Revista Cubana de Ciencias Informáticas
Vol. 15, No. 4, Octubre-Diciembre, 2021
ISSN: 2227-1899 | RNPS: 2301
http://rcci.uci.cu
Pág. 76-104

Tipo de artículo: Artículo original
Temática: Desarrollo de aplicaciones informáticas
Recibido: 17/08/2021 | Aceptado: 02/10/2021

Sistema visualizador para un radar de seguimiento con reducción de costo, consumo eléctrico y tamaño

Display system for a tracking radar with reduction of cost, electrical consumption and size

Lisvan Guevara Trujillo [1*] https://orcid.org/0000-0002-1830-2045
Bárbaro Nicolás Socarras Hernández [2] https://orcid.org/0000-0001-7924-1226
Leandro Zambrano Méndez [3] https://orcid.org/0000-0002-8243-2813
Wenny Hojas-Mazo [3] https://orcid.org/0000-0002-8298-3439
Margarita André Ampuero [3] https://orcid.org/0000-0001-5088-6039

[1] Centro de Investigación y Desarrollo Naval. Calle Estrada Palma No. 13, Casa Blanca, Regla, La Habana, Cuba. lisvan94trujillo@gmail.com
[2] Centro de Investigación, Desarrollo y Producción “Grito de Baire”. Santa Ana No. 711, Plaza, La Habana, Cuba. bnz2030@gb.reduim.cu
[3] Universidad Tecnológica de La Habana “José Antonio Echeverría”, CUJAE. Calle 114 No. 11901, Marianao, La Habana, Cuba. {lzambrano,whojas,mayi}@ceis.cujae.edu.cu

*Autor para la correspondencia. (lisvan94trujillo@gmail.com)

Revista Cubana de Ciencias Informáticas
Vol. 15, No. 4, Octubre-Diciembre, 2021
ISSN: 2227-1899 | RNPS: 2301
http://rcci.uci.cu
Pág. 76-104

RESUMEN

Un visualizador de radar es un dispositivo electrónico que se utiliza para representar en un formato adecuado la información contenida en la radioseñal reflejada. En los radares de seguimiento, el período de actualización de la información en el visualizador es del orden de las décimas de milisegundos. Este artículo describe el diseño e implementación de un sistema que permite procesar y visualizar la información obtenida desde un radar de seguimiento en sustitución de un visualizador analógico. La solución propuesta se compone de hardware sustentado en ordenadores de placa reducida y un software elaborado con el entorno de desarrollado multiplataforma Qt Creator. Mediante una herramienta de análisis de rendimiento se identificó la función de mayor consumo de tiempo en el software y luego se utilizó la programación paralela en dicha función. Esto propició disminuir el tiempo de ejecución, incrementar la aceleración y cumplir con el período de actualización del radar en los dispositivos empleados, a diferencia de la variante secuencial donde sólo un dispositivo cumplió con esta restricción temporal. Los resultados evidenciaron la factibilidad del empleo de Qt Creator, los ordenadores de placa reducida y la programación paralela en aplicaciones de radar.

Palabras clave: ordenador de placa reducida; software multiplataforma; análisis de rendimiento; programación paralela; visualizador de radar de seguimiento.

ABSTRACT

A radar display is an electronic device used to represent the information contained in the reflected radio signal in a suitable format. In tracking radars, the information update period on the display is of the order of tenths of a milliseconds. This article describes the design and implementation of a system that allows the information obtained from a tracking radar to be processed and displayed as a substitute for an analog display. The proposed solution consists of hardware supported by single board computers and software developed with the Qt Creator multiplatform development environment. Using a performance analysis tool, the most time-consuming function in the software

Revista Cubana de Ciencias Informáticas
Vol. 15, No. 4, Octubre-Diciembre, 2021
ISSN: 2227-1899 | RNPS: 2301
http://rcci.uci.cu
Pág. 76-104

was identified and then parallel programming was used in this function. This led to a decrease in execution time, increased acceleration and compliant with the radar update period in the devices used, unlike the sequential variant where only one device complied with this time restriction. The results evidenced the feasibility of using Qt Creator, single board computers and parallel programming in radar applications.

Keywords: single board computer; multiplatform software; performance analysis; parallel programming; tracking radar display.

Introducción

Los radares datan de la primera mitad del siglo XX y continúan siendo de interés para la comunidad científica internacional, pues son elementos fundamentales de los sistemas de control de tráfico terrestre, aéreo y marítimo (Fominaya, 2004). Uno de los tipos de radar más comunes es el de seguimiento, cuya función principal es el seguimiento de objetivos (IEEE, 2017). Este tipo de radar tiene una tasa de actualización de la información del orden de 100 – 10 Hz (Butler, 1998). En su forma más rudimentaria, un sistema de radar consta de cinco elementos: un radiotransmisor, un radiorreceptor sintonizado a la frecuencia del transmisor, dos antenas y un visualizador, véase Figura 1 (Stimson et al., 2014).

Revista Cubana de Ciencias Informáticas
Vol. 15, No. 4, Octubre-Diciembre, 2021
ISSN: 2227-1899 | RNPS: 2301
http://rcci.uci.cu
Pág. 76-104

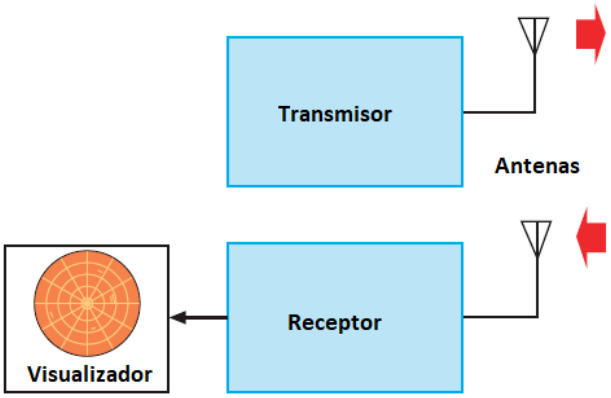

Fig. 1 – Elementos básicos que componen un radar.

El visualizador es un dispositivo electrónico que se utiliza para representar la información contenida en la señal de retorno en un formato adecuado para la interpretación del operador (Kavyashree et al., 2017b). Dicha información puede contener grandes cantidades de bits de datos por segundo. Existe una variedad de formatos usados para mostrar la información. En un esfuerzo por estandarizar la nomenclatura de los formatos de los visualizadores de radar, se establecieron indicadores denotados por las letras de la A a la P, más R (IEEE, 2017). Los tipos más comunes de indicadores son el de tipo A, el Indicador de Altura-Distancia (*Range Height Indicator*, RHI), Indicador de Posición en el Plano (*Plan Position Indicator*, PPI) o indicador de tipo P, indicador de tipo B e indicador tipo C (Kaushik, 2014). Estos indicadores deben presentar al observador una imagen gráfica continua, la cual permita comprender de forma fácil la posición relativa de los objetos detectados (Kaushik, 2014, Sulistyaningsih et al., 2019). La representación de los objetivos con su respectiva información durante la búsqueda y seguimiento, permite una mejor toma de decisión a los operadores. Uno de los medios empleados para la representación de las señales son los monitores basados en tubos de rayos catódicos (Kaushik, 2014, Sulistyaningsih et al., 2019). En la actualidad, una gran parte de los visualizadores de señales que utilizan los radares de seguimiento en Cuba se sustentan en tecnología analógica. Estos visualizadores son altos consumidores de energía eléctrica, poseen gran volumen y peso, sus piezas están obsoletas y existe incapacidad de adquirir los repuestos en el mercado

internacional. Además, están sometidos a largos períodos de mantenimiento, al presentar inestabilidad durante su funcionamiento.

La contribución principal de este trabajo es la obtención de un sistema de procesamiento y representación de señales para un radar de seguimiento que cumple con el período de actualización establecido por este sensor y posee bajo costo, consumo eléctrico, peso y tamaño respecto al visualizador analógico. Durante la revisión del estado del arte se prestó atención a los materiales empleados, a elementos de diseño del software y la validación del sistema. A continuación, se realiza un análisis de soluciones dadas por diversos autores en la implementación de visualizadores digitales de señales de radar.

