Entorno de integración continua para validación de sistemas embebidos de tiempo real

Waldo Valiente, Esteban Carnuccio, Mariano Volker, Graciela De Luca, Raúl Villca, Matías Adagio
Departamento de Ingeniería e Investigaciones Tecnológicas
Universidad Nacional de La Matanza
Dirección: Florencio Varela 1703 – CP 1754 – {wvaliente, ecarnuccio, mvolker, gdeluca}@unlam.edu.ar, {raul.villcasd, mati.adagio}@gmail.com

RESUMEN

Cada año se desarrollan billones de sistemas embebidos. Por lo que el proceso de selección del dispositivo a utilizar se vuelve una tarea compleja. En parte por la cantidad de fabricantes y características en sus modelos, sumado a la necesidad del cumplimiento de requisitos no funcionales, tales como la limitación en el consumo energético y la criticidad de los tiempos de respuesta a eventos externos. Este trabajo se centrará en que esta tarea sea factible en tiempos razonables desde el punto de vista del negocio, como así también del proceso de selección/elaboración de un sistema embebido de tiempo real para lograr, incluso, su óptima replicación y fabricación. Por ese motivo se pretende desarrollar un entorno de mejora continua que facilite el desarrollo, pruebas y optimización de los sistemas embebidos. La mejora continua permite generar ciclos de medición y retroalimentación de las tres partes que forman el proceso de desarrollo. Además se pretende que el conjunto pueda ejecutar sobre distintos dispositivos, tanto real como virtual. Permitiendo así la correcta selección del dispositivo, incluso antes de ser adquirido.

Palabras clave: *Sistemas Embebidos, Tiempo Real, Optimización, Virtual, Mejora Continua.*

CONTEXTO

Nuestra Línea de Investigación es parte del proyecto "*Entorno de integración continua para validación de sistemas embebidos de tiempo real"*, dependiente de la Unidad Académica del *Departamento de Ingeniería e Investigaciones Tecnológicas*, perteneciente al programa de Investigaciones CyTMA2 de la Universidad Nacional de La Matanza, el cual es formado por docentes e investigadores de la carrera de ingeniería en informática. Este proyecto es continuación de los trabajos que viene realizando el grupo de investigación en sistemas operativos, computación de alto rendimiento y en el área de Internet de las cosas.

1. INTRODUCCIÓN

En sus inicio los Sistemas Embebidos (SE), se requerían para funcionalidades sencillas, con el fin de obtener programas de reducido tamaño, debido a las limitaciones en la memoria que disponía el hardware en esa época. Con el paso del tiempo y gracias a los avances tecnológicos, los microcontroladores incrementaron la cantidad de memoria. Por ende se volvieron más sofisticados, por lo que requirió mayor complejidad en la programación, hasta el punto de agregar funcionalidades complejas que pertenecen al ámbito de los Sistemas Operativos de Tiempo Real (RTOS). Con ello aparecieron múltiples modelos de dispositivos,

confeccionados por diferentes fabricantes, cada uno con su propia configuración y conjunto de herramientas de uso. Con el auge de Internet de las cosas apareció un creciente desarrollo de los SE, por año se crean 6 billones de SE nuevos [1], mientras que computadoras de escritorio, solo rondan los 100 millones [2]. Existen proyectos científicos que acompañan esta evolución. Por un lado la *National Science Foundation*, el año pasado finalizó un proyecto de optimización semántica para RTOS que utilizan SE [3]. Mientras que la Comunidad Europea, planteó un proyecto con el objetivo de bajar el consumo energético para RTOS con altas prestaciones que ejecutan en SE [4]. Así mismo desarrollar software para Sistemas Embebidos de Tiempo Real (RTSE) es una tarea compleja. Está afirmación es justificada por la experiencia adquirida de parte del equipo de investigación en los proyectos "Sistema de monitoreo y alarma para personas adultas mayores" y "Dispositivo de asistencia de personas mediante monitoreo y análisis de datos en la nube".

