UNIVERSIDAD CATOLICA DE SANTA MARIA

FACULTAD DE CIENCIAS E INGENIERIAS FISICAS Y FORMALES

PROGRAMA PROFESIONAL DE INGENIERIA ELECTRONICA

"DISEÑO Y APLICACIÓN DE UN ANALIZADOR DE CALIDAD DE ENERGIA ELECTRICA"

TESIS PARA OBTENER EL GRADO PROFESIONAL DE:

INGENIERO ELECTRONICO

AUTOR:

VALENCIA ALEJO, MARTIN

PERU - 2016

PRESIDENTE

SECRETARIO

DECLARACION JURADA

Yo, MARTIN VALENCIA ALEJO, estudiante del Programa Profesional de Ingeniería Electrónica de la Universidad Católica de Santa María, identificado con DNI 71659861 con la tesis titulada "DISEÑO Y APLICACIÓN DE UN ANALIZADOR DE CALIDAD DE ENERGIA ELECTRICA".

Declaro bajo juramento que:

1. La tesis es de mi autoría.
2. La tesis no ha sido autoplagiada; es decir, no ha sido publicada ni presentada anteriormente para obtener algún grado académico previo o título profesional.
3. Los datos presentados en los resultados son reales no han sido falseados, ni duplicados, ni copiados y por tanto los resultados que se presenten en la tesis se constituirán en aportes a la realidad investigada.

De idenficarse la falta de fraude (datos falsos), plagio (información sin citar autores), autoplagio (presentar como nuevo algún trabajo de investigación propio que ya ha sido publicado), o falsificación (representar falsamente las ideas de otros), asumo las consecuencias y sanciones que de mi acción se deriven, sometiéndome a la normatividad vigente de la Universidad Católica de Santa María.

Arequipa, Viernes 22 de Abril del 2016

Martin Valencia Alejo

71659861

RESUMEN

En el presente trabajo presento el diseño y aplicación de un analizador de calidad de energía eléctrica monofásico con una medición máxima de voltaje de 600V y 100 A para medición de Corriente.

Este diseño tiene un mayor número de mediciones a la señal de voltaje, corriente y se aplican algoritmos de Procesamiento digital de Señales con un microcontrolador de tecnología CORTEX- ARM de alta gama de la familia Texas Instruments.

El microcontrolador vizualiza la señal AC en una pantalla táctil de 7 pulgadas y procesa las señales con algoritmos matemáticos como Fourier, Convolucion en frecuencia, métodos estadísticos como promedio, máximos y mínimos de un vector de datos.

El equipo puede desarrollar 6 tipos diferentes de tareas como:

- Gráficos XY de la forma de Onda del voltaje y corriente.
- Calculo de las potencias alternas y gráficos XY de frecuencia vs tiempo.
- Visualizaciones de armónicos de voltaje, corriente y potencia hasta el #59.
- Captura de Transitorios en el Voltaje, hasta 20 eventos.
- Captura de Fluctuaciones y variaciones de tensión, corriente, hasta 7 Horas.

El equipo también cuenta con una memoria interna donde almacena eventos de cada una de las tareas independientes, en formato Excel; Permite al usuario comunicar el equipo por medio de Bluetooth para recibir los datos de la medición por medio de un dispositivo móvil; El sistema es compacto e integra un número reducido de componentes. El circuito fue diseñado utilizando el software EAGLE Cadsoft.

ABSTRACT

In this tesis, present the design and implementation of a power quality analyzer single phase with a measurement maximum voltage 600V and for current measurement 100A.

This design has a greater number of measurements to the signal voltage, current and are applied algorithms of Digital Signal Processing with an ARM-CORTEX, microcontroller high-end technology from Texas Instruments family.

The microcontroller displays the AC signal into a 7-inch touch screen and processes the signals with mathematical algorithms as Fourier, frequency convolution, statistical methods such as average, maximum and minimum of a vector data.

The team can do 6 different types of tasks such as:

• XY graphs waveform voltage and current.

• Calculation of AC powers and XY graphs frequency versus time.

• Views harmonic voltage, current and power to # 59.

• Capture Transient Voltage up to 20 events.

• Capture Fluctuations and variations of voltage, current, up to 7 hours.

The system also has an internal memory which stores events of each of the separate tasks in Excel format; It allows the user to communicate by Bluetooth to receive measurement data by a mobile device; The system is compact and integrates a small number of components. The circuit was designed using the Cadsoft EAGLE software.

DEDICATORIA

A mis padres y hermana, por su gran apoyo y cariño que me dieron en los momentos más difíciles.

¡Muchas gracias, esto se logró gracias a ustedes!

¡Yo nunca deje de soñar!

Martin V.

AGRADECIMIENTO

Le agradezco a Dios por haberme ayudado con las ideas más difíciles, a mi asesor de Tesis Ing. Eduardo Esquivel Zenteno por su gran apoyo y consejos.

Estoy muy agradecido con Anthony F. Seely y Blake Ethridge, ambos ingenieros de la empresa Texas Instruments que me brindaron su apoyo y el soporte para aprender a utilizar los microcontrolador Hercules RMx.

A mi amigo Jan Cumps (Belgica), quien me enseño a implementar las librerías DSP ARM.

A mi Amigo Miguel Bellido, por su ayuda con los detalles finales.

Muchas gracias por su apoyo en la elaboración de este proyecto.

CONTENIDO

FIGURAS

DIAGRAMAS DE BLOQUES

ESQUEMAS CIRCUITALES

FLUJOGRAMAS

TABLAS

NOMENGLATURA

A	Ampere
CA	Corriente Alterna
CI	Circuito Integrado
dB	Decibelios
DSP	Procesamiento Digital de Señales
Hz	Hertz
PWM	Modulacion por Ancho de Pulso
Rms	Raiz Cuadrada media
RAM	Read Access Memory
P	Potencia Activa
S	Potencia Aparente
Q	Potencia Reactiva
Vrms	Voltaje RMS
Irms	Corriente RMS
Vcc	Fuente fija de voltaje
KW	Kilo Watt
KVAR	Kilo Var
PF	Factor de Potencia
THD	Distorsion Armonica total
GB	Giga Byte
TFT	Pantalla a colores LCD
L	Inductancia
C	Capacitancia
J	Joule, unidad de energía
“	Simbolo de Pulgadas
Imax	Corriente Maxima soportada

Emax	Maxima energía soportada
Veff	Tension de servicio
Bw	Ancho de Banda
LSB	Bit menos significativo
AND	Operación lógica binaria AND
XOR	Operación lógica binaria XOR
TS	Periodo de Muestreo
Δf	Resolucion de frecuencia en el espectro de Fourier
T	Longitud total del registro de muestreo
N	Número de muestras en una señal análoga

CAPITULO 1

INTRODUCCION

En este capítulo se presenta la identificación del problema, descripción del problema, justificación, objetivos, aportaciones y la estructura del proyecto de tesis; Se expone los elementos utilizados para analizar el problema, como también los métodos aplicados, se indica los aportes de esa tesis y la estructura del trabajo.

I. IDENTIFICACION DEL PROBLEMA

Ruidos e impulsos

Son perturbaciones de tensión las que tienen lugar entre los conductores; por lo general son frecuentes y diferente voltaje.

Si son transitorios y de valor elevado (cientos de voltios), se denominan impulsos, con una duración inferior a 2 ms.

De todas las perturbaciones, son las más aleatorias y menos predecibles. Una forma de onda de tensión con ruido eléctrico se muestra en la Figura 1(a), mientras que en la figura 1(b), se muestra una forma de onda de tensión con la presencia de un impulso eléctrico.

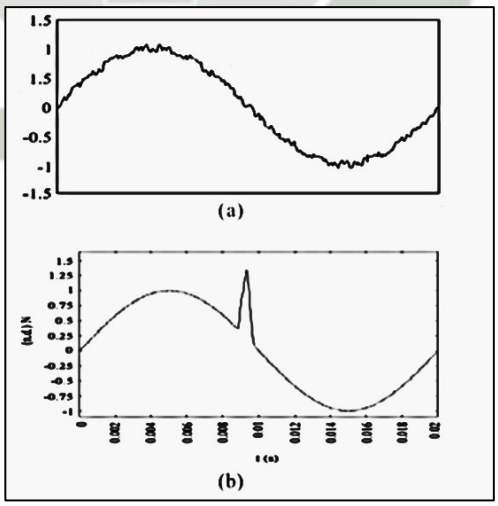


Figura 1.1 (a) Ruido eléctrico presente en la señal senoidal proporcionada por la red eléctrica, (b) Impulso eléctrico.

Variaciones lentas y rápidas de tensión (Sags y Swells)

Se considera una variación lenta de tensión, aquella que se presenta con una duración de 10 segundos o más; Este tipo de variación de tensión se muestra en la figura 2.

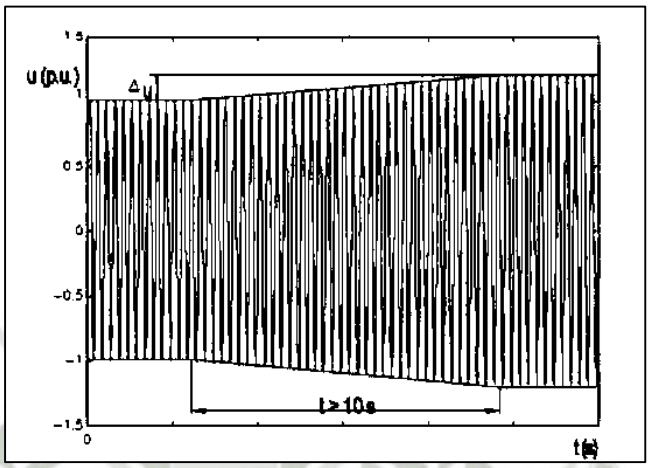


Figura 1.2 Variación lenta de tensión

Por otra parte, una variación rápida de tensión tiene una duración menor a los 10 segundos. Se producen debido a la conexión y desconexión de cargas grandes y maniobras en las líneas de la red eléctrica.

Parpadeo (flicker)

Es una variación rápida de tensión de forma repetitiva, similar a la modulación de amplitud de una onda de alta frecuencia por una onda de baja frecuencia, la cual se puede observar en la figura 3.

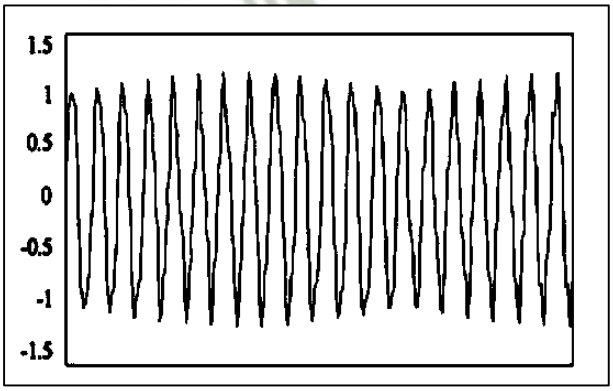


Figura 1.3 Variación rápida de tensión.

Microcortes

Son anulaciones en la tensión de la red eléctrica (o reducciones por debajo del 60% de su valor nominal) con una duración menor a un ciclo.

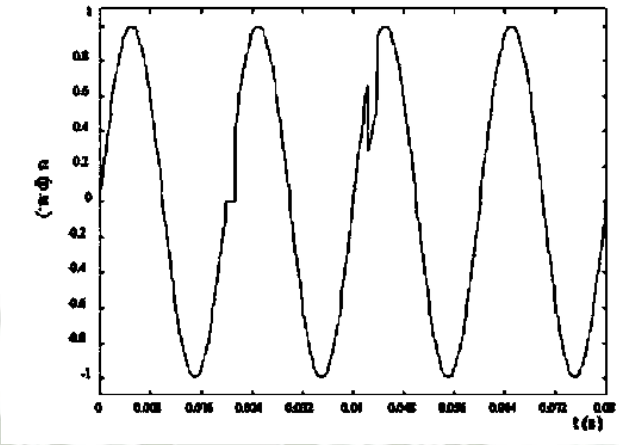


Figura 1.4 Microcortes de tensión.

Cortes largos

Son anulaciones de la tensión de red (o reducciones por debajo del 50% de su valor nominal) de duración mayor a un ciclo.

La figura 5, muestra un corte largo de tensión, que si bien, ésta no cae a cero, si es menor al 50% del valor nominal.

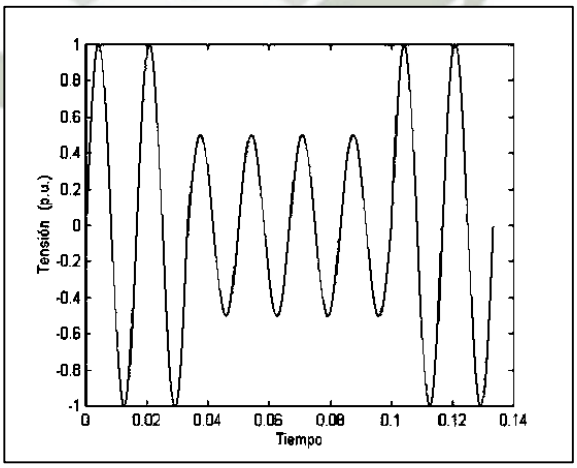


Figura 1.5 Corte largo de tensión.

Distorsión

Es una deformación de la forma de onda de tensión, debida a la presencia de armónicos. Su nombre técnico es Distorsión Armónica Total (THD por sus siglas en inglés). Casi todas las cargas críticas como lo son los equipos electrónicos soportan una distorsión máxima del 5%.

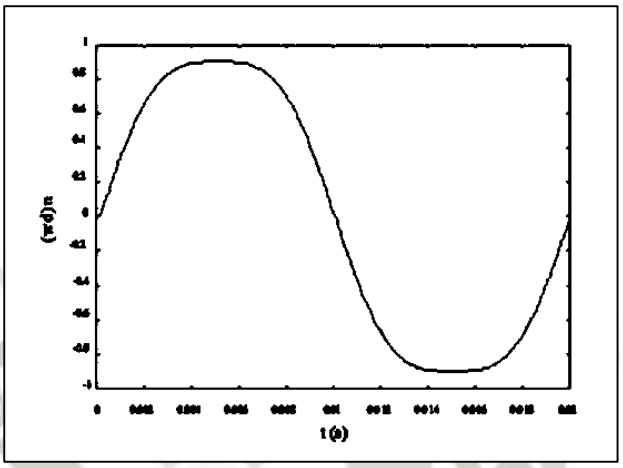


Figura 1.6 Forma de onda de tensión con gran distorsión armónica.

Variaciones de frecuencia

Son cambios en la frecuencia de señal senoidal proporcionada por la red, que en nuestro país es de 60 Hz.

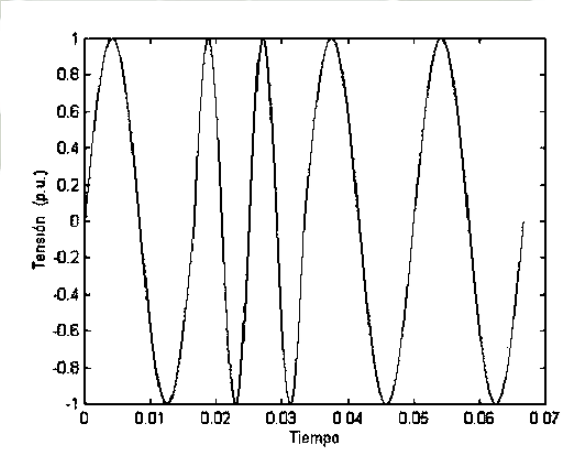


Figura 1.7 Forma de onda de tensión con variación de frecuencia.

II. DESCRIPCION DEL PROBLEMA

Ruidos e impulsos

Los ruidos eléctricos se producen debido al funcionamiento de máquinas eléctricas con escobillas, soldadoras de arco, timbres, interruptores con cargas no lineales.

Por otro lado, los impulsos eléctricos suelen producirse por conexión y desconexión de bancos de condensadores, funcionamiento de hornos de arco, máquinas con escobillas, interruptores, termostatos y por descargas eléctricas.

Variaciones lentas y rápidas de tensión

Se produce debido a la variación de las cargas en redes eléctricas con impedancia alta de cortocircuito.

Parpadeo (flicker)

Produce en las lámparas un parpadeo visible y molesto (de aquí el nombre); se debe principalmente al funcionamiento de hornos de arco y equipos de soldadura.

En general no produce daños en los equipos a menos que la variación sea muy pronunciada.

Microcortes

Se deben principalmente a defectos en la red eléctrica o en la propia instalación del usuario.

Cortes largos

Se producen generalmente por fallas o desconexión de las líneas de alimentación y por averías en los centros de generación y de transformación.

Distorsión

Se debe principalmente a la conexión a la red eléctrica de máquinas con núcleo magnético saturado, convertidores estáticos (rectificadores controlados y no controlados, sistemas de alimentación ininterrumpida, fuentes conmutadas) y otras cargas no lineales.

Variaciones de frecuencia

Normalmente resulta muy raro que se presente este problema en la red eléctrica en condiciones normales y puede llegar a ocurrir debido a la interconexión de los centros de generación de energía eléctrica. Generalmente sólo se producen en centros con generación aislada de tensión como lo pueden ser redes eléctricas rurales aisladas que obtienen energía eléctrica a partir de generadores de combustión interna, paneles fotovoltaicos.

III. JUSTIFICACION

En la actualidad el consumo de energía eléctrica por parte del progreso industrial es elevado y esto implica el uso de nuevas tecnologías, elementos de estado sólido y cargas no lineales, por su naturaleza estos producen una gran cantidad de perturbación en la línea Eléctrica, afectando a equipos electrónicos a su alrededor, creando problemas y elevando el costo del consumo de energía eléctrica.

El concepto de perturbaciones eléctricas y calidad de energía eléctrica es un tema que se encuentra normado por leyes nacionales "*Norma Técnica de Calidad de energía Electrica",* En la norma se establece aspectos, parámetros e indicadores sobre los que se evalúan la calidad del servicio eléctrico.

Las entidades involucradas en la prestación del servicio están obligadas a adquirir e instalar la infraestructura necesaria para la medición y registros de los parámetros de la calidad del producto, calidad del suministro.

Por lo que el objetivo es el diseño de un analizador de calidad de energía con el cual se mida, procese y visualize los parámetros de voltaje, corriente, potencia activa, potencia reactiva, potencia aparente, frecuencia de línea, factor de potencia, factor de cresta y perturbaciones eléctricas y armónicos.

IV. OBEJTIVOS

a. GENERALES

Diseñar e implementar un analizador de calidad de energía con un microcontrolador ARM-CORTEX R5F, el cual llevara a cabo mediciones independientes de voltaje, corriente y frecuencia; con la finalidad de evaluar las variaciones a las tolerancias en los niveles de tensión, frecuencia y perturbaciones en los puntos de entrega que afectan la calidad de energía eléctrica.

b. ESPECIFICOS

- Aplicar la teoría de Procesamiento digital de señales como la transformada rápida de Fourier para el análisis de las perturbaciones eléctricas y teoría de filtros digitales para el análisis del ruido, impulsos y flicker que se generen en la red eléctrica.

- Implementar tiempos de medición de los parámetros a los que se les denomina “Intervalos de Medición”; en un Intervalo de Medición se comprueba que los parámetros se encuentren dentro del rango tolerable, dichos intervalos son mediciones de periodo variable en minutos.

- Se implementará un módulo interfaz para el usuario con una pantalla LCD TFT a colores, donde el usuario podrá tener acceso a los diferentes menús de tareas como, medición de corriente y voltaje, potencias, visualización de forma de onda y parámetros de energía.

V. APORTACIONES

Las aportaciones al desarrollo de la ingeniería nacional brindadas en esta tesis son las siguientes:

- Un equipo prototipo de bajo costo a comparación de los existentes en el mercado.
- Metodología para la aplicación de Procesamiento Digital de Señales.
  - Algoritmo para la detección de Transitorios.
  - Aplicación de la FFT, análisis en magnitud y fase.
  - Analisis en el dominio de frecuencia.
  - Aplicación de Nyquist para el muestreo de las señales.
  - Aplicación de la convolucion discreta.
  - Aplicación de VALOR CUADRATICO MEDIO de gran cantidad de datos.
- Metodología para el uso de pantalla graficas TFT
  - Gráficos XY.
  - Interpolaciones por pixeles
  - Métodos para el control de la TFT LCD.
- Uso e implementación de procesadores CORTEX
- Creación de un sistema operativo aplicado a Análisis de Energía.
- Interfaz humano – equipo.
- Sistema para almacenar y administrar datos en memorias de gran capacidad.

VI. ESTRUCTURA DE LA TESIS

Esta tesis está conformada por 6 capítulos organizados de la siguiente forma:

CAPITULO 1

En este capítulo se da una introducción a los diferentes tipos de problemas que se pueden identificar en una red eléctrica, su descripción, también se presente la justificación, objetivos y aportaciones de la tesis.

CAPITULO 2

En este capítulo se da a conocer sobre lo que es un analizador de calidad de energía y su funcionamiento, también se expone las diferentes características técnicas del equipo.

CAPITULO 3

En este capítulo se presente el esquema circuital, dando a conocer la disposición de los circuitos, especificaciones, funcionamiento, selección y cálculo respectivo para su diseño.

CAPITULO 4

En este capítulo se presenta los diagramas de Flujo del software a implementar. Se profundiza cada etapa del programa con diagramas de Flujo y su respectiva explicación.

CAPITULO 5

En este capítulo se muestra las pruebas realizadas a cada uno de los circuitos que constituyen el analizador de calidad de energía y se presentan los resultados.

CAPITULO 6

En este capítulo se presenta la conclusión del trabajo y recomendaciones para futuros trabajos.

CAPITULO 2

MARCO TEORICO

En este capítulo se da a conocer las caracteristicas del analizador de calidad de energía y su principio de funcionamiento, también se expone las especificaciones técnicas con las que del equipo.

I. PRINCIPIO DE FUNCIONAMIENTO

El analizador de calidad de energía eléctrica, mide la corriente alterna (AC y DC) y la tensión eléctrica (AC y DC), para el ultimo cuenta con un circuito de reducción de voltaje y acondicionamiento de manera la señal de entrada y la señal muestreada por el microcontrolador tenga la máxima fidelidad.

Luego la corriente alterna y continúa por medio de un sensor de efecto Hall, de manera que nos entrega una señal en tensión DC con excelente fidelidad a la señal de corriente eléctrica para luego acondicionarla y ser muestreada por el microcontrolador para su respectivo procesamiento.

El procesamiento de los datos los llevara a cabo el microcontrolador, el cual muestreara las señales de tensión y corriente acondicionadas en la etapa descrita anteriormente, una vez almacenados los datos en su memoria el microcontrolador ejecutara con ellos algoritmos como el cálculo de la transformada de Fourier, cálculo del verdadero valor eficaz, cálculo de Potencias Eléctricas, medición de la frecuencia de línea eléctrica.

Una vez desarrollados los cálculos por el microcontrolador este procederá a mostrar los resultados en la pantalla LCD, almacenar los datos importantes en su memoria o transmitir los datos por medio de su módulo inalambrico.

II. ESPECIFICACIONES DEL EQUIPO

CARACTERISTICAS DEL EQUIPO

- Mide Voltaje hasta 600Vrms
- Mide Corriente hasta 100 Arms
- Visualiza los valores máximo, promedio, factor de cresta de voltaje y corriente
- Calcula y visualiza potencia aparente, potencia activa, potencia reactiva y factor de potencia.
- Mide y registra parámetros de potencia KW, KVAR,VA
- Visualiza distorsión armónico total THD de tensión, corriente y potencia,
- Visualiza valores armónicos individuales hasta el valor 59.
- Registra, captura y visualiza fluctuaciones de corriente y tensión.
- Almacenamiento con formato FAT32, máximo 100GB.
- Max y Min valores rms son calculados por periodo.

ESPECIFICACIONES DEL EQUIPO

Voltaje RMS	: 600V fase a neutro
Corriente	: 100A
Sistema	: Monofásico
Frecuencia	: 50-60Hz.
Mediciones de Potencia	: kW, kVAR.
Mediciones de Energía	: kWh, kVARh, kVAh en Excel.
Medición de factores	: Factor de Potencia PF, THD
Medición y registro transitorio	: Voltaje y Corriente
Medición de Armónicos	: Hasta el #59.
Medición y registro	: THD Distorsión armónica.
Frecuencia de Muestreo	: 460 mediciones por ciclo.
Capacidad de Almacenamiento	: 8GB.
Fuente de Alimentación	: AC 220Vrms /Batería.
Funcionamiento a Batería	: 12600mAh
Comunicación	: Bluetooh
Compatibilidad EMC	: Filtrado integrado, cable apantallado 5mts
Visualizacion	: TFT Touch LCD
Grado de Protección	: Prototipo
Temperatura Operación	: 0-50°C
Humedad Relativa	: 95% sin condensación.

ESPECIFICACIONES ELETRICAS DE PROTECCION

- Voltaje de entrada : 220Vac +/-10%
- Sistema : Monofásico
- Frecuencia de entrada : 40-70Hz
- Protección contra EMI : Filtro LC a la entrada
- Sobrecorrientes : Fusible
- Protección transitorios : Varistor de 130 Joules

ESPECIFICIACIONES DEL SISTEMA DE MEDICION Y PROCESAMIENTO

- Procesador :ARM CORTEX-R5F
- Numero de Bits : 32 & 16 bits
- Velocidad procesador : 547MIPS/ 330MHz
- Conversor ADC : 12 bits / 3.3 v

ESPECIFICACIONES DE MODULO DE VISUALIZACION

El módulo de interfaz para el usuario es una pantalla LCD táctil que se utiliza para:

  - Configuraciones
  - Navegación entre programas
  - Visualizar parámetros
  - Visualizar Alarmas

- Visualizador : Pantalla LCD a colores de 7“
- Resolución : 800 X RGB X 480 Pixel
- Entrada : Panel Táctil.

SISTEMA DE COMUNICACIÓN

- Interfaz : Inalámbrica
- Formato de Salida : ASCII 8N1 (8 bits, No paridad, 1 bit de parada)
- Velocidad de transmisión : 9600
- Comunicación Inalámbrica : Bluetooth.

CAPITULO 3

DISEÑO DE HARDWARE

En este capítulo se describe el proceso de diseño de los diferentes circuitos que conforman cada etapa del Analizador de Calidad de Energía. Se muestra un diagrama de bloques etapas de diseño.

I. DIAGRAMA DE BLOQUES GENERAL

En el siguiente diagrama especificamos cada etapa del circuito la relación que tiene con otros, se puede diferenciar tres etapas, la etapa de control que consta del circuito de medicion de voltaje y corriente, visualización e interfaz, etapa de memoria, etapa de comunicación y la etapa de potencia, como se puede observar en la figura 3.1.

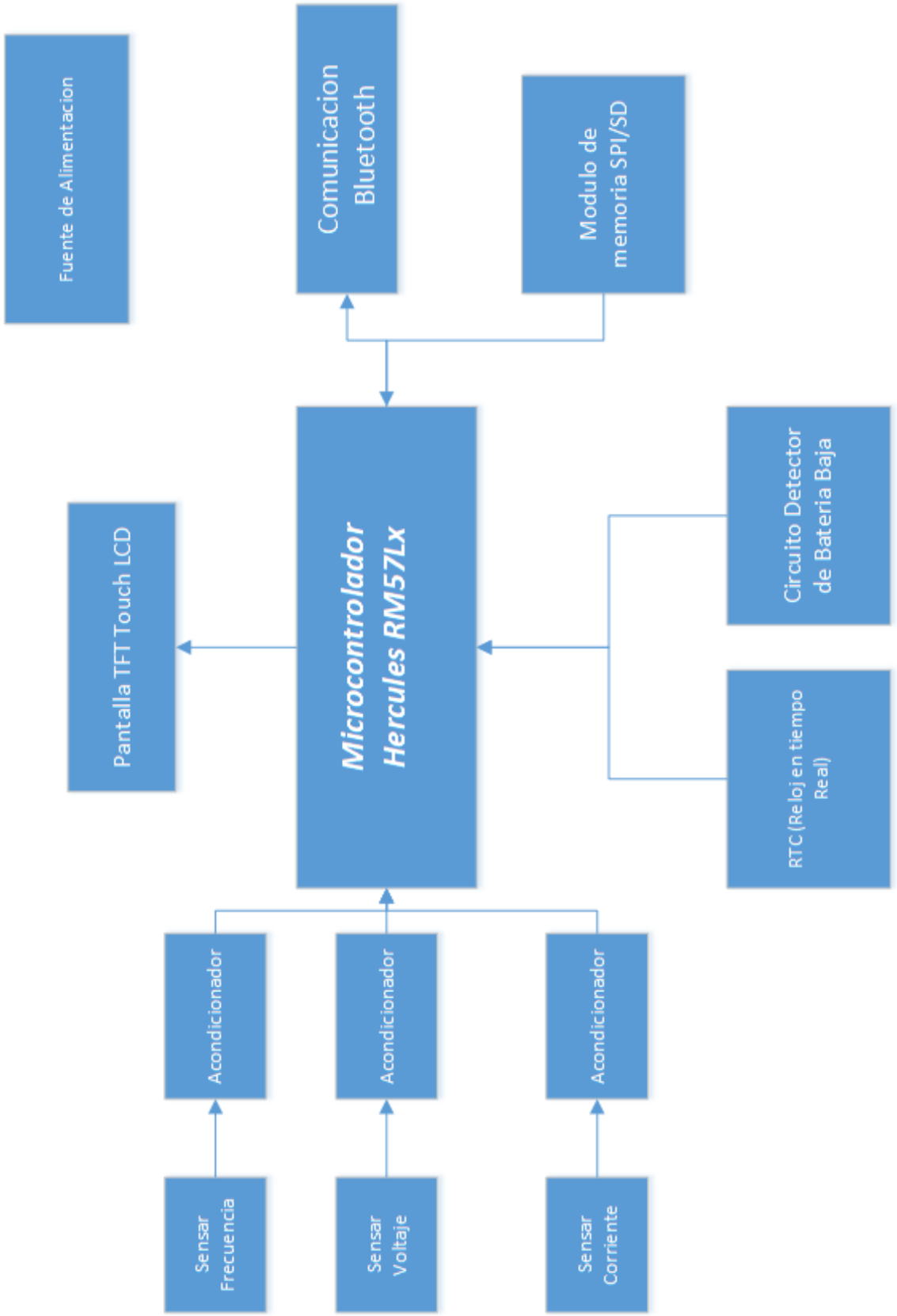


FIGURA 3.1 Diagrama de bloques general del Analizador de calidad de Energía.

a. DISPOSICION DE CIRCUITOS

i. Circuito N° 1: Sensor de Voltaje.

ii. Circuito N° 2: Sensor de Corriente.

iii. Circuito N° 3: Sensor de Frecuencia.

iv. Circuito N° 4: Circuito de Visualización.

v. Circuito N° 5: Circuito de Bluetooth

vi. Circuito N° 6: Circuito Fuente de Alimentación

vii. Circuito N° 7: Circuito para Reloj y Memoria

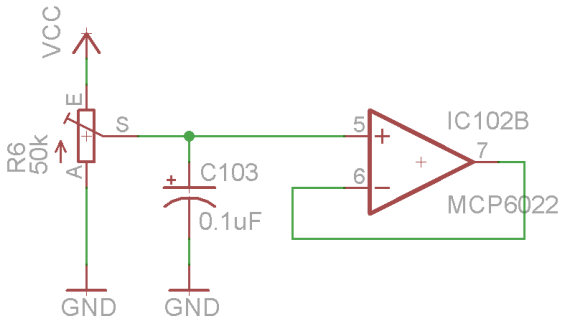
II. ELEMENTOS DE APLICACIÓN GENERAL

a. CIRCUITO DE CONTROL OFFSET

El circuito está conformado por un potenciómetro, un capacitor y un OPAMP en configuración seguidor de voltaje.

Este circuito nos permite una operación con OPAMP que trabajan en con una sola fuente de alimentación, generando un tierra virtual para el amplificador

Los OPAMP de fuente simple requieren una tierra virtual, usualmente un voltaje igual a Vcc/2.

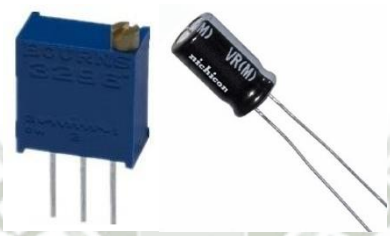


Esquema 3.1 Circuito generador de tierra virtual

El condensador C103 forma un filtro de paso bajo para eliminar el ruido realizado en la línea de tensión.

Para generar la tierra virtual se utilizó un potenciómetro de 50k y multivuelta, para mejorar la precisión de voltaje y para evitar la variación de voltaje se utilizó un condensador de 10uF con un voltaje de 16Voltios.

Ambos forman una red RC, que actúa como filtro pasabajo.



Esquema 3.2 Potenciómetro multivuelta y condensador polarizado

b. CARACTERISTICAS DEL OPAMP IMPLEMENTADO

MICROCHIP MCP6022

• Rail-to-Rail Input/Output

• Wide Bandwidth: 10 MHz (typical)

• Low Offset Voltage:

- Industrial Temperature: ±500 µV (maximum)

- Extended Temperature: ±250 µV (maximum)

• Low Supply Current: 1 mA (typical)

• Power Supply Range: 2.5V to 5.5V

• Temperature Range:

- Industrial: -40°C to +85°C / Extended: -40°C to +125°C

III. DIAGRAMA DE BLOQUES DEL CIRCUITO SENSOR DE VOLTAJE

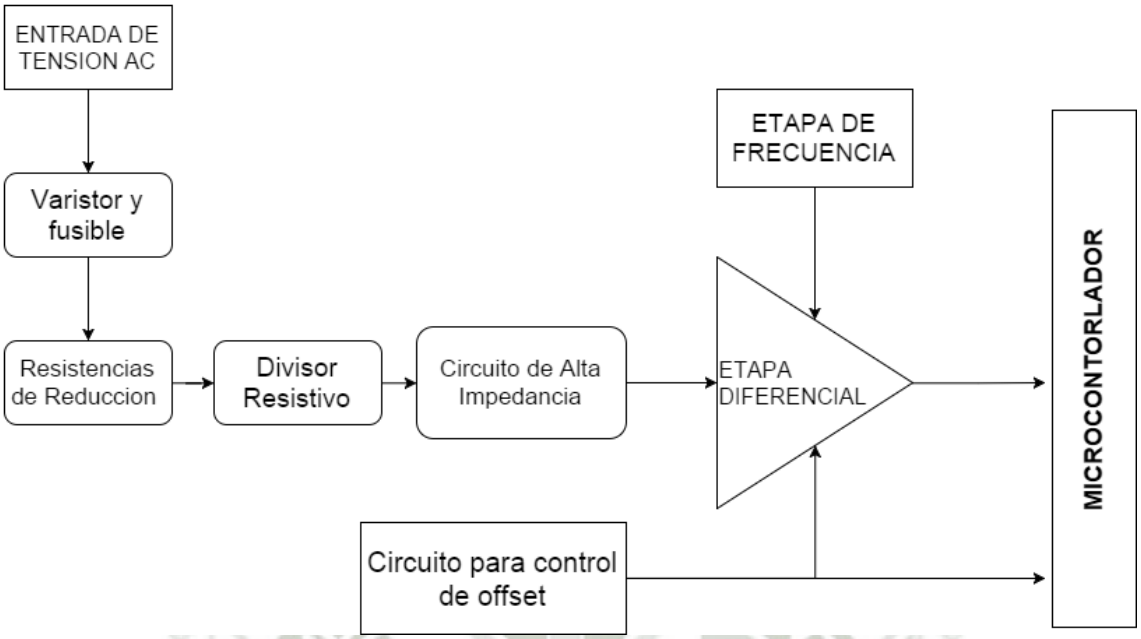


FIGURA 3.2 Diagrama de bloques del circuito sensor de voltaje

a. FUNCIONAMIENTO

i. CIRCUITO VARISTOR Y FUSIBLE

El Varistor ofrece una protección a nuestro circuito si es que existiese una sobre tensión, en este caso por picos voltaje en la línea eléctrica alterna, este siempre debe estar acompañado de un fusible.

El Varistor para proteger el circuito contra una sobretensión, baja su resistencia y esto genera un alto consumo de corriente, el fusible al detectar esto se abre e impide el paso de la tensión eléctrica.

ii. CIRCUITO ATENUADOR DE VOLTAJE

Este circuito se enfoca en reducir el voltaje sensado de 600Vrms a una señal alterna de baja tensión (mV) y muy poca corriente.

Es un juego de resistencias entre las cuales se reparte la potencia eléctrica de entrada como también provee una gran impedancia de entrada, lo que genera un bajo consumo de corriente.

iii. DIVISOR RESISTIVO

El objetivo del circuito divisor resistivo es generar un voltaje de offset en el circuito para que este esté sobrepuesta la señal alterna ya reducida por las resistencias, también esto me permite crear una tierra virtual de referencia para la señal alterna de modo que los OPAMP pueden trabajar con una señal alterna con offset.

iv. CIRCUITO DE ALTA IMPEDANCIA

El circuito de alta impedancia está conformado por OPAMPs en configuración seguidor de voltaje.

Gracias al mínimo consumo de corriente que este tiene en sus entradas, es posible generar una etapa de protección y alta impedancia para el circuito sensor de voltaje.

v. ETAPA DIFERENCIAL

Este circuito trabaja con ambas entradas, invertidas y no invertidas para producir una salida de tensión que es igual a la diferencia de las dos entradas.

Se aplica un voltaje offset de modo que este nos permita mover la salida de tensión.

b. ESPECIFICACIONES TECNICAS

Voltaje de Entrada	:	600Vrms
Voltaje de Salida	:	1000 Veces menos
Frecuencia de Trabajo	:	50-60Hz

c. SELECCIÓN Y CALCULOS

i. CIRCUITO VARISTOR Y FUSIBLE

Para seleccionar el Varistor se debe tener en cuenta:

Tensión de servicio (Veff)

Corriente de choque (i<imax)

Absorción de Energía (E<Emax)

En nuestra aplicación el Varistor no debe soportar una gran cantidad de corriente alterna, debido de que gracias a las resistencias de reducción, la corriente que circula por el varistores está en el orden de los microamperios, por lo tanto la característica importante sería la tensión de servicio la cual debe estar por encima de los 600Vrms.

La Absorción de Energía no debe ser muy alta debido a que no hay un alto consumo de corriente eléctrica.

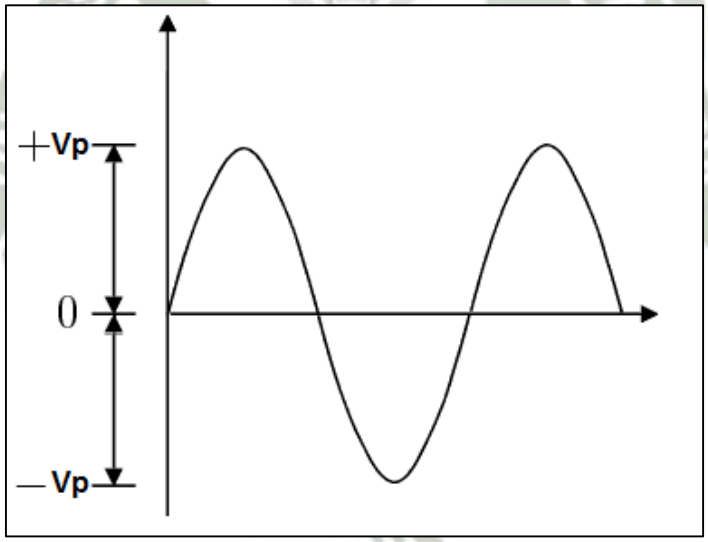
ii. CIRCUITO ATENUADOR DE VOLTAJE Y DIVISOR RESISTIVO

Para el diseño de esta circuito se debe tener en consideración el voltaje pico del máximo voltaje de trabajo en la red eléctrica AC.

Dentro de nuestras especificaciones el voltaje máximo es de 600Vrms, pero nuestro voltaje Pico seria

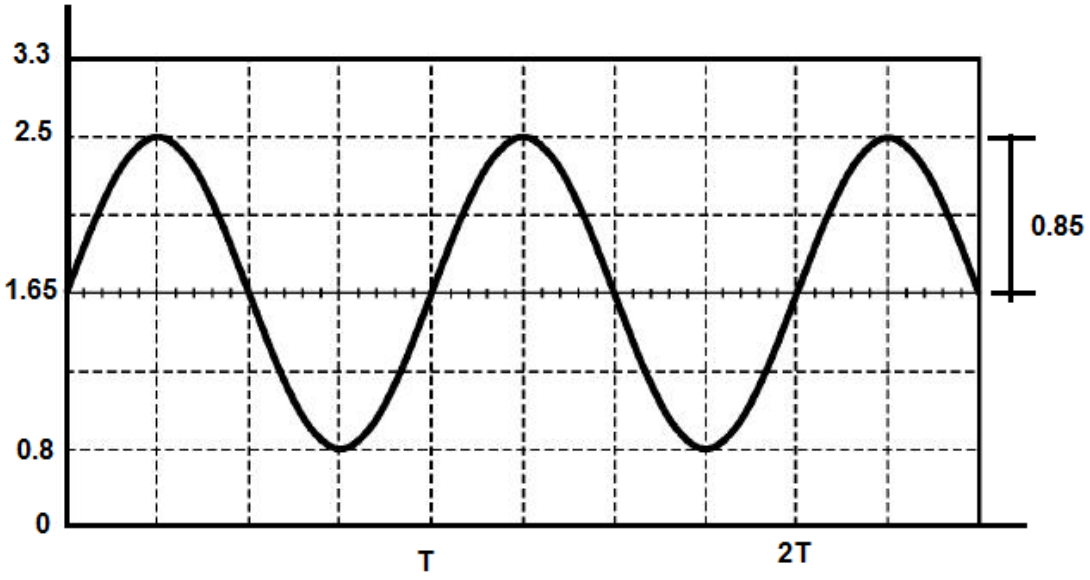
$$Vp = \sqrt{2} * 600Vrms$$

$$Vp = 849\,Vp$$



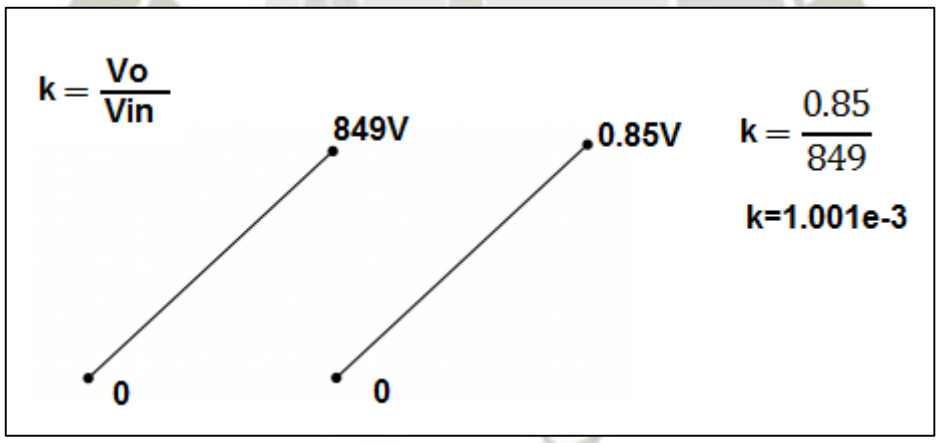
Esquema 3.3 Señal alterna con máximos y mínimos AC

De acuerdo a este último calculo nosotros tenemos una tensión máxima instantánea de 849 Voltios, esto debe ser reducido 1000 veces para que pueda entrar en una ***ventana de voltaje*** entre 0 y 2.5Voltios.



Esquema 3.4 Señal alterna dentro de ventana de voltajes 0 a 3.3v

Para conseguir lo mencionado anteriormente es necesario tener un realizar una ecuación de relación, entre la entrada de voltaje y la salida de voltaje deseado:

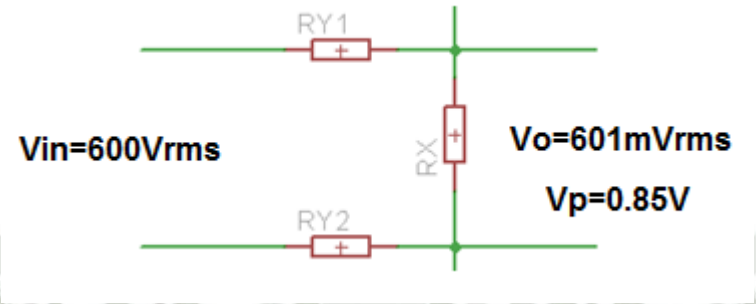


Esquema 3.5 Relación entre la entrada de voltaje y la salida deseada

$$\frac{Vo}{Vin} = \frac{1}{1000}$$ (Ecuación 3.1)

Considerando los siguientes cálculos, se tiene que la tensión de entrada está dentro de 0 y 849 Voltios Pico, por lo tanto la salida de tensión debe estar dentro de los 0 y 0.85Voltios Pico, respetando la atenuación de 1000 veces menos y trabajando dentro de los valores límites de la venta de voltaje

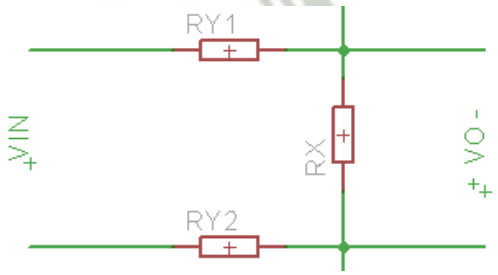
Por lo tanto se debe tener el nivel en RMS seria:



Esquema 3.6 Divisor Resistivo para el Sensado de voltaje

Para el diseño del circuito de CIRCUITO ATENUADOR DE VOLTAJE, se tuvo en mente dos ideas:

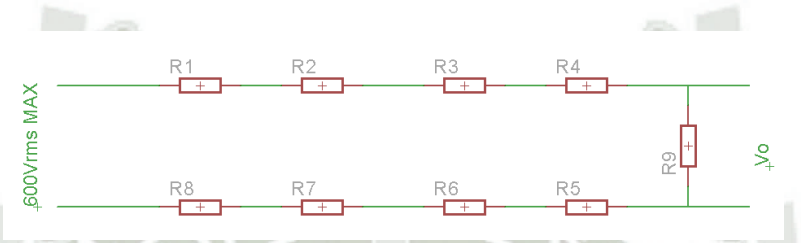
Resistencias de alto valor RY1, RY2, son resistencias cercanas a los 0.5 Mega ohmios, de modo que existía un bajo consumo de corriente eléctrica por la red resistiva.



Esquema 3.7 Divisor Resistivo como variable

El problema con este juego de resistencias es la baja potencia eléctrica que soporta, por motivo de ser resistencias de ¼ de watt, estas no podría soportar una sobretensión.

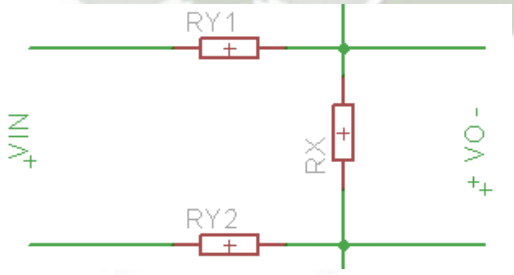
Por lo tanto se repartió la carga en pequeñas resistencias que en su totalidad sumaran la de una de alta impedancia, de esta manera se tiene varias resistencias de ¼ watt, que al sumar sus potencias generan una impedancia que puede soportar una sobretensión eléctrica sin daños en las resistencias.



Esquema 3.8 Divisor Resistivo de varias impedancias

DISEÑO

Para el cálculo del valor de las resistencias se debe tener en cuenta el valor de atenuación hallado anteriormente:



$$Vo = \frac{Vin * RX}{2RY + RX} \qquad \text{(Ecuación 3.2)}$$

$$\frac{Vo}{Vin} = \frac{RX}{2RY + RX}$$

Reemplazando 3.1 en 3.2

$$\frac{RX}{2RY + RX} = \frac{1}{1000}$$

La relación entre RX y RY es:

$$RX = 2.002e - 3 * RY \text{ (3.3)}$$

Seleccionando los valores de RY y RX, con el criterio de que estas deben ser resistencias de alto valor para que generar un consumo mínimo de corriente.

$$RY = 476K\Omega$$

$$RX = 470K\Omega * 2.02e - 3$$

$$RX = 949.4\Omega$$

Los valores Seleccionador por diseño son:

$$RY = 470K\Omega$$

$$RX = 1k\Omega$$

Para conocer la potencia disipada en las resistencias, es necesario la cantidad de corriente alterna que ira por ellas, aplicamos Kirchoff en la red RY-RX-RY.

$$I = \frac{Vp}{2RY+RX} \text{ (Ecuación 3.4)}$$

Donde:

Vp: es el voltaje pico de la señal alterna

Rx, RY: resistencias de entrada.

Vp=850 Voltios

$$I = \frac{850}{2(470K) + 1K}$$

$$I = 891uA$$

Potencia en RY:

$$Py = I^2 Ry = 0.35Watt$$

Potencia en Rx:

$$Px = I^2 Rx = 793uWatt$$

Como se había planteado anteriormente la potencia disipada por Ry es bastante alta por lo que la mejor opción fue distribuir la carga de potencia entre diferentes resistencias.

Se escogió 4 resistencias de diferentes valores como:

$$Ry = R1 + R2 + R3 + R4 \quad \text{(Ecuación 3.5)}$$

No se consideró ninguna relación en especial para la selección de las resistencias en la ecuación 3.5.

$$R1 = R2 = 100K\Omega$$

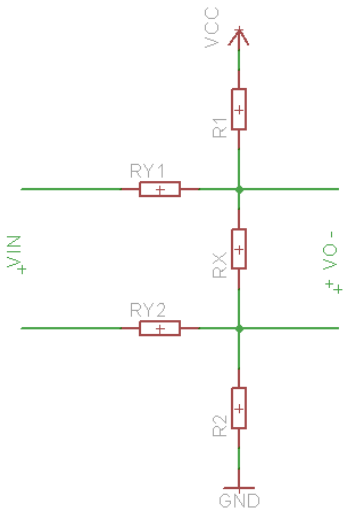
$$R2 = 220K\Omega$$

$$R4 = 56K\Omega$$

El nuevo valor para RY es de 476kΩ

Importante: La salida de este circuito da una señal alterna sin referencia a masa del circuito, por lo que se hizo un arreglo de resistencias de modo que obtenga un voltaje offset y una referencia a masa del circuito.

Arreglo de resistencias para dar referencia a masa



Esquema 3.9 Divisor Resistivo con referencia a masa y fuente DC.

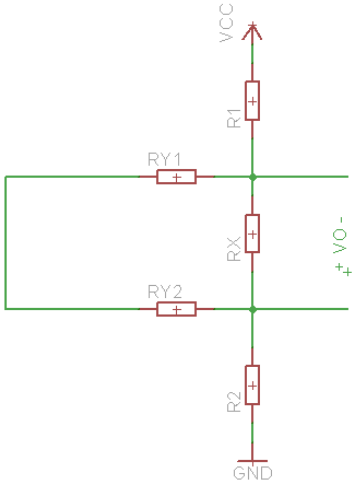
El circuito propuesto se analiza aplicando teoría de **superposición de voltajes.**

"Este teorema establece que el efecto que dos o más fuentes tienen sobre una impedancia es igual, a la suma de cada uno de los efectos de cada fuente tomados por separado, sustituyendo todas las fuentes de voltaje restantes por un corto circuito, y todas las fuentes de corriente restantes por un circuito abierto." (Wikipedia)

$$V_T = f(V_1, V_2) = f(V_1, 0) + f(0, V_2)$$

Desarrollo:

Según el teorema de la superposición de voltaje, para encontrar la ecuación del voltaje de salida con respecto a la primera fuente de alimentación es necesario cortocircuitar la fuente de tensión alterna.



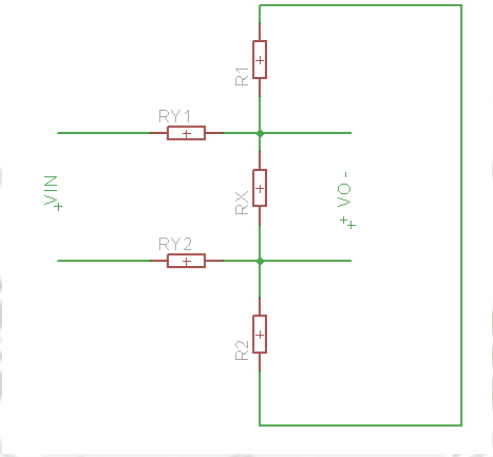
Esquema 3.10 Fuente AC en cortocircuito.

Del siguiente circuito es posible encontrar una relación entre el voltaje de salida y la tensión DC de alimentación.

(Ecuación 3.6)

$$Vo(vcc) = \frac{Vcc(2Ry \parallel Rx)}{2Ry \parallel Rx + R1 + R2}$$

Fuente de tensión continúa cortocircuitada



Esquema 3.11 Fuente DC en cortocircuito.

(Ecuación 3.7)

$$Vo(Vin) = \frac{Vin(Rx \parallel (R1 + R2))}{2Ry + Rx \parallel (R1 + R2)}$$

Voltaje de salida en función del voltaje DC. (0-3.3V)	$Vo(Vcc) = \frac{Vcc(2Ry \parallel Rx)}{2Ry \parallel Rx + R1 + R2}$
Voltaje de salida en función del voltaje AC. (0-600Vrms)	$Vo(Vin) = \frac{Vin(Rx \parallel (R1 + R2))}{2Ry + Rx \parallel (R1 + R2)}$

Según el teorema de Superposición, ambas soluciones se deben sumar:

(Ecuación 3.8)

$$Vo(Vin, Vcc) = Vo(Vin, 0) + Vo(0, Vcc)$$

$$Vo(Vin, Vcc) = \frac{Vin(Rx \parallel (R1 + R2))}{2Ry + Rx \parallel (R1 + R2)} + \frac{Vcc(2Ry \parallel Rx)}{2Ry \parallel Rx + R1 + R2}$$

R1=R2=R >>Rx **:** R1 y R2 son de igual valor y 10 veces mayores a Rx para evitar un alto consumo de corriente

Para Rx=1.5k

R=22k >>1.5k

R1=22k

R2=22k

Valores de las resistencias en el circuito:

RY	476K
RX	1.5K
R1	22K
R2	22K
Vcc	3.3V
Vin	V(t)

Nota: los valores de las resistencias Rx, R1 y R2, fueron seleccionados luego de diferentes pruebas hechas con el circuito implementado.

Reemplazando los valores en la ecuación 3.8

$$Vo(Vin, Vcc) = \frac{V(t)(1.5k \parallel (22k + 22k))}{2(476k) + (1.5k \parallel (22k + 22k))} + \frac{3.3(2(476k) \parallel 1.5k)}{(2(476k) \parallel 1.5k) + 22k + 22k}$$

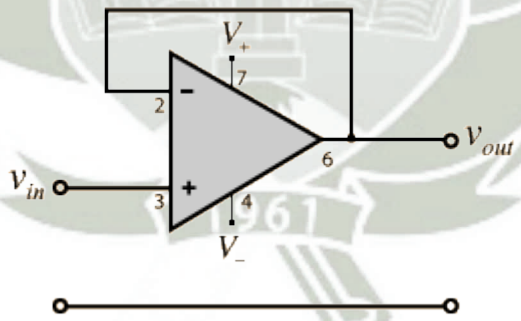
$$Vo(Vin, Vcc) = 1.52136e - 3 * V(t) + 0.1086V$$

De esta última ecuación se desprende que existe una atenuación de 1.5213e-3 para el voltaje alterna.

El circuito divisor resistivo agrega un voltaje offset de 0.1086V, este valor es aún muy alto y es necesario reducirlo; sin la necesidad de alterar los valores de las resistencias, simplemente se le resta el valor offset via software del Microcontrolador.

iii. CIRCUITO DE ALTA IMPEDANCIA

El circuito de alta impedancia está conformado, por OPAMP en configuración seguidor de voltaje



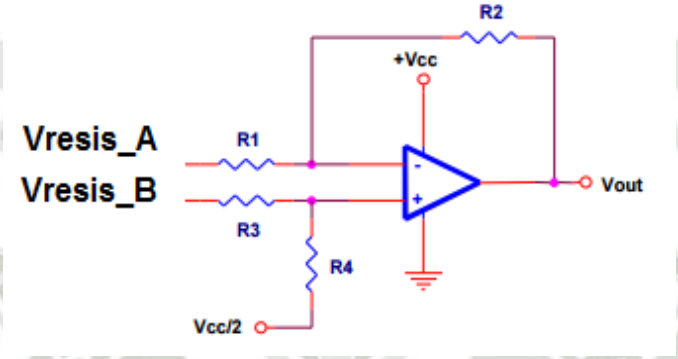
Esquema 3.12 OPAMP configuración seguidor de voltaje

La configuración seguidor de voltaje no produce una amplificación de voltaje, este simplemente entrega el mismo voltaje que entra por su pin V+.

Pero su función dentro del circuito es aportar una alta impedancia de entrada de modo que las ecuación (3.8), no se vea afectada por las resistencias de otros circuitos.

iv. ETAPA DIFERENCIAL

El circuito amplificador diferencial, entrega una salida de tensión en función de la resta de sus entradas.



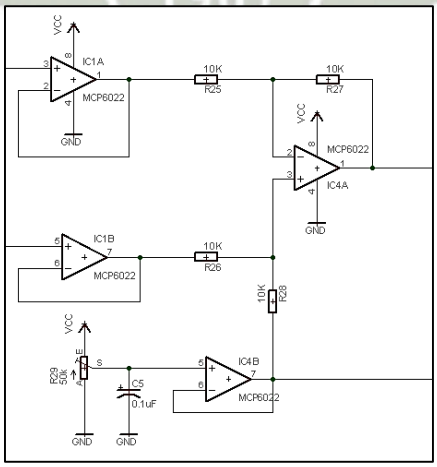
Esquema 3.13 OPAMP Etapa diferencial

En la implementación el circuito tiene ganancia unitaria; R2=R1=R3=R4=10k.

(Ecuación 3.9)

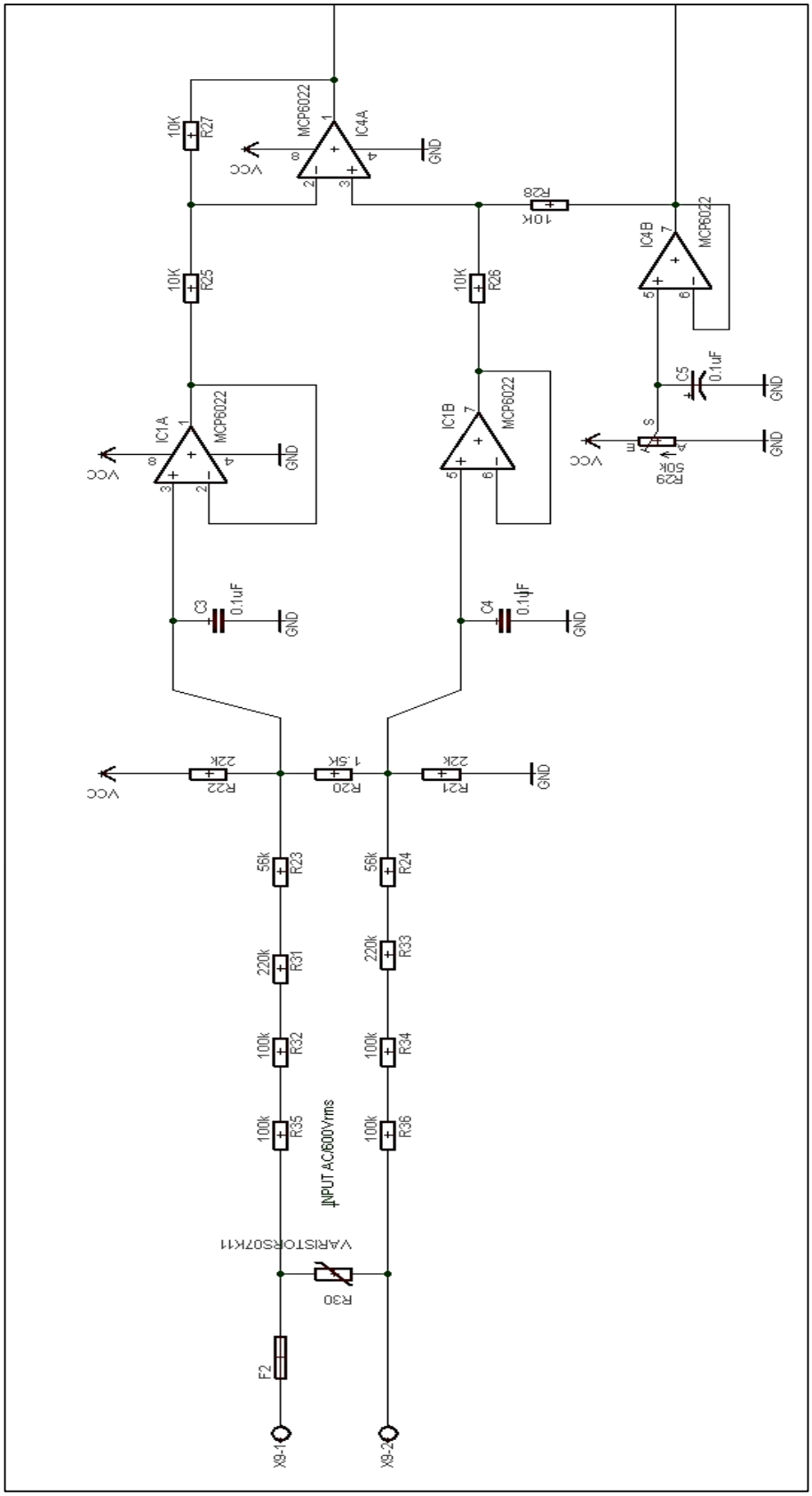
$$\boldsymbol{Vo = \frac{R2}{R1}(V2 - V1)}$$

Utilizando la etapa diferencial y el circuito offset, se obtiene una tierra virtual para señal Análoga que ingresara al microcontrolador.



Esquema 3.14 Circuito generador de tierra virtual

d. DISEÑO DEL CIRCUITO



Esquema 3.15 Circuito general

e. DISPOSITIVOS SELECCIONADOS

i. CIRCUITO VARISTOR Y FUSIBLE

Esquema 3.16 Varistor

Circuito para protección contra sobre voltajes.

