Universidad Autónoma de Querétaro
Facultad de Ingeniería
Campus San Juan del Río

Sistema de bajo costo para monitorización de calidad de la energía basado en computador de placa única.

Opción de titulación
Tesis

Que como parte de los requisitos para obtener el Título de
Ingeniero Electromecánico
Línea Terminal en Mecatrónica

Presenta:
Christian Jonathan Sánchez González

Dirigido por:
Dr. Roque Alfredo Osornio Rios
M. en C. David Alejandro Elvira Ortiz

San Juan del Río, Qro.
Agosto de 2019

Agradecimientos

Primeramente, quiero agradecer a mi familia, a mis padres y hermanos, que sin ellos este momento no habría llegado, por su apoyo incondicional a lo largo de toda la carrera y mis decisiones en la vida.

Quiero agradecer a la universidad y a mis profesores que me aportaron una educación de calidad y compartieron conmigo sus valiosas enseñanzas dándome las herramientas para continuar con mi vida profesional o de investigación.

Y a mis amigos y compañeros, con los cuales compartí años de mi vida, momentos inolvidables, y crecimos juntos, a todos ellos que conozco incluso antes de la universidad, a los amigos con los que inicié la carrera y aquellos que conocí a lo largo de la misma.

Gracias a todos, por tanto.

Índice General

Índice de Figuras

Índice de Tablas

Capítulo 1

1. Introducción

La energía eléctrica es un recurso indispensable que se origina en plantas generadoras, con distintos procesos como puede ser: geotérmica, hidroeléctrica, nuclear o con las nuevas tendencias de energía renovables como son: los aerogeneradores o las celdas fotovoltaicas, pero cualquiera que sea su forma de generación no deja de ser imperfecta, es decir que la energía eléctrica que se utiliza diariamente ya sea en la casa, en las escuelas o cualquier industria, recibe una señal de energía eléctrica que rara vez es la señal ideal, una senoidal pura de 60 Hz y amplitud constante, donde sus razones para no ser perfecta no son únicamente desde su generación, la forma de distribución, las cargas conectadas, la calidad de la instalación eléctrica, son algunas otras formas por la que se ve afectada la señal eléctrica, y es precisamente la imperfección de la energía eléctrica que da pauta para la creación del término de calidad de la energía eléctrica.

La calidad de la energía eléctrica es un término que se ha estudiado desde ya hace algunos años debido a la importancia que ha adquirido principalmente desde la aparición de los sistemas o dispositivos electrónicos y de potencia, los cuales son fuentes conocidas de múltiples disturbios que afectan la calidad de la energía, como agregar armónicos a la red o ser cargas no lineales, y debido a que es cada vez más común la presencia de estos dispositivos en la red eléctrica también es más importante entonces conocer la calidad de la energía de la red. Por esta razón ha tomado importancia el concepto de calidad de la energía eléctrica y de las regulaciones o estándares que la energía debe cumplir para que se considere que tiene un cierto nivel de calidad por ello cada vez son más las instituciones privadas y de gobierno de los diferentes países del mundo que se preocupan por normalizar todo lo relacionado con la calidad de la energía como es el caso de la institución internacional conocida como, el instituto de ingenieros eléctricos y electrónicos

(IEEE) quienes han clasificado los problemas que puede presentar la red eléctrica que afectan en mayor o menor medida la calidad de la energía. Dichos problemas afectan a los dispositivos conectados a la red eléctrica. Cada vez es más común que los dispositivos requieran de cierta calidad energética para su óptimo funcionamiento.

Para llevar a cabo el proceso de medición y monitoreo de la calidad de la energía existen dispositivos comerciales de marcas reconocidas que están aprobados bajo diferentes estándares internacionales como dispositivos para medir la calidad de la energía eléctrica, el problema principal que tienen estos dispositivos es que son muy costosos y en muchos casos son incosteables para particulares o pequeñas empresas, además de ser de arquitectura cerrada lo que quiere decir que no es posible modificar nada del sistema, es decir que el equipo no es personalizable según los requerimientos particulares del usuario, no se tiene una actualización o complementación al equipo para mejorar o crecer en sus algoritmos empleados, de igual manera no se podrían modificar su sistema aun en caso de que existiera alguna modificación a la norma en la que se diseñó el equipo lo cual provoca que el equipo no se encuentre actualizado o se quede fuera de las normas competentes para los equipos de medición de la calidad de la energía.

Por otra parte, aprovechando las nuevas tecnologías como las llamadas computadoras de placa única, las cuales sustituyen los equipos de cómputo convencionales mucho más grandes en volumen y en comparación más difíciles de trasladar, se logra reducir bastante el tamaño de un sistema completo al cual se le introducen las señales, se les aplica el procesamiento adecuado según el tipo de problema con la señal eléctrica y después se puede mostrar todo en una interfaz gráfica, intuitiva y amigable con el usuario a través de una pantalla táctil. Este tipo de dispositivos son muy prácticos debido a que es toda una computadora en la palma de la mano, además incorpora un microcontrolador como el que utilizan las tarjetas de desarrollo de proyectos como lo es Arduino, es decir que se tiene la computadora y el Arduino en la misma placa contando así con dos dispositivos en uno.

En la actualidad ya existen dispositivos para monitorear la calidad de la energía de bajo costo, pero son con tecnologías distintas y se componen de un sistema más grande que requieren computadoras de escritorio además de un sistema conocido como Arreglo de Compuertas Programables en campo (FPGA por sus siglas en inglés) como González (2016) que tiene su sistema para monitoreo de la calidad de la energía en maquinaria de control numérico computarizado (CNC por sus siglas en inglés) utilizando FPGA. Es importante notar que en estos proyectos que no solo se requirió FPGA, sino que también una computadora de escritorio o portátil, para analizar las señales que recibe el FPGA a través de un software en su caso Matlab, el cual es un programa de computadora que le permite al usuario realizar una gran cantidad de cálculos, a través de su lenguaje de programación propio y una vasta cantidad de herramientas y librerías que puede aprovechar el usuario para tareas de múltiples áreas de aplicación, Chapra (2012).

El siguiente trabajo de investigación presenta el desarrollo de un equipo para el monitoreo de calidad de la energía de bajo costo utilizando un computador de placa única, la manera de concretar el proyecto, es primeramente llevar a cabo una investigación de proyectos anteriores, el problema a resolver con el proyecto, la justificación del mismo, los objetivos y la forma de realizarlo, posteriormente se establecerá una fundamentación teórica con los temas necesarios para realizar el proyecto, la metodología a seguir para cumplirlo, los resultados del proyecto de investigación y conclusiones generales sobre el mismo.

1.1 Antecedentes

Monitorización de la Calidad de la Energía

La calidad de la energía eléctrica es un término que se ha estudiado desde ya hace algunos años debido a la importancia que ha adquirido principalmente desde la aparición de los sistemas o dispositivos electrónicos y de potencia, los cuales son fuentes conocidas de múltiples disturbios que afectan la calidad de la energía, como agregar armónicos a la red o ser cargas no lineales, y debido a que es cada vez más común la presencia de estos dispositivos en la red eléctrica también es más

importante entonces conocer la calidad de la energía de la red, y por ello cada vez se realiza más investigación respecto al tema, algunos trabajos desarrollados internacionalmente son como el de Batista et al (2003), quienes presentan un sistema de bajo costo para monitorear la calidad de la energía, utilizando una tarjeta de adquisición de datos MIO-PCI-6024E y una computadora personal con el software de LabVIEW para visualizar las señales adquiridas y realizarles los diferentes procesamientos ya existentes como funciones en LabVIEW. Funciones como la distorsión armónica total (THD por sus siglas en ingles), *p-q Theory* o *Wave Shape* entre otras, donde se concluyó que es posible tener un sistema para monitorear la calidad de la energía, que sea confiable, pero sobre todo que no sea muy costoso como los equipos comerciales. Ángel (2005), diseñó un equipo para monitorear la calidad de la energía enfocándose en las interrupciones y en la caída de tensión de la red eléctrica. En su equipo utiliza una tarjeta de diseño propio para adquirir y acondicionar la señal y luego enviarla por comunicación serial a una PC donde un software se encarga de analizar y mostrar la señal recibida. Su tarjeta utiliza como microcontrolador un controlador de interfaz periférico (PIC por sus siglas en ingles). Lima et al (2008), propusieron una arquitectura de software-hardware para la detección, clasificación y caracterización de eventos de calidad de energía eléctrica, utilizando el algoritmo de *wavelets*, con el fin de crear una arquitectura factible para implementación como parte de un sistema de supervisión de calidad de la energía integral. Jaya et al (2013), presentaron una propuesta de clasificación de perturbaciones referentes a la calidad de la energía utilizando la Transformada S Ortogonal Discreta (DOST por sus siglas en inglés) debido a su eficiencia en comparación con otras transformadas para su caso de aplicación.

Dentro de la UAQ también se han desarrollado diversos trabajos que involucran la calidad de la energía, como es el caso de Ramírez (2012), quien se dio cuenta que era importante realizar un análisis y monitoreo de la calidad de la energía en el laboratorio de Automatización en la Facultad de Ingeniería de la Universidad Autónoma de Querétaro para tener una mayor eficiencia energética de dicho laboratorio, para ello utilizó aparatos de medición comerciales para la calidad de la energía; un analizador de calidad de la energía monofásico Fluke 43B y un

analizador de la calidad de la energía trifásico Amprobe DMIII MultiTest. Mejía (2013), implementó un equipo para monitoreo remoto de la calidad de la energía a través de internet, esto con el fin de poder observar en todo momento y desde cualquier computadora con internet, la calidad energética de una conexión monofásica. González (2016), diseñó un sistema para la detección de dos importantes afectaciones de la calidad de la energía el Sag y el Swell. Ambas son problemas que afectan la red eléctrica de corta duración pero que son importantes para los equipos instalados en la red eléctrica, este trabajo se centró en maquinaria CNC y utilizó un sistema FPGA para la detección de los disturbios. Molano (2013), desarrolló un analizador de la calidad de la energía para motores trifásicos haciendo uso de FPGA para comprobar que es posible tener equipos confiables para el monitoreo y el análisis de las señales eléctricas particularmente en un motor trifásico, sin tener que recurrir a equipos de medición mucho más costosos, y teniendo una confiabilidad de las mediciones bastante aceptable. También dentro de la UAQ, Valtierra et al (2013), desarrollaron una metodología para el monitoreo de armónicos en señales estacionarias y transitorias que satisface el estándar internacional IEC610000-4-7, la tecnología para implementar la metodología propuesta fue llevada a cabo en FPGA para tener un monitoreo continuo y en línea de los sucesos eléctricos.

PQ-UAQ

El dispositivo PQ-UAQ es un dispositivo desarrollado por el grupo de Investigación en el que participé para el desarrollo de esta tesis en la Universidad Autónoma de Querétaro y tiene como objetivo recabar información eléctrica para realizar un análisis de la calidad de la energía, esto a través de sondas de corriente y voltaje que adquieren las señales que posteriormente son estas últimas las que se procesan en un dispositivo FPGA, con todo un circuito adecuado para acondicionar la señal de entrada de la corriente alterna y pueda ser procesada en el FPGA. La tarjeta también cuenta con un dispositivo Bluetooth para enviar las señales y poder visualizarlas en una aplicación de celular, Morales et al (2017), presentan el equipo PQ-UAQ como un sensor inteligente para medir la calidad de

la energía en instalaciones eléctricas. En este mismo trabajo se habla más detalladamente sobre el principio de funcionamiento del PQ-UAQ los componentes internos que forman el equipo y algunas pruebas realizadas en una empresa, un hospital y una casa habitacional. Dzul (2016), en su trabajo de tesis de maestría desarrolla un software para manejar las grandes cantidades de información que se recaban con el PQ-UAQ. Fernández (2016), presenta en su trabajo de tesis la metodología para la calibración del equipo PQ-UAQ utilizando métodos estadísticos para validar las señales adquiridas por el PQ-UAQ y demostrar que es un equipo confiable para la medición de las señales eléctricas. Sin embargo, hace falta que las metodologías de análisis y monitoreo no dependan de una PC o celular. Una por exceso (PC) y otra por falta de recursos (celular).

1.2 Descripción del Problema

El tema de la calidad de la energía no es un tema nuevo, pero sí que ha tomado más relevancia en los últimos años. Esto es debido a que cada vez es más común y a la vez preocupante el estado de la calidad de la energía ya que las cargas electrónicas que son mucho más comunes al día de hoy son cargas no linéales. Este tipo de cargas afectan directamente a la calidad de la energía de la instalación eléctrica donde se conectan los aparatos electrónicos. Además, muchos otros aparatos se han vuelto más sensibles a las perturbaciones de la red eléctrica debido a que, una pobre calidad de la energía eléctrica afecta directamente a su funcionamiento, deteriorando los equipos y reduciendo el tiempo de vida útil para el que originalmente fueron diseñados los equipos en condiciones normales.

Para ello es necesario tener equipos que realicen el monitoreo de dicha calidad de la energía para conocer el estado actual de la instalación eléctrica y en el caso de que la calidad sea deficiente realizar las acciones necesarias para mejorar la calidad de la energía y entonces mejorar la eficiencia eléctrica y asegurar un óptimo funcionamiento de los equipos conectados. Por esta razón se destacan dos problemas por parte de los equipos comerciales:

1. **Costo muy elevado**, los equipos para la medición de la calidad de la energía eléctrica ya sean monofásicos o trifásicos son muy costosos, por ejemplo, un equipo el Fluke 435 el cual es un analizador y medidor de la calidad de la energía trifásica tiene un precio de lista de 10,662 dólares más IVA, Cede (2017).
2. **Arquitectura cerrada**, los equipos comerciales no permiten ningún tipo de modificación a su sistema, las técnicas o algoritmos programados son todo lo que tiene y no pueden modificarse o agregarse nuevas técnicas.

Por otra parte, la Universidad Autónoma de Querétaro ya cuenta con un equipo propio, que está orientado hacia la calidad de la energía, el equipo es mejor conocido como PQ-UAQ donde PQ viene de *Power Quality* o calidad de la energía en español. El PQ-UAQ en realidad es un Datalogger, es decir que únicamente almacena los datos o las señales adquiridas para su posterior análisis o procesamiento. Este equipo tiene cuatro canales de corriente y de voltaje, es decir que permite mediciones de señales trifásicas, las cuales llegan a una tarjeta específicamente diseñada para recibir esas señales. A través de un sistema analógico ha digital (ADS por sus siglas en inglés), se convierten las señales a digitales y luego son procesadas en un FPGA. Finalmente, las señales son almacenadas en una memoria micro SD. El sistema también cuenta con algunos indicadores led y un módulo Bluetooth el cual sirve para conectar el equipo PQ-UAQ con el celular y poder visualizar las señales de entrada en el teléfono celular. Con lo anterior mencionado se destacan entonces cuatro problemas principales que son:

1. **Procesamiento de la señal fuera de línea,** esto quiere decir que el Datalogger PQ-UAQ no tiene un procesamiento de las señales en tiempo real, debido a que las señales que entran al PQ-UAQ requieren ser extraídas normalmente desde la micro SD almacenadas en un banco de datos para posteriormente analizar y procesar las señales en una computadora externa.
2. **Excesiva cantidad de almacenamiento de datos,** debido a que el equipo PQ-UAQ está configurado para que tomé 8000 muestras por segundo de

hasta 7 señales simultáneamente, esto genera una cantidad muy elevada de información que hoy en día supera los 12 terabytes de almacenamiento.

3. **NO cuenta con una visualización de los disturbios,** en este caso si bien ya se mencionó previamente que posee un módulo bluetooth para visualizar la señal de entrada desde un dispositivo externo como un teléfono celular, una computadora o una Tablet, la visualización que se tiene es de la señal de entrada no de una señal que ha recibido un tratamiento digital para determinar su calidad energética o bien de igual manera no muestra la señal enfocándose únicamente en los disturbios que se presenta en lugar de toda la señal.

   NO es un equipo de medición, propiamente tampoco se puede decir que el equipo PQ-UAQ sea un equipo de medición, es únicamente un Datalogger, y para que fuera considerado un equipo de medición se requiere que tenga una visualización y análisis de las señales de entrada en tiempo real.

1.3 Justificación

Teniendo en cuenta la creciente demanda de conocer la calidad de la energía eléctrica, es importante desarrollar un sistema para monitorizar señales eléctricas; que además sea de bajo costo para que éste puede ser accesible para muchas más personas interesadas en conocer y monitorizar la calidad de la energía, y así poder realizar acciones en caso de que se pretenda mejorar la calidad de la energía de la instalación eléctrica, para tener mejor eficiencia energética y aprovechar los beneficios que esto conlleva. Por ello para solucionar los problemas anteriormente planteados se propone un equipo que cuente con las siguientes características:

Bajo Costo: Que el equipo sea de bajo costo es realmente importante ya que la gran mayoría de equipos diseñados para la medición y monitoreo de la calidad de la energía suelen ser muy costosos, por ejemplo, el Fluke 435 para la medición y monitoreo de la calidad de la energía tiene un costo aproximado de 213,240 pesos mexicanos. Por ello se busca obtener un equipo con el mismo

propósito (monitorizar la calidad de la energía), con funciones similares, pero de mucho menor costo. Se estima que sea aproximadamente del 20% del costo de un equipo Fluke 435 es decir, menos de 45,000 pesos mexicanos.

Arquitectura Abierta: Que el sistema sea de arquitectura abierta también representa un beneficio para quienes estén interesados en modificar la funcionalidad de software o adaptar el sistema a sus necesidades. Además, se tiene la opción de hacer crecer el equipo con nuevas técnicas para el análisis de disturbios eléctricos en trabajos posteriores.

Por otra parte, para solucionar los problemas con los que cuenta el Datalogger PQ-UAQ se propone los siguiente:

Selección de las señales de entrada: Con una selección de las señales de entrada en tiempo real, seleccionando únicamente las señales o el periodo de tiempo en que la señal presentó un disturbio, se logrará reducir en gran medida la cantidad de información que se almacena, debido a que mucha de esa información almacenada, no es útil para el análisis por no presentar ningún tipo de disturbio o anomalía y por lo tanto es información desechable que ocupa espacio en el disco duro o banco de datos.

Visualización de los disturbios: El equipo cuenta con una pantalla táctil LCD la cual, junto con una interfaz gráfica de usuario (GUI), es capaz de mostrar gráficamente los disturbios de entrada y el resultado de las técnicas propuestas para el análisis de la señal eléctrica para la calidad de la energía.

Equipo de medición: Una vez que se ha integrado un medio para la visualización y análisis en línea de los disturbios de la calidad de la energía, se podría considerar el Datalogger PQ-UAQ junto a la computadora de placa única y la pantalla táctil, todo como un solo equipo de medición

1.4 Objetivos

1.4.1 Objetivo General

Desarrollar un equipo complementario al Datalogger PQ-UAQ para el análisis en línea y representación gráfica de los disturbios seleccionados utilizando un computador de placa única de bajo costo.

1.4.2 Objetivos Particulares

1. Seleccionar una computadora de placa única a partir de sus características, para contar con el dispositivo más apropiado acorde a la aplicación.
2. Realizar una búsqueda de técnicas para la detección y tratamiento digital de señales eléctricas, en bibliografías y artículos para tener un antecedente de las técnicas que existen para analizar los disturbios de una señal eléctrica.
3. Realizar la programación de las técnicas para la detección y tratamiento digital de señales seleccionadas que son; La transformada de Fourier, la detección de la envolvente y la supresión de la frecuencia fundamental, utilizando C++ como lenguaje de programación para implementar los algoritmos.
4. Realizar la programación de los indicadores y los disturbios de señales eléctricas para determinar la calidad de la energía: para los indicadores; Distorsión total Armónica (THD), valor eficaz (rms) y factor de potencia, para los disturbios; Swell, Sag, Variaciones de tensión, Armónicos e Interarmónicos
5. Incorporar el equipo PQ-UAQ en el sistema, como elemento primario para la adquisición de las señales eléctricas para posteriormente las señales adquiridas enviarlas a la computadora de placa única para su análisis
6. Diseñar una carcasa para todo el sistema, teniendo en cuenta todos los dispositivos, que son; la batería, el Datalogger PQ-UAQ, la computadora Lattepanda, un ventilador, orificios para los cuatro canales de corriente y los cuatro de voltaje, orificios para la alimentación para cargar la batería,

interruptores, Leds, SD’s, USB’s y HDMI, para tener un equipo compacto y portátil.

7. Realizar pruebas sobre el equipo utilizando señales sintéticas para validar los algoritmos, seguido de realizar pruebas utilizando el Datalogger PQ-UAQ para validar los algoritmos y el funcionamiento del programa en general.
8. Diseñar una interfaz gráfica que integre cada uno de los algoritmos programados y que además permita la interacción del usuario a través de una pantalla táctil para que este pueda visualizar diferentes disturbios eléctricos y tener acceso a funciones adicionales.

1.5 Planteamiento General

El planteamiento general de este proyecto es desarrollar un sistema para monitorear la calidad de la energía de una señal eléctrica, utilizando una tarjeta de adquisición de datos de desarrollo de la Universidad Autónoma de Querétaro, una computadora de placa única para hacer el procesamiento de la señal de entrada y aplicarle las distintas técnicas para determinar la calidad de la energía medida, y por último mostrar en una interfaz gráfica diferentes pantallas según lo que desee ver el usuario como la señal de entrada y la señal de salida una vez aplicándose alguna de las técnicas para análisis de la señal.

En la Figura 1 se muestra un diagrama a bloques general de los elementos que compondrán el sistema de monitoreo de calidad de la energía de bajo costo teniendo a la entrada señales eléctricas que pueden ser de datos adquiridos de una planta de generación fotovoltaica, de una fracción del consumo eléctrico de una universidad o bien del consumo de una casa habitación, esas señales entran a una tarjeta de adquisición de datos que desarrollo la Universidad Autónoma de Querétaro en proyectos anteriores llamada PQ-UAQ la cual ya nos entrega una señal que se puede enviar directamente al computador de placa única para realizar el procesamiento y análisis de la señal para después mostrar dichas señales en una interfaz gráfica a través de una pantalla táctil.

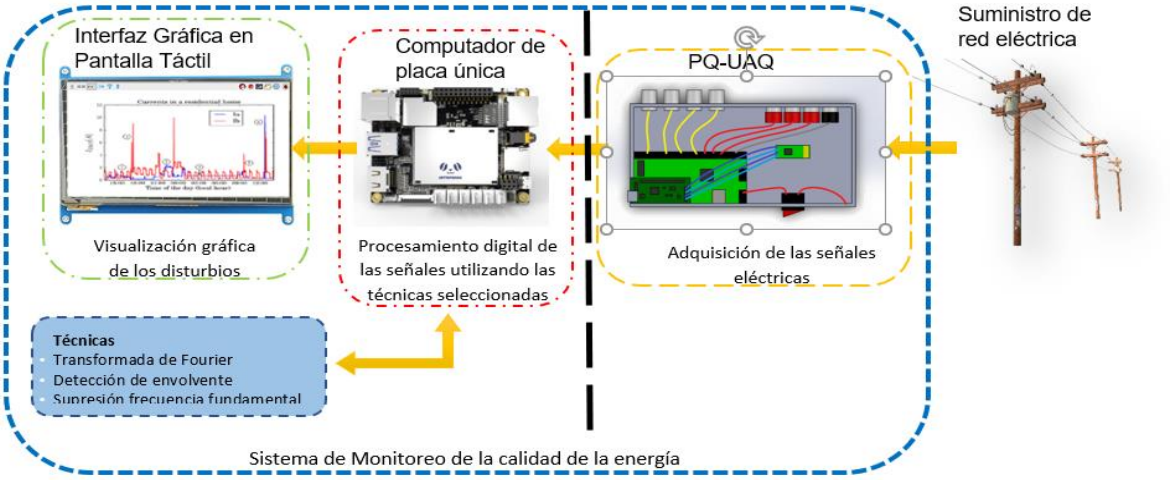


Figura 1 Diagrama a bloques del sistema de monitorización de la calidad de la energía de bajo costo

Capítulo 2

2. Fundamentación Teórica

La calidad de la energía es un concepto que inicio en la década de los 80's y que ha ganado cierta popularidad he importancia en los últimos años, y ese crecimiento se debe a cuatro principales razones de acuerdo con (Enríquez, 1999) las cuales son:

1. Las cargas cada día son más sensibles a las variaciones del suministro eléctrico debido a que cada vez son más los equipos conectados, tanto del área industrial como residencial, que hacen uso de microprocesadores y de la electrónica de potencia, como son los robots industriales, las computadoras, los electrodomésticos, entre muchos otros, que siendo cargas no lineales afectan no solo a la red eléctrica local conectada, sino que también afecta a la compañía de suministro eléctrico.
2. El incremento del concepto de mayor eficiencia en los sistemas eléctricos, provoca la aplicación de dispositivos de alta eficiencia, que, como los controladores de velocidad en motores eléctricos, hacen uso de bancos de capacitores para mejorar el factor de potencia y reducir perdidas, esto a su vez trae como consecuencias un incremento en los niveles de armónicas en los sistemas eléctricos, problema que afecta a la calidad de la energía.
3. Una mayor atención por parte de los usuarios a problemas con la calidad del suministro eléctrico, que pueden afectar a sus equipos, debido a: interrupciones de servicio, los transitorios, depresiones y elevaciones de voltaje, etcétera.
4. La creciente tendencia por la interconexión de sistemas eléctricos al nivel de sistemas de potencia e industriales, que trae como consecuencia una mayor cantidad de procesos integrados, lo que quiere decir que las fallas de un solo componente tienen consecuencias más graves.

Por estas razones es que es tan importante monitorear la calidad de la energía eléctrica, conocer las técnicas para su detección y análisis, los principales disturbios asociados a la calidad de la energía y los indicadores para saber si el suministro eléctrico es aceptable o no.

2.1 Computador de Placa Única

El computador de placa única también conocido como SBC (Single Board Computer por sus siglas en inglés) en esencia son computadoras con todos o la gran mayoría de sus componentes montados en una solo placa o PCB (Printed Circuit Board por sus siglas en inglés) pero que cuentan con las tres siguientes características principales: (García, 2014).

1. Reducido tamaño, las SBC son computadoras de muy reducidas dimensiones menores a las de una mini computadora montada sobre una placa que tienen medidas que oscilan por los 17x17 cm, usualmente las SBC tienen medidas menores a los 10x10 cm por ejemplo un modelo Lattepanda 2G/32G tiene medidas de 7x8.1 cm.
2. Bajo costo, otra característica es que suelen ser muy económicas, más que una Pc equivalente y su precio es muy variado, pero suele rondar por los 100 dólares lo cual hace que muchas personas se interesen en estos equipos para llevar a cabo sus proyectos.
3. Potencia aceptable, si bien es cierto que en su mayoría no poseen la potencia y las características de las computadoras convencionales o incluso las mini Pc's, la potencia que tienen suele ser suficiente para llevar a cabo una gran variedad de aplicaciones incluso de multimedia, pero esto depende mucho de cada SBC.

2.2 Calidad de la Energía

La definición exacta de calidad de la energía es algo aún indeterminado, ya que se aborda más a partir de los problemas que interesan para la calidad de la energía. Es decir que una definición de calidad de la energía puede definirse como;

una ausencia de interrupciones, sobretensiones, deformaciones producidas por armónicas en la red y variaciones de voltaje *rms* suministrado al usuario.

El objetivo de la calidad de la energía es encontrar la forma de corregir, disminuir o evitar los disturbios eléctricos y las variaciones de voltaje para el usuario y proponer soluciones e información para los fallos que se presentan por parte de la compañía que suministra la red eléctrica, para lograr que ambas partes cuenten con un suministro eléctrico de calidad.

(Enríquez, 1999), enlista tres tendencias que han hecho que los problemas de la calidad de la energía se agraven y por tanto que provocan un aumento en la importancia de conocer la calidad de la energía eléctrica.

a) Una mayor utilización de equipos para procesamiento de datos y comunicaciones.
b) Los equipos eléctricos modernos se han vuelto más sensibles al voltaje, donde sus componentes están implementados en sus límites sin tener valores de sobra.
c) El número de disturbios eléctricos se ha incrementado, pues la demanda de las cargas conectadas ha crecido más rápido que la generación de la energía.

2.2.1 Normas Referentes a la Calidad de la Energía

Actualmente aún se trabajan en normatividades para la calidad de la energía, pero ya existen algunas instituciones internacionales que se han preocupado por este tema como son la IEEE y The International Electrotechnical Commission (IEC), siendo estas dos instituciones las principales colaboradoras a la normalización de la calidad de la energía.

Algunas de las normas más importantes referentes a la calidad de la energía, desarrolladas por el IEEE y IEC se muestran en la Tabla 1.

Tabla 1 Descripción de las normas relacionadas con la Calidad de la Energía. (Ramírez, 2012)

Perturbación	Categoría de Normalización	Estándares IEEE	Estándares IEC
Tensiones y corrientes armónicas (resonancia TIF y muescas)	Ambiente/compatibilidad	Ninguna	IEC 61000-2-1/2
	Emisión/Limites de inmunidad	IEEE 519	IEC 61000-3-2/4 (555)
	Pruebas y medidas	Ninguna	IEC 61000-4-7/13
	Instalación/Mitigación	IEEE 519a	IEC 61000-5-5
	Componente térmica	IEEE/ANSI C57.110	Ninguna
Nivel de Tensión (Regulación, desbalance, fluctuaciones y parpadeo)	Ambiente/compatibilidad	IEEE 141, 241, 1100	IEC 38/BTTF 68-6
	Emisión/Limites de inmunidad	ANSI C84.1	IEC 61000-3-3/5 (555)
	Pruebas y medidas	Ninguna	IEC 61000-4-1/14/15
	Instalación/Mitigación	IEEE 141, 241, 1100	IEC 61000-5-X
	parpadeo de luz	IEEE 141.519	IEC 868(61000-4-15)
Huecos de Tensión	Ambiente/compatibilidad	IEEE 1250	IEC 61000-2-4
	Emisión/Limites de inmunidad	IEEE P 1346	IEC 61000-3-3/5 (555)
	Pruebas y medidas	Ninguna	IEC 61000-4-1/11
	Instalación/Mitigación	IEEE 446, 1100, 1159	IEC 61000-5-X
	Apertura de fusible	IEEE 242 (Protección)	IEC 364
Transitorios y Sobre tensiones	Ambiente/compatibilidad	IEEE/ANSI C62.41	IEC 61000-2-5
	Emisión/Limites de inmunidad	Ninguna	IEC 61000-3-X
	Pruebas y medidas	IEEE/ANSI C62.45	IEC 61000-4-1/2/4/5/12
	Instalación/Mitigación	C62 Series, 1100	IEC 61000-5-X
	Ruptura de aislamiento	Ninguna	IEC 664

2.3 Técnicas para la Detección y Análisis de Señales Eléctricas

Existen un gran número de técnicas para la detección y análisis de señales eléctricas algunas técnicas clásicas que son muy populares por su sencillez de implementación y los pocos recursos computacionales, pero perdiendo precisión y son ineficaces para la detección de ciertos disturbios, por ejemplo la transformada de Fourier es una técnica muy utilizada para las señales eléctricas que sean estáticas, es decir que no son variantes en el tiempo, por lo que disturbios transitorios de corta duración son imperceptibles por este tipo de análisis, para ello existen muchas otras técnicas que aun que más complejas de implementar y que requieren muchos más recursos computacionales, también son más apropiadas para el procesamiento de señales como es el algoritmo MUSIC o algunos otros algoritmos genéticos.

Para este trabajo de investigación se utilizarán las técnicas clásicas principalmente por los pocos recursos computacionales que se requieren para implementar estas técnicas o algoritmos y así tener un análisis y monitoreo en línea de las señales de entrada.

2.3.1 Transformada de Fourier

La transformada de Fourier es una herramienta matemática que resulta útil en el diseño de sistemas lineales invariantes en el tiempo o LTI (linear time-invariant por sus siglas en inglés) , según el tipo de señales se le puede conocer de una u otra forma pero en esencia son lo mismo, una descomposición de la señal en componentes sinusoidales (o exponenciales complejas), representadas en el dominio de la frecuencia, si se trata de una señal periódica la descomposición se conoce como serie de Fourier, y para el caso de señales de energía finita, la descomposición se conoce como transformada de Fourier, mostrada en la ecuación 1 , (Proakis et al, 2007).

$$F(\omega) = \int_{-\infty}^{\infty} f(x) e^{-i\omega x} dx \qquad (1)$$

2.3.1.1 Transformada Rápida de Fourier

La transformada rápida de Fourier o FFT (Fast Fourier Transform por sus siglas en inglés) son en realidad algoritmos utilizados para un cálculo más optimizado de la transformada discreta de Fourier o DFT (Discret Fourier Transform por sus siglas en inglés), este tipo de algoritmos disminuyen en mayor o menor medida el tiempo de cálculo computacional requerido, pero con los mismos resultados, es decir que son algoritmos más rápidos y eficientes para calcular la DFT de una señal. Algunos de estos algoritmos son:

- Divide y Vencerás para calcular la DFT
- FFT de base 2 (Radix-2)
- FFT de base 4 (Radix-4)
- Algoritmos FFT de base dividida

FFT de base 2 (Radix-2)

El Algoritmo de FFT de base 2, son en realidad dos algoritmos que son considerados los más simples, pero también más ampliamente utilizados para el cálculo de la DFT, esto gracias a que se mejora considerablemente la velocidad de la DFT, esto gracias a que se disminuyen el número de multiplicaciones complejas y por tanto mejora la velocidad del algoritmo, (Proakis et al, 2007).

Esto se ilustra en la Tabla 2.

Tabla 2 Comparación de la complejidad del cálculo directo de la DFT contra el algoritmo FFT base 2

Número de puntos, N	Multiplicaciones complejas en el cálculo directo, N^2	Multiplicaciones complejas en el algoritmo FFT, $(N/2)\log_2 N$	Factor de mejora de la velocidad
4	16	4	4.0
8	64	12	5.3
16	256	32	8.0
32	1,024	80	12.8
64	4,096	192	21.3
128	16,384	448	36.6
256	65,536	1,024	64.0
512	262,144	2,304	113.8
1,024	1,048,576	5,120	204.8

FFT base 2 de Diezmado temporal

El cálculo básico consiste en tomar dos números complejos, una pareja (a, b), multiplicar b por W_N^r, y luego sumar y restar el producto de a para formar los dos nuevos números complejos (A, B), este cálculo básico, se ilustra en la Figura 2 y se le conoce como diagrama de flujo de mariposa, (Proakis et al, 2007).

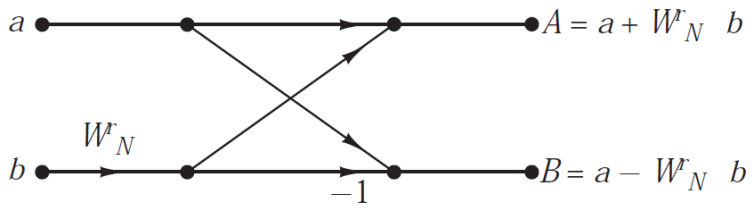


Figura 2 Mariposa básica del algoritmo FFT de diezmado temporal

Un ejemplo del cálculo del algoritmo de diezmado temporal de 8 puntos (N=8) se muestra en la Figura 3.

