# Arquitectura Embebida de Tiempo Real para Sistemas Domóticos

José Félix, Antonio Gutiérrez, Walter Mata, Apolinar González, Facultad de Ingeniería Mecánica y Eléctrica Universidad de Colima, Av. Universidad # 333, Colima, México,{jfelix, jose_gutierrez, wmata, apogon}@ucol.mx

***Resumen*— El presente artículo propone una plataforma para implementar aplicaciones domóticas con sistemas embebidos distribuidos y redes de sensores en ambientes inalámbricos mediante dos enfoques desconocidos en los sistemas domóticos existentes. El primer enfoque utiliza un hipervisor que ejecuta varios sistemas operativos independientes con aplicaciones divididas en dominios directamente sobre el hipervisor para soportar aplicaciones con requerimientos de seguridad crítica y en tiempo real; mientras que el segundo enfoque es mediante la implementación de un middleware llamado micro DDS (µDDS) el cual permite la interoperabilidad entre sistemas embebidos al proporcionar una API (por sus siglas en inglés, Application Programming Interface) para el desarrollo de aplicaciones basadas en el protocolo que sigue la especificación del estándar del Sistema Distribuido de Datos (DDS) [1]. Como resultado de la integración de estas tecnologías tenemos una plataforma informática robusta y segura para aplicaciones orientadas a la domótica y ambientes inteligentes.**

***Palabras Clave*— Sistemas embebidos, Virtualización, Hipervisor, Middleware, µDDS, Sistemas distribuidos, Tiempo real, DDS.**

## I. Introducción

La presencia cada vez mayor de sistemas embebidos en productos y servicios en la actualidad da una gran oportunidad de crecimiento en el futuro para diferentes áreas [2]. Los sistemas embebidos ya desempeñan un papel importante no sólo en los dispositivos de electrónica de uso común, sino también en sistemas de seguridad crítica [3]. Como consecuencia, existe interés tanto en lo conceptual como en lo práctico para desarrollar herramientas que permitan crear aplicaciones sobre sistemas embebidos. Dichos sistemas están creciendo en popularidad al existir una gran variedad de aplicaciones haciendo cada vez nuestras vidas más dependientes de los mismos. Se ha estimado que el 99% de la producción mundial de microprocesadores se utilizan en sistemas embebidos [4], siendo prácticamente invisibles para los consumidores. Esto incluye no sólo aplicaciones de seguridad crítica tales como controladores de dispositivos para aviones, automóviles, ferrocarriles, industria aeroespacial, salud y dispositivos médicos, sino también en comunicaciones, sistemas móviles, medio ambiente, automatización de casas, teléfonos móviles, PDA's, reproductores de DVD, cámaras, etc [5] [6] [7] [8].

El desarrollo de aplicaciones embebidas está entrando en nuevos dominios debido a la disponibilidad de los nuevos procesadores de alta velocidad que proporcionan una mayor potencia de procesamiento y menor consumo de energía a un costo menor. Una de las nuevas áreas donde los sistemas embebidos están teniendo un gran desarrollo es la domótica, la cual es un conjunto de sistemas que enriquecen una vivienda mediante tecnología que responde a los objetivos del usuario [9] en sus tareas comunes de una forma automática y dinámica sin interrumpir la ejecución del sistema [10]. Algunas de las plataformas para aplicaciones domóticas más conocidas en la actualidad son X10 [11], KNX, EIB, EHS, Batibus, Lonworks [12], de las cuales la mayoría su funcionamiento es centralizado y dependen de las líneas eléctricas lo que las hace costosas y dificulta su instalación. A partir de los sistemas embebidos distribuidos inalámbricos le damos a nuestra plataforma primero un procesamiento distribuido y al ser inalámbrico bajo el estándar de comunicación IEEE 802.15.4 [13] reducimos el consumo energético, el costo y además facilita su instalación.

