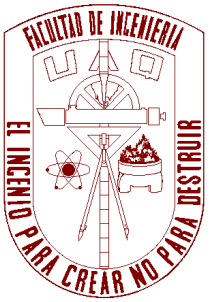
Universidad Autónoma de Querétaro

Facultad de ingeniería

Campus San Juan del Río



Software de procesamiento de señales eléctricas trifásicas

TESIS

Que como parte de los requisitos para obtener el grado de

Maestría en ciencias (Mecatrónica)

Presenta:

Walter Alexander Dzul Ayala

Asesores:

Dr. Luis Morales Velázquez

San Juan del Río, Querétaro. Julio de 2016

Universidad Autónoma de Querétaro
Facultad de Ingeniería
Especialidad en Mecatrónica

Software de procesamiento de señales eléctricas trifásicas

Opción de titulación
Tesis

Que como parte de los requisitos para obtener el Grado de
Maestría en Ciencias (Mecatrónica)

Presenta:
Walter Alexander Dzul Ayala

Dirigido por:
Dr. Luis Morales Velázquez

Dr. Luis Morales Velázquez Presidente	Firma
Dr. Oscar Duque Pérez Secretario	Firma
Dr. Roque Alfredo Osornio Ríos Vocal	Firma
Dr. Miguel Trejo Hernández Suplente	Firma
Rene Romero Troncoso Suplente	Firma

Dr. Aurelio Domínguez Gonzáles
Director de la Facultad

Dra. Ma. Guadalupe Flavia Loarca-Piña
Directora de Investigación y Posgrado

Centro Universitario
Querétaro, Qro.
Fecha Noviembre 2016

RESUMEN

El monitoreo de energía eléctrica permite el detectar e identificar perturbaciones en una red eléctrica como perdidas y picos de voltaje, harmónicos y ruido de alta frecuencia, regularmente vistos en aplicaciones de tablero industrial y comercial. El presente trabajo de tesis expone una interfaz gráfica basada en c/c++. Y de plataforma libre, que tiene como finalidad el manejo de grandes cantidades de información recabadas por la tarjeta de adquisición de datos propietaria PQ-UAQ para su posterior visualización y procesamiento. Cada una de las funciones de la interfaz mencionada fueron desarrolladas con la finalidad de satisfaces las necesidades establecidas por regulaciones internacionales de monitoreo de señales eléctricas. También se tomaron en cuenta limitaciones se procesamiento y almacenamiento presentadas por las especificaciones de hardware de los equipos de cómputo utilizados. La interfaz fue probada con señales eléctricas trifásicas capturadas en intervalos de tiempo entre 24 horas y una semana. Realizadas en instalaciones de alto consumo energético.

Palabras clave: Estructuras de datos, adquisición de datos, conversión de datos, manejo de datos, procesamiento de señales, visualización de datos, calidad de la energía, análisis de datos.

SUMMARY

Electric energy monitoring allows the detection and identification of disturbances in an electric grid, such as sags, swells, harmonics and high frequency noise, regularly seen in industrial applications. This thesis work exposes a free platform graphic interface based in c/c++. Which has the objective of managing large quantities of information captured by the proprietary acquisition data card PQ-UAQ for subsequent visualization and processing. Each function of this interface were developed based to meet the international electric signal monitoring regulations. Processing and storage limitations were also taken in account. The interface was tested with electrical signals captured in intervals of 24 hours and a week. This captures were done in installations of high energetic consumption.

Key words: Data structures, data acquisition, data conversion, data handling, signal processing, data visualization, power quality, data analysis.

AGRADECIMIENTOS

A la Universidad Autónoma de Querétaro por brindarme las instalaciones para realizar mis estudios.

Al consejo nacional de ciencia y tecnología por apoyarme económicamente durante mis estudios de posgrado.

A mis profesores, sinodales y asesores que me brindaron su tiempo, conocimiento y evaluación crítica para realizar mi trabajo.

ÍNDICE GENERAL

ÍNDICE DE FIGURAS

ÍNDICE DE CUADROS

1 INTRODUCCIÓN

La necesidad del monitoreo de redes eléctricas ha incrementado en las recientes décadas como resultado del incremento del uso de elementos de alta precisión en la industria. La funcionalidad y vida útil e estos elementos se ve afectada por perturbaciones eléctricas, causando fallas de lectura y funcionamiento significando pérdidas económicas. Por lo cual varias compañías e instituciones académicas en todo el mundo desarrollan nuevas técnicas para detectar, clasificar y diagnosticar estas perturbaciones llamadas eventos de calidad de la energía. Con el objetivo de monitorear señales se desarrolló una interfaz de usuario que manejara la información capturada por la tarjeta de adquisición de datos propietaria PQ-UAQ. Visualizar y procesar dicha información utilizando metodologías utilizadas para la detección y clasificación de eventos de calidad de la energía con el fin de facilitar el estudio de las señales capturadas.

Este trabajo de tesis se encuentra dividido en capítulos con contenidos específicos. En el primer capítulo se abordan antecedentes donde se mencionan trabajos realizados previamente dentro y fuera de la institución. En este capítulos se presentan los objetivos generales y específicos, así como la hipótesis y justificación que dan las razones para el desarrollo de este trabajo. Por último se presenta un diagrama del planteamiento general con el cual se pretenden alcanzar los objetivos. En el segundo capítulo se presenta la fundamentación teórica, donde se describen las herramientas (algoritmos, software, estándares, matemáticas) utilizadas para realizar este trabajo. El tercer capítulo describe la metodología donde se describen a detalle el procedimiento realizado para alcanzar cada uno de los objetivos propuestos. En el cuarto capítulo se presentan los resultados del trabajo obtenidos, se discute si los objetivos fueron alcanzados y si la hipótesis planteada fue correcta. En el quinto capítulo se presentan las conclusiones obtenidas y el trabajo futuro que se considera puede llegar a realizarse.

1.1 ANTECEDENTES

En cuanto a los trabajos de calidad de la energía García-Hernández *et al*.(2013) determinaron que la calidad de la energía juega un papel importante en el rendimiento de sistemas electrónicos. Establecen que no es un concepto nuevo, pero que en años recientes su aplicación en diversas áreas ha resuelto distintos problemas que tienen que ver con la operación errónea de aplicaciones específicas de equipo electrónico, incluso ha ayudado a prevenir daños totales o parciales de equipos electrónicos sensibles. En instalaciones eléctricas donde los problemas de calidad de energía no son considerados, los problemas eléctricos y pérdidas económicas pueden ser considerables. Rodríguez *et al*.(2012) mencionan que se han propuesto varias técnicas de clasificación de eventos de calidad de la energía. Estas técnicas son similares en el aspecto que la señal tiene que ser filtrada por medio de un procesamiento de señales, para que posteriormente los datos procesados sean evaluados en un módulo de identificación y clasificación.

A continuación se mencionan algunos trabajos de monitoreo de calidad de la energía y procesamiento de señales locales, nacionales e internacionales.

En la Universidad Autónoma de Querétaro se han desarrollado métodos de detección de fallas en sistemas electrónicos y de potencia cuya repercusión ha sido a nivel internacional. Valtierra-Rodriguez *et al*.(2013) desarrollaron una metodología de monitoreo armónico a través de señales estacionarias y transitorias que satisface el estándar IEC61000-4-7. La tecnología fue implementada en un FPGA (Field Programmable Gate Array, Arreglo de Compuertas Programable de Campo) para un monitoreo continuo en línea. Romero-Troncoso *et al.*(2011) presentaron una nueva metodología que mezcla el análisis de información entrópica con lógica difusa para identificar fallas y defectos en motores de inducción. Dicha metodología es implementada a través de un FPGA. Cabal-Yepez *et al*.(2013) presentaron el diseño e implementación de un SoC (Sistem on a Chip, Sistema En un Chip) para el análisis de equipo industrial a través de STFT (short-time Fourier transform, Transformadas de Fourier en Tiempo Reducido) y transformaciones wavelet. Granados-Lieberman *et al*.(2013)realizaron un estudio que correlaciona las bajas de tensión en el maquinado de piezas en una máquina

CNC, denotando la velocidad y profundidad de corte. Así como condiciones de maquinado inadecuadas como errores de control o piezas de maquinado defectuosas. Valtierra-Rodriguez *et al*.(2013) desarrollaron una metodología basada en redes neuronales para detectar y clasificar perturbaciones de potencia individuales y combinadas, tales como bajadas, subidas y cortes de voltaje así como harmónicos e inter-armónicos. Granados-Lieberman *et al.*(2013) propusieron un sensor inteligente que fuera capaz de detectar, clasificar y cuantificar perturbaciones de calidad de la energía.

En el ámbito nacional en general se han desarrollado también metodologías y aplicaciones para mejorar la calidad de energía en sistemas eléctricos y de potencia. Moreno*et al*.(2014) describieron un algoritmo simple que puede detectar y aislar perturbaciones de voltaje con el mismo tiempo de muestreo usado en mediciones de estado estable. Establecen que si se implementara en un PQA (Power Quality Analyzer, Analizador de Calidad de Energía) se podrían grabar todos los tipos de perturbaciones en estado estable utilizando los algoritmos en el PQA sin incremento substancial de carga de procesamiento. García-Hernández *et al*.(2013) propusieron el monitoreo constante de parámetros de calidad de energía como una manera de conservar o mejorar la resolución de sistemas, evitando la influencia de ruido extrínseco excesivo. González *et al.*(2014) presentaron una mejor y factible estrategia de control para DVR (Dynamic Voltage Restorer, Restauradores de Voltaje Dinámicos) utilizando PBC (Passivity Based Control, Control Basado en Pasividad) para una respuesta más rápida que en los controladores clásicos, y que además no presentaba sobretiros. Cisneros-Magaña *et al*.(2014) propusieron una metodología que evaluaba la estimación de estado de la calidad de la energía utilizando el método de END (Enhanced Numeric Difference, Diferenciación Numérica Mejorada), el cual obtenía la solución en estado estable periódica de una red de energía, con el fin de determinar la estimación harmónica de condiciones de harmónicos variantes en el tiempo y la estimación de fallas transitorias.

En el ámbito internacional se han desarrollado diversos métodos de detección y clasificación de perturbaciones eléctricas, aunque al ser la mayoría de naturaleza experimental, muchos no concuerdan con los estándares de la IEEE, (2009) y EN 50160, (2004). También se investiga el efecto del monitoreo de calidad de energía en distintos

ámbitos energéticos. Ferreira *et al.*(2015) proponen un método basado en el análisis de canales independientes para la clasificación de perturbaciones de calidad de energía. El método consiste en descomponer el sistema de energía en varios componentes los cuales son clasificados, para que al final sean clasificadas 5 clases de perturbaciones singulares y doce perturbaciones múltiples, cada una con un 97% de eficiencia. *Lima et al*.(2008) propusieron una arquitectura de software-hardware para detección, clasificación y caracterización de eventos de calidad de energía, utilizando wavelets en todos los algoritmos. Con el fin de crear una arquitectura factible para implementación como parte integral de sistemas de supervisión. Patsalides*et al*.(2012) discuten sobre las posibles limitaciones en la generación de energía fotovoltaica debido a las limitaciones de calidad de la energía. Shareef *et al*.(2013) propusieron una técnica para visualizar y detectar eventos de perturbaciones de calidad de energía. Su técnica estaba basada en procesamiento de imagen usando imágenes binarias e imágenes en escala de grises, capturando la imagen de gris de una forma de onda del voltaje y procesándola, argumentando que es posible detectar eventos de calidad de energía. Dehghani *et al.*(2013)determinaron un nuevo acercamiento para la clasificación de perturbaciones de energía usando el HMM (Hidden Markov Model, Modelo Oculto de Márkov) y WT (Wavelet Transforms, Trasformaciones Wavelet). En su trabajo evaluaron matrices de entrenamiento HMM las distribuciones de energía obtenidas a través de las transformaciones wavelet con el fin de maximizar la precisión de clasificación. Jaya-Bharata-Reddy *et al*.(2013) presentaron un acercamiento para la clasificación de eventos de perturbaciones de calidad de la energía utilizando la DOST (Discrete, Orthogonal, S Transform, Transformada de Laplace Ortogonal Discreta) debido a su eficiencia en comparación con las otras transformadas en casos particulares. Liberado *et al*.(2015) definieron una manera de determinar un compensador de calidad de energía con una relación eficiencia-costo aceptable, generando un tomador de decisiones difuso que se basa en un historial de decisiones tomadas y comparándolas con sus respectivos resultados.

Es evidente que lo más recurrente en trabajos que conciernen a calidad de energía es crear nuevos métodos de diagnóstico y clasificación a un nivel básico. Sin

embargo, se requiere aun la generación de herramientas de bajo costo y que estén basadas en los estándares internacionales, utilizando herramientas clásicas de diagnóstico y clasificación de eventos de calidad de energía, además de una interfaz amigable para el usuario con el fin de facilitar su operación.

La utilización de GUI (Graphical User Interface, interfaz gráfica de usuario), es uno de los aspectos fundamentales en cualquier desarrollo, pues su utilización en conjunto con hardware diseñado para cumplir con la tarea de detectar y clasificar perturbaciones en una red de energía para luego desplegarla de manera procesada, es un aspecto que, aunque no reciente, es de un costo elevado en la industria.

A continuación se analizarán algunos ejemplos de interfaces gráficas de usuario en conjunto con hardware FPGA. Agrawal y Prakash (2014) utilizaron un FPGA como unidad de procesamiento de un juego de sensores en un robot, conectando las lecturas de dichos sensores con una interfaz gráfica desarrollada en el lenguaje de programación Processing por medio de protocolo serial RS-232. Ravanasa (2014) utilizó una interfaz gráfica basada en RDP (MS Windows Remote Control Desktop, Plataforma de Control Remoto de Windows) para acceder a los datos de laboratorio recabados por un FPGA de forma remota. Deepa (2013) controló un dispositivo de seguimiento solar basado en el control de un motor a pasos con PWM implementado en FPGA, cuyos parámetros de control son determinados por una interfaz de usuario basada en la plataforma LabVIEW por medio de BLUETOOTH con el fin de tener un control remoto de alta precisión. Khan (2014) describió un sistema de interfaz gráfica de usuario, la cual era capaz de captar y monitorear un juego de sensores, y a la vez controlar un número determinado de máquinas por medio de procesamiento paralelo, con el fin de implementar dicha interfaz en sistemas de control en la industria. Jayarajan *et al*.(2013) diseñaron un telémetro laser basado en FPGA en conjunto con una interfaz gráfica de monitores basada en LabVIEW con el fin de ofrecer una alternativa de bajo costo. Callejas *et al*.(2013) implementaron una red neuronal artificial en un FPGA para obtener la resolución de trayectorias tipo laberinto, con un despliegue de información gráfica en una interfaz basada en MatLab. Con el fin de analizar de manera visual los efectos de aprendizaje del móvil al pasar por el laberinto. Horvat *et al*.(2014)diseñaron un sistema de control por eventos de un

sistema mecatrónico, utilizando un FPGA como centro de procesamiento y una interfaz gráfica como centro de configuración y despliegue de información basada en PASCAL. El objetivo de este trabajo fue proponer un sistema robusto, seguro y de control transparente para el control por eventos de un sistema. Akiyama *et al*.(2011) presentan un trabajo con el objetivo de desarrollar un generador de pulsos de alto rendimiento usando un FPGA y una interfaz gráfica basada en lenguaje de programación de alto nivel C#, con el fin de ofrecer un generador de pulsos con aplicaciones para los campos de esterilización y tratamiento de aguas. Gilliland *et al*.(2014) presentaron la implementación de un FPGA para procesar los datos de pruebas no invasivas ultrasónicas, las cuales generaron un gran volumen de información, en conjunto con una interfaz gráfica escrita en C++/Qt y OpenGL para el despliegue y análisis de información procesada.

Un requerimiento muy importante para asegurar la calidad de energía en sistemas de distribución es el monitoreo rendimiento de sistemas de potencia. El monitoreo de calidad e energía no es una tarea fácil, y generalmente incluye la instrumentación de hardware y software sofisticado. Un punto clave es el análisis de grandes cantidades de datos con la mínima intervención de herramientas expertas de calidad de energía. Por lo tanto es deseable desarrollar herramientas para el análisis automatizado de clasificación de eventos de calidad de la energía

1.2 DESCRIPCIÓN DEL PROBLEMA

Las señales eléctricas capturadas por la tarjeta de adquisición de datos propietaria PQ-UAQ son almacenadas sin formato en una tarjeta SD, la interfaz gráfica desarrollada tiene como tarea decodificar la información almacenada. Y formatearla para su posterior lectura y procesamiento.

Las señales capturadas representan vectores de información que pueden alcanzar decenas de Gigabytes. Por lo cual es necesaria una metodología de indexado y fragmentación de información para facilitar el manejo de las capturas. También es necesario considerar que los vectores de información requieren un espacio de almacenamiento considerable. Por lo que es necesario utilizar una técnica de almacenamiento eficiente.

El procesamiento de la información capturada requiere cargar a memoria volátil vectores de información capturada para realizar las operaciones y procesamiento de señales deseado. Por lo que se necesita una metodología de procesamiento que considere las limitaciones de hardware de la computadora a utilizar.

Graficar tanto los datos de captura como los datos procesados de grandes cantidades de información representan una carga considerable para el procesador por lo que un método de graficado de información eficiente debe ser implementado.

1.3 HIPÓTESIS Y OBJETIVOS

1.3.1 HIPÓTESIS

Mediante Módulos de visualización y procesamiento integrados en una plataforma de software libre, será posible analizar al menos 3 disturbios eléctricos comunes a un costo económico significativamente menor que el de las plataformas comerciales existentes.

1.3.2 OBJETIVOS

Objetivo general

Desarrollar una herramienta de software libre para la visualización y manipulación de señales eléctricas trifásicas.

Objetivos particulares

1. Determinar los requerimientos de la aplicación que deberán ser tomados en cuenta para el diseño basado en software libre.
2. Desarrollar los elementos de software básicos para la visualización de señales sintéticas a través de software libre.
3. Generar al menos tres modos diferentes de visualización. Gráficas de señales en el tiempo, herramientas estadísticas y gráficas polares para el despliegue de información de las señales eléctricas.
4. Generar el código de la interfaz gráfica que haga uso de tres metodologías para el análisis de señales eléctricas a través de la herramienta de software libre Qt.

1.4 JUSTIFICACIÓN

Este trabajo surge del hecho que los dispositivos de monitoreo de calidad de la energía disponibles en el tienen precios muy elevados y de la propuesta de generar un programa capaz de analizar por lo menos 3 disturbios eléctricos haciendo uso de una computadora con especificaciones de hardware de media gama.

Un aspecto importante del funcionamiento de la tarjeta de adquisición de datos propietaria PQ-UAQ es captura la señal eléctrica completa. Los dispositivos de monitoreo realizan el análisis de la señal eléctricas en tiempo real. Detectando eventos de calidad de la energía y generando una bitácora de eventos para mostrar al usuario. La interfaz gráfica presentada en este trabajo tiene como objetivo el permitir su modificación para que desarrolladores puedan agregar metodologías propias de procesamiento de señales con el fin de desarrollar nuevas técnicas de detección de eventos de calidad de la energía o técnicas conocidas optimizadas para esta plataforma con fines académicos y de investigación.

1.5 PLANTEAMIENTO GENERAL

La figura 1 describe la estructura general de la interfaz gráfica a desarrollar. Dividiendo la estructura general en 3 principales bloques que representan las principales funciones de la interfaz a desarrollar.

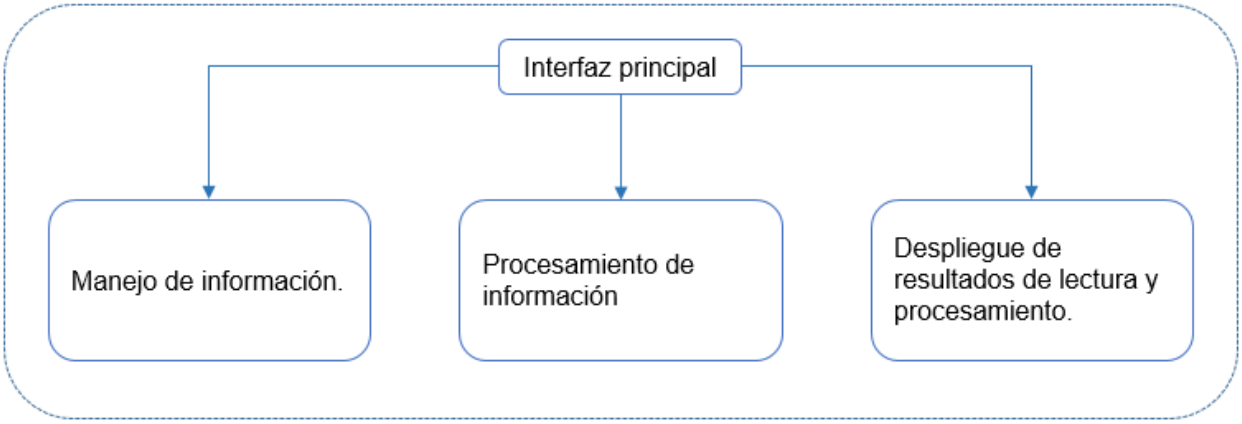


Figura 1 Estructura general del programa a desarrollar.

Para realizar los objetivos propuestos y comprobar la hipótesis planteada, los bloques de la figura 1 están sujetos a consideraciones de tiempo de procesamiento, consumo de recursos computacionales y normativas internacionales de procesamiento de señales para la calidad de la energía.

En el primer bloque de la figura 1 se presenta la función de manejo de datos. Esta función debe ser desarrollada tomando en cuenta la estructura de información recabada por la tarjeta de adquisición de datos propietaria PQ-UAQ para su decodificación y formato. Los vectores de información deben ser manejables así que se propone una metodología de fragmentación y formateo de información tomando como referencia para esta tarea las normativas internacionales de calidad de la energía.

En el segundo bloque se propone una función de lectura de información de captura generada por la interfaz o una señal sintética generada por otra plataforma, tomando en consideración los recursos computacionales equipos de media gama para su ejecución. Las funciones de procesamiento a utilizadas se seleccionan en base técnicas de procesamiento de señales para la detección, clasificación y diagnóstico de eventos de calidad de la energía utilizados en el ámbito industrial y de investigación.

En el tercer bloque se presentan funciones de graficado de información tanto en bruto como procesada, tomando en consideración nuevamente las limitaciones de procesamiento del equipo de cómputo para su ejecución. También se presentan al menos 3 disturbios detectables mediante estos procesos de procesamiento.

2 FUNDAMENTACIÓN TEÓRICA

2.1 ESTADO DEL ARTE

La filosofía del monitoreo de la calidad de energía ha evolucionado de un objetivo de operatividad a múltiples procesos, asumiendo un estándar legal, determinando prácticas y estableciendo una línea base predictiva. En conjunción con los cambios de objetivo, el monitoreo de equipo ha incorporado numerosos avances tecnológicos en las áreas de comunicación, captura de datos, tamaño físico y costo. Como los datos se han estado haciendo disponibles más rápido, el deseo por usar esa información para alcanzar ciertos objetivos resulta natural. Con el incremento del énfasis en la confiabilidad, la necesidad de asegurar tanto la precisión como la profundidad de medición se ha vuelto un objetivo muy importante.

En la década de los 70 se desarrollaron, el voltímetro y la cinta perforada y se introducen en el mercado los ingenieros de servicio de campo. En la década de los 80 se desarrollaron el osciloscopio y las gráficas en el tiempo y se introducen en el mercado los grupos de utilidad de calidad de la energía. En la década de los 90 se desarrollaron, el procesamiento digital de señales, la integración de PC, el almacenamiento masivo de datos y la comunicación de Ethernet, se introducen en el mercado las compañías de utilidad, las plantas e instalaciones industriales de calidad de energía, así como ingenieros y consultores (Khan, 2001). En la década del 2000 se desarrollan el reconocimiento de patrones, el minado de datos, toma de decisiones, la utilización del internet como herramienta de control remota. Se introducen en el mercado las agencias regulatorias, ISO, y la IPP.

Un esquema general de la clasificación de sistema está definido por los siguientes estados.

Pre procesamiento. En este estado se realiza la estimación de componentes, luego se implementa un algoritmo para la segmentación de la señal en diferentes etapas.

Etapa de extracción. Generalmente se utilizan wavelets para cuantificar características de diferentes tipos de eventos de sistemas de energía.

Clasificación. Basada en reglas definidas, es la información de sistemas expertos utilizada para la discriminación de diferentes tipos de eventos.

Toma de decisiones. En esta etapa el tipo de evento es asignado a un evento actual. En muchos casos las etapas de clasificación y toma de decisiones se combinan en un proceso utilizando redes neuronales, lógica difusa, bayesianos o técnicas de reconocimiento de patrones (Flores, 2002).

2.2 CALIDAD E LA ENERGÍA

La calidad de la energía es el conjunto de límites y propiedades eléctricas que permiten la funcionalidad deseada de sistemas eléctricos sin pérdidas significativas de rendimiento o vida útil. El término se usa para describir el potencial eléctrico que controla una carga eléctrica, y la capacidad de dicha carga para funcionar apropiadamente con este potencial eléctrico.

El proceso de desregularización que está teniendo lugar en la industria de potencia eléctrica alrededor del mundo, provocará problema de calidad de energía (Power Quality, PQ) esto lleva a los consumidores a considerarla como un concepto muy preocupante.

En marzo del 2009 el instituto de ingeniería Eléctrica y Electrónica (IEEE) aprobó el estándar para el monitoreo de calidad de energía IEEE, (2009), esta regularización propuesta tiene como fin reducir las perturbaciones generadas por componentes

electrónicos, las cuales deterioran la calidad de la energía y por consiguiente generan pérdidas económicas.

2.2.1 NORMATIVAS

Existen numerosos estándares de monitoreo y medición de calidad de la energía. A continuación se presentan dos estándares oficiales más recurrentes en el ámbito de calidad de la energía.

2.2.1.1 ESTANDARES IEEE1159 y EN 50160

Estas normas determinan parámetros principales de voltaje, y sus rangos de variación permisible. Desde un punto de voltaje público tanto bajo como medio en sistemas de distribución de energía bajo condiciones de operación normales. Se presentan prácticas recomendadas para el monitoreo de características eléctricas de sistemas de energía de una fase y poli-fase. Incluye descripciones consistentes de fenómenos conducidos y variaciones que ocurren en un sistema de energía. Esta práctica describe condiciones y variaciones nominales que pueden ser originadas dentro de la fuente o carga del equipo, o de interacciones entre la fuente y la carga. Esta norma discute dispositivos de monitoreo y técnicas de interpretación de resultado de monitoreo.

2.2.2 EVENTOS DE CALIDAD DE LA ENERGÍA

Categorías	Contenido espectral típico	Duración típica	Magnitud de voltaje típica
1. Transitorios			
1.1 Impulsivos			
1.1.1 Nanosegundos	Impulso de 5ns	< 50 ns	
1.1.2 Microsegundos	Impulso de 1 µs	50 ns – 1 s	
1.1.3 Milisegundo	Impulso de 0.1 ms	> 1 ms	
1.2 Oscilaciones			
1.2.1 Baja frecuencia	< 5 Hz	0.3 - 50 ms	0 – 4 pu (per unit)
1.2.2 Media frecuencia	5-500 kHz	20 µs	0 – 8 pu
1.2.3 Alta frecuencia	0.5-5 MHz	5 µs	0 – 4 pu
2 Variaciones rms de corto tiempo			
2.1 instantáneos			
2.1.1 sag (baja)		.05 – 30 ciclos	0.1 – 0.9 pu
2.1.2 swell (alta)		.05 – 30 ciclos	1.1 – 1.8 pu
2.2 Momentáneos			
2.2.1 interrupciónes		.05 ciclos – 3 s	< .01 pu
2.2.2 bajas		30 ciclos – 3 s	0.1 – 0.9 pu

	2.2.3 altas		30 ciclos – 3 s	1.1 – 1.4 pu
	2.3 Temporales			
	2.3.1 interrupción		> 3 s – 1 min	< 0.1 pu
	2.3.2 bajas		> 3 s – 1 min	0.1 – 0.9 pu
	2.3.3 altas		> 3 s – 1 min	1.1 – 1.2 pu
3	Variaciones rms de larga duración			
	3.1 interrupción sostenida		> 1 min	0.0 pu
	3.2 bajas de voltaje		> 1 min	0.8 – 0.9 pu
	3.3 Sobre voltajes		> 1 min	1.1 – 1.2 pu
	3.4 Sobrecarga de corriente		> 1 min	
4	Desequilibrios			
	4.1 Voltaje		Estado estable	0.5 – 2 %
	4.2 Corriente		Estado estable	1.0 – 30 %
5	Distorsión de formas de onda			
	5.1 Offset de DC		Estado estable	0 – 0.1 %
	5.2 Harmónicos	0-9 kHz	Estado estable	0 – 20 %
	5.3 Interarmónicos	0-9 kHz	Estado estable	0 – 2 %
	5.4 Ruido	Banda ancha	Estado estable	0 – 1 %
6	Fluctuaciones de voltaje	-25 Hz	Intermitente	0.1 – 7 %
7	variaciones de potencia en frecuencia		< 10 s	± .0.10 Hz

Tabla 1 categorías y características disturbios eléctricos

2.3 TARJETA DE ADQUISICIÓN DE DATOS

El dispositivo de monitoreo de calidad de la energía propietario (PQ-UAQ) se compone de un microprocesador basado en FPGA como se aprecia en (1) en la figura 2 y tiene como objetivo capturar señales eléctricas completas y almacenarlas en una tarjeta SD como se muestra en (2) en la figura 2 a una alta frecuencia de manera continua. Las terminales voltajes son conectados directamente a la fuente de energía a medir. Las terminales de corriente requieren pinzas de corriente i400 de Fluke. (FLUKE, 2004) Estas pinzas miden corrientes de CA con una salida de 1 mA/A, están diseñadas para capturar hasta 400 A. los voltajes fueron conectados a cada fase de la red. Cada tarjeta de adquisición de datos utiliza un conjunto de ganancias determinadas por cada canal y dispositivo de medición, estas ganancias deben ser tomadas en cuenta para reinterpretar las señales capturadas de manera correcta. Estas ganancias se encuentran en un archivo de texto .csv para ser utilizado por cualquier plataforma que necesite estos valores.

