# Enseñanza y Aprendizaje de Ingeniería de Computadores

**Revista de Experiencias Docentes en Ingeniería de Computadores**

**Número 5, Mayo 2015**

**Edita:** Departamento de Arquitectura y Tecnología de Computadores

**Colabora:** Vicerrectorado para la Garantía de la Calidad

**ENSEÑANZA Y APRENDIZAJE DE INGENIERÍA DE COMPUTADORES**
**Revista de Experiencias Docentes en Ingeniería de Computadores**

TEACHING AND LEARNING COMPUTER ENGINEERING
Journal of Educational Experiences on Computer Engineering

**Número 5, Año 2015**

**Comité Editorial:**

**ISSN:** 2173-8688 **Depósito Legal:** GR-899/2011
**Edita:** Departamento de Arquitectura y Tecnología de Computadores
**Imprime:** Copicentro Editorial



Printed in Spain Impresa en España

# Índice

DESDE EL PUPITRE
(Experiencias de estudiantes)

# Editorial

Presentamos el quinto número de la revista Enseñanza y Aprendizaje de Ingeniería de Computadores agradeciendo su contribución a los autores de los doce artículos que incluye. De los trabajos que se publican, cinco provienen de la Universidad de Granada (en tres casos se trata de contribuciones de estudiantes, fundamentalmente), dos de la Universidad Complutense de Madrid, y el resto de las Universidades Autónoma de Barcelona, Córdoba, Nacional de San Luis (Argentina), Politécnica de Valencia, y Sevilla. Por tanto, se mantiene la participación de autores de otras Universidades, que empezó a ser significativa a partir del último número, en 2014, coincidiendo con la ampliación del Comité Editorial en el que ya se han incluido profesores de 23 Universidades. Aprovechamos esta editorial para agradecer el trabajo de todos ellos, tanto difundiendo la revista, como participando en el proceso de revisión de los artículos.

En este curso 2014/2015 se han empezado a impartir los denominados "Máster profesionalizantes" de Ingeniería Informática e Ingeniería de Telecomunicación en la ETSIIT de Granada. Además, el Máster de Ingeniería de Computadores y Redes que impartía el Departamento de Arquitectura y Tecnología de Computadores ha sido sustituido por el Máster en Ciencia de Datos e Ingeniería de Computadores, en el que se mantiene una especialidad en Ingeniería de Computadores y Redes. Precisamente en el presente número de la revista se incluye un artículo dedicado a este nuevo Máster ("Máster en Ciencia de Datos e Ingeniería de Computadores: una apuesta por la formación especializada en el sector de las TIC" de F. Rojas, A. Cano, M. Gómez, J. Ortega, F. Herrera, R. Romero, J. González), junto con otro que describe una de las asignaturas de la especialidad de Ingeniería de Computadores y Redes ("Ingeniería de Computadores en la era del Big Data: Computación de altas prestaciones en clasificación y optimización" de J. Ortega, P. Martín, J. González, M. Damas).

Los tópicos relacionados con la seguridad informática son esenciales, y su relevancia crece día a día, como se pone de manifiesto a través de las asignaturas que sobre estos temas se cursan en diferentes Planes de estudios. En este número se incluyen dos trabajos relacionados con la seguridad: "Sistema de Seguridad Basado en una Plataforma Heterogénea Distribuida" de D. Lora, P. Cerro, A.A. Del Barrio, y G.Botella; y "Servicios y Seguridad, un enfoque basado en estrategias de ataque y defensa" de M. Castro, F.Boixader, M. Taboada, D.Rexachs, y E. Luque.

La importancia de utilizar referencias a la historia de la evolución de los computadores, tanto en los aspectos relacionados con su hardware como con su software, para motivar a los estudiantes, y su integración en la docencia de las asignaturas de Ingeniería Informática se ha abordado en artículos de números anteriores de la revista. En este número se incluye también una contribución en esta línea, "Retroinformática para la enseñanza y el aprendizaje en la Ingeniería Informática" de X. Molero, A. Pont, A. Robles, y M. Martínez. También en la línea de buscar propuestas orientadas a mejorar la motivación de los estudiantes para el aprendizaje de una asignatura específica se incluye un artículo relacionado con la utilización de entornos virtuales: "Una experiencia para fortalecer los procesos de enseñanza de la programación mediante el uso de entornos virtuales de aprendizaje" de J.F. Aguirre, H.J. Viano, y B. García.

Sobre temas más específicamente relacionados con las asignaturas de Ingeniería de Computadores, este número de 2015 incluye tres artículos. Dos de ellos describen simuladores. Así, en "Diseño e implementación de un simulador software basado en el procesador MIPS32", de M. Rivas, M. Domínguez, F. Gómez, A. Linares, G. Jiménez, y A.Civit, se proporciona una herramienta que permite elaborar prácticas relacionadas con el lenguaje ensamblador y la comprensión de la microarquitectura de un procesador, mientras que "Simuladores de Planificadores de Sistemas en Tiempo Real" de F.J. Aliaga, I.M. Aliaga, J. Olivares, J.C. Gámez, y J.M. Palomares, considera los aspectos relacionados con las herramientas de simulación para elaborar prácticas de planificación cuando hay restricciones de tiempo real. El otro artículo al que nos hemos referido dentro de los que abordaban temas específicos de la Ingeniería de Computadores es "Una Orquesta Sinfónica como Ejemplo de Aplicación de un Sistema Empotrado Distribuido", de F. Parrales, A.A. Del Barrio, y G.Botella, donde se ilustra una aplicación para los sistemas empotrados distribuidos que, sin duda, resulta muy oportuna para aumentar la motivación de los estudiantes hacia las asignaturas que abordan los tópicos relacionados con el diseño de sistemas de propósito específico.

Para completar este número de la revista, tenemos la sección que iniciamos en el número de 2014 con artículos elaborados por estudiantes (con o sin la participación de profesores) y que denominamos "Desde el pupitre". Para una revista que incluye entre sus objetivos la difusión de los contenidos y las utilidades de la Ingeniería de Computadores, es importante contar con la realimentación que nos proporcionan los estudiantes en sus contribuciones, y poner de manifiesto los proyectos que pueden ser capaces de llevar a cabo con éxito. En esta ocasión, la sección incluye tres contribuciones. En "Aplicación de las materias de Ingeniería de Computadores en la mejora de los Algoritmos Meméticos y Metaheurísticas en general" de F. Palacios y J.Chamorro, se pone de manifiesto la utilidad del conocimiento de detalles relacionados con las plataformas de cómputo en el desarrollo e implementación de algoritmos (en este caso algoritmos meméticos y metaheurísticas). La relevancia del estudio de arquitecturas con paralelismo de datos como las GPU y el interés que despierta en los estudiantes se pone de manifiesto en el artículo "Convolución paralela con GPU", de F.J. Mesa y M.G. Arenas. El tercer artículo de esta sección, "Proyecto Prácticas Procesadores Integrados: Self-Balancing Robot basado en Arduino" de C. Bailón y A.F.

Díaz, describe los aspectos a considerar en el diseño hardware y software dentro de un proyecto de implementación de un robot basado en Arduino, una plataforma de hardware libre muy popular actualmente y con muchas posibilidades para el diseño de prácticas de Ingeniería de Computadores.

Como en los cuatro números anteriores de la revista, algunos de los contenidos de este número están relacionados en gran medida con las Jornadas de Coordinación Docente y Empresas, JCDE (http://atccongresos.ugr.es/jcde/), que en su quinta edición, se celebraron en Diciembre de 2014. Nuestro agradecimiento a todos los que han contribuido a la celebración de esas V JDCE. En primer lugar, agradecemos a Joaquín Fernández Valdivia, director de la E.T.S.I.I.T. por su participación en el acto de inauguración de la JDCE y al equipo de dirección de la Escuela por facilitarnos el uso de las instalaciones; a los ponentes de las distintas sesiones (Juan Julián Merelo, Javier Díaz, Antonio Cañas, y Samuel Romero, profesores del Departamento de Arquitectura y Tecnología de Computadores, y Jesús Chamorro, Director de la OTRI de la Universidad de Granada) por sus presentaciones y su participación en las mesas redondas correspondientes. También agradecemos al Departamento de Arquitectura y Tecnología de Computadores, y especialmente al Vicerrectorado de Garantía de la Calidad de la Universidad de Granada el apoyo y la financiación recibida..

El Comité Editorial

# Máster en Ciencia de Datos e Ingeniería de Computadores: una apuesta por la formación especializada en el sector de las TIC

Fernando Rojas[1], Andrés Cano[2], Manuel Gómez[2], Julio Ortega[1], Francisco Herrera[2], Rocío Romero-Zaliz[2], Jesús González[1]

[1]Departamento de Arquitectura y Tecnología de Computadores. ETSI Informática y de Telecomunicación. Universidad de Granada.
{frojas,jortega,jesusgonzalez}@ugr.es

[2]Departamento de Ciencias de la Computación e Inteligencia Artificial. ETSI Informática y de Telecomunicación. Universidad de Granada.
{acu, mgomez, herrera, rocio}@decsai.ugr.es

**Resumen.** En el curso 2014-2015 se ha impartido por vez primera el "Máster en Ciencia de Datos e Ingeniería de Computadores" en la Universidad de Granada. Este máster surge de la unión de los anteriores "Máster en Soft Computing y Sistemas Inteligentes" y "Máster en Ingeniería de Computadores y Redes", renovando completamente sus contenidos y ofreciendo una oportunidad de asociación entre varios grupos de investigación con un enorme potencial en el ámbito de las Tecnologías de la Información y de las Comunicaciones (TIC). Este máster proporciona al estudiante la oportunidad de formarse profesionalmente en campos con una gran demanda de personal cualificado y de enorme interés social, tales como el análisis de grandes cantidades de datos (*big data*) o el desarrollo de sistemas de propósito específico y plataformas de computación de altas prestaciones.

**Palabras Clave:** Ciencia de datos, ingeniería de computadores, Espacio Europeo de Educación Superior (EEES), máster, big data, computación de altas prestaciones, sistemas específicos.

