Universidad Autónoma de Querétaro
Facultad de Ingeniería
Maestría en Ciencias (Mecatrónica)

Aplicación para dispositivos móviles con sistema operativo Android enfocado al monitoreo eléctrico en campo

Opción de titulación
Tesis

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

Presenta:
Pablo César Ramírez Echeverría

Dirigido por:
Dr. René de J. Romero Troncoso

Dr. René de J. Romero Troncoso Presidente	Firma
Dr. Luis Morales Velázquez Secretario	Firma
Dr. Roque Alfredo Osornio Ríos Vocal	Firma
Dr. Daniel Moriñigo Sotelo Suplente	Firma
Dr. J. Jesús de Santiago Pérez Suplente	Firma

Dr. Aurelio Domínguez González
Director de la Facultad

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

Centro Universitario
Querétaro, Qro.
Noviembre del 2016

RESUMEN

El monitoreo de calidad de la energía se ha convertido en un caso de estudio de gran importancia en redes de sistemas eléctricos, debido a los altos costos que resulta tener una instalación eléctrica ineficiente. En este trabajo se desarrolló una aplicación para dispositivos móviles, la cual es capaz de realizar el monitoreo de un sistema de adquisición de datos para el análisis de la calidad energética de la red eléctrica, que realice el pre-procesamiento e identificación de perturbaciones. Este trabajo se encuentra en un campo que es un nuevo horizonte para la investigación relacionada a la calidad energética y puede traer ventajas como: fácil utilización, adaptabilidad y poco costo. Esta aplicación intenta satisfacer la necesidad de tener una herramienta de monitoreo que pueda trabajar de manera rápida y económica, así como también al mismo tiempo poseer la capacidad de graficar en tiempo real con por lo menos 5 metros de rango y realizar una sincronización efectiva dentro de las normas con varios índices básicos de la calidad energética. La interfaz gráfica de esta aplicación fue desarrollada por medio de reglas de diseño heurísticas, por las cuales se puede obtener una interfaz gráfica de usuario sencilla y de fácil interpretación. La aplicación desarrollada para este trabajo fue programada en lenguaje Android por medio de la herramienta Android Estudio y haciendo uso de las librerías más básicas del sistema Android, aumentando el rango de compatibilidad de dispositivos móviles en los cuales se puede usar la aplicación. De igual manera, se realizaron varias pruebas de diseño para comprobar la eficiencia de la aplicación.

(**Palabras clave:** Android, GUI, Heuristica, Monitoreo, Mobil, Calidad de la Energia)

SUMMARY

Monitoring power quality has become a case study of great importance in networks of electrical systems, due to the cost of having an inefficient electrical installation. In this paper, a mobile application for monitoring data and visualization of data applied to power quality systems data loggers to perform identification of power quality disturbances is developed. This application tries to meet the need for a monitoring tool that can work quickly and economically while possessing the ability to plot in real-time with at least five meters range and make an effective synchronization within the norms, with several basic indexes of energy quality. The graphical interface of this application was developed by heuristic rules of design by which you can get a simple graphical user interface and easy interpretation. The application in this work was developed in Android language by Android Studio tool and using the most basic libraries of the Android system, increasing the range of compatible mobile devices which can use the application. Several design tests were performed to test the efficiency of the application.

(**Key words:** Android, GUI, heuristics, monitoring, mobile, Power Quality)

AGRADECIMIENTOS

Mis principales agradecimientos son, a mis padres por otorgarme su apoyo moral a lo largo de este camino y a los profesores que se han preocupado por darme las herramientas necesarias para superar los retos que se puedan presentar en mi vida más adelante en especial a mis asesores que ha tenido mucha paciencia para guiarme en este trabajo. Al Consejo Nacional de Ciencia y Tecnología (CONACYT) por otorgarme una beca y de esta manera poder realizar los estudios de posgrado para este trabajo. A la Universidad Autónoma de Querétaro por aceptarme en uno de sus programas de posgrado de calidad y a la Universidad de Valladolid por apoyarme durante la estancia en España.

TABLA DE CONTENIDO

Índice de figuras

Índice de Tablas

I. INTRODUCCIÓN

En el mundo del ámbito tecnológico, se ha producido la necesidad de crear nuevos y mejores procesos para todas las aplicaciones posibles, siempre buscando la comodidad, velocidad, potencia y flexibilidad.

Entre los diferentes ámbitos en los cuales existe innovación tecnológica, se encuentra la calidad energética, este es uno de los temas de más interés crítico no solo para el sector industrial, pero para muchas instituciones en diferentes sectores, esto se debe a las desventajas generadas por las descomposturas o fallas que puede generar una red que presente fenómenos relacionados a la calidad de la energía. Por tal razón, un gran número de investigaciones se producen para desarrollar software y hardware con el propósito de realizar diferentes tareas y encontrar fallas en la red eléctrica, muchos de estos desarrollos tecnológicos cuentas con sistemas de monitoreo, ya sea directamente incorporado a los dispositivos o conectados de manera inalámbrica; sin embargo, actualmente las interfaces de dichos trabajos se realizan generalmente con sistemas aparatosos y poco prácticos para un sistema de monitoreo, provocando que se mueva de montaje eléctrico periódicamente. Una alternativa a este problema, es contar con una plataforma móvil con la cual se pueda realizar un monitoreo sencillo de la calidad energética.

Una de las plataformas que cuentan con las características necesarias para realizar el monitoreo son los teléfonos inteligentes Android, con los cuales se pueden monitorear diferentes tipos de dispositivos de adquisición de datos para diversas aplicaciones en el campo de la calidad energética, para facilitar el desarrollo es preferible contar con una arquitectura abierta y con fácil accesibilidad.

En esta investigación se desarrolló una aplicación móvil para el análisis rápido, eficiente y flexible de la red eléctrica con el fin de detectar y documentar eventos relacionados a la calidad energética.

La tesis aborda en primer lugar los antecedentes relacionados al tema, posteriormente se plantea la problemática de la investigación, la justificación del trabajo de investigación, la hipótesis y los objetivos general y particulares, la revisión de literatura que incluye el estado del arte y la fundamentación teórica, seguida de la metodología de la investigación, los resultados y por último las conclusiones del trabajo.

1.1 PLANTEAMIENTO GENERAL

En la actualidad muchas instituciones resultan afectados por el fenómeno de la calidad energética, con el propósito de identificar este fenómeno de Power Quality (PQ) en la red eléctrica, se han desarrollados varios dispositivos de análisis y adquisición de datos con diferentes tecnologías, tal es el caso del PQ-UAQ, un dispositivo generado por la Universidad Autónoma de Querétaro, el propósito general de este aparato es la adquisición de datos en la red eléctrica, por medio de este instrumento se pueden realizar varias investigaciones científicas, sin embargo, este sistema está especializado a la función de adquisición de datos.

El dispositivo descrito, por el momento no cuente con una manera sencilla de visualizar las mediciones tomadas para verificar el estado de la conexión y por lo tanto el monitoreo del sistema puede resultar complicado. En la actualidad se requiere que el dispositivo evolucione a hacia un sistema de gran movilidad y practicidad.

Una de las tecnologías emergentes es la de los teléfonos inteligentes, estos dispositivos tienen un gran número de implementaciones en otros campos científicos; sin embargo, son relativamente nuevos para el análisis de calidad de la energía y los existentes tienen exclusividad con sistemas comerciales. Es importante indicar que mediante una aplicación para un dispositivo móvil se puede generar una interfaz gráfica, con la cual se permita realizar el monitoreo de la red eléctrica y el procesamiento de las señales de manera práctica y de fácil movilidad.

1.2 ANTECEDENTES

Un monitoreo de un dispositivo de adquisición necesita de una forma de visualizar los datos y de realizar pequeños cambios de ser necesario. El uso de una interfaz gráfica permite tener mejor eficiencia, reducir costos, tiempos de operación e intercomunicación. En este trabajo se implementará una Interfaz Gráfica de Usuario (GUI - Graphical User Interface) para un dispositivo de adquisición de datos para investigación en calidad de la energía, el cual estará implementada por medio de un Arreglo de Compuertas Programables de Campo (FPGA - Field Programmable Gate Array) a un sistema de adquisición de datos para un análisis de PQ, este es un término que asocia la calidad de la señal del suministro de voltaje y corriente consumida por la carga. Una mala calidad en las redes eléctricas ocasiona innumerables problemas a los equipos eléctricos, lo cual representa en la industria altos costos y tiempos perdidos.

El primer paso para resolver problemas de detección de ineficiencia energética en la línea eléctrica, es instalando permanentemente un sensor que

permitan medir variables para poder observar fenómenos en la línea eléctrica. Posteriormente se necesita generar un sistema de monitoreo energético, en el cual se tengan todas las señales obtenidas de los sensores de cada sistema eléctrico.

En la Universidad Autónoma de Querétaro se han realizado varias investigaciones relacionadas a la calidad de energía, Valtierra-Rodriguez *et al* (2013) desarrollaron una metodología de estimación de harmónicos en redes trifásicas por medio de redes neuronales, la cual se implementa por medio de FPGA, esto permite una comunicación continua de la red trifásica y la detección de inter-armónicos, con el fin de tener un sistema más eficiente de monitoreo eléctrico a razón de la comunicación en paralelo, la rapidez de procesamiento y reconfiguración que se posee al ser implementado por medio de un FPGA.

Cabal-Yepez *et al* (2013) generaron un sistema de monitoreo para aplicaciones industriales que permite ejecutar un análisis de frecuencia en tiempo real en los sistemas de la red eléctrica, se usó un sistema FPGA-SoC (System on Chip, Sistema en un Chip) para la modificación o la redefinición de los algoritmos implementados con el propósito de realizar el análisis requerido para el sistema indicado, esto con la meta de ejecutar diferentes tipos de análisis de calidad de la energía en la red eléctrica sin tener que cambiar el dispositivo.

Granados-Lieberman *et al* (2013) crearon un sensor inteligente para estimar la frecuencia de un sistema eléctrico en tiempo real. Este sensor cumple con los estándares para la detección de los fenómenos que afectan a la calidad de energía en sistemas eléctricos como picos de voltaje, raíz media cuadrada, factor de cresta y distorsión armónica, obteniendo una estimación rápida, precisa, con alta resolución y pequeñas divergencias, este sensor usa una pinza para medir corriente y monitorea de manera eficiente las perturbaciones eléctricas.

Romero-Troncoso *et al.* (2011) propusieron una metodología, la cual se puede implementar en hardware, además de unir ciencias de la información con lógica difusa con el fin de identificar fallas en la maquinaria y otros dispositivos eléctricos que forman parte de la red eléctrica.

Valtierra-Rodriguez *et al* (2013) generaron una metodología novedosa por medio de redes neuronales para detectar y clasificar anomalías que puedan afectar a la calidad energética, gracias a la clasificación de patrones mediante los histogramas vertical y horizontal de un voltaje en específico, lo cual se consiguió a través del uso de redes neuronales para su posterior clasificación.

En el aspecto regional se tienen varios trabajaos en los cuales se investigan y desarrollan distintos aspectos de la calidad energética. Moreno *et al* (2014) propusieron un algoritmo con la capacidad de diagnosticar variaciones y perturbaciones en el voltaje, este algoritmo se realizó por medio de operaciones aritméticas con el fin de tener un método simplificado, en el cual se eviten las transformaciones tiempo-frecuencia y sea capaz de detectar y aislar los transitorios de otras perturbaciones.

Cisneros-Magaña *et al* (2014) presentaron una metodología basada en un filtro Kalman y derivación numérica para tener una mayor eficiencia en la estimación energética de redes eléctricas.

Los trabajos de carácter internacional con respecto a eficiencia energética son variados e interdisciplinarios, García-Hernández *et al*. (2013) propusieron un sistema de monitoreo de eficiencia energética para evitar defectos o mejorar la resolución en sistemas nucleares espectroscópicos.

Shareef *et al* (2013) generaron un método novedoso para visualizar diferentes tipos de defectos en red eléctrica, el método propuesto hace uso del concepto de visión por computadora utilizando escala de grises e imágenes binarias. Las imágenes en escala de grises revelan la información con respecto a qué tipo de perturbación se presenta.

Dehghani *et al* (2013) hicieron uso de un sistema de monitoreo de señales para identificar perturbaciones eléctricas por medio del Modelo Oculto de Markov (HMM - Hidden Markov Model), para identificar los vectores correspondientes de la señal trifásica, este sistema es entrenado por medio de un análisis estadístico.

Junfeng *et al* (2012) propusieron un sistema de análisis de eficiencia energética basado en una computadora embebida industrial, basándose en las capacidades del sistema eléctrico, se combinó Windows con aritmética para mejorar la detección de anomalías en la red eléctrica y se desarrolló una interfaz del sistema.

Otro concepto importante a considerar en los sistemas de monitoreo, es la conciencia situacional. La obtención de la información para tener una conciencia situacional es de gran importancia y por lo tanto contar con una visión general de la situación, esta permite al usuario poseer un mejor juicio de la situación y ser más eficiente al realizar el trabajo o el entendimiento de la causa del efecto de un fenómeno. En la literatura se encuentran varios documentos e investigaciones que recalcan la importancia de la conciencia situacional.

Panteli *et al* (2015) explicaron cómo los sistemas al forman parte de una red energética, es una operación extremadamente complicada debido a la complejidad y al gran número de contingencias que pueden ocurrir. Por lo tanto los operadores deberían de tener la suficiente información para entender la situación

del sistema y ser capaces de realizar decisiones con el propósito de comprender el comportamiento del sistema a futuro.

Siguiendo a Panteli *et al* presentan la importancia de implementar los sistemas de conciencia situacional y como es que tienen un papel clave en preservar la integridad de los sistemas energéticos modernos. Cuando se necesita tratar la eficiencia del sistema y se requiere desarrollar herramientas que puede eficientemente apoyar el proceso de toma de decisiones en una red energética la conciencia situacional se constituye en una herramienta fundamental.

Salmon *et al* (2006) realizaron un análisis sobre el uso de la conciencia situacional en sistema de c4i (Command, Control, Communication, Computers and Intelligence, Comando, Control, Comunicación, Computadores e Inteligencia) y dan a conocer eventos en los cuales la falta de un buen sistema de conciencia situacional afecta la toma de decisiones y como esto ha generado varios accidentes que afectan a la eficiencia de los sistemas c4i debido a la falta de comunicación y una interfaz adecuada.

Hauss (2003) propuso un cambio radical en los sistemas de administración de tráfico aéreo, en su trabajo discute la eficiencia del SAGAT como método para determinar la conciencia situacional y propone un nuevo método llamado SALSA, el cual toma en cuenta que los controladores aéreos generan una representación mental del tráfico aéreo, además de realizar pruebas las cuales arrojaron resultados donde se demuestra que el método SALSA es más eficiente y la conciencia situacional toma un papel muy relevante. Los sistemas basados en conciencia situacional, son de gran importancia debido a que se tiene como objetivo que el sistema sea fácil de usar y sea intuitivo, sin embargo, en la mayoría de los trabajos

de la literatura, no se tiene desarrollado una interfaz para el monitoreo de un sistema de identificación de señales en la red eléctrica considerando la conciencia situacional, lo cual hace poco flexible y difícil de usar a operarios nuevos.

En cuanto al diseño de sistemas se tiene como objetivo cumplir las reglas de diseño heurísticas, hay varios artículos que hablan de métodos heurísticos. Sandanayake *et al* (2015) desarrollaron un algoritmo basado en las reglas de diseño heurísticas para determinar el diseño de una bancada en una mina de manera más eficiente y rápida, incorporando variaciones de la bancada de manera que se resuelva el complejo problema de optimización en el proceso de la obtención de la bancada.

Graditi *et al* (2014) propusieron una manera de manejar cargas cambiantes mediante el uso de un algoritmo, lo cual reduce la necesidad de optimización de variables y la adopción de una nueva forma de obtención de métodos donde se usan reglas de diseño heurística, la eficiencia del algoritmos es comparada con otros algoritmos en los sistemas eléctricos propuestos.

Chen *et al (*2015) determinaron un nuevo sistema el de Árbol de Máximo Balanceo de Carga (MCLCT - Maximum Vonnected Load Balancing Cover Tree) en el cual se usaron algoritmos heurísticos para la creación de un algoritmo que pudiera mantener el control de una red de sensores.

García (2014) presentó un trabajo en el que se muestra una plataforma de monitoreo inercial con el protocolo Modbus, desarrollado como un sistema embebido basado en FPGA y se apoya en una herramienta informática para el control del sistema. Como se puede observar el desarrollo de sistemas basados en

las reglas de diseño heurística pueden resultar en una mejora en la eficiencia de la generación del diseño de sistemas o algoritmos.

Todo sistema hardware requiere de un desarrollo software, por lo cual el desarrollo de GUI es muy importante, la importancia de las interfaces gráficas es debido a la necesidad de tener un entorno virtual, intuitivo y sencillo para permitir la comunicación del hardware con el usuario, debido a que muchos sistemas actuales no cuentan con una GUI apropiada, se tiene como resultado la necesidad de gastar tiempo y dinero en entrenar a un operador especializado.

En la literatura se observan varios temas relacionados al de desarrollo de interfaces gráficas en el que se recalca la importancia del uso de una GUI apropiada.

Gilliland *et al* (2014) desarrollaron una interfaz gráfica con Qt para visualizar información de señales ultrasónicas, las cuales se obtienen por medio de un SoC y un FPGA para aumentar la eficiencia de procesamiento en el cual se usó OpenCL para acelerar la eficiencia de la computadora.

Khan(2014) generó una interfaz gráfica capaz de monitorear múltiples sensores y controlar diversos sistemas eléctricos. Este sistema tiene la flexibilidad de ser usado en la industria de producción de químicos, medicina y empacadoras.

Callejas *et al* (2013) diseñaron un sistema de control por medio de redes neuronales con aprendizaje no supervisado tipo Mapas Auto-Organizados de Kohonen (SOM - Self-Organizing Maps), usando un FPGA Spartan 6 para la determinación del comportamiento de un robot móvil en la resolución de trayectorias de tipo laberinto. Para ello fue necesario realizar una etapa de exploración en la cual

el robot recopila la información del entorno, se desarrolló una interfaz gráfica para controlar y visualizar los procesos realizados por el robot.

Valentina *et al* (2013) crearon, implementaron e integraron un sistema HMI (Human Machine Interface, Interfaz Máquina-Usuario) para un sensor de temperatura industrial, la interfaz gráfica fue desarrollada en Qt por medio del cual se desarrolló una interfaz ágil y útil para el monitoreo del sensor de temperatura.

Los sistemas móviles presentan una opción para una interfaz gráfica, la cual puede generar una mejor conciencia situacional para la institución en la que se encuentran, esto es debido a los reciente avances de Traer Tu Propio Dispositivo (BYOD - Bring Your Own Device), lo cual consiste en usar los dispositivos personales para el trabajo de manera que el usuario se capaz de obtener datos en un aparato en el que ya tiene cierta capacidad y experiencia, al igual que confianza en el mismo.

También se pueden consultar varios artículos científicos que prueban la importancia de las aplicaciones móviles en el sector privado y como al introducir una red de dispositivos puede resultar en un incremento de la eficiencia en la productividad.

Funk (2006) discutió sobre el futuro de los sistemas móviles basados en aplicaciones intranet, usando información del mercado japonés y modelos para la evolución de las industrias, estos modelos sugieren que habría un cambio drástico en estos dispositivos a lo largo que sistemas más predominantes y más eficientes que cuentan con el aspecto móvil irían emergiendo. La emergencia de un diseño o diseños dominantes para aplicaciones intranet acelera la difusión de las

aplicaciones móviles y en cierta medida determinar al ganador de la competencia entre varias firmas.

Sheng *et al* (2005) demostraron que la tecnología móvil puede ser usada como una herramienta estratégica en una organización y examinaron el impacto de la tecnología móvil en una organización, además de realizar un estudio cualitativo de la producción que dio como resultado una percepción en una implicación estratégica usando tecnología móvil para apoyar a las ventas y marketing en la organización donde se realizó la toma de datos. Al igual, se definieron tres maneras en que la tecnología móvil puede ser positiva desde un punto estratégico para la empresa, estos son: mejoras en el proceso de trabajo, aumento en la comunicación y distribución de información y mejorar la eficiencia de ventas y de marketing en la empresa.

Funk (2009) presentó la evolución del papel de los instrumentos móviles en el sector privado, la investigación muestra: como acuerdos entre los diferentes estándares de interface han logrado generar una conexión entre los aparatos móviles y otras industrias, los productos resultantes y otros servicios mayormente reflejan la capacidad tecnológica de los aparatos móviles en otras industrias, cada nueva interfaz requiere una cantidad critica de usuarios, una cantidad critica de usuarios generalmente se genera a partir de otra cantidad critica de usuarios. Esta investigación sugiere que el mercado occidental no ha cumplido con su potencial y que es necesario un cambio en las políticas del uso de dispositivos móviles.

Miller *et al* (2014) discutieron que el monitoreo por medio de dispositivos móviles presenta diferentes retos como son: la limitada potencia de la batería y recursos de procesamiento de información, además de oportunidades únicas, en

particular la habilidad de capturar información incorporando el contexto de las actividades monitoreadas. Por otra parte, se propone una solución configurable integrada para dispositivos móviles, el sistema es eficiente en la implementación de varios requisitos de monitoreo y permite al usuario supervisar simple información o recibir notificaciones de eventos que ocurren durante el monitoreo.

En la literatura no hay artículos en los que se use un dispositivo Android para la obtención de información en el monitoreo de un sistema para la identificación de fallas en el sistema eléctrico, sin embargo, hay varios artículos en los que se usa un dispositivos Android para la obtención de información en otras aplicaciones.

Widhiasi *et a*l (2013) generaron una aplicación para la detección y comunicación remota con pacientes y proveer una buena retroalimentación, no solo con un sistemas de monitoreo que permite a pacientes con ischemia ser remotamente monitoreados pero también tener contacto con la base de datos de HeartPALS (Heart Pediatric Life Support, Soporte Vital Pediátrico del Corazón).

Evans *et al* (2014) informaron sobre el desarrollo de una aplicación móvil, la cual está basada en dispositivos Android y plataformas iOS. Dicha aplicación tiene como propósito generar la espectroscopia por medio de vibraciones, la cual se puede usar para un gran rango de materiales, este análisis contiene información sobre la composición química de la estructura analizada, estos datos pueden ser documentados rápidamente por un dispositivo móvil. Esto es debido al alto crecimiento en vida de batería y rapidez en el procesamiento de la información.

Zhao *et al* (2014) describieron una aplicación desarrollada para dispositivos móviles en la plataforma Android la cual es capaz de recibir información en tiempo real de notificaciones de eventos astrofísicos. Gracias a los mensajes en VOEevent

Format que es un formato para trasmitir información obtenida en la recopilación de eventos astrofísicos. Sin embargo, la obtención de éstos ya se tenía por medios de Servicio de Mensajes Cortos (SMS - Short Message Service), pero el uso de una aplicación móvil basada en Android es una novedad y hace más fácil la obtención de información por medio del dispositivo.

Khan *et al* (2015) presentaron una aplicación para mantener el monitoreo de una capsula inalámbrica endoscópica la cual está desarrollada en un FPGA de bajo consumo energético de MachXO2-2000, la cual tiene una comunicación directa por medio de la red inalámbrica con un dispositivo móvil Android, el sistema tiene varias funcionalidades y la aplicación fue desarrollada para obtener el video en tiempo real, se realizaron pruebas en organismos vivos para la validación del sistema.

Sin embargo, a pesar de que muchas de las investigaciones cuentan con varias de las características tomadas en esta investigación, no cuentan con un sistema de monitoreo en el que se tenga una evidente rapidez de procesamiento, comunicación en paralelo y adaptabilidad, lo cual es posible lograr con un FPGA y una interfaz gráfica de clasificación y diagnóstico de la calidad de energía que tenga una eficiente calibración y las ventajas de tener un sistema móvil basado en Android, por el cual se puedan hacer un análisis rápido y más eficiente.

1.3 PROBLEMA GENERAL

En la Actualidad existe un gran número de usuarios que emplean una buena cantidad de dispositivos sensibles a anomalías en las redes eléctricas. Las estadísticas a nivel mundial en relación a los problemas de la calidad de la energía se incrementan cada año, estos fenómenos son uno de los principales causantes de fallas en herramientas y equipo industrial, dispositivos de cómputo y dispositivos comunes como es la iluminación, escaleras eléctricas, elevadores, etc.

Por otra parte, existen diferentes fallas en equipos eléctricos dentro de áreas no críticas en instituciones que pueden parecer poco importantes, sin embargo, este tipo de perturbaciones en la línea pueden interrumpir otros procesos más complicados.

Ante la problemática descrita se han desarrollado varios dispositivos que se encargan de monitorear la red eléctrica con el propósito de realizar un análisis para identificar las fuentes de los eventos de PQ que se presenten en una instalación. Cuando se diseñan los dispositivos se necesita de una forma de visualizar los datos adquiridos en el tiempo de la instalación para tener revisiones periódicas de los datos adquiridos y el estado del dispositivo, por esta razón, se han desarrollado varias interfaces gráficas en ordenadores, sistemas dedicados incorporados en la tarjeta y sistemas modulares, estos últimos son un tipo de diseño que está tomando popularidad para diferentes aplicaciones institucionales.

El diseño planteado generalmente dividen el trabajo de la interfaz gráfica a un ordenador o una computadora de escritorio o portátil, sin embargo, últimamente el uso de dispositivos móviles está resultando una mejor opción debido a la

practicidad de este tipo de aplicaciones, sin embargo muchas de las aplicaciones actuales no son compatibles con el hardware desarrollado por empresas o instituciones externas.

El desarrollo para aplicaciones de dispositivos móviles es muy reciente y la cantidad de investigación científica sobre su efectividad es muy escasa. De acuerdo a los trabajos reportados, en la actualidad no existen aplicaciones para el monitoreo de las fallas en equipo eléctrico para áreas no criticas de hospitales o industrias. Los trabajos existentes se enfocan a realizar una correlación o tener un sistema de identificación aplicado en un computador, pero no hay trabajos en los cuales se cuente con una aplicación móvil para realizar el análisis y monitoreo de la red eléctrica en tiempo real.

1.4 JUSTIFICACION

Este trabajo surge debido a la necesidad de una interfaz gráfica para realizar el monitoreo de sistemas desarrollados para investigación en calidad energética y hasta la fecha no se han reportado en la literatura metodologías en las cuales se realice el monitoreo de disturbios eléctricos de dispositivos conectados a una red eléctrica, el cual sea de fácil acceso e implementación por medio de la aplicación en una plataforma Android con el cual se puede contar con un diseño modular el cual podría traer consigo varias ventajas para los usuarios.

El desarrollo de esta aplicación beneficiará al sector productivo, ya que puede proveer la capacidad de tener una mejor conciencia situacional y por lo tanto reducir los costos de mantenimiento en equipos industriales, médicos y comerciales. Es por estas razones, que se justifica la realización de esta aplicación para un sistema de monitoreo propietario llamado PQ-UAQ de ésta institución, con el cual se permitirá tener un monitoreo de la línea constantemente, fácil de entender, de prevenir graves descomposturas a los dispositivos eléctricos y ser de bajo costo. La investigación también busca impulsar el uso de aplicaciones móviles en el sector industrial y científico, debido a la carencia que existe en la actualidad. Por otra parte, las ventajas de implementar un dispositivo móvil para el monitoreo de la línea puede traer consigo la contribución en innovación de las aplicaciones móviles en maquinaria instalada en redes eléctricas. Así también, la lectura se realizará en un sistema con mayor inmunidad al ruido que pueda existir en la línea de alimentación eléctrica y se fomentará el trabajo con instituciones educativas foráneas, debido a los acuerdos que tiene la Universidad Autónoma de Querétaro con la UVA (Universidad de Valladolid) y el Hospital Universitario Río Hortega en Valladolid, España.

1.5 HIPÓTESIS Y OBJETIVOS

1.5.1 HIPÓTESIS

Mediante una metodología de diseño basadas en aplicaciones móviles libre será posible cumplir con los requerimientos de un monitoreo en línea de algunos parámetros básicos como valores pico-pico, RMS y media de la calidad de energía en base a las normas necesarias.

1.5.2 OBJETIVO GENERAL

El objetivo principal de este trabajo es desarrollar una aplicación basada en el sistema operativo Android para dispositivos móviles, con la finalidad de realizar el monitoreo eléctrico en campo de la red eléctrica utilizando el equipo PQ-UAQ.

1.5.3 OBJETIVOS ESPECÍFICOS

Con el propósito de lograr la consecución del objetivo general se han establecido los siguientes objetivos particulares:

- Establecer los requerimientos de la aplicación a desarrollar basado en las características del equipo y las necesidades del usuario, de manera que se pueda obtener una aplicación intuitiva.

- Programar la aplicación de software basada en las herramientas heurísticas para el diseño de la interfaz de monitoreo eléctrico.

- Desarrollar los módulos de comunicación inalámbrica, visualización, procesamiento en línea y sincronización basados en Android.

- Realizar las pruebas a la aplicación mediante las reglas de diseño heurístico para su validación.

- Monitorear parámetros relacionados a la calidad de energía en equipos basados en PQ-UAQ y la aplicación desarrollada.

II. FUNDAMENTACIÓN TEÓRICA

2.1 ESTADO DEL ARTE

El desarrollo en la materia del monitoreo de la calidad de energía ha ido evolucionando, en la década de los 90 la tecnología aplicada a clasificación de perturbaciones eléctricas se unificó con varias ramas de la ingeniería, principalmente procesamiento de señales. Generalmente se usan en Computadoras Personales (PC - Personal Computer), sin embargo, estos sistemas frecuentemente son dependientes de una PC para realizar el procesamiento de la señal (Junfeng *et al*, 2012, Ferreira *et al*, 2013), lo cual permite ejecutar el monitoreo en tiempo real debido al sistema operativo inherente, además de disminuir la portabilidad del sistema.

La mayoría de las técnicas utilizadas para la estimación de parámetros de eficiencia energética están basadas en la Transformada Rápida de Fourier (FFT - Fast Fourier Transform) según Deokar *et al* (2014). También se tiene el caso de un análisis de Curtosis Espectral por González De La Rosa et al (2013),de igual manera el uso de técnicas como la Transformada S (ST - S Transform) de acuerdo con Erişti et al (2014), tanto la FFT y la ST son métodos de estimación de defectos en la red eléctrica.

También se usan wavelets (Meher et al, 2004) para el estudio de inter-armónicos y se han desarrollado sistemas de adaptables como redes neuronales Romero-Troncoso et al (2011) y algoritmos evolutivos Wang et al (2011). En la implementación se han hecho investigaciones relacionadas a la aplicación de FPGA

en sistemas de sensores para el diagnóstico de eficiencia energética Humphreys *et al* (2014). Estos métodos han sido comparados en términos de su resolución en frecuencia y los efectos en la determinación de los componentes espectrales.

Desde la invención de los programas de computadora en 1940 hasta el día de hoy, las interfaces gráficas de usuario han ido evolucionando, la programación se ha convertido en una habilidad técnica para millones de personas y conforme ha ido creciendo la importancia de tener una GUI, la evolución con respecto a los sistemas de hardware y a los compiladores desarrollados que cuentan con una interfaz, hoy en día se presentan miles de usuarios usando interfaces gráficas como Matlab ,c#, java, Qt, c++, etc.

Los sistemas móviles son una novedad de este siglo y han estado tomando interés debido a la evolución de la sociedad en una organización más comunicada, se han hecho investigaciones relacionadas a la importancia de los sistemas móviles y como es que estos van a influenciar al futuro de las organizaciones Funk (2006), de igual manera se han realizado estudios en los que se puede observar la mejora en la eficiencia por medio del uso de dispositivos móviles Sheng et al (2005), en general las investigaciones actuales tratan sobre cómo se puede mejorar la eficiencia, la intercomunicación y el gasto Guesmi et al (2014).

El interés de los sistemas móviles ha impulsado el desarrollo de nuevos sistema en los que se usa el sistema Android, esto se debe a la disponibilidad del dispositivo, el tema de Android es un caso muy reciente debido a que el software-hardware usado en los sistemas móviles Android fue introducido en el año 2007,

aun así ya se tienen investigaciones en instituciones de salud Naik *et al* (2015), investigaciones forense Kaart *et al* (2014) e ingeniería Ilarri et al (2015).

2.2 FUNDAMENTACIÓN TEÓRICA

En esta sección se presentaran los diferentes temas relacionados a los conocimientos necesarios para entender el trabajo y la función que va a realizar, como es la calidad energética, procesos relacionados a la calidad energética, dispositivos FPGA, métodos heurísticos de diseño de interfaces y Android.

2.2.1 CALIDAD ENERGÉTICA

La Calidad energética cubre diversos aspectos como es la medición, el análisis y la mejora de la línea eléctrica, los fenómenos relacionados a la calidad energética son generalmente fenómenos físico estocásticos, esto se debe a que en muchos casos aparecen y desaparecen arbitrariamente.

La medición de disturbios resulta más complicada que una simple medición de un parámetro eléctrico, por eso es necesario almacenar los datos de la línea eléctrica por un periodo de tiempo lo suficientemente prolongado para tener una gran cantidad de información y poder encontrar e identificar mediante uno o varios análisis los disturbios que tiene la línea eléctrica.

La principal fuente de anomalías de calidad energética en el suministro eléctrico, es debido a señales armónicas producidas por cargas no lineales de la

las cuales son todas aquellas cargas que no tienen relación lineal entre el voltaje y corriente.

Dispositivos como controladores eléctricos para motores, hornos de inducción, alumbrado y lámparas con balastros eléctricos, se encuentran entre las principales cargas no lineales más comunes. Las corrientes armónicas generadas por cargas no lineales se encuentran desfasadas noventa grados con respecto al voltaje en donde son producidas, esto provoca que fluyan a una potencia distorsionante de la fuente a la red eléctrica y viceversa, que solo es consumida como perdidas por efecto joule que se transforma en calor, de forma equivalente a la potencia reactiva fundamental relacionada al factor de potencia de desplazamiento de acuerdo con Granados (2013).

Hay que tener en cuenta que el monitoreo solo, no es la solución a los problemas generados por los disturbios de la calidad energética, para resolver este problema otras medidas aparte del monitoreo de la red eléctrica tienen que ser tomadas en cuenta, sin embargo tener un sistema de monitoreo puede ser la llave para obtener una solución más eficiente y conlleve a la reducción de costos al momento de obtener o identificar errores, Mazlumi (2011).

2.2.2 INDICADORES DE LA CALIDAD ENERGÉTICA

En el momento de la instalación del dispositivo es necesario tener una forma de medir y garantizar que el sistema está adquiriendo y mandando datos correctamente y al mismo tiempo, comprobar el estado del montaje eléctrico en donde se está instalando, hay que agregar que una de las principales funciones es el identificar pequeños cambios y perturbaciones en la señal eléctrica en tiempo real. Es por esto, que es necesario implementar índices básicos de la calidad energética para adquirir los valores pico-pico de la señal, los valores de la Raíz Cuadrada Media (RMS - Root Mean Square) y los valores de la media. Los valores pico-pico se obtienen simplemente de la diferencia de los valores máximos y mínimos presentes en el intervalo, estos valores están definidos por las ecuaciones (1-6).

$$Vmax = \max(V(t)) \quad (1)$$

$$Imax = \max(V(t)) \;) \quad (2)$$

$$Vmin = \max(V(t)) \quad (3)$$

$$Imin = \max(I(t)) \quad (4)$$

$$Vpp = \max\big(V(t)\big) - \min(V(t)) \quad (5)$$

$$Ipp = \max\big(I(t)\big) - \min(I(t)) \quad (6)$$

Cuando se habla de valores de Corriente Alterna (AC ó Alternating Current) uno debe referirse a los valores RMS. Por definición el valor eficaz de una señal AC es el valor de la tensión de corriente directa que se debe de aplicar a una carga resistiva para que produzca la misma disipación de potencia que si se conectara a la misma carga resistiva de una señal AC.

