad eléctrica de soluciones acuosas, el cual servirá para medir destilados de la Planta Piloto, para su uso cotidiano. Es por este motivo que se investigaron diferentes métodos que no requieren de una solución estándar para su calibración.

El instrumento diseñado se basa en la técnica de Van der Pauw; es una técnica de cuatro electrodos para la medición de conductividad de un sólido. En una investigación realizada por Moron, se demostró que el método de Van der Pauw puede ser utilizado para la medición de conductividad de soluciones electrolíticas.

Capítulo 1: Introducción Teórica de Conductancia y su Medición

A continuación se explicara la diferencia entre dos conceptos que aunque parezcan similares, es importarte resaltar su diferencia. Estos dos conceptos son la conductividad y la conductancia.

La conductancia eléctrica es la propiedad inversa de la resistencia. Nos permite determinar la corriente que circula por el conductor cuando se aplica entre sus extremos una diferencia de potencial conocida. Depende tanto de la sustancia del conductor como de sus dimensiones y se mide en Siemens [S]. La conductancia especifica o conductividad de un conductor, es la conductancia que ofrece el material cuya longitud y sección son iguales a la unidad; se expresa en Siemens por metro [Siemens/m].

La conductividad eléctrica es una propiedad que no puede ser medida directamente; debe ser calculada a partir de la medición de la relación con la resistencia R como muestra la Ecuación 1:

$$R = \rho\left(\frac{l}{A}\right) = \left(\frac{1}{\kappa}\right)\left(\frac{l}{A}\right) = \frac{G}{\kappa}$$

Ecuación 1

Donde R es la resistencia, ρ es la resistividad eléctrica, κ es la conductividad eléctrica, l es la longitud efectiva de la trayectoria de la corriente, A es la sección transversal efectiva de la trayectoria de corriente, y G es el factor de célula.

Las celdas de conductancia pueden ser o no calculables. Una celda calculable es aquella cuyo factor de forma es constante y puede ser calculado a partir de la geometría de la celda. Su principal ventaja es que no requieren proceso de calibración y pueden ser utilizadas como estándar primario.

La medición de conductividad electrónica tiene diversas aplicaciones, como el monitoreo de aguas, determinación de sustancias, control de salinidad del mar entre muchas otras aplicaciones. Hay una serie de parámetros que siempre se controlan en el agua para uso de laboratorio como por ejemplo la conductividad o resistividad, la dureza y el pH entre otros. La Planta Piloto de la Facultad de Ingeniería Química de la UNMdP propone el diseño de un conductímetro para soluciones acuosas.

Diagrama en Bloque: Instrumento de medida.

Para obtener una medida de la conductividad eléctrica de cualquier solución es necesario aplicar una señal eléctrica, sensar la respuesta ante dicha señal y tener en cuenta la geometría de la muestra del cuerpo o sustancia en estudio.

Es por este motivo que se diseñó un dispositivo que genera una señal de tensión sinusoidal, con una frecuencia fija para que los electrodos no se polaricen. Esta señal de tensión, es amplificada y convertida en una corriente de igual frecuencia y amplitud constante para el rango de medición del instrumento. La misma es aplicada a la solución electrolítica contenida en la celda de conductancia. La señal de tensión de respuesta es medida a través de una etapa de sensado, filtrado y detección del valor eficaz. Finalmente, las señales de corriente aplicada y tensión sensada se procesan con un microcontrolador (µC) el cual envía los resultados a una pantalla LCD donde se visualiza la medida de conductividad electrolítica. El dispositivo cuenta con dos conectores USB; uno para la alimentación y otro para transmitir los datos calculados a una computadora.

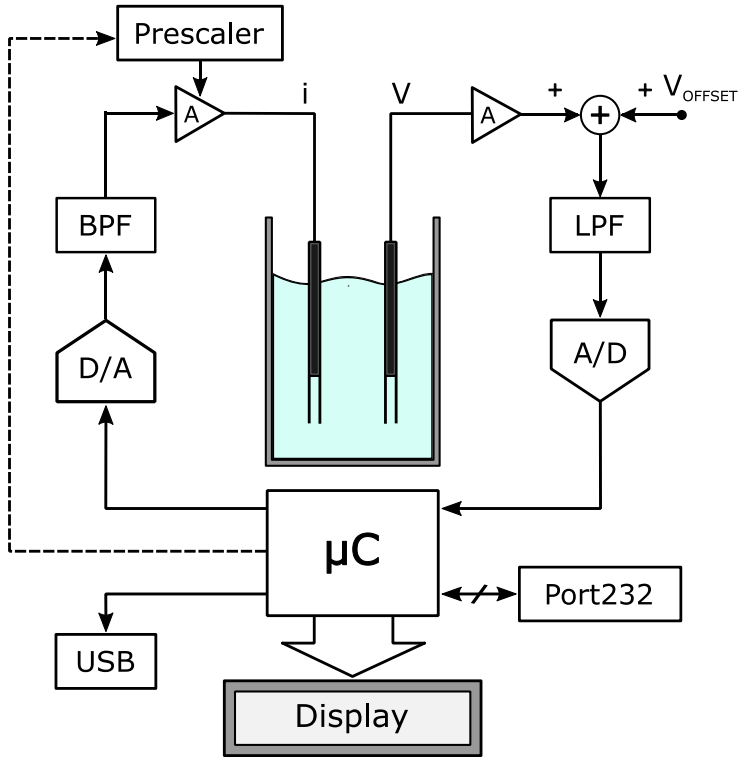

Figura 1 - Diagrama en Bloques del conductímetro

Fundamento Teórico: Métodos para la medición de conductividad

Existen diversos métodos de medición para determinar la conductividad eléctrica, tales como el método de dos hilos, de cuatro hilos y el Método de Van der Pauw, que fueron desarrollados para el estudio de distintos materiales.

Método de 2-hilos

Consiste en la utilización de dos electrodos ubicados en los extremos de la muestra para la aplicación de una señal conocida de corriente. Con el uso de un voltímetro se mide la corriente en la muestra y así se calcula su resistencia; además de ello debe contar con un área transversal uniforme y longitud conocida para facilidad de la determinación del factor geométrico. El objetivo es medir sólo la resistencia de la solución (R_{sol}). Sin embargo, la resistencia R_e, causada por la polarización de los electrodos y los efectos de campo, interfiere en la medición y se mide tanto R_{sol} como R_e. Además, necesita una solución estándar para su calibración.

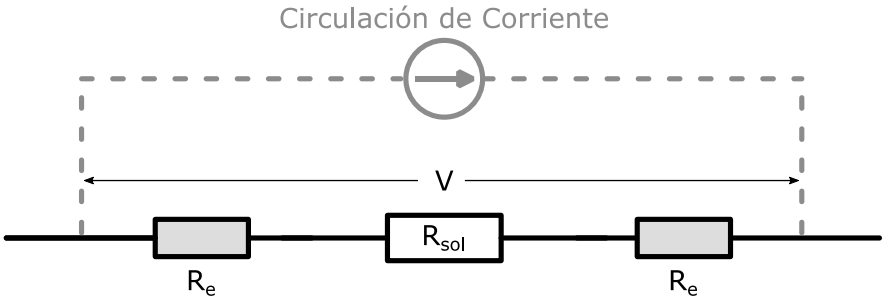

Figura 2 - Esquema Simplificado de una Célula de Conductividad de 2-Hilos

Método de 4-hilos

Este método requiere del uso de 4 electrodos, dos de ellos para aplicar la señal de corriente, y los otros dos para sensar la respuesta en tensión. Es tan importante la forma de la muestra, como la posición de los electrodos. También necesita una solución primaria para ser calibrado antes de su utilización.

Se muestra un diagrama simplificado de la célula en la Figura 3, donde una corriente conocida es aplicada entre los electrodos externos (1 y 4), generando una diferencia de tensión entre los electrodos internos (2 y 3). Dicho valor, que se mide entre las líneas de corriente, se mantiene constante. La medición de tensión se realiza con un voltímetro con una resistencia interna del orden de los 10 MΩ; es por este motivo

que prácticamente no circula corriente por dicha rama, permitiendo despreciar la resistencia R_e. De esta manera, el método de medición a cuatro hilos permite medir resistencias con mayor precisión, ya que elimina la contribución de las resistencias de cableado en la medición y de los potenciales de contacto. Finalmente la conductividad será directamente proporcional a la corriente aplicada.

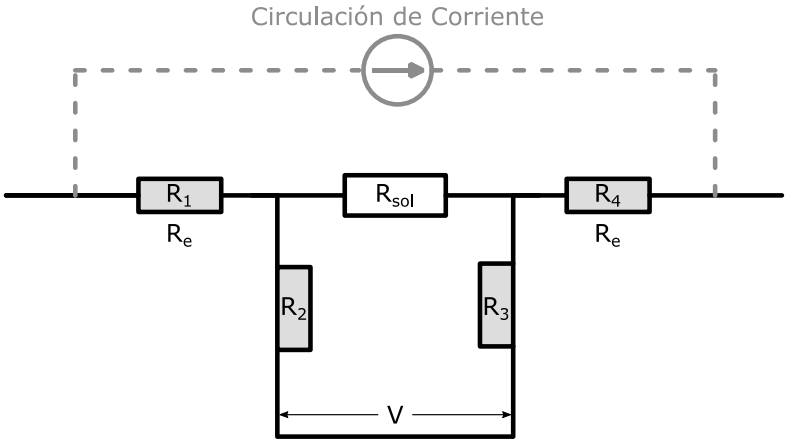

Figura 3 - Esquema Simplificado de la Célula de Conductancia de 4 Hilos

Técnica de Van der Pauw

Se propone una variación del método de 4 Hilos con la ventaja de que permite la medición de forma arbitraria con espesor uniforme, sin necesidad de ser calibrada.

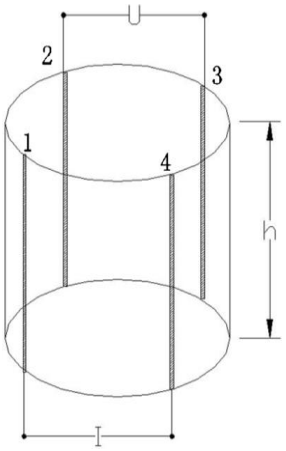

Figura 4 - Medición de la Conductividad Electrolítica de una Solución usando la teoría de Van der Pauw

El método de Van der Pauw propone una estructura de cuatro electrodos cuando se mide la resistividad y el efecto Hall de los materiales sólidos, con un grosor uniforme y una forma arbitraria. Luego, Moron demostró que este método también se puede

aplicar a la medición absoluta de la conductividad electrónica de una solución acuosa.

El principio básico de la teoría de Van der Pauw propone: una solución que se encuentra contenida en un cilindro de altura h (Figura 4), cuatro electrodos de metal ubicados en la circunferencia del cilindro, paralelos al eje vertical. Los electrodos 1 y 4, transportan la corriente eléctrica I_{14} de la solución, y al mismo tiempo se mide la diferencia de tensión U_{23} entre los electrodos restantes 2 y 3. De acuerdo a la teoría de Van de Pauw, se establece la Ecuación 2:

$$e^{(-\sigma\pi h R_{14,23})} + e^{(-\sigma\pi h R_{12,34})} = 1$$

Ecuación 2

Donde σ es la conductividad eléctrica de la solución y $R_{14,23} = V_{2,3}/I_{1,4}$ se define como la resistencia equivalente de la solución, cuando los electrodos 2 y 3 son los que miden la diferencia de potencial y los electrodos 1 y 4 son los electrodos que transportan la corriente, similarmente obtenemos $R_{12,34} = V_{3,4}/I_{1,2}$. Llamaremos $\overline{R}$ a la resistencia equivalente de la solución, la cual puede ser expresada como:

$$\overline{R} = \frac{(R_{12,34} + R_{14,23})}{2} f\left(\frac{R_{12,34}}{R_{14,23}}\right)$$

Ecuación 3

Siendo f es una función que depende de la relación $R_{12,34}/R_{14,23}$ (ver Figura 5). Cuando $R_{12,34}$ y $R_{14,23}$ son casi iguales, la función f se puede expresar de la forma:

$$f \approx 1 - \left(\frac{R_{12,34} - R_{14,23}}{R_{12,34} + R_{14,23}}\right)^2 \frac{\ln(2)}{2} - \left(\frac{R_{12,34} - R_{14,23}}{R_{12,34} + R_{14,23}}\right)^4 \left\{\frac{\ln(2)^2}{4} - \frac{\ln(2)^3}{12}\right\}$$

Ecuación 4

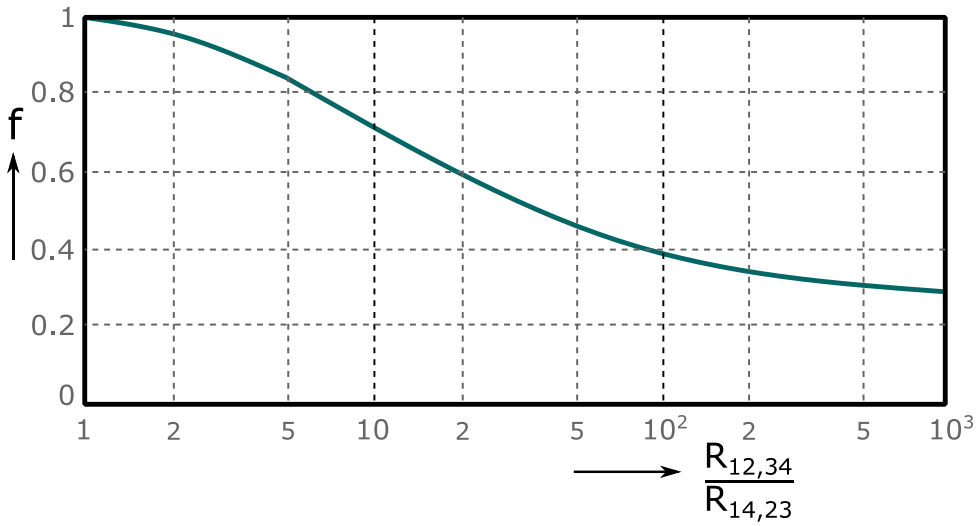

Figura 5 - Función f utilizada para determinar la resistividad específica, en función de $R_{12,34}/R_{23,14}$

Cuando la configuración de los 4 electrodos de metal son perfectamente simétricos, se cumple que $R_{12,34} = R_{14,23} = \overline{R}$ y $f = 1$, la Ecuación 2 puede ser expresada como indica la siguiente ecuación:

$$\sigma = \kappa . G = \frac{\ln(2)}{\pi h}\frac{1}{\overline{R}} = \frac{\ln(2)}{\pi h}\frac{1}{R_{12,34}} = = \frac{\ln(2)}{\pi h}\frac{1}{R_{14,23}}$$

Ecuación 5

Donde κ es el factor de célula, $\ln(2)/\pi h$, G es la conductancia de la solución.

A partir de la Ecuación 4, para una configuración de electrodos simétricos cuando $R_{12,34} = R_{14,23} \pm 1\%$, el error causado en lo más alto de la función f es provocado por el error en la posición, el cual es menor que 0.001%; es por este motivo que puede ser ignorado. La conductividad eléctrica de la solución puede ser calculada midiendo la longitud de los electrodos y la resistencia equivalente de la solución. La constante de célula $\kappa = \frac{\ln(2)}{\pi h}$, la cual depende sólo de la altura h de los electrodos, puede ser calculada más precisamente durante el proceso electrolítico de medición de la conductividad.

Sin importar el método que se utilice, la medición es válida sólo cuando el factor de célula es constante, es decir, las trayectorias de corriente no varían con las propiedades eléctricas del líquido, de los electrodos y/o del recipiente contenedor de la solución. Es por este motivo que los dos primeros métodos son de baja precisión;

además de que es necesario contar con una solución estándar (con una conductividad conocida) para la calibración del dispositivo, mientras que en el método de Van der Pauw se puede usar para la determinación de la conductividad de soluciones estándar.

Se asume que la constante de célula definida está dada para una determinada célula, a una determinada temperatura, la cual toma un valor no variable. La mayoría de las células de conductividad no cumplen estos requerimientos. No poseen una geometría teóricamente calculable y sus constantes varían, ambas en relación a la conductividad y frecuencia medida (esta variación es debida a la polarización de los electrodos y los efectos eléctricos parásitos). Es por este motivo que las células requieren calibración a condiciones cercanas a las que se desean medir. Para ello se utiliza una solución con conductividad eléctrica estándar. Una alternativa es calibrar comparando con una célula que posea una constante conocida.

En este trabajo se propone usar una célula calculable para la determinación de conductividad eléctrica de soluciones estándar, las cuales pueden ser utilizadas como una célula de conducción para le medición de cualquier tipo de solución; éstas debería cumplir las siguientes características:

- bien definida, con geometría calculable.
- valor de constante de célula invariable, independiente del medio y las condiciones de medición que consiste en asegurar un patrón de caminos de corriente y distribución de potencial constante.
- buena precisión de la conductividad del medio medido.
- valor de constante de célula es sensible ante cambios de temperatura.

La célula de Van der Pauw tiene la característica principal que su constante de célula que depende solo de la altura h (para el arreglo simétrico de electrodos, solo una medición es suficiente). Resulta de su estructura geométrica, que cuando asegura altura homogénea h, entrega una distribución de potencial paralela en el espacio de medición de la solución. Por este motivo se resuelve el problema de la determinación de la constante de célula, a partir del análisis del campo a través de una sección plana de la célula y puede resolverse utilizando la representación conforme y la teoría de potencial complejo. En el tipo de célula del método de Van de Pauw, los electrodos tienen anchos muy pequeños comparados con el radio de

circunferencia de la célula y está basado en el método de 4 puntas .Una virtud de este método es que la célula es ligeramente susceptible a la influencia de impedancia de los electrodos y la no homogeneidad de la superficie de los electrodos.

Consideraciones para la medición de soluciones electrolíticas

Para disminuir efectos de electrólisis descritos en las leyes de Faraday, se recomienda inyectar una señal con una frecuencia de alrededor de 1kHz, para medir la conductividad afectando lo menor posible la solución. Otro factor importante es el efecto capacitivo en la interfaz electrodo electrolito, lo que provoca variación de la conductividad medida en función de la frecuencia. Dicho efecto se debe al exceso de cargas, ya sean electrones, iones o dipolos, en la interfaz produciendo un campo eléctrico. Además de ello, debe realizarse una corrección por temperatura ya que ella influye en la conductividad de la solución.

El método de prueba de 4 puntas elaborado por Van der Pauw, originalmente aplicado para determinar la resistividad de materiales semiconductores, puede ser también aplicado en la medición de conductividad eléctrica en conductores iónicos, especialmente para soluciones electrolíticas. Conocer los patrones de los caminos de corriente en una muestra de forma arbitraria es innecesario si la muestra posee espesor homogéneo, los electrodos tienen dimensiones despreciables y se encuentran en la circunferencia de la muestra, y la superficie de la muestra está conectada individualmente.

Transferir el método de medición de un sólido a una solución iónica requiere consideraciones de factores adicionales, como la influencia de los electrodos, la impedancia del electrodo y el grosor relativamente grande de la muestra. El potencial del electrodo inmerso en la solución, en general, no es igual al potencial del electrodo antes de la inmersión. Dos electrodos idénticos, hechos con el mismo material, generan idéntica distorsión de la distribución de potencial y de ahí ellos pueden compensarse el uno al otro. De todas maneras, en la práctica, es difícil asegurar idénticos electrodos. Por ese motivo es ventajoso minimizar su superficie. Algunas desviaciones de las suposiciones del método, por ejemplo las dimensiones finitas de los electrodos y la ubicación no ideal de los mismos sobre la

circunferencia, pueden ser calculadas teóricamente. Otros efectos pueden ser modelados o determinados experimentalmente.

En la realización práctica de la célula, los electrodos tienen una forma de barras delgadas colocadas cerca de la pared del cuerpo de célula o tiras muy finas que se obtienen a partir de corte de surcos en la pared. La segunda opción es más ventajosa, se adapta mejor a las condiciones del método y los electrodos tienen una ubicación fija. También hay una tercera solución posible; consiste en ubicar los electrodos en espacios individuales, separados del recipiente de medición a través de ranuras angostas de espesor “s”.

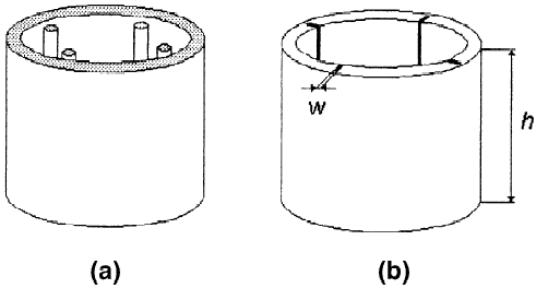

Figura 6 -Células cilíndricas de 4 electrodos en forma de: (a) barras delgadas, (b) tiras muy finas.

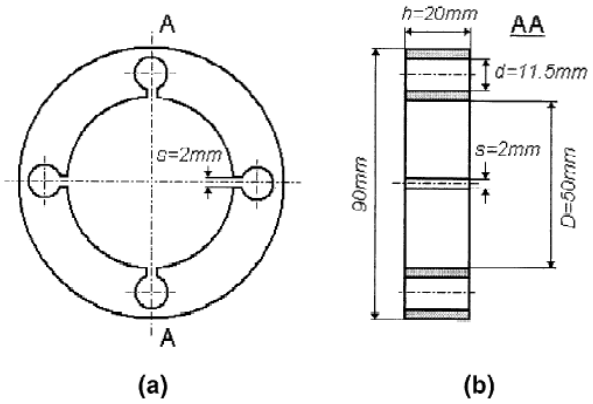

Figura 7 - Cuerpo de la célula del nuevo diseño (III): (a) vista superior, (b) sección transversal ($\Delta h = \pm 0.05$ mm).

No es necesario cambiar el circuito de medición para una configuración determinada de electrodos -la medición puede ser realizada simultáneamente con dos configuraciones distintas- aplicando dos frecuencias medibles. Tiene implementación práctica significativa porque cambiar los electrodos puede ser dificultoso.

Celda de medición

Se diseñó una celda tipo Moron basada en la técnica de Van der Pauw para soluciones. Dicha celda fue seleccionada debido a que no necesita calibración, ni requiere de un proceso de construcción preciso. Ha demostrado ser la mejor celda calculable diseñada hasta el momento y puede ser empleada tanto en procedimientos de medida cotidianos, como en la determinación absoluta de conductividad eléctrica, es decir, como estándar primario.

Cuadro Comparativo de los Métodos de Medición

A continuación se analiza, a través de un cuadro comparativo, las ventajas y desventajas de los métodos mencionados.

Método	Ventajas	Desventajas
2 Hilos	*Fácil de mantener* *Económico* *Usado para medios viscosos o muestras con suspensión*	*La corriente se expande en todas las direcciones posibles, según las propiedades eléctricas la solución y la proximidad de los electrodos a las paredes de la celda, y el factor de célula no es constante.* *Las células deben estar ubicadas en el centro de la solución.* *Polarización en muestras con alta conductividad.* *Necesita una solución primaria con un valor de conductividad similar a la que se desea medir para calibrar el instrumento.*
4 Hilos	*Lineal en un gran rango de conductividad.* *Células por inmersión.* *Bueno para soluciones con alta conductancia.* *Calibración y medición de distintos rangos.* *Reduce efecto de resistencia del cable respecto del método de 2 puntas.*	*Necesita una inmersión de 3 o 4 cm.* *No sirve para muestras cargadas.* *Posee errores de geometría.*
Van der Pauw	*Las celdas de conductancia calculables son una muy buena alternativa para su utilización en situaciones prácticas, así como estándar primario.* *Permite el uso de bajas frecuencias para medición* *No se requiere platinar los electrodos.* *Elimina la necesidad de usar soluciones de calibración.* *No requiere de un proceso de fabricación extremadamente preciso* *Disminuye los efectos relacionados con la interfaz electrodo-electrolito.*	*Se usa para sólidos de forma aleatoria y se adapta a soluciones acuosas si se considera la corriente de Faraday y la interfaz electrodo electrolito.*

Capítulo 2: Diseño e Implementación del Sistema de Medición

Fuente de Corriente

Una vez definido el método de medición, se procede al estudio de las distintas fuentes de corriente. Es importante generar una corriente alterna para evitar que los electrodos se polaricen.

Para el diseño del equipo es necesario contar con una fuente de corriente que tenga una alta impedancia de salida y que presente una apropiada respuesta en frecuencia junto con la estabilidad antes posibles oscilaciones.

En esta etapa el dispositivo se emplea la señal de tensión senoidal generada por el microcontrolador (μC); la cual se aplica a la entrada de una fuente que genera una corriente senoidal de la misma frecuencia que la generada por el μC y una amplitud constante prefijada. La corriente que genere esta fuente debe estar bien controlada y definida ya que en base a esta se realizara el procesamiento para calcular la conductancia.

El análisis de la resistencia de salida ratifica la definición de fuente de corriente, en donde la resistencia de salida de una fuente de corriente ideal es infinita. Por lo tanto, una fuente de corriente puede modelarse idealmente como una fuente de corriente en paralelo con la impedancia de la fuente Z_s, como se muestra en la Figura 8. Nos interesa tener una fuente de corriente con una impedancia relativamente alta para que cuando se modifique la impedancia de carga en el rango esperado la corriente que se inyecta no cambie significativamente. Esta circunstancia puede observarse de forma clara a través de la Ecuación 6.

$$I_L = I_s \left(\frac{Z_s}{Z_s + Z_L} \right)$$

Ecuación 6

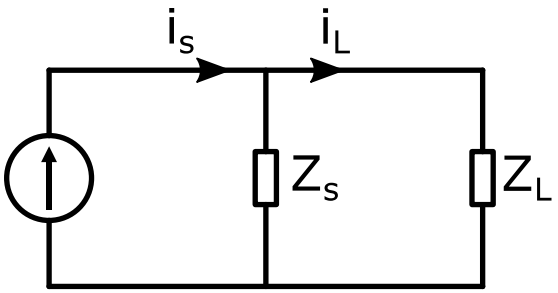

Figura 8 - Modelado de una fuente de corriente.

En base a los valores de conductancia que se desean medir con el conductímetro y la corriente que se debe inyectar es del orden de los 10 mA a una frecuencia de 200 Hz.

A continuación explicare los distintos tipos de fuentes de corriente que se investigaron:

Fuente de Corriente Howland

La propuesta, en base a la bibliografía consultada, fue de utilizar una fuente de corriente Howland. El circuito de dicha fuente se muestra en la Figura 9.

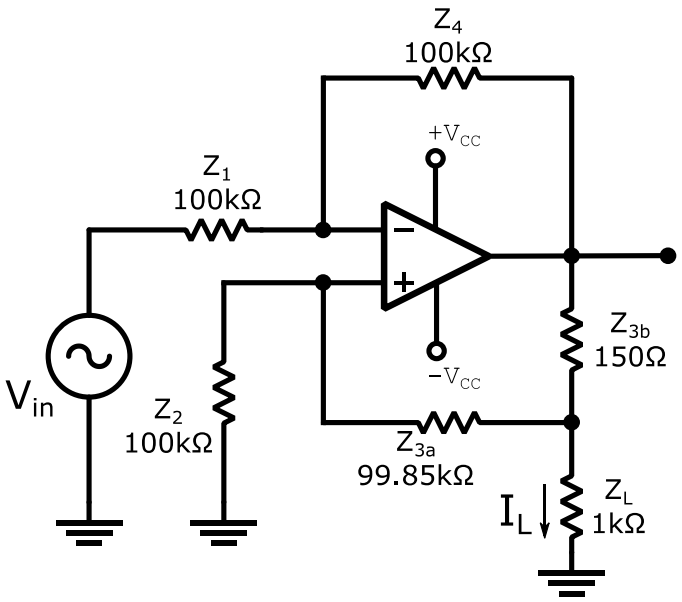

Figura 9 - Fuente de corriente Howland simulada

Una de las condiciones para el correcto funcionamiento de la fuente de corriente Howland es que las resistencias de carga Z_L debe ser mucho menor en relación al resto de las resistencias de la fuente ($Z_L << Z_1$, Z_2, Z_{3a}+ Z_{3b}, Z_4). De esta forma se consigue tener una corriente en la carga que sólo depende de la fuente de tensión

de entrada V_{in} y de la resistencia Z_{3b}. Asumiendo que lo mismo es posible, ya que Z_L=1 kΩ y la máxima impedancia, se procede a calcular $Z_{3b} = V_{in}/I_{L\ Deseada}$ = 150Ω.