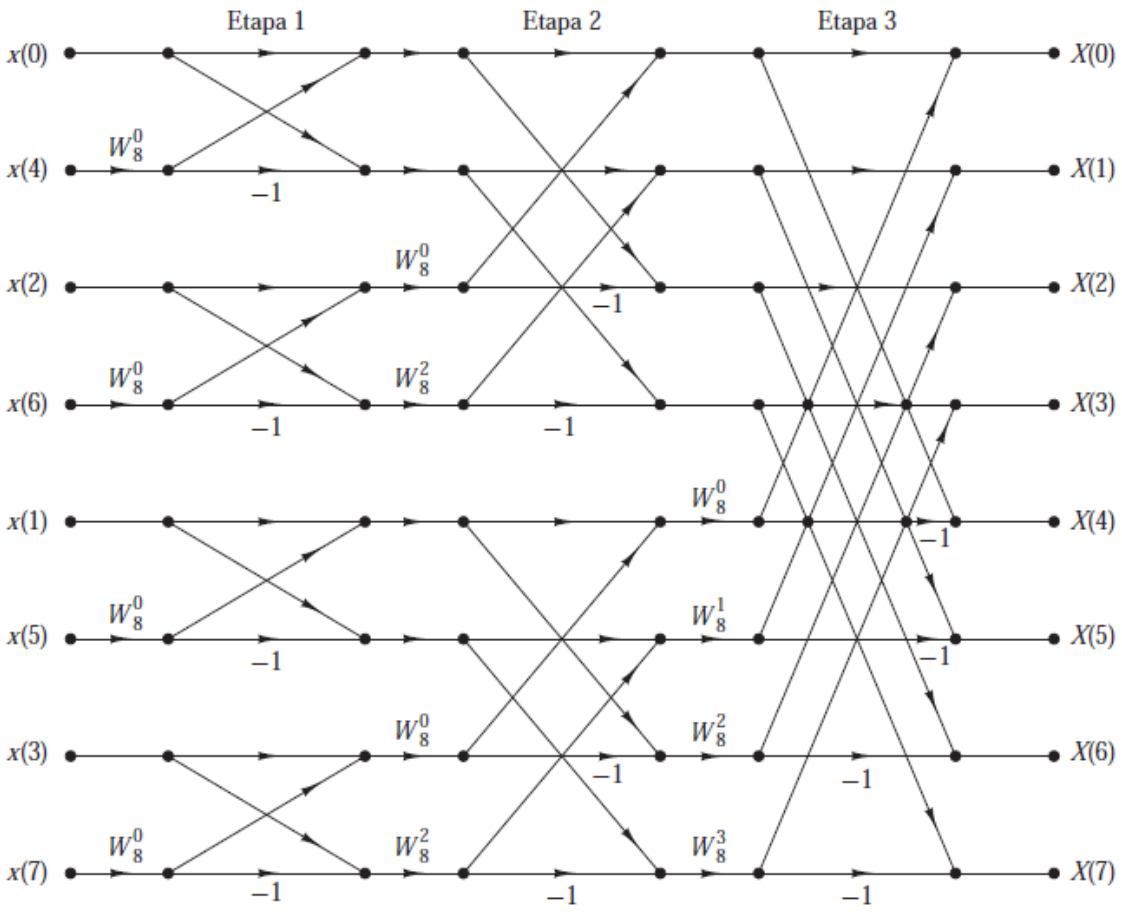


Figura 3 Algoritmo FFT de diezmado temporal de 8 puntos

FFT base 2 de diezmado en frecuencia

El proceso requiere de v=$\log_2$N etapas de diezmado, donde cada etapa requiere de N/2 operaciones mariposa básicas como se muestra en la Figura 4, por lo que se requiere de (N/2) $\log_2$N multiplicaciones complejas y N$\log_2$N sumas complejas, un ejemplo del algoritmo FFT de diezmado en frecuencia de ocho puntos se muestra en la Figura 5, (Proakis et al, 2007).

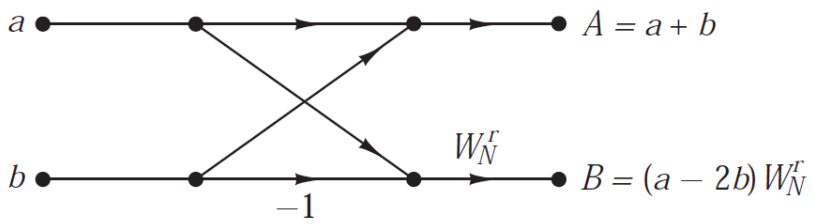


Figura 4 Mariposa básica del algoritmo FFT de diezmado en frecuencia

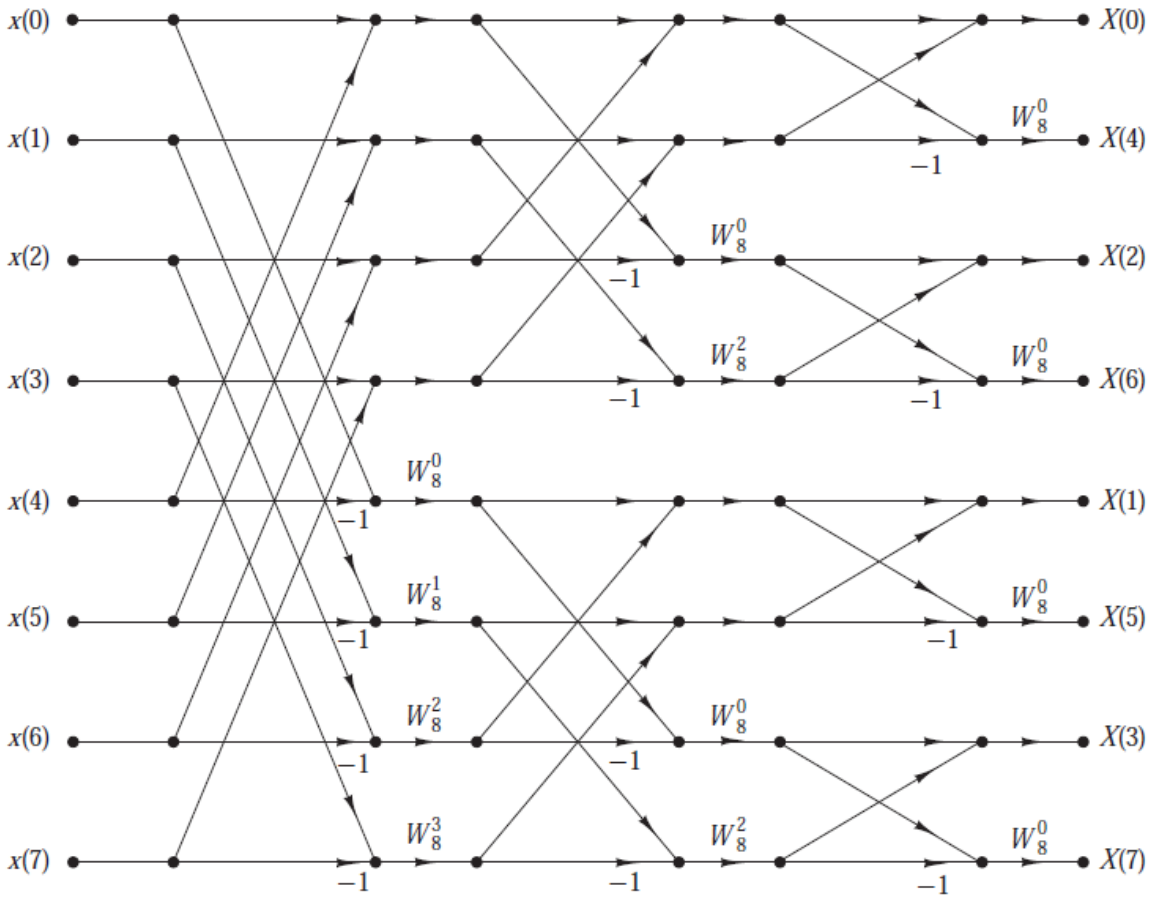


Figura 5 Algoritmo FFT de diezmado en frecuencia de 8 puntos

2.3.2 Detección de la Envolvente

La envolvente de una señal se puede entender de diferentes formas, pero en general, se refiere a una curva generada a partir de los límites de una señal cualquiera, se puede generar una envolvente superior y una envolvente inferior de la señal, aunque la más utilizada es la envolvente superior. La Figura 6 muestra una representación de la envolvente superior e inferior de una señal.

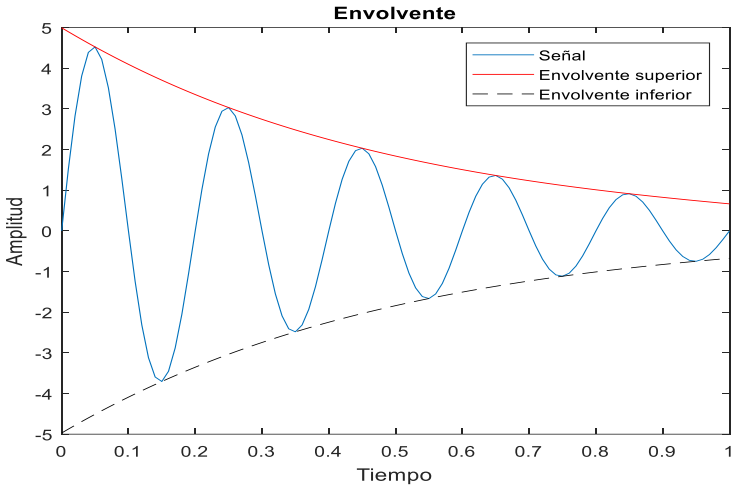


Figura 6 Envolventes de una señal

2.3.2.1 Detección de Máximos Locales

Es un método que sirve para la detectar los puntos máximos de la señal, pero dentro de un rango o únicamente con sus vecinos cercanos, los máximos locales también llamados máximos relativos, la Figura 7 muestra gráficamente los máximos locales de una señal senoidal con ruido.

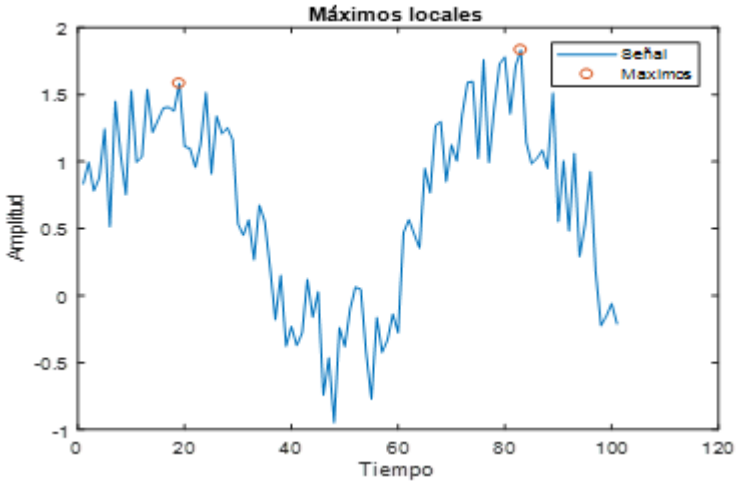


Figura 7 Máximos locales

Se define como: si existe un entorno B de P_0 tal que, para cada otro punto $P = (x_1, x_n) \in B$, (Chapra et al, 2007), con la ecuación 2.

$$f(x_1^0, \dots, x_n^0) \geq f(x_1, \dots, x_n) \qquad (2)$$

2.3.2.2 Interpolación

Son técnicas matemáticas que tienen como objetivo encontrar un valor aproximado de una magnitud dentro de un rango de valores donde se conocen los extremos de dicho rango. El método más común es la interpolación polinomial, por ser simplemente un polinomio de n-ésimo grado de tal forma que cumple con la fórmula 3:

$$f(x) = a_0 + a_1 x + a_1 x^2 + \cdots + a_n x^n \qquad (3)$$

Sin embargo, no es el único utilizado existen distintas formas de interpolación debido a que la interpolación por polinomios suele generar muchos errores principalmente cuando se requiere de un polinomio de grado superior.

- Interpolación lineal
- Interpolación polinomial
- Interpolación Spline

Spline

Son funciones matemáticas ampliamente utilizadas como técnicas de interpolación, también conocidas como interpolación segmentaria o trazadores y consiste en dividir un intervalo en subintervalos polinómicos de cierto orden, los trazadores son una técnica desarrollada para utilizar polinomios de menor orden y que dan mejor aproximación que los polinomios de grado superior, dentro de los Spline se pueden ver de primer grado o trazadores lineales, de segundo grado o trazadores cuadráticos y los más utilizados en la práctica además de ser los que mejor resultados da son los de tercer grado o trazadores o Spline cúbicos, (Chapra et al, 2007). Como se muestra en la ecuación 4.

$$f_i(x) = a_i + b_i x + c_i x^2 + d_i x^3 \qquad i = 0,1, \dots, n-1 \qquad (4)$$

Para evaluar los Spline cúbicos se requiere que cumplan con las siguientes condiciones:

1. Los valores de la función deben ser iguales en los nodos interiores (2n-2 condiciones).
2. La primera y última función deben pasar a través de los puntos extremos (2 condiciones).
3. Las primeras derivadas en los nodos interiores deben ser iguales (n-1 condiciones).
4. Las segundas derivadas en los nodos interiores deben ser iguales (n-1 condiciones).
5. Las segundas derivadas en los nodos extremos son cero (2 condiciones).

2.3.3 Supresión de la Frecuencia Fundamental

Es una técnica que consiste en cancelar o eliminar la frecuencia fundamental de una señal para de esa forma analizar únicamente los armónicos e Interarmónicos de la señal, también pueden eliminarse los armónicos para amplificar el efecto que generan los Interarmónicos.

2.3.3.1 Mínimos Cuadrados

Es un método para la aproximación de un conjunto de valores o puntos los cuales que requiere de una función. Puede ser una regresión lineal, una regresión polinomial, regresión múltiple o regresión no lineal, (Chapra et al, 2007).

Regresión lineal:

Es la aproximación más simple por mínimos cuadrados y se trata de una línea recta que se ajusta a los diferentes valores, la expresión matemática para la línea recta se muestra en la ecuación 5, (Chapra et al, 2007).

$$y = a_0 + a_1 x + e \qquad (5)$$

Donde a_0 y a_1 son coeficientes que representan la intersección con el eje '*y*' y la pendiente, respectivamente, *e* es el error, entre el modelo y las observaciones.

Regresión polinomial:

A pesar de que la regresión lineal es el método más utilizado no es el más conveniente para todos los casos, cuando se busca hacer uso de curvas para adaptarse mejor al conjunto de datos se utiliza una regresión polinomial, (Chapra et al, 2007). La expresión matemática para un polinomio de m-ésimo grado es la ecuación 6;

$$y = a_0 + a_1 x + a_2 x^2 + \cdots + a_m x^m + e \qquad (6)$$

Regresión múltiple

También llamada regresión lineal múltiple es cuando *y* es una función lineal de dos o más variables independientes (Chapra et al, 2007). La expresión matemática para m dimensiones se expresa en la ecuación 7;

$$y = a_0 + a_1 x_1 + a_2 x_2 + \cdots + a_m x_m + e \qquad (7)$$

Regresión no lineal

Como en el caso de los mínimos cuadrados lineales, la regresión no lineal se basa en la determinación de los valores de los parámetros que minimizan la suma de los cuadrados de los residuos, sin embargo, en el caso no lineal, la solución debe realizarse en una forma iterativa, por ejemplo, por el método de Gauss-Newton (Chapra et al, 2007). La expresión general de una ecuación no lineal se muestra en la ecuación 8;

$$y_i = f(x_i; a_0, a_1, \dots, a_m) + e_i \qquad (8)$$

2.4 Indicadores de la Calidad de la Energía

Los indicadores de la calidad de la energía eléctrica son aquellos parámetros que definen, que tan aceptable es la señal eléctrica en base a valores aprobados

por las distintas normas. Se consideran principalmente tres indicadores para definir la calidad de la energía que son:

- Distorsión Armónica Total (THD)
- Valor Eficaz (RMS)
- Factor de Potencia

Estos indicadores se observan directamente sobre el suministro de la red eléctrica debido a que son algunos de los factores que cuida la compañía distribuidora.

2.4.1 Distorsión Armónica Total

También conocido como THD (Total Harmonic Distortion), es un factor que expresa la cantidad de distorsión armónica contenida en la onda distorsionada (Enríquez, 1999). Es decir, la onda fundamental más sus componentes armónicas, y se encuentra tanto para corriente como para voltaje, usualmente se expresa en porcentaje, y se obtiene a partir de las ecuaciones 9 o 10 dependiendo si es para voltaje o corriente respectivamente.

$$THDv = \frac{\sqrt{v_2^2 + v_3^2 + \cdots + v_n^2}}{v_1(fundamental)} x100 \quad (9)$$

$$THDi = \frac{\sqrt{i_2^2 + i_3^2 + \cdots + i_n^2}}{i_1(fundamental)} x100 \quad (10)$$

2.4.2 Valor Eficaz

Valor eficaz o RMS (Root Mean Square), es el valor que tendría una corriente continua que produjera la misma potencia que en corriente alterna, al aplicarse sobre una misma resistencia (Enríquez, 1999). Se calcula para tensión y corriente como se observan en las ecuaciones 11 y 12, para voltaje y corriente respectivamente.

$$Vrms = \frac{v_{max}}{\sqrt{2}} \quad (11)$$

$$Irms = \frac{I_{max}}{\sqrt{2}} \quad (12)$$

2.4.3 Factor de Potencia

El factor de potencia o FP es un indicador que usualmente sirve para describir la cantidad de energía eléctrica que se ha convertido en trabajo (Enríquez, 1999). Se define como el cociente de la relación de la potencia activa entre la potencia aparente como se muestra en la ecuación 13;

$$FP = \frac{P\ (potencia\ activa)}{S\ (potencia\ aparente)} = Cos\ \varphi \quad (13)$$

Se entiende que un FP=1 es una conversión total de la energía eléctrica en trabajo, y es el valor deseado o ideal para el factor de potencia en todas las instalaciones eléctricas.

Para entender la fórmula del factor de potencia se tiene que entender que dentro de la potencia en corriente alterna surgen tres parámetros que generan un efecto distinto según sea el caso, inductancia, capacitancia y resistencia, esto a su vez origina tres tipos de potencia; potencia activa, potencia reactiva y potencia aparente, la Figura 8 muestra lo que se llama el triángulo de potencias y se puede observar gráficamente que entonces el factor de potencia es igual al coseno del ángulo φ.

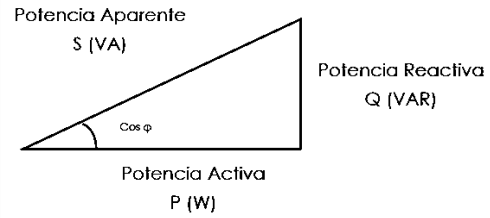


Figura 8 Triángulo de potencias

2.4.3.1 Potencia Aparente

Es la potencia que resulta de considerar la tensión que se consume aplicada a la corriente que se demanda (Enríquez, 1999). Se expresa como la suma de los vectores de potencia activa y la potencia reactiva, se expresa en Volts-amperes (VA) para distinguir del resto de las potencias y su fórmula se expresa en la ecuación 14.

$$S = V * I = \sqrt{P^2 + Q^2} \qquad (14)$$

2.4.3.2 Potencia Reactiva

La potencia reactiva esta 90° desfasada de la potencia activa, es la potencia que requieren las bobinas y capacitores para generar campos magnéticos o eléctricos y no produce ningún trabajo (Enríquez, 1999). Su unidad son Volts-Amperes Reactivos (Var), su fórmula se muestra en la ecuación 15.

$$Q = V * I * Sin\ \varphi \qquad (15)$$

2.4.3.3 Potencia Activa

Es la potencia que se aprovecha como energía útil en las distintas cargas (Enríquez, 1999). Es también conocida como potencia real y es la potencia consumida únicamente por la resistencia, se expresa en watts, a partir de la ecuación 16.

$$P = V * I * Cos\ \varphi \qquad (16)$$

2.5 Disturbios en Señales Eléctricas

Existen distintos tipos de disturbios que pueden afectar a las señales eléctricas, por esto la norma IEEE 1159 clasifica los fenómenos electromagnéticos más importantes y que tienen que ver con la calidad de la energía, problemas como: impulsos, oscilaciones, Sags, Swells, interrupciones, bajo voltaje, sobre voltaje, offset o compensación de DC, armónicos, Inter armónicos, muescas, ruido,

fluctuaciones de voltaje o flicker y variaciones de frecuencia. Como se muestra en la Tabla 3.

Tabla 3 Clasificación de disturbios que afectan a la calidad de la energía. (Batista et al, 2003)

Categoría			Duración Típica	Amplitud Típica
Transitorios		Impulsos	ns a ms	-
		Oscilaciones	3 µs a 5 ms	0 a 8 pu
Corta Duración	Instantáneos	Sag	0.5 a 30 ciclos	0.1 a 0.9
		Swell	0.5 a 30 ciclos	1.1 to 1.8
	Momentáneos	Interrupción	0.5 ciclos a 3 s	< 0.1 pu
		Sag	30 ciclos a 3 s	0.1 a 0.9
		Swell	30 ciclos a 3 s	1.1 a 1.4
	Temporales	Interrupción	3 s a 1 min.	< 0.1 pu
		Sag	3 s a 1 min.	0.1 a 0.9
		Swell	3 s a 1 min.	1.1 a 1.4
Larga Duración		Interrupción	> 1 min.	-
		Bajo-Voltaje	> 1 min.	0.8 a 0.9
		Sobre voltaje	> 1 min.	1.1 a 1.2
Voltaje Desbalanceado			Estado estable	0.5 a 2 %
Distorsión Forma de Onda		DC Offset	Estado estable	0 a 0.1 %
		Armónicos	Estado estable	0 a 20 %
		Interarmónicos	Estado estable	0 a 2 %
		Notching	Estado estable	-
		Noise	Estado estable	0 a 1 %
Fluctuaciones de Voltage (flicker)			Estado estable	0.1 a 7 %
Variaciones de Frecuencia			< 10 s	-

2.5.1 Swell

El incremento de la tensión nominal mejor conocido como Swell, es cuando la tensión nominal se eleva sobre 1.1 p.u. y con una duración de entre 0.5 ciclos (8.33 segundos) y hasta 30 ciclos (1 minuto), son variaciones de tensión no tan comunes como los sag's, es un tipo de disturbio provocado principalmente por fallas

en el sistema eléctrico y la conexión de bancos de capacitores, su principal consecuencia es la reducción de vida útil en los dispositivos conectados y desgaste en el aislamiento de máquinas como motores y transformadores.

La Figura 9 muestra una representación gráfica del Swell.

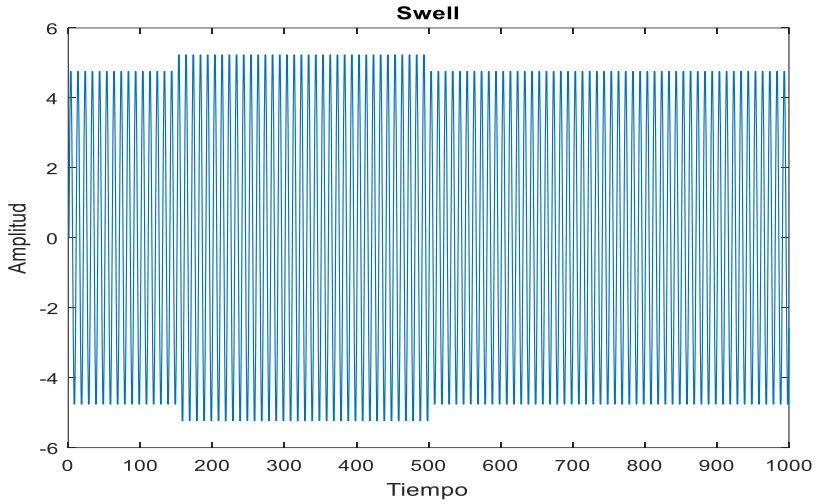


Figura 9 Elevación de tensión Swell

2.5.2 Sag

El decremento de la tensión o Sag, es cuando la tensión nominal disminuye entre 0.1 p.u. y 0.9 p.u. por un tiempo que va desde los 0.5 ciclos (8.33 s) y hasta 30 ciclos (1 minuto), son disturbios generalmente producidos por fallas en el sistema eléctrico, la conexión de cargas grandes y el arranque de motores, tiene como consecuencia el envejecimiento de los sistemas electrónicos y digitales, en algunas ocasiones se puede detectar visualmente con la disminución de intensidad en lámparas.

La Figura 10 muestra la representación gráfica de un Sag.

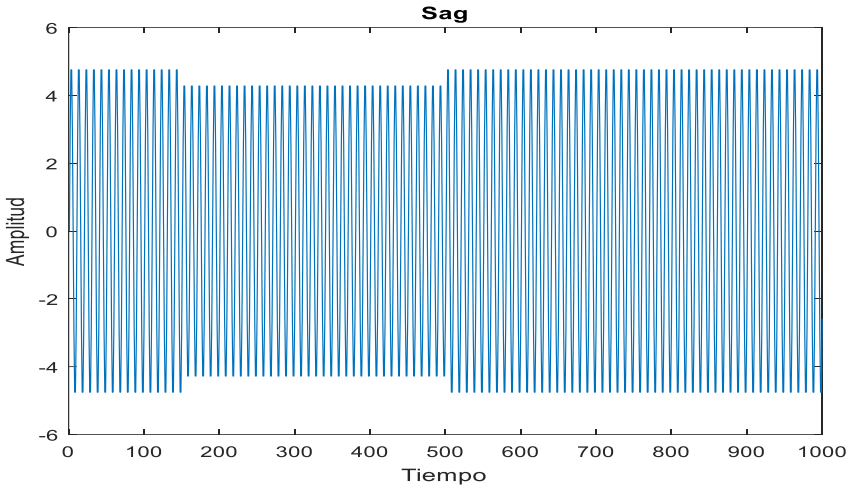


Figura 10 Decremento de la tensión Sag

2.5.3 Interrupciones

Se considera que existe una interrupción cuando el valor nominal de tensión cae por debajo de 0.1 p.u. y pueden clasificarse según el estándar IEEE-1250 como:

- Interrupción instantánea; entre 0.5 y 30 ciclos
- Interrupción momentánea; entre 30 ciclos y 2 segundos
- Interrupción temporal; entre 2 segundos y 2 minutos
- Interrupción sostenida; más de 2 minutos de duración.

Este tipo de fallas se generan principalmente por fallas en el sistema eléctrico, fallas en el transformador, tormentas eléctricas, etc. Como consecuencia por la pérdida de tensión los equipos pueden llegar a desconectarse y eso trae graves consecuencias a los equipos cuando no cuentan con las protecciones adecuadas.

2.5.4 Transitorios

Son incrementos del valor nominal de tensión o corriente, con la característica de ser un incremento mucho mayor al esperado en un Swell, además de ser de menor duración, por debajo de los milisegundos, pueden ser micro o nano segundo de duración para un transitorio. Los transitorios pueden tener cierta

respuesta amortiguada o no amortiguada lo que genera que transitorios tipo impulso y transitorios oscilatorios, (Enríquez, 1999).

Transitorio tipo impulso

Es una variación brusca y repentina de la potencia a una frecuencia distinta de la fundamental, es unidireccional normalmente positivo, principalmente generados por descargas atmosféricas. Al ser de altas frecuencias se ven rápidamente amortiguados por la parte resistiva del sistema y es cuando excitan la resonancia del circuito cuando se generan los transitorios oscilatorios.

Transitorios oscilatorios

Es un caso de transitorios donde el valor instantáneo de voltaje cambia muy rápidamente de polaridad. Ocasionados principalmente por la conexión y desconexión de bancos de capacitores. Los transitorios oscilatorios pueden clasificarse según frecuencia como se muestra en la Tabla 4.

Tabla 4 Clasificación de los transitorios oscilatorios por frecuencia

Categoría	Magnitud	Duración
Baja Frecuencia	< 5 kHz	0.3-5 µSeg
Media Frecuencia	5-500 kHz	20 µSeg
Alta Frecuencia	0.5-5 kHz	5 µSeg

Dependiendo de la amplitud y duración del transitorio pude generar desde casi ningún problema a los equipos y a la instalación en general, o bien puede tener consecuencias devastadoras con pérdidas o fallas graves en los equipos conectados a la red donde ocurrió el transitorio. Figura 11 muestra una representación de los dos tipos de transitorios.

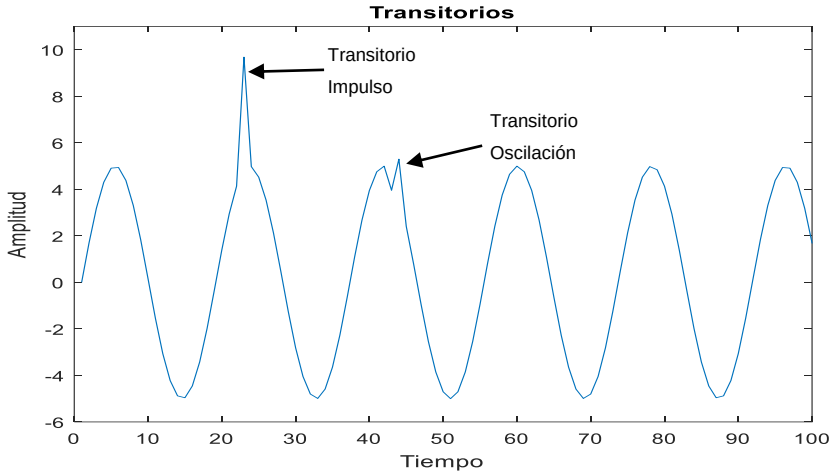


Figura 11 Transitorios tipo impulso y oscilación

2.5.5 Armónicos

Los armónicos son voltajes y/o corrientes con frecuencias que son múltiplos enteros de la frecuencia fundamental de la red eléctrica. Los armónicos distorsionan la forma de onda de la señal senoidal proporcionada durante algunos pocos a muchos ciclos.

Cada armónico se expresa en términos de su orden, por ejemplo, el segundo, tercero y cuarto armónico son 120, 180 y 240 Hz respectivamente de una frecuencia fundamental de 60 Hz.

El fenómeno de la distorsión armónica no es precisamente algo nuevo, por ello se han estudiado a detalle para disminuir sus efectos e identificar las fuentes que producen armónicos, para sus fuentes de generación pueden clasificarse en; fuentes de armónicas tradicionales, fuentes nuevas de armónicas y fuentes futuras, donde una de estas fuentes que destaca hoy en día son las fuentes no líneas que contribuyen cada vez más a la generación de armónicas, y estas fuentes son los dispositivos electrónicos de hoy en día, (Enríquez, 1999).

Se encuentran una gran variedad de consecuencias de la presencia de armónicas, algunas de estas consecuencias son:

- Perdidas adicionales en los bancos de capacitores,

- Mal funcionamiento de motores y transformadores,
- Interferencia telefónica,
- Causa frecuencias de resonancia, entre otros problemas.

La Figura 12 muestra una señal senoidal, que es la señal fundamental, una señal que es el tercer armónico de la fundamental y la señal de onda resultante de la suma de la fundamental con el tercer armónico.

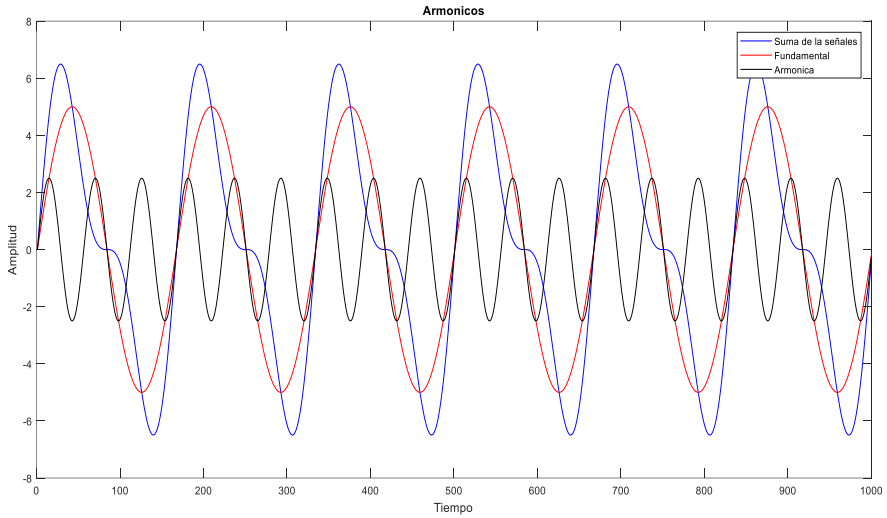


Figura 12 Tercer armónico

2.5.6 Interarmónicos

Son muy parecidos a los armónicos con la única diferencia de que sus frecuencias son múltiplos no enteros de la de la frecuencia fundamental.

2.6 Interfaz Gráfica de Usuario

La interfaz gráfica de usuario o GUI (Graphical User Interface por sus siglas en ingles), es un programa informático que actúa como interfaz de usuario, utilizando objetos gráficos o metáforas gráficas, permite una interacción didáctica con el usuario de manera visual, comunicando la pantalla o GUI con el software de la computadora, (Lamarca, 2018).

De tal manera que la interfaz gráfica cumple con las siguientes características básicas:

- Facilidad de comprensión, aprendizaje y uso
- Representación fija y permanente de un contexto de acción (fondo)
- El objeto de interés ha de ser de fácil identificación
- Diseño ergonómico mediante el establecimiento de barras, menús e iconos de fácil acceso.
- Las interacciones se basarán en acciones físicas sobre elementos de código visual o auditivo (iconos, botones, imágenes, texto, videos, etc.).
- Las operaciones serán rápidas con efectos inmediatos
- Prevención del error y ayuda de acuerdo al nivel del usuario.

2.7 Pantalla Táctil Capacitiva

Una pantalla táctil o touch screen, es una pantalla que mediante un toque directo sobre su superficie permite la entrada de datos y ordenes al dispositivo. Actúa como un periférico de entrada y salida de datos, existen distintas tecnologías para las pantallas táctiles como, capacitivas, resistivas y onda acústica de superficie.

La pantalla táctil capacitiva como su nombre lo indica está basada en sensores capacitivos, son cada vez más utilizadas este tipo de pantallas por ser las que cuentan con mejor resolución de video, mejor respuesta táctil y algunas permiten el multitáctil que es múltiples toques simultáneamente. Consiste en una capa de aislamiento eléctrico, como cristal, recubierto de un conductor transparente, ITO (tin-doped indium oxide), de tal forma que cuando el cuerpo humano como conductor eléctrico entra en contacto con la pantalla genera una distorsión del campo electrostático que se mide a través de la capacitancia y determinada su posición por un controlador.

La Figura 13 muestra una fotografía de la pantalla táctil compatible con la SBC Lattepanda, y como se puede observar posee dos conectores que van a la

tarjeta, uno es para la parte visual únicamente y el otro conector es para el sobre pantalla capacitivo que permite hacer táctil la pantalla.