Como resultado de la nueva generación de hardware, hay un creciente interés en permitir que varias aplicaciones compartan un solo procesador y memoria, concepto que introducimos en nuestra plataforma. Para facilitar este tipo de arquitectura, el tiempo de ejecución y la memoria de cada aplicación debe ser protegida de otras aplicaciones en el sistema, esta protección es la que nos brinda niveles de seguridad. En un sistema operativo con particiones, la memoria se divide entre las mismas utilizando el aislamiento de los subsistemas. Este concepto se conoce comúnmente como virtualización [14]. La importancia de la virtualización en el ámbito de la computación embebida está comenzando a despegar llegando no solo a áreas como la aeronáutica y cuestiones espaciales sino también a áreas como los ambientes inteligentes. Hipervisor [15] es el término que se refiere a esta capa de software (en combinación con los mecanismos de hardware) que proporciona esta virtualización la cual permite que un solo microprocesador pueda ejecutar (espacial y temporal) varios sistemas operativos independientes en un único equipo esto nos permitirá que las aplicaciones sobre la plataforma estén integradas y a la vez tengan cierta independencia. Una de las características más importantes de un hipervisor [16] es que se

debe limitar gastos generales para que las aplicaciones puedan ejecutarse en casi la misma velocidad que en el sistema operativo nativo donde la arquitectura también juega un papel importante. El desafío es mantener un cierto grado de flexibilidad para aumentar la reutilización de componentes de software. El concepto de hipervisor dentro de la domótica es totalmente nuevo y dentro del presente desarrollo su tarea principal será brindar las características de tiempo real para aplicaciones de seguridad crítica y el aislamiento de las aplicaciones de tal manera que al presentarse alguna falla en cierta área del sistema este tendrá la capacidad de seguir funcionando.

Con la implementación de un middleware llamado µDDS (micro DDS) sobre la plataforma, esta permite a los sistemas embebidos y redes inalámbricas el acceso para operar sin problemas, al proporcionar una interfaz de desarrollo de aplicaciones así como la interoperabilidad del protocolo basado en la distribución estándar de la especificación de servicio de datos (DDS) [17].

Dentro del desarrollo de este artículo podremos encontrar la descripción general de la plataforma propuesta y como maneja el sistema de red, la interfaz de usuario, la automatización y control y los servicios multimedia así como el porqué de la integración de los conceptos de hipervisor y middleware observando los beneficios que podemos obtener de un hipervisor en una plataforma para aplicaciones domóticas y los servicios que nos proporciona el middleware para las comunicaciones de sistemas distribuidos. Se realiza al final una evaluación de desempeño con la cual se obtienen resultados tangibles del funcionamiento de hipervisor y el middleware sobre la plataforma para las aplicaciones domóticas.

## II. PLATAFORMA PARA APLICACIONES DOMÓTICAS SEGURAS

Hoy, el reto principal en una casa inteligente es crear una conexión de software simple entre todos los dispositivos inalámbricos, para asegurar que puedan trabajar juntos de manera óptima en la generación de un nuevo conjunto de escenarios funcionales para los usuarios y al mismo tiempo mantener un cierto grado de flexibilidad. Sin embargo, cada componente o dispositivo a menudo realiza una sola función y es difícil de integrar con los demás. Los sistemas de seguridad en una casa no suelen estar integrados, por ejemplo las cámaras de seguridad de vídeo digital y puertas automáticas.

Para lograr lo anteriormente dicho, es necesaria una nueva arquitectura para casas inteligentes, una que pueda explotar el gran potencial de las nuevas tecnologías y que permita una mejor integración de dispositivos para así crear un entorno inteligente.

### *A. Plataforma Global*

En la Fig 1 se puede apreciar que la arquitectura global de la plataforma donde el hipervisor se ejecuta directamente en la arquitectura de hardware. El hipervisor completamente aislado de los distintos ámbitos, permite la comunicación a través de las colas de mensajes, hiperllamadas y memoria compartida. Como resultado, los distintos sistemas operativos se pueden ejecutar de forma independiente y el hipervisor en tiempo real, controla la ejecución de cada uno de ellos. El Middleware basado en el patrón P/S (por sus siglas en inglés Publish/Subscribe) del DDS implementa un subconjunto de las interfaces estándar de esta especificación, para realizar publicaciones y subscripciones en las aplicaciones domóticas desarrolladas. Estas aplicaciones pueden difundir y recolectar información a través de una interfaz P/S proporcionada por el middleware. La plataforma global muestra particiones diferentes, donde cada partición está conformada por uno o más procesos concurrentes (implementado por el sistema operativo de cada partición), que comparten el acceso a los recursos del procesador en función de las necesidades de la aplicación. El código de la partición puede ser una aplicación compilada para ser ejecutada sobre el hardware (máquina al desnudo), un sistema operativo en tiempo real (soporte en tiempo de ejecución) y sus aplicaciones, o un sistema operativo de propósito general y sus aplicaciones. El dominio puede ser descrito de la siguiente manera:

