DESDE 2013
https://repository.uaeh.edu.mx/revistas/index.php/icbi/issue/archive
Pädi Boletín Científico de Ciencias Básicas e Ingenierías del ICBI
ISSN: 2007-6363

Publicación Semestral Pädi Vol. 12 No. Especial 2 (2024) 74-79

Comparativa de un sistema de visión de tiempo real bajo Xenomai y PREEMPT_RT
Comparative of a real-time vision system under Xenomai and PREEMPT_RT

J. L. Muñoz-Torres[a,*], E. Bugarin-Carlos[a], E. Rodríguez-Orozco[a], J. A. Rojas-Quintero[a], A. Y. Aguilar-Bustos[a]

[a] *Departamento de Ingeniería Eléctrica y Electrónica, Tecnológico Nacional de México / IT de Ensenada, 22780, Ensenada, Baja California, México.*

Resumen

Este artículo presenta el diseño y programación de un sistema de visión sobre dos sistemas operativos de tiempo real. Lo anterior con la finalidad de hacer una comparación de su rendimiento. Los sistemas operativos utilizados fueron Xenomai y Linux con el parche PREEMPT_RT. La estructura del sistema de visión en ambos casos se divide en tres hilos: el primero encargado de la adquisición y el procesamiento de las imágenes, el segundo que muestra las imágenes en pantalla y el último para la transferencia de datos (y posterior análisis fuera de línea). El objetivo de esta estructura planteada es la optimización de los tiempos de ejecución; los cuales pueden ser menores comparados con el empleo de un solo proceso o hilo para todo el sistema de visión. Finalmente, se describen experimentos haciendo énfasis en la periodicidad lograda realizando un procesamiento de imagen correspondiente al cálculo de un centroide que da información sobre el movimiento de un péndulo simple.

Palabras Clave: Visión por computadora, sistemas de visión de tiempo real, sistemas operativos de tiempo real.

Abstract

This article presents the design and programming of a vision system on two real-time operating systems. The above is for the purpose of making a comparison of their performance. The operating systems used were Xenomai and Linux with the PREEMPT_RT patch. The structure of the vision system in both cases is divided into three threads: the first one in charge of acquiring and processing the images, the second one that displays the images on the screen and the last one for the data transfer (and subsequent analysis out of line). The objective of this proposed structure is the optimization of the execution times; which may be smaller compared to using a single process or thread for the entire vision system. Finally, experiments are described emphasizing the periodicity achieved by performing an image processing corresponding to the calculation of a centroid that gives information about the movement of a simple pendulum.

Keywords: Computer vision, real-time vision systems, real-time operating systems.

1. Introducción

Un sistema de tiempo real es aquel en el cual los aspectos de comportamiento temporal forman parte de su especificación (Crespo y Alonso, 2006). El correcto funcionamiento de estos sistemas no sólo depende de la exactitud de los resultados aritméticos y lógicos, sino también del momento en que éstos son producidos. Esto ocurre frecuentemente en sistemas que deben interactuar con un entorno que se modifica constantemente, como es el caso de sistemas de control de procesos, robótica, telecomunicación y realidad virtual.

Si el sistema de tiempo real incluye una computadora entonces debe utilizarse un sistema operativo de tiempo real (SOTR). Los SOTRs dan preferencia a los procesos prioritarios, brindan soporte para el manejo de tareas y sincronización, manejo de memoria, comunicación entre procesos, manejo de dispositivos; así como del reloj de la computadora (González *et al.*, 2017).