**Abstract.** The Master's Degree in Data Science and Computer Engineering has been taught at the University of Granada for the first time during academic year 2014-2015. This master comes from the combination of former masters in Soft Computing and Computer Engineering, but it completely renovates its contents and it provides an opportunity for partnership and collaboration between several research groups with a great potential in the field of Information Technology and Communications (ICT). The proposed Master's Degree trains students as researchers in areas with a high demand for qualified staff and huge social interest, such as the analysis of large amounts of data (big data) or the development of embedded systems and platforms for high-performance computing.

**Keywords:** Data science, computer engineering, European Higher Education Area (EHEA), Master's Degree, embedded systems, high-performance computing.

# 1 Introducción

## 1.1 Motivación

Los estudios de un máster tienen como principal finalidad que el estudiante adquiera una formación avanzada, ya sea de carácter especializado o multidisciplinar. Éste es el escenario en que se propone el Máster Universitario Oficial en Ciencia de Datos e Ingeniería de Computadores por la Universidad de Granada, siguiendo la disposición de los Reales Decretos 1393/2007 y 861/2010 por los cuales se establece que los estudios de máster universitario de investigación constituyen el período formativo previo al de doctorado.

El Máster Universitario Oficial en Ciencia de Datos e Ingeniería de Computadores se presenta como una oportunidad de sinergia entre varios grupos de investigación en temas de gran demanda profesional y científica, que sin ninguna duda, ya forma parte del perfil profesional demandado en la actualidad en el ámbito de las Tecnologías de la Información y de las Comunicaciones. Por una parte, la necesidad de analizar grandes volúmenes de datos hacen de la Ciencia de Datos una de las disciplinas que mayor demanda de empleo generará en el futuro inmediato en todo el mundo. Por otra parte, el ingente volumen de datos a tratar precisa de plataformas de cómputo de altas prestaciones y sistemas específicos en el ámbito de la Ingeniería de Computadores.

Este máster proporciona al estudiante la oportunidad de formarse profesionalmente en estas líneas a través de un personal docente integrado en diferentes grupos de investigación de reconocido prestigio internacional. Se presenta como un título único que integra los requerimientos de cómputo de altas prestaciones y de sistemas de uso específico ante la necesidad de obtener información útil a partir de grandes cantidades de datos. Se pueden encontrar todos los detalles sobre este título en la página web del máster [1] y, de forma resumida, en el tríptico del máster (se reproduce la portada en la Figura 1), disponible en formato digital en la misma dirección URL.

## 1.2 Antecedentes

El Máster Universitario Oficial en Ciencia de Datos e Ingeniería de Computadores se propone como un nuevo máster que ha surgido de la unión de los anteriores “Máster en Soft Computing y Sistemas Inteligentes” y “Máster en Ingeniería de Computadores y Redes”.

El Máster en Soft Computing y Sistemas Inteligentes surgió como la evolución natural del Programa de Doctorado “Diseño, Análisis y Aplicaciones de Sistemas Inteligentes” al Espacio Europeo de Educación Superior (EEES). Dicho programa obtuvo la Mención de Calidad del Ministerio de Educación y Ciencia desde el año 2003 (Mención de Calidad MCD2003-00509) hasta el 2006, año en el que se sustituye por el Posgrado en Ciencias de la Computación y Tecnología Informática, obteniendo la mención de calidad durante su impartición (MCD2007-00212).

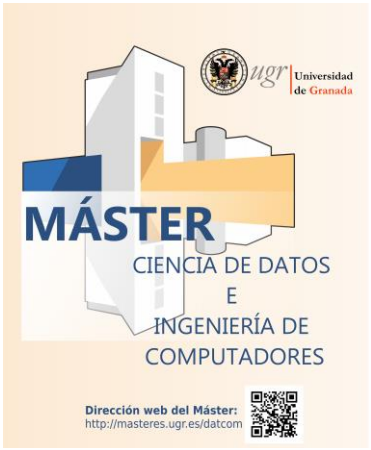


**Figura 1.** Portada del tríptico informativo del Máster en Ciencia de Computadores e Ingeniería de Computadores (disponible en http://masteres.ugr.es/datcom/).

El Máster Universitario en Ingeniería de Computadores y Redes surgió igualmente a partir del proceso de adaptación al EEES (R.D. 56/2005) del Programa de Doctorado "Ingeniería de Computadores: Perspectivas y Aplicaciones". Este Programa de Doctorado se impartió ininterrumpidamente desde el curso 1999-2000 por el departamento de Arquitectura y Tecnología de Computadores de la Universidad de Granada, integrado por profesores de las áreas de Arquitectura y Tecnología de Computadores e Ingeniería de Sistemas y Automática, todos ellos miembros del grupo de investigación CASIP (Circuitos y Sistemas para el Procesamiento de la Información), reconocido como grupo de excelencia de la Junta de Andalucía con la referencia TIC-117. El programa obtuvo la Mención de Calidad del Ministerio (MCD2004-00438, BOE 5/7/2004). Desde ese momento, el programa ha concurrido a todas las convocatorias que se han sucedido para la renovación y seguimiento, habiéndose mantenido la Mención de Calidad desde 2004. Actualmente, estos dos programas de doctorado se encuentran en proceso de extinción, siendo sustituidos por el Programa Oficial de Doctorado en Tecnologías de la Información y la Comunicación de la Universidad de Granada [2].

### 1.3 Proceso de verificación

La solicitud de verificación del actual Máster en Ciencia de Datos e Ingeniería de Computadores se realizó a través de la Universidad de Granada, aprobada por Consejo de Gobierno en su reunión de 8 de abril de 2014. En [3] se puede consultar el documento de solicitud de verificación completo. En respuesta, se recibió el informe favorable por parte de la Dirección de Evaluación y Acreditación de la Agencia Andaluza del Conocimiento (DEVA) para la verificación del título oficial en su reunión celebrada en Córdoba el 30 de junio de 2014. La Consejería de Economía, Innovación, Ciencia y Empleo de la Junta de Andalucía a través del Decreto 113/2014, de 15 de julio, por el que se autoriza la implantación de enseñanzas universitarias de Grado, Máster y Doctorado de las Universidades públicas andaluzas

para el curso 2014/2015 recoge, entre los títulos de máster que se van a impartir, al Máster en Ciencia de Datos e Ingeniería de Computadores [4]. La resolución de 2 de octubre de 2014 de la Secretaría General de Universidades publica el Acuerdo del Consejo de Ministros de 26 de septiembre de 2014 por el que se establece el carácter oficial de determinados títulos de Máster (entre los cuales figura el que aquí se describe) y su inscripción en el Registro de Universidades, Centros y Títulos [5]. Finalmente, la resolución de 25 de noviembre de 2014 de la Universidad de Granada publicada en [6] publica el plan de estudios del Máster en Ciencia de Datos e Ingeniería de Computadores.

### 1.4 Diferenciación con títulos similares en la Universidad de Granada

La Universidad de Granada incluye entre su oferta actual varios títulos de Máster en el ámbito de las Tecnologías de la Información y Comunicación. En cualquier caso, el Máster en Ciencia de Datos e Ingeniería de Computadores se diferencia notablemente de todos ellos en cuanto a su justificación, competencias y plan de estudios.

En primer lugar, se ofrece un máster universitario que se imparte en la ETS de Ingeniería Informática y Telecomunicación denominado "Máster en Desarrollo de Software" [7]. Este máster se integra en el Programa Oficial de Doctorado en Tecnologías de la Información y la Comunicación. No obstante, sus contenidos son completamente diferentes a los propuestos en este máster, centrándose en tareas de desarrollo e ingeniería del software e investigación en la interacción persona-ordenador.

Adicionalmente se oferta el máster de orientación profesional "Máster en Gestión y Tecnologías de Procesos de Negocio" [8] que podría incluirse en el ámbito de las TIC, pero que se diferencia claramente del descrito aquí, debido a su orientación específica a la gestión empresarial y las tecnologías asociadas.

Finalmente, la Universidad de Granada ha puesto en marcha los Másteres Universitarios que habilitan para el ejercicio de la profesión de Ingeniero de Telecomunicación [9] y de Ingeniero en Informática [10]. Estos másteres proporcionan competencias profesionales reguladas y no tiene un perfil marcado en un área concreta de las tecnologías de la información y las comunicaciones.

## 2 Justificación e interés del máster para la sociedad

El Máster en Ciencia de Datos e Ingeniería de Computadores ofrece un valor diferenciador a través de las posibilidades de obtener una formación interdisciplinar y una óptima propuesta para la inserción laboral e investigadora, actualizada e incorporando líneas de investigación y transferencia muy significativas en el ámbito profesional de las Tecnologías de la Información y la Comunicación.

Todos los alumnos que cursen este máster obtienen un título oficial con reconocimiento en todo el Espacio Europeo de Educación Superior (EEES), y además les permite acceder a la realización de la tesis doctoral dentro del Programa de Doctorado Tecnologías de la Información y las Comunicaciones [2], siempre que se cumplan las condiciones que exige la normativa vigente.

La demanda laboral de especialistas en ciencia de datos e ingeniería de computadores hace de este perfil informático uno de los más solicitados por parte de todo tipo de empresas tanto a nivel nacional como internacional. El sector de las TIC es uno de los pocos que puede presumir de tener paro cero y las expectativas son aún mejores, asegurándose desde el Colegio de Ingenieros Informáticos de España que "*en los próximos cinco años el mercado demandará medio millón de puestos de trabajo relacionados con la economía digital*" [11]. Si se restringe el ámbito de las TIC a la ciencia de datos e ingeniería de computadores, las perspectivas son mucho mejores: "*El 'big data' arrasa en las escuelas de negocios*" [12], "*El científico de datos: una novedosa y necesaria profesión*" [13], etc.

En el ámbito local, en palabras del propio Director de la ETS de Ingenierías Informática y de Telecomunicación, Joaquín Fernández Valdivia, las TIC constituyen el nuevo motor económico de la ciudad de Granada, constituyendo lo que denomina una "*isla tecnológica*" en España en la que confluyen la excelencia en formación, investigación y transferencia para ofrecer un "*ecosistema único a través del cual es posible brindar verdadero apoyo a buenas ideas y prometedores proyectos de alto contenido tecnológico al disponer de una abundante oferta de profesionales universitarios de altísima preparación tecnológica unida a una proliferación de vocaciones empresariales*" [14].