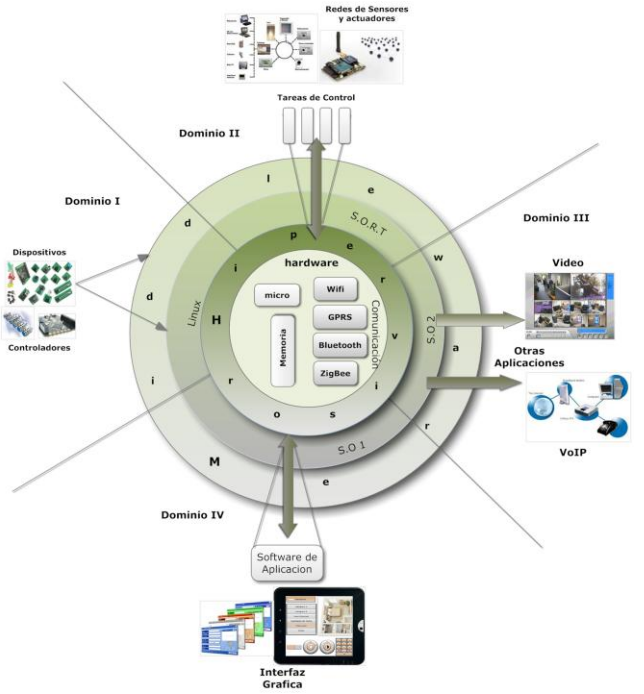


Fig. 1: Plataforma Global

#### *1) Sistemas de red*

La primera partición de la Fig.1 virtualiza los sistemas de red. Todos los componentes que se utilizan para conectar una máquina a otra son una fuente de vulnerabilidad. Las pilas de protocolos, tarjetas de interfaz, routers, switches y los medios de comunicación originalmente diseñados para mover datos de manera óptima. La aplicación y la pila de protocolos están protegidas unos de otros, lo que reduce considerablemente la vulnerabilidad del sistema total. Afortunadamente, las pilas de protocolos en movimiento fuera del núcleo en sus propias particiones pueden ser completamente transparentes para las aplicaciones que los utilizan.

*2) Interfaz de usuario*

Una interfaz de usuario (UI - User Interface por sus siglas en inglés) para la casa inteligente provee de un usuario el acceso a algunos o todos los dispositivos que están integrados en el hogar. Una vez que el usuario selecciona un dispositivo, su estado actual se puede mostrar, así como el menú de comandos disponibles. El equipo puede estar asociado a un determinado grupo de actividades, incluyendo la iluminación, entretenimiento, seguridad.

El acceso a cualquier dispositivo específico está sujeto a la autorización y la capacidad de las interfaces de usuario para controlar sus funciones. Los dispositivos de interfaz de usuario pueden tener varios niveles de sofisticación; por ejemplo, el dispositivo de interfaz puede reproducir música o facilitar el acceso a los sistemas de entretenimiento doméstico controlado por interfaces de usuario simple, mientras que las interfaces avanzados pueden controlar los teléfonos móviles, PDA's, pantallas táctiles y alta definición de interfaces multimedia, entre otros. Los dispositivos más avanzados tienen la ventaja de permitir un mejor uso de los gráficos y software, y puede presentar distintas pantallas.

Para esta aplicación se ha desarrollado una interfaz de usuario sobre el sistema operativo Android [18], con el SDK de este sistema en su versión 2.3. Como se mencionó esta interfaz consta de niveles de sofisticación, por lo tanto tendremos el desarrollo de aplicaciones locales como aplicaciones remotas con dispositivos móviles (celular, Tablet, Laptop, PDA, etc).

Para la comunicación con los dispositivos móviles existe un servidor el cual tiene intercomunicación con los dispositivos de entorno como sensores y actuadores a través del protocolo IEEE 802.15.4 y con los móviles a través de WIFI, mediante un servidor web, el cual hace la conexión del exterior hacia el interior, para acceder a los datos del sistema y poder realizar funciones de control y automatización.