Existen dos tipos de SOTRs: duros (hard) y suaves (soft). Los primeros son aquellos donde es absolutamente imperativo que las respuestas se produzcan dentro de los límites de tiempo especificados; mientras que en los segundos el tiempo es importante, pero el sistema seguirá funcionando si el tiempo de respuesta sobrepasa el límite de tiempo ocasionalmente. En un sistema computacional de tiempo real la computadora interactúa normalmente con algún equipamiento físico y se

*Autor para la correspondencia: alm17760517@ite.edu.mx
Correo electrónico: alm17760517@ite.edu.mx (Jorge Leonardo Muñoz-Torres), ebugarin@ite.edu.mx (Eusebio Bugarin-Carlos), erodriguez@ite.edu.mx (Eduardo Rodríguez-Orozco), jrojas@ite.edu.mx (Juan Antonio Rojas-Quintero), aaguilar@ite.edu.mx (Ana Yaveni Aguilar-Bustos).

Historial del manuscrito: recibido el 15/12/2023, última versión-revisada recibida el 15/12/2023, aceptado el 08/03/2024, publicado el 22/04/2024. **DOI:** https://doi.org/10.29057/icbi.v12iEspecial2.12268

dedica a monitorear o controlar la operación de dicho equipamiento; a esto se le puede conocer como un sistema embebido de tiempo real (Burns y Wellings, 2002).

Existen muchos SOTRs, de los cuales se destacan Xenomai y Linux con el parche PREEMPT_RT. Como ejemplo donde se ha utilizado el segundo de estos SOTRs se tiene el presentado por Rodríguez *et al*. (2021) donde se aprovecha para la programación de un sistema de visión en un robot humanoide. En Johansson (2018) se realiza una validación de Xenomai sobre una Raspberry pi 3 midiendo la latencia en una señal. Otro ejemplo similar al anterior es el descrito en Adam (2021) donde se evalúa el rendimiento de PREEMPT_RT montado sobre las tarjetas Raspberry Pi y BeagleBoard midiendo la periodicidad de tareas.

Xenomai es un proyecto de software libre en el que ingenieros con una amplia experiencia colaboran para construir un núcleo de tiempo real robusto y eficiente con recursos para Linux (ver https://source.denx.de/Xenomai/xenomai/-/wikis/Start_Here#user-content-how-does-xenomai-deliver-real-time). Xenomai entrega tiempo real de dos formas:

- Con el núcleo Cobalt que complementa al núcleo o kernel de Linux (configuración de doble núcleo). El cual se encarga de los tiempos críticos de las actividades; como el manejo de interrupciones y la programación de tiempo real de hilos. El núcleo Cobalt tiene mayor prioridad sobre el kernel nativo de Linux.
- Utilizando las capacidades de tiempo real del kernel nativo de Linux, formando el núcleo Mercury (configuración de núcleo único).

Por otro lado, el parche PREEMPT_RT para Linux se conforma de una serie de parches primarios trabajados por un grupo de desarrolladores expertos; la primera versión de este parche fue basado en el Kernel 2.6.11 del año 2005. El objetivo de este parche es modificar el rendimiento del sistema operativo de uso general Linux para que trabaje con predictibilidad y bajas latencias, a fin de permitir que los programadores puedan desarrollar aplicaciones de tiempo real fácilmente (Reghenzani *et al.,* 2019).

Para verificar el rendimiento de un SOTR diversos autores han utilizado una gran variedad de procedimientos; como el presentado en Brown y Martin (2010) donde se compara el rendimiento del parche PREEMPT_RT, Xenomai y el kernel nativo de Linux montado sobre un hardware específico (BeagleBoard C4) midiendo el tiempo de envío y recepción de datos por los puertos GPIO. Otro ejemplo es el mostrado en González *et al*. (2017) donde se compara el parche PREEMPT_RT y Xenomai montado sobre una Raspberry Pi 3 utilizando un algoritmo para inversión de matrices; además de comparar el rendimiento utilizando prioridades y políticas de planificación.

Ahora bien, los sistemas de visión son aquellos que proporcionan información por medio de la captura de imágenes y su procesamiento (Frost, 2018). Estos sistemas de visión se componen de cámaras con óptica especializada; de hardware y de software necesarios para procesar, analizar y medir características del entorno y estar en condiciones de tener una buena toma de decisiones.