En (Stamatović et al., 2013) se desarrolló un visualizador de señales de radar para remplazar un antiguo indicador basado en tubo de rayos catódicos, el producto fue nombrado Indicador de Radar Digital (*Digital Radar Indicator*, DRI). En (Manasa et al., 2015) se desarrolló un visualizador de datos de radar genérico en tiempo real, el cual es usado para probar y analizar los algoritmos del radar (proceso de detección, conversión de coordenadas y simulación de trayectorias) y para el entrenamiento de los usuarios. En (Ravindra et al., 2017) se diseñó e implementó un visualizador de señales de radar para darle seguimiento a múltiples objetos en tiempo real. El dibujo de los objetivos se realizó empleando QPainter y según los autores puede ser empleado en radares antiguos. En (Kavyashree et al., 2017a, Kavyashree et al., 2017b) los autores abordan el desarrollo de un generador de objetos aéreos en tiempo real para la comprobación de los sistemas visualizadores de los radares. La recepción de datos fue implementada haciendo uso de QThread permitiendo que se ejecute en un hilo independiente. En (Saputera et al., 2018, Sulistyaningsih et al., 2019) se diseñó y desarrolló un visualizador de señales de un radar para monitorear el espacio aéreo de Indonesia, los autores plantean que uno de los parámetros más importantes a tener en cuenta durante el procesamiento y representación es lograr realizarlo en tiempo real.

En las Tablas 1 y 2 se resumen las principales características de los componentes hardware y software de las soluciones descritas.

Revista Cubana de Ciencias Informáticas
Vol. 15, No. 4, Octubre-Diciembre, 2021
ISSN: 2227-1899 | RNPS: 2301
http://rcci.uci.cu
Pág. 76-104

Tabla 1 – Resumen de las propuestas del componente hardware de la literatura.

Autor	Tipo de indicador	Dispositivo	Interfaz de comunicación	Protocolo de comunicación
(Stamatović et al., 2013)	PPI	Computador de propósito general	No declarado	No declarado
(Manasa et al., 2015)		No declarado	Ethernet	TCP,UDP
(Ravindra et al., 2017)		No declarado		UDP
(Kavyashree et al., 2017a, Kavyashree et al., 2017b)		No declarado		
(Saputera et al., 2018, Sulistyaningsih et al., 2019)		Computador de propósito general		No declarado

Elaboración propia.

La Tabla 1 muestra que todas las propuestas identificadas emplean el indicador PPI, lo cual puede estar relacionado a la posibilidad de representar la información en los 360 grados. La mayoría de los trabajos emplean como interfaz de comunicación Ethernet, con el uso del protocolo de comunicaciones UDP. Los pocos trabajos que declaran el dispositivo, lo que utilizan son computadores de propósito general, lo cual podría tener un costo superior de desarrollo del proyecto, un aumento de consumo eléctrico y volumen con respecto al empleo de otras tecnologías como los ordenadores de placa reducida (SBC, por sus siglas en inglés).

Tabla 2 – Resumen de las propuestas del componente software de la literatura.

Autor	Sistema operativo	Patrón de diseño	IDE	Lenguaje de programación
(Stamatović et al., 2013)	Windows 7	Modelo Vista Presentador		C++
(Manasa et al., 2015)	GNU/Linux	Modelo Vista Controlador		
(Ravindra et al., 2017)		No declarado		

(Kavyashree et al., 2017a, Kavyashree et al., 2017b)	No declarado	No declarado	Qt Creator	
(Saputera et al., 2018, Sulistyaningsih et al., 2019)	GNU/Linux	No declarado		C

Elaboración propia.

La Tabla 2 muestra que todas las propuestas identificadas emplean el Entorno de Desarrollo Integrado (IDE, por sus siglas en inglés) Qt Creator, lo cual puede estar motivado a la facilidad que brinda este IDE para obtener un código multiplataforma. Los trabajos que declaran los patrones de diseño emplean modelo vista presentador y modelo vista controlador durante el desarrollo del software. El lenguaje de programación mayormente empleado fue el C++. La mayoría de los trabajos emplean distribuciones de GNU/Linux, no siendo así en (Stamatović et al., 2013) que emplearon Windows 7. Esta decisión provoca un aumento de costo al ser este un sistema operativo privativo por el que se debe pagar por la licencia que autoriza su utilización y no se tuvo en cuenta aspectos como la estabilidad, seguridad, disponibilidad del código fuente y eficiencia en términos de uso de recursos de cómputo.

Los trabajos consultados carecen de un diseño del indicador donde se representa la información y de claridad en el procesamiento de las muestras para lograr una fiel visualización de la situación aérea. En estos trabajos no se fundamenta la selección de la arquitectura hardware, ya que además de tener que cumplir con requerimientos de fiabilidad, rendimiento en tiempo real y flexible ante posibles cambios, se debería considerar el consumo de energía, el costo, el tamaño y peso. Además, no realizan experimentos detallados donde se incluya el análisis de rendimiento en el dispositivo seleccionado, ni se utilizan métricas de evaluación del rendimiento del software, para sustentar la calidad de las soluciones propuestas.

El contenido de este artículo está organizado en tres secciones fundamentales. En la sección de métodos o metodología computacional se exponen los criterios tenidos en cuenta para la selección de la arquitectura hardware-software, así como los principales elementos tenidos en cuenta durante el diseño e implementación del software. En la sección de resultados y discusión

se expone el análisis de las soluciones implementadas y la selección de la mejor variante. La última sección corresponde a las conclusiones, donde se exponen las ideas principales del análisis realizado en la sección anterior.

Métodos o Metodología Computacional

Esta sección está organizada en tres subsecciones. En la primera se fundamenta la selección de la tecnología necesaria para el desarrollo de la solución, en la segunda y tercera subsecciones, se exponen elementos fundamentales para el diseño e implementación de la solución.

Selección de la arquitectura hardware-software

Para la selección del dispositivo a emplear se valoraron los computadores de propósito general, los kits de FPGA (*field-programmable gate array*) y los ordenadores de placa reducida (SBC, por sus siglas en inglés), debido a la capacidad de procesamiento que poseen estas plataformas hardware. Se optó por emplear un ordenador de placa reducida, ya que cumplen con los requerimientos propuestos para el sistema de procesamiento y representación. Entre las ventajas del empleo de estas placas respecto a un computador de propósito general se encuentran el bajo consumo energético, tamaño reducido, menor costo y peso. Además, se optó por emplear un SBC (*Single Board Computer*) frente al uso de hardware reconfigurable. Esta decisión se basa en primer lugar por el costo del dispositivo, ya que desarrollar una placa con FPGA que cuente con salida de video e interfaz Ethernet para la recepción de datos o la compra de un kit con FPGA el costo sería mayor respecto a los SBC disponibles. La tecnología de los dispositivos FPGAs es considerada secreto industrial, por lo que no es posible conocer toda la información de la configuración, lo que imposibilita el diseño de herramientas libres (Brizuela, 2017). Los SBC son fáciles de programar, ya que es posible utilizar las mismas herramientas y bibliotecas que se usan para las aplicaciones de computadoras de propósito general, lo cual acorta significativamente la curva de aprendizaje necesaria para dominar una nueva arquitectura de

hardware y lenguaje de programación (Novo, 2019). También se tuvo en cuenta que en la propia FPGA no se puede desarrollar, es decir, que se necesita de un computador que cumpla con los requisitos mínimos para la instalación del software que permitirá realizar la síntesis del hardware, a diferencia de una SBC que en ella se puede desarrollar. Además, ha existido un crecimiento en el uso de software y hardware libre en los SBC. Para el desarrollo del sistema se cuenta con tres ordenadores de placa reducida: Raspberry Pi 4 modelo B, Odroid-XU4 y el Odroid-N2.