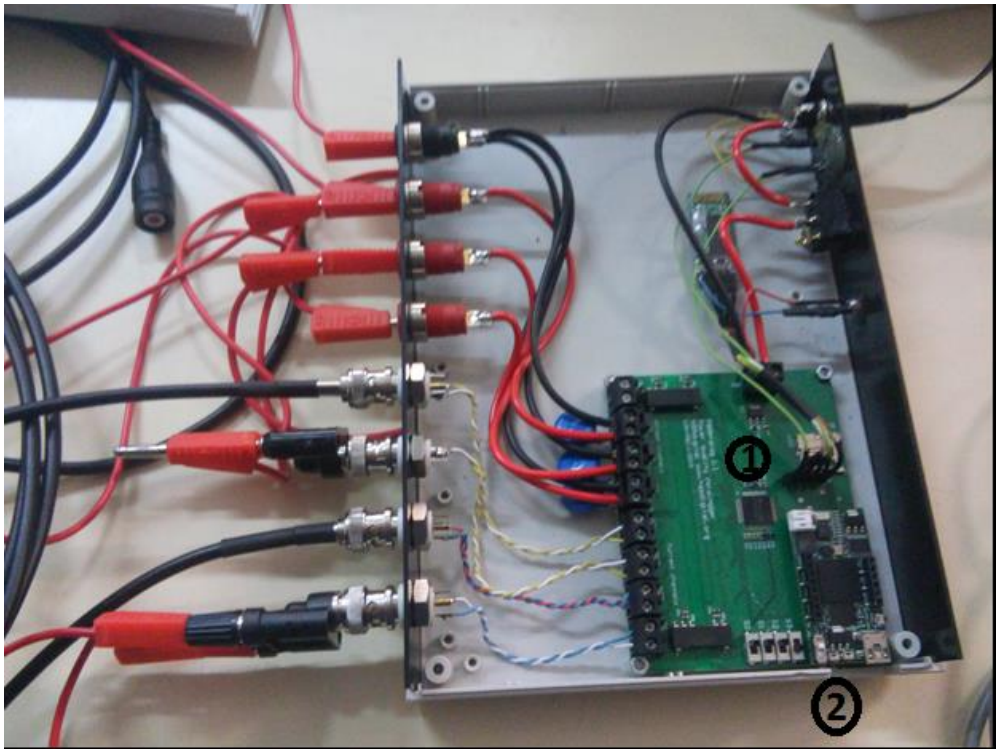


Figura 2 Tarjeta de adquisición de datos

2.3.1 ESTRUCTURA DE DATOS ALMACENADOS EN LA TARJETA SD

El sistema de archivos diseñado para esta aplicación consiste de bloques de 8 KB de información escritos en serie en una tarjeta SD, al comienzo de cada bloque se determina un encabezado de 16 Bytes para su decodificación.

La Tabla 2 muestra los 11 valores determinados necesarios para la codificación de los datos y fragmentación.

El primer valor funciona como una bandera que confirma la locación del encabezado y su estructura. El segundo valor determina el nombre del dispositivo que está realizando la lectura. La señal determina el tipo de datos capturados, ya sean voltajes o corrientes. El canal determina cuál de los 8 canales está realizando la lectura. La secuencia determina permite guardar registro de la continuidad de las lecturas, día, hora, minutos, segundos y milisegundos representan el tiempo real de la captura de datos. La frecuencia de muestreo es la cantidad de datos capturados en un segundo por la tarjeta. 8000 ksps para mantener consistencia.

Los archivos generados después de la decodificación la información en la tarjeta SD son formateados y comprimidos para facilitar su manejo, así como determinar una nueva estructura para los archivos.

Objeto	Estructura	tamaño (Bytes)
Bandera	hexadecimal	1
ID	Número	1
Señal	Número	1
Canal	Número	1
Secuencia	Número	4
Días	Número	1
Horas	Valor BCD	1
Minutos	Valor BCD	1
Segundos	Valor BCD	1
Milisegundoss	Número	2
Frecuencia de muestreo	Número	2

Tabla 2 Estructura de datos de capturas en la tarjeta SD

2.4 PROCESAMIENTO DE SEÑALES

Una señal se define como cualquier cantidad física que varía en el tiempo, espacio o cualquier otra variable independiente. Matemáticamente, una señal se describe como una función de una o más variables independientes. Muchas de las señales encontradas en ingeniería y la ciencia son análogas en naturaleza, esto quiere decir que son continuas en función a una variable, como el espacio o el tiempo y usualmente toman valores en rangos continuos. El procesamiento de señales digitales nos proveen de un método alternativo para procesar las señales análogas, para esto es necesaria la implementación de un convertidor analógico-digital (ADC) el cual se encarga de discretizar una señal con una resolución determinada.

Existen muchas razones por las cuales es preferible el procesamiento digital de una señal análoga al procesamiento de análogo directo. Primero, un sistema programable permite la reconfiguración fácil de operaciones de procesamiento. La reconfiguración de procesamiento de señales análogas conlleva generalmente a

reestructurar el hardware. Las señales digitales son más fáciles de almacenar, sin deterioro o pérdida de fidelidad. En algunos casos el procesamiento digital es más barato que su contraparte analógica (Proakis, 2006).

2.4.1 TÉCNICAS PARA EL PROCESAMIENTO DE SEÑALES

Para el análisis de las señales de voltaje y corriente es necesario conocer las componentes que integran las señales. Para ello existen diversas herramientas que permiten transformar el dominio del tiempo al dominio de la frecuencia. A continuación se presentan se presentan la definiciones matemáticas de las técnicas más recurrentes en el procesamiento de señales en el ámbito de calidad de la energía.

STFT (Short Time Fourier Transform, Transformada de Fourier a tiempo reducido)

La transformada más común para conocer las componentes armónicas en una señal es mediante el uso de la transformada de Fourier. La STFT toma la transformada de Fourier de la señal no estacionaria de tiempo reducido por una ventana de tiempo localizado, lo cual permite conocer el espectro de la señal con respecto de sus intervalos de tiempo. El cuadrado de $\mathrm{STFT}\ \mathrm{SP_x(w,t)}$ es conocido como es espectrograma.

La definición matemática de la STFT se enuncia en la ecuación 1.

$$\mathrm{SP_x(t,w)} = |\mathrm{S_x(t,w)}|^2 = \left|\frac{1}{\sqrt{2\,\pi}}\int_{-\infty}^{\infty} \mathrm{x(t)h(t-\tau)\,e^{-i\omega t}dt}\right|^2 \qquad (1)$$

Donde x(t) es la señal analizada y está localizada en la ventana $\mathrm{h(t)}$ con tiempo cambiante (τ) (Proakis, 2006).

DWT (Discrete Wavelet Transform, Transformada Ondícula Discreta)

Los wavelets son formulados por la familia funciones de base ortogonal cero, lo que significa que describen señales en un formato de tiempo y frecuencia localizados. Las funciones básicas de los Wavelets son generadas por una función singular Wavelet Madre$\mathrm{w(d,t)}$ por dilatación y transformación.

Su definición matemática se enuncia en la ecuación 2

$$\mathrm{W(\tau,d)} = \int_{-\infty}^{\infty} \mathrm{X(t)W(d,t-\tau)dt} \qquad (2)$$

Donde x(t) es la señal analizada y τ es la cantidad de tiempo cambiante (Liu, 2010).

RMS (Root Mean Square, media cuadrática)

Los valores de voltaje y corriente rms están definidos por las ecuaciones en (3) y (4) respectivamente.

$$Vrms = \frac{1}{\sqrt{2}}\sqrt{\sum_{n=1}^{N} V_n^2} \qquad (3)$$

$$Irms = \frac{1}{\sqrt{2}}\sqrt{\sum_{n=1}^{N} I_n^2} \qquad (4)$$

Potencia aparente

Es la potencia total consumida por el circuito, el cual es igual al producto del voltaje y corriente rms.

$$S = Vrms \cdot Irms \qquad (5)$$

Potencia real

Es la potencia consumida por la carga resistiva y se define a continuación.

$$P = \frac{1}{2}\sum_{n=1}^{N} Vn\ In \cos(\theta n - \emptyset n) \qquad (6)$$

Potencia reactiva

Es la potencia almacenada en forma de campo eléctrico y magnético en cargas inductivas y capacitivas, esta potencia representa un consumo de energía que no genera trabajo útil.

$$Q = \frac{1}{2}\sum_{n=1}^{N} Vn\ In \sin(\theta n - \emptyset n) \qquad (7)$$

Factor de potencia

Expresa el desfase entre las componentes fundamentales de voltaje y corriente, tomando en cuenta su signo.

$$FP = \cos(\theta_1 - \emptyset_1) \qquad (8)$$

Donde θ_1 es la fase de la componente fundamental de voltaje y $\emptyset_1$ de la corriente.

2.5 INTERFACEZ GRÁFICAS DE USUARIO

Una herramienta marco de trabajo compresivo desarrollado en C++ para crear interfaces graficas de usuario (GUI) multiplataforma usando un acercamiento “Escribe una vez, compila donde sea”. Qt permite a los programadores utilizar un único árbol fuente para aplicaciones que pueden correr en Windows 98 a Vista, Mac OS X, Linux, HP-UX, y muchas otras versiones de Unix con X11. Las librerías y herramientas de Qt son también parte Qt/embebido Linux, un producto que provee su propio sistema de ventanas en sima de Linux embebido (Blanchette y Summerfield, 2008).

2.6 DISEÑO HEURÍSTICO

La evaluación heurística (HE) es una técnica utilizada para analizar la facilidad de uso en las primeras etapas del diseño de una interfaz de usuario. Se desarrolló en Dinamarca con el objetivo de crear un método para analizar el diseño de interfaces de una manera informal, manejable y fácil de ensenar.

Para llevar a cabo una evaluación heurística es necesario que varias personas analicen el diseño de interfaz de usuario para verificar si infringe alguna de las

heurísticas, si lo hace es necesario modificar el diseño. El análisis puede estar en cualquier etapa de desarrollo del producto, puede ser tan preliminar como un borrador a lápiz del sistema, un prototipo parcial en algún paquete o librería de desarrollo o un sistema operacional completo. Sin embargo, conforme el sistema esté más avanzado, será más complicado arreglar los problemas de usabilidad, por lo tanto un análisis heurístico tiene más valor en los primeros borradores o prototipos. De manera general, varias personas harán evaluación HE de un diseño ya que las investigaciones han Demostrado que diferentes personas encontraran diferentes infracciones. A continuación se encuentran algunas heurísticas importantes a considerar en el diseño de cualquier interfaz de usuario enumeradas en el trabajo de (Icarnegie, 2008).

1. Visibilidad del estatus del sistema.

El dispositivo debe mantener al usuario informado de lo que está ocurriendo, las entradas que ha recibido la aplicación, el proceso que se está llevando a cabo y los resultados del procesamiento. La interfaz debe proveerle información al usuario de manera visual o a través de algún sonido. Si no es así, el usuario no tiene la información necesaria para entender lo que está sucediendo.

2. Comparación entre el sistema y el mundo real.

El diseño de la interfaz del usuario debe utilizar los conceptos, lenguaje, y las convenciones del mundo real que sean familiares para el usuario. Esto significa que el programador del sistema debe comprender la tarea que el sistema debe desempeñar, desde el punto de vista del usuario. Los aspectos culturales se hacen relevantes para el diseño de sistemas computacionales globales.

3. Control y libertad del usuario.

Permite que el usuario tenga control de la interacción. Los usuarios deben poder Corregir sus acciones fácilmente, salirse de la interacción rápidamente en cualquier momento, y no verse forzados a hacer una serie de acciones controladas por el sistema. Los usuarios cometerán errores y por lo tanto, necesitan una opción fácil para corregirlos.

4. Estándares y consistencia.

La información que es igual debe verse igual (mismas palabras, iconos y posiciones en la pantalla). La información que es diferente debe ser expresada de manera diferente. Tal consistencia debe mantenerse dentro de la aplicación y de la plataforma.

5. Prevención de errores.

El sistema debe prevenir que se cometan errores tanto como sea posible, por ejemplo, si hay una cantidad limitada de acciones legales en una parte de la aplicación, ayuda a los usuarios a seleccionarlas, en lugar de permitir que lleven a cabo una acción para luego decirles que cometieron un error.

6. Reconocer en vez de recordar.

Se debe mostrar al usuario todos los objetos y acciones disponibles. No pedirle al usuario que se acuerde de la información de otra pantalla de la aplicación. Esta heurística es una aplicación directa de las teorías de la memoria humana. Es más fácil para alguien reconocer lo que tiene que hacer si existen ciertas pistas en el Ambiente que le lleguen a la memoria de trabajo a través de la percepción

7. Flexibilidad y eficiencia en el uso.

El diseño debe contar con atajos en tecleado que permita que los usuarios expertos puedan acelerar su interacción (en lugar de siempre usar los menús e íconos con un mouse). Los usuarios expertos deben de poder adaptar la interfaz para que las acciones frecuentes se hagan con mayor rapidez.

8. Diseño estético y minimalista.

Eliminar los elementos irrelevantes de la pantalla que solo representan desorden y confusión para el usuario.

9. Ayudar a los usuarios a reconocer, diagnosticar y recuperarse de errores.

Los mensajes de error deben ser escritos en un lenguaje simple, deben explicar el problema dar consejo constructivo de como corregir el error. Una vez más, esto se puede considerar parte de la primera heurística, visibilidad del estatus del sistema, pero es tan importante e ignorada tan frecuentemente que merece su propia heurística.

10. Ayuda y documentación.

Si el sistema no es extremadamente sencillo, va a ser necesario que cuente con sus opciones de ayuda y con una documentación adecuada. La ayuda y documentación debe estar siempre disponible, fácil de buscar y debe aportar consejos directos quesean aplicables a las tareas del usuario.

2.7 LIBRERÍAS

A continuación se presentan las librerías de licencia libre utilizadas en este trabajo.

2.7.1 QCustomPlot

Es un elemento desarrollado en la plataforma Qt, en lenguaje c++ diseñado para el despliegue de información eficiente de manera gráfica, no requiere de dependencias adicionales y permite interacciones de interfaz.

2.7.2 MatIO

Es una librería que permite la escritura y lectura de archivos .mat de Matlab desde C o Fortran, además permite la compresión de información en los archivos generados utilizando la dependencia ZLIB incluida.

3 METODOLOGÍA

A continuación se presenta la metodología del proyecto y desarrollo de la herramienta de interfaz gráfica diseñada para este trabajo. Con el fin de cumplir con los objetivos. La metodología general para realizar el presente trabajo se muestra en la figura 3 donde se presentan tres módulos principales. Estos módulos de dividen en las tareas que se explicarán a detalle a continuación.

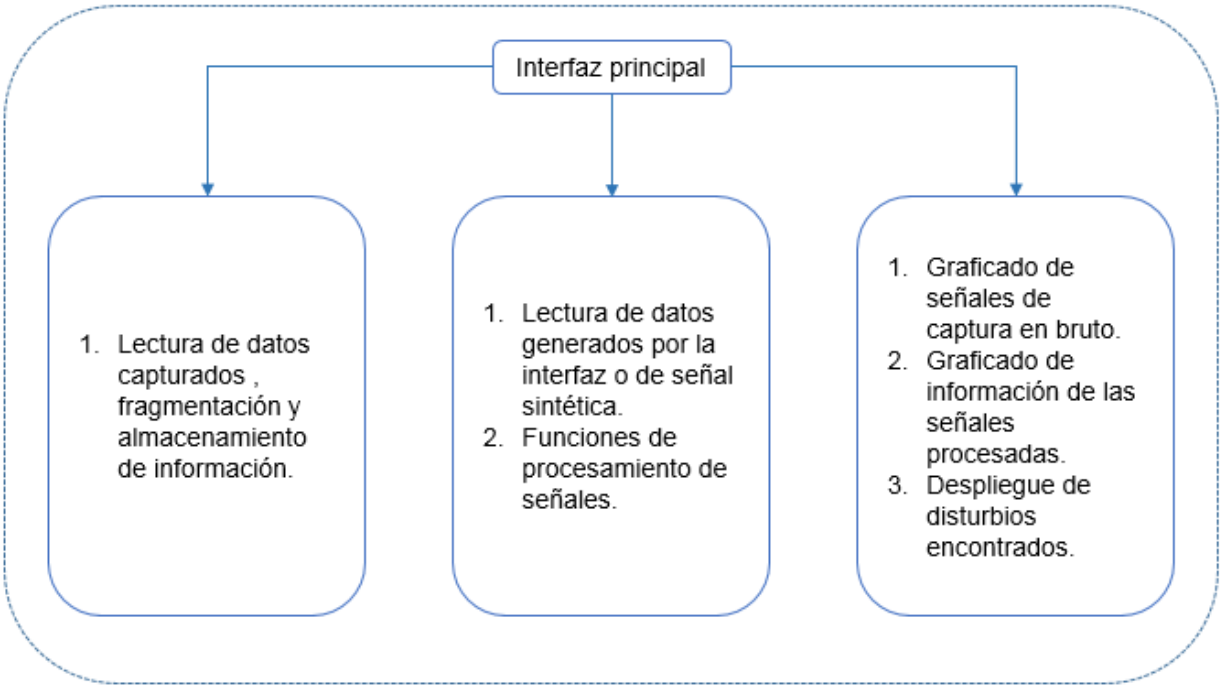


Figura 3 Metodología general del proyecto

3.1 MANEJO DE LA INFORMACIÓN CAPTURADA

En esta sección se presentan las metodologías de manejo de información implementadas para la decodificación de información capturada por la tarjeta de adquisición de datos. Y la subsecuente fragmentación y formato de archivos como se muestra en la figura 4.

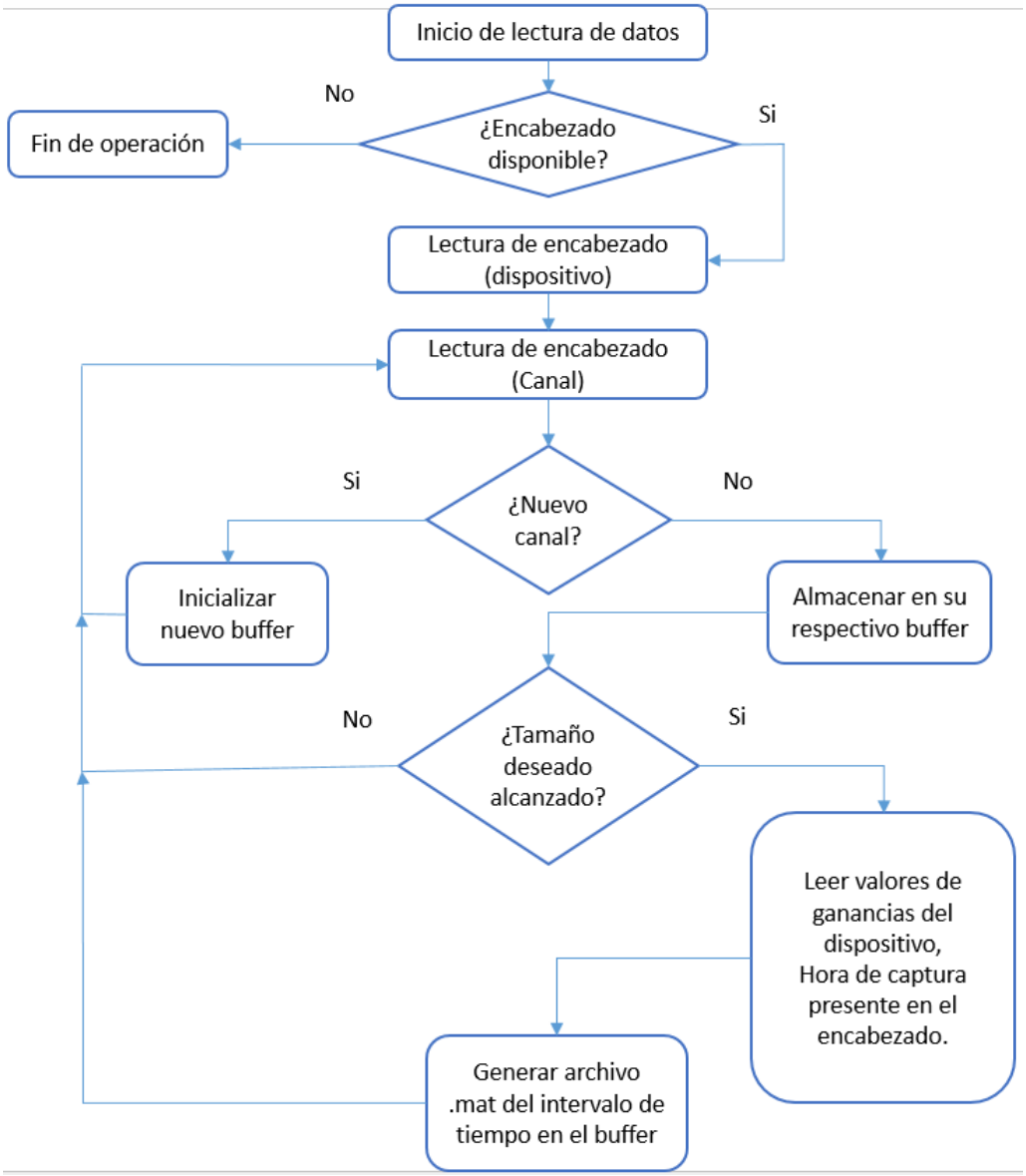


Figura 4 Metodología de lectura, fragmentación y formato de información

Como se mencionó anteriormente, la información capturada se encuentra escrita en bloques de 8 KB lo cual representa un vector de tiempo de muy pequeño, por lo cual es necesario acumular estos datos hasta generar un vector de información útil para su estudio.

De acuerdo con el estándar IEEE 1159 las señales eléctricas se muestrean a 128, 256 y 512 muestras por ciclo y de acuerdo con el estándar EN 50160 medición y prueba de la calidad de fuentes de voltaje requiere intervalos de medición de 10 minutos. Se puede inferir que la información capturada por el dispositivo PQ-UAQ necesita una frecuencia de captura superior a 6.4 ksps y esta información debe ser separada en intervalos de 10 minutos. Estos archivos de información de captura deben tener suficientes datos para su posterior reinterpretación.

Tomando en consideración lo anterior, una lectura in interrumpida de un periodo de una semana generaría un total de 1080 archivos por cada uno de los 7 canales disponibles. Se propone implementar una función de indexado de información con el objetivo de aligerar el manejo de información realizado por la interfaz gráfica y al mismo tiempo facilitar la movilidad y el manejo de la información para el usuario como se muestra en la figura 5.

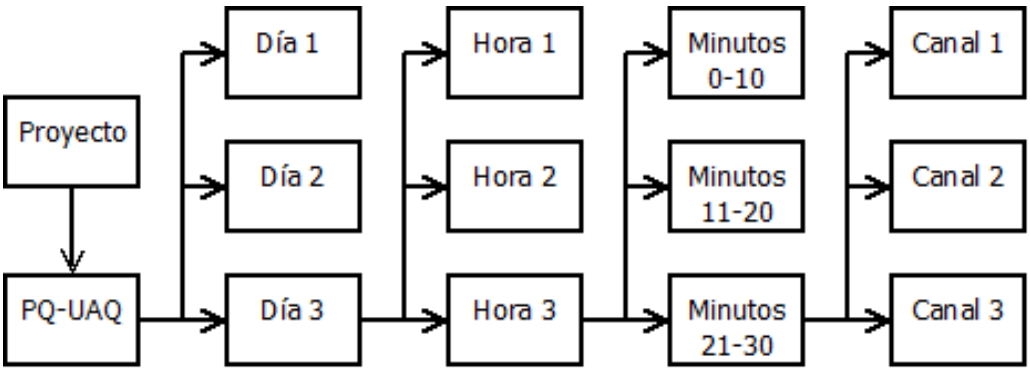


Figura 5 Metodología de fragmentación de información.

3.2 LECTURA Y PROCESAMIENTO DE DATOS

En esta sección se presentan las funciones de lectura de información, para su posterior procesamiento y análisis. El análisis de datos requiere una búsqueda de todos los archivos disponibles, por esto una función de búsqueda de archivo se realiza para el acceso de vectores de información determinados por el usuario.

Un proyecto puede contener las lecturas de varias tarjetas de adquisición de datos, estos dispositivos disponibles son listados en la ventana principal. El usuario debe seleccionar los dispositivos que se desean analizar. Una vez un dispositivo es seleccionado de la lista, se genera una nueva instancia, la cual realiza la búsqueda de archivos y manda llamar las funciones de procesamiento de datos. Los apartados de memoria y vectores de datos son instanciados en una nueva ventana. La cual tiene como objetivo obtener una serie de vectores de datos que comprenden intervalos de las señales a analizar. Una vez el usuario determina el intervalo de tiempo a analiza. Las funciones de procesamiento pueden ser efectuadas como se muestra en la figura 6.

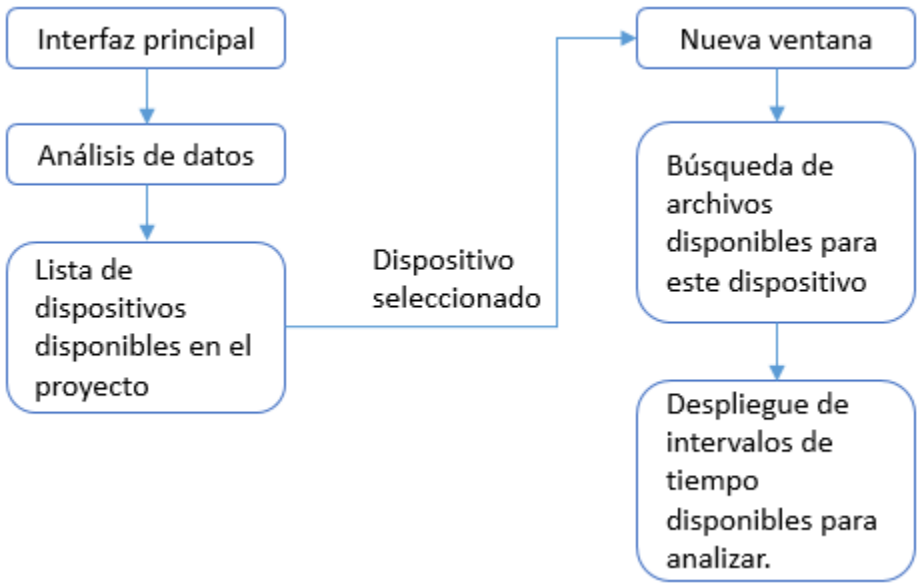


Figura 6 Búsqueda de archivos y lectura de datos

3.2.1 PROCESAMIENTO DE DATOS

Una vez el vector de datos a analizar ha sido determinado por el usuario, las rutinas de procesamiento pueden comenzar por medio de la activación de un botón. Las técnicas de procesamiento tienen como objetivo calcular índices de calidad de la energía, así como calcular componentes de las señales analizadas. Angulo de desfase entre las señales, y componentes harmónicos presentes en las lecturas.

3.2.1.1 ALGORITMO STOCKHAM

El algoritmo Stockham (descrito en la figura 7) es una técnica de cálculo de la transformada rápida de Fourier. Un algoritmo que utiliza un espacio de trabajo para evitar la función de inversión de bits que generalmente se usa en FFTs. Este aspecto del algoritmo representa un tiempo reducido de cómputo para vectores considerablemente grandes.

```
for q = 1:t
        L <- 2^q; r <- n/L

        L_2 <- L/2; r_2 <- n/L_2
        y <- x
        for j=0:L_2 -1
                ω <- cos (2πj/L) – i  sin(2πj/L)
                for k = 0:r - 1
                τ <- ω y (jt_2+k+r)
                x( j r + k ) <- y(j r2 + k) + τ
                x((j+L_2)r + k) <- y(j r_2 + k) – τ
                end
        end
end
```

Figura 7 seudocódigo del algoritmo implementado para el cálculo de la FFT

3.2.1.2 DESCOMPOSICIÓN WAVELET

El algoritmo utilizado (figura 8) realiza la transformada wavelet de una dimensión en lugar, remplazando el vector original por el vector procesado

```
Output = length (input)

For (lenght = length(input) >> 1;  lenght >>=1)
{
        for (i =0; i<lenght; ++i)
        {
                sum = input[1*2]+input[i+2+1];
                difference = input[i*2] – input[1*2+1];
                output [i] = sum;
                output [lenght + i]
        }
        if (lenght == 1)
        reutrn output;
}
```

Figura 8 seudocódigo del algoritmo implementado para el cálculo de la transformada wavelet

3.2.1.3 ÍNDICES DE CALIDAD DE LA ENERGÍA

Los índices implementados en esta interfaz calculan los valores de desfase de las señales eléctricas y las variaciones de voltaje y corriente utilizadas para la detección de sags, swells, e interrupciones en una señal.

3.3 GRAFICADO DE INFORMACIÓN

En esta etapa, se emplean funciones de graficado y operaciones de escalado. Para la visualización de los datos deseados. Estas funciones son las únicas realizadas dentro de la interfaz gráfica. La instancia principal se encarga de generar todos los vectores necesarios para el graficado de las capturas. Sus transformaciones y valores obtenidos a través del cálculo de índices de calidad de la energía.

Los valores graficados en la interfaz son los de variaciones de voltaje, variaciones de corriente, factor de potencia, fasores, densidad espectral, periodograma. Cada una de estas funciones requieren operaciones subsecuentes para la interpretación de correcta de las gráficas. Para la conveniencia del usuario, todas la gráficas fueron separadas en pestañas en la interfaz de análisis de datos como se muestra en la figura 9.