La visión por computadora es el estudio del reconocimiento y localización de objetos mediante el procesamiento de imágenes, con el objetivo de la construcción de sistemas con capacidades similares a la visión humana. El objetivo de la visión por computadora es extraer características de una imagen para su descripción e interpretación (Sucar y Gómez, 2011).

Algunos usos para un sistema de visión se muestran en Bugarin y Aguilar (2014) donde se emplea un sistema de visión de tiempo real para la propuesta de un controlador visual (control servovisual) en la formación de dos robots móviles bajo el esquema líder seguidor; en Kanellakis y Nikolakopoulos (2017) donde se presentan las diversas técnicas de visión por computadora aplicadas a robots móviles (control de movimiento, estimación de posición, mapeo y detección de objetos, etc.); en Ruijiang y Yan (2001) donde se describe el diseño de un sistema de visión de tiempo real para la detección de personas realizando diversas acciones.

Finalmente, con la intención de mostrar desempeños de un sistema de visión de tiempo real para el control servovisual de robots bajo Xenomai y PREEMPT_RT, en este artículo se presenta una comparativa utilizando ambos SOTRs programados en un procesador Intel.

2. Desarrollo

En este apartado se presentan las especificaciones del sistema de cómputo, de la cámara y de los SOTRs utilizados en la realización de la comparativa del sistema de visión de tiempo real. La estructura del programa diseñado se describe por medio de un diagrama de flujo.

2.1. Especificaciones del hardware y software

La Tabla 1 muestra las especificaciones del sistema de cómputo que se emplea para programar el sistema de visión de tiempo real. En la Tabla 2 se presentan los datos de la cámara. Finalmente, en la Tabla 3 se muestran las versiones de los sistemas operativos utilizados.

Tabla 1: Especificaciones del sistema de cómputo

Sistema de cómputo	
Marca	Dell
Modelo	Latitude e6330
Procesador	Intel i5 de 2 núcleos a 2.70 GHz
Memoria RAM	8 GB

Tabla 2: Especificaciones de la cámara

Cámara Flea3	
Marca	Flir
Modelo	FL3-U3-13S2C-CS
Resolución máxima	1328 x 1048
Cuadros por segundo	120
Colores	Escala de grises y color
Protocolo	USB 3.0

Tabla 3: Sistemas operativos de tiempo real (SOTRs)

Sistemas operativos de tiempo real	
SOTR 1	Linux PREEMPT_RT 5.10.47-rt46
SOTR 2	Xenomai 3.1(doble núcleo)

2.2. Estructura del programa

El diagrama de flujo (ver Figura 1) muestra la estructura del programa diseñado. El algoritmo inicia con la configuración de la cámara (resolución, brillo y formato de color). El algoritmo contiene 3 hilos. El primero es el que se encarga de la adquisición de las imágenes y su procesamiento (cálculo de un centroide) por un determinado tiempo (por ejemplo, 30 segundos). El segundo hilo muestra en pantalla las imágenes. Las imágenes serán visualizadas en pantalla sólo cada cierto número de ellas (por ejemplo, cada NF = 10 imágenes) para no sobrecargar al sistema y que se garantice la operación de tiempo real (no interesa tanto la visualización, pero sí el procesamiento de todas las imágenes). Y el último hilo transfiere a un archivo de texto los datos del experimento (tiempos y las coordenadas del centroide). Es importante mencionar que la cámara se configuró para funcionar en escala de grises "mono 8" con una resolución de 640 x 420 píxeles.