- Tipo: MOV-20D102K
- Voltaje Max r.m.s: 625V
- Voltaje DC: 825Vdc
- Corriente Máxima: 100 A
- Capacitancia @1KHz = 400pF
- Máxima Energía: 310.0 Joule

ii. CIRCUITO ATENUADOR DE VOLTAJE Y DIVISOR RESISTIVO

Para el circuito reductor de potencia, se aplicaron resistencias de precisión al %1, con una potencia de 1/4w, dado que son resistencias SMD su nivel de precisión es bastante alta; en el primer prototipo se utilizaron resistencias de carbón con el 5% de variación, esto genera mediciones erróneas con el tiempo.

Esquema 3.17 Resistencias SMD

i. CIRCUITO DE ALTA IMPEDANCIA, ETAPA DIFERENCIAL

Para esta etapa es de vital importancia un OPAMP con una salida de tensión RAIL TO RAIL y un bajo voltaje offset.

También se requiere que el OPAMP trabaje con una sola fuente de alimentación.

IV. DIAGRAMA DE BLOQUES DEL CIRCUITO SENSOR DE CORRIENTE

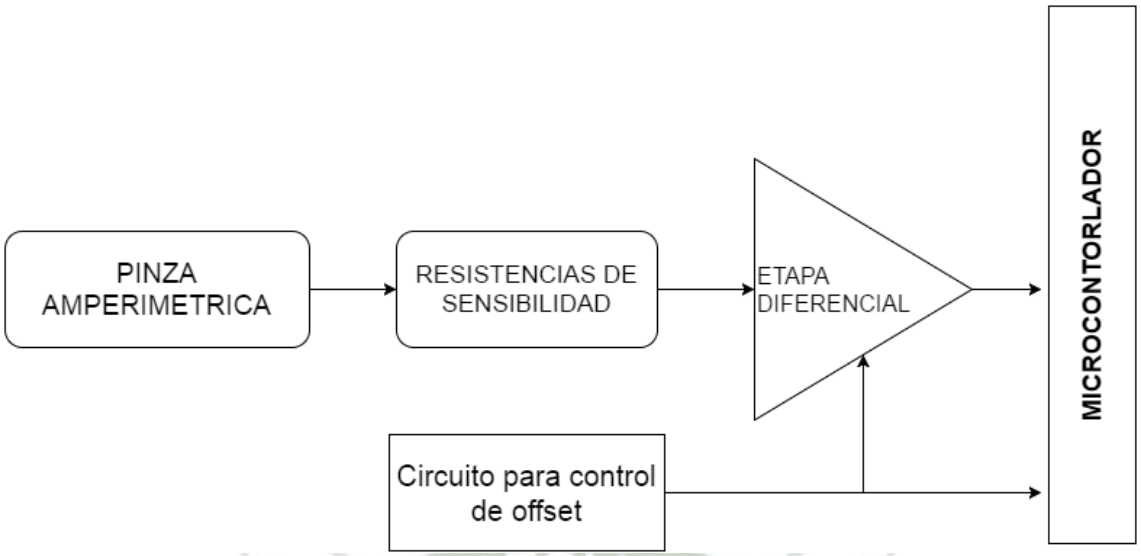


FIGURA 3.3 Diagrama de bloques del circuito sensor de Corriente

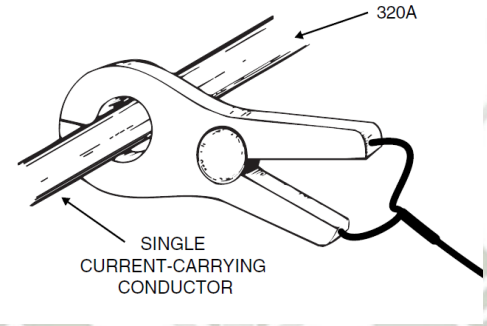
a. FUNCIONAMIENTO

Este circuito se encarga de censar la corriente que consume la carga eléctrica, esto es posible gracias a la pinza amperimetrica la cual entrega una señal que debe ser acondicionada para ser censada por el microcontrolador.

i. PINZA AMPERIMETRICA

El funcionamiento de la pinza se basa en la medida indirecta de la corriente circulante por un conductor a partir del campo magnético o de los campos que dicha circulación de corriente genera. Recibe el nombre de pinza porque consta de un sensor, en forma de pinza, que se abre y abraza el cable cuya corriente queremos medir.

Este método evita abrir el circuito para efectuar la medida, así como las caídas de tensión que podría producir un instrumento clásico. Por otra parte, es sumamente seguro para el operario que realiza la medición, por cuanto no es necesario un contacto eléctrico con el circuito bajo medida ya que, en el caso de cables aislados, ni siquiera es necesario levantar el aislante.



Esquema 3.18 Pinza Amperimetrica

ii. RESISTENCIAS DE SENSIBILIDAD

También conocidos como resistencias shunt, nos permite convertir la corriente eléctrica en voltaje.

Debido a que la pinza eléctrica cuenta con un alto rango de mediciones es necesario incorporar una etapa donde se pueda modificar la resolución de la corriente.

Es necesario utilizar diferentes valores de resistencias para poder tener una buena precisión en los diferentes rangos de corriente a medir.

iii. ETAPA DIFERENCIAL

Esta etapa del circuito recibe la señal de voltaje que cae en las resistencias de sensibilidad, tiene como ganancia unitaria y no aplica ningún tipo de filtro.

No es necesario aplicar filtros análogos porque estos cambiarían la fase y eliminaría algunos armónicos importantes de la señal de corriente alterna.

b. ESPECIFICACIONES TECNICAS

Corriente Máxima	:	100 Arms
Corriente de Salida	:	1000 Veces menor
Frecuencia de Trabajo	:	50-60Hz
Ancho de Banda	:	50 KHz
Corriente Minima	:	1A

c. SELECCIÓN Y CALCULOS

i. PINZA AMPERIMETRICA

Para la implementación del dispositivo sensor de corriente se utilizó una pinza amperimetrica de la marca FLUKE.

Especificaciones:

Modelo	: 80i-600ª.
Salida	: 1miliamperio por Amperio (1mA/A).
Voltaje de trabajo	: 750 V ac rms máximo.
Ancho de banda	: 10 Hz hasta 50 KHz.

Rango de sobrecorriente: 0.1 A hasta 2000 A por 5 segundos.

ii. RESISTENCIAS DE SENSIBILIDAD

En este apartado se considera la pinza amperimetrica ya definida de modo que nosotros tenemos la siguiente tabla dada por el fabricante FLUKE

Rango de corriente de entrada.	Rango de resistencias recomendado
1A to 10A	20 mA (10Ω Shunt)
10A to 200A	200 mA (1Ω Shunt)
200A to 600A	2000 mA (0.1Ω Shunt)

Tabla 3.1 Guía para selección de resistencias.

Partiendo de esta tabla se extrae lo siguiente:

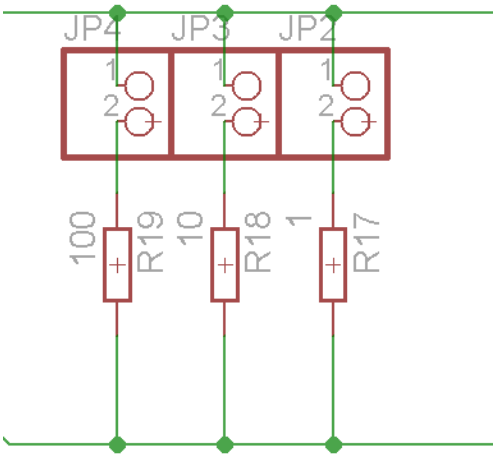
1 OHM para 10 A hasta 200 A

10 OHM para 1 A hasta 10 A

Basándonos en esta relación nosotros podemos afirmar lo siguiente:

100 OHM para 0.1 hasta 1 A

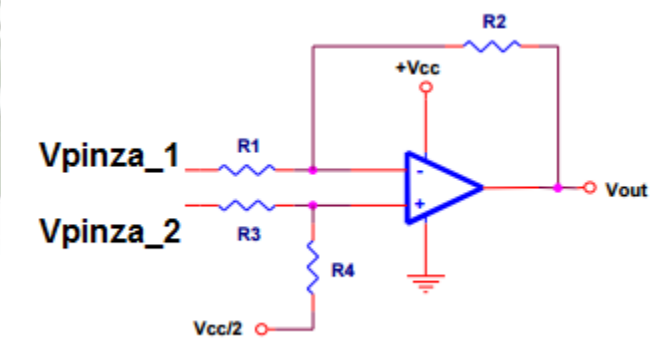
Con el valor de las resistencias seleccionadas nosotros podemos hacer un arreglo de resistencias en paralelo, todas estas estarán solamente conectadas por un JAMPER, de modo que nosotros podemos seleccionar el rango de corriente entre las que trabajaremos.



Esquema 3.19 Resistencias de sensibilidad

iii. ETAPA DIFERENCIAL

El circuito amplificador diferencial, entrega una salida de tensión en función de la resta de sus entradas.



Esquema 3.20 OPAMP Etapa diferencial

En nuestra implementación el circuito tiene ganancia = 10, de modo que la forma de onda pueda ser mejor apreciada en la pantalla LCD (mayor detalle en la etapa de software).

R2=R1=10k

R3=R4=100k

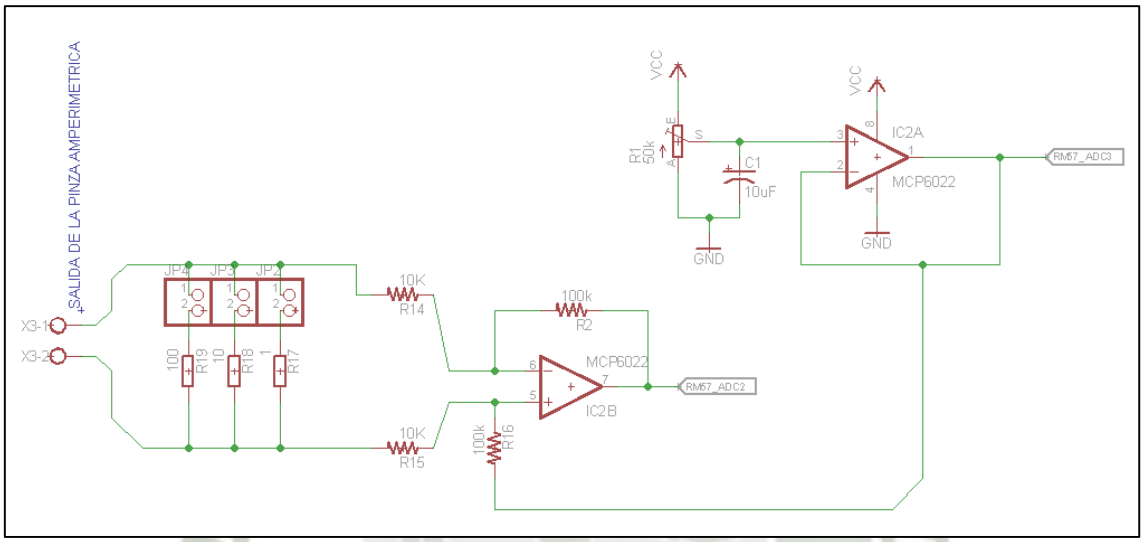
(Ecuación 3.11)

$$Vo = \frac{R2}{R1}(V2 - V1)$$

$$Vo = \frac{100k}{10k}(V2 - V1)$$

$$Vo = 10(V2 - V1)$$

d. DISEÑO DEL CIRCUITO



Esquema 3.21 OPAMP Etapa diferencial

e. DISPOSITIVOS SELECCIONADOS

i. RESISTENCIAS DE SENSIBILIDAD

Para el circuito reductor de potencia, se aplicaron resistencias de precisión al %5, con una potencia de 1/4w.

Esquema 3.22 Resistencias de Carbón

ii. ETAPA DIFERENCIAL

Para esta etapa es de vital importancia un OPAMP con una salida de tensión RAIL TO RAIL y un bajo voltaje offset.

También se requiere que el OPAMP trabaje con una sola fuente de alimentación.

V. DIAGRAMA DE BLOQUES DEL CIRCUITO SENSOR DE FRECUENCIA

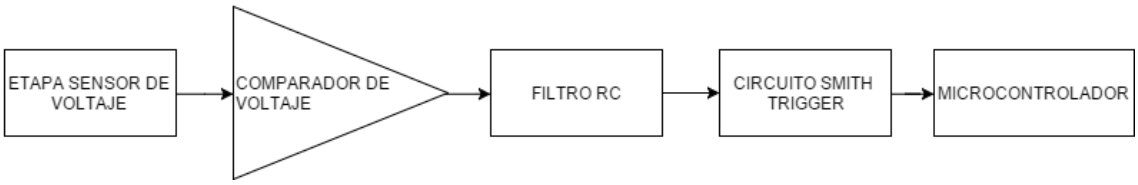


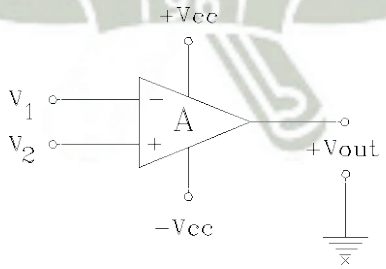
FIGURA 3.4 Diagrama de bloques del circuito sensor de frecuencia

a. FUNCIONAMIENTO

Este circuito tiene como función el tomar la señal de voltaje sensado por el divisor resistivo y comprarlo con un señal de referencia para así obtener una señal cuadrada con la misma frecuencia que la señal de voltaje de línea.

i. COMPARADOR DE VOLTAJE

Los compradores son amplificadores que sirven para comprar dos señales y determinar tiene un nivel de voltaje mayor a la otra.



Esquema 3.23 Comparador de voltaje

Su utilización en las aplicaciones de generación de señal, detección, modulación de señal, etc, es muy importante y constituye un bloque analógico básico en muchos circuitos. La función del comparador es comparar dos tensiones obteniéndose como resultado una tensión alta (V_{OH}) o baja (V_{OL}).

La ecuación de un comparador se puede expresar como:

Vo=-Vcc si V1>V2

Vo=Vcc si V1<V2

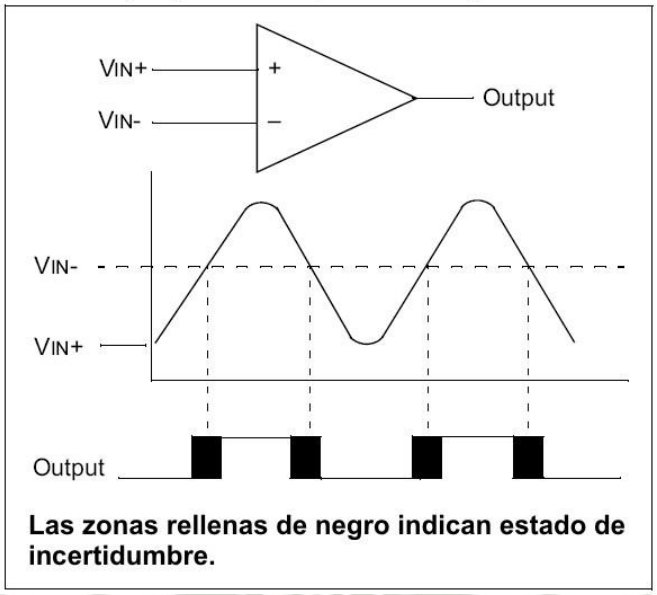
ii. FILTRO RC PASIVO

El comparador de voltaje presenta una zona de incertidumbre donde la salida de tensión tiene cambios de estado por pequeños microsegundos.

Dado que la precisión del temporizador interno del microcontrolador es en microsegundos, el microcontrolador puede medir estos pequeños cambios de estado y los asume como una frecuencia, dando mediciones falsas sobre esta en el equipo.

Por lo que es necesario implementar un circuito filtro, que evite los cambios de estado indeseados en la salida de tensión.

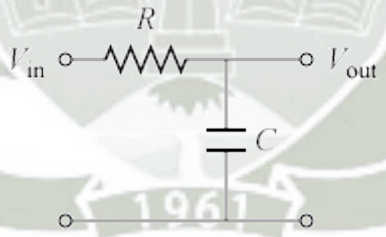
Ejemplo de comparador simple



Las zonas rellenas de negro indican estado de incertidumbre.

El filtro RC es un circuito compuesto por una resistencia y un condensador alimentados por una fuente de voltaje.

Este circuito puede utilizarse para filtrar señales de voltaje con cierta frecuencia y dejar pasar otras.



Esquema 3.24 Filtro pasabajo

Su ecuación para la frecuencia de corte es la siguiente:

$$f_c = \frac{1}{2\pi RC.}$$

(Ecuación 3.9)

iii. CIRCUITO SCHMITT TRIGGER

Este circuito usa el principio de histéresis para prevenir que el ruido tenga alguna influencia no deseada en la salida de señal deseada.

El circuito falsos cambios de estado si el nivel de entrada y el de referencia tienen el mismo valor de voltaje.

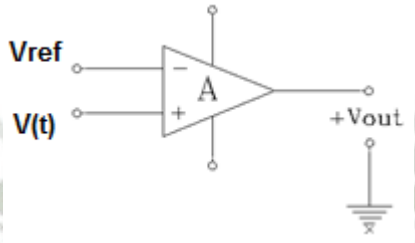
b. ESPECIFICACIONES TECNICAS

Voltaje de entrada	: 0 - 3.3 V (Análoga)
Voltaje de Salida	: 0 - 3.3V (Binaria)
Frecuencias de trabajo	: 40 - 70 Hz

c. SELECCIÓN Y CALCULOS

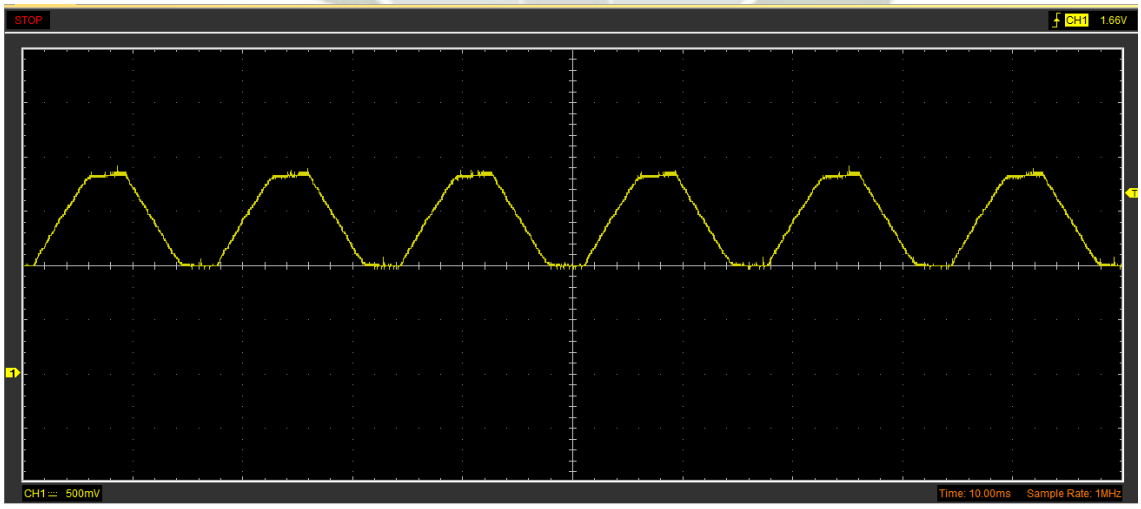
i. COMPARADOR DE VOLTAJE

Tomando como referencia el circuito control de offset del Diagrama de bloque del circuito sensor de voltaje, obtenemos la señal de referencia para el comparador la cual es una señal que con un voltaje vcc/2.



Esquema 3.25 Comparador de Voltaje

Cuando la señal de V(t) supera en voltaje al nivel de referencia, el comparador entrega una señal de salida Vcc o GND cuando el voltaje es menor al nivel de referencia.



Esquema 3.26 Señal V(t), antes de ingresar al comparador.

ii. FILTRO RC PASIVO

En esta etapa de diseño estamos definiendo el ancho de banda en el que trabajara el equipo analizador de calidad de energía.

Para esta etapa del diseño se debe tener en cuenta que el equipo debe tener una frecuencia de trabajo entre 40-70 Hz, como se definió en las especificaciones del equipo en el capítulo 2.

Aplicando la ecuación 3.9

$$f_c = \frac{1}{2\pi RC}$$

Para aplicar la ecuación 3.9 es necesario dar un valor a C=0.1uF y nuestra frecuencia limite será f=70Hz.

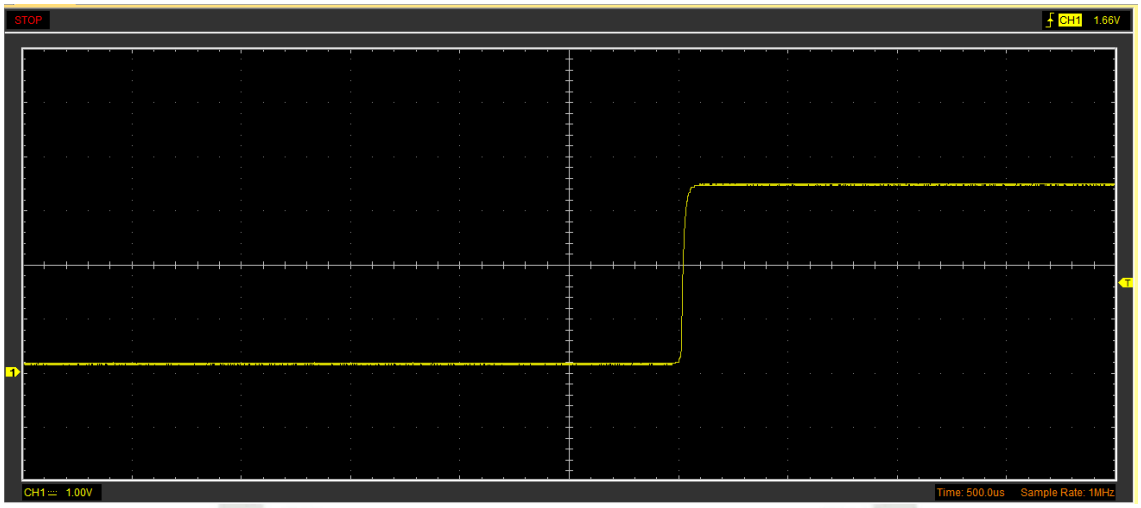
Der la ecuación despejamos R:

$$R = \frac{1}{2\pi RF}$$

Si reemplazamos los valores en la ecuación obtenemos el valor de R:

R=22.73Kohm

El valor estándar seleccionado de la resistencia seria R=22Kohm.

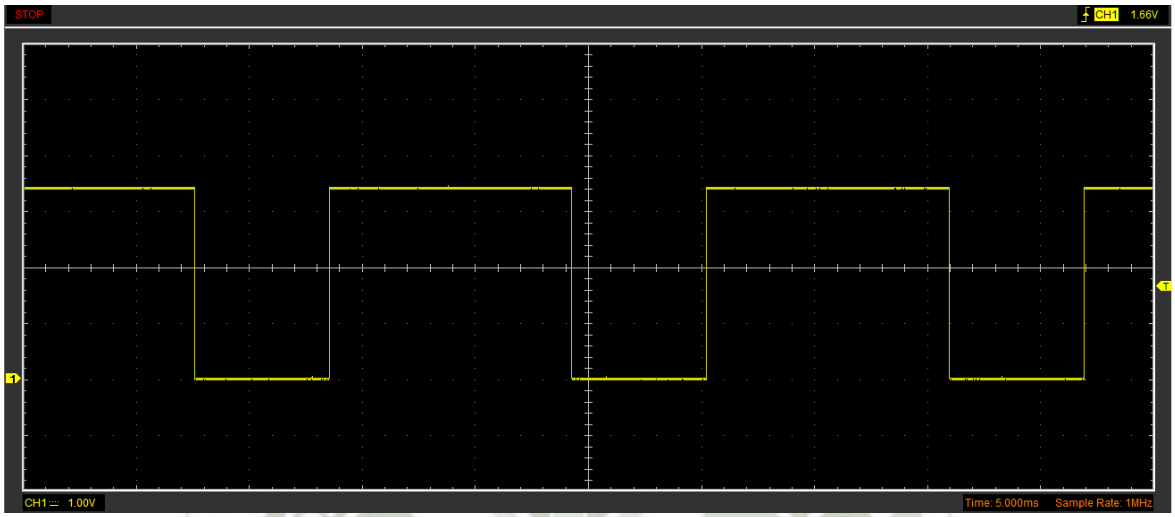


Esquema 3.27 Señal Binaria, después del filtro pasabajo, se puede observar que el flanco de subida no se encuentra correctamente definido.

iii. CIRCUITO SCHMITT TRIGGER

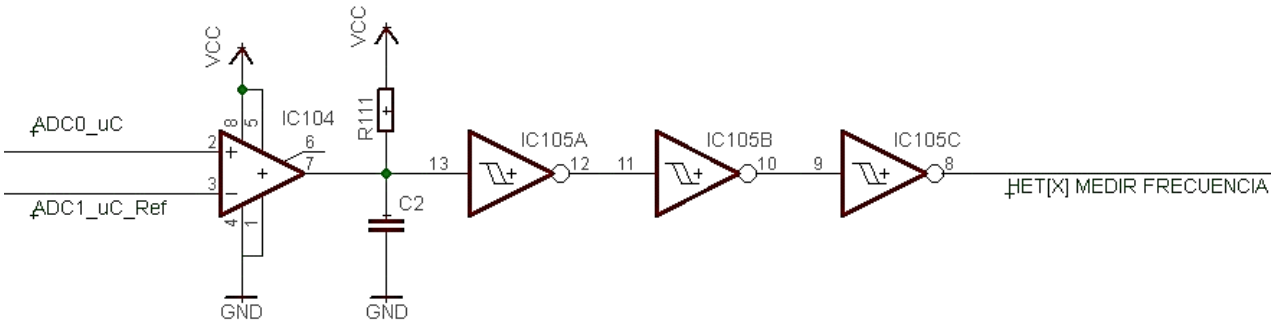
Como Se puede observar en la esquema 3.28, la salida del filtro pasabajo, nos entrega una señal cuadrada no muy bien definida y esto genera mediciones erradas en el temporizador del microcontrolador.

La solución más razonable, es hacer pasar la señal por un juego de compuertas Schmitt Trigger, con el fin de que entreguen una señal perfectamente cuadrada y manteniendo la frecuencia de la señal análoga original.



Esquema 3.28 Señal de salida SCHMITT

d. DISEÑO DEL CIRCUITO



Esquema 3.29 Circuito Sensor de Frecuencia

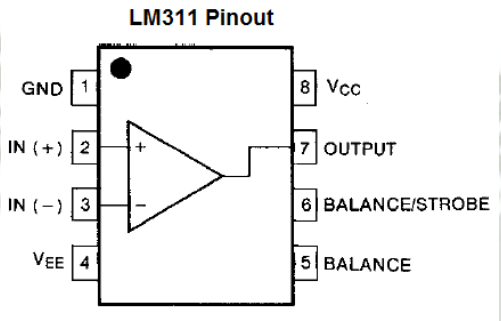
e. DISPOSITIVOS SELECCIONADOS

i. COMPARADOR DE VOLTAJE

El circuito integrado LM311.

Características

- Mínima Corriente de consumo: 250nA (Max) .
- Baja entrada de corriente offset: 50nA (Max).
- Alimentación diferencial: ±30V.
- Simple fuente de alimentación : 5.0V

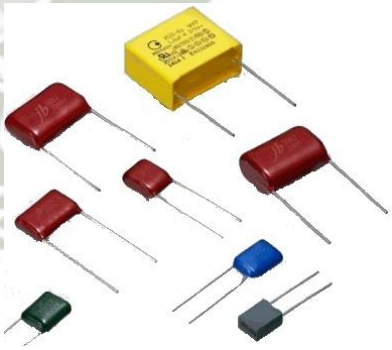


Esquema 3.30 LM311 Comparador

ii. FILTRO RC PASIVO

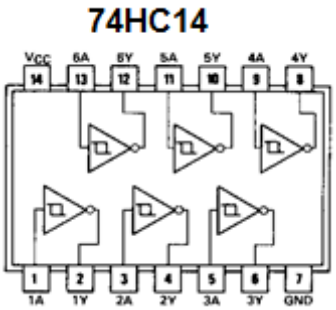
Resistencia de carbón (1/4W).

Condensador de poliéster.



Esquema 3.31 Condensadores no polarizados

iii. CIRCUITO SCHMITT TRIGGER



Esquema 3.32 Circuito Schmitt Trigger

VI. DIAGRAMA DE BLOQUES DEL CIRCUITO DE VISUALIZACION Y ENTRADA

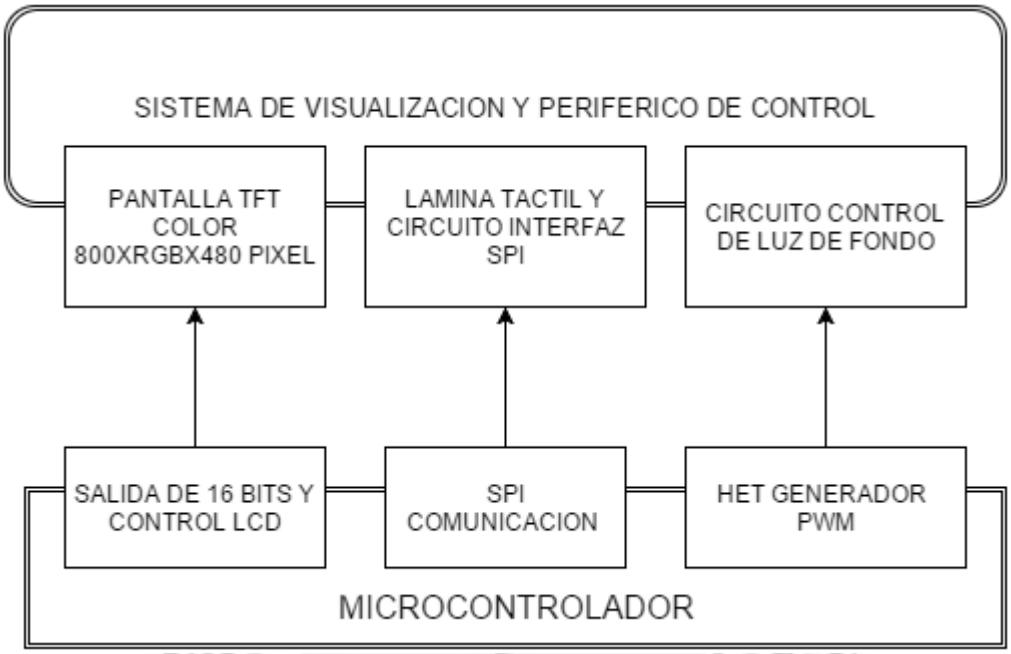


FIGURA 3.5 Diagrama de bloques del circuito de visualización de entrada

a. FUNCIONAMIENTO

El equipo cuenta con una pantalla táctil por donde se seleccionan las tareas que el equipo debe de desarrollar, además de poder visualizar la información que ha sido procesada como:

i. Voltaje, corriente, potencia, frecuencia.

ii. Formas de Onda, Espectro de Fourier

iii. Configuración del equipo

iv. Teclado para la configuración de parámetros.

b. ESPECIFICACIONES TECNICAS

i. Voltaje del Luz de fondo: 5Vdc
ii. Voltaje del LCD microcontrolador: 3.3Vdc
iii. Voltaje de I/O digital: 3.3Vdc
iv. Salida paralela del microcontrolador de 16 bits a LCD.
v. Tamaño de la pantalla: 7 Pulgadas
vi. Resolución: 800XRGBx480
vii. Numero de Colores: 16.7 Millones
viii. Touch: Lamina resistiva.

c. SELECCIÓN Y CALCULOS

A diferencias de la versión LCD de 8bits que la intensidad de su luz de fondo es controlada por una resistencia, las pantallas TFT LCD, se conectan directamente a la fuente y se le controla de forma digital con una señal de PWM.

Según las especificaciones del fabricante este PWM debe tener las siguientes características:

F=300Hz

El ancho de pulso o Dutty Cicle, es el que controla la iluminación de la luz de fondo, luego de varias pruebas se encontró que el 30% es el valor adecuado para una buena iluminación para el ojo humano y disminuir el consumo de corriente de la luz de fondo.

La descripción de cada pin se encuentra detallada en la siguiente Tabla

Pin No.	Symbol	I/O	Function	Remark
1	NC	-	No connection	Note 8
2	NC	-	No connection	Note 8
3	NC	-	No connection	Note 8
4	NC	-	No connection	Note 8
5	GND	P	Power ground	
6	V_{COM}	I	Common voltage	
7	DV_{DD}	P	Power for Digital Circuit	
8	MODE	I	DE/SYNC mode select	Note 1
9	DE	I	Data Input Enable	
10	VS	I	Vertical Sync Input	
11	HS	I	Horizontal Sync Input	
12	B7	I	Blue data(MSB)	
13	B6	I	Blue data	
14	B5	I	Blue data	
15	B4	I	Blue data	
16	B3	I	Blue data	
17	B2	I	Blue data	
18	B1	I	Blue data	Note 2
19	B0	I	Blue data(LSB)	Note 2
20	G7	I	Green data(MSB)	
21	G6	I	Green data	
22	G5	I	Green data	
23	G4	I	Green data	
24	G3	I	Green data	
25	G2	I	Green data	
26	G1	I	Green data	Note 2

27	G0	I	Green data(LSB)	Note 2
28	R7	I	Red data(MSB)	
29	R6	I	Red data	
30	R5	I	Red data	
31	R4	I	Red data	
32	R3	I	Red data	
33	R2	I	Red data	
34	R1	I	Red data	Note 2
35	R0	I	Red data(LSB)	Note 2
36	GND	P	Power Ground	
37	DCLK	I	Sample clock	Note 3
38	GND	P	Power Ground	
39	L/R	I	Left / right selection	Note 4,5
40	U/D	I	Up/down selection	Note 4,5
41	V_{GH}	P	Gate ON Voltage	
42	V_{GL}	P	Gate OFF Voltage	
43	AV_{DD}	P	Power for Analog Circuit	
44	RESET	I	Global reset pin.	Note 6
45	NC	-	No connection	
46	V_{COM}	I	Common Voltage	
47	DITHB	I	Dithering function	Note 7
48	GND	P	Power Ground	
49	NC	-	No connection	
50	NC	-	No connection	

Tabla 3.2 Descripción de los pines de la pantalla LCD

d. DISEÑO DEL CIRCUITO

La conexión entre el microcontrolador está hecha por un Bus de 25 cables directamente entre pin y pin.

Por lo que la siguiente Tabla es la descripción en comunicación.

TFT CONTROL PINS	
LCD_RS	HET1/28
LCD_WR	HET1/8
LCD_CS	HET1/23
LCD_RST	HET1/11
DB0	HET1/2
DB1	HET1/18
DB2	HET1/16
DB3	HET1/30
DB4	HET1/14
DB5	HET1/12
DB6	GIOA/5
DB7	GIOA/2
DB8	GIOA/1
DB9	GIOA/0
DB10	HET1/22
DB11	HET1/25
DB12	GIOB/3
DB13	HET1/29
DB14	GIOB/2
DB15	HET1/10

TOUCH PINS	
DCLOCK	GIOB/1
CS	GIOB/0
DIN	HET1/31
DOUT	HET2/10
IRQ	HET2/9

Hercules RM57L

Tabla 3.3 Comunicación entre los Pines del MCU A LCD.

e. DISPOSITIVOS SELECCIONADOS

i. Pantalla TFT TACTIL

Esquema 3.33 Pantalla LCD 7 Pulgadas

- Pantalla TFT LCD a colores 800XRGBX480
- Voltaje: 5V, tiene una compatibilidad de entradas digitales de 3.3V.
- Controlador Interno SSD1963, las librerías que eran necesarios para controlar la pantalla fueron modificadas a partir de unas ya existentes.
- De modo que se acondiciono estas librerías para el Microcontrolador Cortex-ARM R5F.

VII. DIAGRAMA DE BLOQUES DE BLUETOOTH

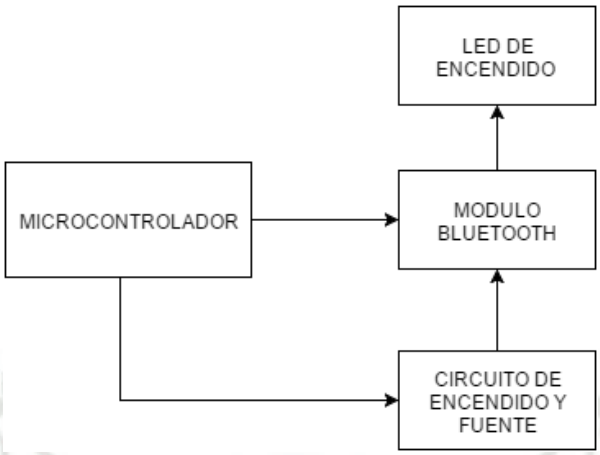
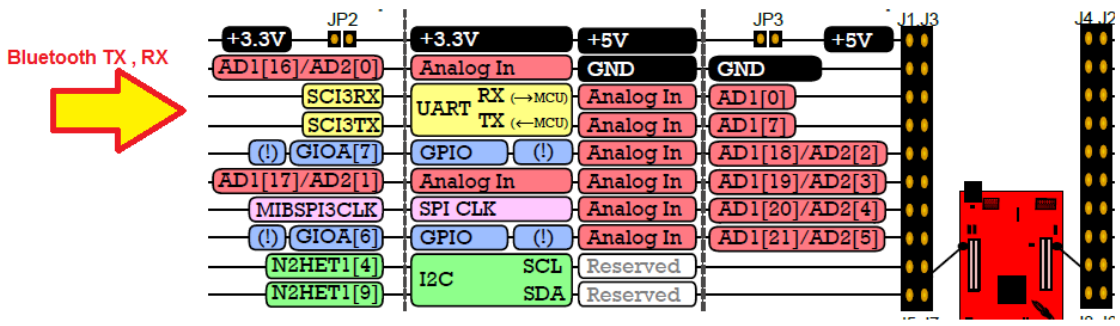


FIGURA 3.6 Diagrama de bloques de Bluetooth

a. FUNCIONAMIENTO

La función de comunicación es importante para los equipos de medición, por lo que se implementa un módulo Bluetooth con el propósito de enviar datos hacia un dispositivo móvil.

Se utiliza el tercer módulo de comunicación SCI del microcontrolador Hercules RM57, para entablar una comunicación con el dispositivo móvil a través del dispositivo Bluetooth.



Esquema 3.34 Modulo SCI Microcontrolador

El dispositivo Bluetooth HC-05, funciona con 3.3 V para sus señales de comunicación y el microcontrolador RM57, eléctricamente también envía los datos con ese mismo valor de tensión eléctrica, por lo que no es necesario un circuito acondicionador de señal para que los dispositivos trabajen correctamente.

i. CIRCUITO DE ENCENDIDO Y ALIMENTACION

Se utiliza un transistor de ***canal PNP ver Esquema 3.36-Control de fuente (Izquierdo),*** para alimentar el modulo Bluetooth, este no siempre se encuentra funcionando, por lo que es necesario tenerlo apagado y así ahorrar el consumo de corriente hacia la batería.

ii. LED DE ALERTA ENCENDIDO Y OPERACIÓN

El modulo Bluetooth tiene un Pin de salida que indica cuando este está correctamente conectado con el dispositivo receptor de datos.

Si el modulo no encuentra un dispositivo de recepción cercano este se quedara parpadeando con intervalos muy cortos de tiempo.

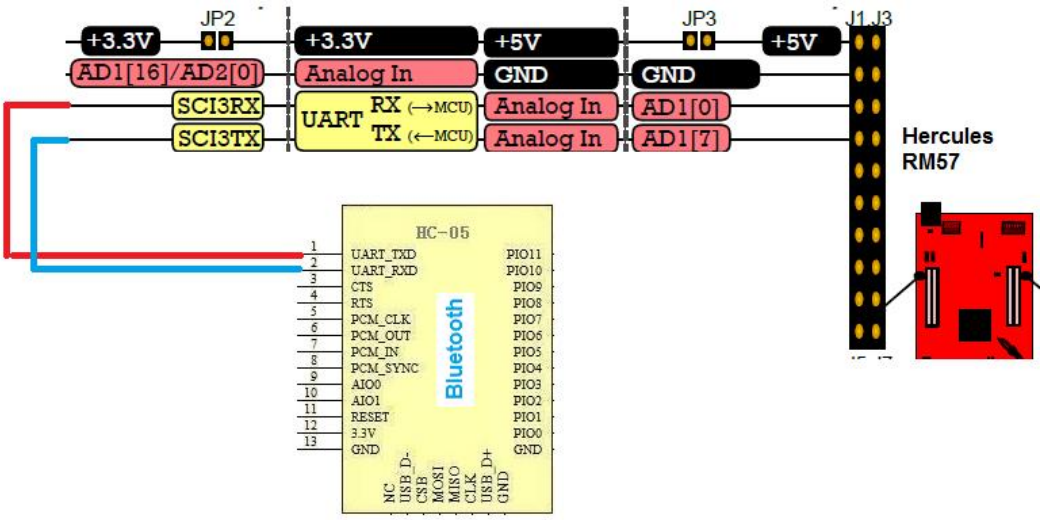
Para un funcionamiento simple, conectamos este Pin a un LED, color rojo para conocer si tuvo una conexión exitosa.

b. ESPECIFICACIONES TECNICAS

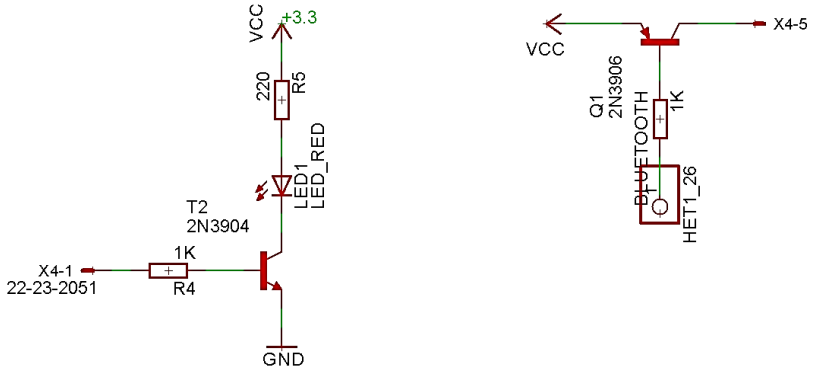
Voltaje de Alimentación: 3.3 V

Protocolo de comunicación: Serial / RS232

c. DISEÑO DEL CIRCUITO



Esquema 3.35 Conexión entre el Modulo HC-05 y el MCU



Esquema 3.36 Control de LED (Derecha), Control de fuente (Izquierda)

VIII. DIAGRAMA DE BLOQUES DE BATERIA Y FUENTE DE ALIMENTACION

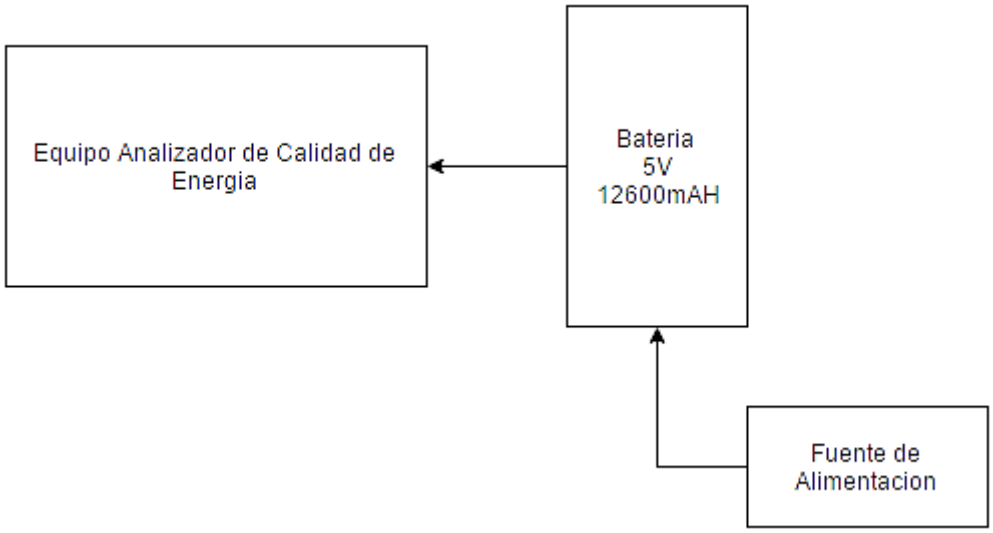


FIGURA 3.7 Diagrama de bloques de Fuente y Batería

a. FUNCIONAMIENTO

La batería externa ***POWER BANK ST-POWER199*** Solar de **12600mAh** es compatible con cualquier marca y modelo de dispositivos móviles, se puede utilizar en diferentes aplicaciones que requieran un alto consumo de corriente eléctrica, también cuenta con un indicador de carga conformado de 4 LEDS.

La batería también puede cargarse sin cables gracias a su panel solar incorporado.

La batería POWER BANK, puede utilizar cualquier tiene de fuente de alimentación para su recarga, esto permite la libertad de utilizar cargadores de celulares u otro tipo de fuentes de alimentación.

Para la recargada de la Batería es necesario una fuente de alimentación de 5Voltios y 2 A de corriente de Salida.

b. ESTIMACION DEL CONSUMO

El siguiente cálculo presentado es una estimación del consumo eléctrico según información de la hoja de datos de los dispositivos.

Dispositivo	Consumo en (mA)
Hercules RM57	**880**
LCD	**200**
BACKLIGHT LCD	**400*Dutty Cycle**
OTROS	**100**
TOTAL	**1180+400*Dutty Cycle**

Tabla 3.4 Tabla de Consumo en Corriente.

El consumo del BACKLIGHT LCD es proporcional al dutty cycle con el que se alimentala pantalla, este es del 30%, entonces la estimación del consumo de corriente en general es de 1300mAh, en un modo de operación donde el BACKLIGHT LCD, esta encendido en todo momento.

La corriente de consumo del microcontrolador puede variar de 880mA a 1 A, cuando este realiza operación matemáticas complejas como Fourier; considerando que el microcontrolador cuenta con una velocidad de 330MHz y esta operacion solo se realiza en los modos ANALISIS POTENCIA Y ARMONICOS, teniendo una duración entre microsegundos y milisegundos, para los cálculos se utiliza como base el valor de 800mA.

i. Duración de la batería en Alto consumo:

Ixeq: Corriente consumida por el equipo completo.

Ixbatt: Corriente entregada por la Batería

H: horas de duración de la batería.

(Ecuación 3.10)

$$H = \frac{Ixbatt}{Ixeq}$$

$$H = \frac{12600mAH}{1300mA}$$

$$H = 9.69\ horas$$

En un modo de alto consumo la duración de la batería es de 9 Horas y 41 Minutos.

ii. Duración de la batería en bajo consumo:

Para obtener una máxima duración de batería el equipo solo debe apagar el BACKLIGHT LCD, el que tiene un consumo de 400mA.

Ixeq=1300mA-400mA*0.3

Ixeq=1180mA

Se aplica la ecuación 3.10

$$H = \frac{12600mAH}{1180mA}$$

$$H = 10.67\ H$$

En un modo de bajo consumo la duración de la batería es de 10 Horas y 40 Minutos.

IX. DIAGRAMA DE BLOQUES DEL SISTEMA DE RELOJ Y MEMORIA

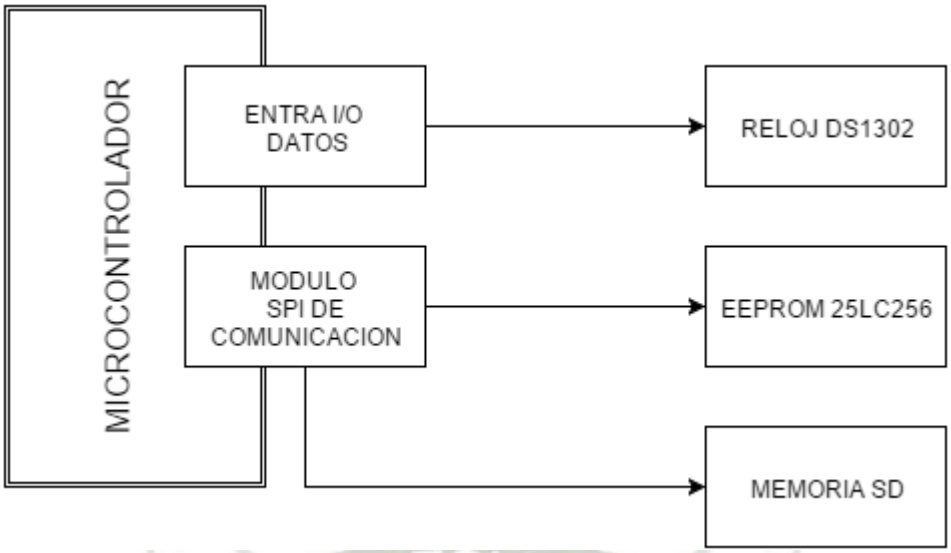


FIGURA 3.8 Diagrama de sistema de reloj y memoria

a. FUNCIONAMIENTO

i. RELOJ EN TIEMPO REAL

El reloj de tiempo real y calendario proporciona información sobre segundos, minutos, horas, día, fecha, mes y año.

El final del mes se debe ajustar automáticamente durante los meses con menos de 31 días, incluyendo las correcciones para los años bisiestos.

El reloj debe funcionar tanto en el formato de 24 horas o de 12 horas con un indicador AM / PM.

ii. Memoria de Configuraciones

La memoria de configuración es la que guarda las configuraciones como:

a. Módulos activados (Bluetooth)

b. Valores de Inicialización

c. Banderas de Inicialización

d. Configuraciones de LCD.

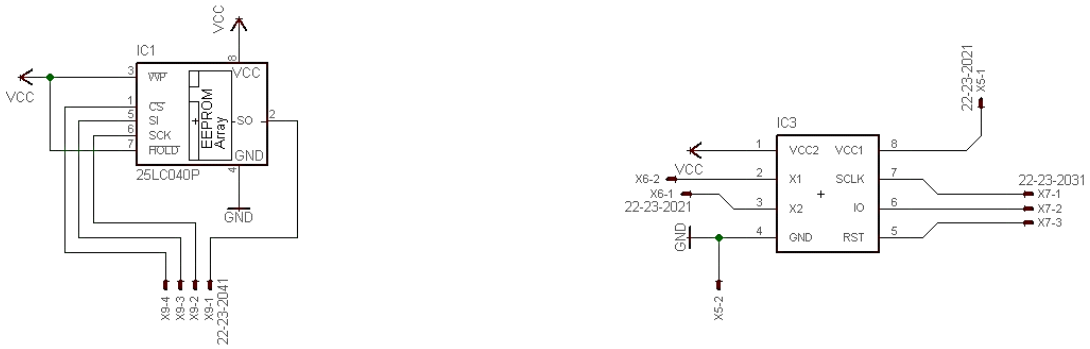
e. Resolución de conversión para Voltaje y Corriente.

iii. MEMORIA SD

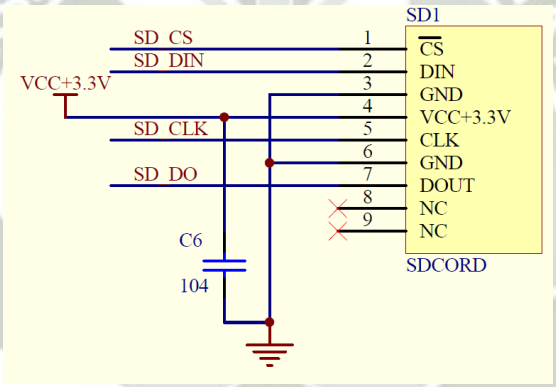
El equipo requiere de un dispositivo masivo de almacenamiento, donde se puedan almacenar información en archivos Excel por lo tanto es de importancia que el equipo cuente con una memoria de alta capacidad en la que se almacene formas de onda y eventos que suceden mientras se ejecutan ciertas tareas.

La memoria debe tener la capacidad de poder ser retirada para su lectura por computadora.

b. DISEÑO DEL CIRCUITO



Esquema 3.37 Memoria 25LC256 (Derecha), DS1302 (Izquierda)



Esquema 3.38 Interfaz de la Memoria SD y Microcontrolador.

c. DISPOSITIVOS SELECCIONADOS

i. RTC DS1302 RELOJ EN TIEMPO REAL

El Circuito RTC de cronometraje DS1302 contiene un reloj de tiempo real y calendario.

Características principales

- Gestiona todas las funciones clásicas de un reloj
- Reloj en tiempo real válido hasta 2100.
- Sencillo protocolo de comunicación para el microcontrolador.
- Sencillo interfaz de 3 cables.
- TTL-compatible (VCC = 5V)
- Operación entre 2,0 V a 5,5 V.
- Utiliza menos de 300 nA en 2.0V
- Rango de temperatura opcional Industrial: -40 ° C a + 85 ° C

Referencia:

https://www.maximintegrated.com/en/products/digital/real-time-clocks/DS1302.html

ii. EEPROM 25LC256 MEMORIA SERIAL

El Microchip Technology Inc. 25LC256 EEROM es una serie de memorias eléctricamente borrables de 256Kbit.

La memoria se accede a través de una sencilla interfaz periférica (SPI). Las señales de bus requeridos son una entrada de reloj (SCK), además de los datos por separado en (SI) y salida de datos (SO).

El acceso al dispositivo se controla a través de una entrada de Selección de Chip (CS). La comunicación con el dispositivo se puede detener mediante el pasador de mantenimiento (HOLD).

Especificiaciones Principales:

- Máx. Reloj 10 MHz.
- Bajo consumo de potencia: Escritura: 5 mA a 5.5 V, 10 MHz, Lectura: 6 mA a 5.5 V, 10MHz- Corriente espera: 1uA para 5,5 V.
- Organización 32.768 x 8 bits.
- 64-Byte página.
- Resistencia: 1.000.000 de borrado /escritura.
- Retención de datos ciclos: mayor a 200 años.

iii. MEMORIA SD

En el mercado se puede obtener una memoria SD de gran capacidad.

El equipo trabaja con formato FAT32, por lo tanto cualquier memoria implementada en el equipo no presentara problema en su correcto funcionamiento.

Las MMC o SD están actualmente disponibles en tamaños de hasta 100GB anunciados. Se usan en casi cualquier contexto donde se usen tarjetas de memoria, como teléfonos móviles, reproductores de audio digital, cámaras digitales y PDAs.

Desde la introducción de la tarjeta Secure Digital y la ranura SDIO (Secure Digital Input/Output), pocas compañías fabrican ranuras MMC en sus dispositivos, pero las MMCs, ligeramente más delgadas y de pines compatibles, pueden usarse en casi cualquier dispositivo que soporte tarjetas SD si lo hace su software/firmware.

CAPITULO 4

DISEÑO DE SOFTWARE

En este capítulo se presenta los esquemas generales de software, diagramas de flujo y ecuaciones que permiten una mejor comprensión del software que el microcontrolador ejecuta para realizar cada una de las tareas.

INTERFAZ GRAFICA

La familia Hercules RMx de la empresa Texas Instruments, cuenta con una interfaz llamada HalCoGen, que trabaja en conjunto con el IDE Code Composer Studio, ambas sirven para configurar y programar la MCU.

HalCoGen proporciona una interfaz gráfica al usuario que le permite configurar los periféricos internos del microcontrolador.

Una vez configurado el dispositivo, nos permite utilizar el Code Composer Studio para completar el código y realizar nuestra aplicación.

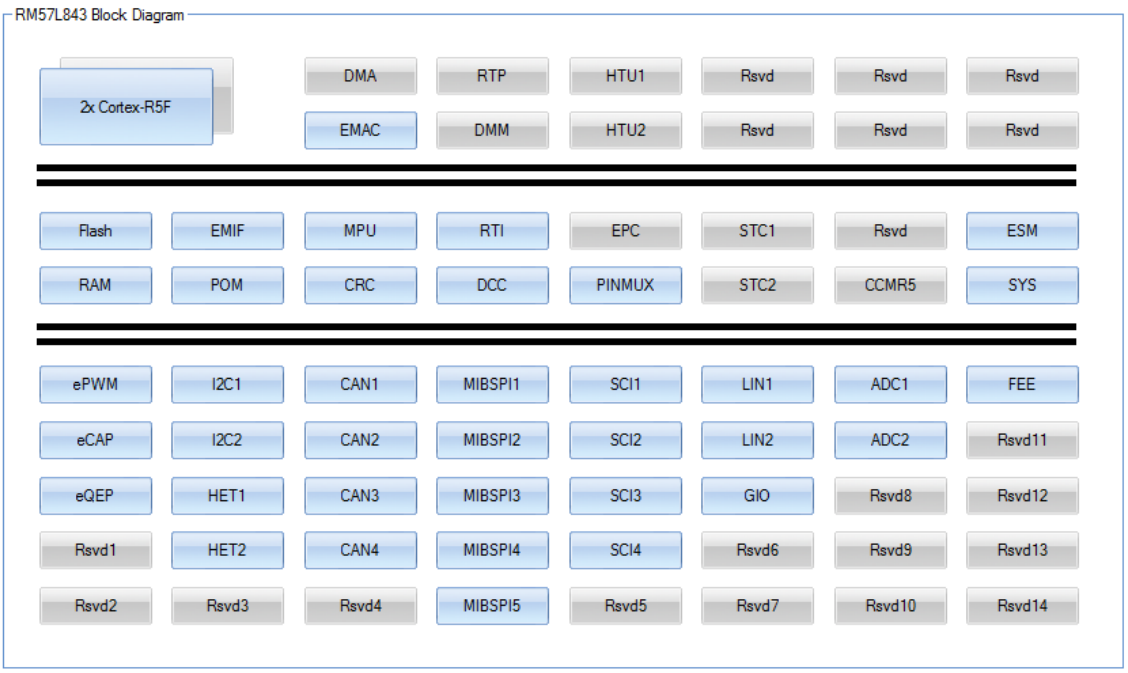


FIGURA 4.1 Diagrama de bloques especifico de la RM57L843

En la Esquema 4 podemos apreciar los diferentes módulos internos con los que el MCU cuenta; los de color celeste son los que están habilitados en el Chip, mientras que los de color gris son módulos reservados y controles de registros, transparentes para el usuario.

I. MODULO CONVERSOR ANALOGO DIGITAL (ADC)

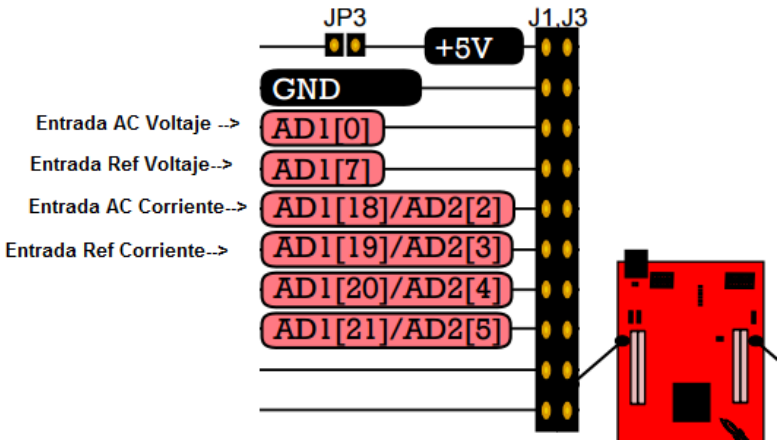
Como se puede en la figura 4.31, la MCU cuenta con dos módulos independientes, ADC1 y ADC2, esto nos permite hacer trabajar ambos de forma individual.

El modulo conversor análogo digital de la MCU, cuenta con una resolución de 12 bits la cual puede ser cambiada a 10bits y 8bits.

El tiempo que demora la MCU en comparar el nivel de voltaje y entregar el dato en su registro es de 0.56 microsegundos.

Este tiempo se suma a medida que se utiliza más canales del conversor.

El equipo trabaja con 4 canales Análogo digital, con la siguiente distribución:



Esquema 4.1 Distribuciones de Canales ADC

Pines Operativos:

AD1 [0]: Por este canal ingresa la señal de voltaje Alterno

AD1 [7]: Por este canal ingresa la señal de referencia solo Voltaje.

AD2 [2]: Por este canal ingresa la señal de corriente alterna.

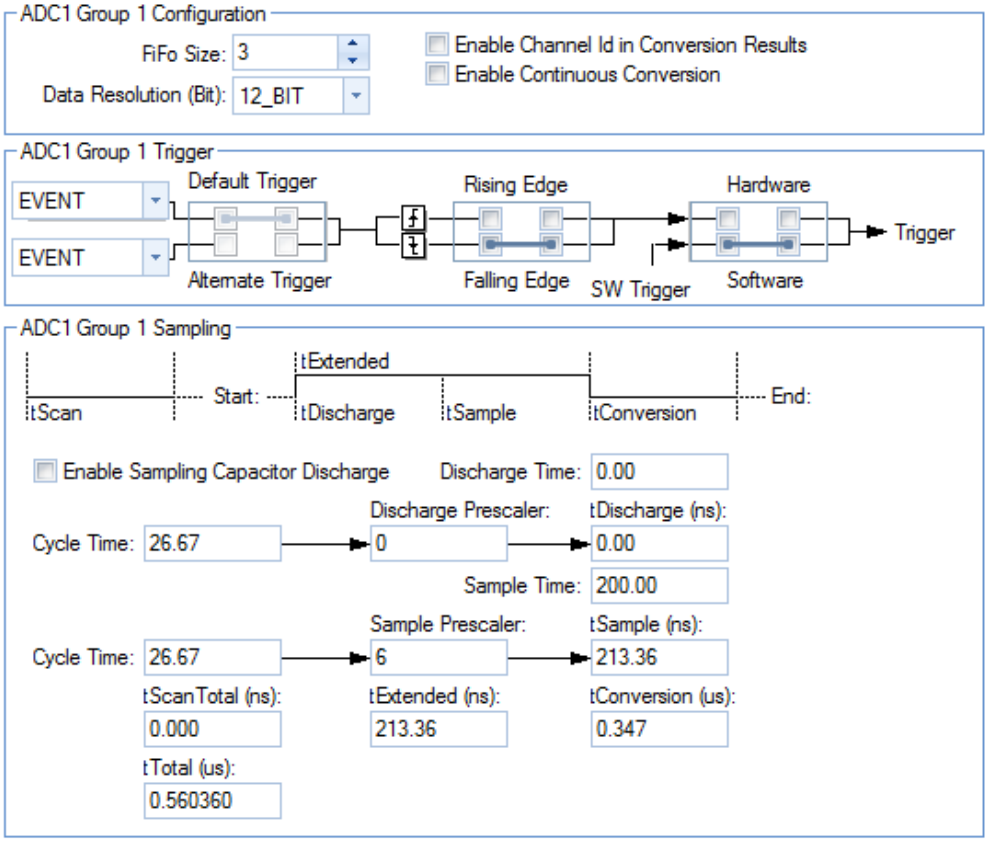
AD2 [3]: Por este canal ingresa la señal de referencia solo corriente.