Figura 13. Fotografía pantalla táctil compatible con Lattepanda

2.8 Señales Sintéticas

Las señales sintéticas son señales generadas a partir de una ecuación o modelos matemáticos, son señales programadas que tienen un comportamiento muy parecido o idéntico a un fenómeno real. Las señales sintéticas son utilizadas para una gran cantidad de aplicaciones o bien como medio de comprobación y validación de algoritmos antes de que se apliquen en señales reales o en campo.

2.9 Qt Creator

Es una empresa internacional Qt Group con presencia en más de 12 países, cumbre con necesidades de software a más de un millón de desarrolladores, a los cuales proporciona un software multiplataforma libre y gratuito Qt Creator, para la compilación de diversos lenguajes de programación, entre los cuales destaca C++, pero también cuenta con soporte para aplicaciones web, móviles y lenguajes de programación como Java, Ruby y C.

Capítulo 3

3. Metodología

La metodología simplificada a seguir en este proyecto se ve representada en la Figura 14 donde el elemento central del proyecto es la SBC Lattepanda debido a que será la computadora central, a la cual se le instalará el software que será necesario para llevar acabo el procesamiento de las señales utilizando los algoritmos propuestos, seguido de una fase de pruebas para corroborar los algoritmos programados, utilizando señales sintéticas, para después pasar a la incorporación del Datalogger PQ-UAQ, siendo este el que administre las señales de entrada para la SBC que requieren procesamiento, de tal forma que las señales ya procesadas se muestren en la pantalla táctil que se conecta a la SBC permitiendo la interacción con el usuario a partir de la interfaz gráfica.

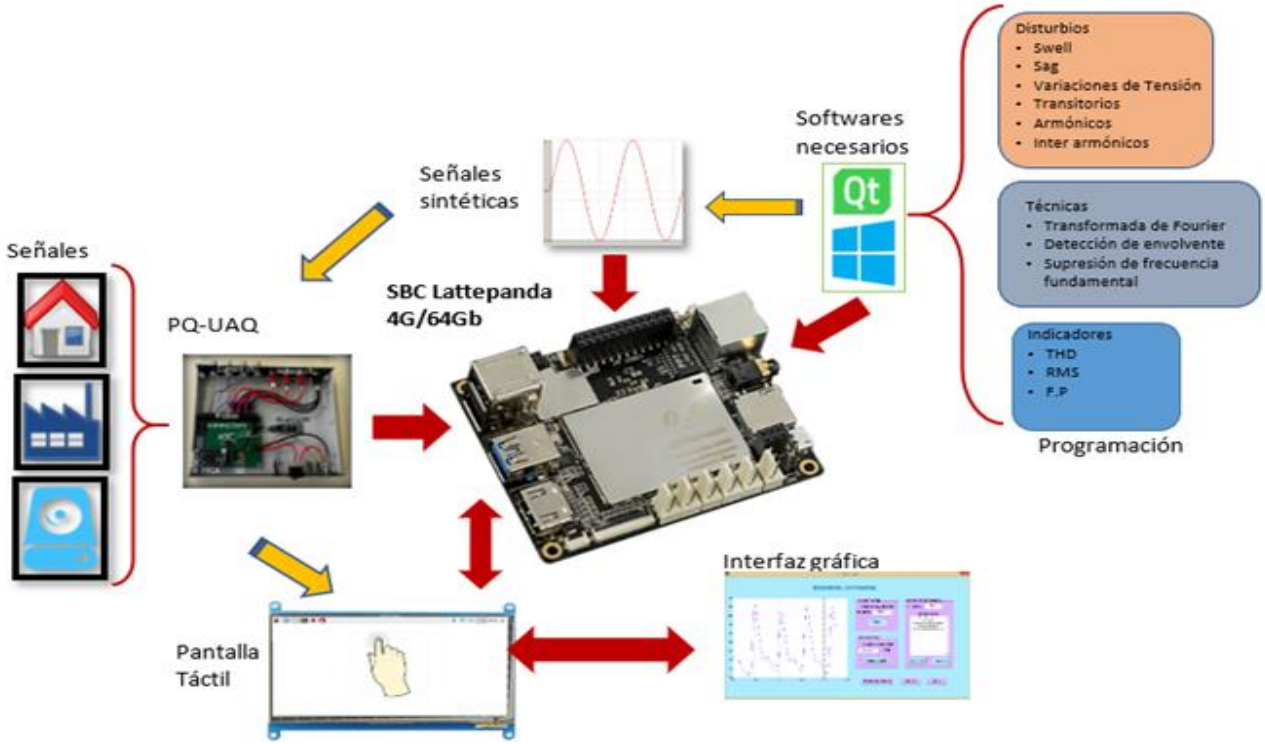


Figura 14 Metodología Simplificada

3.1 Diseño físico del equipo

Para realizar el diseño físico del proyecto, es decir la carcasa que contendrá todos los elementos necesarios para tener un único elemento más parecido a un instrumento de medición portable, es necesario tomar en cuenta los siguientes aspectos para el diseño final.

Dimensiones, el diseño de la carcasa no debe exceder de ciertas dimensiones considerando aspectos como el confort o comodidad al sostener el equipo con una o ambas manos, debido a que se pretende mantener cierta practicidad y portabilidad del equipo lo cual indica que debería ser de fácil sujeción aun con una sola mano, por otra parte se tiene cierta limitación en las dimensiones donde debe ser mayor que algunos de los componentes que se deben incorporar en su interior los componentes que principalmente contribuyen a esta limitación son, la pantalla táctil y la batería que son los dos dispositivos que más espacio requieren a considerar en el diseño.

Peso, el equipo a diseñar debe cumplir con una restricción de peso de tal forma que se pueda cargar con un o ambas manos durante un cierto periodo de tiempo sin que se requiera de mucho esfuerzo físico, para ello hay que evitar elementos que puedan agregar una gran cantidad de peso al sistema como es principalmente la batería.

Tamaño de los componentes, para el volumen total de la carcasa es necesario considerar el volumen individual de cada uno de los elementos que contendrá la carcasa, es decir que se debe conocer de ante mano cuáles serán los elementos y sus dimensiones.

Espacio para conexiones, es importante considerar que el cable y las conexiones para las terminales de corriente y voltaje necesitan de cierto espacio para que tengan el tamaño adecuado de cable no precisamente justo, pero tampoco muy sobrado, por otra parte, de que debe ser relativamente fácil meter la mano para llevar acabo las conexiones manuales de las terminales con el PQ-UAQ

Material y Fabricación, es necesario considerar el material que se pretende utilizar para llevar acabo la carcasa y el método de fabricación, debido a que se podrían utilizar materiales como acrílico o aluminio, pero estos requieren de un proceso de fresado manual o automático en una máquina CNC (Control Numérico Computarizado), también se podría utilizar materiales como el PLA (ácido poli láctico) que requieren de un proceso de impresión 3D. Es entonces debido a que existen distintas opciones para el material y proceso de fabricación es necesario determinar el más óptimo para el diseño, esto a partir de la complejidad de diseño, costos de fabricación, tiempo de fabricación y resistencia del material, para tener la carcasa más adecuada.

3.2 Metodología a detalle

Un desarrollo más detallado de lo que es el proyecto se observa en la Figura 15 donde es representa en forma de diagrama los pasos a seguir para llegar a la conclusión del proyecto de tesis.

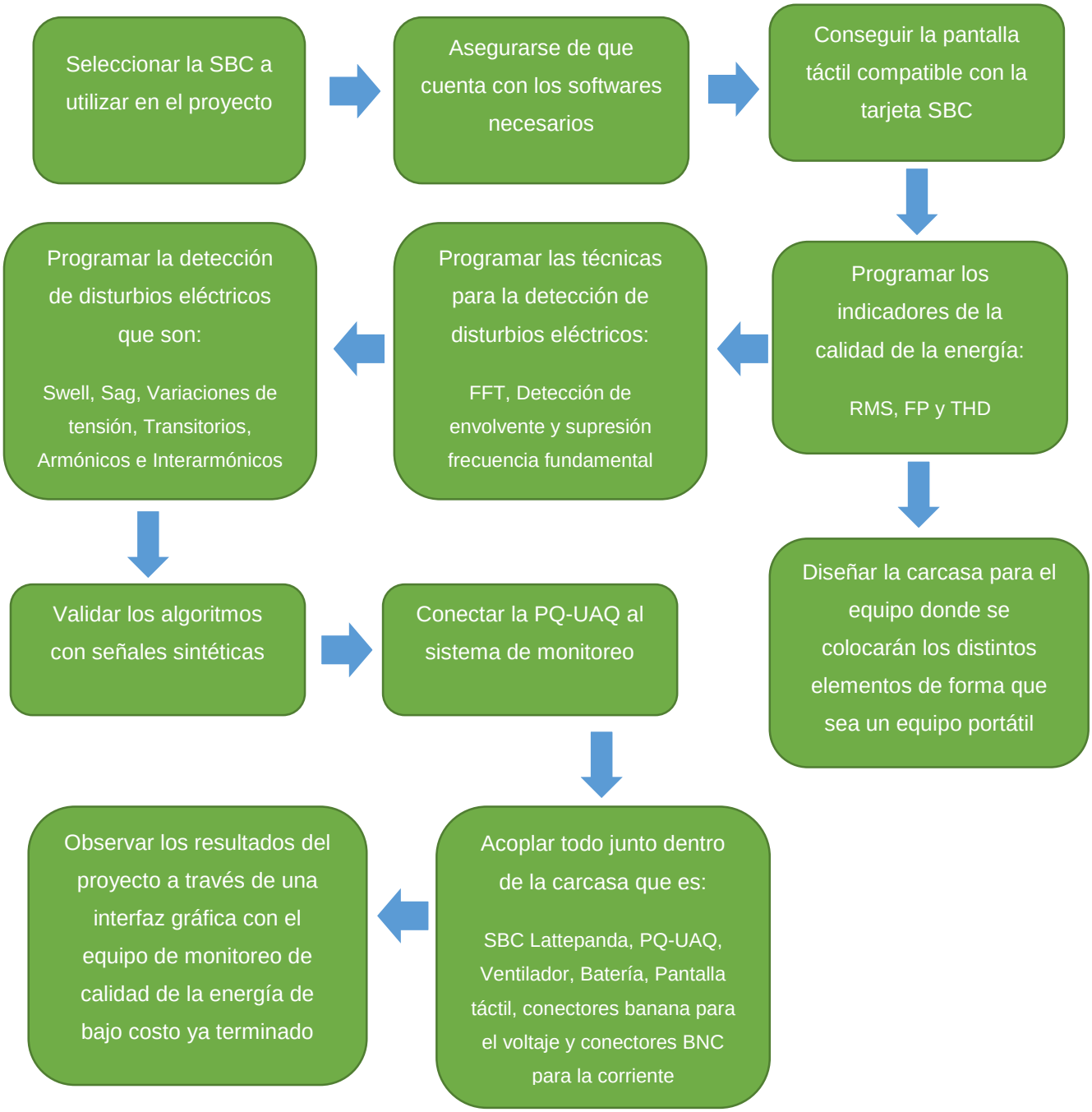


Figura 15 Metodología a detalle

3.3 Selección de SBC

La selección de la SBC se llevó a cabo a partir de sus características concentrándonos en la memoria RAM en la memoria de almacenamiento y en el procesador que integra la computadora, además de considerar de que cuente con mejores características que las que se tiene en una RaspBerry Pi y a su vez considerar un precio adecuado para estar dentro de la clasificación de elemento de bajo costo.

Considerando lo anterior se seleccionó una computadora modelo Lattepanda 4g/32Gb por considerar ser la mejor opción considerando la potencia, el precio y su comparativa con la competencia principal la cual se muestra en la **¡Error! No se encuentra el origen de la referencia.** siguiente.

Tabla 5 Comparativa Lattepanda-RaspBerry pi

Características:	**Lattepanda 4Gb/64G**	**Raspberry Pi 3 Model B+**
Procesador	Intel Cherry Trail Z8350 Quad Core 1.8GHz	Broadcom BCM2837B0, Cortex-A53 (ARMv8) 64-bit SoC 1.4 GHz
Unidad de procesamiento gráfico (GPU)	Intel HD Graphics, 12 EUs @200-500 Mhz	VideoCore IV 400 MHz
Conectividad inalámbrica	WiFi y Bluetooth 4.0	2.4GHz / 5GHz IEEE 802.11.b/g/n/ac Bluetooth 4.2, BLE
Entradas y Salidas (E/S)	6 E/S del procesador Cherry Trail, 20 E/S de Arduino Leonardo, 6 E/S para sensores de gravedad.	E/S 40 pines
Sistema operativo	Windows 10	Raspbian
Precio	$ 4,000 MX	$ 1,100 MX

3.4 Programación y validación de los algoritmos

Para la programación los algoritmos estos se implementaron en lenguaje de programación C++ haciendo uso de Qt como compilador de los archivos, los algoritmos primeramente se realizaron como una aplicación de consola para poder visualizar los resultados y demás variables importantes del código.

Los algoritmos programados fueron primeramente los indicadores de voltaje y corriente que son el valor rms para voltaje, ecuación 11, la potencia aparente, ecuación 14, la potencia reactiva, ecuación 15, potencia activa, ecuación 16 y factor de potencia, ecuación 13, continuando con la programación de la transformada de Fourier, utilizando el método de Radix-2, presentado en la sección pasada, con los valores de la señal en frecuencia, se calcula los indicadores de THD y TIHD para las fases de voltaje, ecuación 9. También se programó la detección de envolvente como se indica en la sección 2.3.2 y por último se programan los casos para la detección de los disturbios de la sección 2.5. El procedimiento descrito se puede observar resumido en el diagrama de flujo de la Figura 16.

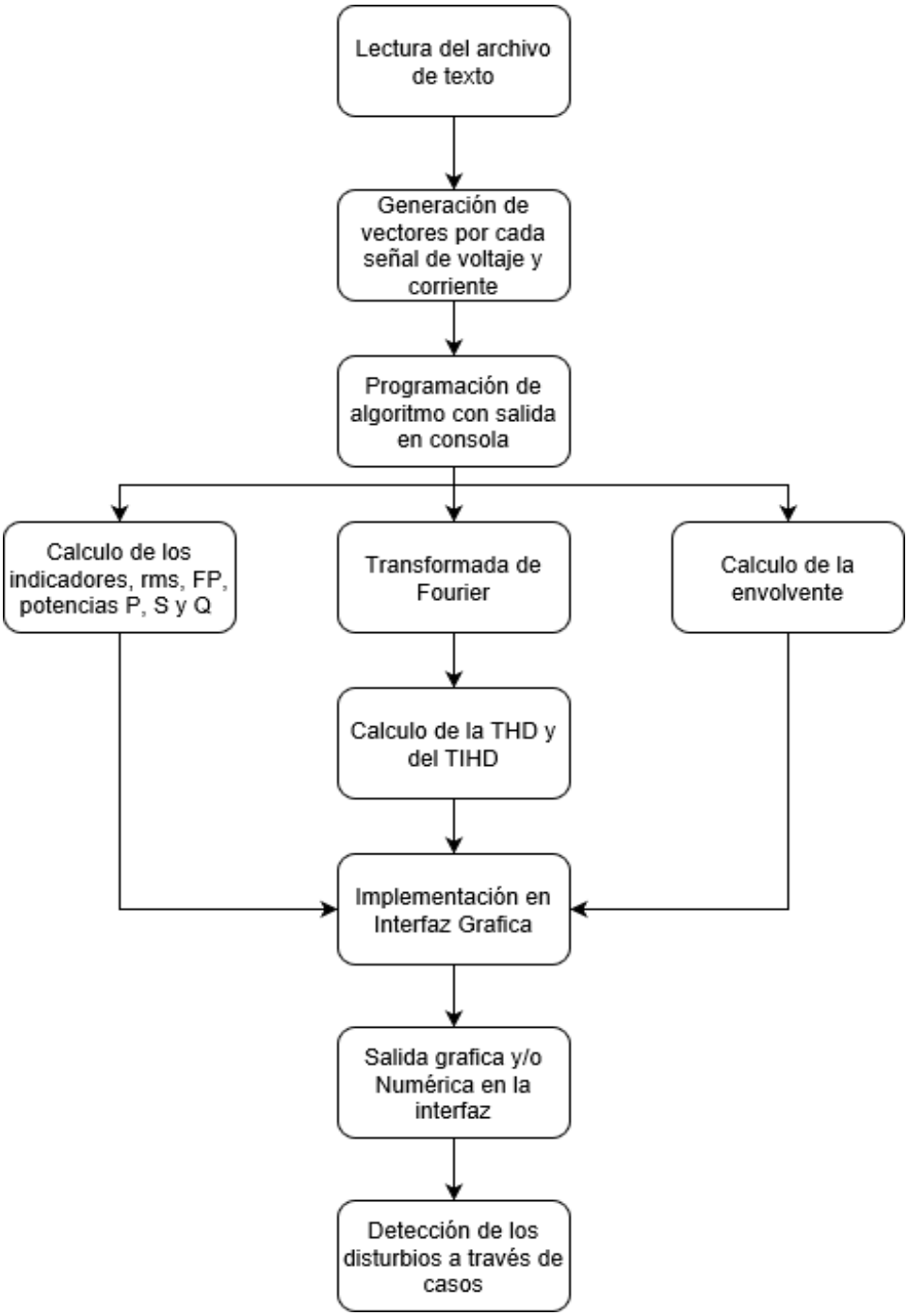


Figura 16 Diagrama del proceso de programación

Para llevar a cabo su validación lo que se realizó fue, hacer un script en Matlab con una señal sintética donde todos sus parámetros son conocidos, esta señal se guarda en un archivo de texto o .txt, a continuación el programa lee los datos que representan la señal en el archivo de texto, guarda todo en una variable

y realiza el procesamiento correspondiente según sea el algoritmo, los resultados obtenidos por el programa se guardan nuevamente en un archivo de texto, y este nuevo archivo se lee y grafica en Matlab para su mejor representación. Lo anterior se resume de manera gráfica en la Figura 17.

Figura 17 Diagrama Validación de algoritmos

3.5 Interfaz gráfica

Para llevar a cabo la interfaz gráfica es necesario tomar en cuenta algunos aspectos importantes para el diseño, como el tamaño de la ventana, para que se ajuste al tamaño de la pantalla táctil compatible con la SBC, el tamaño de los botones o elementos ahí presentados, al igual que el texto, para que tenga el tamaño adecuado para que sea de fácil lectura por el usuario.

Capítulo 4

4. Resultados

Los resultados de este proyecto se muestran siguiendo la metodología, donde es la realización de un prototipo físico del equipo de medición planeado, la programación y validación de los algoritmos con señales sintéticas, la interfaz gráfica, pruebas con señales reales y la implementación funcional en el dispositivo.

4.1 Equipo físico

El diseño del equipo se realizó primero en un programa de modelado de piezas 3D llamado SolidWorks, en este programa se diseñaron cada una de las piezas de la estructura y luego se realizó un ensamble con todas las piezas y demás elementos que componen el equipo de medición, los elementos se aprecian mejor en la Figura 18.

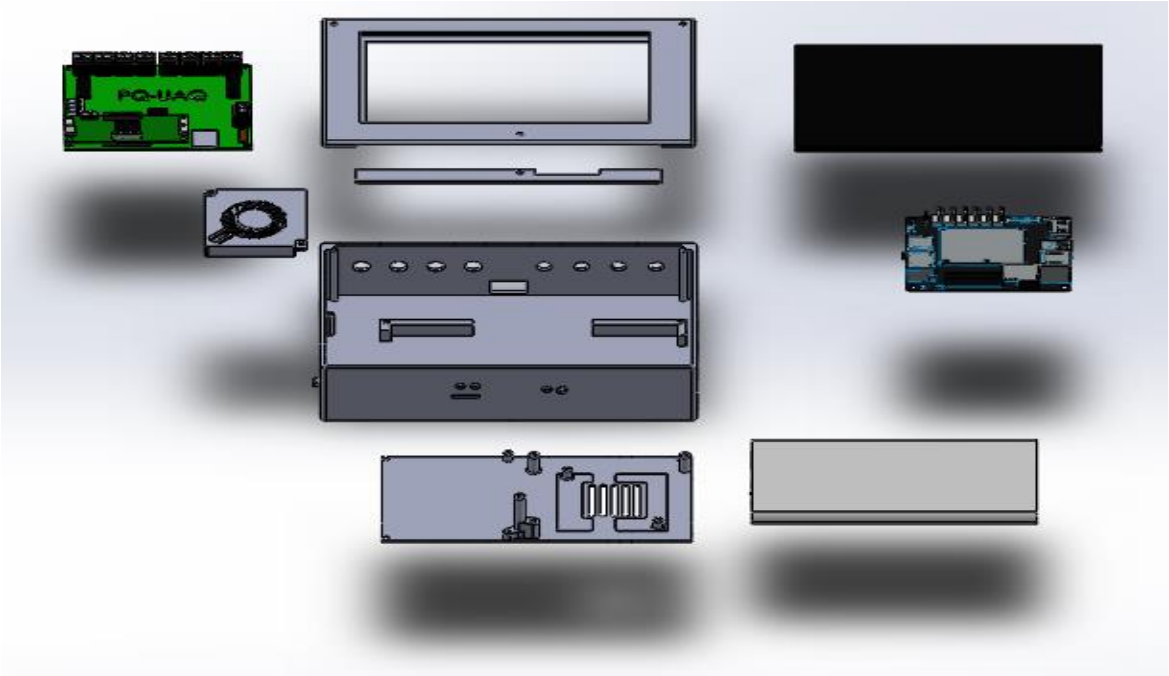


Figura 18 Piezas y componentes del ensamble 3D

En la Figura 19 se puede observar el armado completo del dispositivo en 3D, que es la union de todas las piezas mostradas en la Figura 18, mostrando el ensamble y otorga una vision clara de como debe de quedar el prototipo final.

Se diseño primeramente en 3D para poder realizar los cambios y modificaciones necesarios antes de fabricar el diseño final, ademas de poder ajustar las dimensiones según las piezas, y es indispensable si se planea manufacturar utilizando algun proceso de control numerico CNC, como es el caso de la impresión 3D, debido a que se requiere de archivos .STL para poder imprimir las piezas y este tipo de archivos es generado en SolidWorks.

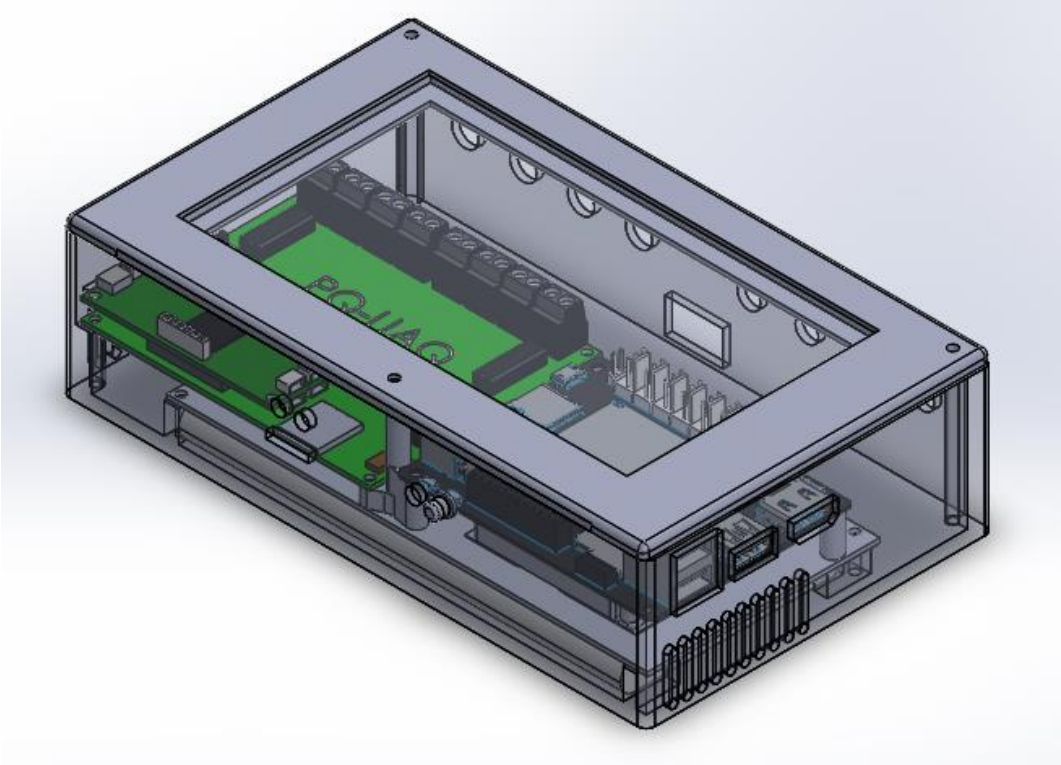


Figura 19 Ensamble completo en el dispositivo en 3D

Una vez definido el diseño en 3D, se mandaron a imprimir las piezas correspondientes en una impresora 3D, utilizando PLA como material para la impresión, la Figura 20 siguiente, muestra una fotografía de la base del dispositivo impreso en PLA, la cual fue impresa en una máquina propiedad de la escuela, y tardo aproximadamente 12 horas en ser concretada, y gasto una cantidad mínima de material.

Figura 20 Fotografía de base del sistema impresa en PLA

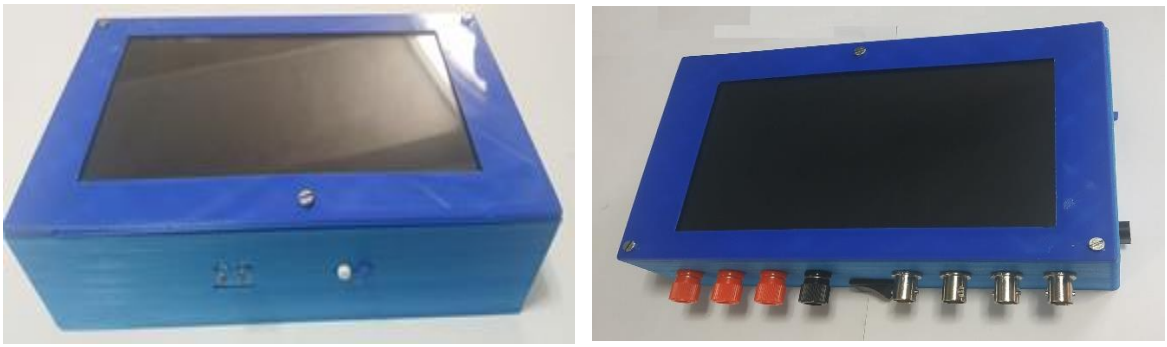
En la imagen siguiente la Figura 21 se muestra una fotografía del dispositivo armado con todas las piezas planteadas en el diseño 3D mostrando el resultado de imprimir e incorporar todos los elementos, todas las piezas impresas tomaron poco más de 18 horas en ser completadas y consumieron menos de una cuarta parte del rollo de PLA destinado únicamente para ese propósito.

Figura 21 Fotografía del dispositivo armado

En la Figura 22 de la a) a la d) muestra distintas vistas del dispositivo armado, permitiendo observar las ranuras botones, y demás elementos del dispositivo, en la Figura 22 a), permite observar dos botones el azul para el encendido y el blanco para resetear, dos leds, para conocer el estado de la PQ-UAQ y una ranura para una micro SD de la PQ-UAQ, Figura 22 b), la vista trasera, permite observar los conectores para las sondas de voltaje y corriente que le llegan a la PQ-UAQ, 4

conectores banana hembra para chasis y 4 conectores BNC hembra para chasis, además de un interruptor de encendido para la batería, Figura 22 c), la vista lateral derecha, permite observar 3 entradas USB para la SBC Lattepanda, y una salida de HDMI, Figura 22 d), la vista lateral izquierda, cuenta con un interruptor para la batería, un conector hembra para un cargador de 5V, y una ranura para la segunda memoria micro SD de la PQ-UAQ.

a) b)



c) d)

Figura 22 Fotografías de diferentes vistas del dispositivo a) Frontal, b) Trasera, c) Lateral Derecha y d) Lateral Izquierda

4.2 Señales Sintéticas y prueba de los algoritmos

Para la generación de las señales sintéticas se utilizó Matlab realizando un script con el cual podemos programar la cantidad deseada de señales, con cualquier amplitud y frecuencia, además de poder programar disturbios eléctricos, tales como Sag o Swell, agregar armónicos, es decir generar señales completamente personalizadas, el código del Script para la generación de señales sintéticas se

muestra en el Anexo 1, en la Figura 23 se muestra un diagrama con las posibilidades de las señales sintéticas, donde todo comienza con un script de Matlab para generar la señal dentro del cual se puede modificar prácticamente cualquier parámetro de la señal o incluso agregarle disturbios si se desean.

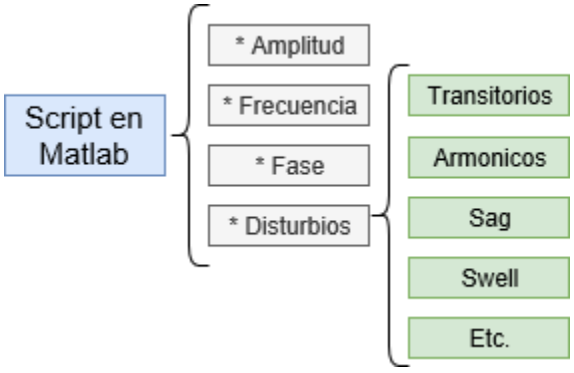


Figura 23 Diagrama Señales sintéticas

Para la prueba de los algoritmos se realizó uno a uno utilizando las señales sintéticas para la validación de los mismos, ya que los algoritmos están programados para la lectura de un archivo de texto el cual contiene la señal sintética generada, y corroborando los resultados con las herramientas de Matlab, un ejemplo de ello se muestra en la Figura 24 donde se observan dos graficas comparando el resultado del algoritmo FFT programado, la gráfica de color rojo, y utilizando las herramientas de Matlab, la gráfica de color azul.

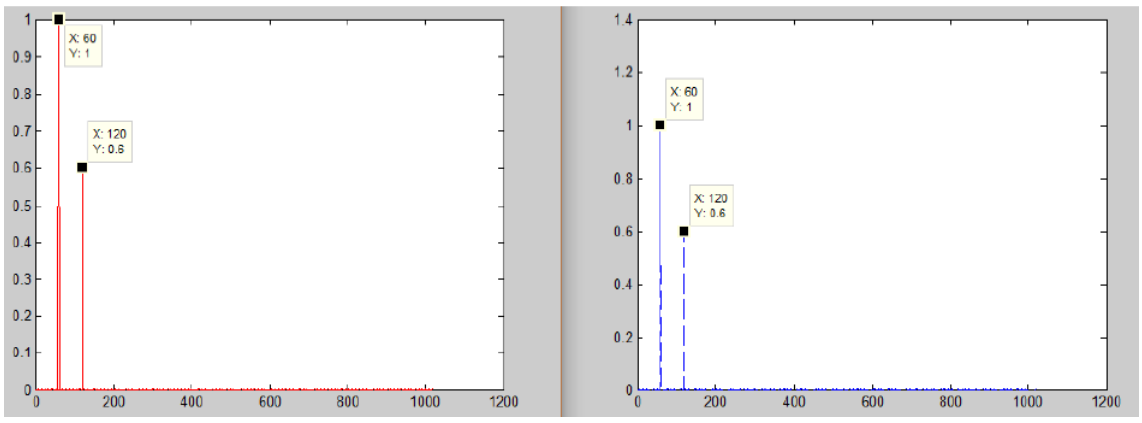


Figura 24 Comparación grafica del algoritmo FFT

4.3 Interfaz Gráfica

Se desarrolló una interfaz gráfica para la implementación y agrupación de los algoritmos diseñados, utilizando Qt como espacio de trabajo y C++ como lenguaje de programación.

La interfaz gráfica cuenta con una ventana principal con un espacio para la representación gráfica, una ventana donde se despliegan el número de puntos para la representación gráfica en potencia de 2 desde 128 hasta 8192, se tiene el botón graficar, el cual nos muestra la gráfica cargada en caso de que se desee cambiar el tamaño de la ventana posteriormente de cargar los datos, un apartado donde se muestran los calores de los indicadores eléctricos para las tres fases, cuenta con una barra de herramientas, donde se encuentra la opción archivo, herramientas y eventos, y 6 casillas seleccionables para visualizar o no, cada una de las fases sus valores de corriente y voltaje, como se observa en la Figura 25.

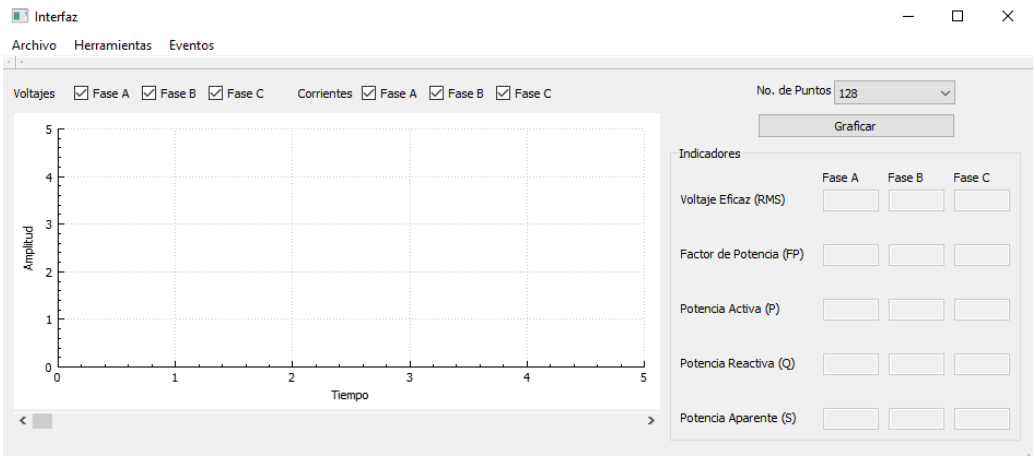


Figura 25 Ventana principal de la interfaz grafica

Al dar clic en el botón de archivo, seguido en el botón de abrir se despliega la ventana de búsqueda de archivos y está configurada para leer únicamente archivos de texto es decir archivos de formato .txt, la Figura 26 muestra la ventana de abrir archivos encima de la ventana principal.