Una vez determinado el rango de impedancia que presenta la solución, se procede a simular la fuente de corriente Howland asumiendo un AO ideal (ganancia infinita y ancho de banda infinito).

$$I_L = \frac{Z_4(Z_2 + Z_{3a})}{Z_1.Z_{3b}(Z_2 + Z_{3a}) + Z_L(Z_1Z_{3a} + Z_1Z_{3b} - Z_2Z_4)} \cdot V_{in}$$

Ecuación 7

Cumpliendo los criterios de diseño propuestos en la bibliografía, los cuales son: $Z_1 = Z_4$ y $Z_2 = Z_{3a} + Z_{3b}$ en base a la Ecuación 7, que relaciona la corriente en la carga (I_L) con la tensión de entrada (V_{in}), se concluye en la Ecuación 8.

$$I_L = -\frac{1}{Z_{3b}} \cdot V_{in}$$

Ecuación 8

A partir del valor aproximado de corriente de carga deseado de 10mA y el valor de la fuente de tensión V_{in}=1.5 Volt, se calculó $Z_{3b} \approx$ 150Ω en base a la relación indicada previamente. Con lo cual adoptó Z_2=100KΩ y determinó Z_{3a}= 99.85 KΩ

Siguiendo las indicaciones de la bibliografía, se adoptó valores de impedancia Z_1= Z_4= 100KΩ y se procedió a realizar la simulación del circuito mostrando sus resultados en la Figura 10 y Tabla 1.

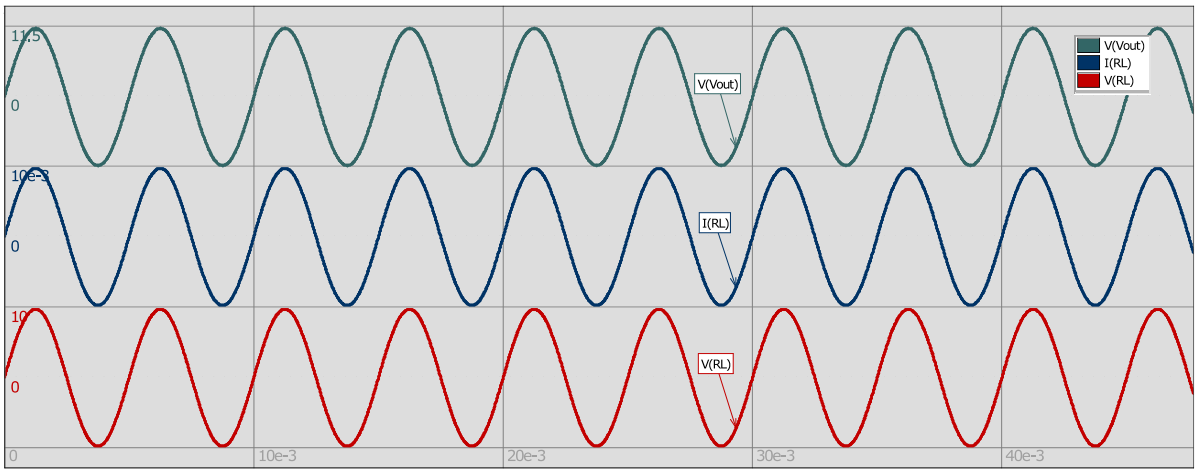

Figura 10 - Simulación de entradas y respuesta a la fuente Howland

	min	max	pp	freq
Cursors				
V(Vout)	-11.5063	11.5063	23.0126	200
I(RL)	-9.99991e-3	9.99991e-3	19.9998e-3	200
V(RL)	-9.99991	9.99991	19.9998	200

Tabla 1 - Valores obtenidos en la simulación de la fuente Howland

Se observó en la simulación que para una R_L mínima de 1.15kΩ, la tensión que tendría a la salida del OP07 sería de 11.5Volt, para entregar una corriente de 10mA en la carga. Mientras que en la hoja de datos del componente, indica que la máxima tensión que el AO podría entregar para esa carga es de ±10Volt, recortando así la forma de onda. Es por este motivo que se descartó la posibilidad de utilizar una fuente de corriente Howland en el proyecto.

Fuente de Corriente con amplificador clase AB

A continuación se planteó otra fuente de corriente (amplificador de audio de baja potencia) como que se muestra a continuación:

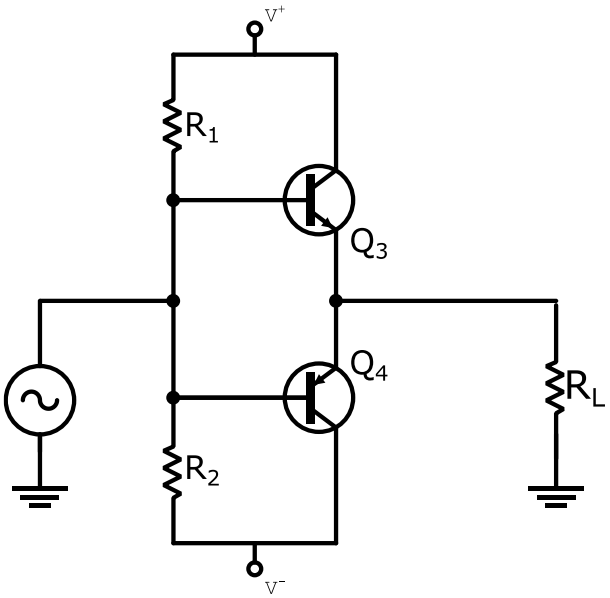

Figura 11 - Circuito de fuente de corriente con amplificador clase B

Para $R_L = 1k\Omega$ (propone máxima corriente), una caída de tensión de $V_L = 10Volt$ y $\beta = 50,$ se determinó una corriente sobre la carga $i_L = {}^{10V}/_{1k\Omega} \cong 10\,mA$, la cual es la que atraviesa el colector. Siendo $i_b = \frac{1}{\beta} \cdot i_c$ y $i_T = 2.5\, i_b$, se calculó el valor de las resistencias $R_1 = R_2 = \frac{(10V - 0.7V)}{2.5 \cdot i_b} = 18.6\ k\Omega$.

La simulación de la respuesta temporal del circuito obtenido se observa en la Figura 12

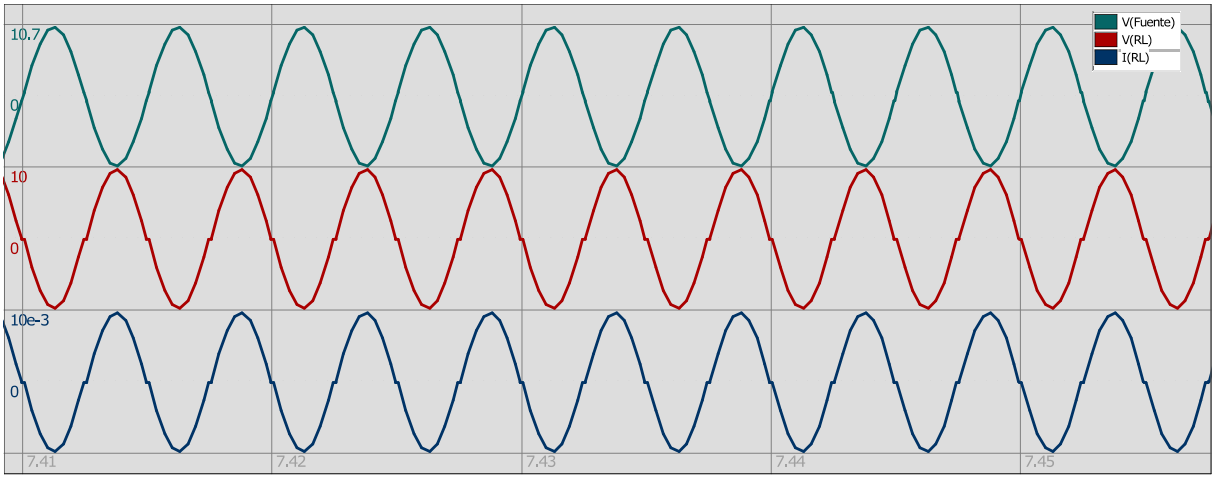

Figura 12 - Simulación de entrada y respuesta de la fuente de corriente con amplificador clase B

En la gráfica temporal se observa que se produce una distorsión de cruce por cero, debido a que durante la transición de la señal de positiva a negativa (o viceversa) existe una no linealidad en la señal de salida.

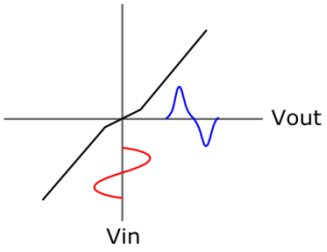

Figura 13 - Relación temporal distorsión de cruce por cero

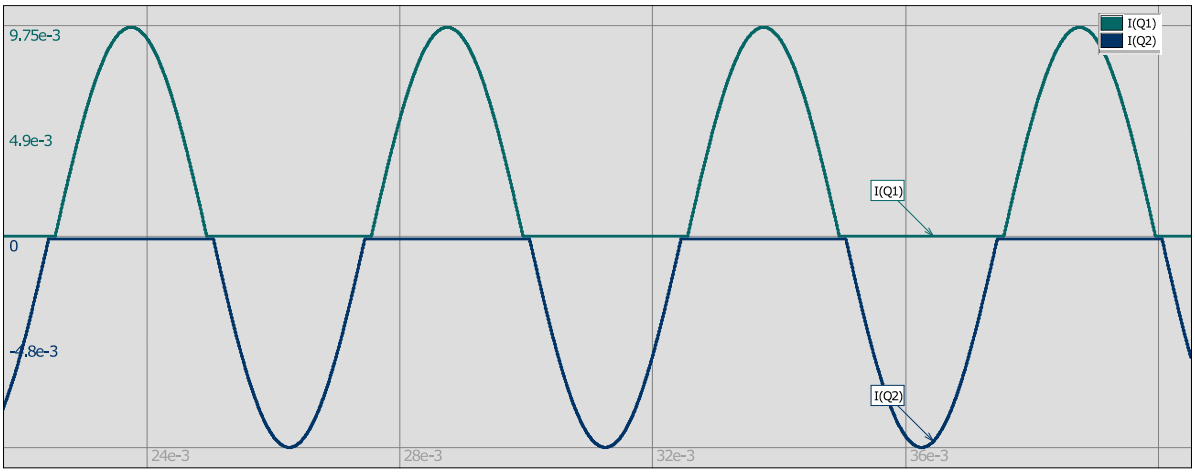

Figura 14 - Simulación CON distorsión de cruce por cero en corriente de salida.

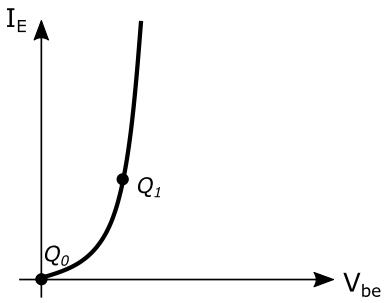

Figura 15 - Curva no lineal de un transistor

Si los transistores permanecen en corte, la corriente en R_L empieza a manifestarse desde que el transistor comienza a conducir; como V_{in} se aplica entre la base y el emisor, la zona no lineal de la característica de este diodo se hace manifiesta debido a que la situación de su punto de reposo Q_0. En cambio, si el punto de trabajo se desplaza hasta Q_1, esta zona se excluye de la región de trabajo normal de los transistores, y cuando la señal de entrada los excita, estos trabajan en la región totalmente lineal de su característica, dando como resultado la eliminación de la distorsión, la cual se produce en el momento del cruce de la conducción de un transistor a la del otro.

Este problema se resuelve mediante la utilizacion de transistores los cuales generan una pequeña caída de tension que compensa esta caida que se presenta entre la base y el emisor. de esta manera, los transistores ya no trabajan en clase B, sino en clase AB.

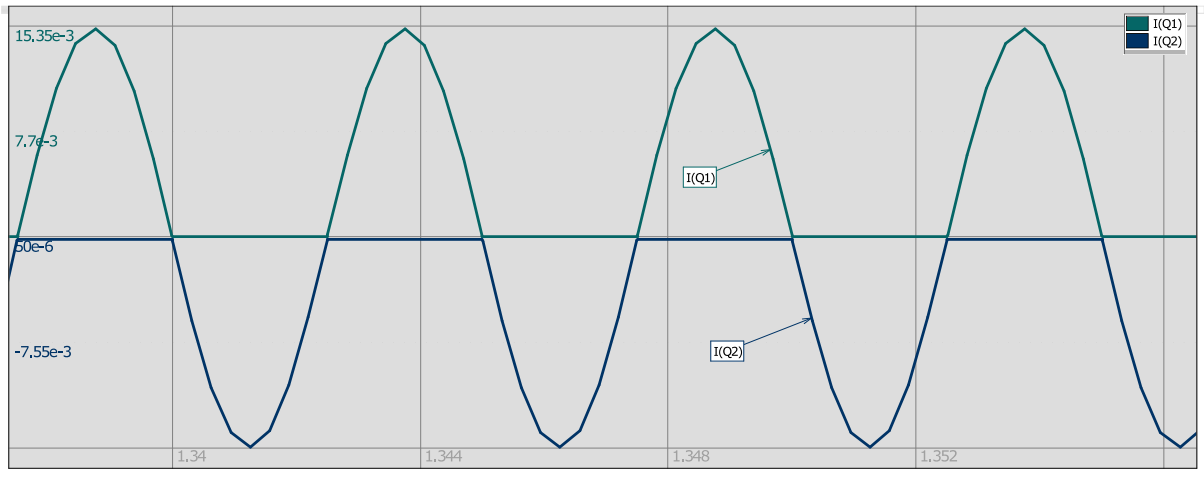

Figura 16 - Simulación SIN distorsión de cruce por cero en corriente de salida.

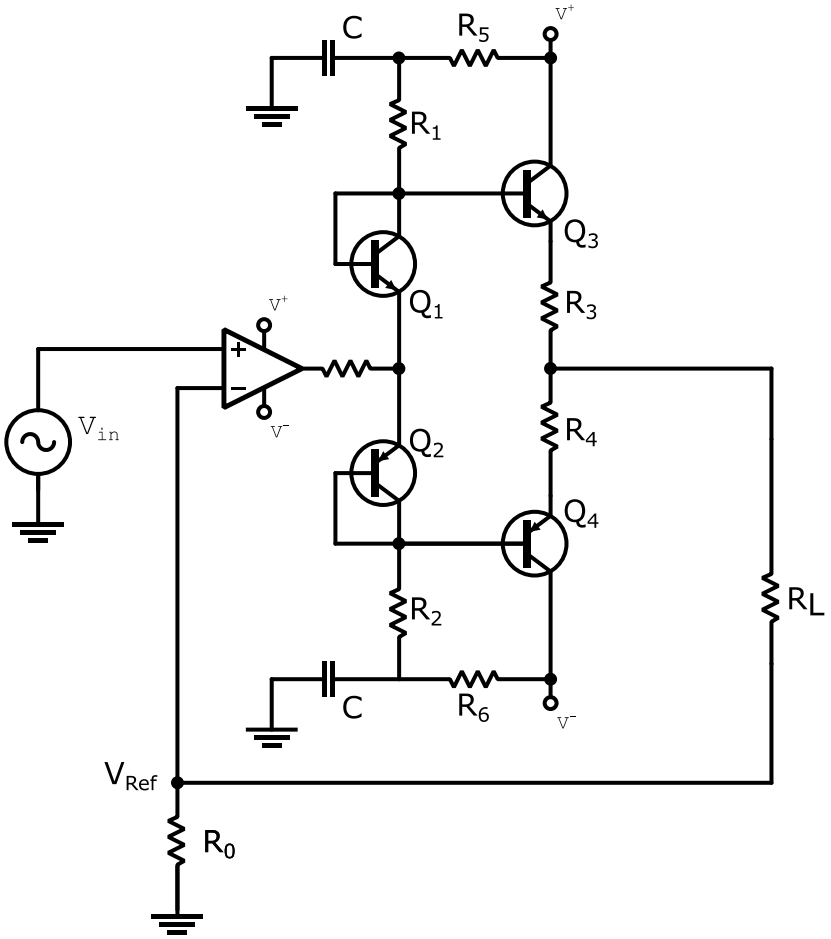

Figura 17 – Fuente de Corriente realimentada

Se agregaron componentes con el fin de mejorar la performance. Las resistencias R_3 y R_4 permiten estabilizar la corriente frente a la temperatura y protegen ante imperfecciones del TBJ. Luego, R_5 y C, al igual que R_6 y C, forman un filtro de alta frecuencia, permitiendo pasar sólo la tensión de alimentación ±Vcc.

Sabiendo que los valores de resistencia que presenta la sustancia a medir se encuentran aproximadamente entre 1KΩ y 1MΩ, se fija la caída de tensión máxima que se quiere tener en la sustancia ($V(R_L)_{máx}$= 10Volt) y se calcula la corriente mínima que circula, a partir de la $R_{L\ máx}$ = 1MΩ. La corriente que circula es de 10μA. Esta será la corriente que se desea tener en esta rama. Se calcula el valor que debe tomar R_0, sabiendo que la caída de tensión en dicha resistencia es la tensión de la pata V^- del AO (senoidal de 1.5 Volt de amplitud pico aproximadamente) y el valor de la corriente que la atraviesa, se calcula el valor de $R_{0\ máx}$ = 150KΩ.

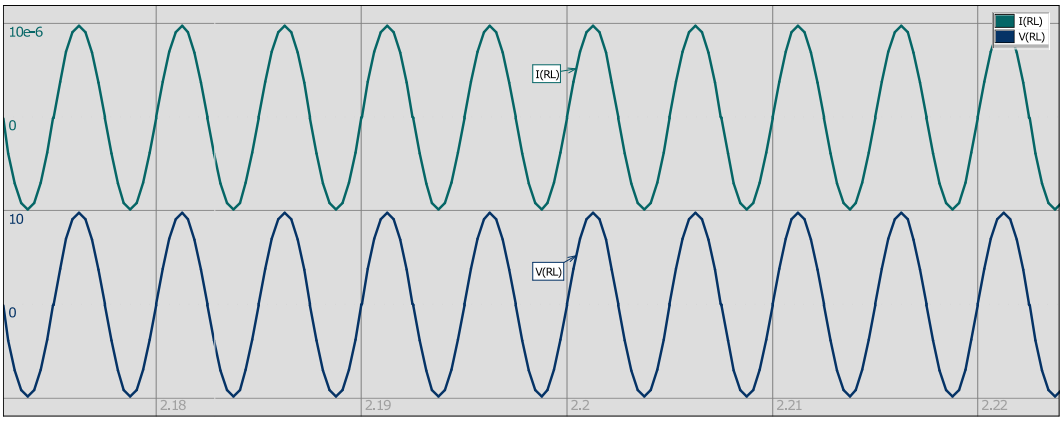

Figura 18 - Simulación de formas de onda para configuración R_L =1MΩ y R_0=150KΩ

	min	max	pp	freq
Cursors				
I(RL)	-9.99982e-6	9.99982e-6	19.9996e-6	200
V(RL)	-9.99982	9.99982	19.9996	200

Tabla 2- Valores obtenidos en la simulación de la configuración R_L =1MΩ y R_0=150KΩ

Si se fija un único valor para R_0, generando una corriente de 10μA, y surge la necesidad de medir sustancias que presenten una resistencia menor a 100KΩ, la caída de tensión en la carga sería del orden de los milivolt. Es por este motivo que se diseña un juego de resistencias R_0 controladas por relees, permitiendo calibrar 3 rangos de medición según la sustancia. Limitando así el rango de tensión que cae en la carga (10~1Volt).

Se realiza el mismo procedimiento para el cálculo de $R_{0\ mín}$. Se fija la mínima caída de tensión que se quiere tener en la sustancia $V(R_L)_{\ mín}$ =1 Volt y conociendo el valor de $R_{L\ mín}$=1KΩ, se obtiene la corriente en la rama. Luego $I_{máx}$ = 1mA y la resistencia R_0 que fija dicha corriente debe ser de 1.5KΩ.

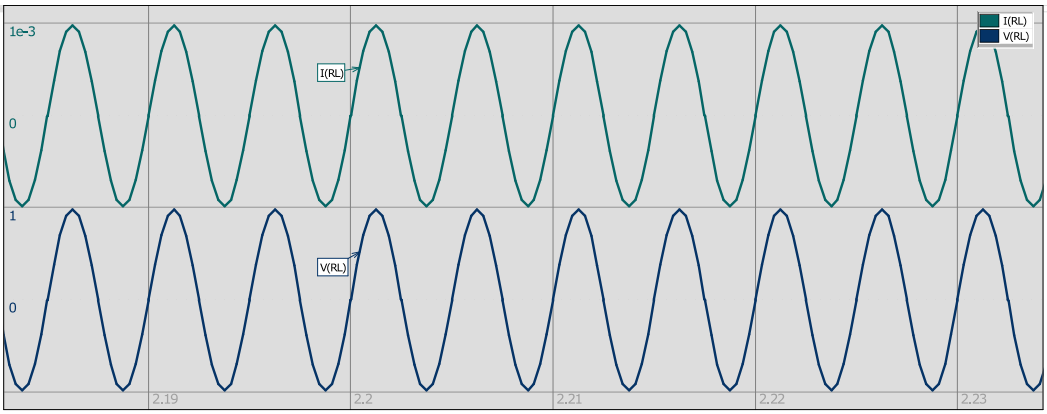

Figura 19 - Simulación de formas de onda para configuración R_L =1KΩ y R_0=1.5KΩ

	min	max	pp	freq
Cursors				
I(RL)	-999.974e-6	999.974e-6	1.99995e-3	200
V(RL)	-999.974e-3	999.974e-3	1.99995	200

Tabla 3 - Valores obtenidos en la simulación de la configuración R_L =1KΩ y R_0=1.5KΩ

Se decidió duplicar los valores de resistencias R_0 calculados, con el fin de ampliar considerablemente los rangos de valores de R_L que se permite medir, a costa de tener mayor error de medición para las resistencias R_L que se encuentren en el intervalo [1kΩ, 2kΩ).

Se reemplazó la resistencia R_0 de la Figura 17, por el circuito de la Figura 21, con el fin de configurar los 3 rangos de medición que se muestran en la Figura 20. Para ello se utilizaron dos Relés controlados desde el µC para medir las diferentes resistencias de carga.

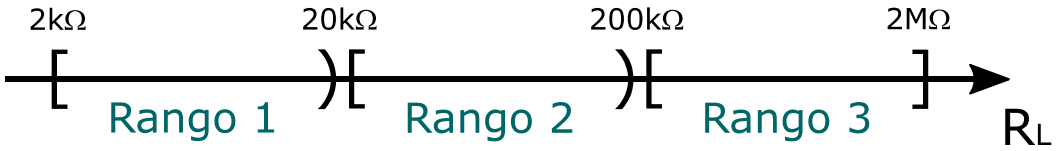

Figura 20 -Distribución de Rangos en función al valor de RL

Según su combinación, se puede tener $3K\Omega$, $30K\Omega$ y $300K\Omega$ como resistencia R_0. Como se muestra en la Figura 21, para el Rango 1: el relé1 se enciende y $R_0 = R_{0A}$. En cambio, en el Rango 2, el relé1 se apaga y se enciende el relé2 teniendo $R_0 =$

$R_{0A} + R_{0B}$. Luego en la configuración Rango3, ambos relés se apagan, consiguiendo $R_0 = R_{0A} + R_{0B} + R_{0C}$.

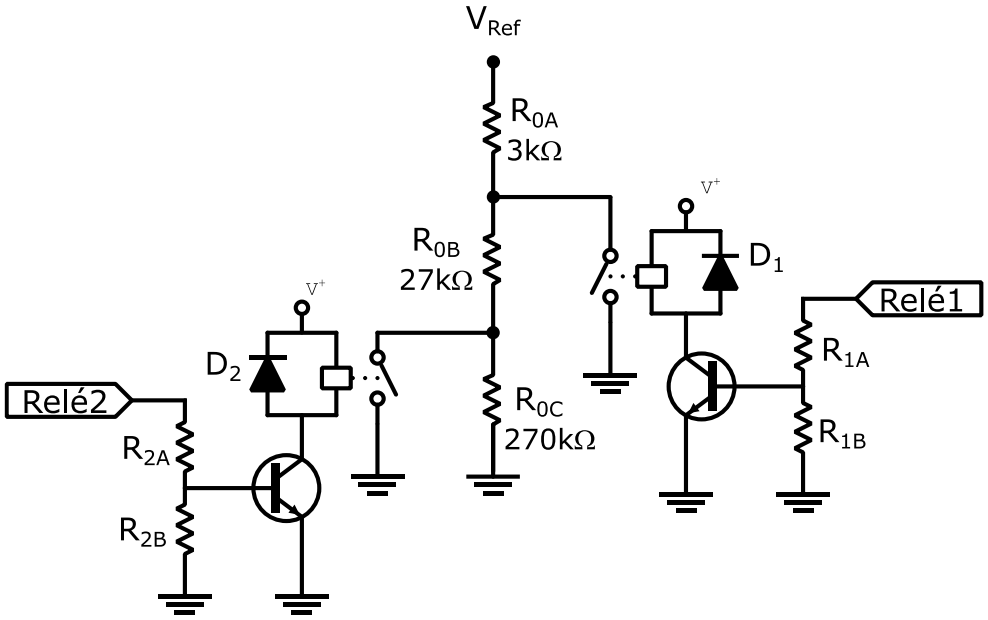

Figura 21 - Circuito para la configuración de rangos de medición

	R_L	V_L	i_L	R_0
Rango 1	*2 KΩ*	*1 Volt*	*500 μA*	*3 KΩ*
	20 kΩ	*10 Volt*		
Rango 2	*20 KΩ*	*1 Volt*	*50 μA*	*30 KΩ*
	200 KΩ	*10 Volt*		
Rango 3	*200 KΩ*	*1 Volt*	*5 μA*	*300 KΩ*
	2 MΩ	*10 Volt*		

Tabla 4 - Distribución de Rangos de Medición

Amplificador de Instrumentación

La función del Amplificador de instrumentación se basa en el Acondicionamiento de la Señal. En esta etapa se realiza el sensado y acondicionamiento de la señal de tensión (señal de respuesta ante la corriente eléctrica aplicada en la celda).

Para ello necesitamos una ganancia precisa y estable, alta impedancia de entrada, baja impedancia de salida y alto rechazo en modo común.

Ante las exigencias de medida que impone el proyecto, se diseña un amplificador de instrumentación.