Mediante esta interfaz podemos visualizar representaciones de la casa donde está instalado el sistema domótico, además nos brinda la información necesaria para conocer el estado en el que se encuentra nuestro hogar y a partir de estos datos poder tomar decisiones, cambiar y controlar los recursos a nuestra conveniencia en cuestión de seguridad, confort, ahorro energético, etc.


Fig. 2: Diseño de la Interfaz de Usuario

Se puede observar en la fig. 2 que la interfaz contiene un plano de la casa, en este se puede acceder a cada área y ver cuál es su estado. Una vez dentro se ve de manera gráfica una representación del lugar, así como información importante, tal como, temperatura, presencia, grado de luminosidad, estado de la alarma, entre otras. Además podemos hacer cambios como encender las luces, el aire acondicionado, monitorear el sistema de alarma entre otras aplicaciones, todo esto de manera remota.

En la Fig. 2 se puede aprecia la aplicación antes descrita funcionando sobre un teléfono inteligente (Smartphone), esto nos proporciona un nivel más alto de sofisticación. Esta aplicación podrá ser instalada sobre cualquier plataforma o dispositivo que tenga instalado android; se están realizando pruebas con celulares, tablets y con la conversión (porting) de este sistema para arquitectura x86, que en algunas aplicaciones tendrá el rol de servidor de datos. Esta aplicación es importante debido a que nos proporcionará la interacción del usuario con el sistema domótico de una manera fácil e intuitiva.

*3) Automatización y control*

Una casa inteligente típica consiste en sistemas heterogéneos, tales como los sistemas de acceso, controles de iluminación, gestión energética, seguridad y vigilancia digital, entre otros. Estos sistemas heterogéneos permiten a sus propietarios especificar las acciones y tareas que se llevarán a cabo en momentos específicos. Esto puede lograrse mediante el desarrollo de una arquitectura distribuida para dispositivos virtuales que se ejecutan en la parte superior de los niveles de aplicación y varios puntos de control en toda la casa, mientras que proporciona una interfaz de usuario para las personas que van administrar de forma remota.

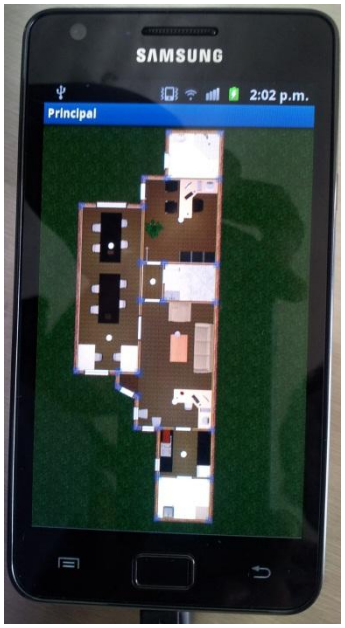


Fig. 3: Aplicación funcionando sobre Smart Phone con Android

En muchos casos, estas interfaces reflejan la interfaz basada en navegadores para el control y seguimiento. El alojamiento (hosting) virtual en módulos de terminales en el hogar

proporciona acceso a aplicaciones de mayor capacidad de procesamiento y una mayor conectividad, que es más preferible que operen a nivel local con más recursos limitados. Los algoritmos inteligentes en este tipo de sistemas heterogéneos pueden utilizar virtualizaciones de mayor poder de procesamiento, conectividad y tamaños de almacenamiento.

*4) Entretenimiento y servicio multimedia*

Uno de los dominios más populares para aplicaciones domóticas es el entretenimiento, y dentro de éste, uno de los mayores desafíos es la gestión del contenido. Los propietarios de viviendas están expuestos a una gran variedad de música, imágenes y medios de comunicación, gran parte de este contenido puede terminar en un solo dispositivo, en lugar de ser distribuidos entre los diferentes dispositivos que se utilizan de una forma más expandida entre un mayor número de miembros de la familia ubicados en diferentes lugares de la casa. La centralización de los datos crea dificultades para los habitantes de una casa porque necesitan cambiar de un dispositivo a otro para poder acceder a diferentes tipos de contenido. Usando la virtualización podría resolver este cuello de botella mediante la implementación de la virtualización de contenidos en una plataforma distribuida. Usando la virtualización también se garantiza la organización, el etiquetado y el análisis de todos los contenidos de entretenimiento almacenados de una manera más eficiente y accesible inmediatamente. Por otro lado, la virtualización también prepara el camino para el reconocimiento de la actividad de las personas en función de su recuperación y utilización de contenidos específicos.