Problemática a investigar:

Los dispositivos utilizados en SE poseen múltiples diferencias, ya que cada uno cuenta con sus herramientas propias por lo que trabajar con ellos requiere de reunir constantemente habilidades y conocimientos técnicos para desenvolverse en dicha área. También existen dos complicaciones en la selección y el uso de cada microcontrolador, ya que cada uno posee una tecnología propia. La primera de ellas surge debido a la correcta elección del dispositivo, más allá de las especificaciones básicas, junto con el costo y la facilidad de adquisición en el mercado local. Cuando se plantean requisitos no funcionales estrictos, como el tiempo de respuesta ante distintos eventos, o que el consumo energético permita que el dispositivo funcione con batería asegurando la disponibilidad, existe una amplia variedad de modelos, que pueden estar sobre dimensionados o no llegar a satisfacer estos requerimientos. No es económicamente viable adquirir tantos modelos de dispositivos para elegir el que mejor se adapte a los requisitos. Por lo que se propone de utilizar un SE virtual para realizar las pruebas de rendimiento, verificando distintas optimizaciones, antes de llegar a adquirir el hardware del dispositivo. La segunda complicación, ya con el dispositivo seleccionado, surge por el amplio conjunto de herramientas suministrado por los fabricantes de SE. Estas se enfocan en el modelado, desarrollo, depuración e implementación del programa en el SE. Por lo que se busca armar un entorno de trabajo que permita realizar la ejecución de SE virtuales y reales, y medir las características de optimización que aseguren los requisitos no funcionales, en el uso de los recursos del SE.

Por otra parte se encontraron problemas comunes inherentes a la funcionalidad de los RTSE, estos se describen a continuación.

Problemas Comunes

Según [5], "En sistemas embebidos el objetivo principal de la programación del proceso es garantizar una respuesta rápida a eventos externos y cumplir el tiempo de ejecución del mismo". Para lo cual las funciones de código específicas pueden dividirse en tareas. Cada una puede tener una prioridad, esta dependerá de su periodicidad o si son activados como respuesta a eventos externos. En los SE suele haber procesadores de un solo núcleo, por eso lo que se suele multiplexar uso del procesador. Esto se efectúa para alternar el uso del procesador entre tareas, generando concurrencia. Para los SE complejos pueden tener procesadores de varios núcleos, logrando verdadera paralelización de tareas o realizando distintas tareas concurrentemente, lo que puede ocasionar retardos no buscados por problemas de

sincronización debido al uso de semáforos, memoria compartida, tuberías, entre otros.

Los sistemas operativos que al principio se ejecutaban en un SE, eran programas con funciones muy simples de control, que esperaban eventos desde el exterior para funcionar. Actualmente como la demanda de los sistemas multifuncionales aumenta, también lo hace la infraestructura donde se ejecutan. Por lo que se dispone de sistemas operativos cada vez más complejos, como los Smartphone que tienen casi tantas características computacionales como los de una computadora de escritorio. Algunos poseen memoria en el orden de los Gigabytes, procesadores con varios núcleos y pantallas de alta definición. Por lo que ya no se suele realizar diseños propios del sistema operativo para cada SE. Lo que se realiza es adaptar el sistema operativo tradicional, mediante configuraciones a estas arquitecturas, por ejemplo Android baso su versión de GNU/Linux o Microsoft Windows® para Microsoft Surface Go®.

En el último tiempo hubo un incremento exponencial de Sistemas Embebidos de Tiempo Real (RTOS) [6]. El requisito de Tiempo Real es necesario para completar su trabajo y entregar servicios regularmente, ya que tienen requisitos de tiempo estricto que deben cumplirse. Todos los días estos sistemas proporcionan tareas importantes, por ejemplo al conducir, al volar, monitoreo de pacientes, etc. La diferencia de estos sistemas con el de PC tradicional, es que los últimos ejecutan aplicaciones complejas que no requieren de tiempo real, como navegadores, procesadores de texto, entre otros. La vertiente actual busca utilizar los recursos de SE junto con funciones específicas de concurrencia que poseen los Sistemas Operativos tradicionales. RTOS es la opción que logra combinar ambos mundos, ya que están preparados para ejecutar tareas que poseen requerimiento de tiempos estrictos. Un RTOS debe responder inmediatamente a eventos externos con la menor latencia de interrupción. Además, debe completar los pedidos de servicio antes que se cumpla el tiempo límite. Para lograr cumplir con estos requisitos tienen las siguientes capacidades [5]: Menor latencia de interrupción, regiones críticas cortas, administrador de tareas apropiativo y algoritmos de planificación dinámicos. Los dos primeros requisitos tienen que ver con acortar los tiempos de ejecución de RTOS y que las tareas tengan mayor tiempo de procesador. Los otros dos requisitos tienen que ver como son administrados los tiempos de procesador, permitiendo a tareas de mayor prioridad cumplir su tiempo límite. Algunos de los RTOS que cumplen con estos requisitos son FreeRTOS, MicroC/OS, NuttX y QNX.