Dicho amplificador consta de dos etapas: etapa pre-amplificación y etapa diferencial. La configuración más utilizada como amplificador de instrumentación está constituida por tres amplificadores operacionales utilizados como en el esquema de la figura:

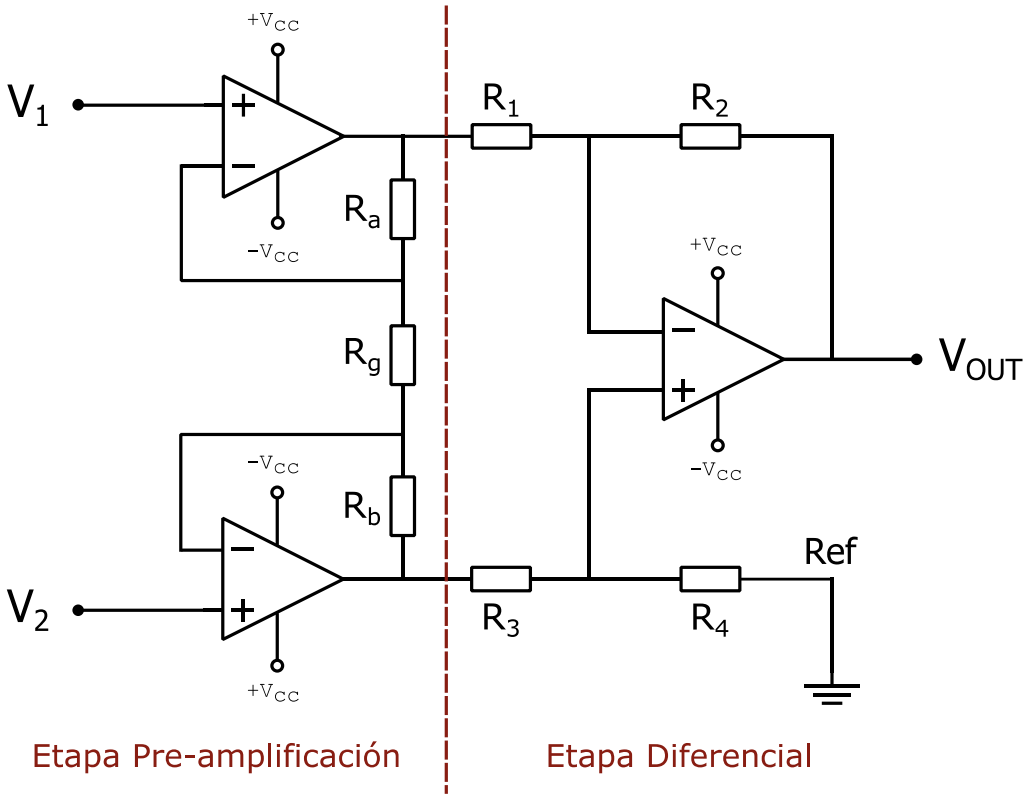

Figura 22 - Amplificador de Instrumentación

$$V_A = V_1\left(\frac{R_A}{R_G}+1\right)-\frac{R_A}{R_G}V_2$$

Ecuación 9

$$V_B = V_2\left(\frac{R_B}{R_G}+1\right)-\frac{R_B}{R_G}V_1$$

Ecuación 10

Si en el lugar de R_g se abre el circuito ($R_g \to \infty$), se consigue un seguidor de tensión y al mismo tiempo un amplificador de instrumentación con impedancia de entrada infinita.

El amplificador de Instrumentación tiene como objetivo sensar la tensión de la solución acuosa (la cual posee valores entre 1~10 Volt) y acondicionarla de modo tal que en la salida haya una señal de tensión en un rango de ±1.5 Volt (proporcional al rango medido) con la misma forma y frecuencia que la señal medida. La etapa diferencial del amplificador de Instrumentación (V_2-V_1) tiene que tener una ganancia:

$$A_d = \frac{V_{d\,out}}{(V_2 - V_1)} = 0.15$$

Ecuación 11

Considerando el valor de V_{ref} = 0, la salida diferencial y de modo común del amplificador de instrumentación serán respectivamente:

$$V_{d\,out} = -\frac{R_2}{R_1} \cdot V_1 + \frac{R_2 + R_1}{R_1} \cdot \frac{R_4}{R_3 + R_4} \cdot V_2$$

Ecuación 12

$$V_{CM\,out} = \left(-\frac{R_2}{R_1} + \frac{R_2 + R_1}{R_1} \cdot \frac{R_4}{R_3 + R_4}\right) \cdot V_{CM}$$

Ecuación 13

Luego si $R_1 = R_3$ y $R_2 = R_4$, obtenemos:

$$V_{CM\,out} = 0$$

Ecuación 14

$$V_{d\,out} = \frac{R_2}{R_1} \cdot (V_2 - V_1)$$

Ecuación 15

Comparando la Ecuación 11 y la Ecuación 15:

$$\frac{R_2}{R_1} = 0.15$$

Ecuación 16

Filtro Antialiasing

La salida del amplificador diferencial entrega al µC la señal analógica que se desea recuperar. Se acondicionó dicha señal, antes de hacer una conversión analógica-digital, para limitar el ancho de banda de la señal de entrada y evitar errores debido a aliasing de señales por encima de la frecuencia de Nyquist. Se diseñó un LPF

antialiasing con frecuencia de corte (f_c) por debajo de la frecuencia de Nyquist, y al menos una década por encima de la frecuencia de la señal de referencia. ($F_s > 2f_c$).

El filtrado se aplicó en ambas entradas diferenciales colocando capacitores y resistencias que fijen la frecuencia de corte del filtro, y determine la ganancia de la etapa diferencial requerida en la Ecuación 16 y en la Ecuación 19.

Se adopta R_1=10KΩ y R_2=1.5 kΩ. Luego R_2, por no ser un valor comercial, se obtiene a partir del paralelo R_{2a} = 5kΩ y R_{2b}= 2k15Ω (siendo $R_{2a} = R_{4a}$ y $R_{2b} = R_{4b}$).

Para evitar el solapamiento se colocaron dos capacitores. Uno en la realimentación del amplificador diferencial (en paralelo con la resistencia R_{2a} y R_{2b}) y el segundo en paralelo a R_{4a} // R_{4b}.

La tensión de salida del amplificador diferencial obtenida en la Ecuación 15 se modificó, obteniendo la Ecuación 17.

$$V_{d\,out} = \frac{R_2}{R_1} \cdot \left(\frac{1}{1 + sR_2C}\right)(V_2 - V_1)$$

Ecuación 17 - para $R_2 = R_{2a}$ // R_{2b}

Se adoptó el valor del capacitor C=20nF, fijando una frecuencia de corte $f_c = 5.3kHz$. De esta manera se verifica lo propuesto en la Ecuación 18:

$$F_s > 10.6\ kHz$$

Ecuación 18

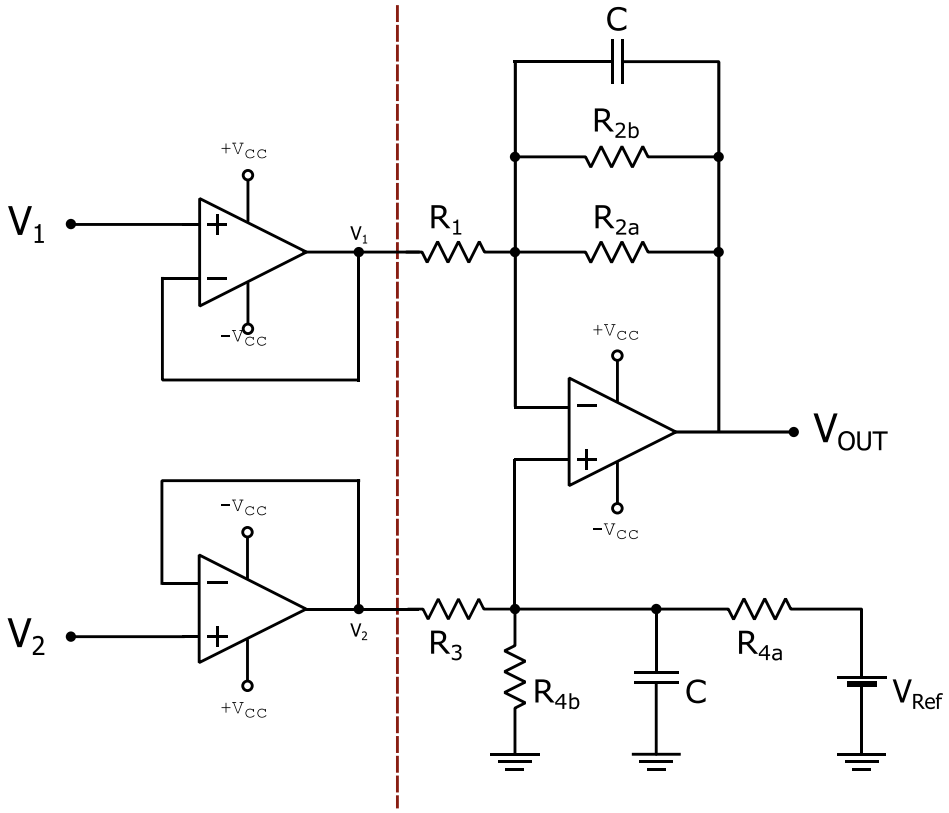

Figura 23

Luego calculamos V_{ref} para sumar una tensión continua de 1.5Volts a la V_{out}, para obtener, de esta manera, una señal de $0 < V_{out} < 3$ Volt.

$$1.5\ \text{Volt} = \frac{\left(\frac{R_{2b}\cdot R_{2a}}{R_{2b}+R_{2a}} + R_1\right)}{R_1} \cdot \left(\frac{R_{4b}\cdot R_3}{R_{4b}+R_3}\right) \cdot \frac{V_{ref}}{\left(R_{4a} + \frac{R_{4b}\cdot R_3}{R_{4b}+R_3}\right)}$$

Ecuación 19

$$1.5\ \text{Volt} = 1.15 \cdot 1\text{k}77\Omega \cdot \frac{V_{ref}}{6\text{k}77\Omega}$$

$$\mathbf{V_{ref} \approx 5Volt}$$

Para generar la V_{ref} se utiliza un diodo Zener polarizado que fija una tensión de 5.1Volt.

La tensión de salida simulada y su tabla de valores correspondiente, se observan en las siguientes figuras:

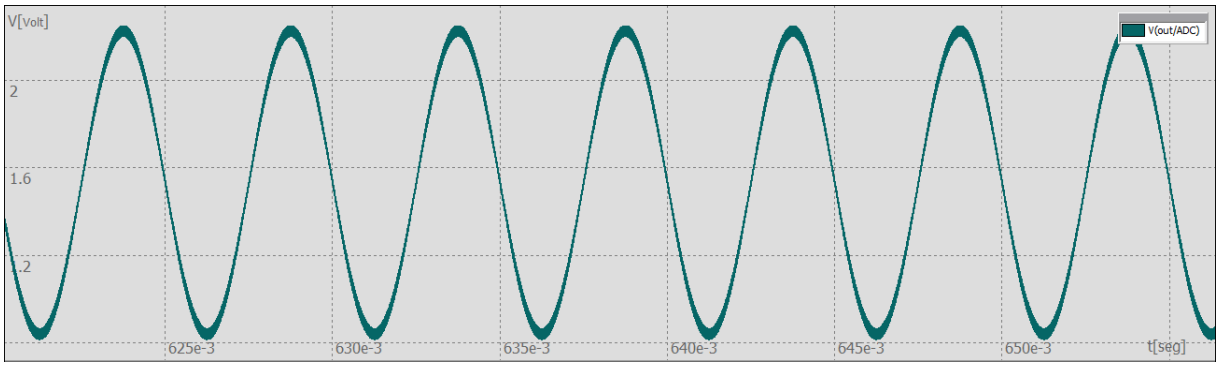

Figura 24 – Simulación de la Tensión de Salida del Amplificador de Instrumentación

	min	max	pp	mean	freq
Cursors					
V(out/ADC)	818.957e-3	2.25246	1.4335	1.51841	200

Tabla 5 - Valores de Tensión de Salida Simulados

Filtro Pasa Banda

La onda senoidal de 200Hz generada por µC se acondicionó, colocando un filtro pasa banda (BPF) pasivo, entre la salida del DAC y la entrada al generador de corriente, para eliminar el OFFSET y suavizar los escalones de la senoidal digital. El BPF se compuso por un filtro pasa bajos (LPF) "RC" y un filtro pasa altos (HPF) "CR" conectados en serie, con frecuencia de corte ubicadas, aproximadamente, una década por debajo (f_{c_HPF}) y una década por encima (f_{c_LPF}), respecto la frecuencia deseada de 200 Hz.

Se adoptan los componentes como indica la Figura 25:

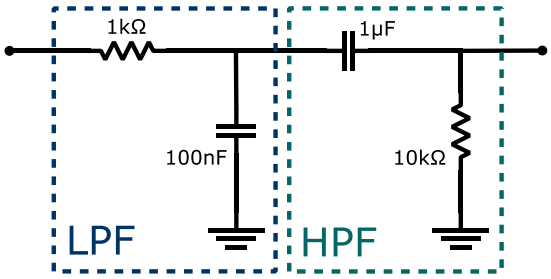

Figura 25 - Filtro pasa banda

Y sus respectivas frecuencias de corte se calculan en las Ecuación 20 y Ecuación 21.

$$f_{c\_LPF} = \frac{1}{2\pi \cdot 1k\Omega \cdot 100nF} = 1591Hz$$

Ecuación 20

$$f_{c\_HPF} = \frac{1}{2\pi \cdot 10k\Omega \cdot 1\mu F} = 15{,}9Hz$$

Ecuación 21

En las Figura 26 y Figura 27, se observan las simulaciones de la respuesta temporal y en frecuencia del BPF diseñado.

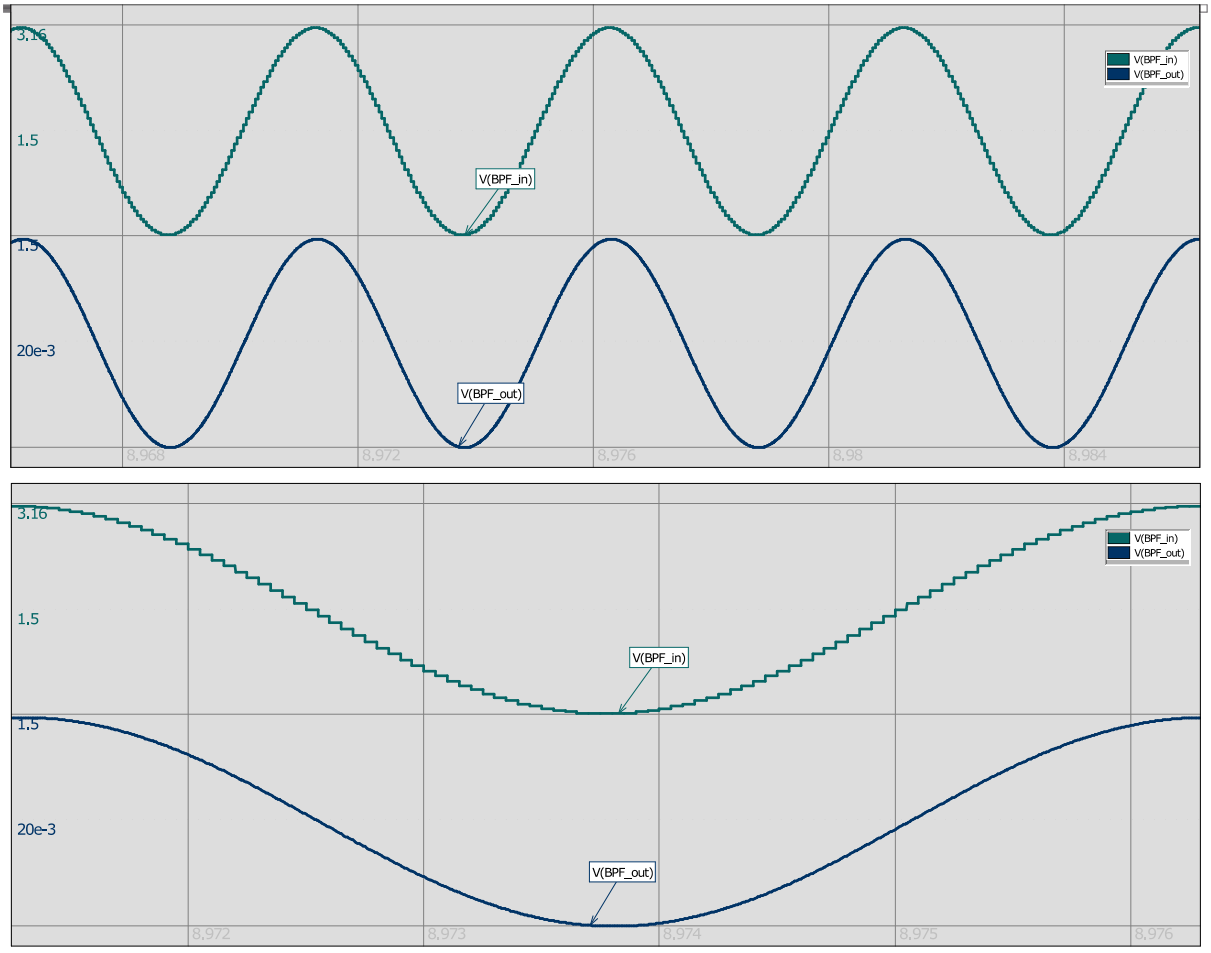

Figura 26 - Simulación temporal BPF

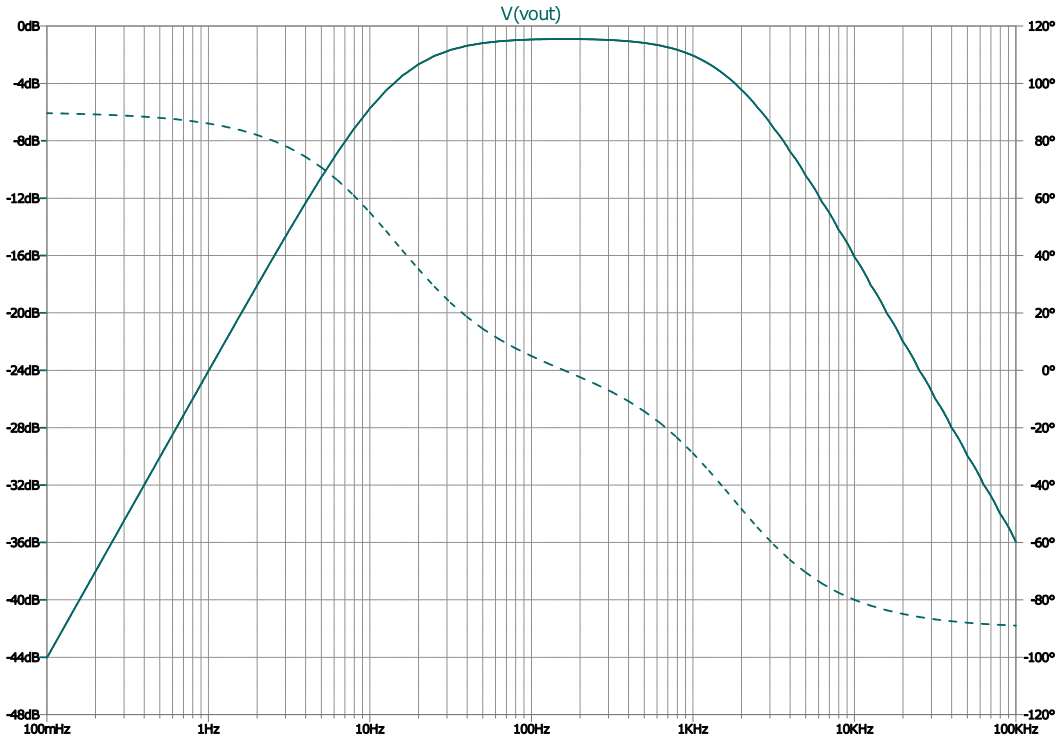

Figura 27 - Bode BPF

Fuentes de alimentación

Para el proyecto se necesitan tres fuentes de alimentación distintas, para abastecer las diferentes partes del circuito. El microcontrolador (µC), el sensor de luz y el conector para programar se alimentan con V_{DD} (3.3 Volt). La pantalla LCD y el adaptador de USB/UART necesitan 5 Volts para su funcionamiento y, finalmente, la parte analógica del circuito se alimenta con $\pm V_{CC}$ (±12 Volt).

A partir de una fuente de 5 Volt, se generan las distintas fuentes de alimentación debido a que el circuito no demanda mucha potencia. Y además, a través de una simple ficha USB de uso frecuente, se puede acceder a la tensión de alimentación necesaria.

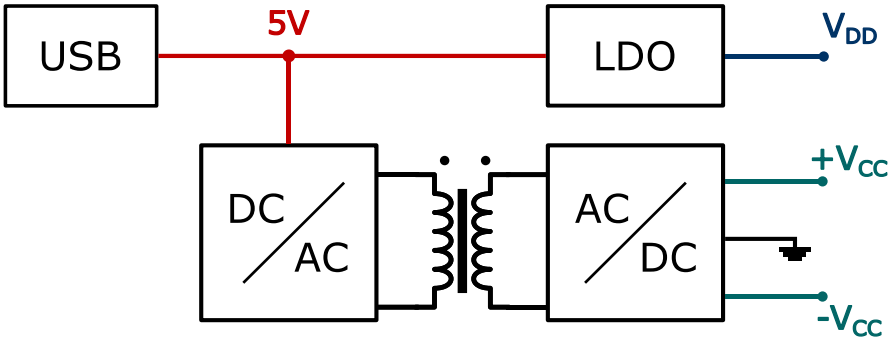

Figura 28- Diagrama en Bloques de las Fuentes de Alimentación

Como se muestra en la Figura 28, a partir de la tensión que provee el conector USB (5 Volt), se obtienen las fuentes V_{DD} y $\pm V_{CC}$. Los 3.3 Volt se consiguen a través de un regulador lineal de voltaje, mientras que para los ±12 Volt se utiliza un convertidor Push-Pull (Apéndice A).

La Figura 29 muestra el funcionamiento de la fuente Push-Pull implementada, basada en la nota de aplicación (Chu, 2018).

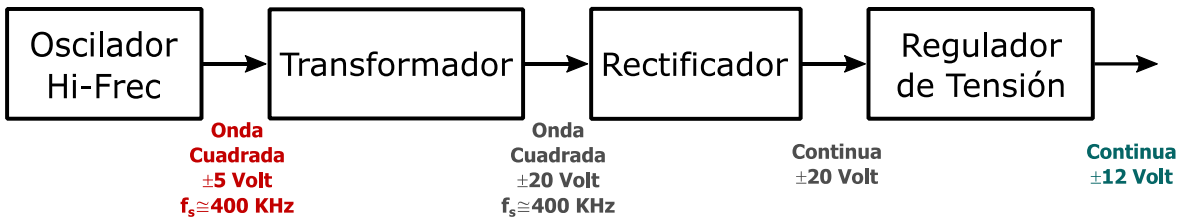

Figura 29 - Diagrama en Bloques de la Fuente V_{CC}

Para los bloques de la fuente, se utilizaron componentes recomendados en la nota de aplicación. Salvo el bloque que corresponde al transformador, en donde fue necesario realizar el diseño para que cumpla dos funciones. La primera, para elevar la tensión de ±5V a ±20V; y la segunda para que tenga un diseño planar, integrado en el PCB.

El diseño del transformador se realizó en base a la bibliografía, la cual propone construir el núcleo con un material de ferrita tal como el 3C8; sin embargo dicho material es obsoleto. En su lugar, se utilizó el 3C95, como resultado de una comparación de materiales similares dentro de la familia 3C9x. A partir de la Figura 30, se observa que la densidad de potencia de pérdida frente a las curvas de temperatura para 3C95, es muy plana y las pérdidas varían poco desde la temperatura ambiente hasta los 100 ° C. Esto es válido para diversas condiciones de frecuencia y densidad de flujo.

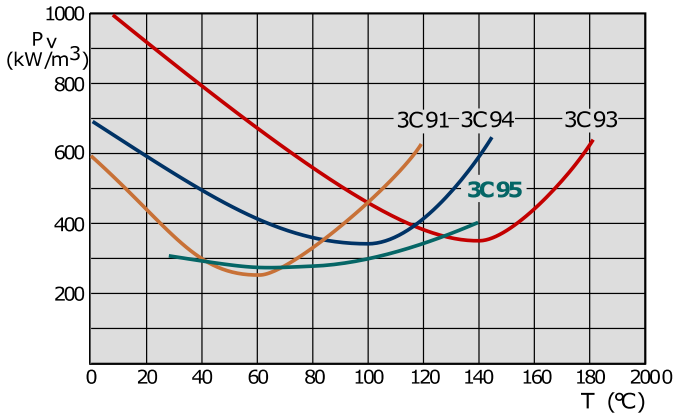

Figura 30 - Comparación de la Potencia de Pérdida

Una vez seleccionado el material del núcleo del transformador, se utilizó la Figura 31 de su hoja de datos y se estableció una condición sobre la densidad de flujo pico ($\hat{B}$) máxima:

$$\hat{B} < 100\ mT$$

Ecuación 22

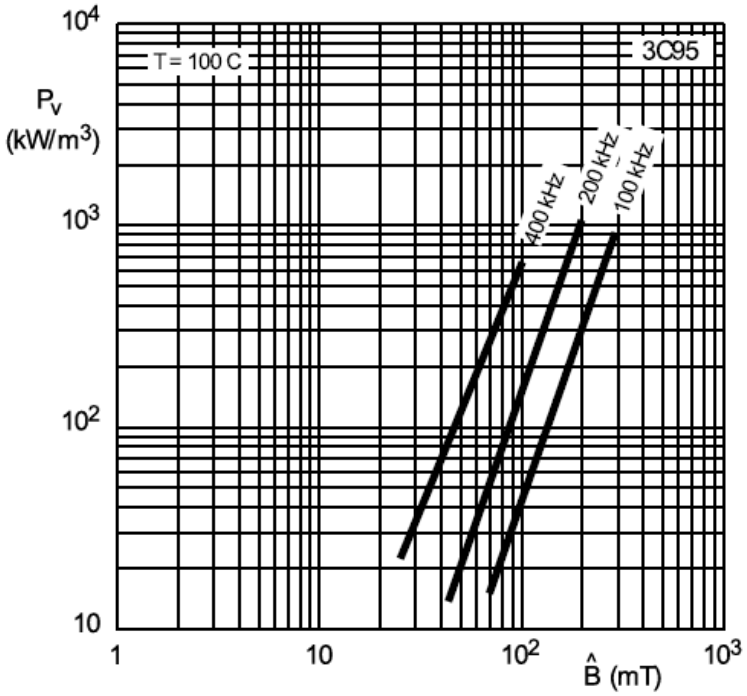

Figura 31 - Pérdidas de potencia específicas en función de la densidad de flujo pico con la frecuencia como parámetro (Ferroxcube, 2014)

Además de la condición sobre $\hat{B}$ que se desprende de la hoja de datos, la bibliografía indicó que para un convertidor Push-Pull se debía cumplir también:

$$\hat{B} < B_{sat}$$

Ecuación 23

Para el material elegido, el $B_{sat} = 410\ mT$. Luego se adopta $\hat{B} = \frac{B_{sat}}{5} = 80\ mT$, pues cumple con las condiciones de la Ecuación 22 y Ecuación 23

.

Se dispuso de un núcleo planar E 18/4/10/R-3C95, con un área efectiva $A_e = 39.5mm^2$ y un volumen efectivo $V_e = 830mm^3$.

Para un convertidor Push-Pull, con una relación de trabajo de 0.5, se calcula el número de vueltas del devanado primario (N_p) a partir de la Ecuación 24

.

$$\hat{B} = \frac{V_p}{4 f_s A_e N_p}$$

Ecuación 24

A partir de esta ecuación, se llega a un resultado decimal en representación de la cantidad de vueltas, $en\ base\ al\ \hat{B}\ adoptado\ (N_p = 1.3)$. Debido a esto, se calculó el valor de la densidad de flujo pico para una y dos vueltas en el devanado primario:

$$\hat{B}_{(N_p=1)} = \frac{5V}{4 \cdot 300kHz\ \cdot [(39.5\ mm^2) \cdot 10^{-6}] \cdot\ 1} = 105{,}5\ mT$$

$$\hat{B}_{(N_p=2)} = \frac{5V}{4 \cdot 300kHz\ \cdot [(39.5\ mm^2) \cdot 10^{-6}] \cdot\ 2} = 52{,}7\ mT$$

Se observó que $\hat{B}_{(N_p=1)}$ no cumple con la condición de diseño de la Ecuación 22; es por este motivo que se diseñó un transformador con $N_p = 2$ y $N_s = 8$ para que cumpla la relación de vueltas n deseada dada por la Ecuación 25.

$$n = 4 = \frac{V_s}{V_p} = \frac{N_s}{N_p}$$

Ecuación 25

Para verificar que la elección de vueltas en el devanado primario fue acertada, se analizan las perdidas en el núcleo; para ello, se parte de la Figura 31, para un valor de densidad de flujo $\hat{B}_{(N_p=2)} \approx 50\ mT$ y la potencia de perdida específica (P_v) a 100°C

y 400kHz que se obtiene a partir de este grafico, es $P_v = 100\ kW/m^3$, es decir una perdida en el núcleo de:

$$P_{Núcleo} = P_v \cdot V_{Núcleo} = 100\frac{kW}{m^3} \cdot (830 \cdot 10^{-9}\ m^3\ ) = 0.083\ W$$

Finalmente, se llega a la conclusión que el transformador es viable.

Conector USB/µC

Se utiliza un puerto USB (Universal Serial Bus) para conectar y transmitir los datos procesados en el microcontrolador a una computadora.

En la Figura 32, se muestra el diagrama en bloques correspondiente al conector.