## III. ARQUITECTURA DEL HIPERVISOR

XtratuM es un hipervisor desarrollado por el Instituto de Automática e Informática Industrial de la Universidad Politécnica de Valencia, que proporciona un marco para ejecutar varios sistemas operativos (tiempo real) en un entorno particionado y robusto que ha sido parcialmente financiado por el proyecto FRESCOR Europea y el Centro Nacional de dEtudes Espaciales de Francia (CNES, la agencia espacial francesa), y que se utilizará en el procesador LEON2.

XtratuM se puede utilizar para construir Multiples Niveles independientes de seguridad (Multiple Independent Levels of Security por sus siglas en inglés MILS) de la arquitectura [19]. La característica más interesante de XtratuM es su capacidad de compartir el mismo hardware entre varios sistemas operativos ejecutándose en forma concurrente [20].

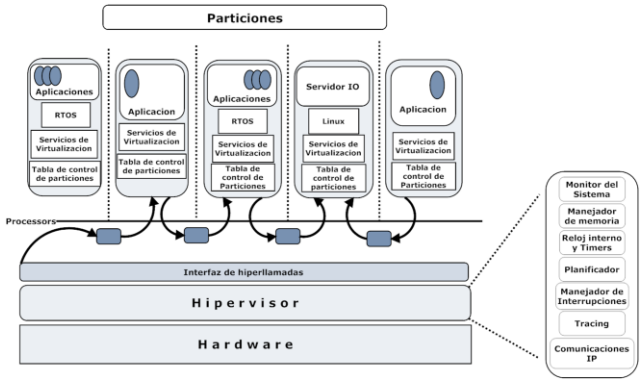


Fig. 4: Arquitectura interna de XtratuM

La Fig. 4 muestra la arquitectura interna de XtratuM, este dispone de un monitor del sistema, manejador de memoria, reloj interno, planificador, manejador de interrupciones y comunicaciones IP y proporciona estos servicios a las particiones, que, al estar en aislamiento nos brinda un nivel independiente de seguridad y a la ves la capacidad de interacción entre las múltiples particiones con las comunicaciones existentes.

### *A. Beneficios de un hipervisor dentro de la domótica*

La hipervisión de sistemas computacionales permite que un solo sistema informático o terminal pueda albergar varias máquinas virtuales al mismo tiempo y controlar varios aspectos de cada máquina virtual, incluyendo la configuración de los equipos, capacidades de red, etc. Estas cualidades permiten los siguientes tipos de configuración de uso: La utilización de recursos, entorno de red, la independencia de la plataforma, soporte para sistemas legados y de administración entre otros. En un ambiente de casa inteligente, la restricción de implementar la hipervisión está relacionada con los requisitos dinámicos. Hay un número de aplicaciones potenciales para hipervisión en los entornos doméstico inteligentes. Estas máquinas virtuales ocultas pueden servir a un variado número de funciones; por ejemplo, se pueden remplazar dispositivos físicos para entretenimiento, soporte remoto a servicios de salud y así poder definir su casa en módulos de control de tareas del hogar de forma automatizada.

## IV. IMPLEMENTACIÓN DEL UDDS PARA APLICACIONES DOMÓTICAS

La arquitectura de la plataforma de una casa inteligente que se ejecuta en cada nodo se muestra en la Fig. 5. Existen dos implementaciones, una para la plataforma XtratuM, estaciones base y para interfaces de usuario; y la otra para la plataforma de sensores y actuadores que utilizan los sistemas operativos de tiempo real. La arquitectura de comunicación para las dos plataformas de hardware que se ejecutan en el hipervisor XtratuM, Partikle OS y µDDS.