Voltaje y corriente Rms están definidos por la ecuaciones (7-8), en donde Vn e In se refieren a cada uno de los valores obtenidos del muestreo de la señal, estas mediciones son usadas en el análisis de la señal eléctrica para encontrar pequeñas perturbaciones y poder identificar eventos de la calidad energética.

$$\text{Vrms} = \frac{1}{\sqrt{2}}\sqrt{\sum_{n=1}^{N} {V_n}^2} \qquad (7)$$

$$\text{Irms} = \frac{1}{\sqrt{2}}\sqrt{\sum_{n=1}^{N} {I_n}^2} \qquad (8)$$

Otro de los valores calculados es la media de las señales obtenidas por el sistema de adquisición de datos. Esta medición es usada para obtener el desequilibrio del voltaje, con el cual se pueden detectar fallas en la instalación del montaje eléctrico. Esta medición se encuentra definida por las ecuaciones (9-10) (IEEE ,2009; Granados, 2013)

$$\text{Vmedia} = \frac{1}{\text{n}}\sum_{i=1}^{n} V_i \qquad (9)$$

$$\text{Imedia} = \frac{1}{\text{n}}\sum_{i=1}^{n} I_i \qquad (10)$$

2.2.3 DISPOSITIVOS DE LÓGICA PROGRAMABLE FPGA

Los FPGA’s ofrecen velocidades temporizadas por hardware y fiabilidad, pero sin requerir altos volúmenes de recursos para compensar al gran gasto que genera un diseño personalizado de Circuito Integrado de Aplicación Especifica (ASIC - Application-Specific Integrated Circuit).

El silicio reprogramable también tiene la misma flexibilidad que un software que se ejecuta en un sistema basado en procesador, pero no está limitado por el número de núcleos de procesamiento disponibles. A diferencia de los procesadores, los FPGAs son verdaderamente paralelos por naturaleza, así las diferentes operaciones de procesamiento ni tienen que competir por los mismos recursos. Cada tarea de procesamiento independiente es asignada a una sección del Chip y pueden ejecutarse de manera autónoma sin ser afectada por otros bloques de lógica. Como resultado, el rendimiento de una aplicación no se ve afectado cuando se agregan otros procesos. Cada chip de FPGA está hecho de un número limitado de recursos predefinidos con interconexiones programables para implementar un circuito digital reconfigurable y bloques de entrada y salida para permitir que los circuitos tengan acceso al mundo exterior.

Las especificaciones de recursos de FPGA a menudo incluyen el número de bloques de lógica configurables, numero de bloques de lógica de función fijos como multiplicadores y el tamaño de los recursos de memoria como RAM en bloques embebidos. De las muchas partes del chip FPGA, estos son generalmente los más importantes cuando se seleccionan y comparan FPGA para una aplicación en particular. Todos los FPGA consisten de la misma arquitectura y contienen generalmente tres capacidades básicas, una interfaz de entrada y salida, bloques básicos e interconexiones.(Smith, 2010, Xilinx, 2012).

. 2.2.3.1 ENTRADAS Y SALIDAS

Las interfaces de entrada y salidas son medios por los cuales la información es obtenida o enviada, las señales de la interface las cuales pueden ser unidireccionales o bidireccionales, varias salidas o con una salida .El propósito general de las I/O (Inputs and Outputs, Entradas y Salidas) puede variar, dependiendo del productor, sin embargo, la función general es la misma. Smith (2010).

. 2.2.3.2 BLOQUES BÁSICOS

Los bloques básicos de construcción son recursos pre-configurables que conforman el diseño de un programa, los cuales están interconectadas en varias maneras para determinar funciones definidas en el diseño. Los bloques usados son definidos por la manufacturera del producto, pero generalmente se tienen bloques similares, Smith (2010).

. 2.2.3.3 INTERCONEXIÓN

Interconexión implica conectar los bloques básicos para generar funciones específicas, al igual que conectar la lógica interna con la interfaz externa I/O, la interconexión se realiza por medio de la herramienta de implementación, sin embargo, algunos sistemas permiten al usuario modificar la interconexión manualmente, Smith (2010).

. 2.2.4 LENGUAJE DE PROGRAMACIÓN VHDL

VHDL es un lenguaje hardware de alto nivel usado para describir circuitos digitales, los cuales pueden ser programados a un sistema FPGA. Es un lenguaje que tiene características parecidas a sistemas de programación de software, el cual fue desarrollado por el Departamento de Defensa de los Estados Unidos de América y posteriormente fue adoptado por Instituto de Ingeniería Eléctrica y Electrónica (IEEE ó Institute of Electrical and Electronics Engineers), el cual se determinó como un estándar, Smit (2010).

. 2.2.5 SISTEMA DE ADQUISICIÓN PQ_UAQ

El sistema usado es un dispositivo de monitoreo de variables eléctricas en los sistemas de potencia, esta labor constituye una necesidad esencial para diversas aplicaciones industriales y científicas, tales como el monitoreo de la calidad de la energía, diagnóstico y monitoreo de máquinas eléctricas, sistemas de protección, control entre otros. Debido a la gran diversidad de aplicación existentes en el monitoreo de variables eléctricas, es de gran ayuda contar con un sistema de adquisición de señales eléctricas.

En esta investigación se usó un sistema de adquisición de datos de señales eléctrica basado en tecnología FPGA desarrollado por la Universidad Autónoma de Querétaro llamado PQ_UAQ, el cual se puede observar en la Figura 1. Este sistema es capaz de adquirir las señales de voltaje y corriente en los sistemas trifásicos, así mismo el sistema ofrece la posibilidad de trasmitir datos por medio de bluetooth para su activación y otras instrucciones de control.

A pesar de que el dispositivo debe ser considerado como una incógnita, se tiene que tener en cuenta el formato en el que el sistema envía y recibe datos. La tarjeta PQ_UAQ puede adquirir dos comandos al inicio del sistema, los cuales le indican que información va a ser trasmitida, ya sean los datos para iniciar la sincronización de la PQ_UAQ o el índice de inicio de dibujo de las variables eléctricas del sistema en línea.

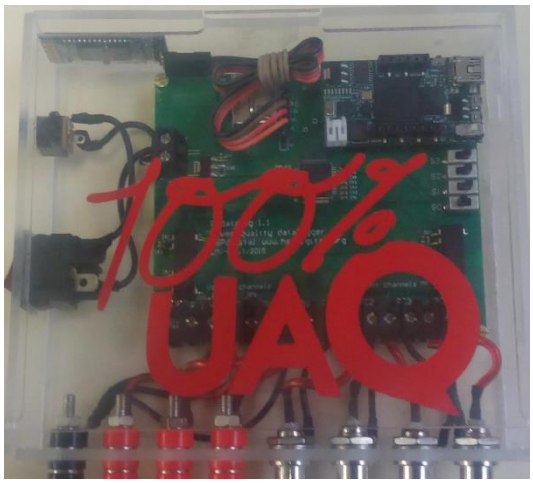


Figura 1. Tarjeta PQ_UAQ

Para poder trasmitir los datos y adquirir instrucciones de control un dispositivo HC-06 fue instalado en la tarjeta PQ_UAQ y se puede observar en la Figura 2. Esta tecnología bluetooth es un dispositivo común en el mercado y da la capacidad de una fácil re-configuración, hay que agregar que este dispositivo usa tecnología bluetooth 2.0, la cual tiene una media de rango de 10 metros, este rango puede aumentar si se amplifica la antena del dispositivo.

Figura 2. Dispositivo HC-06

2.2.6 BLUETOOTH

Cuando se creó la tecnología bluetooth, se tomaron en cuenta varias metas principales entre las cuales se incluyen: operación a nivel mundial, poco costo, robustez, corto rango de la señal, y uso de poca energía. La tecnología de bluetooth, es una de las más económicas en implementar y desarrollar en un sistema en el que se necesita una comunicación segura entre dos dispositivos.

Debido a sus características de diseño, Bluetooth es una de las tecnologías más populares cuando se trata de comunicación inalámbrica, características de robustez y poco consumo eléctrico, lo que la hace una de las mejores tecnologías para trasmitir información. Debido al avance y la inversión de varias compañías, los teléfonos inteligentes Android son capaces de mandar comandos a los dispositivos integrados con bluetooth por medio de un módulo embebido de bluetooth.

La comunicación inalámbrica entre el maestro y el esclavo, se encarga de los procesos de inicio e intercambio de datos, mientras que el protocolo es definido

por la capa de abstracción del Bluetooth. La manera en el cual se genera una conexión y trasmisión de datos entre dos dispositivos bluetooth está definida por el Grupo con Especial Interés en Bluetooth (SIG - Bluetooth Special Interest Group), en el cual se puede encontrar el Protocolo de Adaptación y Control Entre Enlaces Lógicos (L2CAP - Logical Link Control and Adaptation Layer Protocol), que define los detalles de la trasmisión de datos y la configuración de cada paquete. Entre las características del protocolo L2CAP, se tiene multiplexación de canales, segmentación y construcción de datos, control de flujo entre canales, control de errores y manejo de grupos enlazados. La cantidad de datos que se puede adquirir por medio del protocolo L2CAP, tiene como límite 64 kilobytes y se recomienda que la velocidad de trasmisión no sobrepase los 113,200 de velocidad, sobre todo sí se van a realizar pruebas en un ambiente con muchas interferencias físicas y ruido en radiofrecuencia, Bluetooth Special Interest Group (2004).

2.2.7 ARCHIVOS CSV

Por razones de practicidad y constancia se necesitó crear un documento el cual almacene la mayoría de las configuraciones. Estas propiedades están definidas en el documento de Excel, la cual se obtuvo por medio de la calibración de los diferentes tipos de tarjetas a monitorear, los valores y función de las elementos que se encuentran dentro de este documento son mostrados en parte en la tabla 1 y en la tabla 2, Heaton & ebrary, Inc (2002).

Tabla 1 Elementos del archivo pquaq.csv y función.

Opciones	Descripción
ID	Número de tarjeta
Device	Nombre del dispositivo
Port	Puerto para realizar conexión por medio del ordenador
MAC	Cambio de tipo de línea de la señal eléctrica
Comments	referencia de que pinzas usar
Channels	Canales que manda la tarjeta
Name	Nombre del canal
Color	Código hexadecimal de los colores correspondientes al canal
Line	Tipo de línea del canal
Gain	Ganancia del canal
Offset	Desplazamiento que el canal debe de tener
Units	Unidades usadas en el canal

Tabla 2 Elementos del archivo pauaq.csv y función

1	ID	Device	Port	MAC	Comments	Channels	Name	Color
2	1	PQUAQ_01-4	rfcomm3	20:15:04:20:3	Use Voltage	8	Ia	#FF0000
3	2	PQUAQ_02-3	rfcomm4	20:15:08:13:8	Use Voltage	8	Ia	#FF0000
4	3	PQUAQ_03-F	rfcomm5	20:15:04:16:2	Use Voltage	8	Ia	#FF0000
5	4	PQUAQ_04-F	rfcomm6	20:15:04:29:0	Use Voltage	8	Ia	#FF0000
6	5	PQUAQ_05-F	rfcomm7	20:15:05:15:6	Use Voltage	8	Ia	#FF0000
7	6	PQUAQ_06-4	rfcomm2	20:15:04:20:3	Use Voltage	8	Ia	#FF0000

2.2.8 EVALUACIÓN HEURÍSTICA

La Evaluación Heurística (HE - Heuristics Evaluation) 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 enseñar.

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 HE de un diseño, ya que las investigaciones han demostrado que diversas personas encontraran diferentes infracciones, Icarnegie (2008). A continuación se mencionan algunas heurísticas importantes a considerar en el diseño de cualquier interfaz de usuario, las cuales son señaladas 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 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 iconos 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 ignorado 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 que sean aplicables a las tareas del usuario.

2.2.9 SINCRONIZACIÓN DE SISTEMAS DE MONITOREO

La medida del tiempo es una cuestión muy importante en los sistemas distribuidos, esta es igual importante en sistemas centralizados sin embargo no se le da importancia ya que no resulta complicado lograr que todos los dispositivos tengan una misma referencia ya que comparten el mismo reloj. En cambio en múltiples sistemas con diferentes relojes, como en el caso de los sistemas distribuidos se torna difícil poder sincronizarlos. El formato de la medida de tiempo en estos casos igual resulta importante, pues es normal tener que saber a qué hora del día han producido ciertos eventos en un sistema de adquisición.

La precisión requerida para la sincronización del tiempo varía, siendo los sistemas de tiempo real los que posiblemente requieren una mayor precisión. Sin embargo, se presentan casos en donde incluso los sistemas de propósito general requieren de una sincronía horaria. Cada sistema de monitoreo cuenta con un reloj físico, este reloj es un dispositivo electrónico basado en un cristal de cuarzo que oscila a una determinada frecuencia y que además puede programarse para generar interrupciones a intervalos determinados.

Conociendo la frecuencia de las interrupciones, se puede llevar una cuenta del tiempo y por lo tanto la fecha y la hora actual. Esta fecha y hora puede ser utilizada para indicar el momento en el que suceden ciertos eventos en sistemas distribuidos, a estas indicaciones se les llama marca de tiempo. Sin embargo es prácticamente imposible que dos relojes “iguales” oscilen exactamente a la misma frecuencia. Por ejemplo, los relojes basados en cristales están sujetos a una desviación, es decir que cuentan el tiempo a velocidades distintas, por lo que progresivamente sus contadores van distanciados por muy pequeño que sea el periodo de oscilación de los dos relojes, la diferencia acumulada después de muchas oscilaciones conduce a una diferencia claramente observable. La diferencia entre la velocidad de un reloj es el cambio por unidad de tiempo en la diferencia de valores entre su contador y el de un reloj prefecto de referencia. Para los relojes de cristal de cuarzo, esta velocidad de deriva esta alrededor de 10^{-6} lo que significa una diferencia de un segundo cada 11.6 días.

Debido a las diferencias que se producen, se hace necesario tener un sistema conectado a los relojes internacionales en la red, el cual se encargue de

dar una referencia de tiempo para poder realizar un análisis en el momento de adquirir la información. Sin embargo, en el mundo se cuenta con una gran variedad de relojes o tiempos, los cuales se pueden usar para realizar una sincronización. Los relojes más precisos utilizan osciladores atómicos, cuya precisión es del orden de 10^{14} , lo cual corresponde a tener un error de un segundo cada 1400000 años.

Diversos laboratorios de todo el mundo indican el número de tics de los relojes atómicos periódicamente al Buro Internacional de l`Heyre (BIH - Bureau Internationale de l`Heyre), el cual calcula su valor medio generando el Tiempo Atómico Internacional (TAI - International Atomic Time). No obstante este tipo de formato, puede ser difícil de interpretar para usuarios sin experiencia y es una mala decisión cuando las referencias tienen que ser analizadas por usuarios no expertos; constantemente esto se debe a que las unidades de tiempo que se utilizan tienen un origen astronómico, es decir están definidas en función de los periodos de rotación y traslación de la Tierra alrededor del Sol, y resulta que este periodo de rotación se está alargando gradualmente, pues el planeta Tierra está frenando, principalmente debido a la fricción de las mareas, efectos atmosféricos y a las corrientes de convección en el núcleo de la tierra, esto implica, que el tiempo atómico y el universal tienden a desfasarse.

La gradual relentización de la rotación de la Tierra da lugar a que en la actualidad, un día TAI sea 3 milisegundos menores que el día solar. Por esto, cuando la diferencia llega los 900 ms, el BIH añade un segundo “bisiesto” al contador TAI, dando lugar al Tiempo Universal Coordinado (UTC), el cual es la base de todas las medidas políticas, civiles y militares del tiempo. El tiempo UTC está disponible a través de emisoras de radio que emiten señales horarias, mediante

satélites geoestacionarios y GPS cuyas precisiones son de 0,1 a 10 ms y 0,1 a 1 ms respectivamente; esta medición se encuentra actualmente accesible en dispositivos móviles, gracias al internet y actualizaciones constantes que se realizan de manera automática en estos dispositivos.

2.2.10 SISTEMA OPERATIVO ANDROID

El sistema operativo Android fue originalmente desarrollado en Android, Inc. y fue un proyecto fundando en Palo Alto en el año 2003. Varios desarrolladores crearon el sistema operativo en base a un Kernel de Linux. Android ha visto un rápido crecimiento a partir de su lanzamiento en 2008, en 2014 registro 1.5 millones de usuarios al día y proyectando a 1 billón de usuarios para finales de ese año, esto hace a Android una de las tecnologías con más rápido crecimiento en la historia.

Uno de los principales factores en la rápida adopción de dispositivos Android es el hecho que Google libremente licencio el sistema operativo como un software de arquitectura abierta esto abrió la posibilidad de que varias marcas de dispositivos móviles puedan desarrollar teléfonos inteligentes sin la necesidad de desarrollar su propio sistema operativo, ejemplo de esto son empresas como LG, Motorola, Samsung, Sony. etc., las cuales usas el sistema operativo Android en sus dispositivos móviles.

Para desarrollar en Android, se requiere que el dispositivo se ajuste a ciertas especificaciones de hardware y términos de contrato. Para Google el programa de compatibilidad es una manera de reducir la variación en la funcionalidad del hardware, para así crear un ambiente más estable para desarrolladores. De igual manera Google publico el código fuente sin restricciones

de certificación, llamado Proyecto Android de Fuente Abierta (AOSP ó Android Open Source Project)), este es el sistema operativo completo, el cual cualquiera puede descargar y modificar, esto abrió la posibilidad de que empresas como Amazon y Xiaomi tengan su propia versión de Android, al igual otras compañías que no desarrollan en dispositivos móviles Pon et al (2014).

Gracias al avance constante en el desarrollo de dispositivos móviles, la tecnología Android se ha hecho más eficiente cada generación y la cantidad de sensores y hardware se han incrementado debido a la necesidad de los consumidores y a la competencia, al contar con un dispositivo móvil Android uno adquiere la posibilidad de tener un sistema de desarrollo que cuenta con sensores y capacidad de procesamiento en la palma de la mano, Meier (2010).

2.2.12 FRAGMENTOS

Cuando empezaron a aparecer los dispositivos de gran tamaño tipo Tablet, el equipo de Android tuvo que solucionar el problema de la adaptación de la interfaz gráfica de las aplicaciones a esa escala de la pantalla. Esto es debido a que cuando se diseña una interfaz gráfica para funcionar en una pantalla de un teléfono móvil, esta GUI no se llega a adaptar a una pantalla de varias pulgadas mayor. La solución de esto vino en forma del componente llamado Fragment.

Android que introdujo el elemento de Fragment desde la versión 3.0, permitió que los desarrolladores puedan contar con una GUI más flexible y dinámica. El uso de fragmentos permite la creación de tales diseños sin la necesidad de hacer grandes cambios a la forma en cómo se crea la interfaz de la aplicación, esto lo

logra al dividir la ventana principal en distintos fragmentos y esto permite modificar la apariencia de la actividad en tiempo real y guardad esos cambios en un proceso simple y con un simple procesamiento.

Una de las principales características de Fragment, es la capacidad de combinar múltiples fragmentos en una sola actividad, para construir una actividad que cuente con varios paneles y de igual manera reusar estos fragmentos en otras actividades que se ejecutan. Un fragmento siempre se encuentra enlazado a el ciclo de vida de la actividad, de esta manera un fragmento se encuentra directamente afectado por los procesos que la actividad realiza. Esto quiere decir, que si la actividad es cerrada, el fragmento igual es parado y si la actividad es pausada, todos los fragmentos son igualmente pausados. Sin embargo, cuando la actividad está corriendo se pueden ejecutar funciones diferentes e individuales en cada uno de los fragmentos.

Para crear un fragmento es necesario llamar a la subclase Fragment, los Fragmentos tienen código parecido a las actividades principales. Los Fragmentos igual contiene métodos disparados por ciertos eventos tales como onCreate (), onStar (), onPause (), y onStop (). De hecho se pueden transferir todas las funciones y métodos usados en la actividad principal a los fragmentos. Un Fragmento igual es capaz de realizar operaciones sin necesidad de una pantalla de diseño propia, lo cual le permite funcionar como una clase o una actividad, The Android Open Source Project (2016).

2.2.13 HILOS

Los hilos son elementos que nos permites ejecutar varios procesos a la vez. Por lo tanto, esto permite hacer programas que se ejecuten en menor tiempo y sean

más eficientes. Evidentemente no se pueden ejecutar infinitos procesos de forma concurrente ya que el hardware tiene sus limitaciones, las cuales dependen de la cantidad de núcleos con el que cuente el dispositivo.

En Android para utilizar multitarea debemos usar la clase thread, esta clase implementa la interface Runnable, por default todos los componentes de la misma aplicación corren en los mismos componentes.

En proyectos reales no se pueden pretender todas las acciones que lleva a cabo una aplicación ya sea simples o concisas. En ocasiones hay instrucciones que toman unos segundos en terminarse, esta es una de las mayores causas de la terminación abrupta de las aplicaciones. Sin embargo, muchos de estos problemas se pueden evitar si se entiende apropiadamente la carga de cada hilo, los tiempos en los que se terminan y al impedir malas prácticas de programación al manejar los hilos, The Android Open Source Project (2016).

2.2.14 HANDLERS

Un handler permite al usuario mandar mensaje y objetos que formen parte del hilo principal, al igual que introducir métodos con funciones específicas. Generalmente es preferible que se ejecuten procesos complicados y pesados en el handler, como cálculos o procesos conectados con el hilo de la interfaz gráfica, debido a que este elemento es el que está más cercano al hilo principal del sistema y al igual los procesos ejecutados en el handler toman prioridad al momento de dividir la carga de tiempo y procesamiento en la aplicación, es generalmente en donde se va a concentrar la carga en procesamiento en la mayoría de las aplicaciones, es debido a esto que es de gran importancia optimizar el código que se encuentre dentro del handler. Un handler igual tiene la propiedad de cambiar y mandar mensajes a otros

hilos, al igual que levantar banderas las cuales afecten a los Fragmentos en la aplicación, The Android Open Source Project (2016).

2.2.15 CANVAS Y PAINT

Las clases Canvas y Paint del sistema operativo Android son métodos que existen desde su primera versión, esto quiere decir que todos los dispositivos Android deberían de ser capaces de usar ambas librerías. Un objeto Paint es generalmente usado para definir colores al definir un objeto, pero este objeto almacena más que colores, el objeto Paint encapsula el estilo, colores complejos con opciones e información de renderizado, este objeto puede ser aplicado a un gráfico como podría ser una figura o incluso un texto especifico.

Se puede establecer el color del objeto Paint con el sencillo método setColor, con el cual se puede decidir entre un gran rango de colores, sin embargo generalmente se terminan usando colores estándar por su fácil acceso, estos colores están predefinidos en la clase Android.Graphics.Color, mediante esta clase un valor entero puede ser usado para definir el color base para el objeto Paint, sin embargo si se necesita de colores más específicos se tiene el método setRGB (). Para definir los colores en la gama RGB (Red, Green, and Blue, Rojo, Verde y Azul). De igual manera se definen los estilos del objeto Paint. Por ejemplo si definimos el estilo de un objeto Paint como estilo STROKE, el objeto Paint será dibujado por medio de líneas y no será llenado.

Es importante mencionar que la clase Paint igual puede realizar el renderizado del texto en la pantalla en diferentes fuentes y estilos, en este caso los formatos de textos no están limitados a los ofrecidos desde el Android Studio, debido a que se pueden cargar estilos de texto por medio de archivos descargados

en el AssetManager. Cuando en una aplicación desea realizar un dibujo y control de animaciones gráficas, se recomienda que se haga por medio del objeto Canvas.

El objeto Canvas contiene las características de forma y dimensiones al crear un dibujo. Las dimensiones del objeto Canvas están delimitadas por el contenedor del View, el cual tiene los datos del tamaño de la pantalla, de tal manera se hace simple adquirir los valores y dimensiones de la ventana en donde se encuentran para así obtener un factor de escala.

Un Canvas funciona como una superficie en donde gráficas y figuras serán dibujadas. El Canvas hace uso de evento onDraw (), por medio de este evento resulta sencillo crearlo, ya que solo es necesario hacer las llamadas a las funciones para construir diferentes tipos de métodos. Mediante Canvas todas las figuras serán creadas sobre un Bitmap, el cual es creado en la ventana que encapsule al objeto Canvas, The Android Open Source Project (2016.).

2.2.16 SURFACEVIEW

El surfaceView es una subclase que ofrece un área de dibujo dedicada dentro del View o la ventana de la superficie asignada. El propósito de esto es ofrecer a la superficie de dibujo un hilo secundario, de tal manera que la aplicación no necesite esperar hasta que el sistema esté listo para dibujar. Esto le da la propiedad a la aplicación de que en lugar de esperar por otros métodos u objetos, la aplicación puede dibujar a su propio paso en sus propias dimensiones.

Primero se necesita crear una clase que extienda SurfaceView. Esta clase también debería de implementar *SurfaceHOlder.Callback*. Esta subclase es una interfaz que notificara al usuario con información de la superficie en donde se está

dibujando, tales como la destrucción, cambios y creación de tal objeto o superficie. Estos eventos son importantes de tal manera que se pueda saber cuándo comenzar a dibujar o si se necesitan hacer ajustes dependiendo de las nuevas superficies definidas.

Para dibujar en el SurfacHolder uno tiene que pasar el hilo en donde el SurfacHolder ha sido creado y adquirir el Canvas por medio de *lockCanvas ()* y de esta forma asignarle al Canvas un SurfaceHolder definido, para ejecutar los dibujos necesarios, The Android Open Source Project (2016).

2.2.17 APLICACIONES MÓVILES ANDROID

A pesar que la mayoría de los dispositivos móviles están bajo reglas de diseño de compatibilidad, Al desarrollar en dispositivos android se encuentran con varias retos si se desea compatibilidad entre diferentes dispositivos diferencias entre dispositivos como el tamaño del dispositivo y falta de sensores o en algunos casos cambios en el código fuente de Android, al igual al tener un dispositivo con menor tamaño se presentaran problemas de baja capacidad de procesamiento, limitado espacio de almacenamiento, poca resolución de pantalla, lenta transferencia de datos, peores enlaces de conexión y limitada vida de la batería. Sin embargo de igual manera se cuenta con un gran avance y desarrollo mejorando diferentes aspectos como memoria flash y memoria en estado sólido, tamaños de pantalla incrementados, más núcleos, debido a esto hay una gran diferente gama de productos, lo cual genera un reto a la hora de diseñar, es importante entonces tener en cuenta el uso de recursos cuando se diseña una aplicación móvil, para de esta manera poder crear una buena interfaz, sobre todos cuando la mayoría de los

usuarios necesitan o prefieren una interfaz gráfica que contenga una buena cantidad de información en una representación simple y sencilla de entender, al igual tener en cuenta que la tendencia de hacer dispositivos móviles con pantallas grandes está incrementando así que es importante diseñar a pensando en el futuro uso de la aplicación, de manera que la interfaz sea capaz de escalarse y adaptarse a tamaños chicos y grandes de pantallas Meier (2010).

2.2.18 ANDROID STUDIO

Android Studio es un IDE (integrated development environment, ambiente de desarrollo integrado) similar a eclipse, sin embargo este software está basado en Intellij IDEA, la cual es una herramienta con gran popularidad entre los desarrolladores de Java, muchas de las características que posee el paquete de ADT se pueden implementar desde Android Studio. Vale agregar que muchas características que posee el android estudio no son accesibles en el software de ADT o eclipse. El programa Android estudio es el software que se eligió para la generación de la aplicación, la interfaz en si se divide en una vista general, el directorio del proyecto y de estructura, una ventana para definir la parte grafica de la aplicación y una división en donde se especifica la funcionalidad de cada uno de los componentes que conforman a la aplicación Annuzzi et al (2010).

III. METODOLOGÍA

En esta sección se muestran a detalle los pasos y tareas a realizar para alcanzar los objetivos. Entre estos pasos se encuentra el diseño general de la aplicación, el desarrollo de módulos, pruebas de la aplicación, desarrollo de la interfaz gráfica, por último la obtención de gráficas y coeficientes de CE para determinar un monitoreo exitoso de la tarjeta de adquisición y por último la obtención de resultados. EL diagrama de la metodología de la aplicación es presentada en la Figura 3.

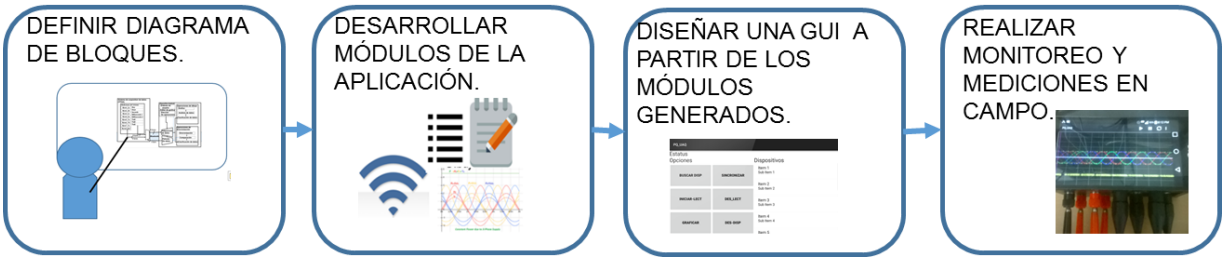


Figura 3. Metodología de diseño de la aplicación

3.1. DIAGRAMA DE BLOQUES

En el diagrama de bloques presentado en la Figura 4, se observa el proceso básico que se realiza en la comunicación del dispositivo, en esta comunicación se pueden ver dos módulos principales, la tarjeta de adquisición de datos y el dispositivo Android.

La tarjeta FPGA se considera para fines de esta investigación como una caja negra de la cual lo único que se conoce es la velocidad de trasmisión, el tamaño de la información trasmitida y el formato de los paquetes trasmitidos.

La aplicación por otra parte está representada en su totalidad, para manejar el intercambio de datos inalámbrico y se utilizó la tecnología Bluetooth, la cual envía un requisito de información dependiendo de las instrucciones del usuario. Si el usuario decide que necesita una sincronización se envía una cadena de datos de referencia, de igual manera una cadena es enviada para avisar al FPGA que el proceso de sincronización ha comenzado.

Si el usuario necesita realizar un dibujo de una lectura a la red eléctrica, la aplicación se encarga de mandar una cadena de datos indicándole a el FPGA que el inicio del monitoreo ha comenzado, marcando el inicio de la adquisición de los datos de la de la señal trifásica, al adquirir los datos esta información es dividida, catalogada y usada para mostrar los indicadores de la calidad energética de voltaje y corriente pico y usar la información de dibujo.

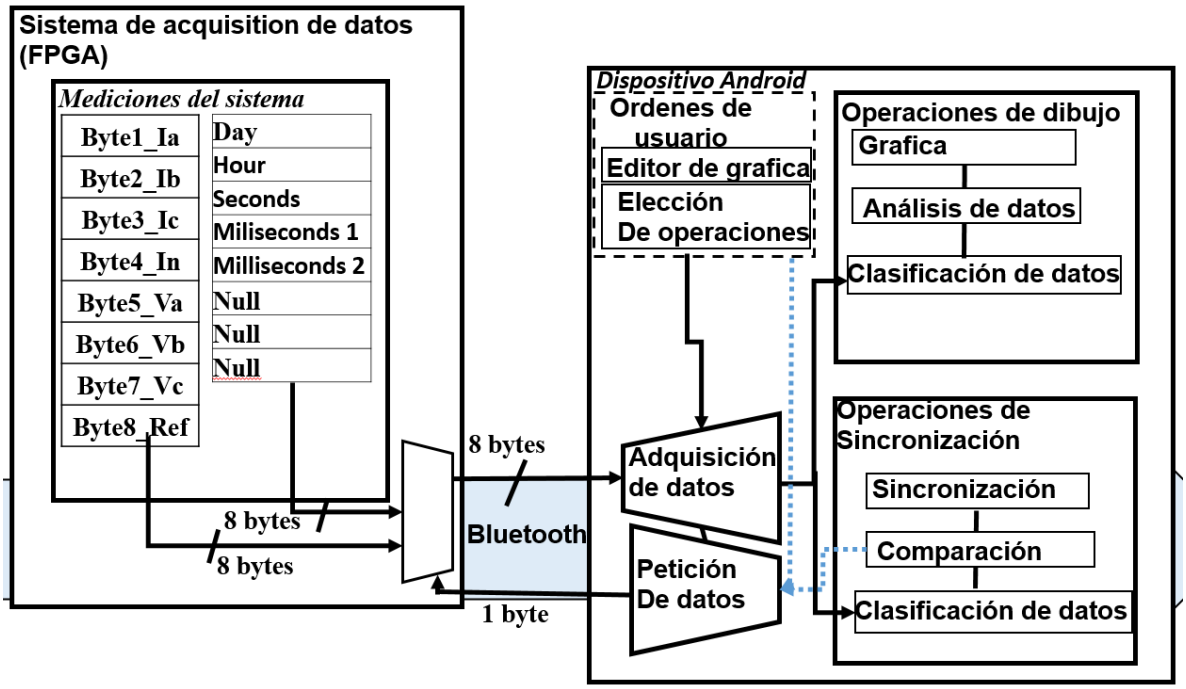


Figura 4. Diagrama de bloques de la aplicación

3.2. MÓDULOS DE LA APLICACIÓN

Con la finalidad de cumplir los objetivos de la investigación, el proyecto fue dividido en varios módulos como se muestra en la Figura 5, entre estos el módulo que cuenta con un carácter más crítico, es el módulo de comunicación de bluetooth, a partir del cual se adquieren los formatos y arreglos de información.

Una vez ya sabiendo el tamaño y características de los paquetes obtenidos un módulo encargado del envío y recepción de datos necesita ser desarrollado y probado, ya teniendo la interpretación de los datos, un módulo para ejecutar ordenes de usuario se desarrolla para contar con la característica de elección entre varias opciones, es de igual importancia construir una interfaz comprensible que sigua las reglas de diseño

heurístico, hay que recalcar que una referencia de las mediciones realizadas necesita ser almacenada para su futuro uso en el momento de hacer una correlación de información y de esta manera encontrar eventos de PQ más eficientemente; es con esta finalidad que se determinó el desarrollo de una bitácora, finalmente módulos de dibujo y sincronización para cumplir el requerimiento de dibujar los datos adquiridos y el manejo de la sincronización de las tarjetas respectivamente.

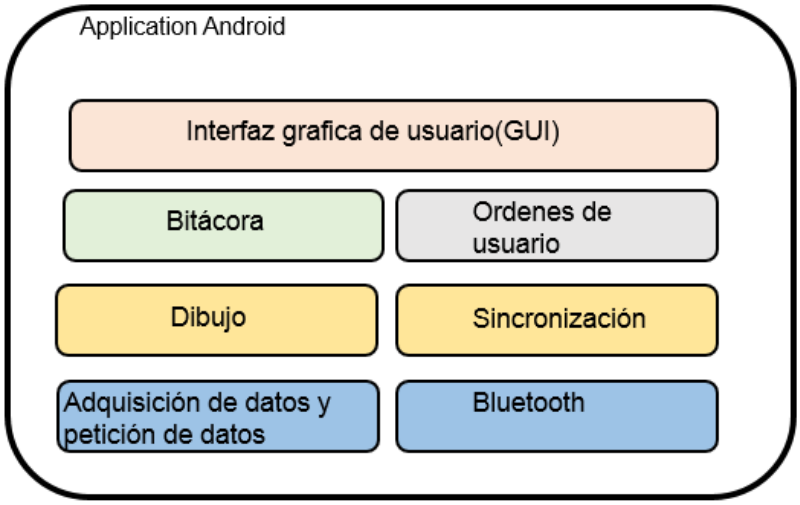


Figura 5. Módulos de la aplicación

3.2.1. MODULO BLUETOOTH

Como se ha mencionado antes la comunicación planteada se realiza mediante tecnología bluetooth, Los dispositivos móviles Android desde la primera versión comercial han contado con tecnología bluetooth, de igual manera el sistema operativo Android cuenta con clases y métodos de simple uso por los cuales se puede acceder y establecer una comunicación con otros dispositivos bluetooth de

la misma generación. Estas clases y métodos permiten al teléfono móvil intercambiar información, habilitando una comunicación punto a punto y multipunto con dispositivos bluetooth.