El nivel de referencia, es el voltaje offset enviado por la configuración de los OPAMP. Cada una de las señales de voltaje y corriente tiene un voltaje offset diferente, por lo que es necesario obtener este dato durante el muestreo.

Esto nos permite tener una señal con valores positivos y negativos en la memoria del Microcontrolador.

Ademas esto ayuda eficientemente en el procesamiento digital de las señales.

Configuración del ADC1 y ADC2, son similares:



Esquema 4.2 Diagrama de bloques especifico de la RM57L843

Cada canal utiliza un tiempo total de tTotal= 0.560360 microsegundos.

Por lo tanto para Nchadc= 4 canales utilizan un tiempo total de:

$$Tx4ch = Nchadc * tTotal \quad \text{(Ecuación 4.1)}$$

$$Tx4ch = 4 * 0.560360\ us$$

$$Tx4ch = 2.24\ us$$

La conversión de los 4 canales ADC, tiene un tiempo de retardo de 2.24 microsegundos, lo que nos da una frecuencia de muestreo máxima de:

(Ecuación 4.2)

$$Fx4ch = \frac{1}{T4xch}$$

$$Fx4ch = \frac{1}{2.24us}$$

$$Fx4ch = \ 446.1417\ KHz$$

Nuestra Ancho de banda es hasta el armónico 59 de la frecuencia de 60Hz, por lo tanto seria de Bw= 3.540 KHz

Entonces:

$$Fx4ch >> Bw$$

Dado que se cumple la condición, el conversor ADC es perfecto para el muestro de la señal Alterna en nuestro equipo.

La resolución del conversor ADC es de 12 bits y un voltaje máximo de 3.3 Voltios, entonces se tiene un LSB de:

$$LSB = \frac{Vmax}{2^n}$$ Ecuación 4.3

$$LSB = \frac{3.3}{2^{12}}$$

$$LSB = 805.66uV$$

Con este último dato y conociendo la atenuación dada por la red resistiva para el muestreo de voltaje podemos calcular la sensibilidad del equipo ante los cambios de voltaje y corriente.

a. Sensibilidad del voltaje ($Vacxsbd$)

Para el desarrollo de esta etapa se requiere un dato del:

Capítulo 3, en la ecuación 8.

Atenuación por resistencias (AtxR) = 1.049e-3

$$Vacxsbd = \frac{1}{\text{AtxR}} * LSB \qquad \text{Ecuación 4.4}$$

$$Vacxsbd = \frac{1}{1.049e-3} * 805.66uV$$

$$Vacxsbd = 0.76802\ V$$

El equipo puede capturar variaciones de voltaje mínimas de 0.76 Voltios.

b. Sensibilidad de la Corriente ($Aacxsbd$)

Atenuación por pinza Amperimetrica (Atxclmp) = 1000

$$Aacxsbd = \text{Atxclmp} * LSB \qquad \text{Ecuación 4.5}$$

$$Aacxsbd = 1000 * 805.66uV$$

$$Aacxsbd = 0.80566\ A$$

El equipo puede capturar variaciones de Corriente mínimas de 0.805 Amperios.

Como se puede observar el módulo ADC es la parte principal del equipo, debido a que la mayor parte de la información para procesar se obtiene de esta interfaz.

II. MODULO TEMPORIZADOR DE ALTA PRECISION (NHET) Y GPIO (GIO)

El módulo NHET es un módulo temporizador avanzado inteligente, que proporciona funciones de sincronización para aplicaciones en tiempo real.

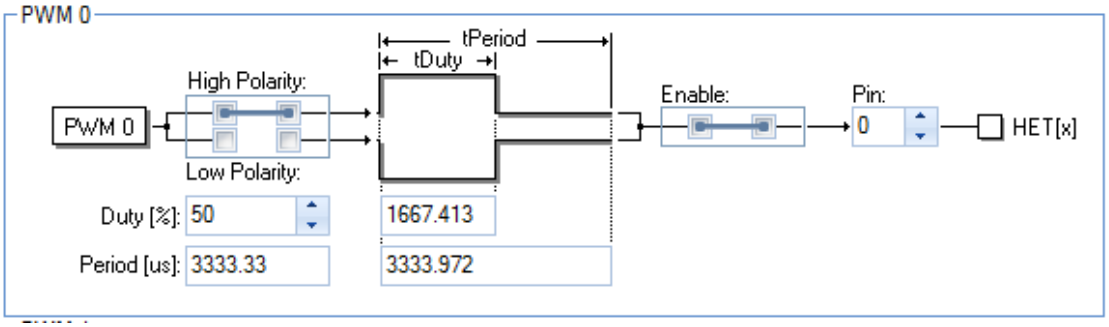
Sus canales de Hardware de alta resolución le permiten una mayor precisión para las función de medición de pulsos, comparación y señales PWM también puede configurarse como entrada y salida de digital.

El módulo GIO tiene el propósito de funcionar como entrada y salida digital, con la diferencia de que este está compuesto por un registro de 8 bits y sus interrupciones son más sencillas de definir en la programación.

a. MODULO DE TEMPORIZADOR DE ALTA PRECISION

1. Generador de PWM

El módulo PWM cuenta con una resolución de microsegundos, se puede seleccionar la polaridad de este también, como se puede ver en la imagen.



Esquema 4.3 Diagrama de configuración PWM

La aplicación del PWM en el equipo es el control del brillo de la pantalla LCD (BACKLIGHT LCD) las características de su trabajo se definen en:

"CAPITULO 3, DIAGRAMA DE BLOQUES DEL CIRCUITO DE VISUALIZACION Y ENTRADA, SELECION Y CALCULOS"

Aquí es donde se define que es necesario una señal cuadrada con una frecuencia de 300Hz con un Dutty Cycle variable.

(Ecuación 4.6)

$$Period = \frac{1}{300Hz}$$

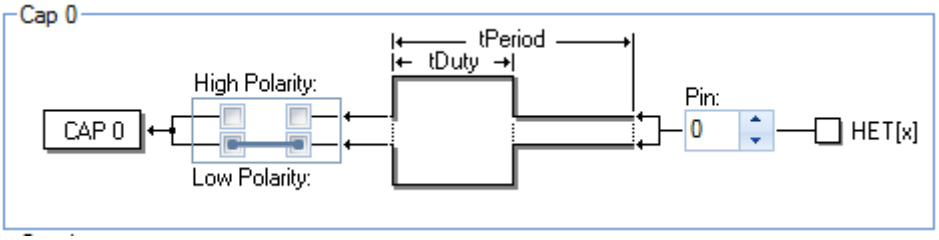
$$Period = 3333.33\ us$$

2. Modulo Captura de ancho de pulso

El modulo temporizar, está especializado para medir anchos de pulso, dicho temporizador inicia su conteo cuando se detecta un flanco de subida o de bajada según este programado, para luego finalizar su conteo una vez reciba un nuevo flanco.

Carga el buffer de datos con el valor equivalente en decimal, este valor es equivalente en microsegundos a su equivalente, es decir:

Registro (valor decimal) = 16736, es equivalente a 16736 microsegundos.



Esquema 4.4 Diagrama de configuración Capturado Temporal

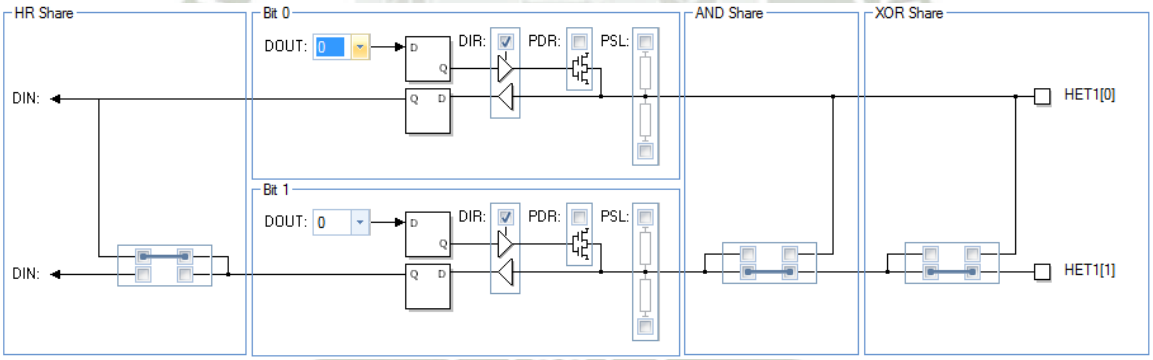
3. Entrada y Salida Digital

El MCU RM57, puede multiplexar su módulo NHET y configurarlos como salida y entradas de datos digitales.

El microcontrolador cuenta con 64 pines I/O en total, todos pueden configurarse para funcionar como salidas de PWM, temporizadores o contadores de pulsos.

Dentro de sus tareas también cuenta con compuertas lógicas en sus salidas y entradas como la AND y XOR, que se pueden combinar entre sí.

En el equipo este módulo I/O es la base de control para la pantalla TFT LCD, ya que se utiliza 25 pines para el control de la pantalla.



Esquema 4.5 Diagrama de configuración como I/O Digital

III. MODULO DE INTERRUPCION EN TIEMPO REAL(RTI)

Las interrupciones en tiempo real es un módulo que proporciona funciones de temporización en el sistema.

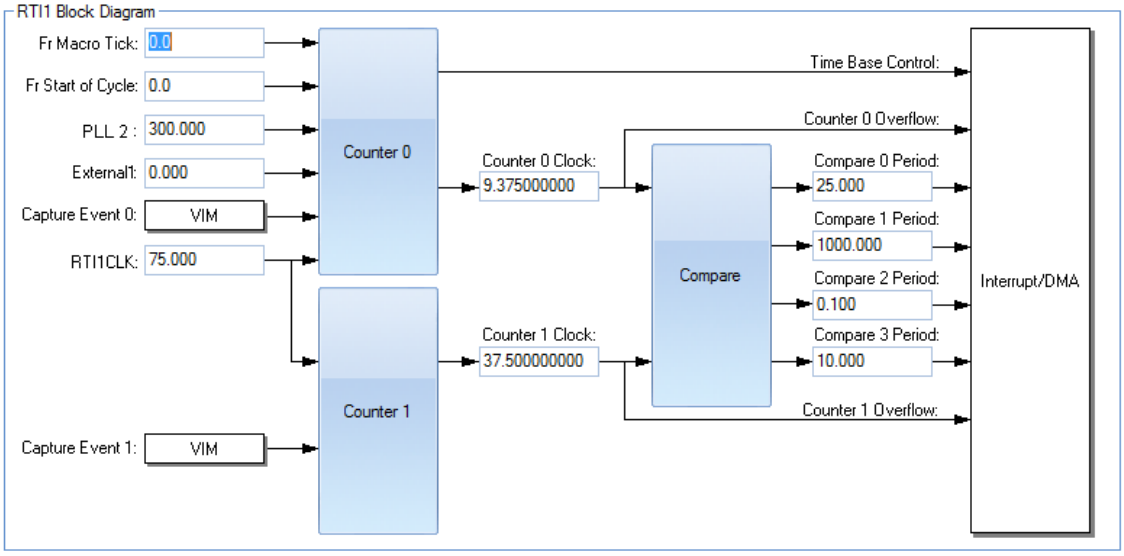
El módulo RTI está compuesto por varios contadores en los que se puede configurar sus bases de tiempo según sea la necesidad.

a. DIAGRAMA DEL MODULO RTI GENERAL

En esta vista general del módulo RTI, se puede observar que está compuesto por dos submódulos, dos contadores y un comprador.

Para definir la frecuencia en la que debe operar el modulo se hace a través del registros RTI1CLK el cual está configurado a 75MHz.

Una vez que la temporización culmina se activa una bandera de interrupción y ejecuta la tarea que fue programada.



Esquema 4.6 Diagrama bloques RTI General

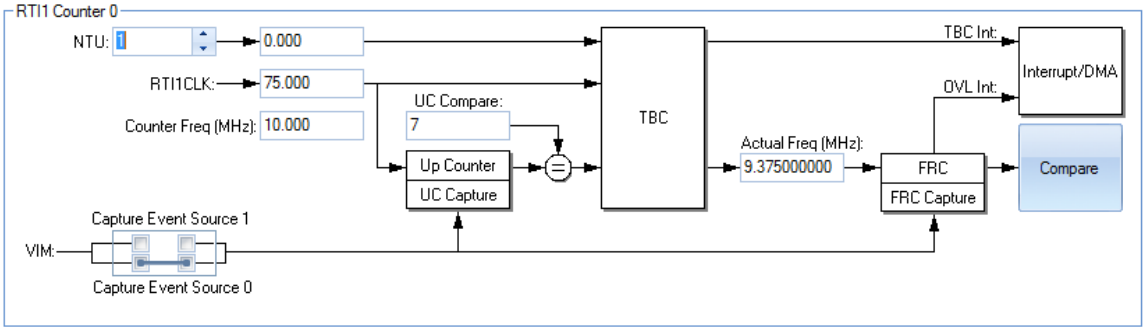
b. DIAGRAMA DE CONTADORES

Este módulo nos permite configurar la frecuencia base para el tiempo de conteo, esto es un equivalente al Prescaler en otras familias de microcontroladores.

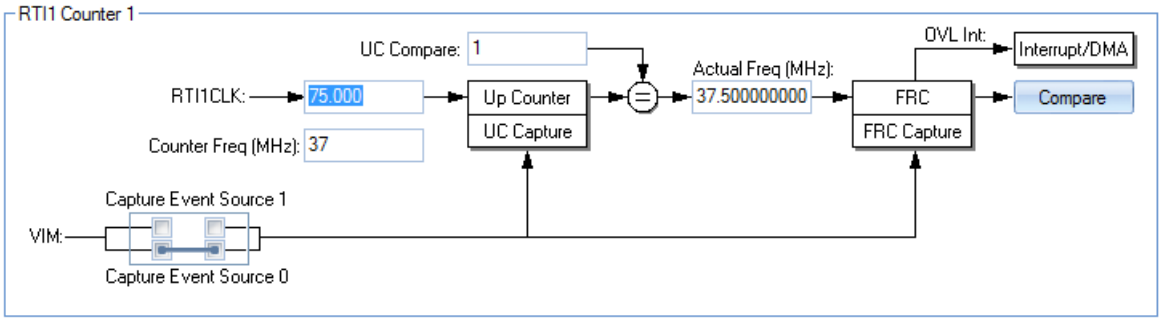
EL módulo RTI cuenta con dos módulos que trabajan independiente del otro, de modo que uno se puede configurar con una frecuencia de conteo base y el otro con un valor totalmente diferente sin que afecten su trabajo entre ellos.

Esto es una gran ventaja por el hecho de que se puede configurar un contador para temporizaciones largas, mientras que podemos configurar otro contador para temporizaciones muy cortas.

Aprovechándonos de esta característica del módulo para nuestra aplicación, se configuro los contadores con diferentes características:



Esquema 4.7 Diagrama bloques Contador 1



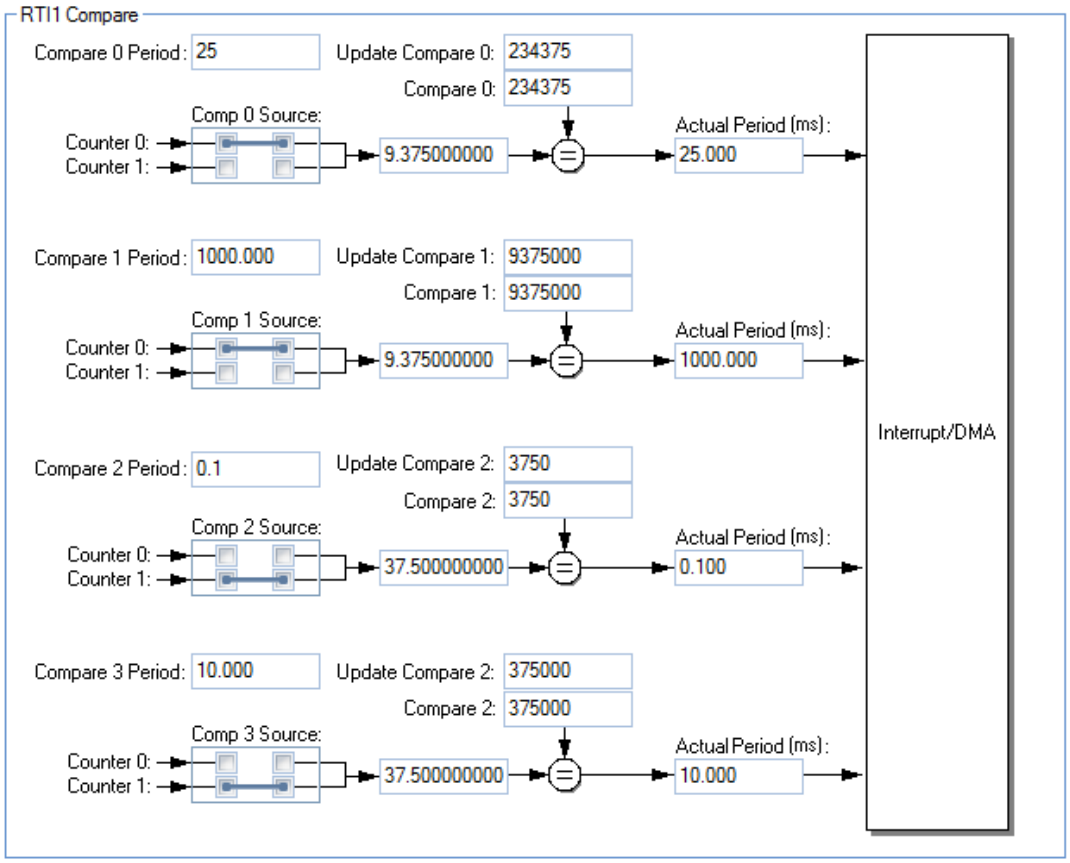
Esquema 4.8 Diagrama bloques Contador 2

	Contador 0	Contador 1
Actual Freq (MHz)	9.375	37.5

Tabla 4.1 Frecuencia de Contadores

c. DIAGRAMA DE COMPARADORES

En esta etapa se explicara el motivo del valor para cada uno de los valores configurados en los comparadores.



Esquema 4.9 Diagrama Bloques Comparador

Comparador	**Periodo (ms)**	**Tarea que ejecuta**
0	25	Tiene como tarea medir el ancho de pulso de la frecuencia del voltaje AC y cargarla en un registro de lectura general, donde otras funciones pueden hacer uso de este.
1	1000	Sirve para refrescar la pantalla LCD TFT. Se aplica un arreglo numérico para multiplicar este valor por 2 o 5 y obtener temporizaciones mas largas.
2	0.1	Sirve como temporizador para capturar el ADC, pero su valor puede ser cambiado por las diferentes funciones que el equipo ejecuta.
3	10	Se encarga de ejecutar un temporizador para mantener la comunicación con la memoria SD.

Tabla 4.2 Configuración de comparadores

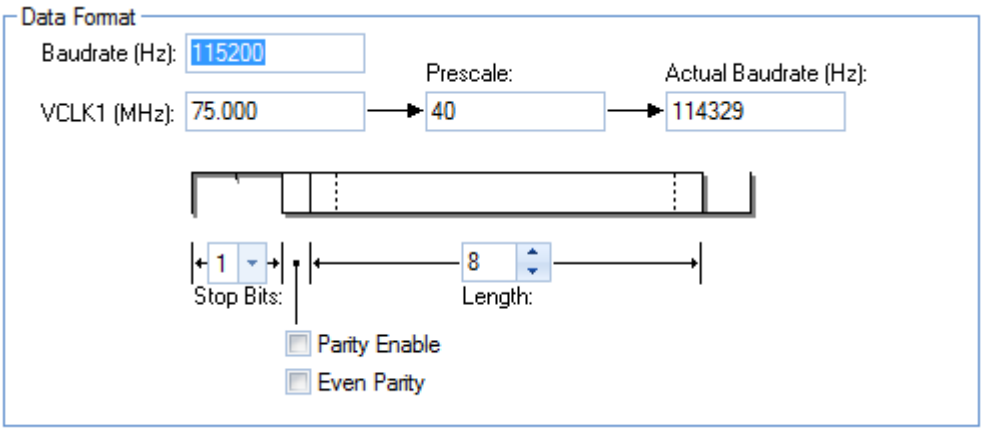
IV. MODULO INTERFAZ DE COMUNICACIÓN SERIAL (SCI)

El sistema hace uso de dos módulos de comunicación serial en el equipo.

Modulo	Tarea	Baudrate (Hz)	Otros
SCI 1	Comunicación con la PC y administración de memoria SD.	115200	1 Bit Stop Sin Paridad
SCI 3	Comunicación con el Modulo Bluetooth.	9600	1bit Stop Sin paridad

Tabla 4.3 Tarea de los módulos SCI

Uno de ellos está configurado para comunicarse con la Hyperterminal de la PC y ejecuta un programa donde se puede administrar los datos que se encuentran en la memoria SD, mientras que el otro envía y recibe información enviada por el módulo Bluetooh.



Esquema 4.10 Diagrama Bloques SCI 1

V. MODULO INTERFAZ SERIAL PERIFERICOS (SPI)

El equipo hace uso de 3 módulos SPI a continuación se describe cada uno.

Módulo SPI	Baudrate(kHz)	Módulos Conectados
SPI 1	300	Configurado para el control de la memoria SD.
SPI 3	500	Configurado para el control de una memoria EEPROM (25LC256), la cual almacena la configuración general del equipo.
SPI Touch	125	Configurado para comunicarse con el modulo que controla la pantalla Touch. ADS7843 Touch Screen Controller.

Tabla 4.4 Asignación de tareas por módulo SPI.

METODOS Y ANALISIS MATEMATICOS

I. APLICACION DEL TEOREMA DE NYQUIST PARA EL PERIODO DE MUESTREO CON LA RTI

Se considera capturar 5 periodos de la señal alterna con el fin de mostrar en la pantalla LCD la similitud entre diferentes formas de onda, para esto se realiza los siguientes cálculos:

$$F = 60Hz$$

$$T = \frac{1}{f}$$

$$T = 16.67ms$$

Si se desea capturar 5 periodos, la longitud total del tiempo muestreado será:

$$Tt = 5 * T = 5 * 16.67ms$$

$$Tt = 83.35ms$$

Para mostrar los 5 periodos en la pantalla LCD, nosotros debemos considerar la longitud horizontal en pixeles, esta es de 800 pixeles, por lo tanto se capturara 800 valores del ADC.

Ahora la configuración del número de muestras esta predeterminada con un valor de N=800.

Para la selección del periodo de muestreo se debe considerar que se ara captura de 800 valores del ADC en un tiempo de 83.35ms, con esta premisa se puede calcular el valor del periodo de muestreo Ts:

$$Ts = \frac{Tt}{N}$$

$$Ts = \frac{83.35ms}{800}$$

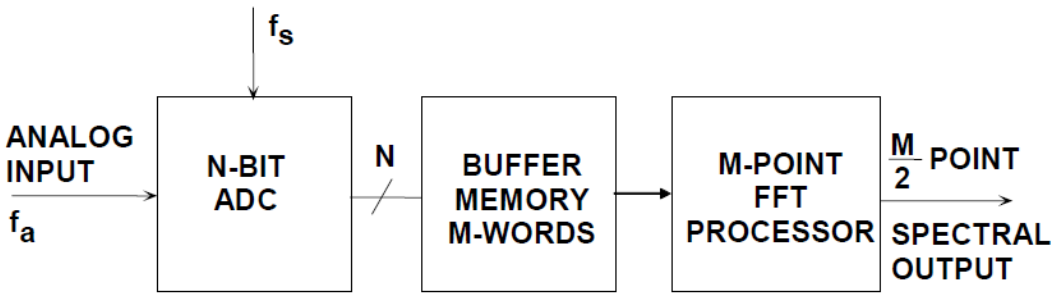
$$Ts = 104.18\ us$$

La velocidad mínima previamente calculada es de $Tx4ch = 2.24\ us$, por lo tanto el periodo de muestreo cumple con la siguiente condición:

$$Ts >> Bw$$

II. APLICACION DEL ANALISIS DE FOURIER PARA DETERMINACION DE MAGNITUD Y FASES

Existe diferentes maneras de cuantificar la distorsión y el ruido en una señal, todas ellas están basadas en el análisis de Fourier.



Esquema 4.11 Diagrama del algoritmo de Fourier

[Analog Devices MT-003 TUTORIAL]

a. CALCULO DEL PERIODO DE MUESTREO

Dado que para la aplicación se utilizan librerías DSP para la familia de procesadores Cortex, estos tienen una normatividad en cuanto a la longitud de datos que se deben ingresar en el algoritmo de Fourier.

"Internamente, la función utiliza una decimación radix-4 en el algoritmo de frecuencia (DIF) y el tamaño de la FFT se apoya de las longitudes [16, 64, 256, 1024]."

Por lo tanto los algoritmos deben operar con el doble de muestras de la longitud de la FFT, esto se interpreta de la siguiente manera:

Si yo deseo un FFT con una longitud de 1024 valores, debo capturar 2048 valores con el conversor Análogo digital y almacenarlos en la memoria.

Datos

N: $Numero\ de\ muestras\ en\ la\ señal\ v(t)$

Δf: $Resolución\ de\ frecuencia$

Ts: $Periodo\ de\ Muestreo$

T: $longitud\ total\ del\ registro\ de\ la\ muestra$

Se seleccionó una resolución de frecuencia Δf =10Hz, para que la muestra discreta sea siempre entera, también se escogió capturar 2048 valores del conversor ADC, para obtener una resolución del espectro con una longitud de 1024 valores.

Calculo:

N=2048

Δf =10Hz

$Ts = ?$

Ecuacion 4.7

$$\Delta f = \frac{1}{T}$$

$$T = Ts * N$$

$$\Delta f = \frac{1}{Ts * N}$$

$$10Hz = \frac{1}{Ts(2048)}$$

$$Ts = 48.828us$$

b. METODO DE ANALISIS DE SOLO ARMONICOS

Inicialmente se planteó el cálculo hasta el armónico 51, para una frecuencia de Lina de 60Hz, el ancho de banda requerido seria

$$BW = 51(60) = 3060\ Hz$$

Previamente se planteó un dato importante para esta etapa es la resolución de la frecuencia $\Delta f = 10Hz$.

Como el análisis de Fourier es discreto, el dato es almacenado de la siguiente manera:

Vectores en memoria RAM	Definición
Vector_Magnitud	Es un vector que contiene todo el espectro de la señal análoga, está compuesto por 1024 posiciones de memoria con precisión de punto flotante.
Posicion_vector	es el índice de los elementos dentro del vector magnitud

Tabla 4.5 Definición de vectores / Armónicos.

En esta etapa trabajaremos con estos datos.

El objetivo con el análisis de Fourier es encontrar los armónicos de la señal muestreada y obtener información como magnitud, fase del espectro

Se toma como ejemplo la siguiente ecuación, transformada discreta de una señal cuadrada:

Ecuación 4.8

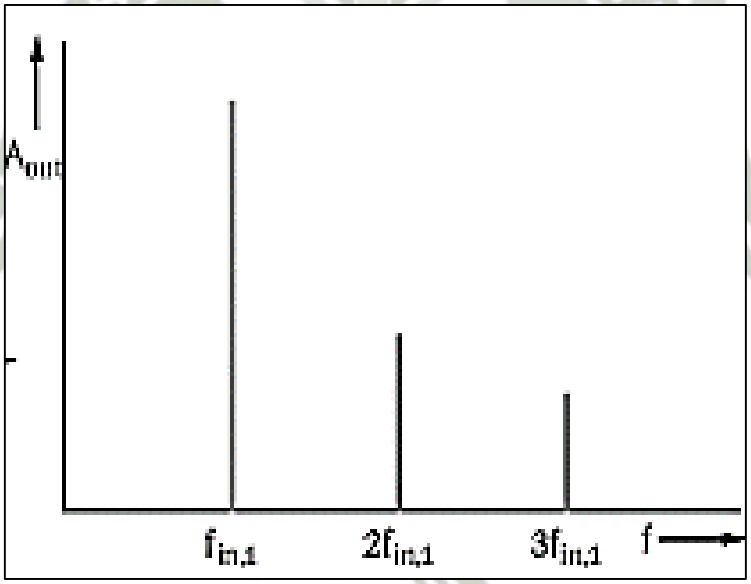
$$f(t) = a_0 + \sum_{n=1}^{\infty} [a_n \cos(n\omega_0 t) + b_n \operatorname{sen}(n\omega_0 t)]$$

Dado que se trabaja con un espectro discreto este se almacena en un vector de valores, donde se debe extraer la información de solo los armónicos.

Para ello concentramos el análisis en la sumatoria, la sumatoria tiene un inicio y final, el inicio está indicado por la variable ***n***, que solo pueden ser números enteros.

Mientras que en este caso el valor límite debe ser computacional, no puede ser infinito, por lo tanto también debe ser finito.

En el espectro de frecuencias esto se entiende de la siguiente manera.



Esquema 4.12 Espectro de solo armónicos

"Los armónicos dentro del vector son múltiplos enteros de la frecuencia fundamental"

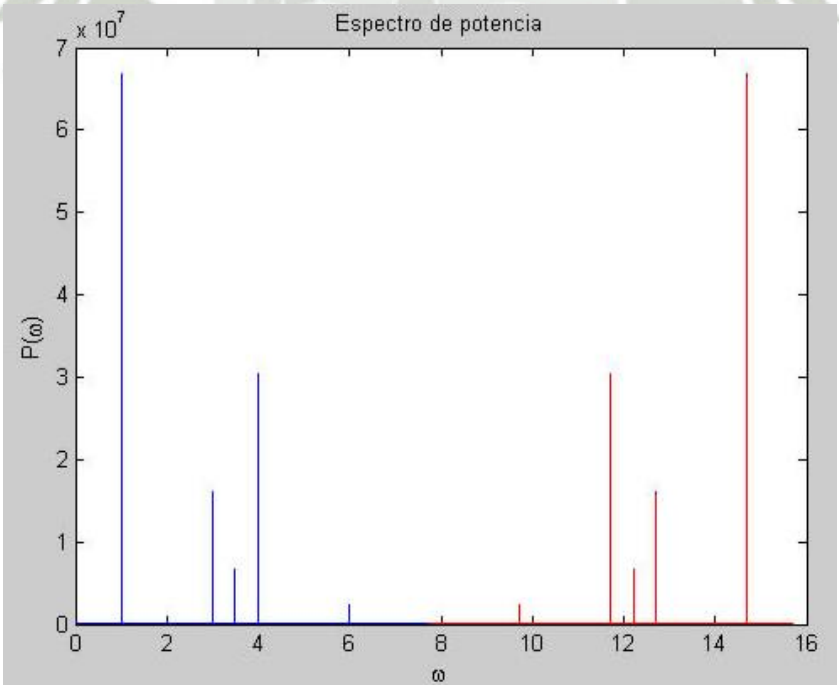
En el ***Vector_Magnitud***, se tiene la información de todo el espectro en frecuencia, por lo tanto es necesario filtrar la información de los armónicos, para esto se efectúa el siguiente calculo.

Previamente se definió la resolución de frecuencia como:

$$\Delta f = 10Hz$$

El ***Vector_Magnitud*** tiene 1024 posiciones en los que almacena el espectro completo de la señal análoga.

Debido al método ***"radix-4 decimation",*** que utiliza el algoritmo de Fourier, esto genera que la mitad de los datos sean un reflejo de su otra mitad.



Esquema 4.13 FFT usando Radix 4 Decimación

De este modo dentro del ***Vector_Magnitud,*** solo se debe trabajar con 512 valores, de aquí calculamos el ancho de banda total del espectro de Fourier.

$$BWs = 512 * \Delta f \qquad \text{Ecuación 4.8}$$

$$Bws = 512 * 10Hz$$

$$BW = 5120Hz$$

El ancho de banda del espectro de Fourier es de 5120 Hz.

El número de armónicos que se pueden mostrar dentro de este ancho de banda seria:

Se define:

Narmnic= Numero de armónicos en el espectro completo.

Fline = Frecuencia base con la que se desea trabajar

BW = Ancho de banda del espectro completo.

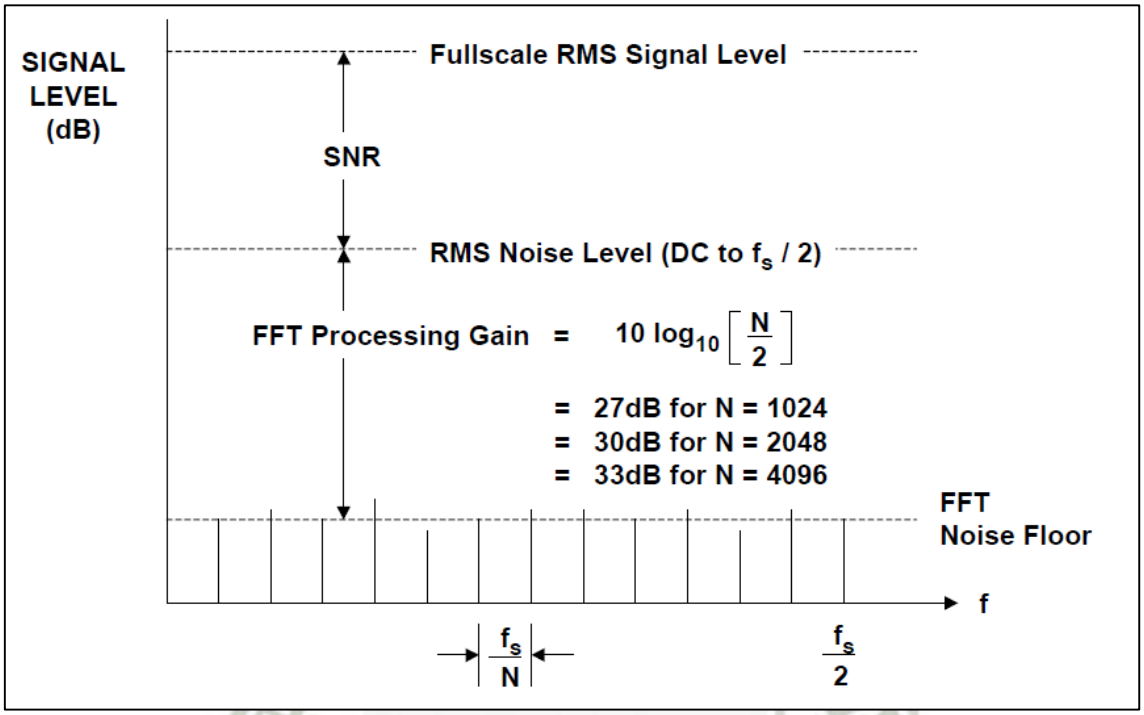
$$Narmnic = \frac{BW}{Fline}$$

$$Narmnic = \frac{5120}{60}$$

$$Narmnic = 85.33$$

Narmnic "debe ser un número entero por el hecho de que trabajamos con muestras discretas enteras"

Dentro de este espectro se puede analizar hasta el armónico 85.



Esquema 4.14 Grafica de ADC y Ruido

En la Esquema 4.14 se puede extraer que el Piso de Ruido es la suma de ganancia de procesamiento más el SNR.

Según la ecuación:

N: Numero de muestras en el espacio de tiempo

$FFTpg$: Ganancia de Procesamiento FFT.

Se definió anteriormente el valor de N=2048

$$FFTpg = 10\log(\frac{N}{2})$$

$$FFTpg = 10\log(\frac{2048}{2})$$

$$FFTpg = 30.1dB$$

Observe que la ganancia de procesamiento FFT se determina por el número de N de la FFT.

La FFT actúa como un analizador de espectro analógico con un ancho de banda acotado desde 0 hasta $\frac{fs}{N}$

.

"Aumentar el número de N, aumenta la resolución FFT y se estrecha su ancho de banda, lo que reduce el ruido de fondo"

Dado que se trabaja con un algoritmo de Fourier con un valor de N limitado máximo de 2048 valores, obteniendo una ganancia de procesamiento de 30.1dB.

Los armónicos de mayor frecuencia tienden a tener valores de potencia muy bajos.

Con una relación de 30.1dB entre la potencia de armónicos cercanos a $\frac{fs}{N}$ solo estaríamos analizando y aplicando a nuestros cálculos la potencia obtenida de solo el ruido de fondo.

Por lo tanto se debe acotar el número de armónicos con los que se va a trabajar.

En las especificaciones del equipo se planteó 51 armónicos, pero dada las dimensiones de la pantalla LCD de 800 pixeles de longitud.

Se vio mejorar el número de armónicos que se mostraran hasta el Numero 59.

c. FASE DEL ESPECTRO

Una forma de onda periódica no sinusoidal que cumple determinadas condiciones puede describirse mediante una serie de Fourier de señales sinusoidales. ***(Referencia Hart Daniel W. Electronica de potencia Cap 2-8)***

La serie de Fourier presenta senos y cosenos de la misma frecuencia que pueden combinarse en una misma sinusoide, dando como resultado una expresión alternativa para la serie de Fourier (Ecuación 4.8)

$$f(t) = a_0 + \sum_{n=1}^{\infty} C_n \cos(n\omega_0 t + \theta_n)$$

Ecuación 4.9

Donde:

$$C_n = \sqrt{a_n^2 + b_n^2} \quad \text{y} \quad \theta_n = \tan^{-1}\left(\frac{-b_n}{a_n}\right)$$

Ecuaciones 4.10

La transformada discreta o DFT, sigue el mismo principio de funcionamiento que en las ecuaciones 4.9 y 4.10.

Cuando se aplica la transformada discreta de Fourier, el algoritmo nos devuelve dos vectores.

El primer vector de memoria llena de números reales y el segundo con los números imaginarios.

Se podría definir de la siguiente Manera:

An= Primer Vector de números reales.

Bn= Segundo vector de números imaginarios.

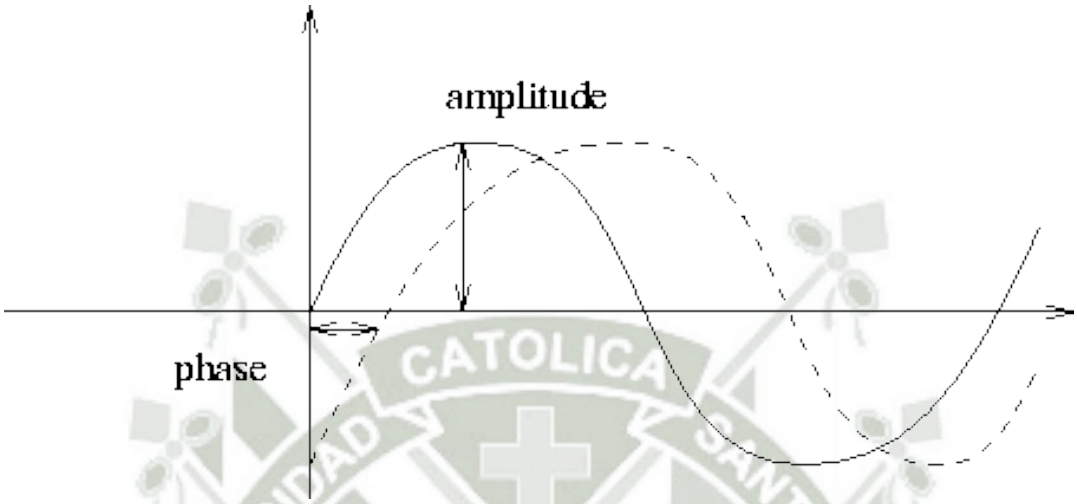
Por lo que para graficar el espectro en frecuencia solo se debe reemplazar los valores de dichos vectores en la ecuación 4.10.

El análisis de esta etapa solo se basa en la fase del espectro de Fourier.

$$\theta_n = \tan^{-1}\left(\frac{-b_n}{a_n}\right)$$

Ecuación 4.11

El objetivo es aplicar los conceptos teóricos previamente explicados para encontrar el angulo de fase de la señal de voltaje, luego el de la corriente, restar dichos ángulos me permite encontrar el Angulo de desfase entre dos señales.



Esquema 4.15 Desfase entre dos señales Alternas

III. CONVOLUCION EN EL DOMINIO DE FRECUENCIA

a. PRINCIPIOS MATEMATICOS

Esta operación se produce de forma inevitable en el dominio tiempo, cuando tenemos un producto de espectros en el dominio de frecuencia o viceversa.

Por eso, llegamos a este punto donde es necesario definir la operación de convolucion y su relación con Fourier.

Se puede definir la convolucion como el producto de dos funciones temporales (Brigham, 1988)

$$h(t) = f_1(t) * f_2(t) = \int_{-\infty}^{\infty} f_1(\tau) f_2(t-\tau) d\tau$$

Ecuación 4.12

En la ecuación 4.12, se define la convolucion en el dominio tiempo, pero esta operación también puede llevarse a cabo en el dominio frecuencia mediante la transformada de Fourier. (Bath, 1974)

$$\begin{matrix} f_1(t) \leftrightarrow F_1(\omega) \\ f_2(t) \leftrightarrow F_2(\omega) \end{matrix} \Rightarrow f_1(t) * f_2(t) \leftrightarrow F_1(\omega) F_2(\omega)$$

$$F_1(\omega) F_2(\omega) = H(\omega) \leftrightarrow h(t)$$

Ecuación 4.13

Se define H(w), como el espectro de Fourier de la función h(t) de la ecuación 4.12.

Entonces si quiero conocer la convolucion de dos funciones, en lugar de resolver la integral correspondiente, es más rápido obtener los espectros de Fourier de cada una de las señales y luego aplicar una multiplicación simple.

Análogamente hubiéramos podido definir la convolucion en el dominio frecuencia, pues esta propiedad es igualmente cierta para dicha operación (Bath, 1974; Brigham, 1988).

Así, notamos que la transformada de Fourier es una herramienta muy útil, para realizar multitud de operaciones habituales en el análisis de datos.

b. ESPECTRO DE POTENCIA

Como se definió en la ecuación 4.13, realizar una multiplicación en el dominio de la frecuencia es equivalente a una convolucion en el dominio de tiempo.

Desde el punto de vista del Microcontrolador es mucho más sencillo realizar una multiplicación que desarrollar una integral de valores discretos.

A continuación se define el consumo de memoria RAM, de una convolucion en el dominio tiempo y en el dominio frecuencia.

1. Premisas

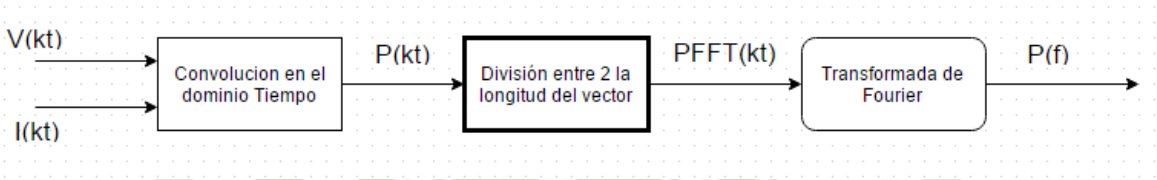
Algoritmo Convolucion: Recibe dos vectores de datos discretos con una longitud N y devuelve el resultado en un vector de longitud 2N.

Algoritmo de Fourier: Recibe un vector en el dominio de tiempo con una longitud N y devuelve un vector de longitud N/2.

Señales de Entrada: Se define como V(kt), I(kt), como la señales discretas muestreadas en el dominio tiempo, ambas tienen una longitud N = 2048, Referencia ***"APLICACION DEL ANALISIS DE FOURIER PARA DETERMINACION DE MAGNITUD Y FASES - METODOS Y ANALISIS MATEMATICOS".***

2. Espectro de Potencia por convolucion en el Dominio Tiempo

En el siguiente diagrama se puede observar el flujo de operaciones que se deben realizar para obtener el espectro de frecuencia de la potencia si se realiza la operación a través del dominio tiempo.



Esquema 4.16 Convolucion en el dominio Tiempo.

Función	Memoria	Descripción
V(kt)	2048	Almacena la señal alterna de voltaje.
I(Kt)	2048	Almacena la señal alterna de la corriente
P(kt)	4096	Almacena el resultado de la convolucion en el tiempo
PFFT(kt)	2048	Almacena la mitad de valores del vector P(kt)
P(f)	1024	Almacena el espectro de

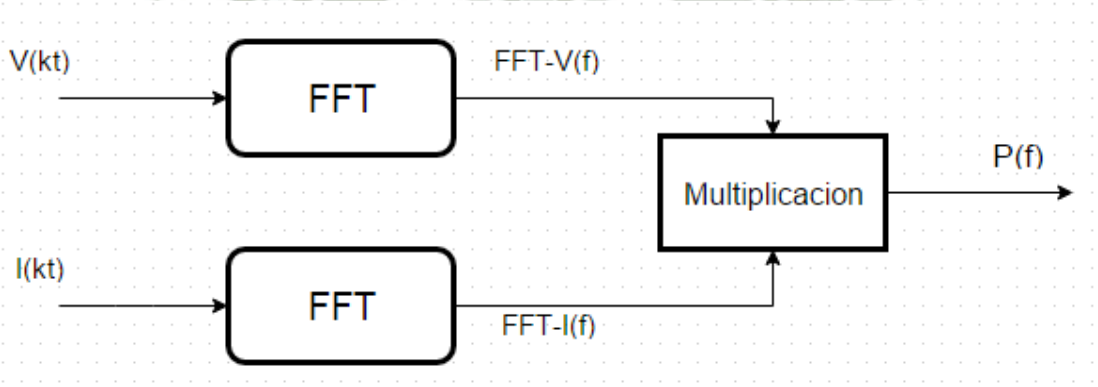
Tabla 4.6 Consumo de memoria – Convolucion en Dominio tiempo.

Al sumar los valores de memoria definidos en la tabla 4.6, se puede obtener un tamaño de 11264 posiciones de memoria que serán necesarios para implementar el algoritmo.

Se debe tener en cuenta que el algoritmo de la convolucion en el tiempo, ocupa muchos ciclos de reloj en el procesador, por lo que la implementación del algoritmo Esquema 4.16, con un microcontrolador ARM CORTEX-R5F de 330MHz, puede demorar varios milisegundos.

3. Espectro de Potencia por convolucion en el Dominio Frecuencia

En el siguiente diagrama se puede observar el flujo de operaciones que se deben realizar para obtener el espectro de frecuencia de la potencia si se realiza la operación a través del dominio Frecuencia.



Esquema 4.17 Convolucion en el dominio Frecuencia.

Función	Memoria	Descripción
V(kt)	2048	Almacena la señal alterna de voltaje.
I(Kt)	2048	Almacena la señal alterna de la corriente
FFT-V(f)	1024	Almacena la FFT del voltaje
FFT-I(f)	1024	Almacena la FFT de la corriente
P(f)	1024	Almacena la multiplicación de los espectros V(f)*I(f)

Tabla 4.7 Consumo de memoria – Convolucion en Dominio Frecuencia.

Al sumar los valores de memoria definidos en la tabla 4.7, se puede obtener un tamaño de 7168 posiciones de memoria que serán necesarios para implementar el algoritmo.

Dado que el algoritmo tiene mucho menos posiciones de memoria con los que trabajar, le tomara microsegundos realizar toda la operación al microcontrolador.

El equipo Analizador de calidad de energía, utiliza el algoritmo de la figura 4.17 para calcular el espectro de la potencia eléctrica.

IV. FUNCIONES MATEMATICAS DE ANALISIS

a. VALOR CUADRATICO MEDIO (RMS)

"El término "RMS" es sinónimo de "Root-Mean-Squared". La mayoría de los libros definen esto como ***"La cantidad de energía AC que produce el mismo efecto de calentamiento como una alimentación de CD equivalente".***

El término RMS, se refiere sólo a voltajes sinusoidales variables en el tiempo, las corrientes o formas de onda complejas eran la magnitud de los cambios de forma de onda en el tiempo y no se utiliza en el análisis de circuitos de corriente continua o cálculos eran la magnitud es siempre constante.

En otras palabras, el valor efectivo es un valor de CC equivalente que le indica cuántos voltios o amperios de corriente continua que una forma de onda sinusoidal variable en el tiempo es igual a en términos de su capacidad para producir la misma potencia" Referencia: http://www.electronics-tutorials.ws/accircuits/rms-voltage.html

El algoritmo RMS, se encuetra dentro de las librerias DSP,internamete esta configurado para trabajar con punto flotante, teniando una alta precision y velocidad en el momento de calcular el valor RMS de una señal.

El algoritmo de la librerias se basa en la formula:

$$x_{\mathrm{RMS}} = \sqrt{\frac{1}{N}\sum_{i=1}^{N} x_i^2} = \sqrt{\frac{x_1^2 + x_2^2 + \cdots + x_N^2}{N}}$$

Ecuación 4.14

b. MAXIMO, MINIMO Y MEDIA

Máximo: el algoritmo busca el valor máximo dentro un vector de valores, la función retorna el valor máximo y la posición de este dentro del vector.

Mínimo: el algoritmo busca el valor mínimo dentro un vector de valores, la función retorna el valor mínimo y la posición de este dentro del vector.

Media: calcula la media de un vector, la media se define como el promedio de elementos en un vector.

c. CALCULO DE POTENCIAS ELECTRICAS

La potencia aparente es el producto de las magnitudes de la tensión eficaz y la corriente eficaz y se utiliza frecuentemente para especificar el valor nominal de los equipos de potencia, como transformadores, UPS, Estabilizadores Ferro resonantes, fuentes Conmutadas. Referencia Daniel W. Electrónica de Potencia

La potencia aparente se expresa de la siguiente forma:

$$S = Vrms * Irms \quad \text{(Ecuacion 4.15)}$$

$$P = Vrms * Irms * fp \quad \text{(Ecuación 4.16)}$$

El factor de potencia de una carga se define como el cociente de la potencia activa y la potencia aparente

(Ecuación 4.17)

$$fp = \frac{P}{S}$$

En los circuitos que utilizan señales sinusoidales, los cálculos anteriores dan como resultado $fp = cos(\varphi)$, donde φ es el Angulo de fase entre las señales sinusoidales de tensión y de corriente.

Sin embargo, este es un caso especial y debería ser utilizado solo cuando tanto la tensión como la corriente sean sinusoides.

1. Fuente sinusoidal y carga no lineal

Si una fuente de tensión sinusoidal se aplica a una carga no lineal, la forma de onda de la corriente no será sinusoidal, pero se puede representar como una serie de Fourier.

Si la tensión de sinusoide es:

$$v(t) = V_1 \operatorname{sen}(\omega_0 t + \theta_1)$$

Y la corriente de Fourier se puede representar mediante la serie:

$$i(t) = I_0 + \sum_{n=1}^{\infty} I_n \operatorname{sen}(n\omega_0 t + \phi_n)$$

La potencia activa absorbida por la carga se calcula a partir de:

$$P = V_0 I_0 + \sum_{n=1}^{\infty} \left(\frac{V_{n,\,\text{máx}} I_{n,\,\text{máx}}}{2} \right) \cos(\theta_n - \phi_n)$$

$$= (0)(I_0) + \left(\frac{V_1 I_1}{2} \right) \cos(\theta_1 - \phi_1) + \sum_{n=2}^{\infty} \left(\frac{(0) I_{n,\,\text{máx}}}{2} \right) \cos(\theta_n - \phi_n)$$

$$= \left(\frac{V_1 I_1}{2} \right) \cos(\theta_1 - \phi_1) = V_{1,\,\text{rms}} I_{1,\,\text{rms}} \cos(\theta_1 - \phi_1)$$

Podemos notar que el único término de la potencia distinto de cero es el correspondiente a la frecuencia de la tensión aplicada.

Por lo tanto el factor de potencia de la carga se calcula a partir de la ecuación 4.17

$$fp = \frac{P}{S} = \frac{P}{V_{\text{rms}} I_{\text{rms}}}$$

$$= \frac{V_{1,\text{rms}} I_{1,\text{rms}} \cos(\theta_1 - \phi_1)}{V_{1,\text{rms}} I_{\text{rms}}} = \left(\frac{I_{1,\text{rms}}}{I_{\text{rms}}}\right) \cos(\theta_1 - \phi_1)$$

(Ecuación 4.18)

Donde la corriente eficaz se calcula a partir de:

$$I_{\text{rms}} = \sqrt{\sum_{n=0}^{\infty} I_{n,\text{rms}}^2} = \sqrt{I_0^2 + \sum_{n=1}^{\infty} \left(\frac{I_n}{\sqrt{2}}\right)^2}$$

(Ecuación 4.19)

El cociente entre el valor eficaz a la frecuencia fundamental y el valor eficaz total, $\frac{I1rms}{Irms}$

en la ecuación 4.18, es el factor de distorsión:

$$\text{FD} = \frac{I_{1,\text{rms}}}{I_{\text{rms}}}$$

Ecuación 4.20

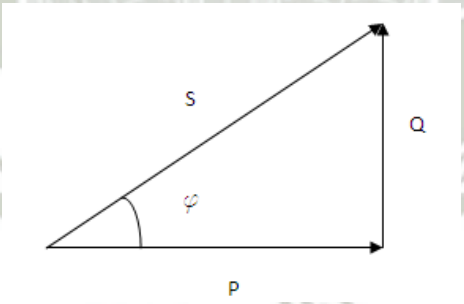
2. Selección de Fórmulas para el Algoritmo

El equipo calcula el valor RMS del voltaje y la corriente de la forma de onda que este a muestreado y almacenado en su memoria previamente.

Al tener la señal muestreada en la memoria RAM del microcontrolador también se puede aplicar la transformada de Fourier y obtener la fase de la señal de voltaje y corriente. **(APLICACION DEL ANALISIS DE FOURIER PARA DETERMINACION DE MAGNITUD Y FASES – FASE DEL ESPECTRO)**

En conclusión se tiene como variables conocidas el Vrms, Irms y el Angulo de desfase entre dos señales φ.

Calcular: Potencia Aparente [S], Potencia Activa [P] y Potencia Reactiva [Q].



Esquema 4.18 Triangulo de Potencias.

Las ecuaciones que el equipo ejecuta como algoritmo son:

$$S = Vrms * Irms$$

$$P = S * fp$$

$$fp = \frac{I_1 rms}{Irms} cos(\varphi)$$

Si se aplica Pitágoras al triangulo de impedancia podemos obtener la relación para el cálculo de la potencia Reactiva:

$$Q = \sqrt{S^2 - P^2}$$

(Ecuación 4.21)

d. CALCULO DISTORSION ARMONICA TOTAL THD

La distorsión armónica total es otro término utilizado para cuantificar la propiedad no sinusoidal de una forma de onda. El valor de THD es la relación entre el valor eficaz de todos los términos correspondientes a la frecuencias distintas de la fundamental y el valor eficaz del termino correspondiente a la frecuencia fundamental. (Hart Daniel W-Electrónica de Potencia).

Cuando se habla de los armónicos en las instalaciones de energía, son los armónicos de corriente los más preocupantes, puesto que son corrientes que generan efectos negativos. Es habitual trabajar únicamente con valores correspondientes a la distorsión armónica total (THD)

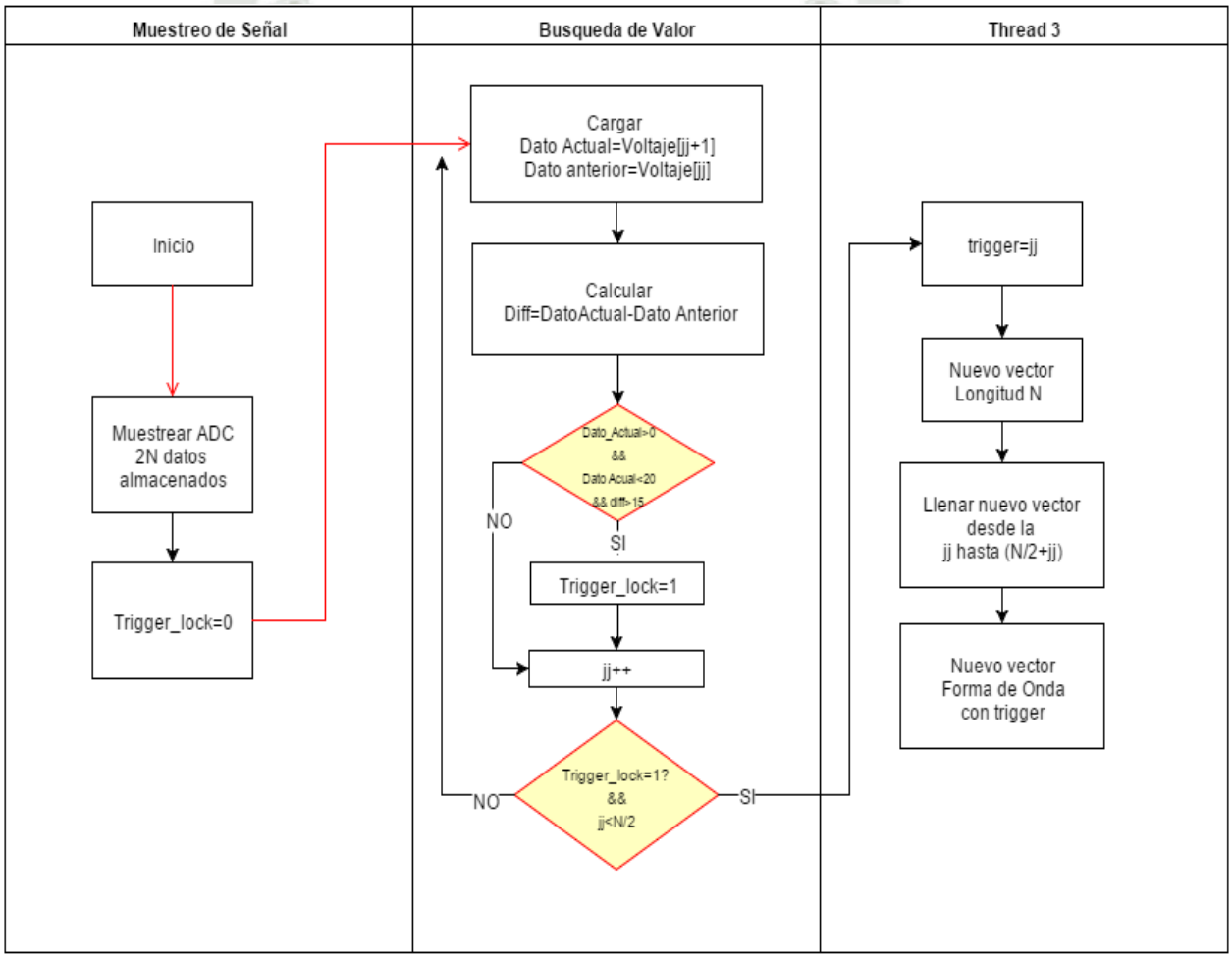
$$\text{THD} = \frac{\sum \text{Potencia de los armonicos}}{\text{Potencia de la frecuencia fundamental}} = \frac{P_1 + P_2 + P_3 + \cdots + P_N}{P_0}$$

(Ecuación 4.22)

METODOS Y GENERACION DE GRAFICOS EN PANTALLA

I. DIAGRAMA DE BLOQUES DEL TRIGGER DIGITAL

Es sistema de disparo nos permite sincronizar las capturas que hace el osciloscopio con la señal que queremos observar. En resumidas cuentas, nos permiten estabilizar la imagen de la señal para poder observarla.



Esquema 4.19 Diagrama de Trigger Digital

a. FUNCION DEL TRIGGER

El sistema de trigger nos permite capturar las señales (o formas de onda) cuando se cumple una condición determinada en la señal (supera un umbral un flanco de subida o de bajada).

Los trigger por flanco suelen ser los más comunes y los que más se suelen utilizar. La captura de la señal se produce cuando la señal que estamos observando supera un umbral.

b. FUNCIONAMIENTO

Variables	Función
N	Numero de Muestras del ADC.
Tigger_lock	Bandera que activa o desactiva el trigger.
Dato Actual	Almacena el valor de la posición [jj+1].
Dato Anterior	Almacena el valor de la posición [jj].
Diff	Variación entre dos valores actual y anterior de la forma de onda.
jj	Contador, ayuda a barrer todas las posiciones de memoria.
Nuevo Vector	Almacena una copia de la forma de onda, partiendo de la posición jj hasta (N+jj).

Tabla 4.8 Variables del Trigger

El valor de N será la longitud horizontal de la pantalla LCD = 800 pixeles, dado que se desea muestrear en toda la longitud la forma de onda.

El sistema captura en un vector una longitud de 2N o 1600 valores del ADC, es necesario tener esta cantidad de valores para que la búsqueda echa por el algoritmo del trigger no presente errores lógicos.

Luego se inicializa una bandera Trigger_lock, que tiene como tarea salir del bucle de búsqueda si las condiciones de trigger se cumplen.

Las variables Dato Actual y Anterior, almacenan el valor dentro de la posición [jj] del vector que almacena la forma de onda, la diferencia de estos valores es almacenado en Diff.

Si la variable Diff tiene un valor negativo la señal tiene una transición de positivo a negativo (flanco descendente), pero si es valor positivo la señal tiene una transición de negativo a positivo (flanco ascendente).

La condicional pregunta si Diff>15 y 0<dato actual<20; el valor de 15 y 20, fueron ajustados luego de muchas pruebas realizadas.

Si las condiciones se cumplen la bandera Trigger_lock se carga con valor de 1 y el programa sale del bucle de búsqueda.

Mientras se realiza la búsqueda de un valor que cumpla las condiciones, el contador jj, se incremente de uno en uno y se define una vez que la condición de búsqueda se cumpla.

El valor de jj, nos indica la posición del valor que cumple las condiciones de búsqueda

En resumen el valor jj, almacena la posición del valor en donde se produce el flanco de subida o bajada de la señal muestreada.

Luego se puede graficar la forma de onda en la pantalla LCD, partiendo de la posición jj, se considerar que siempre se graficara en la pantalla 800 valores.

II. DIAGRAMAS DE BLOQUES DE GENERACION DE GRAFICOS EN PANTALLA

La pantalla LCD tiene dimensiones de 800x480 pixeles, dentro de estas dimensiones se debe mostrar un Menú con botones que permitan la navegación y operación de las diferentes tareas que puede cumplir el equipo, como también mostrar las formas de onda, el espectro de Fourier y las diferentes formas de onda.