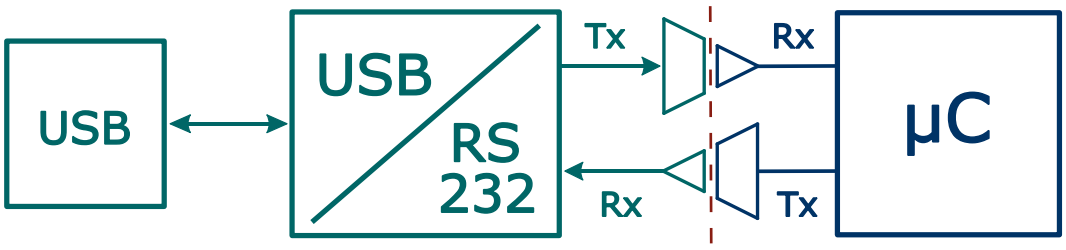

Figura 32 - Diagrama en Bloques del Conector USB

En el diagrama, se muestra una línea vertical punteada que indica una aislación entre los dos primeros bloques y el microcontrolador para, de esa forma, evitar que la PC introduzca ruido a la medición.

Procesamiento

Para generar la señal sinusoidal de tensión de 200 Hz con el µC, se utilizó una tabla con valores de tensión, de un periodo de la sinusoidal, y una base de tiempo para reconstruir la señal que cumpla con la condición de la Ecuación 18.

La duración de base de tiempo (Δt) se calculó con el objetivo de fijar la frecuencia de una interrupción, durante la cual el µC debe: actualizar el valor del DAC (T_{DAC}), muestrear la señal de tensión (T_S) y efectuar el algoritmo para el cálculo de la conductividad (T_C), como se muestra en la Figura 33,

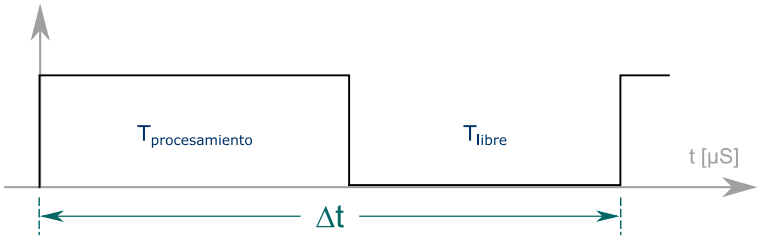

Figura 33 - Distribución de tiempos BDT; donde Tprocesamiento=TDAC +TS+TC.

Se decidió tomar 100 valores de la tabla para representar un periodo $T = 5mSeg$ de la senoidal. Obteniendo la base de tiempo de la Ecuación 26:

$$\Delta t = \frac{1}{F_s} = \frac{5mSeg}{100} = 50\mu Seg$$

Ecuación 26 - Duración de la BDT

Para construir un algoritmo con mayor precisión, se debería haber usado datos en punto flotante de precisión doble o simple. Sin embargo, se requería una sobrecarga significativa del procesador, para realizar cálculos de punto flotante, como resultado de la falta de soporte en el hardware (el STM32-M0 no posee FPU). Dicha sobrecarga limita la tasa de iteración efectiva de un algoritmo.

Para mejorar el rendimiento matemático o aumentar la tasa de ejecución repetitiva del algoritmo, se utilizó la representación de punto fijo (Ver Apéndice B).

La tabla que se utilizó contenía 4000 valores de un ciclo de la senoidal, en notación de Punto Fijo I2Q30. Cada valor representaba la amplitud de la función para cada ángulo, o índice (θ), entre 0y 2π.

Para conseguir una tabla de 100 valores, se incrementa el índice $\Delta\theta$ de la tabla original, según la Ecuación 27.

$$\Delta\theta = \frac{Cantidad\ de\ Muestras\ Original}{Cantidad\ de\ Muestras\ Deseadas} = \frac{4000}{100} = 40$$

Ecuación 27 - Incremente en el índice

La forma de onda obtenida se muestra en la Figura 34:

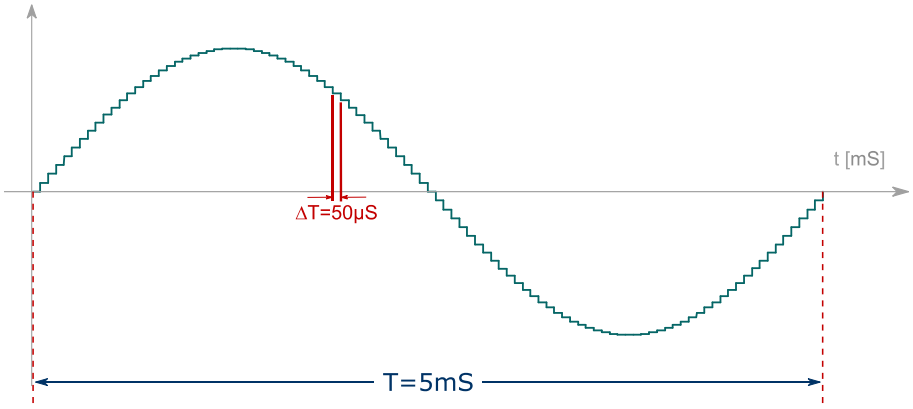

Figura 34 - Muestreo

Ver Apéndice D que muestra el código ejecutado.

Algoritmo

Para el cálculo la conductividad eléctrica, se utilizaron los valores RMS de corriente aplicada y tensión medida en la solución. El objetivo es calcular la tensión V_{RMS}, como un valor representativo de la tensión medida, para luego dividir la corriente I_{RMS} por dicho valor y obtener así la conductancia, como muestra la Ecuación 28:

$$C = \frac{I_{RMS}}{V_{RMS}} \cdot \frac{\ln(2)}{\pi \cdot 2}\ [S/cm]$$

Ecuación 28

La corriente que se inyecta al transductor es una señal conocida para cada uno de los rangos de medición establecidos. (Ver Tabla 4). De forma general, podemos

estimar el su valor RMS para corrientes sinusoidales, a partir del valor de tensión pico $\hat{V}_I$ de la entrada inversora del AO de la fuente de corriente.

$$I_{RMS} = \frac{\hat{V}_I}{R_0} \cdot \frac{\sqrt{2}}{2}$$

Ecuación 29

Se utilizó la Ecuación 30 para estimar la tensión V_{RMS} en la solución, a partir del valor medio de la sinusoidal rectificada.

$$V_{RMS} = \frac{\sqrt{2}}{4}\pi \cdot \bar{V}_{sol}$$

Ecuación 30

Para evitar la realización un procedimiento de ajuste de offset al inicio de cada medición, se realizó un filtrado de la señal para eliminar el offset. Se diseñó un filtro pasa alto con frecuencia de corte de $2Hz$, antes del cálculo del V_{RMS}, como muestra la Ecuación 31:

$$H_{HPF}(Z) = \frac{(1 - Z^{-1})}{(1 - C_{1\_HPF} \cdot Z^{-1})}$$

Ecuación 31

Con $C_{1\_HPF} = e^{-2\pi \cdot 2 \cdot 50e-6}$, diagrama de Bode que muestra la Figura 35:

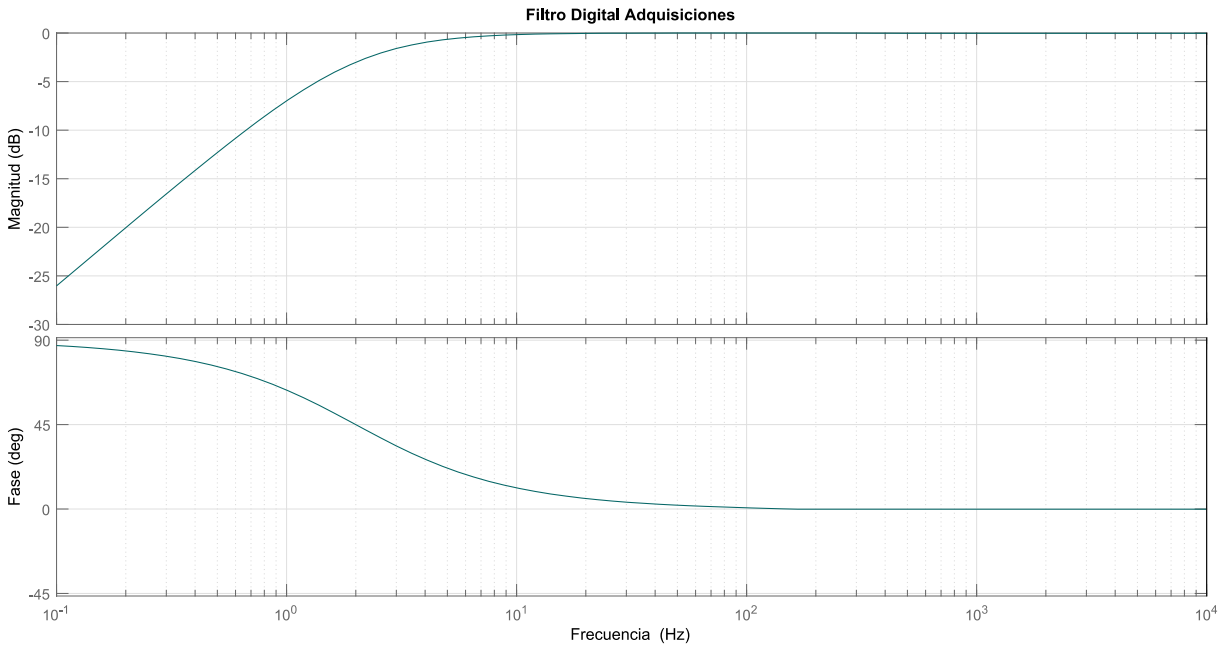

Figura 35 - Diagrama de Bode HPF Digital

Una vez filtrada la tensión de salida del amplificador diferencial, se calculó por software el valor medio de la señal rectificada de tensión $\bar{V}_{sol}$, en base a los valores adquiridos por el ADC de 12 bits del STM32, durante la interrupción de la base de tiempo. Para ello se armó un vector $x[k]$ que permitió almacenar las últimas N muestras del ADC en punto fijo I12Q20, correspondientes a un ciclo de la señal a filtrar. El valor de "N" es la cantidad de muestras correspondiente a un periodo de la señal de entrada.

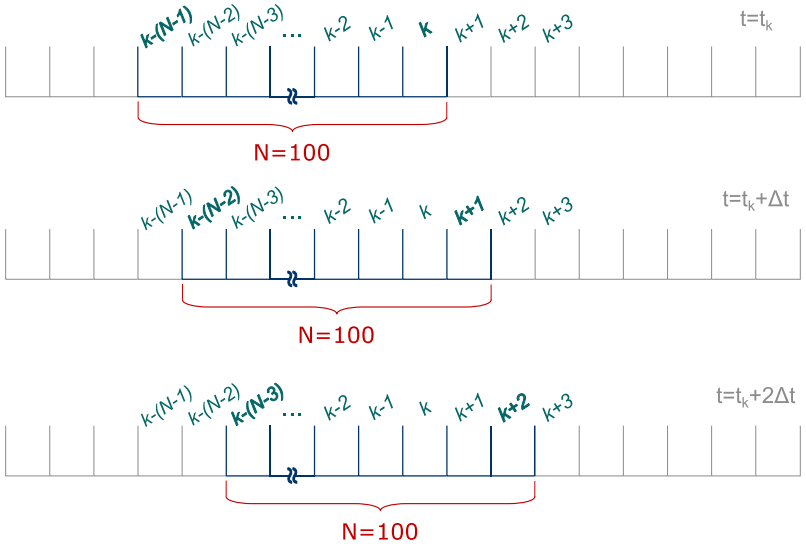

Figura 36

El promedio de las primeras 100 muestras ($y[k]$) se obtuvo a través de la Ecuación 32:

$$y[k] = \frac{1}{N}\sum_{i=0}^{N-1} x[k-i]$$

Ecuación 32

Al actualizar la medición del ADC, el valor de la ventana, que forma el vector, se desplazó una posición a la derecha como indica la Figura 36; así, el nuevo valor muestreado, se almacenó en la posición $k+1$ y se descartó el valor de la posición $k-(N-1)$, actualizando también los valores del vector. Luego, se calculó el promedio por medio de la Ecuación 33:

$$y[k+1] = \frac{1}{N}\left(\sum_{i=0}^{N} x[k-i] + x[k+1] - x[k-(N-1)]\right)$$

Ecuación 33

$$y[k+1] = y[k] + \frac{1}{N}(x[k+1] - x[(k+1)-N])$$

Ecuación 34

Se aplicó un cambio de variables "$k+1=\kappa$", para trabajar con un sistema causal, resultando:

$$y[\kappa] = y[\kappa-1] + \frac{1}{N}(x[\kappa] - x[\kappa-N])$$

Ecuación 35

Aplicando la "Transformada Z", se obtiene la Ecuación 36:

$$Y(Z) = Y(Z)\cdot Z^{-1} + \frac{1}{N}[X(Z) - X(Z)\cdot Z^{-N}]$$

Ecuación 36

$$\frac{Y(Z)}{X(Z)} = \frac{1}{N}\cdot\frac{1-Z^{-N}}{1-Z^{-1}}$$

Ecuación 37

Obteniendo así un filtro digital, que promedia las muestras de la entrada y por lo tanto suprime variaciones rápidas, característica que le otorga el carácter de filtro pasa bajo.

Se representó la Ecuación 36 mediante el Diagrama en Bloques de la Figura 37, y se simuló la respuesta temporal en la entrada y salida del filtro como muestra la Figura 38:

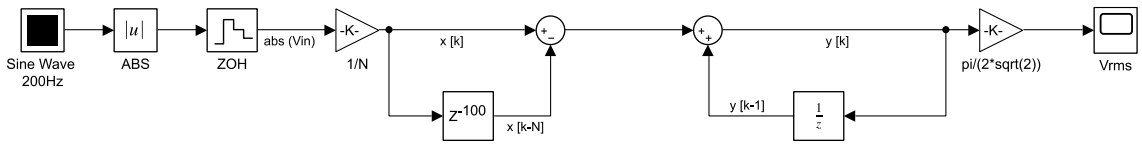

Figura 37- Diagrama en Bloques LPF Digital

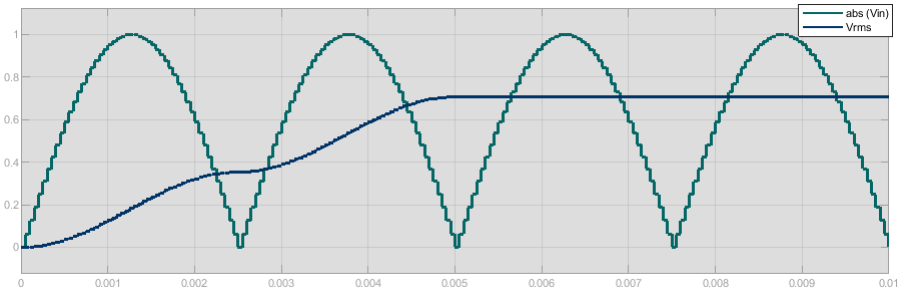

Figura 38 - Respuesta Temporal cálculo Vrms

En la Figura 38 se observa que el valor V_{RMS} (curva azul) se obtiene luego de $5\mu s$, cuando ya se adquirió 100 datos, correspondientes a un periodo de la senoidal.

El bode de la Figura 39 muestra que se elimina la componentes fundamental de la sinusoidal de 200Hz y sus componentes armónicas (que coinciden con la frecuencia de la señal valor absoluto), permitiendo pasar la continua que representa el V_{RMS} deseado.

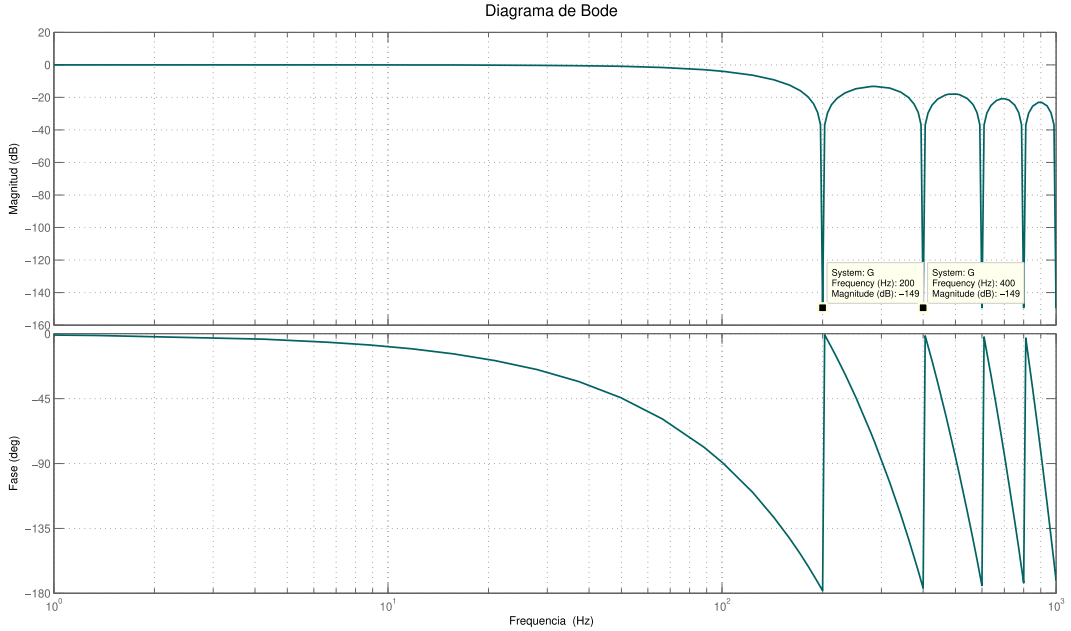

Figura 39 - Bode del Filtro Digital para el Cálculo de Vrms

Finalmente se diseñó un filtro pasa bajos de la Ecuación 38 atenuar aún más las oscilaciones en la estima de la tensión V_{RMS} obtenida y, de esa forma, disminuir el error en el cálculo de Conductividad.

$$H_{LPF}(Z) = \frac{C_{2\_LPF}}{(1 - C_{1\_LPF} \cdot Z^{-1})}$$

Ecuación 38

Para $C_{1\_LPF} = e^{-2\pi \cdot 5Hz \cdot 50e-6}$ y $C_{2\_LPF} = 1 - C_{1\_LPF}$

Finalmente, la respuesta temporal V_{RMS} a la salida de los tres filtros se observa en la Figura 40:

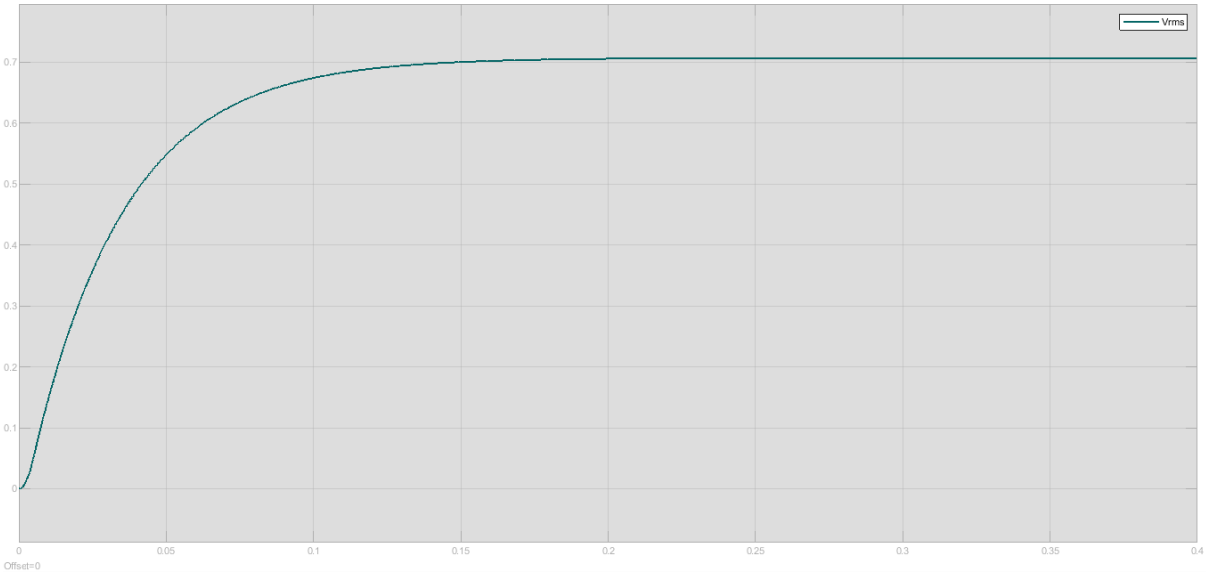

Figura 40 - Simulación de la respuesta temporal del filtro digital

Interfaz de Usuario

La interfaz de usuario del dispositivo está compuesta por una pantalla LCD y cinco botones; estos últimos se disponen en el conductímetro como muestra la Figura 41.

Para implementar la interfaz de usuario se utilizó el formalismo de máquina de estado. Cada uno de los estados se muestra en la pantalla LCD y forma parte del menú de opciones, al cual puede acceder un usuario a través de la utilización de pulsadores, permitiendo la transición de un estado a otro de la máquina de estados de la Figura 42Figura 41.

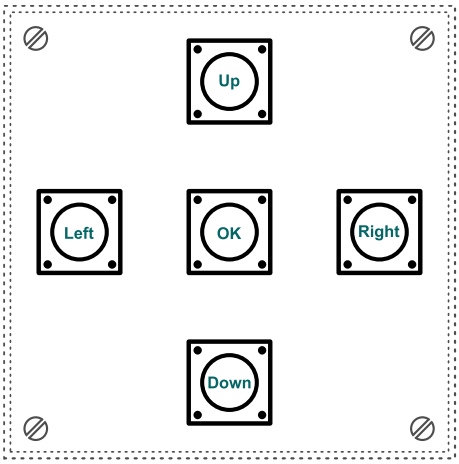

Figura 41- Distribución de Botones de la Interfaz de Usuario.

En el momento en que se enciende el dispositivo, el primer estado es el que muestra en la pantalla la palabra “Menú”. Solo los pulsadores “Up” o “Down” permiten la transición a los estados que muestran en pantalla “Iniciar Medición” o “Iniciar Calibración”, respectivamente.

El estado “Iniciar Calibración” se utilizó con el objetivo de proponer al usuario que seleccione el rango según la solución que desee medir; para ello, el dispositivo espera información del usuario hasta que presione el pulsador “OK” o “Right”, pasando el estado “Configurar Rango 1”. En cambio, si el usuario no desea calibrar, puede presionar el botón “Up” o “Down”, accediendo así al estado “Iniciar Medición” del menú principal.

A partir de la pantalla “Configurar Rango 1”, se puede acceder a los estados “Configurar Rango 2” y “Configurar Rango 3” por medio de la utilización de los pulsadores “Down” y “Up”, respectivamente; o volver al estado “Iniciar Calibración” presionando el botón “Left”. De lo contrario, el dispositivo espera recibir información del pulsador “OK” o “Right” para combinar los relés R_1 y R_2, configurando así el rango de ganancia de la fuente de corriente que el usuario seleccionó, pasando luego al estado “Iniciar Medición”.

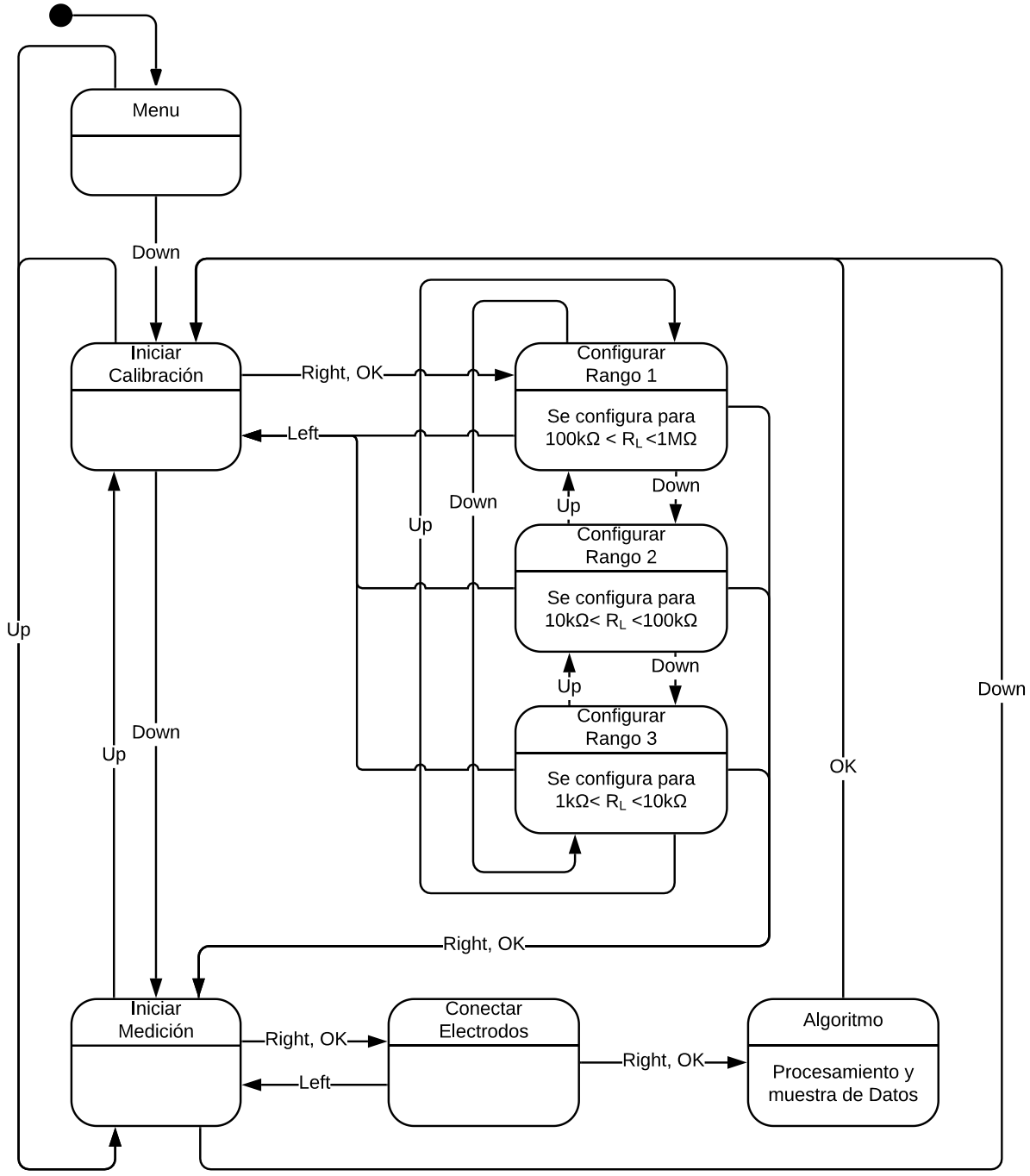

Figura 42- Maquina de Estado

El estado “Iniciar Medición” proponer al usuario comenzar el proceso de medición de la solución. El dispositivo espera hasta recibir la información del usuario para realizar una transición de estado. Por medio del pulsador “Ok” o “Right” inicia tal proceso,

pasando al estado “Conectar Electrodos”. En cambio, puede pasar al estado “Iniciar Calibración” por medio del botón “Up” o “Down”.

A través de la pantalla “Conectar Electrodos” se le recuerda al usuario que verifique que los electrodos estén sumergidos en la solución si desea continuar con la medición. El dispositivo espera recibir la información del usuario. Si se presiona el botón “Ok” o “Right”, el dispositivo cambia al estado “Algoritmo”; también tiene la posibilidad de volver al estado “Iniciar Medición” si presiona el pulsador “Left”.

Por último, el estado “Algoritmo” procesa los datos sensados y muestra en pantalla los resultados obtenidos. Solo a través del pulsador “Ok” cambia de estado, volviendo así a la pantalla “Iniciar Calibración”.

Capítulo 3: Resultados Experimentales

La Figura 43 muestra la duración de la base de tiempo y las Figuras Figura 44, Figura 45 y Figura 46, muestran los tiempos de procesamiento de las tres fases que se realizan durante la interrupción de la base de tiempo.