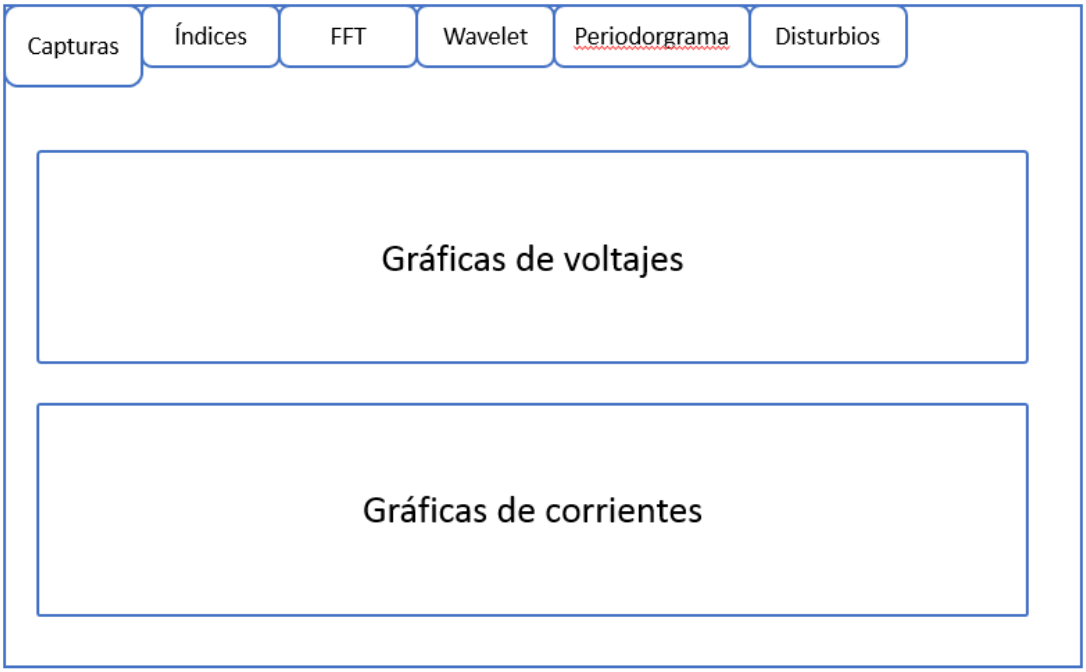


Figura 9 Diagrama de la estructura de la interfaz

4 RESULTADOS

4.1 COMPRESIÓN DE ARCHIVOS Y ARBOL DE DIRECTORIOS

Los archivos generados serian aún incompatibles con cualquier aplicación. Para facilitar la administración y procesamiento de datos. Todos los archivos fueron convertidos a formato .mat para ser compatibles con la aplicación MATLAB, entre otras aplicaciones. La estructura de los archivos generados se representa en la Tabla 2. Para optimizar el tamaño de cada archivo se utilizó una librería libre.

Objeto	Esctructura
Color	String
Comentario	string
Datos	Vector
Dispositivo	String
Ganancia	Número
Offset	Número
Tiempo Real	String
Número de	Número
Frecuencia de	Número
Señal	String
Unidad	char

Tabla 3 Estructura de datos posterior a la fragmentación

4.2 TIEMPO DE DESCOMPRESIÓN

Una vez la captura de los datos termina. La recuperación de dichos datos puede realizarse. El total de lecturas de una semana se recupera en las siguientes velocidades.

La Tabla 3 describe el tiempo de que toma la recuperación de datos de la tarjeta SD, el proceso de fragmentación, formateo, compresión e indexado.

La compresión de datos es efectuada durante la recuperación de datos y afecta el tamaño de los archivos en base a las variaciones de los datos.

La información sin comprimir recabada durante una captura de datos es invariante. Un archivo de 10 minutos es de 9.15 MB. Mientras que un archivo comprimido varía entre 3 MB para corrientes y 8MB para voltajes dependiendo de su información. Tomando el promedio de estos datos se obtiene un 40 por ciento ahorro de espacio en el disco.

Especificaciones	PC 1	PC 2
procesador	2nd. Gen. I5	4rth. Gen. I5
ram	8 Gb DDR3	16 Gb DDR3
Disco duro	5400 RPM	7200 RPM
Tiempo en realizar la tarea.	2:10 Horas	1:50 Horas

Tabla 4 Tiempos de descompresión de información

4.3 PROCESAMIENTO DE SEÑALES

A través de la interfaz principal se puede seleccionar la ubicación del proyecto que se desea analizar figura 9, el programa se encarga de detectar cada uno de los dispositivos utilizados en dicho proyecto, esta tarea toma tiempo pues determina el número de archivos e integridad de datos de cada dispositivo.

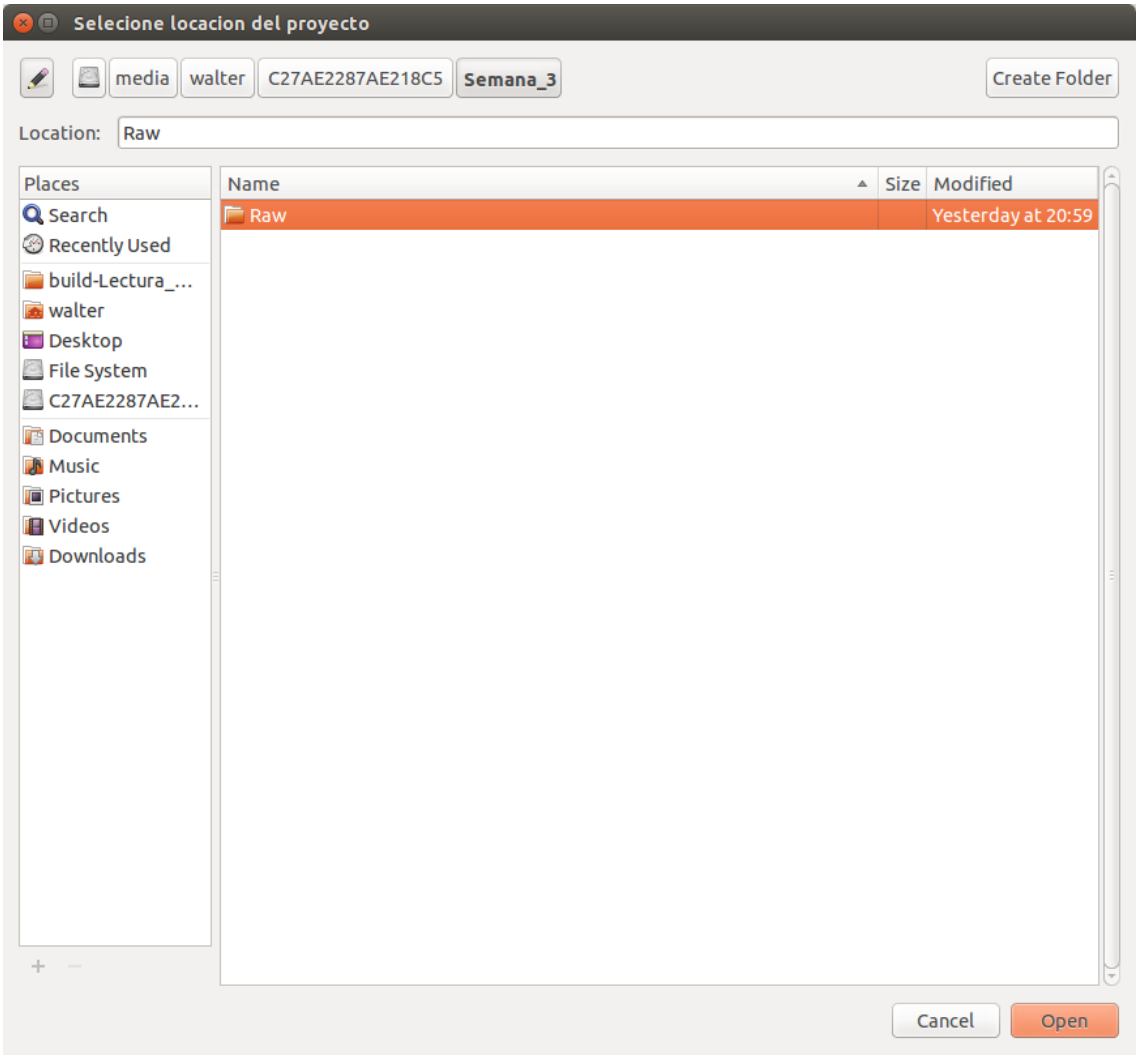
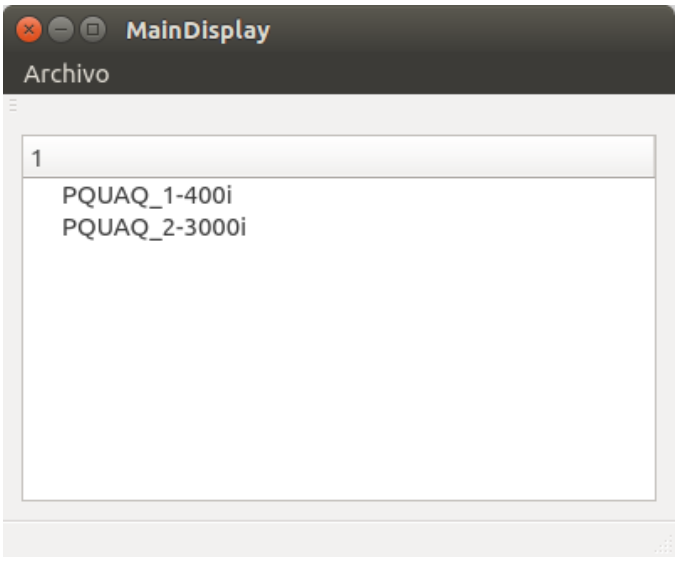


Figura 10 Selección de locación de proyecto

Posteriormente la interfaz principal lista los dispositivos presentes en el proyecto, y el usuario puede seleccionar cualquiera para desplegar información de dicho dispositivo dando doble clic en el dispositivo de la lista. Esto genera un apartado de memoria para este dispositivo, por lo cual es posible seleccionar otro dispositivo de la lista sin perder los datos de los dispositivos ya seleccionados figura 11.



Lista de dispositivos disponibles 11

Los valores de voltaje y corriente se despliegan en dos gráficas distintas en una se muestran los valores de voltaje y en la otra se muestran los valores de amperaje como se muestra en la figura 12.

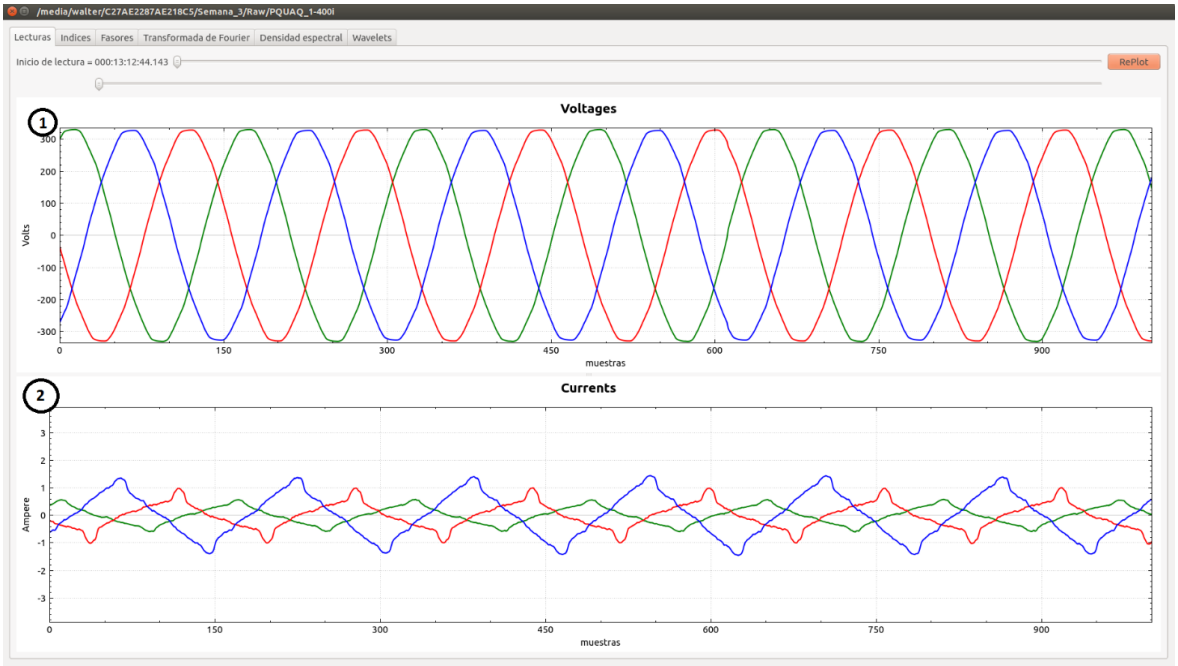


Figura 12 Gráficas de valores de voltaje y amperaje capturadas.

En la gráfica 11 en (1) se pueden apreciar los datos de voltaje capturados por la tarjeta PQ-UAQ. Cada fase está representada por un color. En verde se representa los valores de voltaje de la fase “A”, en rojo se representa la fase “B” y en azul la fase “C”. Análogamente los valores de corriente se grafican de la misma manera en (2).

4.4 ÍNDICES DE CALIDAD DE LA ENERGÍA

En la figura 13, se muestran los índices de calidad de la energía calculado en un periodo de tiempo determinado por el usuario

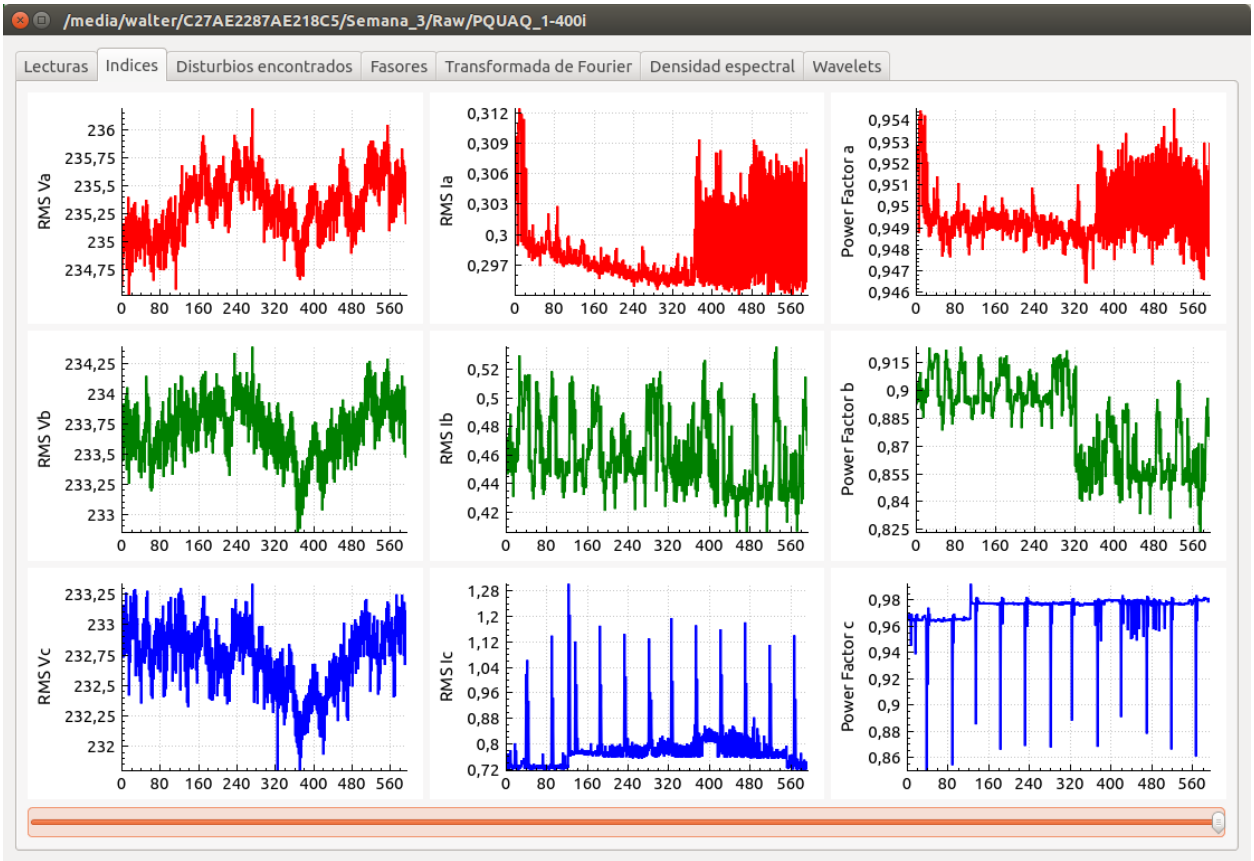


Figura 13 Representación gráfica de variaciones de voltaje, variaciones de corriente y factor de potencia.

4.5 TRANSFORMADA DE FOURIER

En la figura 14 se muestran las densidades espectrales calculadas por la transformada de Fourier. De las señales eléctricas capturadas.

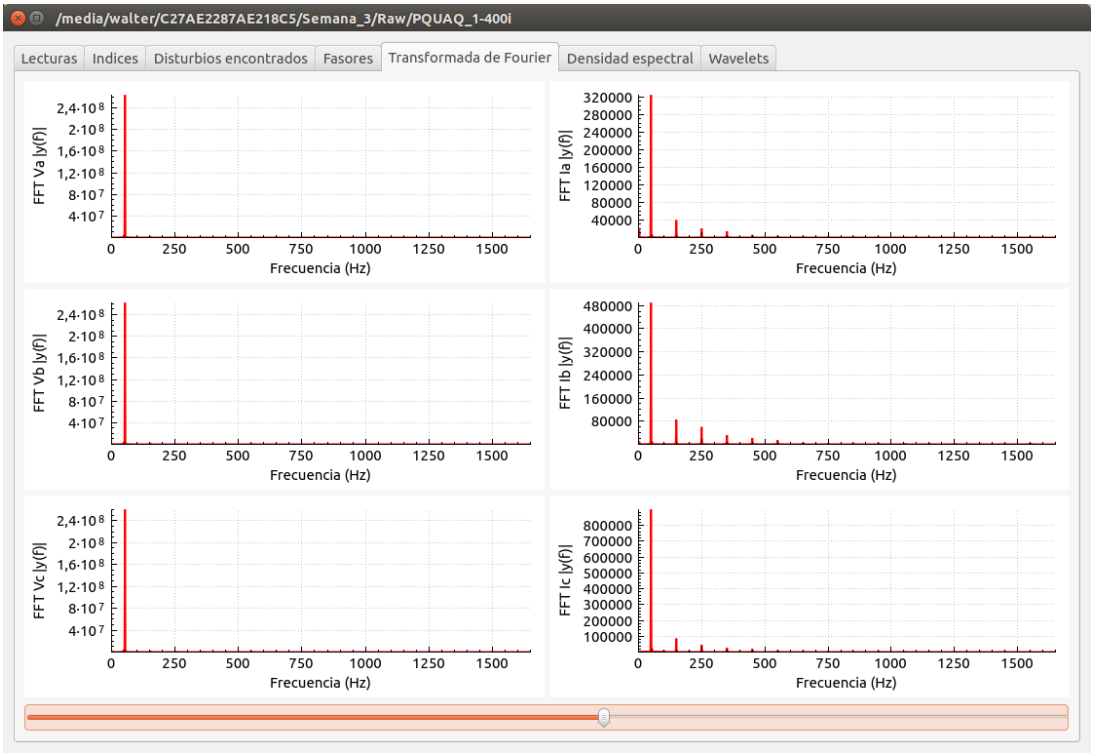


Figura 14 Transformadas de Fourier

La interfaz de la figura 14 está dividida en 6 secciones del lado izquierdo se muestran los valores de la transformada de Fourier de los voltajes de las fases “A”,”B” y “C” (de arriba hacia abajo). Del lado derecho se muestran los valores de la fft de las corrientes de las mismas fases.

4.6 PERIODOGRAMA

El periodograma actúa como una función de acumulación de componentes harmónicos de una señal. Es por esto que esta función fue implementada para que el usuario pueda identificar los componentes harmónicos de las señales más fácilmente como se muestra en la figura 15.

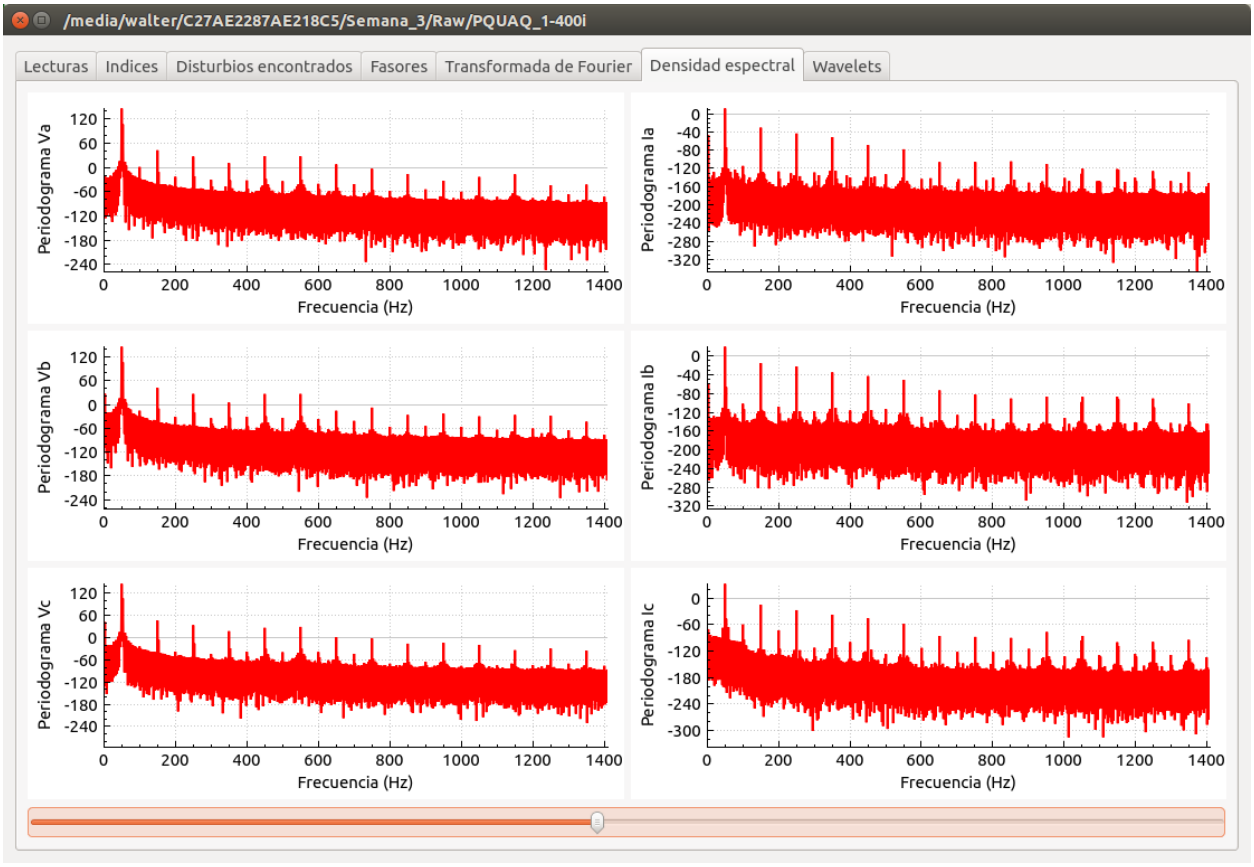


Figura 15 Periodogramas de las señales eléctricas

4.7 DESCOMPOSICIÓN WAVELET

En la figura 16 se muestra la descomposición wavelet de los voltajes de la fase “A”, como se puede apreciar en la figura, la transformada wavelet se aplica a todas las fase. Realizando un total de 6 descomposiciones de 6 niveles, tres para los voltajes de cada fase y tres para las corrientes.

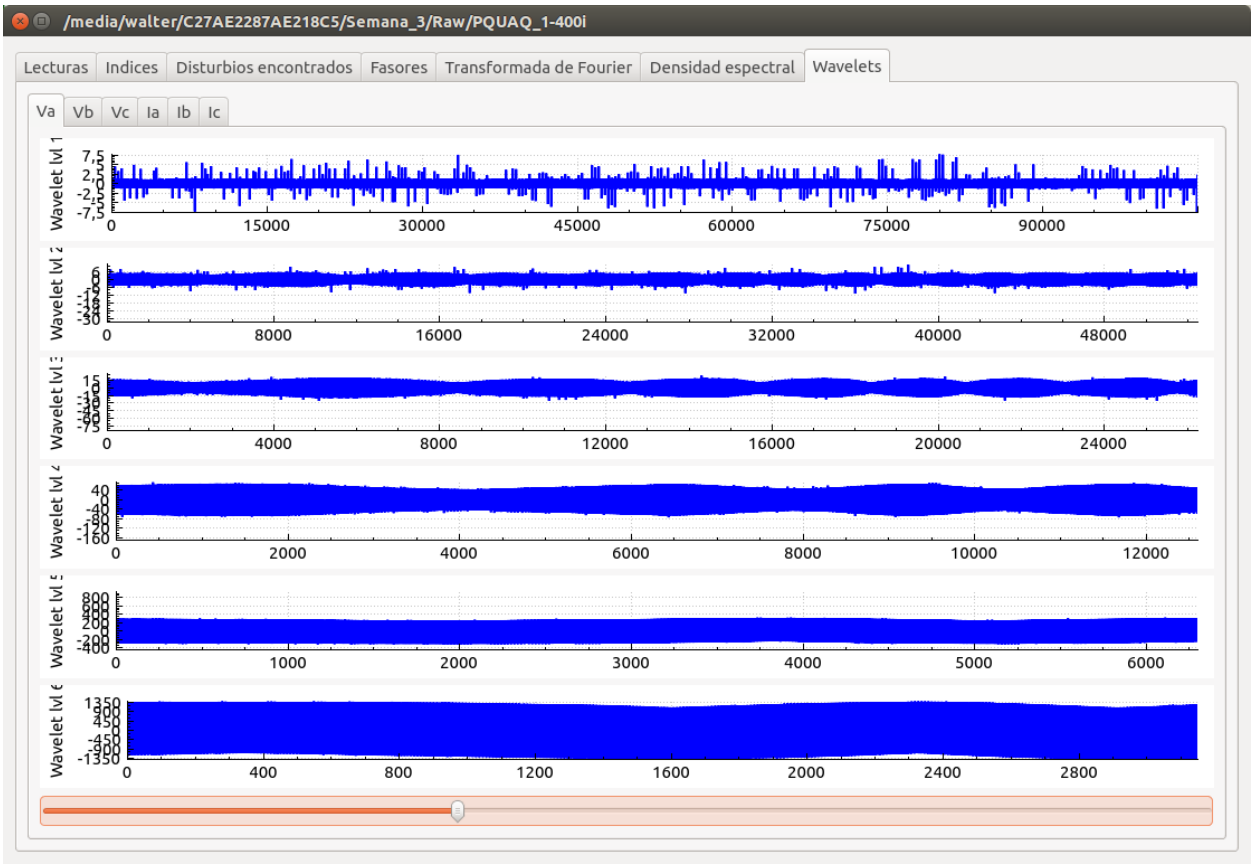


Figura 16 Descomposición wavelet de la señal de voltaje de la fase “a”.

4.8 FASORES

El cálculo de los fasores se realiza con el mismo vector de información de voltajes y corrientes utilizados en todas las transformaciones e índices presentados anteriormente. Con un total de 6 flechas los fasores determinan el ángulo de desfase entre voltajes y corrientes, en la figura 21 se pueden apreciar 2 flechas de cada color, las flechas largas representan los voltajes y las cortas representan las corrientes, el ángulo de diferencia entre las flechas cortas y las largas representan el desfase. En este caso, el ángulo de desfase es muy pequeño.

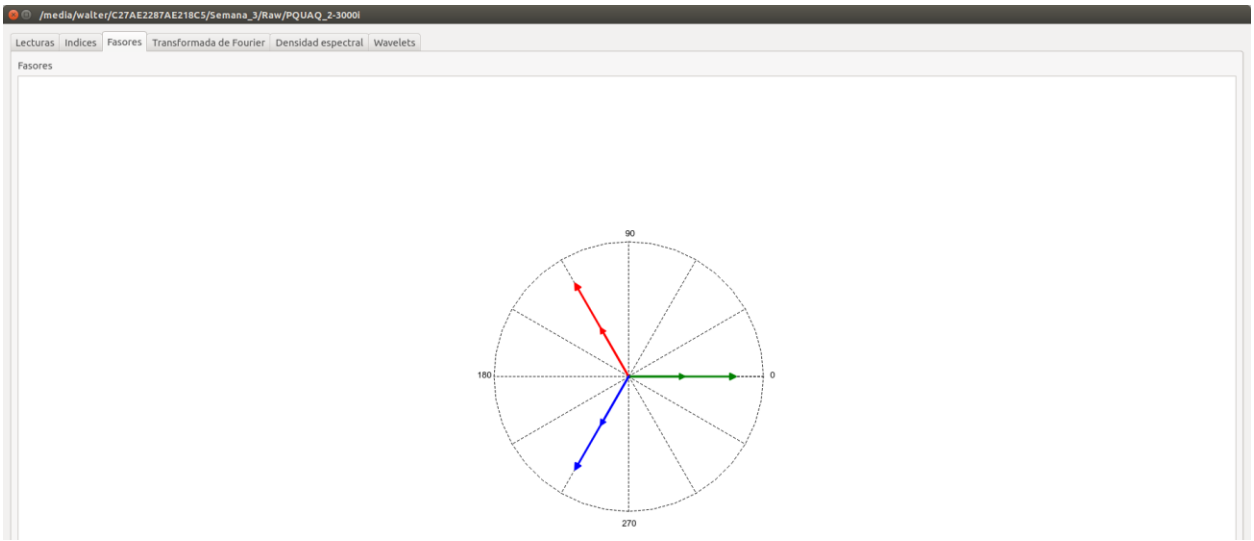


Figura 17 fasores

4.9 ANÁLISIS DE SEÑALES SINTÉTICAS

El programa desarrollado fué diseñado para la lectura de señales de energía capturadas por la tarjeta de adquisición de datos PQ-UAQ así como la lectura y procesamiento de señales generadas por otra plataforma compatible con archivos .mat.