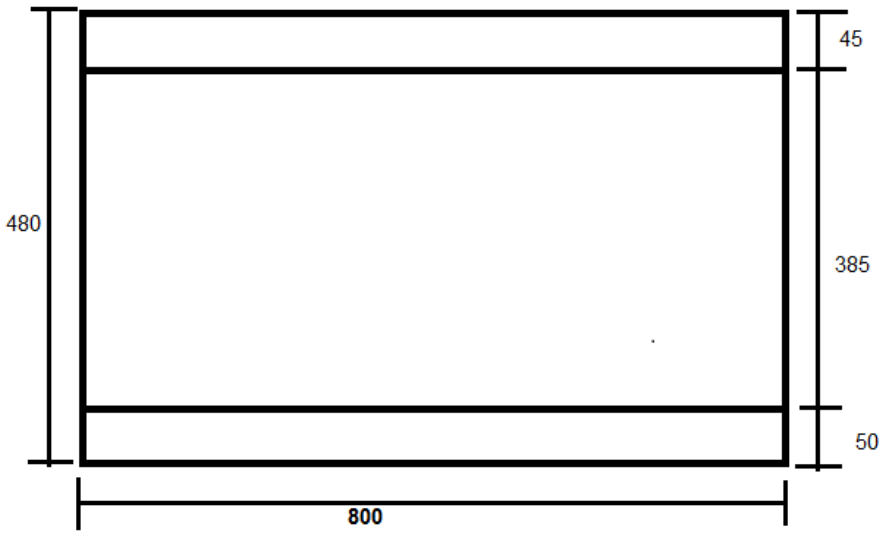
La forma de onda nos proporciona una valiosa información sobre la señal. En cualquier momento podemos visualizar la altura que alcanza y, por lo tanto, saber si el voltaje ha cambiado en el tiempo (si observamos, por ejemplo, una línea horizontal podremos concluir que en ese intervalo de tiempo la señal es constante).

a. ESCALAMIENTO DE SEÑALES PARA LA PANTALLA LCD

Una forma de onda es la representación gráfica de una onda. Una forma de onda de tensión siempre se presentará con el tiempo en el eje horizontal (X) y la amplitud en el eje vertical (Y).

El equipo debe mostrar dos formas de onda, voltaje y corriente, ambas formas de onda deben tener determinadas una cierta cantidad de pixeles en la pantalla LCD.

Considerando que la pantalla debe contar con un pequeño menú de botones en la pantalla superior de 45 pixeles e inferior de 50 pixeles de altura, para la navegación entre las diferentes tareas.



Esquema 4.20 Dimensiones de la pantalla LCD en pixeles

El espacio que ocupan los caracteres para mostrar información en la pantalla tiene una altura de 20 pixeles, por lo que se considera como un espacio donde no se puede mostrar la forma de onda, así que se resta estos pixeles del total del espacio considerado para mostrar la forma de onda.

Dimensiones de pantalla 800x480 pixeles

$$Lr = \ altura\ para\ las\ graficas$$

$$Lxy = \ altura\ para\ los\ graficos\ XY$$

$$Lseñal = \ altura\ para\ las\ formas\ de\ onda$$

$$Lr = 480 - 45 - 45$$

$$Lr = 390$$

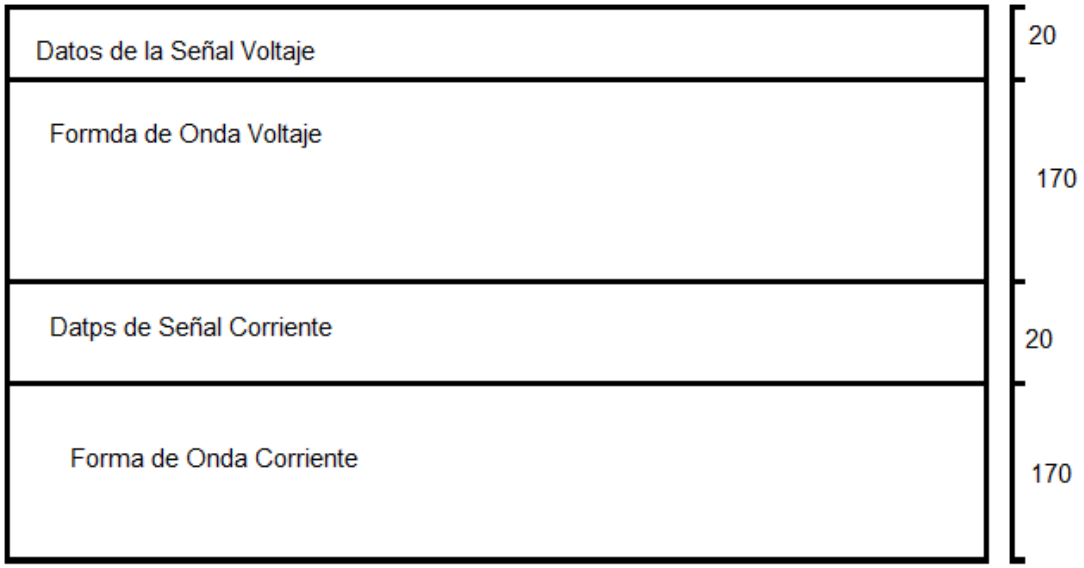
Resto la altura del texto

$$Lxy = 390 - 20 - 20$$

$$Lxy = 350$$

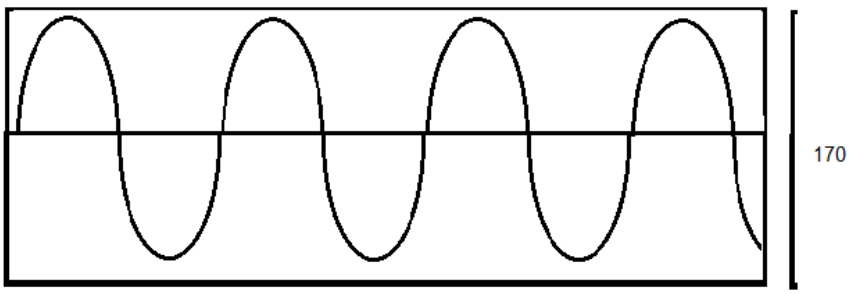
$$Lseñal = \frac{Lxy}{2} = \ 175\ pixels$$

De acuerdo a estos últimos cálculos la altura máxima para las formas de onda es de 175 Pixeles.



Esquema 4.21 Distribución de la pantalla LCD

Dentro de estos 175 pixeles, se debe mostrar a forma de onda de pico a pico, dejando como la mitad de este recuadro como el eje y=0.



Esquema 4.22 Zona para la forma de Onda

Ecuaciones

$$Longitud\ Vertical, LV = 170$$

$$Maximo\ Pico\ en\ pixeles$$

$$Mppx = \frac{LV}{2} = 85$$

$$Maximo\ Voltaje\ rms\ registrado\ por\ el\ equipo$$

$$MVrms = 600Vrms$$

$$Pico\ del\ Voltaje\ MVp = 600\sqrt{2} = 848.52\ V$$

Aplicando la ecuación 3.8

$$Vo(Vin, Vcc) = 1.049e - 3 * V(t) + 0.156V$$

Tenemos un valor de entrada en el conversor ADC de:

$$Vinadc = 848.52 * 1.049e - 3$$

$$Vinadc = 0.890V$$

Valor binario ADC (Ecuación 4.23)

$$Vbin = 0.890 * \left(\frac{4095}{3.3}\right)$$

$$Vbin = 1104$$

Por lo tanto se tiene una relación de 1104 valores para 85 pixeles

Valores	Pixeles
1104	85
Vxbin	Xpixel

(Ecuación 4.24)

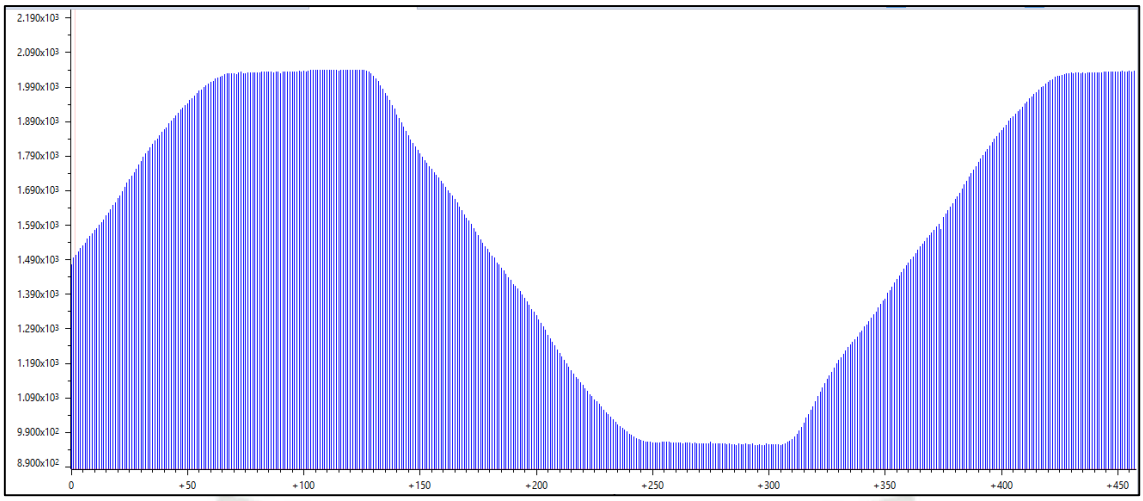
$$Xpixel = Vxbin * \frac{85}{1104}$$

b. INTERPOLACION PARA SEÑALES DIGITALES

La **interpolación** consiste en hallar un dato dentro de un intervalo en el que conocemos los valores en los extremos.

Dado que la señal en la memoria del microcontrolador son datos discretos (Esquema 4.23).

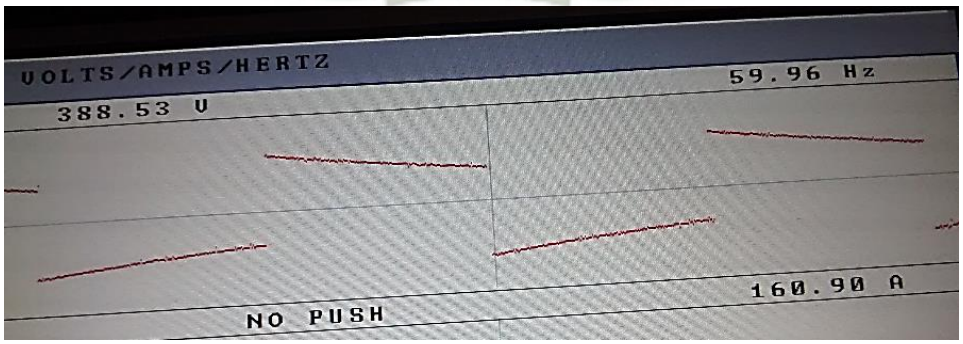
Se puede considerar un dato en la posición [k], como el extremo de inicio y el dato en la posición [k+1], como el extremo final.



Esquema 4.23 Onda Discreta – Sin interpolación

De modo que es necesario rellenar los espacios vacíos entre un dato y otro para una correcta visualización de la forma de onda en la pantalla LCD.

Una de las formas de onda con cambios muy abruptos es la cuadrada que presentan cambios abruptos, que no pueden ser muestreados por el ADC del microcontrolador (Esquema 4.20), siendo indispensable un algoritmo que llene estos espacios vacíos con valores numéricos que mantengan una relación con la forma de onda y puedan visualizarse en la pantalla LCD.



Esquema 4.24 Señal Cuadrada sin Interpolar

Esto también incluye los valores picos que se sobreponen sobre la señal análoga, estos son valores con un cambio rápido y abrupto en el tiempo, por lo que cual el conversor análogo digital no puede capturarlos en su forma completa, pero se puede obtener valores puntuales, que gracias al algoritmo de interpolación se pueden visualizar correctamente.

Se planeó la solución de utilizar rectas entre dos puntos, que cumplan ciertas condiciones de distancia entre ellos, no siempre se dibujara rectas entre los puntos y por el motivo de que generaría retardos de cálculo en el procesador.

Dentro de la memoria del microcontrolador se encuentra la señal discreta almacenada, por consiguiente se conoce los siguientes datos:

$$Y(k+1) = \ una\ muestra\ adelante$$

$$Y(k) = \ muestra\ actual$$

$$K = \ Numero\ de\ la\ muestra$$

$$K+1 = \ numero\ de\ la\ muestra\ + 1\ posicion$$

Las librerías para la pantalla LCD cuentan con una función que dibuja una línea recta entre dos puntos $(x0, y0) \rightarrow (x1, y1)$.

Se define como:

$$x0 = k$$

$$x1 = k + 1$$

Para los valores del eje Y:

$$y0 = y(k)$$

$$y1 = y(k + 1)$$

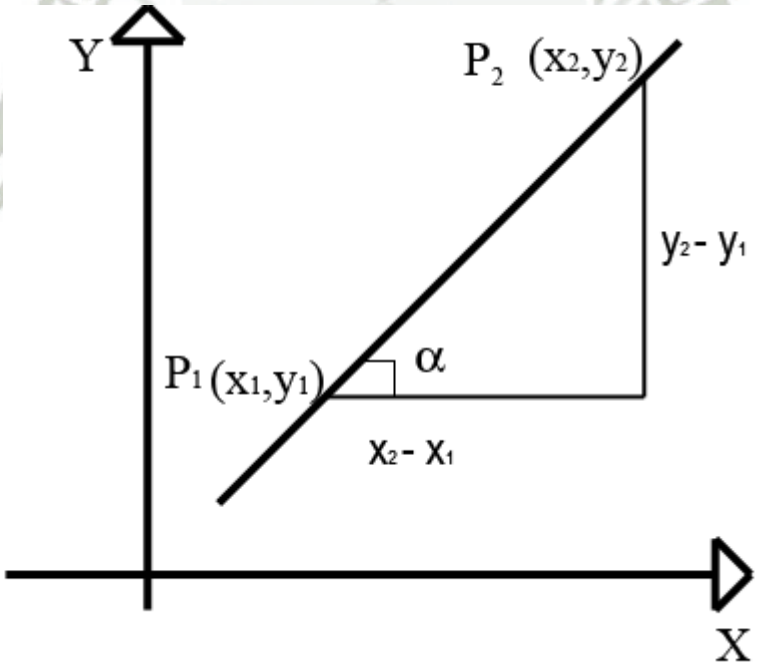
Las coordenadas son enviadas a la función quien se encarga de dibujar rectas entre los puntos, siempre y cuando se cumpla la condición de superar la pendiente, esto se verá más adelante explicado de manera mas amplia.

1. Sensibilidad de la Interpolación

Las coordenadas $(x1, y1)$ y $(x2, y2)$ son puntos dentro de un vector ya escalado por la ecuación 4.24, también son utilizadas para el cálculo de Sensibilidad.

Sin la sensibilidad el algoritmo de interpolación estaría activada todo el tiempo y esto ocuparía el procesador, generando retardos notables en las demás tareas.

La pendiente de la recta permite hallar la sensibilidad para el algoritmo de interpolación lineal.



Esquema 4.25 Pendiente de una Recta

Dado que dibujar rectas tendría al microcontrolador ocupado la mayor parte del tiempo, se debe cumplir la siguiente condición para que el microcontrolador dibuje una recta entre dos puntos:

$$y2 - y1 > 5 \quad \text{(Ecuación 4.25)}$$

La diferencia entre dos puntos en el eje y, debe ser mayor a 5 pixeles.

Se escogió una distancia de 5 pixeles, dado que la resolución de la pantalla LCD es de 800x480 y con 5 pixeles de distancia entre dos puntos en su matriz, no es muy notable para el ojo humano.

Si hace menor este valor, el microcontrolador dibujando rectas la mayor parte del tiempo y esto genera retraso.

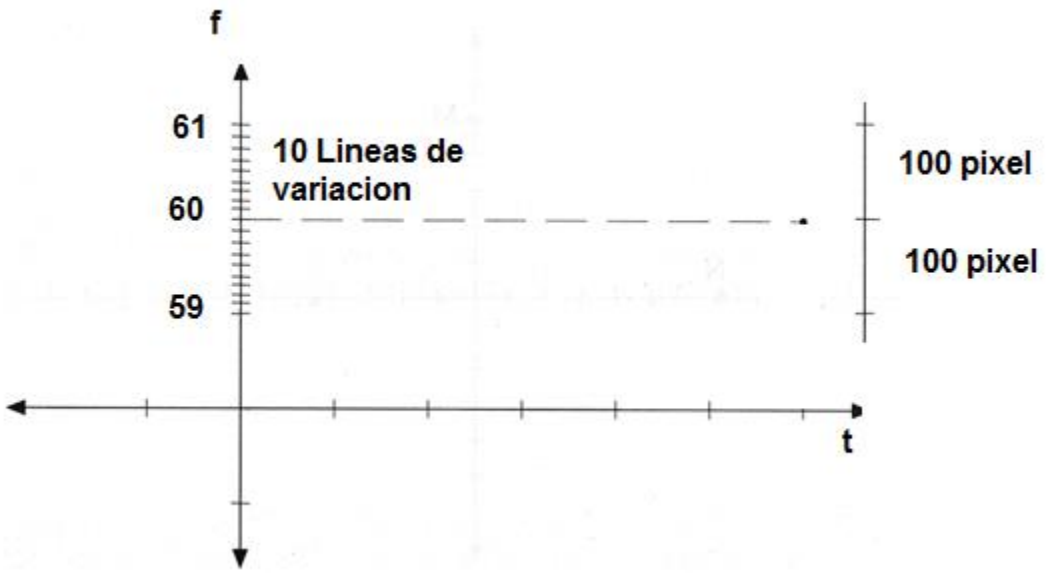
c. INTERPOLACION PARA GRAFICO FRECUENCIA VS TIEMPO

La mejor forma de visualizar los cambios de frecuencia durante un cierto periodo de tiempo, es generar un gráfico donde se presenta la variación de la frecuencia de línea, durante un periodo de 5 segundos.

En esta etapa se utiliza el **Modulo Captura de ancho de pulso.**

Para generar este grafico se necesita, tener almacenada en la memoria RAM del microcontrolador, el valor actual de la frecuencia.

Premisas:



Esquema 4.26 Dimensiones LCD, Grafica F VS t

Se desea observar la frecuencia base de línea y la variación de ±1Hz.

Frecuencia de Línea en Perú: 60Hz

Variación entre: [59,61] Hz

Pixeles asignados para la tarea: 200 pixeles eje todo el eje Y.

Lineas de variación en Frecuencia = 10

Calculo

Se puede concluir del Esquema 4.26 lo siguiente:

Resolución de frecuencia

$$Rfrec = 60 - 59 = \frac{1}{10}$$

$$Rfrec = 0.1Hz$$

Resolución de pixeles

$$Rpxl = \frac{100}{10}$$

$$Rpxl = 10\ pixeles$$

La relación entre pixeles y resolución de frecuencia es

Frecuencia		Pixeles
0.1	----------	10
$Frec$	----------	Pxl

Ecuación 4.26

$$Pxl = Frec(\frac{10}{0.1})$$

Se debe tener en cuenta que $Frec$, es solo la variación entre la frecuencia medida y la frecuencia ideal, ejemplo:

Frecuencia medida = 60.152 Hz

Frecuencia Ideal = 60.000 Hz

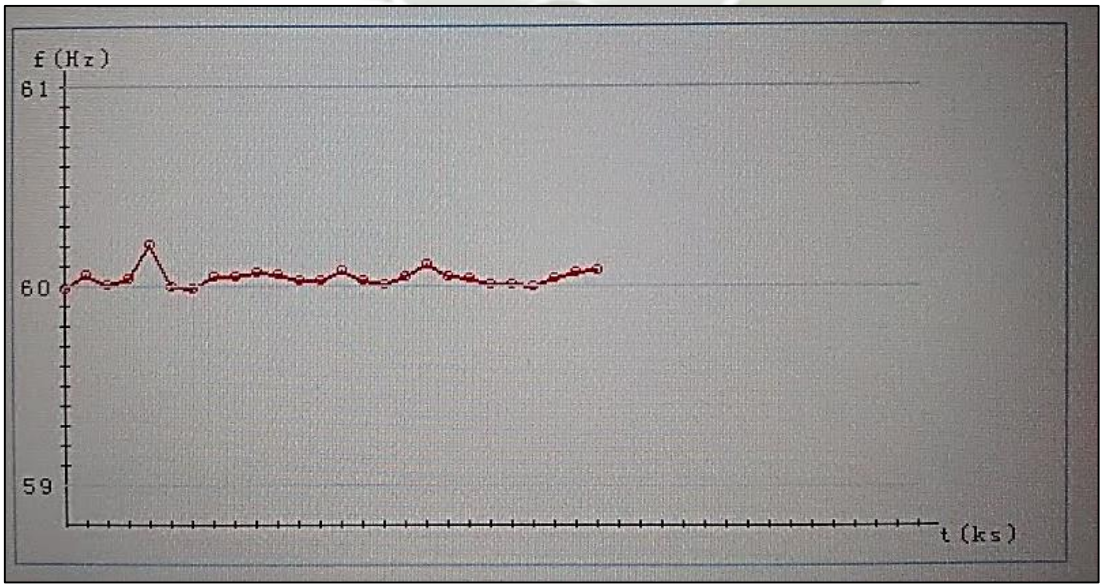
Ecuación 4.27

$$Frec = Frecuencia\ medida - Frecuencia\ ideal$$

$$Frec = 60.152 - 60.000$$

$$Frec = 0.152\ Hz$$

En la esquema 4.27, se puede observar la variación de la frecuencia dentro de 10 valores (eje Y) y el tiempo (eje X).



Esquema 4.27 Grafica Frecuencia VS Tiempo

d. INTERPOLACION PARA GRAFICO VALOR RMS VS TIEMPO

Siguiendo el mismo principio que el grafico Frecuencia VS Tiempo, la función de interpolación es utilizada para graficar el valor RMS de la señal de voltaje y corriente durante un periodo de muestreo de 1 segundo.

La muestra se almacena dentro de la memoria del microcontrolador, de la cual se extraerán el máximo valor RMS y el mínimo valor RMS de cada señal independientemente.

Esto permitirá al microcontrolador generar un gráfico de alta definición donde se pueda observar la variación del voltaje RMS y la corriente RMS.

Para esta sección se utiliza dimensiones de la pantalla mucho más grandes y se requiere graficar el valor RMS del voltaje y la corriente.

Se debe escalar el valor RMS del voltaje y la corriente de acuerdo a las dimensiones requeridas dentro de la pantalla LCD.

Dado que la pantalla se maneja como coordenadas, se le asigna al grafico de voltaje RMS, las siguientes dimensiones:

0 hasta 800 pixeles de ancho, las coordenadas de altura serán de 90 hasta 240 pixeles.

Calculo para la ecuación del voltaje

Datos

Vinf: El menor voltaje RMS almacenado en la memoria.

Vsup: El mayor voltaje RMS almacenado en la memoria.

Recta	Voltaje	YPixeles
(x0,y0)	Vsup	90
(x1,y1)	Vinf	240

Calculo de pendiente

$$m = \frac{90 - 240}{20 + Vsu - Vinf}$$

$$= -\frac{150}{20 + Vsu - Vinf}$$

Ecuación de la recta y-y0=m(x-x0)

$$y - 240 = m(x - Vin + 10)$$

$$y = \left(\frac{-150}{20 + Vsu - Vinf}\right)(x - Vin + 10) + 240$$

Ecuación 4.28

Se agrega un valor de 10 al Vin por si existiera algún tipo de fluctuación.

Vinf: El menor voltaje RMS almacenado en la memoria.

Vsup: El mayor voltaje RMS almacenado en la memoria.

Recta	Voltaje	YPixeles
(x0,y0)	Vsup	90
(x1,y1)	Vinf	240

Calculo de pendiente

$$m = \frac{90 - 240}{20 + Vsu - Vinf}$$

$$= -\frac{150}{20 + Vsu - Vinf}$$

Ecuación de la recta y-y0=m(x-x0)

$$y - 240 = m(x - Vin + 10)$$

$$y = \left(\frac{-150}{20 + Vsu - Vinf}\right)(x - Vin + 10) + 240$$

Ecuación 4.28

Se agrega un valor de 10 al Vin por si existiera algún tipo de fluctuación.

De la misma manera se relaciona las dimensiones para la corriente

Iinf: El menor corriente RMS almacenado en la memoria.

Isup: El mayor corriente RMS almacenado en la memoria.

Recta	Voltaje	YPixeles
(x0,y0)	Isup	290
(x1,y1)	Iinf	420

Calculo de pendiente

$$m = \frac{290 - 420}{20 + Isup - Iinf}$$

$$= -\frac{130}{20 + Isup - Iinf}$$

Ecuación de la recta y-y0=m(x-x0)

$$y - 420 = m(x - Iin + 10)$$

$$y = \left(\frac{-130}{20 + Isup - Iinf}\right)(x - Vin + 10) + 420$$

Ecuación 4.29

Se agrega un valor de 10 al Vin por si existiera algún tipo de fluctuación.

Esquema 4.28 Grafica Voltaje/Corriente VS Tiempo

e. INTERPOLACION PARA VISUALIZACION DE FORMA DE ONDA ALMACENADAS EN MEMORIA RAM

Utiliza el mismo algoritmo de interpolación para señales.

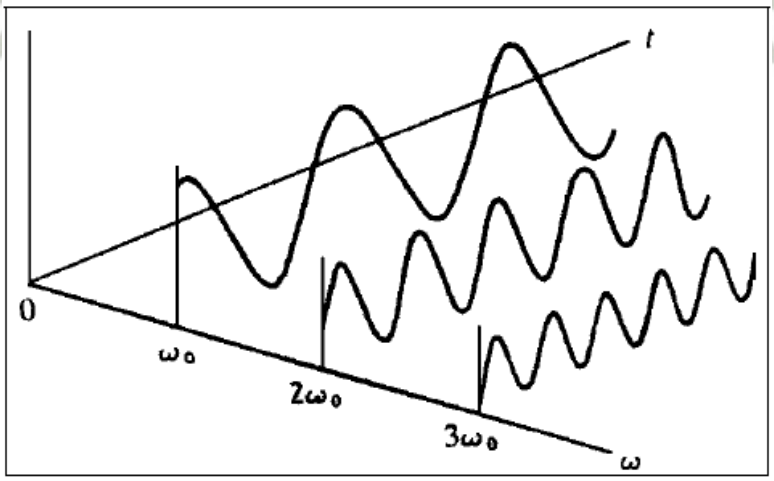
La única diferencia es que las señales visualizadas en la pantalla LCD fueron almacenadas en una matriz de memoria.

f. VISUALIZACION DE LOS ARMONICOS EN EL DOMINIO DE LA FRECUENCIA

Para continuar con esta etapa es necesario que se tengan en cuenta los datos y métodos explicados en: "**METODO DE ANALISIS DE SOLO ARMONICOS, CAPITULO 4**".

En la Tabla 4.5 "Definición de vectores / Armónicos", se describe el vector de nombre "Vector_magnitud", este contiene todo el espectro de Fourier de la señal previamente cargada.

Dado que se trabaja con señales periódicas en el tiempo, mantendrá siempre una misma forma en su espectro de Fourier.



Esquema 4.29 Armónicos de un Espectro

Los armónicos se generan en frecuencias múltiplos de la frecuencia fundamental y la Resolución de frecuencia Δf, permite conocer que posición que ocupara cierta frecuencia dentro del Vector_Magnitud.

Ejemplo: ¿Qué posición que ocupara la frecuencia fundamental de línea dentro del vector Magnitud?

Ecuación 4.30

$$K_f = \frac{Frec}{\Delta f}$$

$$= \frac{60Hz}{10Hz} = 6$$

En la posición 6 del vector se encuentra la componente fundamental de la señal muestreada.

Como se mencionó los armónicos se generan en frecuencias múltiplos de modo que si deseamos saber la posición del segundo y tercer armónico, estas serían:

Armónico	Frecuencia	$K_f = \frac{Frec}{10}$
2do	120	12
3er	180	18

Tabla 4.9 Tabla Armónico/Posición I

De acuerdo al último calculo, podemos definir lo siguiente:

Armónico	$K_f = \frac{Frec}{10}$
1	6
2	12
3	18
. . .	. . .
59	354

Tabla 4.9 Tabla Armónico/Posición II

De acuerdo a estos últimos datos, se puede encontrar un armónico para una frecuencia de 60 Hz, cada 6 posiciones y para extraer la información de solo los armónicos del Vector_Magnitud, se implementó una instrucción "for", que tenga:

Inicio =6

Final =6*59=354

Tamaño de paso = 6

```
for (x = INICIO; x < FINAL; x = x + TAMAÑO DE PASO)
```

En el caso de una frecuencia de 50Hz

$$K_f = \frac{Frec}{10} = \frac{50}{10} = 5$$

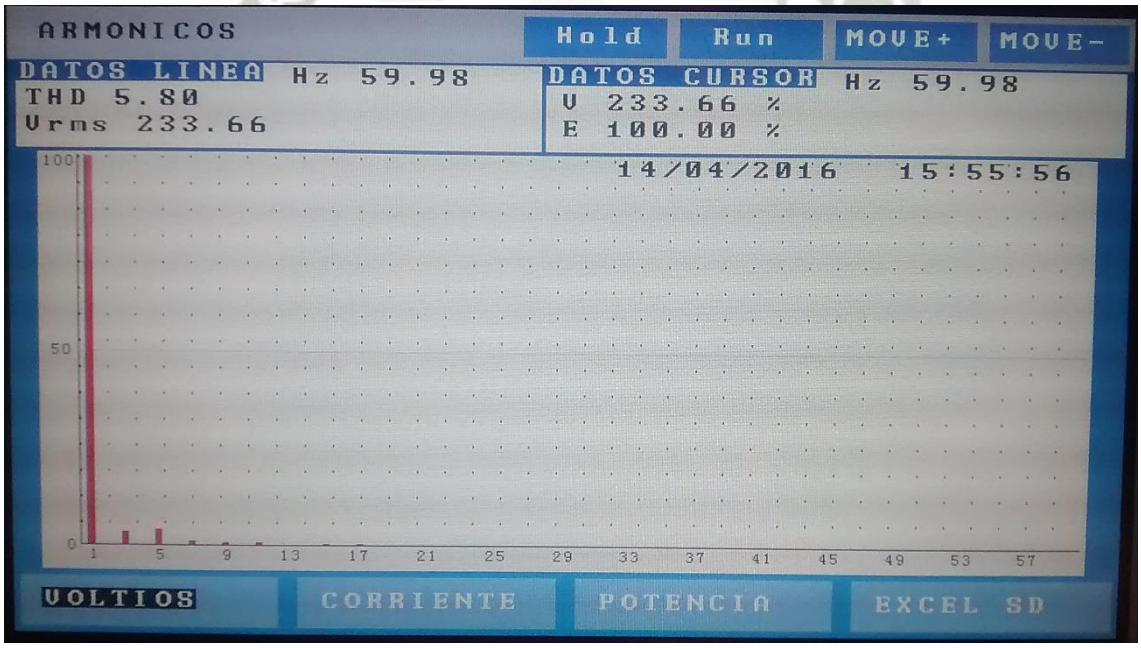
Con este dato se puede calcular:

Inicio =5

Final =5*59

Tamaño de paso = 5

Para otras frecuencias, el algoritmo calcula automáticamente el Inicio, el final y el tamaño de paso.



Esquema 4.30 Grafica de Solo Armónicos

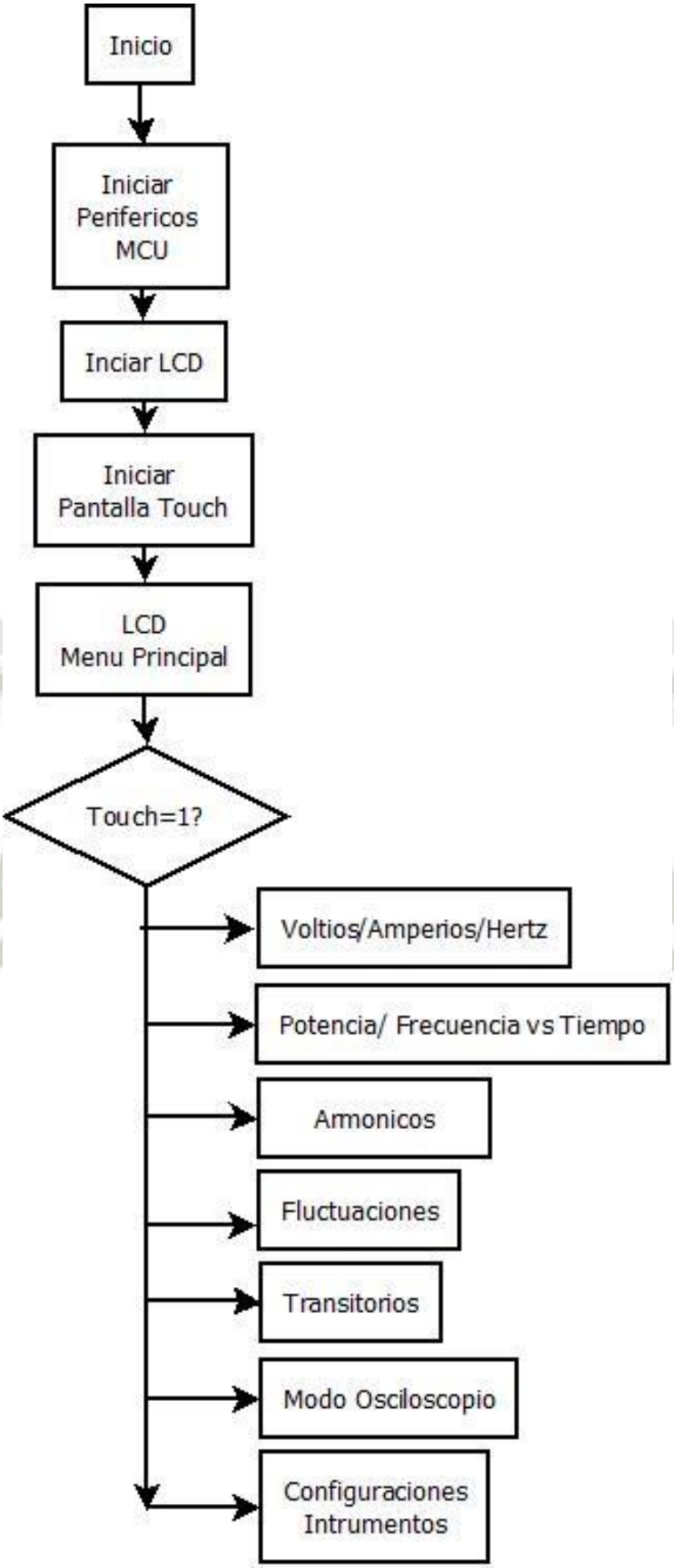
SISTEMA OPERATIVO DE ANALISIS ELECTRICO

I. DIAGRAMA DE FLUJO GENERAL DEL SISTEMA DE CONTROL

a. DISPOSICION DE PROGRAMAS

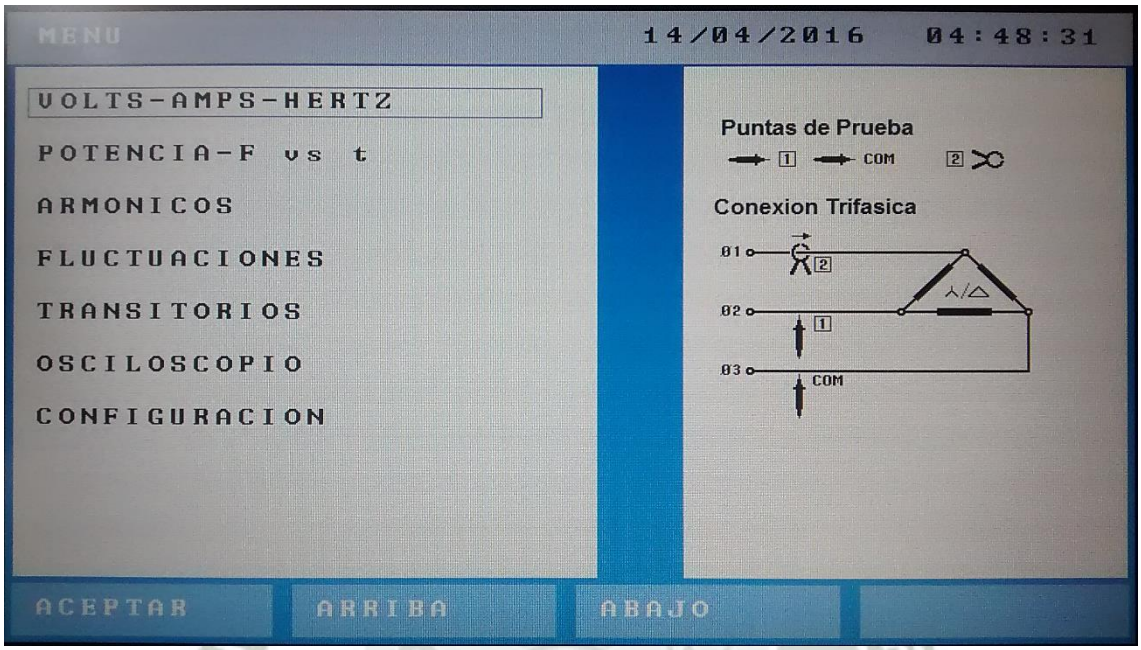
En el flujograma 4.1 se muestra el diagrama de flujo en general, este nos indica cual es el orden de las diferentes tareas que puede realizar el equipo Analizador de Calidad de Energía.

El funcionamiento del equipo se da mediante 6 tareas, entre las que se puede navegar a través del teclado en pantalla y la bandera Touch.



Flujograma 4.1 Sistema de Control General

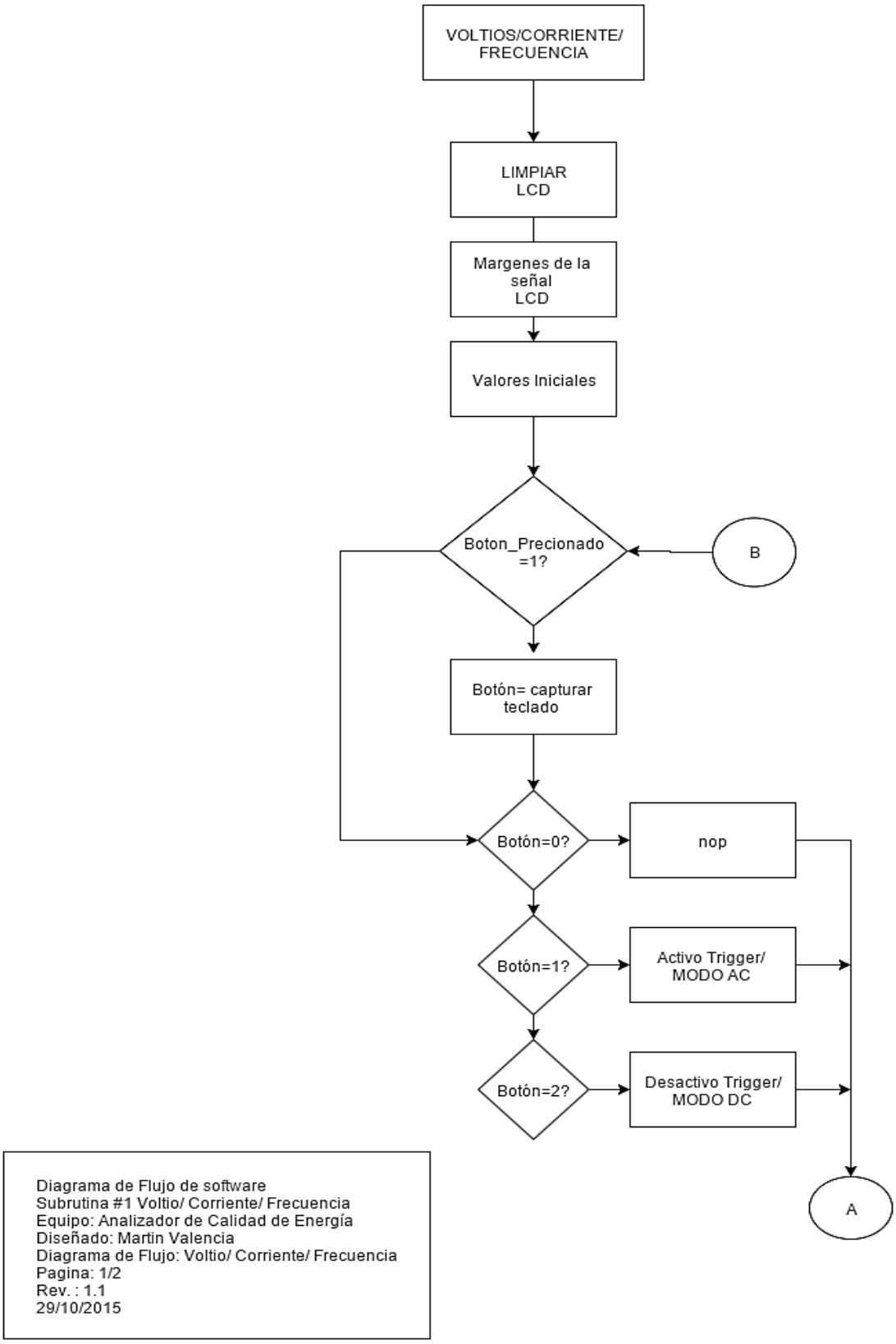
Al encender el equipo, muestra en pantalla un Menú de operaciones, se puede navegar entre ellos con los botones de “ARRIBA” and “ABAJO” y para seleccionar una tarea a ejecutar se utiliza el botón “ACEPTAR”.



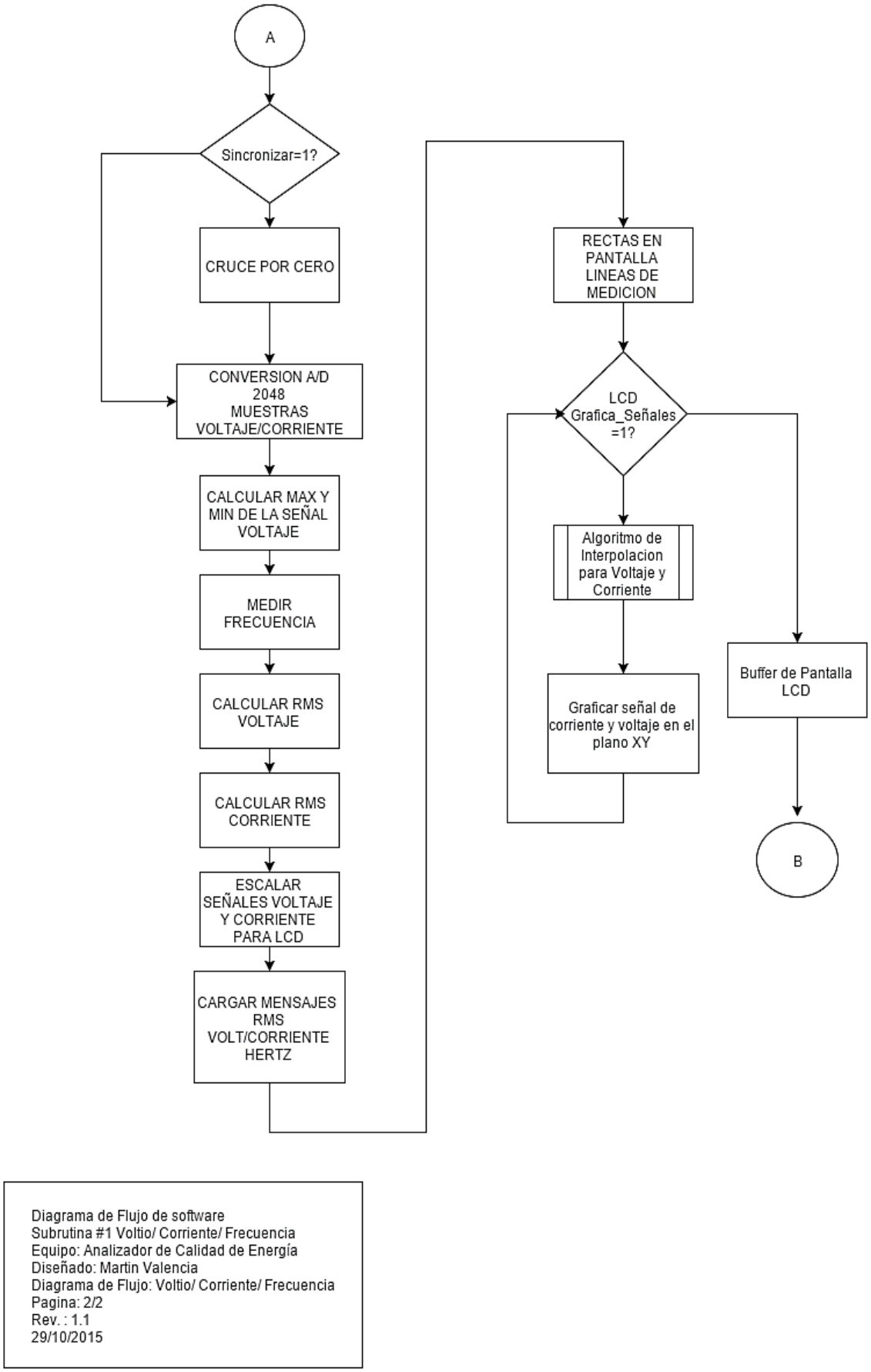
Esquema 4.31 Menú Principal en Pantalla

Con este último botón se abandona la pantalla principal y en su lugar se ejecuta el programa seleccionado.

II. DIAGRAMA DE BLOQUES DEL PROGRAMA VOLTAJE, CORRIENTE Y FRECUENCIA, OSCILOSCOPIO.



Flujograma 4.2.a Voltaje, Corriente y Frecuencia / Osciloscopio



Flujograma 4.2.b Voltaje, Corriente y Frecuencia / Osciloscopio

El Flujograma 4.2, contiene el algoritmo de operación de las tareas Voltaje, Corriente y Frecuencia y al mismo tiempo se utiliza el mismo algoritmo en la tarea Osciloscopio, con la única diferencia que este último no tiene limitada su pantalla con rectángulos, permitiendo al usuario un mayor juego de amplitud para la señal como se verán a continuación.

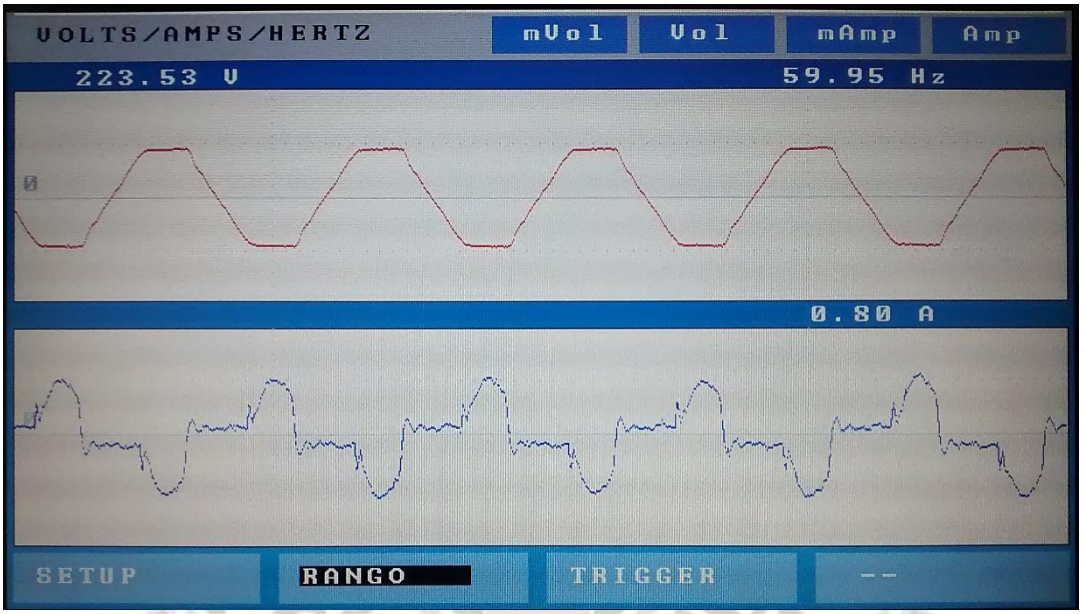
c. FUNCIONAMIENTO

Se acede a la tarea desde el Menú principal del sistema, se procede a limpiar la pantalla LCD y se generan los márgenes de la pantalla esto es solo en el modo "Voltaje, Corriente y Frecuencia", mientras que el modo "Osciloscopio" no se dibujan dichos márgenes.

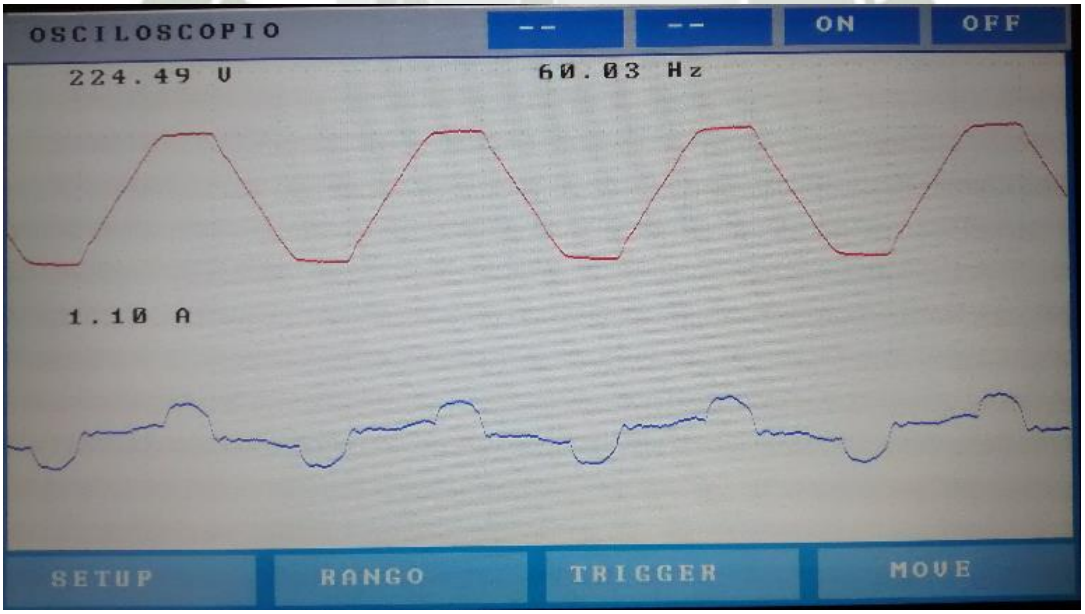
Se leen la memoria interna que contiene la configuración de resolución para visualizar las señales en pantalla.

Se dibujan botones en pantalla que permiten navegar entre las diferentes tareas que el algoritmo puede ejecutar.

d. PRUEBAS DE FUNCIONAMIENTO

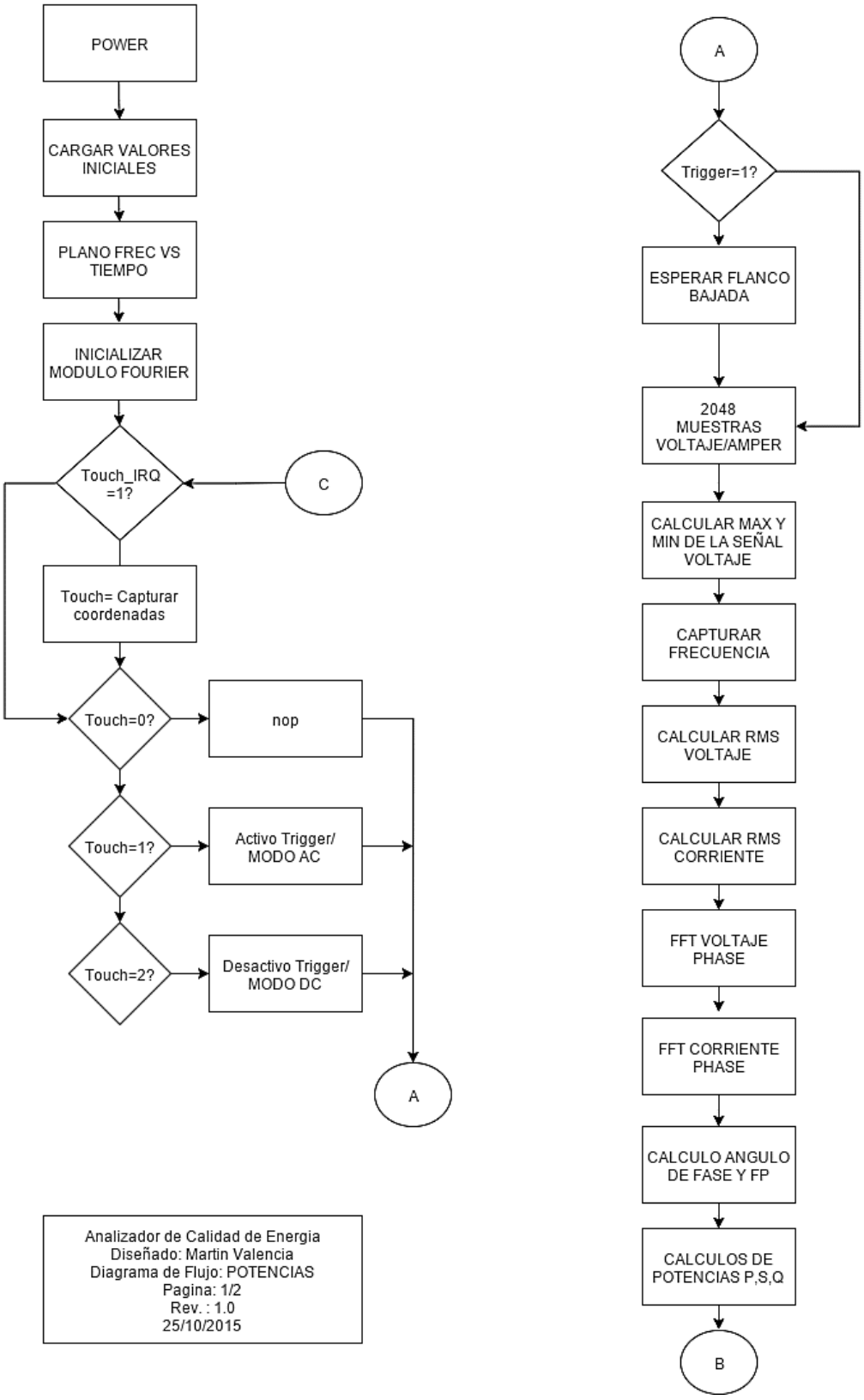


Esquema 4.32 Pantalla Modo Voltaje, Corriente y Frecuencia

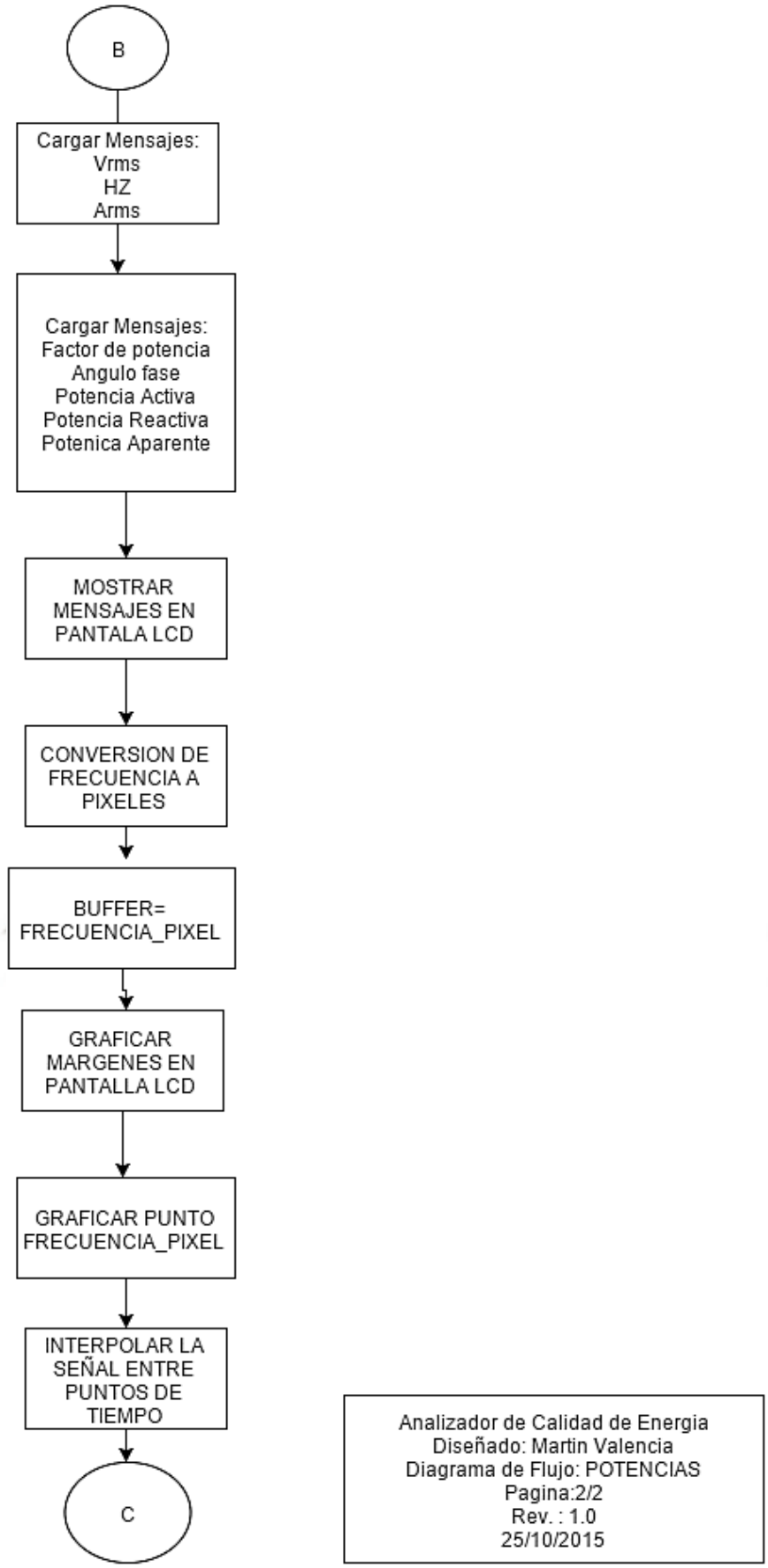


Esquema 4.33 Pantalla Modo Osciloscopio

III. DIAGRAMA DE BLOQUES DEL POTENCIA, FRECUENCIA VS TIEMPO



Flujograma 4.3.a Potencia



Flujograma 4.3.b Potencia

a. FUNCIONAMIENTO

Se acede a la tarea desde el Menú principal del sistema, se procede a limpiar la pantalla LCD y se generan en pantalla los márgenes de un plano XY para visualizar una gráfica de Frecuencia vs Tiempo, esta será actualizada cada 5 segundos.

El algoritmo principal utiliza la transformada de Fourier para calcular el angulo de desfase y posteriormente encontrar las diferentes potencias eléctricas y mostrarlas en pantalla.

El algoritmo cuenta con un sistema de trigger de alta precisión, este siempre espera un evento producto del flanco de subida que viene de las compuertas Smith Trigger, esto permite que el angulo de fase entre la señal de corriente y voltaje tenga una alta precisión. Para activar dicho Trigger, solo se debe acceder por pantalla y activarlo.

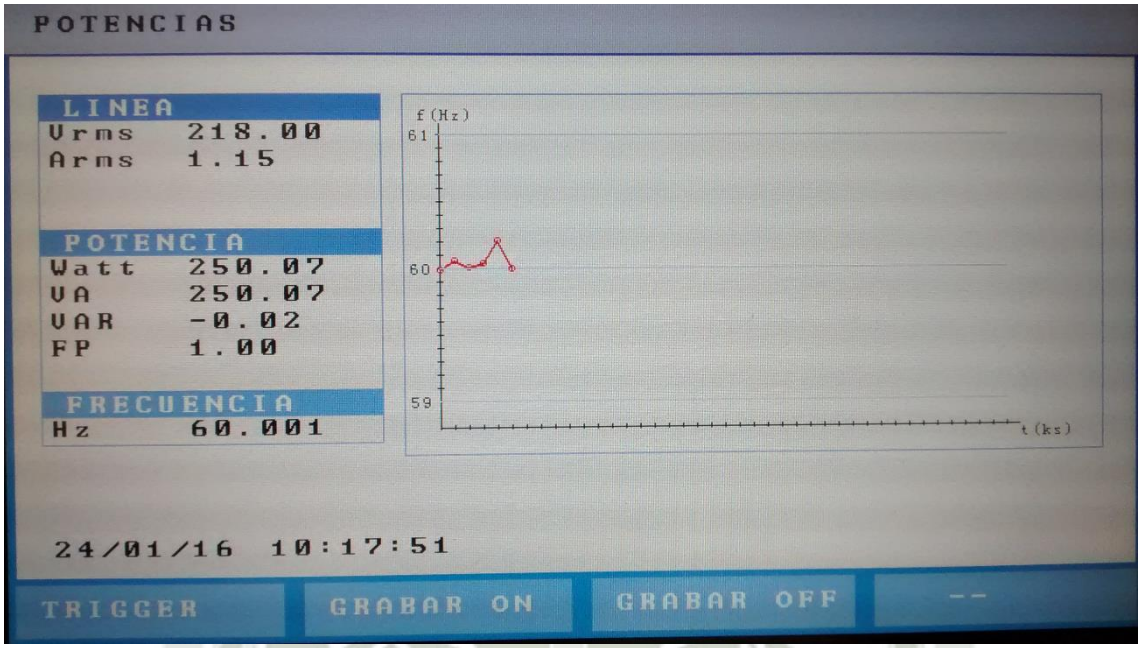
El algoritmo muestra en pantalla el Voltaje y Corriente RMS, Angulo de fase, Potencia Activa, Reactiva, Aparente, Frecuencia y Factor de potencia.

El Algoritmo también permite almacenar los datos en forma de ITEMS, con un número máximo de 2000 ITEM.

Se define como ITEM a una captura total de datos, el equipo muestrea un ITEM de datos cada 5 segundos como se mencionó anteriormente.

Esto permite al equipo almacenar un total de 2 Horas y 47 Minutos, de información referente a solo la potencia Eléctrica de la señal Monofásica que estamos analizando.