La calidad de la formación que se va a proporcionar en este máster está avalada por la excelencia de la Universidad de Granada en el campo de la Informática (*Computer Science*). Existen dos rankings o clasificaciones de calidad de las universidades de reconocido prestigio internacional que se describen: el ranking de Shanghai (*Academic Ranking of World Universities*, ARWU) [15] y el ranking de Taiwan (*National Taiwan University Ranking, NTU Ranking*) [16]. En la edición de 2014 del ranking de Shanghai, la Universidad de Granada ocupa el puesto 43 del mundo en el campo de la Informática (*Computer Science*), siendo la primera universidad española y la séptima de Europa. En cuanto a la clasificación elaborada por la *National Taiwan University* a partir de los datos de publicaciones científicas de las bases de datos *Web of Science* de Thomson Reuters, la Universidad de Granada se sitúa también como la primera universidad española en Informática y en la posición 20ª del mundo (Figura 2) y 3ª de Europa. Las siguientes universidades españolas en esta clasificación son la Universidad Politécnica de Cataluña, en el puesto 46, y la Universidad Politécnica de Valencia en el 77.

# 3 Estructura del Máster

## 3.1 Especialidades que incluye el Máster

El máster propuesto incluye dos especialidades que se corresponden, en cierta medida, a los anteriores másteres de *Soft Computing* y Sistemas Inteligentes y el de Ingeniería de Computadores y Redes. El estudiante puede elegir una especialidad de entre las dos ofertadas, o bien realizar una configuración multidisciplinar en su perfil de optatividad. Las especialidades propuestas se describen a continuación.

| World Rank | University | Total Score details |
|---|---|---|
| 1 | Nanyang Technological University | 89.5 |
| 2 | Massachusetts Institute of Technology | 87.6 |
| 3 | University of California- Berkeley | 85.8 |
| 3 | Stanford University | 85.8 |
| 5 | Tsinghua University | 81.2 |
| 6 | National University of Singapore | 79.0 |
| 7 | Harvard University | 78.8 |
| 8 | University of California- San Diego | 78.1 |
| 9 | City University of Hong Kong | 77.6 |
| 10 | University of Illinois- Urbana-Champaign | 77.2 |
| 11 | Carnegie Mellon University | 76.7 |
| 12 | Swiss Federal Institute of Technology - Zurich | 74.5 |
| 13 | Zhejiang University | 74.4 |
| 14 | Harbin Institute of Technology | 73.2 |
| 15 | Hong Kong Polytechnic University | 73.0 |
| 16 | Georgia Institute of Technology | 72.2 |
| 17 | Shanghai Jiao Tong University | 72.1 |
| 18 | University of Michigan- Ann Arbor | 70.1 |
| 19 | Swiss Federal Institute of Technology - Lausanne | 69.8 |
| 20 | University of Granada | 69.7 |
| 21 | The University of Texas- Austin | 69.6 |
| 22 | University of Southern California | 69.5 |
| 23 | University of Maryland- College Park | 68.0 |
| 23 | University of Oxford | 68.0 |
| 25 | Huazhong University of Science & Technology | 67.4 |

**Figura 2**. 'Top 25' de Universidades a nivel mundial en el campo de la Informática (*Computer Science*), de acuerdo con el 'Ranking de Taiwán 2014'. (http://nturanking.lis.ntu.edu.tw/DataPage/TOP300.aspx?query=ComputerScience&y=2014)

– Especialidad en Ingeniería de Computadores y Redes. Las perspectivas en la ingeniería de computadores y las redes están determinadas tanto por las posibilidades que ofrecen las mejoras tecnológicas como por los condicionantes que establecen las aplicaciones dominantes en el mercado: se trata de disponer de computadores que aprovechen la tecnología de forma óptima, ejecutando eficientemente las aplicaciones (fundamentalmente las más demandadas), y ajustándose a las restricciones que impone el mercado. El hecho de disponer de un cada vez mayor número de datos y la necesidad de obtener información útil a partir de dichos datos en un tiempo razonable implica la demanda imprescindible de plataformas de cómputo de altas prestaciones (HPC, High performance Computing) [17] y sistemas de propósito específico que se adapten al entorno de uso tanto a nivel de hardware como de software [18, 19].

- Especialidad en Ciencia de Datos y Tecnologías Inteligentes. Big data y la ciencia de datos utilizan algoritmos y tecnología de estadística de datos para procesar grandes cantidades de datos con el objetivo de extraer conocimiento [20-24]. Se aplican a áreas tan diversas como la genómica, física, energía, transportes, finanzas, marketing, medicina, educación, etc. Todas las previsiones indican que su aplicación crecerá de forma notable en los próximos años, ya que el proceso de recogida de datos mediante ordenadores, dispositivos inteligentes, sensores y web, entre otros, aumenta de forma notable. Ante esta avalancha de datos, hay una percepción creciente de la necesidad de analizarlos, ordenarlos y visualizarlos en busca de relaciones, patrones y agrupaciones que no pueden obtenerse mediante su simple observación. En definitiva, todo apunta a que habrá una gran demanda de profesionales dedicados a esta tarea en los próximos años.

Las dos especialidades propuestas se complementan mutuamente, ya que el tratamiento de grandes masas de datos sólo es posible con una adecuada infraestructura de cómputo. Esto hace que los equipos profesionales de trabajo en ciencia de datos e ingeniería de computadores tengan que conformarse con personal cualificado en ambos perfiles de conocimiento. De esta forma, se justifica la propuesta de ambas especialidades de forma conjunta y coordinada en este máster.

### 3.2 Propuesta formativa

Para obtener el título de Máster se han de superar 60 créditos ECTS de entre un total de 128 ECTS ofertados. La estructura del máster es la siguiente (Figura 3):

- Asignaturas obligatorias: 12 créditos repartidos en 3 asignaturas de 4 ECTS que incluyen materias comunes relacionadas con la metodología de la investigación en TIC, técnicas de investigación, análisis de datos, oportunidades de innovación, etc.
- Asignaturas de nivelación de conocimientos: se ofertan 16 créditos de los que el estudiante cursará 2 asignaturas (8 ECTS) para complementar su formación y atender a las diferentes necesidades para el acceso al bloque de especialización con suficientes garantías.
- Asignaturas optativas: el alumno debe cursar 28 créditos de optatividad. El estudiante podrá elegir una especialidad de entre las 2 planteadas o bien realizar una configuración multidisciplinar en su perfil de optatividad. Para obtener la distinción de una especialidad dentro del título de máster el estudiante deberá haber superado al menos 24 créditos dentro de la especialidad correspondiente. El resto de créditos (4) se pueden elegir libremente de entre todos los ECTS optativos. Se ofrecen 44 ECTS por especialidad, para un total de 88 ECTS ofertados.
- Trabajo Fin de Máster: el máster concluirá con la elaboración y defensa de un Trabajo de Fin de Máster de 12 ECTS que deberá estar orientado a la evaluación de competencias asociadas al título.

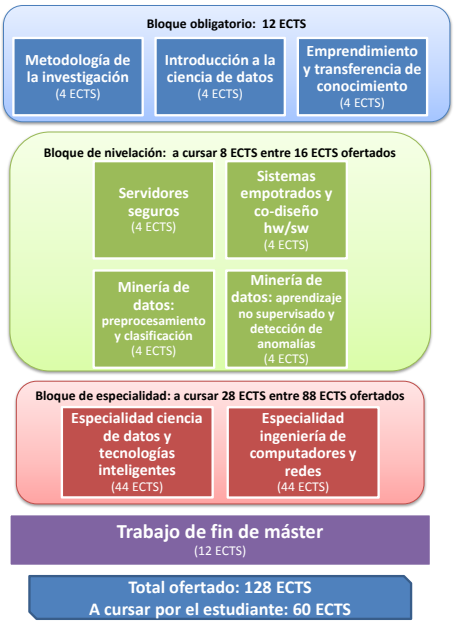


**Figura 3**. Resumen de la Propuesta formativa del Máster en Ciencia de Datos e Ingeniería de Computadores

### 3.3 Descripción detallada del plan de estudios

Las Tablas 1, 2 y 3 detallan los nombres y asignación de créditos ECTS de las asignaturas que componen cada uno de los bloques descritos en la Figura 3.

**Tabla 1**. Materias que componen los módulos comunes (obligatorio, nivelación y TFM)

| **Módulo** | **Materias** | **ECTS** |
|---|---|---|
| Módulo Obligatorio | Metodología de la investigación | 4 |
| | Introducción a la ciencia de datos | 4 |
| | Emprendimiento y transferencia de conocimiento | 4 |
| Módulo de Nivelación de Conocimientos | Servidores seguros | 4 |
| | Sistemas empotrados y co-diseño hw/sw | 4 |
| | Minería de datos: preprocesamiento y clasificación | 4 |
| | Minería de datos: aprendizaje no supervisado y detección de anomalías | 4 |
| Trabajo Fin de Máster | Trabajo fin de máster | 12 |

**Tabla 2.** Materias que componen la especialidad "Ingeniería de Computadores y Redes"

| Especialidad | Módulo | Materias | ECTS |
|---|---|---|---|
| Especialidad en Ingeniería de Computadores y Redes | Computación de Altas Prestaciones | Computación de altas prestaciones para clasificación y optimización | 4 |
| | | Biología computacional con big data-omics e ingeniería biomédica | 4 |
| | | Ingeniería de servidores web | 4 |
| | | Modelado de sistemas y predicción de series temporales | 4 |
| | | Procesamiento de la señal de altas prestaciones en biomedicina | 4 |
| | Sistemas de Aplicación Específica | Internet de las cosas | 4 |
| | | Arquitecturas de altas prestaciones para visión | 4 |
| | | Mecatrónica y sistemas aero-espaciales | 4 |
| | | Neurociencia computacional y neuroingeniería | 4 |
| | | Sistemas de visión bioinspirados | 4 |
| | | Robótica móvil y neurobótica | 4 |