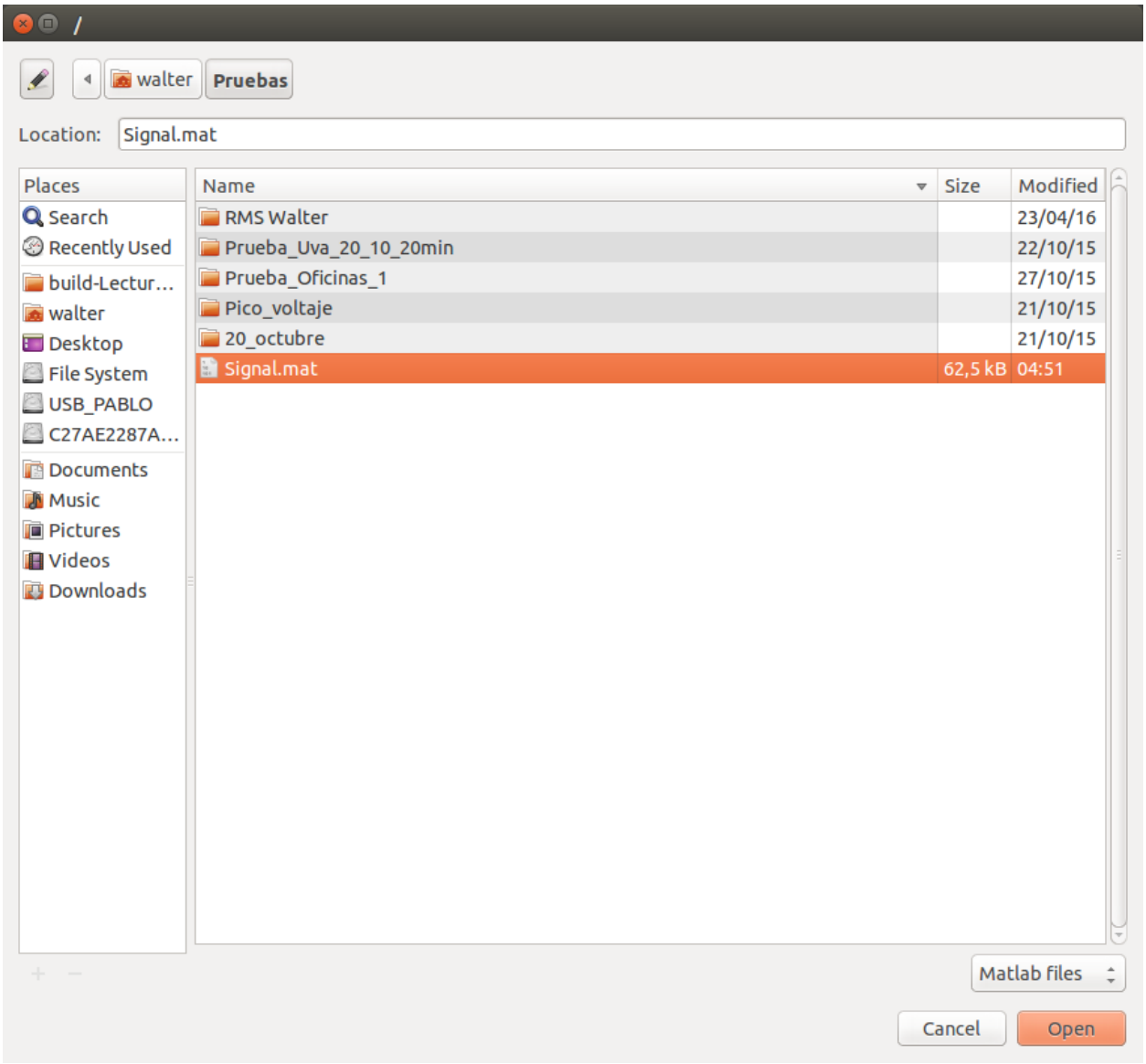


Figura 18 Ventana de explorador de archivos .mat

Como se aprecia en la figura 19 se puede apreciar una señal generada que consiste de diferentes amplitudes y una sola frecuencia. Con el fin de comprobar la validez de las herramientas de procesamiento de datos empleadas de este trabajo, se compararon las mismas funciones ejecutadas por Matlab.

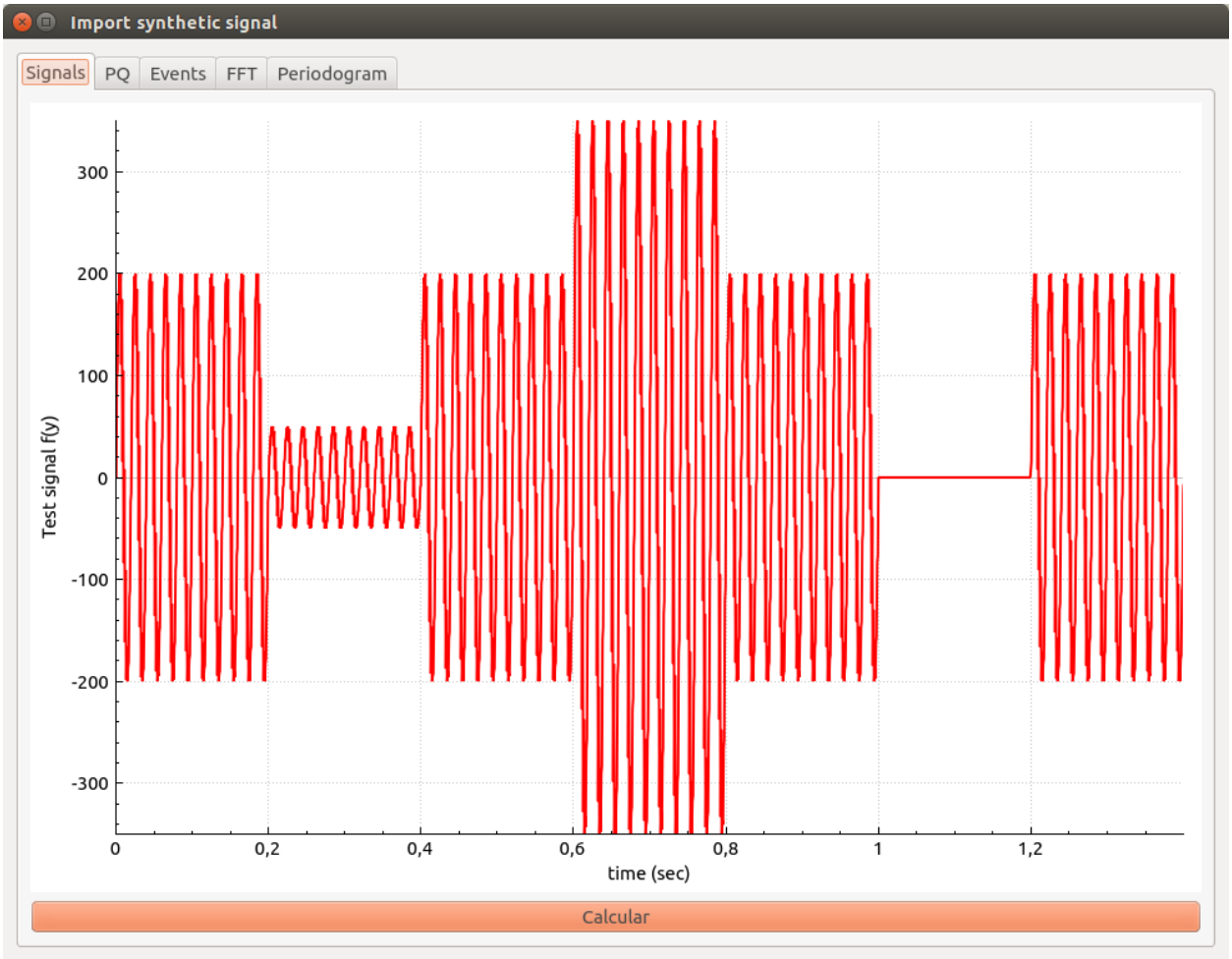


Figura 19 Señal sintética generada por Matlab.

4.10 COMPARACIÓN DE RESULTADOS DE PROCESAMIENTO

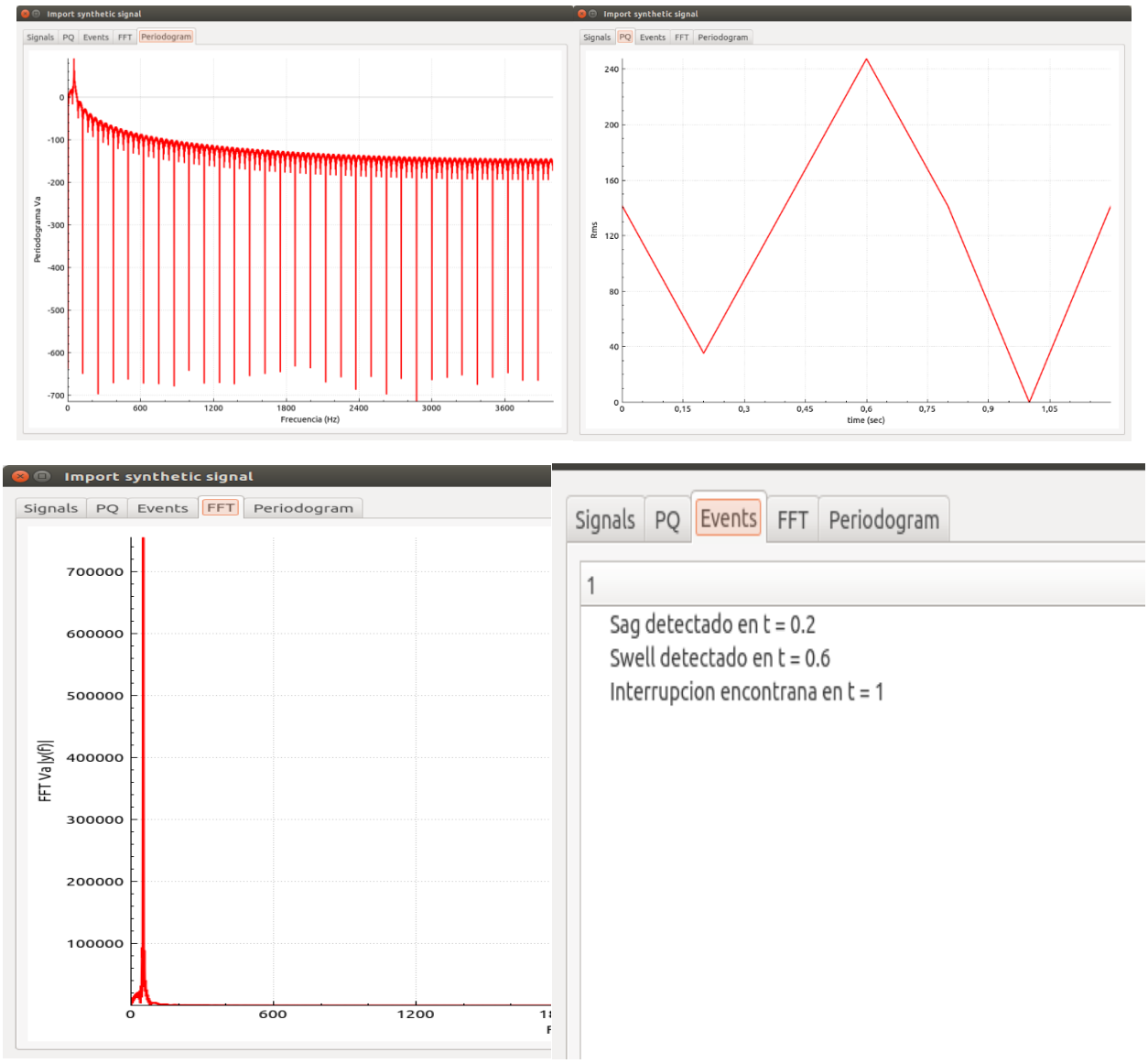


Figura 20 Resultados de procesamiento de señal sintética

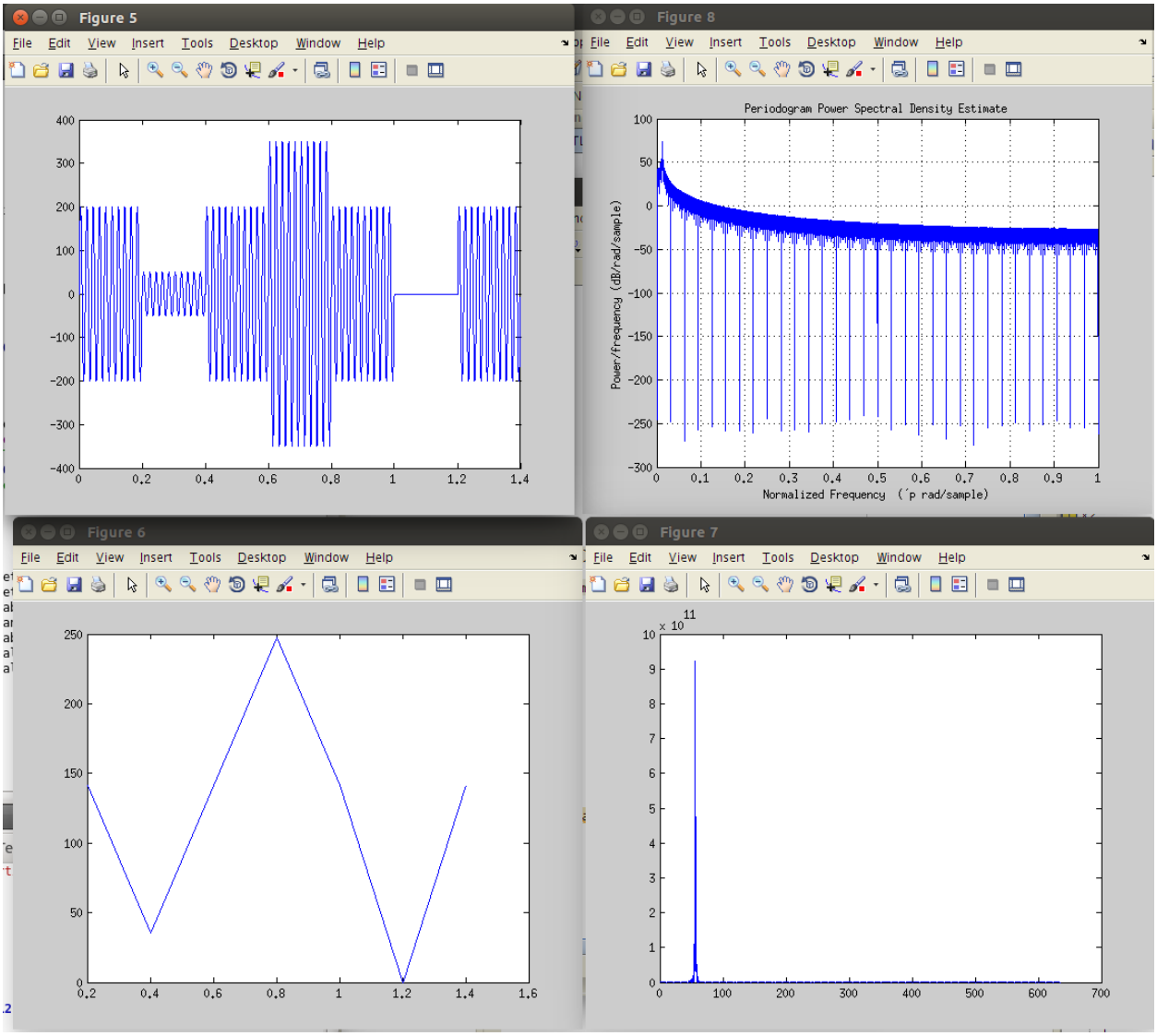


Figura 21 resultados de procesamiento de señal sintética por medio de Matlab.

Como se puede apreciar, la interfaz es capaz de graficar la información de la señal correctamente, realizar transformaciones para encontrar el contenido harmónico que contiene la señal. Y calcular las rms. Correctamente con esta información se puede determinar la presencia de sags, swell o interrupciones de manera confiable.

5 CONCLUSIONES

El primero de los objetivos particulares dictamina determinar los requerimientos de la aplicación. Los requerimientos que se toman en cuenta son capacidad de cómputo de media gama. Capacidad de almacenamiento masivo. Superior a 1 TB. Los procesos computacionales efectuados pueden ser realizados por cualquier computadora de media gama, con el inconveniente del tiempo de procesamiento. Que puede ser superior a los 2 minutos.

El segundo objetivo es el desarrollar, elementos básicos de visualización. Estos elementos fueron desarrollados para el graficado valores polares, la generación de vectores en el tiempo y frecuencia así como los índices de calidad de la energía que arrojan valores angulares y estadísticos.

El tercer objetivo es el de generar modos diferentes de visualización. Se presentan formas de visualización de señales en el tiempo, en frecuencia, una gráfica polar y por medio de herramientas estadísticas proporcionadas por las regulaciones internacionales es posible la detección de disturbios.

El cuarto objetivo presenta la tarea de generar el código de metodologías para el análisis de señales. Por lo que se implementaron técnicas de procesamiento comenzando con los índices de calidad de energía, la transformada rápida de Fourier, descomposición wavelet, periodograma y fasores.

La hipótesis indica que mediante técnicas de visualización y procesamiento integrados en una plataforma libre será posible la detección de al menos tres disturbios eléctricos. Los cuales son (Sags, Swells e Interrupciones). Con las capacidades de procesamiento de una computadora de costo promedio, el cual es mucho menor al de los dispositivos de monitoreo de de energía disponibles en el mercado.

REFERENCIAS

Agrawal, S., & Prakash, R. N. (2013). Implementation of WSN which can simultaneously monitor Temperature conditions and control robot for positional accuracy.

Akiyama, M., Goh, T., Nakagawa, Y., Suematsu, M., Sakamoto, T., & Akiyama, H. (2011). PULSED POWER GENERATOR DRIVEN BY FPGA AND PC.

Blanchette, J., & Summerfield, M. (2008). *C++ GUI Programming with Qt 4, Second Edition* (Second). Prentice Hall.

Cabal-Yepez, E., Garcia-Ramirez, A. G., Romero-Troncoso, R. J., Garcia-Perez, A., & Osornio-Rios, R. a. (2013). Reconfigurable monitoring system for time-frequency analysis on industrial equipment through STFT and DWT. *IEEE Transactions on Industrial Informatics*, *9*(2), 760–771. http://doi.org/10.1109/TII.2012.2221131

Callejas, I., Piñeros, J., Rocha, J., Hernández, F., & Delgado, F. (2013). Implementación de una red neuronal artificial tipo SOM en una FPGA para la resolución de trayectorias tipo laberinto. *2013 2nd International Congress of Engineering Mechatronics and Automation, CIIMA 2013 - Conference Proceedings*. http://doi.org/10.1109/CIIMA.2013.6682790

Cisneros-Magaña, R., Medina, A., & Segundo-Ramírez, J. (2014). Efficient time domain power quality state estimation using the enhanced numerical differentiation Newton type method. *International Journal of Electrical Power & Energy Systems*, *63*, 414–422. http://doi.org/10.1016/j.ijepes.2014.05.076

Deepa, P. (2013). Solar tracking System with High precision implementation on FPGA with GUI on LabVIEW, (1), 373–376.

Dehghani, H., Vahidi, B., Naghizadeh, R. a., & Hosseinian, S. H. (2013). Power quality disturbance classification using a statistical and wavelet-based Hidden Markov Model with Dempster–Shafer algorithm. *International Journal of Electrical Power & Energy Systems*, *47*, 368–377. http://doi.org/10.1016/j.ijepes.2012.11.005

Ferreira, D. D., Seixas, J. M. De, & Cerqueira, A. S. (2015). A method based on independent component analysis for single and multiple power quality disturbance classification. *Electric Power Systems Research*, *119*, 425–431. http://doi.org/10.1016/j.epsr.2014.10.028

Flores, R. a. (2002). State of the art in the classification of power quality events, an overview. *10th International Conference on Harmonics and Quality of Power. Proceedings (Cat. No.02EX630)*, *1*, 1–4. http://doi.org/10.1109/ICHQP.2002.1221398

FLUKE. (2004). AC i400 current clamp, *2004*(December), 1–2.

García-Hernández, J. M., Ramírez-Jiménez, F. J., Mondragón-Contreras, L., López-Callejas, R., Torres-Bribiesca, M. a., & Peña-Eguiluz, R. (2013). Power quality considerations for nuclear spectroscopy applications: Grounding. *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, *729*, 147–152. http://doi.org/10.1016/j.nima.2013.06.051

Gilliland, S., Boulet, C., Gonnot, T., & Saniie, J. (2014). Multidimensional Representation of Ultrasonic

Data Processed by Reconfigurable Ultrasonic System-on-Chip Using OpenCL High-Level Synthesis, 1936–1939.

González, M., Cárdenas, V., & Espinosa, G. (2014). Advantages of the passivity based control in dynamic voltage restorers for power quality improvement. *Simulation Modelling Practice and Theory*, *47*, 221–235. http://doi.org/10.1016/j.simpat.2014.06.009

Granados-Lieberman, D., Osornio-Rios, R. a., Rivera-Guillen, J. R., Trejo-Hernandez, M., & Romero-Troncoso, R. J. (2013). Torque reduction and workpiece finishing effects due to voltage sags in turning processes. *Proceedings of the Institution of Mechanical Engineers, Part B: Journal of Engineering Manufacture*, *228*(1), 140–148. http://doi.org/10.1177/0954405413497940

Granados-Lieberman, D., Valtierra-Rodriguez, M., Morales-Hernandez, L., Romero-Troncoso, R., & Osornio-Rios, R. (2013). A Hilbert Transform-Based Smart Sensor for Detection, Classification, and Quantification of Power Quality Disturbances. *Sensors*, *13*, 5507–5527. http://doi.org/10.3390/s130505507

Horvat, R., Jezernik, K., Member, L., & Curkoviˇ, M. (2014). An Event-Driven Approach to the Current Control of a BLDC Motor Using FPGA, *61*(7), 3719–3726.

Institute of Electrical and Electronics Engineers. (2009). *IEEE Std 1159 - IEEE Recommended Practice for Monitoring Electric Power Quality. IEEE Std 1159-2009 (Revision of IEEE Std 1159-1995)* (Vol. 2009). http://doi.org/10.1109/IEEESTD.2009.5154067

Jaya Bharata Reddy, M., Raghupathy, R. K., Venkatesh, K. P., & Mohanta, D. K. (2013). Power quality analysis using Discrete Orthogonal S-transform (DOST). *Digital Signal Processing*, *23*(2), 616–626. http://doi.org/10.1016/j.dsp.2012.09.013

Jayarajan, J., Kumaran, R., Paul, S., Parmar, R. M., & Koringa, P. (2013). Design of High Precision Electronics for Laser Range Finder, 1–6.

Khan, a. K. (2001). Monitoring power for the future. *Power Engineering Journal*, *15*, 81. http://doi.org/10.1049/pe:20010204

Khan, S. R. (2014). GUI Based Industrial Monitoring and Control System, (Pestse), 0–3.

Liberado, E. V., Marafão, F. P., Simões, M. G., Souza, W. A. De, & Pomilio, J. A. (2015). Novel expert system for defining power quality compensators. *Expert Systems with Applications*, *42*(7), 3562–3570. http://doi.org/10.1016/j.eswa.2014.12.032

Lima, R., Quiroga, D., Reineri, C., & Magnago, F. (2008). Hardware and software architecture for power quality analysis. *Computers & Electrical Engineering*, *34*(6), 520–530. http://doi.org/10.1016/j.compeleceng.2007.12.003

Liu, C.-L. (2010). A Tutorial of the Wavelet Transform. *History*, 1–72.

Markiewicz, H. (Cooper D. A., & Klajn, A. (Wroclaw U. of T. (2004). Voltage Disturbances. *Power Quality Application Guide*, *5.4.2*, 4–11. Retrieved from www.cda.org.uk; www.brass.org; www.eurocopper.org;

Moreno, R., Visairo, N., Núñez, C., & Rodríguez, E. (2014). A novel algorithm for voltage transient detection and isolation for power quality monitoring. *Electric Power Systems Research*, *114*, 110–

117. http://doi.org/10.1016/j.epsr.2014.04.009

Patsalides, M., Stavrou, A., Efthymiou, V., & Georghiou, G. E. (2012). Towards the establishment of maximum PV generation limits due to power quality constraints. *International Journal of Electrical Power & Energy Systems*, *42*(1), 285–298. http://doi.org/10.1016/j.ijepes.2012.03.043

Proakis, J. G. (2006). Digital Signal Processing 4th edition.

Ravanasa, K. (2014). vLab , A High Speed Multi-Accesses Parallel Processing Remote Laboratory Access for FPGA Design Technology, 377–381.

Rodríguez, a., Aguado, J. a., Martín, F., López, J. J., Muñoz, F., & Ruiz, J. E. (2012). Rule-based classification of power quality disturbances using S-transform. *Electric Power Systems Research*, *86*, 113–121. http://doi.org/10.1016/j.epsr.2011.12.009

Romero-Troncoso, R. J., Saucedo-Gallaga, R., Cabal-Yepez, E., Garcia-Perez, A., Osornio-Rios, R. a., Alvarez-Salas, R., … Huber, N. (2011). FPGA-based online detection of multiple combined faults in induction motors through information entropy and fuzzy inference. *IEEE Transactions on Industrial Electronics*, *58*(11), 5263–5270. http://doi.org/10.1109/TIE.2011.2123858

Shareef, H., Mohamed, A., & Ibrahim, A. A. (2013). An image processing based method for power quality event identification. *International Journal of Electrical Power & Energy Systems*, *46*, 184–197. http://doi.org/10.1016/j.ijepes.2012.10.049

Valtierra-Rodriguez, M., Osornio-Rios, R. A., Garcia-Perez, A., & Romero-Troncoso, R. D. J. (2013). FPGA-based neural network harmonic estimation for continuous monitoring of the power line in industrial applications. *Electric Power Systems Research*, *98*, 51–57. http://doi.org/10.1016/j.epsr.2013.01.011

Valtierra-Rodriguez, M., Romero-Troncoso, R. D. J., Osornio-Rios, R. a, & Garcia-Perez, a. (2013). Detection and Classification of Single and Combined Power Quality Disturbances using Neural Networks. *Industrial Electronics, IEEE Transactions on*, *PP*(c), 1. http://doi.org/10.1109/TIE.2013.2272276