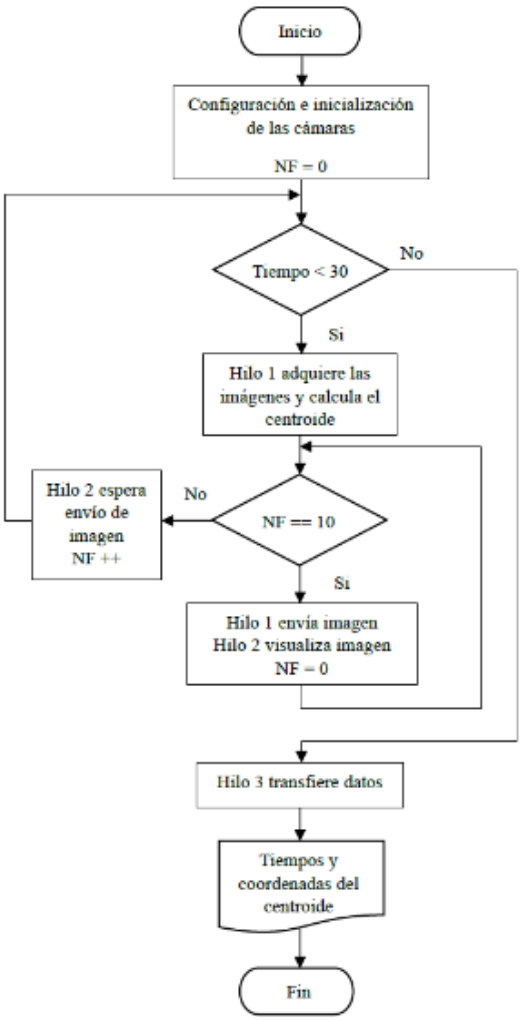


Figura 1: Diagrama de flujo del programa del sistema de visión de tiempo real.

2.3. Área de pruebas

El área de pruebas donde se realizaron los experimentos se presenta en la Figura 2. Para las pruebas se utilizó un péndulo que consiste en una esfera de color negro sostenida por una cuerda; la cual está atada a una estructura de madera. A esta esfera es a la que se le calcula el centroide. Para reducir tiempos no se procesa la imagen completa sino solo a una ventana que encierra a la esfera mediante segmentación binaria y seguimiento de la característica de la imagen (Bugarin y Aguilar, 2014).

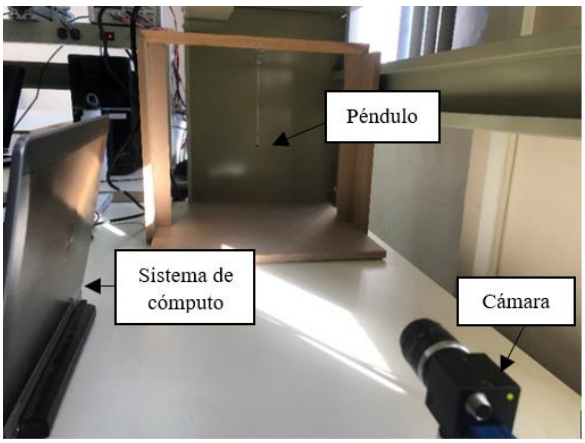


Figura 2: Área de pruebas.

3. Resultados

En este apartado se describen los resultados de los experimentos realizados con el sistema de visión programado tanto en Xenomai como en PREEMPT_RT. Para las pruebas se tomó un tiempo de 30 segundos. Los experimentos se desarrollaron con el péndulo en movimiento (con movimientos diferentes para cada corrida por supuesto; y con ligeras diferencias en la postura de la cámara); es decir, cada experimento consistió en soltar el péndulo desde una posición inicial alejada de la vertical y con el algoritmo grabar y procesar su movimiento durante 30 segundos. La Figura 3 presenta resultados para el sistema de visión programado en Xenomai. Y la Figura 4 corresponde a los resultados para el sistema de visión programado en PREEMPT_RT. En el inciso (a) de estas figuras se muestra el movimiento que realizó el centroide del péndulo visto desde la perspectiva de la cámara, en (b) se presenta el movimiento del centroide del péndulo en el eje x contra el tiempo y en (c) el movimiento en el eje y contra el tiempo.