Gracias a las APIs de bluetooth una aplicación puede realizar escaneos por otros dispositivos, otra de las propiedades que tiene este método es la capacidad de organizar y alinear dispositivos anteriormente enlazados, establecer canales de RFCOMM, transferir información a otros dispositivos, administrar múltiples conexiones. En esta aplicación un módulo de bluetooth fue desarrollado con el fin de intercambiar y mandar información entre dos dispositivos, se usó de base una librería ya conocida y de software abierto conocido como *BluetoothChat*.

En esta investigación fue necesario realizar algunos cambios como son: los permisos de Bluetooth en el código tuvieron que ser establecidos, estos permisos son necesarios para todo tipo de comunicación bluetooth, tales como pedir conexiones, aceptar conexiones y transferencia de información. Para que una aplicación pueda descubrir dispositivos por si misma se necesita el permiso de *BLUETOOTH_ADMIN*. La mayoría de las aplicaciones necesitan de este permiso simplemente para habilitar la característica de descubrir nuevos dispositivos. Sin embargo, al establecer conexiones básicas el permiso de *BLUETOOTH* tiene que ser declarado en el manifiesto de la aplicación.

En la aplicación se tiene varias clases que se encargan de configurar y administrar la conexión bluetooth con otros dispositivo, esto se logra por medio de un hilo que está a la espera de conexiones entrantes, al igual otro hilo se encarga de conexiones con otros dispositivos y un último hilo para realizar las trasmisiones

una vez conectado, de manera que cuando se lee un mensaje esto activa un evento en el hilo de lectura, llamado mensaje_leido en el código y cuando se quiere enviar un mensaje simplemente se introduce la información que se quiere enviar en el evento del hilo de mesaje_escrito, posteriormente la información es administrada por los módulos de adquisición de datos y petición de datos, mientras tanto la información obtenida de la dirección MAC del dispositivo es usada para la selección de ganancias correspondientes por medio del archivo CSV, de igual manera configuraciones de color son seleccionados desde dicho archivo.

3.2.2. ADQUISICIÓN DE DATOS Y PETICIÓN DE DATOS

El diagrama del proceso para realizar la adquisición de datos y la petición de datos se muestra en la Figura 6, todo el proceso de manejo de datos se inicializa al momento de que el evento de la función OnCreate( ) es disparado, en este evento se realizan las primeras instrucciones de manera que es recomendable hacer declaraciones e instanciar cualquier tipo de variable que sea necesaria, a partir de los elementos inicializados de la GUI como botones en el primer fragmento, el usuario es capaz introducir instrucciones, posteriormente se realizar las operaciones indicadas por el usuario.

En el momento de realizar la conexión con el dispositivo PQ_UAQ, la trasmisión y recepción de la información son administrados por el API de Bluetooth, esta trasmisión y recepción dependen totalmente de las librerías de Bluetooth, al igual que la interacción con el handler del main el cual espera por eventos de envío y recepción de datos, el evento de trasmisión es activado llamando a la operación *Mensaje_escrito* del handler el cual se encarga de mandar los datos al módulo de

Bluetooth, este se encarga de dividir el mensaje en tramas de bytes y mandarlo por medio del protocolo de bluetooth.

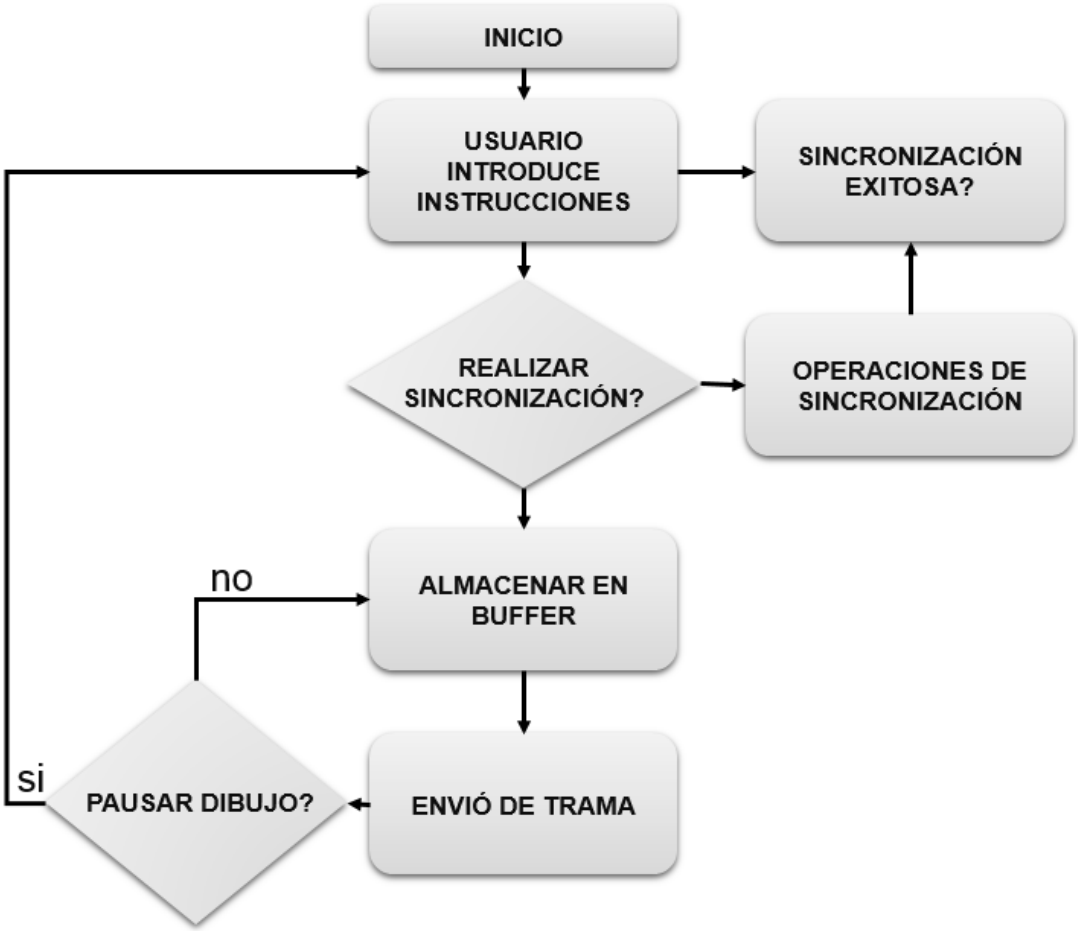


Figura 6. Diagrama básico de la adquisición y petición de datos

Cuando se activa el evento de recepción en el módulo del bluetooth toda la información de una trama es almacenada en una FIFO en la cual se almacenan tramas de 1 byte, cada byte recibido es almacenado y dependiendo de la operación

seleccionado para una variable y uso específico, sin embargo se conoce que el tamaño de las tramas de información recibidas del FPGA es de 8 bytes para cada una de las operaciones a realizar y cada vez que el modulo del Bluetooth recibe un byte de información de la tarjeta PQ_UAQ se activa un evento de recepción, por lo tanto es posible contar la cantidad de datos recibidos por la aplicación. Igual es importante agregar que la información almacenada en la FIFO puede ser accedida desde cualquier clase que se encuentre en el proyecto de la aplicación, esto se debe a que se creó una copia global de la FIFO la cual se actualiza cada vez que los eventos del handler sean ejecutados.

El handler seguirá llenando la FIFO hasta llegar al final de la trama, cuando se dispara el evento de fin de trama se manda una señal para iniciar el módulo de dibujo o de sincronización. Una vez terminado el proceso de recepción de una trama, la FIFO es vaciada para garantizar que cuando se reciba la información de la siguiente trama, la aplicación no se llene de datos basura que mostrarían datos erróneos. En la figura 7 y 8 se puede observar los caracteres que se envían y la organización de los paquetes recibidos por la tarjeta PQ_UAQ para la operación de dibujo y sincronización respectivamente.

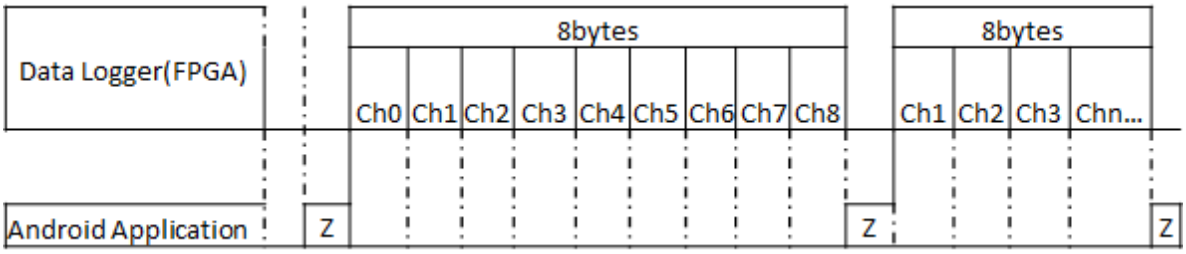


Figura 7. Proceso de petición de adquisición de datos para el dibujo de señales.

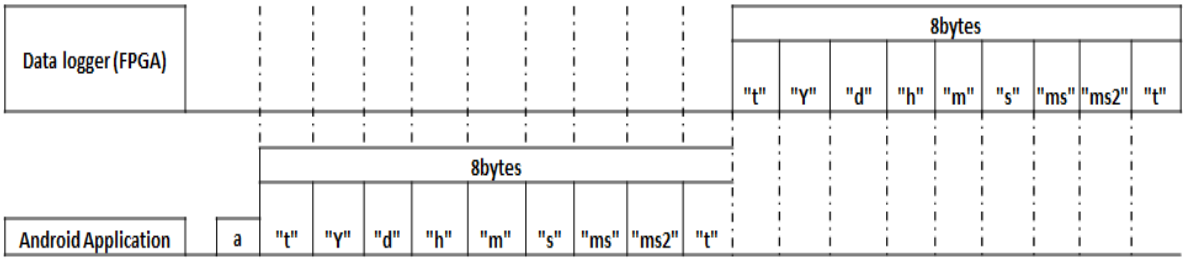


Figura 8. Proceso de petición y adquisición de datos para la sincronización.

3.2.3. INTERFAZ GRAFICA DE USUARIO

Con el fin de que el usuario pueda tener una interacción con la aplicación y el dispositivo de monitoreo, es necesario crear una GUI. En este proyecto, la GUI fue desarrollada usando las opciones de creación de Layouts del software Android Studio.

Android Studio cuenta con un creador de Layout, el cual permite editar y modificar Layouts básicos de la aplicación, al igual que editar características por medio de XML. Esto da la opción de desarrollador una creación más fácil de una interfaz gráfica.

Una de las primeras tareas al desarrollar una aplicación, es entender el funcionamiento del software a desarrollar y determinar los elementos necesarios. Las principales funciones que la aplicación debe de tener son: mostrar dispositivos de bluetooth a conectar, dar opciones para activar el inicio de muestreo, sincronización e inicio de dibujo, realizar un monitoreo constante de las señales obtenidas, permitir guardar datos adquiridos del muestre de las señales en una bitácora para usar en referencias en investigación o reportes del estado del

monitoreo, la Figura 9 presenta la función general de cada fragmento y el orden que siguen.

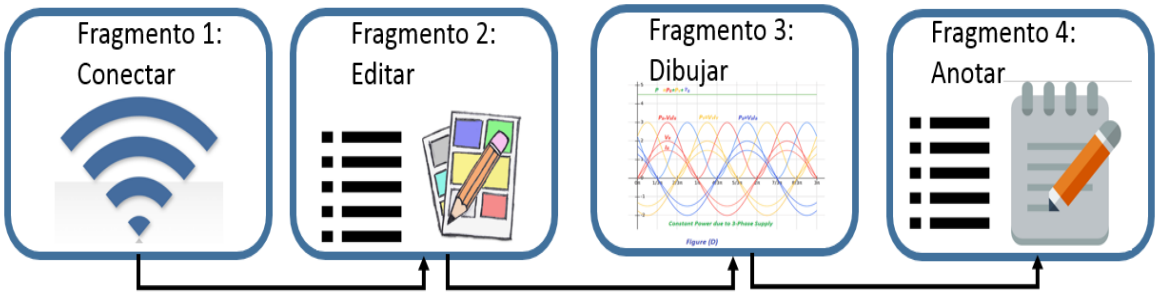


Figura 9. Fragmentos que conforman la GUI de la aplicación y función general

La interfaz gráfica está dividida en varios fragmentos para poder separar las operaciones y módulos requeridos en ventanas visibles y fáciles de entender, además de una manera fluida de trasportarse entre ventanas sin destruir la actividad actual que se encarga de la mayoría de los procesos de cálculo. Dichos fragmentos son controlados por un PageAdapter que se encarga de organizar, ordenar y decidir la distribución de las ventanas mostradas. Cada fragmento se proyectó pensando en las reglas de diseño heurístico con un diseño sencillo y simple.

3.2.4. CONEXIÓN

La primera ventana que se necesita visualizar en la aplicación es el fragmento de conexión, en donde se sitúa la ventana de conectividad, dando así la opción de generar un enlace con diferentes dispositivos Bluetooth 2.0, de igual manera el primer fragmento contiene opciones de sincronización y activación de

dibujo en la misma ventana, en la tabla 3 se muestra cada una de las operaciones implementadas en el primer fragmento y la descripción de sus funciones específicas

Tabla 3 Operaciones del primer fragmento y descripciones

Opciones	Descripción
1 Buscar Dispositivos	Por medio de esta operación se detectan nuevos dispositivos de bluetooth, para su futura conexión..
2 iniciar Lectura	Este comando se encarga de comenzar el inicio de la adquisición de datos en el PQ_UAQ conectado, igual inicializa los valores del *Paint* de cada uno de los canales a dibujar durante la graficación.
3 Desactivar lectura	Detención de adquisición de datos para preservar segura la memoria de la tarjeta USB de la PQ_UAQ.
4 Sincronizar	Este comando inicia el proceso de sincronización y puede ejecutarse cuantas veces se necesite para tener una sincronización exitosa.
5 Desconectar Dispositivo	Desconectar la tarjeta de tal manera que no se presenten problemas de conexión con otros dispositivos de bluetooth.
6 Graficar	Este Botón se encarga de crear la ventana de graficación y la bitácora para realizar el monitoreo del sistema.

Estas funciones son activadas por medio de botones, mientras que la lista de dispositivos encontrados fue creada en un listview y la edición se realizó por medio de un adaptador el cual se encarga de poner las opciones correspondientes en la ventana de manera dinámica, en la Figura 10 se presenta el diseño del fragmento de conexión. En la parte superior se cuenta con un TextView que muestra

el estado de las operaciones de activación y sincronización, esto con el propósito de seguir las reglas de diseño heurístico para la característica visibilidad del estatus del sistema.

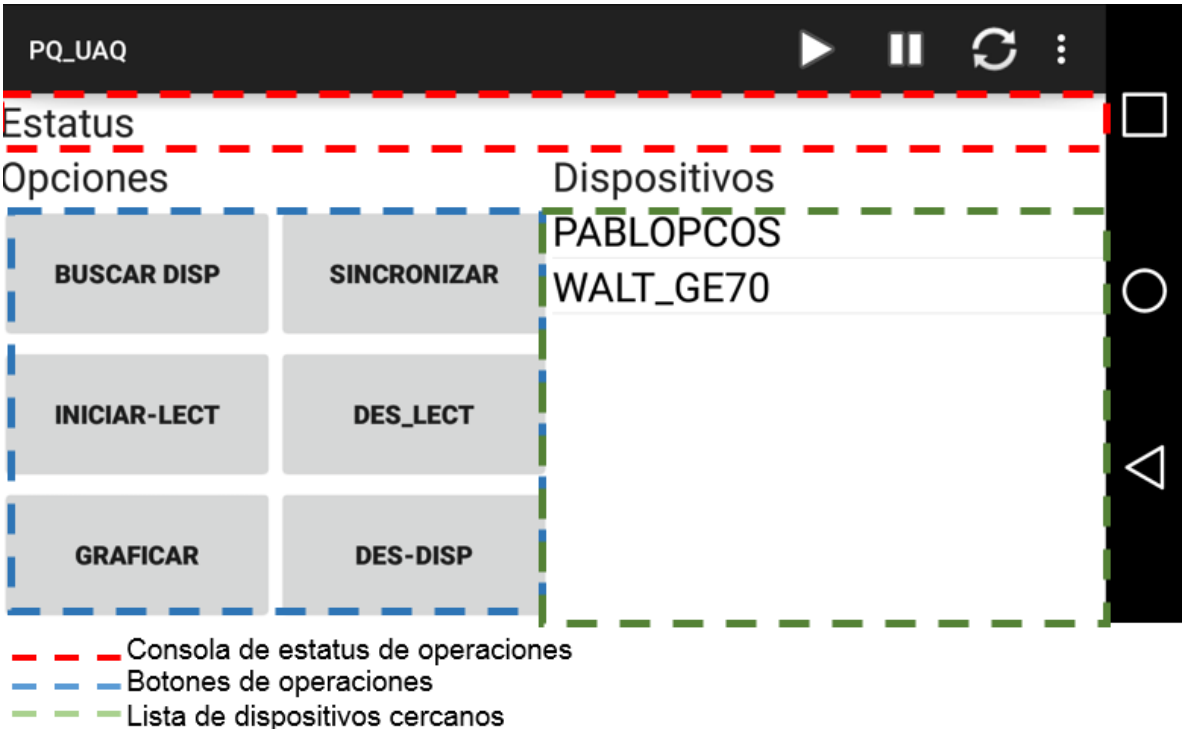


Figura 10 Primer Fragmento para las operaciones de conexión

3.2.4.1. SINCRONIZACIÓN

Este módulo se realizó con la función de que se pueda generar la sincronización entre varios dispositivos y de esta manera investigar la propagación de un fenómeno de la calidad energética en los diferentes dispositivos. En la Figura 11 se presenta un diagrama de la composición de los paquetes enviados durante la sincronización.

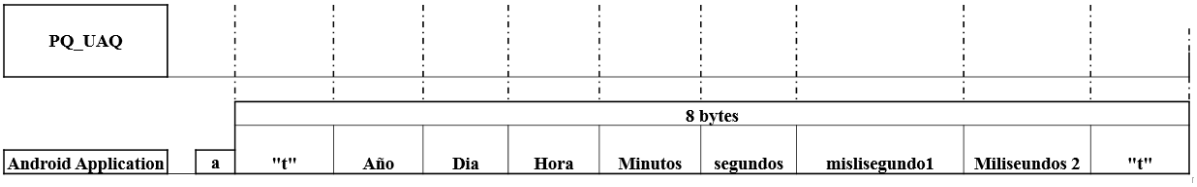


Figura 11 Composición de los paquetes enviados durante la sincronización

En la sincronización los paquetes de información son mandados para adaptarse a el Código Decimal Codificado a Binario (BCD ó Binnary Coded to Decimal), en la Figura12 se puede observar que los paquetes cuentan con un carácter al inicio de la comunicación y posteriormente se manda el primer paquete en BCD del dispositivo Android el cual se encarga de mandar un tiempo de referencia que concuerde con el tiempo del dispositivo móvil, como se observa en la Figura 11, los paquetes mandados y recibidos comienza y termina con una carácter ¨t¨ el cual es usado para determinar el inicio y fin de los datos trasmitidos.

Los valores enviados corresponden a los valores en código BCD de año, día, hora, minutos, segundos, milisegundos1 y milisegundos 2, estos dos últimos resultan de la división del milisegundo debido a que el tamaño de cada uno de los

datos trasmitidos está limitado a un byte y los datos de un milisegundos expresado en BCD pueden llegar a exceder el byte de información.

En la Figura 12 se muestra el diagrama de flujo básico por el cual se ejecuta el proceso de sincronización, las variables Hand corresponde al tiempo del dispositivo Android en código UTC, mientras que Hfpga es el tiempo del FPGA adquirido del dispositivo PQ_UAQ, en este proceso se mandan caracteres U y D, que sirven para atrasar y adelantar 10 minutos el tiempo de la tarjeta FPGA en caso de atrasos o adelantos en el reloj interno del FPGA.

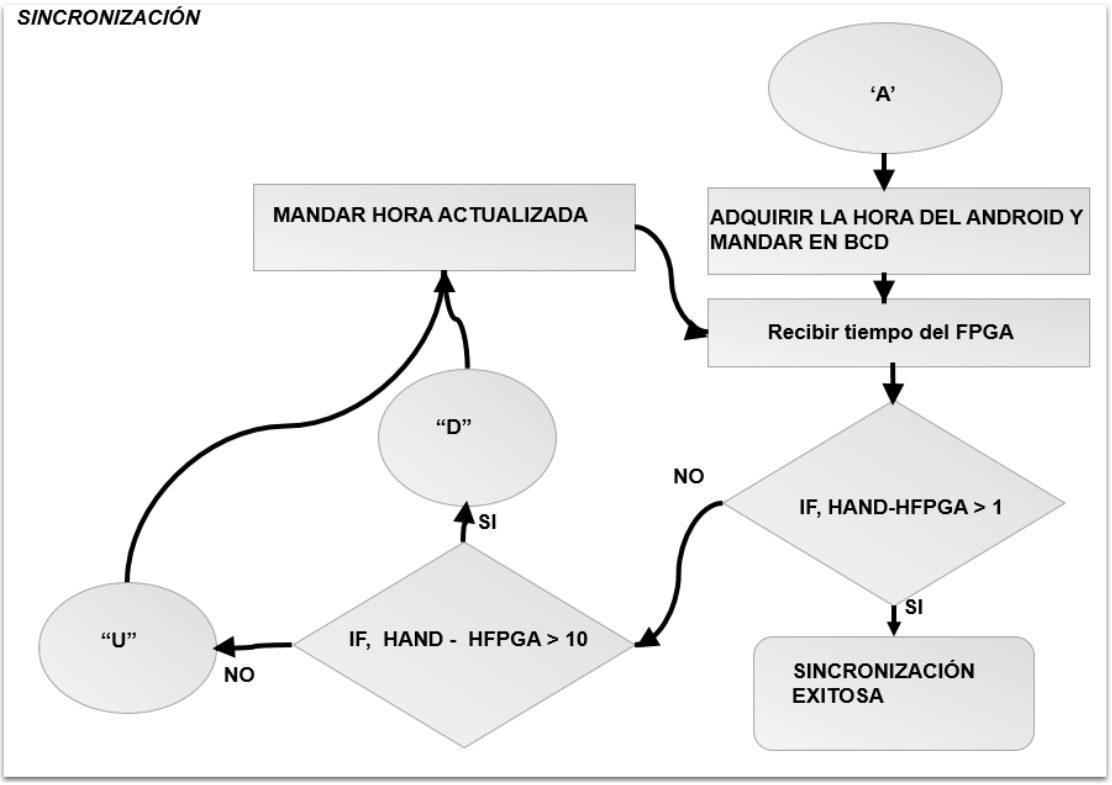


Figura 12. Diagrama de flujo del proceso de sincronización

Este diagrama de flujo contiene el proceso base para realizar la sincronización sin embargo este proceso se realiza por medio de varias funciones, en la Tabla 4 se muestran cada una de las funciones implementadas con una descripción de lo que hace cada una.

Tabla 4. Funciones usadas para la sincronización y descripciones.

Funcion	Descripcion
sendMessageBCD	Se encarga de enviar el mensaje codificado a la tarjeta de adquisición
BtnSync()	Es el botón que dispara la función de sincronización
DoRunable()	Este proceso inicializa el runable SyncR dependiendo de las veces que se hagan peticiones de sincroniza-ción
SyncR	Proceso principal de sincronización envía, recibe datos de hora, para hacer comparaciones.
sendSyncTime()	Envía mediante sendmessageBSD cada una de las tramas de los valores de tiempo
BDivide	Se encarga de dividir y decodificar el tiempo adquirido
read_rtc_time	Asigna cada una de las tramas de información a variables fáciles de usar en bytes.
bcd_2_bin	Ayuda a decodificar convirtiendo los valores obtenidos en BCD a bytes.
STimeVars	Determina las variable de tiempo de la aplicación para su posterior trasmisión
St_2_B_BCD	codifica los valore de tiempo a BCD para ser mandados por medio de sendSyncTime

3.2.5. EDITOR

Durante el desarrollo de la aplicación se vio la necesidad de visualizar señales individuales o en diferentes combinaciones, por este propósito un editor de la gráfica fue generado en donde se eligen los canales que se desean ver, estos cambios se realizan por medio de la clase DibujoSwitch(), la cual se encarga de guardar los datos del buffer para posteriormente ser graficados, esta clase cuenta

con un Switch en el cual se introducen valores enteros y por medios de estos, la aplicación decide que graficar de manera sencilla, estos enteros son determinados por los botones en el segundo Fragmento el cual se puede observar en la Figura 13, al apretar estos botones se forma el valor entero que será guardado en las variables Vfull para el voltaje y Afull para las corrientes, de igual manera se provee de una forma visual de observar los cambios gracias a los cambios en los botones y a la información mostrada en las etiquetas.

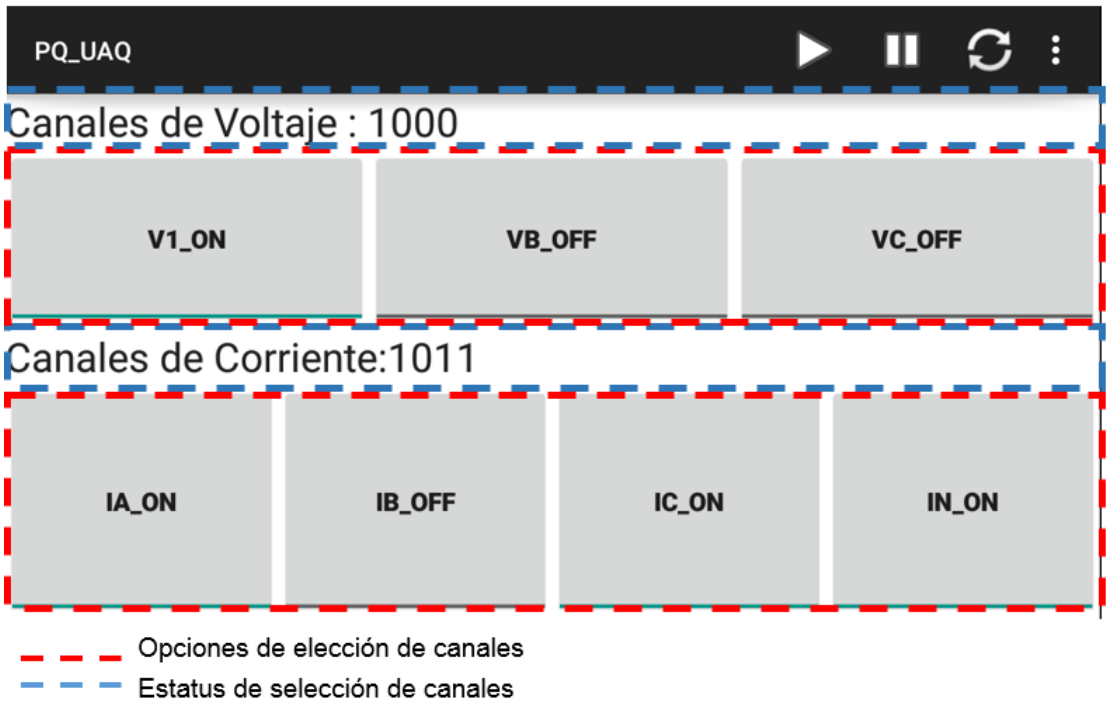


Figura 13. GUI de segundo Fragmento para opciones de edición

En la Tabla 5 y 6 se muestra las diferentes combinaciones de los cambios que se pueden producir al cambiar Vfull y Afull respectivamente.

Tabla 5 Combinaciones y efectos de Vfull

	Vfull	canal V1	canal V2	canal V3
	0001			**x**
	0010		**x**	
	0011		**x**	**x**
	0100	**x**		
	0101	**x**		**x**
	0110	**x**	**x**	
	0111	**x**	**x**	**x**

Tabla 6 Combinaciones y efectos de Afull

Afull	chanal A1	canal A2	canal A3	canal AN
0001				**x**
0010			**x**	
0011			**x**	**x**
0100		**x**		
0101		**x**		**x**
0110		**x**	**x**	
0111		**x**	**x**	**x**
1000	**x**			
1001	**x**			**x**
1010	**x**		**x**	
1011	**x**		**x**	**x**
1100	**x**	**x**		
1101	**x**	**x**		**x**
1110	**x**	**x**	**x**	
1111	**x**	**x**	**x**	**x**

Otras opciones que afectan a la gráfica como colores y estilos de líneas, son determinadas por el usuario modificando el archivo CSV, ya que a partir de estos datos se definen todas las características de estilos de cada uno de los canales.

3.2.6. DIBUJO

El módulo de dibujo es uno de las partes más importantes de la aplicación, ya que se usa para el monitoreo principal de las señales mostradas. Este módulo usa las tramas de información obtenidas de la FIFO del módulo de transición y recepción para crear un dibujo de las señales muestreadas que el usuario selecciono desde el segundo fragmento.

En la Figura 14 se puede observar el proceso simplificado en un diagrama de flujo por el cual se realiza el dibujo de las tramas de la información obtenida en los paquetes trasmitidos por el sistema PQ_UAQ. El evento por el cual comienza este proceso es disparado por un botón el cual se encuentra en la barra de actividad de la aplicación, en esta igual se da la capacidad de pausar la gráfica.

Este evento es disparado sí y solo sí, la aplicación se encuentra realizando el proceso de adquisición de datos de la tarjeta FGPA y si el usuario se encuentra en la ventana de dibujo o de graficar. La impresión de los valores procesados se imprime al mismo tiempo que la gráfica realizada y es controlado por los mismos procesos del dibujo que la señales.

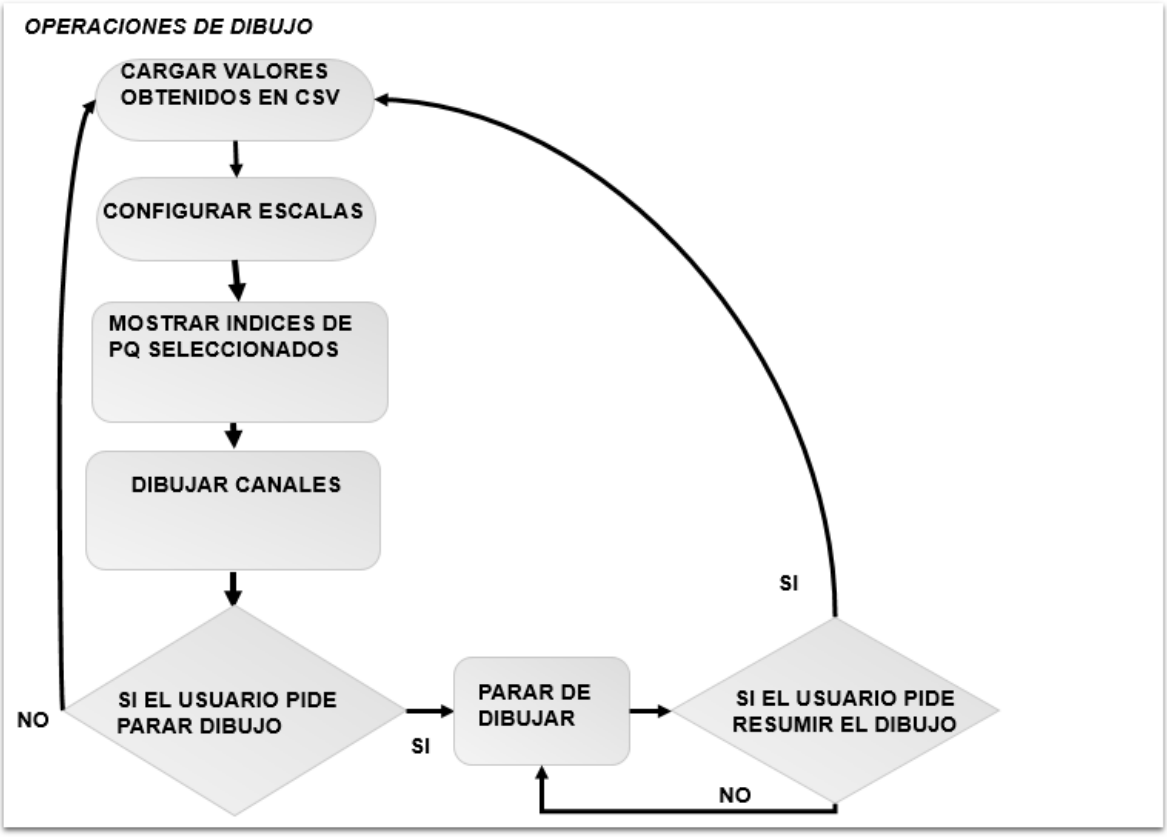


Figura 14. Proceso de petición y adquisición de datos para el dibujo

EL dibujo de las gráficas es un proceso que se tiene que hacer constantemente y tiene que ser ejecutado cada que la aplicación sea capaz de realizarlo, es por esto que se hace uso de dos hilos que controlan el funcionamiento y orden de los datos del módulo de dibujo.

El primer proceso el cual se encuentra dentro de un hilo, forma parte del handler y es un proceso que se realiza una vez obtenida la FIFO de la petición de datos en donde la información obtenida se divide en 8 buffers que son capaces de

almacenar 512 datos, estos buffers son declarados globales y levanta banderas cada vez que se llenan para almacenar datos en el siguiente buffer.

Consta agregar que de igual manera, por cada trama de datos se realiza las operaciones de procesamiento en las cuales se calculan los coeficientes mostrados en las gráficas. Esto se hace debido a que el proceso de graficación es un proceso realizado en el hilo de la UI el cual necesita de mucha memoria para realizarse y procesos como cálculos y operaciones básicas alentarían el procesamiento del dibujo provocando que algunas tramas adquiridas no sean dibujadas o se dibujen con una evidente desincronización, esto provocaría que el usuario no tenga una sensación de estar graficando en tiempo real.

El otro proceso necesario para ejecutar el módulo de dibujo es un hilo dedicado a iniciar la clase que se encarga de dibujar, el cual es creado en el evento OnCreate () de la aplicación y es ejecutado cada vez que el hilo principal de la aplicación despierta a este proceso. El Fragmento utilizado para el módulo de dibujo es una ventana simple el cual posee un SurfaceView que se encarga de proporcionar una superficie de dibujo editable. Debido a las características de un SurfaceView, esto nos permite modificar el dibujo realizado y manipular la gráfica dinámicamente para cambiar posiciones y formatos de la gráfica en tiempo real.

Es importante mencionar que esta aplicación va a ser usada en diferentes tipos de dispositivos, debido a esto se implementó una sección de código dedicada a sacar un factor de escala de los datos obtenidos de manera que la aplicación pueda ser usada en cualquier tipo de pantalla, no obstante las pequeñas diferencias eléctricas de la instrumentación de los dispositivos PQ_UAQ, igual pueden provocar

pequeñas diferencias en las mediciones, es importante tener en cuenta las diferentes características eléctricas de cada uno de los dispositivos, es por esta razón que se realizaron pruebas para obtener valores de ganancia y offset de las gráficas que se obtienen desde el archivo CSV y posteriormente son trasmitidas a un documento de texto para su interpretación y asignación.

El método Drawline el cual es usado para dibujar cada una de las líneas que conforman a la gráfica pertenece de igual manera dentro de la clase canvas y es usado junto con un ciclo para graficar y adaptar posición de toda la trama de información, en la Figura 15 se presenta la sección de código en donde se realiza esta operación, de igual manera canvas se usa para dibujar el texto por el cual se presentan los valores de procesamiento de las señales eléctricas tales como pico y RMS son mostradas por medio del método drawtext.