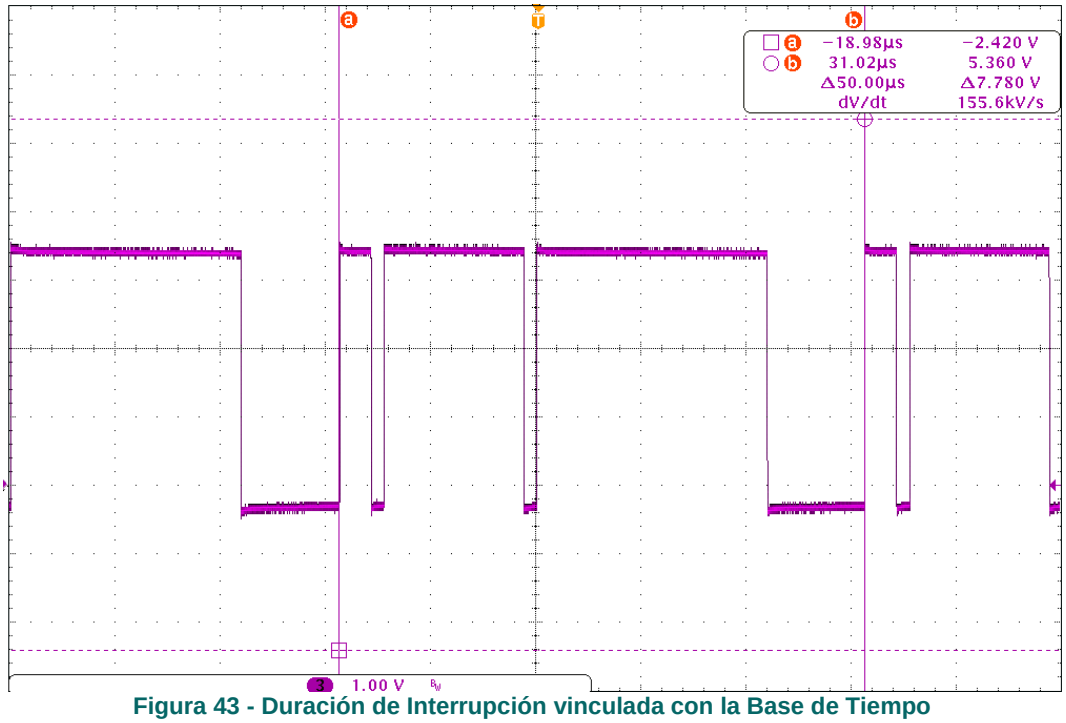

Figura 43 - Duración de Interrupción vinculada con la Base de Tiempo

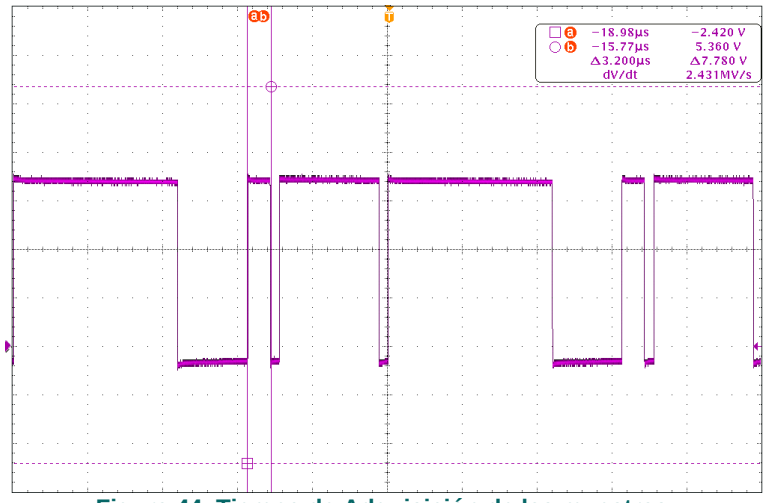

Figura 44 Tiempo de Adquisición de las muestras

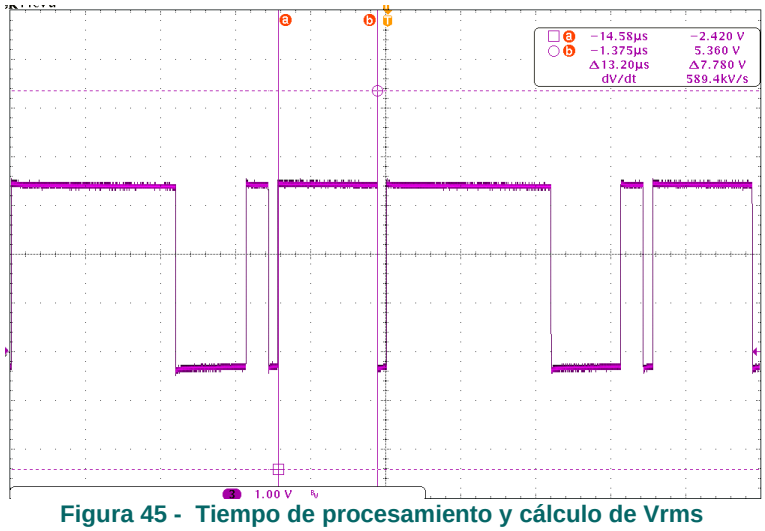

Figura 45 - Tiempo de procesamiento y cálculo de Vrms

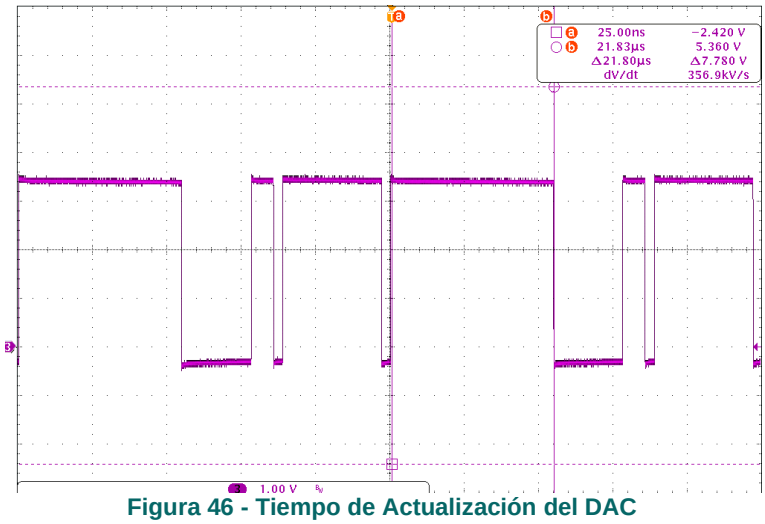

Figura 46 - Tiempo de Actualización del DAC

En la Figura 47 se puede observar la respuesta temporal de los filtros digitales, cuando se aplica una senoidal de 1 Volt pico, aplicada en la entrada del ADC. Además del valor de tensión al cual converge (0.707 Volt), se puede apreciar el tiempo de establecimiento de la señal V_{RMS}

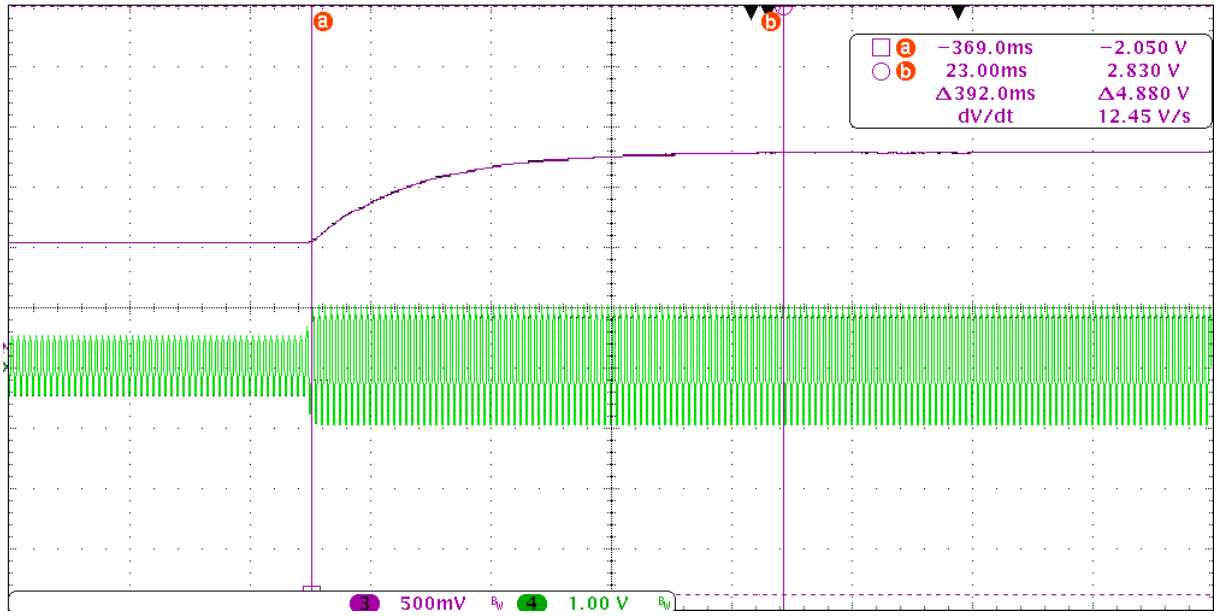

Figura 47 - Tiempo de respuesta para el cálculo del Vrms

La salida del amplificador diferencial es el 15% de la tensión de 2 Vpp aplicada en la entrada diferencial

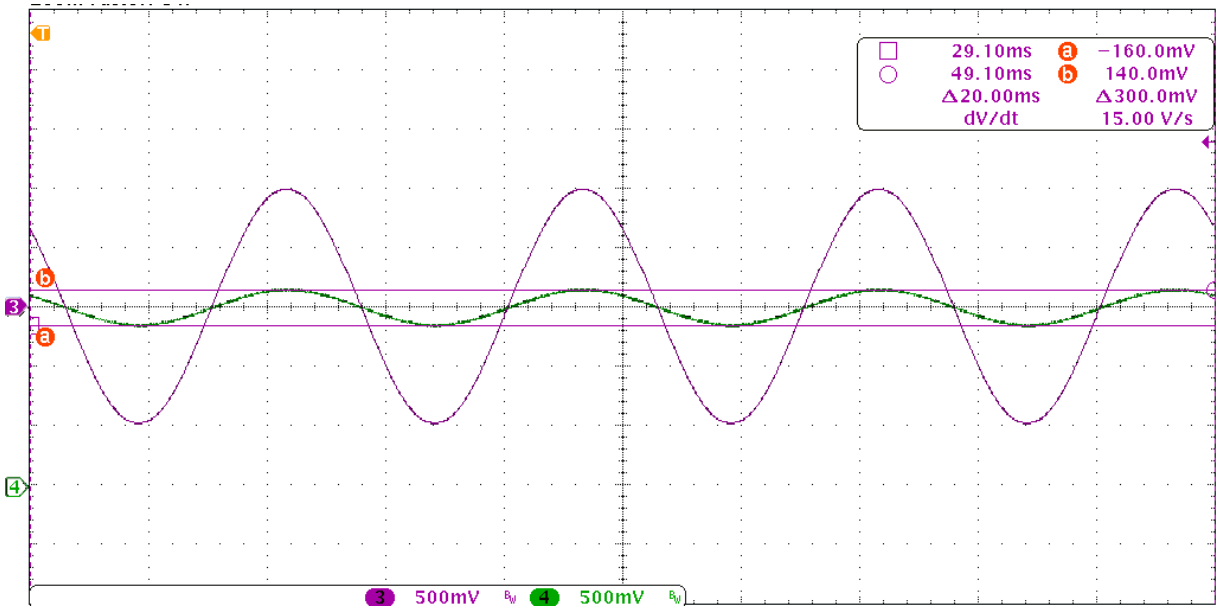

Figura 48 - Forma de onda de la entrada y salida del amplificador diferencial

Finalmente se analizó el error porcentual de medición. Para ello se utilizaron resistencias con valores conocidos y se comparó con lo mostrado en pantalla. Los datos registrados y su error relativo porcentual se presentan en la Tabla 6.

Rango Configurado de Medicion	Resistencia Teórica	Resistencia Medida	Error Relativo
Rango 1	1,2 kΩ	1268,1 Ω	5,67%
	6,6 kΩ	6693,2 Ω	1,41%
	10 kΩ	9944,4 Ω	-0,56%
	19,2 kΩ	18852 Ω	-1,81%
Rango 2	10 kΩ	10578,5 Ω	5,79%
	38,5 kΩ	40125 Ω	4,22%
	47 kΩ	48668,9 Ω	3,55%
	82,2 kΩ	82338,4 Ω	0,17%
	119,8 kΩ	118343,2 Ω	-1,22%
	190 kΩ	187300 Ω	-1,42%
	200 kΩ	196695,5 Ω	-1,65%
Rango 3	119,8 kΩ	129600,8 Ω	8,18%
	380 kΩ	401929,3 Ω	5,77%
	570 kΩ	584112 Ω	2,48%
	914 kΩ	898472 Ω	-1,70%
	1,5 MΩ	1490000 Ω	-0,67%
	2,4 MΩ	2293578 Ω	-4,43%

Tabla 6 - Valores Experimentales de Resistencia y Relaciones

Análisis de Resultados y Errores esperados

Para la realización del conductímetro, primero se investigaron distintas opciones para su desarrollo; luego del análisis se hicieron los cálculos y simulaciones pertinentes a través de programas como NL5 y LTSpice. Una vez verificado los resultados teóricos, se utilizó la placa de desarrollo STM32F072RB, como parte de un prototipo para la parte digital del proyecto; al igual que una placa experimental con componentes de tecnología pasante (THT) para el ensayo de los circuitos analógicos simulados. Finalmente se diseñó y armó la placa PCB (Ver Apéndice C).

Es por este motivo que las fuentes de error fueron disminuyendo a lo largo del desarrollo del conductímetro. Sin embargo, durante las últimas pruebas del dispositivo, se registraron alinealidades en el DAC y el ADC; las cuales pueden haber generado deformaciones en la V_{REF} y en el V_{RMS}, respectivamente. Además, se detectó una tensión de offset en la entrada del ADC la cual se corrigió con un filtro digital HPF sobre los datos muestreados.

También se observó que el porcentaje de error, aumentaba para las resistencias de carga R_L ubican en los extremos de los rangos de medición, como muestra la Tabla 6. Esta situación podría corregirse, como indica la Figura 49, realizando una aproximación lineal de la ganancia real, a partir de la vinculación de la tensión analógica de la entrada y su valor digitalizado. Finalmente, por software, se corregiría la ganancia para lograr mediciones con mayor precisión.

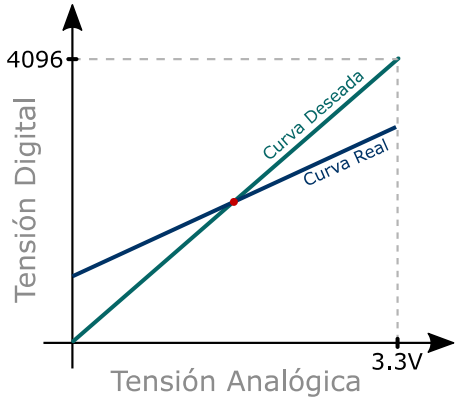

Figura 49 - Ganancia del ADC, Real Vs Deseada

Para el cálculo de la corriente I_{RMS} se utilizaron valores teóricos. Se realizo el cociente de la tensión V_{ref} medida, respecto de la resistencia R_0 de cada rango y luego se lo afecto por el valor teórico $\sqrt{2}/2$.

Capítulo 4: Conclusiones y Trabajos Futuros

Conclusión

La finalización de este proyecto fue no solo la culminación de mi transito como estudiante en la UNMdP, sino también la gestación de mi faceta como profesional, para afianzarme en el campo laboral dentro del marco de la ingeniería.

El aprendizaje no fue solo vinculado con el desarrollo de un conductímetro desde el punto de vista de la electrónica, sino que también adquirí otras herramientas tales como la estructura de pensamiento para el análisis y desarrollo de un proyecto de diseño; así mismo, la resolución de situaciones donde los resultados obtenidos no son siempre los esperados.

Durante el diseño del conductímetro, logré incorporar todo lo visto en mis años como alumna y adquirir la capacidad de integrar todos estos conocimientos para desarrollar mi proyecto.

En primer término, en lo que se refiere a la construcción de la parte analógica, realicé el diseño teórico, las simulaciones en el tiempo y en la frecuencia, la implementación práctica en un prototipo utilizando una placa experimental con componentes THT, para luego diseñar una placa de circuito impreso (PCB) de 4 capas, con componentes de montaje superficial (SMD) .

En segundo lugar, el manejo de una placa de desarrollo (STM32F072RB); así como también el lenguaje de programación C y el software “System Workbench for STM32” para manejar el estado de un programa y finalmente el procesamiento digital de señales dentro del procesador.

También el uso de componentes de interfaces de usuario como un display LCD y el manejo de botones, ambos controlados desde el µC. Y el diseño de componentes magnéticos gracias al convertidor push-pull para generar la fuente $\pm V_{cc}$.

La realización de este conductímetro tenía por objetivo brindar una herramienta de medición a la planta piloto de Ingeniería Química y considero que se logró favorablemente; y, a pesar de que algunos resultados no fueron los esperados, los errores encontrados están dentro del rango permitido, pudiéndose mejorar a través de lo propuesto en el apartado Trabajos Futuros.

Trabajos futuros

Se propone rediseñar la placa PCB para que ambos conectores USB permiten la alimentación del circuito y rehacer las huellas (footprint) del rectificador de tensión, los relés y pulsadores.

Además, definir un protocolo de comunicación por puerto serie y una interfaz gráfica para la PC, que permita enviar datos en función de las demandas del usuario.

Otro punto que se propone es cambiar la pantalla LCD por una que tenga luz de background y así, habilitar por software, el sensor de Luz. También, se podría cambiar de interfaz para incorporar una pantalla de uso táctil VM800B43. De esta manera, se optimiza aún más su utilización.

Por último, se puede diseñar una caja plástica que contenga la placa para asegurar su integridad y la comodidad del usuario.

Apéndices

Apéndice A

Los convertidores Push-Pull requieren transformadores con tomas centrales para transferir energía del primario al secundario.

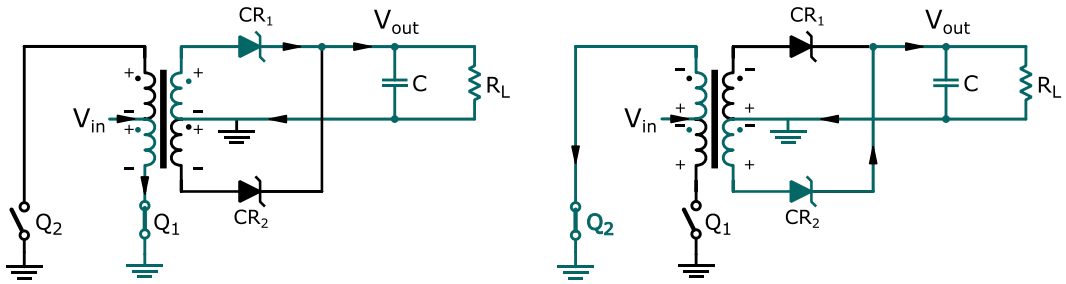

Figura 50 - Circuitos de Conmutación Push-Pull

Los transistores Q_1 y Q_2 conducen de forma alternada para evitar que se produzca un corto circuito entre los extremos del devanado.

Cuando Q_1 conduce y Q_2 queda en alta impedancia, V_{IN} genera una corriente a través de la mitad inferior del devanado primario a tierra, creando así una caída de tensión en el extremo primario inferior con respecto de V_{IN} en la derivación central.

Las dos fuentes de voltaje, cada una de las cuales equivale a V_{IN}, aparecen en serie y causan un potencial de voltaje en el extremo abierto del primario de $2V_{IN}$ con respecto a tierra.

Por convención de puntos, las mismas polaridades de voltaje que ocurren en el primario también ocurren en el secundario. El potencial positivo del extremo secundario superior, por lo tanto, polariza en directa al diodo CR_1. La corriente secundaria que comienza desde el extremo secundario superior fluye a través de CR_1, carga el condensador C y regresa a través de la impedancia de la carga R_L de vuelta a la derivación central.

Cuando Q_2 conduce, Q_1 pasa a alta impedancia y las polaridades de voltaje en el primario y secundario se invierten. Ahora el extremo inferior del primario presenta el extremo abierto con un potencial de $2V_{IN}$ respecto a tierra. En este caso CR_2 tiene polarización directa, mientras que CR_1 tiene polarización inversa y la corriente fluye desde el extremo secundario inferior a través de CR_2, cargando el condensador y volviendo a través de la carga a la derivación central. Volver al informe

Apéndice B

Hay dos enfoques, en la informática moderna, para almacenar números; estas son la notación de punto fijo y la notación de punto flotante. A continuación se analizarán los detalles de estos formatos de almacenamiento.

Representación de Punto Fijo

La notación de punto fijo es una representación de un número fraccional tal como se almacena en la memoria, y se expresa como un entero con signo en formato de complemento a dos.

“Qf” o “ImQf” es una representación fraccional de punto fijo de números de “m+f” bits. Donde “f” indica la cantidad de bits destinados a la representación fraccionaria, permitiendo cambiar la resolución, y “m” la cantidad de bits correspondiente a la parte entera. La notación de punto fijo ubica el punto de raíz (o punto decimal) en cualquier lugar de la representación binaria, dividiendo así la parte entera (a la izquierda de la raíz) de la parte decimal.

Para interpretar los bits del entero con signo almacenado en la memoria, se ubica el punto de la raíz y se multiplica el entero almacenado por un factor de escala fijo. El factor de escala en binario siempre es 2 elevado a un exponente fijo. Como se muestra en la Figura 51, la parte entera es la secuencia habitual de potencias positivas de 2 y la parte fraccionaria es la secuencia de potencias negativas de 2.

Figura 51 - Representación de Punto Fijo

Rango de valores: $[-(2^{m-1}), 2^{m-1} - 2^{-f}]$

Resolución: 2^{-f}

Para sumar o restar dos valores del mismo tipo de punto fijo, es suficiente sumar o restar los enteros subyacentes y mantener su factor de escala común. El resultado puede representarse exactamente en el mismo tipo, siempre que no se produzca un desbordamiento. Si los números tienen diferentes tipos de punto fijo, con diferentes factores de escala, entonces uno de ellos debe convertirse al otro

antes de la suma, manteniendo el signo, a través del desplazamiento a la derecha para alinear el lugar decimal.

La multiplicación de cantidades $I_m Q_f$ y $I_n Q_g$ da como resultado el formato $I_{m+n} Q_{f+g}$. El resultado puede ser representado en la forma $I_m Q_f$ realizando un desplazamiento aritmético a la derecha, perdiendo así precisión.

La multiplicación por potencias de 2 puede implementarse como un desplazamiento aritmético a la izquierda y la división por una potencia de la raíz; como un desplazamiento aritmético a la derecha; En muchos procesadores, los cambios son más rápidos que la multiplicación y la división.

Punto Flotante

En informática, la aritmética de punto flotante es aritmética que utiliza la representación formulada de números reales como una aproximación para respaldar una compensación entre rango y precisión. Por esta razón, el cálculo de punto flotante a menudo se encuentra en sistemas que incluyen números reales muy pequeños y muy grandes.

Un número de punto flotante puede ser representado usando precisión simple (SP) con 32 bits; el bit 31 es el signo, los bits 30 al 23 representan el exponente y los bits 22 al 0 representan los bits decimales.

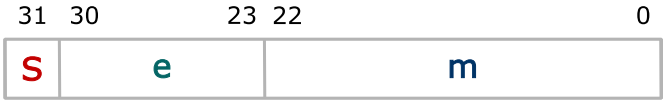

Figura 52 - Representación IEEE754 con 32 bits

Donde S es el bit de signo, M se denomina mantisa, E es el exponente de 8 bits, con un $OFFSET = +127$; tanto la mantisa como el exponente, son valores enteros con signo.

$$M = 1 + \sum_{i=1}^{23} m_i \cdot 2^{-i}$$

Con $1 \leq M < 2$, pues $m_0 = 1$; los números reales se pueden representar de forma binaria como:

$$Z = (-1)^S . M \times 2^{E-OFFSET}$$

En la SP, el rango del exponente es de $[-127,\ 127]$; la magnitud de los números representables en Punto Flotante es: $0.1111\ldots11\ x\ 2^{127} \approx 2^{127} \approx 10^{38}$ a $0.1000\ldots0\ x\ 2^{-127} \approx 10^{-38} \approx 2^{-128}$. Pero la precisión no va a superar los 24 bits $\Rightarrow$ Precisión máxima es $2^{24} \approx 10^{7}$, o sea 7 dígitos decimales. La mantisa limita la precisión y el exponente el rango.

Se llama “Unidad de Punto Flotante” (FPU), coloquialmente un coprocesador matemático, a un componente del µC especializado en el cálculo de operaciones aritméticas en coma flotante. No todos los µC tienen una FPU dedicada; en ese caso, el µC puede utilizar código para emular una función en coma flotante a través de la unidad aritmético-lógica (ALU), la cual reduce el costo del hardware a cambio de pérdida de velocidad. En la práctica, la forma en que estas operaciones se llevan a cabo en la lógica digital puede ser bastante compleja.

Volver al informe

Apéndice C

Para el diseño del PCB, se utilizó el software libre KiCad. A continuación se muestra el diagrama esquemático de los circuitos analógico y digital, respectivamente (Figura 53 y Figura 54). La distribución de los footprints y caminos de las 4 capas del PCB (Figura 55Figura 56, Figura 57, Figura 58) y la representación gráfica en tres dimensiones (3D) de la placa con algunos de sus componentes (Figura 59Figura 60, Figura 61).