2. LINEAS DE INVESTIGACIÓN Y DESARROLLO

Por esta creciente necesidad de programas más robustos se tuvo que establecer un proceso de desarrollo para SE, este se compone de tres etapas [7], representadas en (Fig. 1):

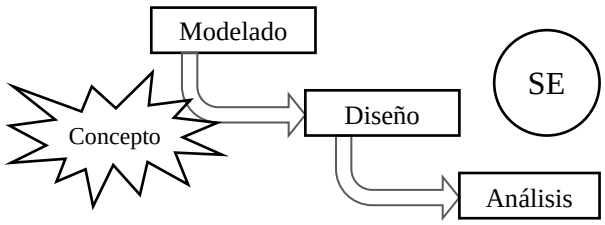

Fig. 1 Etapas del proceso de construcción de SE.

La concepción del SE parte desde una idea o concepto que se debe plasmar en el hardware.

1°) La primera etapa, la del Modelado, se centra en la definición de la dinámica del comportamiento de las partes que formarán el SE, como los eventos físicos serán medidos y convertidos en pulsos eléctricos por los sensores. El paradigma que representa mejor esta lógica es a través de máquinas de estados concurrentes, donde los sensores van

registrando eventos que accionan funcionalidades del SE para dar una salida por sus actuadores. Las herramientas que suelen utilizarse para el modelado son *Simulink*® y *LabVIEW*®. Estas son herramientas que permiten realizar pruebas, mediciones, controles y depuración en un entono simulado o real y embebido. Posibilita realizar todas las verificaciones antes de construir el hardware. Este programa puede conectarse al software de diseño electrónico llamado Proteus, a través de la interfaz emulador virtual serial [8], permitiendo así la simulación del microcontrolador. Cabe aclarar que estos programas son de licenciamiento pago.

2°) La etapa correspondiente al Diseño se centra en cómo se realiza la integración de las partes del SE y como obtener datos precisos de los sensores, ya sea por calibración o filtrado. Así como la funcionalidad que debe realizar el microcontrolador con esa información (sobre cómo el sistema interactúa con su entorno). En esta etapa se plantean las técnicas que debe emplearse para cumplir con requisitos específicos sobre el SE. Estas técnicas pueden requerir atender a varios sensores simultáneamente, como así también enfocar el diseño para lograr un menor consumo energético, para que funcione con baterías. Otro requisito, que también se diseña, tiene que ver con la forma de realizar cálculos complejos en el propio SE. Las herramientas que se utilizan en esta etapa son provistas por el fabricante de SE, ya que permite programarlo, muchas de ellas se basan en la interfaz de programación Eclipse. Para arquitecturas Arduino: Eclipse C++, para procesadores ARM: posee un conjunto de herramientas con licenciamiento pago *KIEL*® y la versión libre *System Workbench*, entre otros.

3°) La etapa de Análisis. Es utilizada para garantizar que el SE resultante, cumpla con la funcionalidad correcta, como con los requisitos planteados en las etapas de Modelado y Diseño. Debe incluir especificaciones, tales como seguridad, consumo, fiabilidad, entre otras, que deben ser definidas como propiedades en forma precisas, expresadas técnicamente, para evitar ambigüedad, facilitando la recolección y comparación de resultados. El análisis debe ser realizado cuantitativamente para lograr validar las definiciones de características o alcance del SE. En consecuencia existen técnicas para analizar tiempos dinámicamente [6], tales como: **Circuito Emulador:** Es un dispositivo de propósito especial que se comporta como un procesador particular, pero con mejores capacidades para depuración e inspección del programa que ejecuta. **Osciloscopios:** Son herramientas destinadas a medir con exactitud atributos de las señales eléctricas que recorren circuitos, por lo que pueden ser utilizados para medir tiempos de respuesta de las entradas y salidas de SE. **Analizadores lógicos:** Son dispositivos que permiten leer la memoria o el bus de direccionamiento y analizar las instrucciones que son leídas. **Dispositivo de rastreo:** Los puertos de rastreo son incorporados a los microcontroladores actuales. Estos permiten depurar el programa en ejecución e inspeccionar sus datos. **Temporizadores en el código:** Se puede agregar dentro del programa, funciones específicas que miden el intervalo de tiempo transcurrido en ejecutar partes del código. **Contadores de procesador:** Procesadores como X86, PowerPC o MIPs poseen contadores dentro del procesador, que se incrementan a medida que ejecutan instrucciones. **Herramientas de inspección de desempeño:** Obtienen el rendimiento del hardware, a través de código agregado por el compilador. **Simuladores:** Programa que trata de reproducir