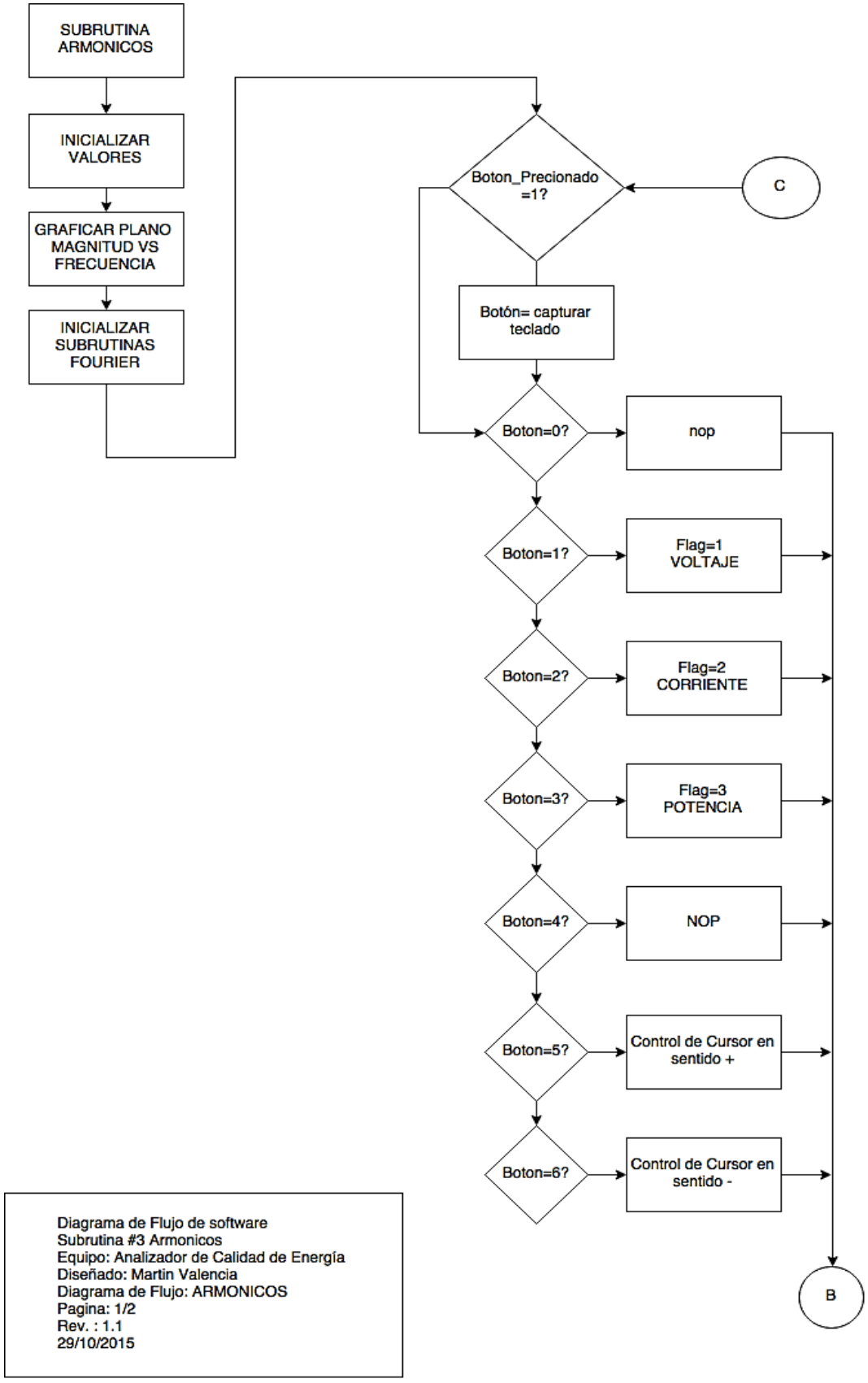
b. PRUEBAS DE FUNCIONAMIENTO



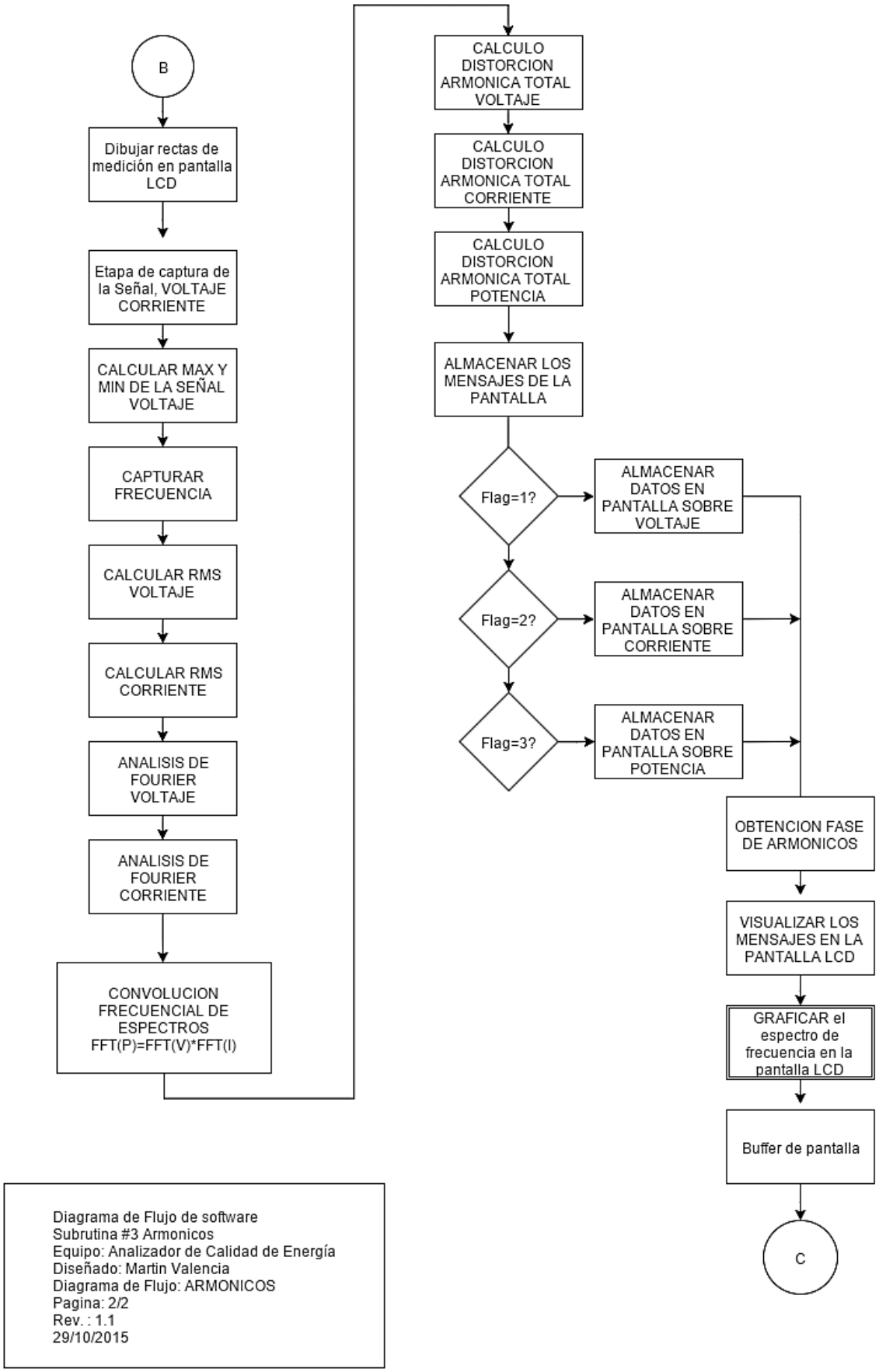
Esquema 4.34 Pantalla Modo Potencia

NOTA: EN UN ARCHIVO DENTRO DEL CD DE TESIS SE PRESENTA EL DOCUMENTO EXCEL GENERADO POR EL EQUIPO CON NOMBRE: "POTXX"

IV. DIAGRAMA DE BLOQUES DEL PROGRAMA ARMONICOS



Flujograma 4.3.a Armónicos



Flujograma 4.3.b Armónicos

a. FUNCIONAMIENTO

Se acede a la tarea desde el Menú principal del sistema, este tiene como función principal trabajar con la transformada de Fourier para el análisis de los armónicos de la línea Alterna.

Entre sus diferentes tareas a ejecutar, este algoritmo puede mostrar los armónicos de la señal de voltaje y corriente y a partir de estos armónicos el equipo calcula por convolucion frecuencial los armónicos de la señal de potencia.

Entre los diferentes resultados que el algoritmo muestra en la pantalla tenemos, valor RMS, THD, Energía aportada por el armónico, frecuencia de línea, frecuencia del armónico.

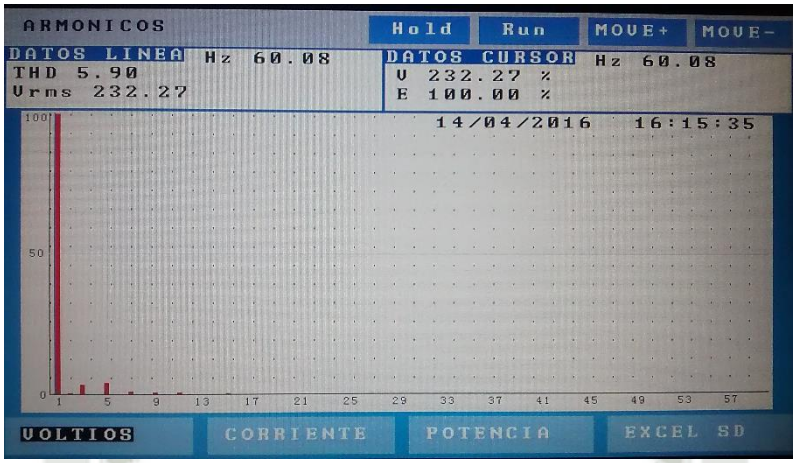
En la parte inferior de la pantalla cuenta con botones VOLTAJE, CORRIENTE, POTENCIA Y GRABAR SD.

Al presionar sobre uno de ellos, se selecciona los armónicos a calcular, en cuanto a la opción GRABAR SD, el algoritmo almacena un vector con los valores del espectro y los parámetros mencionados anteriormente en formato EXCEL en la memoria SD del equipo.

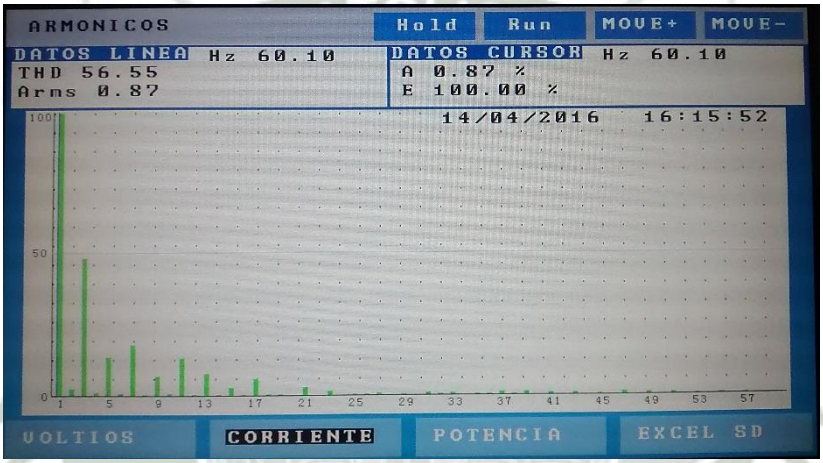
Se puede acceder a la información desde una PC, donde se puede recrear el grafico del espectro en la hoja de Excel.

NOTA: EN UN ARCHIVO DENTRO DEL CD DE TESIS SE PRESENTA EL DOCUMENTO EXCEL GENERADO POR EL EQUIPO CON NOMBRE: "HARXX"

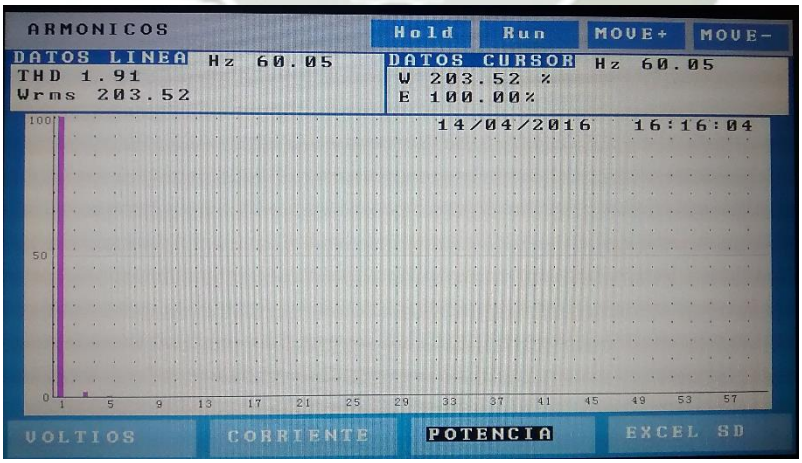
b. PRUEBAS DE FUNCIONAMIENTO



Esquema 4.35 Armónicos Voltaje

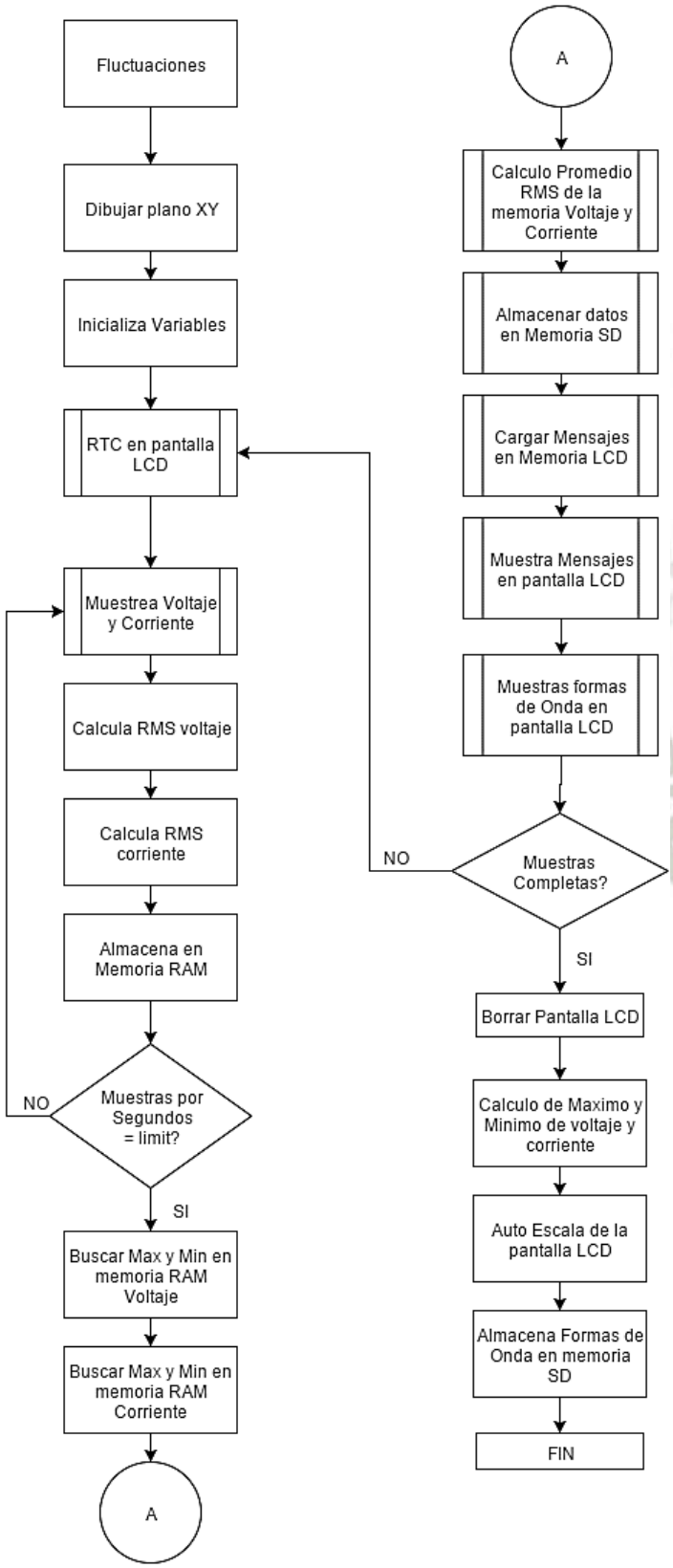


Esquema 4.36 Armónicos Corriente



Esquema 4.37 Armónicos Potencia

V. DIAGRAMA DE BLOQUES DEL PROGRAMA FLUCTUACIONES



Flujograma 4.4 Fluctuaciones

a. FUNCIONAMIENTO

Se acede a la tarea desde el Menú principal del sistema, este tiene como función principal medir las fluctuaciones producidas en el voltaje y la corriente en la linea AC, Durante este modo el equipo representa los parámetros periódicamente a lo largo de lo que se denomina “Intervalo de Trazado”.

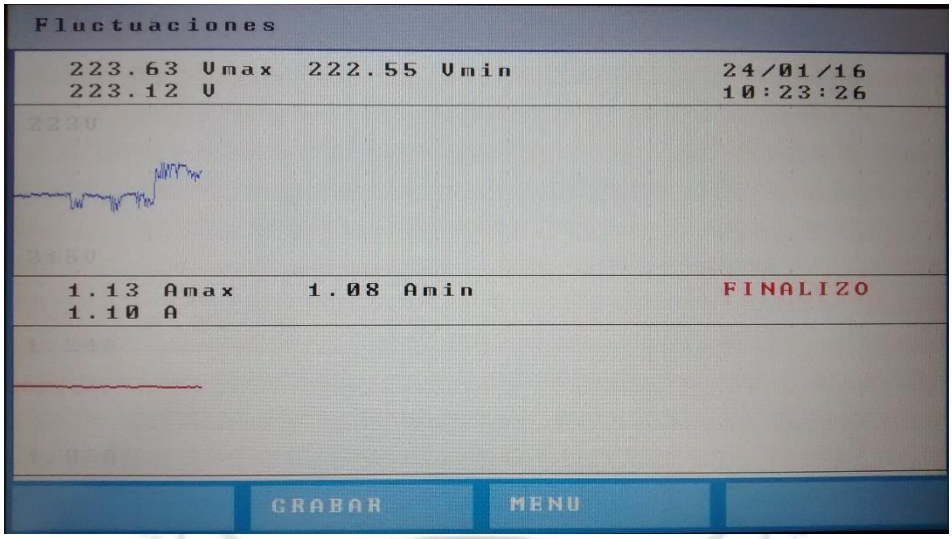
El tiempo de Registro es el tiempo total que le toma al equipo completar todo el ancho de la pantalla con las grafica de fluctuaciones.

En la siguiente tabla se da a conocer la relación entre un tiempo de registro y el intervalo de trazado, para esto se tiene en consideración de que siempre se grafican 800 puntos a lo ancho de la pantalla.

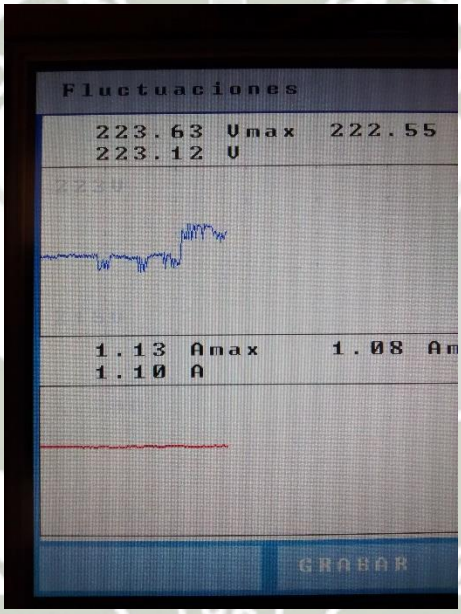
Tiempo de Registro (Minutos)	**Intervalo de trazado (Segundos)**
12	**1**
26	**2**
52	**4**
1 H 15 min	**8**
3H 30min	**16**
7H –Min	**32**

El equipo toma en cuenta todas las mediciones realizadas durante un intervalo de trazado y registra la lectura máxima, mínima y promedio de la señal de voltaje y corriente.

b. PRUEBAS DE FUNCIONAMIENTO



Esquema 4.38 Grafica de Fluctuaciones Voltaje y Corriente

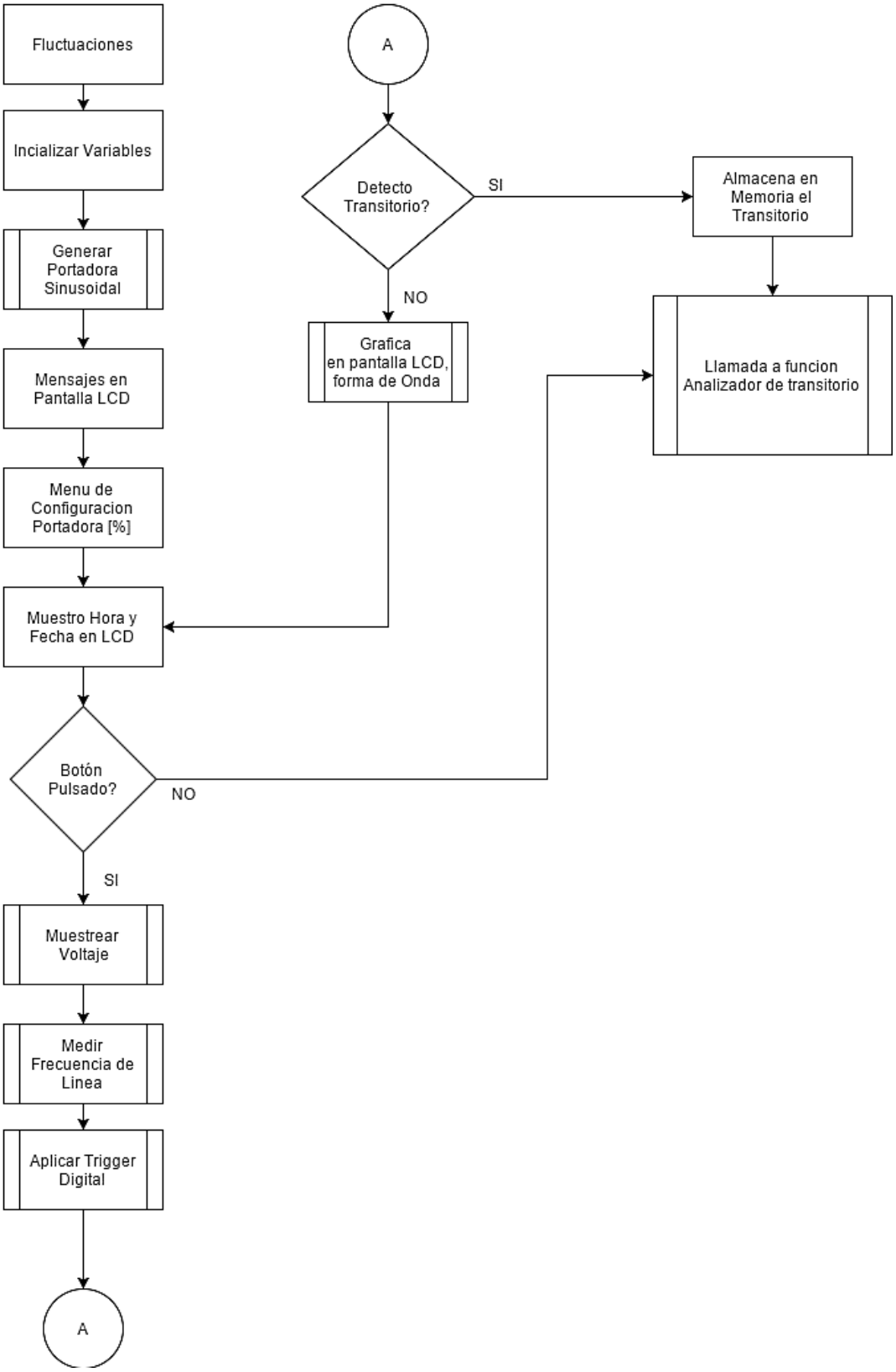


Esquema 4.39 Datos Mostrados durante la medición en fluctuación

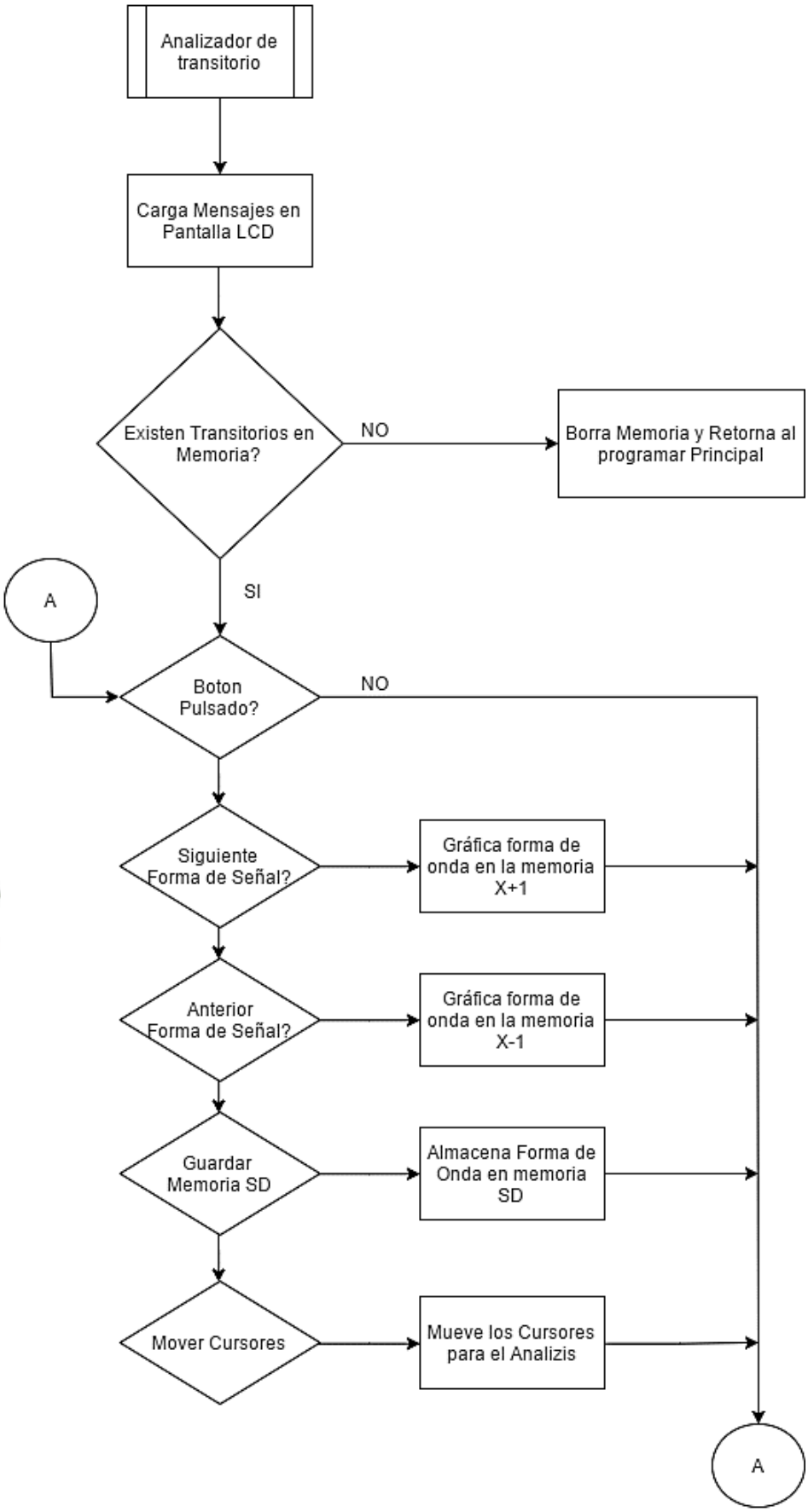
El Equipo puede registrar las formas de onda y almacenarlas en la memoria SD en formato Excel.

NOTA: EN UN ARCHIVO DENTRO DEL CD DE TESIS SE PRESENTA EL DOCUMENTO EXCEL GENERADO POR EL EQUIPO CON NOMBRE: "FLUCXX"

VI. DIAGRAMA DE BLOQUES DEL PROGRAMA TRANSITORIOS



Flujograma 4.5.a Transitorios



Flujograma 4.5.b Transitorios

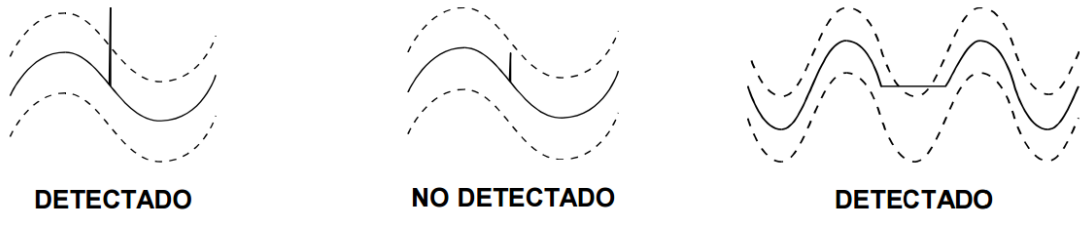
a. FUNCIONAMIENTO

Se acede a la tarea desde el Menú principal del sistema, este tiene como función principal Analizar los fenómenos transitorios los cuales son picos momentáneos y cambios rápidos en la señal de tensión.

Su algoritmo puede detectar picos en la señal de tensión y guardar la forma de onda en su memoria como una imagen de la señal,

Su principio de funcionamiento se basa cuando el transitorio supera una señal envolvente alrededor de la forma de onda de la tensión.

El ancho de la envolvente puede configurarse manualmente.

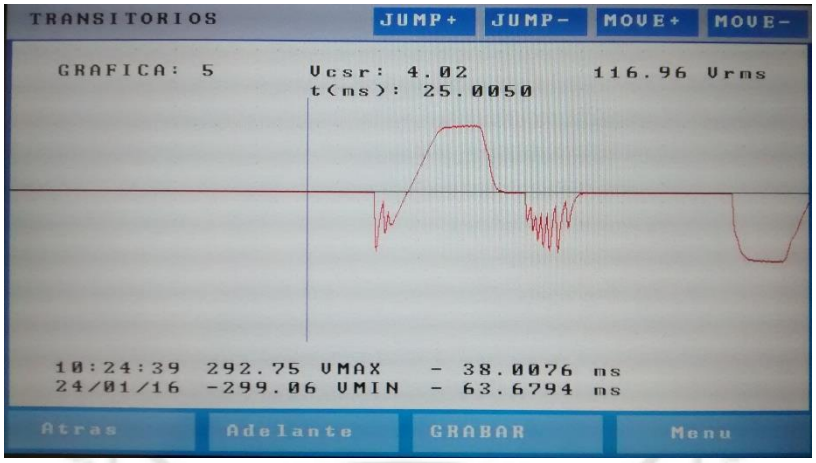


Esquema 4.40 Envolvente digital (Fluke 43b Manual de Usuario)

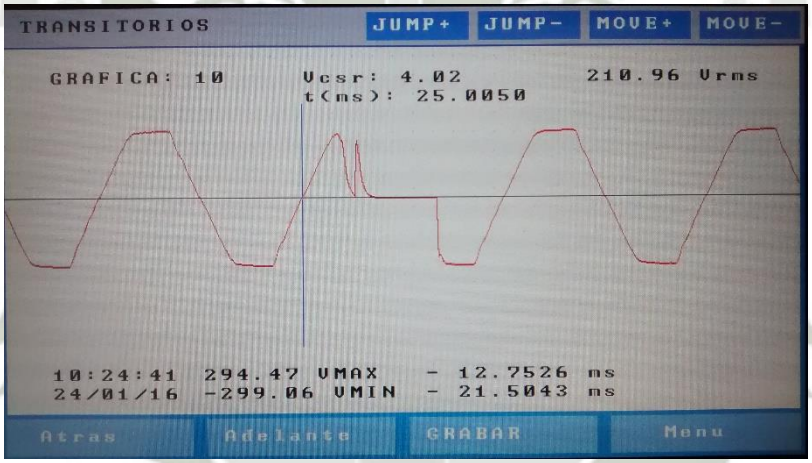
El equipo puede almacenar en su memoria hasta 20 formas de onda en su memoria RAM para luego poder ser almacenas en su memoria SD en formato EXCEL.

NOTA: EN UN ARCHIVO DENTRO DEL CD DE TESIS SE PRESENTA EL DOCUMENTO EXCEL GENERADO POR EL EQUIPO CON NOMBRE: "TRANCXX"

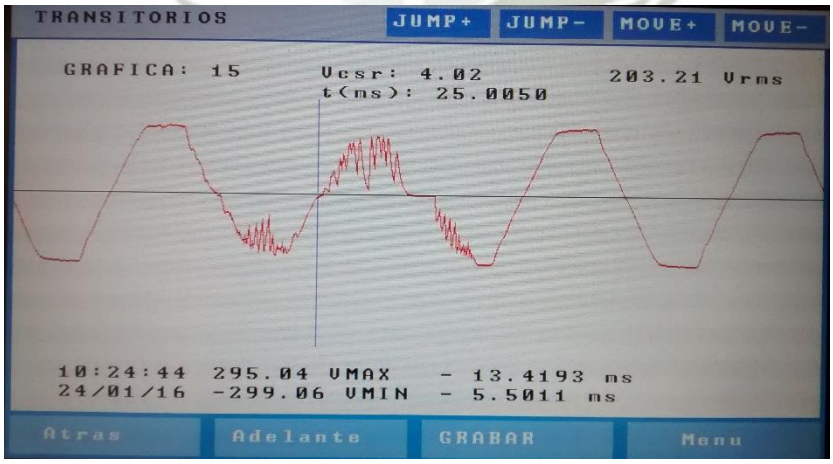
b. PRUEBAS DE FUNCIONAMIENTO



Esquema 4.41 Captura 1 de Transitorio

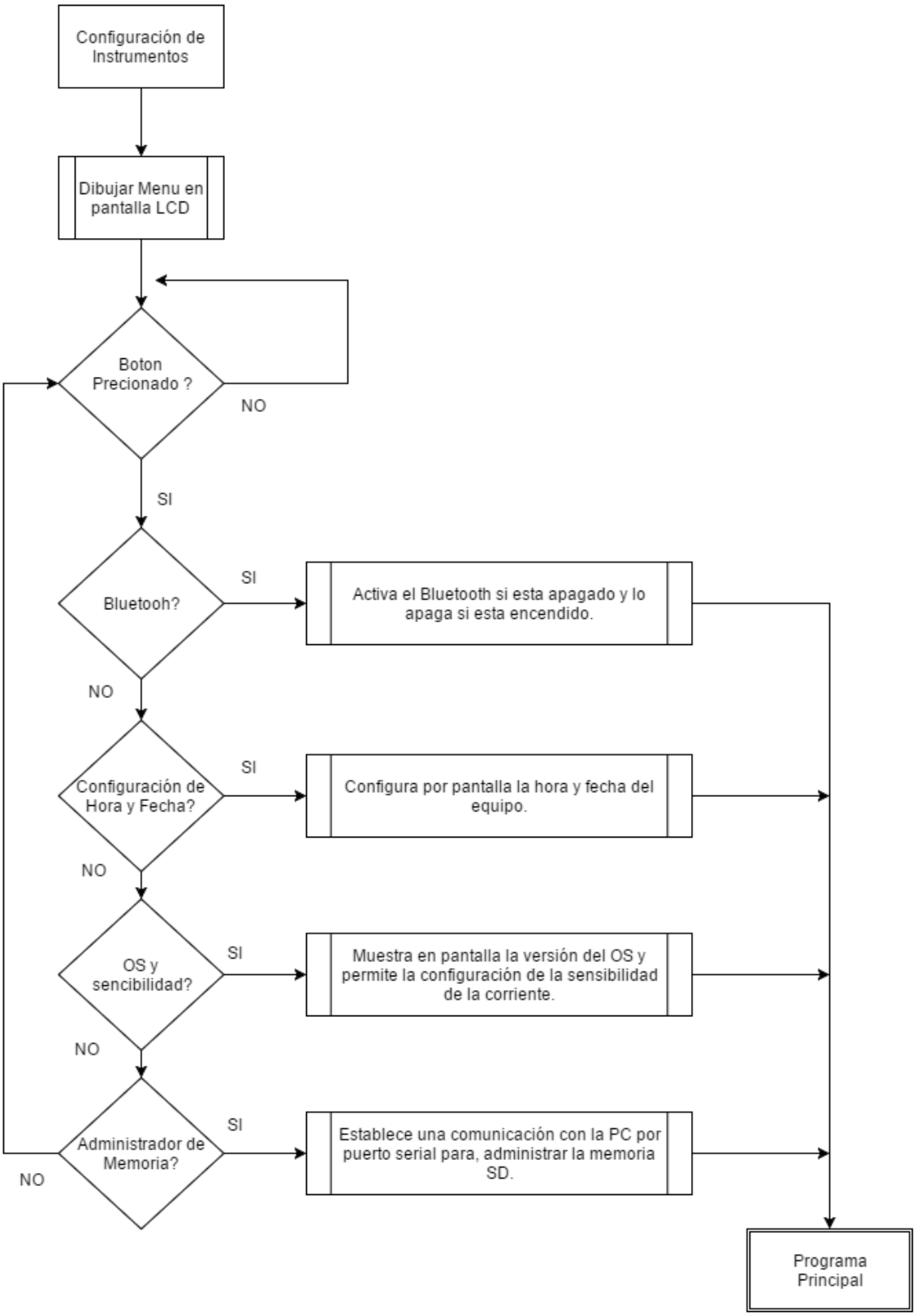


Esquema 4.42 Captura 2 de Transitorio



Esquema 4.43 Captura 3 de Transitorio

VII. DIAGRAMA DE BLOQUES DEL PROGRAMA CONFIGURACIONES DEL INSTRUMENTO

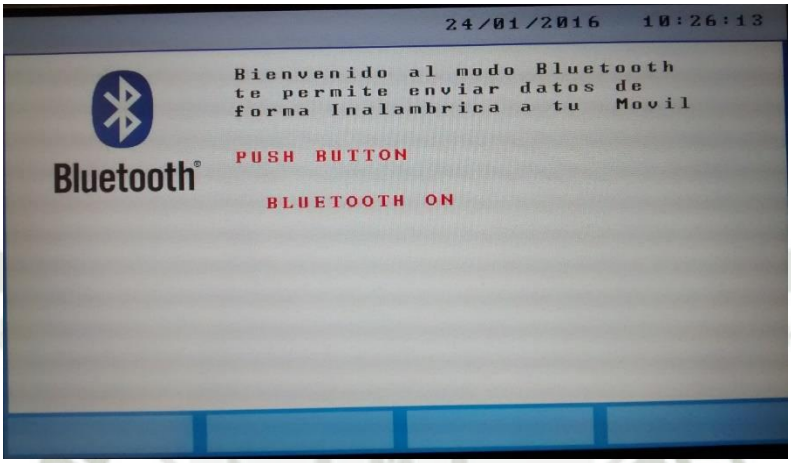


Flujograma 4.6 Menú de Configuraciones

a. FUNCIONAMIENTO

Se acede a la tarea desde el Menú principal del sistema, este tiene como función principal administrar las diferentes configuración del equipo, entre estos tenemos el encender y apagar el modo Bluetooth, fecha y hora y el rango para la corriente.

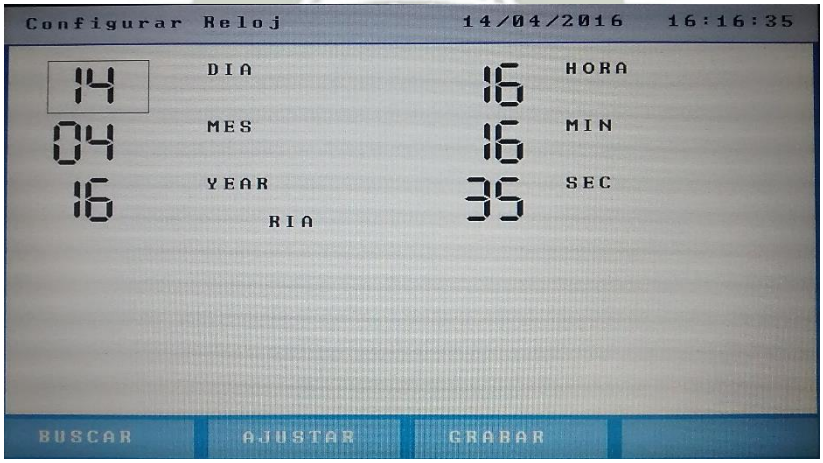
i. BLUETOOTH



Esquema 4.44 Pantalla de Bluetooth

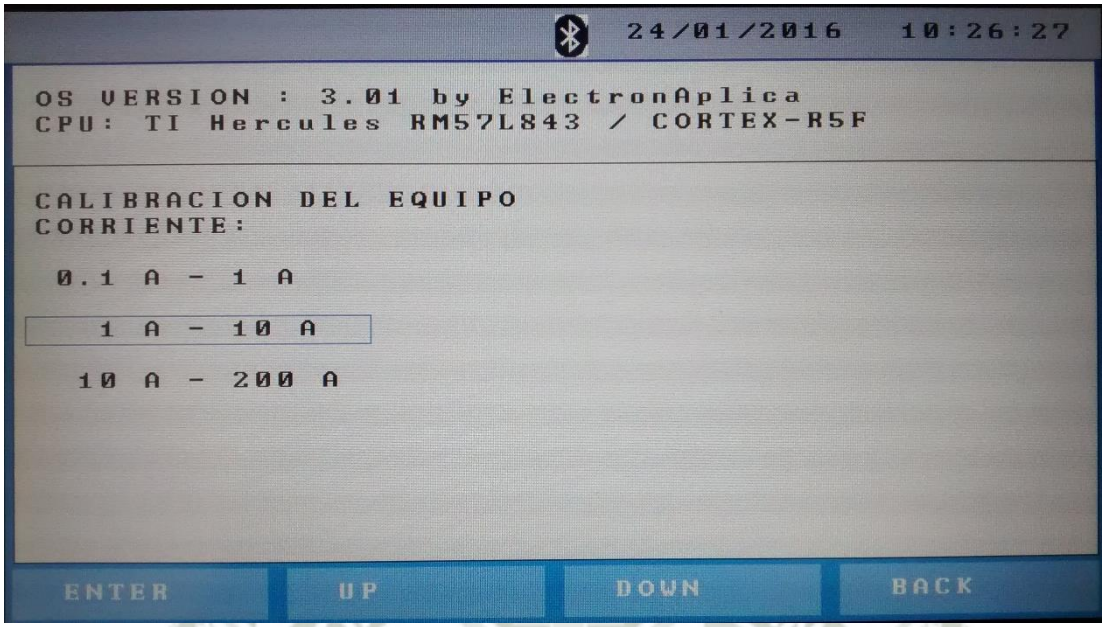
ii. FECHA Y HORA

El ajuste horario puede realizarse desde esta pantalla, donde se puede configurar el dia, mes, año, hora, minutos y los segundos se resetan automáticamente una vez fijada la nueva configuración de reloj.



Esquema 4.45 Pantalla de Reloj

iii. VERSION Y CALIBRACION



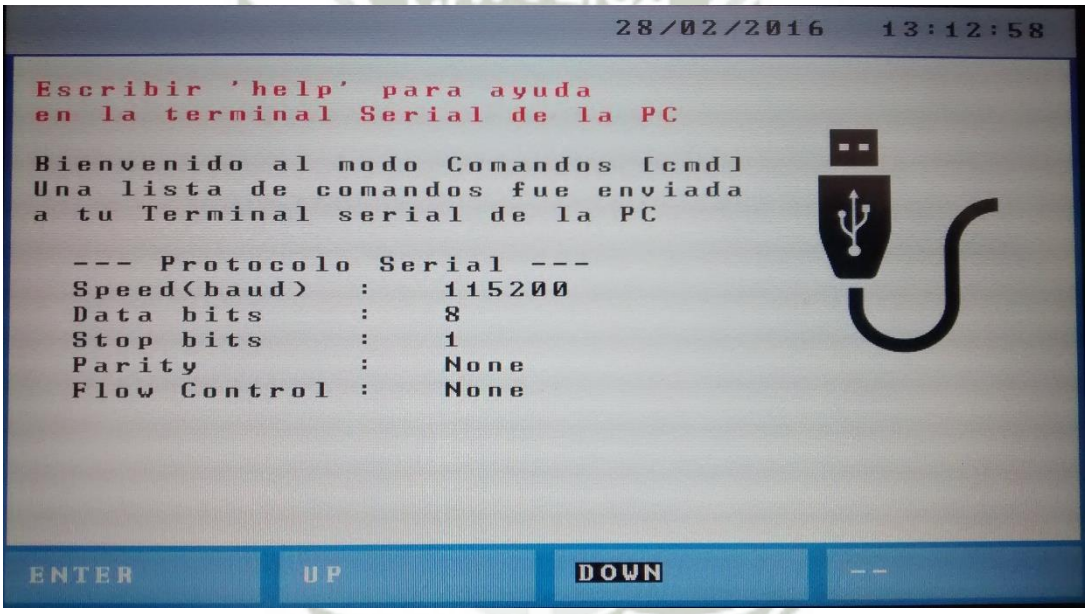
Esquema 4.46 Pantalla de calibración

VIII. SISTEMA DE ADMINISTRACION DE MEMORIA

El sistema de administración de memoria se ejecuta desde el terminal serial de Windows, utilizando comandos similares de Linux para poder acceder a la memoria SD y navegar entre sus diferentes archivos.

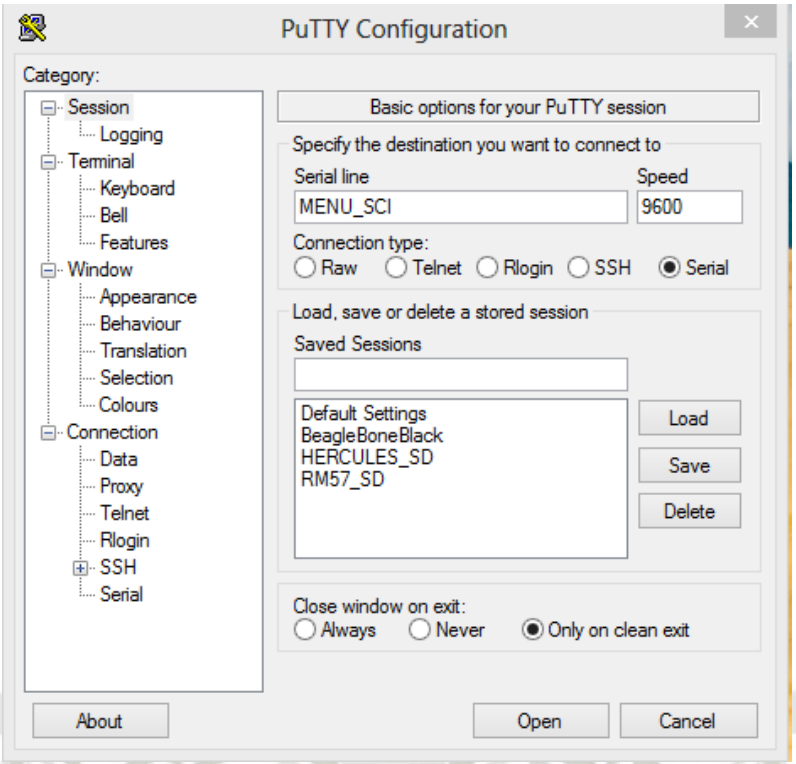
La ruta para ingresar al menú es:

INSTRUMENTS SETUP > ADMINISTRAR MEMORIA



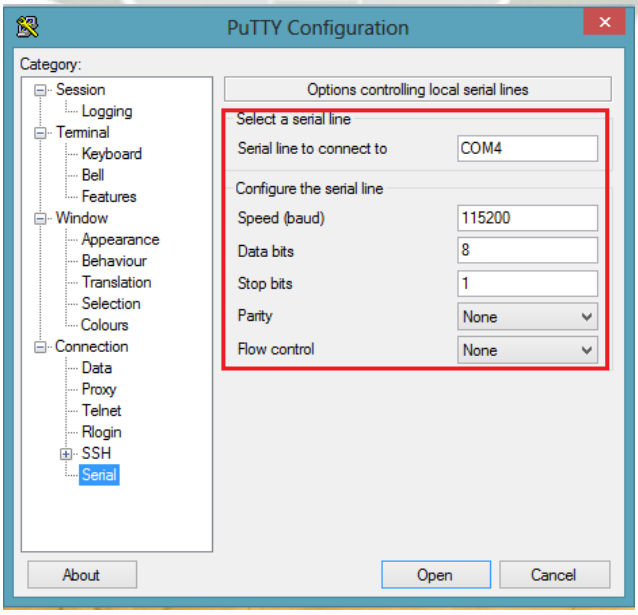
Esquema 4.47-1 Analizador de Calidad modo Terminal

Para entablar una comunicar entre la PC y el equipo se requiere de una terminal en la PC, en esta aplicación se utilizó el programa PuTTY.



Esquema 4.48-2 Terminal PuTTY en Windows

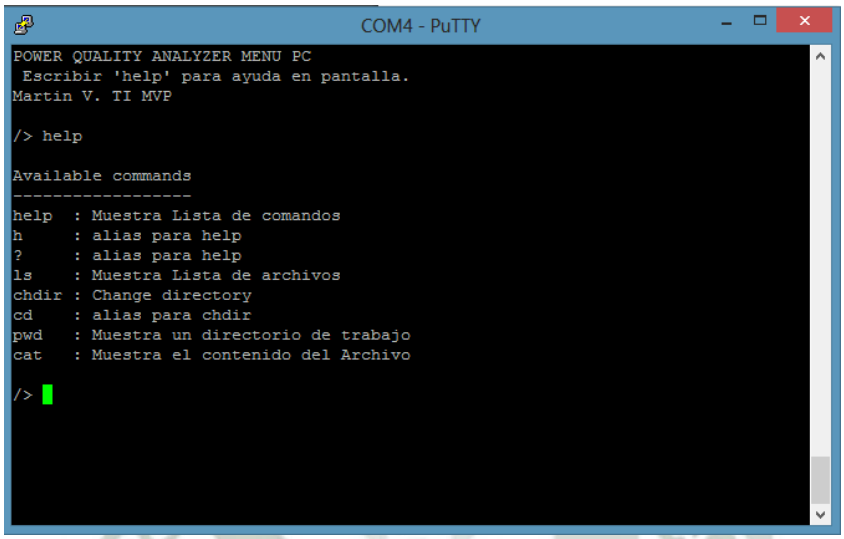
Se configura la terminal en la PC, con las características de comunicación que aparecen en la Esquema Esquema 4.49-3.



Esquema 4.49-3 Configuracion de PuTTY

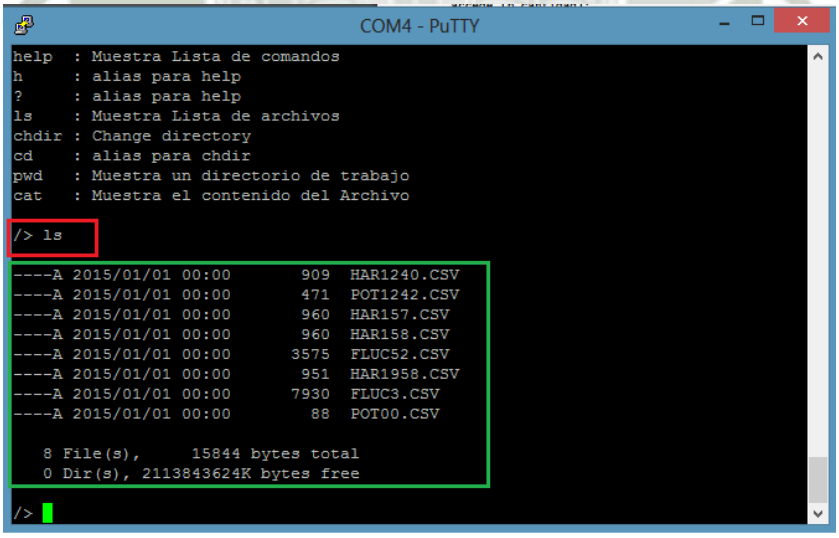
Con las siguientes configuraciones realizadas se procede a realizar la conexión, pulsando el botón “open”, esto desplegara una ventana con diferentes comandos.

En la Esquema 4.50, se puede observar los diferentes comandos que utiliza el administrador de memoria.



Esquema 4.50 Terminal Serial, Administrador de memoria.

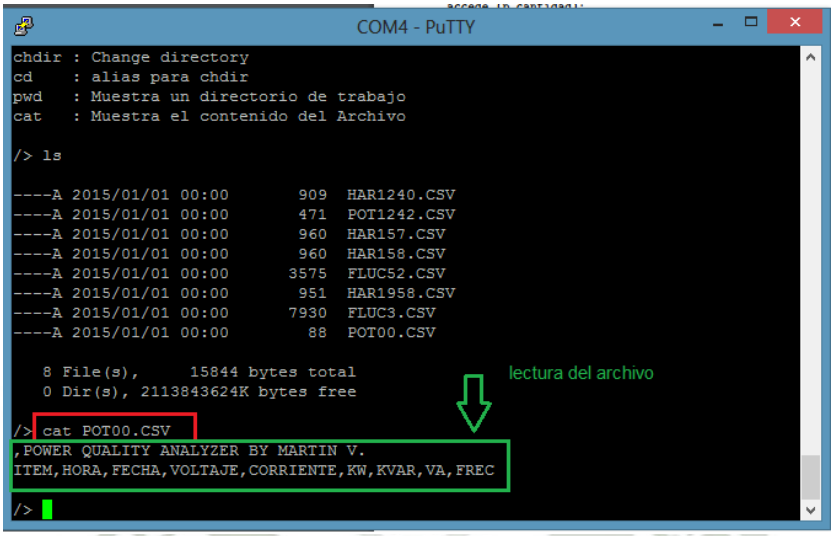
Se utiliza el comando “ls”, para poder mostrar una lista de archivos que se encuentran en la memoria.



Esquema 4.51 Comando “ls” y los archivos en memoria

El comando desplegar la lista de archivos dentro de memoria, la cantidad de espacio libre y ocupado que tiene la memoria SD.

Utilizando el comando “cat+nombre del archivo”, se puede acceder a la información del archivo, esto se puede observar en la figura 4.52



Esquema 4.52 Comando “cat”

El equipo almacena todos sus archivos sin la necesidad de crear carpetas, ya que este les genera un nombre especifico.

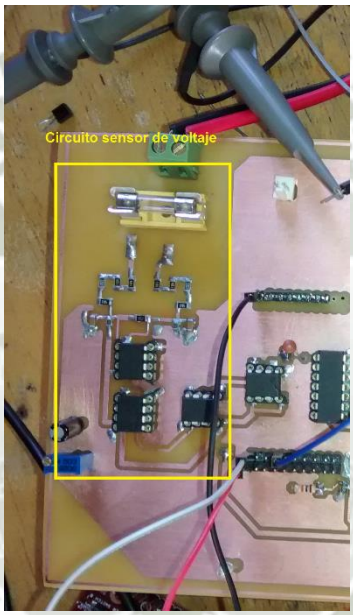
CAPITULO 5

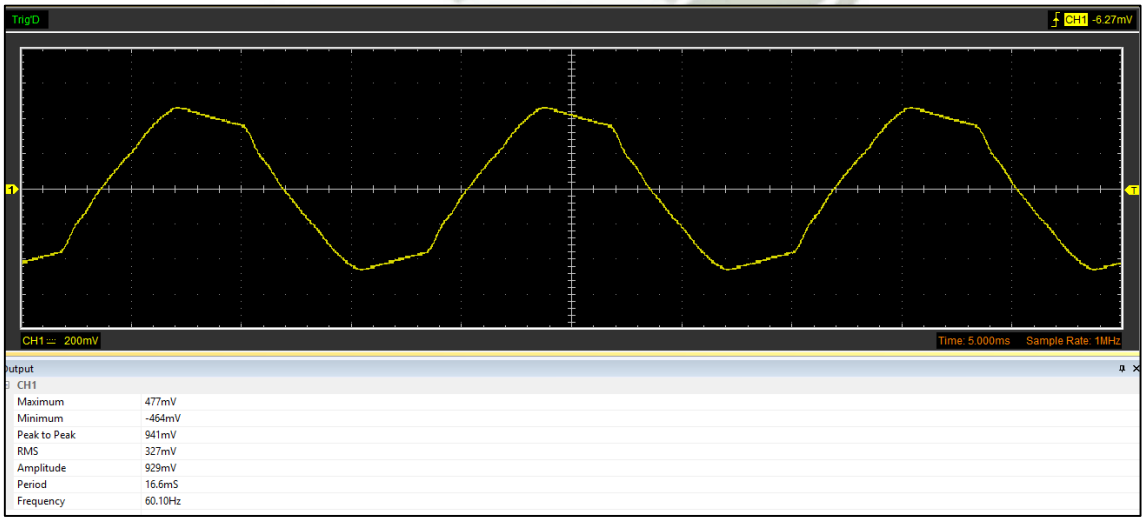
PRUEBAS Y RESULTADOS

En este capítulo se describen las pruebas realizadas en todos los circuitos diseñados para el Analizador de Calidad de Energía como también se muestran los resultados obtenidos, estas pruebas se realizan según la disposición de los circuitos.

I. CIRCUITO N°1 CIRCUITO DE MEDICION DE VOLTAJE

Se realizó la medición de la señal de voltaje AC, para observar que el circuito sensor de voltaje, actúa de acuerdo a lo diseñado en el esquema 5.1 que la señal de voltaje se redujo después la red de resistencias.

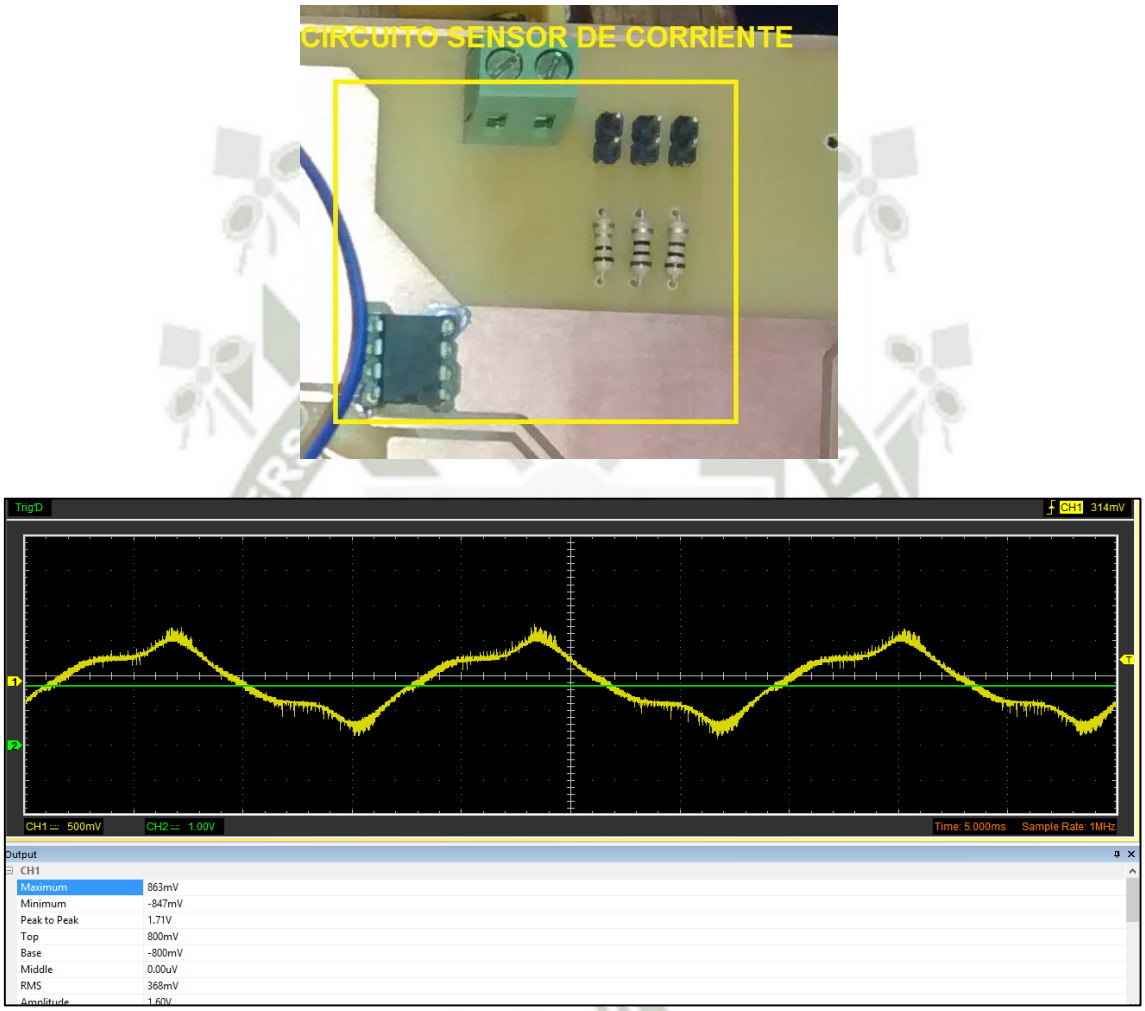




Esquema 5.1 Medición de la señal de voltaje

II. CIRCUITO N°2 CIRCUITO DE MEDICION DE CORRIENTE

Se realizó la medición de la señal de la corriente AC, para observar que las resistencias de escala funcionaran correctamente y poder visualizar la señal de salida en el osciloscopio, esto se puede observar en el Esquema 5.2.