**Tabla 3.** Materias que componen la especialidad "Ciencia de Datos y Tecnologías Inteligentes"

| Especialidad | Módulo | Materias | ECTS |
|---|---|---|---|
| Especialidad en Ciencia de Datos y Tecnologías Inteligentes | Modelos avanzados de ciencia de datos | Modelos gráficos probabilísticos | 4 |
| | | Extracción de características en imágenes | 3 |
| | | Series temporales y minería de flujos de datos | 3 |
| | | Sistemas de recuperación de información y de recomendación | 3 |
| | | Minería de datos: Aspectos avanzados | 3 |
| | | Modelos de ciencia de datos no numéricos. Aplicaciones en redes sociales, web y gestión de procesos | 6 |
| | Big data y cloud computing | Big data y cloud computing | 6 |
| | Tecnologías inteligentes e inteligencia computacional | Soft computing: Conjuntos y sistemas difusos | 4 |
| | | Técnicas de Soft Computing para Aprendizaje y optimización. Redes Neuronales y Metaheurísticas, programación evolutiva y bioinspirada | 3 |
| | | Visión por Computador | 3 |
| | Aplicaciones de Ciencias de Datos y Tecnologías Inteligentes | Aplicaciones de ciencia de datos y tecnologías inteligentes | 6 |

En la especialidad en Ciencia de Datos y Tecnologías Inteligentes es relevante la participación de los estudiantes en competiciones propias del máster y organizadas mediante la plataforma Kaggle [25]. La experiencia adquirida por los estudiantes ha permitido que, de forma particular, se hayan involucrado en competiciones profesionales encargadas por empresas y en las que se remunera a los ganadores. Los estudiantes del máster han trabajado con R [26], uno de los lenguajes más usados en ciencia de datos y con más de 6700 librerías disponibles en el repositorio CRAN (*Comprehensive R Archive Network*). También cabe destacar que en este perfil los estudiantes tienen acceso a un clúster propio (hadoop.ugr.es) para la realización de las prácticas de Cloud Computing y Big Data, procesando los datos bajo los paradigmas Hadoop y Spark.

Por su parte, en la especialidad de Ingeniería de Computadores y Redes se pueden destacar por una parte las asignaturas relacionadas con la computación de altas prestaciones (HPC), implementando aplicaciones reales que maximicen el rendimiento de los sistemas de cómputo y analizando las características de éstos. Por otro lado, desde el bloque de asignaturas relacionado con los sistemas de aplicación específica, se estudia y trabaja de forma directa con sistemas que precisan de ingeniería y desarrollo a medida, tanto a nivel de hardware como de software. Los estudiantes podrán programar e implementar sus propios robots, actuadores y sistemas de control, diseñar sistemas de visión artificial o controlar un vehículo aéreo no tripulado, entre muchas otras aplicaciones.

Los detalles sobre los docentes que imparten cada materia, sus contenidos, objetivos, metodología docente, procedimiento de evaluación e información ampliada se pueden encontrar en la guía docente de cada asignatura. Estas guías de pueden consultar a través de la página web del Máster en Ciencia de Datos e Ingeniería de Computadores, en la sección "Descripción detallada del programa de estudios": http://masteres.ugr.es/datcom/pages/info_academica/plan_estudios.

En cuanto al calendario de impartición de las lecciones presenciales, éste comprende desde mediados del mes de octubre hasta final de mayo. Se ha procurado conciliar el posible desempeño de una actividad laboral por parte de los estudiantes, concentrando la docencia en horario de tarde y preferentemente de lunes a jueves. Además se incide en el carácter aplicado del máster, reservando un 50% de las lecciones presenciales para aplicaciones prácticas. Referente a la evaluación, aunque cada materia puede establecer sus propios matices y criterios, se propone un modelo basado en la evaluación continua, según lo descrito en la Normativa de evaluación y de calificación de los estudiantes de la Universidad de Granada [27].

## 4 Conclusiones

La Universidad de Granada ofrece la posibilidad de realizar la formación de posgrado en una de las universidades con mayor prestigio a nivel mundial en el sector de las TIC. Además, el Máster en Ciencia de Datos e Ingeniería de Computadores que se describe en esta contribución supone una oferta única que integra dos especialidades de gran interés social y en I+D+i: la ciencia de datos y la ingeniería de computadores.

Los alumnos que superen este máster obtienen un título oficial con reconocimiento en todo el Espacio Europeo de Educación Superior (EEES).

Este máster se propone para su realización en un curso académico y combina formación rigurosa en técnicas avanzadas con un contacto muy cercano con aplicaciones reales y con una plantilla docente altamente cualificada. En definitiva, el Máster en Ciencia de Datos e Ingeniería de Computadores apuesta por la calidad y la excelencia en la formación de un perfil con una enorme proyección laboral e investigadora en la actualidad.

## Referencias


1. Universidad de Granada, Máster Universitario Oficial en Ciencia de Datos e Ingeniería de Computadores, http://masteres.ugr.es/datcom/
2. Universidad de Granada, Programa Oficial de Doctorado en Tecnologías de la Información y la Comunicación, http://doctorados.ugr.es/tic/
3. Consejo de Gobierno de la Universidad de Granada: Normativa de Órganos Colegiados: NCG80/2. Máster Universitario en Ciencia de Datos e Ingeniería de Computadores. Boletín Oficial de la Universidad de Granada, nº80. 11 de abril de 2014. (2014)
4. Consejería de Economía, Innovación, Ciencia y Empleo, Junta de Andalucía: Decreto 113/2014, de 15 de julio. Boletín Oficial de la Junta de Andalucía, Boletín número 138 de 17/07/2014 (2014)
5. Ministerio de Educación, Cultura y Deporte, Gobierno de España: Resolución de 2 de octubre de 2014, de la Secretaría General de Universidades, por la que se publica el Acuerdo del Consejo de Ministros de 26 de septiembre de 2014, por el que se establece el carácter oficial de determinados títulos de Máster y su inscripción en el Registro de Universidades, Centros y Títulos., Boletín Oficial del Estado, núm. 253 de 18 de octubre de 2014, páginas 84664 a 84675 (2014)
6. Universidad de Granada: Resolución de 25 de noviembre de 2014, de la Universidad de Granada, por la que se publica el plan de estudios de Máster en Ciencia de Datos e Ingeniería de Computadores., Boletín Oficial del Estado, núm. 304, de 17 de diciembre de 2014, páginas 102247 a 102249 (2014)
7. Universidad de Granada, Máster Universitario en Desarrollo de Software, http://masteres.ugr.es/master-desarrollo-software/
8. Universidad de Granada, Máster Universitario en Gestión y Tecnologías de Procesos de Negocio, http://masteres.ugr.es/mbagestiontic/
9. Universidad de Granada, Máster Universitario en Ingeniería de Telecomunicación, http://masteres.ugr.es/telecomunicacion/
10. Universidad de Granada, Máster Universitario en Ingeniería Informática, http://escuelaposgrado.ugr.es/static/EP_Management/*/showCard/M50/56/2
11. El Confidencial, España necesita medio millón de expertos en informática más de los que puede aportar, http://www.elconfidencial.com/espana/2015-04-17/espana-necesita-medio-millon-de-expertos-en-informatica-mas-de-los-que-puede-aportar_761546/
12. Ediciones Cinco Días, El 'big data' arrasa en las escuelas de negocios, http://cincodias.com/cincodias/2015/05/01/sentidos/1430493583_087196.html
13. Universia España, El científico de datos: una novedosa y necesaria profesión, http://noticias.universia.es/ciencia-nn-tt/noticia/2014/05/06/1095994/cientifico-datos-novedosa-necesaria-profesion.html
14. Valdivia, J.F.: Las TIC: el nuevo motor económico de Granada. Diario IDEAL, Granada 12/4/2015, 33 (2015)

15. Center for World-Class Universities (CWCU) of Shanghai Jiao Tong University,, 2014 Academic Ranking of World Universities (ARWU), http://www.shanghairanking.com/es/
16. National Taiwan University, 2014 National Taiwan University Ranking (NTU Ranking), http://nturanking.lis.ntu.edu.tw/
17. Duranton, M., Bosschere, K.D., Cohen, A., Maebe, J., Munk, H.: HiPEAC Vision 2015. High Performance and Embedded Architecture and Compilation. In: FP7 HiPEAC Network of Excellence (ed.), http://www.hipeac.org/assets/public/publications/vision/hipeac-vision-2015_Dq0boL8.pdf
(2015)
18. Fernández de Vega, F., Hidalgo Pérez, J.I., Lanchares, J.: Parallel Architectures and Bioinspired Algorithms. Studies in Computational Intelligence, pp. 286. Springer, Berlin (2012)
19. Hwang, K., Fox, G.C., Dongarra, J.J.: Distributed and cloud computing : from parallel processing to the Internet of things. Elsevier Morgan Kaufmann, Waltham [Massachusetts] (2012)
20. Mayer-Schönberger, V., Cukier, K., Iriarte, A.: Big data : la revolución de los datos masivos. Turner, Madrid (2013)
21. Provost, F., Fawcett, T.: Data science for business : what you need to know about data minig and data-analytic thinking. O'Reilly, Sebastopol, CA (2013)
22. Leskovec, J., Rajaraman, A., Ullman, J.D.: Mining of massive datasets. Cambridge University Press, Cambridge (2014)
23. Schutt, R., O'Neil, C.: Doing data science : straight talk from the frontline. O'Reilly, Sebastopol [California] (2014)
24. Herrera, F.: Big data: Procesando los datos en la sociedad digital. Revista española de Física 28, 40-44 (2014)
25. Kaggle Community, Kaggle: The Home of Data Science, http://www.kaggle.com
26. R Foundation, R: The R Project for Statistical Computing, http://www.r-project.org/
27. Consejo de Gobierno de la Universidad de Granada, Normativa de evaluación y de calificación de los estudiantes de la Universidad de Granada, http://secretariageneral.ugr.es/pages/normativa/fichasugr/ncg7121

# Ingeniería de Computadores en la era del *Big Data*: Computación de altas prestaciones en clasificación y optimización

Julio Ortega; Pedro Martín-Smith; Jesús González Peñalver; Miguel Damas
Departamento de Arquitectura y Tecnología de Computadores
E.T.S.I.I.T., Universidad de Granada
jortega@ugr.es

**Resumen.** Este artículo describe las necesidades que las aplicaciones de la ciencia de datos plantean a las arquitecturas de computador y las consecuencias que comportan en la enseñanza de asignaturas de este ámbito. Como ejemplo, de esta relación entre aplicaciones y arquitectura de computadores se describe la asignatura Computación de Altas Prestaciones en Clasificación y Optimización, impartida en el Máster en Ciencia de Datos e Ingeniería de Computadores de la Universidad de Granada.