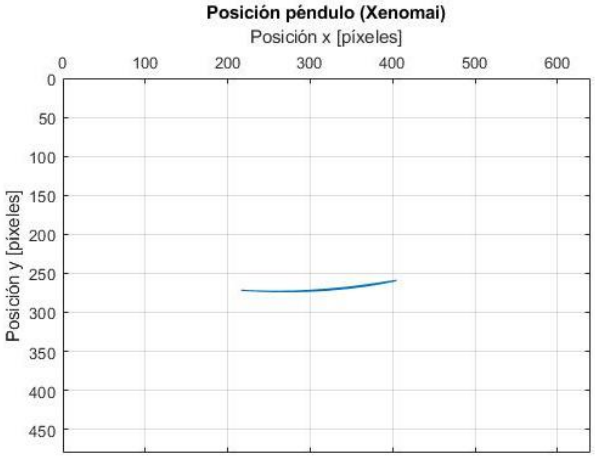
Para el análisis de la periodicidad de los experimentos se tomaron en cuenta las gráficas del tiempo que tarda cada ciclo (entre imágenes consecutivas procesadas) en ejecutarse. En las figuras 5 y 6 se muestra el tiempo de ciclo para cada caso.

En las figuras 7 y 8 se observan los histogramas correspondientes a los experimentos realizados en Xenomai y PREEMPT_RT (en adelante, por cuestiones de tiempo consumido por la inicialización de la cámara, se presentan datos estadísticos sin considerar los primeros 10 ciclos). En estos histogramas se muestran el número de ciclos que se ejecutaron por tiempo de ciclo.

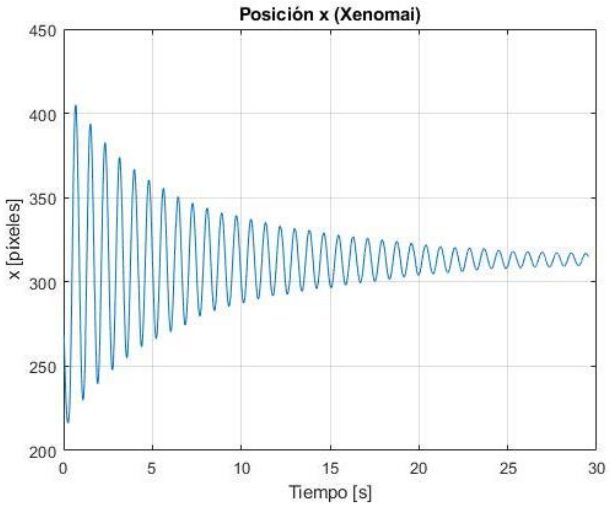
La Tabla 4 ilustra los datos estadísticos correspondientes a los tiempos de ciclo para el hilo 1 (el hilo de procesamiento) de cada experimento.

Tabla 4: Datos estadísticos del algoritmo (hilo 1)

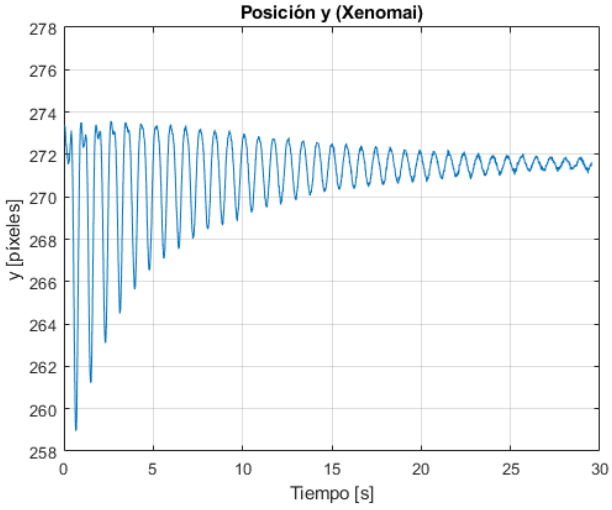
	Xenomai [s]	PREEMPT_RT [s]
Media	0.008329	0.008327
Mayor tiempo	0.008974	0.009895
Menor tiempo	0.007459	0.006504
Desviación estándar	0.000113	0.0001962