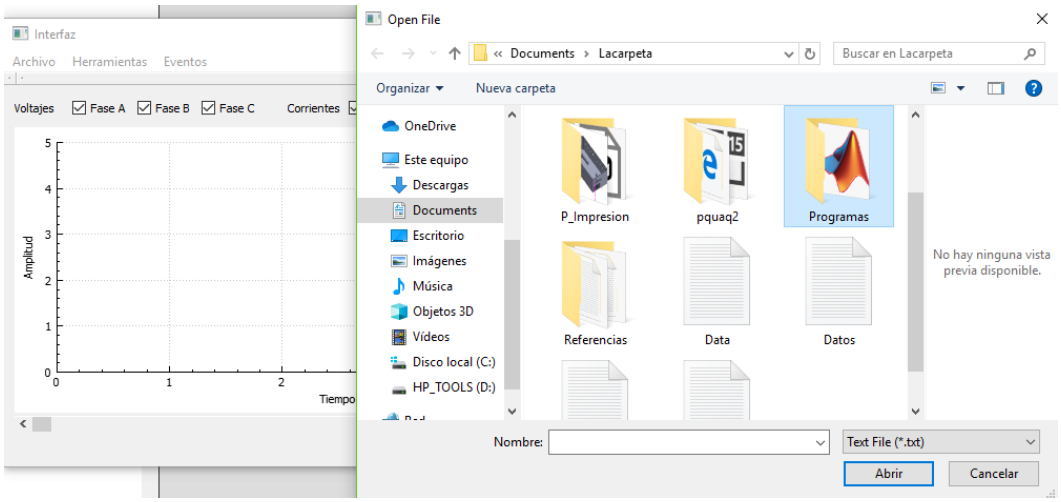


Figura 26 Interfaz gráfica con ventana de búsqueda de archivos

La manera que se visualizan los datos utilizando una señal real adquirida por el PQ-UAQ se muestran en la Figura 27, y como se puede observar en la imagen, de la señal se grafican las 3 señales de voltaje y 3 de corriente valores de los indicadores entre más elementos previamente mencionados.

Figura 27 Ventana de la interfaz gráfica con datos reales

La forma en que se presenta la transformada rápida de Fourier se muestra en la Figura 28 donde además de observar gráficamente la FFT se observan los valores de distorsión armónica total y distorsión interarmónica total, THD y TIHD respectivamente, para cada una de las 3 fases.

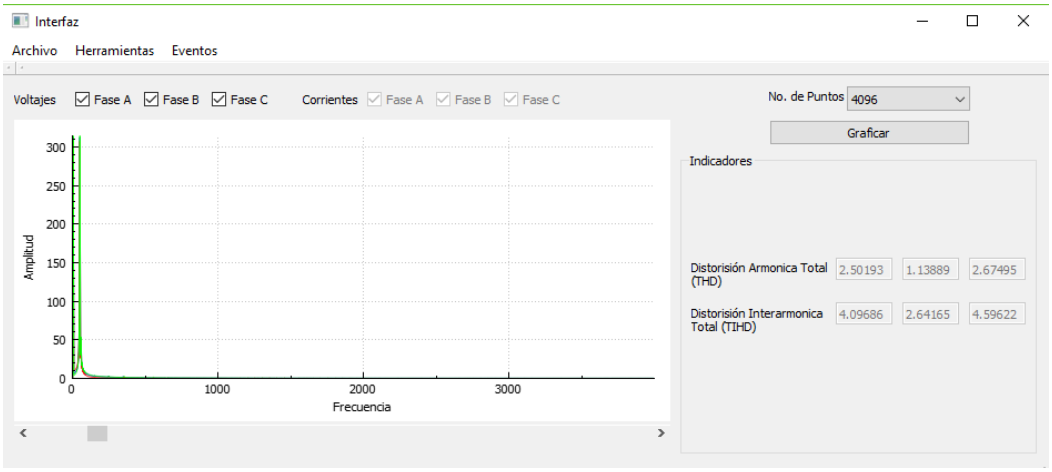


Figura 28 Interfaz gráfica con la visualización de FFT

La envolvente de igual manera tiene su apartado dentro del menú de herramientas, como se observa en la Figura 29, donde se observa la envolvente sobre cada una de las fases de voltaje.

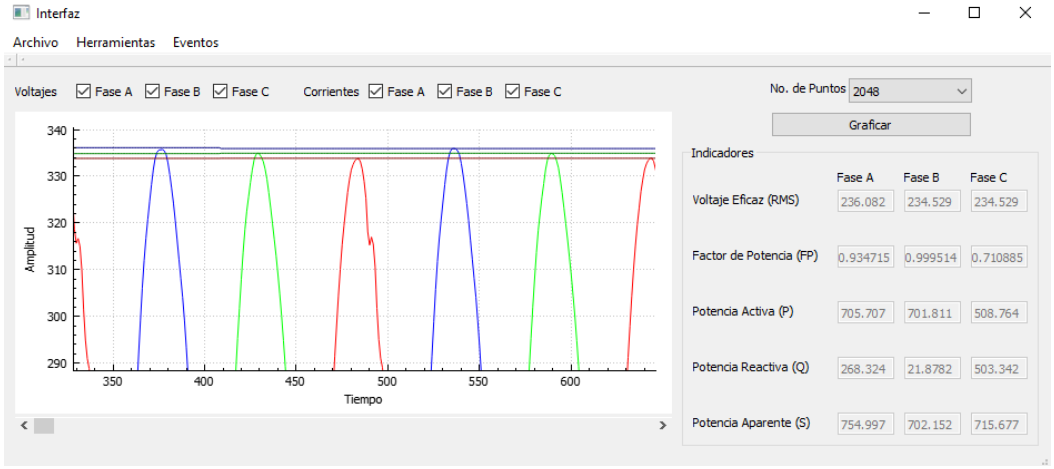


Figura 29 Interfaz gráfica en el apartado de envolvente de la señal

En la sección de eventos se registran los disturbios eléctricos detectados, ahí se numeran, se da una breve descripción del motivo de su detección y se coloca enseguida el valor detectado que infringió los límites permitidos para cada caso, la lista de disturbios es posible borrarla completamente y guardarla en un archivo de texto. Figura 30 muestra la pantalla de la interfaz gráfica una vez presionado el botón de eventos de la pantalla principal.

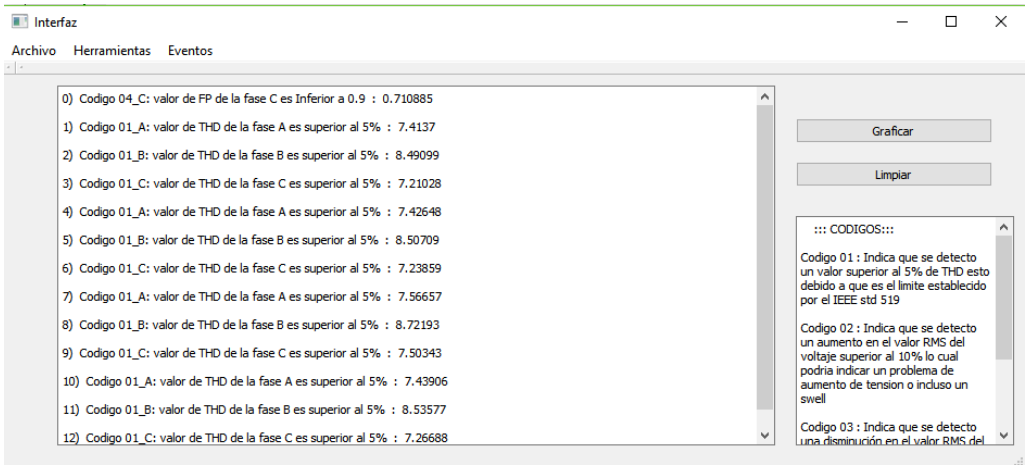


Figura 30 Interfaz gráfica, pantalla de registro de eventos

Para mayor información sobre el funcionamiento de la interfaz gráfica y utilización del dispositivo, ir al manual de usuario del equipo en el Anexo 4.

4.4 Adquisición y Pruebas con señales reales

Para la adquisición de las señales eléctricas se utiliza el Datalogger PQ-UAQ, el cual estuvo conectado a una central de energía eólica en España, cuyos datos fueron almacenados en discos duros por la gran cantidad de información y son dichos datos los cuales se utilizan para comprobar el funcionamiento del dispositivo y los algoritmos de la interfaz gráfica.

Las señales almacenadas que se generan automáticamente por la adquisición de datos del dispositivo PQ-UAQ en formato .mat y para poder reconstruir las señales es necesario utilizar un script en Matlab para poder generar los datos en un formato manejable donde se observen claramente las señales de corriente y voltaje, el script para la reconstrucción de las señales reales adquiridas por el PQ-UAQ se muestra en el Anexo 2, además se cuenta con un script para escribir las señales reconstruidas en archivos de texto con formato .txt los cuales son los archivos que son leídos por la interfaz gráfica.

Las pruebas de las señales reales se realizaron cargando individualmente cada señal adquirida por la PQ-UAQ, la primer señal a analizar se muestra en la Figura 31.

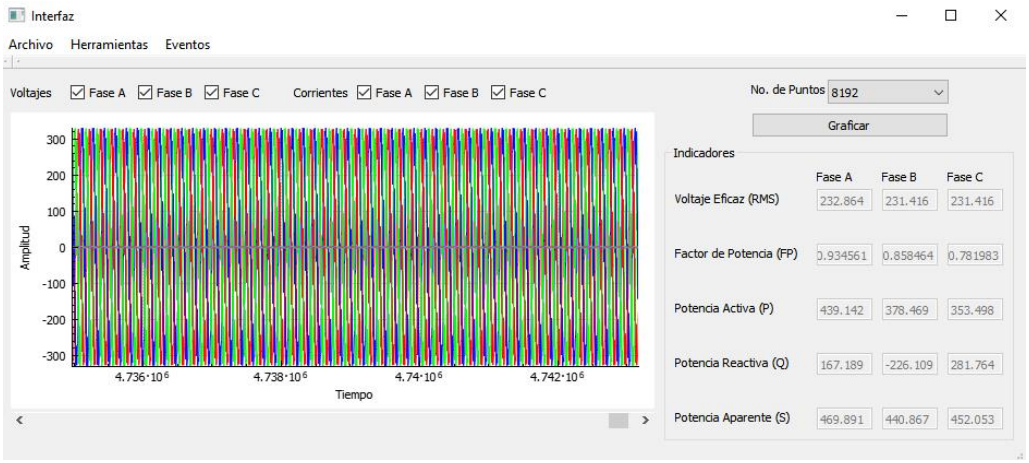


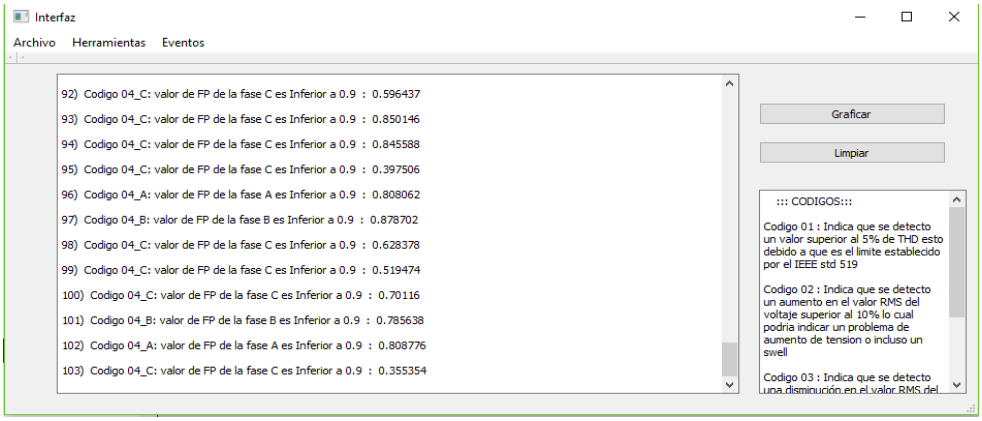
Figura 31 Interfaz gráfica, prueba de primer señal real

De la primera señal analizada se observaron un total de 77 disturbios eléctricos todos ellos correspondientes a tener un valor por debajo de los límites que impone la CFE, es decir que el factor de potencia está por debajo de 0.9, y esto se puede observar en repetidas ocasiones y en cualquiera de las 3 fases de voltaje, el umbral de detección puede ser modificado según las especificaciones del usuario puede disminuirse o aumentarse hasta un valor máximo de 1.

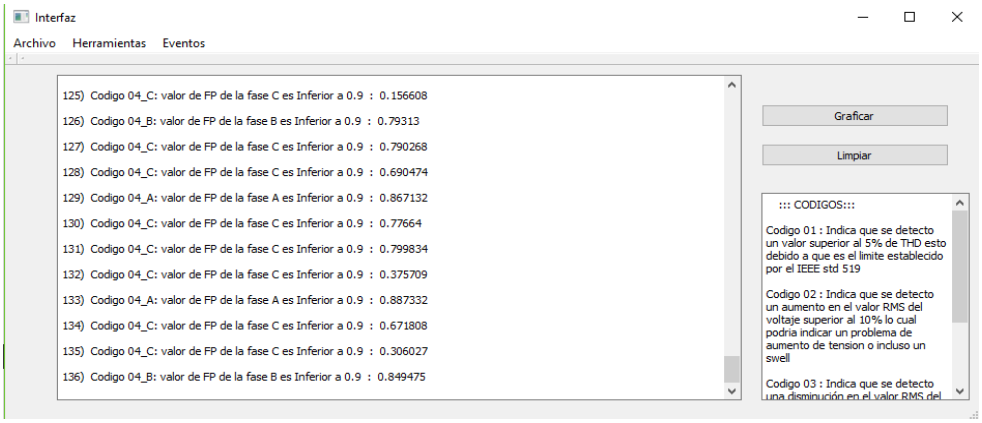
Figura 32 Interfaz Gráfica, disturbios detectados de la primer señal real de prueba

En las pruebas siguientes nuevamente se obtuvieron únicamente problemas con el factor de potencia, la Figura 33 a), b), c), d) y e) muestra los resultados del análisis directamente en la interfaz gráfica con un total de 103, 136, 110, 72 y 117 respectivamente.

a)

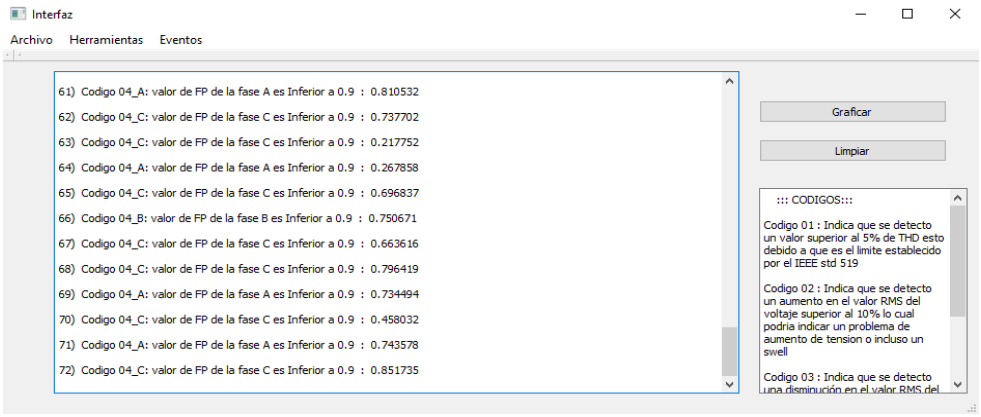


b)



c)

d)



e)

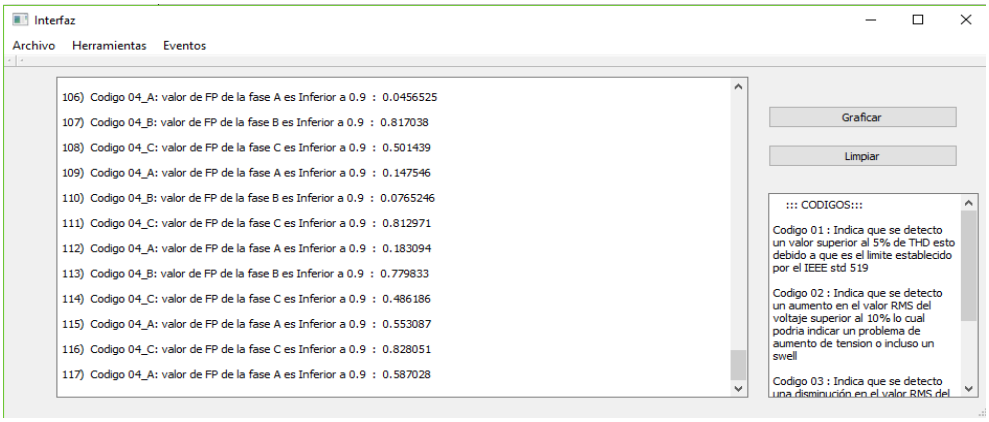


Figura 33 Pruebas a distintas señales reales

Ahora bien tras haber realizado el análisis de cada señal y observar los disturbios de cada una de estas señales se logra reducir una gran cantidad de información, ya que conociendo los disturbios presentes en cada señal se puede decidir si se desea conservar la señal para un mayor análisis futuro o desecharla definitivamente y liberar espacio en el disco duro, debido a que cada señal almacenada en un archivo de texto ocupa 290 Mega Bytes aproximadamente y un archivo de salida tras el análisis de la interfaz gráfica tiene un tamaño aproximado de 5.85 Kilo Bytes, y este archivo es suficiente cuando se quiere tener un registro, es decir que podría sustituir al archivo original y habría una disminución de espacio muy considerable siendo el archivo de salida cerca de 50 mil veces más pequeño en espacio de memoria.

4.5 Implementación funcional del equipo

La Figura 34 muestra una fotografía del equipo para el análisis de la calidad de la energía utilizando una de las señales reales.

Figura 34 Fotografía del equipo corriendo la interfaz grafica

En la Figura 35 se puede apreciar al dispositivo realizando la transformada rápida de Fourier.

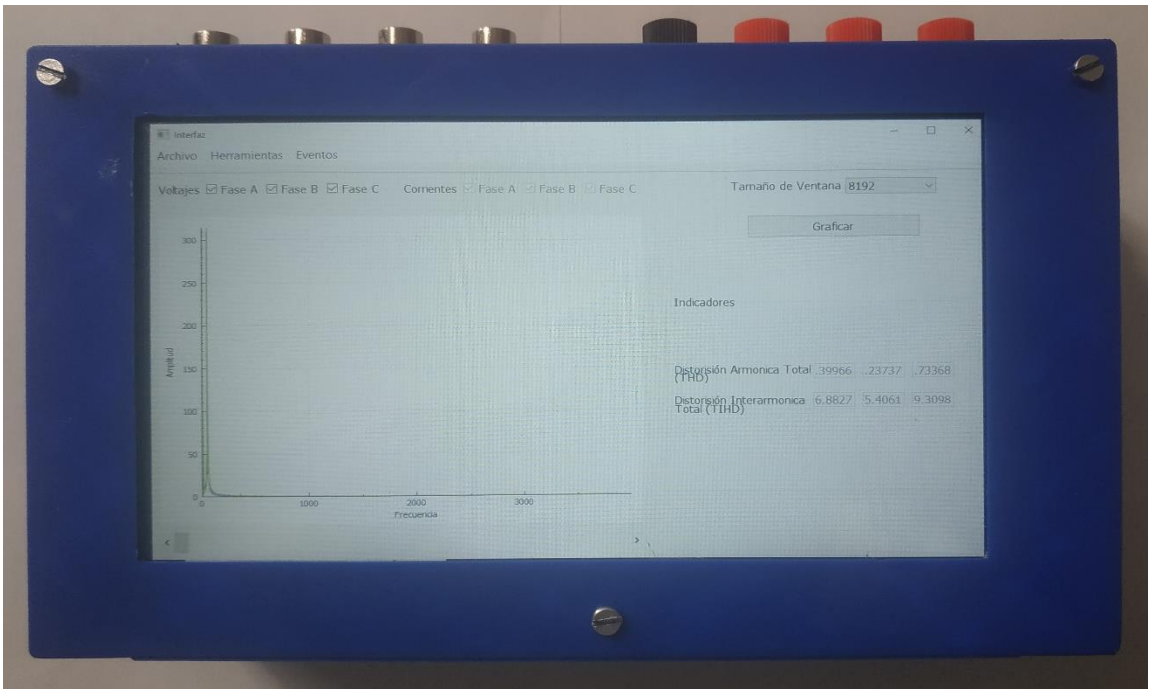


Figura 35 Fotografía del dispositivo en el apartado de FFT

Por su parte la Figura 36 muestra al dispositivo realizando el apartado de envolvente de la señal trifásica.

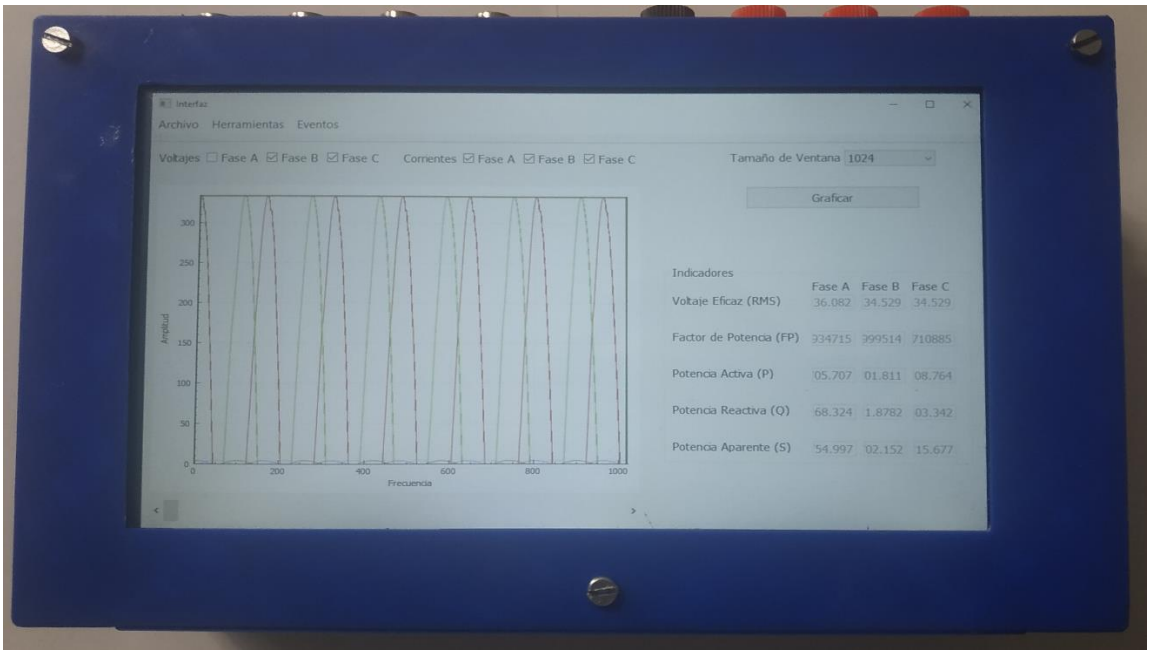


Figura 36 Fotografía del dispositivo realizando la envolvente de la señal

Por último en la Figura 37 muestra una fotografía del dispositivo en el apartado de registro de eventos analizando la señal.

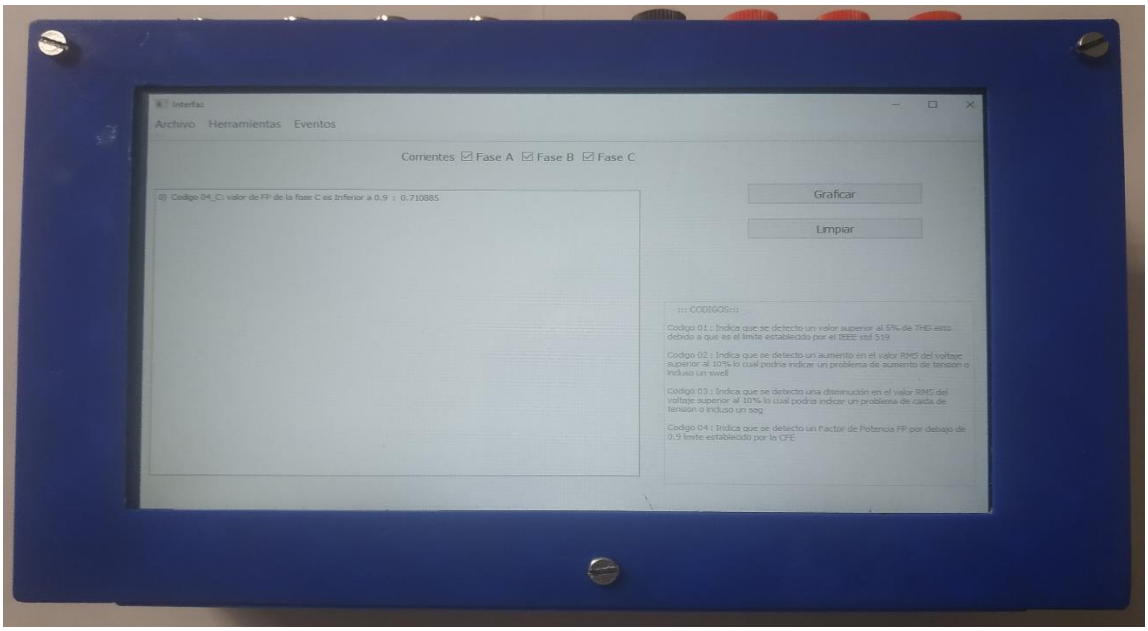


Figura 37 Fotografía del dispositivo en el apartado de registro de eventos

Capítulo 5

Conclusiones

La elaboración de un dispositivo resulta ser una tarea más complicada de la esperada, debido a que se requiere conocer a fondo la problemática a resolver, en este caso poder visualizar y analizar disturbios de la red eléctrica, generar una investigación profunda sobre el tema y saber seleccionar la información, en este caso saber seleccionar adecuadamente las técnicas para el análisis de señales que se ajusten a las necesidades y sean efectivas, caso es el de la transformada rápida de Fourier, de la cual existen múltiples algoritmos para su solución, o inclusive existen variantes de la misma transformada de Fourier que podría dar mejores resultados pero estos suelen ser más complejos de programar o requieren más recursos computacionales.

Se logró observar que el problema más frecuente, observado en las señales eléctricas analizadas, es la variación del factor de potencia, el cual no implica gran daño a los equipos conectados, pero si genera gastos adicionales para las empresas, ya que al menos en México existe una sanción económica para las empresas que proporcionen un factor de potencia menor a 0.9. Hay que resaltar que el método del análisis utiliza ventanas mucho menores que el que utilizaría la compañía administradora, la cual promedia cada 5 minutos, esto con el objeto de profundizar en el análisis y obtener un mejor entendimiento de cada señal parte a parte de cada ventana analizada.

Para el proyecto resulto interesante la investigación de las SBC esto a que permite conocer un nuevo grupo de dispositivos electrónicos que resultan interesantes y permite explorar nuevas opciones de movilidad sin perder una gran cantidad de recursos computacionales como ocurre con los microcontroladores habituales, además de observar que existen una gran cantidad de modelos y marcas disponibles cada uno con sus particularidades.

El dispositivo para el análisis de la calidad de la energía ofrece al usuario una nueva posibilidad de utilizar el equipo PQ-UAQ, el cual es un equipo diseñado por la Universidad Autónoma de Querétaro, que originalmente está diseñado para adquirir las señales eléctricas y almacenar dichas señales en memorias micro SD, pero con el complemento de la computadora, la batería, la carcasa y demás componentes, se logra aumentar su funcionalidad, al poder leer los datos generados por el Datalogger PQ-UAQ, analizarlos y presentarlos gráficamente.

Prospectivas

Algunos puntos que requieren atención futura, son la velocidad y conexión con la PQ-UAQ, debido a que a través del proyecto se pudo observar que la cantidad de información recibida es excesiva, y resulta en una gran carga computacional la cual incluso el SBC seleccionado tiene problemas para procesar, esto se observa en el tiempo que tarda en abrir un archivo, aproximadamente dos minutos, y la lentitud con la que responde ante el cambio grafico de la barra de desplazamiento,

en cuanto la conexión con la PQ-UAQ es principal problema es la disponibilidad de la misma y que se requiere mayor capacitación para su optima utilización.

Más trabajo futuro interesante y posible, es de continuar ampliando el catálogo de herramientas utilizadas para analizar las señales eléctricas, de igual manera ampliar los disturbios que se pueden detectar, probar con técnicas más complejas, pero más precisas para el análisis de las señales, por ejemplo, podría ser utilizar la transformada de tiempo corto de Fourier, o la transformada wavelet.

De igual manera al ser un equipo de arquitectura abierta permite la modificación prácticamente total, es decir, tanto software como hardware, y un trabajo futuro interesante con el cual no existirían problemas de velocidad y el análisis seria completamente en línea y de tiempo real, podría ser utilizando un FPGA como tarjeta para el procesamiento de las señales y quizás remplazar la SBC para tener un equipo de gama alta, más complejo y complicado de programar pero mucho más eficiente y sin retraso alguno por el tiempo de procesamiento.

Referencias.

- Ángel Silva, Miguel. 2005. Diseño y construcción de un prototipo como alternativa para la monitorización de interrupciones y caídas de tensión.
- Batista, José. Martins, Julio y Afonso, Joao. 2003. Low-Cost digital system for power quality monitoring.
- Chapra, Steven y Canale, Raymond. 2006. Métodos numéricos para ingenieros. Editorial McGraw-Hill Interamericana
- Dzul Ayala, Walter. 2016. Software de procesamiento de señales eléctricas trifásicas.
- Enríquez Harper, Gilberto. 1999. El ABC de la calidad de la energía. Editorial Limusa.

- Fernández Negroe, Bernardo. 2016. Metodología de calibración para equipo de monitoreo eléctrico propietario PQ-UAQ.
- González Abreu, Artvin. 2016. Sistema basado en FPGA para detección de Sag y Swell en fresadora CNC mediante micro algoritmos genéticos y análisis de su dinámica ante el disturbio.
- Jaya Bharata Reddy, M. Raghupathy, R. Venkatesh, K. y Mohanta, D. 2013. Power quality análisis using Discrete Orthogonal S-Transform (DOST).
- Lima, R. Quiroga, D. Reineri, C. y Mognago, F. 2008. Hardware and software architecture for power quality analysis.
- Mejía Barrón, Arturo. 2013. Monitoreo remoto vía ethernet de la calidad de la energía en sistemas de baja tensión.
- Molano Clemente, Martin. 2013. Desarrollo e implementación de un analizador de calidad de energía con base en FPGA para motores trifásicos.
- Morales Velázquez, Luis. Romero Troncoso, Rene de Jesús. Herrera Ruiz, Gilberto. Morínigo Sotelo, Daniel y Osornio Rios, Roque. 2017. Smart sensor network for power quality monitoring in Electrical installations.
- Proakis, John y Manolakis, Dimitris. 2007. Tratamiento digital de señales. Editorial Pearson
- Ramírez Duran, Iván. 2012. Monitoreo y Análisis de la calidad de la energía eléctrica en el laboratorio de automatización
- Valtierra Rodríguez, Martin. Osornio Rios, Roque. García Pérez, Arturo y Romero Troncoso, Rene de Jesús. 2013. FPGA Based neural network Harmonic estimation for continuous monitoring of the power line in industrial applications.
- Chapra, Steven. 2012. Applied numerical methods with MATLAB for engineers and scientists. Editorial McGraw-Hill

Anexos

Anexo 1: Script generación de señales sintéticas

```
%Script para generar señales
clear variables
clear all
clc

%Generar señal sintetica
f = [50 100];
A = [135 25];
Fs = 2048; %8000,512,1000
N = 2^11;

for k=0:N-1
    x(k+1)=A(1)*sin(2*pi*f(1)*k/Fs)+A(2)*sin(2*pi*f(2)*k/Fs);%+A(3)*sin(2*pi*f(3)*k/Fs);%↙
+A(4)*sin(2*pi*f(4)*k/Fs);
    y(k+1)=A(1)*sin(2*pi*f(1)*k/Fs);
    z(k+1)=A(2)*sin(2*pi*f(2)*k/Fs);
end
plot(x)
hold on
plot(y,'r')
hold on
plot(z,'g')
legend('Suma de ambas','Fundamental','Armonica');

fileID=fopen('Datos.txt','w');
fprintf(fileID,'%4.6f\r\n',x);
fprintf(fileID,'\n');
fclose(fileID);

H=N/2;
for k=0:H-1
    freq(k+1)=k*Fs/N;
end
xf=abs(fft(x));
xff=xf(1:H)/H;

figure
plot(freq,xff)
```

Anexo 2: Script reconstrucción de señales reales

```
function [I V t tick] = load_signals(day, hour, minute)
%day = 000;
%hour = 022;
%minute = 001;
 dia = strcat('C:\Users\USER\Documents\Lacarpeta\dia_', num2str(day,'%.3d'));
%dia = strcat('/dia_', num2str(day,'%.3d'));
hor = strcat('/hora_',num2str(hour,'%.3d'));
min = strcat('/minz_',num2str(minute,'%.3d'));
dir = strcat(dia,hor,min,'/');%strcat('.',dia,hor,min,'/');%

file = strcat(dir,'Ia.mat');
if( ~exist(file,'file')); return; end;
Ia = load(file);

file = strcat(dir,'Ib.mat');
if( ~exist(file,'file')); return; end;
Ib = load(file);

file = strcat(dir,'Ic.mat');
if( ~exist(file,'file')); return; end;
Ic = load(file);

% file = strcat(dir,'In.mat');
% if( ~exist(file,'file')); return; end;
% In = load(file);
In=zeros(1,length(Ic.Data));

file = strcat(dir,'Va.mat');
if( ~exist(file,'file')); return; end;
Va = load(file);

file = strcat(dir,'Vb.mat');
if( ~exist(file,'file')); return; end;
Vb = load(file);

file = strcat(dir,'Vc.mat');
if( ~exist(file,'file')); return; end;
Vc = load(file);


I = [ Ia.Gain*-1*(double(Ia.Data)-Ia.Offset)
    Ib.Gain*1*(double(Ib.Data)-Ib.Offset)
    Ic.Gain*1*(double(Ic.Data)-Ic.Offset)
    In]';
    %In.Gain*1*(double(In.Data)-In.Offset) ]';
% I = [ 0.0064564791*10*(double(Ia.Data)-Ia.Offset)
%     0.0064770145*10*(double(Ib.Data)-Ib.Offset)
%     0.0064809287*10*(double(Ic.Data)-Ic.Offset)
%     0.0063252307*10*(double(In.Data)-In.Offset) ]';

V = [ Va.Gain*(double(Va.Data)-Va.Offset)
    Vb.Gain*(double(Vb.Data)-Vb.Offset)
```

```
    Vc.Gain*(double(Vc.Data)-Vc.Offset) ]';

t = (0:(double(Ia.SampleNumber)-1))/Ia.SamplingFreq;

rt = datenum(Ia.RealTime,'dd:HH:MM:SS.FFF');
datestr(rt,'dd:HH:MM:SS.FFF');
for k=0:59

    %rtt = addtodate(rt,integer(k*(max(t)/60)),'second');
    rtt = addtodate(rt,ceil(k*(max(t)/60)),'second');
    st = datestr(rtt,'dd HH:MM:SS.FFF');
    %set(st,'HorizontalAlignment','right','VerticalAlignment','top','Rotation',45);

    tick.t(k+1) = k*10;
    tick.m(k+1,:) = st;

end
```

Anexo 3: Script para la escritura de señales en archivo de texto

```
[I V] = load_signals(0, 22, 5);

%V(:,3)=[];
%V(:,2)=[];
I(:,4)=[];
M = [V I];
fileID=fopen('Data5.txt','w');
%fprintf(fileID,'%4.6f\r\n',V);
fprintf(fileID,'%4.6f %4.6f %4.6f %4.6f %4.6f %4.6f\r\n',M.');
fprintf(fileID,'\n');
fclose(fileID);
```

Anexo 4: Manual de usuario

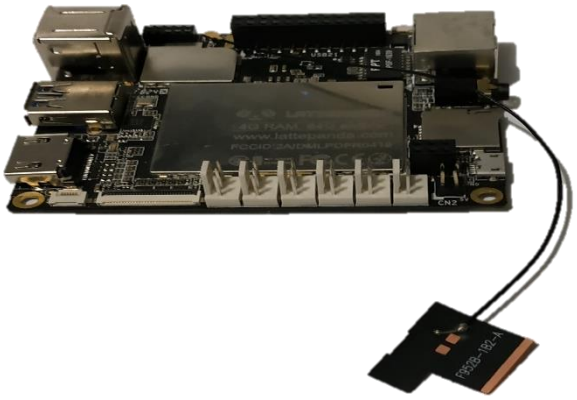
El presente manual tiene como objeto dar a conocer a detalle el dispositivo para monitorización de calidad de la energía de bajo costo utilizando SBC, y constará de dos partes, la primer será la parte física o elementos que componen al equipo y la segunda parte es la utilización de la interfaz gráfica incorporada en el equipo.

