Proyecto final: Sistema de Medición en Tiempo Real de Variables Eléctricas.

Giraldo S. Lohana
lmgiraldos@unal.edu.co
Osorio S. Andrés
anosoriosa@unal.edu.co
UNIVERSIDAD NACIONAL DE COLOMBIA

***Abstract*— In the present document the results of the design and implementation of a real-time measurement system for electrical variables are shown, where the objective was to apply the acquired knowledge in the Processors course, by using a raspberry pi3.**

***Keywords*— Real Time Processing, Raspberry pi, Shared Memory, Communication Protocol, Data acquisition.**

I. INTRODUCCIÓN

En los últimos años, se ha visto incrementada la importancia de monitorear las variables más importantes de las redes eléctricas, con el fin de analizar su comportamiento, identificar anormalidades y tomar acciones correctivas que permitan garantizar calidad de la energía y operatividad adecuada de los equipos. Cuando no se consideran las características de las cargas conectadas, pueden presentarse incrementos de consumo, caídas de voltaje, incremento de pérdidas por efecto Joule y calentamiento, entre otros. Por lo anterior, resulta importante contar con un sistema que permita monitorear en tiempo real las variables más importantes de la red.

En el presente documento, se plantea la implementación académica de un sistema de medición en tiempo real de bajo costo, que permita al usuario interactuar al seleccionar el tipo de variable que desea visualizar a través de una pantalla o de una página web, mientras estos datos se almacenan en la memoria interna, con el fin de que puedan ser accedidos posteriormente para efectuar análisis detallados.

Pese a que en la industria existe una amplia oferta de dispositivos de este tipo, la aproximación a la implementación de sistemas de medición en tiempo real, abre paso a futuras investigaciones y posibles implementaciones de dispositivos de medición no solo de variables eléctricas, sino de demás variables que puedan ser requeridas por un usuario.

II. MARCO TEÓRICO

A. RASPBERRY PI

La Raspberry Pi, es un ordenador de placa reducida, de bajo costo diseñado en Reino Unido inicialmente para fomentar la enseñanza de computación alrededor del mundo [1].

El modelo utilizado corresponde a la Raspberry PI 3 B, que posee una arquitectura RISC de 64 bits. Esta posee un chip integrado Broadcom BCM2837 64bit, con un procesador ARMv7 de cuatro núcleos de 1.2 GHz y una unidad de procesamiento de gráficos (GPU) a 400 MHz, el dispositivo posee una memoria RAM de 1GB, y una ranura para tarjeta micro SD (memoria no volátil) [1], [2].

Existen varios sistemas operativos compatibles con la Raspberry pi, en el presente trabajo se utilizará el sistema operativo Raspbian Lite.

Esta tarjeta posee además posee wifi y bluetooth integrados, para la conexión con diferentes periféricos o para envío y recepción de datos. Adicionalmente, posee un puerto Ethernet, cuatro puertos USB 2.0, un conector audiovisual (AV), una interfaz serial para cámara (CSI), un puerto HDMI y 40 pines GPIO distribuidos en dos filas [1].

B. TRANSFORMADA RÁPIDA DE FOURIER (FFT)

La transformada rápida de Fourier es un algoritmo agiliza el cálculo de la Transformada Discreta de Fourier (DFT) ya que elimina gran parte de los cálculos repetitivos. El algoritmo fue propuesto en 1965 por Cooley y Tukey [3].

Los conjuntos de datos en el tiempo discreto se convierten en una representación de frecuencia discreta. Esta es una variante numérica de la transformada de Fourier, que convierte un vector de n (potencia de 2) amplitudes de entrada en un vector de n frecuencias [3].

$$\hat{f}(k) = \sum_{n=0}^{N-1} f(n)\, e^{\frac{-j2\pi kn}{N}}$$

Con $N = 2^n$

La aplicación de la Transformada Rápida de Fourier, reduce el número de operaciones de N^2 a $NLog(N)$, donde N Es el tamaño, dimensión del vector de entrada [3].

Donde, para el caso analizado en el presente informe, la amplitud de la onda objeto de análisis corresponde al doble de la amplitud del número complejo obtenido a partir de la implementación de la FFT, y la fase de la onda corresponde a la fase del número complejo.