Un sistema embebido simple y de bajo costo puede realizarse sin usar un sistema operativo. Sin embargo, considerando el tiempo y esfuerzo de diseño, se presenta como mejor opción el uso de un sistema operativo embebido (Seo et al., 2017, Ramos et al., 2021). Una de las opciones de sistemas operativos conocidos por las ventajas que posee, son las distribuciones GNU/Linux. La selección de Linux se debe a la calidad y disponibilidad de código, amplio soporte de hardware, implementación de diversos protocolos de comunicación e interfaces para la programación de aplicaciones (API, por sus siglas en inglés), variedad de herramientas de desarrollo disponibles, condiciones favorables de licencia, independencia de proveedores y costo (Leppinen, 2017, Ramos et al., 2021). Se seleccionó el sistema operativo Ubuntu para la implementación del sistema por varios aspectos: es una distribución de GNU/Linux muy conocida y utilizada a nivel mundial por su simplicidad y fácil uso. Está basado en el sistema operativo GNU/Linux Debian, heredando la robustez, estabilidad y seguridad de este. Es fácilmente configurable y puesta a punto para la ejecución de la aplicación. Al contarse con experiencia de trabajo sobre la plataforma, se ahorra tiempo de configuración y se aborda rápidamente el problema a resolver. Es un sistema operativo multitarea y multihilo, el cual es muy eficiente en términos de uso de recursos del sistema. Es un sistema operativo estable, característica necesaria para aplicaciones de este tipo, donde el mantenimiento del sistema es esporádico y el tiempo de ejecución continua predomina. Destaca debido a que se actualiza cada seis meses lo que repercute en una mejor detección del hardware y la disponibilidad de versiones más modernas de las aplicaciones incluidas. Es menos propenso a malware y virus. La mayoría del software compatible con Ubuntu

también es gratuito, está disponible en "repositorios" en línea y tiene un amplio soporte de la comunidad.

Como resultado del análisis bibliográfico, se pudo evidenciar que los lenguajes de programación pueden jugar un papel importante en la eficiencia energética de un software y que los lenguajes compilados son más eficientes y se ejecutan en menor tiempo que los lenguajes semi-compilados e interpretados. En (Hundt, 2011), un ingeniero de Google demuestra que un algoritmo puede comportarse de forma muy diferente dependiendo del lenguaje de programación utilizado para su implementación, pues los resultados mostraron variaciones de tiempo de ejecución de hasta 12 veces y el C++ tuvo menor tiempo de ejecución y ocupó menor espacio en memoria que los otros tres lenguajes. Miembros de la Universidad Estatal de Texas en (Abdulsalam et al., 2014) estudiaron el impacto de los lenguajes C, C++, Java y Python en la eficiencia energética. Se implementaron tres optimizaciones del lenguaje C y C++ durante la ejecución de la Transformada Rápida de Fourier y Quicksort, produciendo una mejora por encima de un 50 % en tiempo de ejecución y eficiencia energética con respecto al no optimizado. El código C demostró ser ligeramente más rápido y más eficiente energéticamente que el C++ en el caso que no estén optimizados, sin embargo, la versión optimizada del C++ fue más rápida y más eficiente que la versión optimizada de C, evidenciándose que C++ puede hacer un buen uso de los recursos de CPU y memoria. En (Pereira et al., 2017, Pereira et al., 2021) se monitorearon 27 lenguajes de programación empleando un computador de escritorio con el sistema operativo Ubuntu Server 16.10 y los resultados obtenidos mostraron que los tres lenguajes de programación con menor tiempo de ejecución y más eficientes energéticamente fueron C, Rust y C++, destacándose el C y C++ por tener un menor consumo de memoria. El estudio de eficiencia energética realizado en (Georgiou et al., 2018) demuestra que existen diferencias en el empleo de un lenguaje de programación en plataformas distintas. El análisis fue realizado a 14 lenguajes de programación en las siguientes plataformas: servidor Dell Vostro 470 con CPU Intel i7-3770, laptop HP EliteBook 840 G3 con CPU Intel i7-6500U y el SBC Raspberry Pi 3 modelo B. Como resultado se

evidenció que las implementaciones realizadas en C obtuvieron el mejor rendimiento en las tres plataformas y C++ obtuvo un buen desempeño en el SBC por detrás del C.

A pesar de variar un poco el orden de los lenguajes dependiendo de parámetros propios de la plataforma, versiones de los compiladores, versiones de las bibliotecas, máquinas virtuales y los parámetros de optimización, se concluye que C y C++ poseen los mejores tiempos de ejecuciones y eficiencia de energía. C++ incluye muchos aspectos del lenguaje C, pero además incorpora muchas características sofisticadas tales como: programación orientada a objetos, excepciones, sobrecarga de operadores y uso de plantillas (*templates*) (Espinosa, 2011). También C++ posee control absoluto de memoria y se pueden usar todas las técnicas de manejo de punteros y conteo de referencia (Espinosa, 2011). Teniendo en cuenta estos resultados se opta por escoger como lenguaje de programación el C++ para aprovechar su velocidad de ejecución, eficiencia y todas las potencialidades que ofrece de manera general. Además, existen diversos trabajos donde se utiliza este lenguaje en el desarrollo de visualizadores de radar (Stamatović et al., 2013), (Manasa et al., 2015), (Ravindra et al., 2017) y (Kavyashree et al., 2017a, Kavyashree et al., 2017b).

Para la elección del IDE se tuvo en cuenta que fuera una aplicación de software libre, multiplataforma y que permita programar en C++, por lo que se seleccionó Qt Creator. La API Qt es muy amplia e incluye estructura de datos, sistemas de redes, entrada y salida, soporte para hilos, entre otras potencialidades (Beaupre et al., 2018). Las aplicaciones de Qt son capaces de adaptarse a 12 plataformas diferentes incluidas Android, iOS, macOS, Windows, Linux y otras plataformas embebidas (Beaupre et al., 2018). Johan Thelin plantea que Qt es una buena opción para usar en sistemas embebidos (Thelin, 2007). Ray Rischpater considera que para plataformas embebidas, actualmente, es mejor usar las clases de Qt, debido a que poseen un bajo consumo de memoria, son legibles y portables en diferentes plataformas (Rischpater, 2013). Qt es una popular elección en sistemas embebidos y ha sido usada para crear interfaces gráficas de usuario en diversos sistemas. Por ejemplo, en (Shengwen et al., 2011, Bugay, 2016) se

Revista Cubana de Ciencias Informáticas
Vol. 15, No. 4, Octubre-Diciembre, 2021
ISSN: 2227-1899 | RNPS: 2301
http://rcci.uci.cu
Pág. 76-104

diseñaron sistemas de control de carga y descarga de baterías. En estos trabajos se empleó la plataforma de desarrollo embebido basada en Linux y el software embebido Qt. Los autores percibieron que el Qt embebido fue la mejor opción debido a su alto rendimiento, bajo consumo de memoria y por las posibilidades de ser desplegado en diferentes sistemas sin cambiar el código fuente. Además, ha sido empleado para el desarrollo de visualizadores digitales de señales de radar en (Stamatović et al., 2013), (Manasa et al., 2015), (Ravindra et al., 2017), (Kavyashree et al., 2017a, Kavyashree et al., 2017b) y (Saputera et al., 2018, Sulistyaningsih et al., 2019) lográndose visualizar la situación aérea en tiempo real. Adicionalmente, se tomó en cuenta que Qt Creator posee novedosas concepciones como es el caso de la comunicación entre módulos para la programación en ambientes gráficos, mediante el mecanismo que lo hace diferir de la mayoría de las plataformas: los "*signals*" y los "*slots*".

Diseño del sistema

La Figura 2 muestra el esquema de solución del sistema visualizador de señales de un radar de seguimiento:

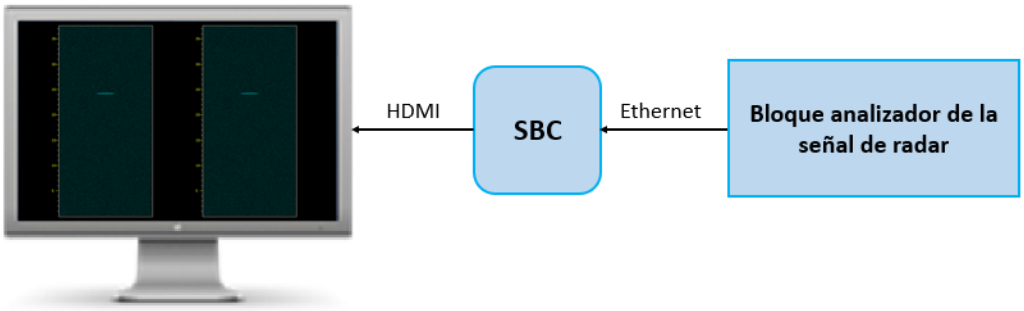

Fig. 2 – Esquema de solución del visualizador digital de señales de un radar de seguimiento.