A continuación, se presentan listados los componentes del equipo.

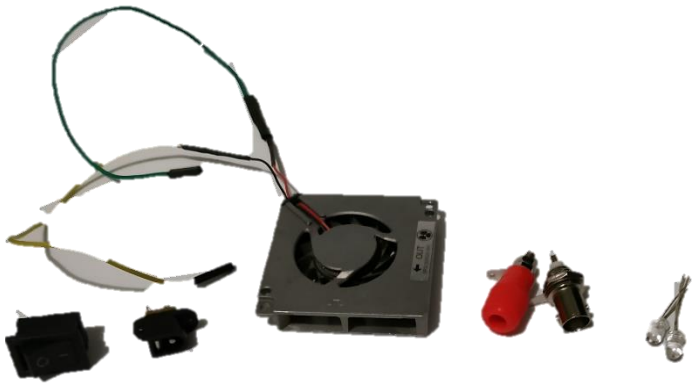
- Batería recargable de litio de 20,000 mAh.

- Pantalla Touch-Screen compatible con la SBC Lattepanda

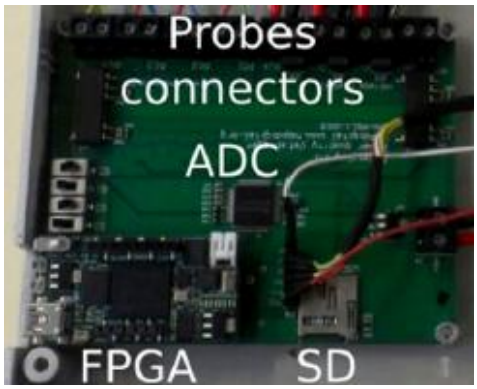
- Computador de placa única SBC Lattepanda



- Componentes extras como interuptor, plug para eliminador de recarga, conectores banana hembra, conectores BNC hembra, Leds y Ventilador 5 V.

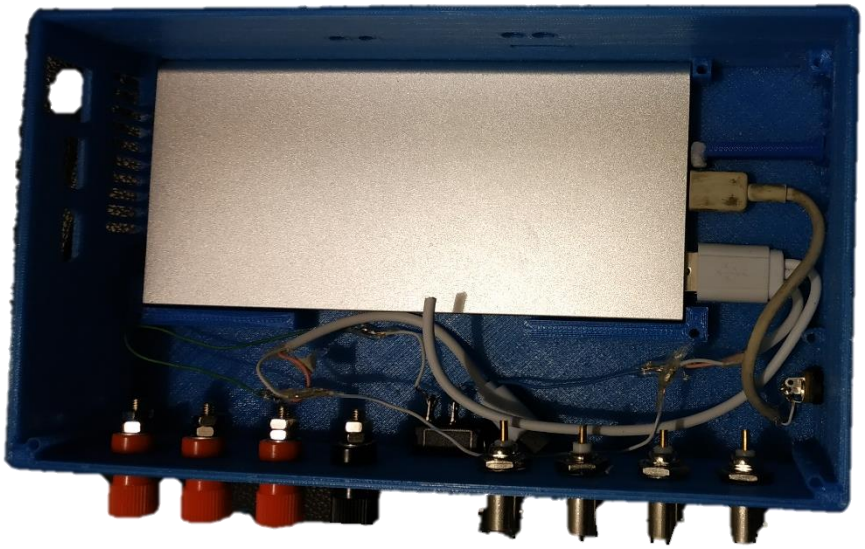


- Datalogger PQ-UAQ



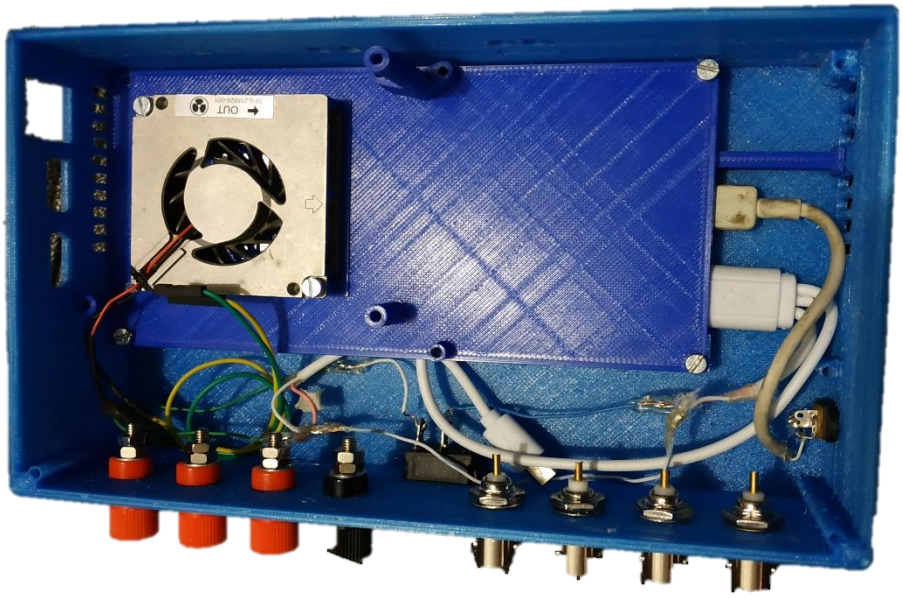
- Carcasa del dispositivo.

Para el armado y conexión de los componentes previamente mencionados, es a través de una forma de apilamiento, donde se coloca un componente sobre otro colocando primeramente la batería en el espacio designado como se muestra en la imagen siguiente, donde se pueden realizar las conexiones de la batería y los conectores de voltaje y corriente.

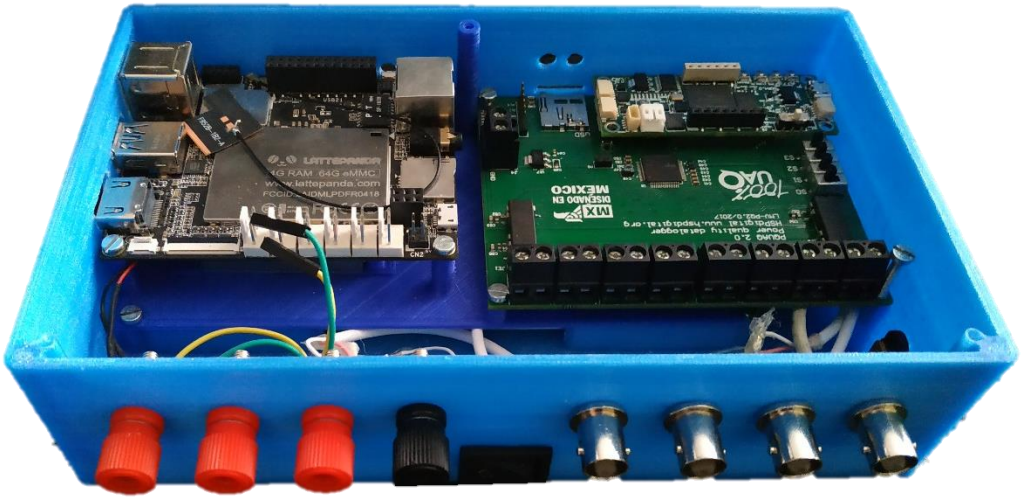


Enseguida se coloca la pieza del segundo nivel que sirve como soporte de los demás componentes y como una forma de asegurar que la batería no se salga de su lugar incluso con el dispositivo de cabeza como se observa en la imagen siguiente.

El primer elemento que tiene que ser colocado es el ventilado sobre el que descansara la SBC debido a que esta última tiene un fuerte problema con sobrecalentamiento.

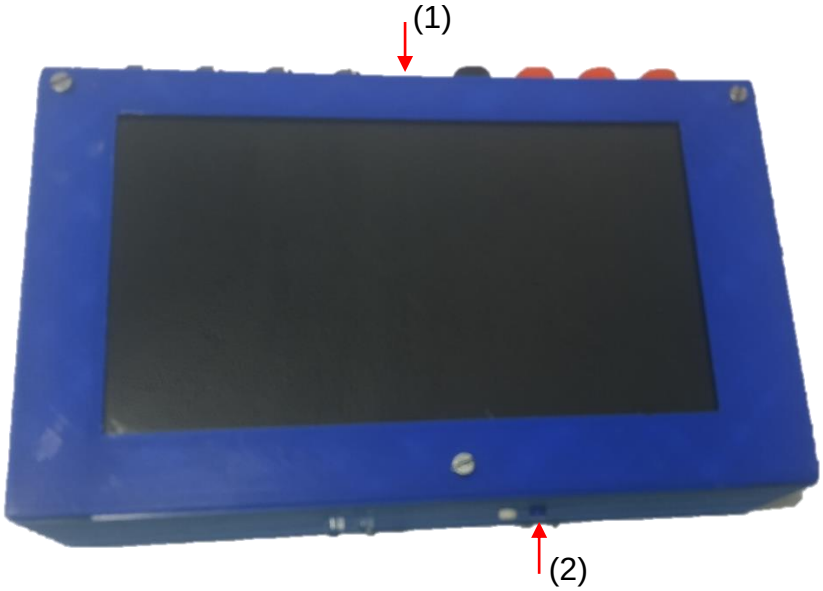


A continuación, es tiempo de colocar la SBC sobre el ventilador con el cual está en contacto y la PQ-UAQ se coloca como se muestra en la imagen siguiente.



Por ultimo se realizaron las conexiones de las terminales hacia la PQ-UAQ, la pantalla hacia la SBC, se colocan los botones y leds.

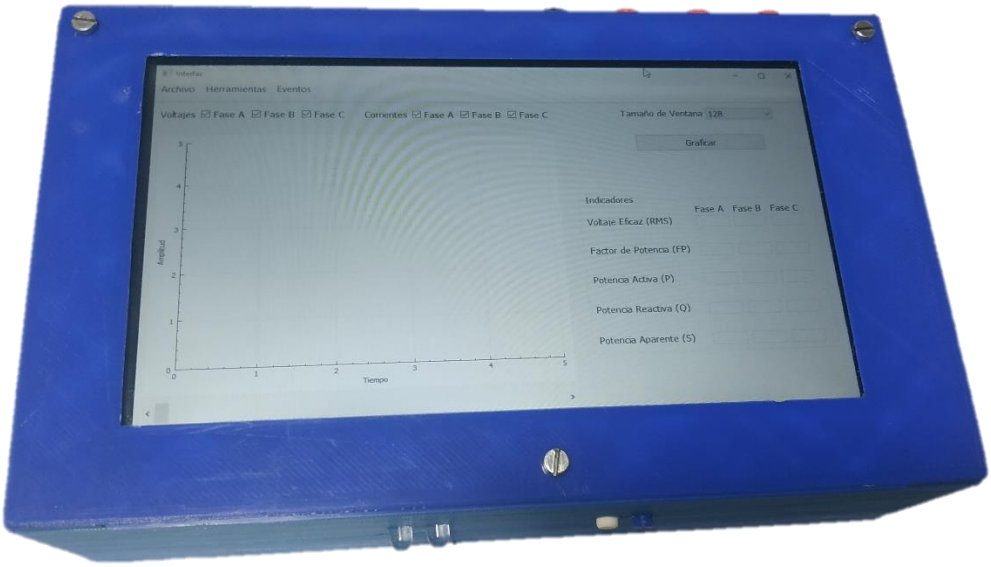
Una vez que terminamos de armar el dispositivo el método de encendido se realiza activando el interruptor (1) que tiene el dispositivo en la parte posterior para permitir la activación de la batería, a continuación, se presiona el botón azul (2) que se encuentra en la parte delantera del dispositivo para encender la SBC y la pantalla conectada



Al momento de presionar el botón de encendido y con el interruptor activado se debe observar que se enciendan los leds de la batería en el costado izquierdo que a su vez indican el porcentaje de carga de la misma y del costado derecho se debe apreciar un led, que indica que el computador esta encendido y funcionando además la primera pantalla del dispositivo muestra el icono de Lattepanda.

Al iniciar al sistema se destaca que el sistema operativo de la computadora es Windows 10 y al centro se encuentra el icono de la interfaz gráfica, la cual se abre al momento de apretar dicho botón.

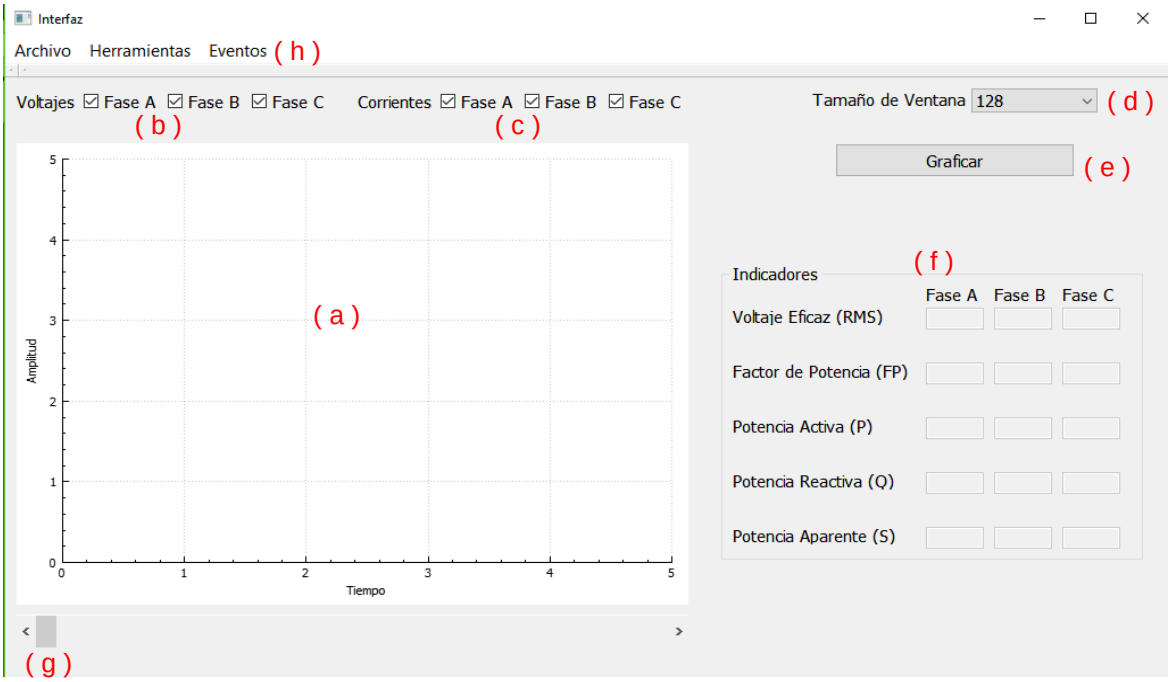
Al momento de apretar el botón se abre la interfaz gráfica y se muestra automáticamente como se observa en la imagen siguiente.



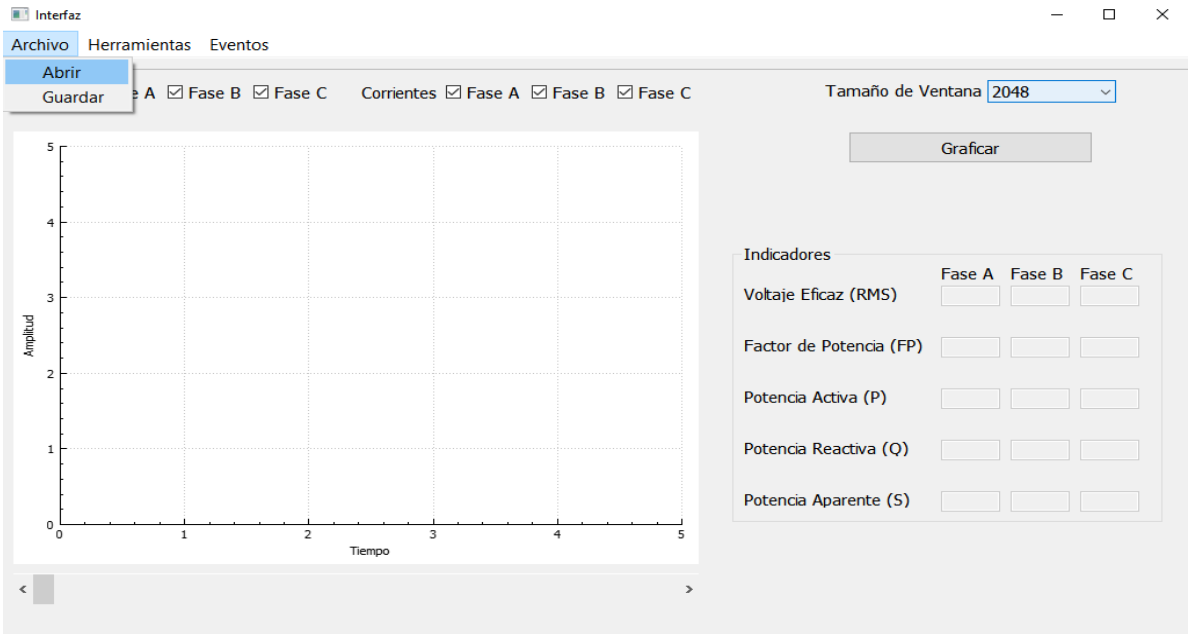
Y para apagar el equipo basta con el mismo procedimiento que se hace en cualquier sistema operativo Windows y se puede apagar el interruptor para conservar energía, ya que el ventilador continúa funcionando sin que se encienda la computadora.

Problemas comunes que se pueden suscitar durante los pasos anteriores es que el equipo se apague repentinamente, esto se debe a que el equipo Lattepanda tiene una programación de fábrica que en el momento que se sobre caliente o la administración de corriente este por debajo de los 2 A se pone en un estado similar a estar suspendida y no se reactiva hasta que se corrijan los problemas o la manera más rápida seria apagando y prendiendo el interruptor principal.

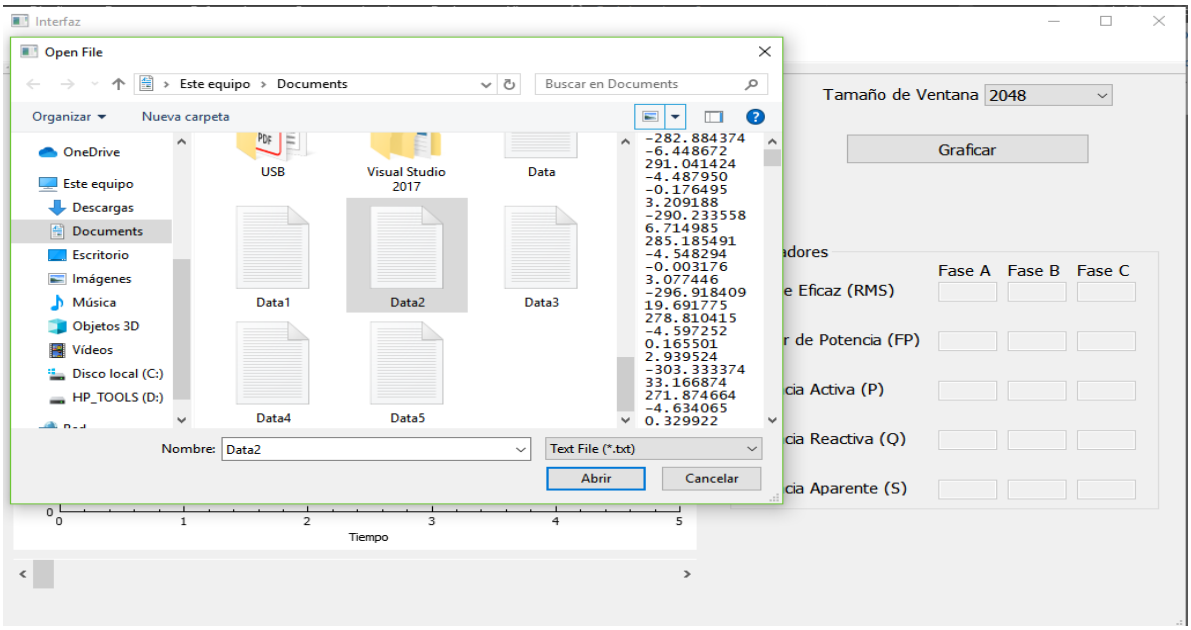
En el apartado de la interfaz graficase inicia con la pantalla principal de inicio que como se puede observar en la imagen siguiente cuenta con una ventana grafica (a), con cuadros seleccionables para ocultar o mostrar cada una de las fases de voltaje (b) y corriente (c), cuenta con una cuadro para seleccionar el tamaño de ventana (d) que va desde los 128 y hasta los 8192 puntos, un botón para graficar nuevamente la señal (e) si es que se ha cambiado el tamaño de ventana, con un apartado donde se muestran los valores de los indicadores para cada fase (f), una barra deslizable (g) para recorrer la gráfica de las señales he ir recalculando cada parámetro y por ultimo una barra menú (h) done se encuentra el botón de archivo, Herramientas y Eventos.



El primer paso es cargar alguna señal las cuales tienen el nombre de Data y se encuentran en formato de texto, para ello es necesario presionar en el menú archivo-> abrir. Como se muestra en la imagen siguiente.

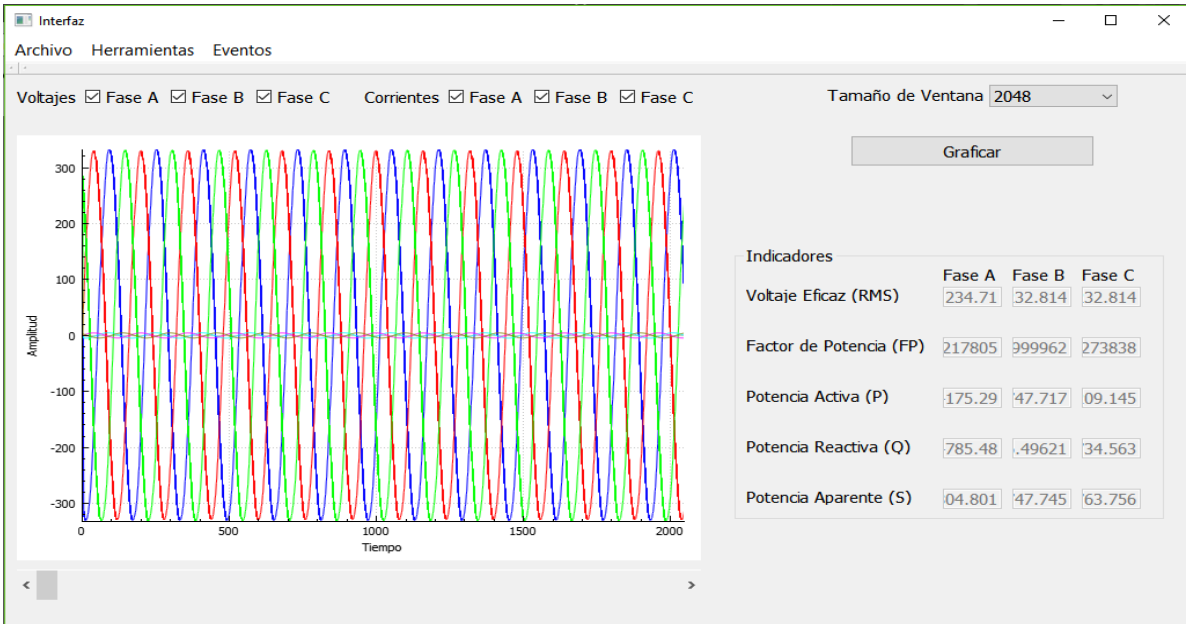


Al presionar el botón abrir se despliega la ventana de búsqueda de archivos y se debe seleccionar alguno de los documentos de texto con el nombre de Data que se encuentran en la carpeta de documentos.

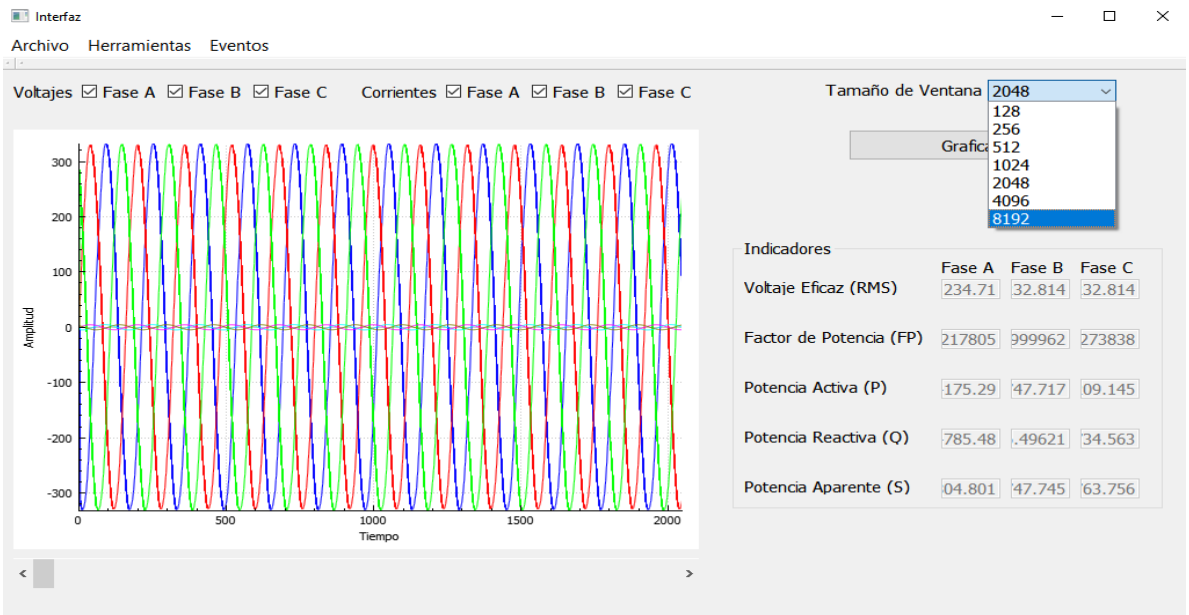


Al seleccionar alguno de los documentos de las señales, tarda aproximadamente dos minutos en cargar completamente las señales debido a la cantidad de datos de

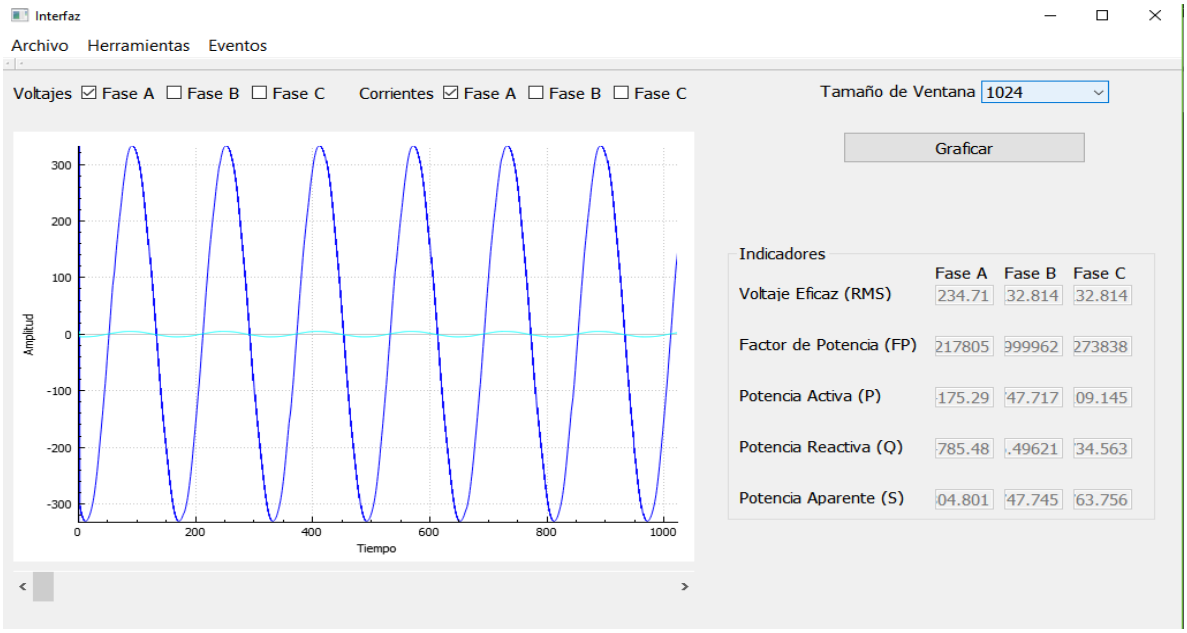
la señal, pero una vez termina de cargar la señal se muestra como la imagen siguiente



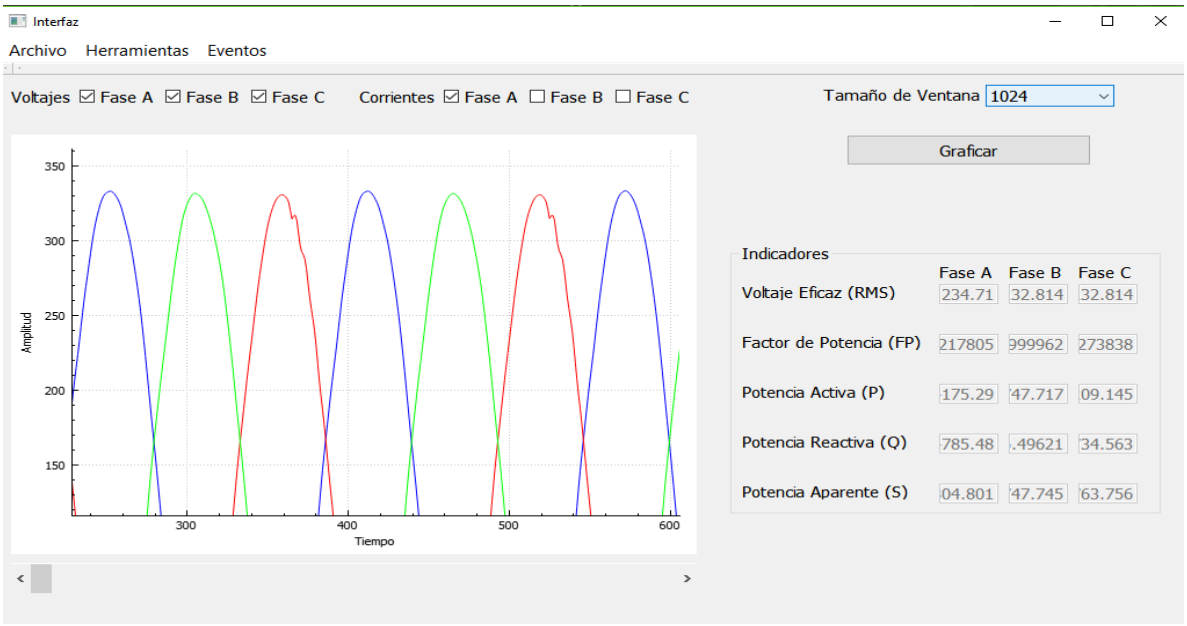
Si se desea puede cambiar el tamaño de la ventana para ello hay que desplegar donde dice tamaño de ventana y al seleccionar un valor se debe presionar el botón graficar.



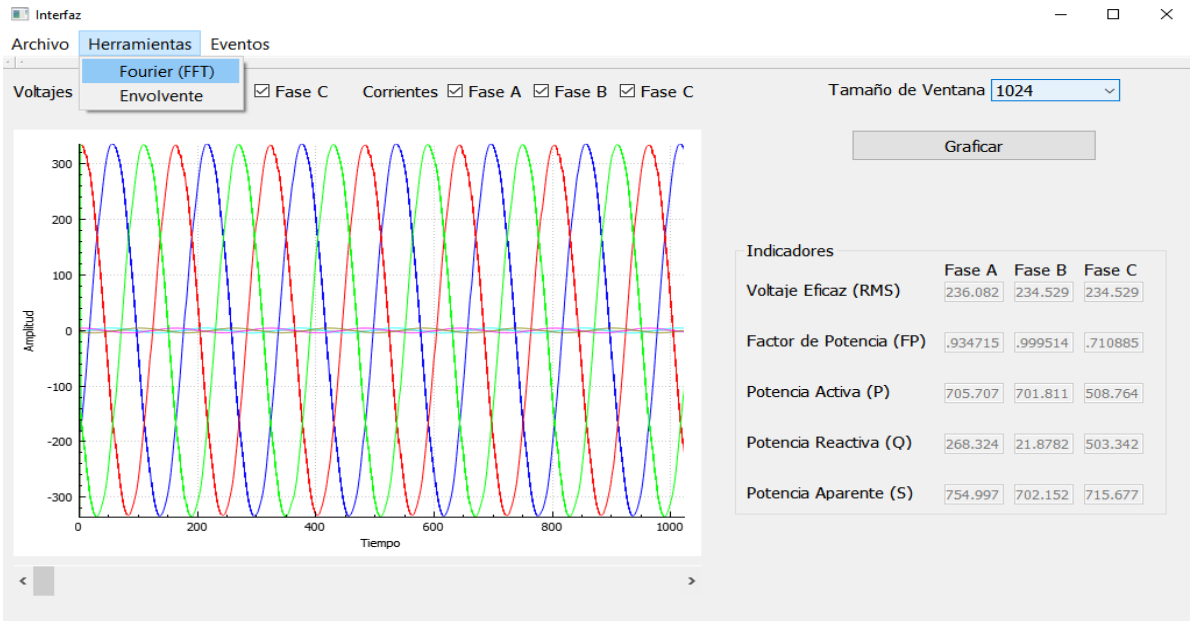
Ya que se ha cargado la señal es posible ocultarla si se desea utilizando el cuadro seleccionable de la señal que se desea ocultar y se muestra de la siguiente manera.