(a)

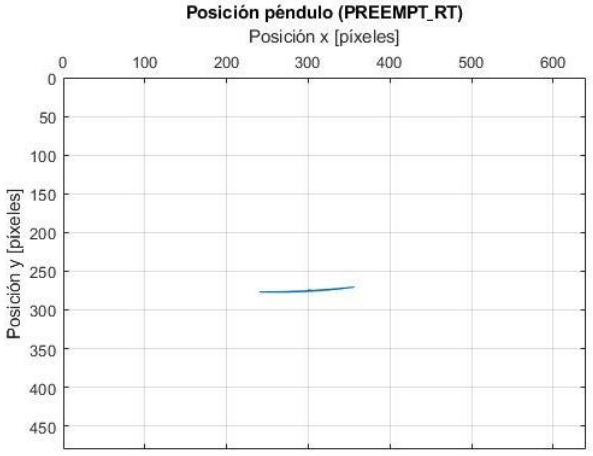


(b)

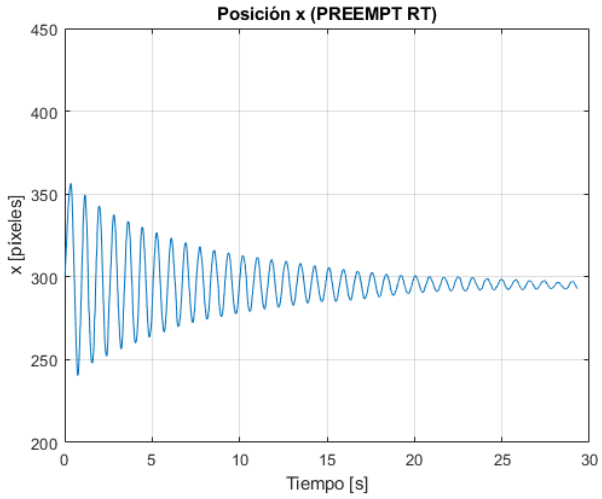


(c)

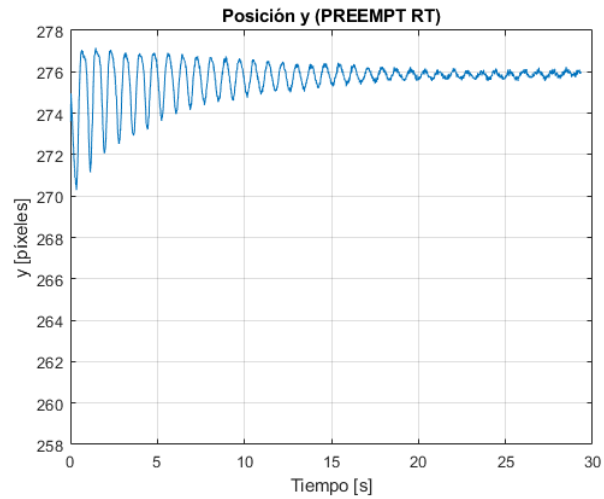
Figura 3: Resultados del sistema de visión programado en Xenomai. (a) Posición del péndulo. (b) Posición en el eje x. (c) Posición en el eje y.



(a)



(b)



(c)

Figura 4: Resultados del sistema de visión programado en PREEMPT_RT. (a) Posición del péndulo. (b) Posición en el eje x. (c) Posición en el eje y.