C. COMUNICACIÓN UDP

El protocolo de comunicación UDP (User Data Protocol) se define en la RFC 768, publicado por Postel en 1980. Permite el envío de paquetes de datos independientes llamados datagramas sin necesidad de establecer una conexión. Este protocolo no garantiza la entrega ni la secuencia de los datagramas por lo que la aplicación que lo usa, debe hacerse responsable del control de errores. Este protocolo, sin embargo, permite minimizar los tiempos de transmisión de datos debido a que no hay un retardo adicional por el establecimiento de la conexión, lo que lo hace útil para aplicarlo en sistemas de tiempo real [4].

Los mensajes transmitidos se dividen en dos partes, una cabecera de 8 bytes y un campo de datos, acompañados de información que identifica el ordenador y puerto de destino [4].

El formato de los datagramas transmitidos es como sigue:

- Puerto UDP de origen (16 bits, opcional). Número de puerto máquina de origen, e indica dónde se envían las respuestas.
- Puerto UDP de destino (16 bits). Número de puerto de la máquina destino.
- Longitud del datagrama UDP (16 bits). Especifica la longitud medida en bytes del mensaje UDP incluyendo la cabecera.
- Checksum (16 bits, opcional). Suma de comprobación de errores del mensaje, y en caso de detección de error el dato es descartado. Cuando no se emplea se pone en cero.
- Datos. Datos transmitidos.

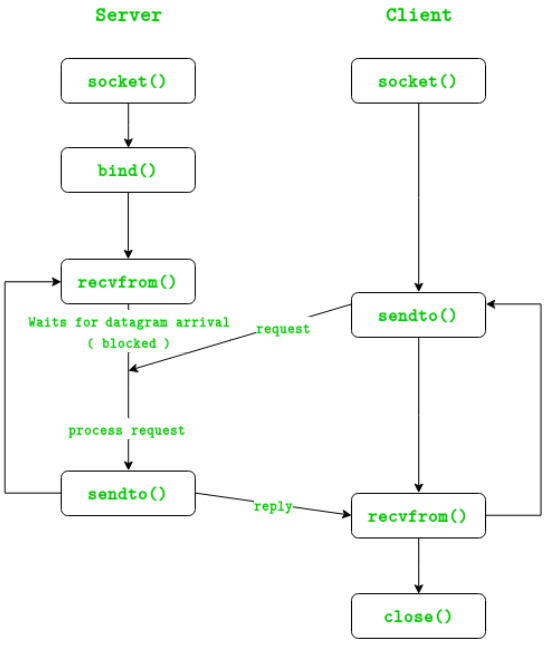

Figura 1. Estructura para transmisión de datos por protocolo UDP [7]

D. FORMATO LITTLE ENDIAN

Corresponde a uno de los formatos en los que pueden guardarse los datos que ocupan más de un byte en un ordenador, en este formato, el byte menos significativo de un dato de varios bytes se almacena primero, a diferencia del formato big endian, donde este se almacena de último. Por ejemplo, un dato formado por la secuencia 00000111 00000000 en formato Little Endian se puede interpretar como un número 7, mientras que en formato Big Endian equivale al número 1792 [5].

De lo anterior, se identifica la importancia de reconocer en qué plataforma fue creado un fichero de datos, y su correspondiente formato, con el fin de interpretarlo de manera adecuada.

E. MEMORIA COMPARTIDA

Es un mecanismo de memoria que permite a dos o más procesos compartir un segmento de memoria, y los datos que en este se almacenan, y por tanto se constituye como el método más rápido de comunicación entre procesos. Para asegurar la transferencia de información, es necesario realizar una sincronización de acceso entre los diferentes procesos y garantizar exclusión mutua a las secciones compartidas.

F. MULTIHILO

Hace referencia a la ejecución de varias tareas en forma simultánea, los hilos, son pequeños procesos independientes ejecutados dentro de otro proceso, que pueden programarse para que se ejecuten de manera simultánea dentro del mismo programa.

El paralelismo a nivel de hilo, o thread level parallelism (TLP), tiene como característica que los hilos comparten las unidades funcionales del procesador, sin embargo, este posee estructuras independientes para cada uno, por ejemplo, el contador de programa [7].