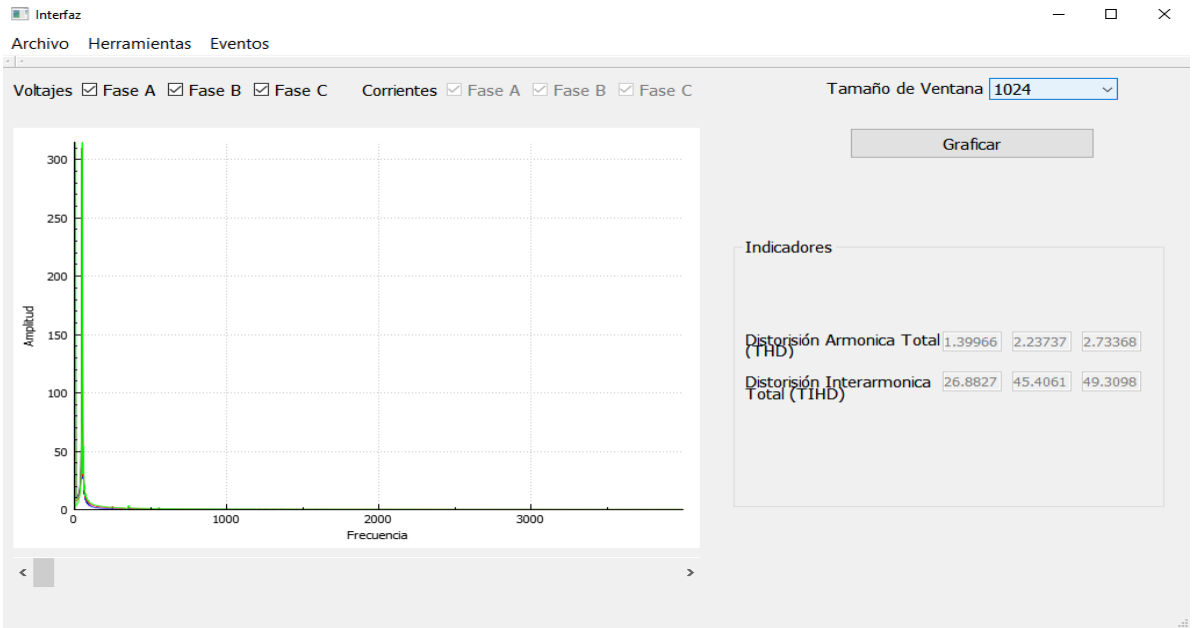
En cuanto en el apartado grafico es posible hacer zoom a las señales o moverlas para una mejor visualización, esto con los gestos comunes de una pantalla táctil.



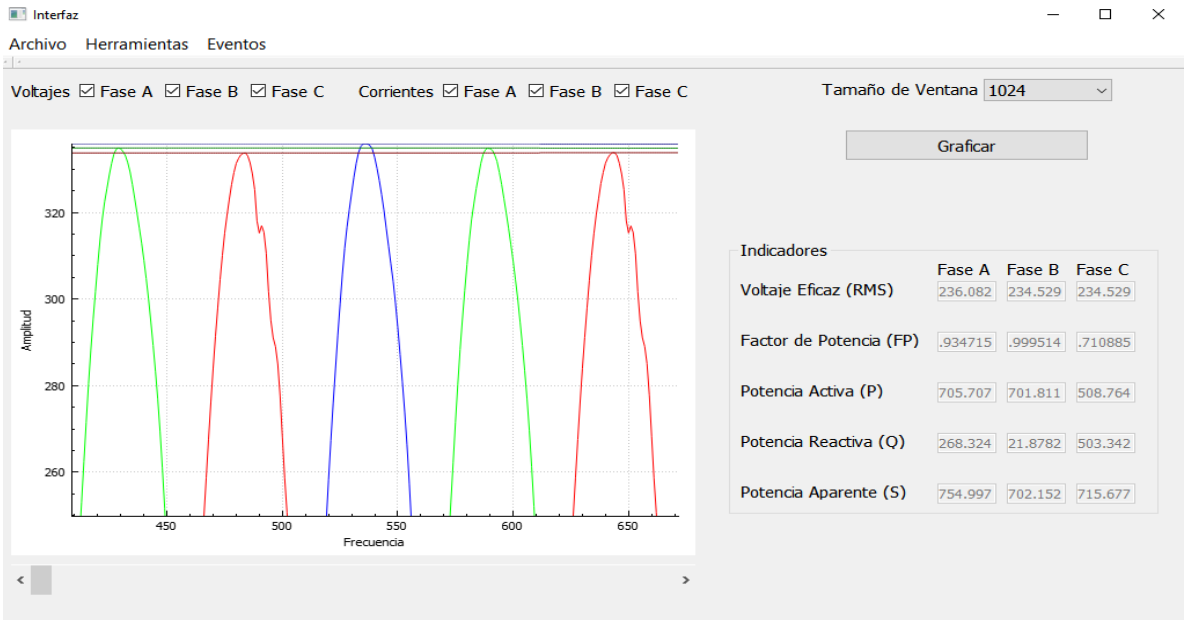
En el menú de herramientas se encuentran dos botones uno para la transformada rápida de Fourier FFT y el otro para la envolvente, como a continuación.



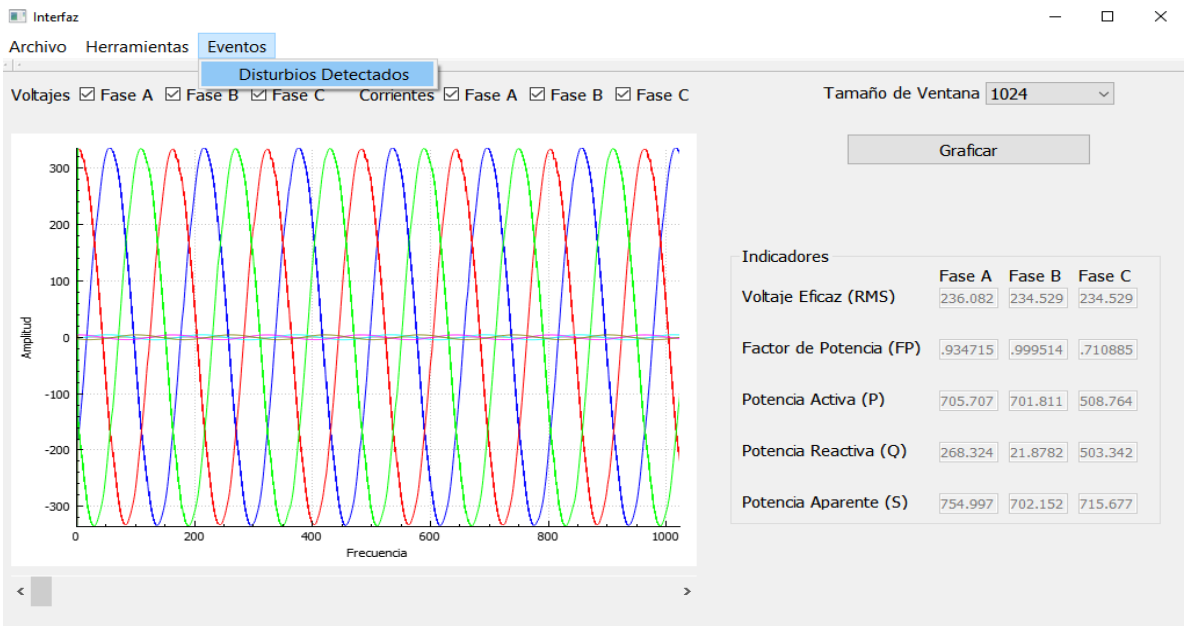
Al presionar el botón de FFT se calcula la transformada rápida de Fourier de igual forma para cierta ventana de datos y la barra deslizable sigue funcionando desplazándonos por toda la señal, en este punto los indicadores son remplazados por el valor de distorsión armónica total (THD) y el valor de distorsión interarmónica total (TIHD) para las tres fases de voltaje.



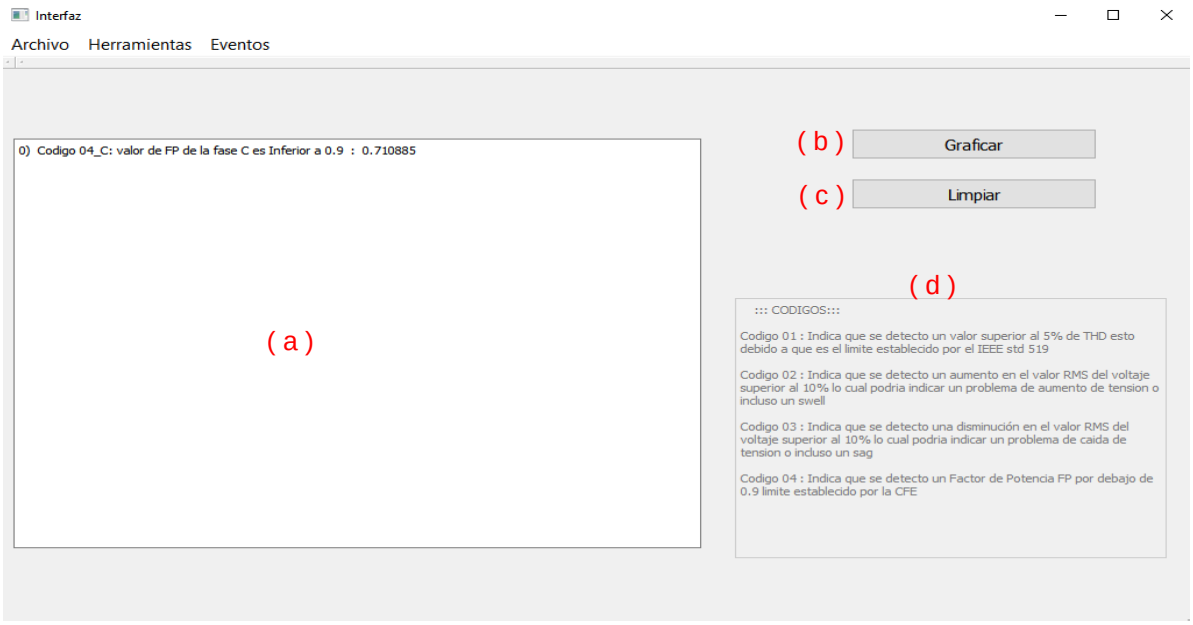
Para calcular la envolvente es necesario presionar el botón graficar para regresar a la gráfica en tiempo de la señal, y al presionar el botón envolvente se muestra de la siguiente manera, se aplicó un zoom para mejor visualización.



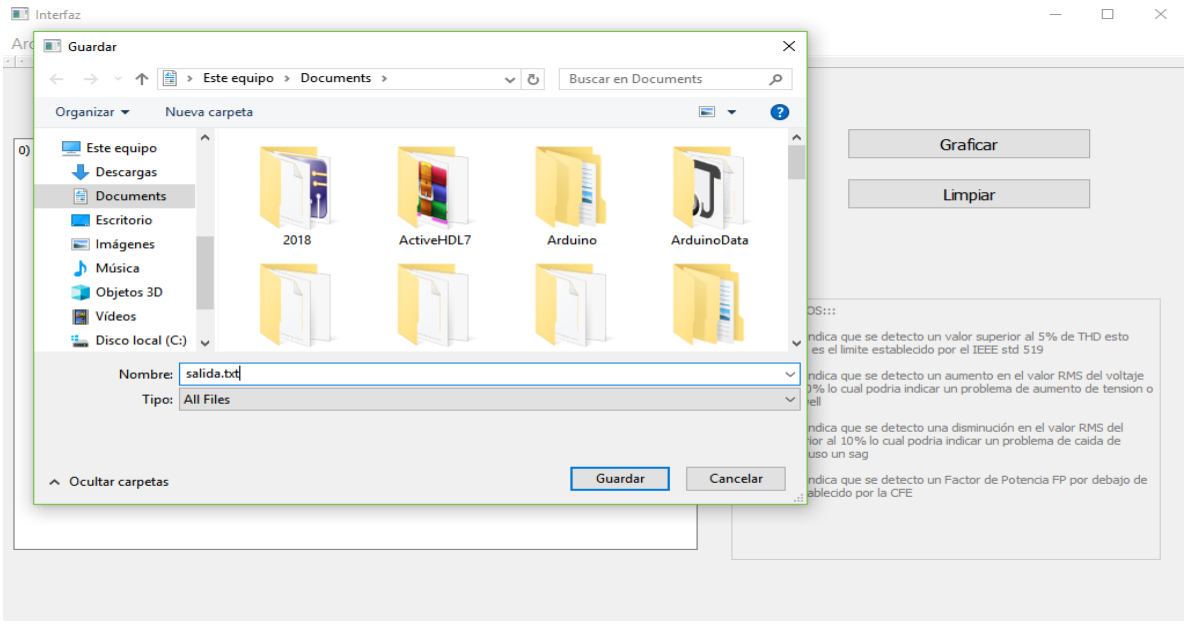
Para acceder al registro de eventos o disturbios ocurridos, se debe seleccionar del menú principal Eventos -> Disturbios Detectados.



Al momento de presionar el botón se muestra la pantalla de la imagen siguiente, donde cambia ligeramente la ventana, ahora se muestra un cuadro de texto (a) donde se van escribiendo los disturbios detectados, con un contador para el total de disturbios, el código de disturbio, la fase a la que aplica, y el valor motivo de su detección, se conserva el botón de graficas (b), se agrega el botón de limpiar (c), para borrar el registro de eventos detectados y un cuadro de texto (d) con una breve descripción de los códigos posibles y el motivo de su detección.

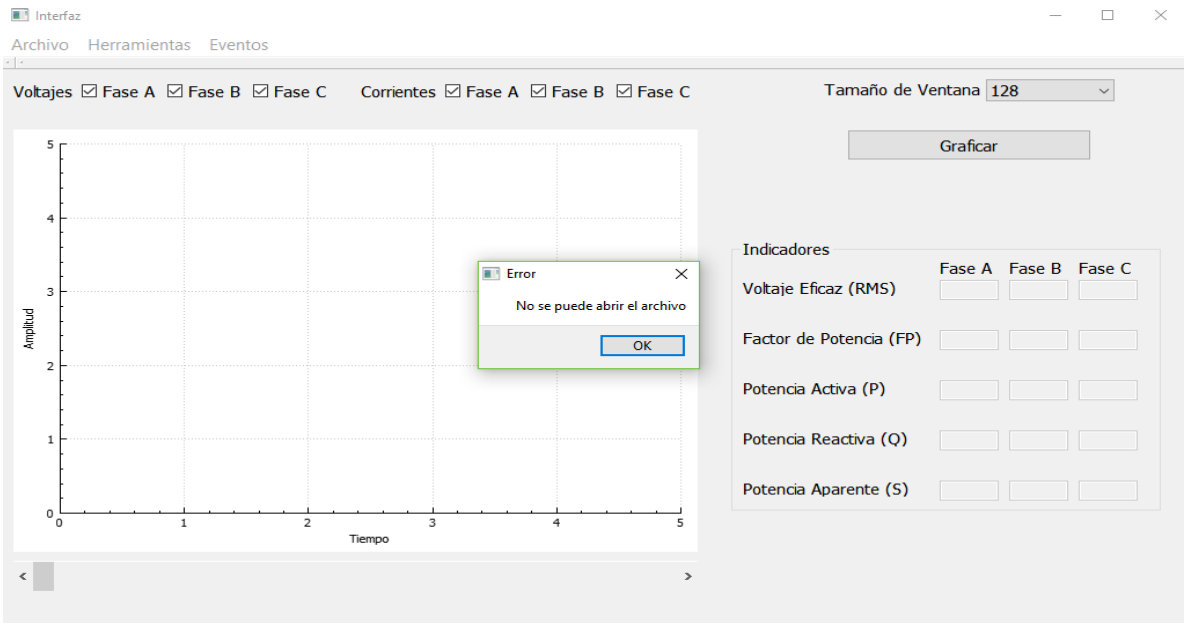


Si se desea guardar los eventos detectados, en el menú principal Archivo -> Guardar se abrirá un cuadro para otorgar un nombre y un formato de archivo que debe ser .txt, y seleccionar una ruta donde guardar el archivo.

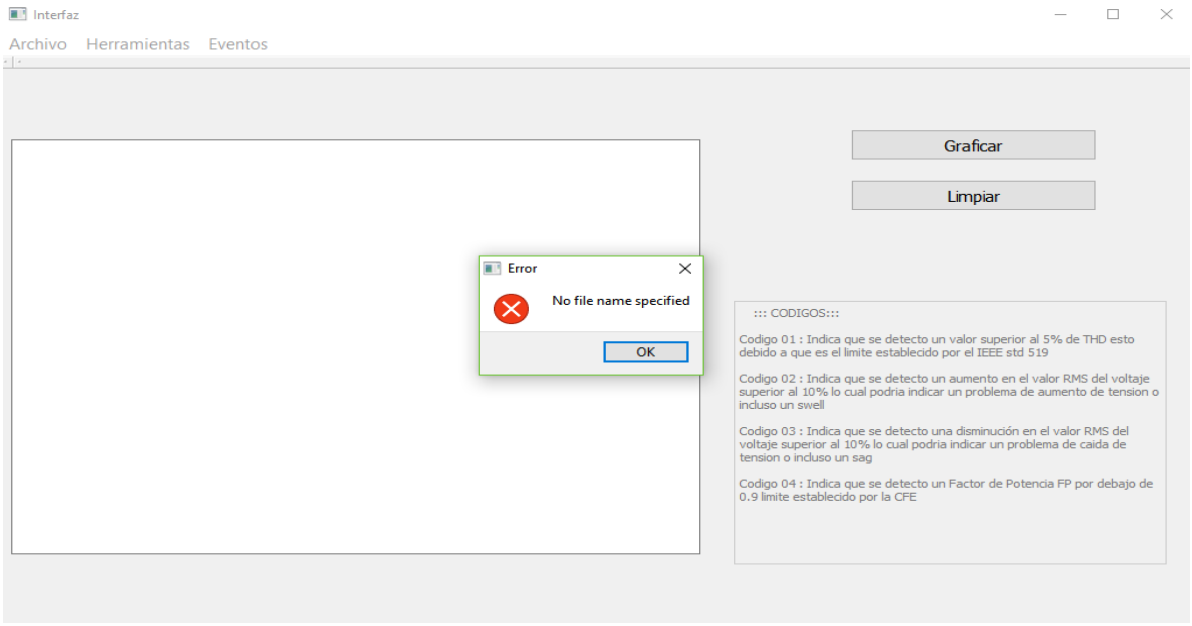


Por ultimo algunos errores que pueden ocurrir en la interfaz gráfica.

- No se seleccionó ningún archivo o no es válido a la hora de abrir la señal.



- No se estableció ningún nombre a la hora de intentar guardar los eventos.



Anexo 5: Código de la interfaz gráfica y Algoritmos

PQ-UAQ.pro 1

```
#-------------------------------------------------
#
# Project created by QtCreator 2018-11-15T13:08:33
#
#-------------------------------------------------

QT       += core gui printsupport

greaterThan(QT_MAJOR_VERSION, 4): QT += widgets

TARGET = PQ-UAQ
TEMPLATE = app

# The following define makes your compiler emit warnings if you use
# any feature of Qt which has been marked as deprecated (the exact warnings
# depend on your compiler). Please consult the documentation of the
# deprecated API in order to know how to port your code away from it.
DEFINES += QT_DEPRECATED_WARNINGS

# You can also make your code fail to compile if you use deprecated APIs.
# In order to do so, uncomment the following line.
# You can also select to disable deprecated APIs only up to a certain version of
Qt.
#DEFINES += QT_DISABLE_DEPRECATED_BEFORE=0x060000    # disables all the APIs
deprecated before Qt 6.0.0

CONFIG += c++11

SOURCES += \
        main.cpp \
        interfaz.cpp \
    qcustomplot.cpp

HEADERS += \
        interfaz.h \
    qcustomplot.h

FORMS += \
        interfaz.ui
```

```
FORMS += \
        interfaz.ui

# Default rules for deployment.
qnx: target.path = /tmp/$${TARGET}/bin
else: unix:!android: target.path = /opt/$${TARGET}/bin
!isEmpty(target.path): INSTALLS += target
```

interfaz.h 1

```
#ifndef INTERFAZ_H
#define INTERFAZ_H

#include <QMainWindow>

namespace Ui {
class Interfaz;
}

class Interfaz : public QMainWindow
{
    Q_OBJECT

public:
    explicit Interfaz(QWidget *parent = nullptr);
    ~Interfaz();

private slots:
    void on_actionAbrir_triggered();

    void on_Bgraf_clicked();

    void on_actionFourier_FFT_triggered();

    void on_Cfa_stateChanged(int arg1);

    void on_Cfb_stateChanged(int arg1);

    void on_Cfc_stateChanged(int arg1);

    void on_Ccfa_stateChanged(int arg1);

    void on_Ccfb_stateChanged(int arg1);

    void on_Ccfc_stateChanged(int arg1);

    void on_horizontalScrollBar_actionTriggered(int action);

    void on_actionDisturbios_Detectados_triggered();


  void on_thd_textChanged(const QString &arg1);

  void on_thd1_textChanged(const QString &arg1);

  void on_thd2_textChanged(const QString &arg1);

  void on_rms_textChanged(const QString &arg1);

  void on_rms1_textChanged(const QString &arg1);

  void on_rms2_textChanged(const QString &arg1);

  void on_fp_textChanged(const QString &arg1);
```

```
    void on_fp1_textChanged(const QString &arg1);

    void on_fp2_textChanged(const QString &arg1);

    void on_actionEnvolvente_triggered();

    void on_actionGuardar_triggered();

    void on_Blimp_clicked();

private:
    Ui::Interfaz *ui;
};

#endif // INTERFAZ_H
```

main.cpp 1

```
#include "interfaz.h"
#include <QApplication>

int main(int argc, char *argv[])
{
    QApplication a(argc, argv);
    Interfaz w;
    w.show();

    return a.exec();
}
```