El esquema muestra que el SBC recibirá la información de la señal del radar por interfaz Ethernet desde el bloque analizador. Los datos recibidos corresponden a muestras de amplitud de la señal del radar. Cada recepción es un barrido del radar, el cual contiene 6250 muestras de 8 bits. Una exploración del espacio aéreo lo componen 128 barridos. El período de actualización de la

Revista Cubana de Ciencias Informáticas
Vol. 15, No. 4, Octubre-Diciembre, 2021
ISSN: 2227-1899 | RNPS: 2301
http://rcci.uci.cu
Pág. 76-104

información de cada exploración es de 40 milisegundos, constituyendo esta la restricción de tiempo impuesto por el radar.

Cada barrido recibido será procesado en el SBC y luego representado en un monitor. En este trabajo se hará uso del indicador de tipo B para la representación de la información, por ser ampliamente utilizado para el aterrizaje de precisión de aeronaves. El indicador de tipo B es una pantalla rectangular en la que cada objetivo aparece como un destello de intensidad (IEEE, 2017). El indicador de tipo B proporciona una representación bidimensional del espacio, en el eje vertical se representa la distancia y en el eje horizontal el acimut (Kaushik, 2014). El espacio es barrido hacia arriba en el eje Y, con distancias mayores en la parte superior de la pantalla (Kaushik, 2014).

El software estará constituido por tres procesos, los cuales serán ejecutados mientras se reciban datos. Como se muestra en la Figura 3, el software puede ser visto como una secuencia de recepción, procesamiento y representación de los datos.

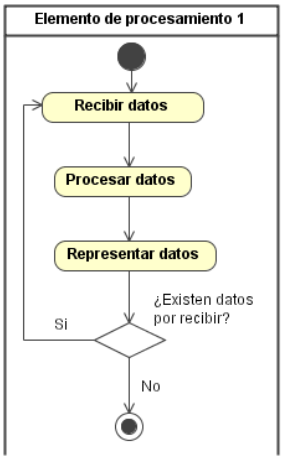

Fig. 3 – Diagrama de actividades de la variante secuencial del software.

En el diagrama de actividades se aprecia que para iniciar cada una de las etapas en las que se divide la ejecución del software, es necesario esperar a que termine la etapa anterior.

La complejidad de un sistema y los problemas recurrentes que contiene, requiere que los desarrolladores organicen su código de tal manera que sea fácil de comprender y solucionar problemas. En la actualidad existen varios patrones para el diseño de interfaces de usuario, cada uno con sus facilidades y dificultades. El uso de patrones de diseño brindará una aplicación más estructurada, organizada y escalable (Medina et al., 2018). La arquitectura modelo vista controlador se utiliza para separar los datos de la clase encargada de la representación. Se trata de un modelo maduro y que ha demostrado su validez a lo largo de los años en todo tipo de aplicaciones y, sobre multitud de lenguajes y plataformas de desarrollo (Servicio de Informática, 2021). Además este patrón de diseño se basa en las ideas de reutilización de código y la separación de conceptos, para su posterior mantenimiento (Medina et al., 2018). El modelo será la clase encargada de la recepción de los datos. En esta clase es donde se configurará la dirección IP y puerto por donde se recibirán los datos. La clase controladora hará de intermediario en la comunicación entre la clase de recepción de datos y la de representación. En esta clase además se realizará el procesamiento de los datos. La vista será la interfaz de usuario donde se representará la información contenida en la señal de radar.

Para la representación de los datos en pantalla fue necesario determinar las dimensiones del indicador. Para ello se tuvo en cuenta las características de la señal y que la resolución del monitor que se usaría es de 1366 x 768 píxeles. Conociendo que el radar realiza 128 barridos por exploración, se propuso que el indicador posea un ancho de 256 píxeles, representando 2 columnas de píxeles por barrido. En el eje vertical del indicador se visualizará cada uno de los barridos del radar. Los píxeles representan la intensidad de las muestras de la señal. Conociendo que por cada barrido se reciben 6250 muestras y que la resolución vertical del monitor es de 768 píxeles, no es posible representar una muestra por píxel, decidiéndose que el indicador va a poseer una longitud vertical de 625 píxeles, representando 1 píxel por cada 10 muestras. Para la representación de 10 muestras por píxel en el eje vertical, se hace necesario un método de

Revista Cubana de Ciencias Informáticas
Vol. 15, No. 4, Octubre-Diciembre, 2021
ISSN: 2227-1899 | RNPS: 2301
http://rcci.uci.cu
Pág. 76-104

integración para determinar cuál es el valor que tomará el píxel, para ello se tuvieron en cuenta tres métodos:

1. El promedio de las muestras, el cual consiste en promediar las 10 muestras de amplitud a representar. Su inconveniente es que disminuye considerablemente la amplitud del objeto aéreo.
2. Valor máximo de las muestras, el cual consiste en determinar el valor máximo de amplitud de las 10 muestras a representar. Su inconveniente es que resalta los picos de ruido.
3. Valor máximo de promedios, en el cual se combinan los métodos anteriores. Consiste en dividir las 10 muestras, en 2 grupos de 5 muestras. Cada uno de estos grupos se promedian, disminuyendo así la amplitud de los picos de ruido, y después se selecciona el mayor promedio de estos dos grupos, permitiendo que el valor de amplitud del objeto aéreo no disminuya tanto.

En la Figura 4 se muestra el pseudocódigo del método valor máximo de los promedios, el cual fue seleccionado para el procesamiento de las muestras.

```
Función Procesar (muestras)
    Para i ← 0 Hasta 6249 Con Paso 10 Hacer
        sumaGrupo1 ← muestras(i) + muestras(i+1) + muestras(i+2) + muestras(i+3) + muestras(i+4)
        sumaGrupo1 ← muestras(i+5) + muestras(i+6) + muestras(i+7) + muestras(i+8) + muestras(i+9)
        mayor ← Max(sumaGrupo1, sumaGrupo2)
        listaIntg(i/10) ← mayor/5
    Fin Para
    Devolver listaIntg
Fin Función
```

Fig. 4 – Pseudocódigo del algoritmo de procesamiento de las muestras.

Implementación del sistema

La aplicación fue desarrollada en el IDE Qt Creator, empleando el lenguaje de programación C++ y el sistema operativo Ubuntu 18.04.

La recepción de los datos se realizó empleando el protocolo de comunicación UDP mediante la clase QUdp, disponible en Qt Creator. Este receptor vincula el puerto y la dirección IP de la interfaz de red por donde se va escuchar con *bind()*. Los 6250 valores recibidos, como resultado de la transmisión de cada barrido, son almacenados en un arreglo y luego procesados. Los 625 valores resultantes del procesamiento son representados en el indicador de tipo B, empleando la clase QPainter. Estos valores de intensidad actualizan el color de las 2 columnas de píxeles correspondiente a cada barrido obteniéndose, al recibir los 128 barridos, una imagen de 256 x 625 píxeles que muestra el sector de exploración del radar.

Se comprobó que el software implementado se puede ejecutar en los siguientes tres ordenadores de placa reducida:

1. La Raspberry Pi 4 modelo B es un ordenador de placa reducida, de bajo costo, desarrollado en Reino Unido por la Raspberry Pi Foundation. Esta diminuta placa, de 85x56x20 milímetros, aloja un chip Broadcom BCM2711 con un procesador ARM formado por 4 núcleos Cortex-A72 a 1.5 GHz, 4GB de RAM LPDDR4, 2 puertos USB 2.0 y 2 puertos USB 3.0, con dos salidas micro HDMI con soporte 4K e interfaz Ethernet a 1Gbps (Ltd, 2021a, Ltd, 2021b). Véase Figura 5a.
2. Odroid-XU4 es una familia de ordenadores creados por Hardkernel, una compañía de hardware libre con base en Corea del Sur. Este dispositivo posee una arquitectura multi-núcleo asimétrica basada en ARM. Está provista de un SoC (System on chip, en lengua inglesa) Samsung Exynos 5422 big.LITTLE de ocho núcleos (Österberg, 2017), formado por dos clúster de 4 núcleos cada uno, el Big, formado por 4 núcleos Cortex-A15 a 2 GHz y el Little por 4 núcleos Cortex-A7 a 1.4 GHz (Rondx, 2019). Además, posee 2 GB de RAM LPDDR3, 2 puertos USB 3.0, 1 puerto USB 2.0, interfaz Ethernet a 1 Gbps, almacenamiento microSD y eMMC 5.0, alcanzando este último mayor velocidad de lectura y escritura. El consumo eléctrico está entre 10 y 20 W (Roy et al., 2017). Véase Figura 5b.