### *A. Middleware para sistemas embebidos inalámbricos de tiempo real*

La especificación [1] del DDS proporciona una interfaz de programación de aplicaciones (API), donde una aplicación distribuida puede utilizar el mecanismo publicar/suscribir de

comunicación centrado en los datos. Se basa en el modelo de la arquitectura impulsada por MDA (por sus siglas en inglés Model-Driven Arquitecture) [21] [4], que define un modelo independiente de plataforma (PIM). Como resultado, los desarrolladores de middleware pueden desarrollar modelos específicos de la plataforma (PSM), que se puede ajustar a los requisitos de aplicación específica, que permite a los desarrolladores construir implementaciones basadas en el DDS diferentes, dedicadas a necesidades muy específicas [1]. Uno de los aspectos importantes a considerar en el DDS es la calidad de servicio (QoS), que es un concepto que se utiliza para especificar ciertos parámetros relacionados con la forma de un servicio se entrega. Los QoS proporcionan la capacidad de controlar y limitar el uso de recursos como el ancho de banda de red, memoria, fiabilidad, puntualidad, y la persistencia de datos, entre otros.

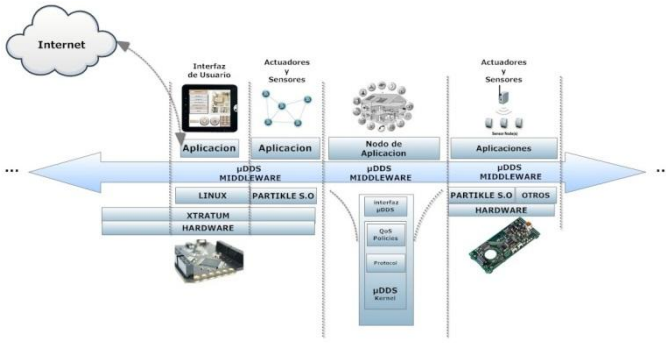


Fig. 5: Arquitectura de comunicaciones

µDDS es un middleware basado en el patrón P/S para sistemas inalámbricos integrados basados en la especificación DDS, el cual funciona mediante la implementación de un subconjunto de las interfaces estándar para la realización de suscripciones y publicaciones para ser utilizado por las aplicaciones desarrolladas. Diferentes protocolos de enrutamiento se puede utilizar para implementar la red, este middleware se está aplicando actualmente en los dispositivos Wi-Fi, que puede soportar topologías en estrella, árbol y malla. Los nodos pueden comunicarse con otros dispositivos mediante un módulo de comunicación inalámbrico.

## V. Evaluación de desempeño

La valoración inicial con XtratuM se realizó ejecutando 2 particiones, una con PaRTiKle y otra con GNU/Linux. Para efectos de este estudio, las variables que se midieron fueron cambio de contexto de la partición y la pérdida de rendimiento debido al número de particiones. El cambio de contexto es importante ya que en el momento de ejecución de las aplicaciones sobre la plataforma, el cambio de contexto entre sistemas operativos y aplicaciones es vital para un buen desempeño en cuestión de prioridades, además es uno de los niveles de seguridad que proporciona el hipervisor al tener dominios de ejecución. Mientras que la importancia de la perdida de rendimiento debido al número de particiones se considera al ser necesaria una respuesta de tiempo real en las aplicaciones orientadas a la domótica, a continuación describiremos estas pruebas antes mencionadas.

### *A. Pruebas de Desempeño de XtratuM*

#### *1) Cambio de Contexto de la Partición*

Cambio de contexto de partición, es el tiempo que necesita el hipervisor para cambiar entre dos particiones. Con el fin de determinar el tiempo de cambio, la actividad interna de XtratuM se cuantificó, añadiendo puntos de interrupción al principio y al final del código a medir. Uno de los objetivos de este estudio es determinar si el cambio de contexto aumenta la latencia cuando se utiliza un protocolo de comunicación. Esto es importante porque se minimiza el gasto de recursos el cual es un objetivo importante en cualquier entorno de ejecución para sistemas de tiempo real.

Para esta prueba se iniciaron dos particiones con el mismo código al mismo tiempo. El eje Y (Time) representa el tiempo en milisegundos en el cual se ejecuta el cambio de partición, mientras el eje X (samples) representa el número de muestras de cambios de contexto realizados durante la prueba.

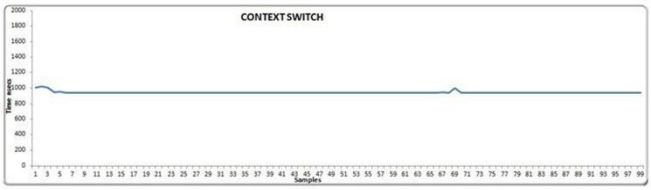


Fig. 6: Tiempo de Cambio de contexto.