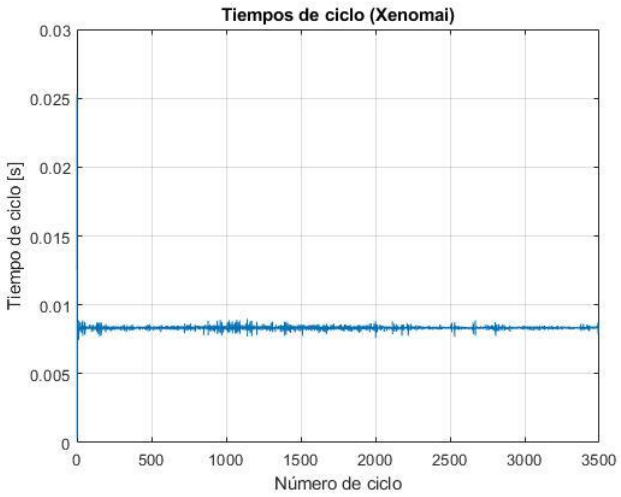


Figura 5: Gráfica de tiempos de ciclo del hilo 1 (Xenomai).

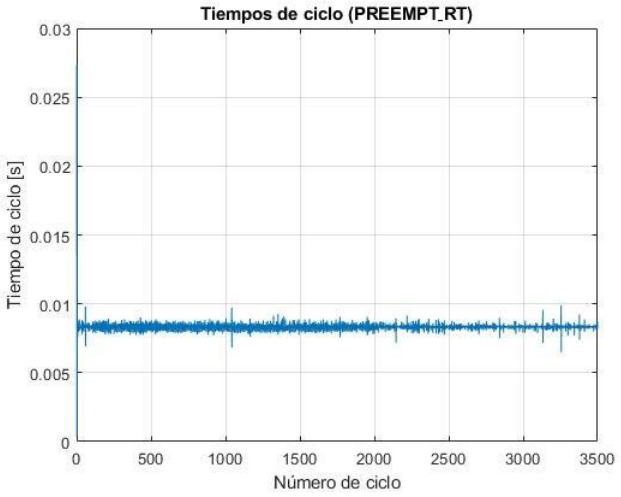


Figura 6: Gráfica de tiempos de ciclo del hilo 1 (PREEMPT_RT).

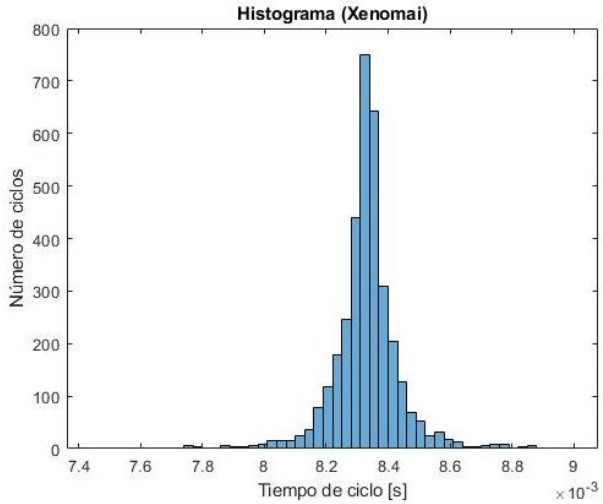


Figura 7: Histograma de tiempos de ciclo del hilo 1 (Xenomai).

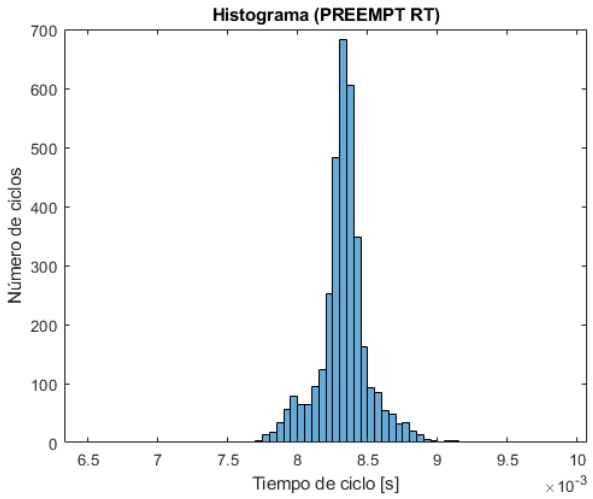


Figura 8: Histograma de tiempos de ciclo del hilo 1 (PREEMPT_RT).