III.ARQUITECTURA

A. HARDWARE

La arquitectura de hardware consiste en la conexión entre un ordenador y la Raspberry Pi a través de el enrutador utilizando el sistema ethernet.

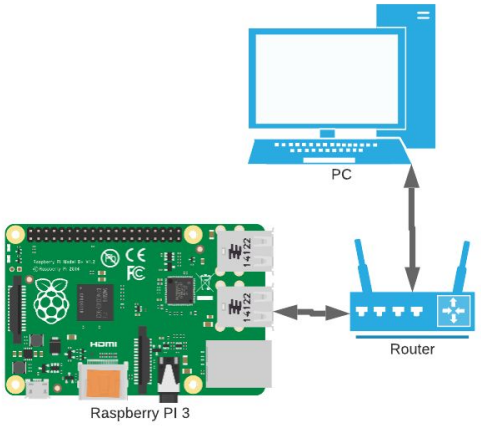

Figura 2. Arquitectura de Hardware utilizada.

B. SOFTWARE

La arquitectura de software tiene varias etapas.

En la etapa de programación se utilizó Visual Studio 2019 en Windows, con su soporte cross-platform para compilar los ejecutables y enviarlos a la memoria de la Raspberry a través de ethernet. La Raspberry se encuentra conectada con una IP local fija al igual que el ordenador.

Para la ejecución, se habilitó el acceso a la Raspberry a través del protocolo SSH y se realizó la conexión utilizando un terminal llamado MobaXterm.

El ordenador inicializa un servidor UDP abierto a la recepción de datos en la red local.

Una vez se ejecuta el programa de simulación, se ejecuta el hilo de procesamiento en paralelo, que utiliza la librería alglib para procesar los datos (fft)

1) Librería ALGLIB

Es una librería que tuvo sus inicios en el año 1999, para el análisis numérico y minería de datos. Actualmente es compatible con varios lenguajes de programación y varios sistemas operativos.

La librería posee múltiples funciones clasificadas en 11 paquetes, entre las que se pueden encontrar solucionadores de ecuaciones diferenciales, algoritmos de minería de datos, ***Transformada Rápida de Fourier***, algoritmos de integración

numéricos, de interpolación, de álgebra lineal, de optimización, solucionadores lineales y no lineales, funciones de análisis de datos (clasificación/regresión, estadística), entre otros [8].

Una vez se procesan los datos, se inicia un socket de cliente para enviar los datos al servidor UDP en el ordenador, para su visualización y almacenamiento posterior.

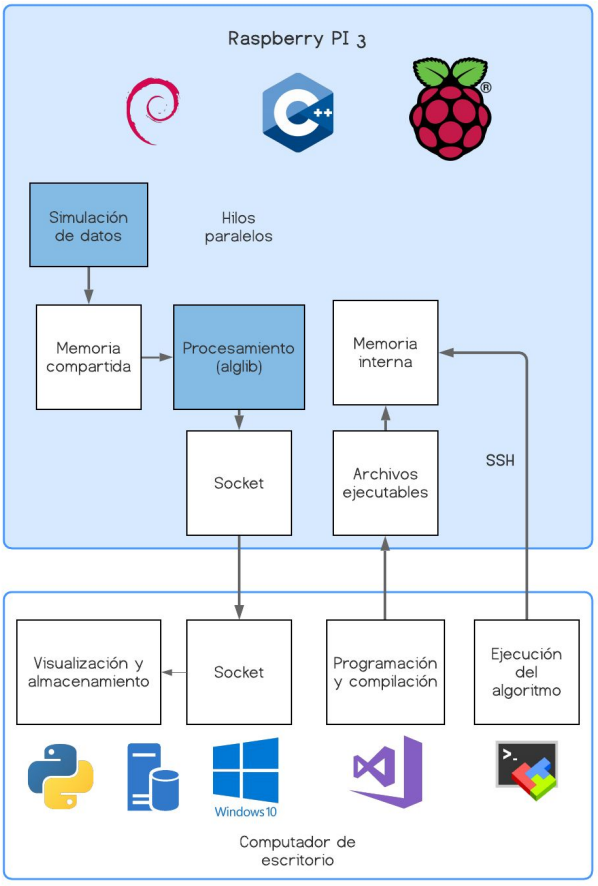

Figura 3. Arquitectura de Software utilizada.