**Palabras clave:** *Big Data*, Ciencia de Datos, Clasificación, Computadores de Altas Prestaciones, Ingeniería de Computadores, Optimización.

**Abstract.** This paper describes the requirements that present big data applications demand from computer architectures and their consequences in the teaching of subjects in this area. As an example of these relationships between applications and computer architectures, the subject High Performance Computing on Classification and Optimization included in the Master of Data Science and Computer Engineering of the University of Granada (Spain) is described.

**Keywords:** Big Data, Classification, Computer Engineering, Data Science, High Performance Computers, Optimization.

## 1 Introducción

La capacidad para generar y almacenar información ha permitido disponer de grandes cantidades de datos y plantear nuevas aplicaciones de gran interés socio-económico que ahora sí podrían abordarse. Extraer conocimiento de estos grandes volúmenes de información plantea retos importantes, no solo a las técnicas de minería de datos existentes, sino también a las plataformas de cómputo donde se ejecutan [1]. Así, para conseguir una implementación eficiente de las aplicaciones de minería de datos que definirán la denominada era del *Big Data*, las arquitecturas de computador deben

satisfacer ciertos requisitos que determinarán las características de los computadores por la relevancia de las aplicaciones de datos y, por tanto, deberían tenerse en cuenta en el planteamiento de estrategias docentes para diseñar el proceso de enseñanza-aprendizaje en muchas de las asignaturas relacionadas con la ingeniería de computadores.

El Máster de Ciencia de Datos e Ingeniería de Computadores que se imparte en la Universidad de Granada [2] pone de manifiesto esta necesidad de tener en cuenta los requisitos de las aplicaciones de *Big Data* y las características de los computadores que permiten su implementación eficiente. En este artículo, además de detallar algunas de las implicaciones de la Ciencia de Datos en la Ingeniería de Computadores, se describe la asignatura de Computación de Altas Prestaciones en Clasificación y Optimización, como un ejemplo concreto con el que ilustrar esas implicaciones.

Así, la Sección 2 del artículo analiza las demandas que plantea la denominada Ciencia de Datos a la Ingeniería de Computadores, y las características tecnológicas que harán posible aplicaciones de los grandes volúmenes de datos existentes, y definirán el camino hacia la era del *Big Data*. La Sección 3 proporciona información sobre el Máster de Ciencia de Datos e Ingeniería de Computadores que se imparte en la Universidad de Granada. La asignatura Computación de Altas Prestaciones en Clasificación y Optimización se encuentra incluida en la especialidad de Ingeniería de Computadores y Redes del máster mencionado, y se analiza con detalle en la Sección 4. Finalmente, las conclusiones del artículo se recogen en la Sección 5 y las referencias se indican en la Sección 6.

## 2 Computadores para el *Big Data*

En el informe “Frontiers in Massive Data Analysis" del Nacional Research Council de Estados Unidos [1] se describen, entre otros aspectos los límites para el desarrollo de las aplicaciones de análisis de grandes volúmenes datos. Los datos generados en experimentos, simulaciones, o recogidos por sensores dan lugar a bases de datos con tamaños del orden de los petabytes. En 2010 se generaron 1.2 Zettabytes ($10^{21}$ bytes, es decir, mil trillones de bytes), Facebook genera alrededor de 10 Tbytes/día, y Twitter, 7 TB/día. Compañías como Google o Microsoft disponen de datos en cantidades del orden del exabyte ($10^{18}$ bytes), es decir, trillones de bytes (hay que tener en cuenta que desde el Big Bang se estima que han pasado $10^{17}$ segundos). Se estima que el volumen de datos almacenado crece a un ritmo mayor que el que marca la ley de Moore para el número de transistores en un circuito integrado (que sirve también como ritmo de referencia para el ritmo de mejora de la capacidad de los computadores). Así, frente al factor de crecimiento de 2 cada dos años que parece indicar la ley de Moore actualmente, el volumen de datos almacenados parece presentar un factor de crecimiento próximo a 2.5 cada dos años. Esta diferencia en los ritmos de crecimiento, denominada brecha de los datos (*data deluge gap*) plantea retos importantes en cuanto a la captura, procesamiento, distribución, comunicación, y análisis de los datos. Por tanto, esta situación también tiene consecuencias importantes en el desarrollo de arquitecturas de computación eficientes [3].

En líneas generales, se puede decir que el almacenamiento, acceso y procesamiento de grandes volúmenes de datos requiere de sistemas de cómputo paralelos y distribuidos. De hecho, ya desde hace algún tiempo, los microprocesadores incorporan varios núcleos de procesamiento para poder mantener el ritmo de mejora de prestaciones que marca la ley de Moore, con las restricciones en la frecuencia de reloj que plantean las limitaciones en el consumo energético del microprocesador. Pero la capacidad de procesamiento constituye solo una parte de lo que deben ofrecer las arquitecturas. Las E/S y el almacenamiento también deben ser capaces de proporcionar acceso rápido a los datos. Los datos deben estar próximos a los procesadores de las plataformas paralelas/distribuidas que los procesan, y para conseguir un ancho de banda de E/S aceptable, también el almacenamiento (y los sistemas de ficheros) deben encontrarse distribuidos. La jerarquía de memoria también es esencial. Cada vez es más frecuente el uso de aceleradores que cooperan con las CPUs de los nodos. Entre estos aceleradores, las GPU (Unidades de Procesamiento Gráfico) se vienen utilizando frecuentemente y constituyen una alternativa muy popular de cara a conseguir ganancias de velocidad considerables con consumos muy competitivos. La comunicación a través de la memoria con estos aceleradores, y el uso de los distintos niveles de almacenamiento de las tarjetas de GPU también son elementos a tener en cuenta.

Además de su conocida ley sobre el paralelismo, Gene Amdahl también propuso una serie de leyes, o más bien reglas prácticas (denominadas en inglés *rules of thumb*) para el diseño de computadores [4,5]: la ley del sistema equilibrado (un sistema necesita un bit de E/S por segundo por instrucción por segundo), la ley de la memoria (en un sistema equilibrado la relación α=Bytes/IPS es igual a 1), y la ley de las E/S (los programas hacen una E/S por cada 50.000 instrucciones). Estas *leyes* pueden seguir teniéndose en cuenta actualmente, aunque con algunas modificaciones en los valores de algunos de los parámetros [5]. Así en [6] y [7] se analizan desde ese punto de vista los requisitos que plantean, respectivamente, los centros de datos, y los entornos de programación de aplicaciones de Big Data como Hadoop y DataMPI. En [7] se habla de la denominada segunda ley de Amdahl (que incluye las leyes del sistema equilibrado y de la memoria: por cada instrucción por segundo de velocidad de CPU se debe proporcionar un bit/s de E/S y un Byte de memoria principal) y, para evaluar los sistemas, se utilizan el número de memoria de Amdahl, o relación α=GBytes/GIPS (GIPS es Giga Instrucciones por segundo), y el número de E/S de Amdahl, NESA=Ancho de Banda (en bits/s)/IPS, que debería ser igual a 1 en un sistema equilibrado. Mientras que en sistemas de cómputo de altas prestaciones, para reducir el coste del sistema, se tienen números de Amdahl (valor mínimo de α y NESA) tan bajos como 0.001, en sistemas diseñados para el procesamiento de datos intensivo (DataScope, GrayWulf, sistema de análisis de datos de eBay) se tienen valores entre 0.5 y 1 [7].

Las tareas de adquisición, búsqueda o visualización tienen una complejidad que crece linealmente con el volumen de los datos. Sin embargo las tareas de minería de datos suelen tener complejidades no lineales, y por ejemplo, muchos algoritmos de clustering pueden tener una complejidad superior a $N^2$, siendo N el número de datos sobre los que se aplica: un incremento de mil en el tamaño de los ficheros de datos supone un aumento en el tiempo de procesamiento de un factor de más de un millón

[8]. Por otra parte, el carácter inherentemente distribuido de las aplicaciones implica una clara dependencia de la capacidad de las redes que interconectan los elementos del sistema. Esta dependencia no sólo se da a nivel del sistema de E/S sino también en la jerarquía de memoria de forma que, a medida que aumenta la utilización del procesador, también lo hace la de la memoria y las E/S [6]. Además, en la arquitectura de capas de un sistema de Big Data hay que tener en cuenta [7] tanto la capa de hardware (por ejemplo un cluster de computadores), la de software de sistema (sistema operativo, middleware de gestión del servidor, etc.), la del entorno para Big Data (Hadoop, DataMPI, etc.) y la de la aplicación propiamente dicha (aplicaciones de ordenación, generación de histogramas de datos, clustering, etc.). Por ejemplo, en el conjunto de benchmarks *BigDataBench* [9] se encuentran cargas de trabajo que presentan perfiles diferentes. Así, para MapReduce, *WordCount* (cuenta las veces que aparece una palabra en el conjunto de datos de entrada) implica una carga intensiva en el uso del procesador, *Sort* (ordena los datos de entrada en función de unas claves dadas) es intensiva en cuanto a E/S, y *K-means* (algoritmo que agrupa los datos en K *clusters* o agrupaciones) también es intensivo en cuanto al uso de CPU, y además ejecuta varias iteraciones de forma que el resultado de una iteración es la entrada de la siguiente, permitiendo analizar el uso que se hace de la memoria.

El análisis de las ejecuciones de los benchmarks con distintas aplicaciones y cargas de trabajo considerando su efecto en los tiempos de ejecución, fallos de cache y TLBs, tasa de error en las predicciones de los saltos, junto con los anchos de banda de E/S, de comunicación, uso de memoria y de CPU permiten analizar hasta qué punto una determinada configuración de aplicación/entorno de Big Data/SO/hardware es más o menos eficiente. Por ejemplo, en [7] se obtiene, para un conjunto de benchmarks con distintas cargas de trabajo (volúmenes de datos) y sobre una misma plataforma SO/hardware, que Hadoop es más lento que DataMPI, con números de Amdahl más pequeños. Con ello se tiene que DataMPI, al permitir mejores prestaciones con menos exigencia en cuanto a ancho de banda de E/S y memoria, es más eficiente que Hadoop. En [10] se utilizan los números de Amdahl para evaluar distintos sistemas de bases de datos para distintas cargas de trabajo y se proponen una serie de conclusiones a tener en cuenta en el diseño de una plataforma para Big Data, poniendo de manifiesto el gran potencial para el desarrollo de nuevas arquitecturas orientadas al Big Data. En general, y dada la dinámica y diversidad (tanto dentro de una aplicación como entre distintas aplicaciones) en cuanto a patrones de uso y requisitos de los recursos, se promueven los sistemas heterogéneos con subsistemas equilibrados entre sí, y optimizados para las distintas fases de una carga de trabajo, y para las cargas de trabajo más representativas.