Revista Cubana de Ciencias Informáticas
Vol. 15, No. 4, Octubre-Diciembre, 2021
ISSN: 2227-1899 | RNPS: 2301
http://rcci.uci.cu
Pág. 76-104

3. El Odroid-N2 posee un SoC Amlogic S922X big.LITTLE Hexa-core, con 4 núcleos ARM Cortex-A73 a 1.8 GHz y 2 núcleos Cortex-A53 a 1.9. Este SoC utiliza la GPU Mali-G52 a 846Mhz. Este SBC posee 4 GB de RAM a 1320Mhz. El dispositivo puede arrancar desde una microSD o desde una eMMC. La conectividad del dispositivo incluye HDMI 2.1 con soporte 4K, Ethernet a Gigabit, 4 puertos USB 3.0, puerto micro-USB 2.0 OTG y un conector de alimentación de corriente directa de 12V/2A (Larabel, 2019, Lee, 2019, Paula, 2019). Véase Figura 5c.

a)- Raspberry Pi 4 Modelo B

b)- Odroid-XU4

c)- Odroid-N2

Fig. 5 – Ordenadores de placa reducida empleados para la representación de las señales de radar.

Considerando la propiedad de calidad de un sistema/software eficiencia de desempeño, la cual está relacionada con el rendimiento de un dispositivo en función de su comportamiento temporal, entre otras definidas en la Norma ISO/IEC 25023:2016, se midió el tiempo de ejecución del software en cada uno de los SBC. En esta medición se detectó que, de los 3 ordenadores de placa reducida empleados, la Raspberry Pi 4 modelo B y el Odroid-N2 incumplieron con el período de actualización del radar. En (Piñero et al., 2021) se plantea que la eficiencia del desempeño puede ser afectado por la velocidad de ejecución del procesador, el ancho de banda de la red, la carga de la red, entre otros. Como el ancho de banda empleado es de 1 Gigabit/s, siendo este el máximo soportado por los ordenadores de placa reducida empleados y suficiente para garantizar la recepción de los datos, entonces los esfuerzos fueron concentrados en la velocidad de ejecución del software.

En (Mena et al., 2017) se plantea que la programación paralela es un área de la computación que permite aprovechar los recursos de hardware para mejorar el tiempo de ejecución de los algoritmos y que uno de los pasos a seguir para el diseño de un algoritmo paralelo es la identificación de las partes del algoritmo que son más costosas y que puedan ejecutarse de forma concurrente. En (Piñero et al., 2021) se plantea como buena práctica asociada a la eficiencia del desempeño, el uso de herramientas dedicadas a las pruebas, la monitorización y el análisis de registros de la eficiencia del desempeño en el software. Para la identificación de las funciones más costosas temporalmente se empleó la herramienta de análisis Gprof, la cual es una herramienta de análisis dinámico que extrae información en tiempo de ejecución (Janjusic et al., 2015, Cho, 2018). Dentro de la información recolectada por esta aplicación se encuentra el por ciento de tiempo de ejecución de las funciones, el tiempo de ejecución de cada función, el número de veces que se ejecuta cada función y el tiempo consumido por llamada de cada función (Cho, 2018, Singhal et al., 2019). Esta herramienta de software libre ordena los valores almacenados de forma decreciente como resultado del muestreo realizado durante 0,01 segundos (Singhal et al., 2019). Esta herramienta permite optimizar un programa a partir de la información obtenida, pues se logran identificar las funciones de mayor consumo de tiempo (Cho, 2018). Para hacer uso de esta herramienta desde Qt Creator se agregaron las siguientes líneas en el archivo .pro:

```
QMAKE_CXXFLAGS += -pg
QMAKE_LFLAGS += -pg
```

Gprof permitió detectar que las funciones más costosas temporalmente son en primer lugar, la encargada de la recepción de datos (representando entre un 53 y 57% del tiempo total), en segundo lugar, la encargada del procesamiento de los datos (aproximadamente un 22%) y, por último, la de representación con aproximadamente un 18% del tiempo total.

Revista Cubana de Ciencias Informáticas
Vol. 15, No. 4, Octubre-Diciembre, 2021
ISSN: 2227-1899 | RNPS: 2301
http://rcci.uci.cu
Pág. 76-104

En la variante secuencial para realizar el procesamiento de los datos y la representación, es necesario primero concluir con la recepción de los datos de un barrido. Utilizando el paralelismo se puede lograr que una vez recibidos los datos correspondientes al primer barrido, pueda realizarse de forma secuencial el procesamiento y representación de estos datos y a la misma vez, la recepción de los datos del segundo barrido, ya que la recepción consume más tiempo que el procesamiento y representación de manera secuencial. Debido a que más del 50 % del tiempo de ejecución del software lo consume la recepción de datos, no es necesario un hilo de ejecución independiente para el procesamiento, ni para la representación de los datos. La propuesta de la variante paralela se muestra en la Figura 6.

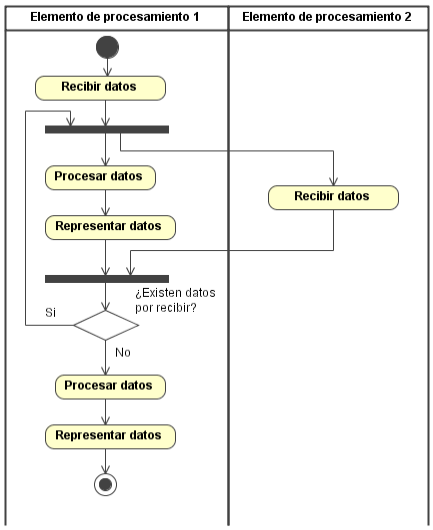

Fig. 6 – Diagrama de actividades de la variante paralela del software.

En la figura anterior se muestra que, para disminuir el tiempo de ejecución del software, mediante la paralelización del sistema, es necesario que el ordenador donde sea ejecutado posea como mínimo dos elementos de procesamiento, condición que cumplen los SBC disponibles. Para el desarrollo de la versión paralela del software se hizo uso del paradigma de memoria compartida. Este paradigma consiste en que múltiples núcleos pueden acceder a la misma memoria. Las principales ventajas que posee el uso de este paradigma es la facilidad de programar y rapidez al

compartir los datos entre los hilos de ejecución (Mena et al., 2017), siendo estas las principales razones por las que se seleccionó este paradigma. Para garantizar que el objeto encargado de la recepción de datos se ejecute en otro hilo se empleó la clase QThread.

Resultados y discusión

Esta sección está organizada en tres subsecciones. En la primera se exponen los principales elementos del diseño experimental que aseguran la obtención de datos apropiados y conducen a deducciones válidas. En la segunda subsección, se explica las métricas empleadas durante la evaluación del rendimiento del software para sustentar la calidad de las soluciones propuestas. En la tercera subsección, se analiza la influencia de las diferentes configuraciones del sistema sobre el tiempo de ejecución del software.

Diseño del proceso experimental

El proceso experimental a analizar es la ejecución del software encargado del procesamiento y la representación de la información de señales de un radar de seguimiento. La variable de respuesta es el tiempo de ejecución en milisegundos. Los factores controlables del experimento son la variante de programación y el dispositivo empleado. La variante de programación puede ser secuencial o paralelo. Los dispositivos empleados en las pruebas fueron la Raspberry Pi 4 Modelo B de 4 GB de RAM, Odroid-XU4 y el Odroid-N2 de 4 GB. Por lo que se realizaron 6 combinaciones y cada combinación fue replicada 10 veces para disminuir el error experimental, por lo que en total se realizaron 60 corridas.

Para que el ambiente en el que se aplican los tratamientos sea lo más uniforme posible, se realizó en orden aleatorio. En las ejecuciones se empleó la misma señal patrón y el sistema operativo Ubuntu 18.04, excepto en la Raspberry Pi 4 que se utilizó Raspbian con el panel de escritorio LXPanel 0.10.0. En el desarrollo de la versión paralela del software, sólo se emplearon 4 núcleos en el Odroid-XU4 y en el Odroid-N2, garantizando un menor consumo energético y

mayor eficiencia que al utilizar todos los disponibles. En el caso de la Raspberry Pi 4 se emplearon los 4 procesadores con que dispone.

Métricas de rendimiento

En (Goya et al., 2016) se plantea que el rendimiento de un algoritmo paralelo puede ser analizado con una mayor exactitud cuando se compara con su mejor versión secuencial. A los criterios de evaluación de los algoritmos paralelos, también se les llama métricas de rendimiento o medidas de rendimiento (Mena et al., 2017).