6 APÉNDICE

6.1 CÓDIGOS

```
QT       += core gui

greaterThan(QT_MAJOR_VERSION, 4): QT += widgets printsupport

TARGET = Lectura_Mat_Test_12
TEMPLATE = app


SOURCES += main.cpp\
        maindisplay.cpp \
    pq_display.cpp \
    filemanager.cpp \
    qcustomplot.cpp \
    plotscene.cpp \
    filegen.cpp \
    fft.cpp \
    synthetic_display.cpp

HEADERS  += maindisplay.h \
    pq_display.h \
    filemanager.h \
    qcustomplot.h \
    plotscene.h \
    filegen.h \
    fft.h \
    synthetic_display.h
```

```
FORMS    += maindisplay.ui \
   pq_display.ui \
   synthetic_display.ui

unix:!macx: LIBS += -L$$PWD/../../../../usr/local/lib/ -lmatio

INCLUDEPATH += $$PWD/../../../../usr/local/include
DEPENDPATH += $$PWD/../../../../usr/local/include

unix:!macx: PRE_TARGETDEPS += $$PWD/../../../../usr/local/lib/libmatio.a

unix:!macx: LIBS += -L$$PWD/../../../../usr/local/lib/ -lgsl

INCLUDEPATH += $$PWD/../../../../usr/local/include
DEPENDPATH += $$PWD/../../../../usr/local/include

unix:!macx: PRE_TARGETDEPS += $$PWD/../../../../usr/local/lib/libgsl.a

unix:!macx: LIBS += -L$$PWD/../../../../usr/local/lib/ -lgslcblas

INCLUDEPATH += $$PWD/../../../../usr/local/include
DEPENDPATH += $$PWD/../../../../usr/local/include

unix:!macx: PRE_TARGETDEPS += $$PWD/../../../../usr/local/lib/libgslcblas.a

#ifndef FFT_H
#define FFT_H
```

```
#include <stdio.h>
#include <math.h>
#include <stdlib.h>
#include <assert.h>
#include <iostream>

#define  REALSIZE     8
typedef double real;
typedef struct { real Re; real Im; }  complex;

                          #undef M_PI
#if (REALSIZE==16)
#define sin  sinl
#define cos  cosl
#define fabs  fabsl
#define M_PI 3.14159265358979323846264338327951L
#else
#define M_PI 3.141592653589793238E0
#endif


class fft
{
public:
   fft();
   void fft1d(complex x[],int n, int flag);
   void stockham(complex x[], int n, int flag, int n2, complex y[]);
```

```
};

#endif // FFT_H

#ifndef FILEGEN_H
#define FILEGEN_H

#include <QString>
#include <iostream>



class FileGen
{
public:
   FileGen();
   void Generator(QString ProjectRoute);
};

#endif // FILEGEN_H

#ifndef FILEMANAGER_H
#define FILEMANAGER_H
#include <QList>
#include <QString>
#include <QDebug>
#include <iostream>
#include <QDirIterator>
#include <QVector>
```

```
#include <QFileDialog>
#include <QInputDialog>
#include <QTreeWidget>
#include <QMessageBox>
#include <matio.h>

class FileManager
{
public:
   FileManager();
   int Generador();
   void DeviceListUpdate(QList<QString> &DeviceList, QString RutaProjecto);
   //void PathIterator(QString RutaDispositivo, QList<PQFile> );
};

class PQFile
{
public:
   PQFile();
   QString FilePath;
   QString Channel;
   QString RealTIme;
   u_int64_t Secuence;
};

#endif // FILEMANAGER_H
```

```
#ifndef MAINDISPLAY_H
#define MAINDISPLAY_H

#include <QMainWindow>
#include <QtCore>
#include <QtGui>
#include <QTreeWidgetItem>
#include "pq_display.h"
#include "synthetic_display.h"
#include "filegen.h"
#include <QList>
#include <QString>

namespace Ui {
class MainDisplay;
}

class MainDisplay : public QMainWindow
{
   Q_OBJECT

   void AddRoot(QString name);

public:
   explicit MainDisplay(QWidget *parent = 0);
   ~MainDisplay();

private slots:
```

```
    void on_actionAnalizar_Lecturas_triggered();

    void on_actionGenerar_Archivos_triggered();

    void on_treeWidget_itemDoubleClicked(QTreeWidgetItem *item, int column);

    void on_actionImportar_triggered();

private:
    Ui::MainDisplay *ui;
    QList <PQ_Display*> DisplayList;
    QList <Synthetic_display*> SDisplayList;
    QString RutaProyecto;

    int displaycount;
};

#endif // MAINDISPLAY_H

#ifndef PLOTSCENE_H
#define PLOTSCENE_H

#include <qobject.h>
#include <QGraphicsScene>
#include <QGraphicsView>
#include <QGraphicsItem>
#include <math.h>
```

```
class PlotScene : public QGraphicsView
{
   Q_OBJECT
public:
   PlotScene();
   void DrawCircle(void);
   void PhasorPlot(void);
   void fasores(double degree, int phase, int size);

private:
   QGraphicsScene *scene;
};

#endif // PLOTSCENE_H

#ifndef PQ_DISPLAY_H
#define PQ_DISPLAY_H

#include <QDialog>
#include "filemanager.h"
#include <plotscene.h>
#include <fft.h>
#include <gsl/gsl_sf_bessel.h>
#include <gsl/gsl_sort.h>
#include <gsl/gsl_wavelet.h>

namespace Ui {
class PQ_Display;
```

```
}

class PQ_Display : public QDialog
{

    Q_OBJECT

    void AddRootf1(QString name);
    void AddRootf2(QString name);
    void AddRootf3(QString name);

public:
    QList<QVector< double > > Display_y_axis;
    QList<QVector< double > > Display_x_axis;
    QString NameTag;
    QString IterationName;
    QList<QString>  PathList;
    QList<QList<PQFile> >  FullFileDir;
    u_int64_t ReadStart;
    u_int64_t ReadTime;
    u_int64_t ReadEnd;
    int FileCount;
    void PathIterator();
    void SliderSet();
    void PlotSelected();
    explicit PQ_Display(QWidget *parent = 0);
    ~PQ_Display();
    Q_DISABLE_COPY(PQ_Display)
```

```
private slots:
    void on_pushButton_5_clicked();

    void on_horizontalSlider_valueChanged(int value);

    void on_horizontalSlider_3_valueChanged(int value);

    void selectionChanged();
    void mousewheel();



    void on_horizontalSlider_2_valueChanged(int value);

    void on_horizontalSlider_4_valueChanged(int value);

    void on_horizontalSlider_5_valueChanged(int value);

    void on_horizontalSlider_6_valueChanged(int value);

    void on_horizontalSlider_7_valueChanged(int value);

    void on_horizontalSlider_8_valueChanged(int value);

    void on_horizontalSlider_9_valueChanged(int value);

    void on_horizontalSlider_10_valueChanged(int value);
```

```
    void on_horizontalSlider_11_valueChanged(int value);

private:
    Ui::PQ_Display *ui;
    PlotScene *plot1;



};

#endif // PQ_DISPLAY_H

#ifndef SYNTHETIC_DISPLAY_H
#define SYNTHETIC_DISPLAY_H

#include <QDialog>
#include <QList>
#include <QString>
#include <QDebug>
#include <iostream>
#include <QDirIterator>
#include <QVector>
#include <QFileDialog>
#include <QInputDialog>
#include <QTreeWidget>
#include <QMessageBox>
#include <matio.h>
#include <fft.h>
```

```
namespace Ui {
class Synthetic_display;
}

class Synthetic_display : public QDialog
{
   void AddRoot(QString name);

   Q_OBJECT

public:
   QVector<double>Ctime;
   QVector<double>Csignal;

   QVector<double>testtime;
   QVector<double>testsignal;

   explicit Synthetic_display(QWidget *parent = 0);
   ~Synthetic_display();

private:

   void LoadFile(QString path);
   Ui::Synthetic_display *ui;
};

#endif // SYNTHETIC_DISPLAY_H
```

```
#include "fft.h"



fft::fft()
{
}

void fft::stockham(complex x[], int n, int flag, int n2, complex y[])
{
   complex  *y_orig, *tmp;
     int  i, j, k, k2, Ls, r, jrs;
     int  half, m, m2;
     real  wr, wi, tr, ti;

     y_orig = y;
     r = half = n >> 1;
     Ls = 1;

     while(r >= n2) {
       tmp = x;
       x = y;
       y = tmp;
       m = 0;
       m2 = half;
       for(j = 0; j < Ls; ++j) {
         wr = cos(M_PI*j/Ls);
         wi = -flag * sin(M_PI*j/Ls);
```

```
        jrs = j*(r+r);
        for(k = jrs; k < jrs+r; ++k) {
          k2 = k + r;
          tr =  wr*y[k2].Re - wi*y[k2].Im;
          ti =  wr*y[k2].Im + wi*y[k2].Re;
          x[m].Re = y[k].Re + tr;
          x[m].Im = y[k].Im + ti;
          x[m2].Re = y[k].Re - tr;
          x[m2].Im = y[k].Im - ti;
          ++m;
          ++m2;
        }
      }
      r  >>= 1;
      Ls <<= 1;
    };

    if (y != y_orig) {
      for(i = 0; i < n; ++i) {
        y[i] = x[i];
      }
    }

    assert(Ls == n/n2);
    assert(1 == n || m2 == n);
}

void fft::fft1d(complex x[], int n, int flag)
```

```
{
   complex *y;

   assert(1 == flag || -1 == flag);
   y = (complex *) malloc( n*sizeof(complex) );
   assert(NULL != y);
   stockham(x, n, flag, 1, y);
   free(y);
}

#include "filegen.h"

FileGen::FileGen()
{
}

void FileGen::Generator(QString ProjectRoute)
{
   QString ProjectRoute2="./pqExtract -s /dev/mmcblk0 -o ";
   ProjectRoute2 += ProjectRoute;
   ProjectRoute2+=" -c pquaq.csv -f dia=-1/hora=24/minz=6 -n 4800000";
   system(ProjectRoute2.toStdString().c_str());
}

#include "filemanager.h"

FileManager::FileManager()
{
```

```
}
int FileManager::Generador()
{
   return 0;
}
void FileManager::DeviceListUpdate(QList<QString> &DeviceList , QString RutaProjecto)
{

}

PQFile::PQFile()
{

}

#include "maindisplay.h"
#include <QApplication>

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

   return a.exec();
}

#include "maindisplay.h"
```

```
#include "ui_maindisplay.h"
#include "pq_display.h"
#include "filemanager.h"

MainDisplay::MainDisplay(QWidget *parent) :
   QMainWindow(parent),
   ui(new Ui::MainDisplay)
{
   ui->setupUi(this);




}

MainDisplay::~MainDisplay()
{
   for(int i=0; i<DisplayList.size();i++)
   {
      DisplayList[i]->close();
   }
   while(!DisplayList.isEmpty())
      {
         DisplayList.pop_front();
      }
   delete ui;
}

void MainDisplay::AddRoot(QString name)
```

```
{
   QTreeWidgetItem *itm = new QTreeWidgetItem(ui->treeWidget);
   itm->setText(0,name);
   ui->treeWidget->addTopLevelItem(itm);
}

void MainDisplay::on_actionAnalizar_Lecturas_triggered()
{
   for(int i=0; i<DisplayList.size();i++)
   {
      DisplayList[i]->close();
   }
   while(!DisplayList.isEmpty())
      {
         DisplayList.pop_front();
      }
   for(int i=0; i<DisplayList.size();i++)
   {
      DisplayList[i]->close();
   }
   ui->treeWidget->clear();
   bool flag=true;
   RutaProyecto = QFileDialog::getExistingDirectory(this,tr("Selecione locacion del
proyecto"),"/run/user/1000/gvfs/ftp:host=calenerhosp.uva.es/datos/",QFileDialog::DontUseCustomDire
ctoryIcons);
   QDirIterator it(RutaProyecto.toStdString().c_str());
   while(it.hasNext())
   {
      QDir DevicePath(it.next());
```

```
            for(QString::const_iterator
itr(DevicePath.dirName().begin());itr!=DevicePath.dirName().end();itr++)
            {
               if(*itr == '.')
               {
                  flag = false;
               }
            }
            if(flag == true)
            {
               AddRoot(DevicePath.dirName());
            }
            flag=true;
   }
}

void MainDisplay::on_actionGenerar_Archivos_triggered()
{
   QString ProjectRoute = QFileDialog::getExistingDirectory(this,tr("Slect project
location"),"/path/to/file/",QFileDialog::ShowDirsOnly | QFileDialog::DontResolveSymlinks);
   FileGen NewGen;
   NewGen.Generator(ProjectRoute);
}

void MainDisplay::on_treeWidget_itemDoubleClicked(QTreeWidgetItem *item, int column)
{
   QList<QTreeWidgetItem *> ItemList;
   ItemList = ui->treeWidget->selectedItems();
   foreach (item, ItemList)
```

```
{
   RutaProyecto+="/";
   RutaProyecto+=item->text(column);

}

bool flag=true;
qDebug()<<DisplayList.size();
if(DisplayList.size()==0)
{
   PQ_Display *NewDispplay = new PQ_Display();
   NewDispplay->setWindowTitle(RutaProyecto);
   DisplayList.push_back(NewDispplay);
   DisplayList[0]->NameTag=RutaProyecto;

   DisplayList[0]->PathIterator();
   DisplayList[0]->SliderSet();
   DisplayList[0]->open();



}
else
{
   for(int i=0;i<DisplayList.size();i++)
   {
      if(DisplayList[i]->NameTag==RutaProyecto)
      {
         flag=false;
```

```
            qDebug()<<"Already Running";
            if(DisplayList[i]->isHidden())
            {DisplayList[i]->show();}
          }
        }
        if(flag)
        {
          PQ_Display *NewDisplay = new PQ_Display();
          NewDisplay->NameTag = RutaProyecto;
          NewDisplay->setWindowTitle(RutaProyecto);
          DisplayList.push_back(NewDisplay);

          DisplayList[DisplayList.size()-1]->PathIterator();
          DisplayList[DisplayList.size()-1]->open();
        }
        flag = true;
    }

    foreach(item, ItemList)
    {
      RutaProyecto.chop(item->text(column).size()+1);
    }
}

void MainDisplay::on_actionImportar_triggered()
{
    Synthetic_display *newsynth = new Synthetic_display();
    newsynth->setWindowTitle("Import synthetic signal");
```

```
    SDisplayList.push_back(newsynth);
    SDisplayList[SDisplayList.size()-1]->open();
}

#include "plotscene.h"
#include <qdebug.h>

PlotScene::PlotScene()
{
    QBrush blackBrush(Qt::white);
    scene = new QGraphicsScene(this);
    setScene(scene);
    setBackgroundBrush(blackBrush);
    setRenderHints(QPainter::Antialiasing | QPainter::SmoothPixmapTransform);
}

void PlotScene::DrawCircle(void)
{
    QGraphicsSimpleTextItem *text1;


    /*text1 = scene->addSimpleText("Phasors0",QFont("Arial",12));
    text1->setPos(0,0); text1->setBrush(QBrush(Qt::white));
    text1->setFlag(QGraphicsItem::ItemIsMovable);*/

    text1 = scene->addSimpleText("0",QFont("Arial",10));
    text1->setPos(210,-10); text1->setBrush(QBrush(Qt::black));
    text1->setFlag(QGraphicsItem::ItemIsMovable);
```

```
text1 = scene->addSimpleText("90",QFont("Arial",10));
text1->setPos(-5,-220); text1->setBrush(QBrush(Qt::black));
text1->setFlag(QGraphicsItem::ItemIsMovable);

text1 = scene->addSimpleText("180",QFont("Arial",10));
text1->setPos(-225,-10); text1->setBrush(QBrush(Qt::black));
text1->setFlag(QGraphicsItem::ItemIsMovable);

text1 = scene->addSimpleText("270",QFont("Arial",10));
text1->setPos(-10,210); text1->setBrush(QBrush(Qt::black));
text1->setFlag(QGraphicsItem::ItemIsMovable);
QVector <double> circle_x,circle_y;

int radio=200;
for(double i=0;i<2*M_PI;i+=((2*M_PI)/36))
{
   circle_y.push_back(sin(i)*radio);
   circle_x.push_back(cos(i)*radio);
   //qDebug()<<"x = "<<sin(i)*radio<<"--- y = "<<cos(i)*radio;
}

QPainterPath line;
QPen outlinePen(Qt::black);
outlinePen.setStyle(Qt::DashLine);
outlinePen.setWidth(1);
line.moveTo(circle_x[0],circle_y[0]);
for (int i=1;i<circle_x.size();i++)
```

```
    {
        line.lineTo(circle_x[i],circle_y[i]);
    }

    for(int i=0;i<circle_x.size();i+=3)
    {
        line.moveTo(0,0);
        line.lineTo(circle_x[i],circle_y[i]);
    }

    scene->addPath(line,outlinePen);
    //text1->setPos(); text1->setBrush(QBrush(Qt::white));
}

void PlotScene::PhasorPlot(void)
{
    QGraphicsSimpleTextItem *text1;


    /*text1 = scene->addSimpleText("Phasors0",QFont("Arial",12));
    text1->setPos(0,0); text1->setBrush(QBrush(Qt::white));
    text1->setFlag(QGraphicsItem::ItemIsMovable);*/

    text1 = scene->addSimpleText("0",QFont("Arial",10));
    text1->setPos(210,-10); text1->setBrush(QBrush(Qt::white));
    text1->setFlag(QGraphicsItem::ItemIsMovable);

    text1 = scene->addSimpleText("90",QFont("Arial",10));
```

```
text1->setPos(-5,-220); text1->setBrush(QBrush(Qt::white));
text1->setFlag(QGraphicsItem::ItemIsMovable);

text1 = scene->addSimpleText("180",QFont("Arial",10));
text1->setPos(-225,-10); text1->setBrush(QBrush(Qt::white));
text1->setFlag(QGraphicsItem::ItemIsMovable);

text1 = scene->addSimpleText("270",QFont("Arial",10));
text1->setPos(-10,210); text1->setBrush(QBrush(Qt::white));
text1->setFlag(QGraphicsItem::ItemIsMovable);
QVector <double> circle_x,circle_y;

int radio=200;
for(double i=0;i<2*M_PI;i+=((2*M_PI)/36))
{
   circle_y.push_back(sin(i)*radio);
   circle_x.push_back(cos(i)*radio);
   //qDebug()<<"x = "<<sin(i)*radio<<"--- y = "<<cos(i)*radio;
}

QPainterPath line;
QPen outlinePen(Qt::white);
outlinePen.setStyle(Qt::DashLine);
outlinePen.setWidth(1);
line.moveTo(circle_x[0],circle_y[0]);
for (int i=1;i<circle_x.size();i++)
{
   line.lineTo(circle_x[i],circle_y[i]);
```

```
}

for(int i=0;i<circle_x.size();i+=3)
{
   line.moveTo(0,0);
   line.lineTo(circle_x[i],circle_y[i]);
}

scene->addPath(line,outlinePen);
//text1->setPos(); text1->setBrush(QBrush(Qt::white));

double angle=M_PI/4; //in rads


//for(angle=0;angle<(M_PI*2);angle+=0.1)
//{
double radius=150;
 QPointF origin(0,0), end(0,0);

    end.setY( -1*(sin(angle)*radius));
    end.setX( (cos(angle)*radius));

    QPen _Pen;
    _Pen.setColor(Qt::red);
    _Pen.setWidth(3);

    QGraphicsLineItem* _Line=new QGraphicsLineItem(origin.x(),origin.y(),end.x(),end.y());
    _Line->setPen(_Pen);
```

```
_Line->setVisible(true);
_Line->setFlags(QGraphicsLineItem::ItemIsSelectable | QGraphicsLineItem::ItemIsMovable);

scene->addItem(_Line);

QTransform t;
t.rotate(90-angle*(180/M_PI));

QPen HeadConf(Qt::red);
HeadConf.setWidth(2);
QBrush HeadFill(Qt::SolidPattern);
HeadFill.setColor(Qt::red);
double HeadSize=10;

QPointF p1(0,0),p2(0,0),p3(0,0);
p1.setX(HeadSize/2);
p1.setY(0);

p2.setX(0);
p2.setY(-HeadSize);

p3.setX(-HeadSize/2);
p3.setY(0);

QPolygonF arrow;
arrow<<p1<<p2<<p3<<p1;

QPolygonF NewHead = t.map(arrow);
```

```
        NewHead.translate(end);

        scene->addPolygon(NewHead,HeadConf,HeadFill);

    //}

}

void PlotScene::fasores(double degree, int phase, int size)
{
    double angle=degree; //in rads

        QPen _Pen;
        QBrush HeadFill(Qt::SolidPattern);
        QPen HeadConf;
    switch (phase)
    {
    case 1:
        HeadFill.setColor(Qt::darkGreen);
        _Pen.setColor(Qt::darkGreen);
        HeadConf.setColor(Qt::darkGreen);
        break;
    case 2:
        HeadFill.setColor(Qt::red);
        _Pen.setColor(Qt::red);
        HeadConf.setColor(Qt::red);
        break;
    default:
```

```
    HeadFill.setColor(Qt::blue);
    _Pen.setColor(Qt::blue);
    HeadConf.setColor(Qt::blue);
    break;
}


double radius=150/size;
 QPointF origin(0,0), end(0,0);

     end.setY( -1*(sin(angle)*radius));
     end.setX( (cos(angle)*radius));



     _Pen.setWidth(3);

     QGraphicsLineItem* _Line=new QGraphicsLineItem(origin.x(),origin.y(),end.x(),end.y());
     _Line->setPen(_Pen);
     _Line->setVisible(true);
     _Line->setFlags(QGraphicsLineItem::ItemIsSelectable | QGraphicsLineItem::ItemIsMovable);

     scene->addItem(_Line);

     QTransform t;
     t.rotate(90-angle*(180/M_PI));
```

```
        HeadConf.setWidth(2/size);



        double HeadSize=10;

        QPointF p1(0,0),p2(0,0),p3(0,0);
        p1.setX(HeadSize/2);
        p1.setY(0);

        p2.setX(0);
        p2.setY(-HeadSize);

        p3.setX(-HeadSize/2);
        p3.setY(0);

        QPolygonF arrow;
        arrow<<p1<<p2<<p3<<p1;

        QPolygonF NewHead = t.map(arrow);
        NewHead.translate(end);

        scene->addPolygon(NewHead,HeadConf,HeadFill);
}

#include "pq_display.h"
#include "ui_pq_display.h"

#include <stdio.h>
```

```
#include <string>



PQ_Display::PQ_Display(QWidget *parent) :
    QDialog(parent),
    ui(new Ui::PQ_Display)
{
    ui->setupUi(this);




    plot1 = new PlotScene();
    plot1->setMinimumHeight(300);

    ui->tab_4->layout()->addWidget(plot1);

    connect(ui->widget, SIGNAL(selectionChangedByUser()),this, SLOT(selectionChanged()));
    connect(ui->widget, SIGNAL(mouseWheel(QWheelEvent*)),this, SLOT(mousewheel()));
    connect(ui->widget_2, SIGNAL(mouseWheel(QWheelEvent*)),this, SLOT(mousewheel()));


}

PQ_Display::~PQ_Display()
{
    for(int i=0;i<FullFileDir.size();i++)
    {
        for(int j=0;j<FullFileDir[i].size();j++)
```

```
        {
            FullFileDir[i].pop_front();
        }
        FullFileDir.pop_front();
    }
    delete ui;
}

void PQ_Display::AddRootf1(QString name)
{
    QTreeWidgetItem *itm = new QTreeWidgetItem(ui->treeWidget);
    itm->setText(0,name);
    ui->treeWidget->addTopLevelItem(itm);
}

void PQ_Display::AddRootf2(QString name)
{
    QTreeWidgetItem *itm = new QTreeWidgetItem(ui->treeWidget_2);
    itm->setText(0,name);
    ui->treeWidget_2->addTopLevelItem(itm);
}

void PQ_Display::AddRootf3(QString name)
{
    QTreeWidgetItem *itm = new QTreeWidgetItem(ui->treeWidget_3);
    itm->setText(0,name);
    ui->treeWidget_3->addTopLevelItem(itm);
}
```

```
void PQ_Display::PathIterator()
{
    ReadStart=0;
    ReadTime=0;
    ReadEnd=0;

    int VarCount=0;
    FileCount=0;
    QString VarName;
    QDirIterator DirIt(NameTag.toStdString().c_str(),QDirIterator::Subdirectories);
    while(DirIt.hasNext())
    {
        DirIt.next();
        if(QFileInfo(DirIt.filePath()).isFile()&&QFileInfo(DirIt.filePath()).suffix()=="mat")
        {
            if(PathList.isEmpty())
            {
                VarName = DirIt.fileName();
                VarName.chop(4);
                if(VarName != "rd")
                {
                    PathList.append(VarName);
                }
            }
            else
            {
                VarName = DirIt.fileName();
```

```
            VarName.chop(4);
            for(int i=0;i<PathList.size();i++)
            {
               if(PathList[i]==VarName || VarName=="rd")
               {
                  break;

               }
               if(PathList[i]!=VarName&& i == PathList.size()-1)
               {
                  PathList.append(VarName);

               }
            }
         }
      }
      if(VarCount > 2)
      {
         break;
      }

}
//qDebug()<<"Full Dir";
for(int i=0;i<PathList.size();i++)
{
   PQFile NewPQFile;
   NewPQFile.Secuence=0;
   QList<PQFile> NewList;
```

```
    NewPQFile.Channel = PathList[i];
    NewList.append(NewPQFile);
    FullFileDir.append(NewList);
    //qDebug()<<FullFileDir[i][0].Channel;
  }
  bool error=false;
  int start[2]={0,0},edge[2]={1,1},stride[2]={1,1},err;
  QDirIterator
DirIt2(NameTag.toStdString().c_str(),QDirIterator::Subdirectories|QDirIterator::FollowSymlinks);
  while(DirIt2.hasNext())
  {
    DirIt2.next();
    if(QFileInfo(DirIt2.filePath()).isFile()&&QFileInfo(DirIt2.filePath()).suffix()=="mat")
    {
      VarName = DirIt2.fileName();
      VarName.chop(4);
      for(int i=0;i<FullFileDir.size();i++)
      {
        if(VarName==FullFileDir[i][0].Channel)
        {
          PQFile NewPQFile;
          NewPQFile.FilePath = DirIt2.filePath();

          mat_t *matfp;
          matvar_t *matvar;

          matfp = Mat_Open(DirIt2.filePath().toStdString().c_str(),MAT_ACC_RDONLY);
          if(NULL != matfp)
```

```
{
    matvar = Mat_VarReadInfo(matfp,"RealTime");
    char *TimeString = new char[matvar->dims[1]];
    //qDebug()<<matvar->dims[1];
    edge[0]=matvar->dims[0];
    edge[1]=matvar->dims[1];
    Mat_VarReadDataAll(matfp,matvar);
    TimeString = (char*)matvar->data;
    NewPQFile.RealTIme = QString::fromUtf8(TimeString);
    //qDebug()<<NewPQFile.RealTIme;
    NewPQFile.Secuence =
((NewPQFile.RealTIme.split(":")[0].toUInt())*86400000)+((NewPQFile.RealTIme.split(":")[1].toUInt())*36
00000)+((NewPQFile.RealTIme.split(":")[2].toUInt())*60000)+((NewPQFile.RealTIme.split(":")[3].toDoubl
e())*1000);
    if(FileCount==0)
    {
       ReadStart = NewPQFile.Secuence;
    }
    //qDebug()<<NewPQFile.Secuence;
    /*
    qDebug()<<"File Opened";
    matvar = Mat_VarReadInfo(matfp,"SampleNumber");
    //char *TimeString = new char[matvar->dims[1]];
    u_int64_t *sn = new u_int64_t[matvar->dims[1]];
    qDebug()<<matvar->dims[1];
    edge[0]=matvar->dims[0];
    edge[1]=matvar->dims[1];
    err=Mat_VarReadData(matfp,matvar,sn,start,stride,edge);
    Mat_VarPrint(matvar,0);
```

```
            //NewPQFile.RealTIme = string(TimeString);
            qDebug()<<sn[0];*/
        }
        else
        {
            qDebug()<<"File Not Opened cycle dropped";
            error = true;
        }
        FullFileDir[i].append(NewPQFile);
        /*for(int k=0;k<FullFileDir[i].size();k++)
        {
            qDebug()<<"millis "<<FullFileDir[i][k].Secuence;
        }*/
        Mat_VarFree(matvar);
        Mat_Close(matfp);
        FileCount++;
      }
    }
  }

  if(error)
  {
    break;
  }
}
qDebug()<<"Files chaecked"<<FileCount;
mat_t *matfp;
```

```
   matvar_t *matvar;
   matfp = Mat_Open(FullFileDir[0][FullFileDir[0].size()-
1].FilePath.toStdString().c_str(),MAT_ACC_RDONLY);
   if(NULL != matfp)
   {
      matvar = Mat_VarReadInfo(matfp,"Data");
      ReadEnd = matvar->dims[1];
   }
   Mat_VarFree(matvar);
   Mat_Close(matfp);

   /*for(int i=0;i<FullFileDir.size();i++)
   {
      for(int j=1;j<FullFileDir[i].size();j++)
      {
         //qDebug()<<FullFileDir[i][j].FilePath;
         //qDebug()<<FullFileDir[i][j].RealTIme;
         //qDebug()<<FullFileDir[i][j].Secuence;