## 3 El Máster de Ciencia de Datos e Ingeniería de Computadores

El máster de Ciencia de Datos e Ingeniería de Computadores de la Universidad de Granada es impartido por profesores de los Departamentos de Arquitectura y Tecnología de Computadores y Ciencias de la Computación e Inteligencia Artificial desde el curso 2014/15. En la página web del máster [2] se puede encontrar información sobre los objetivos y las competencias, el plan de estudios, y el resto de

datos académicos del título. Tal y como se recoge en dicha página web, a través de este máster se busca la sinergia entre ámbitos de investigación que abarcan desde los niveles del desarrollo de algoritmos inteligentes hasta su implementación en distintas plataformas hardware, con el denominador común de las aplicaciones que implican el análisis de grandes volúmenes de datos. Teniendo en cuenta la tendencia actual hacia sistemas que interconectan una gran cantidad de dispositivos y sensores y a la disponibilidad de infraestructuras con grandes volúmenes de almacenamiento, es previsible la demanda de especialistas competentes en los contenidos del máster. El máster está estructurado en dos especialidades (especialidad en Ingeniería de Computadores y Redes, y especialidad en Ciencia de Datos y Tecnologías Inteligentes) que se corresponden, respectivamente, con los niveles más relacionados con el desarrollo de algoritmos y con la configuración de plataformas físicas, aunque incluyendo asignaturas que contemplan la interrelación entre las características del hardware y los requisitos de los algoritmos para una ejecución eficiente.

Tabla 1. Estructura de módulos y créditos del Máster en Ciencia de Datos e Ingeniería de Computadores de la Universidad de Granada [2]

<table>
<tr><td>Módulo Obligatorio<br>(12 créditos)</td><td>Metodología de la Investigación<br>Introducción a la Ciencia de Datos<br>Emprendimiento y Transferencia del Conocimiento</td><td></td></tr>
<tr><td>Módulo de Nivelación de Conocimientos<br>(8 créditos)</td><td>Servidores Seguros<br>Sistemas empotrados y co-diseño hw/sw<br>Minería de datos: preprocesamiento y clasificación<br>Minería de datos: aprendizaje no supervisado y detección de anomalías</td><td></td></tr>
<tr><td rowspan="2">Especialidad<br>(24 créditos de especialidad elegida + 4 de cualquier especialidad</td><td>Especialidad de Ingeniería de Computadores y Redes</td><td>Módulo de Computación de Altas Prestaciones<br>Módulo de Sistemas de Aplicación Específica</td></tr>
<tr><td>Especialidad en Ciencia de Datos y Tecnologías Inteligentes</td><td>Módulo de Modelos Avanzados de Ciencia de Datos<br>Módulo de Big Data y Cloud Computing<br>Módulo de Tecnologías Inteligentes e Inteligencia Computacional<br>Módulo de Aplicaciones en Ciencia de Datos y Tecnologías Inteligentes</td></tr>
<tr><td>Proyecto de Fin de Máster<br>(12 créditos)</td><td colspan="2"></td></tr>
</table>

En la Tabla 1 se muestra la organización en módulos y los créditos que debe superar el estudiante en cada uno de ellos para alcanzar el título. Como se puede ver, se deben cursar 12 créditos de un bloque obligatorio que incluye asignaturas fundamentales de formación para el investigador en TIC (Metodología de la Investigación, Introducción a la Ciencia de Datos, y Emprendimiento y Transferencia del Conocimiento). Hay un bloque de nivelación de conocimientos que incluye cuatro asignaturas de las que el estudiante debe cursar dos (8 créditos) para completar su formación previa en los aspectos que sean precisos para la especialidad que vaya a cursar. Finalmente el estudiante debe cursar las asignaturas de la especialidad que haya elegido: un total de 24 créditos de asignaturas de dicha especialidad, más cuatro créditos que pueden ser de la propia especialidad o de la que no ha elegido. Los 60 créditos del máster se completan con el Proyecto de Fin de Máster de 12 créditos. Más detalles sobre este máster se pueden encontrar en [11].

La especialidad de Ingeniería de Computadores y Redes incluye dos módulos, el de Computación de Altas Prestaciones, y el de Sistemas de Aplicación Específica. La asignatura de Altas Prestaciones en Clasificación y Optimización se imparte dentro del módulo de Computación de Altas Prestaciones en la especialidad de Ingeniería de Computadores y, precisamente, se encuentra entre esas asignaturas, a las que nos referimos más arriba, que ilustran los beneficios que los distintos tipos de computadores paralelos pueden aportar a la ejecución eficiente de procedimientos recientemente propuestos para resolver problemas de clasificación y optimización complejos. En la siguiente sección se analizan sus contenidos.

## 4 Computación de Altas Prestaciones en Clasificación y Optimización

El modelo de mente basada en el reconocimiento de patrones pone de manifiesto la importancia de la clasificación en el ámbito de los sistemas cognitivos y en los proyectos relacionados con el estudio del cerebro y las teorías acerca de la mente. Por otra parte, muchas de las aplicaciones del aprendizaje automático (*machine learning*) y la inteligencia artificial implican problemas de clasificación y optimización costosos en cómputo y almacenamiento. En las aplicaciones de minería de datos se busca descubrir patrones potencialmente útiles en los conjuntos de datos de los que se dispone, y se plantea la construcción de modelos descriptivos y predictivos eficientes que se ajusten a los datos disponibles, tengan capacidad generalizadora y permitan decidir acerca de acciones futuras. Por tanto, la minería de datos implica resolver, entre otros, problemas de selección de características, clasificación, clustering, y optimización. Las aproximaciones con que se abordan estos problemas, y las arquitecturas de cómputo que hacen posible su ejecución, se ven afectadas por el incremento exponencial de los volúmenes de datos a procesar en las aplicaciones de Big Data [12]. La asignatura se centra en los problemas de clasificación, clustering, y optimización, que constituyen el núcleo de muchas de las aplicaciones de Big Data.

Los temas a través de los cuales se han organizado los contenidos a impartir son los siguientes:

1. Arquitecturas de computador paralelas
2. Introducción a los Modelos con paralelismo implícito en problemas de clasificación y optimización: Modelos bioinspirados (redes neuronales y computación evolutiva)
3. Clasificación y Clustering con Modelos Neuronales: Sistemas Autoorganizativos.
4. Computación evolutiva paralela en problemas de optimización mono y multi-objetivo.
5. Implementaciones en plataformas paralelas y distribuidas: arquitecturas multi-núcleo, multicomputadores, e infraestructuras de cloud.
6. Aplicaciones de los problemas de clasificación y optimización complejos: Big Data, BCI, bioinformática, detección de intrusos en redes, etc.

El primer bloque de contenidos incluye los conceptos relacionados con las arquitecturas de computador que pueden dar respuesta a las necesidades de las aplicaciones de clasificación y optimización. Se revisan los distintos tipos de arquitecturas paralelas, el tipo de paralelismo que implementan, y las tendencias futuras en las mismas, poniendo de manifiesto la interacción entre los requisitos que plantean las aplicaciones más demandadas en cada momento y las innovaciones que van incorporando las arquitecturas de los computadores. En esta línea, se estudiarán las arquitecturas multinúcleo, incluyendo las arquitecturas multinúcleo heterogéneas y las unidades de procesamiento gráfico (GPU, Graphic Processing Units) muy extendidas actualmente como coprocesadores para acelerar la ejecución de ciertas aplicaciones. También se ilustrará la importancia de la jerarquía de memoria a través de los distintos tipos de memorias existentes en arquitecturas multinúcleo como las GPU y los procesadores de red (NP), y no menos importante será el análisis de la gestión del almacenamiento de disco y las consecuencias que plantean en el mismo las aplicaciones que utilizan grandes volúmenes de datos.

Una vez descritas las plataformas, se pasa a estudiar los algoritmos que utilizarán esas arquitecturas para proporcionar el nivel de prestaciones requerido por las aplicaciones que los usan. Así, se consideran los modelos bioinspirados como son las redes neuronales artificiales y los algoritmos evolutivos. Aparte de analizar sus características y de ilustrar el tipo de aplicaciones en las que se utilizan, también se contempla su perfil desde el punto de vista del paralelismo implícito que poseen, y de su implementación en plataformas paralelas y distribuidas de altas prestaciones, para abordar problemas de clasificación y optimización complejos que aparecen en aplicaciones de interfaz cerebro-computador (BCI, *Brain-Computer Interface*), bioinformática, detección de intrusos en redes, etc. Tras estudiar características de estos modelos bioinspirados, entre los que están la autoorganización, se analizan con detalle los mapas autoorganizativos (SOM) [13] y su uso en problemas de clasificación y clustering, para pasar a la computación evolutiva en problemas mono y multiobjetivo. A continuación se considerarán los aspectos relacionados con su implementación eficiente en distintos tipo de plataformas (entre ellas las de tipo

cloud), los paradigmas de programación y las técnicas con buenas características de elasticidad y disponibilidad (entre ellas MapReduce).