Las métricas permiten realizar un análisis cualitativo del rendimiento del algoritmo. Entre las métricas más empleadas se encuentran: la aceleración o *speed up,* y la eficiencia (Mena et al., 2017).

La aceleración es una medida que captura el beneficio relativo de resolver un problema en paralelo (Goya et al., 2016). Se define como la proporción del tiempo empleado secuencialmente con respecto al tiempo empleado en la versión paralela, aplicados a procesos idénticos (Mena et al., 2017). Esta métrica permite conocer cuántas veces es más rápido el programa paralelo con respecto al programa secuencial. Matemáticamente está definida por la siguiente ecuación:

$$S = \frac{T_S}{T_p}$$

Donde S es la aceleración, Ts representa el tiempo de ejecución secuencial y Tp el tiempo de ejecución paralelo.

La eficiencia es una medida de la fracción de tiempo para la cual un elemento de procesamiento es útilmente empleado (Goya et al., 2016), la cual denota qué tan bien se han utilizado los procesadores. En un sistema ideal la eficiencia es uno, pero en la práctica está entre cero y uno

Revista Cubana de Ciencias Informáticas
Vol. 15, No. 4, Octubre-Diciembre, 2021
ISSN: 2227-1899 | RNPS: 2301
http://rcci.uci.cu
Pág. 76-104

(Goya et al., 2016). La eficiencia de los programas paralelos decrece con el aumento del número de elementos de procesamiento para un tamaño de problema dado (Goya et al., 2016). Matemáticamente, está definida como la razón entre la aceleración S y el número de procesadores P empleados:

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

Análisis de los resultados

La Figura 7 muestra las gráficas de los valores de tiempo de ejecución en milisegundos de las 10 ejecuciones secuenciales y paralelas en cada uno de los dispositivos.

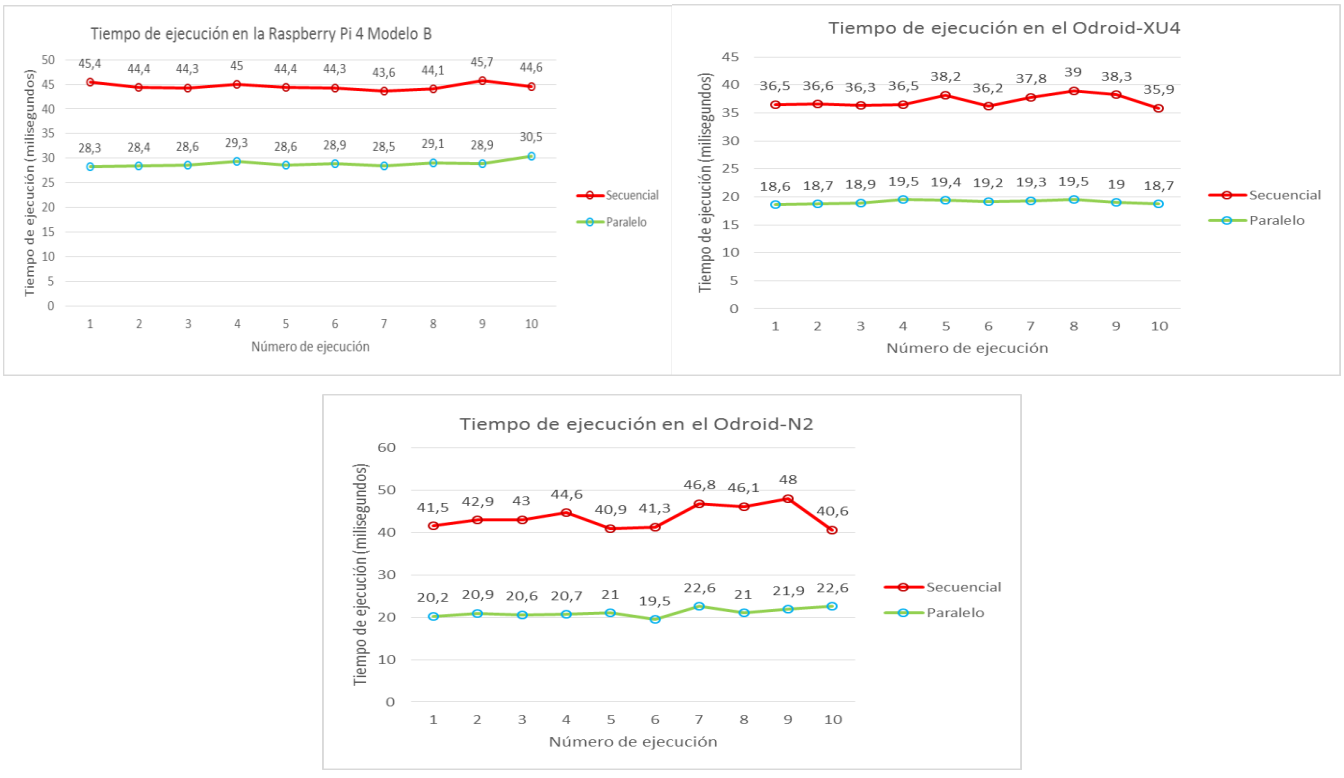

Fig. 7 – Gráficas de tiempo de ejecución del software.

Revista Cubana de Ciencias Informáticas
Vol. 15, No. 4, Octubre-Diciembre, 2021
ISSN: 2227-1899 | RNPS: 2301
http://rcci.uci.cu
Pág. 76-104

En las gráficas se aprecia que en la variante de programación secuencial sólo el Odroid-XU4 cumplió con la restricción temporal impuesta por el radar que es de 40 milisegundos. En la variante de programación paralela los tres dispositivos cumplieron con el requisito de tiempo.

En la Tabla 3 se muestran los valores de las métricas aplicadas al software en cada uno de los dispositivos empleados.

Tabla 3 – Métricas de rendimiento aplicadas al software.

Métrica		**Raspberry Pi 4**	**Odroid-XU4**	**Odroid-N2**
Promedio del tiempo de ejecución en ms	**Tiempo secuencial**	44,58	37,13	43,57
	Tiempo paralelo	28,91	19,08	21,1
Aceleración		1,54	1,94	2,06
Eficiencia		0,38	0,48	0,51

La tabla muestra disminución del tiempo de ejecución del software a partir del empleo de la variante de programación paralela. Los valores de aceleración mayores que 1 confirman la disminución del tiempo de ejecución de la variante paralela, alcanzando el Odroid N2 una disminución del tiempo de ejecución en un 51% respecto a la variante secuencial. Los valores de eficiencia mostrados en la tabla indican que el aprovechamiento de los recursos de cómputo de la Raspberry Pi 4, del Odroid-XU4 y del Odroid N2 fue de un 38%, 48% y 51%, respectivamente.

Durante la realización del análisis se procedió a generar un conjunto de gráficas sobre los experimentos realizados. El objetivo es investigar los efectos de las variables de entrada, conocido como factores, sobre la variable de respuesta. Para ello se empleó el software Minitab 19. A partir del análisis a los datos recolectados se determinará la configuración de factores que optimiza los resultados.

El diagrama de Pareto permite detectar los efectos determinantes de factores e interacciones en el rendimiento del proceso o sistema (Antony, 2014). Este diagrama muestra el valor absoluto de los efectos y se traza una línea de referencia en el gráfico. Cualquier efecto que sobrepase esta

Revista Cubana de Ciencias Informáticas
Vol. 15, No. 4, Octubre-Diciembre, 2021
ISSN: 2227-1899 | RNPS: 2301
http://rcci.uci.cu
Pág. 76-104

línea de referencia es potencialmente importante para el rendimiento del proceso (Antony, 2014). Los efectos son organizados según su valor en orden decreciente. Además, permite visualizar las variables que tienen un mayor impacto sobre la variable de respuesta. Como se aprecia en la Figura 8 la variante de programación, el dispositivo y la interacción de estos dos factores influyen significativamente en el tiempo del software.