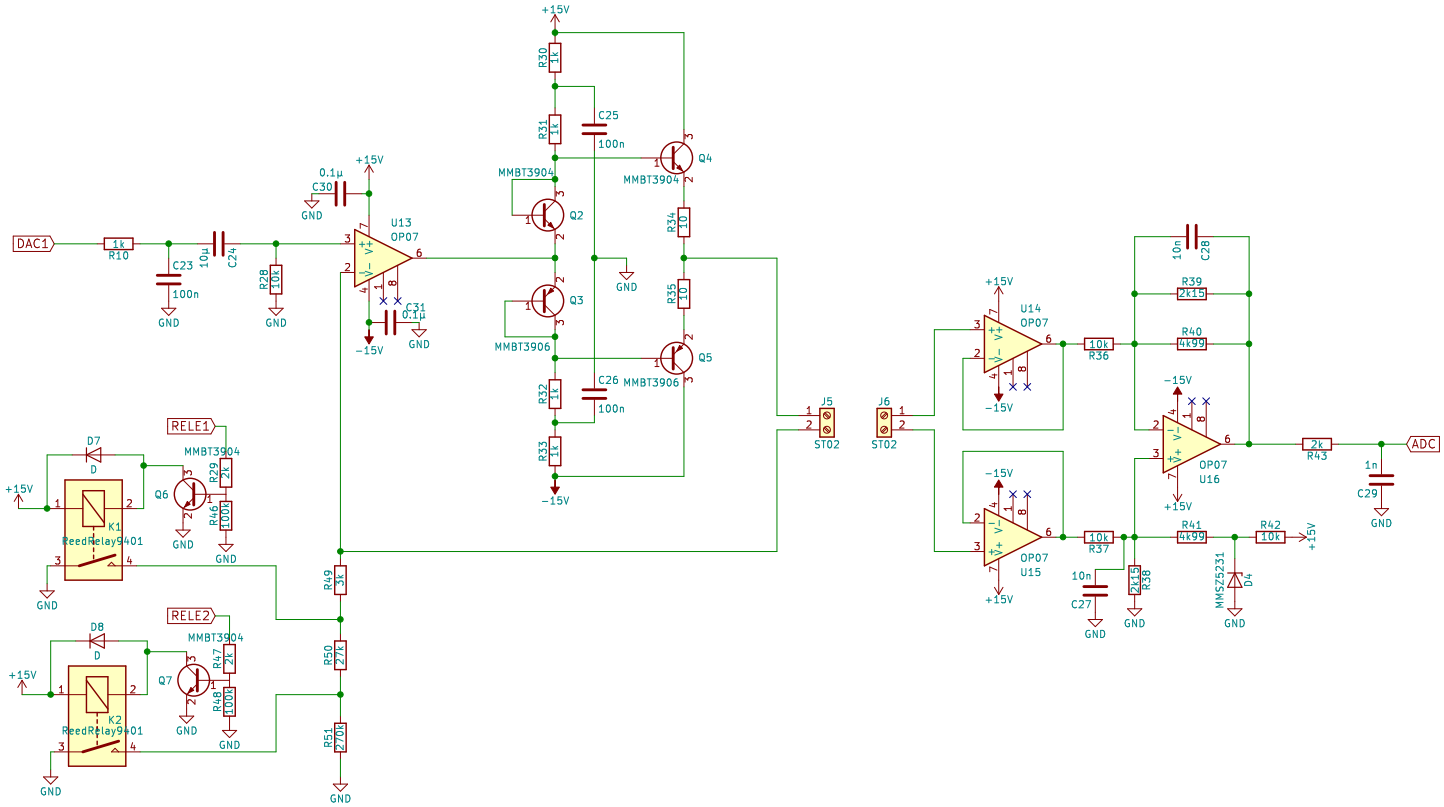

Figura 53 - Diagrama Esquemático del Circuito Analógico

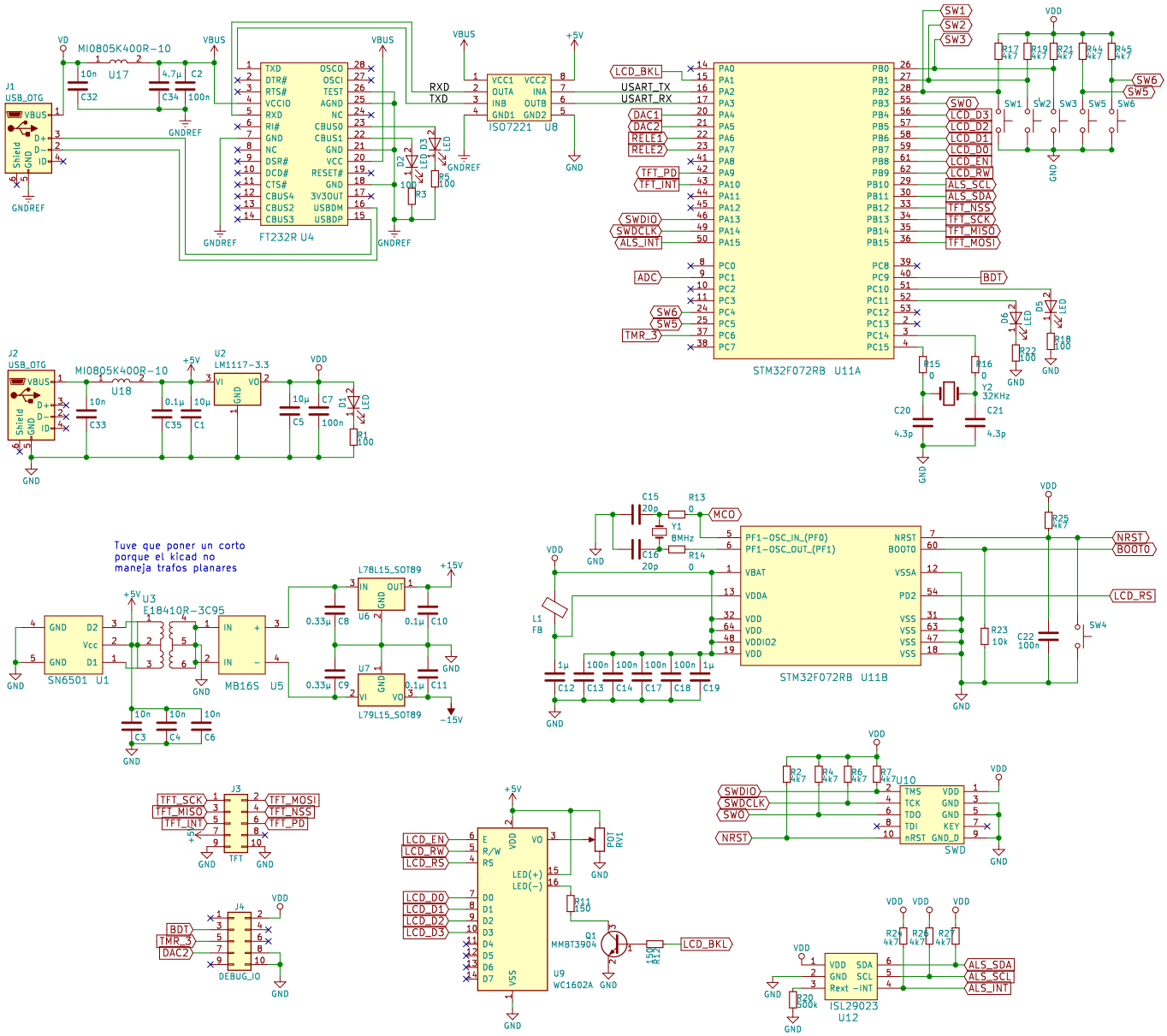

Figura 54- Diagrama Esquemático del Circuito Digital

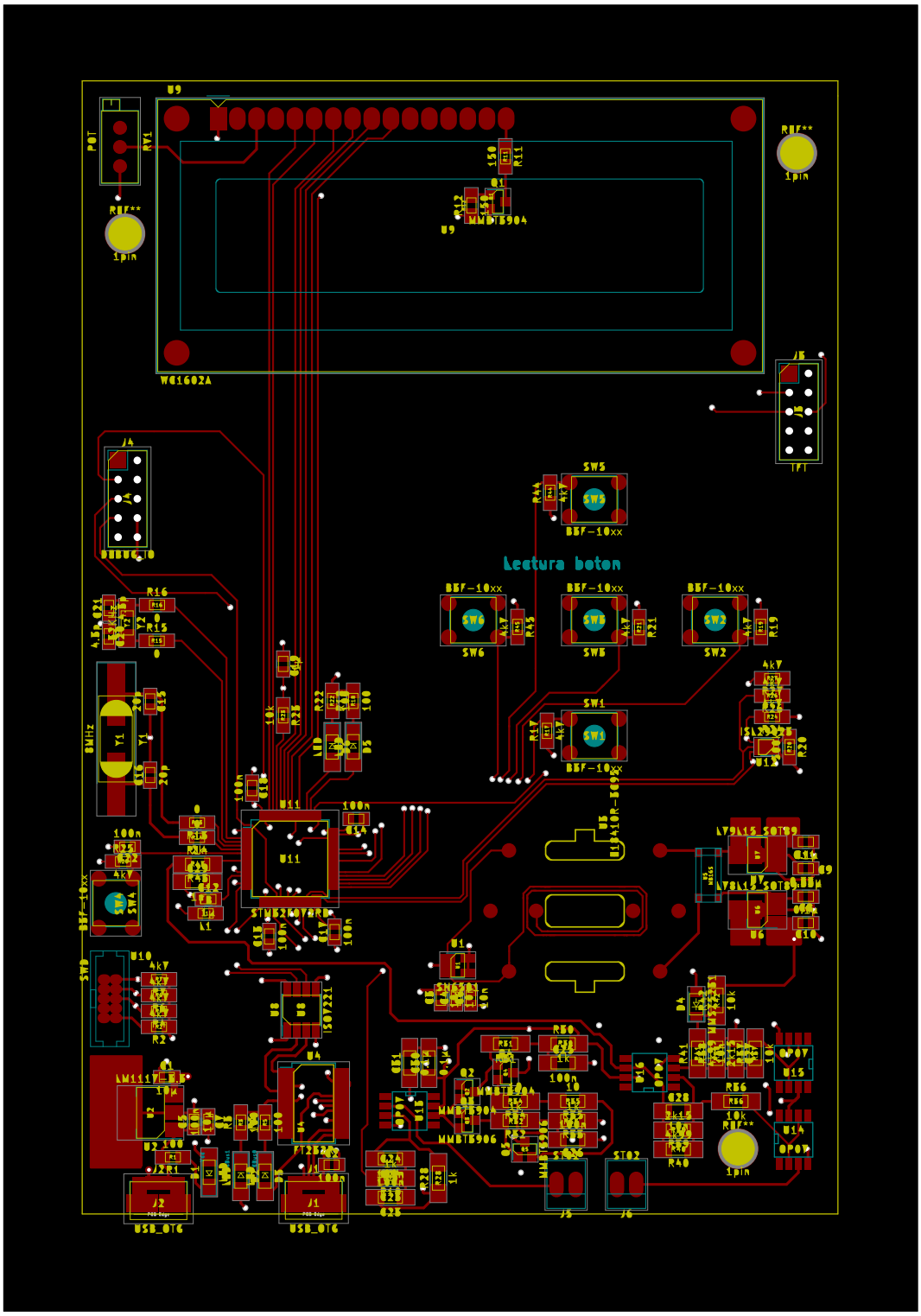

Figura 55 - Capa Top del PCB

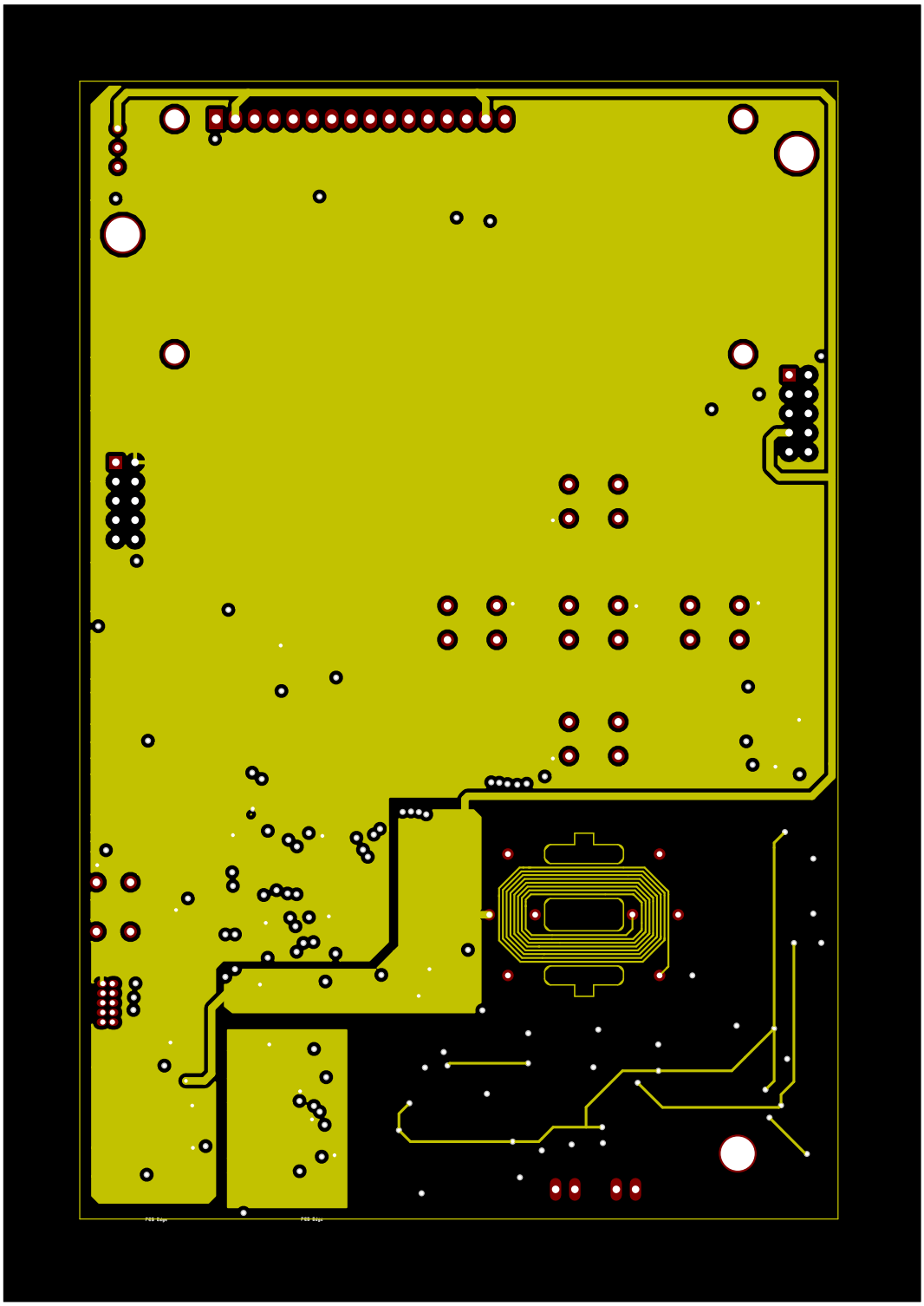

Figura 56- Capa Mid1, Distribución de los Caminos de Alimentación del PCB

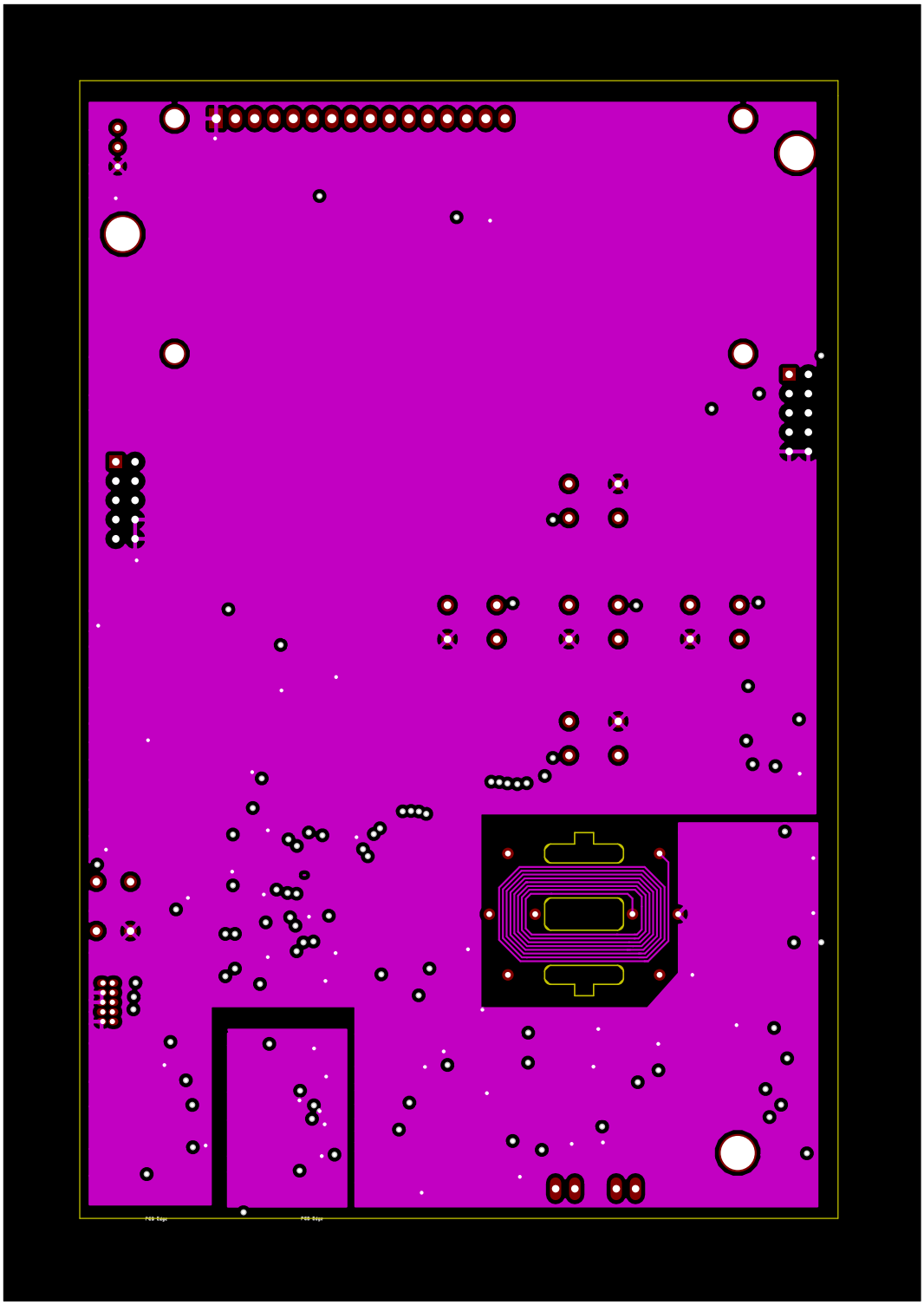

Figura 57- Capa Mid2, Distribución de las Masas del PCB

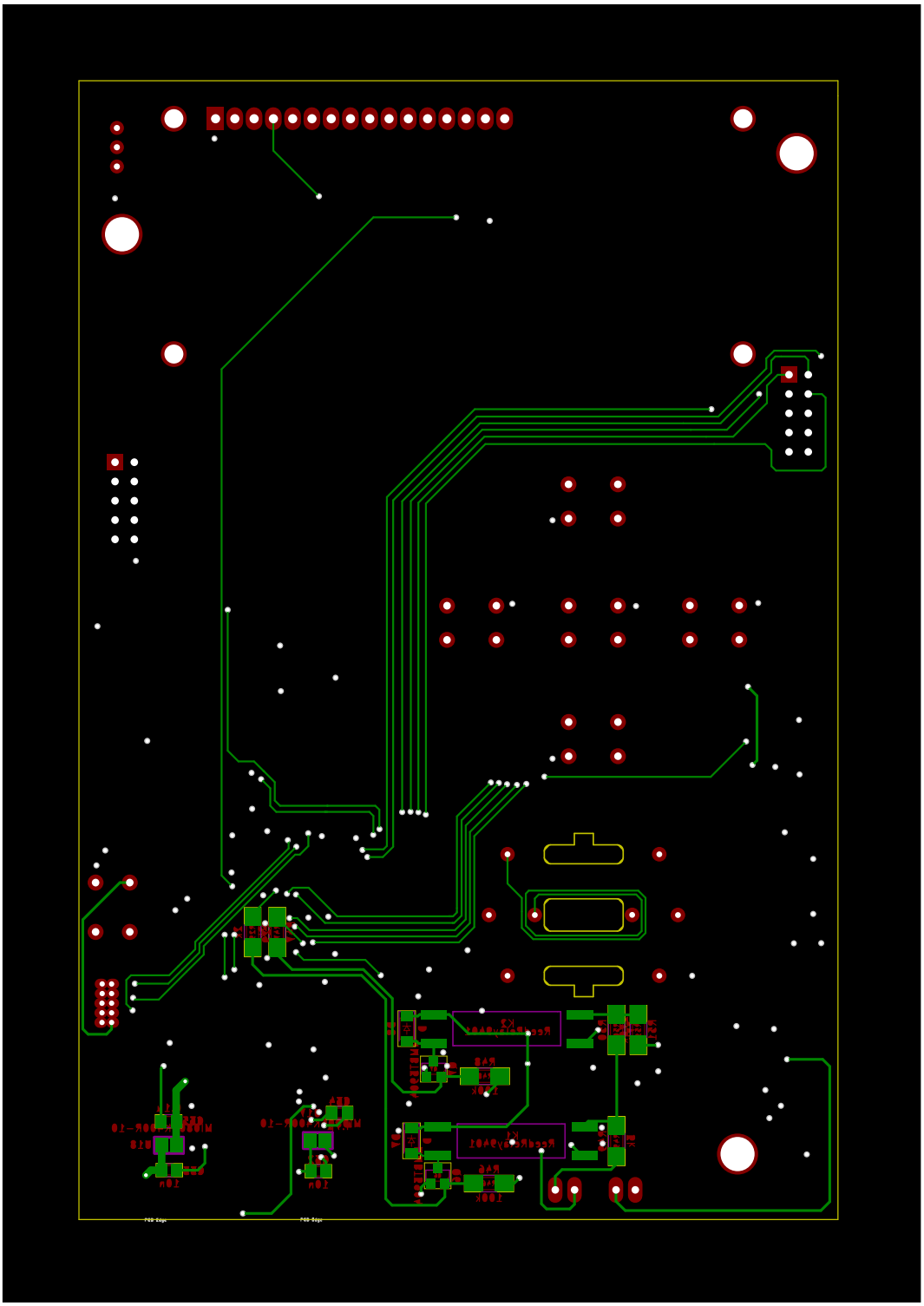

Figura 58- Capa Bottom del PCB

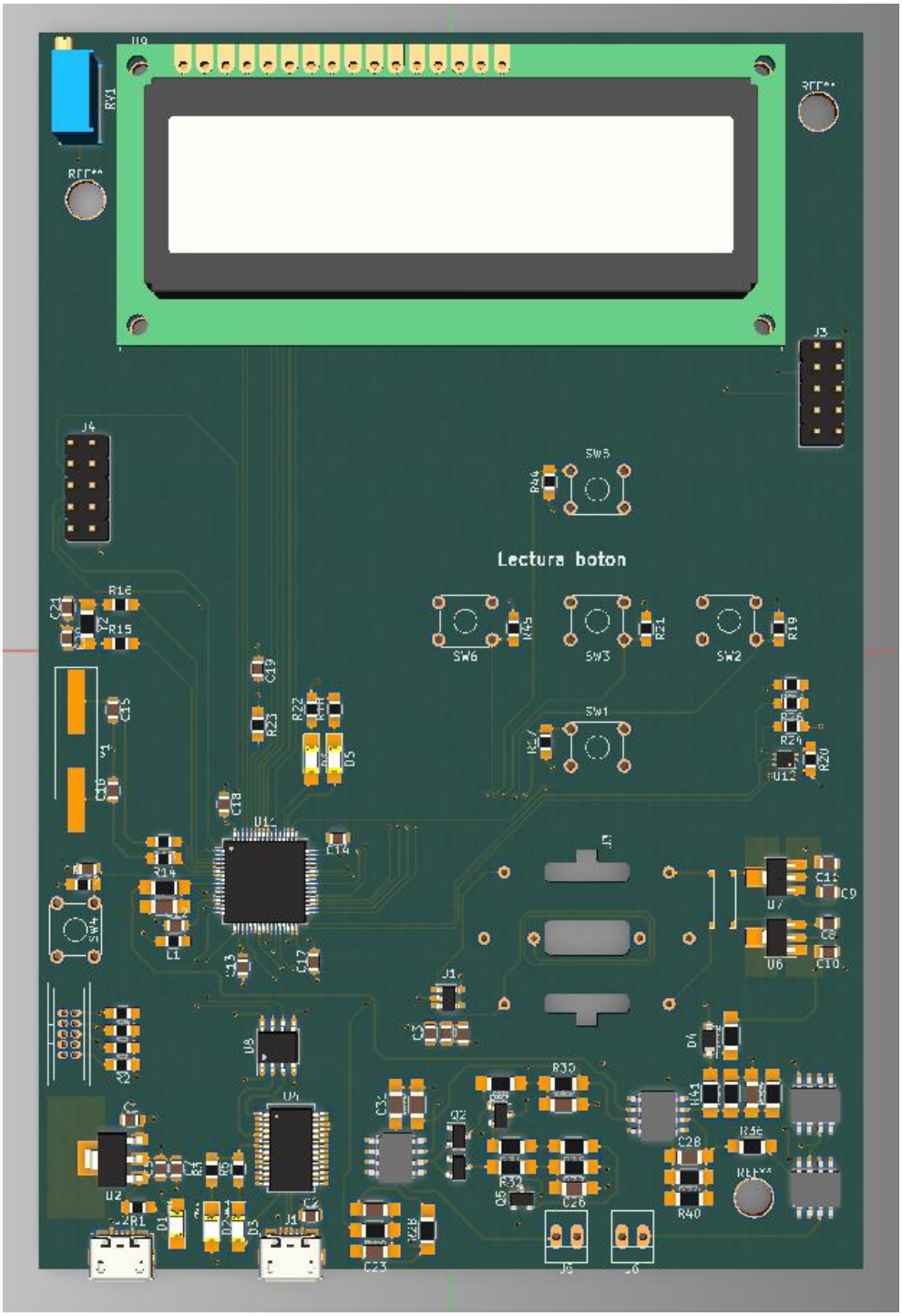

Figura 59- Vista 3D Frontal

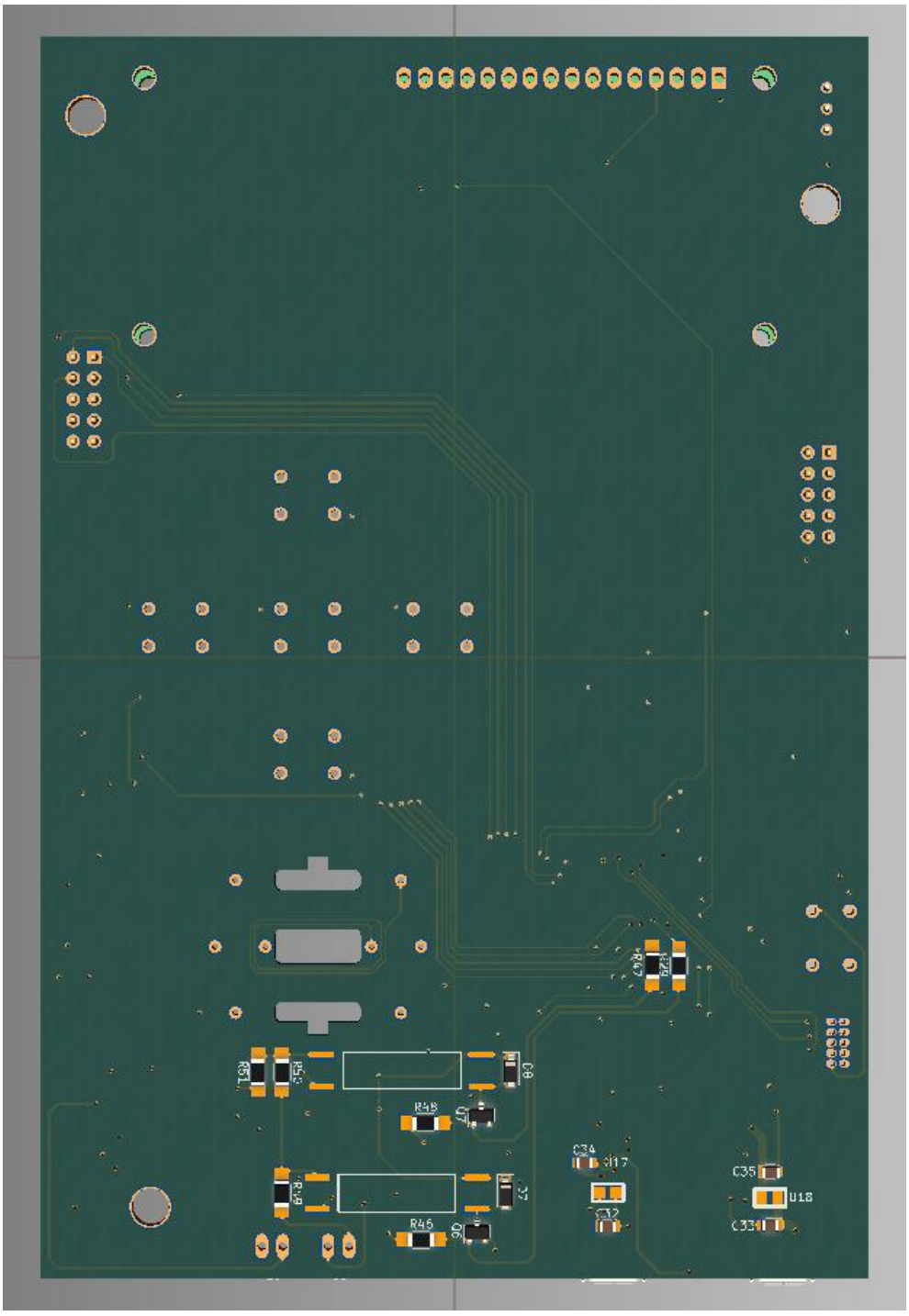

Figura 60 - Vista 3D Trasera

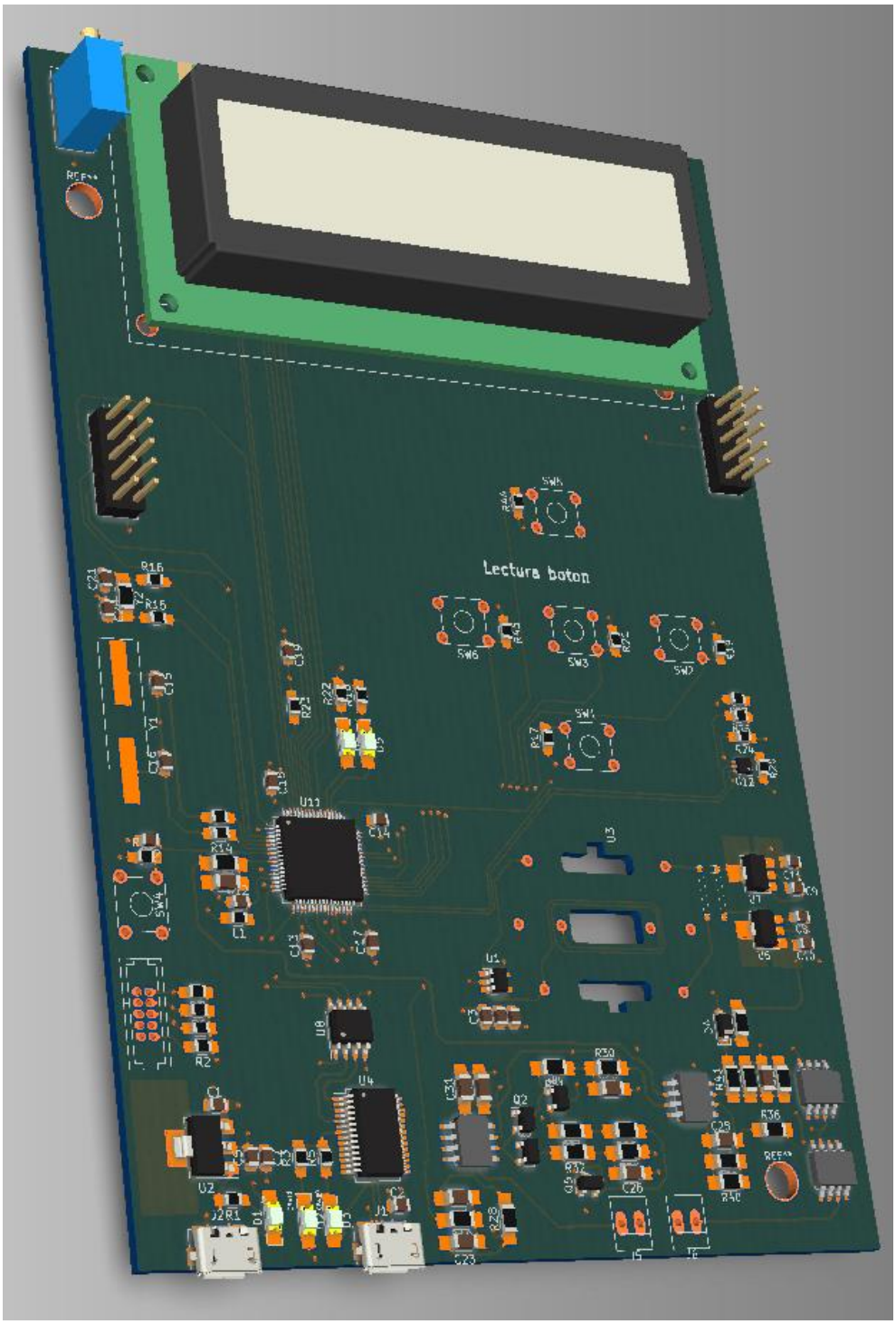

Figura 61 - Vista 3D, Plano Inclinado

Apéndice D

```
// C library includes
#include <stdint.h>
#include <stdio.h>
#include <stdlib.h>
#include <stdbool.h>
#include <string.h>
#include <math.h>

// stm32 std peripheral library includes
#include "stm32f0xx.h"
#include "stm32f0xx_nucleo.h"
#include "stm32f0xx_gpio.h"             // General Port Input/Output Librery
#include "stm32f0xx_tim.h"              // Timer Librery
#include "stm32f0xx_rcc.h"              // Reset and Clock Librery
#include "stm32f0xx_misc.h"             // interrup Librery
#include "stm32f0xx_dac.h"              // DAC librery
#include "stm32f0xx_usart.h"
#include "stm32f0xx_i2c.h"
#include "stm32f0xx_exti.h"
//#include "stm32f0xx_it.h"

// local includes
#include "fpmath.h"
#include "lutq.h"


#undef USE_DBG_SW
#define Q                   ( 20 )
#define SYSCLOCK_FREQ       ( 48e6 )    // [Hz]
#define TIM3_PERIOD_S       ( 50e-6 )   // [s]
#define SINUSOID_FREQ       ( 200.0 )   // [Hz]
#define TIM3_PERIOD_CNT     ( TIM3_PERIOD_S * SYSCLOCK_FREQ )
#define DELTA_THETA         ( LUT_LEN / (1.0 / (SINUSOID_FREQ * TIM3_PERIOD_S)) )
#define ADC_GAIN            TOFIX(3.3 / 4096.0, Q) // ganancia de 3.3Volt a +/-10V
#define ADC_MEAN            TOFIX(1.646158218, Q) // Gain experim de 3.3V a +/-10V
#define DAC_GAIN            TOFIX(2048.0 / 1.65, Q) // ganancia de 3.3V a +/-10V

//#define ADC_MEAN          TOFIX(3.3 / 2.0, Q) // Gain 3.3Volt a +/-10Volt
#define N                   100
#define INV_N               TOFIX(1.0 / (double) N, Q)
#define M_PI_2              1.57079632679489661923
#define M_SQRT2             1.41421356237309504880
#define RMS_CONST           TOFIX( (100.0 / 15.0) * M_PI_2 / M_SQRT2 / (double)N, Q)
#define MED_CONST           TOFIX( 4096/3.3, Q)
#define IRMS_R1             TOFIX(0.5 * M_SQRT2 * 1.21 / 0.003 , Q)
#define IRMS_R2             TOFIX(0.5 * M_SQRT2 * 1.21 / 0.030 , Q)
#define IRMS_R3             TOFIX(0.5 * M_SQRT2 * 1.21 / 0.300 , Q)
#define K                   TOFIX((M_LN2)/(2*M_PI),Q)

# define MillisecondsIT          ( 10 )

volatile fixq_t amp = TOFIX(0.40, 30);
volatile fixq_t off = TOFIX(0.5, 30);
volatile int16_t theta = 0;
volatile fixq_t dac_value;
```

```
volatile int32_t adc_value = 0;
volatile uint16_t medio;
volatile int32_t calib = 0;
volatile int32_t adc_offset = 1960;

// filtro pasa altos (F_sample= 50e-6)
#define CTE1_HPF   TOFIX(0.9993718788, Q)      //HPF:e^(-(2pi)*(2Hz)*(50µs)) FC=2Hz,
#define CTE1_LPF   TOFIX(0.2078795764, Q)      //LPF:e^(-(2pi)*(5kHz)*(50µs))FC=5kHz
#define CTE2_LPF   TOFIX(0.7921204236, Q)      //1-CTE1_LPF
#define CTE1B_LPF  TOFIX(0.9993718788, Q)      //LPF:e^(-(2pi)*(5kHz)*(50µs))FC=5kHz
#define CTE2B_LPF  TOFIX(1-0.9993718788, Q)    //1-CTE1_LPF



volatile fixq_t v_in[2] = {0, 0};       //Tension entrada

volatile fixq_t v_inf[2] = {0, 0};      //Tension entrada filtrada

volatile fixq_t v_fin[2] = {0, 0};      //Tension salida

typedef enum {         /* BOTON */
    OK    = 0,      /* Center Input */
    RIGHT,          /* Right Button */
    LEFT,           /* Left Button */
    DOWN,           /* Down Button */
    UP,             /* Up Button */
       NONE         /* Limpia la variable Button*/
} TInputs;

typedef enum {            // ** ESTADO/PANTALLA **
       MENU   = 1,        // Opcion del Menú - Pantalla 1
       CALIB,             // Opcion del Menú - Pantalla 2
       RANGO1,            // Configuracion de Relés - RANGO1
       RANGO2,            // Configuracion de Relés - RANGO2
       RANGO3,            // Configuracion de Relés - RANGO3
       MEDIC,             // Opcion del Menú - Pantalla 3
       CONEX_ELECTR,      // Aviso conexion Electrodos - Pantalla 4
       TX_PC              // Transmitiendo Datos a PC

} Screen;

volatile TInputs evento = NONE;
volatile Screen estado = MENU;

volatile bool mostrar = false;

int RSB = 0; // Register select signal Bit
int RWB = 0; // Data Read / Write Bit

int init = 0;
int fin = 1;
volatile fixq_t X[N] = {0,};
volatile fixq_t Y[2] = {0,0};

volatile fixq_t U;
volatile fixq_t Vrms;
volatile int32_t Vmed;
volatile fixq_t TxValue;
```

```
volatile fixq_t Irms;
int contador = 299999;


// para generar una base de tiempo de muestreo de 50us
static const uint32_t timer_period = TIM3_PERIOD_CNT;
static void dbg_gpio_init(void)
{
    GPIO_InitTypeDef port;

    // enable clock
    RCC_AHBPeriphClockCmd(RCC_AHBPeriph_GPIOC, ENABLE);

    // configure gpio
    GPIO_StructInit(&port);
    port.GPIO_Pin = GPIO_Pin_9;
    port.GPIO_Mode = GPIO_Mode_OUT;
    port.GPIO_Speed = GPIO_Speed_50MHz;
    GPIO_Init(GPIOC, &port);

    // set initial state
    GPIO_WriteBit(GPIOC, GPIO_Pin_9, Bit_SET);
}


static void dbg_on(void)
{
    GPIO_SetBits(GPIOC, GPIO_Pin_9);
}


static void dbg_off(void)
{
    GPIO_ResetBits(GPIOC, GPIO_Pin_9);
}


// ADC configuration
void adc_gpio_init(void)
{
    GPIO_InitTypeDef port;

    // enable gpio clocks
    RCC_AHBPeriphClockCmd(RCC_AHBPeriph_GPIOC, ENABLE);

    // configure PC1 as ADC_IN11 input
    GPIO_StructInit(&port);
    port.GPIO_Pin = GPIO_Pin_1;
    port.GPIO_Mode = GPIO_Mode_AN;
    port.GPIO_PuPd = GPIO_PuPd_NOPULL;
    GPIO_Init(GPIOC, &port);

    // configure PC0 as ADC_IN11 input / Se configura PC0 para verificar que PC1
esta Ok
      GPIO_StructInit(&port);
      port.GPIO_Pin = GPIO_Pin_0;
      port.GPIO_Mode = GPIO_Mode_AN;
      port.GPIO_PuPd = GPIO_PuPd_NOPULL;
```

```
      GPIO_Init(GPIOC, &port);
}


void adc_init(void)
{
    ADC_InitTypeDef adc1;
    //---------------------------------------
    ADC_InitTypeDef adc0;
    //-----------------------------------------------

    // enable adc system clock
    RCC_APB2PeriphClockCmd(RCC_APB2Periph_ADC1, ENABLE);

    // initialize adc
    ADC_StructInit(&adc1);
    adc1.ADC_Resolution = ADC_Resolution_12b;
    adc1.ADC_ContinuousConvMode = DISABLE;
    adc1.ADC_ExternalTrigConv = ADC_ExternalTrigConv_T3_TRGO;
    adc1.ADC_ExternalTrigConvEdge = ADC_ExternalTrigConvEdge_Rising;
    adc1.ADC_DataAlign = ADC_DataAlign_Right;
    adc1.ADC_ScanDirection = ADC_ScanDirection_Upward;

    // Config channels
    ADC_ChannelConfig(ADC1, ADC_Channel_11, ADC_SampleTime_7_5Cycles);   //  el  ADC
toma 239.5 ciclos de reloj para hacer muestra (1/48e6)

    /* *** ADC Calibration *** */
    // Calibrate ADC before enabling
    // ADC_GetCalibrationFactor(ADC1);
    ADC_Init(ADC1, &adc1);

    // Enable ADC interrupt
    ADC_ITConfig(ADC1, ADC_IT_EOC, ENABLE);

    // Enable timer interrupt handler
    NVIC_InitTypeDef nvicStructure;
    nvicStructure.NVIC_IRQChannel = ADC1_COMP_IRQn;
    nvicStructure.NVIC_IRQChannelPriority = 0;
    nvicStructure.NVIC_IRQChannelCmd = ENABLE;
    NVIC_Init(&nvicStructure);

    // ADC_GetCalibrationFactor(ADC1);
    calib = ADC_GetCalibrationFactor(ADC1);
    // enable ADC1
    ADC_Cmd(ADC1, ENABLE);
    // Wait until ADC enabled

    while(ADC_GetFlagStatus(ADC1, ADC_FLAG_ADEN) == RESET);

    ADC_Cmd(ADC1, ENABLE);
    ADC_StartOfConversion(ADC1);
}


//SWITCHES
void sw_gpio_init(void)
{
```

```
#if !defined(USE_DBG_SW)
       // define el tipo de variable "sw" que es de tipo GPIO_InitTypeDef (generico)
       GPIO_InitTypeDef sw_gpio;
       EXTI_InitTypeDef sw_exti;
    NVIC_InitTypeDef sw_nvic;

       // GPIOB clock enable
       RCC_AHBPeriphClockCmd(RCC_AHBPeriph_GPIOB, ENABLE);
       // GPIOC clock enable
       RCC_AHBPeriphClockCmd(RCC_AHBPeriph_GPIOC, ENABLE);
       // Habilito el clock del periferico SYSCFG
       RCC_APB2PeriphClockCmd(RCC_APB2Periph_SYSCFG, ENABLE);

        // Configuro PB0 PB1 y PB2
       GPIO_StructInit(&sw_gpio);
       sw_gpio.GPIO_Pin =  GPIO_Pin_0 | GPIO_Pin_1 | GPIO_Pin_2;
       sw_gpio.GPIO_Mode = GPIO_Mode_IN;
       sw_gpio.GPIO_OType = GPIO_OType_OD;
       sw_gpio.GPIO_PuPd = GPIO_PuPd_UP;      // Este pin en el kit no tiene pull up
       GPIO_Init(GPIOB, &sw_gpio);

       // Initialize SWITCH line 0, line 1 y line 2
    EXTI_StructInit(&sw_exti);
    sw_exti.EXTI_Line = EXTI_Line0 | EXTI_Line1 | EXTI_Line2;
    sw_exti.EXTI_Mode = EXTI_Mode_Interrupt;
    sw_exti.EXTI_Trigger = EXTI_Trigger_Falling ;
    sw_exti.EXTI_LineCmd = ENABLE;
    EXTI_Init(&sw_exti);


    //Select the input source pin for the EXTI line (PB0)
    //(Vinculo PB0, PB1 y PB2 con la linea 0, 1 y 2 del EXTI)
    SYSCFG_EXTILineConfig(EXTI_PortSourceGPIOB, EXTI_PinSource0);
    SYSCFG_EXTILineConfig(EXTI_PortSourceGPIOB, EXTI_PinSource1);
    SYSCFG_EXTILineConfig(EXTI_PortSourceGPIOB, EXTI_PinSource2);

       // Configuro PB0 PB1 y PB2
       GPIO_StructInit(&sw_gpio);
       sw_gpio.GPIO_Pin =  GPIO_Pin_4 | GPIO_Pin_5;
       sw_gpio.GPIO_Mode = GPIO_Mode_IN;
       sw_gpio.GPIO_OType = GPIO_OType_OD;
       sw_gpio.GPIO_PuPd = GPIO_PuPd_UP;      // Este pin en el kit no tiene pull up
       GPIO_Init(GPIOC, &sw_gpio);

       // Initialize SWITCH line 4 y line 5
   EXTI_StructInit(&sw_exti);
   sw_exti.EXTI_Line = EXTI_Line4 | EXTI_Line5;
   sw_exti.EXTI_Mode = EXTI_Mode_Interrupt;
   sw_exti.EXTI_Trigger = EXTI_Trigger_Falling ;
   sw_exti.EXTI_LineCmd = ENABLE;
   EXTI_Init(&sw_exti);

   //Select the input source pin for the EXTI line (PC4 y PC5)
   //(Vinculo PC4 y PC5 con la linea 4 y 5 del EXTI)
   SYSCFG_EXTILineConfig(EXTI_PortSourceGPIOC, EXTI_PinSource4);
   SYSCFG_EXTILineConfig(EXTI_PortSourceGPIOC, EXTI_PinSource5);
```

```
    // Habilito la interrupcion a nivel NVIC para PB0 y PB1
    sw_nvic.NVIC_IRQChannel = EXTI0_1_IRQn;
    sw_nvic.NVIC_IRQChannelPriority = 3;    // 3 es la prioridad más baja
    sw_nvic.NVIC_IRQChannelCmd = ENABLE;
    NVIC_Init(&sw_nvic);

    // Habilito la interrupcion a nivel NVIC para PB2
       sw_nvic.NVIC_IRQChannel = EXTI2_3_IRQn;
       sw_nvic.NVIC_IRQChannelPriority = 3;    // 3 es la prioridad más baja
       sw_nvic.NVIC_IRQChannelCmd = ENABLE;
       NVIC_Init(&sw_nvic);

    // Habilito la interrupcion a nivel NVIC para PB4 y PB5
       sw_nvic.NVIC_IRQChannel = EXTI4_15_IRQn;
       sw_nvic.NVIC_IRQChannelPriority = 3;    // 3 es la prioridad más baja
       sw_nvic.NVIC_IRQChannelCmd = ENABLE;
       NVIC_Init(&sw_nvic);

}

// USART configuration
void usart_gpio_init(void)
{
       GPIO_InitTypeDef portTx;
       GPIO_InitTypeDef portRx;

       // Enables or disables the High Speed APB (APB2) peripheral clock.
       RCC_AHBPeriphClockCmd(RCC_AHBPeriph_GPIOA, ENABLE);         //

       // Configure PA2 as USART1 to Tx
       GPIO_StructInit(&portTx);
       portTx.GPIO_Pin = GPIO_Pin_2;
       portTx.GPIO_Mode = GPIO_Mode_AF;
       portTx.GPIO_OType = GPIO_OType_PP;
       portTx.GPIO_PuPd = GPIO_PuPd_UP;
       portTx.GPIO_Speed = GPIO_Speed_2MHz;

       //Initializes the GPIOA peripheral according to the specified parameters in
the GPIO_InitStruct.
       GPIO_Init(GPIOA, &portTx);

       /* GPIOA PA3 alternative function Rx */
       GPIO_StructInit(&portRx);
       portRx.GPIO_Pin = GPIO_Pin_3;
       portRx.GPIO_Speed = GPIO_Speed_2MHz;
       portRx.GPIO_OType = GPIO_OType_PP;
       portRx.GPIO_PuPd = GPIO_PuPd_NOPULL;
       portRx.GPIO_Mode = GPIO_Mode_AF;

       //Initializes the GPIOA peripheral according to the specified parameters in
the GPIO_InitStruct.
       GPIO_Init(GPIOA, &portRx);

       // Writes data to the specified GPIO data port.
       GPIO_PinAFConfig(GPIOA, GPIO_PinSource2, GPIO_AF_1);
       GPIO_PinAFConfig(GPIOA, GPIO_PinSource3, GPIO_AF_1);

}
```

```
void usart_init(void)
{
	USART_InitTypeDef usart2;

	// Enables or disables the Low Speed APB (APB1) peripheral clock.
	RCC_APB1PeriphClockCmd(RCC_APB1Periph_USART2 , ENABLE);
	//Habilito el Clock para la USART2, ubicada en PA2, en el PIN 8, de CN9  o
pin 23 de CN10

	// Fills each USART_InitStruct member with its default value.
	USART_StructInit(&usart2);

	//VER!!!   - Modo Rx es necesario???
	usart2.USART_BaudRate = 115200;		//9600;
	usart2.USART_WordLength = USART_WordLength_8b;
	usart2.USART_StopBits = USART_StopBits_1;
	usart2.USART_Parity = USART_Parity_No ;
	usart2.USART_HardwareFlowControl = USART_HardwareFlowControl_None;
	usart2.USART_Mode = USART_Mode_Rx | USART_Mode_Tx;		//   USART1
habilitada para Tx y Rx
	//     usart1.USART_HardwareFlowControl = USART_HardwareFlowControl_None;

	// Initializes the USARTx peripheral according to the specified parameters in
the USART_InitStruct .
	USART_Init(USART2, &usart2);

	// Enables or disables the specified USART peripheral.
	USART_Cmd(USART2, ENABLE);
}


// LED configuration
void led_gpio_init(void)
{
	// declara la variable led_cfg como GPIO_InitTypeDef
	GPIO_InitTypeDef led_cfg;

    // GPIOC clock enable
    RCC_AHBPeriphClockCmd(RCC_AHBPeriph_GPIOC, ENABLE);

    // Fills each GPIO_InitStruct member with its default value.
    GPIO_StructInit(&led_cfg);

    // Configura PC10 y PC11 as LEDs
    led_cfg.GPIO_Pin =  GPIO_Pin_10 | GPIO_Pin_11;
    led_cfg.GPIO_Mode = GPIO_Mode_OUT;
    led_cfg.GPIO_PuPd = GPIO_PuPd_NOPULL;
    GPIO_Init(GPIOC, &led_cfg);

}


// RELE configuration
void rele_gpio_init(void)
{
	// declara la variable rele_cfg como GPIO_InitTypeDef
```

```
	GPIO_InitTypeDef rele_cfg;

    // GPIOC clock enable
    RCC_AHBPeriphClockCmd(RCC_AHBPeriph_GPIOA, ENABLE);

    // Fills each GPIO_InitStruct member with its default value.
    GPIO_StructInit(&rele_cfg);

    // Configura PA6 y PA7 as LEDs
    rele_cfg.GPIO_Pin =  GPIO_Pin_6 | GPIO_Pin_7;
    rele_cfg.GPIO_Mode = GPIO_Mode_OUT;
    rele_cfg.GPIO_PuPd = GPIO_PuPd_NOPULL;
    GPIO_Init(GPIOA, &rele_cfg);
}


void rele1_en(int LEB)
{

	if (LEB == 1) {
		GPIO_SetBits(GPIOA, GPIO_Pin_6);
	}
	else if (LEB == 0) {
		GPIO_ResetBits(GPIOA,GPIO_Pin_6);
	}
}


void rele2_en(int LEB)
{

	if (LEB == 1) {
		GPIO_SetBits(GPIOA, GPIO_Pin_7);
	}
	else if (LEB == 0) {
		GPIO_ResetBits(GPIOA,GPIO_Pin_7);
	}
}


// DAC configuration
void dac_gpio_init(void)
{
    GPIO_InitTypeDef port;

    // GPIOA clock enable
    RCC_AHBPeriphClockCmd(RCC_AHBPeriph_GPIOA, ENABLE);

    // Configure PA4 as DAC_OUT1
    GPIO_StructInit(&port);
    port.GPIO_Pin =  GPIO_Pin_4;
    port.GPIO_Mode = GPIO_Mode_AN;
    port.GPIO_PuPd = GPIO_PuPd_NOPULL;
    GPIO_Init(GPIOA, &port);

    // Configure PA5 as DAC_OUT2
    GPIO_StructInit(&port);
    port.GPIO_Pin =  GPIO_Pin_5;
```

```
    port.GPIO_Mode = GPIO_Mode_AN;
    port.GPIO_PuPd = GPIO_PuPd_NOPULL;
    GPIO_Init(GPIOA, &port);
}


void dac_init(void)
{
    DAC_InitTypeDef dac0;
    DAC_InitTypeDef dac1;

    // enable dac clock
    RCC_APB1PeriphClockCmd(RCC_APB1Periph_DAC, ENABLE);

    // initialize dac
    DAC_StructInit(&dac0);
    dac0.DAC_Trigger = DAC_Trigger_None;
    dac0.DAC_OutputBuffer = DAC_OutputBuffer_Enable;

    // Initializes the DAC peripheral according to the specified parameters in the
DAC_InitStruct.
    DAC_Init(DAC_Channel_1, &dac0);

    // Enables or disables the specified DAC channel.
    DAC_Cmd(DAC_Channel_1, ENABLE);

//-------------------------------------------------------------------------------
-----------------

    // Fills each DAC_InitStruct member with its default value.
    DAC_StructInit(&dac1);
    dac1.DAC_Trigger = DAC_Trigger_None;                          //   Change   some
default values
    dac1.DAC_OutputBuffer = DAC_OutputBuffer_Enable;

    // Initializes the DAC peripheral according to the specified parameters in the
DAC_InitStruct.
    DAC_Init(DAC_Channel_2, &dac1);

    // Enables or disables the specified DAC channel.
    DAC_Cmd(DAC_Channel_2, ENABLE);
}


// TIMER_3 configuration
void tim_gpio_init(void)
{
    GPIO_InitTypeDef port;

    // enable gpio clocks
    RCC_AHBPeriphClockCmd(RCC_AHBPeriph_GPIOC, ENABLE);

    // configure PC6 as TIM3 pwm output
    GPIO_StructInit(&port);
    port.GPIO_Mode = GPIO_Mode_AF;
    port.GPIO_Pin = GPIO_Pin_6;
    port.GPIO_Speed = GPIO_Speed_50MHz;
    GPIO_Init(GPIOC, &port);
```

```
}


void tim_init(void)
{
    TIM_TimeBaseInitTypeDef timer;

    // Enable timer clock
    RCC_APB1PeriphClockCmd(RCC_APB1Periph_TIM3, ENABLE);

    // Initialize timer
    TIM_TimeBaseStructInit(&timer);
    timer.TIM_Prescaler = 0;       // no prescaler for TIM3 clock, TIM3_CLK =
SystemCoreClock
    timer.TIM_Period = timer_period;
    timer.TIM_ClockDivision = TIM_CKD_DIV1;
    timer.TIM_CounterMode = TIM_CounterMode_Up;
    timer.TIM_RepetitionCounter = 0;
    TIM_TimeBaseInit(TIM3, &timer);

    // configure timer trigger output for using is as ADC start of conversion
signal
    TIM_SelectOutputTrigger(TIM3, TIM_TRGOSource_Update);

    // Enable timer
    TIM_Cmd(TIM3, ENABLE);
}


// LCD configuration
void lcd_gpio_init(void)

{
       // GPIO Bits de Control (GPIOB)
       GPIO_InitTypeDef lcdControlB;

       // Enables or disables the High Speed APB (APB2) peripheral clock.
       RCC_AHBPeriphClockCmd(RCC_AHBPeriph_GPIOB, ENABLE);

       // Configure PB8-PB9-PD2 as control bits (LED)
       GPIO_StructInit(&lcdControlB);

       lcdControlB.GPIO_Pin = GPIO_Pin_8 | GPIO_Pin_9;
       lcdControlB.GPIO_Mode = GPIO_Mode_OUT;
       lcdControlB.GPIO_OType =  GPIO_OType_PP;
       lcdControlB.GPIO_PuPd = GPIO_PuPd_DOWN ;      //bits se mantienen en estado
bajo
       lcdControlB.GPIO_Speed = GPIO_Speed_Level_1 ;// 2 MHz Idem Led Speed

       //Initializes the GPIOB peripheral according to the specified parameters in
the GPIO_InitStruct.
       GPIO_Init(GPIOB, &lcdControlB);

       // GPIO Bits de Control (GPIOD)
       GPIO_InitTypeDef lcdControlD;

       // Enables or disables the High Speed APB (APB2) peripheral clock.
       RCC_AHBPeriphClockCmd(RCC_AHBPeriph_GPIOD, ENABLE);
```

```
	// Configure PD2 as control bits (LED)
	GPIO_StructInit(&lcdControlD);

	lcdControlD.GPIO_Pin = GPIO_Pin_2;
	lcdControlD.GPIO_Mode = GPIO_Mode_OUT;
	lcdControlD.GPIO_OType =  GPIO_OType_PP;
	lcdControlD.GPIO_PuPd = GPIO_PuPd_DOWN ;
	lcdControlD.GPIO_Speed = GPIO_Speed_Level_1 ;

	//Initializes the GPIOB peripheral according to the specified parameters in
the GPIO_InitStruct.
	GPIO_Init(GPIOD, &lcdControlD);


	// GPIO Datos
	GPIO_InitTypeDef lcdData;

	// Enables or disables the High Speed APB (APB2) peripheral clock.
	RCC_AHBPeriphClockCmd(RCC_AHBPeriph_GPIOB, ENABLE);		//  Habilito  el
Clock para la USART1_TX , ubicada en PA9, en el PIN 1, de CN5 (usb DM)

	// Configure PB3-PB4-PB5-PB6 as Data bits (LCD)
	GPIO_StructInit(&lcdData);


	lcdData.GPIO_Pin = GPIO_Pin_7 | GPIO_Pin_4 | GPIO_Pin_5 | GPIO_Pin_6;    //
para PLACA ANA
	lcdData.GPIO_Mode = GPIO_Mode_OUT;
	lcdData.GPIO_OType =  GPIO_OType_PP;
	lcdData.GPIO_PuPd = GPIO_PuPd_DOWN ;
	lcdData.GPIO_Speed = GPIO_Speed_Level_1 ;            // 2 MHz Idem Led Speed

	//Initializes the GPIOB peripheral according to the specified parameters in
the GPIO_InitStruct.
	GPIO_Init(GPIOB, &lcdData);


	// declara la variable lcd_bkl como GPIO_InitTypeDef con el fin de activar el
BackLight
	GPIO_InitTypeDef lcd_bkl;

    // GPIOA clock enable
    RCC_AHBPeriphClockCmd(RCC_AHBPeriph_GPIOA, ENABLE);

    // Fills each GPIO_InitStruct member with its default value.
    GPIO_StructInit(&lcd_bkl);

    // Configura Backlight del LCD
    lcd_bkl.GPIO_Pin =  GPIO_Pin_1;
    lcd_bkl.GPIO_Mode = GPIO_Mode_OUT;
    lcd_bkl.GPIO_PuPd = GPIO_PuPd_NOPULL;
    GPIO_Init(GPIOA, &lcd_bkl);

    GPIO_SetBits(GPIOA, GPIO_Pin_1);

}
```

```
void lcdDelayControl_gpio_init(void)
{
    GPIO_InitTypeDef port;

    // enable gpio clocks
    RCC_AHBPeriphClockCmd(RCC_AHBPeriph_GPIOC, ENABLE);

    // configure PC1 as ADC_IN11 input
    GPIO_StructInit(&port);
    port.GPIO_Pin = GPIO_Pin_5;
    port.GPIO_OType =  GPIO_OType_PP;
    port.GPIO_Mode = GPIO_Mode_OUT;
    port.GPIO_Speed = GPIO_Speed_Level_1 ;           // 2 MHz Idem Led Speed

      GPIO_Init(GPIOC, &port);
}


void lcdDelayControl_en(int LEB) // PB8
{

      if (LEB == 1) {
            GPIO_SetBits(GPIOC, GPIO_Pin_5);
      }
      else if (LEB == 0) {
            GPIO_ResetBits(GPIOC,GPIO_Pin_5);
      }
}


void lcd_delay(int cycles)       // uS: Indica la cantidad de uSegundos que desea
esperar.
{
      int i=1;
      while (i <= cycles) {
            i++;
      }
}


void lcd_en(int LEB)       // PB8
{

      if (LEB == 1) {
            GPIO_SetBits(GPIOB, GPIO_Pin_8);
      }
      else if (LEB == 0) {
            GPIO_ResetBits(GPIOB,GPIO_Pin_8);
      }
}


void lcd_rw(int RWB)       // PB9
{
      if (RWB == 1) {
            GPIO_SetBits(GPIOB, GPIO_Pin_9);
      }
```

```
	else if (RWB == 0) {
		GPIO_ResetBits(GPIOB,GPIO_Pin_9);
	}
}


void lcd_rs(int RSD)	// PD2
{
	if (RSD == 1) {
		GPIO_SetBits(GPIOD, GPIO_Pin_2);
	}
	else if (RSD == 0){
		GPIO_ResetBits(GPIOD,GPIO_Pin_2);
	}
}


void lcd_strobe(void)		//// aviso que la info esta en el bus de datos
{
// mantengo el estado BAJO por 5 uSeg