Un ejemplo de uso de arquitecturas de altas prestaciones en una aplicación que incluye optimización y clasificación, y que ilustra la perspectiva desde la que se plantea la asignatura, se puede encontrar en las tareas propias de las BCI basadas en la clasificación de electroencefalogramas (EEG). Por un lado se utilizan bases de datos correspondientes a electroencefalogramas de distintos sujetos y estados mentales a clasificar. Estos ficheros pueden tener un tamaño considerable al incluir señales recogidas a través de varios electrodos, y dadas sus características de baja relación señal-ruido de las señales, y el carácter temporal y no estacionario de las mismas. Requieren un preprocesamiento relativamente elaborado, y la caracterización de las mismas para su clasificación no es trivial. Así, usualmente, el número de características o componentes (*features* en inglés) que describen un EEG suele ser muy elevado, y puesto que, por otra parte, el número de EEG de que se dispone no es demasiado grande (dado el relativo coste temporal que implica la obtención de EEGs), el problema de clasificación de EEGs suele presentar el denominado problema de la maldición de la dimensionalidad (*curse of dimensionality*, en inglés). Es necesario llevar a cabo una selección de las características esenciales que permitan disponer de una relación suficientemente grande entre patrones (EEGs correspondientes a distintos estados mentales) y componentes de dichos patrones (características de los EEGs).

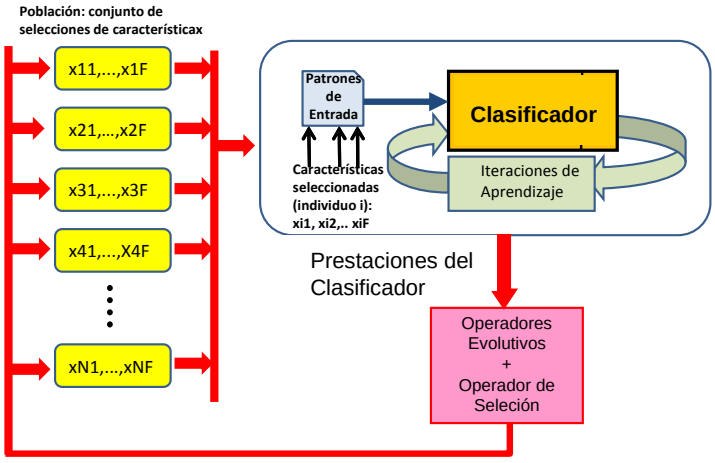


**Figura 1.** Procedimiento de tipo *wrapper* para la selección mediante optimización

En la Figura 1 se representa un procedimiento que permite la selección de características mediante un algoritmo de optimización evolutivo que tiene en cuenta las prestaciones que alcanza un clasificador entrenado (ajustado) con patrones que tienen como componentes las características cuya selección está codificada a través de los individuos de la población del algoritmo evolutivo. En este procedimiento de tipo *wrapper*, la calidad de los individuos de la población del algoritmo evolutivo viene dada por las prestaciones alcanzadas por el clasificador utilizado, y estas prestaciones

pueden definirse no sólo a través de un índice como puede ser el índice de aciertos de clasificación para los patrones utilizados como conjunto de test, sino también evaluar sus propiedades de generalización. Así, se puede introducir el uso de la optimización multi-objetivo en la selección de características. Igualmente, en el caso de que no se disponga de patrones etiquetados con sus clases, la evaluación de las características seleccionadas se puede hacer a partir de un procedimiento de *clustering* no supervisado, y se pueden estudiar el uso de mapas autoorganizativos, por ejemplo.

En cuanto al uso de arquitecturas paralelas, dado que el número de características entre las que hay que encontrar la mejor selección es muy elevado (cientos de características), estamos ante un problema de alta dimensionalidad en el espacio de variables de decisión del algoritmo de optimización multiobjetivo. En esta situación, el uso de computadores paralelos es esencial. En la asignatura se analizarán los distintos tipos de algoritmos paralelos (evaluación paralela del fitness de los individuos de la población, evaluación de subpoblaciones, etc.) y sus implementaciones (maestro-trabajador, islas, etc.), y se proporcionarán modelos de prestaciones que permitan entender las condiciones en las que una alternativa puede ser más beneficiosa que otra. Una alternativa para el aprovechamiento del paralelismo de datos que está recibiendo mucha atención es el uso de las GPU (*Graphic Processing Units*). La Figura 2 proporciona un esquema que ilustra las ventajas y los problemas de la implementación de un algoritmo evolutivo (en este caso un algoritmo de optimización multiobjetivo basado en el NSGA-II) con paralelismo basado en evolución de subpoblaciones en una arquitectura de GPU. Si bien es posible aprovechar el paralelismo de datos al evaluar en paralelo los individuos de la población, hay que tener en cuenta que deben implementarse operadores de selección, determinación de relaciones de dominancia, operadores de cruce, etc., que implican la comparación de varios individuos, y por lo tanto cierta comunicación y sincronización entre hebras. Por otra parte, si la evaluación de las soluciones implica ajustar un clasificador, o un mapa autoorganizativo, utilizando información de ficheros de patrones de un tamaño superior al de la memoria local disponible, el costo de las transferencias de memoria para acceder a dicha información (conexiones de los bloques de *fitness evaluation* a la memoria global de la Figura 2) puede comprometer el beneficio que aporta el aprovechamiento del paralelismo de la arquitectura. En cualquier caso, se trata de una oportunidad única para poner de manifiesto la interacción entre procesadores, jerarquía de memoria, y sistemas de interconexión, y analizar el compromiso entre paralelismo y localidad que exigen las aplicaciones consideradas y que determina la capacidad de una arquitectura dada para satisfacer los requisitos de prestaciones que se establecen.

En resumen, y teniendo en cuenta estos planteamientos, a través del temario de la asignatura se pretende que el estudiante sea capaz de:

- Identificar el paralelismo implícito en los modelos bioinspirados neuronales y evolutivos y las posibilidades de las arquitecturas de cómputo paralelas actuales que pueden aprovechar.
- Proponer modelos neuronales o evolutivos plausibles en la resolución de problemas de clasificación y optimización.

- Identificar los principios del comportamiento autoorganizativo y aplicarlos en problemas de clustering y clasificación.
- Distinguir entre problemas de optimización mono y multi-objetivo y estimar las diferencias en complejidad que plantea su resolución mediante aproximaciones basadas en computación evolutiva paralela.
- Identificar y proponer distintas alternativas para la implementación paralela de procedimientos de clasificación y optimización teniendo en cuenta las características de las arquitecturas de cómputo a utilizar (multiprocesadores, multicomputadores, o plataformas distribuidas) y los paradigmas de programación (entre estos, paradigmas como MapReduce para el tratamiento de grandes volúmenes de datos).
- Proponer procedimientos de clasificación y optimización de altas prestaciones en ejemplos de aplicaciones de complejidad elevada o que impliquen un procesamiento de información no estructurada (análisis de datos complejos, Brain-Computer Interfaces, etc.).
- Implementar los procedimientos de clasificación y optimización estudiados a través de herramientas de programación (como por ejemplo Matlab u Octave).

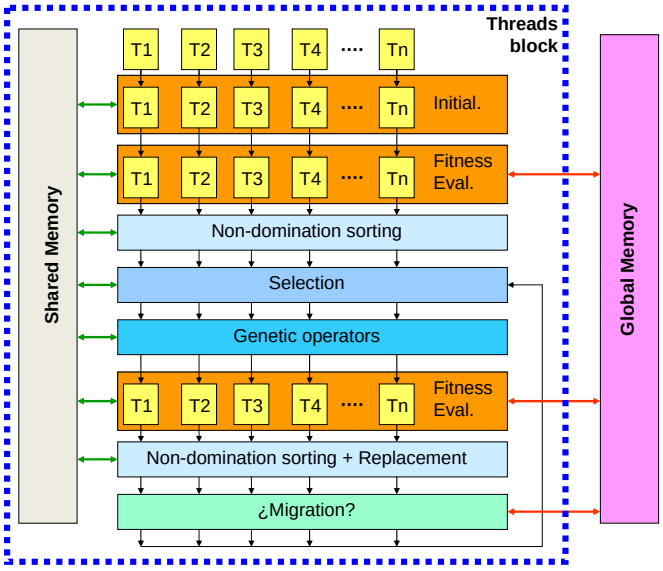


**Figura 2.** Esquema de implementación en una GPU de la evolución de una subpoblación en un procedimiento evolutivo de optimización multiobjetivo.

## 5 Conclusiones

La asignatura Computación de Altas Prestaciones en Clasificación y Optimización pone de manifiesto los aspectos de esas aplicaciones que requieren el uso de arquitecturas paralelas, y la evolución previsible en las arquitecturas de computador para dar respuesta a las exigencias de la minería de datos en el contexto del Big Data. Las competencias específicas del título de máster en Ciencia de Datos e Ingeniería de Computadores que el estudiante adquirirá a través de esta asignatura son las siguientes:

- Capacidad para la aplicación de técnicas y metodologías que permitan abordar desde nuevas perspectivas los problemas de interés, gracias a la disponibilidad de las plataformas de computación y comunicación con altos niveles de prestaciones.
- Capacidad de análisis de aplicaciones en ámbitos de biomedicina y bioinformática, optimización y predicción, control avanzado, y robótica bioinspirada, tanto desde el punto de vista de los requisitos para una implementación eficaz de los algoritmos y las técnicas de computación que se usan para abordarlas, como de las características deseables en las arquitecturas donde se ejecutan.