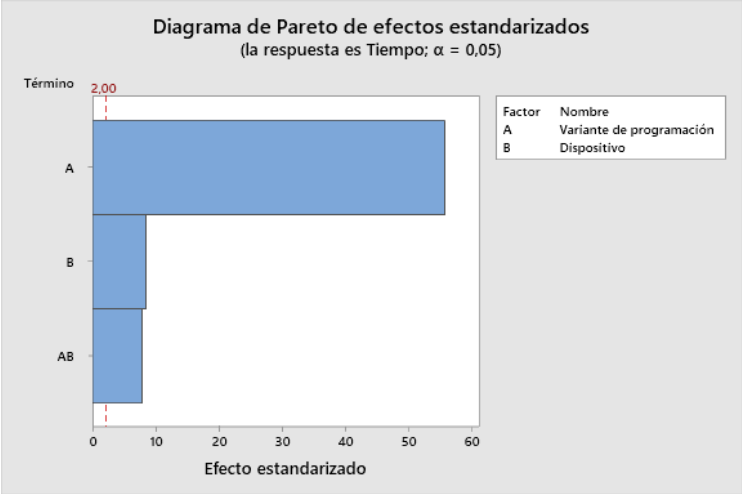

Fig. 8 – Diagrama de Pareto.

Puesto que el diagrama de Pareto muestra el valor absoluto de los efectos, se determinó que el efecto variante de programación es el más significativo.

Para analizar la magnitud del efecto de cada factor y comparar fácilmente el efecto entre distintos factores, se emplea la gráfica de efectos principales (Antony, 2014). Un efecto principal es la diferencia en la respuesta media entre los niveles de un factor. En esta gráfica se muestran las medias de tiempo utilizando los dos niveles de variante de programación y las medias de tiempo utilizando los tres dispositivos. La línea horizontal muestra la media de tiempo de procesamiento para todas las corridas. En la Figura 9 se muestra el efecto que tiene cada uno de los factores por separado en el rendimiento del problema.

Revista Cubana de Ciencias Informáticas
Vol. 15, No. 4, Octubre-Diciembre, 2021
ISSN: 2227-1899 | RNPS: 2301
http://rcci.uci.cu
Pág. 76-104

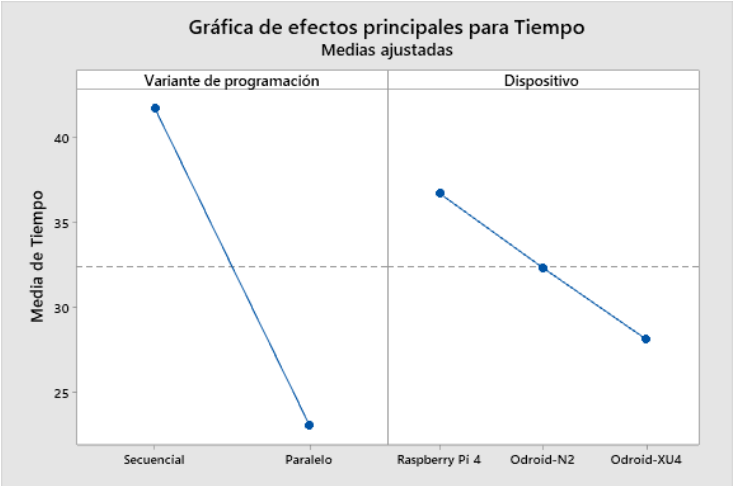

Fig. 9 – Gráfica de efectos principales para Tiempo en milisegundos.

El factor variante de programación al poseer en la gráfica la línea más inclinada, indica que mayor es la magnitud del efecto principal. El nivel en el que se obtienen los mejores valores de rendimiento es en la variante paralela, teniendo un efecto directo sobre el rendimiento. En el caso del factor dispositivo, el nivel en el que se obtiene una pequeña disminución del tiempo es con el Odroid-XU4. Con esta gráfica se corrobora lo obtenido en el diagrama de Pareto, pues el factor más significativo es la variante de programación.

Teniendo en cuenta que existen interacciones significativas entre los dos factores, también se debe examinar la gráfica de interacción que se muestra en la Figura 10. Cada punto de la gráfica de interacción muestra el tiempo de procesamiento medio con diferentes combinaciones de los niveles de los factores.

Revista Cubana de Ciencias Informáticas
Vol. 15, No. 4, Octubre-Diciembre, 2021
ISSN: 2227-1899 | RNPS: 2301
http://rcci.uci.cu
Pág. 76-104

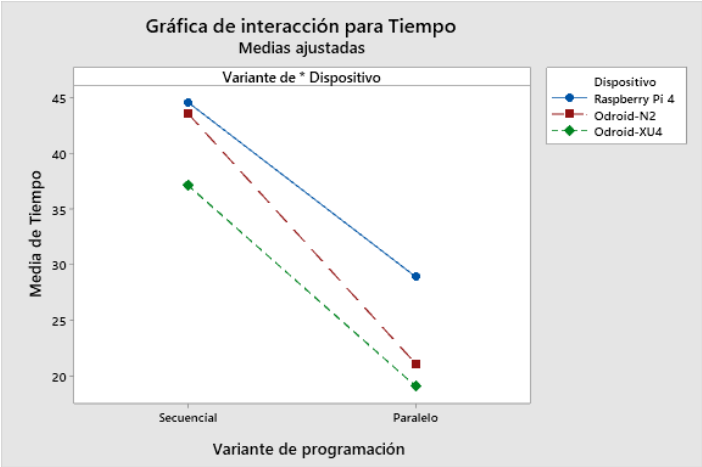

Fig. 10 – Gráfica de interacción para Tiempo en milisegundos.

La gráfica de interacción indica que con la utilización de la variante paralela y el Odroid-XU4, el software se ejecutó con menor tiempo. Con la ejecución del software secuencial se aumenta el tiempo de ejecución y la Raspberry Pi 4 y el Odroid-N2 no cumplen con el período de actualización del radar que es de 40 milisegundos. Además, al usar la variante paralela se observa que el Odroid-N2 tuvo la mayor reducción de tiempo, aproximadamente 2,06 veces, con respecto a la variante secuencial.

Conclusiones

El sistema propuesto posibilita la representación digital de señales cumpliendo con el período de actualización de un radar de seguimiento. Esta arquitectura hardware-software requiere un bajo costo de implementación, debido al empleo de ordenadores de placa reducida y software libre.

Las pruebas experimentales utilizando las métricas de rendimiento tiempo de ejecución, aceleración y eficiencia demostraron el adecuado desempeño de la variante paralela del software.

El análisis de los resultados permitió determinar que los factores tipo de dispositivo y variante de programación influyen en el tiempo de ejecución del software y de manera significativa el efecto variante de programación. Además, se recomienda el uso de la variante de programación paralela y del Odroid-XU4 para disminuir el tiempo de ejecución del software.

Referencias

Abdulsalam, S.; Lakomski, D., Et Al. Program Energy Efficiency: The Impact Of Language, Compiler And Implementation Choices. International Green Computing Conference, Texas, Usa, 3-5 November 2014, P. 1-6.

Antony, J. Design Of Experiments For Engineers And Scientists, 2nd Ed. Netherlands, Elsevier, 2014. 221 P.

Beaupre, T.; Doan, K., Et Al. Developing A Benchmark For Qt On Embedded Platforms. Massachusetts, Worcester Polytechnic Institute, 2018. 36 P.

Brizuela, L. L. Empleo De La Plataforma Arduino Para El Desarrollo De Actividades Prácticas Relacionadas Con Las Mediciones. Tesis De Ingeniería, Universidad Central ¨Marta Abreu¨ De Las Villas, Cuba, 2017.

Bugay, N. C. Embedded Graphical User Control Interface For An Advanced Battery Management System. Florida International University, Miami, 2016.

Butler, J. M. Tracking And Control In Multi-Function Radar. Tesis Doctoral, University College London, United Kingdom, 1998.

Cho, S.-Y. A Program Optimization Method For Embedded Software Developed Using Open Sources. International Journal On Advanced Science Engineering Information Technology 2018, 8 (4): P. 1692-1697.

Espinosa, E. C. Desarrollo De Un Software Para La Sincronización De Intersecciones Semafóricas. Tesis De Diploma, Instituto Superior Politécnico "José Antonio Echeverría", La Habana, Cuba, 2011.

Fominaya, J. A. G. Nuevas Técnicas De Localización, Clasificación E Identificación Para Radares De Vigilancia Superficial Y Alta Resolución En Escenarios Lpi. Tesis Doctoral, Universidad Politécnica De Madrid, España, 2004.

Georgiou, S.; Kechagia, M., Et Al. What Are Your Programming Language's Energy-Delay Implications? Proceedings Of The 15th International Conference On Mining Software Repositories, Gothenburg, Sweden, 27 May -3 June 2018, P. 303-313.