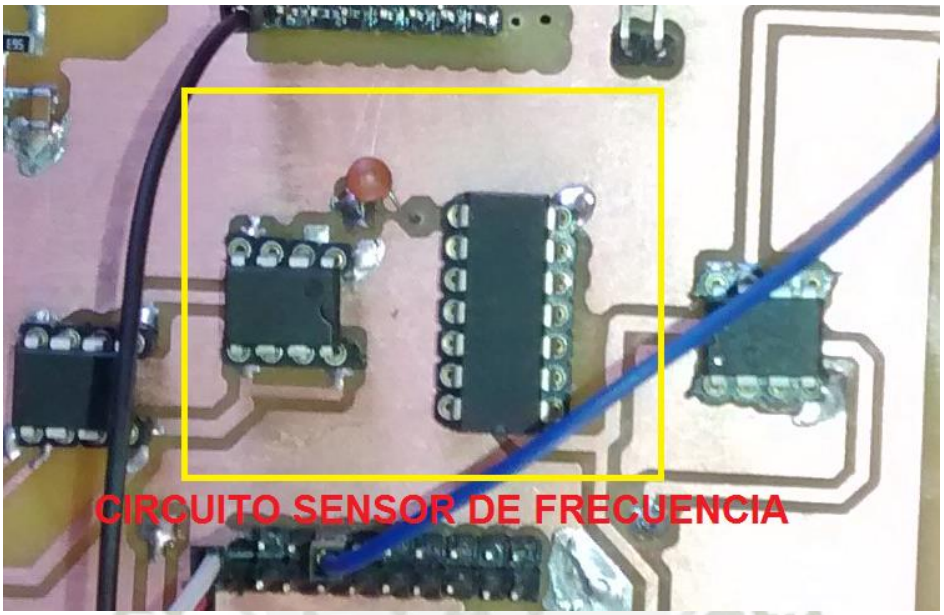


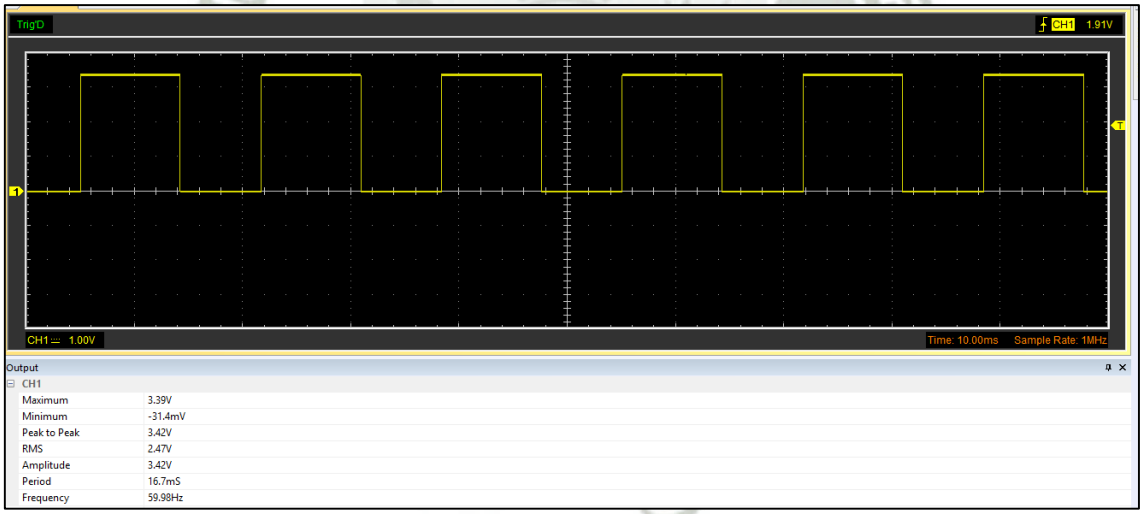
Esquema 5.2 Medición de la señal de Corriente

La señal de corriente alterna es sensado por la pinza amperimetrica Fluke, la cual atenúa la señal 1000 veces y luego por el circuito amplificador de le da una ganancia de 10 veces para luego ajustar los parámetros vía software, entregando la medición real que pasa por la pinza.

III. CIRCUITO N°3 CIRCUITO DE MEDICION DE FRECUENCIA

El circuito sensor de corriente se basa en un comparador y en compuertas lógicas inversoras, se utiliza la señal de voltaje como la fuente para la medición de frecuencia.



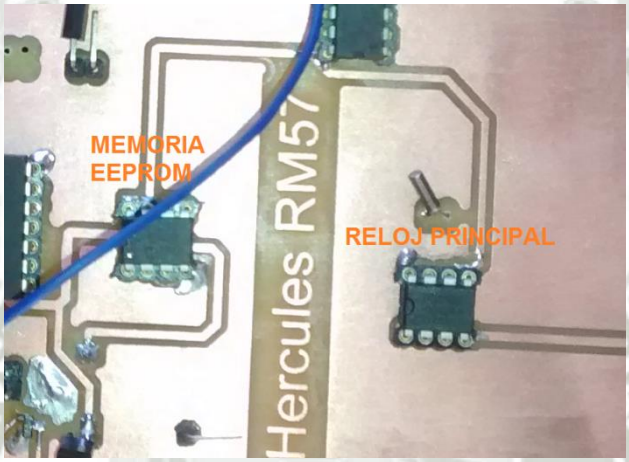


Esquema 5.3 Medición de frecuencia

En el esquema 5.3, se puede observar la señal en el osciloscopio de la forma de onda de la frecuencia.

IV. CIRUCITO N°4 CIRCUITO DE PERIFERICOS

En el esquema 5.4, se puede observar la implementación de la memoria Eeprom 25LC256 que es necesario para almacenar la configuración de operaciones principales del equipo y también se puede observar el IC de reloj principal DS1302, gracias a este el sistema puede almacenar la información con impresión de fecha y hora.



Esquema 5.4 Memoria Eeprom y Reloj

V. CIRCUITO N°5 CIRCUITO DE COMUNICACION BLUETOOTH

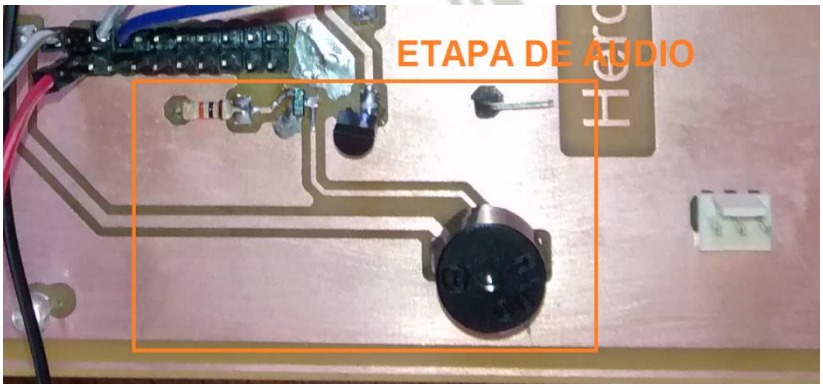
El modulo Bluetooth implementado es el HC-05 de montaje superficial.

El cual se soldó directamente en la placa principal por la zona inferior para que no ocupara espacio dentro de la caja que contiene el equipo, se comunica a través de comunicación Serial y tiene un led color azul que se enciende cada vez que esté está operativo.

Esquema 5.4 Modulo HC-05 Bluetooth

VI. CIRCUITO N°6 CIRCUITO INDICADOR AUDITIVO

El equipo Analizador de Calidad de Energía, tiene implementado un buzzer que sirve como alarma auditiva y aviso para los eventos que se generan.



Esquema 5.5 Buzzer para audio.

CAPITULO 6

CONCLUSIONES Y SUGERENCIAS

En este proyecto de tesis se diseñó e implemento completamente un Analizador de Calidad de Energia para lineas monofásicas. Se ha logrado tambien la integración de una interfaz con pantalla a colores táctil (Touch) en la que se puede visualizar formas de onda y acceder a las diferentes tareas que el equipo puede desarrollar. De las pruebas anteriores se puede concluir lo siguiente:

III. CONCLUSIONES

1. El proyecto se diseñó utilizando una tarjeta de desarrollo Hercules RM57 Launchpad de la marca Texas Instruments, el cual es un microcontrolador de gama alta, diseñado para aplicaciones industriales, la cual permite una alta precisión en la medición y procesamiento de parámetros eléctricos.

2. En cuanto a las técnicas de medición, para el muestreo de las señales alternas se utilizó satisfactoriamente las ecuaciones de Nyquist-Shannon, utilizando como ancho de banda la máxima frecuencia dada por el armonico 59 de la señal de corriente, las normas de los equipos comerciales solo concideran hasta el armonico 50.

3. El diseño de la interfaz gráfica para la pantalla LCD, se desarrollo utilizando un motor generador de gráficos (MGG), compuesto por diferentes algoritmos que fusionando sus resultados se obtiene una interfaz interactiva y de fácil uso para el usuario.

4. En cuanto a la medición de corriente, el cambio de resolución se realiza de forma manual dado que la señal es completamente alterna y el equipo funciona con una sola fuente por lo que no se puede realizar la escala utilizando algún componente que requiera una sola fuente, si se utilizaran un juego de interruptores para hacer el cambio de resistencias, el equipo aumentaría su consumo de corriente de la bateria, por lo que la mejor opción fue utilizar un selector manual.

5. Para obtener el espectro de frecuencia de la potencia se utilizo la ***teoría de convolucion en frecuencia***, utilizando los espectros del voltaje y la corriente con una multiplicación, obteniendo con esto una gran reducción de ciclos maquina del microcontrolador; dado que el microcontrolador varia su consumo de corriente de acuerdo a la tarea que realiza, la convolucion en el dominio de tiempo significava un alto consumo de corriente de la batería, por lo que la aplicación de la convolucion en frecuencia fue totalmente satisfactoria para el uso de un equipo que funciona a batería.

6. El equipo utiliza librerías FatFs R0.11 a, con un sistema de archivos FAT32 lo que permite utilizar memorias de alta capacidad como 8, 16, 32 GB en el equipo Analizador de Calidad de Energia sin presentar problemas en la compatibilidad de si la memoria es de rápida escritura u otras características en especial.

7. Se logro satisfactoriamente utilizar la transformada de Fourier para diferentes tareas, como encontrar el angulo de desfase entre señales alternas, espectros de frecuencia y factor de distorsion de corriente.

8. El analizador de calidad de energía no utiliza ningún tipo de filtro análogo para el tratamiento de la señal previa al microcontroaldor, dado que se debe procesar la señal en su forma original tanto en amplitud, forma de de onda y fase, sin ningún cambio de fase o alteración por parte de dichos filtros analogos, incluso los amplificadores utilizados en el proyectos presentan un voltaje offset muy bajo y ancho de banda mucho mayor al ancho de banda seleccionado para el equipo, por lo tanto estos tienen un comportamiento ideal dentro de este rango de trabajo del equipo.

9. El modo osciloscopio cuenta con una pequeña cantidad de tareas, por el hecho de que este equipo fue diseñado con un conversor Analogo digital de velocidad relativamente lenta con respecto a los conversores análogos con frecuencias de GHz, por lo que no es posible analizar tiempo de subida y bajada, entre otras tareas realizadas por un osciloscopio; pero para el analisis de armónicos en una red eléctrica de 50Hz o 60Hz, es mas que suficiente.

10. Los equipos comerciales utilizan 250 muestras de la señales de Voltaje y Corriente para realizar sus cálculos de RMS, se logro satisfactoriamente capturar 460 muestras de dicas señales obteniendo una mayor precisión en el calculo del valor RMS del Voltaje y la Corriente.

IV. SUGERENCIAS DE MODIFICACIONES

1. Dado que el motor generador de graficos es una respuesta de varios algoritmos trabajando para generar la interfaz, se sugiere unificar estos algoritmos en una sola matriz de tareas que permitan al usuario utilizar esto para otros trabajos con la pantalla grafica.

2. Se puede mejorar la etapa de medicion de corriente utilizando compuertas CD4066 y una fuente de alimentación Flyback con salidas de voltaje positivo y negativo para su correcta operación, de esta manera el cambio de resolución se realizara via software y de forma automática, esto haría mas complejos los algoritmos de medición.

REFERECIAS

1. Microchip.

   Amplificadores Operacionales MCP6022

2. Fairchild (2001)

   Comparador LM311

   Hex Schmitt trigger CD40106BC

3. Fluke

   Pinza Amperimetrica I800AC

4. Texas Instruments

   Microcontrolador ARM Cortex-R5F Dual Core

   Hercules RM57

5. Solomon Systech Semiconductor Technical Data

   Pantalla LCD de 7 Pulgadas Tactil de 16 bits

   SSD1963 LCD Display SRAM Controlador

6. MOV-20DxxxK Series - Metal Oxide Varistor

BIBLIOGRAFIA

1. Electrónica de Potencia

   Daniel W. Hart

   Editorial Prentice Hall

2. Fluke 43B Analizador Electrico Avanzado

   Manual de Uso

3. Fluke 434/435 Tres Fases Medidor de Calidad de Energia

   Manual de Uso

4. Perturbaciones en la red eléctrica

   Víctor Sánchez Huerta

5. HC Serial Bluetooth

   Manual de Instrucciones para el Usuario

6. A Single-Supply Op-Amp Circuit Collection (2000)

   Texas Instruments

7. EKG-Based Heart-Rate Monitor Implementation on the LaunchPad Using

   MSP430G2xx (2011)

   Texas Instruments

8. How to create a CMSIS DSP Library Project for Hercules in CCS

   Jan Cumps

   Launch Your Desing –Texas Instruments

9. Procesamiento Digital de Señales

   U.N.S. 2011

10. RM57Lx 16/32-Bit RISC Flash Microcontroller Technical Reference Manual

    Texas Instruments

APRENDICES

APRENDICE A
PROGRAMA DE MENU GENERAL

```
/* USER CODE BEGIN (0) */
/* USER CODE END */

/* Include Files */

#include "HL_sys_common.h"

/* USER CODE BEGIN (1) */
#include "HL_system.h"
#include "HL_het.h"
#include "HL_gio.h"
#include "HL_spi.h"
#include "HL_adc.h"
#include "HL_rti.h"
#include "HL_sci.h"
#include "math.h"
#include "stdio.h"
#include "TFT_LCD.h"
#include "stdlib.h"
#include "arm_math.h"
//#include "type_defs.h"
#include "EE25LC256.h"
#include "DS1302.h"
//#include "EE25LC256.h"
#include "TFT_menumain.h"

#include "HL_MMG.h"                    // Archivo de estructuras Principal
/* USER CODE END */

/** @fn void main(void)
 *   @brief Application main function
 *   @note This function is empty by default.
 *
 *   This function is called after startup.
 *   The user can use this function to implement the application.
 */

/* USER CODE BEGIN (2) */
```

```
char Tiempo_imprime[40] = "";
char Tiempo_imprime1[40] = "";
extern uint8_t BigFont[];

spiDAT1_t dataconfig1_t;

int flag = 1;
int day, mth, year, dow, hr, min, sec, rticont = 0;

extern const unsigned short LOGO_TI[];
extern const unsigned short logo_bluetooth[];

//extern const unsigned short PROYECTO[];

int spi_men[40];
void screen_clock();
//void Btn_TouchAction(btnMensajes Btntxt);

hetSIGNAL_t Duty_Period1;
double T_prom;

int get_adc_signal = 0;
uint8 Touch, Menu_CSelect = 0;

uint8 Mux_rti_1 = 0, Mux_rti_2 = 0;
boolean Refrescar_LCD = 0, Refrescar_LCD_POW = 0;
extern Modo_Osciloscopio Conf_Osciloscope;
boolean Bluetooh_ON_OFF = 0;
extern Harmonics_data Voltaje, Ampere, Potencia;
extern btnMensajes BtnMensaje;

/* USER CODE END */

void main(void) {
        /* USER CODE BEGIN (3) */

        hetInit();
        gioInit();
        tftInit(); /*Inicializa la pantalla LCD - Touch*/
        adcInit();
        spiInit();
        rtc_init(); //Iniacializa RTC
        sciInit(); /* initialize sci/sci-lin    */
        rtiInit();
```

```
        /* Enable RTI Compare 0 interrupt notification */
        rtiEnableNotification(rtiREG1, rtiNOTIFICATION_COMPARE0); // sirve para
capturar la frecuencia de linea

        rtiEnableNotification(rtiREG1, rtiNOTIFICATION_COMPARE3);
        /* Enable IRQ - Clear I flag in CPS register */
        /* Note: This is usually done by the OS or in an svc dispatcher */
        _enable_IRQ_interrupt_();

        /* Start RTI Counter Block 0 */
        rtiStartCounter(rtiREG1, rtiCOUNTER_BLOCK0);
        rtiStartCounter(rtiREG1, rtiCOUNTER_BLOCK1);
//      rtc_set_datetime(18,02,16, 0, 16, 47); //Comando para la configuracion de la hora
en el RTC

        /*dataconfig1_t.CS_HOLD = FALSE;
         dataconfig1_t.WDEL = TRUE;
         dataconfig1_t.DFSEL = SPI_FMT_0;*/
        spiDAT1_t dataconfig1_t;
        dataconfig1_t.CS_HOLD = TRUE;
        dataconfig1_t.WDEL = TRUE;
        dataconfig1_t.DFSEL = SPI_FMT_0;
        dataconfig1_t.CSNR = 0xFE;

        arm_fill_f32(0.00, &Voltaje.wave_sampling, TEST_LENGTH_SAMPLES);
        arm_fill_f32(0.00, &Ampere.wave_sampling, TEST_LENGTH_SAMPLES);
        arm_fill_f32(0.00, &Potencia.wave_sampling, TEST_LENGTH_SAMPLES);

        gioSetBit(gioPORTA, 7, 1);          //Buzzer Pin control
        MMG_BackFond();

        MMG_MenuInit();

        /*Inicializo los valores de los mensajes en botones*/
        sprintf(BtnMensaje.Btn1, "ENTER");
        sprintf(BtnMensaje.Btn2, "UP");
        sprintf(BtnMensaje.Btn3, "DOWN");
        sprintf(BtnMensaje.Btn4, "--");
        BtnMensaje.GetValue = 0;    // Ningun Boton Selecionado
        Btn_TouchAction(&BtnMensaje);

        setColor(0x77, 0x99, 0xdd);
        setBackColor(0xff, 0xff, 0xff);
```

```
        drawBitmap(400, 100, 209, 164, LOGO_TI, 1);
        setColor(0x22, 0x77, 0xbb);
        drawRect(15, 58, 380, 58 + 22);
        pwmSetDuty(hetRAM2, pwm0, 50);          // POTENCIA DEL BACKIGHT
LCD

        /*
         TFT_print("UP", 240, 445, 0);
         TFT_print("DOWN", 440, 445, 0);
         TFT_print("--", 640, 445, 0);*/

        gioSetBit(gioPORTA, 7, 0);

        Bluetooh_ON_OFF = Read_25LC(40);
        if (Bluetooh_ON_OFF == 1) {
                drawBitmap(400, 5, 24, 33, logo_bluetooth, 1);
                gioSetBit(hetPORT1, 26, 0);          //Bluetooth Pin_control
        }

        /* Conf_Osciloscope.Corriente_factor=1;
         Conf_Osciloscope.Rango_conf.Amperio_A=500;
         Conf_Osciloscope.Rango_conf.Voltaje_V=1104;
         Conf_Osciloscope.Inteporl_SENSIBILIDAD=5;


WRITE_FLOAT_EXT_EEPROM(1,(float)Conf_Osciloscope.Corriente_factor);

WRITE_FLOAT_EXT_EEPROM(10,(float)Conf_Osciloscope.Rango_conf.Amperio_
A);

WRITE_FLOAT_EXT_EEPROM(20,(float)Conf_Osciloscope.Rango_conf.Voltaje_V)
;

WRITE_FLOAT_EXT_EEPROM(30,(float)Conf_Osciloscope.Inteporl_SENSIBILID
AD);*/


        /*  Sectores de memoria Interna
         * DIRECCION              DATO
        TAMAÑO(BYTES)
         * 1                      FACTOR DE CORRIENTE          4
         * 10                     AMPERIO_A                    4
         * 20                     VOLTAJE_A                    4
         * 30                     SENSIBILIDAD_INTEPO          4
```

```
 * 40                    BLUETOOTH
1              */

while (1) {

        flag = digitalRead(IRQ);
        if (flag == 0) {

                Touch = Touch_Key();

                if (Touch == 2) {
                        if (Menu_CSelect > 0)
                                Menu_CSelect--;
                        else
                                Menu_CSelect = 0;

                } else if (Touch == 3) {
                        if (Menu_CSelect < 6)
                                Menu_CSelect++;
                        else
                                Menu_CSelect = 6;
                }

                if (Touch == 1) {
                        setColor(0xff, 0xff, 0xff);
                        fillRect(400, 100, 400 + 209, 100 + 164);
                        MMG_Menu_deled();

                        if (Menu_CSelect == 0)
                                VOLTS_AMPS_HERT();
                        else if (Menu_CSelect == 1)
                                POWER();
                        else if (Menu_CSelect == 2)
                                HARMONICS();
                        else if (Menu_CSelect == 3)
                                SAGS_SWELLS();
                        else if (Menu_CSelect == 4)
                                TRANSIENTS();
                        else if (Menu_CSelect == 5)
                                SCOPE();
                        else if (Menu_CSelect == 6)
                                INTRUMENTS_SETUP();

                }
```

```
                    BtnMensaje.GetValue = Touch;       // Ningun Boton Selecionado
                    Btn_TouchAction(&BtnMensaje);

                    MMG_MenuMain();
                    MMG_SelectMenu(Menu_CSelect);
                }
                screen_clock();

        }

        /* USER CODE END */
}
```

APENDICE B
PROGRAMA DE VOLTAJE, AMP Y HERTZ

```
void VOLTS_AMPS_HERT() {

    int LENGTH_SAMPLES = 1600;

    /* ---------------------------------------------------------------
     ** Limpiar la parte superior de la pantalla con el
     ** fin de mostrar el nuevo mensaje
     ** ---------------------------------------------------------------- */
    setColor(0xd2, 0xd2, 0xd2);
    fillRect(0, 0, 799, 40);

    setColor(0, 0, 0);
    setBackColor(0xd2, 0xd2, 0xd2);
    TFT_print("VOLTS/AMPS/HERTZ", 20, 10, 0);
    /* ---------------------------------------------------------------
     **
     ** ---------------------------------------------------------------- */
    setColor(0x00, 0x00, 0x00);
    drawRect(5, 63, 795, 233);
    drawRect(5, 255, 795, 425);

    /* ---------------------------------------------------------------
     **      Genera los cuadros superiores
     ** ---------------------------------------------------------------- */
    /*      setColor(0x77, 0x99, 0xdd);
     fillRect(365, 5, 465, 35);
     fillRect(475, 5, 575, 35);
     fillRect(585, 5, 685, 35);
     fillRect(695, 5, 795, 35);*/

    /* ---------------------------------------------------------------
     **      Inicializa la barra inferior del Menu para el control
     ** ---------------------------------------------------------------- */

    /*Inicializo los valores de los mensajes en botones*/
    sprintf(BtnMensaje.Btn1, "SETUP   ");
    sprintf(BtnMensaje.Btn2, "RANGO   ");
    sprintf(BtnMensaje.Btn3, "TRIGGER ");
    sprintf(BtnMensaje.Btn4, " --     ");
    BtnMensaje.GetValue = 0;    // Ningun Boton Selecionado
    Btn_TouchAction(&BtnMensaje);
```

```
        /* ------------------------------------------------------------------
         **      Valores Iniciales del equipo
         ** ------------------------------------------------------------------ */

        Conf_Osciloscope.Corriente_factor = (int) READ_FLOAT_EXT_EEPROM(1);
        Conf_Osciloscope.Rango_conf.Amperio_A = (int)
READ_FLOAT_EXT_EEPROM(10);
        Conf_Osciloscope.Rango_conf.Voltaje_V = (int)
READ_FLOAT_EXT_EEPROM(20);
        Conf_Osciloscope.Inteporl_SENSIBILIDAD = (int)
READ_FLOAT_EXT_EEPROM(30);

        Conf_Osciloscope.Trigger_on_off = 0;
        Conf_Osciloscope.HOLD_RUN = 1;
        Conf_Osciloscope.Submenus = 0;

        Conf_Osciloscope.Rango_conf.Tiempo_s = 0.10418e-3;
        set_sample_period(Conf_Osciloscope.Rango_conf.Tiempo_s);

        int pixel_sensibilidad = Conf_Osciloscope.Inteporl_SENSIBILIDAD;
        Bluetooh_ON_OFF = Read_25LC(40);
        if (Bluetooh_ON_OFF == 1) {
                drawBitmap(400, 5, 24, 33, logo_bluetooth, 1);
                gioSetBit(hetPORT1, 26, 0);          //Bluetooth Pin_control
        }
        int Touch = 0;

        while (1) {

                if (digitalRead(IRQ) == 0) {
                        Touch = Touch_Key();

                        if (Touch == 1) {
                                Conf_Osciloscope.Submenus = 1;
                                Setup_scope();


                        } else if (Touch == 2) {
                                Conf_Osciloscope.Submenus = 2;
                                Rango_scope();


                        } else if (Touch == 3) {
                                Conf_Osciloscope.Submenus = 3;
```

```
                    Trigger_scope();

                } else if (Touch == 4) {
                    //Conf_Osciloscope.Submenus = 4;
                    //main();

                } else if (Touch == 5) {
                    if (Conf_Osciloscope.Submenus == 1) {
                        Conf_Osciloscope.Rango_conf.Tiempo_s -=
0.005e-3;
                        if (Conf_Osciloscope.Rango_conf.Tiempo_s < 0)
                            Conf_Osciloscope.Rango_conf.Tiempo_s =
0.01e-3;

        set_sample_period(Conf_Osciloscope.Rango_conf.Tiempo_s);
                    }
                    if (Conf_Osciloscope.Submenus == 2)
                        Conf_Osciloscope.Rango_conf.Voltaje_V -= 50;

                } else if (Touch == 6) {
                    if (Conf_Osciloscope.Submenus == 1) {
                        Conf_Osciloscope.Rango_conf.Tiempo_s +=
0.005e-3;

        set_sample_period(Conf_Osciloscope.Rango_conf.Tiempo_s);
                    }

                    if (Conf_Osciloscope.Submenus == 2)
                        Conf_Osciloscope.Rango_conf.Voltaje_V += 50;

                } else if (Touch == 7) {
                    if (Conf_Osciloscope.Submenus == 1)
                        Conf_Osciloscope.HOLD_RUN = 0;
                    if (Conf_Osciloscope.Submenus == 2)
                        Conf_Osciloscope.Rango_conf.Amperio_A -= 50;
                    else if (Conf_Osciloscope.Submenus == 3)
                        Conf_Osciloscope.Trigger_on_off = 1;

                } else if (Touch == 8) {
                    if (Conf_Osciloscope.Submenus == 1)
                        Conf_Osciloscope.HOLD_RUN = 1;
                    if (Conf_Osciloscope.Submenus == 2)
                        Conf_Osciloscope.Rango_conf.Amperio_A += 50;
                    else if (Conf_Osciloscope.Submenus == 3)
```

```
                    Conf_Osciloscope.Trigger_on_off = 0;
                }

                BtnMensaje.GetValue = Touch;       // Ningun Boton Selecionado
                Btn_TouchAction(&BtnMensaje);

            }// if touch

            Touch = 0;

            if (Conf_Osciloscope.HOLD_RUN == 1) {

                /* ---------------------------------------------------------------
                 ** Captura de los conversor adc
                 ** --------------------------------------------------------------- */
                for (i = 0; i < LENGTH_SAMPLES; i++) {
                    adcStartConversion(adcREG1, adcGROUP1); //Start ADC
conversio
                    while                (!adcIsConversionComplete(adcREG1,
adcGROUP1))
                        ;       // ConfigutaR halcogen el trigger por
sofware

                    ch_count    =    adcGetData(adcREG1,    adcGROUP1,
&adc_data[0]);
                    ch_count = ch_count;
                    Voltaje.wave_sampling[i] = (adc_data[0].value
                            - adc_data[1].value) + 280;

                    adcStartConversion(adcREG2, adcGROUP1); //Start ADC
conversio
                    while                (!adcIsConversionComplete(adcREG2,
adcGROUP1))
                        ;       // ConfigutaR halcogen el trigger por
sofware

                    ch_count    =    adcGetData(adcREG2,    adcGROUP1,
&adc_data[0]);
                    ch_count = ch_count;
                    Ampere.wave_sampling[i]    =    adc_data[0].value    -
adc_data[1].value;

                    //delay_seg(40.69e-6);        // Retardo entre muestras de
83microsegundos
```

```
                while (get_adc_signal == 0) {
                }
                get_adc_signal = 0;

        }

        /* ---------------------------------------------------------------
         ** Tigger Digital por algoritmo
         ** ---------------------------------------------------------------- */
        int jh = 0, trigger_look = 0, diff_val = 0, centrotrigger =
                        LENGTH_SAMPLES / 2;
        int Dato_Actual = 0, Dato_Anterior = 0;
        if (Conf_Osciloscope.Trigger_on_off == 1) {

                do {
                        Dato_Actual                                     =
Voltaje.wave_sampling[centrotrigger + jh + 1];
                        Dato_Anterior                                   =
Voltaje.wave_sampling[centrotrigger + jh];
                                diff_val = Dato_Actual - Dato_Anterior;

                        if (Dato_Actual > 0 && Dato_Actual < 20 &&
diff_val > 15)

                                trigger_look = 1;
                        else {
                                trigger_look = 0;
                        }

                        jh++;
                } while ((trigger_look == 0) && (jh <
LENGTH_SAMPLES / 2));

        }

// Acondiciono el valor RMS de acuerdo a su ecuacion
        int sk = 0;
        for (i = jh; i < (LENGTH_SAMPLES / 2) + jh; i++) {
                float xy_acondition = Voltaje.wave_sampling[i] * 85
                                / Conf_Osciloscope.Rango_conf.Voltaje_V;
                int ypoint = (int) (xy_acondition + 150);
                if (ypoint < 65)
                        ypoint = 66;
                else if (ypoint > 230)
                        ypoint = 229;
```

```
                vol_xy_plot[sk] = ypoint;

                xy_acondition = Ampere.wave_sampling[i] * 50
                                /
Conf_Osciloscope.Rango_conf.Amperio_A;

                ypoint = (int) (xy_acondition + 340);
                if (ypoint < 256)
                        ypoint = 257;
                else if (ypoint > 423)
                        ypoint = 422;
                amp_xy_plot[sk] = ypoint;
                sk++;

                }

                /* ----------------------------------------------------------------
                 ** Calculo del Maximo y Minimo de la señal de voltaje
                 ** ---------------------------------------------------------------- */
                arm_max_f32(Voltaje.wave_sampling, LENGTH_SAMPLES,
                                &Voltaje.valor_MAX, &Voltaje.max_marks1);
                arm_min_f32(Voltaje.wave_sampling, LENGTH_SAMPLES,
                                &Voltaje.valor_MIN, &Voltaje.min_marks2);

                frec_line = 0.00;
                int Amplitud_wave = Voltaje.valor_MAX - Voltaje.valor_MIN;

                if (Amplitud_wave > 200)
                        frec_line = (float) (1e6 / T_prom);

                /* ----------------------------------------------------------------
                 ** Call the RMS function to calculate rms marks among
numStudents
                 ** ---------------------------------------------------------------- */
                arm_rms_f32(Voltaje.wave_sampling, LENGTH_SAMPLES,
                                &Voltaje.valor_RMS);
                Voltaje.valor_RMS = (Voltaje.valor_RMS * 3.3 / 4095) /
0.0014039;
                /*if (Voltaje.valor_RMS < 0.00)
                 Voltaje.valor_RMS = 0.00;*/

                arm_rms_f32(Ampere.wave_sampling, LENGTH_SAMPLES,
                                &Ampere.valor_RMS);
                Ampere.valor_RMS = (Ampere.valor_RMS * 3.3 / 4095)
```

```
                    * Conf_Osciloscope.Corriente_factor;

                sprintf(Mensajes_LCD.str_RMS,      "%3.2f      V            ",
Voltaje.valor_RMS);
                sprintf(Mensajes_LCD.str_Hz, "%2.2f Hz  ", frec_line);
                sprintf(Mensajes_LCD.str_Amplitud_harmonic, "%3.2f A  ",
                        Ampere.valor_RMS);
                //sprintf(Mensajes_LCD.str_Amplitud_harmonic,   "Tger=   %d
",jh);

                setColor(0, 0, 0);
                setBackColor(0xff, 0xff, 0xff);
                TFT_print(Mensajes_LCD.str_RMS, 50, 45, 0);
                TFT_print(Mensajes_LCD.str_Hz, 580, 45, 0);
                TFT_print(Mensajes_LCD.str_Amplitud_harmonic, 600, 236, 0);

                setColor(0xc0, 0xc0, 0xc0);
                TFT_print("0", 10, 150 - 20, 0);
                //TFT_print("210 V", 10, 150 - 62, 0);

                TFT_print("0", 10, 340 - 20, 0);
                //TFT_print("370 A", 10, 340 - 62, 0);

                setColor(0xd2, 0xd2, 0xd2);
                drawLine(6, 150, 794, 150);
                int j;
                for (j = 70; j < 230; j = j + 10) {
                        for (i = 200; i < 800; i += 400)
                                drawLine(i, j, i, j + 2);
                }
                for (i = 10; i < 800; i += 10) {
                        drawLine(i, 108, i, 108 - 2);
                        drawLine(i, 192, i, 192 + 2);
                }

                drawLine(6, 340, 794, 340);
                for (j = 250; j < 425; j = j + 10) {
                        for (i = 200; i < 800; i += 400)
                                drawLine(i, j, i, j + 2);
                }
                for (i = 10; i < 800; i += 10) {
                        drawLine(i, 298, i, 298 - 2);
                        drawLine(i, 382, i, 382 + 2);
                }
```

```
/*
 setColor(0xd2, 0xd2, 0xd2);
 drawLine(6, 150 - 42, 794, 150 - 42);
 drawLine(6, 150 + 42, 794, 150 + 42);

 setColor(0x00, 0x00, 0x00);
 drawLine(6, 340, 794, 340);
 setColor(0xd2, 0xd2, 0xd2);
 drawLine(6, 340 + 42, 794, 340 + 42);
 drawLine(6, 340 - 42, 794, 340 - 42);
 setColor(0x00, 0x00, 0x00);

 int x_axis_wave = 170;

 drawLine(x_axis_wave, 64, x_axis_wave, 232);
 drawLine(x_axis_wave, 256, x_axis_wave, 424);

 x_axis_wave = 535;

 drawLine(x_axis_wave, 64, x_axis_wave, 232);
 drawLine(x_axis_wave, 256, x_axis_wave, 424);
 */
int k, m_pendiente;
for (k = 6; k < 794; k++) {

        /* ------------------------------------------------------------
--
         ** Interpolacion de la señal para borrar en la pantalla
         **  ------------------------------------------------------------
- */
        setColor(0xff, 0xff, 0xff);
        drawPixel(k, vol_xy_buff[k]);
        drawPixel(k, vol_xy_buff[k] + 1);
        m_pendiente = abs(vol_xy_buff[k + 1] - vol_xy_buff[k]);
        if (m_pendiente > pixel_sensibilidad) {
                drawLine(k, vol_xy_buff[k], k + 1, vol_xy_buff[k +
1]);

                drawLine(k, vol_xy_buff[k] + 1, k + 1,
                                vol_xy_buff[k + 1] + 1);
        }

        drawPixel(k, amp_xy_buff[k]);
        drawPixel(k, amp_xy_buff[k] + 1);
```

```
                        m_pendiente = abs(amp_xy_buff[k + 1] -
amp_xy_buff[k]);
                        if (m_pendiente > pixel_sensibilidad) {
                                drawLine(k, amp_xy_buff[k], k + 1,
amp_xy_buff[k + 1]);
                                drawLine(k, amp_xy_buff[k] + 1, k + 1,
                                                amp_xy_buff[k + 1] + 1);
                        }
                        /* ----------------------------------------------------------------
--
                         ** Interpolacion de la señal para mostrar en la pantalla
                         ** ----------------------------------------------------------------
- */
                        setColor(0xff, 0, 0);
                        drawPixel(k, vol_xy_plot[k]);
                        drawPixel(k, vol_xy_plot[k] + 1);
                        m_pendiente = abs(vol_xy_plot[k + 1] - vol_xy_plot[k]);
                        if (m_pendiente > pixel_sensibilidad) {
                                drawLine(k, vol_xy_plot[k], k + 1, vol_xy_plot[k +
1]);
                                drawLine(k, vol_xy_plot[k] + 1, k + 1,
                                                vol_xy_plot[k + 1] + 1);
                        }
                        setColor(0, 0x00, 0xff);
                        drawPixel(k, amp_xy_plot[k]);
                        drawPixel(k, amp_xy_plot[k] + 1);
                        m_pendiente = abs(amp_xy_plot[k + 1] - amp_xy_plot[k]);
                        if (m_pendiente > pixel_sensibilidad) {
                                drawLine(k, amp_xy_plot[k], k + 1, amp_xy_plot[k
+ 1]);
                                drawLine(k, amp_xy_plot[k] + 1, k + 1,
                                                amp_xy_plot[k + 1] + 1);
                        }

                }
                /*Almacenamiento en buffer de datos para una proxima
comparacion*/
                for (i = 0; i < TEST_LENGTH_SAMPLES; i++) {
                        vol_xy_buff[i] = vol_xy_plot[i];
                        amp_xy_buff[i] = amp_xy_plot[i];
                }
        }
    }
}
```

APRENDICE C
PROGRAMA DE POTENCIA Y FRECUENCIA

```
void POWER() {

        rtiEnableNotification(rtiREG1, rtiNOTIFICATION_COMPARE1); // Sirve como reloj para refrescar la pantalla LCD
        int sample_cont = 0;            //Sirve para mostrar una distancia en el eje x
        int sample_hz = 0, y_sample_frec = 0, x_sample_frec;

//      setColor(0xff, 0xff, 0xff);
//      fillRect(6, 46, 794, 430);

        setColor(0xd2, 0xd2, 0xd2);
        fillRect(0, 0, 799, 40);
        /*Concidiones Iniciales para la grafica*/
        plotlcd.X_axis = 300;

        setFont(SmallFont);
        /*Recta superior para el eje de las ordenadas*/
        setColor(0x00, 0x00, 0x00);
        drawLine(plotlcd.X_axis, 320, 710, 320);
        drawLine(plotlcd.X_axis, 85, plotlcd.X_axis, 320);
        setBackColor(0xff, 0xff, 0xff);
        TFT_print("f(Hz)", plotlcd.X_axis - 15, 80, 0);
        TFT_print("t(ks)", 710, 320, 0);
        TFT_print("59", plotlcd.X_axis - 20, 295, 0);
        TFT_print("60", plotlcd.X_axis - 20, 195, 0);
        TFT_print("61", plotlcd.X_axis - 20, 95, 0);

        int yy = 0, ylines = 0;
        for (yy = 0; yy < 20; yy++) {
                drawLine(plotlcd.X_axis - 2, 100 + ylines, plotlcd.X_axis + 2,
                                100 + ylines);
                ylines += 10;
        }

        yy = 0, ylines = 0;
        for (yy = 0; yy < 41; yy++) {
                drawLine(plotlcd.X_axis + ylines, 318, plotlcd.X_axis + ylines, 322);
                ylines += 10;
        }

        setFont(BigFont);
```

```
setColor(0, 0, 0);
setBackColor(0xd2, 0xd2, 0xd2);
TFT_print("POTENCIAS", 20, 10, 0);

/*        setColor(255, 255, 255);
 setBackColor(0x77, 0x99, 0xdd);
 TFT_print(" AC  ", 40, 445, 0);
 TFT_print(" DC * ", 240, 445, 0);*/

/* ------------------------------------------------------------------
 **       Inicializa la barra inferior del Menu para el control
 ** ------------------------------------------------------------------ */

/*Inicializo los valores de los mensajes en botones*/
sprintf(BtnMensaje.Btn1, "TRIGGER");
sprintf(BtnMensaje.Btn2, "GRABAR ON ");
sprintf(BtnMensaje.Btn3, "GRABAR OFF");
sprintf(BtnMensaje.Btn4, " --  ");
BtnMensaje.GetValue = 0;    // Ningun Boton Selecionado
Btn_TouchAction(&BtnMensaje);

/*DIBUJO PANTALLA PRINCIPAL DE MEDIDAS*/
setColor(0x77, 0x99, 0xdd);

fillRect(25, 70, 260, 90);
fillRect(25, 170, 260, 170 + 20);
fillRect(25, 290, 260, 290 + 20);
drawRect(25, 70, 260, 330);

drawRect(275, 70, 770, 340);

setColor(255, 255, 255);
setBackColor(0x77, 0x99, 0xdd);
TFT_print("LINEA", 40, 72, 0);
TFT_print("POTENCIA", 40, 172, 0);
TFT_print("FRECUENCIA", 40, 292, 0);

Conf_Osciloscope.Corriente_factor = (int) READ_FLOAT_EXT_EEPROM(1);
if (Bluetooh_ON_OFF == 1) {
        drawBitmap(400, 5, 24, 33, logo_bluetooth, 1);
        gioSetBit(hetPORT1, 26, 0);          //Bluetooth Pin_control
}
```

```
float Period_sampling_wave = 0.488281e-3;          //0.04069us
set_sample_period(Period_sampling_wave);

int item = 0, flag_frec = 0;
Conf_Osciloscope.Trigger_on_off = 0;
Conf_Osciloscope.RECORD = 3;              // vALOR IMPOSIUBLE

arm_cfft_radix4_instance_f32 S;
/* Initialize the CFFT/CIFFT module */
arm_status status = arm_cfft_radix4_init_f32(&S, fftSize, ifftFlag,
                    doBitReverse);

while (1) {

          if (digitalRead(IRQ) == 0) {
                    Touch = Touch_Key();

                    if (Touch == 1) {
                              Trigger_scope();
                              Conf_Osciloscope.Submenus = 1;
                    } else if (Touch == 2) {
                              Conf_Osciloscope.RECORD = 1;

                    } else if (Touch == 3) {
                              Conf_Osciloscope.RECORD = 0;
                    } else if (Touch == 4) {
                              //main();

} else if (Touch == 5) {

                    } else if (Touch == 6) {

                    } else if (Touch == 7) {
                              if (Conf_Osciloscope.Submenus == 1)
                                        Conf_Osciloscope.Trigger_on_off = 1;
                    } else if (Touch == 8) {
                              if (Conf_Osciloscope.Submenus == 1)
                                        Conf_Osciloscope.Trigger_on_off = 0;

                    }
                    BtnMensaje.GetValue = Touch;       // Ningun Boton Selecionado
                    Btn_TouchAction(&BtnMensaje);

          }         // if touch
```

```
Touch = 0;
if (Refrescar_LCD_POW == 1) {
        /* ---------------------------------------------------------------
         ** Tigger Digital por cruce por cero
         ** ---------------------------------------------------------------- */

        if (Conf_Osciloscope.Trigger_on_off == 1) {
                loop0: if (gioGetBit(hetPORT2, 6) != 1)
                        goto loop0;
                loop1: if (gioGetBit(hetPORT2, 6) != 0)
                        goto loop1;
        }
        /* ---------------------------------------------------------------
         ** Captura de los conversor adc
         ** ---------------------------------------------------------------- */
        for (i = 0; i < TEST_LENGTH_SAMPLES; i++) {
                adcStartConversion(adcREG1, adcGROUP1); //Start ADC
conversio
                while                (!adcIsConversionComplete(adcREG1,
adcGROUP1))
                        ;               // ConfigutaR halcogen el trigger por
sofware

                ch_count     =     adcGetData(adcREG1,     adcGROUP1,
&adc_data[0]);
                ch_count = ch_count;
                Voltaje.wave_sampling[i] = (adc_data[0].value
                                - adc_data[1].value) + 280;

                adcStartConversion(adcREG2, adcGROUP1); //Start ADC
conversio
                while                (!adcIsConversionComplete(adcREG2,
adcGROUP1))
                        ;               // ConfigutaR halcogen el trigger por
sofware

                ch_count     =     adcGetData(adcREG2,     adcGROUP1,
&adc_data[0]);
                ch_count = ch_count;
                Ampere.wave_sampling[i]     =     adc_data[0].value     -
adc_data[1].value;

//delay_seg(40.69e-6);// Retardo entre muestras de 83microsegundos
```

```
                while (get_adc_signal == 0) {
                }
                get_adc_signal = 0;
        }
        /* ----------------------------------------------------------------
         ** Calculo del Maximo y Minimo de la señal de voltaje
         ** ---------------------------------------------------------------- */
        arm_max_f32(Voltaje.wave_sampling,
TEST_LENGTH_SAMPLES,
                        &Voltaje.valor_MAX, &Voltaje.max_marks1);
        arm_min_f32(Voltaje.wave_sampling,
TEST_LENGTH_SAMPLES,
                        &Voltaje.valor_MIN, &Voltaje.min_marks2);

        frec_line = 0.00;
        int Amplitud_wave = Voltaje.valor_MAX - Voltaje.valor_MIN;

        if (Amplitud_wave > 200)
                frec_line = (float) (1e6 / T_prom);    //RTI0,    cada    10ms
muestrea la frecuencia de linea.
        flag_frec = 1;
        if (frec_line > 61) {
                frec_line = 61;

        } else if (frec_line < 59)
                flag_frec = 0;

        /* ----------------------------------------------------------------
         ** Etapa para el calculo de los valores RMS (V,I)
         ** ---------------------------------------------------------------- */
        arm_rms_f32(Voltaje.wave_sampling,
TEST_LENGTH_SAMPLES,
                        &Voltaje.valor_RMS);
        // Acondiciono el valor RMS de acuerdo a su ecuacion
        Voltaje.valor_RMS   =   (Voltaje.valor_RMS   *   3.3   /   4095)   /
0.0014039;

        if (Voltaje.valor_RMS < 0.00)
                Voltaje.valor_RMS = 0.00;

        arm_rms_f32(Ampere.wave_sampling,
TEST_LENGTH_SAMPLES,
                        &Ampere.valor_RMS);
        Ampere.valor_RMS = (Ampere.valor_RMS * 3.3 / 4095)
```

```
                                * Conf_Osciloscope.Corriente_factor;
                        /* -----------------------------------------------------------------
                         ** Etapa para el calculo de la transformada discreta de fourier
                         ** Voltaje,Corriente y convolucion dominio frecuencia para potencia
                         ** ----------------------------------------------------------------- */

                        /*FFT de la señal de voltaje*/
                        arm_cfft_radix4_f32(&S, Voltaje.wave_sampling);/* Process the data through the CFFT/CIFFT module */
                        arm_copy_f32(Voltaje.wave_sampling, Voltaje.fft_PHASE, fftSize);/*Copio el vector en formato complejo para calculas posteriormente la fase*/
                        /*FFT de la señal de corriente*/
                        arm_cfft_radix4_f32(&S, Ampere.wave_sampling);/* Process the data through the CFFT/CIFFT module */
                        arm_copy_f32(Ampere.wave_sampling, Ampere.fft_PHASE, fftSize);/*Copio el vector en formato complejo para calculas posteriormente la fase*/

                        /* -----------------------------------------------------------------
                         ** Calculo del factor de distorcion FD
                         ** ----------------------------------------------------------------- */
                        arm_cmplx_mag_f32(Ampere.wave_sampling, Ampere.fft_MAG, fftSize);/* Process the data through the Complex Magnitude Module for calculating the magnitude at each bin */

                        arm_max_f32(Ampere.fft_MAG, fftSize / 2, &Potencia.valor_MAX,&Potencia.max_marks1);

                        int nxar=0;     // barre entre los armonicos de la señal
                        double FDsum=0;     // almacena la supersuma


        for(nxar=Potencia.max_marks1;nxar<8*Potencia.max_marks1;nxar+=Potencia.max_marks1)

        FDsum+=Ampere.fft_MAG[nxar]*Ampere.fft_MAG[nxar];// La operacion solo trabaja hasta el 8vo armonico

                        FDsum=sqrt(FDsum);

        Potencia_data.Factor_DISTORSION=Potencia.valor_MAX/FDsum;


                        /* -----------------------------------------------------------------
```

```
                    ** Algoritmo para buscar la fundamental en Fourier
                    ** ------------------------------------------------------------------ */
                    arm_max_f32(Voltaje.fft_PHASE, fftSize / 2,
&Voltaje.valor_MAX,
                                &Voltaje.max_marks1);
                    arm_max_f32(Ampere.fft_PHASE, fftSize / 2,
&Ampere.valor_MAX,
                                &Ampere.max_marks1);

                    /* ------------------------------------------------------------------
                    ** Calculo del angulo de fase y FP
                    ** ------------------------------------------------------------------ */
                    Voltaje.Harmonic_Angulo = (180 / 3.14159265)
                                * atan(

        Voltaje.fft_PHASE[Voltaje.max_marks1 + 1]
                                                    /
Voltaje.fft_PHASE[Voltaje.max_marks1]);
                    Ampere.Harmonic_Angulo = (180 / 3.14159265)
                                * atan(

        Ampere.fft_PHASE[Ampere.max_marks1 + 1]
                                                    /
Ampere.fft_PHASE[Ampere.max_marks1]);

                    Potencia_data.Angulo_fase = Voltaje.Harmonic_Angulo-
Ampere.Harmonic_Angulo;

                    Potencia_data.Factor_Potencia =
Potencia_data.Factor_DISTORSION*cos(Potencia_data.Angulo_fase * 3.14159265 /
180);

                    /* ------------------------------------------------------------------
                    ** Calculo de las Potencias Electricas
                    ** ------------------------------------------------------------------ */
                    Potencia_data.Potencia_Aparente = Voltaje.valor_RMS*
Ampere.valor_RMS;  // S=Vrms*Irms
                    Potencia_data.Potencia_Activa =
Potencia_data.Potencia_Aparente* Potencia_data.Factor_Potencia; //P=S*Fp

                    //Potencia_data.Potencia_Reactiva =
Potencia_data.Potencia_Aparente* sin(Potencia_data.Angulo_fase * 3.14159265 / 180);
//Q=S*sin(Angle)
```

```
Potencia_data.Potencia_Reactiva=sqrt(pow(Potencia_data.Potencia_Aparente,2)
-pow(Potencia_data.Potencia_Activa,2));

                    /* ----------------------------------------------------------------
                     ** Almacenamiento en Memoria SD 5 horas,33 minutos y 20
segundos
                     ** ---------------------------------------------------------------- */
                    if (Conf_Osciloscope.RECORD == 1) {

                            Data_Memoria.Voltaje[item] = (int) (Voltaje.valor_RMS *
100);
                            Data_Memoria.Corriente[item]      =      (int)
(Ampere.valor_RMS * 100);
                            Data_Memoria.Pot_Watt[item] =
                                    (int) (Potencia_data.Potencia_Activa * 100);
                            Data_Memoria.Pot_VAR[item] =
                                    (int)   (Potencia_data.Potencia_Reactiva   *
100);
                            Data_Memoria.Pot_VA[item] =
                                    (int)   (Potencia_data.Potencia_Aparente   *
100);
                            Data_Memoria.Frec[item] = (int) (frec_line * 100);

                            Reloj.hora[item] = bcdToDec(read_ds1302(0x85));
                            Reloj.min[item] = bcdToDec(read_ds1302(0x83));
                            Reloj.day[item] = bcdToDec(read_ds1302(0x87));
                            Reloj.mes[item] = bcdToDec(read_ds1302(0x89));
                            Reloj.anio[item] = bcdToDec(read_ds1302(0x8d));
                            Reloj.seg[item] = bcdToDec(read_ds1302(0x81));

                            item++;
                            if (item > 2000) {
                                    sprintf(Item_count, "Exeso de Memoria");
                                    item = 2001;
                            } else {
                                    sprintf(Item_count, "REC ITEM = %d   ", item);
                            }

                            setColor(0xff, 0, 0);
                            setBackColor(0xff, 0xff, 0xff);
                            TFT_print(Item_count, 400, 400, 0);

                    } else if (Conf_Osciloscope.RECORD == 0) {
```

```
                    Conf_Osciloscope.RECORD = 3;

                    sprintf(mylabel,    "POT%d%d.CSV",    Reloj.hora[1],
Reloj.min[1]);
                    const char * TEST_FILENAME = mylabel;

                    WRITE_SD_CARD(item, TEST_FILENAME, 2, &Reloj,
&Data_Memoria);

                    sprintf(Item_count, "GUARDADO EN SD");
                    setColor(0xff, 0, 0);
                    setBackColor(0xff, 0xff, 0xff);
                    TFT_print(Item_count, 400, 400, 0);
                    item = 0;
                } else if (Conf_Osciloscope.RECORD == 3) {
                    sprintf(Item_count, "                ");
                    setColor(0xff, 0, 0);
                    setBackColor(0xff, 0xff, 0xff);
                    TFT_print(Item_count, 400, 400, 0);
                }

                sprintf(Mensajes_LCD.str_RMS,    "Vrms    %03.2f    ",
Voltaje.valor_RMS);
                sprintf(Mensajes_LCD.str_Hz, "Hz    %2.3f ", frec_line);
                sprintf(Mensajes_LCD.str_Amplitud_harmonic, "Arms  %03.2f  ",
                        Ampere.valor_RMS);
                sprintf(Potencias_LCD.str_Factor_Potencia,    "FP        %3.2f
",Potencia_data.Factor_Potencia);
                sprintf(Potencias_LCD.str_Factor_DISTORSION,  "FD      %3.2f
",Potencia_data.Factor_DISTORSION);

                sprintf(Potencias_LCD.str_Activa, "Watt  %4.2f   ",
                        Potencia_data.Potencia_Activa);
                sprintf(Potencias_LCD.str_Reactiva, "VAR   %4.2f   ",
                        Potencia_data.Potencia_Reactiva);
                sprintf(Potencias_LCD.str_Aparente, "VA    %4.2f   ",
                        Potencia_data.Potencia_Aparente);

                /* ----------------------------------------------------------------
                 ** Mensajes para el modulo Bluetooh
                 ** ---------------------------------------------------------------- */
                if (Bluetooh_ON_OFF == 1) {
                    sprintf(Mensajes_Bluetooh.str_RMS, "Vrms= %03.2f\n\r",
```

```
                                        Voltaje.valor_RMS);
                        sprintf(Mensajes_Bluetooh.str_Hz,    "Hz=    %2.4f\n\r",
frec_line);
                        sprintf(Mensajes_Bluetooh.str_Amplitud_harmonic,
                                        "Arms= %03.2f\n\r", Ampere.valor_RMS);
                        sprintf(Potencias_Bluetooh.str_Factor_Potencia,      "Fp=
%3.2f\n\r",
                                        Potencia_data.Factor_Potencia);
                        sprintf(Potencias_Bluetooh.str_Factor_DISTORSION,
"FD= %3.2f\n\r",Potencia_data.Factor_DISTORSION);

                        sprintf(Potencias_Bluetooh.str_Activa, "Watt= %4.2f\n\r",
                                        Potencia_data.Potencia_Activa);
                        sprintf(Potencias_Bluetooh.str_Reactiva,             "VAR=
%4.2f\n\r",
                                        Potencia_data.Potencia_Reactiva);
                        sprintf(Potencias_Bluetooh.str_Aparente,             "VA=
%4.2f\n\r",
                                        Potencia_data.Potencia_Aparente);
                        sprintf(Mensaje, "********** \n\r");

                        sciSend(sciREG3, sizeof(Mensaje), Mensaje);

                        sciSend(sciREG3, sizeof(Mensajes_Bluetooh.str_RMS),
                                        Mensajes_Bluetooh.str_RMS);
                        sciSend(sciREG3,

        sizeof(Mensajes_Bluetooh.str_Amplitud_harmonic),

        Mensajes_Bluetooh.str_Amplitud_harmonic);
                        sciSend(sciREG3, sizeof(Mensajes_Bluetooh.str_Hz),
                                        Mensajes_Bluetooh.str_Hz);
                        sciSend(sciREG3, sizeof(Potencias_Bluetooh.str_Activa),
                                        Potencias_Bluetooh.str_Activa);
                        sciSend(sciREG3,
sizeof(Potencias_Bluetooh.str_Aparente),
                                        Potencias_Bluetooh.str_Aparente);
                        sciSend(sciREG3,
sizeof(Potencias_Bluetooh.str_Reactiva),
                                        Potencias_Bluetooh.str_Reactiva);
                        sciSend(sciREG3,
sizeof(Potencias_Bluetooh.str_Factor_Potencia),
                                        Potencias_Bluetooh.str_Factor_Potencia);
```

```
                    sciSend(sciREG3,
sizeof(Potencias_Bluetooh.str_Factor_DISTORSION),

        Potencias_Bluetooh.str_Factor_DISTORSION);

                    }

                    setColor(0, 0, 0);
                    setBackColor(0xff, 0xff, 0xff);

                    TFT_print(Mensajes_LCD.str_RMS, 30, 90, 0);
                    TFT_print(Mensajes_LCD.str_Amplitud_harmonic, 30, 110, 0);

                    TFT_print(Potencias_LCD.str_Activa, 30, 190, 0);
                    TFT_print(Potencias_LCD.str_Aparente, 30, 210, 0);
                    TFT_print(Potencias_LCD.str_Reactiva, 30, 230, 0);
                    TFT_print(Potencias_LCD.str_Factor_Potencia, 30, 250, 0);
                    TFT_print(Potencias_LCD.str_Factor_DISTORSION, 30, 270, 0);
                    TFT_print(Mensajes_LCD.str_Hz, 30, 310, 0);

                    setColor(0x77, 0x99, 0xdd);
                    drawRect(260, 70, 260, 330);
                    drawRect(275, 70, 275, 340);

                    /*Linea Recta que sirve para los 60Hz*/
                    setColor(0xd2, 0xd2, 0xd2);
                    drawLine(plotlcd.X_axis, 200, 700, 200);
                    drawLine(plotlcd.X_axis, 100, 700, 100);
                    drawLine(plotlcd.X_axis, 300, 700, plotlcd.X_axis);

                    if (flag_frec == 1) {
                            float diff_frecuencia = (float) (frec_line - 60);
                            int Pixel_per_Hz = diff_frecuencia * 100;

                            int Pixel_frec_ploty = (int) (-Pixel_per_Hz + 200);

                            vol_xy_buff[sample_hz] = Pixel_frec_ploty; //almacena el
buffer de frecuencia para borrar una vez terminada la cuenta

                            /* ----------------------------------------------------------------
--
                             ** Muestra la grafica de la fecuencia vs tiempo
                             ** ----------------------------------------------------------------
- */
```

```
                    setColor(0xff, 0x00, 0x00);

                    drawCircle(sample_cont + plotlcd.X_axis,
Pixel_frec_ploty, 2);

                    if (sample_cont > 9) {
                        drawLine(x_sample_frec, y_sample_frec,
                                sample_cont + plotlcd.X_axis,
Pixel_frec_ploty);
                        drawLine(x_sample_frec, y_sample_frec + 1,
                                sample_cont + plotlcd.X_axis,
Pixel_frec_ploty + 1);
                    }

                    y_sample_frec = Pixel_frec_ploty;
                    x_sample_frec = sample_cont + plotlcd.X_axis;

                    sample_hz++;
                    sample_cont += 10;
// para 20 muestras en pantalla por cada 5 pixeles
                    if (sample_cont > 400) {
                        int jj = 0;
                        sample_cont = 0;
                        setColor(0xff, 0xff, 0xff);
                        for (jj = 0; jj < 41; jj++) {

                            drawCircle(sample_cont + plotlcd.X_axis,
                                    vol_xy_buff[jj], 2);

                            if (sample_cont > 9) {
                                drawLine(x_sample_frec,
y_sample_frec,
                                        sample_cont +
plotlcd.X_axis,
                                        vol_xy_buff[jj]);
                                drawLine(x_sample_frec,
y_sample_frec + 1,
                                        sample_cont +
plotlcd.X_axis,
                                        vol_xy_buff[jj] + 1);
                            }

                            y_sample_frec = vol_xy_buff[jj];
```

```
                                        x_sample_frec = sample_cont +
plotlcd.X_axis;

                                        sample_cont += 10;
                                    }

                                    sample_cont = 0;
                                    sample_hz = 0;
                                }
                            }
                            Refrescar_LCD_POW = 0;

                        }

                        day = bcdToDec(read_ds1302(0x87));
                        mth = bcdToDec(read_ds1302(0x89));
                        year = bcdToDec(read_ds1302(0x8d));
                        hr = bcdToDec(read_ds1302(0x85));
                        min = bcdToDec(read_ds1302(0x83));
                        sec = bcdToDec(read_ds1302(0x81));

                        sprintf(Mensaje, "%02u/%02u/%02u ", day, mth, year);
                        sprintf(Mensaje6, "%02u:%02u:%02u ", hr, min, sec);
                        setColor(0, 0, 0);
                        setBackColor(0xff, 0xff, 0xff);

                        TFT_print(Mensaje, 30, 400, 0);
                        TFT_print(Mensaje6, 180, 400, 0);

                }

}
```