//	GPIO_ResetBits(GPIOB, GPIO_Pin_8);
    lcd_delay(500);

// mantengo el estado ALTO por 5 uSeg
	GPIO_SetBits(GPIOB, GPIO_Pin_8);
	lcd_delay(500);

// vuelvo al estado BAJO y lo mantengo po 5 uSeg.
	GPIO_ResetBits(GPIOB,GPIO_Pin_8);
	lcd_delay(500);
}


void lcd_data(char control)
{
	// me quedo con los 4 bits menos significativos
	uint8_t dataLed = (0x0F & control);
	GPIO_Write(GPIOB, dataLed << 4);
}


void lcd_write(char DATA)  // Escribe los datos en el GPIOB. (primero escribe las 4-
MSB y luego los 4-LSB)
{
	lcd_delay(4000);                                                    //
delay de 40 uSeg aprox
	lcd_data((0xF0 & DATA) >> 4);
	lcd_strobe();
	lcd_data((0x0F & DATA) >> 0);
	lcd_strobe();
}


void lcd_clear(void)
{
	lcd_rs(0);
	lcd_write(0x01u);		//
	lcd_delay(2000);		// Delay de 2 ms
```

```
}


void lcd_init()                         // inicializacion del LCD
{
       lcd_en(0);
       lcd_rs(0);
       lcd_rw(0);
       lcd_delay(15000);          // Delay minimo de 15 mSeg // 15000 ciclos * 50e-6
= 15e-3
       lcd_delay(15000);          // Delay minimo de 15 mSeg // 15000 ciclos * 50e-6
= 15e-3

       lcd_data(0x03u);                 // Cargo el GPIOB  con 0 0 0011 (rs rw d7
d6 d5 d4)
       lcd_strobe();              // aviso que la info esta en el bus de datos

       lcd_delay(5000);           //  Delay  minimo  de  4.1  mSeg  (a  una  frec  de
ClockCycle de 48e6)
       lcd_delay(5000);           //  Delay  minimo  de  4.1  mSeg  (a  una  frec  de
ClockCycle de 48e6)
       lcd_strobe();              // aviso que la info esta en el bus de datos

       lcd_delay(500);                  // Delay minimo de 100µSeg (a una frec de
ClockCycle de 48e6)
       lcd_delay(500);                  // Delay minimo de 100µSeg (a una frec de
ClockCycle de 48e6)
       lcd_strobe();              // aviso que la info esta en el bus de datos

       lcd_delay(500);                  // Delay minimo de 100µSeg (a una frec de
ClockCycle de 48e6)
       lcd_delay(500);                  // Delay minimo de 100µSeg (a una frec de
ClockCycle de 48e6)
       lcd_strobe();              // aviso que la info esta en el bus de datos

       lcd_data(0x02u);                 // Cargo el GPIOB  con 0 0 0011 (rs rw d7
d6 d5 d4)
       lcd_strobe();              // aviso que la info esta en el bus de datos

    lcd_write(0x28u);             //
       lcd_write(0x0Fu);          //     Display On, Cursor On, cursor Blink
       lcd_clear();               //     Limpia la pantalla
       lcd_write(0x06u);          //     Queda en modo entrada */

}


void lcd_goto(uint8_t pos) // Especifica la posicion del display del LCD sobre la
cual quiero escribir
{
    lcd_rs(0u);
    lcd_write(0x80 + pos); //
}


void lcd_writeln(char* buff, uint8_t size, uint8_t line) // Escribe la palabra
"buff" en el LCD (4 bit mode)
{
```

```
    uint8_t i;

    // Cambia segun la linea sobre la cual quiere escribirse, implementado para dos
lineas.
    if (line == 1u) {
        // selecciona la primera linea para escribir
       lcd_goto(0x00u);    // especifica la posicion en hexadecimal (sin signo)
sobre la cual quiero escribir
    }
    else if (line == 2u) {
        // selecciona la segunda linea para escribir
        lcd_goto(0x40u);  // especifica la posicion hexadecimal sobre la cual
quiero escribir.
    }
    else {
        // default line
        lcd_goto(0x00u);
    }

    // write characters operation
    lcd_rs(1u);
    // send buffer contents to lcd
    for (i = 0; i < size; i++) {
        lcd_write(buff[i]);
    }
}
void calibf (void){
       /* muestra en pantalla */
       char lineas[2][17] = {
              "  Calibracion   ",
              "                "
       };
       lcd_writeln(lineas[0], strlen(lineas[0]), 1);
       lcd_writeln(lineas[1], strlen(lineas[1]), 2);

       estado=CALIB;
       evento=      NONE;

}
void medicf(void){
       /* muestra en pantalla */
       char lineas[2][17] = {
              "    Medicion    ",
              "                "
       };
       lcd_writeln(lineas[0], strlen(lineas[0]), 1);
       lcd_writeln(lineas[1], strlen(lineas[1]), 2);

       estado=MEDIC;
       evento=NONE;

}

void rango1f(void){
       /* muestra en pantalla */
       char lineas[2][17] = {
              " Calibracion >  ",
              "    RANGO 1     "
```

```
	};
	lcd_writeln(lineas[0], strlen(lineas[0]), 1);
	lcd_writeln(lineas[1], strlen(lineas[1]), 2);

	estado=RANGO1;
	evento=	NONE;

}

void rango2f(void){
	/* muestra en pantalla */
	char lineas[2][17] = {
		" Calibracion >  ",
		"    RANGO 2     "
	};
	lcd_writeln(lineas[0], strlen(lineas[0]), 1);
	lcd_writeln(lineas[1], strlen(lineas[1]), 2);

	estado=RANGO2;
	evento=	NONE;

}


void rango3f(void){
	/* muestra en pantalla */
	char lineas[2][17] = {
		" Calibracion >  ",
		"    RANGO 3     "
	};
	lcd_writeln(lineas[0], strlen(lineas[0]), 1);
	lcd_writeln(lineas[1], strlen(lineas[1]), 2);

	estado=RANGO3;
	evento=	NONE;

}

void conctarf(void){
	/* muestra en pantalla */
	char lineas[2][17] = {
		"   Conectar     ",
		"  Electrodos >  "
	};
	lcd_writeln(lineas[0], strlen(lineas[0]), 1);
	lcd_writeln(lineas[1], strlen(lineas[1]), 2);

	estado=CONEX_ELECTR;
	evento=	NONE;
}

void calibR1f(void) {
	/* muestra en pantalla */
	char lineas[2][17] = {
		"   Calibrando   ",
		"     Rango 1    "
	};
	lcd_writeln(lineas[0], strlen(lineas[0]), 1);
```

```
	lcd_writeln(lineas[1], strlen(lineas[1]), 2);

	//Configuracion de Rele para RANGO 1 (3K)
	rele1_en(1);
	//Especifico la corriente RMS para el Rango 1 (3K)
	Irms=IRMS_R1;

	lcd_delay(3000000);
	estado=CALIB;
	evento= UP;
}

void calibR2f(void) {
	/* muestra en pantalla */
	char lineas[2][17] = {
		"   Calibrando   ",
		"     Rango 2    "
	};
	lcd_writeln(lineas[0], strlen(lineas[0]), 1);
	lcd_writeln(lineas[1], strlen(lineas[1]), 2);

	//Cconfiguracion de los Reles en RANGO 2 (30K)
	rele1_en(0);
	rele2_en(1);

	//Especifico la corriente RMS para el Rango 2 (30K)
	Irms=IRMS_R2;

	lcd_delay(3000000);
	estado=CALIB;
	evento= UP;
	//evento=    NONE;
}

void calibR3f(void) {
	/* muestra en pantalla */
	char lineas[2][17] = {
		"   Calibrando   ",
		"     Rango 3    "
	};
	lcd_writeln(lineas[0], strlen(lineas[0]), 1);
	lcd_writeln(lineas[1], strlen(lineas[1]), 2);

	//Cconfiguracion de los Reles en RANGO 3 (300K)
	rele1_en(0);
	rele2_en(0);
	//Especifico la corriente RMS para el Rango 3 (300K)
	Irms=IRMS_R3;

	lcd_delay(3000000);
	estado= CALIB;
	evento= UP;
	//evento=    NONE;
}

void UsartTx(void){

	unsigned char bytes[4];
```

```
	//Si Vrms= AABBCCDD
	//TxValue=Vrms;
	bytes[0]=(TxValue>>24)&0xFF;		//bytes=[0]=AA
	bytes[1]=(TxValue>>16)&0xFF;		//bytes=[1]=BB
	bytes[2]=(TxValue>>8)&0xFF;		//bytes=[2]=CC
	bytes[3]=TxValue&0xFF;			//bytes=[3]=DD

	USART_SendData(USART2, 0x3c);
	USART_SendData(USART2, 0x00);
	USART_SendData(USART2, 0x01);
	USART_SendData(USART2, 0x02);

	USART_SendData(USART2, 0x03);
	USART_SendData(USART2, 0x04);
	USART_SendData(USART2, 0x05);
	USART_SendData(USART2, 0x06);
}

void algoritmof (void){

	contador++;
	if (contador ==300000) {
	// Mostrar datos procesados
	// transmitir datos por puerto serie
	/* muestra en pantalla */
	char lineas[2][18] = {
			"  Conductancia  ",
			"                "
		};
	//char lineas[2][18] = {
	//    " Conductividad: ",
	//    ".               "
	//};
	fixq_t Conductivity= FPMUL(FPDIV(Irms,Vrms,Q),K,Q);
	// Calculo la conductividad en PFijo,
	fixq_t Cond = TOFIX(Conductivity,Q);
	sprintf(lineas[1], "    %2.3f\xe4S     ", TOFLT_F(Conductivity, Q));
	lcd_writeln(lineas[0], strlen(lineas[0]), 1);
	lcd_writeln(lineas[1], strlen(lineas[1]), 2);
	TxValue=Conductivity;
	UsartTx();
	contador=0;
	}


	estado=TX_PC;
	evento=NONE;
}

int main(void)
{

	SystemInit ();
	SystemCoreClockUpdate (); //
	SysTick_Config (SystemCoreClock / MillisecondsIT);
	NVIC_EnableIRQ(SysTick_IRQn);

	lcd_gpio_init();
```

```
lcdDelayControl_gpio_init();
lcd_init();

// user code
dbg_gpio_init();
dbg_on();
dbg_off();

adc_gpio_init();
adc_init();

dac_gpio_init();
dac_init();

tim_gpio_init();
tim_init();

usart_gpio_init();
usart_init();

rele_gpio_init();

sw_gpio_init();

/* muestra en pantalla */
char lineas[2][17] = {
"  *** MENU ***  ",
"                "
};
lcd_writeln(lineas[0], strlen(lineas[0]), 1);
lcd_writeln(lineas[1], strlen(lineas[1]), 2);

while (1) {
	switch (estado)
	{
		case(MENU):
					switch (evento)
					{
						case (OK):
									calibf();
									break;
						case (DOWN):
									calibf();
									break;
						case (UP):
									medicf();
									break;
						default:
									break;
					}
					break;

		case(CALIB):
					switch (evento)
					{
						case (OK):
									rango1f();
									break;
```

```
                        case (RIGHT):
                                        rango1f();
                                        break;
                        case (DOWN):
                                        medicf();
                                        break;
                        case (UP):
                                        medicf();
                                        break;
                        default:
                                        break;
                }
                break;

case(MEDIC):
                switch (evento)
                {
                        case (OK):
                                        conctarf();
                                        break;
                        case (RIGHT):
                                        conctarf();
                                        break;
                        case (DOWN):
                                        calibf();
                                        break;
                        case (UP):
                                        calibf();
                                        break;
                        default:
                                        break;
                }
                break;

case(RANGO1):
                switch (evento)
                {
                        case (OK):
                                        calibR1f();
                                        break;
                        case (RIGHT):
                                        calibR1f();
                                        break;
                        case (LEFT):
                                        calibf();
                                        break;
                        case (DOWN):
                                        rango2f();
                                        break;
                        case (UP):
                                        rango3f();
                                        break;
                        default:
                                        break;
                }
                break;

case(RANGO2):
```

```
                        switch (evento)
                        {
                               case (OK):
                                                      calibR2f();
                                                      break;
                               case (RIGHT):
                                                      calibR2f();
                                                      break;
                               case (LEFT):
                                                      calibf();
                                                      break;
                               case (DOWN):
                                                      rango3f();
                                                      break;
                               case (UP):
                                                      rango1f();
                                                      break;
                               default:
                                                      break;
                        }
                        break;

case(RANGO3):
                        switch (evento)
                        {
                               case (OK):
                                                      calibR3f();
                                                      break;
                               case (RIGHT):
                                                      calibR3f();
                                                      break;
                               case (LEFT):
                                                      calibf();
                                                      break;
                               case (DOWN):
                                                      rango1f();
                                                      break;
                               case (UP):
                                                      rango2f();
                                                      break;
                               default:
                                                      break;
                        }
                        break;

        case(CONEX_ELECTR):

                        switch (evento)
                        {
                               case (OK):
                                                      algoritmof();
                                                      break;
                               case (RIGHT):
                                                      algoritmof();
                                                      break;
                               case (LEFT):
                                                      medicf();
                                                      break;
```

```
                                        default:
                                                                break;
                                }
                                break;

                        case(TX_PC):
                                algoritmof();
                                switch (evento)
                                {
                                        case (OK):
                                                                calibf();
                                                                break;
                                        case (RIGHT):
                                                                calibf();
                                                                break;
                                        default:
                                                                break;
                                }
                                break;
                        default:
                                break;
                }
        }
        mostrar = false;

        while (1){
                if (mostrar) {


                        sprintf(lineas[1], "    %1.3f", TOFLT_F(Vrms, 20));
                        lcd_writeln(lineas[0], strlen(lineas[0]), 1);
                        lcd_writeln(lineas[1], strlen(lineas[1]), 2);

                        mostrar = false;
                }


        }

        for (;;) {
                if (USART_GetFlagStatus(USART2, USART_FLAG_RXNE) != RESET) {
                        /* If received 't', toggle LED and transmit 'T' */
                        char recv = USART_ReceiveData(USART2);
                        // Testing USART1
                        USART_SendData(USART2, recv);// Envio un dato entre 0 y 65536
por la USART2
                        while (USART_GetFlagStatus(USART2, USART_FLAG_TXE) == RESET)
                                ;

                        lcd_delay(1000);          // Delay minimo de 15 mSeg
                        GPIO_Write(GPIOB, 0x0118);
                        lcd_delay(4200);          // Delay minimo de 4,2 mSeg
                        GPIO_Write(GPIOB, 0x0118);
                        lcd_delay(120);                  // Delay minimo de 100 uSeg
                        GPIO_Write(GPIOB, 0x0118);
                }
        }
```

```
	return 0;
}


void EXTI0_1_IRQHandler(void)
{
		if (EXTI_GetITStatus(EXTI_Line0) != RESET) {

			NVIC_DisableIRQ(EXTI0_1_IRQn);

			lcd_delay( 300000 );

			if (EXTI_GetITStatus(EXTI_Line0) != RESET) {


				evento= OK;
				NVIC_EnableIRQ(EXTI0_1_IRQn);
			}
			else if (EXTI_GetITStatus(EXTI_Line0) == RESET) {
				NVIC_EnableIRQ(EXTI0_1_IRQn);
			}

			EXTI_ClearITPendingBit(EXTI_Line0);
	}


	else if (EXTI_GetITStatus(EXTI_Line1) != RESET){

			NVIC_DisableIRQ(EXTI0_1_IRQn);

			lcd_delay( 300000 );


			if (EXTI_GetITStatus(EXTI_Line1) != RESET) {


				evento= RIGHT;
				NVIC_EnableIRQ(EXTI0_1_IRQn);
			}
			else if (EXTI_GetITStatus(EXTI_Line1) == RESET) {
				NVIC_EnableIRQ(EXTI0_1_IRQn);
			}

			EXTI_ClearITPendingBit(EXTI_Line1);

	}


}


void EXTI2_3_IRQHandler (void)
{
	// Me aseguro que la interrupcion fue por la línea 2
		if (EXTI_GetITStatus(EXTI_Line2) != RESET) {


			NVIC_DisableIRQ(EXTI2_3_IRQn);
```

```
                lcd_delay( 300000 );

                if (EXTI_GetITStatus(EXTI_Line2) != RESET) {


                        evento= DOWN;
                        NVIC_EnableIRQ(EXTI2_3_IRQn);
                }
                else if (EXTI_GetITStatus(EXTI_Line2) == RESET) {
                        NVIC_EnableIRQ(EXTI2_3_IRQn);
                }

                EXTI_ClearITPendingBit(EXTI_Line2);

                // ===================================
                // Codigo de usuario para la linea 2
                // ===================================
                evento = DOWN;
                lcd_delay( 50000 );
                EXTI_ClearITPendingBit(EXTI_Line2);
        }
}


void EXTI4_15_IRQHandler (void)
{

        if (EXTI_GetITStatus(EXTI_Line4) != RESET) {


                NVIC_DisableIRQ(EXTI4_15_IRQn);

                lcd_delay( 300000 );

                if (EXTI_GetITStatus(EXTI_Line4) != RESET) {


                        evento= LEFT;
                        NVIC_EnableIRQ(EXTI4_15_IRQn);
                }
                else if (EXTI_GetITStatus(EXTI_Line4) == RESET) {
                        NVIC_EnableIRQ(EXTI4_15_IRQn);
                }

                EXTI_ClearITPendingBit(EXTI_Line4);

        }

        else if (EXTI_GetITStatus(EXTI_Line5) != RESET){


                NVIC_DisableIRQ(EXTI4_15_IRQn);

                lcd_delay( 300000 );
```

```
                    if (EXTI_GetITStatus(EXTI_Line5) != RESET) {

                           evento= UP;
                           NVIC_EnableIRQ(EXTI4_15_IRQn);
                    }
                    else if (EXTI_GetITStatus(EXTI_Line5) == RESET) {
                           NVIC_EnableIRQ(EXTI4_15_IRQn);
                    }

                    EXTI_ClearITPendingBit(EXTI_Line5);
             }

}


void ADC1_COMP_IRQHandler(void)
{
       // int32_t value = 0;

    if (ADC_GetITStatus(ADC1, ADC_IT_EOC) != RESET) {

       //Inicio de conversion ADC
        dbg_on();
       // Clear ADC1 interrupt flag
        ADC_ClearITPendingBit(ADC1, ADC_IT_EOC);
        // get new ADC sample (Q0)
        adc_value = (int32_t) ADC_GetConversionValue(ADC1) - calib;
        dbg_off();
        //fin

        //---------------- INICIO PROCESAMIENTO ------------------------
        dbg_on();

        // filtro pasabajos

        v_inf[0] = FMULG(ADC_GAIN, adc_value, Q, 0, Q);

        // filtro pasaaltos
        v_in[0] = (v_inf[0] - v_inf[1]) + FPMUL(CTE1_HPF, v_in[1], Q);

        // actualizacion de las variables de estados
        v_inf[1] = v_inf[0];
        v_in[1] = v_in[0];

        //  --- ALGORITMO DE PROCESAMIENTO  ---
        X[init] = abs(v_in[0]);

        U = X[init] - X[fin];

        // Vrms DE LA TENSION MEDIDA  (Integracion de la Diferencia (U))
        Y[0]= U + Y[1];

        //Actualizo el vector de salida (Resultado de la integracion en tiempo
discreto)
        Y[1] = Y[0];

       // Filtro pasa bajo de 2 hz a la salida
        v_fin[0] = FPMUL(Y[0], CTE2B_LPF, Q) + FPMUL(v_fin[1], CTE1B_LPF, Q);
```

```
        // actualizacion de las variables de estados
        v_fin[1] = v_fin[0];

        // Valor de medicion a mostrar Vrms
        Vrms = FPMUL(v_fin[0], RMS_CONST, Q);

        //desplazamiento circular del puntero de la posicion inicial del vector
        if (++init >= N) {
           init = 0;
        }

        //desplazamiento circular del puntero de la posicion final del vector
        if (++fin >= N) {
           fin = 0;
        }
        //   --- FIN ALGORITMO DE PROCESAMIENTO ---
        dbg_off();

        //--------------------INICIO CONVERSION DAC -------------------------
        dbg_on();
        // get new sinusoid value from look up table
        dac_value = FPMUL(amp, sin_lookup(theta), 30);

        // convert from Q30 to a 12-bit value (desplazo a la derecha 18 posiciones,
para Tx los 12 LSB)
        dac_value >>= 18;

        // Show digital sinusoid in DAC1
        DAC_SetChannel1Data(DAC_Align_12b_R, 2048 + dac_value);

        // increase sine angle, wrap angle if greater than 2*pi
        theta += DELTA_THETA;
        if (theta > (LUT_LEN - DELTA_THETA)) {
            theta = 0;
        }

        // Muestro valor Vrms en el DAC2
        DAC_SetChannel2Data(DAC_Align_12b_R, 100 + abs(FMULG(DAC_GAIN << 1, Vrms,
Q, Q, 0)));
        dbg_off();
            }
}
void SysTick_Handler (void) {
       static int32_t counter = 0;
       if(counter++ >= 10) {
             mostrar= true;
             counter = 0;
       }
}
```