La clase Paint en este caso es usada en el hilo de dibujo para determinar las diferentes características de color y escala de las señales siendo graficadas, tanto las características de ganancia, color y formato pueden ser modificadas por medio del módulo de edición en la aplicación.

```
while (PlotIndex < PQ_UAQ_V7.VWidth) {
    ch1_dataf[PlotIndex] = HeightF -HeightF/2-PQ_UAQ_V7.ch_PFA1[PlotIndex]*FactorEH + 1;
    ch2_dataf[PlotIndex] = HeightF -HeightF/2-PQ_UAQ_V7.ch_PFA2[PlotIndex]*FactorEH + 1;
    ch3_dataf[PlotIndex] = HeightF -HeightF/2-PQ_UAQ_V7.ch_PFA3[PlotIndex]*FactorEH + 1;
    ch4_dataf[PlotIndex] = HeightF -HeightF/2-PQ_UAQ_V7.ch_PFAN[PlotIndex]*FactorEH + 1;

     ch5_dataf[PlotIndex] = HeightF -HeightF/2-ch_FA1[PlotIndex]*PQ_UAQFRAGMENT1.Tv.get(4).Ganancia + 1;//verde
    ch5_dataf[PlotIndex] = HeightF -HeightF/2-PQ_UAQ_V7.ch_PFV1[PlotIndex]*FactorEH/sqrtF  + 1;//verde
    ch6_dataf[PlotIndex] = HeightF -HeightF/2-PQ_UAQ_V7.ch_PFV2[PlotIndex]*FactorEH/sqrtF  + 1;//azul-voltaje1
    ch7_dataf[PlotIndex] = HeightF -HeightF/2-PQ_UAQ_V7.ch_PFV3[PlotIndex]*FactorEH/sqrtF  + 1;//violeta-voltaje2
    ch8_dataf[PlotIndex] = HeightF -HeightF/2-PQ_UAQ_V7.ch_PFAux[PlotIndex]*FactorEH + 1;//rosado-voltaje3
    PlotIndex++;
}

canvas.drawLine(1, HeightF-MaxValues[0]+1, PQ_UAQ_V7.VWidth, HeightF-MaxValues[0]+1,PQ_UAQ_V7.Iamax);//1
canvas.drawLine(1, HeightF-MaxValues[1]+1, PQ_UAQ_V7.VWidth, HeightF-MaxValues[1]+1,PQ_UAQ_V7.Ibmax);//1
canvas.drawLine(1, HeightF-MaxValues[2]+1, PQ_UAQ_V7.VWidth, HeightF-MaxValues[2]+1,PQ_UAQ_V7.Icmax);//1
canvas.drawLine(1, HeightF-MaxValues[3]+1, PQ_UAQ_V7.VWidth, HeightF-MaxValues[3]+1,PQ_UAQ_V7.Inmax);//1
canvas.drawLine(1, HeightF-MaxValues[4]+1, PQ_UAQ_V7.VWidth, HeightF-MaxValues[4]+1,PQ_UAQ_V7.Vamax);//1
canvas.drawLine(1, HeightF-MaxValues[5]+1, PQ_UAQ_V7.VWidth, HeightF-MaxValues[5]+1,PQ_UAQ_V7.Vbmax);//1
canvas.drawLine(1, HeightF-MaxValues[6]+1, PQ_UAQ_V7.VWidth, HeightF-MaxValues[6]+1,PQ_UAQ_V7.Vcmax);//1
```

Figura 15. Código de dibujo de SurfaceView en la aplicación

3.2.7. BITACORA

La bitácora es una de los principales módulos que se propusieron desde que se determinaron los objetivos de la aplicación, su función principal es almacenar datos de medición y anotaciones importantes sobre los eventos recibidos durante el monitoreo de una tarjeta, en la Figura 16 se muestra un diagrama en el que se observan los pasos necesarios para la creación de los archivos de texto, hay que mencionar que la generación de datos se realiza en diferentes fragmentos, específicamente los fragmentos de conexión y dibujo como se puede observar en la Figura 17 .

Para realizar una anotación y revisión de las mediciones se necesitan guardar todas las mediciones adquiridas para un futuro análisis o realizar una correlación con otras mediciones realizadas. Estas operaciones de análisis se

tienen que realizar sin importar si se cerró la aplicación y se volvió a abrir para realizar una revisión de la medición, ya que todas las variables se destruyen al cerrar la aplicación se necesitó de almacenar la información en un documento de con ayuda de un código CSV (Comma separated Values, Valores Separados por Comma) para garantizar el guardado de estas variables por largos periodos de tiempo.

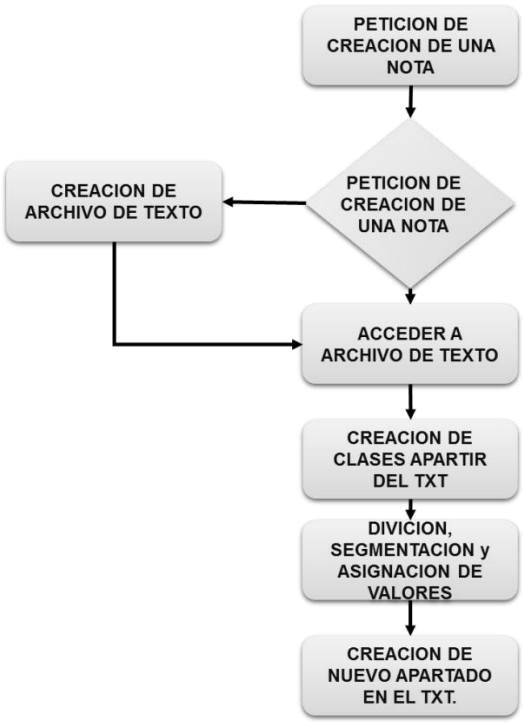


Figura 16. Proceso de generación de nuevos apartados en la bitácora

Los valores guardados obtenidos por las mediciones tienen que ser mostrados en las unidades correspondientes, las cuales se determinan en el archivo

CSV pquaq.csv para cada canal obtenido de las mediciones. Posteriormente estos datos guardados son asignados a su apartado correspondiente en listview en donde otros datos son igual mostrados, en esta aplicación de igual manera se proporciona un EditTextView, en el cual se pueden dejar anotaciones sobre eventos u observaciones durante el monitoreo de la tarjeta, se recomienda agregar eventos cortos repentinos y fallos de maquinaria al momento del monitoreo.

Todos los datos al momento de apretar guardar son almacenados en un archivo de texto. En la Figura 17 se muestra las diferentes variables que son necesarias para crear una sección nueva de la bitácora, las cuales se agregan al documento existente, una vez divididos y asignados, una sección nueva en él es creada en el documento de texto

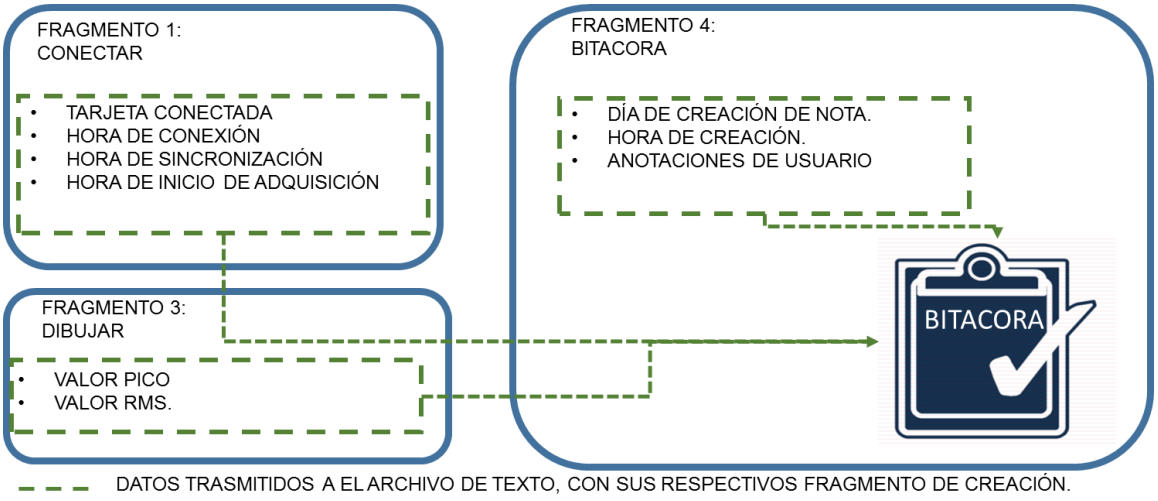


Figura 17. Asignación de variables para la creación de una bitácora

Otros archivos de texto igual, son creados para garantizar que se realicen los objetos correspondientes en la interfaz gráfica, en la Tabla 7 Se muestran los diferentes tipos de documentos de texto creados y su funcionalidad en la aplicación para el módulo de bitácora.

Tabla 7 Documentos de texto usados en la creación de bitácora

	Documento de texto	Funcionalidad
1	StoreLog.txt	Documento principal en donde se guardan todos los datos de la bitácora
2	ListNum.txt	Documento de texto usado para la creación de las listview en el momento que se acede a el evento onCreate () del fragmento.
3	Indice.txt	Trasmisión del índice de elementos creados para ayudar en la interfaz gráfica tanto del fragmento principal como la actividad de anotación.
4	IntentData.txt	Datos de mediciones necesarios al crear una nueva nota

En la Figuras 18 y 19 se muestran las interfaces graficas del fragmento 4 y la actividad de creación de una anotación.

Figura 18 GUI del cuarto fragmento

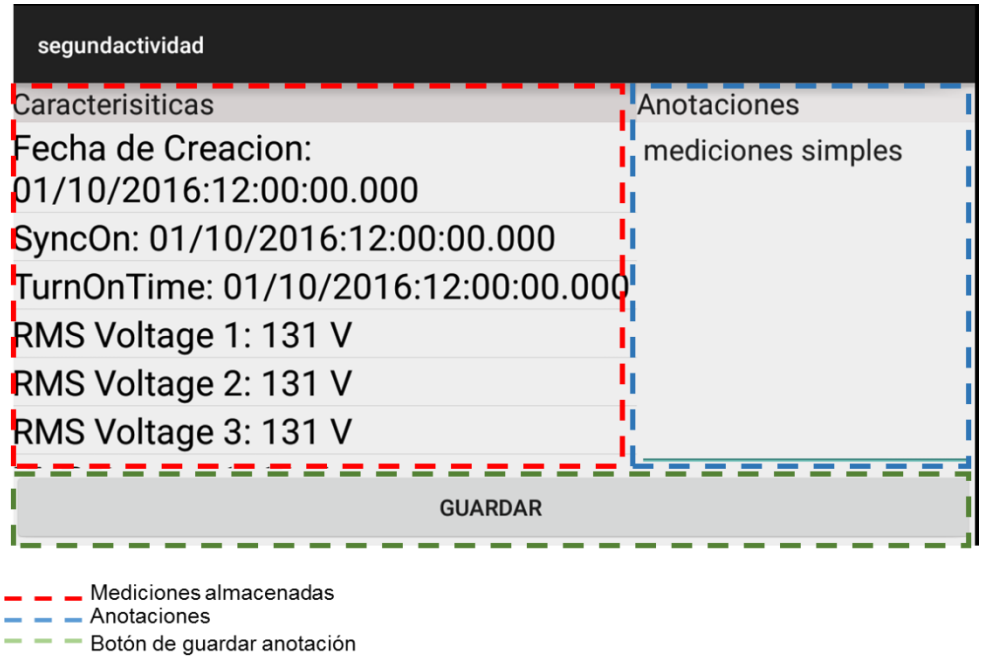


Figura 19. Intent de nota de la bitácora

IV. PRUEBAS Y RESULTADOS

En esta sección se presentan los resultados obtenidos con diferentes pruebas enfocados a determinar el buen funcionamiento de los elementos más importantes para la comunicación dentro de la red como: cada uno de los módulos implementados y la interfaz gráfica.

4.1. DISEÑO

Las pruebas se realizaron con el propósito de comprobar y validar el buen funcionamiento de la aplicación desarrollada, presentar resultados y observaciones de dichas pruebas, evaluar el desempeño de la aplicación en diferentes dispositivos móviles Android y cómo se comportó con otros dispositivos PQ_UAQ.

4.1.2. ENLACE INALAMBRICO

Es importante para lograr una trasferencia de datos entre el dispositivo de monitoreo y el sistema de adquisición, para verificar la conexión se utilizó una simple prueba de graficación usando la señal de referencia generada por la tarjeta de adquisición y se realizó una rutina de graficación, al recibir la gráfica con todos los picos correspondientes, se comprueba que la comunicación funciona, ya que la rutina de graficación envía varios mensajes al dispositivo de adquisición para que este conteste con los valores adquiridos, posteriormente las tramas adquiridas son graficadas y para reafirmar el funcionamiento de la aplicación se utilizaron varias tarjetas.

Por otra parte se comprobó el alcance de la aplicación y se observó retrasos en determinadas distancias menores a 10 metros, que es el rango efectivo promedio

de los dispositivos Bluetooth. Es probable que esta situación haya sido causada por la gran cantidad de procesamientos realizados, agregándose al retraso generado por la distancia e interferencias físicas. En la Figura 20 se muestra varias de las pruebas en diferentes distancias en donde se puede observar la señal de referencia y las señales eléctricas.

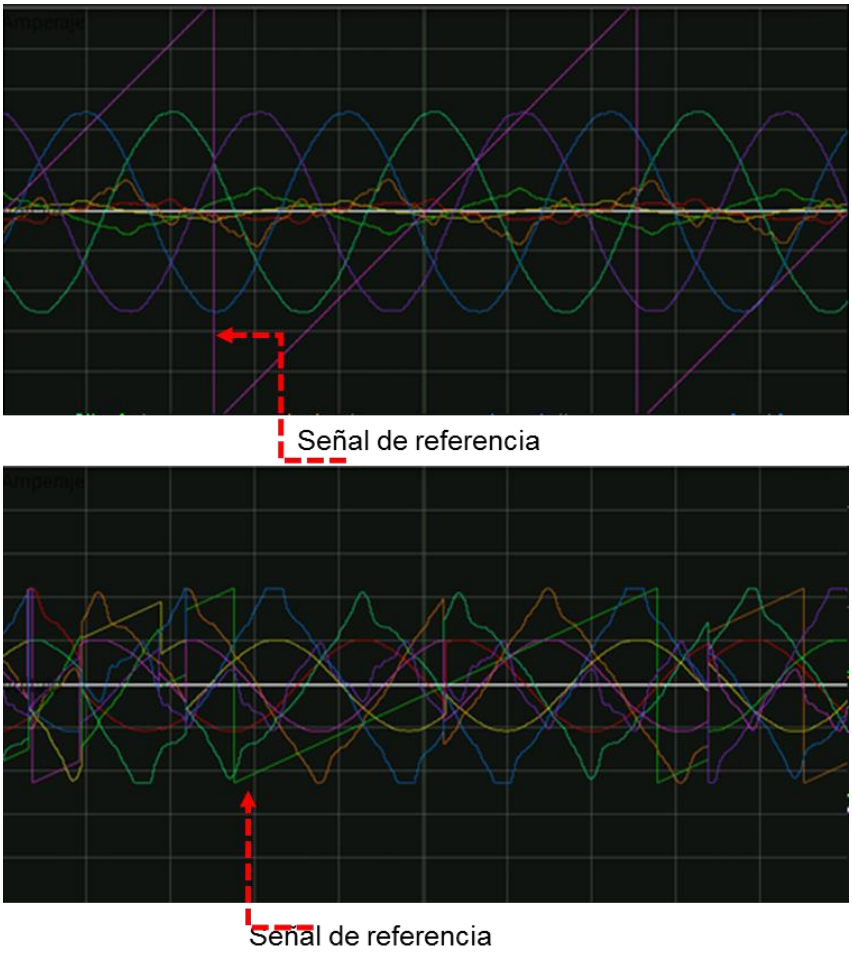


Figura 20. Cambios observados en la graficación a más de 7 metros.

4.2. INTERFAZ GRAFICA DE USUARIO

La GUI desarrollada es bastante sencilla y fácil de entender al momento de operarla, estéticamente se divide en 4 fragmentos como se muestran en las Figuras 21, 22, 23 y 24. El primer fragmento tiene la función de conexión, el segundo es usado para edición de las gráficas, el tercer para graficar la señal monitoreada y el cuarto fragmento se utiliza para generación de anotaciones de eventos importantes y mediciones en una bitácora, para usar en reportes de las mediciones o en comparaciones con otros dispositivos.

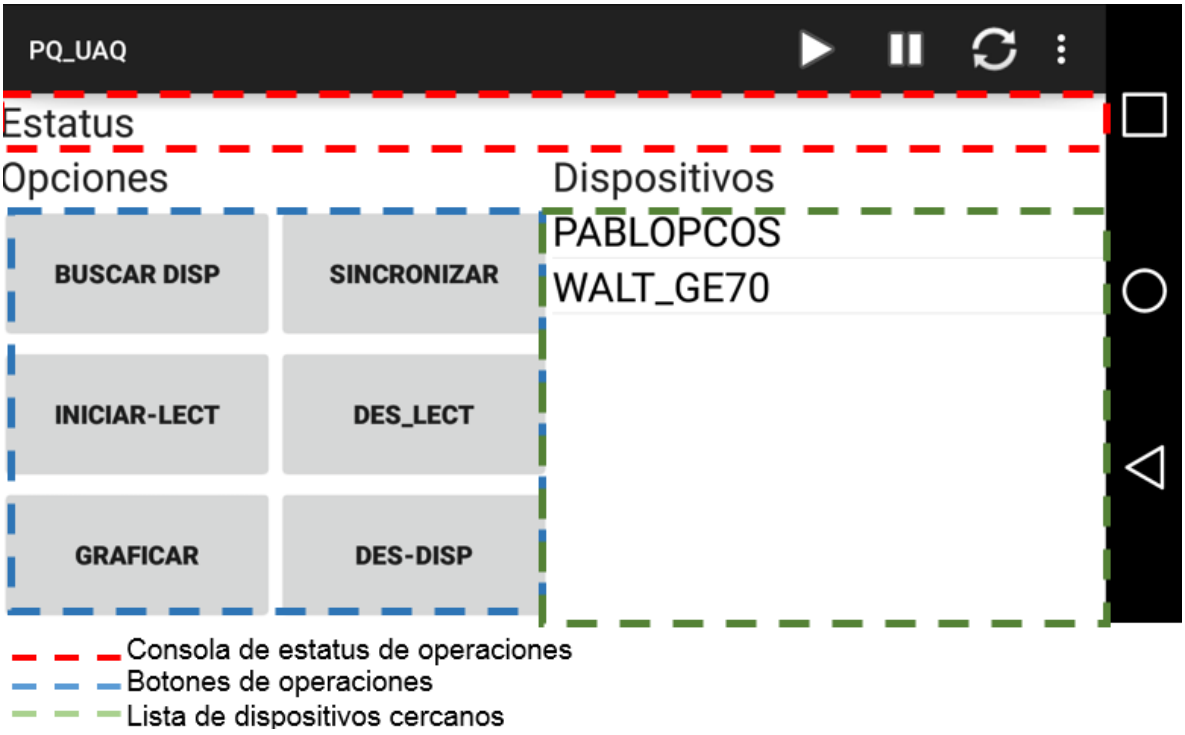


Figura 21 Interfaz gráfica del primer fragmento

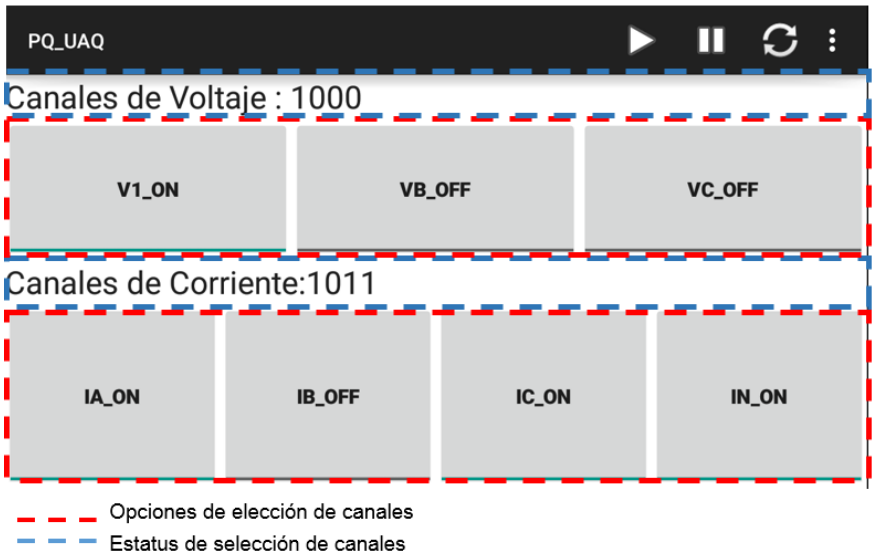


Figura 22. Segundo Fragmento de la aplicación

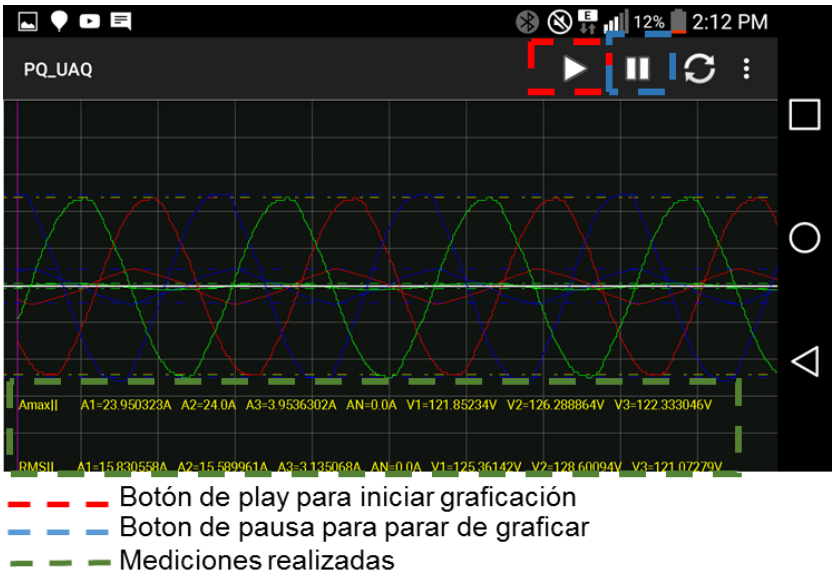


Figura 23. Tercer fragmento de la aplicación

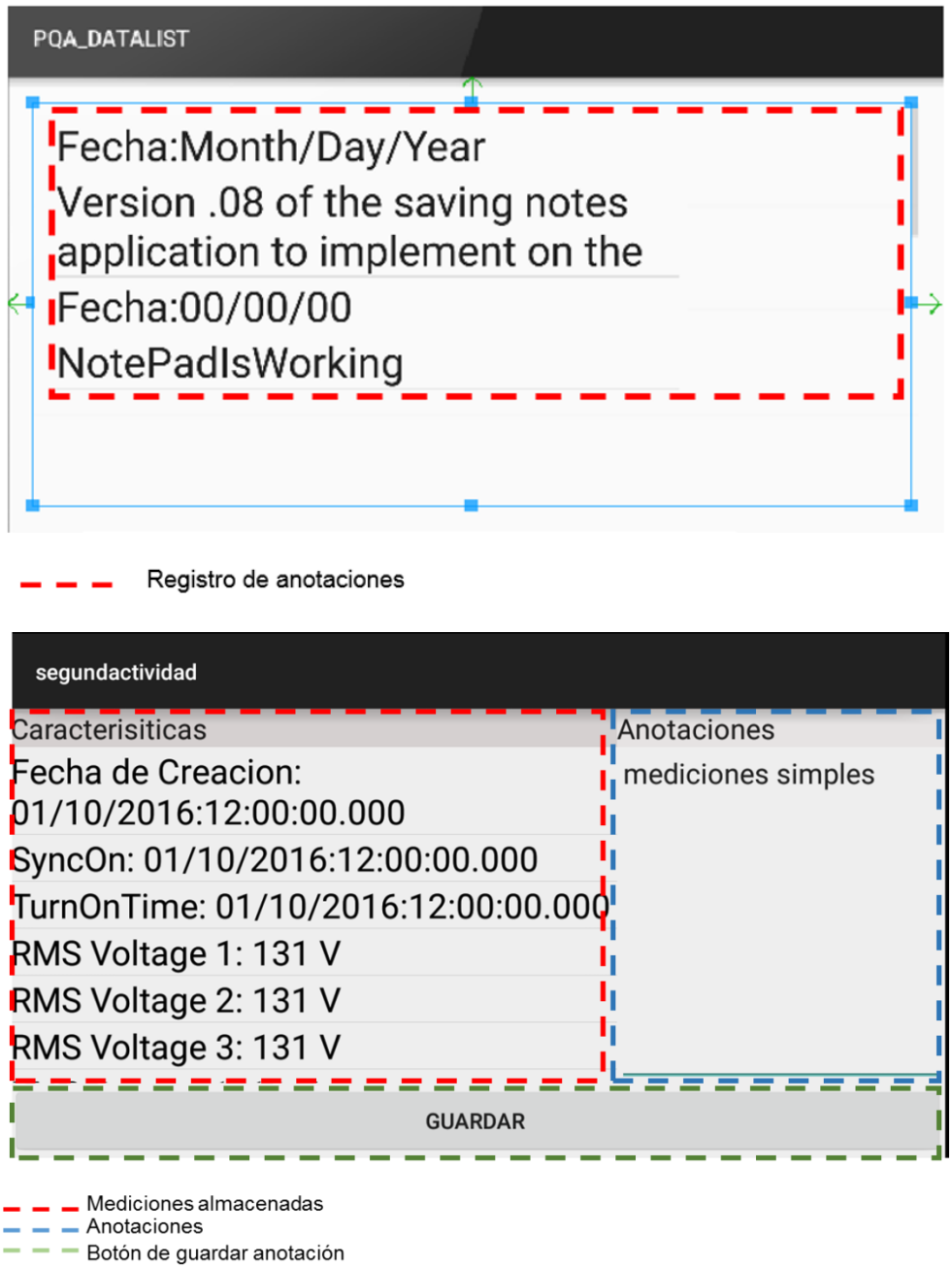


Figura 24. Cuarto fragmento de la aplicación

En cuanto a la usabilidad de la GUI, se cumplió con la mayoría de las heurísticas de diseño recomendadas.

La heurística número uno visibilidad de estatus del sistema se cumple con la implementación del texto de notificación y Test de eventos disparados en el handler de la aplicación de igual manera, al momento de graficar se puede comprobar de manera visual el estatus de la conexión.

La heurística número 2 se cumple ya que no existen botones con nombres extraños o símbolos ajenos, por los cuales no se entienda el funcionamiento. Esto es debido a la simplicidad de la GUI.

La heurística del número 3 y 5 quedan cubiertas, ya que es fácil que el usuario corrija sus acciones además de prevenir errores, si desea monitorear un elemento diferente simplemente deja de monitorear el elemento actual y comienzo a monitorear el siguiente dispositivo.

Hay que agregar que Android como sistema operativo en la mayoría de versiones ya tiene la capacidad en sí de evitar errores por medio de sus operaciones natales. A pesar de conectarse con diferentes dispositivos en diferentes tiempos todas las medidas y unidades siguen siendo las mismas, mientras que no se determine un diferente dispositivo con diferentes ganancias y funcionamiento, las unidades seguirán siendo las mismas, con esto se cumple la regla número 4 de la heurística.

Al usuario no se le permite introducir valores ilegales dentro de la anotación de la bitácora y el editor, cabe mencionar que los valores de estilo son definidos desde el documento CSV a el cual el usuario no tiene acceso desde la aplicación, cumpliendo de esta manera la regla de heurística núm. 5.

La regla número 6 es reconocer en vez de recordar, sí bien se tienen un gran número de pantallas disponibles y opciones a tomar cada uno de los fragmentos son individuales de los otros, claro aún se necesita conectar la tarjeta para realizar un monitoreo, pero si no es necesario se puede ir directamente a la graficación después de conectarse y no se tendrá realmente un mayor cambio en la funcionalidad de la tarjeta sin usar la mayoría de los botones disponibles debido a que se determinaron valores default en caso de que la tarjeta no se encuentre en el archivo CSV para las gráficas y las anotaciones se hacen de forma automática una vez se hace la petición.

Puede que el sistema tenga demasiadas ventanas, lo cual puede provocar un poco de confusión, sin embargo cada ventana es fácil de entender y por lo tanto fácil de aprender, esto es debido a una de las ventajas generales de usar el sistemas Android y poder emplear su dispositivo móvil por el hecho de que el usuario ya tiene un cierto grado de experiencia en manejar su celular con el cual puede experimentar la aplicación sin producir grandes cambios o fallas al sistema.

Una vez ya teniendo un grado de experiencia se espera que el usuario pueda hacer las operaciones más rápidamente, con esto se cumple la regla 7 de las heurísticas.

La regla 8 determina que se debe de tener un diseño estético y minimalista lo cual se logró al reducir la cantidad de funciones y reducir el número de elementos en los Fragmentos, el propósito de esto, es minimizar la cantidad de datos que el usuario ve a la vez.

La regla de la heurística número 9 se cumple desde el hecho que el dispositivo Android ya cuenta con avisos de errores desde el software.

4.3. PRUEBAS Y RESULTADOS DE FUNCIONAMIENTO

Las pruebas de funcionamiento fueron realizadas en la Universidad de Valladolid (UVA), en el hospital Universitario Rio Hortega y en la Facultad de Ingeniería Campus San Juan del Río de la Universidad Autónoma de Querétaro (UAQ), Estas dos primeras son instituciones situadas en Valladolid, España y la última en San Juan del Río, Querétaro en México. En el hospital se realizaron instalaciones y monitoreo semanales, mientras que en la UVA se hicieron pruebas de funcionamiento y graficación de la red eléctrica al tener motores eléctricos, en la UAQ se realizaron pruebas de graficación y elección de canales. La instalación básica del sistema se presenta en la Figura 25, en la que se observan los diagramas de instalación general de los sistemas de monitoreo.

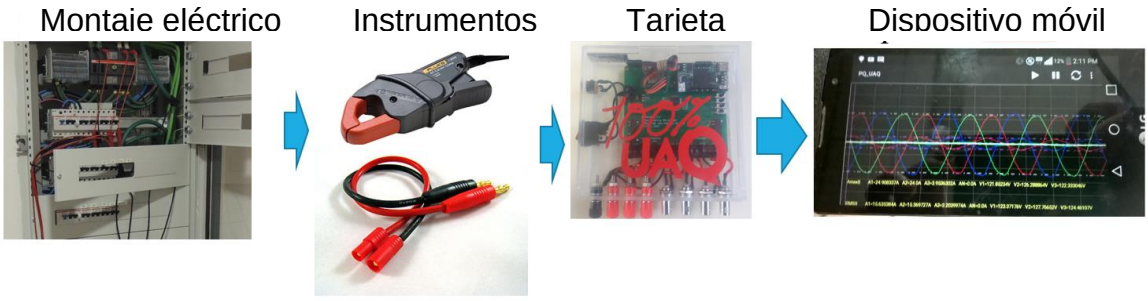


Figura 25. Instalación general del sistema de monitoreo

4.3.1 INSTALACION EN HOSPITALES

Para las pruebas realizadas en la instalación en el hospital Río Hortega, las operaciones más críticas eran el muestreo de la instalación eléctrica y la activación y sincronización de los dispositivos a monitorear. Los dispositivos se cambiaron de

montaje eléctrico semanalmente y se supervisaba cada dos días que las tarjetas funcionaran, para detectar algún problema en los dispositivos, teniendo así un monitoreo constante.

En los muestreos realizados, el uso de una aplicación ayudo en gran parte a la movilidad y rapidez para realizar las medidas necesarias y comprobar que la tarjeta seguía funcionando. En las Figuras 26 y 27 se presenta una imagen de uno de los pisos y montajes en donde se instalaron dispositivos PQ_UAQ, mientras que en la Figura 28 se observan algunas graficas obtenidas durante los muestreos realizados.

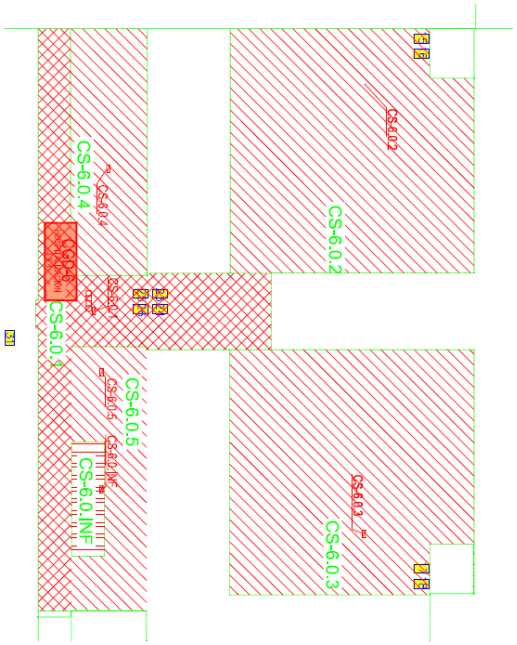
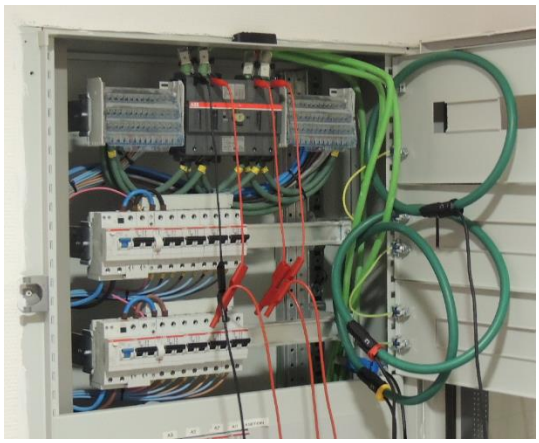
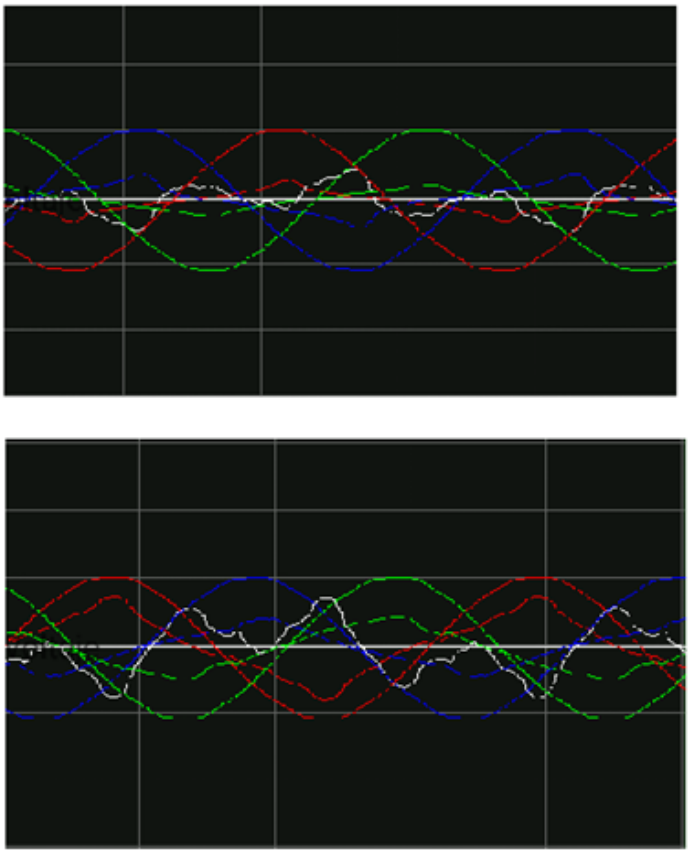


Figura 26. Planos de uno de los pisos en el hospital universitario



Figuran 27 montajes e instalaciones de tarjetas



Figuran 28 Graficas obtenidas

4.3.2 INSTALACION EN LA UVA

Las pruebas realizadas en la UVA consistieron en realizar sincronizaciones, iniciar lecturas y graficas básicas de las señales eléctricas, de esta manera se pudo ayudar a la calibración mostrando el estado de la red. En la Figura 29 se muestra una sincronización e inicio de lectura realizadas y en la Figura 30 se muestran graficas que se tomaron en tales pruebas, es igualmente importante mencionar que las pruebas de distancias efectivas fueron realizadas durante estas actividades.