APRENDICE D
PROGRAMA DE ARMONICOS

```
void HARMONICS() {

        rtiEnableNotification(rtiREG1, rtiNOTIFICATION_COMPARE1); // Sirve como reloj para refrescar la pantalla LCD
        uint8 flag = 1, Touch = 1, Nroamonic = 1;
        int rect_select = 54, buffer_select = 54, spec_select = 6;

        setColor(0xd2, 0xd2, 0xd2);
        fillRect(0, 0, 799, 40);
        setColor(0, 0, 0);
        setBackColor(0xd2, 0xd2, 0xd2);
        TFT_print("HARMONICS", 20, 10, 0);

        /*BONOTES PARTE SUPERIOR DE LA PANTALLA*/
        setColor(0x77, 0x99, 0xdd);
        fillRect(365, 5, 465, 35);
        fillRect(475, 5, 575, 35);
        fillRect(585, 5, 685, 35);
        fillRect(695, 5, 795, 35);

        setColor(0xff, 0xff, 0xff);
        setBackColor(0x77, 0x99, 0xdd);
        TFT_print(" Hold", 370, 10, 0);
        TFT_print(" Run", 480, 10, 0);
        TFT_print("MOVE+", 590, 10, 0);
        TFT_print("MOVE-", 700, 10, 0);

        /* ------------------------------------------------------------------
         **      Inicializa la barra inferior del Menu para el control
         ** ------------------------------------------------------------------ */

        /*Inicializo los valores de los mensajes en botones*/
        sprintf(BtnMensaje.Btn1, "VOLTIOS");
        sprintf(BtnMensaje.Btn2, "CORRIENTE");
        sprintf(BtnMensaje.Btn3, "POTENCIA");
        sprintf(BtnMensaje.Btn4, "EXCEL SD");
        BtnMensaje.GetValue = 1;    // Por default inicia con el voltaje seleccionado
        Btn_TouchAction(&BtnMensaje);
```

```
/**Dibuja el recuadro que contiene el espectro de fourier en su interior*/
//setColor(0x00, 0x00, 0x00);
//drawRect(6, 108, 795, 430);



/**Dibuja el recuadro para los datos de los armonicos*/
setColor(0x22, 0x77, 0xbb);
fillRect(378, 40, 380, 108);

setColor(255, 255, 255);
setBackColor(0x22, 0x77, 0xbb);
TFT_print("DATOS LINEA",6, 40, 0);
TFT_print("DATOS CURSOR",380,40, 0);


/**Dibuja los bordes azules en la pantalla */
//setColor(0x77, 0x99, 0xdd);
setColor(0x22, 0x77, 0xbb);
fillRect(6, 108, 20, 430);
fillRect(795-20, 108, 795, 430);
fillRect(6, 106, 795, 109);

/*Recta superior para el eje de las ordenadas*/
setColor(0x00, 0x00, 0x00);
drawLine(49, 410, 770, 410);          // eje X
drawLine(49, 111, 49, 410);                    // eje Y



setFont(SmallFont);
setColor(0, 0, 0);
setBackColor(0xff, 0xff, 0xff);
TFT_print("0", 40, 405, 0);
TFT_print("50", 30, 254, 0);
TFT_print("100", 25, 110, 0);

int jj = 1, k = 48, x;

for (x = 0; x < 15; x++) {
        sprintf(Mensaje, "%2d", jj);
        TFT_print(Mensaje, k, 412, 0);

        jj = jj + 4;
```

```
        if (jj < 10)
                k = k + 48;
        else if (jj > 10 && jj < 30)
                k = k + 49;

        else if (jj > 30)
                k = k + 48;
}

Conf_Osciloscope.Corriente_factor = (int) READ_FLOAT_EXT_EEPROM(1);
Bluetooh_ON_OFF = Read_25LC(40);
if (Bluetooh_ON_OFF == 1) {
        drawBitmap(400, 5, 24, 33, logo_bluetooth, 1);
        gioSetBit(hetPORT1, 26, 0);          //Bluetooth Pin_control
}

float Period_sampling_wave = 0.048828e-3;        //48.828 us
set_sample_period(Period_sampling_wave);

setFont(BigFont);
arm_cfft_radix4_instance_f32 S;
/* Initialize the CFFT/CIFFT module */
arm_status status = arm_cfft_radix4_init_f32(&S, fftSize, ifftFlag,
                doBitReverse);
uint8 step_harmonic = 6;
Conf_Osciloscope.RECORD = 0;
while (1) {

        if (digitalRead(IRQ) == 0) {
                Touch = Touch_Key();

                if (Touch == 1)
                        flag = 1;                // Espectro Voltios
                else if (Touch == 2)
                        flag = 2;                // Espectro Amperios
                else if (Touch == 3)
                        flag = 3;                // Espectro Potencia
                else if (Touch == 4) {
                        Conf_Osciloscope.RECORD = 1;
                }

                else if (Touch == 5) {
                        Conf_Osciloscope.HOLD_RUN = 1;
                } else if (Touch == 6) {
```

```
                    Conf_Osciloscope.HOLD_RUN = 0;
                } else if (Touch == 7) {
                    rect_select += 12;// Valor que sirve para mover el cursor de medicion en el espectro.
                    spec_select += 6;// Valor que sirve para moverme solo por los armonicos en el espectro.
                    Nroamonic++;// Valor que me indica el numero de armonico, en el que me encuentro.
                } else if (Touch == 8) {
                    rect_select -= 12;
                    spec_select -= 6;
                    Nroamonic--;

                }

                BtnMensaje.GetValue = Touch;       // Ningun Boton Selecionado
                Btn_TouchAction(&BtnMensaje);
            } //if touch fin
            Touch = 0;
            /* ---------------------------------------------------------------
             ** Limites para los cursores de medicion
             ** --------------------------------------------------------------- */
            if (Nroamonic < 1)
                Nroamonic = 1;
            if (spec_select < 6)
                spec_select = 6;
            /* ---------------------------------------------------------------
             ** Dibujo rectas de medicion en la pantalla LCD
             ** --------------------------------------------------------------- */
            if (rect_select < 54)
                rect_select = 54;
            else if (rect_select > 762)
                rect_select = 762;

            setColor(0x00, 0x00, 0xd2);
            drawLine(rect_select, 111, rect_select, 410);

            if (buffer_select != rect_select) {
                setColor(0xff, 0xff, 0xff);
                drawLine(buffer_select, 111, buffer_select, 410);
            }
            buffer_select = rect_select;

            /* ---------------------------------------------------------------
```

```
                ** Etapa de muestreo de la señal alterna
                ** ---------------------------------------------------------- */

                if (Conf_Osciloscope.HOLD_RUN == 0) {
                        for (i = 0; i < TEST_LENGTH_SAMPLES; i++) {
                                adcStartConversion(adcREG1, adcGROUP1); //Start ADC
conversio
                                while                (!adcIsConversionComplete(adcREG1,
adcGROUP1))
                                        ;                       // ConfigutaR halcogen el
trigger por sofware

                                ch_count     =     adcGetData(adcREG1,     adcGROUP1,
&adc_data[0]);
                                ch_count = ch_count;
//Voltaje.wave_sampling[i] = adc_data[0].value - adc_data[1].value;// Señal de voltaje
                                Voltaje.wave_sampling[i] = (adc_data[0].value
                                                - adc_data[1].value) + 280;
                                adcStartConversion(adcREG2, adcGROUP1); //Start ADC
conversio
                                while                (!adcIsConversionComplete(adcREG2,
adcGROUP1))
                                        ;                       // ConfigutaR halcogen el
trigger por sofware

                                ch_count     =     adcGetData(adcREG2,     adcGROUP1,
&adc_data[0]);
                                ch_count = ch_count;
                                Ampere.wave_sampling[i]     =     adc_data[0].value     -
adc_data[1].value;// Señal de corriente

                                //delay_seg(43.71e-6);          // Retardo entre muestras de
83microsegundos

                                while (get_adc_signal == 0) {
                                }
                                get_adc_signal = 0;
                        }

                        /* ----------------------------------------------------------------
                         ** Calculo del Maximo y Minimo de la señal de voltaje
                         ** ---------------------------------------------------------- */
                        arm_max_f32(Voltaje.wave_sampling,
TEST_LENGTH_SAMPLES,
```

```
                                &Voltaje.valor_MAX, &Voltaje.max_marks1);
                arm_min_f32(Voltaje.wave_sampling,
TEST_LENGTH_SAMPLES,
                                &Voltaje.valor_MIN, &Voltaje.min_marks2);

                frec_line = 0.00;
                int Amplitud_wave = Voltaje.valor_MAX - Voltaje.valor_MIN;

                if (Amplitud_wave > 200)
                        frec_line = (float) (1e6 / T_prom);  //RTI0,  cada  10ms
muestrea la frecuencia de linea.
                /*Etapa para identificacar los armonicos de una frecuencia de 50 o
60hz*/
                step_harmonic = 6;
                if (frec_line < 50.5 && frec_line > 49.5)
                        step_harmonic = 5;

                /* ----------------------------------------------------------------
                 ** Etapa para el calculo de los valores RMS (V,I)
                 ** ---------------------------------------------------------------- */
                arm_rms_f32(Voltaje.wave_sampling,
TEST_LENGTH_SAMPLES,
                                &Voltaje.valor_RMS);
// Acondiciono el valor RMS de acuerdo a su ecuacion
                Voltaje.valor_RMS  =  (Voltaje.valor_RMS  *  3.3  /  4095)  /
0.0014039;
                if (Voltaje.valor_RMS < 0.00)
                        Voltaje.valor_RMS = 0.00;

                arm_rms_f32(Ampere.wave_sampling,
TEST_LENGTH_SAMPLES,
                                &Ampere.valor_RMS);
                Ampere.valor_RMS = (Ampere.valor_RMS * 3.3 / 4095)
                                * Conf_Osciloscope.Corriente_factor;
// Calculo de la potencia RMS
                Potencia.valor_RMS = Voltaje.valor_RMS * Ampere.valor_RMS;

                /* ----------------------------------------------------------------
                 ** Etapa para el calculo de la transformada discreta de fourier
                 **  Voltaje,Corriente  y  convolucion  dominio  frecuencia  para
potencia
                 ** ---------------------------------------------------------------- */

                /*FFT de la señal de voltaje*/
```

```
arm_cfft_radix4_f32(&S, Voltaje.wave_sampling);/* Process the
data through the CFFT/CIFFT module */
arm_copy_f32(Voltaje.wave_sampling, Voltaje.fft_PHASE,
fftSize);/*Copio el vector en formato complejo para calculas posteriormente la fase*/
arm_cmplx_mag_f32(Voltaje.wave_sampling, Voltaje.fft_MAG,
fftSize);/* Process the data through the Complex Magnitude Module for calculating the
magnitude at each bin */

/*FFT de la señal de corriente*/
arm_cfft_radix4_f32(&S, Ampere.wave_sampling);/* Process the
data through the CFFT/CIFFT module */
arm_copy_f32(Ampere.wave_sampling, Ampere.fft_PHASE,
fftSize);/*Copio el vector en formato complejo para calculas posteriormente la fase*/
arm_cmplx_mag_f32(Ampere.wave_sampling,
Ampere.fft_MAG, fftSize);/* Process the data through the Complex Magnitude Module
for calculating the magnitude at each bin */

/*FFT de la potencia, multiplifacion DFT(V)*DFT(I)*/
arm_cmplx_mult_cmplx_f32(Voltaje.fft_PHASE,
Ampere.fft_PHASE,
        Potencia.fft_PHASE, fftSize);
arm_cmplx_mag_f32(Potencia.fft_PHASE, Potencia.fft_MAG,
fftSize);
/* ----------------------------------------------------------------
 ** Calculo del THD por cada señal
 ** ---------------------------------------------------------------- */
/*Calculo para el THD Voltaje**/
/* Calculates maxValue and returns corresponding BIN value */
arm_max_f32(Voltaje.fft_MAG, fftSize / 2,
&Voltaje.valor_MAX,
        &Voltaje.max_marks1);

int k_harmonic = 0;
float THD_suma = 0;
for (k_harmonic = 2 * step_harmonic; k_harmonic < 350;
k_harmonic +=
        step_harmonic) {
    THD_suma += Voltaje.fft_MAG[k_harmonic]
            * Voltaje.fft_MAG[k_harmonic];
}
THD_suma = sqrt(THD_suma);
Voltaje.THD_HARMONICS = (float32_t) (THD_suma
        / Voltaje.fft_MAG[Voltaje.max_marks1]) * 100;
/*Calculo para el THD Corriente**/
```

```
                    /* Calculates maxValue and returns corresponding BIN value */
                    arm_max_f32(Ampere.fft_MAG, fftSize / 2,
&Ampere.valor_MAX,
                                        &Ampere.max_marks1);

                    k_harmonic = 0;
                    THD_suma = 0;
                    for (k_harmonic = 2 * step_harmonic; k_harmonic < 350;
k_harmonic +=
                                        step_harmonic) {
                                THD_suma += Ampere.fft_MAG[k_harmonic]
                                                * Ampere.fft_MAG[k_harmonic];
                    }
                    THD_suma = sqrt(THD_suma);
                    Ampere.THD_HARMONICS = (float32_t) (THD_suma
                                        / Ampere.fft_MAG[Ampere.max_marks1]) * 100;
                    /*Calculo para el THD Potencia**/
                    /* Calculates maxValue and returns corresponding BIN value */
                    arm_max_f32(Potencia.fft_MAG, fftSize / 2,
&Potencia.valor_MAX,
                                        &Potencia.max_marks1);

                    k_harmonic = 0;
                    THD_suma = 0;
                    for (k_harmonic = 2 * step_harmonic; k_harmonic < 350;
k_harmonic +=
                                        step_harmonic) {
                                THD_suma += Potencia.fft_MAG[k_harmonic]
                                                * Potencia.fft_MAG[k_harmonic];
                    }
                    THD_suma = sqrt(THD_suma);
                    Potencia.THD_HARMONICS = (float32_t) (THD_suma
                                        / Potencia.fft_MAG[Potencia.max_marks1]) * 100;
            } else {
                    delay_seg(100e-3);
            }
            if (Refrescar_LCD == 1) {

                    /*Grafica los puntos internos*/
                    int j;
                    //setColor(0xd2, 0xd2, 0xd2);
                    setColor(0x00, 0x00, 0x00);
                    for (j = 112; j < 410; j = j + 20) {
                            for (i = 50; i < 770; i += 20) {
```

```
                        drawLine(i, j, i, j + 2);
                }
        }

        for (j = 50; j < 770; j += 20) {
                for (i = 112; i < 410; i += 20)
                        drawLine(j, i, j, i + 2);
        }



        /* ----------------------------------------------------------------
         ** Etapa para cargar Mensajes en Pantalla
         ** ---------------------------------------------------------------- */
        sprintf(Mensajes_LCD.str_Hz, "Hz %3.2f  ", frec_line);
        sprintf(Mensajes_LCD.str_Herz_harmonic, "Hz %4.2f  ",
                        (float) Nroamonic * frec_line);

        if (flag == 1) {
                /*Acondicionamiento del espectro de frecuencia para el
ploteo en la pantalla FFT*/
                arm_max_f32(Voltaje.fft_MAG, fftSize / 2,
&Voltaje.valor_MAX,
                                &Voltaje.max_marks1);
                arm_scale_f32(Voltaje.fft_MAG, 298 /
Voltaje.valor_MAX, buffer,
                                fftSize / 2);
                for (i = 0; i < 350; i++)
                        vol_xy_plot[i] = (uint16) buffer[i];
//Mensajes_LCD

                Voltaje.Harmonic_Porcent = (float32_t)
((buffer[spec_select]
                                / 298) * 100);
                Voltaje.Harmonic_Amplitud = Voltaje.valor_RMS
                                * buffer[spec_select] / 298;

                sprintf(Mensajes_LCD.str_RMS, "Vrms %4.2f   ",
                                Voltaje.valor_RMS);
                sprintf(Mensajes_LCD.str_Amplitud_harmonic, "V %4.2f
",
```

```
                                        Voltaje.Harmonic_Amplitud);
                        sprintf(Mensajes_LCD.str_Porcentaje_harmonic, "E %3.2f
",
                                        Voltaje.Harmonic_Porcent);
                        sprintf(Mensajes_LCD.str_THD, "THD %3.2f   ",
                                        Voltaje.THD_HARMONICS);

                } else if (flag == 2) {
                        /*Acondicionamiento del espectro de frecuencia para el
ploteo en la pantalla FFT*/
                        arm_max_f32(Ampere.fft_MAG, fftSize / 2,
&Ampere.valor_MAX,
                                        &Ampere.max_marks1);
                        arm_scale_f32(Ampere.fft_MAG, 298 /
Ampere.valor_MAX, buffer,
                                        fftSize / 2);
                        for (i = 0; i < 350; i++)
                                vol_xy_plot[i] = (uint16) buffer[i];

                        Ampere.Harmonic_Porcent = (float32_t)
((buffer[spec_select]
                                        / 298) * 100);
                        Ampere.Harmonic_Amplitud = Ampere.valor_RMS
                                        * buffer[spec_select] / 298;

                        sprintf(Mensajes_LCD.str_RMS, "Arms %4.2f ",
Ampere.valor_RMS);
                        sprintf(Mensajes_LCD.str_Amplitud_harmonic, "A %4.2f
",
                                        Ampere.Harmonic_Amplitud);
                        sprintf(Mensajes_LCD.str_Porcentaje_harmonic, "E %3.2f
",
                                        Ampere.Harmonic_Porcent);
                        sprintf(Mensajes_LCD.str_THD, "THD %3.2f  ",
                                        Ampere.THD_HARMONICS);
                } else if (flag == 3) {
                        /*Acondicionamiento del espectro de frecuencia para el
ploteo en la pantalla FFT*/
                        arm_max_f32(Potencia.fft_MAG, fftSize / 2,
&Potencia.valor_MAX,
                                        &Potencia.max_marks1);
                        arm_scale_f32(Potencia.fft_MAG, 298 /
Potencia.valor_MAX,
                                        buffer, fftSize / 2);
```

```
                for (i = 0; i < 350; i++)
                        vol_xy_plot[i] = (uint16) buffer[i];

                Potencia.Harmonic_Porcent = (float32_t)
((buffer[spec_select]
                        / 298) * 100);
                Potencia.Harmonic_Amplitud = Potencia.valor_RMS
                        * buffer[spec_select] / 298;
                sprintf(Mensajes_LCD.str_RMS, "Wrms %4.2f   ",
                        Potencia.valor_RMS);
                sprintf(Mensajes_LCD.str_Amplitud_harmonic, "W %4.2f
",
                        Potencia.Harmonic_Amplitud);
                sprintf(Mensajes_LCD.str_Porcentaje_harmonic, "E
%3.2f%  ",
                        Potencia.Harmonic_Porcent);
                sprintf(Mensajes_LCD.str_THD, "THD %3.2f%  ",
                        Potencia.THD_HARMONICS);
            }

            /* ----------------------------------------------------------------
             ** Etapa de Almacenamiento en Memoria SD
             ** ---------------------------------------------------------------- */

            if (Conf_Osciloscope.RECORD == 1) {
                //wave_sampling = contiene el vector escalado a 100%
                /**Escala el espectro de voltaje**/
                arm_max_f32(Voltaje.fft_MAG, fftSize / 2,
&Voltaje.valor_MAX,
                        &Voltaje.max_marks1);
                arm_scale_f32(Voltaje.fft_MAG,
                        Voltaje.valor_RMS / Voltaje.valor_MAX,
                        Voltaje.wave_sampling, fftSize / 2);
                /**Escala el espectro de corriente**/
                arm_max_f32(Ampere.fft_MAG, fftSize / 2,
&Ampere.valor_MAX,
                        &Ampere.max_marks1);
                arm_scale_f32(Ampere.fft_MAG,
                        Ampere.valor_RMS / Ampere.valor_MAX,
                        Ampere.wave_sampling, fftSize / 2);
                /**Escala el espectro de potencia**/
                arm_max_f32(Potencia.fft_MAG, fftSize / 2,
&Potencia.valor_MAX,
                        &Potencia.max_marks1);
```

```
                    arm_scale_f32(Potencia.fft_MAG,
                            Potencia.valor_RMS / Potencia.valor_MAX,
                            Potencia.wave_sampling, fftSize / 2);

                    /**Almaceno el valor del THD en la memoria SD**/
                    Data_Memoria.THD_AMP = (int) (Ampere.THD_HARMONICS * 100);
                    Data_Memoria.THD_VOL = (int) (Voltaje.THD_HARMONICS * 100);
                    Data_Memoria.THD_POW = (int) (Potencia.THD_HARMONICS * 100);
                    Data_Memoria.Frec[1] = (int) (frec_line * 100);

                    Reloj.hora[1] = bcdToDec(read_ds1302(0x85));
                    Reloj.min[1] = bcdToDec(read_ds1302(0x83));
                    Reloj.day[1] = bcdToDec(read_ds1302(0x87));
                    Reloj.mes[1] = bcdToDec(read_ds1302(0x89));
                    Reloj.anio[1] = bcdToDec(read_ds1302(0x8d));
                    Reloj.seg[1] = bcdToDec(read_ds1302(0x81));

                    i = 0;
                    for (x = step_harmonic; x < 350; x = x + step_harmonic) {
                        Data_Memoria.Voltaje[i] = (int) (Voltaje.wave_sampling[x]
                                * 100);
                        Data_Memoria.Corriente[i] = (int) (Ampere.wave_sampling[x]
                                * 100);
                        Data_Memoria.Pot_Watt[i] = (int) (Potencia.wave_sampling[x]
                                * 100);
                        i++;
                    }

                    sprintf(mylabel, "HAR%d%d.CSV", Reloj.hora[1], Reloj.min[1]);
                    const char * TEST_FILENAME = mylabel;

                    WRITE_SD_CARD(i, TEST_FILENAME, 3, &Reloj, &Data_Memoria);

                    sprintf(Item_count, "SAVE SD");
                    setColor(0xff, 0, 0);
```

```
                    setBackColor(0xff, 0xff, 0xff);
                    TFT_print(Item_count, 640, 80, 0);

                    Conf_Osciloscope.RECORD = 0;//Bloque futuras
grabaciones por error
                } else {
                    sprintf(Item_count, "      ");
                    setColor(0xff, 0, 0);
                    setBackColor(0xff, 0xff, 0xff);
                    TFT_print(Item_count, 640, 80, 0);
                }

                /*Calculo de la fase por harmonico*/
                /*      float angle_harmonics = (180 / 3.14159265)*
atan(Voltaje.fft_PHASE[spec_select + 1]/ Voltaje.fft_PHASE[spec_select]);
                 sprintf(Mensaje6, "%3.2f  ", (float) angle_harmonics);
                 TFT_print(Mensaje6, 470, 70, 0);*/
                /* ------------------------------------------------------------
                 ** Etapa donde se Muestran los Mensajes en pantalla
                 ** ------------------------------------------------------------ */
                setColor(0, 0, 0);
                setBackColor(0xff, 0xff, 0xff);
                TFT_print(Mensajes_LCD.str_THD, 10, 60, 0);
                TFT_print(Mensajes_LCD.str_RMS, 10, 80, 0);
                TFT_print(Mensajes_LCD.str_Hz, 200, 43, 0);

                TFT_print(Mensajes_LCD.str_Herz_harmonic, 590, 43, 0);
                TFT_print(Mensajes_LCD.str_Amplitud_harmonic, 390, 60, 0);
                TFT_print(Mensajes_LCD.str_Porcentaje_harmonic, 390, 80, 0);

                //TFT_print(Mensajes_LCD.str_Angle_harmonic, 470, 70, 0);
                /* ------------------------------------------------------------
                 ** Etapa donde se Dibuja el espectro de frecuencia armonica en
pantalla
                 ** ------------------------------------------------------------ */
                setColor(0xd2, 0xd2, 0xd2);
                drawLine(49, 260, 770, 260);

                int x, ymax = 410, a = 55;

                for (x = step_harmonic; x < 350; x = x + step_harmonic) {
                    setColor(0xff, 0xff, 0xff);
                    fillRect(a, ymax - vol_xy_buff[x], a + 4, ymax - 5);
```

```
                    if (flag == 1)
                        setColor(0xff, 0x00, 0x00);
                    else if (flag == 2)
                        setColor(0x00, 0xfa, 0x00);
                    else if (flag == 3)
                        setColor(0xFF, 0x00, 0xFF);
                    fillRect(a, ymax - vol_xy_plot[x], a + 4, ymax);
                    a = a + 12;
                }

                /* ---------------------------------------------------------------
                 ** Almacena la muestra anterior para limpiar pantalla en la
siguiente
                 ** Muestra
                 ** --------------------------------------------------------------- */
                for (i = 0; i < 350; i++)
                    vol_xy_buff[i] = vol_xy_plot[i];
                Refrescar_LCD = 0;
            }

            day = bcdToDec(read_ds1302(0x87));
            mth = bcdToDec(read_ds1302(0x89));
            year = 2000 + bcdToDec(read_ds1302(0x8d));
            hr = bcdToDec(read_ds1302(0x85));
            min = bcdToDec(read_ds1302(0x83));
            sec = bcdToDec(read_ds1302(0x81));
            sprintf(Mensaje, "%02u/%02u/%02u ", day, mth, year);
            sprintf(Mensaje6, "%02u:%02u:%02u ", hr, min, sec);

            setColor(0, 0, 0);
            setBackColor(0xff, 0xff, 0xff);
            TFT_print(Mensaje6, 630, 110, 0);
            TFT_print(Mensaje, 430, 110, 0);
        }
}
```

APRENDICE E
PROGRAMA DE FLUCTUACIONES

```
void SAGS_SWELLS() {
        int LENGTH_SAMPLES = 256, SECONDS_SAMPLES = 60;
        int LENGTH_LCD = 790;     //790
//
        int i, ii, j;
        boolean flag_end = 0;
        /* ------------------------------------------------------------------
         ** Inicializacion de las variables del programa
         ** ------------------------------------------------------------------ */
//      int sample_cont = 0;   //Sirve para mostrar una distancia en el eje x
        int sample_cont = 0, y_sample_frec = 0, y_sample_amp = 0, x_sample_frec;
        plotlcd.X_axis = 6;

        /* ------------------------------------------------------------------
         ** Inicializacion de las variables del programa
         ** ------------------------------------------------------------------ */

        Conf_Osciloscope.Corriente_factor = (int) READ_FLOAT_EXT_EEPROM(1);
        Bluetooh_ON_OFF = Read_25LC(40);
        if (Bluetooh_ON_OFF == 1) {
                drawBitmap(400, 5, 24, 33, logo_bluetooth, 1);
                gioSetBit(hetPORT1, 26, 0);           //Bluetooth Pin_control
        }
        float Period_sampling_wave = 0.0651e-3;   //0.04069us
        set_sample_period(Period_sampling_wave);
        /* ------------------------------------------------------------------
         ** Pantalla de Inicio del prorgrama
         ** ------------------------------------------------------------------ */
        setColor(0xd2, 0xd2, 0xd2);
        fillRect(0, 0, 799, 40);

        setColor(0, 0, 0);
        setBackColor(0xd2, 0xd2, 0xd2);
        TFT_print("Fluctuaciones", 20, 10, 0);

        /* ------------------------------------------------------------------
         **      Inicializa la barra inferior del Menu para el control
         ** ------------------------------------------------------------------ */

        /*Inicializo los valores de los mensajes en botones*/
        sprintf(BtnMensaje.Btn1, "ENTER");
```

```
sprintf(BtnMensaje.Btn2, "UP");
sprintf(BtnMensaje.Btn3, "DOWN");
sprintf(BtnMensaje.Btn4, " BACK  ");
BtnMensaje.GetValue = 0;    // Ningun Boton Selecionado
Btn_TouchAction(&BtnMensaje);

/* ---------------------------------------------------------------------
 ** Configuraciones Inciales del programa
 ** --------------------------------------------------------------------- */
Menu_Fluctuaciones_Setup();
Menu_CSelect = 0;
setColor(0x22, 0x77, 0xbb);
drawRect(15, 98, 380, 98 + 22);

while (1) {

        flag = digitalRead(IRQ);
        if (flag == 0) {

                Touch = Touch_Key();

                if (Touch == 2) {
                        if (Menu_CSelect > 0)
                                Menu_CSelect--;
                        else
                                Menu_CSelect = 0;

                } else if (Touch == 3) {
                        if (Menu_CSelect < 5)
                                Menu_CSelect++;
                        else
                                Menu_CSelect = 5;
                } else if (Touch == 4) {
                        main();
                }

                if (Touch == 1) {
                        setColor(0xff, 0xff, 0xff);
                        drawRect(15, 98, 380, 98 + 22);
                        Menu_Fluctuaciones_DELED();

                        if (Menu_CSelect == 0)
                                SECONDS_SAMPLES = 60;// 60 SEGUNDOS X
790 MUESTRAS = 13MIN
```

```
                        else if (Menu_CSelect == 1)
                                SECONDS_SAMPLES = 120;
        //26MIN
                        else if (Menu_CSelect == 2)
                                SECONDS_SAMPLES = 240;
        //52MIN
                        else if (Menu_CSelect == 3)
                                SECONDS_SAMPLES = 480;          //1    H
15 MIN
                        else if (Menu_CSelect == 4)
                                SECONDS_SAMPLES = 960;          // 3H Y
30 MIN
                        else if (Menu_CSelect == 5)
                                SECONDS_SAMPLES = 1920;         //7H  Y
420 MIN
                        goto INICIAR_PROGRAMA_FLUCTUACIONES;
                        // BIFURCACION AL PROGRAMA PRINCIPAL
                }

                BtnMensaje.GetValue = Touch;       // Ningun Boton Selecionado
                Btn_TouchAction(&BtnMensaje);
                Menu_Fluctuaciones_Setup();
                Fluctuaciones_SelectMenu(Menu_CSelect);
            }                                       //FIN DEL IF
           //screen_clock();

        }                                       // FIN DEL WHILE

        INICIAR_PROGRAMA_FLUCTUACIONES:

        /*setColor(255, 255, 255);
        setBackColor(0x77, 0x99, 0xdd);
        TFT_print("       ", 20, 445, 0);
        TFT_print("       ", 220, 445, 0);
        TFT_print("       ", 420, 445, 0);
        TFT_print("DETENER ", 640, 445, 0);*/


        /*Inicializo los valores de los mensajes en botones*/
        sprintf(BtnMensaje.Btn1, "       ");
        sprintf(BtnMensaje.Btn2, "       ");
        sprintf(BtnMensaje.Btn3, "       ");
        sprintf(BtnMensaje.Btn4, "DETENER ");
```

```
        BtnMensaje.GetValue = 0;    // Ningun Boton Selecionado
        Btn_TouchAction(&BtnMensaje);


        /*
         sprintf(Item_count, "%d", SECONDS_SAMPLES);
         setColor(0, 0, 0);
         setBackColor(0xff, 0xff, 0xff);*/

        TFT_print(Item_count, 100, 200, 0);
// Draw crosshairs
        setColor(0xd2, 0xd2, 0xd2);
        for (j = 90; j < 245; j = j + 30) {
                for (i = 60; i < 800; i += 60)
                        drawLine(i, j, i, j + 4);
        }

        for (j = 290; j < 425; j = j + 30) {
                for (i = 60; i < 800; i += 60)
                        drawLine(i, j, i, j + 4);
        }

        setColor(0x00, 0x00, 0x00);
        drawRect(5, 90, 795, 245);
        drawRect(5, 290, 795, 425);

        /* ---------------------------------------------------------------
         ** Escala de la corriente y el voltaje
         ** --------------------------------------------------------------- */
        //CORRIENTE
        uint8 Amax, Amin;
        if (Conf_Osciloscope.Corriente_factor == 1) {
                Amax = 1;
                Amin = 0;
        } else if (Conf_Osciloscope.Corriente_factor == 10) {
                Amax = 10;
                Amin = 0;
        } else if (Conf_Osciloscope.Corriente_factor == 100) {
                Amax = 100;
                Amin = 10;
        }
        float m_amp = (float) (135 / (Amax - Amin));
        // VOLTAJE
        int Vlim = 300;
```

```
float m_volt = (float) 155 / Vlim;
/* ---------------------------------------------------------------------
 ** Ejecuta algoritmo principal
 ** --------------------------------------------------------------------- */
while (1) {

        /* ---------------------------------------------------------------------
         ** Almacena el primer valor del reloj al iniciar la Terea
         ** --------------------------------------------------------------------- */
        Reloj.hora[sample_cont] = bcdToDec(read_ds1302(0x85));
        Reloj.min[sample_cont] = bcdToDec(read_ds1302(0x83));
        Reloj.day[sample_cont] = bcdToDec(read_ds1302(0x87));
        Reloj.mes[sample_cont] = bcdToDec(read_ds1302(0x89));
        Reloj.anio[sample_cont] = bcdToDec(read_ds1302(0x8d));
        Reloj.seg[sample_cont] = bcdToDec(read_ds1302(0x81));

        /*Muestras el reloj en pantalla*/
        day = (int) Reloj.day[sample_cont];
        mth = (int) Reloj.mes[sample_cont];
        year = (int) Reloj.anio[sample_cont];
        hr = (int) Reloj.hora[sample_cont];
        min = (int) Reloj.min[sample_cont];
        sec = (int) Reloj.seg[sample_cont];

        sprintf(Mensaje, "%02u/%02u/%02u ", day, mth, year);
        sprintf(Mensaje6, "%02u:%02u:%02u ", hr, min, sec);
        setColor(0, 0, 0);
        setBackColor(0xff, 0xff, 0xff);

        TFT_print(Mensaje, 600, 50, 0);
        TFT_print(Mensaje6, 600, 70, 0);

        for (ii = 0; ii < SECONDS_SAMPLES; ii++) {

                flag = digitalRead(IRQ);
                if (flag == 0) {

                        Touch = Touch_Key();

                        if (Touch == 4)
                                flag_end = 1;

                        BtnMensaje.GetValue = Touch;       //      Ningun      Boton
Selecionado
```

```
                Btn_TouchAction(&BtnMensaje);

            }                                   // if touch

            for (i = 0; i < LENGTH_SAMPLES; i++) {
                adcStartConversion(adcREG1, adcGROUP1); //Start ADC
conversio
                while               (!adcIsConversionComplete(adcREG1,
adcGROUP1))
                    ;               // ConfigutaR halcogen el trigger por
sofware

                ch_count    =    adcGetData(adcREG1,    adcGROUP1,
&adc_data[0]);
                ch_count = ch_count;
                Voltaje.wave_sampling[i]    =    adc_data[0].value    -
adc_data[1].value
                        + 277;

                adcStartConversion(adcREG2, adcGROUP1); //Start ADC
conversio
                while               (!adcIsConversionComplete(adcREG2,
adcGROUP1))
                    ;               // ConfigutaR halcogen el trigger por
sofware

                ch_count    =    adcGetData(adcREG2,    adcGROUP1,
&adc_data[0]);
                ch_count = ch_count;
                Ampere.wave_sampling[i]    =    adc_data[0].value    -
adc_data[1].value;

//delay_seg(65.10e-6);// Retardo entre muestras de 83microsegundos
                while (get_adc_signal == 0) {
                }
                get_adc_signal = 0;
            }

            arm_rms_f32(Voltaje.wave_sampling, LENGTH_SAMPLES,
                    &Voltaje.valor_RMS);
            //Voltaje.valor_RMS  =  (Voltaje.valor_RMS  *  3.3  /  4095)  /
0.0017272;
            //Voltaje.valor_RMS = (Voltaje.valor_RMS * 3.3 / 4095);
```

```
                    Voltaje.valor_RMS = (Voltaje.valor_RMS * 3.3 / 4095) /
0.0014039;
                    arm_rms_f32(Ampere.wave_sampling, LENGTH_SAMPLES,
                            &Ampere.valor_RMS);
                    Ampere.valor_RMS = (Ampere.valor_RMS * 3.3 / 4095)
                            * Conf_Osciloscope.Corriente_factor;

        /*****************************************************/
                    Voltaje.fft_MAG[ii] = Voltaje.valor_RMS;
                    Ampere.fft_MAG[ii] = Ampere.valor_RMS;
                }

                /* ----------------------------------------------------------------
                 ** Calculo del Maximo y Minimo
                 ** ---------------------------------------------------------------- */
                arm_max_f32(Voltaje.fft_MAG,        SECONDS_SAMPLES,
&Voltaje.valor_MAX,
                        &Voltaje.max_marks1);
                arm_min_f32(Voltaje.fft_MAG,        SECONDS_SAMPLES,
&Voltaje.valor_MIN,
                        &Voltaje.min_marks2);

                arm_max_f32(Ampere.fft_MAG,         SECONDS_SAMPLES,
&Ampere.valor_MAX,
                        &Ampere.max_marks1);
                arm_min_f32(Ampere.fft_MAG,         SECONDS_SAMPLES,
&Ampere.valor_MIN,
                        &Ampere.min_marks2);

                /* ----------------------------------------------------------------
                 ** Calculo de la Media
                 ** ---------------------------------------------------------------- */

                arm_mean_f32(Voltaje.fft_MAG,       SECONDS_SAMPLES,
&Voltaje.valor_MEAN);
                arm_mean_f32(Ampere.fft_MAG,        SECONDS_SAMPLES,
&Ampere.valor_MEAN);

                /* ----------------------------------------------------------------
                 ** Almacena en la memoria los valores que seran guardados dentro
                 ** de la tabla excel por la memoria SD
                 ** ---------------------------------------------------------------- */
                Data_Memoria.Voltaje_mean[sample_cont] =
                        (int) (Voltaje.valor_MEAN * 100);
```

```
                Data_Memoria.Voltaje_max[sample_cont] = (int) (Voltaje.valor_MAX *
100);
                Data_Memoria.Voltaje_min[sample_cont] = (int) (Voltaje.valor_MIN *
100);
                Data_Memoria.Corriente_mean[sample_cont]          =          (int)
(Ampere.valor_MEAN
                                * 100);
                Data_Memoria.Corriente_max[sample_cont] =
                                (int) (Ampere.valor_MAX * 100);
                Data_Memoria.Corriente_min[sample_cont] =
                                (int) (Ampere.valor_MIN * 100);

                /* ---------------------------------------------------------------
                 ** Carga los Mensajes que se presentaran en pantalla
                 ** ---------------------------------------------------------------- */
                sprintf(Msn_Vol.str_MAX, "%3.2f Vmax  ", Voltaje.valor_MAX);
                sprintf(Msn_Vol.str_MIN, "%3.2f Vmin  ", Voltaje.valor_MIN);
                sprintf(Msn_Vol.str_MEAN, "%3.2f V  ", Voltaje.valor_MEAN);

                sprintf(Msn_Amp.str_MAX, "%3.2f Amax  ", Ampere.valor_MAX);
                sprintf(Msn_Amp.str_MIN, "%3.2f Amin  ", Ampere.valor_MIN);
                sprintf(Msn_Amp.str_MEAN, "%3.2f A  ", Ampere.valor_MEAN);

                setColor(0, 0, 0);
                setBackColor(0xff, 0xff, 0xff);
                TFT_print(Msn_Vol.str_MAX, 50, 50, 0);
                TFT_print(Msn_Vol.str_MEAN, 50, 70, 0);
                TFT_print(Msn_Vol.str_MIN, 250, 50, 0);

                TFT_print(Msn_Amp.str_MAX, 50, 250, 0);
                TFT_print(Msn_Amp.str_MEAN, 50, 270, 0);
                TFT_print(Msn_Amp.str_MIN, 250, 250, 0);

                /*600, porque el maximo voltaje es de 600 Vrms*/
                //uint16 plot_xy_mean = 240 - (uint16) Voltaje.valor_MEAN * 200 / 600;
                uint16 plot_xy_mean = (-m_volt * (Voltaje.valor_MEAN - Vlim)) + 90;

                //uint16 plot_xy_mean_amp = 420 - (uint16) Ampere.valor_MEAN * 200
/ 200;
                uint16 plot_xy_mean_amp = (-m_amp * (Ampere.valor_MEAN - Amax))
+ 290;

                /* Almacena en Memoria los valores promedio de la corriente y voltaje*/
                FLUCTUACION_VOLTAJE[sample_cont] = Voltaje.valor_MEAN;
```

```
                FLUCTUACION_CORRIENTE[sample_cont] = Ampere.valor_MEAN;

                vol_xy_buff[sample_cont] = plot_xy_mean;
                amp_xy_buff[sample_cont] = plot_xy_mean_amp;
                /* ----------------------------------------------------------------
                 ** Muestra la grafica de la fecuencia vs tiempo
                 ** ---------------------------------------------------------------- */

                setColor(0x00, 0x00, 0xff);
                drawPixel(sample_cont + plotlcd.X_axis, plot_xy_mean);
                drawPixel(sample_cont + plotlcd.X_axis, plot_xy_mean + 1);

                setColor(0xff, 0x00, 0x00);
                drawPixel(sample_cont + plotlcd.X_axis, plot_xy_mean_amp);
                drawPixel(sample_cont + plotlcd.X_axis, plot_xy_mean_amp + 1);

                if (sample_cont > 1) {                  // interpolacion entre puntos
                        setColor(0x00, 0x00, 0xff);
                        drawLine(x_sample_frec,     y_sample_frec,     sample_cont     +
plotlcd.X_axis,
                                        plot_xy_mean);
                        setColor(0xff, 0x00, 0x00);
                        drawLine(x_sample_frec,     y_sample_amp,     sample_cont     +
plotlcd.X_axis,
                                        plot_xy_mean_amp);
                }

                y_sample_frec = plot_xy_mean;
                y_sample_amp = plot_xy_mean_amp;
                x_sample_frec = sample_cont + plotlcd.X_axis;

                sample_cont++;
                //sample_cont++;

// FIN DEL MUESTREO TOTAL
                if (sample_cont > LENGTH_LCD || flag_end == 1) {

                        setColor(0xff, 0X00, 0X00);
                        setBackColor(0xff, 0xff, 0xff);
                        TFT_print("FINALIZO", 600, 250, 0);


                /*      setColor(255, 255, 255);
                        setBackColor(0x77, 0x99, 0xdd);
```

```
                TFT_print("       ", 20, 445, 0);
                TFT_print("GRABAR  ", 220, 445, 0);
                TFT_print("MENU    ", 420, 445, 0);
                TFT_print("       ", 640, 445, 0);*/

                sprintf(BtnMensaje.Btn1, "       ");
                sprintf(BtnMensaje.Btn2, "GRABAR  ");
                sprintf(BtnMensaje.Btn3, "MENU    ");
                sprintf(BtnMensaje.Btn4, "       ");
                BtnMensaje.GetValue = 0;    // Ningun Boton Selecionado
                Btn_TouchAction(&BtnMensaje);

                /*Inicializacion de las variables*/
                flag_end = 0;
                int jj = 0;

                int longitud = sample_cont;

                sample_cont = 0;
                setColor(0xff, 0xff, 0xff);
                for (jj = 0; jj < longitud; jj++) {
                        drawPixel(sample_cont + plotlcd.X_axis, vol_xy_buff[jj]);
                        drawPixel(sample_cont + plotlcd.X_axis, vol_xy_buff[jj] +
1);
                        drawPixel(sample_cont          +          plotlcd.X_axis,
amp_xy_buff[jj]);
                        drawPixel(sample_cont + plotlcd.X_axis, amp_xy_buff[jj]
+ 1);

                        if (sample_cont > 1) {
                                drawLine(x_sample_frec, y_sample_frec,
                                                sample_cont     +     plotlcd.X_axis,
vol_xy_buff[jj]);
                                drawLine(x_sample_frec, y_sample_amp,
                                                sample_cont     +     plotlcd.X_axis,
amp_xy_buff[jj]);
                        }

                        y_sample_frec = vol_xy_buff[jj];
                        y_sample_amp = amp_xy_buff[jj];
                        x_sample_frec = sample_cont + plotlcd.X_axis;

                        sample_cont++;
                }
```

```
                    /*Etapa de calculo para la pantalla LCD en memoria*/
                    /* ----------------------------------------------------------------------
                     ** Calculo del Maximo y Minimo
                     ** ---------------------------------------------------------------------- */
                    arm_max_f32(FLUCTUACION_VOLTAJE,          longitud,
&Voltaje.valor_MAX,
                                &Voltaje.max_marks1);
                    arm_min_f32(FLUCTUACION_VOLTAJE,          longitud,
&Voltaje.valor_MIN,
                                &Voltaje.min_marks2);

                    arm_max_f32(FLUCTUACION_CORRIENTE,        longitud,
&Ampere.valor_MAX,
                                &Ampere.max_marks1);
                    arm_min_f32(FLUCTUACION_CORRIENTE,        longitud,
&Ampere.valor_MIN,
                                &Ampere.min_marks2);

                    float Vinf = Voltaje.valor_MIN;
                    float Vsup = Voltaje.valor_MAX;

                    float Ainf = Ampere.valor_MIN;
                    float Asup = Ampere.valor_MAX;

                    float diff_supinf = Vsup - Vinf;
                    float diff_supinf_amp = Asup - Ainf;

                    for (jj = 0; jj < longitud; jj++) {
                        uint16 yvol = (uint16) ((-150 / (20 + diff_supinf))
                                * (FLUCTUACION_VOLTAJE[jj] - Vinf +
10) + 240);
                        uint16 yamp = (uint16) ((-150 / (20 + diff_supinf_amp))
                                *    (FLUCTUACION_CORRIENTE[jj]    -
Ainf + 10) + 420);
                        amp_xy_buff[jj] = yamp;
                        vol_xy_buff[jj] = yvol;

                    }

                    sprintf(Msn_Vol.str_MIN, "%dV ", (int) Voltaje.valor_MIN);
                    sprintf(Msn_Vol.str_MAX, "%dV ", (int) Voltaje.valor_MAX);
                    sprintf(Msn_Amp.str_MIN, "%3.2fA ", Ampere.valor_MIN);
                    sprintf(Msn_Amp.str_MAX, "%3.2fA ", Ampere.valor_MAX);
```

```
                setColor(0xe2, 0xe2, 0xe2);
                setBackColor(0xff, 0xff, 0xff);
                TFT_print(Msn_Vol.str_MIN, 15, 220, 0);
                TFT_print(Msn_Vol.str_MAX, 15, 100, 0);
                TFT_print(Msn_Amp.str_MAX, 15, 300, 0);
                TFT_print(Msn_Amp.str_MIN, 15, 400, 0);

                sample_cont = 0;

                for (jj = 0; jj < longitud - plotlcd.X_axis; jj++) {
                        setColor(0x00, 0x00, 0xff);
                        drawPixel(sample_cont + plotlcd.X_axis, vol_xy_buff[jj]);
                        drawPixel(sample_cont + plotlcd.X_axis, vol_xy_buff[jj] +
1);

                        setColor(0xff, 0x00, 0x00);
                        drawPixel(sample_cont + plotlcd.X_axis,
amp_xy_buff[jj]);
                        drawPixel(sample_cont + plotlcd.X_axis, amp_xy_buff[jj]
+ 1);

                        if (sample_cont > 1) {
                                setColor(0x00, 0x00, 0xff);
                                drawLine(x_sample_frec, y_sample_frec,
                                                sample_cont + plotlcd.X_axis,
vol_xy_buff[jj]);
                                setColor(0xff, 0x00, 0x00);
                                drawLine(x_sample_frec, y_sample_amp,
                                                sample_cont + plotlcd.X_axis,
amp_xy_buff[jj]);
                        }

                        y_sample_frec = vol_xy_buff[jj];
                        y_sample_amp = amp_xy_buff[jj];
                        x_sample_frec = sample_cont + plotlcd.X_axis;

                        sample_cont++;
                }

                while (1) {
                        flag = digitalRead(IRQ);
                        if (flag == 0) {

                                Touch = Touch_Key();
```

```
                    if (Touch == 2) {
                        sprintf(mylabel, "FLUC%d.CSV",
                                Reloj.min[sample_cont - 1]);
                        const char * TEST_FILENAME = mylabel;

                        WRITE_SD_CARD(longitud,
TEST_FILENAME, 1, &Reloj,
                                &Data_Memoria);

                        setColor(0xff, 0X00, 0X00);
                        setBackColor(0xff, 0xff, 0xff);
                        TFT_print("GRABADO  EXITOSO",  550,
250, 0);

                    } else if (Touch == 3)
                        main();

                }                                   // if touch

                BtnMensaje.GetValue = Touch;     //    Ningun    Boton
Selecionado

                Btn_TouchAction(&BtnMensaje);
            }
        }
    }
}
```

APRENDICE F
PROGRAMA TRANSITORIOS

```
void TRANSIENTS() {

        int i, buffer_select = 6, Evnto = 0;
        int LENGTH_SAMPLES = 1600;
        boolean Flag_trasients = 0;
        /* ---------------------------------------------------------------
         **
         ** ---------------------------------------------------------------- */
        setColor(0xd2, 0xd2, 0xd2);
        fillRect(0, 0, 799, 40);
        setColor(0, 0, 0);
        setBackColor(0xd2, 0xd2, 0xd2);
        TFT_print("TRANSITORIOS", 20, 10, 0);
        /* ---------------------------------------------------------------
         **      Inicializa la barra inferior del Menu para el control
         ** ---------------------------------------------------------------- */
/*      setColor(255, 255, 255);
        setBackColor(0x77, 0x99, 0xdd);
        TFT_print(" START  ", 20, 445, 0);
        TFT_print(" BACK   ", 640, 445, 0);*/


        sprintf(BtnMensaje.Btn1, " START ");
        sprintf(BtnMensaje.Btn2, "UP");
        sprintf(BtnMensaje.Btn3, "DOWN");
        sprintf(BtnMensaje.Btn4, " BACK  ");
        BtnMensaje.GetValue = 0;    // Por default inicia con el voltaje seleccionado
        Btn_TouchAction(&BtnMensaje);

        /*      arm_fill_q31(200,&vol_xy_buff,1024);
         arm_fill_q31(200,&amp_xy_buff,1024);
         */

        rtiEnableNotification(rtiREG1, rtiNOTIFICATION_COMPARE2); // ACTIVA
EL DELAY PARA MUESTREAR LA SEÑAL

        Bluetooh_ON_OFF = Read_25LC(40);
        if (Bluetooh_ON_OFF == 1) {
                drawBitmap(400, 5, 24, 33, logo_bluetooth, 1);
                gioSetBit(hetPORT1, 26, 0);             //Bluetooth Pin_control
        }
```

```
        Conf_Osciloscope.Inteporl_SENSIBILIDAD = (int) READ_FLOAT_EXT_EEPROM(30);
        int pixel_sensibilidad = Conf_Osciloscope.Inteporl_SENSIBILIDAD;
        Conf_Osciloscope.Rango_conf.Voltaje_V = 500;//(int) READ_FLOAT_EXT_EEPROM(20);

        float Period_sampling_wave = 0.08335e-3;  //48.828 us
        set_sample_period(Period_sampling_wave);

        frec_line = 60.0;
        float RTI_sample = Period_sampling_wave;
        for (i = 0; i < LENGTH_SAMPLES / 2; i++) {

                Ampere.wave_sampling[i] = 512* sin(2 * 3.1415 * frec_line * i * RTI_sample);

                /*int variable = (int) (Ampere.wave_sampling[i] * 70 / Conf_Osciloscope.Rango_conf.Voltaje_V + 200);
                amp_xy_plot[i] = variable;*/
                vol_xy_buff[i] = 200;
        }

        int diff_pico = 0;
        float Chng_volt = 0.00;
        /* ------------------------------------------------------------------
         ** Configuraciones Inciales del programa
         ** ------------------------------------------------------------------ */
        Menu_Transitorios_Setup();
        Menu_CSelect = 0;
        setColor(0x22, 0x77, 0xbb);
        drawRect(15, 98, 100, 98 + 22);

        while (1) {

                flag = digitalRead(IRQ);
                if (flag == 0) {

                        Touch = Touch_Key();

                        if (Touch == 2) {
                                if (Menu_CSelect > 0)
                                        Menu_CSelect--;
                                else
```

```
                    Menu_CSelect = 0;

            } else if (Touch == 3) {
                    if (Menu_CSelect < 3)
                            Menu_CSelect++;
                    else
                            Menu_CSelect = 3;
            } else if (Touch == 4) {
                    main();
            }

            if (Touch == 1) {
                    setColor(0xff, 0xff, 0xff);
                    drawRect(15, 98, 100, 98 + 22);
                    Menu_Transitorios_DELED();

                    if (Menu_CSelect == 0)
                            Chng_volt = 0.3;
                    else if (Menu_CSelect == 1)
                            Chng_volt = 0.5;
                    else if (Menu_CSelect == 2)
                            Chng_volt = 1;
                    else if (Menu_CSelect == 3)
                            Chng_volt = 2;

                    goto INICIAR_PROGRAMA_TRANSITORIOS;
                    // BIFURCACION AL PROGRAMA PRINCIPAL
            }

            BtnMensaje.GetValue = Touch;       // Ningun Boton Selecionado
            Btn_TouchAction(&BtnMensaje);
            Menu_Transitorios_Setup();
            Transitorios_SelectMenu(Menu_CSelect);
        }                                       //FIN DEL IF
        //screen_clock();

    }                                           // FIN DEL WHILE

    INICIAR_PROGRAMA_TRANSITORIOS:

/*  setColor(255, 255, 255);
    setBackColor(0x77, 0x99, 0xdd);
    TFT_print("       ", 20, 445, 0);
    TFT_print("       ", 220, 445, 0);
```

```
TFT_print("        ", 420, 445, 0);
TFT_print("DETENER ", 640, 445, 0);*/

sprintf(BtnMensaje.Btn1, "        ");
sprintf(BtnMensaje.Btn2, "        ");
sprintf(BtnMensaje.Btn3, "        ");
sprintf(BtnMensaje.Btn4, "DETENER ");
BtnMensaje.GetValue = 0;    // Por default inicia con el voltaje seleccionado
Btn_TouchAction(&BtnMensaje);


diff_pico = (int) (512 * Chng_volt);

while (1) {

        /*Etapa del reloj del sistema*/

        Reloj.hora[Evnto] = bcdToDec(read_ds1302(0x85));
        Reloj.min[Evnto] = bcdToDec(read_ds1302(0x83));
        Reloj.day[Evnto] = bcdToDec(read_ds1302(0x87));
        Reloj.mes[Evnto] = bcdToDec(read_ds1302(0x89));
        Reloj.anio[Evnto] = bcdToDec(read_ds1302(0x8d));
        Reloj.seg[Evnto] = bcdToDec(read_ds1302(0x81));

        /*Muestras el reloj en pantalla*/
        day = (int) Reloj.day[Evnto];
        mth = (int) Reloj.mes[Evnto];
        year = (int) Reloj.anio[Evnto];
        hr = (int) Reloj.hora[Evnto];
        min = (int) Reloj.min[Evnto];
        sec = (int) Reloj.seg[Evnto];

        sprintf(Mensaje, "%02u/%02u/%02u ", day, mth, year);
        sprintf(Mensaje6, "%02u:%02u:%02u ", hr, min, sec);
        setColor(0, 0, 0);
        setBackColor(0xff, 0xff, 0xff);
        TFT_print(Mensaje6, 50, 400, 0);
        TFT_print(Mensaje, 200, 400, 0);

        //screen_clock();

        flag = digitalRead(IRQ);
        if (flag == 0) {
```

```
                Touch = Touch_Key();

                BtnMensaje.GetValue = Touch;       // Ningun Boton Selecionado
                Btn_TouchAction(&BtnMensaje);
                if (Touch == 4)
                        TRANSIENTS_ANALISIS(Evnto);
        }

        for (i = 0; i < LENGTH_SAMPLES; i++) {
                adcStartConversion(adcREG1,  adcGROUP1);  //Start  ADC
conversio
                while (!adcIsConversionComplete(adcREG1, adcGROUP1))
                        ;       // ConfigutaR halcogen el trigger por sofware

                ch_count = adcGetData(adcREG1, adcGROUP1, &adc_data[0]);
                ch_count = ch_count;
                Voltaje.wave_sampling[i]      =      (adc_data[0].value      -
adc_data[1].value)
                                + 277;

                while (get_adc_signal == 0) {
                }
                get_adc_signal = 0;

        }

        /* ----------------------------------------------------------------
         ** Calculo del Maximo y Minimo de la señal de voltaje
         ** ---------------------------------------------------------------- */
        arm_max_f32(Voltaje.wave_sampling,          LENGTH_SAMPLES,
&Voltaje.valor_MAX,
                        &Voltaje.max_marks1);
        arm_min_f32(Voltaje.wave_sampling,          LENGTH_SAMPLES,
&Voltaje.valor_MIN,
                        &Voltaje.min_marks2);
        frec_line = 0.00;
        int Amplitud_wave = Voltaje.valor_MAX - Voltaje.valor_MIN;

        if (Amplitud_wave > 200)
                frec_line = (float) (1e6 / T_prom);

        /* ----------------------------------------------------------------
         ** Tigger Digital por algoritmo
```

```
 ** ---------------------------------------------------------------- */
int jh = 0, trigger_look = 0, diff_val = 0, centrotrigger =
                LENGTH_SAMPLES / 2;
int Dato_Actual = 0, Dato_Anterior = 0;

do {
        Dato_Actual = Voltaje.wave_sampling[centrotrigger + jh + 1];
        Dato_Anterior = Voltaje.wave_sampling[centrotrigger + jh];
        diff_val = Dato_Actual - Dato_Anterior;

        if (Dato_Actual > 0 && Dato_Actual < 20 && diff_val > 10
                        && diff_val < 20) // MODIFICADO PARA
MAYOR PRECISION
                trigger_look = 1;
        else {
                trigger_look = 0;
        }

        jh++;
} while ((trigger_look == 0) && (jh < LENGTH_SAMPLES / 2));

int sk = 0;

if (frec_line > 40) {

        /* ----------------------------------------------------------------
         ** Acondicionamiento de señal para graficar en LCD
         ** ---------------------------------------------------------------- */
        sk = 0;
        for (i = jh; i < (LENGTH_SAMPLES / 2) + jh; i++) {
                Voltaje.fft_PHASE[sk] = Voltaje.wave_sampling[i];
                sk++;
        }

        float sin_max, sin_min;
        for (i = 6; i < (LENGTH_SAMPLES / 2) - 6; i++) {
                sin_max = Ampere.wave_sampling[i] + diff_pico;
                sin_min = Ampere.wave_sampling[i] - diff_pico;

                if (Voltaje.fft_PHASE[i] > sin_max
                                || Voltaje.fft_PHASE[i] < sin_min) {
                        Flag_trasients = 1;
                        break;
                }       // FIN DEL IF VOLTAJE
```

```
                }       // FIN DEL FOR PARA DETECTAR SOBREPICOS

                if (Flag_trasients == 1) {

                        sprintf(Mensajes_LCD.str_RMS, "Evento: %d   ", Evnto +
1);
                        setColor(0, 0, 0);
                        setBackColor(0xff, 0xff, 0xff);
                        TFT_print(Mensajes_LCD.str_RMS, 60, 45, 0);

                        if (Evnto < 20) {
                                for (i = 0; i < LENGTH_SAMPLES / 2; i++) {
                                        Transient_vol[Evnto][i]                 =
Voltaje.fft_PHASE[i];
                                }

                                Evnto++;
                        } else {
                                TRANSIENTS_ANALISIS(Evnto);
                        }
                        Flag_trasients = 0;
                }       // IF FLAG TRASNIENTS
        }       // IF DE FRECUENCIA >40HZ

        sk = 0;
        for (i = jh; i < (LENGTH_SAMPLES / 2) + jh; i++) {

                float xy_acondition = Voltaje.wave_sampling[i] * 70
                                / Conf_Osciloscope.Rango_conf.Voltaje_V;
                int ypoint = (int) (xy_acondition + 200);
                vol_xy_plot[sk] = ypoint;

                sk++;

        }
        /*      setColor(0x00, 0x00, 0xd2);
         drawLine(n_potion, 90, n_potion, 350);

         if (buffer_select != n_potion) {
         setColor(0xff, 0xff, 0xff);
         drawLine(buffer_select, 90, buffer_select, 350);
         }
         buffer_select = n_potion;*/
```

```
setColor(0x0, 0x0, 0x0);
drawLine(5, 200, 795, 200);
setColor(0xd2, 0xd2, 0xd2);
int j;
for (j = 50; j < 430; j = j + 20) {
        for (i = 10; i < 800; i += 80) {
                drawLine(i, j, i, j + 2);
        }
}

for (j = 10; j < 800; j += 20) {
        for (i = 50; i < 430; i += 80)
                drawLine(j, i, j, i + 2);
}

int k, m_pendiente; //, ancho = (int) (diff_pico * 70/
Conf_Osciloscope.Rango_conf.Voltaje_V);
for (k = 6; k < LENGTH_SAMPLES / 2 - 6; k++) {

        /* ---------------------------------------------------------------
         ** Interpolacion de la señal para borrar en la pantalla
         ** --------------------------------------------------------------- */
        setColor(0xff, 0xff, 0xff);
        drawPixel(k, vol_xy_buff[k]);
        drawPixel(k, vol_xy_buff[k] + 1);
        m_pendiente = abs(vol_xy_buff[k + 1] - vol_xy_buff[k]);
        if (m_pendiente > pixel_sensibilidad) {
                drawLine(k, vol_xy_buff[k], k + 1, vol_xy_buff[k + 1]);
                drawLine(k, vol_xy_buff[k] + 1, k + 1, vol_xy_buff[k + 1]
+ 1);
        }

        /*      drawPixel(k, amp_xy_buff[k] - ancho);
         drawPixel(k, amp_xy_buff[k] + ancho);
         m_pendiente = abs(amp_xy_buff[k + 1] - amp_xy_buff[k]);
         if (m_pendiente > pixel_sensibilidad) {
         drawLine(k, amp_xy_buff[k] - ancho, k + 1,
         amp_xy_buff[k + 1] - ancho);
         drawLine(k, amp_xy_buff[k] + ancho, k + 1,
         amp_xy_buff[k + 1] + ancho);
         }*/

        /* ---------------------------------------------------------------
         ** Interpolacion de la señal para mostrar en la pantalla
```

```
                        ** ------------------------------------------------------------- */
                        setColor(0xff, 0, 0);
                        drawPixel(k, vol_xy_plot[k]);
                        drawPixel(k, vol_xy_plot[k] + 1);
                        m_pendiente = abs(vol_xy_plot[k + 1] - vol_xy_plot[k]);
                        if (m_pendiente > pixel_sensibilidad) {
                                drawLine(k, vol_xy_plot[k], k + 1, vol_xy_plot[k + 1]);
                                drawLine(k, vol_xy_plot[k] + 1, k + 1, vol_xy_plot[k + 1]
+ 1);
                        }

                        /*
                         setColor(0, 0x00, 0xff);
                         drawPixel(k, amp_xy_plot[k] - ancho);
                         drawPixel(k, amp_xy_plot[k] + ancho);
                         m_pendiente = abs(amp_xy_plot[k + 1] - amp_xy_plot[k]);
                         if (m_pendiente > pixel_sensibilidad) {
                         drawLine(k, amp_xy_plot[k] - ancho, k + 1,
                         amp_xy_plot[k + 1] - ancho);
                         drawLine(k, amp_xy_plot[k] + ancho, k + 1,
                         amp_xy_plot[k + 1] + ancho);
                         }*/

                }
//              while (1)

                /*Almacenamiento en buffer de datos para una proxima comparacion*/
                for (i = 6; i < LENGTH_SAMPLES / 2 - 6; i++) {
                        vol_xy_buff[i] = vol_xy_plot[i];
                        //amp_xy_buff[i] = amp_xy_plot[i];
                }

        }

}

void TRANSIENTS_ANALISIS(int Cont_Event) {
        TRANSIENTS_BORRAR_LCD();
/*
        setColor(255, 255, 255);
        setBackColor(0x77, 0x99, 0xdd);
        TFT_print(" Atras    ", 20, 445, 0);
        TFT_print("Adelante  ", 220, 445, 0);
        TFT_print("GRABAR    ", 420, 445, 0);
```

```
        TFT_print(" Menu   ", 640, 445, 0);*/

        sprintf(BtnMensaje.Btn1, " Atras    ");
        sprintf(BtnMensaje.Btn2, "Adelante  ");
        sprintf(BtnMensaje.Btn3, "GRABAR    ");
        sprintf(BtnMensaje.Btn4, " Menu   ");
        BtnMensaje.GetValue = 0;    // Por default inicia con el voltaje seleccionado
        Btn_TouchAction(&BtnMensaje);

        /*BONOTES PARTE SUPERIOR DE LA PANTALLA*/
        setColor(0x77, 0x99, 0xdd);
        fillRect(365, 5, 465, 35);
        fillRect(475, 5, 575, 35);
        fillRect(585, 5, 685, 35);
        fillRect(695, 5, 795, 35);

        setColor(0xff, 0xff, 0xff);
        setBackColor(0x77, 0x99, 0xdd);
        TFT_print("JUMP+", 370, 10, 0);
        TFT_print("JUMP-", 480, 10, 0);
        TFT_print("MOVE+", 590, 10, 0);
        TFT_print("MOVE-", 700, 10, 0);

        if (Cont_Event == 0) {
/*
                TFT_print("         ", 20, 445, 0);
                TFT_print("         ", 220, 445, 0);
                TFT_print("         ", 420, 445, 0);
                TFT_print(" Menu   ", 640, 445, 0);
*/

                sprintf(BtnMensaje.Btn1, "         ");
                sprintf(BtnMensaje.Btn2, "         ");
                sprintf(BtnMensaje.Btn3, "         ");
                sprintf(BtnMensaje.Btn4, " Menu   ");
                BtnMensaje.GetValue = 0;   //  Por  default  inicia  con  el  voltaje
seleccionado
                Btn_TouchAction(&BtnMensaje);

                setColor(255, 255, 255);
                setBackColor(0x77, 0x99, 0xdd);
                TFT_print("SIN EVENTOS REGISTRADOS    ", 20, 300, 0);
                while (1) {
                        if (digitalRead(IRQ) == 0) {
```

```
                    Touch = Touch_Key();
                    BtnMensaje.GetValue = Touch;      //      Ningun      Boton
Selecionado
                    Btn_TouchAction(&BtnMensaje);

                    if (Touch == 4)
                        main();
                } // fin del if touch
            } // Fin del while eterno
        } // fin del if de eventos

        int grafic = 0, buffer_graf = 0, Cursor = 100, buffer_cursor = 100,
                LENGTH_SAMPLES = 800;
        Graficar_LCD(0, 0);

        while (1) {
            if (digitalRead(IRQ) == 0) {
                Touch = Touch_Key();

                if (Touch == 1) {
                    Cursor = 100;
                    if (grafic < 1)
                        grafic = 0;
                    else
                        grafic--;

                    Graficar_LCD(grafic, buffer_graf);

                    buffer_graf = grafic;

                    setColor(0x00, 0X00, 0X00);
                    setBackColor(0xff, 0xff, 0xff);
                    TFT_print("              ", 50, 360, 0);

                } else if (Touch == 2) {
                    Cursor = 100;
                    if (grafic < (Cont_Event - 1))
                        grafic++;
                    else
                        grafic = Cont_Event - 1;

                    Graficar_LCD(grafic, buffer_graf);
                    buffer_graf = grafic;
```

```
                    setColor(0x00, 0X00, 0X00);
                    setBackColor(0xff, 0xff, 0xff);
                    TFT_print("              ", 50, 360, 0);

                } else if (Touch == 3) {

                    for (i = 0; i < LENGTH_SAMPLES; i++)
                        Data_Memoria.VOLTAJE_FLUC_RMS[i]        =
Transient_vol[grafic][i];

                    Data_Memoria.THD_AMP = grafic;

                    sprintf(mylabel, "TRS%d.CSV", grafic);
                    const char * TEST_FILENAME = mylabel;

                    WRITE_SD_CARD(LENGTH_SAMPLES,
TEST_FILENAME, 4, &Reloj,
                                  &Data_Memoria);

                    setColor(0xff, 0X00, 0X00);
                    setBackColor(0xff, 0xff, 0xff);
                    TFT_print("GRABADO EXITOSO", 50, 360, 0);

                } else if (Touch == 4) {
                    main();

                } else if (Touch == 5) {
                    if (Cursor < 780)
                        Cursor += 40;
                    else
                        Cursor = 780;

                    Graficar_Cursor(grafic, Cursor, buffer_cursor);
                    buffer_cursor = Cursor;
                } else if (Touch == 6) {
                    Cursor -= 40;
                    if (Cursor < 40)
                        Cursor = 40;

                    Graficar_Cursor(grafic, Cursor, buffer_cursor);
                    buffer_cursor = Cursor;
                }

                else if (Touch == 7) {
```

```
                    if (Cursor < 794)
                        Cursor++;
                    else
                        Cursor = 794;

                    Graficar_Cursor(grafic, Cursor, buffer_cursor);
                    buffer_cursor = Cursor;
                } else if (Touch == 8) {
                    Cursor--;
                    if (Cursor < 6)
                        Cursor = 6;

                    Graficar_Cursor(grafic, Cursor, buffer_cursor);
                    buffer_cursor = Cursor;
                }

                BtnMensaje.GetValue = Touch;       // Ningun Boton Selecionado
                Btn_TouchAction(&BtnMensaje);
            } // fin del if touch

        } // fin del while
}

void Graficar_LCD(int wrt_graf, int dld_graf) {

        int LENGTH_SAMPLES = 1600, i;

        sprintf(Mensajes_LCD.str_Herz_harmonic, "GRAFICA: %d  ", wrt_graf + 1);

        setColor(0, 0, 0);
        setBackColor(0xff, 0xff, 0xff);
        TFT_print(Mensajes_LCD.str_Herz_harmonic, 50, 65, 0);

        setColor(0xd2, 0xd2, 0xd2);
        int j;
        for (j = 50; j < 430; j = j + 20) {
            for (i = 10; i < 800; i += 80) {
                drawLine(i, j, i, j + 2);
            }
        }

        for (j = 10; j < 800; j += 20) {
            for (i = 50; i < 430; i += 80)
```

```
                    drawLine(j, i, j, i + 2);
            }
            /*
             Conf_Osciloscope.Inteporl_SENSIBILIDAD = (int)
READ_FLOAT_EXT_EEPROM(30);
             int pixel_sensibilidad = Conf_Osciloscope.Inteporl_SENSIBILIDAD;
             */
            for (i = 0; i < LENGTH_SAMPLES / 2; i++) {

                Voltaje.wave_sampling[i] = Transient_vol[wrt_graf][i];

                float xy_acondition = Transient_vol[wrt_graf][i] * 70
                        / Conf_Osciloscope.Rango_conf.Voltaje_V;
                int ypoint = (int) (xy_acondition + 200);
                vol_xy_plot[i] = ypoint;

                xy_acondition = Transient_vol[dld_graf][i] * 70
                        / Conf_Osciloscope.Rango_conf.Voltaje_V;
                ypoint = (int) (xy_acondition + 200);
                vol_xy_buff[i] = ypoint;
            }

            int k, m_pendiente, pixel_sensibilidad = 2;
            for (k = 6; k < LENGTH_SAMPLES / 2 - 6; k++) {

                /* ------------------------------------------------------------------
                 ** Interpolacion de la señal para borrar en la pantalla
                 ** ------------------------------------------------------------------ */
                setColor(0xff, 0xff, 0xff);
                drawPixel(k, vol_xy_buff[k]);
                drawPixel(k, vol_xy_buff[k] + 1);
                m_pendiente = abs(vol_xy_buff[k + 1] - vol_xy_buff[k]);
                if (m_pendiente > pixel_sensibilidad) {
                    drawLine(k, vol_xy_buff[k], k + 1, vol_xy_buff[k + 1]);
                    drawLine(k, vol_xy_buff[k] + 1, k + 1, vol_xy_buff[k + 1] + 1);
                }

                /* ------------------------------------------------------------------
                 ** Interpolacion de la señal para mostrar en la pantalla
                 ** ------------------------------------------------------------------ */
                setColor(0xff, 0, 0);
                drawPixel(k, vol_xy_plot[k]);
                drawPixel(k, vol_xy_plot[k] + 1);
                m_pendiente = abs(vol_xy_plot[k + 1] - vol_xy_plot[k]);
```

```
                if (m_pendiente > pixel_sensibilidad) {
                        drawLine(k, vol_xy_plot[k], k + 1, vol_xy_plot[k + 1]);
                        drawLine(k, vol_xy_plot[k] + 1, k + 1, vol_xy_plot[k + 1] + 1);
                }

        }
        /*DIBUJO EJE X*/
        setColor(0x0, 0x0, 0x0);
        drawLine(5, 200, 795, 200);

        arm_max_f32(Voltaje.wave_sampling, LENGTH_SAMPLES / 2,
&Voltaje.valor_MAX,
                        &Voltaje.max_marks1);
        arm_min_f32(Voltaje.wave_sampling, LENGTH_SAMPLES / 2,
&Voltaje.valor_MIN,
                        &Voltaje.min_marks2);
        arm_rms_f32(Voltaje.wave_sampling, LENGTH_SAMPLES / 2,
&Voltaje.valor_RMS);
        Voltaje.valor_RMS = (Voltaje.valor_RMS * 3.3 / 4095) / 0.0014039;
        Voltaje.valor_MAX = (-Voltaje.valor_MAX * 3.3 / 4095) / 0.0014039;
        Voltaje.valor_MIN = (-Voltaje.valor_MIN * 3.3 / 4095) / 0.0014039;

        // DEBIDOA POSICIONES DE LA GRAFICA, ES INVERTIDO, EL MAXIMO
ES MINIMO Y VICEVERZA
        float Tmn = (float) ((Voltaje.max_marks1 * 0.08335e-3) * 1e3);
        float Tmx = (float) ((Voltaje.min_marks2 * 0.08335e-3) * 1e3);

        sprintf(Msn_Vol.str_MIN, "%3.2f VMAX ", Voltaje.valor_MIN);
        sprintf(Msn_Vol.str_MAX, "%3.2f VMIN ", Voltaje.valor_MAX);
        sprintf(Mensajes_LCD.str_RMS, "%3.2f Vrms  ", Voltaje.valor_RMS);

        sprintf(Msn_Vol.str_MEAN, "- %2.4f ms  ", Tmn);
        sprintf(Msn_Vol.str_RMS, "- %2.4f ms  ", Tmx);

        day = (int) Reloj.day[wrt_graf];
        mth = (int) Reloj.mes[wrt_graf];
        year = (int) Reloj.anio[wrt_graf];
        hr = (int) Reloj.hora[wrt_graf];
        min = (int) Reloj.min[wrt_graf];
        sec = (int) Reloj.seg[wrt_graf];

        sprintf(Mensaje, "%02u/%02u/%02u ", day, mth, year);
        sprintf(Mensaje6, "%02u:%02u:%02u ", hr, min, sec);
```

```
        setColor(0, 0, 0);
        setBackColor(0xff, 0xff, 0xff);
        // ´PRIMERA FILA DE DATOS
        TFT_print(Mensaje6, 50, 380, 0);
        TFT_print(Mensaje, 50, 400, 0);
// SEGUNDA FILA DE DATOS
        TFT_print(Msn_Vol.str_MIN, 200, 380, 0);
        TFT_print(Msn_Vol.str_MAX, 200, 400, 0);
        // TERCERA FILA DE DATOS
        TFT_print(Msn_Vol.str_RMS, 420, 380, 0);
        TFT_print(Msn_Vol.str_MEAN, 420, 400, 0);

        TFT_print(Mensajes_LCD.str_RMS, 580, 65, 0);

}

void Graficar_Cursor(int wrt_graf, int cursor, int buff_curso) {

        float time_us = (float) ((cursor * 0.08335e-3) * 1e6);
        float Vcursor = Transient_vol[wrt_graf][cursor];
        Vcursor = (-Vcursor * 3.3 / 4095) / 0.0014039;

        sprintf(Mensajes_LCD.str_Hz, "Vcsr: %2.2f    ", Vcursor);

        if (time_us < 1000)
                sprintf(Mensajes_LCD.str_THD, "t(us): %4.2f    ", time_us);
        else
                sprintf(Mensajes_LCD.str_THD, "t(ms): %2.4f    ", time_us / 1000);

        // DIBUJA LOS PINES DEL FONDO
        setColor(0, 0, 0);
        setBackColor(0xff, 0xff, 0xff);
        int j;
        for (j = 50; j < 430; j = j + 20) {
                for (i = 10; i < 800; i += 80) {
                        drawLine(i, j, i, j + 2);
                }
        }

        for (j = 10; j < 800; j += 20) {
                for (i = 50; i < 430; i += 80)
                        drawLine(j, i, j, i + 2);
        }
```

```
        setColor(0x00, 0x00, 0xd2);
        //drawLine(cursor, 111, cursor, 410);
        drawLine(cursor, 100, cursor, 360);
        // GRAFICA EL CURSOR EN LA PANTALLA Y GENERA EL
MOVIMIENTO
        if (buff_curso != cursor) {
                setColor(0xff, 0xff, 0xff);
                drawLine(buff_curso, 100, buff_curso, 360);

                float xy_acondition = Transient_vol[wrt_graf][buff_curso] * 70
                                / Conf_Osciloscope.Rango_conf.Voltaje_V;
                int ypoint = (int) (xy_acondition + 200);
                xy_acondition = Transient_vol[wrt_graf][buff_curso - 1] * 70
                                / Conf_Osciloscope.Rango_conf.Voltaje_V;
                int ypoint_1 = (int) (xy_acondition + 200);

                setColor(0xff, 0, 0);
                drawPixel(buff_curso, ypoint);
                drawPixel(buff_curso, ypoint + 1);
                float m_pendiente = abs(ypoint - ypoint_1);
                if (m_pendiente > 2) {
                        drawLine(buff_curso - 1, ypoint_1, buff_curso, ypoint);
                        drawLine(buff_curso - 1, ypoint_1 + 1, buff_curso, ypoint + 1);
                }

                setColor(0x0, 0x0, 0x0);
                drawPixel(buff_curso, 200);
                //      drawLine(5, 200, 795, 200);
        }
        buff_curso = cursor;
        // MENSAJES EN PANTALLA
        setColor(0, 0, 0);
        setBackColor(0xff, 0xff, 0xff);
        TFT_print(Mensajes_LCD.str_Hz, 300, 65, 0);
        TFT_print(Mensajes_LCD.str_THD, 300, 85, 0);

        // DIBUJA RECTA DEL EJE
}
```

```
void TRANSIENTS_BORRAR_LCD() {
        int LENGTH_SAMPLES = 1600;
        Conf_Osciloscope.Inteporl_SENSIBILIDAD = (int)
READ_FLOAT_EXT_EEPROM(30);
        int pixel_sensibilidad = Conf_Osciloscope.Inteporl_SENSIBILIDAD;

        setColor(0xff, 0xff, 0xff);
        setBackColor(0xff, 0xff, 0xff);
        TFT_print(Mensajes_LCD.str_RMS, 60, 45, 0);

        TFT_print("          ", 50, 400, 0);
        TFT_print("          ", 200, 400, 0);

        drawLine(5, 200, 795, 200);

        int j;
        for (j = 50; j < 430; j = j + 10) {
                for (i = 10; i < 800; i += 50) {
                        drawLine(i, j, i, j + 2);
                }
        }

        for (j = 10; j < 800; j += 10) {
                for (i = 50; i < 430; i += 50)
                        drawLine(j, i, j, i + 2);
        }
        int k, m_pendiente;
        for (k = 6; k < LENGTH_SAMPLES / 2 - 6; k++) {

                setColor(0xff, 0xff, 0xff);
                drawPixel(k, vol_xy_buff[k]);
                drawPixel(k, vol_xy_buff[k] + 1);
                m_pendiente = abs(vol_xy_buff[k + 1] - vol_xy_buff[k]);
                if (m_pendiente > pixel_sensibilidad) {
                        drawLine(k, vol_xy_buff[k], k + 1, vol_xy_buff[k + 1]);
                        drawLine(k, vol_xy_buff[k] + 1, k + 1, vol_xy_buff[k + 1] + 1);
                }
        }
}
```