La primera partición obtiene la hora actual en repetidas ocasiones y el rendimiento en un lapso de 100 milisegundos. La segunda partición recupera el tiempo actual una vez que es puesta en ejecución. Se midió el intervalo entre la primera partición y la segunda partición. La Fig.6 muestra que el tiempo promedio entre la primera y segunda partición se mantiene casi constante a un promedio de aproximadamente 0.9 microsegundos, mostrando un comportamiento casi constante entre dos particiones garantizando con esto la estabilidad y robustez necesaria para el soporte de nuestras aplicaciones sobre la plataforma.

### *B. Pruebas de Desempeño del µDDS*

#### *1) Caudal Eficaz (throughput)*

El caudal eficaz o también conocido como throughput es definido como el número total de mensajes (paquetes) transmitidos por periodo de tiempo. En esta prueba, el publicador envía datos a uno o más aplicaciones suscriptoras. El publicador anuncia a la aplicación suscriptora o suscriptoras que comenzará e inicia un reloj. Se puede especificar la duración de la prueba. El suscriptor comienza a contar el número de mensajes recibidos. Cuando el periodo de prueba termina el publicador anuncia al suscriptor que la prueba finalizó. Entonces el suscriptor divide el número de mensajes recibidos por el lapso de tiempo.

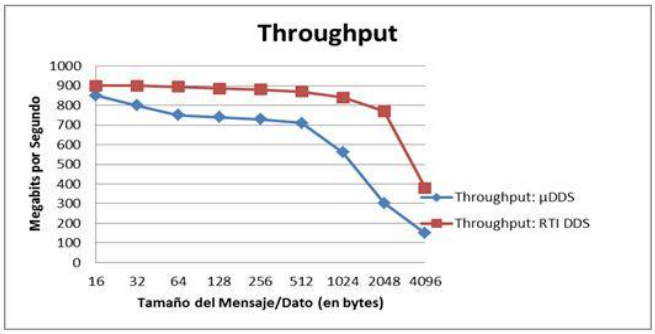


Fig. 7: Prueba de caudal Eficaz de la implementación µDDS.

En la Fig 8. Podemos observar la comparación entre la implementación propia del µDDS y otro middleware que sigue la misma especificación DDS llamado RTI DDS, de igual forma en la gráfica podemos ver que la curva azul perteneciente a nuestra implementación logra el mismo número de mensaje que la curva roja pero en un menor tiempo lo que nos brinda un caudal más eficaz que otras implementaciones dando mayor solides a nuestra plataforma.

## VI. Conclusiones

Como resultado del presente desarrollo tenemos una plataforma sólida, robusta y segura para la implementación de aplicaciones domóticas. De tal manera que mediante redes de sensores inalámbricas podremos obtener datos de una casa habitación los cuales con el uso de un publicador mediante la comunicación inalámbrica implementada bajo el estándar IEEE 802.15.4 una vez recibidos por los diferentes dispositivos de la plataforma el middleware bajo el patrón de publisher/subscriber se encargara de identificar si este nodo está subscrito y le pertenece esta información de lo contrario simplemente la desechada. Una aplicación implementada sobre la plataforma interpretara los datos y a partir de su análisis podrá realizar una toma de decisiones las cuales comunicara a los actuadores los cuales tendrán la tarea de activar/desactivar dispositivos de salida. El hipervisor permite el cambio de contexto entre los sistemas operativos y las aplicaciones además de brindarles protección y administrar los recursos del hardware. El fin de realizar una evaluación de desempeño de XtratuM es con el fin de verificar si cuenta con la robustez necesaria para la plataforma ya que la mayoría de pruebas de este hipervisor están enfocados a otras áreas además comprobamos su capacidad de soportar la implementación del µDDS a la cual también le realizamos algunas pruebas de comunicación para demostrar su eficiente funcionamiento sobre la plataforma y ver reflejada la interoperabilidad que nos proporciona este protocolo que sigue la especificación DDS. Después del desarrollo y el análisis del funcionamiento de la plataforma propuesta podemos afirmar que la implementación de aplicaciones domóticas sobre una arquitectura que basa su funcionamiento en un procesamiento distribuido con capacidades inalámbricas, características de tiempo real, una interfaz de usuario sencilla e intuitiva, niveles de seguridad crítica y una amplia interoperabilidad es una excelente opción para el desarrollo de aplicaciones que serán de bajo consumo de energía y de bajo costo para el usuario, lo que promete un gran futuro y desarrollo para nuevas y mejores aplicaciones inteligentes dirigidas hacia la domótica.