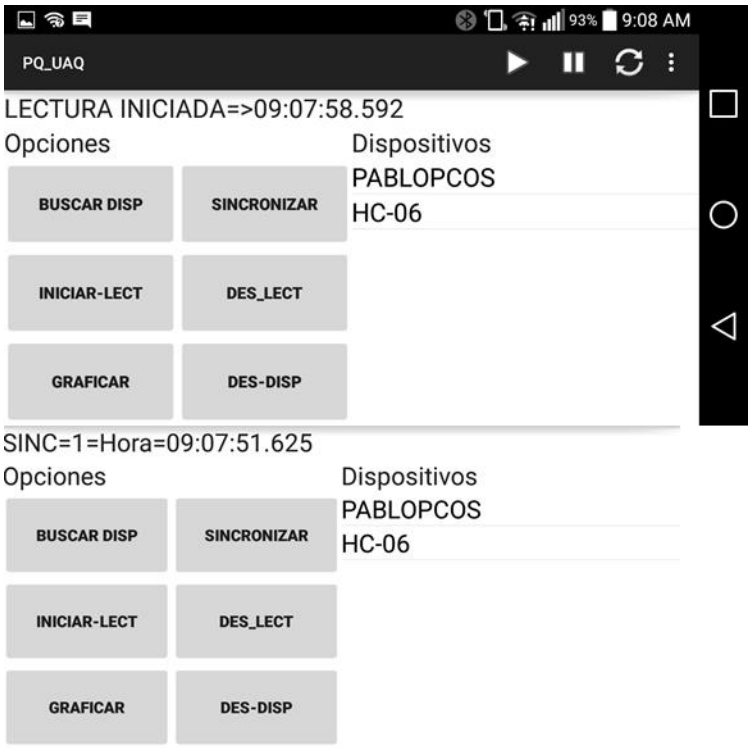


Figura 29. Sincronización e Inicio de lectura en el primer fragmento

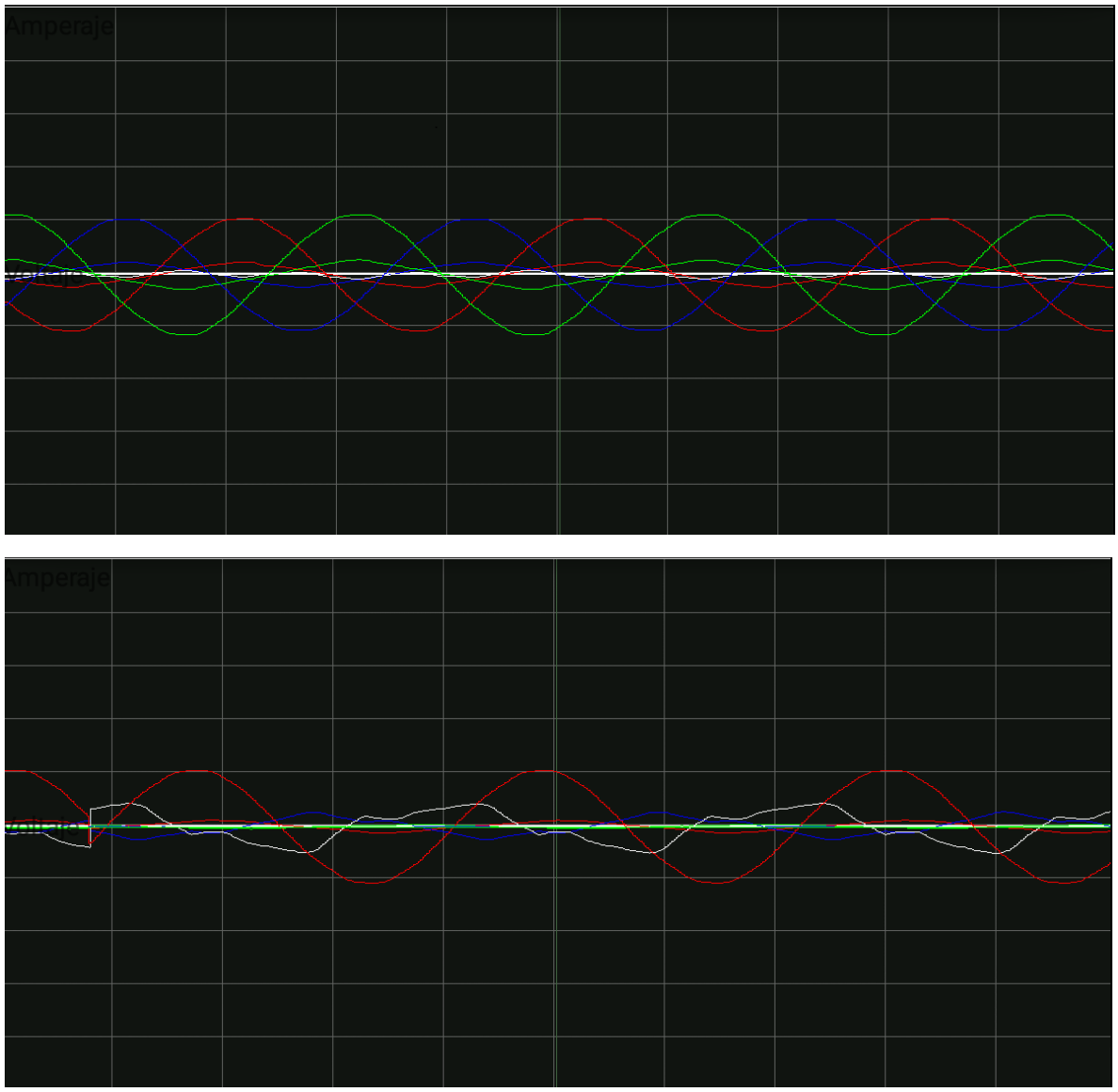


Figura 30. Lecturas tomadas durante la calibración de tarjetas

4.3.3 INSTALACION EN CAMPUS SAN JUAN DEL RIO DE LA UAQ

En la UAQ Campus San Juan del Río se verifico que las medidas obtenidas de los cálculos de los coeficientes estén correctamente realizados y que se encuentren dando mediciones adecuadas, igualmente se hicieron pruebas del funcionamiento de los fragmentos y la capacidad que el usuario tienen al elegir los canales y mostrar los valores pico-pico, así también se realizaron lecturas y se verificaron las mediciones. En la Figura 31 se muestra la instalación de las tarjetas y en la Figura 32 se muestran mediciones, mientras que en la Figura 33 se muestra la elección de canales.

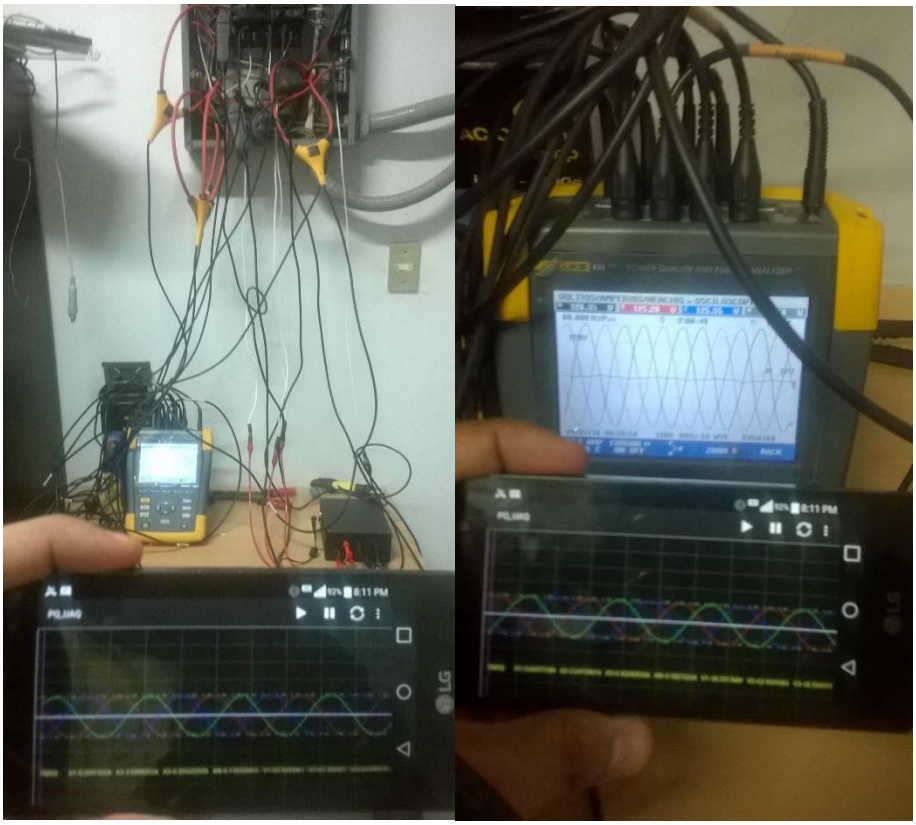


Figura 31. Instalación en el Campus San Juan del Río

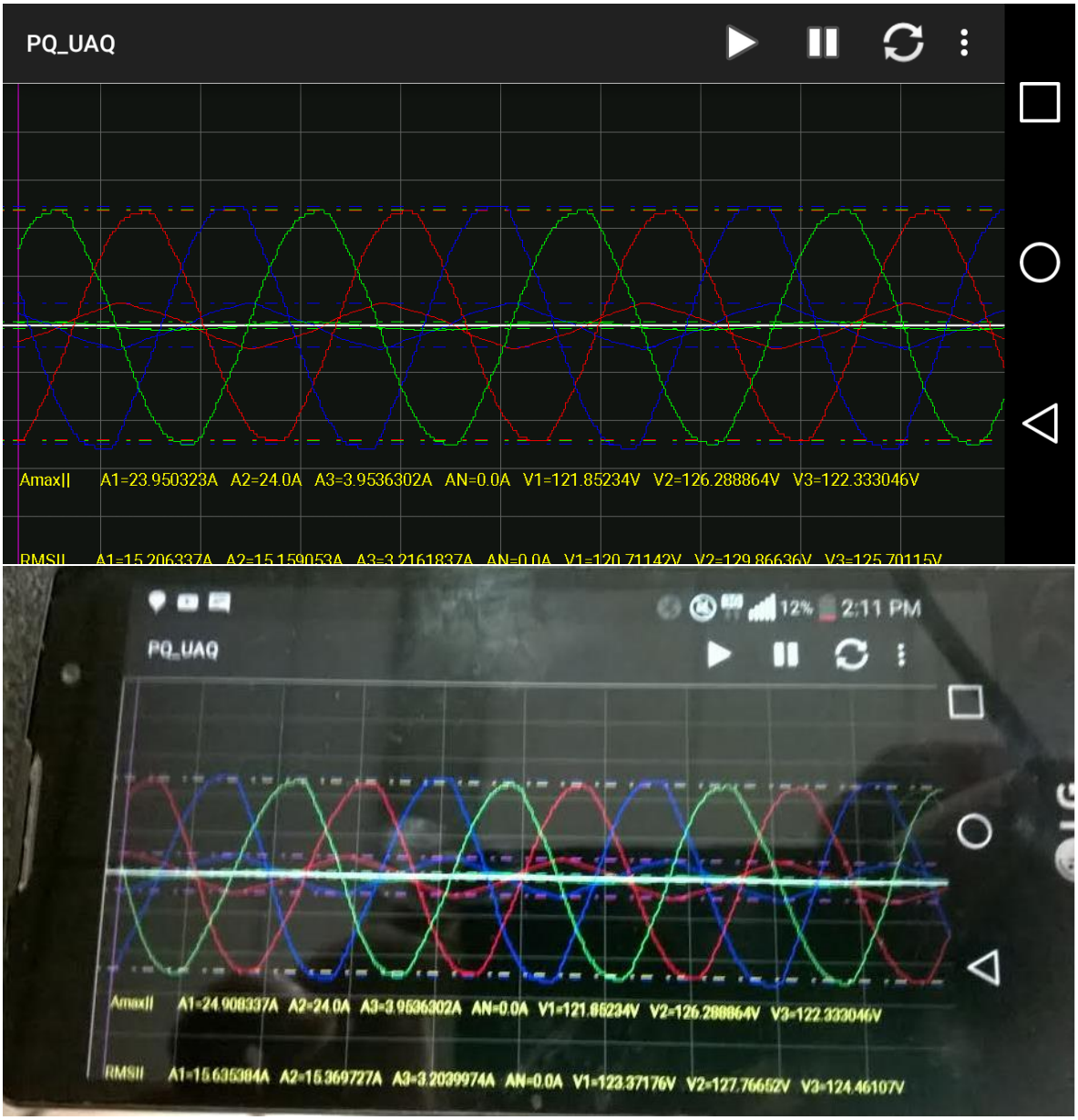


Figura 32. Mediciones realizadas

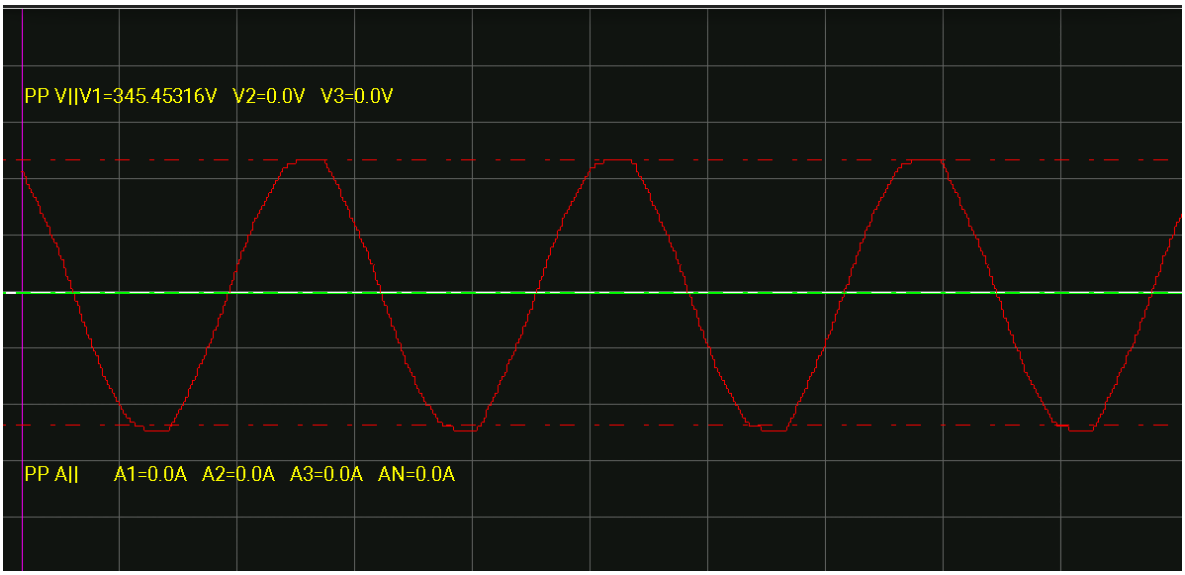


Figura 33. Selección de canales

Por medio de las pruebas realizadas es posible llevar a cabo un análisis de funcionalidad. En el momento de la instalación las tres señales trifásicas podían ser observables de manera sencilla y entendible. Valores como RMS y Pico fueron exitosamente calculados y mostrados en la pantalla en tiempo real, concordando con los tomados por otros dispositivos.

Cada una de las señales de corriente y voltaje podrían ser fácilmente observables, al igual que el lag de la corriente con respecto al voltaje. La manera en como las gráficas están presentadas es una configuración realizada en el editor de graficas por el usuario.

Es importante mencionar que en la actualidad se han creado otras aplicaciones capaces de conectarse con dispositivos PQ, sin embargo estas aplicaciones no cuentan con la flexibilidad de conectividad con otros sistemas de monitoreo y presentan problemas de compatibilidad con la gran mayoría de

dispositivos móviles, ya que fueron creadas usando librerías en las que se necesitan altos niveles de la versión de Android.

V. CONCLUSIONES

El presente trabajo de tesis muestra que es posible desarrollar una aplicación enfocada a el monitoreo de un dispositivo de adquisición de datos o para ser usado ya sea en ambientes industriales o de laboratorio.

El hecho de ser un dispositivo inalámbrico provee que el ruido presentado en la tarjeta, sea trasmitido a la aplicación y por lo tanto se tienen un monitoreo más limpio que sí se hiciera de manera alámbrica. Esto le da mejores prestaciones con respecto a la inmunidad al ruido.

El dispositivo fue creado utilizando conocimiento de trabajos anteriores y en conjunto a trabajos actuales de desarrollo tecnológico por la UAQ, además se desarrolló con plataformas de diseño libre, lo cual dota a esté trabajo de versatilidad de uso, ya que cualquier persona que cuente con un dispositivo Android es capaz de usarlo.

En mención a los resultados obtenidos, se concluye que son satisfactorios, ya que se cumplieron con los objetivos planteados y se validó el trabajo en instituciones externas a la Universidad Autónoma de Querétaro.

También cabe mencionar que es posible realizar algunas mejoras en puntos específicos del trabajo como es: en los módulos de comunicación de bluetooth, ya que sí se usara bluetooth 4.0, se tendrá una comunicación con mayor distancia y más ahorro de energía, así como un procesamiento mejor de la aplicación.

En relación con la publicación del congreso realizada, resulto satisfactorio el compartir conocimientos de la investigación y dar a conocer el trabajo de la universidad en un congreso de carácter internacional.

El trabajo actual destaca la importancia de una área de investigación que últimamente se está haciendo más popular, el desarrollo de software en dispositivos móviles, ya que últimamente el mundo se está volviendo más interconectado, por lo que es necesario desarrollar más investigaciones en las cuales se utilicen nuevas aplicaciones móviles y desarrollar algoritmos para realizar operaciones que actualmente resultan muy caras o necesitan de sistemas muy aparatosos para instalar.

El contar con una aplicación puede mejorar la eficiencia e incluso remplazar algunos procesos, por lo que se sugiere realizar más investigación que involucren el uso de dispositivos móviles.

VI. REFERENCIAS

Annuzzi, J., Darcey, L., & Conder, S. (2010). *Advanced Android Aplication Development*. Michigan, USA: Addison-Wesley.

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

Chen, C., & Mukhopadhyay, S. C. (2015). Efficient Coverage and Connectivity Preservation With Load Balance for Wireless Sensor Networks, *15*(1), 48–62.

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

IEEE (2009). *IEEE Std 1159™-2009, IEEE Recommended Practice for Monitoring Electric Power Quality* (Vol. 2009).

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 and Energy Systems*, *47*, 368–377. http://doi.org/10.1016/j.ijepes.2012.11.005

Deokar, S. a., & Waghmare, L. M. (2014). Integrated DWT–FFT approach for detection and classification of power quality disturbances. *International Journal of Electrical Power & Energy Systems*, *61*, 594–605. http://doi.org/10.1016/j.ijepes.2014.04.015

Erişti, H., Yıldırım, Ö., Erişti, B., & Demir, Y. (2014). Automatic recognition system of underlying causes of power quality disturbances based on S-Transform and Extreme Learning Machine. *International Journal of Electrical Power & Energy Systems*, *61*, 553–562. http://doi.org/10.1016/j.ijepes.2014.04.010

Evans, M. J., Clemens, G., Casey, C., & Baker, M. J. (2014). Developing a mobile app for remote access to and data analysis of spectra. *Vibrational Spectroscopy*, *72*, 37–43. http://doi.org/10.1016/j.vibspec.2014.02.008

Ferreira, D. D., De Seixas, J. M., Cerqueira, A. S., & Duque, C. a. (2013). Exploiting principal curves for power quality monitoring. *Electric Power Systems Research*, *100*, 1–6. http://doi.org/10.1016/j.epsr.2013.02.006

Funk, J. L. (2006). The future of mobile phone-based Intranet applications: A view from Japan. *Technovation*, *26*, 1337–1346. http://doi.org/10.1016/j.technovation.2005.08.009

Funk, J. L. (2009). The emerging value network in the mobile phone industry: The case of Japan and its implications for the rest of the world. *Telecommunications Policy*, *33*, 4–18. http://doi.org/10.1016/j.telpol.2008.09.002

García, E. G., & Velázquez, L. M. (2014). *Implementación de protocolo Modbus RS-485 en sistema embebido basado en FPGA para red de sensores inerciales.* Universidad Autónoma de Querétaro.

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 De La Rosa, J. J., Sierra-Fernández, J. M., Agüera-Pérez, A., Palomares-Salas, J. C., & Moreno-Muñoz, A. (2013). An application of the spectral kurtosis to characterize power quality events. *International Journal of Electrical Power and Energy Systems*, *49*, 386–398. http://doi.org/10.1016/j.ijepes.2013.02.002

Graditi, G., Luisa, M., Silvestre, D., Gallea, R., & Sanseverino, E. R. (2014). Heuristic-based shiftable loads optimal management in smart micro-grids. *IEEE Transactions on Industrial Informatics*, *3203*(1), 1–1. http://doi.org/10.1109/TII.2014.2331000

Granados, L. D. (2013). *Análisis en maquinaria CNC ante variaciones de bajo voltaje y sus efectos de calidad de la energía*. *UAQ*.

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

Guesmi, H., Ben Salem, S., & Bacha, K. (2014). Smart wireless sensor networks for online faults diagnosis in induction machine. *Computers & Electrical Engineering*. http://doi.org/10.1016/j.compeleceng.2014.10.015

Heaton, J., & ebrary, Inc. (2002). *Programming Spiders, Bots, and Aggregators in Java*.

Humphreys, I., Eisenblätter, G., & O'Donnell, G. E. (2014). FPGA based monitoring platform for

condition monitoring in cylindrical grinding. *Procedia CIRP*, *14*, 448–453. http://doi.org/10.1016/j.procir.2014.03.022

Ilarri, S., Stojanovic, D., & Ray, C. (2015). Semantic management of moving objects: A vision towards smart mobility. *Expert Systems with Applications*, *42*(3), 1418–1435. http://doi.org/10.1016/j.eswa.2014.08.057

callejas, I.,Piñeros, J.,Rocha, J., Ferney, H., & 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

Junfeng, H., Hao, S., & Xiaolin, W. (2012). Design of Power Quality Monitor Based on Embedded Industrial Computer. *Physics Procedia*, *24*, 63–69. http://doi.org/10.1016/j.phpro.2012.02.011

Kaart, M., & Laraghy, S. (2014). Android forensics: Interpretation of timestamps. *Digital Investigation*, *11*(3), 234–248. http://doi.org/10.1016/j.diin.2014.05.001

Mazlumi, K (2011). Power Quality Monitoring, Power Quality â€“Â Monitoring, Analysis and Enhancement, Dr. Ahmed Zobaa (Ed.), ISBN: 978-953-307-330-9, InTech, Available from: http: //www.intechopen.com/books/power-quality-monitoring-analysis-and-enhancement/power-qualitymonitoringKhan, S. R. (2014). GUI Based Industrial Monitoring and Control System, (Pestse), 0–3.

Khan, T. H., Shrestha, R., Wahid, K. a., & Babyn, P. (2015). Design of a smart-device and FPGA based wireless capsule endoscopic system. *Sensors and Actuators A: Physical*, *221*, 77–87. http://doi.org/10.1016/j.sna.2014.10.033

Meher, S. K., Pradhan, a. K., & Panda, G. (2004). An integrated data compression scheme for power quality events using spline wavelet and neural network. *Electric Power Systems Research*, *69*, 213–220. http://doi.org/10.1016/j.epsr.2003.10.001

Meier, R. (2010). *Android Application Development*. *Framework* (Vol. 131). http://doi.org/9781118102275

Miller, C., & Poellabauer, C. (2014). Configurable Integrated Monitoring System for Mobile Devices. *Procedia Computer Science*, *34*, 410–417. http://doi.org/10.1016/j.procs.2014.07.046

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

Naik, P. P., Mishra, G. K., Danielsson, B., & Bhand, S. (2015). Android integrated urea biosensor for public health awareness. *Sensing and Bio-Sensing Research*, *3*, 12–17. http://doi.org/10.1016/j.sbsr.2014.11.001

Panteli, M., & Kirschen, D. S. (2015). Situation awareness in power systems: Theory, challenges and applications. *Electric Power Systems Research*, *122*, 140–151. http://doi.org/10.1016/j.epsr.2015.01.008

Pon, B., Seppälä, T., & Kenney, M. (2014). Android and the demise of operating system-based power: Firm strategy and platform control in the post-PC world. *Telecommunications Policy*, *38*(11), 979–991. http://doi.org/10.1016/j.telpol.2014.05.001

Romero-Troncoso, R. J., Cabal-Yepez, E., Garcia-Perez, A., Osornio-Rios, R. a., Alvarez-Salas, R., & Granados-Lieberman, D. (2011). Reconfigurable instrument for power quality monitoring in 3-phase power systems. *SDEMPED 2011 - 8th IEEE Symposium on Diagnostics for Electrical Machines, Power Electronics and Drives*, (October 2012), 186–191. http://doi.org/10.1109/DEMPED.2011.6063622

Salmon, P., Stanton, N., Walker, G., & Green, D. (2006). Situation awareness measurement: A review of applicability for C4i environments. *Applied Ergonomics*, *37*, 225–238. http://doi.org/10.1016/j.apergo.2005.02.001

Sandanayake, D. S. S., Topal, E., & Ali Asad, M. W. (2015). A heuristic approach to optimal design of an underground mine stope layout. *Applied Soft Computing*, *30*, 595–603. http://doi.org/10.1016/j.asoc.2015.01.060

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

Sheng, H., Nah, F. F.-H., & Siau, K. (2005). Strategic implications of mobile technology: A case study using Value-Focused Thinking. *The Journal of Strategic Information Systems*, *14*, 269–290. http://doi.org/10.1016/j.jsis.2005.07.004

Smith, G. R. (2010). *FPGA 101 everything you need to know to get started*. Massachusetts, USA: Elsevier.

The Android Open Source Project (2016). The Android Open Source. Retrieved from http://developer.android.com/intl/es/reference/android/graphics/Canvas.html.

The Android Open Source Project (2016). The Android Open Source. Retrieved from http://developer.android.com/intl/es/reference/java/lang/System.html.

The Android Open Source Project (2016). The Android Open Source. Retrieved from https://developer.android.com/reference/android/os/Handler.html.

The Android Open Source Project (2016). The Android Open Source. Retrieved from

https://developer.android.com/reference/java/lang/Thread.html

The Android Open Source Project (2016). The Android Open Source. Retrieved from https://developer.android.com/reference/android/view/SurfaceView.html

Valentina, R., Ruppert, M., Nenninger, P., & Mendoza, F. (2013). Human machine interface for Virtual Prototyping of Industrial Instruments. *IEEE International Conference on Industrial Informatics (INDIN)*, 270–275. http://doi.org/10.1109/INDIN.2013.6622894

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

Bluetooth Special Interest Grou (2004). Specification of the Bluetooth System Master Table of Contents & Compliance.

Wang, M.-H., & Tseng, Y.-F. (2011). A novel analytic method of power quality using extension genetic algorithm and wavelet transform. *Expert Systems with Applications*, *38*(10), 12491–12496. http://doi.org/10.1016/j.eswa.2011.04.032

Widhiasi, A., Idrus, R., Pasha, M. F., & Syukur, M. (2013). A feasibility study scheme of an android-based integrated wearable ECG monitoring system. *Procedia Computer Science*, *21*, 407–414. http://doi.org/10.1016/j.procs.2013.09.053

Xilinx, D., & Pasos, S. (2012). FPGAs a Fondo, 1–5.

Zhao, Y., Bond, I. a., & Sweatman, W. L. (2014). An Android application for receiving notifications of astrophysical transient events. *Astronomy and Computing*, *6*, 19–27. http://doi.org/10.1016/j.ascom.2014.05.001

VII. APENDICE

7.1 CODIGOS

7.1.1 PQ_UAQ_V7

```
import android.annotation.SuppressLint;
import android.app.Activity;
import android.bluetooth.BluetoothAdapter;
import android.bluetooth.BluetoothDevice;
import android.bluetooth.BluetoothSocket;
import android.content.BroadcastReceiver;
import android.content.Context;
import android.content.Intent;
import android.content.IntentFilter;
import android.graphics.Color;
import android.graphics.DashPathEffect;
import android.graphics.Paint;
import android.os.Bundle;
import android.os.Handler;
import android.os.Message;
import android.os.PowerManager;
import android.support.v4.view.ViewPager;
import android.support.v7.app.AppCompatActivity;
```

```
import android.util.Log;
import android.view.Menu;
import android.view.MenuItem;
import android.view.SurfaceHolder;
import android.view.ViewTreeObserver;
import android.widget.ArrayAdapter;
import android.widget.LinearLayout;
import android.widget.ListAdapter;
import android.widget.ListView;
import android.widget.TextView;
import android.widget.Toast;
import java.io.IOException;
import java.text.SimpleDateFormat;
import java.util.ArrayList;
import java.util.Set;
import java.util.UUID;

@SuppressLint("HandlerLeak")
public class PQ_UAQ_V7 extends AppCompatActivity {

    ////////////////////WaveformView;
    public static volatile float[] ch_FV1;
    public static volatile float[] ch_FV2;
    public static volatile float[] ch_FV3;
    public static volatile float[] ch_FV4;
```

```
public static volatile float[] ch_FA1;
public static volatile float[] ch_FA2;
public static volatile float[] ch_FA3;
public static volatile float[] ch_FA4;


public static volatile String MText="0";
public static volatile String RMSTxt="0";
public static volatile String MVTEXT="0";
public static volatile String RMSVTEXT="0";

public static volatile  Paint V1_color = new Paint();
public static volatile Paint V2_color = new Paint();
public static volatile Paint V3_color = new Paint();
public static volatile  Paint V4_color = new Paint();
public static volatile Paint A1_color = new Paint();
public static volatile Paint A2_color = new Paint();
public static volatile Paint A3_color = new Paint();
public static volatile Paint A4_color = new Paint();
public static volatile Paint TxtColor = new Paint();
public static volatile int  BackGroundCol;
public static volatile Paint ChColor[];
public static volatile ArrayList<Paint> AChPaint=new ArrayList<Paint>();
```

```
public static   ArrayList<DataSetPreset_T1> SaveLog = new ArrayList<DataSetPreset_T1>();

public double sqrtD=Math.sqrt(2);
public float sqrtF=(float)sqrtD;
public float GainHelper = 256f;
////////////////////////////////

public static boolean F3TxtFlag = false;
public static boolean TrashFlag = true;
public static boolean F3FlagQkSwData=false;
public static boolean FlagBuscarDispositivo=false;
public static boolean FlagPrintSync=true;
public boolean RunSyncBool = false;

public static final String TAG = "TAG";

public static volatile float[] ch_PFA1;
public static volatile float[] ch_PFA2;
public static volatile float[] ch_PFA3;
public static volatile float[] ch_PFAN;

public static volatile float[] ch_PFV1;
public static volatile float[] ch_PFV2;
public static volatile float[] ch_PFV3;
public static volatile float[] ch_PFAux;
```

```
public static String Sch_PFA1;
public static String Sch_PFA2;
public static String Sch_PFA3;
public static String Sch_PFAN;
public static String Sch_PFV1;
public static String Sch_PFV2;
public static String Sch_PFV3;


public static int IVChOp;
public static int IAChOp;
public static int ITxtOp1=01;
public static int BackOp;
public static int EscaleOP;

public static long milistime;

public String buffSt;

public int BuffIndice;

public int  n,j;
public int ms,dms,fms;
public long fmsL,dmsL,msL;
```

```
public char bufSyncR[];
public char Syncstr[];
public char Syncbuf[];
public char Syncmsg[];

public static String DEVICESAV;

public static String DEVICESAVE="PQ_UAQ";
public static String CREATESAVE="2016";
public static String MACSAVE;
public static String SyncSAVE;
public static String TurnOnSAVE;
public static String[] RMSSAVE=new String[8];;
public static String[] MAXSAVE=new String[8];;

// Run/Pause status
private boolean bReady = false;

private boolean seleccionador=false;

public static final boolean D = false;
public static final int Mensaje_Leido = 2;
public static final int Mensaje_Escrito = 3;
public static final int Mensaje_Nombre_Dispositivo = 4;
public static final int Mensaje_TOAST = 5;
```

```
public static final int MESSAGE_Desconectado = 6;
public static final int REQUEST_ENABLE_BT = 7;
public static final String DEVICE_NAME = "HC-06_1234";//nombre del dispositivo_codigo
public static final String TOAST = "toast";
private String mConnectedDeviceName = null;
private BluetoothAdapter AdaptadorBT = null;
private Set<BluetoothDevice> pairedDevices;
public int Opcion=R.menu.menu_pq__uaq__v7;
public PowerManager.WakeLock EnergyLock;

public static String Day,Month,Year,Hour,Minute,Second,milisecond;

public WaveformView mWaveform = null;

//clase conexion bt
private ConexionBT Servicio_BT=null;
private Test_Class ServicioTest=null;
private BluetoothSocket Btsocket=null;

private static final int  MAX_LEVEL     = 600;

public int FifoCol=0;
public int FifoRow=0;

public  boolean DoWhileFlag;
```

```
public  boolean DoWhileFlag2;

public int SendTrig=0;
public int FlagBuff=0;
public int FlagFIx =0;

public static String AuxGlobalString;
public static Byte AuxGlobalbyte;


public static volatile int  VWidth =0;
public static volatile int  VHeight=0;

///NewFlags
public int Stt=0;
public int Flag_Z=0;
public int Flag_A =0;
public int Cot =0;
public int ClkFlag =0;
public int Timer =0;
public int Timer2 =0;
///
//Fragment flags
public int F1_Flag=0;
public static boolean SyncFlag=false;
```

```
///val for the other phone 790
    //private final int width = 1165;
    private final int heigth = 1165;
    private final int WidthRes=0;
    //int widthPlus=width+850;
    public SurfaceHolder holder;
    ////////////////////////////////////////////////
    //public WaveformView mWaveform = null;
    private MyPagerAdapter pageAdapter;

    public static int[][] DrawBuff=new int[8][512];
    public static int[][] GlobalBuff=new int[8][512];
    public static int[][] CeroBuff=new int[8][512];

    public static float[] IndiMaxVal=new float[10];
    public static float[] IndiRmsVal=new float[10];

    public static volatile byte[]SyncFpgaTime=new byte[6];

    public static volatile int[][][] OpcionesBufferS;
    public static volatile String BtDispNames[][];
    public static volatile int NumDisp=0;

    public static volatile int OpcionesBuff1,OpcionesBuff2,OpcionesBuff3,OpcionesBuff4;
    public TextView TxtOpciones,TxtPreset,TxtF3QuickDataShow;
```

```
private TextView F1TxtOpciones;

public static volatile int oneFlag=0;
public static volatile int DrawFlag=0;
public static volatile int F3_Flag=0;
public static volatile int F2_Flag=0;
public static volatile boolean TwoFlag=false;
public static volatile WaveformPlotThread plot_thread;
public static volatile Object referee;
public static volatile WaveformView plot_area;
//public static volatile byte[][] ByteAux = new byte[8][8];
private LinearLayout LiLayMain;
public  ArrayList<Configuracion> Configuraciones=null;

private ArrayList<String> mPairedDevicesArrayList;
public  ArrayList<String> mNewDevicesArrayList = new ArrayList<String>();
private ArrayList<CustomizedBluetoothDevice> mDeviceList;

ArrayAdapter<String> adapter,detectedAdapter;
ArrayList<BluetoothDevice> arrayListBluetoothDevices = null;
String DeviceNSt;
String DeviceASt;
ArrayList<String> DevicesListASt=new ArrayList<String>();
ArrayList<String> DevicesListNSt=new ArrayList<String>();
```

```
public static volatile Paint Iamax = new Paint();
public static volatile Paint Ibmax = new Paint();
public static volatile Paint Icmax = new Paint();
public static volatile Paint Inmax = new Paint();
public static volatile Paint Vamax = new Paint();
public static volatile Paint Vbmax = new Paint();
public static volatile Paint Vcmax = new Paint();

public static int VFull=1110;
public static int AFull=1111;
public static int TxtFull;
public static int BackFull;
public static int Escala0;

public static String BTITULO;
public static String BFechaDBuild;
public static String BFechaDesync;
public static String BFechaDInicio;
//Valores Maximos
public static String BValorVpA;
public static String BValorVpB;
public static String BValorVpC;
public static String BValorApA;
public static String BValorApB;
```

```
public static String BValorApC;
public static String BValorApN;
////valores RMS
public static String BValorVrmsA;
public static String BValorVrmsB;
public static String BValorVrmsC;

public static String BValorArmsA;
public static String BValorArmsB;
public static String BValorArmsC;
public static String BValorArmsN;

public DibujoSwitch EditorPlot = new DibujoSwitch();



@Override
protected void onCreate(Bundle savedInstanceState) {
   super.onCreate(savedInstanceState);
   setContentView(R.layout.mainscreen);
   LiLayMain=(LinearLayout)findViewById(R.id.LiLayMainID);
   arrayListBluetoothDevices = new ArrayList<BluetoothDevice>();
   ViewTreeObserver Vtree=LiLayMain.getViewTreeObserver();
```

```
ConfigBT();
pairedDevices = AdaptadorBT.getBondedDevices();
OpcionesBufferS=new int[pairedDevices.size()+100][8][10];

Vtree.addOnGlobalLayoutListener(new ViewTreeObserver.OnGlobalLayoutListener() {
   @Override
   public void onGlobalLayout() {

   }
});



referee=true;
//////////////////////////////////////////////////////////
pageAdapter = new MyPagerAdapter(getSupportFragmentManager());
ViewPager pager = (ViewPager) findViewById(R.id.CVPageID);
pager.setAdapter(pageAdapter);
pager.addOnPageChangeListener(pageChangeListener);

PowerManager pm= (PowerManager)getSystemService(Context.POWER_SERVICE);
this.EnergyLock=pm.newWakeLock(PowerManager.SCREEN_BRIGHT_WAKE_LOCK ,"TAG");
this.EnergyLock.acquire();
```