         if(FullFileDir[i][j-1].Secuence > FullFileDir[i][j].Secuence)
         qDebug()<<"Sequence error";

      }
   }*/
   /*qDebug()<<"Days"<<FullFileDir[0][FullFileDir[0].size()-1].Secuence/86400000;
   qDebug()<<"Hours"<<(FullFileDir[0][FullFileDir[0].size()-1].Secuence%86400000)/3600000;
   qDebug()<<"Minutes"<<((FullFileDir[0][FullFileDir[0].size()-1].Secuence%86400000)%3600000)/60000;
   qDebug()<<"Seconds"<<(((FullFileDir[0][FullFileDir[0].size()-
1].Secuence%86400000)%3600000)%60000)/1000;
```

```
   qDebug()<<"millis"<<(((FullFileDir[0][FullFileDir[0].size()-
1].Secuence%86400000)%3600000)%60000)%1000;*/


   //qDebug()<<" Vars Found";


      //qDebug()<<PathList;
}


void PQ_Display::on_pushButton_5_clicked()
{


   //this->PlotSelected();
   QVector <double> Ia,Ib,Ic,Va,Vb,Vc;
   int cerocount=0;
   double Iarms=0,Ibrms=0,Icrms=0,Varms=0,Vbrms=0,Vcrms=0;///RMS
   double Vapico=0,Vbpico=0,Vcpico=0,Iapico=0,Ibpico=0,Icpico=0;///Valorespico
   double Sa=0,Sb=0,Sc=0;///potencia aparente
   double Pa=0,Pb=0,Pc=0;///potencia activa
   double Qa=0,Qb=0,Qc=0;///potencia reactiva
   double PFa=0,PFb=0,PFc=0;///factor de potencia
   double Rea=0,Reb=0,Rec=0,Ima=0,Imb=0,Imc=0;///Componentes reales e imaginarios de
componentes armonicos
   QString printvalue;
   QVector <double> harmonics;
   QVector <double>
RmsVa,RmsVb,RmsVc,RmsIa,RmsIb,RmsIc,PowerFa,PowerFb,PowerFc,PQIndexXaxis;
   int start[2]={0,0},edge[2]={1,1},stride[2]={1,1},err;
```

```
QString IteratorName2;
IteratorName2 = IterationName;


mat_t *matfp;
matvar_t *matvar;

IteratorName2+="Ia.mat";
matfp = Mat_Open(IteratorName2.toStdString().c_str(),MAT_ACC_RDONLY);
IteratorName2.chop(6);


matvar = Mat_VarReadInfo(matfp,"Gain");
double *Gain1 = new double[matvar->dims[1]];
edge[0]=matvar->dims[0];
edge[1]=matvar->dims[1];
Mat_VarReadData(matfp,matvar,Gain1,start,stride,edge);
//qDebug()<<"Gain = "<<Gain[0];

matvar = Mat_VarReadInfo(matfp,"Offset");
double *Offset1 = new double[matvar->dims[1]];
edge[0]=matvar->dims[0];
edge[1]=matvar->dims[1];
Mat_VarReadData(matfp,matvar,Offset1,start,stride,edge);
//qDebug()<<"Offset = "<<Offset[0];


matvar=Mat_VarReadInfo(matfp,"Data");
```

```
 QVector <double> tiempo;

mat_int16_t *predestino1 = new mat_int16_t[matvar->dims[1]];
edge[0]=matvar->dims[0];
edge[1]=matvar->dims[1];

err = Mat_VarReadData(matfp,matvar,predestino1,start,stride,edge);

 for(int i=0;i<matvar->dims[1];i++)
 {
    Ia.push_back((predestino1[i]*Gain1[0])-(Gain1[0]*Offset1[0]));
    tiempo.push_back(i);
 }
 std::free(predestino1);
 std::free(Gain1);
 std::free(Offset1);



 /////////////////


 IteratorName2+="Ib.mat";
 matfp = Mat_Open(IteratorName2.toStdString().c_str(),MAT_ACC_RDONLY);
 IteratorName2.chop(6);
```

```
matvar = Mat_VarReadInfo(matfp,"Gain");
double *Gain2 = new double[matvar->dims[1]];
edge[0]=matvar->dims[0];
edge[1]=matvar->dims[1];
Mat_VarReadData(matfp,matvar,Gain2,start,stride,edge);
//qDebug()<<"Gain = "<<Gain[0];

matvar = Mat_VarReadInfo(matfp,"Offset");
double *Offset2 = new double[matvar->dims[1]];
edge[0]=matvar->dims[0];
edge[1]=matvar->dims[1];
Mat_VarReadData(matfp,matvar,Offset2,start,stride,edge);
//qDebug()<<"Offset = "<<Offset[0];

matvar=Mat_VarReadInfo(matfp,"Data");

mat_int16_t *predestino2 = new mat_int16_t[matvar->dims[1]];
edge[0]=matvar->dims[0];
edge[1]=matvar->dims[1];

err = Mat_VarReadData(matfp,matvar,predestino2,start,stride,edge);

for(int i=0;i<matvar->dims[1];i++)
{
   Ib.push_back((predestino2[i]*Gain2[0])-(Gain2[0]*Offset2[0]));
}
std::free(predestino2);
std::free(Gain2);
```

```
std::free(Offset2);

/////////////////

IteratorName2+="Ic.mat";
matfp = Mat_Open(IteratorName2.toStdString().c_str(),MAT_ACC_RDONLY);
IteratorName2.chop(6);


matvar = Mat_VarReadInfo(matfp,"Gain");
double *Gain3 = new double[matvar->dims[1]];
edge[0]=matvar->dims[0];
edge[1]=matvar->dims[1];
Mat_VarReadData(matfp,matvar,Gain3,start,stride,edge);
//qDebug()<<"Gain = "<<Gain[0];

matvar = Mat_VarReadInfo(matfp,"Offset");
double *Offset3 = new double[matvar->dims[1]];
edge[0]=matvar->dims[0];
edge[1]=matvar->dims[1];
Mat_VarReadData(matfp,matvar,Offset3,start,stride,edge);
//qDebug()<<"Offset = "<<Offset[0];

matvar=Mat_VarReadInfo(matfp,"Data");

mat_int16_t *predestino3 = new mat_int16_t[matvar->dims[1]];
edge[0]=matvar->dims[0];
edge[1]=matvar->dims[1];
```

```
err = Mat_VarReadData(matfp,matvar,predestino3,start,stride,edge);

 for(int i=0;i<matvar->dims[1];i++)
 {
    Ic.push_back((predestino3[i]*Gain3[0])-(Gain3[0]*Offset3[0]));
 }
 std::free(predestino3);
 std::free(Gain3);
 std::free(Offset3);

 /////////////////


 IteratorName2+="Va.mat";
 matfp = Mat_Open(IteratorName2.toStdString().c_str(),MAT_ACC_RDONLY);
 IteratorName2.chop(6);

 matvar = Mat_VarReadInfo(matfp,"Gain");
 double *Gain4 = new double[matvar->dims[1]];
 edge[0]=matvar->dims[0];
 edge[1]=matvar->dims[1];
 Mat_VarReadData(matfp,matvar,Gain4,start,stride,edge);
 //qDebug()<<"Gain = "<<Gain[0];

 matvar = Mat_VarReadInfo(matfp,"Offset");
 double *Offset4 = new double[matvar->dims[1]];
 edge[0]=matvar->dims[0];
```

```
 edge[1]=matvar->dims[1];
 Mat_VarReadData(matfp,matvar,Offset4,start,stride,edge);
 //qDebug()<<"Offset = "<<Offset[0];

 matvar=Mat_VarReadInfo(matfp,"Data");

mat_int16_t *predestino4 = new mat_int16_t[matvar->dims[1]];
edge[0]=matvar->dims[0];
edge[1]=matvar->dims[1];

err = Mat_VarReadData(matfp,matvar,predestino4,start,stride,edge);

 for(int i=0;i<matvar->dims[1];i++)
 {
    Va.push_back((predestino4[i]*Gain4[0])-(Gain4[0]*Offset4[0]));
 }
 std::free(predestino4);
 std::free(Gain4);
 std::free(Offset4);

 ////////////////

 IteratorName2+="Vb.mat";
 matfp = Mat_Open(IteratorName2.toStdString().c_str(),MAT_ACC_RDONLY);
 IteratorName2.chop(6);

 matvar = Mat_VarReadInfo(matfp,"Gain");
 double *Gain5 = new double[matvar->dims[1]];
```

```
 edge[0]=matvar->dims[0];
 edge[1]=matvar->dims[1];
 Mat_VarReadData(matfp,matvar,Gain5,start,stride,edge);
 //qDebug()<<"Gain = "<<Gain[0];

 matvar = Mat_VarReadInfo(matfp,"Offset");
 double *Offset5 = new double[matvar->dims[1]];
 edge[0]=matvar->dims[0];
 edge[1]=matvar->dims[1];
 Mat_VarReadData(matfp,matvar,Offset5,start,stride,edge);
 //qDebug()<<"Offset = "<<Offset[0];

 matvar=Mat_VarReadInfo(matfp,"Data");

mat_int16_t *predestino5 = new mat_int16_t[matvar->dims[1]];
edge[0]=matvar->dims[0];
edge[1]=matvar->dims[1];

err = Mat_VarReadData(matfp,matvar,predestino5,start,stride,edge);

 for(int i=0;i<matvar->dims[1];i++)
 {
    Vb.push_back((predestino5[i]*Gain5[0])-(Gain5[0]*Offset5[0]));
 }
 std::free(predestino5);
 std::free(Gain5);
 std::free(Offset5);
```

```
/////////////////


 IteratorName2+="Vc.mat";
 matfp = Mat_Open(IteratorName2.toStdString().c_str(),MAT_ACC_RDONLY);
 IteratorName2.chop(6);

 matvar = Mat_VarReadInfo(matfp,"Gain");
 double *Gain6 = new double[matvar->dims[1]];
 edge[0]=matvar->dims[0];
 edge[1]=matvar->dims[1];
 Mat_VarReadData(matfp,matvar,Gain6,start,stride,edge);
 //qDebug()<<"Gain = "<<Gain[0];

 matvar = Mat_VarReadInfo(matfp,"Offset");
 double *Offset6 = new double[matvar->dims[1]];
 edge[0]=matvar->dims[0];
 edge[1]=matvar->dims[1];
 Mat_VarReadData(matfp,matvar,Offset6,start,stride,edge);
 //qDebug()<<"Offset = "<<Offset[0];

 matvar=Mat_VarReadInfo(matfp,"Data");

mat_int16_t *predestino6 = new mat_int16_t[matvar->dims[1]];
edge[0]=matvar->dims[0];
edge[1]=matvar->dims[1];

err = Mat_VarReadData(matfp,matvar,predestino6,start,stride,edge);
```

```
for(int i=0;i<matvar->dims[1];i++)
{
   Vc.push_back((predestino6[i]*Gain6[0])-(Gain6[0]*Offset6[0]));
}
std::free(predestino6);
std::free(Gain6);
std::free(Offset6);

/////////////////

Mat_VarFree(matvar);
Mat_Close(matfp);
//qDebug()<<Ia.size()<<" "<<Ib.size()<<" "<<Ic.size()<<" "<<Va.size()<<" "<<Vb.size()<<" "<<Vc.size();
double interval;
int RMSInterval = 1600;
for(int k=0;k<4800000/RMSInterval;k++)
{
   Iarms = 0;
   Ibrms = 0;
   Icrms = 0;
   Varms = 0;
   Vbrms = 0;
   Vcrms = 0;
   Pa = 0;
   Pb = 0;
   Pc = 0;
for(int i=k*RMSInterval;i<((k*RMSInterval)+RMSInterval);i++)
```

```
{
   /*Iarms = Iarms+((pow(Ia[i],2.0))/RMSInterval);
   Ibrms = Ibrms+((pow(Ib[i],2.0))/RMSInterval);
   Icrms = Icrms+((pow(Ic[i],2.0))/RMSInterval);
   Varms = Varms+((pow(Va[i],2.0))/RMSInterval);
   Vbrms = Vbrms+((pow(Vb[i],2.0))/RMSInterval);
   Vcrms = Vcrms+((pow(Vc[i],2.0))/RMSInterval);*/

   Iarms = Iarms+((pow(Ia[i],2.0))/RMSInterval);
   Ibrms = Ibrms+((pow(Ib[i],2.0))/RMSInterval);
   Icrms = Icrms+((pow(Ic[i],2.0))/RMSInterval);
   Varms = Varms+((pow(Va[i],2.0))/RMSInterval);
   Vbrms = Vbrms+((pow(Vb[i],2.0))/RMSInterval);
   Vcrms = Vcrms+((pow(Vc[i],2.0))/RMSInterval);

   Pa = Pa + (Ia[i]*Va[i]/RMSInterval);
   Pb = Pb + (Ib[i]*Vb[i]/RMSInterval);
   Pc = Pc + (Ic[i]*Vc[i]/RMSInterval);
}


   Iarms = sqrt(Iarms);
   Ibrms = sqrt(Ibrms);
   Icrms = sqrt(Icrms);
   Varms = sqrt(Varms);
   Vbrms = sqrt(Vbrms);
   Vcrms = sqrt(Vcrms);
```

```
RmsVa.push_back(Varms);
RmsVb.push_back(Vbrms);
RmsVc.push_back(Vcrms);

RmsIa.push_back(Iarms);
RmsIb.push_back(Ibrms);
RmsIc.push_back(Icrms);

Sa = Varms*Iarms;
Sb = Vbrms*Ibrms;
Sc = Vcrms*Icrms;

/*Qa = sqrt(pow(Sa,2.0)-pow(Pa,2.0));
Qb = sqrt(pow(Sb,2.0)-pow(Pb,2.0));
Qc = sqrt(pow(Sc,2.0)-pow(Pc,2.0));*/

PFa = Pa/Sa;
PFb = Pb/Sb;
PFc = Pc/Sc;

PowerFa.push_back(PFa);
PowerFb.push_back(PFb);
PowerFc.push_back(PFc);
interval = k;
PQIndexXaxis.push_back(interval/5);
}
```

```
double estimate = 233.3452;
double magnitudef1 = 0,magnitudef2 = 0,magnitudef3 = 0;
for(int i=0;i<RmsVa.size();i++)
{
   ///para fase 1
   magnitudef1 = RmsVa[i]/estimate;
   if(magnitudef1 == 0)
   {
      AddRootf1("Interrupcion encontrana en t = "+ QString::number(i/5));
   }
   if(magnitudef1 > 1.1)
   {
      AddRootf1("Swell detectado en t = "+ QString::number(i/5) );
   }
   if(magnitudef1 <  1 && magnitudef1 > 0)
   {
      AddRootf1("Sag detectado en t = "+ QString::number(i/5));
   }

   ///para fase 1
}

int  N=4194304;
int  N2=524288;

complex *fftData;
fftData = new complex[N];
```

```
    QVector<double> dftVA,dftVb,dftVc,dftIA,dftIb,dftIc;
    //QVector<double> waveVa,waveVb,waveVc,waveIa,waveIb,waveIc;
    QVector<double>
periodogramVa,periodogramVb,periodogramVc,periodogramIa,periodogramIb,periodogramIc;
    QVector<double> fftXaxis,periodogramXaxis,waveXaxis;
    for(int i=0;i<N;i++)
    {
        fftData[i] = {Va[i],0};
    }

    fft newfft;
    newfft.fft1d(fftData,N,1);

    double absolute;
    double psdx;
    double frqX=0;
    for(int i=0;i<N/2+1;i++)
    {
        absolute = sqrt(fftData[i].Re*fftData[i].Re+fftData[i].Im*fftData[i].Im);
        psdx = (1.0/(8000.0*N));
        periodogramVa.push_back(10.0*log(psdx * pow(absolute,2)));
        if(i==N/2)
        {
            periodogramVa[i] = 10*log(2*periodogramVa[i]);
        }

        dftVA.push_back(absolute);
        frqX = (2*3.1416)/(N*.000125);
```

```
   fftXaxis.push_back(i*(frqX/(2*3.1416)));
}

ui->fftva->addGraph();
ui->fftva->graph(0)->setData(fftXaxis,dftVA);
ui->fftva->graph(0)->setPen(QPen(Qt::red,2));
ui->fftva->yAxis->setLabel("FFT Va |y(f)|");
ui->fftva->xAxis->setRange(0,10);
ui->fftva->xAxis->setLabel("Frecuencia (Hz)");
ui->fftva->yAxis->rescale();
ui->fftva->replot();

ui->SpecDenVa->addGraph();
ui->SpecDenVa->graph(0)->setData(fftXaxis,periodogramVa);
ui->SpecDenVa->graph(0)->setPen(QPen(Qt::red,2));
ui->SpecDenVa->yAxis->setLabel("Periodograma Va");
ui->SpecDenVa->xAxis->setRange(0,10);
ui->SpecDenVa->xAxis->setLabel("Frecuencia (Hz)");
ui->SpecDenVa->yAxis->rescale();
ui->SpecDenVa->replot();
///////
for(int i=0;i<N;i++)
{
   fftData[i] = {Vb[i],0};
}
newfft.fft1d(fftData,N,1);

for(int i=0;i<N/2+1;i++)
```

```
{
   absolute = sqrt(fftData[i].Re*fftData[i].Re+fftData[i].Im*fftData[i].Im);
   psdx = (1.0/(8000.0*N));
   periodogramVb.push_back(10.0*log(psdx * pow(absolute,2)));
   if(i==N/2)
   {
      periodogramVb[i] = 10*log(2*periodogramVb[i]);
   }

   dftVb.push_back(absolute);
   //fftXaxis.push_back(i);
}

ui->fftvb->addGraph();
ui->fftvb->graph(0)->setData(fftXaxis,dftVb);
ui->fftvb->graph(0)->setPen(QPen(Qt::red,2));
ui->fftvb->yAxis->setLabel("FFT Vb |y(f)|");
ui->fftvb->xAxis->setRange(0,10);
ui->fftvb->xAxis->setLabel("Frecuencia (Hz)");
ui->fftvb->yAxis->rescale();
ui->fftvb->replot();

ui->SpecDenVb->addGraph();
ui->SpecDenVb->graph(0)->setData(fftXaxis,periodogramVb);
ui->SpecDenVb->graph(0)->setPen(QPen(Qt::red,2));
ui->SpecDenVb->yAxis->setLabel("Periodograma Vb");
ui->SpecDenVb->xAxis->setRange(0,10);
ui->SpecDenVb->xAxis->setLabel("Frecuencia (Hz)");
```

```
ui->SpecDenVb->yAxis->rescale();
ui->SpecDenVb->replot();
//////
for(int i=0;i<N;i++)
{
   fftData[i] = {Vc[i],0};
}
newfft.fft1d(fftData,N,1);

for(int i=0;i<N/2+1;i++)
{
   absolute = sqrt(fftData[i].Re*fftData[i].Re+fftData[i].Im*fftData[i].Im);
   psdx = (1.0/(8000.0*N));
   periodogramVc.push_back(10.0*log(psdx * pow(absolute,2)));
   if(i==N/2)
   {
      periodogramVc[i] = 10*log(2*periodogramVc[i]);
   }

   dftVc.push_back(absolute);
   //fftXaxis.push_back(i);
}

ui->fftvc->addGraph();
ui->fftvc->graph(0)->setData(fftXaxis,dftVc);
ui->fftvc->graph(0)->setPen(QPen(Qt::red,2));
ui->fftvc->yAxis->setLabel("FFT Vc |y(f)|");
ui->fftvc->xAxis->setRange(0,10);
```

```
ui->fftvc->xAxis->setLabel("Frecuencia (Hz)");
ui->fftvc->yAxis->rescale();
ui->fftvc->replot();

ui->SpecDenVc->addGraph();
ui->SpecDenVc->graph(0)->setData(fftXaxis,periodogramVc);
ui->SpecDenVc->graph(0)->setPen(QPen(Qt::red,2));
ui->SpecDenVc->yAxis->setLabel("Periodograma Vc");
ui->SpecDenVc->xAxis->setRange(0,10);
ui->SpecDenVc->xAxis->setLabel("Frecuencia (Hz)");
ui->SpecDenVc->yAxis->rescale();
ui->SpecDenVc->replot();

//////
for(int i=0;i<N;i++)
{
   fftData[i] = {Ia[i],0};
}
newfft.fft1d(fftData,N,1);

for(int i=0;i<N/2+1;i++)
{
   absolute = sqrt(fftData[i].Re*fftData[i].Re+fftData[i].Im*fftData[i].Im);
   psdx = (1.0/(8000.0*N));
   periodogramIa.push_back(10.0*log(psdx * pow(absolute,2)));
   if(i==N/2)
   {
      periodogramIa[i] = 10*log(2*periodogramIa[i]);
```

```
    }

    dftIA.push_back(absolute);
    //fftXaxis.push_back(i);
}

ui->fftia->addGraph();
ui->fftia->graph(0)->setData(fftXaxis,dftIA);
ui->fftia->graph(0)->setPen(QPen(Qt::red,2));
ui->fftia->yAxis->setLabel("FFT Ia |y(f)|");
ui->fftia->xAxis->setRange(0,10);
ui->fftia->xAxis->setLabel("Frecuencia (Hz)");
ui->fftia->yAxis->rescale();
ui->fftia->replot();

ui->SpecDenIa->addGraph();
ui->SpecDenIa->graph(0)->setData(fftXaxis,periodogramIa);
ui->SpecDenIa->graph(0)->setPen(QPen(Qt::red,2));
ui->SpecDenIa->yAxis->setLabel("Periodograma Ia");
ui->SpecDenIa->xAxis->setRange(0,10);
ui->SpecDenIa->xAxis->setLabel("Frecuencia (Hz)");
ui->SpecDenIa->yAxis->rescale();
ui->SpecDenIa->replot();

//////
for(int i=0;i<N;i++)
{
    fftData[i] = {Ib[i],0};
```

```
}
newfft.fft1d(fftData,N,1);

for(int i=0;i<N/2+1;i++)
{
   absolute = sqrt(fftData[i].Re*fftData[i].Re+fftData[i].Im*fftData[i].Im);
   psdx = (1.0/(8000.0*N));
   periodogramIb.push_back(10.0*log(psdx * pow(absolute,2)));
   if(i==N/2)
   {
      periodogramIb[i] = 10*log(2*periodogramIb[i]);
   }

   dftIb.push_back(absolute);
   //fftXaxis.push_back(i);
}

ui->fftib->addGraph();
ui->fftib->graph(0)->setData(fftXaxis,dftIb);
ui->fftib->graph(0)->setPen(QPen(Qt::red,2));
ui->fftib->yAxis->setLabel("FFT Ib |y(f)|");
ui->fftib->xAxis->setRange(0,10);
ui->fftib->xAxis->setLabel("Frecuencia (Hz)");
ui->fftib->yAxis->rescale();
ui->fftib->replot();

ui->SpecDenIb->addGraph();
ui->SpecDenIb->graph(0)->setData(fftXaxis,periodogramIb);
```

```
ui->SpecDenIb->graph(0)->setPen(QPen(Qt::red,2));
ui->SpecDenIb->yAxis->setLabel("Periodograma Ib");
ui->SpecDenIb->xAxis->setRange(0,10);
ui->SpecDenIb->xAxis->setLabel("Frecuencia (Hz)");
ui->SpecDenIb->yAxis->rescale();
ui->SpecDenIb->replot();

//////
for(int i=0;i<N;i++)
{
   fftData[i] = {Ic[i],0};
}
newfft.fft1d(fftData,N,1);

for(int i=0;i<N/2+1;i++)
{
   absolute = sqrt(fftData[i].Re*fftData[i].Re+fftData[i].Im*fftData[i].Im);
   psdx = (1.0/(8000.0*N));
   periodogramIc.push_back(10.0*log(psdx * pow(absolute,2)));
   if(i==N/2)
   {
      periodogramIc[i] = 10*log(2*periodogramIc[i]);
   }

   dftIc.push_back(absolute);
   //fftXaxis.push_back(i);
}
```

```
ui->fftic->addGraph();
ui->fftic->graph(0)->setData(fftXaxis,dftIc);
ui->fftic->graph(0)->setPen(QPen(Qt::red,2));
ui->fftic->yAxis->setLabel("FFT Ic |y(f)|");
ui->fftic->xAxis->setRange(0,10);
ui->fftic->xAxis->setLabel("Frecuencia (Hz)");
ui->fftic->yAxis->rescale();
ui->fftic->replot();

ui->SpecDenIc->addGraph();
ui->SpecDenIc->graph(0)->setData(fftXaxis,periodogramIc);
ui->SpecDenIc->graph(0)->setPen(QPen(Qt::red,2));
ui->SpecDenIc->yAxis->setLabel("Periodograma Ic");
ui->SpecDenIc->xAxis->setRange(0,10);
ui->SpecDenIc->xAxis->setLabel("Frecuencia (Hz)");
ui->SpecDenIc->yAxis->rescale();
ui->SpecDenIc->replot();

////////////////////////////////////////////////////////////////////////////////wavlet va begin
double waveData[N2];int xcount=0;
gsl_wavelet *w;
gsl_wavelet_workspace *work;

for(int i=0; i<N2;i++)
{
   waveData[i] = Va[i];
}
w = gsl_wavelet_alloc(gsl_wavelet_daubechies,4);
```

```
work = gsl_wavelet_workspace_alloc(N2);
gsl_wavelet_transform_forward(w,waveData,1,N2,work);
QVector<double> wva1,wxlv1;
for(int i=N2/2;i<N2-1;i++)
{
   wva1.push_back(waveData[i]);

   wxlv1.push_back(xcount);
   xcount++;
}
xcount =0;
ui->wva1->addGraph();
ui->wva1->addGraph();
ui->wva1->graph(0)->setData(wxlv1,wva1);
ui->wva1->graph(0)->setPen(QPen(Qt::blue,2));
ui->wva1->yAxis->setLabel("Wavelet lvl 1");
ui->wva1->xAxis->setRange(0,1000);
ui->wva1->yAxis->rescale();
ui->wva1->replot();

///////


QVector<double> wva2,wxlv2;
for(int i=N2/4;i<(N2/2)-1;i++)
{
   wva2.push_back(waveData[i]);
   wxlv2.push_back(xcount);
```

```
    xcount++;
}
xcount =0;
ui->wva2->addGraph();
ui->wva2->addGraph();
ui->wva2->graph(0)->setData(wxlv2,wva2);
ui->wva2->graph(0)->setPen(QPen(Qt::blue,2));
ui->wva2->yAxis->setLabel("Wavelet lvl 2");
ui->wva2->xAxis->setRange(0,1000);
ui->wva2->yAxis->rescale();
ui->wva2->replot();

///////

QVector<double> wva3,wxlv3;
for(int i=N2/8;i<(N2/4)-1;i++)
{
    wva3.push_back(waveData[i]);
    wxlv3.push_back(xcount);
    xcount++;
}
xcount =0;
ui->wva3->addGraph();
ui->wva3->addGraph();
ui->wva3->graph(0)->setData(wxlv3,wva3);
ui->wva3->graph(0)->setPen(QPen(Qt::blue,2));
ui->wva3->yAxis->setLabel("Wavelet lvl 3");
ui->wva3->xAxis->setRange(0,1000);
```

```
ui->wva3->yAxis->rescale();
ui->wva3->replot();

///////

QVector<double> wva4,wxlv4;
for(int i=N2/16;i<(N2/8)-1;i++)
{
   wva4.push_back(waveData[i]);
   wxlv4.push_back(xcount);
   xcount++;
}
xcount =0;
ui->wva4->addGraph();
ui->wva4->addGraph();
ui->wva4->graph(0)->setData(wxlv4,wva4);
ui->wva4->graph(0)->setPen(QPen(Qt::blue,2));
ui->wva4->yAxis->setLabel("Wavelet lvl 4");
ui->wva4->xAxis->setRange(0,1000);
ui->wva4->yAxis->rescale();
ui->wva4->replot();

///////

QVector<double> wva5,wxlv5;
for(int i=N2/32;i<(N2/6)-1;i++)
{
   wva5.push_back(waveData[i]);
```

```
    wxlv5.push_back(xcount);
    xcount++;
}
xcount =0;
ui->wva5->addGraph();
ui->wva5->addGraph();
ui->wva5->graph(0)->setData(wxlv5,wva5);
ui->wva5->graph(0)->setPen(QPen(Qt::blue,2));
ui->wva5->yAxis->setLabel("Wavelet lvl 5");
ui->wva5->xAxis->setRange(0,1000);
ui->wva5->yAxis->rescale();
ui->wva5->replot();

///////

QVector<double> wva6,wxlv6;
for(int i=N2/64;i<(N2/32)-1;i++)
{
    wva6.push_back(waveData[i]);
    wxlv6.push_back(xcount);
    xcount++;
}
xcount =0;
ui->wva6->addGraph();
ui->wva6->addGraph();
ui->wva6->graph(0)->setData(wxlv6,wva6);
ui->wva6->graph(0)->setPen(QPen(Qt::blue,2));
ui->wva6->yAxis->setLabel("Wavelet lvl 6");
```

```
ui->wva6->xAxis->setRange(0,1000);
ui->wva6->yAxis->rescale();
ui->wva6->replot();

//////////////////////wavlet va end


/////wavlet vb begin

for(int i=0; i<N2;i++)
{
   waveData[i] = Vb[i];
}
w = gsl_wavelet_alloc(gsl_wavelet_daubechies,4);
work = gsl_wavelet_workspace_alloc(N2);
gsl_wavelet_transform_forward(w,waveData,1,N2,work);
QVector<double> wvb1;
for(int i=N2/2;i<N2-1;i++)
{
   wvb1.push_back(waveData[i]);

}
ui->wvb1->addGraph();
ui->wvb1->addGraph();
ui->wvb1->graph(0)->setData(wxlv1,wvb1);
ui->wvb1->graph(0)->setPen(QPen(Qt::blue,2));
ui->wvb1->yAxis->setLabel("Wavelet lvl 1");
ui->wvb1->xAxis->setRange(0,1000);
```

```
ui->wvb1->yAxis->rescale();
ui->wvb1->replot();

///////


QVector<double> wvb2;
for(int i=N2/4;i<(N2/2)-1;i++)
{
   wvb2.push_back(waveData[i]);
}
ui->wvb2->addGraph();
ui->wvb2->addGraph();
ui->wvb2->graph(0)->setData(wxlv2,wvb2);
ui->wvb2->graph(0)->setPen(QPen(Qt::blue,2));
ui->wvb2->yAxis->setLabel("Wavelet lvl 2");
ui->wvb2->xAxis->setRange(0,1000);
ui->wvb2->yAxis->rescale();
ui->wvb2->replot();

///////

QVector<double> wvb3;
for(int i=N2/8;i<(N2/4)-1;i++)
{
   wvb3.push_back(waveData[i]);
}
ui->wvb3->addGraph();
```

```
ui->wvb3->addGraph();
ui->wvb3->graph(0)->setData(wxlv3,wvb3);
ui->wvb3->graph(0)->setPen(QPen(Qt::blue,2));
ui->wvb3->yAxis->setLabel("Wavelet lvl 3");
ui->wvb3->xAxis->setRange(0,1000);
ui->wvb3->yAxis->rescale();
ui->wvb3->replot();

///////

QVector<double> wvb4;
for(int i=N2/16;i<(N2/8)-1;i++)
{
   wvb4.push_back(waveData[i]);
}
ui->wvb4->addGraph();
ui->wvb4->addGraph();
ui->wvb4->graph(0)->setData(wxlv4,wvb4);
ui->wvb4->graph(0)->setPen(QPen(Qt::blue,2));
ui->wvb4->yAxis->setLabel("Wavelet lvl 4");
ui->wvb4->xAxis->setRange(0,1000);
ui->wvb4->yAxis->rescale();
ui->wvb4->replot();

///////

QVector<double> wvb5;
for(int i=N2/32;i<(N2/6)-1;i++)
```

```
{
   wvb5.push_back(waveData[i]);
}
ui->wvb5->addGraph();
ui->wvb5->addGraph();
ui->wvb5->graph(0)->setData(wxlv5,wvb5);
ui->wvb5->graph(0)->setPen(QPen(Qt::blue,2));
ui->wvb5->yAxis->setLabel("Wavelet lvl 5");
ui->wvb5->xAxis->setRange(0,1000);
ui->wvb5->yAxis->rescale();
ui->wvb5->replot();

///////

QVector<double> wvb6;
for(int i=N2/64;i<(N2/32)-1;i++)
{
   wvb6.push_back(waveData[i]);
}
ui->wvb6->addGraph();
ui->wvb6->addGraph();
ui->wvb6->graph(0)->setData(wxlv6,wvb6);
ui->wvb6->graph(0)->setPen(QPen(Qt::blue,2));
ui->wvb6->yAxis->setLabel("Wavelet lvl 6");
ui->wvb6->xAxis->setRange(0,1000);
ui->wvb6->yAxis->rescale();
ui->wvb6->replot();
//////////////////////wavlet vb end
```

```
///
/////wavlet vc begin

for(int i=0; i<N2;i++)
{
   waveData[i] = Vc[i];
}
w = gsl_wavelet_alloc(gsl_wavelet_daubechies,4);
work = gsl_wavelet_workspace_alloc(N2);
gsl_wavelet_transform_forward(w,waveData,1,N2,work);
QVector<double> wvc1;
for(int i=N2/2;i<N2-1;i++)
{
   wvc1.push_back(waveData[i]);

}
ui->wvc1->addGraph();
ui->wvc1->addGraph();
ui->wvc1->graph(0)->setData(wxlv1,wvc1);
ui->wvc1->graph(0)->setPen(QPen(Qt::blue,2));
ui->wvc1->yAxis->setLabel("Wavelet lvl 1");
ui->wvc1->xAxis->setRange(0,1000);
ui->wvc1->yAxis->rescale();
ui->wvc1->replot();

///////
```

```
QVector<double> wvc2;
for(int i=N2/4;i<(N2/2)-1;i++)
{
   wvc2.push_back(waveData[i]);
}
ui->wvc2->addGraph();
ui->wvc2->addGraph();
ui->wvc2->graph(0)->setData(wxlv2,wvc2);
ui->wvc2->graph(0)->setPen(QPen(Qt::blue,2));
ui->wvc2->yAxis->setLabel("Wavelet lvl 2");
ui->wvc2->xAxis->setRange(0,1000);
ui->wvc2->yAxis->rescale();
ui->wvc2->replot();

///////

QVector<double> wvc3;
for(int i=N2/8;i<(N2/4)-1;i++)
{
   wvc3.push_back(waveData[i]);
}
ui->wvc3->addGraph();
ui->wvc3->addGraph();
ui->wvc3->graph(0)->setData(wxlv3,wvc3);
ui->wvc3->graph(0)->setPen(QPen(Qt::blue,2));
ui->wvc3->yAxis->setLabel("Wavelet lvl 3");
ui->wvc3->xAxis->setRange(0,1000);
ui->wvc3->yAxis->rescale();
```

```
ui->wvc3->replot();

///////

QVector<double> wvc4;
for(int i=N2/16;i<(N2/8)-1;i++)
{
   wvc4.push_back(waveData[i]);
}
ui->wvc4->addGraph();
ui->wvc4->addGraph();
ui->wvc4->graph(0)->setData(wxlv4,wvc4);
ui->wvc4->graph(0)->setPen(QPen(Qt::blue,2));
ui->wvc4->yAxis->setLabel("Wavelet lvl 4");
ui->wvc4->xAxis->setRange(0,1000);
ui->wvc4->yAxis->rescale();
ui->wvc4->replot();

///////

QVector<double> wvc5;
for(int i=N2/32;i<(N2/6)-1;i++)
{
   wvc5.push_back(waveData[i]);
}