## Referencias


[1] OMG, «Model Driven Architecture (MDA),» Document number ormsc/2001-07-01. Technical report, 2001.

[2] E. Schoitsch, «Embedded Systems - Introduction,» *European Research Consortium for Informatics and Mathematics*, nº 52, 2003.

[3] B. Georgio, «Research trends in real-time computing for embedded systems,» *ACM SIGBED Review*, vol. 3, nº 3, pp. 1 - 10, 2006.

[4] A. Burns y A. Wellings, Sistemas de Tiempo Real y Lenguajes de Programación, ADDISON-WESLEY, 2005.

[5] H. Craig, Embedded Linux: Hardware, Software, And Interfacing, ISBN 0672322269 ed., Addison-Wesley Professional, 2002.

[6] H. Christopher, Embedded Linux Primer: A Practical Real-World Approach, Prentice Hall; 1 edition, 2006.

[7] I. Akyildiz y W. Su, *Wireless sensor network: a survey computer networks.*

[8] H. O. Marcy, J. R. Agre, C. Chien, L. P. Clare, N. Romanov y A. Twarowski, «Wireless Sensor Networks for Area Monitoring and Integrated Vehicle,» *AIAA Space Technology Conference and Exposition*, 1999.

[9] F. Hamoui, C. Urtado, S. Vauttier y M. Huchard, «Specification of a Component-based Domotic System to Support,» *International Conference on Software Engineering and Knowledge Engineering*, vol. 21, 2009.

[10] J. C. Augusto y C. D. Nugent, «Smart Homes Can Be Smarter,» *Lecture Notes in Computer Science*, vol. 4008/2006, nº 10.1007/11788485_1, pp. 1-15, 2006.

[11] J. A. Infantes Díaz, «Descripción de X-10, Biblioteca de conexión de Arduinos con el protocolo X10,» Departamento Lenguajes y Ciencias de la Computación, Universidad de Málaga, 2009.

[12] R. Acuña Agost, J. Ahumada Ojeda y M. Avendaño Arriagada, «Protocolo X10,» Universidad de Concepción - Chile, Mayo 2001. [En línea]. Available: http://www2.udec.cl/~racuna/domotica/x10.htm.

[13] N. Sastry y D. Wagner, «Security considerations for IEEE 802.15.4 networks,» *WiSe '04: Proceedings of the 2004 ACM workshop on Wireless security*, vol. 3, pp. 32-42, 2004.

[14] P. Barham, B. Dragovic, K. Fraser, S. Hand, T. Harris, A. Ho, R. Neugebauer, I. Pratt y A. Warfield, «Xen and the art of virtualization,» *ACM SIGOPS Operating Systems Review*, vol. 37, nº 5, Diciembre 2003.

[15] P. A. Karger, «Multi-Level Security Requirements for Hypervisors,» *Proceedings of the 21st Annual Computer Security Applications Conference*, pp. 267 - 275, 2005.

[16] S. Soltesz, H. Pötzl, M. E. Fiuczynski, A. Bavier y L. Peterson, «Container-based operating system virtualization: a scalable, high-performance alternative to hypervisors,» *EuroSys European Conference on Computer Systems*, vol. 41, nº 3, pp. 275-287, 2007.

[17] T. Guesmi, R. Rekik, S. Hasnaoui y H. Rezig, «Design and Performance of DDS-based Middleware for Real-Time Control Systems,» *International Journal of Computer Science and Network Security*, vol. 7, nº 12, 2007.

[18] E. Burnette, Hello, Android: Introducing Google's Mobile Development Platform, Pragmatic Bookshelf; 2nd edition, 2009.

[19] A. Crespo, I. Ripoll, M. Masmano, P. Arberet y J. Metge, «XtratuM an Open Source Hypervisor for TSP Embedded Systems in Aerospace,» *DAta Systems In Aerospace (DASIA)*, 2009, Istanbul, Turkey.

[20] OMG, «Data Distribution Service for Real-Time Systems version 1.2,» OMG Technical Document, Jan. 2007.

[21] OMG, «Overview and guide to OMG architecture,» OMG Technical Document formal/03-06-01, 2000.