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

PQ_UAQ_V7.ChColor= new Paint[8];
PQ_UAQ_V7.V1_color.setColor(Color.rgb(255, 0, 0));//0 Rojo
PQ_UAQ_V7.V2_color.setColor(Color.rgb(255, 255, 0));//1 basura
PQ_UAQ_V7.V3_color.setColor(Color.rgb(0, 255, 0));//2
PQ_UAQ_V7.V4_color.setColor(Color.rgb(255, 128, 0));//3

PQ_UAQ_V7.A1_color.setColor(Color.rgb(0, 255, 128));//4
PQ_UAQ_V7.A2_color.setColor(Color.rgb(0, 128, 255));//5
PQ_UAQ_V7.A3_color.setColor(Color.rgb(153, 51, 255));//7f
PQ_UAQ_V7.A4_color.setColor(Color.rgb(255, 51, 255));//8

Vamax.setColor(Color.RED);
Vbmax.setColor(Color.BLUE);
Vcmax.setColor(Color.GREEN);
Inmax.setColor(Color.WHITE);
Iamax.setColor(Color.RED);
Ibmax.setColor(Color.BLUE);
Icmax.setColor(Color.GREEN);

Iamax.setStyle(Paint.Style.STROKE);
Ibmax.setStyle(Paint.Style.STROKE);
Icmax.setStyle(Paint.Style.STROKE);
Inmax.setStyle(Paint.Style.STROKE);
```

```
        Vamax.setStyle(Paint.Style.STROKE);
        Vbmax.setStyle(Paint.Style.STROKE);
        Vcmax.setStyle(Paint.Style.STROKE);
        Iamax.setPathEffect(new DashPathEffect(new float[]{5, 10, 15, 20}, 0));
        Ibmax.setPathEffect(new DashPathEffect(new float[]{5, 10, 15, 20}, 0));
        Icmax.setPathEffect(new DashPathEffect(new float[]{5, 10, 15, 20}, 0));
        Inmax.setPathEffect(new DashPathEffect(new float[]{5, 10, 15, 20}, 0));
        Vamax.setPathEffect(new DashPathEffect(new float[]{5, 10, 15, 20}, 0));
        Vbmax.setPathEffect(new DashPathEffect(new float[]{5, 10, 15, 20}, 0));
        Vcmax.setPathEffect(new DashPathEffect(new float[]{5, 10, 15, 20}, 0));

//      Iamax.setPathEffect(new DashPathEffect(new float[]{5, 10, 15, 20}, 0));

        PQ_UAQ_V7.TxtColor.setColor(Color.YELLOW);
        PQ_UAQ_V7.TxtColor.setTextSize(20);

    }

    private ViewPager.OnPageChangeListener pageChangeListener = new
ViewPager.OnPageChangeListener() {

        int currentPosition = 0;
```

```
    @Override
    public void onPageSelected(int newPosition) {

      FragmentLifecycle fragmentToHide = (FragmentLifecycle)pageAdapter.getItem(currentPosition);
      fragmentToHide.onPauseFragment();

      FragmentLifecycle fragmentToShow = (FragmentLifecycle)pageAdapter.getItem(newPosition);
      fragmentToShow.onResumeFragment();

      currentPosition = newPosition;
    }

    @Override
    public void onPageScrolled(int arg0, float arg1, int arg2) { }

    public void onPageScrollStateChanged(int arg0) { }
  };

  public static void SaveOpcionesBuffer(int OpcionState) {
    OpcionesBufferS[OpcionesBuff1][OpcionesBuff2][OpcionesBuff3]=OpcionState;
  }

  @Override
```

```
    public void onResume() {
        super.onResume();
//      ConfigBT();
//       pairedDevices = AdaptadorBT.getBondedDevices();
//       OpcionesBufferS=new int[pairedDevices.size()+100][8][10];

//      pageAdapter.notifyDataSetChanged();
//                                  ExampleFragment fragment = (ExampleFragment)
getFragmentManager().findFragmentById(R.id.example_fragment);
//      fragment.<specific_function_name>();
    }

    @Override
    protected void onPause() {
        super.onPause();
    }

    @Override
    public void onDestroy() {
        super.onDestroy();
        Servicio_BT.stop();
        // release screen being on
        if (EnergyLock.isHeld()) {
            EnergyLock.release();
        }
```

```
}

public void find() {

   if (AdaptadorBT.isDiscovering()) {
      AdaptadorBT.cancelDiscovery();
   }

   else {
      // mNewDevicesArrayList.clear();
      AdaptadorBT.startDiscovery();
      if(RunSyncBool==false) {
         registerReceiver(mReceiver, new IntentFilter(BluetoothDevice.ACTION_FOUND));
      }
   }
}

public void BtOFF() {
   Servicio_BT.stop();
   AdaptadorBT.disable();
}

public void pagerADd() {
   pageAdapter.FragmentAdd();
   pageAdapter.notifyDataSetChanged();
```

```
}

public void BtON() {
   AdaptadorBT.enable();
}

public void BtnStringBuilderSearch(){
 find();
   FlagBuscarDispositivo=true;
}

public void BtnConect(String Addres) {
   ConfigBT();
   //find();
   String address = Addres;
   BluetoothDevice device = AdaptadorBT.getRemoteDevice(address);
   Servicio_BT.connect(device);
   if(FlagBuscarDispositivo)
      unregisterReceiver(mReceiver);

   FlagBuscarDispositivo=false;
}

public void BtnDisconect() {
   Servicio_BT.stop();
```

```
//       if(FlagBuscarDispositivo)
//           unregisterReceiver(mReceiver);


   }



   public  void sendMessage(String message) {
      if (Servicio_BT.getState() == ConexionBT.STATE_CONNECTED) {//checa si estamos conectados
a BT
         if (message.length() > 0) {   // checa si hay algo que enviar
            byte[] send = message.getBytes();//Obtenemos bytes del mensaje
            if(D) Log.e("SendM=>", "Mensaje enviado:" + message);
            Servicio_BT.write(send);//Mandamos a escribir el mensaje
         }
      } else;
   }//fin de sendMessage


//
   public  void sendMessageBCD(byte[] message) {
      if (Servicio_BT.getState() == ConexionBT.STATE_CONNECTED) {//checa si estamos conectados
a BT
            byte[] send = message;//Obtenemos bytes del mensaje
            if(D) Log.e("NATBCD", "Mensaje enviado:" + message);
            Servicio_BT.write(send);//Mandamos a escribir el mensaje
```

```
        if(send.length< 0){
            Toast.makeText(this, "write() of 4 bytes failed!", Toast.LENGTH_SHORT).show();
        }
      } else;
   }//fin de sendMessage

private final BroadcastReceiver mReceiver = new BroadcastReceiver() {
   @Override
   public void onReceive(Context context, Intent intent) {
      Message msg = Message.obtain();
      String action = intent.getAction();
      if(BluetoothDevice.ACTION_FOUND.equals(action)){
         BluetoothDevice device = intent.getParcelableExtra(BluetoothDevice.EXTRA_DEVICE);

         if(D)Toast.makeText(context,                    "ACTION_FOUND=>"+device.getName(),
Toast.LENGTH_SHORT).show();
         try
         {
            device.getClass().getMethod("setPairingConfirmation",    boolean.class).invoke(device,
true);
            device.getClass().getMethod("cancelPairingUserInput", boolean.class).invoke(device);
         }
         catch (Exception e) {
            if(D)Log.i("Log", "Inside the exception: ");
            e.printStackTrace();
```

```
        }

        if(arrayListBluetoothDevices.size()<1) // this checks if the size of bluetooth device is 0,then add the
        {                                      // device to the arraylist.
           arrayListBluetoothDevices.add(device);
           if(device==null){

           }else{
              if(RunSyncBool==false) {
//                 detectedAdapter.add(device.getName()+"\n"+device.getAddress());
                 DeviceNSt = device.getName().toString();
                 DeviceASt = device.getAddress().toString();
                 DevicesListNSt.add(DeviceNSt);
                 DevicesListASt.add(DeviceASt);
//                 detectedAdapter.notifyDataSetChanged();
              }
           }
        }
        else
        {
           boolean flag = true;    // flag to indicate that particular device is already in the arlist or not
           for(int i = 0; i<arrayListBluetoothDevices.size();i++)
           {
              if(device.getAddress().equals(arrayListBluetoothDevices.get(i).getAddress())) {
```

```
                flag = false;
            }
        }

        if(flag == true)
        {
            arrayListBluetoothDevices.add(device);
            if(RunSyncBool==false) {
//                  detectedAdapter.add(device.getName()+"\n"+device.getAddress());

                DeviceNSt = device.getName().toString();
                DeviceASt = device.getAddress().toString();
                DevicesListNSt.add(DeviceNSt);
                DevicesListASt.add(DeviceASt);
//                  detectedAdapter.notifyDataSetChanged();
            }
        }

      }
    }

    BtDispNames=new String[3][arrayListBluetoothDevices.size()];
    ListView ListExperiment;
    ListExperiment=(ListView)findViewById(R.id.F1LstVComponentID);
    for(int j =0;j<arrayListBluetoothDevices.size();j++){
```

```
        BtDispNames[0][j] = String.valueOf(DevicesListNSt.get(j));
        BtDispNames[1][j] = String.valueOf(DevicesListASt.get(j));
      }

      ListAdapter List1Adapter = new
PQ_UAQF1Adapter1(getApplicationContext(),BtDispNames[0]);
      ListExperiment.setAdapter(List1Adapter);

      if(D)Toast.makeText(getApplicationContext(),"Showing Paired
Devices",Toast.LENGTH_SHORT).show();
    }
};

   public void ConfigBT(){
      // Obtenemos el adaptador de bluetooth
      AdaptadorBT = BluetoothAdapter.getDefaultAdapter();
      if (AdaptadorBT.isEnabled()) {//Si el BT esta encendido,
         if (Servicio_BT == null) {//y el Servicio_BT es nulo, invocamos el Servicio_BT
            if(D)Toast.makeText(getApplicationContext(), "ConfigBT_ifif ",
Toast.LENGTH_SHORT).show();
            Servicio_BT = new ConexionBT(getApplicationContext(), mHandler);
         }
      }
      else{ if(D) Log.e("Setup", "Bluetooth apagado...");
         Intent enableBluetooth = new Intent(BluetoothAdapter.ACTION_REQUEST_ENABLE);
         startActivityForResult(enableBluetooth, REQUEST_ENABLE_BT);
```

```
    }
}

public BluetoothSocket createSocket(final BluetoothDevice device)
        throws IOException {
    Btsocket= device.createRfcommSocketToServiceRecord(UUID
            .fromString("00001101-0000-1000-8000-00805F9B34FB"));
    return Btsocket;
}

public void onActivityResult(int requestCode, int resultCode, Intent data) {
    //Una vez que se ha realizado una actividad regresa un "resultado"...
    switch (requestCode) {
        case REQUEST_ENABLE_BT://Respuesta de intento de encendido de BT
            if (resultCode == Activity.RESULT_OK) {//BT esta activado,iniciamos servicio
                ConfigBT();
            } else {//No se activo BT, salimos de la app
                finish();}
    }//fin de switch case
}//fin de onActivityResult


//////////////////////////////////////////
///Sync//////////////////////////////////
public void BtnSync(){
```

```
    mHandler.removeCallbacks(r);
    RunSyncBool=true;
    F1TxtOpciones=(TextView)findViewById(R.id.Fn2TxtPresetWordID);
    SyncFlag=false;
    DoWhileFlag=true;
    DoWhileFlag2=true;
    dms = 0;
    j = 0;
    BuffIndice=0;
    bufSyncR=new char[30];
    Syncstr=new char[100];
    Syncmsg=new char[100];
    Syncstr[0] = '\0';
    Syncmsg[0] = '\0';
    sendSyncTime();
    sendMessage("a");
    new Thread(SyncR).start();
    //mHandler.postDelayed(SyncR, 1);
}

public void DoRunable(){
    if (j > 1) {
        buffSt = (dmsL > 10) ? "D" : "U";
        if (D) Log.e("bufSyncR", "bufSyncR=>"+buffSt);
        sendMessage(buffSt);
```

```
        }
////////////////////////////////
        if(DoWhileFlag2==true){
           fmsL=read_rtc_time();
           if (D) Log.d("bufSyncR", "Fms=>"+fmsL);
           DoWhileFlag2=false;
        }

        if(fmsL >= 0 ){
           fmsL=read_rtc_time();
           if (D) Log.d("bufSyncR", "Fms=>"+fmsL);
        }

        ////////////////////////
        msL = STimeVars();
        if (D) Log.e("bufSyncR", "msl=>"+msL);
        dmsL = msL - fmsL;
        Log.d("bufSyncR","J"+j+"==dmsl=="+dmsL+"=(msl=>"+msL+")-(fms=>"+fmsL+")");

        if (dmsL < 0)
           dmsL += 1000;
        ++j;
        j%=500;
     }
```

```
final Runnable SyncR = new Runnable() {
   public void run() {
      if(DoWhileFlag){
         DoRunable();
         DoWhileFlag=false;
      }else if((dmsL > 1) && (j<500)&&DoWhileFlag==false){
         DoRunable();
      }
      if(dmsL < 1){
         mHandler.removeCallbacks(SyncR);
            RunSyncBool=false;
      }else{
         new Thread(SyncR).start();
      }
   }
};

public void BtnPlot(){
   SyncFlag=true;
   preSet();
   TxtPreset=(TextView)findViewById(R.id.Fn2TxtPresetWordID);
   SaveproccesData();
   new Thread(r).start();
}
```

```
public void BtnPreset(){
    SyncFlag=true;
    preSet();
    TxtPreset=(TextView)findViewById(R.id.Fn2TxtPresetWordID);;
}

public void BtnTextStopy(){
}

public void sendSyncTime(){
    byte BO []=new byte[30];
    byte SO[];
    STimeVars();
    SO= BDivide(milisecond);
    BO[0]=St_2_B_BCD("00");
    BO[1]=St_2_B_BCD(Hour);
    BO[2]=St_2_B_BCD(Minute);
    BO[3]=St_2_B_BCD(Second);
    BO[4]=SO[0];
    BO[5]=SO[1];
    sendMessage("t");
    sendMessageBCD(BO);
    if (D) Log.e(TAG, "t"+","+String.valueOf(BO[0])+":"+String.valueOf(BO[1])+":"+String.valueOf(BO[2])+":"+String.valueOf(BO[3])+"ms"+String.valueOf(BO[4])+"Ms"+String.valueOf(BO[5]));
```

```
}

public byte[] BDivide(String c){
    byte ByteBank[]=new byte[2];
    short Shorty;
    Shorty=Short.valueOf(c);
    ByteBank[0]= (byte) ((Shorty)&0xFF);
    ByteBank[1]= (byte) ((Shorty>>8)&0xFF);
    return ByteBank;
}

public String[] MsDivider(String milis){
    String[] TotalSt=new String[2];
    char First = milis.charAt(0);
    char Second = milis.charAt(1);
    char Third = milis.charAt(2);
    char Forth = '0';
    String St1=String.valueOf(First);
    String St2=String.valueOf(Second);
    String St3=String.valueOf(Third);
    String St4=String.valueOf(Forth);
    TotalSt[0]=St1+St2;
    TotalSt[1]=St3+St4;
    return TotalSt;
}
```

```
public long read_rtc_time(){
    long n;
    int dia_r,hora_r,minuto_r,segundo_r,milisegundos;
    //short milisS;
    dia_r=0;
    hora_r=0;
    minuto_r=0;
    segundo_r=0;
    milisegundos=0;
    //ask_current_time();
    dia_r=SyncFpgaTime [0];
    hora_r=bcd_2_bin(SyncFpgaTime [1]);
    minuto_r=bcd_2_bin(SyncFpgaTime [2]);
    segundo_r=bcd_2_bin(SyncFpgaTime [3]);
    milisegundos=((SyncFpgaTime [5]<<8)|SyncFpgaTime[4]);
    milisegundos=UByte(milisegundos);
    n=((hora_r*60+minuto_r)*60+segundo_r)+milisegundos;
    if (D) Log.e("read_rtc_time", "Hora=>" +
hora_r+"Minuto=>"+minuto_r+"Segundos=>"+segundo_r+"Segundos=>"+segundo_r+"milisegund
os"+milisegundos);
    return n;
}

public int bcd_2_bin(byte b){
```

```
    int valor;
    int x1,x2;
    x1=(b&0xF0)>>4;
    x2=(b&0x0F);
    valor=x1*10+x2;
    return valor;
}

private int UByte(int b){
    if(b<0) // if negative
        return (int)( (b&0x7F) + 128 );
    else
        return (int)b;
}

public long STimeVars(){
    long PQMilliseconds = System.currentTimeMillis();
    short mHour,mMinute,mSecond,mmilisecond;
    long HFormatMilies;
    //Date Momment = Calendar.getInstance().getTime();
    SimpleDateFormat simpleDayFormat = new SimpleDateFormat("dd");
    SimpleDateFormat simpleMonthFormat = new SimpleDateFormat("M");
    SimpleDateFormat simpleyearFormat = new SimpleDateFormat("y");
    SimpleDateFormat simpleHourFormat = new SimpleDateFormat("HH");
    SimpleDateFormat simpleMinFormat = new SimpleDateFormat("mm");
```

```
    SimpleDateFormat simpleSecFormat = new SimpleDateFormat("ss");
    SimpleDateFormat simplemsFormat = new SimpleDateFormat("SSS");

    Day=simpleDayFormat.format(PQMilliseconds);
    Month=simpleMonthFormat.format(PQMilliseconds);
    Year=simpleyearFormat.format(PQMilliseconds);
    Hour=simpleHourFormat.format(PQMilliseconds);
    Minute=simpleMinFormat.format(PQMilliseconds);
    Second=simpleSecFormat.format(PQMilliseconds);
    milisecond=simplemsFormat.format(PQMilliseconds);

   // mHour=Integer.valueOf(Hour);
    mHour=Short.valueOf(Hour);
    mMinute=Short.valueOf(Minute);
    mSecond=Short.valueOf(Second);
    mmilisecond=Short.valueOf(milisecond);
    Log.e("STimeVars", "AHour=>" +
mHour+"Amin=>"+mMinute+"Asec=>"+mSecond+"Ams=>"+mmilisecond);
    HFormatMilies= ((mHour*60+mMinute)*60+mSecond)+mmilisecond;
    milistime=HFormatMilies;
    return milistime;
  }

  public byte St_2_B_BCD(String b){
    char first = b.charAt(0);
```

```
    char second = b.charAt(1);
    byte Char1b = (byte)first;
    byte Char2b = (byte)second;
    int x1=(Char1b&0x0F);
    int x2=(Char2b&0x0F);
    byte xfull= (byte) ((x1<<4)|(x2));
    return xfull;
}

public void SetPLotEditorChanges(){
    EditorPlot.DibujoSwitchV(VFull);
    EditorPlot.DibujoSwitchA(AFull);
}

public void ADDPAGER(){
    pageAdapter.fragments.add(new PQ_UAQFRAGMENT3());
    pageAdapter.fragments.add(new PQ_UAQFRAGMENT2());
    pageAdapter.notifyDataSetChanged();
}

public static void BufferChF(){
    PQ_UAQ_V7.ch_FV1 = new float[PQ_UAQ_V7.VWidth];
    PQ_UAQ_V7.ch_FV2 = new float[PQ_UAQ_V7.VWidth];
    PQ_UAQ_V7.ch_FV3 = new float[PQ_UAQ_V7.VWidth];
    PQ_UAQ_V7.ch_FV4 = new float[PQ_UAQ_V7.VWidth];
```

```
    PQ_UAQ_V7.ch_FA1 = new float[PQ_UAQ_V7.VWidth];
    PQ_UAQ_V7.ch_FA2 = new float[PQ_UAQ_V7.VWidth];
    PQ_UAQ_V7.ch_FA3 = new float[PQ_UAQ_V7.VWidth];
    PQ_UAQ_V7.ch_FA4 = new float[PQ_UAQ_V7.VWidth];

}

public void preSet(){
    ch_PFA1 = new float[PQ_UAQ_V7.VWidth];
    ch_PFA2 = new float[PQ_UAQ_V7.VWidth];
    ch_PFA3 = new float[PQ_UAQ_V7.VWidth];
    ch_PFAN = new float[PQ_UAQ_V7.VWidth];
    ch_PFV1 = new float[PQ_UAQ_V7.VWidth];
    ch_PFV2 = new float[PQ_UAQ_V7.VWidth];
    ch_PFV3 = new float[PQ_UAQ_V7.VWidth];
    ch_PFAux = new float[PQ_UAQ_V7.VWidth];
}

public static void SaveChF2(float[] CanalF,int[] bufferF,float Ganancia ){
    for (int x = 0; x < PQ_UAQ_V7.VWidth; x++){
        if(x<512){
            CanalF[x] = ((float)bufferF[x])*256f*Ganancia;
        }else
        {
            CanalF[x] = 0;
```

```
        }
    }
}


public static void Save_0(float[] CanalF,int[] bufferF){
    for (int x = 0; x < PQ_UAQ_V7.VWidth; x++){
        if(x<512){
            CanalF[x] = 0;
        }else
        {
            CanalF[x] = 0;
        }
    }
}

public void SaveproccesData(){
    Log.e(TAG, "Mensaje leido   =w= FAGS");
        SaveChF2(ch_PFA1,  GlobalBuff[0],  PQ_UAQFRAGMENT1.Tv.get(0).Ganancia);
        SaveChF2(ch_PFA2,  GlobalBuff[1],  PQ_UAQFRAGMENT1.Tv.get(1).Ganancia);
    SaveChF2(ch_PFA3,  GlobalBuff[2],  PQ_UAQFRAGMENT1.Tv.get(2).Ganancia);
    SaveChF2(ch_PFAN,  GlobalBuff[3],  PQ_UAQFRAGMENT1.Tv.get(3).Ganancia);
    SaveChF2(ch_PFV1,  GlobalBuff[4],  PQ_UAQFRAGMENT1.Tv.get(4).Ganancia);
    SaveChF2(ch_PFV2,  GlobalBuff[5],  PQ_UAQFRAGMENT1.Tv.get(5).Ganancia);
    SaveChF2(ch_PFV3, GlobalBuff[6], PQ_UAQFRAGMENT1.Tv.get(6).Ganancia);
```

```
    SaveChF2(ch_PFAux, GlobalBuff[7], PQ_UAQFRAGMENT1.Tv.get(7).Ganancia);
}

float max(float [] Canal,int ind){
    float val = 0;
    for(int i=0; i<Canal.length-1; ++i){
        if(val<Canal[i]) val = Canal[i];
    }
    IndiMaxVal[ind]=val;
    return val;
}

float min(float [] Canal,int ind){
    float val = 0;
    for(int i=0; i<Canal.length-1; ++i){
        if(val>Canal[i]) val = Canal[i];
    }
    return val;
}

float rms(float [] Canal,int ind){
    float val = 0f;
    float  valAuxF = 0;
    for(int i=0; i<512; ++i)
        val += Canal[i]*Canal[i];
```

```
    valAuxF=(float)(Math.sqrt(val / 512));//remplace canal.lenght por 512 para que sea la muestra de los datos exepto los valores en cero que se agregan para rellenar la pantalla
    return valAuxF ;
  }

  float pp(float [] Canal,int ind){
    float Max = 0;
    float Min = 0;
    for(int i=0; i<Canal.length-1; ++i){
      if(Max<Canal[i]) Max = Canal[i];
      if(Min>Canal[i]) Min = Canal[i];
    }
    return Max-Min;
  }

  float mean (float [] Canal,int ind){
    float val = 0;
    for(int i=0; i<Canal.length-1; ++i)
      val += Canal[i];
    return val/Canal.length-1;
  }


  final Runnable r = new Runnable() {
    public void run() {
```

```
          FifoRow=0;
          FifoCol=0;
          sendMessage("Z");
//          Log.e(TAG, "Message_write  =w= " + "Z");
          mHandler.postDelayed(r, 600);
      }
   };

   //////////////////////////////////////////
   final Handler mHandler = new Handler() {
      @Override
      public void handleMessage(Message msg) {
         switch (msg.what) {

//>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
>>>>>>>>>>>>>>>>>>>>>>
            case Mensaje_Escrito:
               byte[] writeBuf = (byte[]) msg.obj;//buffer de escritura...
               // Construye un String del Buffer
               String writeMessage = new String(writeBuf);
               Log.e(TAG, "Message_write  =w= " + writeMessage);
               break;

//>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
>>>>>>>>>>>>>>>>>>>>>>
            case Mensaje_Leido:
```

```
if(F2_Flag==1){
   SetPLotEditorChanges();
}
   if (F3TxtFlag) {
      SetSTsMax();
      MText = "Amax||    " +
            "   A1=" + Sch_PFA1 + "A" +
            "   A2=" + Sch_PFA2+ "A" +
            "   A3=" + Sch_PFA3 + "A" +
            "   AN=" + Sch_PFAN + "A"+
            "   V1=" + Sch_PFV1 + "V" +
            "   V2=" + Sch_PFV2+ "V" +
            "   V3=" + Sch_PFV3 + "V"
            ;
      switch (ITxtOp1){
         case 00:
            RMSTxt = "min A||    " +
                  "   A1=" + String.valueOf(min(ch_PFA1, 0)) + "A" +
                  "   A2=" + String.valueOf(min(ch_PFA2, 1)) + "A" +
                  "   A3=" + String.valueOf(min(ch_PFA3, 2)) + "A" +
                  "   AN=" + String.valueOf(min(ch_PFAN, 3)) + "A"
            ;
            RMSVTEXT="min V||"+ "V1=" + String.valueOf(min(ch_PFV1, 4)) + "V" +
                  "   V2=" + String.valueOf(min(ch_PFV2, 5)) + "V" +
                  "   V3=" + String.valueOf(min(ch_PFV3,6)) + "V"
```

```
    ;
    break;
case 01:
    RMSTxt = "RMS A||    " +
          "   A1=" + String.valueOf(rms(ch_PFA1, 0)) + "A" +
          "   A2=" + String.valueOf(rms(ch_PFA2, 1)) + "A" +
          "   A3=" + String.valueOf(rms(ch_PFA3, 2)) + "A" +
          "   AN=" + String.valueOf(rms(ch_PFAN, 3)) + "A"
          ;
    RMSVTEXT="RMS V||"+ "V1=" + String.valueOf(rms(ch_PFV1, 4)) + "V" +
          "   V2=" + String.valueOf(rms(ch_PFV2, 5)) + "V" +
          "   V3=" + String.valueOf(rms(ch_PFV3, 6)) + "V"
    ;
case 02:
    RMSTxt = "PP A||    " +
          "   A1=" + String.valueOf(pp(ch_PFA1, 0)) + "A" +
          "   A2=" + String.valueOf(pp(ch_PFA2, 1)) + "A" +
          "   A3=" + String.valueOf(pp(ch_PFA3, 2)) + "A" +
          "   AN=" + String.valueOf(pp(ch_PFAN, 3)) + "A"
    ;
    RMSVTEXT="PP V||" + "V1=" + String.valueOf(pp(ch_PFV1, 4)) + "V" +
          "   V2=" + String.valueOf(pp(ch_PFV2, 5)) + "V" +
          "   V3=" + String.valueOf(pp(ch_PFV3,6)) + "V"
    ;
    break;
```

```
            default:
                //maxivalues
                break;

        }
    }

    if (SyncFlag) {
        DrawFlag = 1;
        int raw, data_length, x, dataI1, dataI2;
        int Ibyte;
        byte[] readBuf = (byte[]) msg.obj;//buffer de lectura..
        data_length = msg.arg1;
        if (D) Log.d(TAG, "data_Length=>" + data_length);
        for (x = 0; x < data_length; x++) {
            if (D) Log.d(TAG, "Index=>" + x);
            if (D) Log.d(TAG, "Stt=>" + Stt);
            Ibyte = readBuf[x];
            if (D) Log.d(TAG, "Ibyte=>" + Ibyte);
            //PrivateBuff[FifoCol][FifoRow]=0;
            //PrivateBuff[FifoCol][FifoRow] = Ibyte;
            GlobalBuff[FifoCol][FifoRow] = Ibyte;
//          Log.d(TAG, "GlobalBuff=>" + PQ_UAQ_V7.GlobalBuff[FifoCol][FifoRow]);
//          Log.d(TAG, "GlobalBuffix=>Row=>" + FifoRow + "Col=>" + FifoCol);
```

```
        if (FifoCol == 7) {
            if (D) Log.d(TAG, "FifoRow");
            FifoRow++;
            oneFlag = FifoRow;
            FifoRow %= 512;
            Cot = 0;
            SendTrig++;
            SendTrig %= 512;
            Stt = 1;
            if (D) Log.d(TAG, "Mensaje ST=>" + SendTrig);
        }
        if (D) Log.d(TAG, "FifoCol");
        FifoCol++;
        FifoCol %= 8;
    }
    String readMessage = new String(readBuf, 0, msg.arg1);
    if (D) Log.e(TAG, "Mensaje leido   =w= " + readMessage);
    ClkFlag = 1;

} else {
    DrawFlag = 1;
    int raw, data_length, x, dataI1, dataI2;
    int Ibyte;
    byte[] readBuf = (byte[]) msg.obj;//buffer de lectura..
    data_length = msg.arg1;
```

```
                if (D) Log.d(TAG, "data_Length=>" + data_length);
                for (x = 0; x < data_length; x++) {
                   if (D) Log.d(TAG, "Index=>" + x);
                   if (D) Log.d(TAG, "Stt=>" + Stt);
                   Ibyte = readBuf[x];
                   if (D) Log.d(TAG, "readBuf{" + x + "}=>" + readBuf[x]);
                   SyncFpgaTime[BuffIndice] = readBuf[x];
                   if (D) Log.e(TAG, "FifoCol");
                   BuffIndice++;
                   BuffIndice %= 6;
                }
                String readMessage = new String(readBuf, 0, msg.arg1);
                if (D) Log.e(TAG, "Mensaje leido   =w= " + readMessage);

                sendMessageBCDT("a".getBytes());
                ClkFlag = 1;
                Long Vaporware;
                Vaporware = dmsL;
                if (FlagPrintSync) {
                   if (Vaporware <= 1) {
                      if (Vaporware < 0) {
                         F1TxtOpciones.setText("Fallo la sincronizacion=>" + dmsL);
                      } else {
                         F1TxtOpciones.setText("SINC=" + dmsL + "=Hora=" + Hour + ":" + Minute + ":"
+ Second + "." + milisecond);
```

```
                    PQ_UAQ_V7.BFechaDesync= Hour + ":" + Minute + ":" + Second + "." +
milisecond;
                  }
                } else {
                  F1TxtOpciones.setText("Sincronizando...");
                }
              }
            }

          break;

//>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
>>>>>>>>>>>>>>>>>>>>>>>>

        case Mensaje_Nombre_Dispositivo:
          mConnectedDeviceName = msg.getData().getString(DEVICE_NAME); //Guardamos
nombre del dispositivo
          Toast.makeText(getApplicationContext(), "Conectado con "+ mConnectedDeviceName,
Toast.LENGTH_SHORT).show();
          seleccionador=true;
          break;

//>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
>>>>>>>>>>>>>>>>>>>>>>>>

        case Mensaje_TOAST:
          Toast.makeText(getApplicationContext(), msg.getData().getString(TOAST),
              Toast.LENGTH_SHORT).show();
          break;
```

```
//>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
>>>>>>>>>>>>>>>>>>>>>>>

            case MESSAGE_Desconectado:

               if(D) Log.e("Conexion","DESConectados");

               seleccionador=false;

               break;

//>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
>>>>>>>>>>>>>>>>>>>>>>>

         }//FIN DE SWITCH CASE PRIMARIO DEL HANDLER

      }

      public  void sendMessageBCDT(byte[] message) {

         if (Servicio_BT.getState() == ConexionBT.STATE_CONNECTED) {//checa si estamos
conectados a BT

            byte[] send = message;//Obtenemos bytes del mensaje

            if(D) Log.e("SendBCD", "Mensaje enviado:" + message);

            Servicio_BT.write(send);//Mandamos a escribir el mensaje

         } else;

      }//fin de sendMessage





      public void SaveChF2A(float[] CanalF,int[] bufferF,float Ganancia ){

         for (int x = 0; x < 512 - 1; x++){
```

```
        if(x<512){
           CanalF[x] = ((float)bufferF[x]*256f*Ganancia);
        }else
        {
           CanalF[x] = 0;
        }
      }
   }

   public void SaveChF(float[] CanalF,int[] bufferF ){
      for (int x = 0; x < PQ_UAQ_V7.VWidth - 1; x++){
        if(x<512){
           CanalF[x] = (float)bufferF[x];
        }else
        {
           CanalF[x] = 0;
        }
      }
   }
};//Fin de Handler

public void SetSTsMax(){
   Sch_PFA1=String.valueOf(max(ch_PFA1,0));
   Sch_PFA2=String.valueOf(max(ch_PFA2,1));
   Sch_PFA3=String.valueOf(max(ch_PFA3,2));
```

```
   Sch_PFAN=String.valueOf(max(ch_PFAN,3));
   Sch_PFV1=String.valueOf(max(ch_PFV1,4)/sqrtF);
   Sch_PFV2=String.valueOf(max(ch_PFV2,5)/sqrtF);
   Sch_PFV3=String.valueOf(max(ch_PFV3,6)/sqrtF) ;
}

@Override
public boolean onCreateOptionsMenu(Menu menu) {
   // Inflate the menu; this adds items to the action bar if it is present.
   getMenuInflater().inflate(R.menu.menu_pq__uaq__v7, menu);



   return true;
}

@Override
public boolean onOptionsItemSelected(MenuItem item) {

   switch (item.getItemId()) {
      case R.id.APlay:
         if(PQ_UAQ_V7.F3_Flag==1) {
            PQ_UAQ_V7.plot_area.Stopy();
            PQ_UAQ_V7.plot_thread.setRunning(true);
            PQ_UAQ_V7.TwoFlag = true;
            PQ_UAQ_V7.F3TxtFlag = true;
```

```
            PQ_UAQ_V7.plot_thread.start();
        }else{
        }
        return true;

    case R.id.AStop:
        PQ_UAQ_V7.plot_thread.setRunning(false);
        PQ_UAQ_V7.plot_thread.interrupt();//Cause blocking methods like Thread.sleep() to end
with an Interrupted exception
        try {
            PQ_UAQ_V7.plot_thread.join();//Wait until the thread is completed
        } catch (InterruptedException e) {
            e.printStackTrace();
        }