IV. PROCEDIMIENTO UTILIZADO

Se inicializa la memoria compartida, los datos son guardados en una dirección de memoria específica para poder ser consultados desde el otro hilo. se guarda una ventana de datos que va siendo desplazada a medida que se actualizan los datos.
Se simula la medición de valores de voltaje y corriente en uno de los hilos de la raspberry, utilizando las funciones sin() y cos() de la librería estándar, considerando que los ángulos deben de estar en radianes (usualmente en la ingeniería eléctrica se usan en grados). Estos datos son guardados en un vector de tamaño fijo y a su vez, en una estructura que tiene una estampa de tiempo y un indicador de utilización (para no generar conflictos entre los datos mientras se procesan), la función de procesamiento es lanzada cada cierto tiempo utilizando alarmas.

Una vez se guardan los archivos en memoria compartida, se procede al procesamiento, en el cual se inicia un socket de cliente para enviar los datos a un socket de servidor, que los imprime en consola y los almacena en un log de texto.
La ventana de voltaje y de corriente es procesada mediante la fft, se busca en cada uno de sus posiciones la componente de mayor amplitud y se almacena la fase y la magnitud en valores RMS (en este caso solo se dividió por raiz de 2 la magnitud del mayor componente).

La fase del complejo corresponde con la fase de la onda, pero como la ventana se mueve al ser desplazada con nuevos datos, es necesario fijar una referencia, en este caso la referencia de fase es el voltaje, por ende a ambas fases se les resta la fase del voltaje.

Una vez se fija la referencia, se saca el coseno de la diferencia de fases, este es el factor de potencia, y se puede calcular la potencia activa haciendo el producto de las magnitudes de voltaje y corriente y multiplicando por el factor de potencia.

Se hace el formato de una cadena con la función sprintf() y se envían los datos al servidor UDP.

Nota: Debido a las condiciones presentadas para el desarrollo del proyecto, se simulan las mediciones de voltaje y corriente ya que se carece de equipos para la realización de mediciones y pruebas, en caso de implementación, es posible hacer uso de un sensor de voltaje (Integrado LM358-V3), y un módulo sensor de corriente (ACS712 20A).

V. RESULTADOS OBTENIDOS

Se simulan varias ondas de distintas frecuencias (40,50, 58, 60 Hz), y de igual magnitud, se tiene un desfase de 60 grados entre las ondas. Se procesan los datos y se obtienen los siguientes resultados:

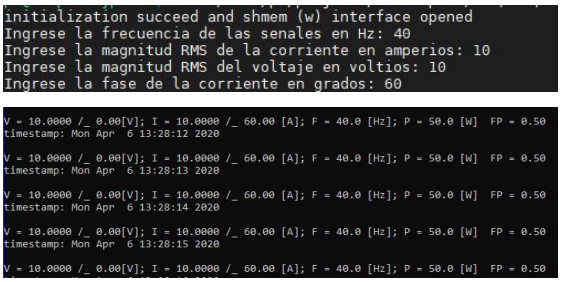

Figura 4. Resultados prueba a 40 Hz.

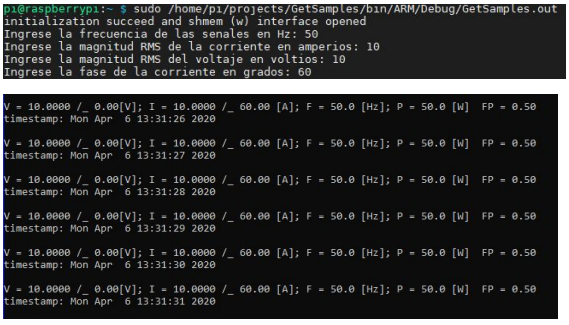

Figura 5. Resultados prueba a 50 Hz.

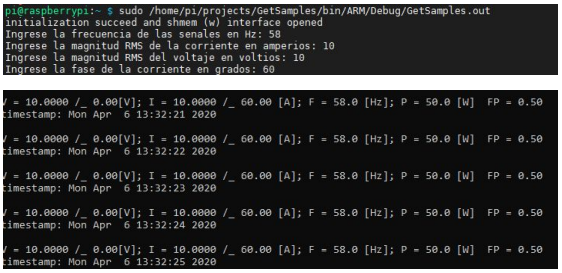

Figura 6. Resultados prueba a 58 Hz.