la conducta de un microcontrolador real [2], a los fines de validar la funcionalidad, medir tiempos y atributos de la ejecución. **Emulador software:** Programa que permite ejecutar el programa SE como si ejecutara sobre la arquitectura real. Son de fácil acceso en las primeras etapas del desarrollo, aun sin contar con el hardware real.

3. RESULTADOS OBTENIDOS/ESPERADOS

Con el propósito de armar el entorno de trabajo de SE sé planifica el proyecto dividiéndolo en dos etapas. En la primera etapa se realizarán análisis sobre sistemas embebidos disponibles en el mercado, además de los tipos de RTOS que se pueden ejecutar sobre los mismos. Para lograr comparar sus características, ventajas y desventajas, se desarrollarán algoritmos y casos de prueba, que se almacenarán en repositorios para su utilización por parte de la comunidad. Estos se comparará sobre SE reales y virtuales, que serán configurados en las distintas pruebas. Se realizarán las compilaciones en el sistema embebido y se ejecutarán automáticamente con alguna herramienta de optimización de desarrollo, que serán evaluadas. Para el segundo año del proyecto se concentrara en el desarrollo de las pruebas automatizadas con herramientas disponibles con licencia GPL o similar. Estas herramientas están destinadas a mejorar el desarrollo de sistemas embebidos de tiempo real, depurando o midiendo rendimiento. Luego se realizarán análisis y pruebas sobre las optimizaciones que se puedan utilizar. De esta manera con la integración de las herramientas y las pruebas automáticas se podrá verificar y comparar el impacto de las optimizaciones sobre distintos SE. Permitiendo la elección del SE que se adecue a las necesidades.

4. FORMACIÓN DE RECURSOS HUMANOS

La presente línea de investigación dentro del departamento de Ingeniería e Investigaciones Tecnológicas forma parte del trabajo que uno de los investigadores se encuentra realizando para su maestría. Completan el grupo de investigación dos de docentes de categoría V y dos ingenieros en formación de investigador.

5. BIBLIOGRAFÍA

[1] X. Fan, Real-Time Embedded Systems: Design Principles and Engineering Practices, 1st ed., Newton, MA: Newnes, 2015.

[2] M. Barr y A. Massa, Programming Embedded Systems: With C and GNU Development Tools, 2 ed., Mumbai, India: O'Reilly Media, Inc., 2006.

[3] F. Eric, "Semantics of Optimization for Real Time Intelligent Embedded Systems (SORTIES)," National Science Foundation, Alexandria, Virginia 22314, USA , 2018.

[4] M. Bertogna, «High-Performance Real-time Architectures for Low-Power Embedded Systems,» 31 12 2018. [En línea]. Available: https://cordis.europa.eu/project/rcn/199161/factsheet/es. [Último acceso: 09 2019].

[5] K. C. Wang, Embedded and Real-Time Operating Systems, 1st ed., vol. 130, Springer Publishing Company, Incorporated, 2017, pp. 24, 36.

[6] I. Lee, J. Y.-T. Leung y S. H. Son, Handbook of Real-Time and Embedded Systems, 1st ed., Chapman & Hall/CRC, 2007.

[7] E. A. Lee y S. A. Seshia, Introduction to Embedded Systems: A Cyber-Physical Systems Approach, 2nd ed., The MIT Press, 2016.

[8] R. Singh, A. Gehlot, B. Singh y S. Choidhury, Arduino-Based Embedded Systems: Interfacing, Simulation, and LabVIEW GUI, CRC Press, 2017.