APENDICE G
PROGRAMA OSCILOSCOPIO

```
void SCOPE() {

        uint8 Touch = 1;
        int LENGTH_SAMPLES = 1600;    // el doble del ancho de la pantalla
//      setColor(0xff, 0xff, 0xff);
//      fillRect(6, 46, 794, 430);

        setColor(0xd2, 0xd2, 0xd2);
        fillRect(0, 0, 799, 40);

        setColor(0, 0, 0);
        setBackColor(0xd2, 0xd2, 0xd2);
        TFT_print("OSCILOSCOPIO", 20, 10, 0);

        /* ------------------------------------------------------------
         **      Inicializa la barra inferior del Menu para el control
         ** ------------------------------------------------------------ */
/*      setColor(255, 255, 255);
        setBackColor(0x77, 0x99, 0xdd);
        TFT_print(" SETUP  ", 20, 445, 0);
        TFT_print(" RANGO ", 220, 445, 0);
        TFT_print("TRIGGER  ", 420, 445, 0);
        TFT_print(" MOVE  ", 640, 445, 0);
        */
        sprintf(BtnMensaje.Btn1, " SETUP  ");
        sprintf(BtnMensaje.Btn2, " RANGO ");
        sprintf(BtnMensaje.Btn3, "TRIGGER  ");
        sprintf(BtnMensaje.Btn4, " MOVE  ");
        BtnMensaje.GetValue = 0;    // Por default inicia con el voltaje seleccionado
        Btn_TouchAction(&BtnMensaje);


        /* ------------------------------------------------------------------
         **      Valores Iniciales del equipo
         ** ------------------------------------------------------------ */

        Conf_Osciloscope.Corriente_factor = (int) READ_FLOAT_EXT_EEPROM(1);
        Conf_Osciloscope.Rango_conf.Amperio_A = (int)
READ_FLOAT_EXT_EEPROM(10);
        Conf_Osciloscope.Rango_conf.Voltaje_V = (int)
READ_FLOAT_EXT_EEPROM(20);
```

```
Conf_Osciloscope.Inteporl_SENSIBILIDAD = (int) READ_FLOAT_EXT_EEPROM(30);

Conf_Osciloscope.Trigger_on_off = 0;
Conf_Osciloscope.HOLD_RUN = 1;
Conf_Osciloscope.Submenus = 0;
Conf_Osciloscope.Mover_up_dawn_amp = 340;
Conf_Osciloscope.Mover_up_dawn_vol = 150;

Conf_Osciloscope.Rango_conf.Tiempo_s = 0.08335e-3;
set_sample_period(Conf_Osciloscope.Rango_conf.Tiempo_s);
Bluetooh_ON_OFF = Read_25LC(40);
if (Bluetooh_ON_OFF == 1) {
        drawBitmap(400, 5, 24, 33, logo_bluetooth, 1);
        gioSetBit(hetPORT1, 26, 0);          //Bluetooth Pin_control
}
Touch = 0;
while (1) {

        if (digitalRead(IRQ) == 0) {
                Touch = Touch_Key();

                if (Touch == 1) {
                        Conf_Osciloscope.Submenus = 1;
                        Setup_scope();

                } else if (Touch == 2) {
                        Conf_Osciloscope.Submenus = 2;
                        Rango_scope();

                } else if (Touch == 3) {
                        Conf_Osciloscope.Submenus = 3;
                        Trigger_scope();
                } else if (Touch == 4) {
                        Conf_Osciloscope.Submenus = 4;

                        Move_scope();
                } else if (Touch == 5) {
                        if (Conf_Osciloscope.Submenus == 1) {//expande o contrae el tiempo de muestreo
                                Conf_Osciloscope.Rango_conf.Tiempo_s -= 0.005e-3;
                                if (Conf_Osciloscope.Rango_conf.Tiempo_s < 0)
```

```
                                        Conf_Osciloscope.Rango_conf.Tiempo_s =
0.01e-3;

        set_sample_period(Conf_Osciloscope.Rango_conf.Tiempo_s);
                                }
                                if (Conf_Osciloscope.Submenus == 2)       //Atenua       o
expande la señal arriba y abajo, Voltaje
                                        Conf_Osciloscope.Rango_conf.Voltaje_V -= 50;
                                if (Conf_Osciloscope.Submenus == 4)       //Mueve        la
señal arriba y abajo, Voltaje
                                        Conf_Osciloscope.Mover_up_dawn_vol -= 10;

                        } else if (Touch == 6) {
                                if  (Conf_Osciloscope.Submenus  ==  1)  {//expande  o
contrae el tiempo de muestreo
                                        Conf_Osciloscope.Rango_conf.Tiempo_s          +=
0.005e-3;

        set_sample_period(Conf_Osciloscope.Rango_conf.Tiempo_s);
                                }

                                if (Conf_Osciloscope.Submenus == 2)       //Atenua       o
expande la señal arriba y abajo, Voltaje
                                        Conf_Osciloscope.Rango_conf.Voltaje_V += 50;
                                if (Conf_Osciloscope.Submenus == 4)       //Mueve        la
señal arriba y abajo, Voltaje
                                        Conf_Osciloscope.Mover_up_dawn_vol += 10;

                        } else if (Touch == 7) {
                                if (Conf_Osciloscope.Submenus == 1)       //  CONGELA
LA PANTALLA
                                        Conf_Osciloscope.HOLD_RUN = 0;
                                if (Conf_Osciloscope.Submenus == 2)       //Atenua       o
expande la señal arriba y abajo, Corriente
                                        Conf_Osciloscope.Rango_conf.Amperio_A -= 50;
                                else  if  (Conf_Osciloscope.Submenus  ==  3)//Activa  el
Trigger de la Señal
                                        Conf_Osciloscope.Trigger_on_off = 1;
                                if (Conf_Osciloscope.Submenus == 4)       //Mueve        la
señal arriba y abajo, corriente
                                        Conf_Osciloscope.Mover_up_dawn_amp -= 10;

                        } else if (Touch == 8) {
```

```
                    if (Conf_Osciloscope.Submenus == 1)        // CONGELA
LA PANTALLA
                        Conf_Osciloscope.HOLD_RUN = 1;
                    if (Conf_Osciloscope.Submenus == 2)        //Atenua      o
expande la señal arriba y abajo, Corriente
                        Conf_Osciloscope.Rango_conf.Amperio_A += 50;
                    else  if  (Conf_Osciloscope.Submenus  ==  3)//Activa  el
Trigger de la Señal
                        Conf_Osciloscope.Trigger_on_off = 0;
                    if  (Conf_Osciloscope.Submenus  ==  4)  //Mueve  la  señal
arriba y abajo, corriente
                        Conf_Osciloscope.Mover_up_dawn_amp += 10;
                }

                BtnMensaje.GetValue = Touch;       // Ningun Boton Selecionado
                Btn_TouchAction(&BtnMensaje);
            }// fin de if touch

            Touch = 0;
            if (Conf_Osciloscope.HOLD_RUN == 1) {
                /*
                 if (Conf_Osciloscope.Trigger_on_off == 1) {
                 loop0: if (gioGetBit(hetPORT2, 6) != 1)
                 goto loop0;
                 loop1: if (gioGetBit(hetPORT2, 6) != 0)
                 goto loop1;
                 }*/

                for (i = 0; i < LENGTH_SAMPLES; i++) {
                    adcStartConversion(adcREG1, adcGROUP1); //Start ADC
conversio
                    while                (!adcIsConversionComplete(adcREG1,
adcGROUP1))
                        ;                  // ConfigutaR halcogen el trigger por
sofware

                    ch_count    =    adcGetData(adcREG1,    adcGROUP1,
&adc_data[0]);
                    ch_count = ch_count;
                    Voltaje.wave_sampling[i] = (adc_data[0].value
                            - adc_data[1].value) + 280;

                    adcStartConversion(adcREG2, adcGROUP1); //Start ADC
conversio
```

```
                while (!adcIsConversionComplete(adcREG2, adcGROUP1))
                        ;               // ConfigutaR halcogen el trigger por sofware

                ch_count = adcGetData(adcREG2, adcGROUP1, &adc_data[0]);
                ch_count = ch_count;
                Ampere.wave_sampling[i] = adc_data[0].value - adc_data[1].value;

                //delay_seg(Tsample);       // Retardo entre muestras de 83microsegundos
                while (get_adc_signal == 0) {
                }
                get_adc_signal = 0;
        }

        /* ----------------------------------------------------------------
         ** Tigger Digital por algoritmo
         ** ---------------------------------------------------------------- */
        int jh = 0, trigger_look = 0, diff_val = 0, centrotrigger =
                        LENGTH_SAMPLES / 2;
        int Dato_Actual = 0, Dato_Anterior = 0;
        if (Conf_Osciloscope.Trigger_on_off == 1) {

                do {
                        Dato_Actual = Voltaje.wave_sampling[centrotrigger + jh + 1];
                        Dato_Anterior = Voltaje.wave_sampling[centrotrigger + jh];
                        diff_val = Dato_Actual - Dato_Anterior;

                        if (Dato_Actual > 10 && diff_val > 20)
                                trigger_look = 1;
                        else {
                                trigger_look = 0;
                        }

                        jh++;
                } while ((trigger_look == 0) && (jh < LENGTH_SAMPLES / 2));

        }
```

```
/* ----------------------------------------------------------------
 ** Calculo del Maximo y Minimo de la señal de voltaje
 ** ---------------------------------------------------------------- */
arm_max_f32(Voltaje.wave_sampling, LENGTH_SAMPLES,
                &Voltaje.valor_MAX, &Voltaje.max_marks1);
arm_min_f32(Voltaje.wave_sampling, LENGTH_SAMPLES,
                &Voltaje.valor_MIN, &Voltaje.min_marks2);

frec_line = 0.00;
int Amplitud_wave = Voltaje.valor_MAX - Voltaje.valor_MIN;

if (Amplitud_wave > 200)
        frec_line = (double) (1e6 / T_prom);//RTI0, cada 10ms muestrea la frecuencia de linea.

/* ----------------------------------------------------------------
 ** Call the RMS function to calculate rms marks among numStudents
 ** ---------------------------------------------------------------- */
arm_rms_f32(Voltaje.wave_sampling, LENGTH_SAMPLES,
                &Voltaje.valor_RMS);
Voltaje.valor_RMS = (Voltaje.valor_RMS * 3.3 / 4095) / 0.0014039;
arm_rms_f32(Ampere.wave_sampling, LENGTH_SAMPLES,
                &Ampere.valor_RMS);
Ampere.valor_RMS = (Ampere.valor_RMS * 3.3 / 4095)
                * Conf_Osciloscope.Corriente_factor;
/* ----------------------------------------------------------------
 **Acondicionamiento de la señal para graficar en pantalla
 ** ---------------------------------------------------------------- */

int sk = 0;
for (i = jh; i < (LENGTH_SAMPLES / 2) + jh; i++) {
        float xy_acondition = Voltaje.wave_sampling[i] * 50
                        / Conf_Osciloscope.Rango_conf.Voltaje_V;
        int ypoint = (int) (xy_acondition
                        + Conf_Osciloscope.Mover_up_dawn_vol);
        if (ypoint < 65)
                ypoint = 66;
        //else if (ypoint > 230)
        //ypoint = 229;
        vol_xy_plot[sk] = ypoint;
```

```
                        xy_acondition = Ampere.wave_sampling[i] * 50
                                        /
Conf_Osciloscope.Rango_conf.Amperio_A;

                        ypoint = (int) (xy_acondition
                                        +
Conf_Osciloscope.Mover_up_dawn_amp);
                        //if (ypoint < 256)
                        //        ypoint = 257;
                        if (ypoint > 423)
                                ypoint = 422;
                        amp_xy_plot[sk] = ypoint;
                        sk++;
                }

                sprintf(Mensajes_LCD.str_RMS,       "%3.2f       V              ",
Voltaje.valor_RMS);
                sprintf(Mensajes_LCD.str_Hz, "%2.2f Hz  ", frec_line);
                sprintf(Mensajes_LCD.str_Amplitud_harmonic, "%3.2f A  ",
                        Ampere.valor_RMS);

                setColor(0, 0, 0);
                setBackColor(0xff, 0xff, 0xff);
                TFT_print(Mensajes_LCD.str_RMS, 50, 45, 0);
                TFT_print(Mensajes_LCD.str_Hz, 400, 45, 0);
                TFT_print(Mensajes_LCD.str_Amplitud_harmonic, 50, 236, 0);

                int k, m_pendiente;
                int j;
                setColor(0xd2, 0xd2, 0xd2);
                for (j = 50; j < 430; j = j + 10) {
                        for (i = 10; i < 800; i += 50) {
                                drawLine(i, j, i, j + 2);
                        }
                }

                for (j = 10; j < 800; j += 10) {
                        for (i = 50; i < 430; i += 50)
                                drawLine(j, i, j, i + 2);
                }

                for (k = 6; k < 794; k++) {
```

```
                        /* ------------------------------------------------------------------
--
                         ** Interpolacion de la señal para borrar en la pantalla
                         ** ------------------------------------------------------------------
- */

                        setColor(0xff, 0xff, 0xff);
                        drawPixel(k, vol_xy_buff[k]);
                        drawPixel(k, vol_xy_buff[k] + 1);
                        m_pendiente = abs(vol_xy_buff[k + 1] - vol_xy_buff[k]);
                        if                      (m_pendiente                    >
Conf_Osciloscope.Inteporl_SENSIBILIDAD) {
                                drawLine(k, vol_xy_buff[k], k + 1, vol_xy_buff[k +
1]);
                                drawLine(k, vol_xy_buff[k] + 1, k + 1,
                                        vol_xy_buff[k + 1] + 1);
                        }

                        drawPixel(k, amp_xy_buff[k]);
                        drawPixel(k, amp_xy_buff[k] + 1);
                        m_pendiente     =     abs(amp_xy_buff[k     +     1]     -
amp_xy_buff[k]);
                        if                      (m_pendiente                    >
Conf_Osciloscope.Inteporl_SENSIBILIDAD) {
                                drawLine(k,     amp_xy_buff[k],     k     +     1,
amp_xy_buff[k + 1]);
                                drawLine(k, amp_xy_buff[k] + 1, k + 1,
                                        amp_xy_buff[k + 1] + 1);
                        }

                        /* ------------------------------------------------------------------
--
                         ** Interpolacion de la señal para mostrar en la pantalla
                         ** ------------------------------------------------------------------
- */
                        setColor(0xff, 0, 0);
                        drawPixel(k, vol_xy_plot[k]);
                        drawPixel(k, vol_xy_plot[k] + 1);
                        m_pendiente = abs(vol_xy_plot[k + 1] - vol_xy_plot[k]);
                        if                      (m_pendiente                    >
Conf_Osciloscope.Inteporl_SENSIBILIDAD) {
                                drawLine(k, vol_xy_plot[k], k + 1, vol_xy_plot[k +
1]);
                                drawLine(k, vol_xy_plot[k] + 1, k + 1,
```

```
                                        vol_xy_plot[k + 1] + 1);
                        }
                        setColor(0, 0, 0xfa);
                        drawPixel(k, amp_xy_plot[k]);
                        drawPixel(k, amp_xy_plot[k] + 1);
                        m_pendiente = abs(amp_xy_plot[k + 1] - amp_xy_plot[k]);
                        if (m_pendiente > Conf_Osciloscope.Inteporl_SENSIBILIDAD) {
                                drawLine(k, amp_xy_plot[k], k + 1, amp_xy_plot[k + 1]);
                                drawLine(k, amp_xy_plot[k] + 1, k + 1,
                                        amp_xy_plot[k + 1] + 1);
                        }

                }
                /*Almacenamiento en buffer de datos para una proxima comparacion*/
                for (i = 0; i < LENGTH_SAMPLES; i++) {
                        vol_xy_buff[i] = vol_xy_plot[i];
                        amp_xy_buff[i] = amp_xy_plot[i];
                }

        }
    }
}
```

APENDICE H
PROGRAMA DE CONFIGURACIONES EQUIPO

```
void INTRUMENTS_SETUP() {

        setColor(0xff, 0xff, 0xff);
        fillRect(400, 100, 400 + 209, 100 + 164);

        drawBitmap(500, 100, 147, 150, setup_logo, 1);

        setColor(0xd2, 0xd2, 0xd2);
        fillRect(0, 0, 799, 40);
        setColor(0x22, 0x77, 0xbb);
        drawRect(15, 58, 380, 58 + 22);
        MENU_Instruments_Setup();
        Menu_CSelect = 0;
        Bluetooh_ON_OFF = Read_25LC(40);
        if (Bluetooh_ON_OFF == 1)
                drawBitmap(400, 5, 24, 33, logo_bluetooth, 1);
        while (1) {

                flag = digitalRead(IRQ);
                if (flag == 0) {

                        Touch = Touch_Key();

                        if (Touch == 2) {
                                if (Menu_CSelect > 0)
                                        Menu_CSelect--;
                                else
                                        Menu_CSelect = 0;

                        } else if (Touch == 3) {
                                if (Menu_CSelect < 3)
                                        Menu_CSelect++;
                                else
                                        Menu_CSelect = 3;
                        }

                        if (Touch == 1) {

                                setColor(0xff, 0xff, 0xff);
                                fillRect(500, 100, 500 + 147, 100 + 150);
                                MENU_Instruments_Setup_deled();
```

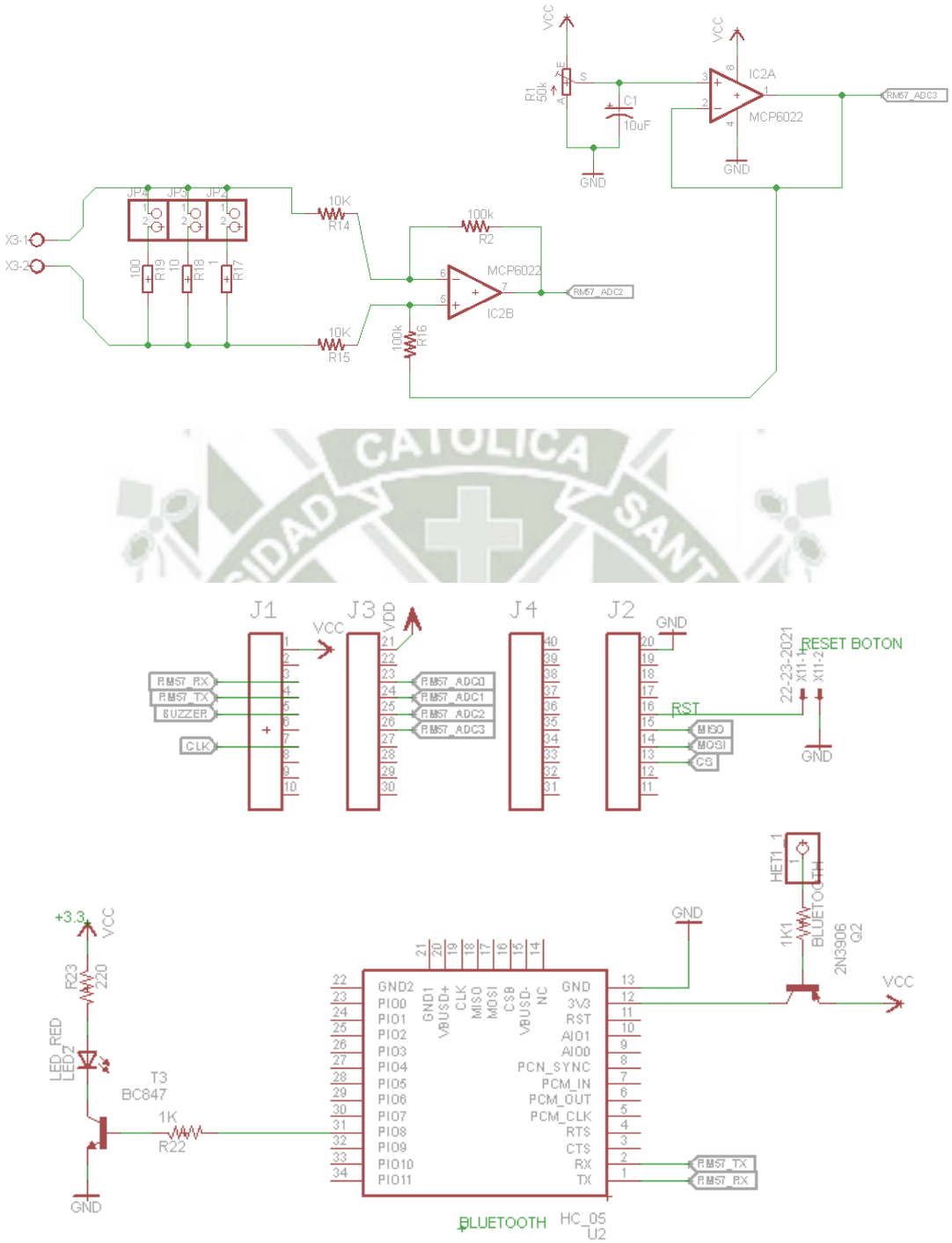
```
                        if (Menu_CSelect == 0)
                                Bluetooh_tx();
                        else if (Menu_CSelect == 1)
                                Set_date_time();
                        else if (Menu_CSelect == 2)
                                Version_Calibration();
                        else if (Menu_CSelect == 3)
                                Administrar_Memoria();

                    }

                    MENU_Instruments_Setup();
                    MMG_SelectMenu(Menu_CSelect);
                }
                screen_clock();

        }

}

void Bluetooh_tx() {

        setColor(255, 255, 255);
        setBackColor(0x77, 0x99, 0xdd);
        TFT_print("          ", 20, 445, 0);
        TFT_print("          ", 220, 445, 0);
        TFT_print("          ", 420, 445, 0);
        TFT_print("          ", 640, 445, 0);

        Bluetooh_ON_OFF = Read_25LC(40);
        drawBitmap(50, 50, 160, 160, Bluetoothlogo, 1);

        setColor(0x00, 0x00, 0x00);
        setBackColor(0xff, 0xff, 0xff);
        TFT_print(" Bienvenido al modo Bluetooth ", 220, 60, 0);
        TFT_print(" te permite enviar datos de ", 220, 80, 0);
        TFT_print(" forma Inalambrica a tu  Movil", 220, 100, 0);

        if (Bluetooh_ON_OFF == 1) {
                drawBitmap(400, 5, 24, 33, logo_bluetooth, 1);
                gioSetBit(hetPORT1, 26, 0);          //Bluetooth Pin_control
        }
```

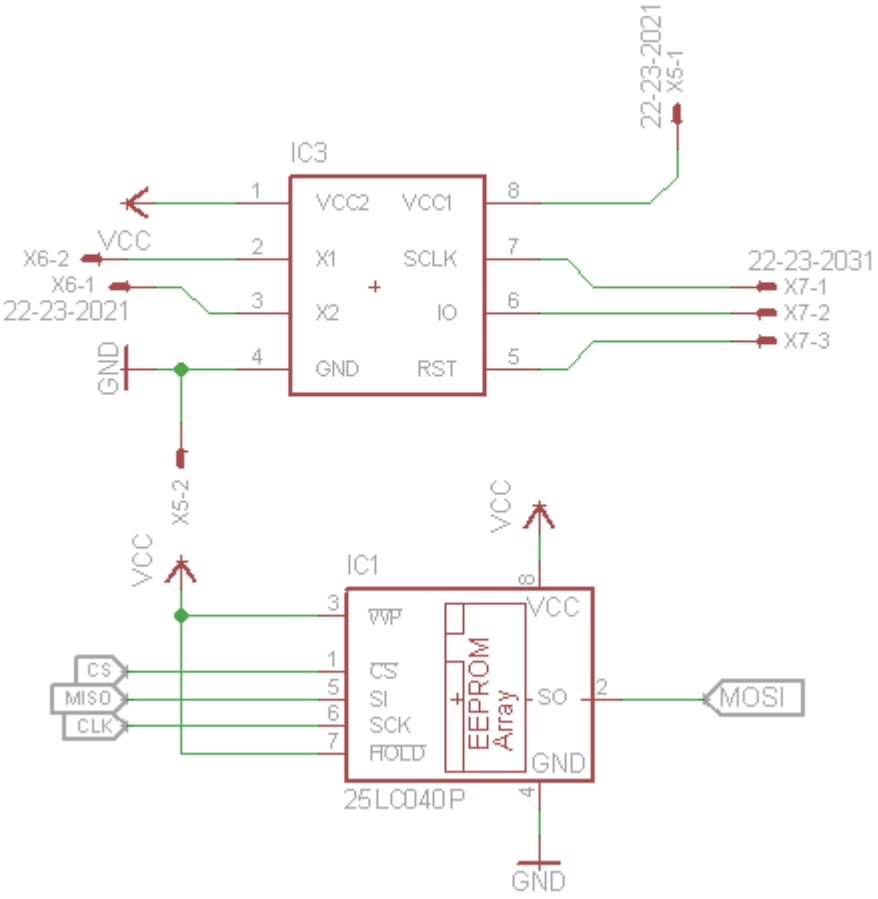
```
        setColor(0xff, 0x00, 0x00);

        TFT_print(" PUSH BUTTON", 220, 150, 0);

        if (Bluetooh_ON_OFF == 0) {
                Bluetooh_ON_OFF = 1;
                TFT_print(" BLUETOOTH ON", 250, 200, 0);
        } else {
                Bluetooh_ON_OFF = 0;
                TFT_print(" BLUETOOTH OFF", 250, 200, 0);
        }

        Write_25LC(40, Bluetooh_ON_OFF);

        while (1) {
                /*flag = digitalRead(IRQ);
                 if (flag == 0) {
                 main();
                 }*/
                screen_clock();

        }

}

void Set_date_time() {
        /* ------------------------------------------------------------------
         **
         ** ------------------------------------------------------------------ */
        setColor(0xd2, 0xd2, 0xd2);
        fillRect(0, 0, 799, 40);
        setColor(0, 0, 0);
        setBackColor(0xd2, 0xd2, 0xd2);
        TFT_print("Configurar Reloj", 20, 10, 0);

        setColor(255, 255, 255);
        setBackColor(0x77, 0x99, 0xdd);
        TFT_print("          ", 20, 445, 0);
        TFT_print("          ", 220, 445, 0);
        TFT_print("          ", 420, 445, 0);
        TFT_print("          ", 640, 445, 0);

        setColor(0, 0, 0);
        setBackColor(0xff, 0xff, 0xff);
```

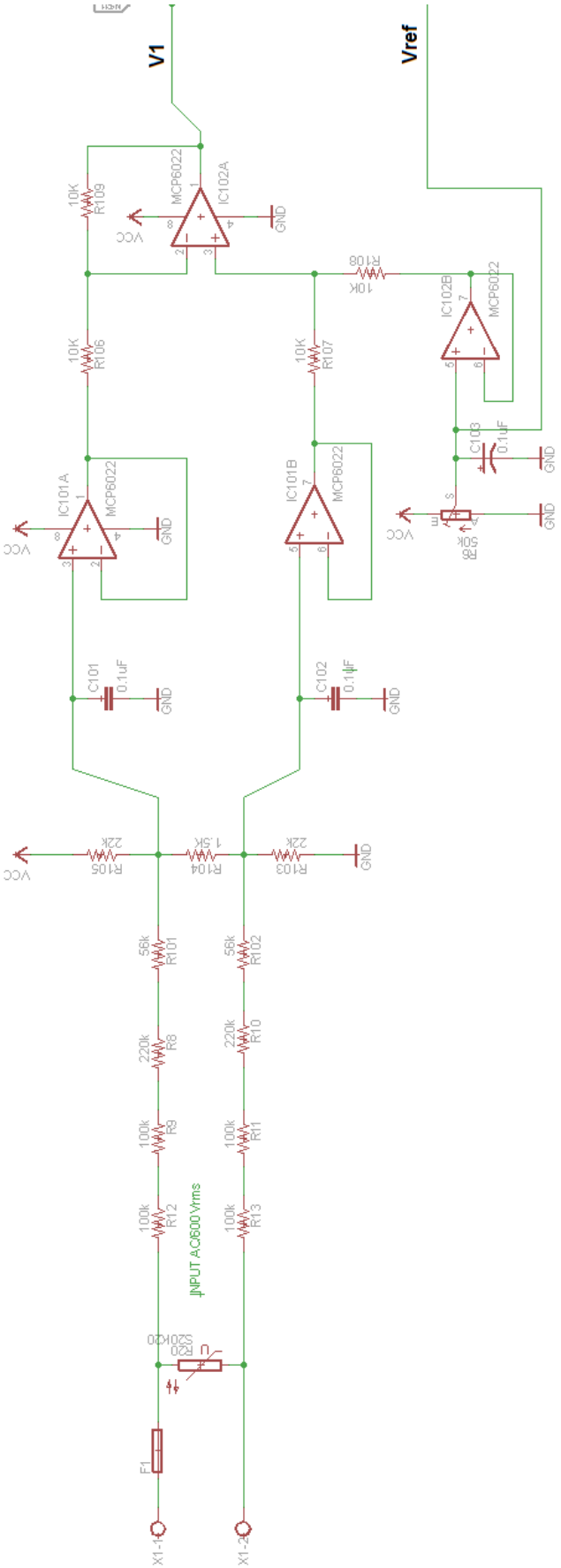
```
        TFT_print("DIA", 200, 60, 0);
        TFT_print("MES", 200, 120, 0);
        TFT_print("YEAR", 200, 180, 0);

        TFT_print("HORA", 550, 60, 0);
        TFT_print("MIN", 550, 120, 0);
        TFT_print("SEC", 550, 180, 0);

        setFont(SevenSegNumFont);

        while (1) {
                day = bcdToDec(read_ds1302(0x87));
                mth = bcdToDec(read_ds1302(0x89));
                year = 2000 + bcdToDec(read_ds1302(0x8d));
                hr = bcdToDec(read_ds1302(0x85));
                min = bcdToDec(read_ds1302(0x83));
                sec = bcdToDec(read_ds1302(0x81));

                sprintf(Mensajes_LCD.str_Amplitud_harmonic, "%02u", day);
                sprintf(Mensajes_LCD.str_Angle_harmonic, "%02u", mth);
                sprintf(Mensajes_LCD.str_Herz_harmonic, "%04u", year);

                sprintf(Mensajes_LCD.str_Hz, "%02u", hr);
                sprintf(Mensajes_LCD.str_Porcentaje_harmonic, "%02u", min);
                sprintf(Mensajes_LCD.str_RMS, "%02u", sec);

                setColor(0, 0, 0);
                setBackColor(0xff, 0xff, 0xff);
                TFT_print(Mensajes_LCD.str_Amplitud_harmonic, 50, 60, 0);
                TFT_print(Mensajes_LCD.str_Angle_harmonic, 50, 120, 0);
                TFT_print(Mensajes_LCD.str_Herz_harmonic, 50, 180, 0);

                TFT_print(Mensajes_LCD.str_Hz, 450, 60, 0);
                TFT_print(Mensajes_LCD.str_Porcentaje_harmonic, 450, 120, 0);
                TFT_print(Mensajes_LCD.str_RMS, 450, 180, 0);

        }

}

void Administrar_Memoria() {
        drawBitmap(600, 100, 138, 180, icon_10270, 1);

        setColor(0xff, 0x00, 0x00);
```

```
        TFT_print("Escribir \'help\' para ayuda ", 20, 60, 0);
        TFT_print("en la terminal Serial de la PC ", 20, 80, 0);
        setColor(0x00, 0x00, 0x00);
        setBackColor(0xff, 0xff, 0xff);
        TFT_print("Bienvenido al modo Comandos [cmd]", 20, 120, 0);
        TFT_print("Una lista de comandos fue enviada", 20, 140, 0);
        TFT_print("a tu Terminal serial de la PC", 20, 160, 0);

        TFT_print("--- Protocolo Serial --- ", 50, 200, 0);
        TFT_print("Speed(baud)  :   115200", 50, 220, 0);
        TFT_print("Data bits    :   8", 50, 240, 0);
        TFT_print("Stop bits    :   1", 50, 260, 0);
        TFT_print("Parity       :   None", 50, 280, 0);
        TFT_print("Flow Control :   None", 50, 300, 0);

        UARTprintf("POWER QUALITY ANALYZER MENU PC\r\n ");
        UARTprintf("Escribir \'help\' para ayuda en pantalla.\r\n");
        UARTprintf("Martin V. TI MVP \r\n");

        SD_Test();

}
```

APENDICE I: DISEÑO DE CIRCUITO Y PLACA IMPRESA EN EAGLE CADSOFT







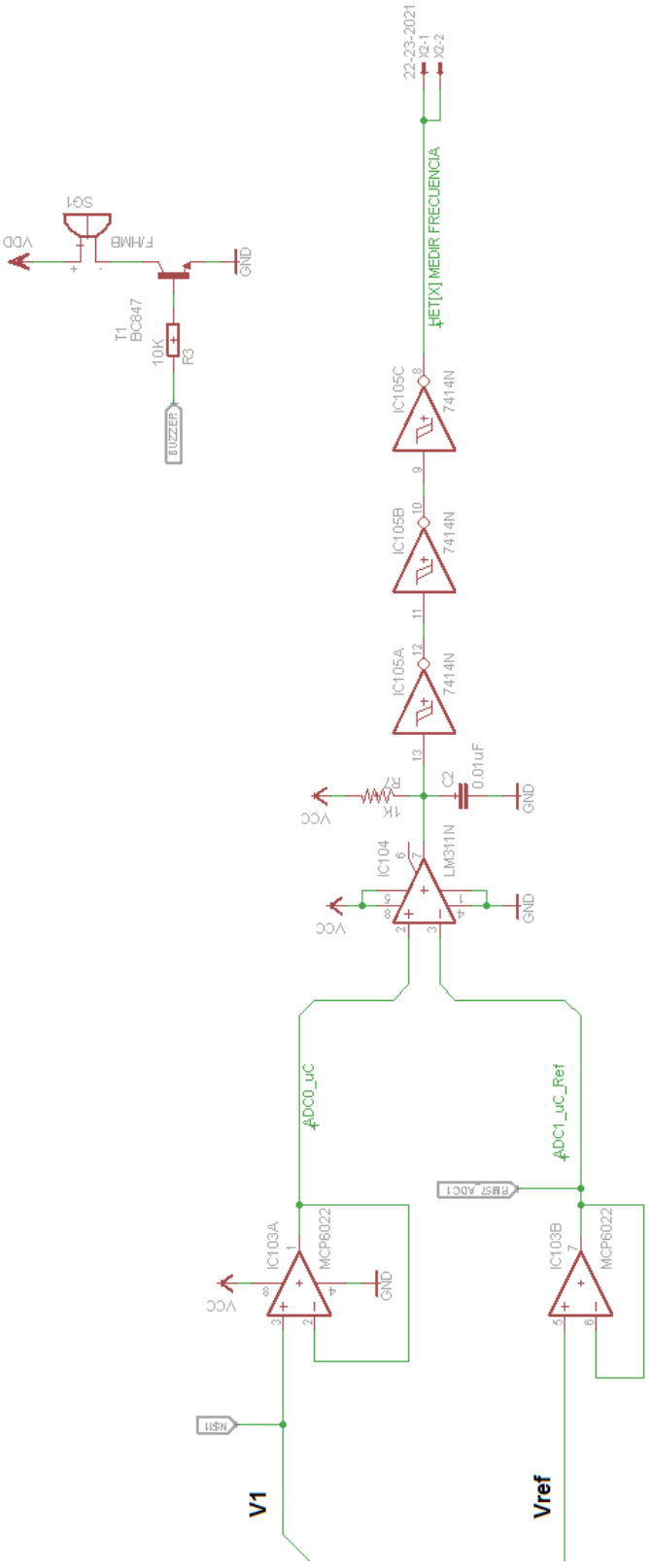


FIGURA I.1 DISEÑO DEL ESQUEMATICO

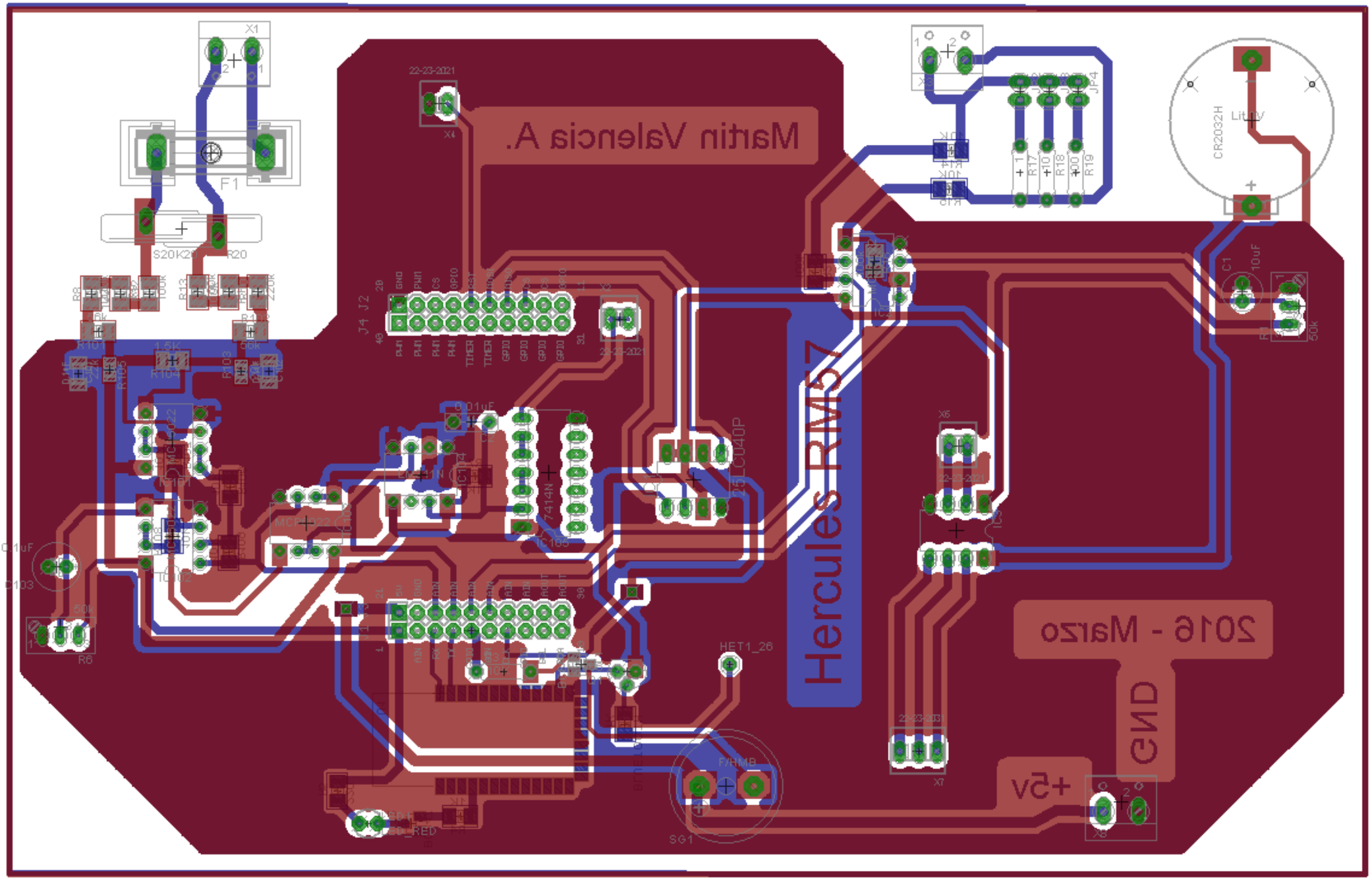


FIGURA I.2 DISEÑO DE LA PLACA IMPLEMENTADA

APENDICE J
FOTOS DE LA IMPLEMENTACION DEL ANALIZADOR DE CALIDAD DE ENERGIA

FIGURA J.1 Vista de Componentes para medición

FIGURA J.2 Vista Superior del A.C.E.

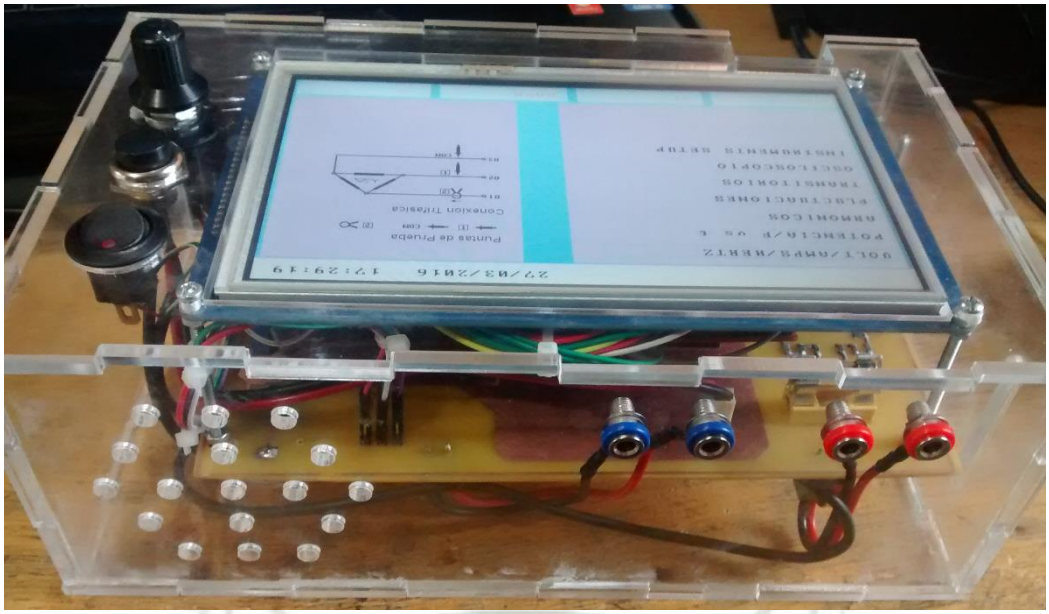


FIGURA J.3 Vista Anterior del A.C.E.

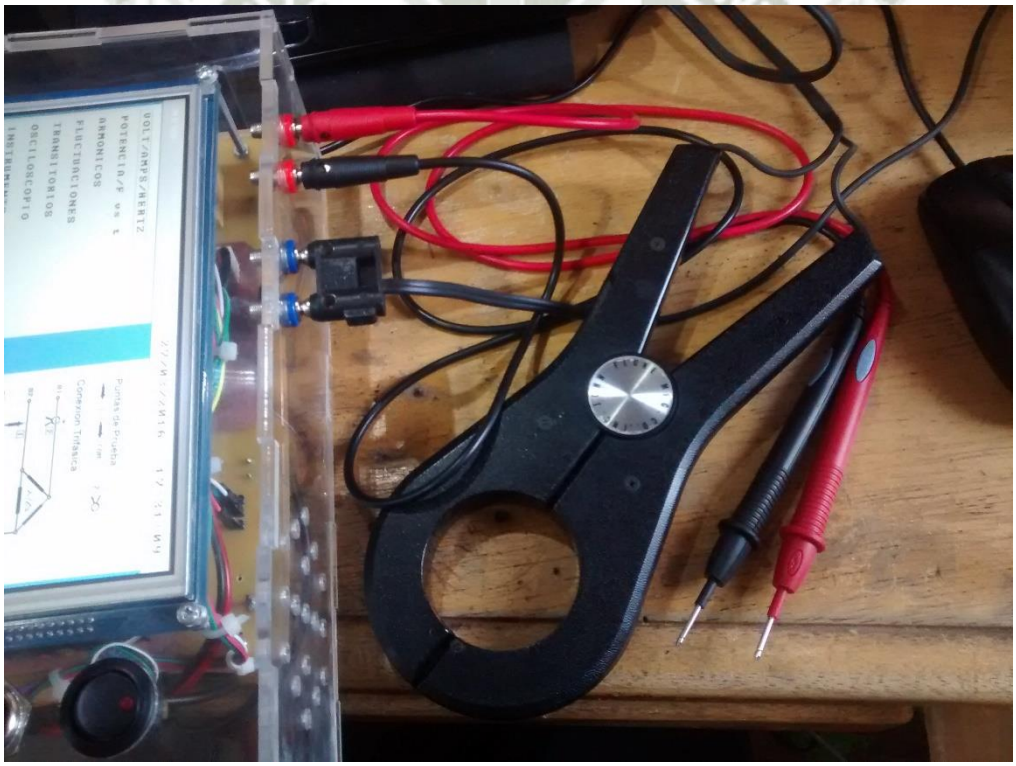
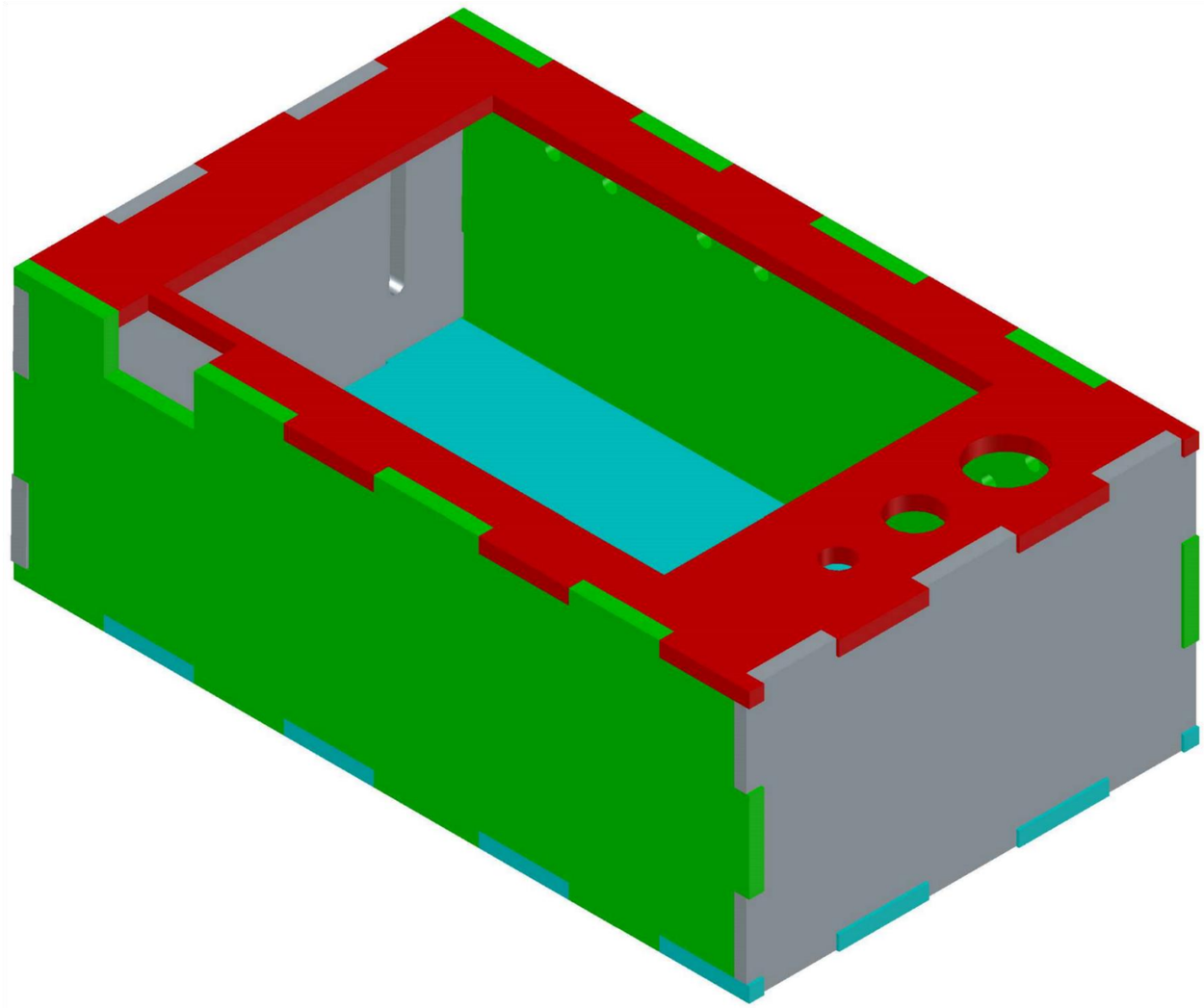


FIGURA J.4 Vista de Conexiones

APENDICE K
DISEÑO 3D DE LA CAJA DE PROTECCION VISTA 1



DISEÑO 3D DE LA CAJA DE PROTECCION VISTA 2