ui->wvc5->addGraph();
ui->wvc5->addGraph();
ui->wvc5->graph(0)->setData(wxlv5,wvc5);
```

```
ui->wvc5->graph(0)->setPen(QPen(Qt::blue,2));
ui->wvc5->yAxis->setLabel("Wavelet lvl 5");
ui->wvc5->xAxis->setRange(0,1000);
ui->wvc5->yAxis->rescale();
ui->wvc5->replot();

///////

QVector<double> wvc6;
for(int i=N2/64;i<(N2/32)-1;i++)
{
   wvc6.push_back(waveData[i]);
}
ui->wvc6->addGraph();
ui->wvc6->addGraph();
ui->wvc6->graph(0)->setData(wxlv6,wvc6);
ui->wvc6->graph(0)->setPen(QPen(Qt::blue,2));
ui->wvc6->yAxis->setLabel("Wavelet lvl 6");
ui->wvc6->xAxis->setRange(0,1000);
ui->wvc6->yAxis->rescale();
ui->wvc6->replot();
////////////////////////////////////////////////////////////////////////////////wavlet vc end

////////////////////////////////////////////////////////////////////////////////wavlet Ia begin


for(int i=0; i<N2;i++)
{
```

```
   waveData[i] = Ia[i];
}
w = gsl_wavelet_alloc(gsl_wavelet_daubechies,4);
work = gsl_wavelet_workspace_alloc(N2);
gsl_wavelet_transform_forward(w,waveData,1,N2,work);
QVector<double> wia1;
for(int i=N2/2;i<N2-1;i++)
{
   wia1.push_back(waveData[i]);
}

ui->wia1->addGraph();
ui->wia1->addGraph();
ui->wia1->graph(0)->setData(wxlv1,wia1);
ui->wia1->graph(0)->setPen(QPen(Qt::blue,2));
ui->wia1->yAxis->setLabel("Wavelet lvl 1");
ui->wia1->xAxis->setRange(0,1000);
ui->wia1->yAxis->rescale();
ui->wia1->replot();

///////


QVector<double> wia2;
for(int i=N2/4;i<(N2/2)-1;i++)
{
   wia2.push_back(waveData[i]);
```

```
}

ui->wia2->addGraph();
ui->wia2->addGraph();
ui->wia2->graph(0)->setData(wxlv2,wia2);
ui->wia2->graph(0)->setPen(QPen(Qt::blue,2));
ui->wia2->yAxis->setLabel("Wavelet lvl 2");
ui->wia2->xAxis->setRange(0,1000);
ui->wia2->yAxis->rescale();
ui->wia2->replot();

///////

QVector<double> wia3;
for(int i=N2/8;i<(N2/4)-1;i++)
{
   wia3.push_back(waveData[i]);
}
ui->wia3->addGraph();
ui->wia3->addGraph();
ui->wia3->graph(0)->setData(wxlv3,wia3);
ui->wia3->graph(0)->setPen(QPen(Qt::blue,2));
ui->wia3->yAxis->setLabel("Wavelet lvl 3");
ui->wia3->xAxis->setRange(0,1000);
ui->wia3->yAxis->rescale();
ui->wia3->replot();

///////
```

```
QVector<double> wia4;
for(int i=N2/16;i<(N2/8)-1;i++)
{
   wia4.push_back(waveData[i]);
}
ui->wia4->addGraph();
ui->wia4->addGraph();
ui->wia4->graph(0)->setData(wxlv4,wia4);
ui->wia4->graph(0)->setPen(QPen(Qt::blue,2));
ui->wia4->yAxis->setLabel("Wavelet lvl 4");
ui->wia4->xAxis->setRange(0,1000);
ui->wia4->yAxis->rescale();
ui->wia4->replot();

///////

QVector<double> wia5;
for(int i=N2/32;i<(N2/6)-1;i++)
{
   wia5.push_back(waveData[i]);
}
ui->wia5->addGraph();
ui->wia5->addGraph();
ui->wia5->graph(0)->setData(wxlv5,wia5);
ui->wia5->graph(0)->setPen(QPen(Qt::blue,2));
ui->wia5->yAxis->setLabel("Wavelet lvl 5");
ui->wia5->xAxis->setRange(0,1000);
```

```
ui->wia5->yAxis->rescale();
ui->wia5->replot();

///////

QVector<double> wia6;
for(int i=N2/64;i<(N2/32)-1;i++)
{
   wia6.push_back(waveData[i]);
}
ui->wia6->addGraph();
ui->wia6->addGraph();
ui->wia6->graph(0)->setData(wxlv6,wia6);
ui->wia6->graph(0)->setPen(QPen(Qt::blue,2));
ui->wia6->yAxis->setLabel("Wavelet lvl 6");
ui->wia6->xAxis->setRange(0,1000);
ui->wia6->yAxis->rescale();
ui->wia6->replot();

//////////////////////wavlet ia end


/////wavlet ib begin

for(int i=0; i<N2;i++)
{
   waveData[i] = Ib[i];
}
```

```
w = gsl_wavelet_alloc(gsl_wavelet_daubechies,4);
work = gsl_wavelet_workspace_alloc(N2);
gsl_wavelet_transform_forward(w,waveData,1,N2,work);
QVector<double> wib1;
for(int i=N2/2;i<N2-1;i++)
{
   wib1.push_back(waveData[i]);

}
ui->wib1->addGraph();
ui->wib1->addGraph();
ui->wib1->graph(0)->setData(wxlv1,wib1);
ui->wib1->graph(0)->setPen(QPen(Qt::blue,2));
ui->wib1->yAxis->setLabel("Wavelet lvl 1");
ui->wib1->xAxis->setRange(0,1000);
ui->wib1->yAxis->rescale();
ui->wib1->replot();

///////


QVector<double> wib2;
for(int i=N2/4;i<(N2/2)-1;i++)
{
   wib2.push_back(waveData[i]);
}
ui->wib2->addGraph();
ui->wib2->addGraph();
```

```
ui->wib2->graph(0)->setData(wxlv2,wib2);
ui->wib2->graph(0)->setPen(QPen(Qt::blue,2));
ui->wib2->yAxis->setLabel("Wavelet lvl 2");
ui->wib2->xAxis->setRange(0,1000);
ui->wib2->yAxis->rescale();
ui->wib2->replot();

///////

QVector<double> wib3;
for(int i=N2/8;i<(N2/4)-1;i++)
{
   wib3.push_back(waveData[i]);
}
ui->wib3->addGraph();
ui->wib3->addGraph();
ui->wib3->graph(0)->setData(wxlv3,wib3);
ui->wib3->graph(0)->setPen(QPen(Qt::blue,2));
ui->wib3->yAxis->setLabel("Wavelet lvl 3");
ui->wib3->xAxis->setRange(0,1000);
ui->wib3->yAxis->rescale();
ui->wib3->replot();

///////

QVector<double> wib4;
for(int i=N2/16;i<(N2/8)-1;i++)
{
```

```
   wib4.push_back(waveData[i]);
}
ui->wib4->addGraph();
ui->wib4->addGraph();
ui->wib4->graph(0)->setData(wxlv4,wib4);
ui->wib4->graph(0)->setPen(QPen(Qt::blue,2));
ui->wib4->yAxis->setLabel("Wavelet lvl 4");
ui->wib4->xAxis->setRange(0,1000);
ui->wib4->yAxis->rescale();
ui->wib4->replot();

///////

QVector<double> wib5;
for(int i=N2/32;i<(N2/6)-1;i++)
{
   wib5.push_back(waveData[i]);
}
ui->wib5->addGraph();
ui->wib5->addGraph();
ui->wib5->graph(0)->setData(wxlv5,wib5);
ui->wib5->graph(0)->setPen(QPen(Qt::blue,2));
ui->wib5->yAxis->setLabel("Wavelet lvl 5");
ui->wib5->xAxis->setRange(0,1000);
ui->wib5->yAxis->rescale();
ui->wib5->replot();

///////
```

```
QVector<double> wib6;
for(int i=N2/64;i<(N2/32)-1;i++)
{
   wib6.push_back(waveData[i]);
}
ui->wib6->addGraph();
ui->wib6->addGraph();
ui->wib6->graph(0)->setData(wxlv6,wib6);
ui->wib6->graph(0)->setPen(QPen(Qt::blue,2));
ui->wib6->yAxis->setLabel("Wavelet lvl 6");
ui->wib6->xAxis->setRange(0,1000);
ui->wib6->yAxis->rescale();
ui->wib6->replot();
//////////////////////wavlet vib end
///
/////wavlet ic begin

for(int i=0; i<N2;i++)
{
   waveData[i] = Ic[i];
}
w = gsl_wavelet_alloc(gsl_wavelet_daubechies,4);
work = gsl_wavelet_workspace_alloc(N2);
gsl_wavelet_transform_forward(w,waveData,1,N2,work);
QVector<double> wic1;
for(int i=N2/2;i<N2-1;i++)
{
```

```
   wic1.push_back(waveData[i]);

}
ui->wic1->addGraph();
ui->wic1->addGraph();
ui->wic1->graph(0)->setData(wxlv1,wic1);
ui->wic1->graph(0)->setPen(QPen(Qt::blue,2));
ui->wic1->yAxis->setLabel("Wavelet lvl 1");
ui->wic1->xAxis->setRange(0,1000);
ui->wic1->yAxis->rescale();
ui->wic1->replot();

///////


QVector<double> wic2;
for(int i=N2/4;i<(N2/2)-1;i++)
{
   wic2.push_back(waveData[i]);
}
ui->wic2->addGraph();
ui->wic2->addGraph();
ui->wic2->graph(0)->setData(wxlv2,wic2);
ui->wic2->graph(0)->setPen(QPen(Qt::blue,2));
ui->wic2->yAxis->setLabel("Wavelet lvl 2");
ui->wic2->xAxis->setRange(0,1000);
ui->wic2->yAxis->rescale();
ui->wic2->replot();
```

```
///////

QVector<double> wic3;
for(int i=N2/8;i<(N2/4)-1;i++)
{
   wic3.push_back(waveData[i]);
}
ui->wic3->addGraph();
ui->wic3->addGraph();
ui->wic3->graph(0)->setData(wxlv3,wic3);
ui->wic3->graph(0)->setPen(QPen(Qt::blue,2));
ui->wic3->yAxis->setLabel("Wavelet lvl 3");
ui->wic3->xAxis->setRange(0,1000);
ui->wic3->yAxis->rescale();
ui->wic3->replot();

///////

QVector<double> wic4;
for(int i=N2/16;i<(N2/8)-1;i++)
{
   wic4.push_back(waveData[i]);
}
ui->wic4->addGraph();
ui->wic4->addGraph();
ui->wic4->graph(0)->setData(wxlv4,wic4);
ui->wic4->graph(0)->setPen(QPen(Qt::blue,2));
```

```
ui->wic4->yAxis->setLabel("Wavelet lvl 4");
ui->wic4->xAxis->setRange(0,1000);
ui->wic4->yAxis->rescale();
ui->wic4->replot();

///////

QVector<double> wic5;
for(int i=N2/32;i<(N2/6)-1;i++)
{
   wic5.push_back(waveData[i]);
}
ui->wic5->addGraph();
ui->wic5->addGraph();
ui->wic5->graph(0)->setData(wxlv5,wic5);
ui->wic5->graph(0)->setPen(QPen(Qt::blue,2));
ui->wic5->yAxis->setLabel("Wavelet lvl 5");
ui->wic5->xAxis->setRange(0,1000);
ui->wic5->yAxis->rescale();
ui->wic5->replot();

///////

QVector<double> wic6;
for(int i=N2/64;i<(N2/32)-1;i++)
{
   wic6.push_back(waveData[i]);
}
```

```
ui->wic6->addGraph();
ui->wic6->addGraph();
ui->wic6->graph(0)->setData(wxlv6,wic6);
ui->wic6->graph(0)->setPen(QPen(Qt::blue,2));
ui->wic6->yAxis->setLabel("Wavelet lvl 6");
ui->wic6->xAxis->setRange(0,1000);
ui->wic6->yAxis->rescale();
ui->wic6->replot();
//////////////////////////////////////////////////////////////////////////wavlet Ic end

/////////
plot1->DrawCircle();
plot1->fasores(0,1,1);
plot1->fasores((M_PI/2-acos(PFa)),1,2);

plot1->fasores(M_PI*120/180,2,1);
plot1->fasores((M_PI*120/180)+(M_PI/2-acos(PFb)),2,2);

plot1->fasores(M_PI*240/180,3,1);
plot1->fasores((M_PI*240/180)+(M_PI/2-acos(PFa)),3,2);

ui->widget->addGraph();
ui->widget->graph(0)->setData(tiempo,Va);
ui->widget->graph(0)->setPen(QPen(Qt::darkGreen,2));
ui->widget->addGraph();
ui->widget->graph(1)->setData(tiempo,Vb);
ui->widget->graph(1)->setPen(QPen(Qt::red,2));
ui->widget->addGraph();
```

```
    ui->widget->graph(2)->setData(tiempo,Vc);
    ui->widget->graph(2)->setPen(QPen(Qt::blue,2));
    ui->widget->xAxis->setLabel("muestras");
    ui->widget->yAxis->setLabel("Volts");
    ui->widget->xAxis->setRange(0,1000);
    ui->widget->yAxis->rescale();


    ui->widget->setInteractions(QCP::iRangeDrag | QCP::iRangeZoom | QCP::iSelectAxes |
QCP::iSelectLegend | QCP::iSelectPlottables);
    ui->widget->axisRect()->setupFullAxesBox();
    ui->widget->plotLayout()->insertRow(0);
    ui->widget->plotLayout()->addElement(0,0,new QCPPlotTitle(ui->widget,"Voltages"));
    ui->widget->replot();


    ui->widget_2->addGraph();
    ui->widget_2->graph(0)->setData(tiempo,Ia);
    ui->widget_2->graph(0)->setPen(QPen(Qt::darkGreen,2));
    ui->widget_2->addGraph();
    ui->widget_2->graph(1)->setData(tiempo,Ib);
    ui->widget_2->graph(1)->setPen(QPen(Qt::red,2));
    ui->widget_2->addGraph();
    ui->widget_2->graph(2)->setData(tiempo,Ic);
    ui->widget_2->graph(2)->setPen(QPen(Qt::blue,2));
    ui->widget_2->xAxis->setLabel("muestras");
    ui->widget_2->yAxis->setLabel("Ampere");
    ui->widget_2->xAxis->setRange(0,1000);
    ui->widget_2->yAxis->rescale();
```

```
    ui->widget_2->setInteractions(QCP::iRangeDrag | QCP::iRangeZoom | QCP::iSelectAxes |
QCP::iSelectLegend | QCP::iSelectPlottables);
    ui->widget_2->axisRect()->setupFullAxesBox();
    ui->widget_2->plotLayout()->insertRow(0);
    ui->widget_2->plotLayout()->addElement(0,0,new QCPPlotTitle(ui->widget_2,"Currents"));
    ui->widget_2->replot();

    ui->Rmsva->addGraph();
    ui->Rmsva->graph(0)->setData(PQIndexXaxis,RmsVa);
    ui->Rmsva->graph(0)->setPen(QPen(Qt::red,2));
    ui->Rmsva->yAxis->setLabel("RMS Va");
    ui->Rmsva->xAxis->setRange(0,10);
    ui->Rmsva->yAxis->rescale();

    ui->Rmsvb->addGraph();
    ui->Rmsvb->graph(0)->setData(PQIndexXaxis,RmsVb);
    ui->Rmsvb->graph(0)->setPen(QPen(Qt::darkGreen,2));
    ui->Rmsvb->yAxis->setLabel("RMS Vb");
    ui->Rmsvb->xAxis->setRange(0,10);
    ui->Rmsvb->yAxis->rescale();

    ui->Rmsvc->addGraph();
    ui->Rmsvc->graph(0)->setData(PQIndexXaxis,RmsVc);
    ui->Rmsvc->graph(0)->setPen(QPen(Qt::blue,2));
    ui->Rmsvc->yAxis->setLabel("RMS Vc");
    ui->Rmsvc->xAxis->setRange(0,10);
    ui->Rmsvc->yAxis->rescale();
```

```
ui->Rmsia->addGraph();
ui->Rmsia->graph(0)->setData(PQIndexXaxis,RmsIa);
ui->Rmsia->graph(0)->setPen(QPen(Qt::red,2));
ui->Rmsia->yAxis->setLabel("RMS Ia");
ui->Rmsia->xAxis->setRange(0,10);
ui->Rmsia->yAxis->rescale();

ui->Rmsib->addGraph();
ui->Rmsib->graph(0)->setData(PQIndexXaxis,RmsIb);
ui->Rmsib->graph(0)->setPen(QPen(Qt::darkGreen,2));
ui->Rmsib->yAxis->setLabel("RMS Ib");
ui->Rmsib->xAxis->setRange(0,10);
ui->Rmsib->yAxis->rescale();

ui->Rmsic->addGraph();
ui->Rmsic->graph(0)->setData(PQIndexXaxis,RmsIc);
ui->Rmsic->graph(0)->setPen(QPen(Qt::blue,2));
ui->Rmsic->yAxis->setLabel("RMS Ic");
ui->Rmsic->xAxis->setRange(0,10);
ui->Rmsic->yAxis->rescale();

ui->PFa->addGraph();
ui->PFa->graph(0)->setData(PQIndexXaxis,PowerFa);
ui->PFa->graph(0)->setPen(QPen(Qt::red,2));
ui->PFa->yAxis->setLabel("Power Factor a");
ui->PFa->xAxis->setRange(0,10);
ui->PFa->yAxis->rescale();
```

```
    ui->PFb->addGraph();
    ui->PFb->graph(0)->setData(PQIndexXaxis,PowerFb);
    ui->PFb->graph(0)->setPen(QPen(Qt::darkGreen,2));
    ui->PFb->yAxis->setLabel("Power Factor b");
    ui->PFb->xAxis->setRange(0,10);
    ui->PFb->yAxis->rescale();

    ui->PFc->addGraph();
    ui->PFc->graph(0)->setData(PQIndexXaxis,PowerFc);
    ui->PFc->graph(0)->setPen(QPen(Qt::blue,2));
    ui->PFc->yAxis->setLabel("Power Factor c");
    ui->PFc->xAxis->setRange(0,10);
    ui->PFc->yAxis->rescale();
}

void PQ_Display::SliderSet()
{   int set=FileCount/FullFileDir.size();
    ui->horizontalSlider->setRange(1,set);
    QString StartTime = "Inicio de lectura = ";
    StartTime += FullFileDir[0][set].RealTIme;
    ui->label->setText(StartTime);

}

void PQ_Display::on_horizontalSlider_valueChanged(int value)
{
    QString StartTime = "Inicio de lectura = ";
```

```
    StartTime += FullFileDir[0][value].RealTIme;
    IterationName = FullFileDir[0][value].FilePath;
    IterationName.chop(6);
    //qDebug()<<IterationName;
    ui->label->setText(StartTime);



}

void PQ_Display::on_horizontalSlider_3_valueChanged(int value)
{
    ui->widget->xAxis->setRange(0,value*48000);
    ui->widget->replot();

    ui->widget_2->xAxis->setRange(0,value*48000);
    ui->widget_2->replot();
}

void PQ_Display::PlotSelected()
{
    Display_x_axis.clear();
    Display_y_axis.clear();
    int start[2]={0,0},edge[2]={1,1},stride[2]={1,1};
    //ui->horizontalSlider->value();
    for(int i=0;i<FullFileDir.size();i++)
    {
        qDebug()<<"Data of File "<<FullFileDir[i][0].Channel;
        ///data types
```

```
        /// gain = double
        /// offset = double
        /// data = int_16
        int time;
        mat_t *matfp;
        matvar_t *matvar;
        matfp = Mat_Open(FullFileDir[i][ui->horizontalSlider-
>value()].FilePath.toStdString().c_str(),MAT_ACC_RDONLY);
        matvar = Mat_VarReadInfo(matfp,"Gain");
        double *Gain = new double[matvar->dims[1]];
        edge[0]=matvar->dims[0];
        edge[1]=matvar->dims[1];
        Mat_VarReadData(matfp,matvar,Gain,start,stride,edge);
        qDebug()<<"Gain = "<<Gain[0];

        matvar = Mat_VarReadInfo(matfp,"Offset");
        double *Offset = new double[matvar->dims[1]];
        edge[0]=matvar->dims[0];
        edge[1]=matvar->dims[1];
        Mat_VarReadData(matfp,matvar,Offset,start,stride,edge);
        qDebug()<<"Offset = "<<Offset[0];

        //std::free(predestino);
        Mat_VarFree(matvar);
        Mat_Close(matfp);
    }
}
```

```
void PQ_Display::selectionChanged()
{

}

void PQ_Display::mousewheel()
{
   if(ui->widget->xAxis->selectedParts().testFlag(QCPAxis::spAxis))
      ui->widget->axisRect()->setRangeZoom(ui->widget->xAxis->orientation());
   else if (ui->widget->yAxis->selectedParts().testFlag(QCPAxis::spAxis))
      ui->widget->axisRect()->setRangeZoom(ui->widget->yAxis->orientation());
   else
      ui->widget->axisRect()->setRangeZoom(Qt::Horizontal|Qt::Vertical);



   if(ui->widget_2->xAxis->selectedParts().testFlag(QCPAxis::spAxis))
      ui->widget_2->axisRect()->setRangeZoom(ui->widget_2->xAxis->orientation());
   else if (ui->widget_2->yAxis->selectedParts().testFlag(QCPAxis::spAxis))
      ui->widget_2->axisRect()->setRangeZoom(ui->widget_2->yAxis->orientation());
   else
      ui->widget_2->axisRect()->setRangeZoom(Qt::Horizontal|Qt::Vertical);
}



void PQ_Display::on_horizontalSlider_2_valueChanged(int value)
{
   ui->Rmsva->xAxis->setRange(0,value*6);
   ui->Rmsva->replot();
```

```
    ui->Rmsvb->xAxis->setRange(0,value*6);
    ui->Rmsvb->replot();

    ui->Rmsvc->xAxis->setRange(0,value*6);
    ui->Rmsvc->replot();

    ui->Rmsia->xAxis->setRange(0,value*6);
    ui->Rmsia->replot();

    ui->Rmsib->xAxis->setRange(0,value*6);
    ui->Rmsib->replot();

    ui->Rmsic->xAxis->setRange(0,value*6);
    ui->Rmsic->replot();

    ui->PFa->xAxis->setRange(0,value*6);
    ui->PFa->replot();

    ui->PFb->xAxis->setRange(0,value*6);
    ui->PFb->replot();

    ui->PFc->xAxis->setRange(0,value*6);
    ui->PFc->replot();
}

void PQ_Display::on_horizontalSlider_4_valueChanged(int value)
{
```

```
    ui->fftva->xAxis->setRange(0,value*30);
    ui->fftva->replot();

    ui->fftvb->xAxis->setRange(0,value*30);
    ui->fftvb->replot();

    ui->fftvc->xAxis->setRange(0,value*30);
    ui->fftvc->replot();

    ui->fftia->xAxis->setRange(0,value*30);
    ui->fftia->replot();

    ui->fftib->xAxis->setRange(0,value*30);
    ui->fftib->replot();

    ui->fftic->xAxis->setRange(0,value*30);
    ui->fftic->replot();
}

void PQ_Display::on_horizontalSlider_5_valueChanged(int value)
{
    ui->SpecDenVa->xAxis->setRange(0,value*30);
    ui->SpecDenVa->replot();

    ui->SpecDenVb->xAxis->setRange(0,value*30);
    ui->SpecDenVb->replot();

    ui->SpecDenVc->xAxis->setRange(0,value*30);
```

```
    ui->SpecDenVc->replot();

    ui->SpecDenIa->xAxis->setRange(0,value*30);
    ui->SpecDenIa->replot();

    ui->SpecDenIb->xAxis->setRange(0,value*30);
    ui->SpecDenIb->replot();

    ui->SpecDenIc->xAxis->setRange(0,value*30);
    ui->SpecDenIc->replot();
}

void PQ_Display::on_horizontalSlider_6_valueChanged(int value)
{
    ui->wva1->xAxis->setRange(0,value*3000);
    ui->wva1->replot();

    ui->wva2->xAxis->setRange(0,value*1500);
    ui->wva2->replot();

    ui->wva3->xAxis->setRange(0,value*750);
    ui->wva3->replot();

    ui->wva4->xAxis->setRange(0,value*360);
    ui->wva4->replot();

    ui->wva5->xAxis->setRange(0,value*180);
    ui->wva5->replot();
```

```
    ui->wva6->xAxis->setRange(0,value*90);
    ui->wva6->replot();


}

void PQ_Display::on_horizontalSlider_7_valueChanged(int value)
{
    ui->wvb1->xAxis->setRange(0,value*3000);
    ui->wvb1->replot();

    ui->wvb2->xAxis->setRange(0,value*1500);
    ui->wvb2->replot();

    ui->wvb3->xAxis->setRange(0,value*750);
    ui->wvb3->replot();

    ui->wvb4->xAxis->setRange(0,value*360);
    ui->wvb4->replot();

    ui->wvb5->xAxis->setRange(0,value*180);
    ui->wvb5->replot();

    ui->wvb6->xAxis->setRange(0,value*90);
    ui->wvb6->replot();
}
```

```
void PQ_Display::on_horizontalSlider_8_valueChanged(int value)
{
   ui->wvc1->xAxis->setRange(0,value*3000);
   ui->wvc1->replot();

   ui->wvc2->xAxis->setRange(0,value*1500);
   ui->wvb2->replot();

   ui->wvc3->xAxis->setRange(0,value*750);
   ui->wvc3->replot();

   ui->wvc4->xAxis->setRange(0,value*360);
   ui->wvc4->replot();

   ui->wvc5->xAxis->setRange(0,value*180);
   ui->wvc5->replot();

   ui->wvc6->xAxis->setRange(0,value*90);
   ui->wvc6->replot();
}

void PQ_Display::on_horizontalSlider_9_valueChanged(int value)
{
   ui->wia1->xAxis->setRange(0,value*3000);
   ui->wia1->replot();

   ui->wia2->xAxis->setRange(0,value*1500);
   ui->wia2->replot();
```

```
    ui->wia3->xAxis->setRange(0,value*750);
    ui->wia3->replot();

    ui->wia4->xAxis->setRange(0,value*360);
    ui->wia4->replot();

    ui->wia5->xAxis->setRange(0,value*180);
    ui->wia5->replot();

    ui->wia6->xAxis->setRange(0,value*90);
    ui->wia6->replot();
}

void PQ_Display::on_horizontalSlider_10_valueChanged(int value)
{
    ui->wib1->xAxis->setRange(0,value*3000);
    ui->wib1->replot();

    ui->wib2->xAxis->setRange(0,value*1500);
    ui->wib2->replot();

    ui->wib3->xAxis->setRange(0,value*750);
    ui->wib3->replot();

    ui->wib4->xAxis->setRange(0,value*360);
    ui->wib4->replot();
```

```
    ui->wib5->xAxis->setRange(0,value*180);
    ui->wib5->replot();

    ui->wib6->xAxis->setRange(0,value*90);
    ui->wib6->replot();
}

void PQ_Display::on_horizontalSlider_11_valueChanged(int value)
{
    ui->wic1->xAxis->setRange(0,value*3000);
    ui->wic1->replot();

    ui->wic2->xAxis->setRange(0,value*1500);
    ui->wic2->replot();

    ui->wic3->xAxis->setRange(0,value*750);
    ui->wic3->replot();

    ui->wic4->xAxis->setRange(0,value*360);
    ui->wic4->replot();

    ui->wic5->xAxis->setRange(0,value*180);
    ui->wic5->replot();

    ui->wic6->xAxis->setRange(0,value*90);
    ui->wic6->replot();
}
```

```
#include "synthetic_display.h"
#include "ui_synthetic_display.h"

Synthetic_display::Synthetic_display(QWidget *parent) :
   QDialog(parent),
   ui(new Ui::Synthetic_display)
{
   ui->setupUi(this);
   QString file = QFileDialog::getOpenFileName(this,tr("/"),tr("Select file"),tr("Matlab files (*.mat)"));;
   LoadFile(file);
}

Synthetic_display::~Synthetic_display()
{
   delete ui;
}
void Synthetic_display::AddRoot(QString name)
{
   QTreeWidgetItem *itm = new QTreeWidgetItem(ui->treeWidget);
   itm->setText(0,name);
   ui->treeWidget->addTopLevelItem(itm);
}
void Synthetic_display::LoadFile(QString path)
{
   int start[2]={0,0},edge[2]={1,1},stride[2]={1,1};
   mat_t *matfp;
   matvar_t *matvar;
   matfp = Mat_Open(path.toStdString().c_str(),MAT_ACC_RDONLY);
```

```
matvar = Mat_VarReadInfo(matfp,"Data");
double *Data = new double[matvar->dims[1]];
edge[0]=matvar->dims[0];
edge[1]=matvar->dims[1];
Mat_VarReadData(matfp,matvar,Data,start,stride,edge);

matvar = Mat_VarReadInfo(matfp,"time");
double *time = new double[matvar->dims[1]];
edge[0]=matvar->dims[0];
edge[1]=matvar->dims[1];
Mat_VarReadData(matfp,matvar,time,start,stride,edge);



for(int i=0;i<matvar->dims[1];i++)
{
   Csignal.push_back(Data[i]);
   Ctime.push_back(time[i]);
}
std::free(Data);
std::free(time);

ui->widget->addGraph();
ui->widget->graph(0)->setData(Ctime,Csignal);
ui->widget->graph(0)->setPen(QPen(Qt::red,2));
ui->widget->yAxis->setLabel("Test signal f(y)");
ui->widget->xAxis->setRange(0,10);
```

```
ui->widget->xAxis->setLabel("time (sec)");
ui->widget->yAxis->rescale();
ui->widget->xAxis->rescale();
ui->widget->replot();


QVector<double> rms,dft,power;
QVector<double> rmsT,powerT;
double Srms=0;
double interval;
int RMSInterval = 1600;
for(int k=0;k<Csignal.size()/RMSInterval;k++)
{
   Srms=0;
for(int i=k*RMSInterval;i<((k*RMSInterval)+RMSInterval);i++)
{
   Srms = Srms+((pow(Csignal[i],2.0))/RMSInterval);
}


Srms = sqrt(Srms);


rms.push_back(Srms);

interval = k;
rmsT.push_back(interval/5);
}
```

```
double estimate = 141.4214;
double magnitudef1 = 0;
double TV;
for(int i=0;i<rms.size();i++)
{
   ///para fase 1
   magnitudef1 = rms[i]/estimate;
   qDebug()<<magnitudef1;
   TV = i;
   TV = TV/5;

   if(magnitudef1 == 0)
   {
      AddRoot("Interrupcion encontrana en t = "+ QString::number(TV));
   }
   if(magnitudef1 > 1.1)
   {
      AddRoot("Swell detectado en t = "+ QString::number(TV) );
   }
   if(magnitudef1 <  .9 && magnitudef1 > 0)
   {
      AddRoot("Sag detectado en t = "+ QString::number(TV));
   }

}

ui->rms->addGraph();
ui->rms->graph(0)->setData(rmsT,rms);
```

```
ui->rms->graph(0)->setPen(QPen(Qt::red,2));
ui->rms->yAxis->setLabel("Rms");
ui->rms->xAxis->setRange(0,10);
ui->rms->xAxis->setLabel("time (sec)");
ui->rms->yAxis->rescale();
ui->rms->xAxis->rescale();
ui->rms->replot();

int  N=8192;
int  N2=N;

complex *fftData;
fftData = new complex[N];

for(int i=0;i<N;i++)
{
   fftData[i] = {Csignal[i],0};
}

fft newfft;
newfft.fft1d(fftData,N,1);

double absolute;
double psdx;
double frqX=0;
for(int i=0;i<N/2+1;i++)
{
   absolute = sqrt(fftData[i].Re*fftData[i].Re+fftData[i].Im*fftData[i].Im);
```

```
    psdx = (1.0/(8000.0*N));
    power.push_back(10.0*log(psdx * pow(absolute,2)));
    if(i==N/2)
    {
        power[i] = 10*log(2*power[i]);
    }

    dft.push_back(absolute);
    frqX = (2*3.1416)/(N*.000125);
    powerT.push_back(i*(frqX)/(2*3.1416));
}

ui->FFT->addGraph();
ui->FFT->graph(0)->setData(powerT,dft);
ui->FFT->graph(0)->setPen(QPen(Qt::red,2));
ui->FFT->yAxis->setLabel("FFT Va |y(f)|");
ui->FFT->xAxis->setRange(0,10);
ui->FFT->xAxis->setLabel("Frecuencia (Hz)");
ui->FFT->yAxis->rescale();
ui->FFT->xAxis->rescale();
ui->FFT->replot();

ui->Periodogram->addGraph();
ui->Periodogram->graph(0)->setData(powerT,power);
ui->Periodogram->graph(0)->setPen(QPen(Qt::red,2));
ui->Periodogram->yAxis->setLabel("Periodograma Va");
ui->Periodogram->xAxis->setRange(0,10);
ui->Periodogram->xAxis->setLabel("Frecuencia (Hz)");
```

```
    ui->Periodogram->yAxis->rescale();
    ui->Periodogram->xAxis->rescale();
    ui->Periodogram->replot();

}
```