Volver al inicio del Apéndice D

Bibliografía

Chu, G. (2018). *Signal and power isolation with 3 . 3 V / 5 V input and 12 V / 15 V output*. (January), 1-13.

Ferroxcube. (2014). *3C95/3C97 Datasheet*. Recuperado de https://5740daf986400cba1fd6-438a983e20136d2f376705dfe1c68aea.ssl.cf3.rackcdn.com/Acal-BFi-Ferroxcube-3C95_3C97-brochure.pdf

Http://www.analytical-chemistry.uoc.gr/files/items/6/618/agwgimometria_2.pdf. (2004). Conductivity theory and practice. Recuperado de accessed 06/11/2017 website: http://www.analytical-chemistry.uoc.gr/files/items/6/618/agwgimometria_2.pdf

Information, D. (2014). *SN6501 Transformer Driver for Isolated Power Supplies*. (1).

Jaimes Morales, S. A., & Valencia Quintero, J. P. (2006). *Diseño de un Medidor de Conductividad Electrica de Soluciones Salinas, Haciendo Uso de una Celda de Conductancia Calculable, para la Calibración de Una Sonda Tetrapolar Utilizada en la Medición de Impedancia Electrica en Tejido Humano*.

Jaimes, S. A., Valencia, J. P., & Miranda, D. A. (2008a). Diseño de medidor de conductividad electrolítica, para operar en el rango de conductividad eléctrica del tejido humano. En *IFMBE Proceedings* (Vol. 18, pp. 456-459). https://doi.org/10.1007/978-3-540-74471-9_106

Jaimes, S. A., Valencia, J. P., & Miranda, D. A. (2008b). Diseño de un medidor de conductividad electrolítica, para operar en el rango de conductividad eléctrica del tejido humano. *IFMBE Proceedings*, *18*, 456-459. https://doi.org/10.1007/978-3-540-74471-9_106

Mohan, N., Undeland, T. M., & Robbins, W. P. (s. f.). *Power electronics*.

Moroń, Z. (2003). Investigations of van der Pauw method applied for measuring electrical conductivity of electrolyte solutions: Measurement of electrolytic conductivity. *Measurement: Journal of the International Measurement Confederation*. https://doi.org/10.1016/S0263-2241(02)00079-9

Ortiz, O. A. A., Vergara, O. J. B., & Mercado, D. A. M. (2007). Criterios de Diseño de la Fuente de Corriente de Holwand. *Revista Uis Ingenierías*, *6*(1). Recuperado de http://revistas.uis.edu.co/index.php/revistauisingenierias/article/view/1950

Schiefelbein, S. L., Fried, N. A., Rhoads, K. G., & Sadoway, D. R. (1998). A high-accuracy, calibration-free technique for measuring the electrical conductivity of liquids. *Review of Scientific Instruments*. https://doi.org/10.1063/1.1149095

Webster, J. G. (s. f.). *The Measurement, Instrumentation and Sensors HANDBOOK*.

Zhang, B., Lin, Z., Zhang, X., Yu, X., Wei, J., & Wang, X. (2014a). System for absolute measurement of electrolytic conductivity in aqueous solutions based on van der Pauw's theory. *Measurement Science and Technology*, *25*(5). https://doi.org/10.1088/0957-0233/25/5/055005

Zhang, B., Lin, Z., Zhang, X., Yu, X., Wei, J., & Wang, X. (2014b). System for absolute measurement of electrolytic conductivity in aqueous solutions based on van der Pauw's theory. *Measurement Science and Technology*, *25*(5). https://doi.org/10.1088/0957-0233/25/5/055005