        MACSAVE= BTITULO;
        SyncSAVE= BFechaDesync;
        TurnOnSAVE=BFechaDInicio;
        RMSSAVE[0]=Sch_PFV1;
        RMSSAVE[1]=Sch_PFV2;
        RMSSAVE[2]=Sch_PFV3;
        RMSSAVE[3]=Sch_PFA1;
        RMSSAVE[4]=Sch_PFA2;
        RMSSAVE[5]=Sch_PFA3;
        RMSSAVE[6]=Sch_PFAN;
```

```
    MAXSAVE[0]=Sch_PFV1;
    MAXSAVE[1]=Sch_PFV2;
    MAXSAVE[2]=Sch_PFV3;
    MAXSAVE[3]=Sch_PFA1;
    MAXSAVE[4]=Sch_PFA2;
    MAXSAVE[5]=Sch_PFA3;
    MAXSAVE[6]=Sch_PFAN;

    return true;
case R.id.ASync:
    SetPLotEditorChanges();
    return true;

case R.id.AChVID:
    mHandler.removeMessages(0);
    return true;

case R.id.AChAID:
    // User chose the "Favorite" action, mark the current item
    // as a favorite...
    return true;


default:
```

```
            // If we got here, the user's action was not recognized.
            // Invoke the superclass to handle it.
            return super.onOptionsItemSelected(item);

    }
}
```

7.15. PQ_UAQFRAGMENT1

```
import android.graphics.Color;
import android.os.Bundle;
import android.support.v4.app.Fragment;
import android.util.Log;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.view.ViewTreeObserver;
```

```
import android.widget.AdapterView;
import android.widget.Button;
import android.widget.LinearLayout;
import android.widget.ListView;
import android.widget.TextView;
import android.widget.Toast;

import java.io.BufferedReader;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.util.ArrayList;
import java.util.Arrays;
import java.util.EventListener;
import java.util.HashMap;
import java.util.List;


public class PQ_UAQFRAGMENT1 extends Fragment implements FragmentLifecycle{

    public float GainDes_0[] = new float [8];
    public float GainDes_1[] = new float [8];
    public float GainDes_2[] = new float [8];
    public float GainDes_3[] = new float [8];
    public float GainDes_4[] = new float [8];
    public float GainDes_5[] = new float [8];
```

```
    public float GainDes_6[] = new float [8];
    public float GainDes_7[] = new float [8];

    private LinearLayout LayF1;
    public boolean FlagFragment = true;

    private EventListener listener;

    public final static String DispConfig="DispConfig.txt";
    public static String Adress="20:15:04:20:35:94";
    private Button BtnConectar,BtnActivar,BtnDesactivar,BtnBuscar,BtnSync,BtnPlot,BtnShowVal,
BtnDesLec;
    private TextView F1TxtOpciones;
    private ListView ListVDevicesList,ListadeOpciones;
    public static String TAG = "PQ_UAQFRAGMENT1 ";

    public static Configuracion Ch1=new Configuracion();
    public static Configuracion Ch2=new Configuracion();
    public static Configuracion Ch3=new Configuracion();
    public static Configuracion Ch4=new Configuracion();
    public static Configuracion Ch5=new Configuracion();
    public static Configuracion Ch6=new Configuracion();
    public static Configuracion Ch7=new Configuracion();
    public static Configuracion Ch8=new Configuracion();
```

```
    public static   ArrayList<Configuracion> Tv =new ArrayList<Configuracion>();

    public static final int Ch_01 = 0;
    public static final int Ch_02 = 1;
    public static final int Ch_03 = 2;
    public static final int Ch_04 = 3;
    public static final int Ch_05 = 4;
    public static final int Ch_06 = 5;
    public static final int Ch_07 = 6;
    public static final int Ch_08 = 7;

    public volatile String IdNum,FullTxt;
    public static   ArrayList<DataSetPreset_T3> SaveLog3 = new ArrayList<DataSetPreset_T3>();

//    PQUAQ.Config_PQUAQ("20:15:04:20:35:94", 0, "RED", "V", 10f, "-", 10f);

    private String PQDeviceName;
    private int    PQChannel;
    private String PQColor;
    private String PQUnits;
    private float  PQGain;
    private String PQlineType;
    private float  PQFrecc;

    public String FullText;
```

```
public String []  DivTxt1;
public String DivTxt2;

public volatile String [] DivFullSt;

public static int ChFlag=0;

public int ArraySizeInt;

public static String GetDownOnIt[][] = new String[10][10];

List<String> expandableListTitle;
HashMap<String, List<String>> expandableListDetail;

@Override
public void onPauseFragment() {
   //Log.i(TAG, "onPauseFragment()");
}

@Override
public void onResumeFragment() {
   Log.i(TAG, "onResumeFragment()");
   PQ_UAQ_V7.BufferChF();
   InitUISizes();
```

```
}
public void InitUISizes(){
   PQ_UAQ_V7.ch_PFA1 = new float[PQ_UAQ_V7.VWidth];
   PQ_UAQ_V7.ch_PFA2 = new float[PQ_UAQ_V7.VWidth];
   PQ_UAQ_V7.ch_PFA3 = new float[PQ_UAQ_V7.VWidth];
   PQ_UAQ_V7.ch_PFAN = new float[PQ_UAQ_V7.VWidth];
   PQ_UAQ_V7.ch_PFV1 = new float[PQ_UAQ_V7.VWidth];
   PQ_UAQ_V7.ch_PFV2 = new float[PQ_UAQ_V7.VWidth];
   PQ_UAQ_V7.ch_PFV3 = new float[PQ_UAQ_V7.VWidth];
   PQ_UAQ_V7.ch_PFAux = new float[PQ_UAQ_V7.VWidth];

}
public void InitConfig(){
   //PQDeviceName="20:15:04:20:35:94";
   PQColor="Red";
   PQUnits="V";
   PQGain=2f;
   PQlineType="-";
   PQFrecc=10f;
}

public void SaveConfig(){
   PQ_UAQ_V7.BackGroundCol=Color.rgb(20,20,20);
   Ch1.Config_PQUAQ(PQDeviceName, Ch_01, "Red",PQUnits,1,PQlineType,PQFrecc);
   Ch2.Config_PQUAQ(PQDeviceName, Ch_02, "Blue",PQUnits,1,PQlineType,PQFrecc);
```

```
    Ch3.Config_PQUAQ(PQDeviceName, Ch_03, "Green",PQUnits,1,PQlineType,PQFrecc);
    Ch4.Config_PQUAQ(PQDeviceName, Ch_04, "White",PQUnits,1,PQlineType,PQFrecc);
    Ch5.Config_PQUAQ(PQDeviceName, Ch_05, "Red",PQUnits,1,PQlineType,PQFrecc);
    Ch6.Config_PQUAQ(PQDeviceName, Ch_06, "Blue",PQUnits,1,PQlineType,PQFrecc);
    Ch7.Config_PQUAQ(PQDeviceName, Ch_07, "Green",PQUnits,1,PQlineType,PQFrecc);
    Ch8.Config_PQUAQ(PQDeviceName, Ch_08, "Magenta", PQUnits, 1, PQlineType, PQFrecc);
    Tv.add(Ch1);
    Tv.add(Ch2);
    Tv.add(Ch3);
    Tv.add(Ch4);
    Tv.add(Ch5);
    Tv.add(Ch6);
    Tv.add(Ch7);
    Tv.add(Ch8);
  }
  @Override
  public View onCreateView(LayoutInflater inflater, final ViewGroup container, Bundle
savedInstanceState) {
    View Fragment1V = inflater.inflate(R.layout.activity_first2, container, false);
    LayF1=(LinearLayout)Fragment1V.findViewById(R.id.LiLayVFrameID);
    //BtnConectar=(Button)Fragment1V.findViewById(R.id.F1BtnConnectID);
    ListVDevicesList=(ListView)Fragment1V.findViewById(R.id.F1LstVComponentID);
    //ListadeOpciones= (ListView)Fragment1V.findViewById(R.id.F1LsVOpcionesID);
    F1TxtOpciones=(TextView)Fragment1V.findViewById(R.id.Fn2TxtPresetWordID);
```

```
        BtnBuscar=(Button)Fragment1V.findViewById(R.id.F1BtnBuscarID);
        BtnSync=(Button)Fragment1V.findViewById(R.id.F1BtnSincID);
        BtnActivar=(Button)Fragment1V.findViewById(R.id.F1BtnInicioID);
        BtnDesLec=(Button)Fragment1V.findViewById(R.id.F1BtnDesLecID);
        BtnPlot=(Button)Fragment1V.findViewById(R.id.F1BtnGrafID);
        BtnDesactivar=(Button)Fragment1V.findViewById(R.id.F1BtnDesactID);
        readFileInEditor();
        SeparandoyProbando();

//      SetActAct1();
//
        InitConfig();
        if(FlagFragment){
           SaveConfig();
           FlagFragment=false;
        }
//      SetValuesGain();
        ViewTreeObserver Vtree=Fragment1V.getViewTreeObserver();
        Vtree.addOnGlobalLayoutListener(new ViewTreeObserver.OnGlobalLayoutListener() {
           @Override
           public void onGlobalLayout() {
              PQ_UAQ_V7.VWidth = LayF1.getWidth();
              PQ_UAQ_V7.VHeight = LayF1.getHeight();
           }
        });
```

```
ListVDevicesList.setOnItemClickListener(
        new AdapterView.OnItemClickListener() {
          @Override
          public void onItemClick(AdapterView<?> parent, View view, int position, long id) {
            String ListDevice = String.valueOf(parent.getItemAtPosition(position));
            Adress = PQ_UAQ_V7.BtDispNames[1][position];
            PQDeviceName = Adress;
            SaveConfig();
            PQ_UAQ_V7.BTITULO = Adress;
            GainSetTesting(Adress);
            //GainSet(Adress);
            ((PQ_UAQ_V7) getActivity()).BtnDisconect();
            ((PQ_UAQ_V7) getActivity()).BtnConect(Adress);
            Toast.makeText(getActivity().getApplicationContext(),                Adress,
Toast.LENGTH_SHORT).show();
            ((PQ_UAQ_V7) getActivity()).BtnPreset();
          }
        }
    );

    BtnBuscar.setOnClickListener(new View.OnClickListener() {
      @Override
      public void onClick(View view) {
        ((PQ_UAQ_V7) getActivity()).BtnStringBuilderSearch();
      }
```

```
    });
    BtnSync.setOnClickListener(new View.OnClickListener() {
      @Override
      public void onClick(View view) {
        ((PQ_UAQ_V7) getActivity()).BtnSync();
      }
    });
    BtnActivar.setOnClickListener(new View.OnClickListener() {
      @Override
      public void onClick(View view) {
        ((PQ_UAQ_V7) getActivity()).STimeVars();
                PQ_UAQ_V7.FlagPrintSync = false;
                F1TxtOpciones.setText("LECTURA INICIADA=>" + PQ_UAQ_V7.Hour + ":" +
PQ_UAQ_V7.Minute + ":" + PQ_UAQ_V7.Second + "." + PQ_UAQ_V7.milisecond);
                PQ_UAQ_V7.BFechaDInicio = PQ_UAQ_V7.Hour + ":" + PQ_UAQ_V7.Minute + ":"
+ PQ_UAQ_V7.Second + "." + PQ_UAQ_V7.milisecond;
                ((PQ_UAQ_V7) getActivity()).sendMessage("s");
                Toast.makeText(getActivity().getApplicationContext(),          "ACTIVADO",
Toast.LENGTH_SHORT).show();
      }
    });
    BtnDesLec.setOnClickListener(new View.OnClickListener() {
      @Override
      public void onClick(View view) {
        ((PQ_UAQ_V7) getActivity()).sendMessage("h");
        PQ_UAQ_V7.FlagPrintSync = false;
```

```
            Toast.makeText(getActivity().getApplicationContext(),                    "DESACTIVADO",
Toast.LENGTH_SHORT).show();
        }
    });
    BtnPlot.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View view) {
            ((PQ_UAQ_V7) getActivity()).BtnPlot();
            ((PQ_UAQ_V7) getActivity()).ADDPAGER();
            PQ_UAQ_V7.F2_Flag=1;
            //initFrag3();
            ((PQ_UAQ_V7) getActivity()).SetPLotEditorChanges();
        }
    });
    BtnDesactivar.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View view) {
            ((PQ_UAQ_V7) getActivity()).BtnDisconect();
        }
    })
    return Fragment1V;
}
public void readFileInEditor()
{
    try {
```

```
        InputStream in = this.getResources().openRawResource(R.raw.pquaq);
        if (in != null) {
           InputStreamReader tmp=new InputStreamReader(in);
           BufferedReader reader=new BufferedReader(tmp);
           String str;
           StringBuilder buf=new StringBuilder();
           while ((str = reader.readLine()) != null){
              buf.append(str);
           }
           in.close();
           FullText = buf.toString();
           DivTxt1 = FullText.split(";");
        }else{
        }
     }
     catch (java.io.FileNotFoundException e) {
// that's OK, we probably haven't created it yet
     }
     catch (Throwable t) {
        Toast.makeText(getActivity().getApplicationContext(), "Exception: " + t.toString(),
Toast.LENGTH_LONG).show();
     }
  }

  public void SeparandoyProbando(){
```

```
        IdNum=DivTxt1[0].trim();
        FullTxt=DivTxt1[1].trim();
        List<String> ArraySets= Arrays.asList(DivTxt1);
        ArraySizeInt= ArraySets.size();
        for(int x=0; x<ArraySizeInt+1; x++) {
           SaveLog3.add(new DataSetPreset_T3());
        }
        for(int x=0; x<ArraySets.size(); x++) {
           SaveLog3.get(x).DataSettings(ArraySets.get(x));
        }
//      TestingArray.DataSaveSet(ArraySets.get(0));
        //TxtData.setText("Complete_BUff=>" + NumSt + "Testing" + SaveLog.get(2).Date_D);
    }
    public void GainSetTesting(String Dispositivo) {
        Log.e(TAG, "GUSANOS");
        for (int x = 0; x < ArraySizeInt; x++) {
           Log.e(TAG, Dispositivo+"-"+SaveLog3.get(x).Mac );
           if (SaveLog3.get(x).Mac.equals(Dispositivo)) {
              Log.e(TAG, "The last of the giants");
              Tv.get(0).Ganancia=Float.parseFloat(SaveLog3.get(x).Gain[0]);
              Tv.get(1).Ganancia=Float.parseFloat(SaveLog3.get(x).Gain[1]);
              Tv.get(2).Ganancia=Float.parseFloat(SaveLog3.get(x).Gain[2]);
              Tv.get(3).Ganancia=Float.parseFloat(SaveLog3.get(x).Gain[3]);
              Tv.get(4).Ganancia=Float.parseFloat(SaveLog3.get(x).Gain[4]);
              Tv.get(5).Ganancia=Float.parseFloat(SaveLog3.get(x).Gain[5]);
```

```
            Tv.get(6).Ganancia=Float.parseFloat(SaveLog3.get(x).Gain[6]);
            Tv.get(7).Ganancia=Float.parseFloat(SaveLog3.get(x).Gain[7]);
          }
        }
    }

    @Override
    public void onResume() {
        super.onResume();
    }
    public static PQ_UAQFRAGMENT1 newInstance(String text) {
        PQ_UAQFRAGMENT1 f = new PQ_UAQFRAGMENT1();
        Bundle b = new Bundle();
        b.putString("msg", text);
        f.setArguments(b);
        return f;
    }

}
```

7.16. PQ_UAQFRAGMENT2

```
import android.content.Intent;
import android.os.Bundle;
import android.support.v4.app.Fragment;
import android.util.Log;
```

```
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.view.ViewTreeObserver;
import android.widget.AdapterView;
import android.widget.Button;
import android.widget.ListAdapter;
import android.widget.ListView;
import android.widget.Toast;

import java.io.BufferedReader;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.io.OutputStreamWriter;
import java.util.ArrayList;
import java.util.Arrays;
import java.util.List;

public class PQ_UAQFRAGMENT2 extends Fragment implements FragmentLifecycle{

    public final static String EDITORMAINLOG="StoreLog.txt";
    public final static String EDITORDEUS="ListNum.txt";
    public final static String LISTROWS="Indice.txt";
    public final static String PASIINGDATA="IntentData.txt";
    private Button BtnPause,BtnPlay,BtnRun;
```

```
    private ListView LstEvento;
    public static final String TAG = "PQ_UAQFRAGMENT2";
    public final static String READDATA = "readdata.txt";
    private static int Str = 0;

    public static volatile int Num,Count,Av,x=0;
    public static volatile String NumSt,CountSt,xSt,Val,Numo,StatusNum;
    public static volatile int AuxCount;

    public int SNum,PosInt;
    public int TestingPrint;
    public String TestingString;
    public boolean StatusCheck;

    public volatile String IdNum,FullText;
    public volatile String [] Eventos;

    ////////////////////////////////////////////////////////
    //variables en donde se guardan los datos separados
    public volatile String [] DivFullSt;
    public volatile String [] DivFullText;

    public int ArraySizeInt;
//    public static int AFull;
```

```
    public AdapterClasses AClass = new AdapterClasses();

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

    public static   ArrayList<DataSetPreset_T2> SaveLog_T2 = new ArrayList<DataSetPreset_T2>();

    public static List<String> OpcionesList_T1= new ArrayList<String>();
    public static List<String> OpcionesList_T2= new ArrayList<String>();
    public static List<String> OpcionesList_T3= new ArrayList<String>();

    public String PageString;


    @Override
    public void onPauseFragment() {
        Log.i(TAG, "onPauseFragment()");
    }

    @Override
    public void onResumeFragment() {
        Log.i(TAG, "onResumeFragment()");
        PQ_UAQ_V7.F2_Flag=1;
//      PQ_UAQ_V7.F2_Flag=1;
    }
```

```
@Override
public View onCreateView(LayoutInflater inflater, ViewGroup container, Bundle savedInstanceState) {
    View Fragment2V = inflater.inflate(R.layout.activity_pq__uaq__v7, container, false);

    Button BtnNuevo = (Button)Fragment2V.findViewById(R.id.F4BtnNuevoID);

    ViewTreeObserver Vtree=Fragment2V.getViewTreeObserver();
    Vtree.addOnGlobalLayoutListener(new ViewTreeObserver.OnGlobalLayoutListener() {
      @Override
      public void onGlobalLayout() {
        if (PQ_UAQ_V7.F3_Flag == 1) {
          PQ_UAQ_V7.plot_thread.setRunning(false);
          PQ_UAQ_V7.plot_thread.interrupt();//Cause blocking methods like Thread.sleep() to end with an Interrupted exception
          try {
            PQ_UAQ_V7.plot_thread.join();//Wait until the thread is completed
          } catch (InterruptedException e) {
            e.printStackTrace();
          }
        }
      }
    });

    Flush();
```

```
WriteInTxt("2", EDITORDEUS);
CreatePreset();

SNum = ReadSingleInfo(EDITORDEUS);
if (SNum == 0) {
}
SNum = ReadSingleInfo(EDITORDEUS);
readFileInEditor();
SeparandoyProbando();
readFileInEditor();
Separador_f1();

CreateGroupList(OpcionesList_T1,OpcionesList_T2);
ListAdapter ADAPTER = new PQ_LvReg_Adapter(getActivity().getApplicationContext(),
OpcionesList_T1, OpcionesList_T2);
ListView ListaEventos = (ListView)Fragment2V.findViewById(R.id.LvData_Id);
ListaEventos.setAdapter(ADAPTER);

ListaEventos.setOnItemClickListener(
    new AdapterView.OnItemClickListener() {
      @Override
      public void onItemClick(AdapterView<?> parent, View view, int position, long id) {

        PQ_UAQ_V7.plot_thread.setRunning(false);
```

```
                PQ_UAQ_V7.plot_thread.interrupt();//Cause blocking methods like Thread.sleep() to
end with an Interrupted exception
                try {
                   PQ_UAQ_V7.plot_thread.join();//Wait until the thread is completed
                } catch (InterruptedException e) {
                   e.printStackTrace();
                }

                String food = String.valueOf(parent.getItemAtPosition(position));

                WriteInTxt(String.valueOf(position), EDITORDEUS);
                WriteInTxt("1",READDATA);
                Intent NoteIntent = new Intent(getActivity().getApplicationContext(),
SingleBitacora.class);
                startActivity(NoteIntent);
              }
            }
      );

      BtnNuevo.setOnClickListener(new View.OnClickListener() {
         @Override
         public void onClick(View view) {
            int SnumeroAuzx;
            savingInstanceMax();
```

```
            SnumeroAuzx=SNum+1;
            WriteInTxt(String.valueOf(SnumeroAuzx), EDITORDEUS);
            WriteInTxt("2",READDATA);
            ((PQ_UAQ_V7) getActivity()).STimeVars();

PQ_UAQ_V7.BFechaDBuild=PQ_UAQ_V7.Year+"/"+PQ_UAQ_V7.Month+"/"+PQ_UAQ_V7.Day;

            PassingData();

            Intent NoteIntent = new Intent(getActivity().getApplicationContext(), SingleBitacora.class);
            startActivity(NoteIntent);

//              Toast.makeText(MainActivity.this, food, Toast.LENGTH_LONG).show();
          }
       });

       return Fragment2V;
    }

    public void savingInstanceMax(){
       PQ_UAQ_V7.BValorApA=String.valueOf(PQ_UAQ_V7.IndiMaxVal[0]);
       PQ_UAQ_V7.BValorApB=String.valueOf(PQ_UAQ_V7.IndiMaxVal[1]);
       PQ_UAQ_V7.BValorApC=String.valueOf(PQ_UAQ_V7.IndiMaxVal[2]);
       PQ_UAQ_V7.BValorApN=String.valueOf(PQ_UAQ_V7.IndiMaxVal[3]);
       PQ_UAQ_V7.BValorVpA=String.valueOf(PQ_UAQ_V7.IndiMaxVal[4]);
```

```
   PQ_UAQ_V7.BValorVpB=String.valueOf(PQ_UAQ_V7.IndiMaxVal[5]);
   PQ_UAQ_V7.BValorVpC=String.valueOf(PQ_UAQ_V7.IndiMaxVal[6]);

   PQ_UAQ_V7.BValorArmsA=String.valueOf(PQ_UAQ_V7.IndiRmsVal[0]);
   PQ_UAQ_V7.BValorArmsB=String.valueOf(PQ_UAQ_V7.IndiRmsVal[1]);
   PQ_UAQ_V7.BValorArmsC=String.valueOf(PQ_UAQ_V7.IndiRmsVal[2]);
   PQ_UAQ_V7.BValorArmsN=String.valueOf(PQ_UAQ_V7.IndiRmsVal[3]);
   PQ_UAQ_V7.BValorVrmsA=String.valueOf(PQ_UAQ_V7.IndiRmsVal[4]);
   PQ_UAQ_V7.BValorVrmsB=String.valueOf(PQ_UAQ_V7.IndiRmsVal[5]);
   PQ_UAQ_V7.BValorVrmsC=String.valueOf(PQ_UAQ_V7.IndiRmsVal[6]);
}


public void CreateGroupList(List ListaT1,List ListaT2){
   for(x=0; x<SNum; x++) {
      ListaT1.add(SaveLog_T2.get(x).TITULO);
      ListaT2.add(SaveLog_T2.get(x).FechaDBuild);
   }
}

public void Separador_f1(){
   if(FullText!=null){
      DivFullText=FullText.split(";");
      Eventos=new String [Num];
```

```
        for(x=0; x<Num; x++) {
           Eventos[x]=DivFullText[x].trim();
        }

    }
}

public void SeparandoyProbando(){
    IdNum=DivFullSt[0].trim();
    FullText=DivFullSt[1].trim();
    List<String> ArraySets= Arrays.asList(DivFullSt);
    ArraySizeInt= ArraySets.size();
    for(int x=0; x<ArraySizeInt+1; x++) {
       SaveLog_T2.add(new DataSetPreset_T2());
    }
    for(int x=0; x<ArraySets.size(); x++) {
       SaveLog_T2.get(x).LogSaveSet(ArraySets.get(x));
    }

}

public void WriteInTxt(String DataSt, String textArch){
    try {
       OutputStreamWriter out= new
OutputStreamWriter(getActivity().openFileOutput(textArch,0));
```

```
            out.write(DataSt);
            out.close();
        }
        catch (Throwable t) {
            Toast.makeText(getActivity().getApplicationContext(), "Exception: " + t.toString(),
Toast.LENGTH_LONG).show();
        }
    }

    public void Flush(){
        int Count=0;
        CountSt=String.valueOf(Count);
        try {
            OutputStreamWriter out= new
OutputStreamWriter(getActivity().openFileOutput(EDITORMAINLOG, 0));
            out.flush();
            out.close();
        }
        catch (Throwable t) {
//          Toast.makeText(this, "Exception: " + t.toString(), Toast.LENGTH_LONG).show();
        }
    }

    public int ReadSingleInfo(String TextFile)
    {
```

```
int CacheData=0;
try {
   InputStream InData = getActivity().openFileInput(TextFile);
   if (InData != null) {
      InputStreamReader TmpData=new InputStreamReader(InData);
      BufferedReader ReaderData=new BufferedReader(TmpData);
      String StrData;
      StringBuilder buf=new StringBuilder();
      while ((StrData = ReaderData.readLine()) != null){
         buf.append(StrData);
      }
      InData.close();
      String CacheSt=buf.toString();
      CacheData = Integer.parseInt(CacheSt, 10);
   }else{


   }
}
catch (java.io.FileNotFoundException e) {
// that's OK, we probably haven't created it yet
}
catch (Throwable t) {
   Toast.makeText(getActivity().getApplicationContext(),  "Exception:  "+TextFile+  t.toString(),
Toast.LENGTH_LONG).show();
}
```

```
    return CacheData;
  }

  public void CreatePreset(){
    WriteInTxt(
            "DIspositivo/IP" +"," + "Fecha de creacion"+","+ "Fecha de Sincronizacion"+","+ "Fecha de Inicio"+","+

            "Valor Pico V A"+","+ "Valor Pico V B"+","+ "Valor Pico V C"+","+ "Valor Pico A A"+","+ "Valor Pico A B"+","+ "Valor Pico A C"+","+ "Valor Pico A N"+","+

            "Valor Rms V A"+","+ "Valor Rms V B"+","+ "Valor Rms V C"+","+ "Valor Rms A A"+","+ "Valor Rms A B"+","+ "Valor Rms A C"+","+ "Valor Rms A N"+","+

            "Valor max V A"+","+ "Valor max V B"+","+ "Valor max V C"+","+ "Valor max A A"+","+ "Valor max A B"+","+ "Valor max A C"+","+ "Valor max A N"+","+

            "Valor min V A"+","+ "Valor min V B"+","+ "Valor min V C"+","+ "Valor min A A"+","+ "Valor min A B"+","+ "Valor min A C"+","+ "Valor min A N"+","+

            "Valor media V A"+","+ "Valor media V B"+","+ "Valor media V C"+","+ "Valor media A A"+","+ "Valor media A B"+","+ "Valor media A C"+","+ "Valor media A N"+","+"Texto"+";"
            +
            "PQ_UAQ_00/20:15:04:20:35:82" +"," + "01/10/2016:12:00:00.000"+","+ "01/10/2016:12:00:00.000"+","+"01/10/2016:12:00:00.000"+","+
```

```
            "120"+","+ "120"+","+ "120"+","+ "4"+","+ "4"+","+ "4"+","+ "1"+","+

            "116"+","+ "116"+","+ "116"+","+ "4"+","+ "4"+","+ "4"+","+ "1"+","+

            "125"+","+ "125"+","+ "125"+","+ "5"+","+ "5"+","+ "5"+","+ "1"+","+

            "-125"+","+ "-125"+","+ "-125"+","+ "-5"+","+ "-5"+","+ "-5"+","+ "-1"+","+

            "50"+","+ "50"+","+ "50"+","+ "50"+","+ "50"+","+ "50"+","+ "50"+","+"Texto"+";"
        ,EDITORMAINLOG);
  }

  public void PassingData(){
    FlushSave();                                        ///fecha de sincronozacion

WriteInTxt(PQ_UAQ_V7.MACSAVE+","+PQ_UAQ_V7.DEVICESAVE+","+PQ_UAQ_V7.BFechaDBuild+
","+"07/06/2016:20:30:13.923"+","+"07/06/2016:20:30:13.923"+","+
        PQ_UAQ_V7.RMSSAVE[0]+","+
        PQ_UAQ_V7.RMSSAVE[1]+","+
        PQ_UAQ_V7.RMSSAVE[2]+","+
        PQ_UAQ_V7.RMSSAVE[3]+","+
        PQ_UAQ_V7.RMSSAVE[4]+","+
        PQ_UAQ_V7.RMSSAVE[5]+","+
        PQ_UAQ_V7.RMSSAVE[6]+","+
        PQ_UAQ_V7.MAXSAVE[0]+","+
```

```
            PQ_UAQ_V7.MAXSAVE[1]+","+
            PQ_UAQ_V7.MAXSAVE[2]+","+
            PQ_UAQ_V7.MAXSAVE[3]+","+
            PQ_UAQ_V7.MAXSAVE[4]+","+
            PQ_UAQ_V7.MAXSAVE[5]+","+
            PQ_UAQ_V7.MAXSAVE[6]+","
            ,PASIINGDATA);
    }

    public void FlushSave(){
        int Count=0;
        CountSt=String.valueOf(Count);
        try {
            OutputStreamWriter out= new
OutputStreamWriter(getActivity().openFileOutput(PASIINGDATA, 0));
            out.flush();
            out.close();
        }
        catch (Throwable t) {
//          Toast.makeText(this, "Exception: " + t.toString(), Toast.LENGTH_LONG).show();
        }
    }

    public void readFileInEditor()
    {
```

```
        try {
           InputStream in = getActivity().openFileInput(EDITORMAINLOG);
           if (in != null) {
              InputStreamReader tmp=new InputStreamReader(in);
              BufferedReader reader=new BufferedReader(tmp);
              String str;
              StringBuilder buf=new StringBuilder();
              while ((str = reader.readLine()) != null){
                 buf.append(str);
              }
              in.close();
              NumSt = buf.toString();
              DivFullSt = NumSt.split(";");
              Num = Integer.parseInt(IdNum, 10);
           }else{
              StatusCheck=true;
              WriteInTxt("PQ_UAQ_01" + "," + "3" + "," + "4" +","+ "1" + "," + "2" + "," + "1"  + ";"
                    +
                    "PQ_UAQ_02" + "," + "3" + "," + "4" +","+ "1" + "," + "2" + "," + "1"  + ";"
                    ,EDITORMAINLOG);
           }
        }
        catch (java.io.FileNotFoundException e) {
    // that's OK, we probably haven't created it yet
        }
```

```
        catch (Throwable t) {
          Toast.makeText(getActivity().getApplicationContext(), "Exception: " + t.toString(), Toast.LENGTH_LONG).show();
        }
    }

    public static PQ_UAQFRAGMENT2 newInstance(String text) {
        PQ_UAQFRAGMENT2 f2 = new PQ_UAQFRAGMENT2();
        Bundle b = new Bundle();
        b.putString("msg", text);
        f2.setArguments(b);
        return f2;
    }
}
```

7.17. PQ_UAQFRAGMENT3

```
import android.graphics.Color;
import android.os.Bundle;
import android.support.annotation.Nullable;
import android.support.v4.app.Fragment;
import android.util.Log;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.view.ViewTreeObserver;
```

```
import android.widget.TextView;

/**
 * Created by Pablo on 8/18/2015.
 */

public class PQ_UAQFRAGMENT3 extends Fragment implements FragmentLifecycle {

    public static final String TAG = "PQ_UAQFRAGMENT3";
    // Run/Pause status
    private boolean bReady = false;

    public static final boolean D = true;
    private static int Str = 0;
    public static final int Mensaje_Estado_Cambiado = 1;
    public static final int Mensaje_Leido = 2;
    public static final int Mensaje_Escrito = 3;
    public static final int Mensaje_Nombre_Dispositivo = 4;
    public static final int Mensaje_TOAST = 5;
    public static final int MESSAGE_Desconectado = 6;
    public static final int REQUEST_ENABLE_BT = 7;
    public static final String DEVICE_NAME = "HC-06_1234";//nombre del dispositivo_codigo
    public static final String TOAST = "toast";
    private TextView F3TxtV1,F3TxtV2,F3TxtV3,F3TxtA1,F3TxtA2,F3TxtA3,F3TxtAN;
```

```
    public ViewGroup GroupView;



    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container, Bundle
savedInstanceState) {
        View FragmentV3 = inflater.inflate(R.layout.activity_scope, container, false);

//      GroupView=container;

        PQ_UAQ_V7.AChPaint.add(PQ_UAQ_V7.V1_color);
        PQ_UAQ_V7.AChPaint.add(PQ_UAQ_V7.V2_color);
        PQ_UAQ_V7.AChPaint.add(PQ_UAQ_V7.V3_color);
        PQ_UAQ_V7.AChPaint.add(PQ_UAQ_V7.V4_color);
        PQ_UAQ_V7.AChPaint.add(PQ_UAQ_V7.A1_color);
        PQ_UAQ_V7.AChPaint.add(PQ_UAQ_V7.A2_color);
        PQ_UAQ_V7.AChPaint.add(PQ_UAQ_V7.A3_color);
        PQ_UAQ_V7.AChPaint.add(PQ_UAQ_V7.A4_color);

        for (int x = 0; x < 8; x++){
            switch (PQ_UAQFRAGMENT1.Tv.get(x).Color){
                case "Red":
                    PQ_UAQ_V7.AChPaint.get(x).setColor(Color.rgb(255, 0, 0));
                    break;
```

```
        case "Blue":
            PQ_UAQ_V7.AChPaint.get(x).setColor(Color.rgb(0, 0, 255));
            break;
        case "Green":
            PQ_UAQ_V7.AChPaint.get(x).setColor(Color.rgb(0, 255, 0));
            break;
        case "Cyan":
            PQ_UAQ_V7. AChPaint.get(x).setColor(Color.rgb(0, 255, 255));
            break;
        case "Magenta":
            PQ_UAQ_V7.AChPaint.get(x).setColor(Color.rgb(255, 0, 255));
            break;
        case "White":
            PQ_UAQ_V7.AChPaint.get(x).setColor(Color.rgb(255, 255, 255));
            break;
    }
}

ViewTreeObserver Vtree=FragmentV3.getViewTreeObserver();
Vtree.addOnGlobalLayoutListener(new ViewTreeObserver.OnGlobalLayoutListener() {
    @Override
    public void onGlobalLayout() {
        if (PQ_UAQ_V7.F3_Flag == 1) {
        }
    }
```

```
        });

        return FragmentV3;
    }

    @Override
    public void onResume() {
        super.onResume();
//      Fragment.instantiate(getView().getContext(),null);
//      FragmentVIew=View.inflate(R.layout.activity_first, GroupView);
    }

    @Nullable
    @Override
    public View getView() {
        return super.getView();
    }



    public static PQ_UAQFRAGMENT3 newInstance(String text) {
        PQ_UAQFRAGMENT3 f = new PQ_UAQFRAGMENT3();
        Bundle b = new Bundle();
        b.putString("msg", text);
        f.setArguments(b);
```

```
        return f;
    }

//    public void CreateGroupList(List Lista,int numero) {
//        for (int x = 0; x<PQ_UAQ_V7.SNum; x++) {
//            Lista.add(PQ_UAQ_V7.SaveLog.get(x).Nombre);
//        }
//    }

    @Override
    public void onPauseFragment() {
        Log.i(TAG, "onPauseFragment()");
        PQ_UAQ_V7.F3_Flag=0;
    }

    @Override
    public void onResumeFragment() {
        Log.i(TAG, "onResumeFragment()");
        PQ_UAQ_V7.SyncFlag=true;
        PQ_UAQ_V7.F3_Flag=1;
        Str = 0;
    }
}
```

7.18. PQ_UAQFRAGMENT_3