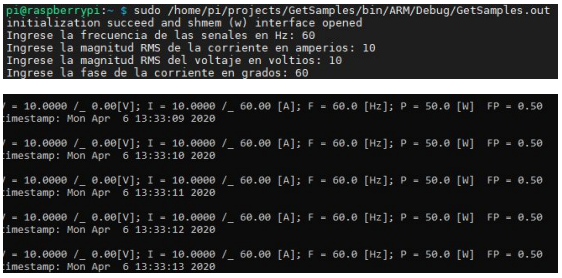

Figura 7. Resultados prueba a 60 Hz.

Se observa que en todos los casos los valores concuerdan con los esperados, cabe resaltar que se pueden tener señales de hasta 1 kHz por la frecuencia de muestreo usada (2 kHz).

VI. CONCLUSIONES

La utilización de memoria compartida, paralelismo y otras técnicas permiten agilizar la ejecución de programas que requieren muchas operaciones matemáticas, es importante analizar en qué casos es conveniente esto, en especial cuando se tienen comunicaciones que requieren que el receptor esté siempre listo para recibir datos y que podría estar ocupado procesando datos.

Una de las optimizaciones podría ser la utilización de librerías de procesamiento de señales de más bajo nivel, como la versión paga de ALGLIB, en este caso al ser un solo sistema basta con la versión normal, pero si se desea implementar una PMU o aplicaciones que requieran tiempo real, es conveniente utilizar otros métodos y además una modificación al kernel de Linux que deshabilite muchos periféricos e interrupciones que tiene el estándar.

A veces basta con tener un servidor abierto a la recepción de datos, pero es preferible utilizar una comunicación encriptada en caso de estar enviando datos sensibles. A pesar de la diferencia en los lenguajes de implementación de los socket, funciona adecuadamente la recepción y almacenamiento en el servidor, pero si el volumen de datos es mayor, se recomienda programación de socket a bajo nivel.

VII. REFERENCIAS

[1] G. Halfacree, “The Official Raspberry Pi Beginner’s Guide How to use your new computer,” Raspberry Pi Trading Ltd, p. 240, 2018. Recuperado de: https://www.raspberrypi.org/magpi-issues/Beginners_Guide_v1.pdf

[2] Raspberry Pi Foundation, “Raspberry Pi 3 Model B+,” Raspberry Pi Website. 2016. Recuperado de https://static.raspberrypi.org/files/product-briefs/Raspberry-Pi-Model-Bplus-Product-Brief.pdf

[3] A. Martínez Manzano, “Transformada rápida de Fourier Implementación y algunas aplicaciones,” Universidad de Murcia, 2018. Recuperado de: https://www.um.es/documents/118351/9850722/Mart%C3%ADnez+Manzano+TF_48705250_v2.pdf/c44507c8-e990-4aac-b282-927acaddcedd1

[4] C. E. Molina C., “Protocolo de datagramas de usuario (UDP).” Recuperado de: http://www.redtauros.com/Clases/Fundamentos_Redes/07_Protocolo_Internet_UDP.pdf

[5] G. Domínguez Aguilar, “El Formato ‘Big Endian’ y el ‘Little Endian.’” p. 1, 2014. Recuperado de: https://docentes.uaa.mx/guido/wp-content/uploads/sites/2/2014/10/BE_LE.pdf

[6] I. Rodero and F. Guim, “Arquitecturas multihilo.” Universitat Oberta de Catalunya. Recuperado de: https://www.exabyteinformatica.com/uoc/Informatica/Arquitecturas_de_computadores_avanzadas/Arquitecturas_de_computadores_avanzadas_(Modulo_3).pdf

[7] “UDP Server-Client implementation in C”. Recuperado de:https://www.geeksforgeeks.org/udp-server-client-implementation-c/

[8] ALGLIB Project. “Manual de referencia”. Recuperado de: https://www.alglib.net/translator/man/manual.cpp.html#int_main