interfaz.cpp 1

```
#include "interfaz.h"
#include "ui_interfaz.h"
#include "math.h"
#include <QMessageBox>
#include <QFile>
#include <QTextStream>
#include <QtGui>
#include <iostream>
QVector<double>
Data(8192),Data1(8192),Data2(8192),CDat(8192),CDat1(8192),CDat2(8192);
QVector<double> ap, ap1, ap2, gg(4096), gg1(4096), gg2(4096), freq(4096), ac, ac1,
ac2;
int cc=0;
Interfaz::Interfaz(QWidget *parent) :
QMainWindow(parent),
ui(new Ui::Interfaz)
{
ui->setupUi(this);
int i,n=0;
for (i=7;i<=13;i++){
n=pow(2.0,i);
ui->comboBox->addItem(QString::number(n));
}
```

```
ui->plot->xAxis->setLabel("Tiempo");
ui->plot->yAxis->setLabel("Amplitud");
ui->Lthd->setVisible(false);
ui->Lthd1->setVisible(false);
ui->Ltihd->setVisible(false);
ui->Ltihd1->setVisible(false);
ui->thd->setVisible(false);
ui->thd1->setVisible(false);
ui->thd2->setVisible(false);
ui->tihd->setVisible(false);
ui->tihd1->setVisible(false);
ui->tihd2->setVisible(false);
ui->cont->setVisible(false);
ui->textEdit->setVisible(false);
ui->cont2->setVisible(false);
ui->Blimp->setVisible(false);
ui->text->setVisible(false);
ui->text->setText(" ::: CODIGOS:::\n\n"
"Codigo 01 : Indica que se detecto un valor superior al 5%
de THD esto debido a que es el limite establecido por el IEEE std 519\n\n"
"Codigo 02 : Indica que se detecto un aumento en el valor
RMS del voltaje superior al 10% lo cual podria indicar un problema de aumento de
tension o incluso un swell\n\n"
"Codigo 03 : Indica que se detecto una disminución en el
valor RMS del voltaje superior al 10% lo cual podria indicar un problema de caida
interfaz.cpp 2
de tension o incluso un sag\n\n"
"Codigo 04 : Indica que se detecto un Factor de Potencia FP
por debajo de 0.9 limite establecido por la CFE");
}
Interfaz::~Interfaz()
{
delete ui;
}
void Interfaz::on_actionAbrir_triggered()
{
int i;
QVector<double> x(8192);
double d1, d2, d3, d4, d5, d6;
QString d=ui->comboBox->currentText();
//int f = d.toInt();
//ui->textBrowser->setText(d);
ui->plot->addGraph();
ui->plot->graph(0)->setPen(QPen(Qt::blue));
ui->plot->addGraph();
ui->plot->graph(1)->setPen(QPen(Qt::red));
ui->plot->addGraph();
ui->plot->graph(2)->setPen(QPen(Qt::green));
ui->plot->addGraph();
ui->plot->graph(3)->setPen(QPen(Qt::cyan));
ui->plot->addGraph();
ui->plot->graph(4)->setPen(QPen(Qt::magenta));
ui->plot->addGraph();
ui->plot->graph(5)->setPen(QPen(Qt::darkYellow));
ui->plot->graph(0)->data()->clear();
ui->plot->graph(1)->data()->clear();
```

```
ui->plot->graph(2)->data()->clear();
ui->plot->graph(3)->data()->clear();
ui->plot->graph(4)->data()->clear();
ui->plot->graph(5)->data()->clear();
ui->Ccfa->setEnabled(true);
ui->Ccfb->setEnabled(true);
ui->Ccfc->setEnabled(true);
//ui->horizontalScrollBar->setVisible(true);
QString filename = QFileDialog::getOpenFileName(this, tr("Open File"),"C://
Users//USER//Documents//", "Text File (*.txt)");
QFile fr(filename); //abre el archivo en donde se encuantran los datos
if(!fr.open(QIODevice::ReadOnly))
{
QMessageBox::about(this,"Error","No se puede abrir el archivo");
interfaz.cpp 3
}
else{
QTextStream in(&fr);
//ui->textBrowser->setText(in.readAll());
//for(i=0;i<d.toInt();++i){
while(!in.atEnd()){
in.readLine();
in >> d1 >> d2 >> d3 >> d4 >> d5 >> d6;
if(in.status() == QTextStream::Ok){
ap.append(d1);
ap1.append(d2);
ap2.append(d3);
ac.append(d4);
ac1.append(d5);
ac2.append(d6);
//std::cout << ap[i] << " " << i << " " << d1 << std::endl;
}
else
break;
}
//int j=0;
for(i=0; i<d.toInt()-1; ++i){
Data[i]=ap[i];
Data1[i]=ap1[i];
Data2[i]=ap2[i];
CDat[i]=ac[i];
CDat1[i]=ac1[i];
CDat2[i]=ac2[i];
x[i]=i;
//std::cout << Data[i] << " " << i << " " << ap[i] << std::endl;
}
}
// for(l=0;l=H;l++){
// freq[l+1]=l*fs/N;
// }
int Fs=8000;
int N=8000;
double ven=Fs/5;
double num=N/ven;
QVector <double> v(8192);
QVector <double> v1(8192);
```

```
QVector <double> v2(8192);
double mult,mult1,mult2,ang=0,ang1=0,ang2=0;
QVector <double> Xrms(10);
QVector <double> Xrms1(10);
QVector <double> Xrms2(10);
QVector <double> max(10);
QVector <double> max1(10);
QVector <double> max2(10);
QVector <double> Yrms(10);
QVector <double> Yrms1(10);
QVector <double> Yrms2(10);
interfaz.cpp 4
QVector <double> maxy(10);
QVector <double> maxy1(10);
QVector <double> maxy2(10);
QVector <double> S(10);
QVector <double> S1(10);
QVector <double> S2(10);
QVector <double> P(10);
QVector <double> P1(10);
QVector <double> P2(10);
QVector <double> Q(10);
QVector <double> Q1(10);
QVector <double> Q2(10);
QVector <double> FP(10);
QVector <double> FP1(10);
QVector <double> FP2(10);
QVector <double> Palt(10);
QVector <double> Palt1(10);
QVector <double> Palt2(10);
double p0=0,p1=0,p01=0,p11=0,p02=0,p12=0;
for(int i=0;i<=num-2;i++){
Palt[i+1]=0;
Palt1[i+1]=0;
Palt2[i+1]=0;
for(int j=ven*i;j<=ven*(i+1);j++){
Xrms[i+1]=Xrms[i+1]+pow(Data[j+1],2);//*datV[j+1]);
Xrms1[i+1]=Xrms1[i+1]+pow(Data1[j+1],2);
Xrms2[i+1]=Xrms2[i+1]+pow(Data2[j+1],2);
//qDebug() << Xrms;
//printf("j: %d\n",j);
if(Data[j+1]>max[i+1]){
max[i+1]=Data[j+1];
}
if(Data1[j+1]>max1[i+1]){
max1[i+1]=Data1[j+1];
}
if(Data2[j+1]>max2[i+1]){
max2[i+1]=Data2[j+1];
}
if(Data[j+1]<0 && p0==0){
p0=j-(1/(Data[j+1]-Data[j]))*Data[j];
}
if(Data1[j+1]<0 && p01==0){
p01=j-(1/(Data1[j+1]-Data1[j]))*Data1[j];
}
```

```
if(Data2[j+1]<0 && p02==0){
p02=j-(1/(Data2[j+1]-Data2[j]))*Data2[j];
}
Yrms[i+1]=Yrms[i+1]+CDat[j+1]*CDat[j+1];
Yrms1[i+1]=Yrms1[i+1]+CDat1[j+1]*CDat1[j+1];
Yrms2[i+1]=Yrms2[i+1]+CDat2[j+1]*CDat2[j+1];
if(CDat[j+1]>maxy[i+1]){
interfaz.cpp 5
maxy[i+1]=CDat[j+1];
}
if(CDat1[j+1]>maxy1[i+1]){
maxy1[i+1]=CDat1[j+1];
}
if(CDat2[j+1]>maxy2[i+1]){
maxy2[i+1]=CDat2[j+1];
}
mult=Data[j+1]*CDat[j+1];
mult1=Data1[j+1]*CDat1[j+1];
mult2=Data2[j+1]*CDat2[j+1];
Palt[i+1]=Palt[i+1]+mult;
Palt1[i+1]=Palt1[i+1]+mult1;
Palt2[i+1]=Palt2[i+1]+mult2;
v[j+1]=Data[j+1];
v1[j+1]=Data1[j+1];
v2[j+1]=Data2[j+1];
if(CDat[j+1]<0&&p1==0){
p1=j-(1/(CDat[j+1]-CDat[j]))*CDat[j];
}
if(CDat1[j+1]<0&&p11==0){
p11=j-(1/(CDat1[j+1]-CDat1[j]))*CDat1[j];
}
if(CDat2[j+1]<0&&p12==0){
p12=j-(1/(CDat2[j+1]-CDat2[j]))*CDat2[j];
}
}
Palt[i+1]=Palt[i+1]/ven;
Palt1[i+1]=Palt1[i+1]/ven;
Palt2[i+1]=Palt2[i+1]/ven;
ang=abs(p1-p0)*(90/p0);
ang1=abs(p11-p01)*(210/p01);
ang2=abs(p12-p02)*(330/p02);
}
// thd=0.117;
double c,c1,c2;
for(int i=0;i<=num-2;i++){
//printf("Indice i: %f Voltaje RMS : %f\n",ven,Xrms[i+1]);
Xrms[i+1]=sqrt(Xrms[i+1]*(1/ven));
Xrms1[i+1]=sqrt(Xrms1[i+1]*(1/ven));
Xrms2[i+1]=sqrt(Xrms2[i+1]*(1/ven));
Yrms[i+1]=sqrt(Yrms[i+1]*(1/ven));
Yrms1[i+1]=sqrt(Yrms1[i+1]*(1/ven));
Yrms2[i+1]=sqrt(Yrms2[i+1]*(1/ven));
c=abs(maxy[i+1])/Yrms[i+1];
c1=abs(maxy1[i+1])/Yrms1[i+1];
c2=abs(maxy2[i+1])/Yrms2[i+1];
S[i+1]=Xrms[i+1]*Yrms[i+1];
```

```
S1[i+1]=Xrms1[i+1]*Yrms1[i+1];
S2[i+1]=Xrms2[i+1]*Yrms2[i+1];
P[i+1]=S[i+1]*cos(ang*(M_PI/180));
P1[i+1]=S1[i+1]*cos(ang1*(M_PI/180));
P2[i+1]=S2[i+1]*cos(ang2*(M_PI/180));
interfaz.cpp 6
Q[i+1]=S[i+1]*sin(ang*(M_PI/180));
Q1[i+1]=S1[i+1]*sin(ang1*(M_PI/180));
Q2[i+1]=S2[i+1]*sin(ang2*(M_PI/180));
FP[i+1]=cos(ang*(M_PI/180));
FP1[i+1]=cos(ang1*(M_PI/180));
FP2[i+1]=cos(ang2*(M_PI/180));
}
ui->rms->setText(QString::number(Xrms[1]));
ui->rms1->setText(QString::number(Xrms1[1]));
ui->rms2->setText(QString::number(Xrms1[1]));
ui->fp->setText(QString::number(FP[1]));
ui->fp1->setText(QString::number(FP1[1]));
ui->fp2->setText(QString::number(FP2[1]));
ui->Pp->setText(QString::number(P[1]));
ui->Pp1->setText(QString::number(P1[1]));
ui->Pp2->setText(QString::number(P2[1]));
ui->Pq->setText(QString::number(Q[1]));
ui->Pq1->setText(QString::number(Q1[1]));
ui->Pq2->setText(QString::number(Q2[1]));
ui->Ps->setText(QString::number(S[1]));
ui->Ps1->setText(QString::number(S1[1]));
ui->Ps2->setText(QString::number(S2[1]));
if(ui->Cfa->isChecked()){
ui->plot->graph(0)->setData(x,Data);
}
if(ui->Cfb->isChecked()){
ui->plot->graph(1)->setData(x,Data1);
}
if(ui->Cfc->isChecked()){
ui->plot->graph(2)->setData(x,Data2);
}
if(ui->Ccfa->isChecked()){
ui->plot->graph(3)->setData(x,CDat);
}
if(ui->Ccfb->isChecked()){
ui->plot->graph(4)->setData(x,CDat1);
}
if(ui->Ccfc->isChecked()){
ui->plot->graph(5)->setData(x,CDat2);
}
ui->plot->graph(0)->rescaleAxes();
ui->plot->graph(1)->rescaleAxes();
ui->plot->graph(2)->rescaleAxes();
//ui->plot->graph(3)->rescaleAxes();
//ui->plot->graph(4)->rescaleAxes();
//ui->plot->graph(5)->rescaleAxes();
ui->plot->setInteractions(QCP::iRangeDrag | QCP::iRangeZoom |
QCP::iSelectPlottables);
ui->plot->replot();
interfaz.cpp 7
```

```
ui->cont->setText("1");
}
void Interfaz::on_Bgraf_clicked()
{
ui->plot->addGraph();
ui->plot->graph(0)->setPen(QPen(Qt::blue));
ui->plot->addGraph();
ui->plot->graph(1)->setPen(QPen(Qt::red));
ui->plot->addGraph();
ui->plot->graph(2)->setPen(QPen(Qt::green));
ui->plot->addGraph();
ui->plot->graph(3)->setPen(QPen(Qt::cyan));
ui->plot->addGraph();
ui->plot->graph(4)->setPen(QPen(Qt::magenta));
ui->plot->addGraph();
ui->plot->graph(5)->setPen(QPen(Qt::darkYellow));
ui->plot->graph(0)->data()->clear();
ui->plot->graph(1)->data()->clear();
ui->plot->graph(2)->data()->clear();
ui->plot->graph(3)->data()->clear();
ui->plot->graph(4)->data()->clear();
ui->plot->graph(5)->data()->clear();
ui->plot->graph(6)->data()->clear();
ui->plot->graph(7)->data()->clear();
ui->plot->graph(8)->data()->clear();
ui->text->setVisible(false);
ui->Ccfa->setEnabled(true);
ui->Ccfb->setEnabled(true);
ui->Ccfc->setEnabled(true);
ui->comboBox->setVisible(true);
ui->label->setVisible(true);
ui->plot->setVisible(true);
ui->textEdit->setVisible(false);
ui->Blimp->setVisible(false);
ui->Lthd->setVisible(false);
ui->Lthd1->setVisible(false);
ui->Ltihd->setVisible(false);
ui->Ltihd1->setVisible(false);
ui->thd->setVisible(false);
ui->thd1->setVisible(false);
ui->thd2->setVisible(false);
ui->tihd->setVisible(false);
ui->tihd1->setVisible(false);
ui->tihd2->setVisible(false);
ui->LA->setVisible(true);
ui->LB->setVisible(true);
ui->LC->setVisible(true);
ui->Lrms->setVisible(true);
ui->Lfp->setVisible(true);
ui->Lpp->setVisible(true);
ui->Lpq->setVisible(true);
ui->Lps->setVisible(true);
interfaz.cpp 8
ui->rms->setVisible(true);
ui->rms1->setVisible(true);
ui->rms2->setVisible(true);
```

```
ui->fp->setVisible(true);
ui->fp1->setVisible(true);
ui->fp2->setVisible(true);
ui->Pp->setVisible(true);
ui->Pp1->setVisible(true);
ui->Pp2->setVisible(true);
ui->Pq->setVisible(true);
ui->Pq1->setVisible(true);
ui->Pq2->setVisible(true);
ui->Ps->setVisible(true);
ui->Ps1->setVisible(true);
ui->Ps2->setVisible(true);
ui->plot->setVisible(true);
ui->horizontalScrollBar->setVisible(true);
ui->groupBox->setVisible(true);
ui->label->setVisible(true);
ui->label_2->setVisible(true);
ui->Cfa->setVisible(true);
ui->Cfc->setVisible(true);
ui->Cfb->setVisible(true);
ui->comboBox->setVisible(true);
//ui->Bgraf->setGeometry(690,40,181,23);
//ui->horizontalScrollBar->setVisible(true);
ui->cont->setText("1");
ui->cont2->setText("0");
QString d=ui->comboBox->currentText();
QVector<double> x(8192);
for(int i=0; i<d.toInt(); ++i){
Data[i]=ap[i];
Data1[i]=ap1[i];
Data2[i]=ap2[i];
CDat[i]=ac[i];
CDat1[i]=ac1[i];
CDat2[i]=ac2[i];
x[i]=i;
}
//qDebug() << ap.length();
int Fs=8000;
int N=8000;
double ven=Fs/5;
double num=N/ven;
QVector <double> v(8192);
QVector <double> v1(8192);
QVector <double> v2(8192);
double mult,mult1,mult2,ang=0,ang1=0,ang2=0;
QVector <double> Xrms(10);
interfaz.cpp 9
QVector <double> Xrms1(10);
QVector <double> Xrms2(10);
QVector <double> max(10);
QVector <double> max1(10);
QVector <double> max2(10);
QVector <double> Yrms(10);
QVector <double> Yrms1(10);
QVector <double> Yrms2(10);
QVector <double> maxy(10);
```

```
QVector <double> maxy1(10);
QVector <double> maxy2(10);
QVector <double> S(10);
QVector <double> S1(10);
QVector <double> S2(10);
QVector <double> P(10);
QVector <double> P1(10);
QVector <double> P2(10);
QVector <double> Q(10);
QVector <double> Q1(10);
QVector <double> Q2(10);
QVector <double> FP(10);
QVector <double> FP1(10);
QVector <double> FP2(10);
QVector <double> Palt(10);
QVector <double> Palt1(10);
QVector <double> Palt2(10);
double p0=0,p1=0,p01=0,p11=0,p02=0,p12=0;
for(int i=0;i<=num-2;i++){
Palt[i+1]=0;
Palt1[i+1]=0;
Palt2[i+1]=0;
for(int j=ven*i;j<=ven*(i+1);j++){
Xrms[i+1]=Xrms[i+1]+pow(Data[j+1],2);//*datV[j+1]);
Xrms1[i+1]=Xrms1[i+1]+pow(Data1[j+1],2);
Xrms2[i+1]=Xrms2[i+1]+pow(Data2[j+1],2);
//qDebug() << Xrms;
//printf("j: %d\n",j);
if(Data[j+1]>max[i+1]){
max[i+1]=Data[j+1];
}
if(Data1[j+1]>max1[i+1]){
max1[i+1]=Data1[j+1];
}
if(Data2[j+1]>max2[i+1]){
max2[i+1]=Data2[j+1];
}
if(Data[j+1]<0 && p0==0){
p0=j-(1/(Data[j+1]-Data[j]))*Data[j];
}
if(Data1[j+1]<0 && p01==0){
p01=j-(1/(Data1[j+1]-Data1[j]))*Data1[j];
interfaz.cpp 10
}
if(Data2[j+1]<0 && p02==0){
p02=j-(1/(Data2[j+1]-Data2[j]))*Data2[j];
}
Yrms[i+1]=Yrms[i+1]+CDat[j+1]*CDat[j+1];
Yrms1[i+1]=Yrms1[i+1]+CDat1[j+1]*CDat1[j+1];
Yrms2[i+1]=Yrms2[i+1]+CDat2[j+1]*CDat2[j+1];
if(CDat[j+1]>maxy[i+1]){
maxy[i+1]=CDat[j+1];
}
if(CDat1[j+1]>maxy1[i+1]){
maxy1[i+1]=CDat1[j+1];
}
```

```
if(CDat2[j+1]>maxy2[i+1]){
maxy2[i+1]=CDat2[j+1];
}
mult=Data[j+1]*CDat[j+1];
mult1=Data1[j+1]*CDat1[j+1];
mult2=Data2[j+1]*CDat2[j+1];
Palt[i+1]=Palt[i+1]+mult;
Palt1[i+1]=Palt1[i+1]+mult1;
Palt2[i+1]=Palt2[i+1]+mult2;
v[j+1]=Data[j+1];
v1[j+1]=Data1[j+1];
v2[j+1]=Data2[j+1];
if(CDat[j+1]<0&&p1==0){
p1=j-(1/(CDat[j+1]-CDat[j]))*CDat[j];
}
if(CDat1[j+1]<0&&p11==0){
p11=j-(1/(CDat1[j+1]-CDat1[j]))*CDat1[j];
}
if(CDat2[j+1]<0&&p12==0){
p12=j-(1/(CDat2[j+1]-CDat2[j]))*CDat2[j];
}
}
Palt[i+1]=Palt[i+1]/ven;
Palt1[i+1]=Palt1[i+1]/ven;
Palt2[i+1]=Palt2[i+1]/ven;
ang=abs(p1-p0)*(90/p0);
ang1=abs(p11-p01)*(210/p01);
ang2=abs(p12-p02)*(330/p02);
}
// thd=0.117;
double c,c1,c2;
for(int i=0;i<=num-2;i++){
//printf("Indice i: %f Voltaje RMS : %f\n",ven,Xrms[i+1]);
Xrms[i+1]=sqrt(Xrms[i+1]*(1/ven));
Xrms1[i+1]=sqrt(Xrms1[i+1]*(1/ven));
Xrms2[i+1]=sqrt(Xrms2[i+1]*(1/ven));
Yrms[i+1]=sqrt(Yrms[i+1]*(1/ven));
Yrms1[i+1]=sqrt(Yrms1[i+1]*(1/ven));
Yrms2[i+1]=sqrt(Yrms2[i+1]*(1/ven));
c=abs(maxy[i+1])/Yrms[i+1];
interfaz.cpp 11
c1=abs(maxy1[i+1])/Yrms1[i+1];
c2=abs(maxy2[i+1])/Yrms2[i+1];
S[i+1]=Xrms[i+1]*Yrms[i+1];
S1[i+1]=Xrms1[i+1]*Yrms1[i+1];
S2[i+1]=Xrms2[i+1]*Yrms2[i+1];
P[i+1]=S[i+1]*cos(ang*(M_PI/180));
P1[i+1]=S1[i+1]*cos(ang1*(M_PI/180));
P2[i+1]=S2[i+1]*cos(ang2*(M_PI/180));
Q[i+1]=S[i+1]*sin(ang*(M_PI/180));
Q1[i+1]=S1[i+1]*sin(ang1*(M_PI/180));
Q2[i+1]=S2[i+1]*sin(ang2*(M_PI/180));
FP[i+1]=cos(ang*(M_PI/180));
FP1[i+1]=cos(ang1*(M_PI/180));
FP2[i+1]=cos(ang2*(M_PI/180));
}
```

```
ui->rms->setText(QString::number(Xrms[1]));
ui->rms1->setText(QString::number(Xrms1[1]));
ui->rms2->setText(QString::number(Xrms1[1]));
ui->fp->setText(QString::number(FP[1]));
ui->fp1->setText(QString::number(FP1[1]));
ui->fp2->setText(QString::number(FP2[1]));
ui->Pp->setText(QString::number(P[1]));
ui->Pp1->setText(QString::number(P1[1]));
ui->Pp2->setText(QString::number(P2[1]));
ui->Pq->setText(QString::number(Q[1]));
ui->Pq1->setText(QString::number(Q1[1]));
ui->Pq2->setText(QString::number(Q2[1]));
ui->Ps->setText(QString::number(S[1]));
ui->Ps1->setText(QString::number(S1[1]));
ui->Ps2->setText(QString::number(S2[1]));
if(ui->Cfa->isChecked()){
ui->plot->graph(0)->setData(x,Data);}
if(ui->Cfb->isChecked()){
ui->plot->graph(1)->setData(x,Data1);}
if(ui->Cfc->isChecked()){
ui->plot->graph(2)->setData(x,Data2);}
if(ui->Ccfa->isChecked()){
ui->plot->graph(3)->setData(x,CDat);
ui->plot->graph(3)->rescaleAxes();}
if(ui->Ccfb->isChecked()){
ui->plot->graph(4)->setData(x,CDat1);
ui->plot->graph(4)->rescaleAxes();}
if(ui->Ccfc->isChecked()){
ui->plot->graph(5)->setData(x,CDat2);
ui->plot->graph(5)->rescaleAxes();}
// let the ranges scale themselves so graph 0 fits perfectly in the visible
area:
ui->plot->graph(0)->rescaleAxes();
ui->plot->graph(1)->rescaleAxes();
ui->plot->graph(2)->rescaleAxes();
interfaz.cpp 12
// same thing for graph 1, but only enlarge ranges (in case graph 1 is smaller
than graph 0):
// Note: we could have also just called customPlot->rescaleAxes(); instead
// Allow user to drag axis ranges with mouse, zoom with mouse wheel and select
graphs by clicking:
ui->plot->setInteractions(QCP::iRangeDrag | QCP::iRangeZoom |
QCP::iSelectPlottables);
ui->plot->replot();
}
int bin(int num,int bit) //conversion de decimal a binario
{
int c,k,i;
int n=0;
int b[100];
for (c = bit-1; c >= 0; c--)
{
k = num >> c;
if (k & 1)
b[c]=1;
else
```

```
b[c]=0;
}
for(i=0;i<bit;++i){
n+=(pow(2.0,bit-1-i)*b[i]); ///convierte el valor binario a decimal
//printf("%d",b[i]);
//std::cout << i << " " << b[i] << std::endl;
}
//printf("\n");
//printf("num=%d, n=%d\n",num,n);
return n;
}
void Interfaz::on_actionFourier_FFT_triggered()
{
ui->plot->xAxis->setLabel("Frecuencia");
ui->plot->yAxis->setLabel("Amplitud");
ui->plot->graph(0)->data()->clear();
ui->plot->graph(1)->data()->clear();
ui->plot->graph(2)->data()->clear();
ui->plot->graph(3)->data()->clear();
ui->plot->graph(4)->data()->clear();
ui->plot->graph(5)->data()->clear();
ui->plot->replot();
ui->cont->setText("2");
ui->text->setVisible(false);
ui->Ccfa->setEnabled(false);
ui->Ccfb->setEnabled(false);
ui->Ccfc->setEnabled(false);
interfaz.cpp 13
ui->comboBox->setVisible(false);
ui->label->setVisible(false);
ui->textEdit->setVisible(false);
//ui->horizontalScrollBar->setVisible(false);
ui->Blimp->setVisible(false);
ui->Lthd->setVisible(true);
ui->Lthd1->setVisible(true);
ui->Ltihd->setVisible(true);
ui->Ltihd1->setVisible(true);
ui->thd->setVisible(true);
ui->thd1->setVisible(true);
ui->thd2->setVisible(true);
ui->tihd->setVisible(true);
ui->tihd1->setVisible(true);
ui->tihd2->setVisible(true);
ui->Lrms->setVisible(false);
ui->Lfp->setVisible(false);
ui->Lpp->setVisible(false);
ui->Lpq->setVisible(false);
ui->Lps->setVisible(false);
ui->LA->setVisible(false);
ui->LB->setVisible(false);
ui->LC->setVisible(false);
ui->rms->setVisible(false);
ui->rms1->setVisible(false);
ui->rms2->setVisible(false);
ui->fp->setVisible(false);
ui->fp1->setVisible(false);
```

```
ui->fp2->setVisible(false);
ui->Pp->setVisible(false);
ui->Pp1->setVisible(false);
ui->Pp2->setVisible(false);
ui->Pq->setVisible(false);
ui->Pq1->setVisible(false);
ui->Pq2->setVisible(false);
ui->Ps->setVisible(false);
ui->Ps1->setVisible(false);
ui->Ps2->setVisible(false);
ui->plot->setVisible(true);
ui->horizontalScrollBar->setVisible(true);
ui->groupBox->setVisible(true);
ui->label->setVisible(true);
ui->label_2->setVisible(true);
ui->Cfa->setVisible(true);
ui->Cfc->setVisible(true);
ui->Cfb->setVisible(true);
ui->comboBox->setVisible(true);
//ui->Bgraf->setGeometry(690,40,181,23);
ui->cont2->setText("0");
const size_t s = 3000;
double init[s];
interfaz.cpp 14
double init1[s];
double init2[s];
double M1[s];
double M11[s];
double M12[s];
double M2[s];
double M21[s];
double M22[s];
double M3[s];
double M31[s];
double M32[s];
double M4[s];
double M41[s];
double M42[s];
double M5[s];
double M51[s];
double M52[s];
double M6[s];
double M61[s];
double M62[s];
double M7[s];
double M71[s];
double M72[s];
double M8[s];
double M81[s];
double M82[s];
double M9[s];
double M91[s];
double M92[s];
double M10[s];
double M101[s];
double M102[s];
```

```
double initi[s];
double initi1[s];
double initi2[s];
double M1i[s];
double M11i[s];
double M12i[s];
double M2i[s];
double M21i[s];
double M22i[s];
double M3i[s];
double M31i[s];
double M32i[s];
double M4i[s];
double M41i[s];
double M42i[s];
double M5i[s];
double M51i[s];
double M52i[s];
double M6i[s];
double M61i[s];
interfaz.cpp 15
double M62i[s];
double M7i[s];
double M71i[s];
double M72i[s];
double M8i[s];
double M81i[s];
double M82i[s];
double M9i[s];
double M91i[s];
double M92i[s];
double M10i[s];
double M101i[s];
double M102i[s];
//QString d=ui->comboBox->currentText();
int i,num;
int d=2048;
int bit;
double a,de;
for(i=0;1;++i) //revisa si el dato es una potencia de 2
{
de=pow(2.0,i);
a=d/de;
//std::cout << i << std::endl;
if(a<1.0)
QMessageBox::about(this,"Error","El numero no es potencia de 2");
else if(a==1.0){
bit=i;
//std::cout << bit << std::endl;
//printf("bit= %d\n",bit);
break;}
}
for(int ii=0;ii<d;++ii){ //(int ii=0;ii<d.toInt();++ii)
//printf("posicion: %d, ",i);
num=bin(ii,bit); //realiza ordenamiento
init[num]=Data[ii];
```

```
init1[num]=Data1[ii];
init2[num]=Data2[ii];
initi[num]=0;
initi1[num]=0;
initi2[num]=0;
//std::cout << Data[i] << " " <<init[i] << " " << i <<std::endl;
//printf("[%f,%f]\n",data[i],initi[num]);
}
int p;
int l=0;
double W[2];
double Wnb[2];
double Wnb1[2];
double Wnb2[2];
interfaz.cpp 16
double val=0;
double pot=0;
double aux[s];
double aux1[s];
double aux3[s];
memcpy(aux, init, sizeof(aux));
memcpy(aux1, init1, sizeof(aux1));
memcpy(aux3, init2, sizeof(aux3));
double auxi[s];
double auxi1[s];
double auxi2[s];
memcpy(auxi, initi, sizeof(auxi));
memcpy(auxi1, initi1, sizeof(auxi1));
memcpy(auxi2, initi2, sizeof(auxi2));
double aux2[s];
double aux21[s];
double aux22[s];
double aux2i[s];
double aux2i1[s];
double aux2i2[s];
double ex=0;
for(i=0;i<bit;++i){ //se desplaza entre las columnas de las mariposas
pot=pow(2,i); //potencia de la mariposa
//printf("ma= %d\n",ma);
//printf("pot= %f\n",pot);
do{ // va tomando grupos
for(int j=0;j<pot;++j){ //evalua las mariposas de cada uno de los
grupos
p=j+(int)pot+l;
ex=(d/(pot*2.0))*j; //(d.toInt()/(pot*2.0))*j;
//std::cout << ex << std::endl;
//exp=pow(2.0,bit-i)*pow(2.0,l);
val=M_PI*(2.0/d)*ex; // M_PI*(2.0/d.toInt())*ex
//std::cout << M_PI << std::endl;
//}
W[0]=cos(val); //valor real
W[1]=-sin(val); //valor imaginario
Wnb[0]=W[0]*aux[p]-W[1]*auxi[p]; //real
Wnb1[0]=W[0]*aux1[p]-W[1]*auxi1[p]; //real
Wnb2[0]=W[0]*aux3[p]-W[1]*auxi2[p]; //real
Wnb[1]=W[0]*auxi[p]+W[1]*aux[p]; //imaginario
```

```
Wnb1[1]=W[0]*auxi1[p]+W[1]*aux1[p]; //imaginario
Wnb2[1]=W[0]*auxi2[p]+W[1]*aux3[p]; //imaginario
//std::cout << W[0] << std::endl;
//printf("Wnb[1]=%d\n",Wnb[1]);
aux2[j+l]=aux[j+l]+Wnb[0]; //real superior
aux21[j+l]=aux1[j+l]+Wnb1[0]; //real superior
aux22[j+l]=aux3[j+l]+Wnb2[0]; //real superior
aux2[p]=aux[j+l]-Wnb[0]; //real inferior
aux21[p]=aux1[j+l]-Wnb1[0]; //real inferior
aux22[p]=aux3[j+l]-Wnb2[0]; //real inferior
aux2i[j+l]=auxi[j+l]+Wnb[1]; //imaginario superior
interfaz.cpp 17
aux2i1[j+l]=auxi1[j+l]+Wnb1[1]; //imaginario superior
aux2i2[j+l]=auxi2[j+l]+Wnb2[1]; //imaginario superior
aux2i[p]=auxi[j+l]-Wnb[1]; //imaginario inferior
aux2i1[p]=auxi1[j+l]-Wnb1[1]; //imaginario inferior
aux2i2[p]=auxi2[j+l]-Wnb2[1]; //imaginario inferior
//std::cout << aux2[p] << std::endl;
/*printf("realSup=%f\n",aux[j+1]);
printf("realInf=%f\n",aux2[p]);
printf("imagSup=%f\n",aux2i[j+1]);
printf("imagInf=%f\n",aux2i[p]);
printf("\n");*/
//std::cout << l << std::endl;
}
l=l+((int)pot*2);
//std::cout << l << std::endl;
}while(l<d); //d.toInt()
l=0;
if(i==0){
memcpy(M1, aux2, sizeof(M1));
memcpy(M11, aux21, sizeof(M11));
memcpy(M12, aux22, sizeof(M12));
memcpy(M1i, aux2i, sizeof(M1i));
memcpy(M11i, aux2i1, sizeof(M11i));
memcpy(M12i, aux2i2, sizeof(M12i));
}
if(i==1){
memcpy(M2, aux2, sizeof(M2));
memcpy(M21, aux21, sizeof(M21));
memcpy(M22, aux22, sizeof(M22));
memcpy(M2i, aux2i, sizeof(M2i));
memcpy(M21i, aux2i1, sizeof(M21i));
memcpy(M22i, aux2i2, sizeof(M22i));
}
if(i==2){
memcpy(M3, aux2, sizeof(M3));
memcpy(M31, aux21, sizeof(M31));
memcpy(M32, aux22, sizeof(M32));
memcpy(M3i, aux2i, sizeof(M3i));
memcpy(M31i, aux2i1, sizeof(M31i));
memcpy(M32i, aux2i2, sizeof(M32i));
}
if(i==3){
memcpy(M4, aux2, sizeof(M4));
memcpy(M41, aux21, sizeof(M41));
```

```
memcpy(M42, aux22, sizeof(M42));
memcpy(M4i, aux2i, sizeof(M4i));
memcpy(M41i, aux2i1, sizeof(M41i));
memcpy(M42i, aux2i2, sizeof(M42i));
}
if(i==4){
memcpy(M5, aux2, sizeof(M5));
memcpy(M51, aux21, sizeof(M51));
memcpy(M52, aux22, sizeof(M52));
interfaz.cpp 18
memcpy(M5i, aux2i, sizeof(M5i));
memcpy(M51i, aux2i1, sizeof(M51i));
memcpy(M52i, aux2i2, sizeof(M52i));
}
if(i==5){
memcpy(M6, aux2, sizeof(M6));
memcpy(M61, aux21, sizeof(M61));
memcpy(M62, aux22, sizeof(M62));
memcpy(M6i, aux2i, sizeof(M6i));
memcpy(M61i, aux2i1, sizeof(M61i));
memcpy(M62i, aux2i2, sizeof(M62i));
}
if(i==6){
memcpy(M7, aux2, sizeof(M7));
memcpy(M71, aux21, sizeof(M71));
memcpy(M72, aux22, sizeof(M72));
memcpy(M7i, aux2i, sizeof(M7i));
memcpy(M71i, aux2i1, sizeof(M71i));
memcpy(M72i, aux2i2, sizeof(M72i));
}
if(i==7){
memcpy(M8, aux2, sizeof(M8));
memcpy(M81, aux21, sizeof(M81));
memcpy(M82, aux22, sizeof(M82));
memcpy(M8i, aux2i, sizeof(M8i));
memcpy(M81i, aux2i1, sizeof(M81i));
memcpy(M82i, aux2i2, sizeof(M82i));
}
if(i==8){
memcpy(M9, aux2, sizeof(M9));
memcpy(M91, aux21, sizeof(M91));
memcpy(M92, aux22, sizeof(M92));
memcpy(M9i, aux2i, sizeof(M9i));
memcpy(M91i, aux2i1, sizeof(M91i));
memcpy(M92i, aux2i2, sizeof(M92i));
}
if(i==9){
memcpy(M10, aux2, sizeof(M10));
memcpy(M101, aux21, sizeof(M101));
memcpy(M102, aux22, sizeof(M102));
memcpy(M10i, aux2i, sizeof(M10i));
memcpy(M101i, aux2i1, sizeof(M101i));
memcpy(M102i, aux2i2, sizeof(M102i));
}
memcpy(aux, aux2, sizeof(aux));
memcpy(aux1, aux21, sizeof(aux1));
```

```
memcpy(aux3, aux22, sizeof(aux3));
memcpy(auxi, aux2i, sizeof(auxi));
memcpy(auxi1, aux2i1, sizeof(auxi1));
memcpy(auxi2, aux2i2, sizeof(auxi2));
}
QVector <double> fin(s), fin1(s), fin2(s); //vector de los resultantes
interfaz.cpp 19
double Fs = 8000;
double N = d;
for(int g=0;g<N;++g){
//fin.resize(d.toInt());
fin[g]=sqrt(pow(aux2[g],2.0)+pow(aux2i[g],2.0)); //se calculan los valores
resultantes de cada uno de los valores
fin1[g]=sqrt(pow(aux21[g],2.0)+pow(aux2i1[g],2.0));
fin2[g]=sqrt(pow(aux22[g],2.0)+pow(aux2i2[g],2.0));
gg[g]=(abs(fin[g])/(N/2));
gg1[g]=(abs(fin1[g])/(N/2));
gg2[g]=(abs(fin2[g])/(N/2));
//std::cout << fin[g] << " " << g << " " << gg [g] << std::endl;
//gg.append(g);
}
for(int h=0;h<N/2;++h){
freq[h]=(h*Fs/N);
gg[0]=0;
//std::cout << freq[h] << " " << h << " " << gg[h] << std::endl;
}
int n=13;
double thd=0,thd1=0,thd2=0,sum=0,sum1=0,sum2=0;
QVector <double> nrms(50),nrms1(50),nrms2(50);
for(int m=0;m<50;++m){
nrms[m]=gg[n];
nrms1[m]=gg1[n];
nrms2[m]=gg2[n];
n=n+13;
nrms[m]=nrms[m]*0.7071;
nrms1[m]=nrms1[m]*0.7071;
nrms2[m]=nrms2[m]*0.7071;
//qDebug() << nrms;
}
for(int n=1;n<50;n++){
sum=sum+pow(nrms[n],2);
sum1=sum1+pow(nrms1[n],2);
sum2=sum2+pow(nrms2[n],2);
thd=((sqrt(sum))/nrms[0])*100;
thd1=((sqrt(sum1))/nrms1[0])*100;
thd2=((sqrt(sum2))/nrms2[0])*100;
}
int o=20;
double tihd=0,tihd1=0,tihd2=0,sumi=0,sumi1=0,sumi2=0;
QVector <double> nrmsi(50),nrmsi1(50),nrmsi2(50);
for(int p=0;p<50;++p){
nrmsi[p]=gg[o];
nrmsi1[p]=gg1[o];
nrmsi2[p]=gg2[o];
o=o+20;
nrmsi[p]=nrmsi[p]*0.7071;
```

interfaz.cpp 20

```
nrmsi1[p]=nrmsi1[p]*0.7071;
nrmsi2[p]=nrmsi2[p]*0.7071;
//qDebug() << nrms;
}
for(int n=1;n<50;n++){
sumi=sumi+pow(nrmsi[n],2);
sumi1=sumi1+pow(nrmsi1[n],2);
sumi2=sumi2+pow(nrmsi2[n],2);
tihd=((sqrt(sumi))/nrmsi[0])*100;
tihd1=((sqrt(sumi1))/nrmsi1[0])*100;
tihd2=((sqrt(sumi2))/nrmsi2[0])*100;
}
ui->thd->setText(QString::number(thd));
ui->thd1->setText(QString::number(thd1));
ui->thd2->setText(QString::number(thd2));
if(thd>5){
ui->textEdit->append(QString::number(cc)+") "+"Codigo 01_A: valor de THD
de la fase A es superior al 5% : " + QString::number(thd) + "\n");
cc++;
}
if(thd1>5){
ui->textEdit->append(QString::number(cc)+") "+"Codigo 01_B: valor de THD
de la fase B es superior al 5% : " + QString::number(thd1) + "\n");
cc++;
}
if(thd2>5){
ui->textEdit->append(QString::number(cc)+") "+"Codigo 01_C: valor de THD
de la fase C es superior al 5% : " + QString::number(thd2) + "\n");
cc++;
}
ui->tihd->setText(QString::number(tihd));
ui->tihd1->setText(QString::number(tihd1));
ui->tihd2->setText(QString::number(tihd2));
if(ui->Cfa->isChecked()){
ui->plot->graph(0)->setData(freq,gg);}
ui->plot->graph(0)->rescaleAxes();
if(ui->Cfb->isChecked()){
ui->plot->graph(1)->setData(freq,gg1);}
ui->plot->graph(1)->rescaleAxes();
if(ui->Cfc->isChecked()){
ui->plot->graph(2)->setData(freq,gg2);}
ui->plot->graph(2)->rescaleAxes();
ui->plot->setInteractions(QCP::iRangeDrag | QCP::iRangeZoom |
QCP::iSelectPlottables);
ui->plot->replot();
}
void Interfaz::on_Cfa_stateChanged(int arg1)
```