```
import android.graphics.Color;
import android.os.Bundle;
import android.support.v4.app.Fragment;
import android.util.Log;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.view.ViewTreeObserver;
import android.widget.Button;
import android.widget.CompoundButton;
import android.widget.ListView;
import android.widget.TextView;
import android.widget.ToggleButton;

import java.io.BufferedReader;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.io.OutputStreamWriter;
import java.util.ArrayList;
import java.util.Arrays;
import java.util.List;
```

```
/**
 * Created by Pablo on 5/17/2016.
 */
public class PQ_UAQFRAGMENT_3 extends Fragment implements FragmentLifecycle {
    public final static String EDITORMAIN="StoreFacts.txt";
    public final static String EDITORLIST="ListNum.txt";
    public final static String SAVEDATAIND="Indice.txt";
    public static SetColorOfClassSets CSet=new SetColorOfClassSets();
    private Button BtnEditar, BtnBorrar;
    private ListView ListOpciones;
    public static final String TAG = "PQ_UAQFRAGMENT_N3";

    public static volatile int Num,Count,Av,x=0;
    public static volatile String NumSt,CountSt,xSt,Val,Numo,StatusNum;
    public static volatile int AuxCount;

    public int SNum,PosInt;
    public boolean StatusCheck;

    public volatile String IdNum,FullText;
    public volatile String [] Eventos;
    //////los numeros son para clasificar el orden de los valores de id, fehca, hora, etc
    public volatile String [] IdEvnt_1;
    public volatile String [] Dia_2;
    public volatile String [] Mes_3;
```

```
    public volatile String [] Anio_4;
    public volatile String [] Hora_5;
    public volatile String [] Min_6;
    public volatile String [] seg_7;
    public volatile String [] Text_8;
    ////////////////////////////////////

    ////////////////////////////////////////////////////////////
    //variables en donde se guardan los datos separados
    public volatile String [] DivFullSt;
    public volatile String [] DivFullText;
    public volatile String [] DivEvVar;
    public volatile String [][] DivEventos = new String [8][];

    public int V0000;
    public int V0001;
    public int V0002;
    public int V0003;
//    public static int VFull;

    public int A0000;
    public int A0001;
    public int A0002;
    public int A0003;
```

```
    public int ArraySizeInt;
//    public static int AFull;

    public AdapterClasses AClass = new AdapterClasses();

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

    public static List<String> OpcionesList= new ArrayList<String>();

    public String PageString;

    @Override
    public void onPauseFragment() {
        Log.i(TAG, "onPauseFragment()");
    }

    @Override
    public void onResumeFragment() {
        Log.i(TAG, "onResumeFragment()");
        PQ_UAQ_V7.F2_Flag=1;

    }

    @Override
```

```
  public View onCreateView(LayoutInflater inflater, ViewGroup container, Bundle savedInstanceState) {
      final View Fragment3NV = inflater.inflate(R.layout.activity_intentf_3aux, container, false);

      ToggleButton BtnOpcionV1 = (ToggleButton) Fragment3NV.findViewById(R.id.F2_TBtnV1ID);
      ToggleButton BtnOpcionV2 = (ToggleButton) Fragment3NV.findViewById(R.id.F2_TBtnV2ID);
      ToggleButton BtnOpcionV3 = (ToggleButton) Fragment3NV.findViewById(R.id.F2_TBtnV3ID);

      ToggleButton BtnOpcionA1 = (ToggleButton) Fragment3NV.findViewById(R.id.F2_TBtnA1ID);
      ToggleButton BtnOpcionA2 = (ToggleButton) Fragment3NV.findViewById(R.id.F2_TBtnA2ID);
      ToggleButton BtnOpcionA3 = (ToggleButton) Fragment3NV.findViewById(R.id.F2_TBtnA3ID);
      ToggleButton BtnOpcionA4 = (ToggleButton) Fragment3NV.findViewById(R.id.F2_TBtnANID);

      BtnOpcionV1.setText("VA_OFF");
      BtnOpcionV1.setTextOn("V1_ON");
      BtnOpcionV1.setTextOff("V1_OFF");

      BtnOpcionV2.setText("VB_OFF");
      BtnOpcionV2.setTextOn("V2_ON");
      BtnOpcionV2.setTextOff("V2_OFF");

      BtnOpcionV3.setText("VC_OFF");
      BtnOpcionV3.setTextOn("V3_ON");
      BtnOpcionV3.setTextOff("V3_OFF");
```

```
        BtnOpcionA1.setText("IA_OFF");
        BtnOpcionA1.setTextOn("IA_ON");
        BtnOpcionA1.setTextOff("IA_OFF");

        BtnOpcionA2.setText("IB_OFF");
        BtnOpcionA2.setTextOn("IB_ON");
        BtnOpcionA2.setTextOff("IB_OFF");

        BtnOpcionA3.setText("IC_OFF");
        BtnOpcionA3.setTextOn("IC_ON");
        BtnOpcionA3.setTextOff("IC_OFF");

        BtnOpcionA4.setText("IN_OFF");
        BtnOpcionA4.setTextOn("IN_ON");
        BtnOpcionA4.setTextOff("IN_OFF");

//      Button BtnNuevo = (Button)Fragment3NV.findViewById(R.id.F2BtnSaveId);
//      Button BtnGuardarComo = (Button)Fragment3NV.findViewById(R.id.F2BtnSaveAsID);
//
//      final ListView ListaEventos = (ListView)Fragment3NV.findViewById(R.id.F2lstVID);
//
//      final EditText Nombres = (EditText)Fragment3NV.findViewById(R.id.F2TxtNombresID);

        final TextView TxtPrintV1 = (TextView) Fragment3NV.findViewById(R.id.F2w2ChVOutptID);
        final TextView TxtPrintA1 = (TextView) Fragment3NV.findViewById(R.id.F2w2ChAOutptID);
```

```
        Flush();
        WriteInTxt("2",EDITORLIST);
        WriteInTxt("PQ_UAQ" + "," + "3" + "," + "4" + "," + "1" + "," + "2" + "," + "1" + ";"
                +
                "Fluke" + "," + "3" + "," + "4" + "," + "1" + "," + "2" + "," + "1" + ";"
                ,EDITORMAIN);

        SNum=ReadSingleInfo(EDITORLIST);
        if(SNum==0){
//          WriteInTxt("2",EDITORLIST);
        }

        SNum=ReadSingleInfo(EDITORLIST);
        readFileInEditor();
        SeparandoyProbando();

        readFileInEditor();
        Separador_f1();

        BtnOpcionV1.setOnCheckedChangeListener(new
CompoundButton.OnCheckedChangeListener() {
            @Override
            public void onCheckedChanged(CompoundButton buttonView, boolean isChecked) {
                if (isChecked) {
```

```
            V0000 = 1;
          } else {
            V0000 = 0;
          }
          PQ_UAQ_V7.VFull = AClass.SaveVOp(V0000, V0001, V0002, 0);
          TxtPrintV1.setText("Canales de Voltaje : " + String.valueOf(PQ_UAQ_V7.VFull));
        }
      });

      BtnOpcionV2.setOnCheckedChangeListener(new
CompoundButton.OnCheckedChangeListener() {
        public void onCheckedChanged(CompoundButton buttonView, boolean isChecked) {
          if (isChecked) {
            V0001 = 1;
          } else {
            V0001 = 0;
          }
          PQ_UAQ_V7.VFull = AClass.SaveVOp(V0000, V0001, V0002, 0);
          TxtPrintV1.setText("Canales de Voltaje : " + String.valueOf(PQ_UAQ_V7.VFull));
        }
      });

      BtnOpcionV3.setOnCheckedChangeListener(new
CompoundButton.OnCheckedChangeListener() {
        public void onCheckedChanged(CompoundButton buttonView, boolean isChecked) {
```

```
            if (isChecked) {
               V0002 = 1;
            } else {
               V0002 = 0;
            }
            PQ_UAQ_V7.VFull = AClass.SaveVOp(V0000, V0001, V0002, 0);
            TxtPrintV1.setText("Canales de Voltaje : " + String.valueOf(PQ_UAQ_V7.VFull));

         }
      });


      BtnOpcionA1.setOnCheckedChangeListener(new
CompoundButton.OnCheckedChangeListener() {
         public void onCheckedChanged(CompoundButton buttonView, boolean isChecked) {
            if (isChecked) {
               A0000 = 1;
            } else {
               A0000 = 0;
            }PQ_UAQ_V7.AFull=AClass.SaveAOp(A0000, A0001, A0002, A0003);
            TxtPrintA1.setText("Canales de Corriente:"+ String.valueOf(PQ_UAQ_V7.AFull));

         }
      });
```

```
    BtnOpcionA2.setOnCheckedChangeListener(new
CompoundButton.OnCheckedChangeListener() {
        public void onCheckedChanged(CompoundButton buttonView, boolean isChecked) {
          if (isChecked) {
            A0001 = 1;
          } else {
            A0001 = 0;
          }PQ_UAQ_V7.AFull=AClass.SaveAOp(A0000, A0001, A0002, A0003);
          TxtPrintA1.setText("Canales de Corriente:"+ String.valueOf(PQ_UAQ_V7.AFull));
        }
    });


    BtnOpcionA3.setOnCheckedChangeListener(new
CompoundButton.OnCheckedChangeListener() {
        public void onCheckedChanged(CompoundButton buttonView, boolean isChecked) {
          if (isChecked) {
            A0002 = 1;
          } else {
            A0002 = 0;
          }
          PQ_UAQ_V7.AFull = AClass.SaveAOp(A0000, A0001, A0002, A0003);
          TxtPrintA1.setText("Canales de Corriente:" + String.valueOf(PQ_UAQ_V7.AFull));


        }
    });
```

```
    BtnOpcionA4.setOnCheckedChangeListener(new
CompoundButton.OnCheckedChangeListener() {
        public void onCheckedChanged(CompoundButton buttonView, boolean isChecked) {
            if (isChecked) {
                A0003 = 1;
            } else {
                A0003 = 0;
            }
            PQ_UAQ_V7.AFull = AClass.SaveAOp(A0000, A0001, A0002, A0003);
            TxtPrintA1.setText("Canales de Corriente:" + String.valueOf(PQ_UAQ_V7.AFull));
        }
    });


    ViewTreeObserver Vtree=Fragment3NV.getViewTreeObserver();
    Vtree.addOnGlobalLayoutListener(new ViewTreeObserver.OnGlobalLayoutListener() {
        @Override
        public void onGlobalLayout() {
            if (PQ_UAQ_V7.F3_Flag == 1) {
                PQ_UAQ_V7.plot_thread.setRunning(false);
                PQ_UAQ_V7.plot_thread.interrupt();//Cause blocking methods like Thread.sleep() to
end with an Interrupted exception
                try {
                    PQ_UAQ_V7.plot_thread.join();//Wait until the thread is completed
                } catch (InterruptedException e) {
```

```
                e.printStackTrace();
            }
        }
    }
});

    return Fragment3NV;
}

////este metodo crea los elementos apartir de los datos optenidos del Archivo txt
    public void CreateGroupList(List Lista,int numero) {
        for (x = 0; x<SNum; x++) {
            Lista.add(PQ_UAQ_V7.SaveLog.get(x).Nombre);
        }
    }

    public void LeeyendoDatos(){
        SNum=ReadSingleInfo(EDITORLIST);
        readFileInEditor();
        SeparandoyProbando();

        readFileInEditor();
        Separador_f1();
    }
```

```
    public void Separador_f1(){
        if(FullText!=null){
            DivFullText=FullText.split(";");
            Eventos=new String [Num];

            for(x=0; x<Num; x++) {
                Eventos[x]=DivFullText[x].trim();
            }

//          DivEventos[0]=Eventos[0].split(",");
//          IdEvnt_1[0]=DivEventos[0][0].trim();
        }
    }


    public void Init(int num_1){
        IdEvnt_1=new String [num_1];
        Dia_2   =new String [num_1];
        Mes_3   =new String [num_1];
        Anio_4  =new String [num_1];
        Hora_5  =new String [num_1];
        Min_6   =new String [num_1];
        seg_7   =new String [num_1];
        Text_8  =new String [num_1];
```

```
}

public void SeparandoyProbando(){
   IdNum=DivFullSt[0].trim();
   FullText=DivFullSt[1].trim();
   List<String> ArraySets= Arrays.asList(DivFullSt);
   ArraySizeInt= ArraySets.size();
   for(int x=0; x<ArraySizeInt+1; x++) {
      PQ_UAQ_V7.SaveLog.add(new DataSetPreset_T1());
   }
   for(int x=0; x<ArraySets.size(); x++) {
      PQ_UAQ_V7.SaveLog.get(x).DataSaveSet(ArraySets.get(x));
   }
}



public void WriteInTxt(String DataSt, String textArch){
   try {
      OutputStreamWriter out= new
OutputStreamWriter(getActivity().openFileOutput(textArch,0));
      out.write(DataSt);
      out.close();
   }
   catch (Throwable t) {
```

```
//                  Toast.makeText(getActivity().getApplicationContext(), "Exception: " + t.toString(),
Toast.LENGTH_LONG).show();
      }
   }

   public void Flush(){
      int Count=0;
      CountSt=String.valueOf(Count);
      try {
         OutputStreamWriter                    out=                    new
OutputStreamWriter(getActivity().openFileOutput(EDITORMAIN, 0));
         out.flush();
         out.close();
      }
      catch (Throwable t) {
//          Toast.makeText(this, "Exception: " + t.toString(), Toast.LENGTH_LONG).show();
      }
   }

   public int ReadSingleInfo(String TextFile)
   {
      int CacheData=0;
      try {
         InputStream InData = getActivity().openFileInput(TextFile);
         if (InData != null) {
```

```
            InputStreamReader TmpData=new InputStreamReader(InData);
            BufferedReader ReaderData=new BufferedReader(TmpData);
            String StrData;
            StringBuilder buf=new StringBuilder();
            while ((StrData = ReaderData.readLine()) != null){
               buf.append(StrData);
            }
            InData.close();
            String CacheSt=buf.toString();
            CacheData = Integer.parseInt(CacheSt, 10);
          }else{


          }
      }
      catch (java.io.FileNotFoundException e) {
// that's OK, we probably haven't created it yet
      }
      catch (Throwable t) {
//          Toast.makeText(getActivity().getApplicationContext(), "Exception: "+TextFile+ t.toString(),
Toast.LENGTH_LONG).show();
      }


      return CacheData;
   }
```

```
public void readFileInEditor()
{
   try {
      InputStream in = getActivity().openFileInput(EDITORMAIN);
      if (in != null) {
         InputStreamReader tmp=new InputStreamReader(in);
         BufferedReader reader=new BufferedReader(tmp);
         String str;
         StringBuilder buf=new StringBuilder();
         while ((str = reader.readLine()) != null){
            buf.append(str);
         }
         in.close();
         NumSt = buf.toString();
         DivFullSt = NumSt.split(";");
         Num = Integer.parseInt(IdNum, 10);
      }else{
         StatusCheck=true;
         WriteInTxt("PQ_UAQ" + "," + "3" + "," + "4" +","+ "1" + "," + "2" + "," + "1"  + ";"
               +
               "Fluke" + "," + "3" + "," + "4" +","+ "1" + "," + "2" + "," + "1"  + ";"
               ,EDITORMAIN);
      }
   }
   catch (java.io.FileNotFoundException e) {
```

```
// that's OK, we probably haven't created it yet
      }
      catch (Throwable t) {
//                  Toast.makeText(getActivity().getApplicationContext(), "Exception: " + t.toString(), Toast.LENGTH_LONG).show();
      }
   }

   public void C(){
      CSet.Config_PQUAQ(Color.BLACK, "Red", "Red", "Red", "Red", "Red", "Red", "Red", "Red");
   }

   public void ModTech(){
//       expandableListTitle = new ArrayList<String>(expandableListDetail.keySet());
   }
}
```

7.2 ARTICULO


UNIVERSIDAD AUTÓNOMA DE QUERÉTARO
12° CONGRESO INTERNACIONAL DE INGENIERÍA
La UNIVERSIDAD AUTÓNOMA DE QUERÉTARO
a través de la
FACULTAD DE INGENIERÍA,
otorga la presente
CONSTANCIA
A:
Pablo Cesar Ramírez Echevarria
Por su participación como asistente.
En la duodécima edición del
Congreso Internacional de Ingeniería
realizado del 01 al 6 de Mayo del 2016
en la ciudad de Santiago de Querétaro, México.
Fi
IEEE
SECCIÓN QUERÉTARO
CONCYTEQ
CONACYT
Dr. Aurelio Domínguez González
Director
Dra. Karen Esquivel Escalante
Coordinadora del Congreso
Internacional de Ingeniería

Mobile application for monitoring and visualization of data applied to power quality systems

P C. Ramirez-Echeverria
Facultad de Ingeniería Querétaro
Universidad Autónoma de Querétaro.
San Juan del Rio, México.
pramirez@hspdigital.org

L. Morales-Velazquez
Facultad de Ingeniería Querétaro
Universidad Autónoma de Querétaro.
San Juan del Rio, México

R. de J. Romero-Troncoso
División de Ingenierías Campus Irapuato
Universidad de Guanajuato.
Salamanca, México.

R. A. Osornino-Rios.
Facultad de Ingeniería Querétaro
Universidad Autónoma de Querétaro.
San Juan del Rio, México

D. Morinigo Sotelo
Departamento de Ingeniería Eléctrica
Universidad de Valladolid
Valladolid, España

***Abstract*— This paper presents the development of a mobile application for the android platform to perform sampling, Synchronization, transmission and control over a data-logger system for power quality analysis. The main function of this system is the sampling process and data storage of voltage and current values taken from the electrical assembly power grid. The power quality system transmits the data via basic Bluetooth radio frequency communication. Afterward, this data gets processed, classified, and displayed in real time.**

Keywords—*Android, Power Quality, Bluetooth, GUI*

I. Introduction

In the last century automation, telecommunications and computer industries have experienced an unprecedented growth. However the new generation of electronics is getting more sensitive to electromagnetic disturbances to the point that any disruption of the supply voltage may have destructive effects on the installed equipment, any disturbance manifested in the voltage, current, and frequency from the standard rate is treated as a power quality (PQ) problem. Power quality covers several aspects including the measuring, analysis and the improvement of the load bus voltage [1]. A lot of the phenomena related to power quality may appear and disappear arbitrarily, because of this finding electric disturbances are more complicated than a simple measurement of an electric parameter. It is necessary to monitor the grid over a long time interval to acquire a big amount of data to get a better idea of what is going on in the power grid [2]. The most typical approach for power quality analysis systems in the industry is having an on-purpose-built monitoring device that takes cares of the plotting, analysis, monitoring and graphic user interface [3]. However, this system is expensive at prices range of 9000 USD with closed software and a lack of versatility on the hardware. There are cases where the number of devices needed to perform a PQ analysis make it economically unreasonable [4]. Therefore, methods that may reduce the price of power quality analyzer (PQA) system are needed. There are several works where the use of a mobile implementation can increase the efficiency of industrial and commercial processes [5-7]. In some cases, several power quality systems get connected to a single power grid to analyze the estate and investigate if a PQ event occurring in one part of the grid can affect other components of the power grid [4]. For this reason, a way to synchronize all the data logging systems and the option to turn on and off the sampling is needed. The motivation for the approach described in this text was to allow on-site technicians to be able to use their everyday mobile phone for wireless diagnostic purposes both in terms of data acquisition as well as signal processing. The PQ system used in this work is a based-field programmable gate array (FPGA) used for data logging. This system can perform a fast and precise data acquisition of the supply voltage and current, it has the capacity to connect to a smartphone to transmit and acquire data wirelessly. In this work the feasibility of using a mobile phone application for signal processing, the monitoring, and synchronizing of a PQ data logging system is presented.

II. Theoretical Background

A. Android mobile device

Thanks to the constant advances in the development of mobile phones, android technology has become more efficient each generation and the amount of sensors and other hardware have increased depending on the necessities of the consumers. Android runs on an open-source Linux Kernel [8], designed from the beginning as an advanced mobile operating system. Nowadays the majority of the people counts with a smartphone, this gives the possibility of having sensors and

data analysis systems on the palm of the hand of the majority of technicians, because of this, the possibility of doing tasks like plotting, graphic interface and data acquisition is possible to perform on a mobile platform and its open philosophy ensures an easy and growing development [9].

B. Bluetooth

Bluetooth is one of the most popular and economic wireless technology, this is because Bluetooth is robust, has low cost, low power consumption and is used worldwide [10]. The communication protocol is defined in the hardware abstraction of the Bluetooth. The pairing and data control are defined by the Bluetooth special interest group (Bluetooth SIG) [11], the packet structure of a Bluetooth transmission gets defined on logical link control and adaptation layer protocol (L2CAP), this define that the max limit of the packet for the serial transmission is 64 kilobytes and it gets recommended that the velocity of transmission do not go over 113200 if the communication gets done in an environment with a lot of physical and radio frequency interferences. Wireless communication via Bluetooth in the Android is managed by the application programming interface (API) Bluetooth, this API exist since the release version number one of Android OS, it permits applications to use the Bluetooth hardware in a mobile phone, to connect to other Bluetooth hardware, scan for other devices and establish R FCOM channels [12].

C. Power Quality Analisis

To have a basic analysis of the state of the data acquisition system indices of power quality $Vpeak$ and $Ipeak$ are acquired in the time domain like the max values in the analyzed interval, as it is shown on (1) and (2) [13].

$$Vpeak = max(V(t)) \tag{1}$$

$$Ipeak = max(I(t)) \tag{2}$$

The root mean square (RMS) are other values that are processed and it is needed to get an average of the values in the channel. Formulas are shown in (3) and (4) [14].

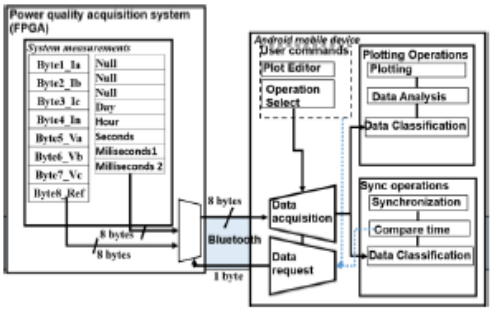
$$Vrms = \frac{1}{\sqrt{2}}\sqrt{\sum_{n=1}^{N} V_n^2} \tag{3}$$

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

III. METODOLOGY

The developed application consist of a synchronization module, a plotting module, and a data acquisition, request and classification modules and the user commands that dictate which operation must be executed. A functional diagram of the monitoring system developed is presented In Fig. 1. The Android mobile device is bonded via Bluetooth to the PQ system. Depending on the control, the instruction sent by the mobile phone the PQ system sends data requested. When the data reach the mobile phone, the ways the data is going to be used depends on the instruction given by the user, this is if the user needs to plot or to synchronize the data-logger FPGA system. Settings such as scale, color, offset, style and name of the signals on the plot are set by the user on the Plot Editor.

Fig. 1. Functional Diagram of the systems used in this paper.



A. User commands

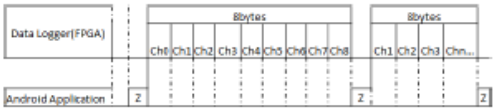
User commands comprehend all the configurations and operations that the user input via the user interface and the *Plot Editor*. In Fig. 3, a flow diagram showing how the *Plot Editor* manages actions is shown. *Plot Editor* is in here in order to change and set values as gains, color, offset. Etc. Actions like deciding to plot or synchronize are selected in this section by the user. This set of commands get implemented primarily by buttons with OnClickListener method events [15] for plotting, synchronization and the *Plot Editor*.

B. Data acquisition ,Data request and data classification

After the connection with the PQ system, data acquisition is administrated by Bluetooth APIs. The data sent by the power quality acquisition system is a total of 8 bytes for each one of the signals acquired from the data logger. This data gets saved on a first in, first out (FIFO) type of buffer declared on the main function like a global buffer and every class has access to a copy of this FIFO when it needs the sampling data. In Fig. 2 a diagram of the packets of the data logger and the Android platform packets transfer is presented. When the Android sends the character "Z" in binary the PQ system sends the data logged sampled values taken in the request moment. In Fig. 1 and Fig. 2, when executing the plotting the data gets separated on channels each one with 1 byte for each point of the plot being drawn. After being divided, each one of the channels gets assigned a label set on the *Plot Editor*. The application can also request for time data when a synchronization is getting requested. In Fig. 3 the data transfer of the synchronization operations is shown, in this process, the data being acquired gets divided into values of time and sent to the

sync operations until the system gets a successful synchronization.

Fig. 2. Data transfer packages between the android application and the PQ data logger.



C.Sync Operations

Sync Operations is the module than the Android application uses in order to get a successful synchronization. In Fig. 3 a diagram of the packages sent is shown. In this diagram, time data is sent and received in binary coded decimal (BCD) in Fig. 3 the composition of the data packages is shown, in this packages the character "t" is used to indicate when does each of the packages ends and begins, the values represented by Y, d, h, m, s, ms1 and ms2 correspond to the values in BCD of the year, day, hour minute, second, the first part of the milliseconds value and the second part of the millisecond value. In Fig 4, the flow diagram of the algorithm used to synchronize is shown. In this figure, Hand is the coordinated universal time (UTC) of the Android phone. Hfpga, is the FPGA time, the character "a" is sent to indicate the beginning of a synchronization to the PQ system, the character "D" is used to set the PQ system time 10 second forward and, character "U" is used to set the time 10 second backwards.

Fig. 3. Data packages in the process of the synchronization.

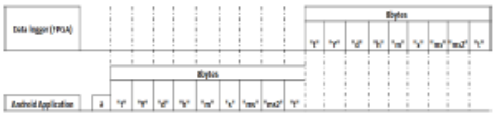
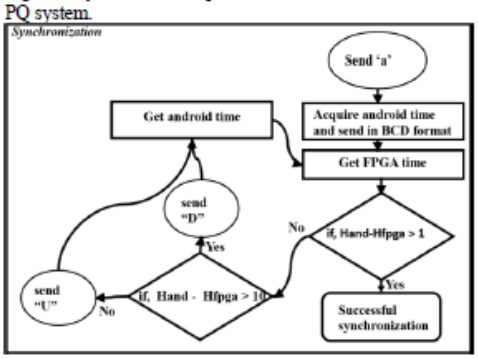


Fig. 4. Synchronization process between Android and the PQ system.



To acquire the UTC time the mobile phone needs to use the method CurrentTimeMillis of the class System [16]. This method always returns UTC time, this measurement of time can be found in the majority of mobile devices and GPS, the method CurrentTimeMillis gives the values of time in milliseconds and the user has to transform or transcribe the data depending on the necessity when an algorithm or the class SimpleDateFormat that permits the user to format and parse dates gets executed. UTC have the precision needed to perform a synchronization and is easy to use for inexperienced users in time formats because the format is almost identical to the local time format. When doing a synchronization, one pulse per second is the minimum to get an acceptable synchronization [17], and precision of the application developed in this work can reach a precision of 1 millisecond. It is worth mentioned that every smartphone has a different internal clock, however, all the Android mobile devices are coordinated or synchronized with the UTC time via internet.

E.Data analisis.

On the data analysis, the RMS and peak values of the voltage and current are obtained using (1), (2), (3) and (4) whith the class math [18], from each one of the channels, and assign a variable to get used in the plot of the signal.

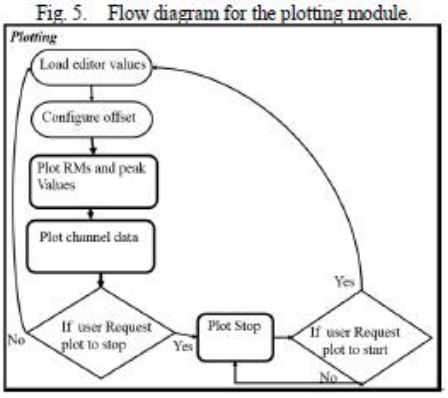
F.Android Paint and Canvas

Classes Canvas [19] and Paint [20] are methods that exist since API number one and are part of the Android Open Source Project [21], this means all the Android devices are able to use these classes, in the Android OS. To draw the screen, you need a valid canvas object by extending the view class. The paint object stores far more than color. The Paint class encapsulates the style and complex color and rendering information that can be applied to a drawable.

G.Plotting

In Fig. 5, the process to plot is presented. When the stream of data gets separated, each channel is painted with Canvas and Paint. Canvas, in this case, is used to determine the view where the plot is going to be draw, Canvas also drawn each one of the points from all the channels via the method drawline [19], plotting the acquired data from the power quality acquisition sampling the values calculated of the RMS and peak, these values are shown using the method drawtext [19]. Paint is used in a dedicated thread, the plot user interface (UI) also has some commands to control when you want to start or stop plotting, the colors chosen to plot each one of the channels and their gains can be modified from the *Plot Editor*.

Fig. 5. Flow diagram for the plotting module.



IV. RESULTS

A. Experimental setup

The data was generated on a dedicated test rig which is presented in Fig. 4, it is comprised of a .74kW induction motor with nominal speed of 3355 RPM.

Fig. 6. Induction motor on the test rig.

The instrumentation in Fig. 5,is comprised of 4 banana cables connected to three phase electric voltage *a*, *b* , c and neutral voltage and 4 AC Current Clamps i200 [22] to get the values of current *a, b, c* and the neutral of the three phase electric power.

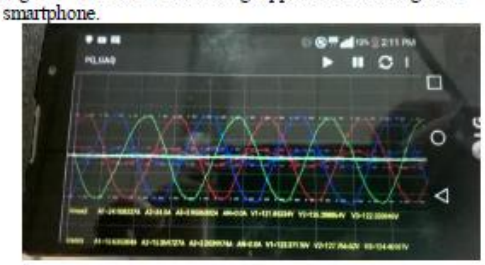
Fig. 7. Instrumentation on the test rig.

The last component of the test rig was the Power Quality data acquisition system PQ_UAQ, this system is an FPGA based system developed by the Universidad Autonoma de Queretaro (UAQ). This main function of this system is to work as a data-logger of the electric events on the motor. It uses an HC-06 Bluetooth to transmit data wirelessly to the mobile phone, and obtains the data through an ADC data acquisition system developed by UAQ. The Android smartphone used in this test was a LG-H420 with Android version 5.01. The test consisted of running the motor, Synchronizing the PQ_UAQ, turning on the data-login in the mobile phone application PQ_UAQ, plotting data from the moment the motor started running and, drawing the data being sent to the Android phone by the PQ_UAQ.

B. Results

This section shows the results of monitoring the power quality system PQ_UAQ by the application installed on a smartphone. At the moment of the test, the sine wave of each voltage on the three-phase power lines connected to the motor could be easily observed, RMS and Peak of the voltage and current during the test were displayed on the screen as shown in Fig. 8 .Each of the Voltage and Current of the three phase connection could be observed, current lag could also be observed. The way the channels displayed look, was edited on the application plot editor by the user. It's worth to mention that currently there are other kinds of android applications to monitor measurement system like fluke connect, however this kind of program doesn't have compatibility with a lot of old or recent Android versions phones and its only possible to use the limitation of only being able to use it with some specific fluke products make it a bad choice to monitor the PQ_UAQ.

Fig. 8. Plot of the monitoring Application working on a smartphone.



V. CONCLUSION

The use of mobile application for monitoring a power quality system enabled that anyone who wanted to monitor the system signal could see in what estate the connection was, problems such as cables disconnected were detected in the moment of the installation and synchronization of the power quality system, this implementation made power quality events easier to detect in the moment of an analysis of the data saved in the

system. Not to mention the use of a mobile application would mean that typical plotting in real time of the power quality signal could be performed without the need for extra equipment which would make the system more costly and less practical. The process of creating the application has proven that the mobile monitoring and synchronizing of a power quality system is possible and it can improve the efficiency of existing systems.

A. Autores y aflicaciones

Pablo Cesar Ramírez Echeverría, masters student of the Maestría en ciencias de la mecatrónica, in the Universidad Autónoma de Querétaro, campus San Juan del Rio.

Acknowledgments

The authors of this work whishes to thanks the Universidad Autónoma de Querétaro, Campus San Juan del Río and the Universidad de Valladolid, España, for facilities granted to carry out this investigation.

This work was partially supported by CONACyT scholarship number 632286 and by projects SEP-CONACyT 222453-2013 and PROMEP 103.5/14/710401.

Partially financed by FOFIUAQ-FIN201613

References

[1] K. Mazlumi, "Power Quality Monitoring", in Power Quality Monitoring, Analysis and Enhancement, A. zobba, Ed. ISBN: 978-953-307-330-9, In Tech, 2011.

[2] G. Găşpăresc, "Methodes of Power Quality Analysis", in Power Quality Monitoring, Analysis and Enhancement, A. zobba, Ed. ISBN: 978-953-307-330-9, In Tech, 2011.

[3] Fluke 430 Series II, Technical Data, Fluke Corporations, 2012.

[4] N. Gupta, A. Swarnkar, and K. R. Niazi, "Distribution network reconfiguration for power quality and reliability improvement using Genetic Algorithms," Int. J. Electr. Power Energy Syst., vol. 54, pp. 664–671, 2014.

[5] P. Rzeszucinski, M. Orman, C. T. Pinto, A. Tkaczyk, and M. Sulowicz, "A signal processing approach to bearing fault detection with the use of a mobile phone," pp. 310–315, 2015.

[6] T. H. Khan, R. Shrestha, K. a. Wahid, and P. Babyn, "Design of a smart-device and FPGA based wireless capsule endoscopic system," *Sensors Actuators A Phys.*, vol. 221, pp. 77–87, 2015.

[7] M. del Rosario, S. Redmond, and N. Lovell, "Tracking the Evolution of Smartphone Sensing for Monitoring Human Movement," *Sensors*, vol. 15, no. 8, pp. 18901–18933, 2015.

[8] B. Pon, T. Seppälä, and M. Kenney, "Android and the demise of operating system-based power: Firm strategy and platform control in the post-PC world," *Telecomm. Policy*, vol. 38, no. 11, pp. 979–991, 2014.

[9] R. Meier, Android Application Development, Wiley Publishing, Inc., vol. 131. 2010.

[10] R. Heydon, Bluetooth low energy : The developer's handbook. " Prentice Hall. 2012.

[11] Specification of the Bluetooth System Master Table of Contents & Compliance, vol. 0,Agere systems, Inc, Ericsson Technology Licensing, AB, IBM Corporation , Intel Corporation, Microsoft Corporation, Microsoft Corporation, Motorola, Inc, Nikia Corporation, Toshiba Corporation, 2004.

[12] L. D. Granados, "Análisis en maquinaria CNC ante variaciones de bajo voltaje y sus efectos de calidad de la energía.", PhD Tesis, Universidad Autónoma de Querétaro, San juan del rio, Querétaro, Jul 2013.

[13] H. Junfeng, S. Hao, and W. Xiaolin, "Design of Power Quality Monitor Based on Embedded Industrial Computer," *Phys. Procedia*, vol. 24, pp. 63–69, 2012.

[14] The Android Open Source Project, Bluetooth [Online]. Aviable: http://developer.android.com/intl/es/guide/topics/connectivity/bluetooth.html.

[15] The Android Open Source Project, View.OnClickListener [Online]. Aviable: http://developer.android.com/intl/es/reference/android/view/View.OnClickListener.html

[16] The Android Open Source Project, System [Online]. Aviable: http://developer.android.com/intl/es/reference/java/lang/System.html.

[17] Y. Zhang, L.T. Yang, J.Ma, "Detailed DSRC-WAVE Architecture", In Unlicensed mobile access technology, Boca Raton, Florida, Us, Taylor & Francis Group , 2009, Ch. 15. Sec. 15.5.2, pp. 311.

[18] The Android Open Source Project, Math [Online]. Aviable: http://developer.android.com/intl/es/reference/java/lang/Math.html.

[19] The Android Open Source Project, Canvas [Online]. Aviable:

http://developer.android.com/intl/es/reference/java/lang/System.html.

[20] The Android Open Source Project, Paint [Online]. Aviable: http://developer.android.com/intl/es/reference/java/lang/System.html.

[21] The Android Open Source Project, Android Open Source Project [Online]. Aviable: https://source.android.com/.

[22] FLUKE i200/i200s Ac Current Clamp, Technical Data, Fluke Corporations, 2005.