Goya, A. A.; Cañizares, D. G., Et Al. Análisis De La Escalabilidad Del Cálculo Paralelo De Medidas De Similitud Entre Pares De Genes. Revista Cubana De Ciencias Informáticas, 2016, 10 (3): P. 71-84.

Hundt, R. Loop Recognition In C++/Java/Go/Scala. 2011: P. 1-10.

Ieee, 2017. Standard For Radar Definitions. New York, Usa: Ieee Standards Association.

Janjusic, T. & Kartsaklis, C. Glprof: A Gprof Inspired, Callgraph-Oriented Per-Object Disseminating Memory Access Multi-Cache Profiler. Procedia Computer Science, 2015, 51: P. 1363-1372.

Kaushik, P. Radar Displays. International Journal Of Innovative Research In Technology, 2014, 1 (7): P. 472-476.

Kavyashree, V.; Madhu Chaitra, P. N., Et Al. A Radar Target Generator For Airborne Targets. International Journal Of Science Technology & Engineering, 2017a, 3 (11): P. 259-266.

Kavyashree, V.; Madhu Chaitra, P. N., Et Al. A Radar Target Generator For Airborne Targets. International Journal Of Advance Research And Innovative Ideas In Education, 2017b, 2 (15): P. 249-261.

Larabel, M. Odroid-N2: Análisis Del Buen Rendimiento En Pruebas Con Linux. Odroid Magazine, 2019, (66): P. 39-44.

Lee, J. Odroid-N2: Ejecuta Ubuntu 18.04 Y Android Pie Con El Más Reciente, Potente Y Rápido Ordenador De Placa Reducida De Hardkernel. Odroid Magazine, 2019, (63): P. 19-25.

Leppinen, H. Current Use Of Linux In Spacecraft Flight Software. Ieee Aerospace And Electronic Systems Magazine, 2017, 32 (10): P. 4-13.

Ltd, R. P. T. Raspberry Pi 4 Computer Model B. 2021a. [En Línea] Disponible En: Https://Datasheets.Raspberrypi.Org/Rpi4/Raspberry-Pi-4-Product-Brief.Pdf [Consultado El: 26 De Julio De 2021].

Ltd, R. P. T. Raspberry Pi 4 Tech Specs. 2021b. [En Línea] Disponible En: Https://Www.Raspberrypi.Org/Products/Raspberry-Pi-4-Model-B/Specifications/ [Consultado El: 26 De Julio De 2021].

Manasa, M. & Hemalatha, M. Design Of Generic Radar Data Visualizer Using Model View Controller Pattern. International Journal For Technological Research In Engineering, 2015, 2 (12): P. 2990-2992.

Medina, G. A.; Menéndez, J., Et Al. Comparative Study Of Performance And Productivity Of Mvc And Mvvm Design Patterns. Kne Engineering, 2018, 3 (1): P. 241-252.

Mena, O. L.; Cordovés, T. C., Et Al. Algoritmo Paralelo Para La Obtención De Predicados Difusos. Revista Cubana De Ciencias Informáticas, 2017, 11 (2): P. 117-133.

Novo, J. A. C. Aceleración Y Optimización Del Consumo Energético De Clasificadores En Cascada Para La Detección De Rostros Sobre Arquitecturas Asimétricas. Tesis Doctoral, Universidad De Granada, Granada, 2019.

Österberg, E. Profiling Memory Accesses On The Odroid-Xu4. Thesis Basic Level, Uppsala Universitet, Suecia, 2017.

Paula, C. E. Análisis Del Odroid-N2. Odroid Magazine, 2019, (69): P. 41-45.

Pereira, R.; Couto, M., Et Al. Energy Efficiency Across Programming Languages: How Do Energy, Time, And Memory Relate? Proceedings Of The 10th Acm Sigplan International Conference On Software Language Engineering, Vancouver, Canada, 23-24 October 2017, P. 256-267.

Pereira, R.; Couto, M., Et Al. Ranking Programming Languages By Energy Efficiency. Science Of Computer Programming, 2021, 205: P. 102609.

Piñero, M.; Marin, A., Et Al. Buenas Prácticas Para Prevenir Los Riesgos De La Eficiencia Del Desempeño En Los Productos De Software. Revista Cubana De Ciencias Informáticas, 2021, 15 (1): P. 89-113.

Ramos, T. R. B.; Pupo, S. D. C., Et Al. Computador De A Bordo: Una Solución Nacional. Revista De Ingeniería Electrónica, Automática Y Comunicaciones, 2021, 42 (1): P. 1-20.

Ravindra, C.; Rajkumar, S., Et Al. Design And Implementation Of Radar Console Displays For Multi Object Tracking Radar Using Qt-Ide. 11th International Radar Symposium India, India, 12-16 December 2017, P. 1-4.

Rischpater, R. Application Development With Qt Creator, 2nd Ed. Birmingham, United Kingdom, Packt Publishing Ltd, 2013. 264 P.

Rondx, J. Minería De Criptomonedas: Cómo Ganar Monedas Verium Con Tu Odroid. Odroid Magazine, 2019, (72): P. 4-11.

Roy, R. & Bommakanti, V. User Manual Odroid Xu4. 2017. [En Línea] Disponible En: Https://Magazine.Odroid.Com/Wp-Content/Uploads/Odroid-Xu4-User-Manual.Pdf [Consultado El: 21 De Julio De 2021].

- Saputera, Y. P.; Sulistyaningsih, Et Al. Radar Software Development For The Surveillance Of Indonesian Aerospace Sovereignty. International Conference On Electrical Engineering And Computer Science, Pangkal, Indonesia, 2-4 October 2018, P. 189-194.

Seo, S.; Kim, J., Et Al. An Analysis Of Embedded Operating Systems: Windows Ce, Linux, Vxworks, Uc/Os-Ii And Osek/Vdx. International Journal Of Applied Engineering Research, 2017, 12 (18): P. 7976-7981.

Servicio De Informática, A. N. M. Modelo Vista Controlador. 2021. [En Línea] Disponible En: Https://Si.Ua.Es/Es/Documentacion/Asp-Net-Mvc-3/1-Dia/Modelo-Vista-Controlador-Mvc.Html [Consultado El: 13 De Junio De 2021].

Shengwen, F.; Licong, C., Et Al. Software Design Of Wind Power Supervisory Control System Based On Embedded Qt. 2011 International Conference On Electrical And Control Engineering, Yichang, China, 16-18 September 2011, P. 3628-3631.

Singhal, S. P.; Gupta, S., Et Al. Profiling Minisat Based On User Defined Execution Time--Gprof. Arxiv E-Prints Arxiv:1909.13058, 2019: P. 1-13.

Stamatović, M.; Jevtić, M., Et Al. Modern Air Situation Picture Display For Air Surveillance Radar Applications. Telfor Journal, 2013, 5 (1): P. 54-59.

Stimson, G. W.; Griffiths, H. D., Et Al. Stimson's Introduction To Airborne Radar, 3rd Ed. Usa, Scitech Publishing, 2014. 774 P.

Sulistyaningsih; Saputera, Y. P., Et Al. Design Of Radar Display Of Indonesian Airspace Monitoring Application. Telkomnika, 2019, 17 (3): P. 1176-1184.

Thelin, J. Foundations Of Qt Development. United States Of America, Apress, 2007. 528 P.

Conflicto de interés

Los autores autorizan la distribución y uso de su artículo.

Contribuciones de los autores

1. Conceptualización: Lisvan Guevara Trujillo
2. Curación de datos: Bárbaro Nicolás Socarras Hernández y Leandro Zambrano Méndez
3. Análisis formal: Leandro Zambrano Méndez
4. Adquisición de fondos: -
5. Investigación: Lisvan Guevara Trujillo
6. Metodología: Bárbaro Nicolás Socarras Hernández y Wenny Hojas-Mazo
7. Administración del proyecto: Lisvan Guevara Trujillo
8. Recursos: Bárbaro Nicolás Socarras Hernández y Leandro Zambrano Méndez

9. Software: Lisvan Guevara Trujillo
10. Supervisión: Wenny Hojas-Mazo y Margarita André Ampuero
11. Validación: Leandro Zambrano Méndez
12. Visualización: Lisvan Guevara Trujillo
13. Redacción – borrador original: Lisvan Guevara Trujillo
14. Redacción – revisión y edición: Wenny Hojas-Mazo y Margarita André Ampuero