interfaz.cpp 21

```
{
QVector<double> x(8192);
QString d=ui->comboBox->currentText();
QString s=ui->cont->text();
if(ui->Cfa->isChecked() and s.toInt() == 1){
for(int i=0; i<d.toInt()-1; ++i){
x[i]=i;
```

```
}
ui->plot->graph(0)->setData(x,Data);
ui->plot->setInteractions(QCP::iRangeDrag | QCP::iRangeZoom |
QCP::iSelectPlottables);
ui->plot->replot();
}
else if(ui->Cfa->isCheckable()){
ui->plot->graph(0)->data()->clear();
ui->plot->replot();
}
if(ui->Cfa->isChecked() and s.toInt() == 2){
ui->plot->graph(0)->setData(freq,gg);
ui->plot->setInteractions(QCP::iRangeDrag | QCP::iRangeZoom |
QCP::iSelectPlottables);
ui->plot->replot();
}
}
void Interfaz::on_Cfb_stateChanged(int arg1)
{
QVector<double> x(8192);
QString d=ui->comboBox->currentText();
QString s=ui->cont->text();
if(ui->Cfb->isChecked() and s.toInt() == 1){
for(int i=0; i<d.toInt()-1; ++i){
x[i]=i;
}
ui->plot->graph(1)->setData(x,Data1);
ui->plot->setInteractions(QCP::iRangeDrag | QCP::iRangeZoom |
QCP::iSelectPlottables);
ui->plot->replot();
}
else if(ui->Cfb->isCheckable()){
ui->plot->graph(1)->data()->clear();
ui->plot->replot();
}
if(ui->Cfb->isChecked() and s.toInt() == 2){
ui->plot->graph(1)->setData(freq,gg1);
ui->plot->setInteractions(QCP::iRangeDrag | QCP::iRangeZoom |
QCP::iSelectPlottables);
ui->plot->replot();
}
}
void Interfaz::on_Cfc_stateChanged(int arg1)
{
interfaz.cpp 22
QVector<double> x(8192);
QString d=ui->comboBox->currentText();
QString s=ui->cont->text();
if(ui->Cfc->isChecked() and s.toInt() == 1){
for(int i=0; i<d.toInt()-1; ++i){
x[i]=i;
}
ui->plot->graph(2)->setData(x,Data2);
ui->plot->setInteractions(QCP::iRangeDrag | QCP::iRangeZoom |
QCP::iSelectPlottables);
ui->plot->replot();
```

```
}
else if(ui->Cfc->isCheckable()){
ui->plot->graph(2)->data()->clear();
ui->plot->replot();
}
if(ui->Cfc->isChecked() and s.toInt() == 2){
ui->plot->graph(2)->setData(freq,gg2);
ui->plot->setInteractions(QCP::iRangeDrag | QCP::iRangeZoom |
QCP::iSelectPlottables);
ui->plot->replot();
}
}
void Interfaz::on_Ccfa_stateChanged(int arg1)
{
QVector<double> x(8192);
QString d=ui->comboBox->currentText();
QString s=ui->cont->text();
if(ui->Ccfa->isChecked() and s.toInt() == 1){
for(int i=0; i<d.toInt()-1; ++i){
x[i]=i;
}
ui->plot->graph(3)->setData(x,CDat);
ui->plot->setInteractions(QCP::iRangeDrag | QCP::iRangeZoom |
QCP::iSelectPlottables);
ui->plot->replot();
}
else if(ui->Ccfa->isCheckable()){
ui->plot->graph(3)->data()->clear();
ui->plot->replot();
}
if(ui->Ccfa->isChecked() and s.toInt() == 2){
ui->plot->graph(3)->data()->clear();
ui->plot->replot();
}
}
void Interfaz::on_Ccfb_stateChanged(int arg1)
{
QVector<double> x(8192);
QString d=ui->comboBox->currentText();
QString s=ui->cont->text();
interfaz.cpp 23
if(ui->Ccfb->isChecked() and s.toInt() == 1){
for(int i=0; i<d.toInt()-1; ++i){
x[i]=i;
}
ui->plot->graph(4)->setData(x,CDat1);
ui->plot->setInteractions(QCP::iRangeDrag | QCP::iRangeZoom |
QCP::iSelectPlottables);
ui->plot->replot();
}
else if(ui->Ccfb->isCheckable()){
ui->plot->graph(4)->data()->clear();
ui->plot->replot();
}
if(ui->Ccfb->isChecked() and s.toInt() == 2){
ui->plot->graph(4)->data()->clear();
```

```
ui->plot->replot();
}
}
void Interfaz::on_Ccfc_stateChanged(int arg1)
{
QVector<double> x(8192);
QString d=ui->comboBox->currentText();
QString s=ui->cont->text();
if(ui->Ccfc->isChecked() and s.toInt() == 1){
for(int i=0; i<d.toInt()-1; ++i){
x[i]=i;
}
ui->plot->graph(5)->setData(x,CDat2);
ui->plot->setInteractions(QCP::iRangeDrag | QCP::iRangeZoom |
QCP::iSelectPlottables);
ui->plot->replot();
}
else if(ui->Ccfc->isCheckable()){
ui->plot->graph(5)->data()->clear();
ui->plot->replot();
}
if(ui->Ccfc->isChecked() and s.toInt() == 2){
ui->plot->graph(5)->data()->clear();
ui->plot->replot();
}
}
void Interfaz::on_horizontalScrollBar_actionTriggered(int action)
{
ui->plot->addGraph();
ui->plot->graph(0)->setPen(QPen(Qt::blue));
ui->plot->addGraph();
ui->plot->graph(1)->setPen(QPen(Qt::red));
ui->plot->addGraph();
ui->plot->graph(2)->setPen(QPen(Qt::green));
ui->plot->addGraph();
ui->plot->graph(3)->setPen(QPen(Qt::cyan));
interfaz.cpp 24
ui->plot->addGraph();
ui->plot->graph(4)->setPen(QPen(Qt::magenta));
ui->plot->addGraph();
ui->plot->graph(5)->setPen(QPen(Qt::darkYellow));
ui->plot->graph(0)->data()->clear();
ui->plot->graph(1)->data()->clear();
ui->plot->graph(2)->data()->clear();
ui->plot->graph(3)->data()->clear();
ui->plot->graph(4)->data()->clear();
ui->plot->graph(5)->data()->clear();
int x;
QVector <double> f(4096);
x = ui->horizontalScrollBar->value();
//qDebug() << x;
QString d=ui->comboBox->currentText();
if(d.toInt()==8192){
ui->horizontalScrollBar->setMaximum(585);
f.resize(8192);
}
```

```
if(d.toInt()==4096){
ui->horizontalScrollBar->setMaximum(1171);
}
if(d.toInt()==2048){
ui->horizontalScrollBar->setMaximum(2343);
f.resize(2048);
}
if(d.toInt()==1024){
ui->horizontalScrollBar->setMaximum(4687);
f.resize(1024);
}
if(d.toInt()==512){
ui->horizontalScrollBar->setMaximum(9375);
f.resize(512);
}
if(d.toInt()==256){
ui->horizontalScrollBar->setMaximum(18750);
f.resize(256);
}
if(d.toInt()==128){
ui->horizontalScrollBar->setMaximum(37500);
f.resize(128);
}
int j=0;
int v=x*d.toInt();
//qDebug() << v;
for(j = 0; j<d.toInt(); ++j){
Data[j]=ap[v];
Data1[j]=ap1[v];
Data2[j]=ap2[v];
CDat[j]=ac[v];
CDat1[j]=ac1[v];
CDat2[j]=ac2[v];
interfaz.cpp 25
f[j]=v;
v++;
//std::cout << v << " " << j << " " << ap[v] << std::endl;
}
QString ss=ui->cont->text();
QString s1=ui->cont2->text();
if(ss.toInt()==1){
int Fs=8000;
int N=8000;
double ven=Fs/5;
double num=N/ven;
QVector <double> v0(8192);
QVector <double> v1(8192);
QVector <double> v2(8192);
double mult,mult1,mult2,ang=0,ang1=0,ang2=0;
QVector <double> Xrms(10);
QVector <double> Xrms1(10);
QVector <double> Xrms2(10);
QVector <double> max(10);
QVector <double> max1(10);
QVector <double> max2(10);
QVector <double> Yrms(10);
```

```
QVector <double> Yrms1(10);
QVector <double> Yrms2(10);
QVector <double> maxy(10);
QVector <double> maxy1(10);
QVector <double> maxy2(10);
QVector <double> S(10);
QVector <double> S1(10);
QVector <double> S2(10);
QVector <double> P(10);
QVector <double> P1(10);
QVector <double> P2(10);
QVector <double> Q(10);
QVector <double> Q1(10);
QVector <double> Q2(10);
QVector <double> FP(10);
QVector <double> FP1(10);
QVector <double> FP2(10);
QVector <double> Palt(10);
QVector <double> Palt1(10);
QVector <double> Palt2(10);
double p0=0,p1=0,p01=0,p11=0,p02=0,p12=0;
for(int i=0;i<=num-2;i++){
Palt[i+1]=0;
Palt1[i+1]=0;
Palt2[i+1]=0;
for(int j=ven*i;j<=ven*(i+1);j++){
Xrms[i+1]=Xrms[i+1]+pow(Data[j+1],2);//*datV[j+1]);
Xrms1[i+1]=Xrms1[i+1]+pow(Data1[j+1],2);
interfaz.cpp 26
Xrms2[i+1]=Xrms2[i+1]+pow(Data2[j+1],2);
//qDebug() << Xrms;
//printf("j: %d\n",j);
if(Data[j+1]>max[i+1]){
max[i+1]=Data[j+1];
}
if(Data1[j+1]>max1[i+1]){
max1[i+1]=Data1[j+1];
}
//qDebug () << max;
if(Data2[j+1]>max2[i+1]){
max2[i+1]=Data2[j+1];
}
if(Data[j+1]<0 && p0==0){
p0=j-(1/(Data[j+1]-Data[j]))*Data[j];
}
if(Data1[j+1]<0 && p01==0){
p01=j-(1/(Data1[j+1]-Data1[j]))*Data1[j];
}
if(Data2[j+1]<0 && p02==0){
p02=j-(1/(Data2[j+1]-Data2[j]))*Data2[j];
}
Yrms[i+1]=Yrms[i+1]+CDat[j+1]*CDat[j+1];
Yrms1[i+1]=Yrms1[i+1]+CDat1[j+1]*CDat1[j+1];
Yrms2[i+1]=Yrms2[i+1]+CDat2[j+1]*CDat2[j+1];
if(CDat[j+1]>maxy[i+1]){
maxy[i+1]=CDat[j+1];
```

```
}
if(CDat1[j+1]>maxy1[i+1]){
maxy1[i+1]=CDat1[j+1];
}
if(CDat2[j+1]>maxy2[i+1]){
maxy2[i+1]=CDat2[j+1];
}
mult=Data[j+1]*CDat[j+1];
mult1=Data1[j+1]*CDat1[j+1];
mult2=Data2[j+1]*CDat2[j+1];
Palt[i+1]=Palt[i+1]+mult;
Palt1[i+1]=Palt1[i+1]+mult1;
Palt2[i+1]=Palt2[i+1]+mult2;
v0[j+1]=Data[j+1];
v1[j+1]=Data1[j+1];
v2[j+1]=Data2[j+1];
if(CDat[j+1]<0&&p1==0){
p1=j-(1/(CDat[j+1]-CDat[j]))*CDat[j];
}
if(CDat1[j+1]<0&&p11==0){
p11=j-(1/(CDat1[j+1]-CDat1[j]))*CDat1[j];
}
if(CDat2[j+1]<0&&p12==0){
p12=j-(1/(CDat2[j+1]-CDat2[j]))*CDat2[j];
}
}
interfaz.cpp 27
Palt[i+1]=Palt[i+1]/ven;
Palt1[i+1]=Palt1[i+1]/ven;
Palt2[i+1]=Palt2[i+1]/ven;
ang=abs(p1-p0)*(90/p0);
ang1=abs(p11-p01)*(210/p01);
ang2=abs(p12-p02)*(330/p02);
}
// thd=0.117;
double c,c1,c2;
for(int i=0;i<=num-2;i++){
//printf("Indice i: %f Voltaje RMS : %f\n",ven,Xrms[i+1]);
Xrms[i+1]=sqrt(Xrms[i+1]*(1/ven));
Xrms1[i+1]=sqrt(Xrms1[i+1]*(1/ven));
Xrms2[i+1]=sqrt(Xrms2[i+1]*(1/ven));
Yrms[i+1]=sqrt(Yrms[i+1]*(1/ven));
Yrms1[i+1]=sqrt(Yrms1[i+1]*(1/ven));
Yrms2[i+1]=sqrt(Yrms2[i+1]*(1/ven));
c=abs(maxy[i+1])/Yrms[i+1];
c1=abs(maxy1[i+1])/Yrms1[i+1];
c2=abs(maxy2[i+1])/Yrms2[i+1];
S[i+1]=Xrms[i+1]*Yrms[i+1];
S1[i+1]=Xrms1[i+1]*Yrms1[i+1];
S2[i+1]=Xrms2[i+1]*Yrms2[i+1];
P[i+1]=S[i+1]*cos(ang*(M_PI/180));
P1[i+1]=S1[i+1]*cos(ang1*(M_PI/180));
P2[i+1]=S2[i+1]*cos(ang2*(M_PI/180));
Q[i+1]=S[i+1]*sin(ang*(M_PI/180));
Q1[i+1]=S1[i+1]*sin(ang1*(M_PI/180));
Q2[i+1]=S2[i+1]*sin(ang2*(M_PI/180));
```

```
FP[i+1]=cos(ang*(M_PI/180));
FP1[i+1]=cos(ang1*(M_PI/180));
FP2[i+1]=cos(ang2*(M_PI/180));
}
ui->rms->setText(QString::number(Xrms[1]));
ui->rms1->setText(QString::number(Xrms1[1]));
ui->rms2->setText(QString::number(Xrms1[1]));
ui->fp->setText(QString::number(abs(FP[1])));
ui->fp1->setText(QString::number(abs(FP1[1])));
ui->fp2->setText(QString::number(abs(FP2[1])));
ui->Pp->setText(QString::number(P[1]));
ui->Pp1->setText(QString::number(P1[1]));
ui->Pp2->setText(QString::number(P2[1]));
ui->Pq->setText(QString::number(Q[1]));
ui->Pq1->setText(QString::number(Q1[1]));
ui->Pq2->setText(QString::number(Q2[1]));
ui->Ps->setText(QString::number(S[1]));
ui->Ps1->setText(QString::number(S1[1]));
ui->Ps2->setText(QString::number(S2[1]));
if(ui->Cfa->isChecked()){
ui->plot->graph(0)->setData(f,Data);
}
if(ui->Cfb->isChecked()){
interfaz.cpp 28
ui->plot->graph(1)->setData(f,Data1);
}
if(ui->Cfc->isChecked()){
ui->plot->graph(2)->setData(f,Data2);
}
if(ui->Ccfa->isChecked()){
ui->plot->graph(3)->setData(f,CDat);
}
if(ui->Ccfb->isChecked()){
ui->plot->graph(4)->setData(f,CDat1);
}
if(ui->Ccfc->isChecked()){
ui->plot->graph(5)->setData(f,CDat2);
}
ui->plot->graph(0)->rescaleAxes();
ui->plot->graph(1)->rescaleAxes();
ui->plot->graph(2)->rescaleAxes();
//ui->plot->graph(3)->rescaleAxes();
//ui->plot->graph(4)->rescaleAxes();
//ui->plot->graph(5)->rescaleAxes();
ui->plot->setInteractions(QCP::iRangeDrag | QCP::iRangeZoom |
QCP::iSelectPlottables);
ui->plot->replot();
if(s1.toInt()==1){
QVector <double> max(6);
QVector <double> max1(6);
QVector <double> max2(6);
QVector <double> tem(8192),tem1(8192),tem2(8192);
QString d=ui->comboBox->currentText();
QVector <double> x(8192);
// ui->plot->addGraph();
// ui->plot->graph(6)->setPen(QPen(Qt::darkBlue));
```

```
// ui->plot->addGraph();
// ui->plot->graph(7)->setPen(QPen(Qt::darkRed));
// ui->plot->addGraph();
// ui->plot->graph(8)->setPen(QPen(Qt::darkGreen));
// ui->plot->graph(6)->data()->clear();
// ui->plot->graph(7)->data()->clear();
// ui->plot->graph(8)->data()->clear();
//int N=8000;
double ven=d.toInt()/5;
double num=d.toInt()/ven;
for(int i=0;i<=num;++i){
for(int j=ven*i;j<=ven*(i+1);++j){
if(Data[j]>max[i]){
max[i]=Data[j];
}
if(Data1[j]>max1[i]){
interfaz.cpp 29
max1[i]=Data1[j];
}
if(Data2[j]>max2[i]){
max2[i]=Data2[j];
}
}
//x.append(i);
}
for(int k =0;k < d.toInt()/5;++k){
tem[k]=max[0];
tem1[k]=max1[0];
tem2[k]=max2[0];
}
for(int k = (d.toInt()/5);k < (d.toInt()/5)*2;++k){
tem[k]=max[1];
tem1[k]=max1[1];
tem2[k]=max2[1];
}
for(int k = (d.toInt()/5)*2;k < (d.toInt()/5)*3;++k){
tem[k]=max[2];
tem1[k]=max1[2];
tem2[k]=max2[2];
}
for(int k = (d.toInt()/5)*3;k < (d.toInt()/5)*4;++k){
tem[k]=max[3];
tem1[k]=max1[3];
tem2[k]=max2[3];
}
for(int k = (d.toInt()/5)*4;k < (d.toInt()/5)*5;++k){
tem[k]=max[4];
tem1[k]=max1[4];
tem2[k]=max2[4];
}
for(int l=0;l<=d.toInt();++l){
x[l]=l;
}
if(ui->Cfa->isChecked()){
ui->plot->graph(0)->setData(f,Data);
ui->plot->graph(6)->setData(x,tem);
```

```
}
if(ui->Cfb->isChecked()){
ui->plot->graph(1)->setData(f,Data1);
ui->plot->graph(7)->setData(x,tem1);
}
if(ui->Cfc->isChecked()){
ui->plot->graph(2)->setData(f,Data2);
ui->plot->graph(8)->setData(x,tem2);
}
if(ui->Ccfa->isChecked()){
ui->plot->graph(3)->setData(f,CDat);
interfaz.cpp 30
}
if(ui->Ccfb->isChecked()){
ui->plot->graph(4)->setData(f,CDat1);
}
if(ui->Ccfc->isChecked()){
ui->plot->graph(5)->setData(f,CDat2);
}
ui->plot->graph(6)->rescaleAxes();
ui->plot->graph(7)->rescaleAxes();
ui->plot->graph(8)->rescaleAxes();
ui->plot->setInteractions(QCP::iRangeDrag | QCP::iRangeZoom |
QCP::iSelectPlottables);
ui->plot->replot();
}
}
else if(ss.toInt() == 2){
const size_t s = 3000;
double init[s];
double init1[s];
double init2[s];
double M1[s];
double M11[s];
double M12[s];
double M2[s];
double M21[s];
double M22[s];
double M3[s];
double M31[s];
double M32[s];
double M4[s];
double M41[s];
double M42[s];
double M5[s];
double M51[s];
double M52[s];
double M6[s];
double M61[s];
double M62[s];
double M7[s];
double M71[s];
double M72[s];
double M8[s];
double M81[s];
double M82[s];
```

```
double M9[s];
double M91[s];
double M92[s];
double M10[s];
double M101[s];
double M102[s];
interfaz.cpp 31
double initi[s];
double initi1[s];
double initi2[s];
double M1i[s];
double M11i[s];
double M12i[s];
double M2i[s];
double M21i[s];
double M22i[s];
double M3i[s];
double M31i[s];
double M32i[s];
double M4i[s];
double M41i[s];
double M42i[s];
double M5i[s];
double M51i[s];
double M52i[s];
double M6i[s];
double M61i[s];
double M62i[s];
double M7i[s];
double M71i[s];
double M72i[s];
double M8i[s];
double M81i[s];
double M82i[s];
double M9i[s];
double M91i[s];
double M92i[s];
double M10i[s];
double M101i[s];
double M102i[s];
//QString d=ui->comboBox->currentText();
int i,num;
int d=2048;
int bit;
double a,de;
for(i=0;1;++i) //revisa si el dato es una potencia de 2
{
de=pow(2.0,i);
a=d/de;
//std::cout << i << std::endl;
if(a<1.0)
QMessageBox::about(this,"Error","El numero no es potencia de
2");
else if(a==1.0){
bit=i;
//std::cout << bit << std::endl;
```

```
//printf("bit= %d\n",bit);
break;}
interfaz.cpp 32
}
for(int ii=0;ii<d;++ii){ //(int ii=0;ii<d.toInt();++ii)
//printf("posicion: %d, ",i);
num=bin(ii,bit); //realiza ordenamiento
init[num]=Data[ii];
init1[num]=Data1[ii];
init2[num]=Data2[ii];
initi[num]=0;
initi1[num]=0;
initi2[num]=0;
//std::cout << Data[i] << " " <<init[i] << " " << i <<std::endl;
//printf("[%f,%f]\n",data[i],initi[num]);
}
int p;
int l=0;
double W[2];
double Wnb[2];
double Wnb1[2];
double Wnb2[2];
double val=0;
double pot=0;
double aux[s];
double aux1[s];
double aux3[s];
memcpy(aux, init, sizeof(aux));
memcpy(aux1, init1, sizeof(aux1));
memcpy(aux3, init2, sizeof(aux3));
double auxi[s];
double auxi1[s];
double auxi2[s];
memcpy(auxi, initi, sizeof(auxi));
memcpy(auxi1, initi1, sizeof(auxi1));
memcpy(auxi2, initi2, sizeof(auxi2));
double aux2[s];
double aux21[s];
double aux22[s];
double aux2i[s];
double aux2i1[s];
double aux2i2[s];
double ex=0;
for(i=0;i<bit;++i){ //se desplaza entre las columnas de las mariposas
pot=pow(2,i); //potencia de la mariposa
//printf("ma= %d\n",ma);
//printf("pot= %f\n",pot);
do{ // va tomando grupos
for(int j=0;j<pot;++j){ //evalua las mariposas de cada uno de los
grupos
p=j+(int)pot+l;
ex=(d/(pot*2.0))*j; //(d.toInt()/(pot*2.0))*j;
//std::cout << ex << std::endl;
interfaz.cpp 33
//exp=pow(2.0,bit-i)*pow(2.0,l);
val=M_PI*(2.0/d)*ex; // M_PI*(2.0/d.toInt())*ex
```

```
//std::cout << M_PI << std::endl;
//}
W[0]=cos(val); //valor real
W[1]=-sin(val); //valor imaginario
Wnb[0]=W[0]*aux[p]-W[1]*auxi[p]; //real
Wnb1[0]=W[0]*aux1[p]-W[1]*auxi1[p]; //real
Wnb2[0]=W[0]*aux3[p]-W[1]*auxi2[p]; //real
Wnb[1]=W[0]*auxi[p]+W[1]*aux[p]; //imaginario
Wnb1[1]=W[0]*auxi1[p]+W[1]*aux1[p]; //imaginario
Wnb2[1]=W[0]*auxi2[p]+W[1]*aux3[p]; //imaginario
//std::cout << W[0] << std::endl;
//printf("Wnb[1]=%d\n",Wnb[1]);
aux2[j+l]=aux[j+l]+Wnb[0]; //real superior
aux21[j+l]=aux1[j+l]+Wnb1[0]; //real superior
aux22[j+l]=aux3[j+l]+Wnb2[0]; //real superior
aux2[p]=aux[j+l]-Wnb[0]; //real inferior
aux21[p]=aux1[j+l]-Wnb1[0]; //real inferior
aux22[p]=aux3[j+l]-Wnb2[0]; //real inferior
aux2i[j+l]=auxi[j+l]+Wnb[1]; //imaginario superior
aux2i1[j+l]=auxi1[j+l]+Wnb1[1]; //imaginario superior
aux2i2[j+l]=auxi2[j+l]+Wnb2[1]; //imaginario superior
aux2i[p]=auxi[j+l]-Wnb[1]; //imaginario inferior
aux2i1[p]=auxi1[j+l]-Wnb1[1]; //imaginario inferior
aux2i2[p]=auxi2[j+l]-Wnb2[1]; //imaginario inferior
//std::cout << aux2[p] << std::endl;
/*printf("realSup=%f\n",aux[j+1]);
printf("realInf=%f\n",aux2[p]);
printf("imagSup=%f\n",aux2i[j+1]);
printf("imagInf=%f\n",aux2i[p]);
printf("\n");*/
//std::cout << l << std::endl;
}
l=l+((int)pot*2);
//std::cout << l << std::endl;
}while(l<d); //d.toInt()
l=0;
if(i==0){
memcpy(M1, aux2, sizeof(M1));
memcpy(M11, aux21, sizeof(M11));
memcpy(M12, aux22, sizeof(M12));
memcpy(M1i, aux2i, sizeof(M1i));
memcpy(M11i, aux2i1, sizeof(M11i));
memcpy(M12i, aux2i2, sizeof(M12i));
}
if(i==1){
memcpy(M2, aux2, sizeof(M2));
memcpy(M21, aux21, sizeof(M21));
memcpy(M22, aux22, sizeof(M22));
memcpy(M2i, aux2i, sizeof(M2i));
memcpy(M21i, aux2i1, sizeof(M21i));
memcpy(M22i, aux2i2, sizeof(M22i));
interfaz.cpp 34
}
if(i==2){
memcpy(M3, aux2, sizeof(M3));
memcpy(M31, aux21, sizeof(M31));
```

```
memcpy(M32, aux22, sizeof(M32));
memcpy(M3i, aux2i, sizeof(M3i));
memcpy(M31i, aux2i1, sizeof(M31i));
memcpy(M32i, aux2i2, sizeof(M32i));
}
if(i==3){
memcpy(M4, aux2, sizeof(M4));
memcpy(M41, aux21, sizeof(M41));
memcpy(M42, aux22, sizeof(M42));
memcpy(M4i, aux2i, sizeof(M4i));
memcpy(M41i, aux2i1, sizeof(M41i));
memcpy(M42i, aux2i2, sizeof(M42i));
}
if(i==4){
memcpy(M5, aux2, sizeof(M5));
memcpy(M51, aux21, sizeof(M51));
memcpy(M52, aux22, sizeof(M52));
memcpy(M5i, aux2i, sizeof(M5i));
memcpy(M51i, aux2i1, sizeof(M51i));
memcpy(M52i, aux2i2, sizeof(M52i));
}
if(i==5){
memcpy(M6, aux2, sizeof(M6));
memcpy(M61, aux21, sizeof(M61));
memcpy(M62, aux22, sizeof(M62));
memcpy(M6i, aux2i, sizeof(M6i));
memcpy(M61i, aux2i1, sizeof(M61i));
memcpy(M62i, aux2i2, sizeof(M62i));
}
if(i==6){
memcpy(M7, aux2, sizeof(M7));
memcpy(M71, aux21, sizeof(M71));
memcpy(M72, aux22, sizeof(M72));
memcpy(M7i, aux2i, sizeof(M7i));
memcpy(M71i, aux2i1, sizeof(M71i));
memcpy(M72i, aux2i2, sizeof(M72i));
}
if(i==7){
memcpy(M8, aux2, sizeof(M8));
memcpy(M81, aux21, sizeof(M81));
memcpy(M82, aux22, sizeof(M82));
memcpy(M8i, aux2i, sizeof(M8i));
memcpy(M81i, aux2i1, sizeof(M81i));
memcpy(M82i, aux2i2, sizeof(M82i));
}
if(i==8){
memcpy(M9, aux2, sizeof(M9));
memcpy(M91, aux21, sizeof(M91));
memcpy(M92, aux22, sizeof(M92));
interfaz.cpp 35
memcpy(M9i, aux2i, sizeof(M9i));
memcpy(M91i, aux2i1, sizeof(M91i));
memcpy(M92i, aux2i2, sizeof(M92i));
}
if(i==9){
memcpy(M10, aux2, sizeof(M10));
```

```
memcpy(M101, aux21, sizeof(M101));
memcpy(M102, aux22, sizeof(M102));
memcpy(M10i, aux2i, sizeof(M10i));
memcpy(M101i, aux2i1, sizeof(M101i));
memcpy(M102i, aux2i2, sizeof(M102i));
}
memcpy(aux, aux2, sizeof(aux));
memcpy(aux1, aux21, sizeof(aux1));
memcpy(aux3, aux22, sizeof(aux3));
memcpy(auxi, aux2i, sizeof(auxi));
memcpy(auxi1, aux2i1, sizeof(auxi1));
memcpy(auxi2, aux2i2, sizeof(auxi2));
}
QVector <double> fin(s), fin1(s), fin2(s); //vector de los resultantes
double Fs = 8000;
double N = d;
for(int g=0;g<N;++g){
//fin.resize(d.toInt());
fin[g]=sqrt(pow(aux2[g],2.0)+pow(aux2i[g],2.0)); //se calculan los
valores resultantes de cada uno de los valores
fin1[g]=sqrt(pow(aux21[g],2.0)+pow(aux2i1[g],2.0));
fin2[g]=sqrt(pow(aux22[g],2.0)+pow(aux2i2[g],2.0));
gg[g]=(abs(fin[g])/(N/2));
gg1[g]=(abs(fin1[g])/(N/2));
gg2[g]=(abs(fin2[g])/(N/2));
//std::cout << fin[g] << " " << g << " " << gg [g] << std::endl;
//gg.append(g);
}
for(int h=0;h<N/2;++h){
freq[h]=(h*Fs/N);
gg[0]=0;
//std::cout << freq[h] << " " << h << " " << gg[h] << std::endl;
}
int n=13;
double thd=0,thd1=0,thd2=0,sum=0,sum1=0,sum2=0;
QVector <double> nrms(50),nrms1(50),nrms2(50);
for(int m=0;m<50;++m){
nrms[m]=gg[n];
nrms1[m]=gg1[n];
nrms2[m]=gg2[n];
n=n+13;
nrms[m]=nrms[m]*0.7071;
nrms1[m]=nrms1[m]*0.7071;
nrms2[m]=nrms2[m]*0.7071;
interfaz.cpp 36
//qDebug() << nrms;
}
for(int n=1;n<50;n++){
sum=sum+pow(nrms[n],2);
sum1=sum1+pow(nrms1[n],2);
sum2=sum2+pow(nrms2[n],2);
thd=((sqrt(sum))/nrms[0])*100;
thd1=((sqrt(sum1))/nrms1[0])*100;
thd2=((sqrt(sum2))/nrms2[0])*100;
}
int o=13;
```

```
double tihd=0,tihd1=0,tihd2=0,sumi=0,sumi1=0,sumi2=0;
QVector <double> nrmsi(50),nrmsi1(50),nrmsi2(50);
for(int p=0;p<50;++p){
nrmsi[p]=gg[o];
nrmsi1[p]=gg1[o];
nrmsi2[p]=gg2[o];
o=o+7;
nrmsi[p]=nrmsi[p]*0.7071;
nrmsi1[p]=nrmsi1[p]*0.7071;
nrmsi2[p]=nrmsi2[p]*0.7071;
//qDebug() << nrms;
}
for(int n=1;n<50;n++){
sumi=sumi+pow(nrmsi[n],2);
sumi1=sumi1+pow(nrmsi1[n],2);
sumi2=sumi2+pow(nrmsi2[n],2);
tihd=((sqrt(sumi))/nrmsi[0])*100;
tihd1=((sqrt(sumi1))/nrmsi1[0])*100;
tihd2=((sqrt(sumi2))/nrmsi2[0])*100;
}
ui->thd->setText(QString::number(thd));
ui->thd1->setText(QString::number(thd1));
ui->thd2->setText(QString::number(thd2));
ui->tihd->setText(QString::number(tihd));
ui->tihd1->setText(QString::number(tihd1));
ui->tihd2->setText(QString::number(tihd2));
if(ui->Cfa->isChecked()){
ui->plot->graph(0)->setData(freq,gg);}
ui->plot->graph(0)->rescaleAxes();
if(ui->Cfb->isChecked()){
ui->plot->graph(1)->setData(freq,gg1);}
ui->plot->graph(1)->rescaleAxes();
if(ui->Cfc->isChecked()){
ui->plot->graph(2)->setData(freq,gg2);}
ui->plot->graph(2)->rescaleAxes();
interfaz.cpp 37
ui->plot->setInteractions(QCP::iRangeDrag | QCP::iRangeZoom |
QCP::iSelectPlottables);
ui->plot->replot();
}
}
void Interfaz::on_actionDisturbios_Detectados_triggered()
{
ui->textEdit->setVisible(true);
ui->Lthd->setVisible(false);
ui->Lthd1->setVisible(false);
ui->Ltihd->setVisible(false);
ui->Ltihd1->setVisible(false);
ui->thd->setVisible(false);
ui->thd1->setVisible(false);
ui->thd2->setVisible(false);
ui->tihd->setVisible(false);
ui->tihd1->setVisible(false);
ui->tihd2->setVisible(false);
ui->Lrms->setVisible(false);
ui->Lfp->setVisible(false);
```

```
ui->Lpp->setVisible(false);
ui->Lpq->setVisible(false);
ui->Lps->setVisible(false);
ui->LA->setVisible(false);
ui->LB->setVisible(false);
ui->LC->setVisible(false);
ui->rms->setVisible(false);
ui->rms1->setVisible(false);
ui->rms2->setVisible(false);
ui->fp->setVisible(false);
ui->fp1->setVisible(false);
ui->fp2->setVisible(false);
ui->Pp->setVisible(false);
ui->Pp1->setVisible(false);
ui->Pp2->setVisible(false);
ui->Pq->setVisible(false);
ui->Pq1->setVisible(false);
ui->Pq2->setVisible(false);
ui->Ps->setVisible(false);
ui->Ps1->setVisible(false);
ui->Ps2->setVisible(false);
ui->plot->setVisible(false);
ui->horizontalScrollBar->setVisible(false);
ui->groupBox->setVisible(false);
ui->label->setVisible(false);
ui->label_2->setVisible(false);
ui->Cfa->setVisible(false);
ui->Cfc->setVisible(false);
ui->Cfb->setVisible(false);
interfaz.cpp 38
ui->comboBox->setVisible(false);
//ui->Bgraf->setGeometry(730,40,181,23);
ui->text->setVisible(true);
ui->Blimp->setVisible(true);
ui->Ccfa->setVisible(false);
ui->Ccfb->setVisible(false);
ui->Ccfc->setVisible(false);
ui->label_3->setVisible(false);
}
void Interfaz::on_thd_textChanged(const QString &arg1)
{
QString thd = ui->thd->text();
if(thd.toDouble()>5){
ui->textEdit->append(QString::number(cc)+") "+"Codigo 01_A: valor de THD
de la fase A es superior al 5% : " + thd + "\n");
cc++;
}
}
void Interfaz::on_thd1_textChanged(const QString &arg1)
{
QString thd1 = ui->thd1->text();
if(thd1.toDouble()>5){
ui->textEdit->append(QString::number(cc)+") "+"Codigo 01_B: valor de THD
de la fase B es superior al 5% : " + thd1 + "\n");
cc++;
}
```

```
}
void Interfaz::on_thd2_textChanged(const QString &arg1)
{
QString thd2 = ui->thd2->text();
if(thd2.toDouble()>5){
ui->textEdit->append(QString::number(cc)+") "+"Codigo 01_C: valor de THD
de la fase C es superior al 5% : " + thd2 + "\n");
cc++;
}
}
void Interfaz::on_rms_textChanged(const QString &arg1)
{
QString rms = ui->rms->text();
if(rms.toDouble()>256){
ui->textEdit->append(QString::number(cc)+") "+"Codigo 02_A: valor de RMS
de la fase A es superior al 10% : " + rms + "\n");
cc++;
}
if(rms.toDouble()<210){
ui->textEdit->append(QString::number(cc)+") "+"Codigo 03_A: valor de RMS
de la fase A es Inferior al 10% : " + rms + "\n");
cc++;
interfaz.cpp 39
}
}
void Interfaz::on_rms1_textChanged(const QString &arg1)
{
QString rms1 = ui->rms1->text();
if(rms1.toDouble()>256){
ui->textEdit->append(QString::number(cc)+") "+"Codigo 02_B: valor de RMS
de la fase B es superior al 10% : " + rms1 + "\n");
cc++;
}
if(rms1.toDouble()<210){
ui->textEdit->append(QString::number(cc)+") "+"Codigo 03_B: valor de RMS
de la fase B es Inferior al 10% : " + rms1 + "\n");
cc++;
}
}
void Interfaz::on_rms2_textChanged(const QString &arg1)
{
QString rms2 = ui->rms2->text();
if(rms2.toDouble()>256){
ui->textEdit->append(QString::number(cc)+") "+"Codigo 02_C: valor de RMS
de la fase C es superior al 10% : " + rms2 + "\n");
cc++;
}
if(rms2.toDouble()<210){
ui->textEdit->append(QString::number(cc)+") "+"Codigo 03_C: valor de RMS
de la fase C es Inferior al 10% : " + rms2 + "\n");
cc++;
}
}
void Interfaz::on_fp_textChanged(const QString &arg1)
{
QString fp = ui->fp->text();
```

```
if(fp.toDouble()<0.9){
ui->textEdit->append(QString::number(cc)+") "+"Codigo 04_A: valor de FP
de la fase A es Inferior a 0.9 : " + fp + "\n");
cc++;
}
}
void Interfaz::on_fp1_textChanged(const QString &arg1)
{
QString fp1 = ui->fp1->text();
if(fp1.toDouble()<0.9){
ui->textEdit->append(QString::number(cc)+") "+"Codigo 04_B: valor de FP
de la fase B es Inferior a 0.9 : " + fp1 + "\n");
cc++;
}
}
interfaz.cpp 40
void Interfaz::on_fp2_textChanged(const QString &arg1)
{
QString fp2 = ui->fp2->text();
if(fp2.toDouble()<0.9){
ui->textEdit->append(QString::number(cc)+") "+"Codigo 04_C: valor de FP
de la fase C es Inferior a 0.9 : " + fp2 + "\n");
cc++;
}
}
void Interfaz::on_actionEnvolvente_triggered()
{
QVector <double> max(6);
QVector <double> max1(6);
QVector <double> max2(6);
QVector <double> tem(8192),tem1(8192),tem2(8192);
QString d=ui->comboBox->currentText();
QVector <double> x(8192);
ui->plot->addGraph();
ui->plot->graph(6)->setPen(QPen(Qt::darkBlue));
ui->plot->addGraph();
ui->plot->graph(7)->setPen(QPen(Qt::darkRed));
ui->plot->addGraph();
ui->plot->graph(8)->setPen(QPen(Qt::darkGreen));
ui->plot->graph(6)->data()->clear();
ui->plot->graph(7)->data()->clear();
ui->plot->graph(8)->data()->clear();
//int N=8000;
double ven=d.toInt()/5;
double num=d.toInt()/ven;
for(int i=0;i<=num;++i){
for(int j=ven*i;j<=ven*(i+1);++j){
if(Data[j]>max[i]){
max[i]=Data[j];
}
if(Data1[j]>max1[i]){
max1[i]=Data1[j];
}
if(Data2[j]>max2[i]){
max2[i]=Data2[j];
}
```

```
}
//x.append(i);
}
for(int k =0;k < d.toInt()/5;++k){
tem[k]=max[0];
tem1[k]=max1[0];
tem2[k]=max2[0];
}
interfaz.cpp 41
for(int k = (d.toInt()/5);k < (d.toInt()/5)*2;++k){
tem[k]=max[1];
tem1[k]=max1[1];
tem2[k]=max2[1];
}
for(int k = (d.toInt()/5)*2;k < (d.toInt()/5)*3;++k){
tem[k]=max[2];
tem1[k]=max1[2];
tem2[k]=max2[2];
}
for(int k = (d.toInt()/5)*3;k < (d.toInt()/5)*4;++k){
tem[k]=max[3];
tem1[k]=max1[3];
tem2[k]=max2[3];
}
for(int k = (d.toInt()/5)*4;k < (d.toInt()/5)*5;++k){
tem[k]=max[4];
tem1[k]=max1[4];
tem2[k]=max2[4];
}
for(int l=0;l<=d.toInt();++l){
x[l]=l;
}
qDebug () << max;
ui->plot->graph(6)->setData(x,tem);
ui->plot->graph(7)->setData(x,tem1);
ui->plot->graph(8)->setData(x,tem2);
ui->plot->graph(6)->rescaleAxes();
ui->plot->graph(7)->rescaleAxes();
ui->plot->graph(8)->rescaleAxes();
ui->plot->setInteractions(QCP::iRangeDrag | QCP::iRangeZoom |
QCP::iSelectPlottables);
ui->plot->replot();
ui->cont2->setText("1");
}
void Interfaz::on_actionGuardar_triggered()
{
QFile arch;
QTextStream io;
QString nomarch;
nomarch = QFileDialog::getSaveFileName(this, "Guardar");
arch.setFileName(nomarch);
arch.open(QIODevice::WriteOnly | QIODevice::Text);
if(!arch.isOpen()){
QMessageBox::critical(this, "Error", arch.errorString());
return;
}
```

```
io.setDevice(&arch);
io << ui->textEdit->toPlainText();
}
void Interfaz::on_Blimp_clicked()
interfaz.cpp 42
{
ui->textEdit->setPlainText("");
cc=0;
}
```

main.cpp 1

```
#include "interfaz.h"
#include <QApplication>

int main(int argc, char *argv[])
{
    QApplication a(argc, argv);
    Interfaz w;
    w.show();

    return a.exec();
}
```