4. Conclusiones

En base a los resultados de la Tabla 4 se tiene que la diferencia entre los tiempos de ciclo de los experimentos realizados tanto en Xenomai como en PREEMPT_RT es mínima. Por lo tanto, utilizar cualquiera de estos dos SORTs será eficiente debido a que tendrán un rendimiento similar. Debe mencionarse que todavía falta por explorar la opción de núcleo único de Xenomai, esto queda como trabajo a futuro para este artículo. El algoritmo diseñado en este trabajo utiliza 3 hilos que favorecen la eficiencia del programa; ya que todas las tareas que realiza el algoritmo son separadas haciendo a la aplicación más óptima tanto en uso de recursos del sistema como en el rendimiento de éste; refiriéndose a los tiempos de ejecución. En este trabajo solamente se realizaron comparaciones con un algoritmo de procesamiento de imágenes ligero; como lo es el cálculo de un centroide. De manera que como trabajo a futuro también se contempla abordar algoritmos más complejos y la utilización de más cámaras en el sistema de visión.

Agradecimientos

Se agradece el apoyo financiero del TecNM, del Instituto Tecnológico de Ensenada y del CONAHCYT. Este trabajo fue desarrollado en el marco de las actividades de la red internacional denominada “Red internacional de control y cómputo aplicados” soportada por TecNM.

Referencias

Adam, G. (2021). Real-Time Performance and Response Latency Measurements of Linux Kernels on Single-Board Computers. Computers, vol. 10. p. 64.

Brown, J., Martin, B. (2010). How fast is fast enough? Choosing between Xenomai and Linux for real-time applications. Twelfth Real-Time Linux Workshop, p. 1- 17.

Bugarin, E., Aguilar-Bustos, A. Y. (2014). Control visual para la formación de robots móviles tipo uniciclo bajo el esquema líder-seguidor. Ingeniería, Investigación y Tecnología. XV(4): 593-602.

Burns, A., Wellings, A. (2002). Sistemas de Tiempo Real y Lenguajes de Programación, Addison Wesley, Madrid.

Crespo, A., Alonso A. (2006). Una panorámica de los sistemas de tiempo real. Revista Iberoamericana de Automática e Informática Industrial, vol. 4, p. 8–18.

Frost, S. (2018). Introducción a la visión artificial. p. 1-24.

González, D., Cano, J., Guevara, P. (2017). Análisis comparativo de los tiempos de ejecución sobre SBC dos sistemas operativos de tiempo real. Pistas Educativas, vol. 39, 572-585.

Johansson, G. (2018). Real-Time Linux Testbench on Raspberry Pi 3 using Xenomai. Institute School of Electrical Engineering and Computer Science, Stockolm, p. 1-80.

Kanellakis, C., Nikolakopoulos, G. (2017). Survey on Computer Vision for UAVs: Current Developments and Trends. Journal of Intelligent & Robotic Systems, vol. 87, p. 141–168.

Sucar, E., Gómez, G. (2011). Visión computacional. Instituto Nacional de Astrofísica, Óptica y Electrónica, México.

Reghenzani, F., Massari, G., Fornaciari, W. (2019). The Real-Time Linux Kernel: A Survey on PREEMPT_RT. ACM Computing Surveys, vol. 52, p. 1-36.

Rodríguez, E., Bugarin, E., Rojas, J. A., Aguilar, A. (2021). Preliminary design and experimental tests of a real-time stereoscopic foveated vision system. Memorias del XXIII Congreso Mexicano de Robótica 2021, p. 26–31.

Ruijiang, L., Yan, G. (2001). Real-time stereo tracking of multiple moving heads. In Proceedings IEEE ICCV Workshop on Recognition, Analysis, and Tracking of Faces and Gestures in Real-Time Systems, p. 55-60.