6.2 ARTÍCULO

Storage management and visualization tool for massive power quality files in SD devices

W. A. Dzul Ayala, L. Morales-Velázquez, R. A. Osornino-Rios, R. J. Romero-Troncoso, O. Duque-Pérez

Resumen: Actualmente la necesidad creciente de almacenamiento de grandes cantidades de información de manera eficiente y rápida se ha vuelto más común en los ámbitos industrial y casero. El análisis de la calidad de la energía utiliza tales atributos de almacenamiento para capturar grandes cantidades de señales eléctricas para procesar, con el objetivo de detectar perturbaciones que pueden ser dañinas para los elementos en una red eléctrica. En este trabajo, se presenta una metodología para escribir datos utilizando todo el espacio disponible en dispositivos de almacenamiento masivo. También el desarrollo de una herramienta capaz de decodificar la información para generar archivos de grandes cantidades de información, y propone una manera de fragmentar el flujo continuo de información para un fácil manejo y almacenamiento; subsecuentemente, generando índices para acceder y desplegar información con facilidad y bajos recursos de procesamiento computacional.

***Palabras Clave*: mediciones eléctricas, calidad de la energía, almacenamiento de información, sistemas embebidos, visualización de datos, Estructuras de datos, Adquisición de datos, Conversión de datos, Manejo de datos.**

Abstract: In present times, an increasing necessity for storage of large quantities of information efficiently and fast has become ever more common in its use over industrial and home environments. Power quality analyzing utilizes such storage attributes to capture large amounts of electric signals to process in order to detect disturbances that may be harmful for elements in a power grid. In this paper, a methodology for writing data utilizing all available space of a massive storage unit is presented. Also, the development of a tool capable of decoding information to generate large amounts of information files, and proposes a way to fragment continuous information streams for easy handling and storage; subsequently, generating indexes to access and display information with ease and low PC resources consumption.

***Keywords*: electric measurement, power quality, information storage, embedded systems, Data visualization, Data structures, Data acquisition, Data conversion, Data handling.**

Walter Alexander Dzul Ayala. wdzul@hspdigital.org
Luis Morales Velázquez. lmorales@hspdigital.org
Roque A. Osornio Rios raor@uaq.mx
Dr. René de Jesús Romero Troncoso. troncoso@hspdigital.org
Facultad de Ingeniería, Universidad Autónoma de Querétaro, 76807.
Dr. Oscar Duque Pérez. oscar.duque@eii.uva.es
Escuela de ingenierías industriales. Universidad de Valladolid, España.

Este trabajo fue parcialmente financiado por CONACyT, beca 631615/334810 y por los proyectos SEP-CONACyT 222453-201, PROMEP 103.5/14/710401 y FOFIUAQ-FIN201613.

I. INTRODUCCIÓN

La necesidad del monitoreo de redes eléctricas ha incrementado en las recientes décadas como resultado del incremento del uso de elementos de alta precisión en la industria. La funcionalidad y vida útil de estos elementos se ve afectada por perturbaciones eléctricas, causando fallas de lectura y funcionamiento significando pérdidas económicas. Por lo cual varias compañías e instituciones académicas en todo el mundo desarrollan nuevas técnicas para detectar, clasificar y diagnosticar estas perturbaciones llamadas eventos de calidad de la energía. Proyectos de investigación internacionales desarrollan nuevas técnicas de procesamiento de señales para la clasificación y diagnóstico de eventos de calidad de la energía, por ejemplo, utilizando redes neuronales [1], métodos de compensación [2], el efecto de la calidad de la energía para energías renovables [3], entre otros. Localmente existe un incremento en el número de proyectos de investigación que estudian la calidad de la energía y sus efectos en redes eléctricas. Los eventos de calidad de la energía afectan equipos diseñados para tareas precisas [4]. También el estudio del efecto de eventos de calidad de la energía en el torque de equipos de maquinado [5]. La investigación de calidad de la energía conlleva desarrollar técnicas de detección y clasificación y diagnóstico de sus eventos [6-7].

Como resultado de la investigación y desarrollo de calidad de la energía a nivel mundial, se utilizan varias guías y regulaciones como estándar. Algunos de los más usados son la IEEE 1159 [8], y la EN 50160 [9].

Regulaciones internacionales estandarizadas permitieron a varias empresas introducir monitores de calidad de la energía propietarios al mercado, estos dispositivos detectan eventos de calidad de la energía y crean bitácoras de dichos eventos. Con la inconveniencia de un alto precio. Por ejemplo el monitor Fluke 1760TR tiene un precio de alrededor de 15300 dólares [10].

A diferencia de los equipos disponibles en el mercado, los sistemas embebidos tienen rangos de precio y tamaño mucho menores, esto con el inconveniente de una capacidad de procesamiento limitada, por lo que estos dispositivos deben ser diseñados con métodos de máxima optimización, lo que los hacen difíciles de programar e interactuar con ellos.

El dispositivo de monitoreo de calidad de la energía propietario (PQ-UAQ) se compone de un microprocesador basado en FPGA y tiene como objetivo capturar señales eléctricas completas y almacenarlas en una tarjeta SD a una alta frecuencia de manera continua. Este trabajo describe consideraciones para la optimización de calidad de la energía y

uso de memoria que lleva al desarrollo de una herramienta diseñada para recuperar grandes cantidades de información, indexado, así como lectura para desplegar la información de manera gráfica, con facilidad para el usuario.

II. METODOLOGÍA

Esta sección describe limitaciones por consideraciones y regulaciones así como arquitecturas diseñadas para la aplicación como se muestra en la **Figura 1**.

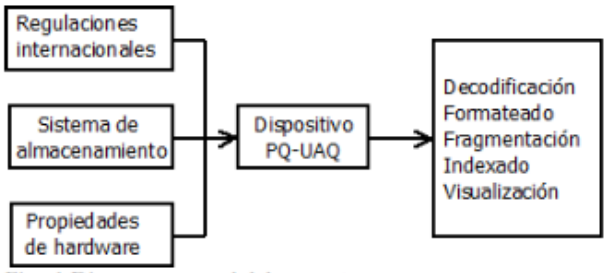


Fig. 1 Diagrama general del proyecto.

A. *Calidad de la Energía*

Detectar perturbaciones de energía en una red eléctrica requiere evaluar corrientes y voltajes, para posteriormente comparar con regulaciones estandarizadas.
De acuerdo con el estándar IEEE 1159 las señales eléctricas se muestrean a 128, 256 y 512 muestras por ciclo.
De acuerdo con el estándar EN 1159 medición y prueba de la calidad de fuentes de voltaje requiere intervalos de medición de 10 minutos.
Se puede inferir que la información capturada por el dispositivo PQ-UAQ necesita una frecuencia de captura superior a 6.4 ksps y esta información debe ser separada en intervalos de 10 minutos.

B. *Sistemas embebidos*

Más del 98 por ciento de los procesadores se vuelven sistemas embebidos. Desde juguetes hasta controladores de plantas nucleares, estos procesadores ayudan a manejar fábricas y permiten un flujo de información, productos y personas [11].
El dispositivo PQ-UAQ es un microprocesador basado en FPGA. Este sistema embebido ejecuta tareas para adquisición de datos, escritura de datos en una tarjeta SD y seguimiento de tiempo real.

El microprocesador tiene 512 KB de capacidad, de los cuales 256 KB se dedican para almacenamiento temporal de datos a través FIFOs de 32 KB, y los 256 KB restantes manejan el convertidor análogo digital, el protocolo de comunicación con la tarjeta SD, reloj, y operaciones lógicas, lo que existe una limitante entre el almacenamiento temporal y escritura de datos en la tarjeta SD.

C. *Sistemas de archivos*

Los sistemas de archivos se utilizan para controlar como los datos se almacenan. Sin un sistema de archivos, la información almacenada sería un bloque de datos sin indicadores que separan un archivo de otro. Al separar los datos en partes con nombres propios, la información es fácilmente identificable.

El sistema FAT (file allocation system) fue diseñado para archivos de menos de 1 MB. Este sistema consiste de tablas de asignación localizadas en varios sectores [12]. El formato FAT32 permite el manejo de archivos de hasta 2GB, lo cual es una mejora pero el dispositivo PQ-UAQ necesita no tener restricción de tamaño de archivos.

El sistema NTFS (New Technology File System) es similar al FAT, diseñado para archivos más grandes. Toda la información se almacena como archivos en la unidad y se accede a estos a través de una tabla de direcciones maestra [13]. Aunque la limitación de tamaño no represente un problema al usar este sistema de archivos, la velocidad con la que los datos serian escritos en la unidad se ve afectada y conlleva a pérdida de datos y variaciones de frecuencia de muestreo.

D. *Captura de datos*

La velocidad de captura está limitada por el CAD integrado en el dispositivo PQ-UAQ, el cual es 8000 ksps, como el mínimo requerido es 6400 ksps de acuerdo con el estándar, los datos adquiridos son válidos para el cálculo de índices de calidad de la energía y procesamiento de señales.

El dispositivo PQ-UAAQ adquiere 8 canales de datos a 8000 ksps a través de un convertidor análogo digital (CAD) para evitar la sobrecarga de información en el microprocesador, tomando en cuenta que la target a SD se compone de bloques de 2 KB. El almacenamiento temporal se ajustó a 8 KB de información por canal.

El sistema de archivos por el cual se almacenarían los datos en la tarjeta SD necesitaba ser rápido y no tener limitación de tamaño de archivos. Por lo cual la información almacenada en la tarjeta SD era escrita sin formato. El almacenamiento de datos sin formato resulta incompatible a la mayoría de los programas y es inútil sin una rutina de decodificación de datos. Por estas razones se desarrolló un sistema de archivos propio descrito en la **Figura 2.**

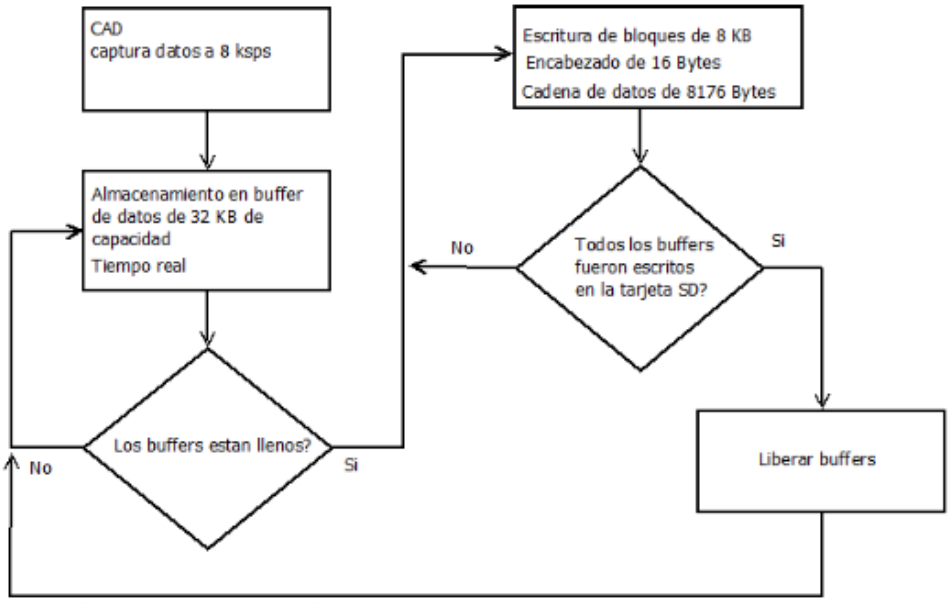


Fig. 2. Descripción del método de escritura de datos

Como esta descrito en la **Figura 2**, el dispositivo PQ-UAQ lee y escribe un flujo continuo de datos en cada uno de sus 8 canales disponibles. En cada canal se escriben 8176 Bytes de información y un encabezado de 16 Bytes, el cual permite la decodificación de la información.

E. Estructura de datos antes del formateo

El sistema de archivos diseñado para esta aplicación consiste de bloques de 8 KB de información escritos en serie en una tarjeta SD, al comienzo de cada bloque se determina un encabezado de 16 Bytes para su decodificación.

La **Tabla 1** muestra los 11 valores determinados necesarios para la codificación de los datos y fragmentación.

El primer valor funciona como una bandera que confirma la locación del encabezado y su estructura. El segundo valor determina el nombre del dispositivo que está realizando la lectura. La señal determina el tipo de datos capturados, ya sean voltajes o corrientes. El canal determina cuál de los 8 canales está realizando la lectura. La secuencia determina permite guardar registro de la continuidad de las lecturas, día, hora, minutos, segundos y milisegundos representan el tiempo real de la captura de datos. La frecuencia de muestreo es la cantidad de datos capturados en un segundo por la tarjeta. 8000 ksps para mantener consistencia.

Los archivos generados después de la decodificación la información en la tarjeta SD son formateados y comprimidos para facilitar su manejo, así como determinar una nueva estructura para los archivos.

Objeto	Estructura	tamaño (Bytes)
Bandera	Hexadecimal	1
ID	Número	1
Señal	Número	1
Canal	Número	1
Secuencia	Número	4
Días	Número	1
Horas	Valor BCD	1
Minutos	Valor BCD	1
Segundos	Valor BCD	1
Milisegundos	Número	2
Frecuencia de muestreo	Número	2

Tabla 1. Estructura de datos en tarjeta SD

F. Sincronización

Si un proyecto requiere más de un dispositivo para realizar lectura simultáneamente. Todos los dispositivos deben estar sincronizados para que el usuario pueda realizar una comparación de las lecturas obtenidas por varios dispositivos en una res eléctrica.

El dispositivo PQ-UAQ puede ser sincronizado vía bluetooth. Esto quiere decir que los dispositivos se sincronizan con el reloj del dispositivo que envía el comando.

G. Fragmentación de datos

Es necesario fragmentar e indexar la información adquirida de la tarjeta SD, para su futuro almacenamiento.

Las pruebas de captura de datos tuvieron una duración de 1 semana. Por lo tanto la información total capturada por cada

canal es de 9.01 GB. Manejar archivos de este tamaño resulta en un excesivo consumo de recursos computacionales. Para aligerar la carga de almacenamiento y procesamiento se implementó un método de fragmentación.

Un algoritmo de fragmentación aplicado durante el método de lectura de la tarjeta SD puede generar archivos de cualquier tamaño deseado. Un intervalo de tiempo a considerar es el mínimo requerido para realizar un análisis de datos válido de calidad de la energía. Que de acuerdo con estándares internacionales [8-9], es un intervalo de 10 minutos.

A lo largo de una semana de lecturas. El total de archivos de 10 minutos excede de 8000. Por lo que es necesario implementar un método de indexado para mantener continuidad y orden en las lecturas.

Como se describe en la **Figura 3**, las lecturas son fragmentadas en días, cada tiene las horas de datos capturados ese día, cada hora tiene una carpeta por cada 10 minutos capturados en esa hora. En cada carpeta de 10 minutos se encuentran 8 archivos, uno por cada canal.

H. Formateo de datos y compresión

Los archivos generados serian aún incompatibles con cualquier aplicación. Para facilitar la administración y procesamiento de datos. Todos los archivos fueron convertidos a formato .mat para ser compatibles con la aplicación MATLAB, entre otras aplicaciones. La estructura de los archivos generados se representa en la **Tabla 2**. Para optimizar el tamaño de cada archivo se utilizó una librería libre fui utilizada [14].

I. Estructura de archivos fragmentados

Objeto	Estructura
Color	String
Comentario	String
Datos	Vector
Dispositivo	String
Ganancia	Número
Offset	Número
Tiempo Real	String
Número de muestras	Número
Frecuencia de muestreo	Número
Señal	String
Unidad	Char

Tabla 2. Estructura de datos después de fragmentación

El *color* indica el color en el que los valores serían visualizados. El *comentario* indica consideraciones para la interpretación de datos. Los *datos* son el vector de información capturada. El *dispositivo* representa que tarjeta realizó la lectura. La *ganancia* y *offset* son valores de calibración de la lectura. *Tiempo real* representa la hora de la lectura. *Numero de muestras* indica cuantos datos hay en el vector de lectura para determinar si hay perdida de datos. La *señal* y la *unidad* determinan que tipo de dato fue capturado.

J. Visualización de datos

La interfaz fue desarrollada en Qt, una herramienta de desarrollo de interfaz basada en c/c++ [15]. Para analizar y visualizar los datos generados por dicha interfaz, es necesario tomar en cuenta la cantidad de memoria necesaria para la tarea, un vector de lectura entero representa vario GB, por lo que cargar y analizar estos archivos es imposible para computadoras de media y baja gama. Por lo que una función de muestreo fue desarrollada para visualizar los datos con bajos recursos computacionales.

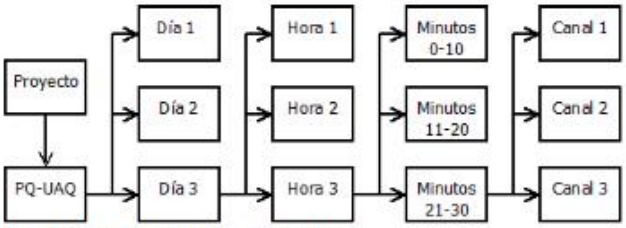


Fig. 3. Diagrama de indexado de datos

III. RESULTADOS

A. Configuración experimental

Fig. 4. Instalación del equipo

El experimento se llevó a cabo en el hospital Rio Hortegay consistió en conectar los dispositivos PQ-UAQ a una red trifásica para medir y capturar corrientes y voltajes.

Las pinzas fueron i400 de Fluke [16]. Estas pinzas miden corrientes de CA con una salida de 1 mA/A, están diseñadas para capturar hasta 400 A. los voltajes fueron conectados a cada fase de la red.

Como se observa en la **Figura 4**, el dispositivo PQ-UAQ está localizado en el número 1, en el número 2 los voltajes y en el 3 las corrientes a medir.

B. Visualización de datos

Toda la visualización de datos fue efectuada en una pantalla de resolución 1920x1080 y los vectores de datos fueron de 1 MB de tamaño. Todo el procesamiento y representación visual generaría un consumo de memoria de 30 MB por instancia.
La Figura 5 es una captura de pantalla que muestra vectores de información visualizados. En un intervalo de tiempo muy pequeño en un momento arbitrario de la lectura. Se pueden observar los voltajes en la gráfica superior y las corrientes en la gráfica inferior.
La figura 6. Muestra los fasores de los valores mostrados en la imagen anterior, cada color representa una fase. Las flechas largas representan las fases de los voltajes y las flechas cortas representan las fases de las corrientes.

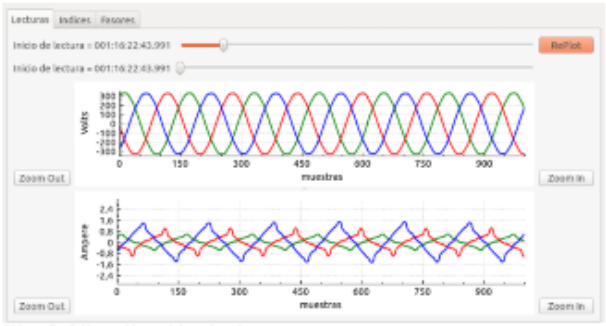


Fig. 5. Visualización de datos

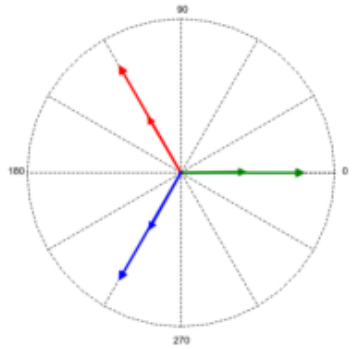


Fig. 6. fasores

C. Resultados de almacenamiento

Una vez la captura de los datos termina. La recuperación de dichos datos puede realizarse. El total de lecturas de una semana se recupera en las siguientes velocidades.

La Tabla 3 describe el tiempo de que toma la recuperación de datos de la tarjeta SD, el proceso de fragmentación, formateo, compresión e indexado.

La compresión de datos es efectuada durante la recuperación de datos y afecta el tamaño de los archivos en base a las variaciones de los datos.

La información sin comprimir recabada durante una captura de datos es invariante. Un archivo de 10 minutos es de 9.15 MB. Mientras que un archivo comprimido varía entre 3 MB para corrientes y 8MB para voltajes dependiendo de su información. Tomando el promedio de estos datos se obtiene un 40 por ciento ahorro de espacio en el disco.

Especificaciones	PC 1	PC 2
Procesador	2nd. Gen. I5	4rth. Gen. I5
RAM	8 Gb DDR3	16 Gb DDR3
Disco duro	5400 RPM	7200 RPM
Tiempo en realizar la tarea.	2:10 Horas	1:50 Horas

Tabla 3. Especificaciones de máquinas usadas.

IV. CONCLUSIONES

Se puede concluir que el tiempo que transcurre durante la lectura de datos de la tarjeta SD varía conforme a las especificaciones de las computadoras que realizaron las tareas. El proceso de lectura, decodificación, formato, fragmentación, compresión e indexado de capturas de datos de 1 semana de duración tomaron alrededor de 2 horas, generando alrededor de 50 GB de archivos. Por dispositivo. En comparación con los datos son comprimir, el espacio requerido es de 70 GB, lo que representa una considerable optimización de espacio.
Los tamaños de los archivos son fáciles de manejar y transferir, y con la estructura de indexado es fácil también encontrar archivos específicos. Además el indexado de archivos facilita a la herramienta de visualización realizar el muestreo para las gráficas utilizando bajos recursos.
Con el indexado y fragmentación de datos, una gran cantidad de archivos que almacenan una gran cantidad de información puede ser visualizada y procesada con máquinas de baja y media gama.

Referencias

[1] M. Seera, C. P. Lim, C. K. Loo, and H. Singh, "A modified fuzzy min–max neural network for data clustering and its application to power quality monitoring," *Appl. Soft Comput.*, vol. 28, pp. 19–29, Mar. 2015.

[2] E. V. Liberado, F. P. Marafão, M. G. Simões, W. A. De Souza, and J. A. Pomilio, "Novel expert system for defining power quality compensators," *Expert Syst. Appl.*, vol. 42, no. 7, pp. 3562–3570, May 2015.

[3] M. Farhoodnea, A. Mohamed, H. Shareef, and H. Zayandehroodi, "Power Quality Impact of Renewable Energy based Generators and Electric Vehicles on Distribution Systems," *Procedia Technol.*, vol. 11, no. Iceei, pp. 11–17, 2013.

[4] D. G. L. (Uaq), "Análisis en maquinaria CNC ante variaciones de bajo voltaje y sus efectos de calidad de la energía," 2013.

[5] D. Granados-Lieberman, R. a. Osornio-Rios, J. R. Rivera-Guillen, M. Trejo-Hernandez, and R. J. Romero-Troncoso, "Torque reduction and workpiece finishing effects due to voltage sags in turning processes," *Proc. Inst. Mech. Eng. Part B J. Eng. Manuf.*, vol. 228, no. 1, pp. 140–148, 2013.

[6] D. Granados-Lieberman, R. J. Romero-Troncoso, R. a. Osornio-Rios, a. Garcia-Perez, and E. Cabal-Yepez, "Techniques and methodologies for power quality analysis and disturbances classification in power systems: a review," *IET Gener. Transm. Distrib.*, vol. 5, no. July 2010, p. 519, 2011.

[7] Z. Liu, Q. Zhang, Z. Han, and G. Chen, "A new classification method for transient power quality combining spectral kurtosis with neural network," *Neurocomputing*, vol. 125, pp. 95–101, Feb. 2014.

[8] Institute of Electrical and Electronics Engineers, *IEEE Std 1159 - IEEE Recommended Practice for Monitoring Electric Power Quality.*, vol. 2009, no. June. 2009.

[9] H. (Cooper D. A. Markiewicz and A. (Wroclaw U. of T. Klajn, "Voltage Disturbances," *Power Qual. Appl. Guid.*, vol. 5.4.2, pp. 4–11, 2004.

[10] F.-F. et al. S. D. Card, "Fluke Test Tools Price List," pp. 1–16, 2016.

[11] B. Group, "Embedded systems glossary," pp. 657–663, 2007.

[12] M. Extensible and F. Initiative, "Hardware White Paper Microsoft Extensible Firmware Initiative FAT32 File System Specification," 2000.

[13] R. Russon and Y. Fledel, "NTFS Documentation," 2004.

[14] CHRISTOPHER C. HULBERT, "MATIO library." 2013.

[15] J. Blanchette and M. Summerfield, *C++ GUI Programming with Qt 4, Second Edition*, Second. Prentice Hall, 2008.

[16] FLUKE, "AC i400 current clamp," vol. 2004, no. December, pp. 1–2, 2004.

Ing. Walter Alexander Dzul Ayala
Recibió el grado de ingeniero en mecatrónica en 2013 en la universidad Modelo, y actualmente se encuentra estudiando para obtener el título de maestría.

Dr. Roque A. Osornio Ríos.
Recibió el grado de Ingeniero por parte del Instituto Tecnológico de Querétaro, Querétaro, México. Y los grados de Maestro en ingeniería y Doctor por parte de la Universidad de Querétaro, Querétaro, México, en 2007. Es un investigador nacional CONACYT.

Dr. Luis Morales Velázquez.
Ingeniero en Electrónica egresado de la Universidad de Guanajuato. Obtuvo el grado de Maestría en Instrumentación y Control además de un Doctorado en la Universidad Autónoma de Querétaro. Es reconocido por CONACYT como investigador.

Dr. René de Jesús Romero Troncoso.
Recibió el grado de Ingeniero y de Maestro por parte de la Universidad de Guanajuato, Salamanca, México. Y el grado de Doctor por honores por parte de la Universidad Autónoma de Querétaro, Querétaro, México, en 2004. Es un investigador nacional nivel 2 CONACYT. En la actualidad es profesor titular en la Universidad de Guanajuato y es un investigador invitado de la Universidad Autónoma de Querétaro. Ha sido asesor de más de 180 tesis, autor de dos libros en sistemas digitales y coautor de más de 80 artículos en revistas internacionales y conferencias. Sus áreas de interés son el procesamiento de señales en hardware y mecatrónica

Dr. Oscar Duque Pérez

Realizó su tesis doctoral en el año 2000, en la universidad de Valladolid, actualmente trabaja como investigador en la Universidad de Valladolid, España