## 6 Referencias


1. National Research Council. “Frontiers in Massive Data Analysis”. Washington, D.C.: The National Academies Press, 2013.
2. Página web del Máster en Ciencia de Datos e Ingeniería de Computadores (Universidad de Granada). http://masteres.ugr.es/datcom/pages/master, 2015.
3. HiPEAC (European Network of Excellence on High Performance and Embedded Architecture and Compilation): “On the road for the HiPEAC Vision 2015”. https://www.hipeac.org/publications/vision/
4. Amdahl, G.M.:”Computer Architecture and Amdahl’s Law”. IEEE Computer, pp.38-46. Diciembre, 2013.
5. Gray, J.; Shenoy P.: “Rules of Thumb in Data Engineering”. In Proc. of the 16th IEEE Int. Conf. on Data Engineering, pp.3-12, 2000.
6. Cohen, D.; Petrini, F.; Day, M.D.; Ben-Yehuda, M.; Hunter, S.W.; Cummings, U.:”Applying Amdahl’s other law to the data center”. IBM J. Research & Development, Vol.53, No. 5, pp.683-694, 2009.
7. Liang, F.; Feng, C.; Lu, X.; Xu, Z.:”Performance characterization of Hadoop and DataMPI based on Amdahl’s second law”. 19th IEEE Int. Conf. on Networking, Architecture and Storage, pp.207-215, 2014.
8. Bell, G.; Gray, J.; Szalay, A.: “Petascale Computational Systems: Balanced CyberInfrastructure in Data-Centric World”. 2005.
9. Wang, L.; et al.:”BigDataBench: a Big Data Benchmark suite from Internet services”. Proc. of the 20th IEEE Int. Symp. On High Performance Computer Architecture (HPCA’14), 2014.
10. Chang, J.; Lim, K.T.; Byrne, J.; Ramirez, L.; Ranganathan, P.: “Workload diversity and dynamics in Big data analytics: Implications to System Designers”. In Proc. 2nd Workshop on Architectures and Systems for Big Data (ASBD’12), 2012.
11. Rojas, F.; Cano, A.; Gómez, M.; Ortega, J.; Herrera, F.; Romero-Zaliz, Rocío, González, J.:”Máster en Ciencia de Datos e Ingeniería de Computadores: una apuesta por la formación especializada en el sector TIC”. Revista de Enseñanza y Aprendizaje de Ingeniería de Computadores, 2015.
12. Wu, X.; Zhu, X.; Wu, G.-Q.; Ding, W.:”Data Mining with Big Data”. IEEE Trans. On Knowledge and Data Engineering, Vol.26, No.1, pp.97-107, 2014.
13. Kohonen, T.:”Self-Organizing Maps”. Springer, 2001.

# Sistema de Seguridad Basado en una Plataforma Heterogénea Distribuida

Diana Lora[+], Pablo Cerro[*], Alberto A. Del Barrio[+], Guillermo Botella[+]

Departamento de Arquitectura de Computadores y Automática, Facultad de Informática de la Universidad Complutense de Madrid
+{dlora, abarriog, gbotella}@ucm.es, * pablo.cerro.canizares@gmail.com

**Resumen.** Este trabajo presenta el proyecto "Ecosistema Digital de Seguridad", realizado en la asignatura Sistemas Empotrados Distribuidos, perteneciente a la titulación Máster en Ingeniería Informática de la Universidad Complutense de Madrid. En este trabajo se describe e implementa un sistema de seguridad por medio de una plataforma heterogénea, cuyo objetivo es la verificación de los usuarios autorizados. En concreto, la placa Raspberry Pi se encarga del reconocimiento de la imagen y la contraseña, mientras que la STM32F4 Discovery sirve de interfaz con el usuario, recibiendo la contraseña y enviándosela a la Raspberry a través de la interfaz RS-232.

**Palabras Clave:** Seguridad, Raspberry Pi, STM32F4 Discovery, Ecosistema de Seguridad, Sistemas Empotrados Distribuidos.

**Abstract.** This work presents the project entitled "Digital Security Ecosystem", carried out as a final project in the Distributed Embedded Systems subject, which belongs to the Computer Engineering Master, at the Complutense University of Madrid. In this work, a security system is described and implemented through a heterogeneous platform, whose objective consists of the authorized users' verification. Concretely, the Raspberry Pi board is responsible for identifying the image and the password, while the STM32F4 Discovery board acts as the user interface, receiving the password which will be sent to the Raspberry by using the RS-232 interface.

**Keywords:** Security, Rasberry Pi, STM32F4 Discovery, Security Ecosystem, Distributed Embedded Systems.

# 1 Introducción

En la actualidad, la globalización y la alta competitividad hacen que las organizaciones tengan mayor cuidado con la información vital que poseen. La necesidad de mantener el conocimiento de una organización en secreto ha traído el rápido desarrollo del sector de la seguridad, con el fin de evitar que información importante se encuentre accesible a la competencia.

Por tanto, son necesarias nuevas medidas de seguridad que vayan de la mano con las tendencias tecnológicas. Por ello, la biometría [2] ha tomado mayor fuerza en el mercado, ya que une la biología del ser humano con la seguridad. La biometría es una tecnología de reconocimiento de características físicas únicas de las personas, como por ejemplo el reconocimiento facial, reconocimiento del iris, la huella digital o tono de la voz. Los sistemas biométricos están compuestos de un dispositivo de captura y un software biométrico que interpreta la muestra y la transforma en una secuencia numérica. Debido a que los seres humanos tenemos características morfológicas únicas que nos diferencian de los demás, la biometría es considerada en la actualidad como el método mejor catalogado para la identificación humana [2].

El reconocimiento facial [3] es un tipo de aplicación de la biometría donde se puede identificar a una persona por medio de patrones del rostro. Este tipo de aplicaciones son comúnmente utilizadas en sistemas de seguridad por la gran fiabilidad y eficiencia que brindan. La mayor parte del software de reconocimiento facial está basado en códigos numéricos llamados *faceprints* [3]. Estos sistemas utilizan 80 puntos nodales del rostro humano para medir diferentes variables como son: ancho o alto de la nariz, la profundidad de las cuencas de los ojos y la forma de los pómulos, etc. Dichos puntos nodales son identificados en una imagen digital de la cara de un individuo, y posteriormente almacenados en forma de código numérico o faceprint. El faceprint se utiliza como base para la comparación con los datos capturados a partir de caras en imágenes o vídeos.

Gracias a la tecnología anteriormente descrita se hace más efectiva la autenticación y acceso del personal de una organización a un sistema o instalación. Previamente al desarrollo de la biometría, el acceso del personal a una instalación se hacía a través de una código secreto que cada individuo posee. Sin embargo, este método no es completamente fiable debido a que otra persona puede ingresar credenciales ajenas. Con la ayuda de la biometría se tiene la garantía de que la persona solicitando acceso a un sistema es en efecto quien dice ser.

Una de las necesidades principales de los sistemas de seguridad es la distribución de sus diferentes componentes y la interacción entre ellos. Por tanto, es un ejemplo de aplicación muy adecuado de los conceptos estudiados en la asignatura Sistemas Empotrados Distribuidos (SEDs) [9]. El sistema digital de seguridad es un SED que consta de dos componentes que interactúan entre sí, por medio de interfaz RS-232, con la finalidad común de verificar las credenciales del usuario.

El resto del artículo se organiza de la siguiente forma: la Sección 2 describe el diseño del sistema, mientras que la Sección 3 trata los detalles de implementación del mismo. En la Sección 4 se evalúa la plataforma por medio de varios experimentos y finalmente en la Sección 5 se presentan nuestras conclusiones, así como posibles líneas de trabajo futuro.

## 2 Diseño del Sistema Digital de Seguridad

Un ecosistema digital es cualquier sistema distribuido, con propiedades de auto-organización, de escalabilidad y sostenibilidad, que está inspirado en los ecosistemas naturales [1]. Dichos sistemas son entornos extendidos e interconectados, donde se intercambian datos entre sus componentes. Un ejemplo claro de ecosistema digital es la propia Internet.

El sistema digital de seguridad del presente proyecto está basado en dichos ecosistemas, y consta de dos componentes, como puede observarse en la Figura 1:

- El módulo de identificación de contraseñas. Este bloque será el encargado de recibir la clave introducida por el usuario, y enviarla al módulo reconocedor. Será implementado sobre la placa STM32F4 Discovery [11].
- El modulo reconocedor de usuarios. Este componente recibirá la clave del anterior módulo, y decidirá si pertenece al sistema o no. Además, realizará el reconocimiento facial del usuario. Combinando ambas informaciones, decidirá si el usuario puede acceder al sistema. Dicho bloque se implementará finalmente con la placa Raspberry Pi [10,12] y una cámara [6].

Estos dos componentes tienen su funcionalidad específica, pero existe una interacción entre ellos para la toma de decisiones y el funcionamiento correcto del sistema.

El flujo del sistema digital de seguridad es el siguiente:

1. Se identifica el usuario por medio de reconocimiento facial.
2. El usuario ingresa la contraseña en la plataforma STM32F4 Discovery.
3. La plataforma STM32F4 Discovery envía la contraseña a la Raspberry Pi.
4. La Raspberry Pi valida la contraseña con la que tiene almacenada en su base de datos.
5. La Raspberry Pi envía 1 para confirmar que el usuario tiene autorización y 0 para denegar acceso.
6. La plataforma STM32F4 Discovery le muestra al usuario el mensaje de “Bienvenido!” o “Denegado” de acuerdo a la respuesta recibida.

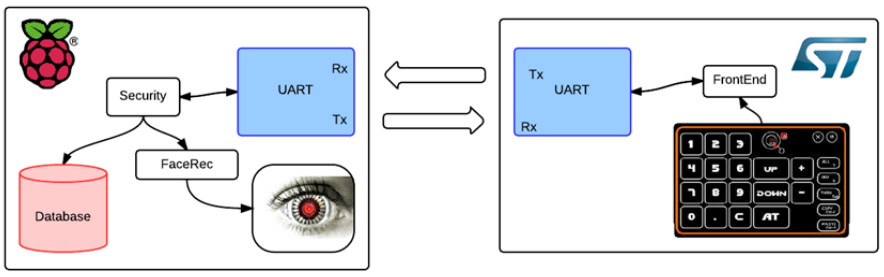


**Figura 1.** Diagrama de comunicación entre plataformas.

# 3 Implementación

En esta sección se describen los detalles de implementación del sistema. Primeramente hablaremos en profundidad de las funcionalidades realizadas por cada uno de los componentes, y posteriormente se explicará el proceso de envío de datos entre ambas placas.

## 3.1 STM32F4 Discovery

En el sistema digital de seguridad propuesto la plataforma STM32F4 Discovery es la encargada de recibir la contraseña ingresada por el usuario, enviarla a la Raspberry Pi y esperar respuesta de autenticación de usuario. La Figura 2 muestra un diagrama de bloques ilustrando estas ideas.

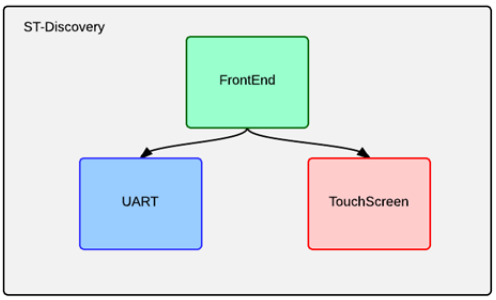


**Figura 2.** Funcionalidades realizadas por la plataforma STM32F4 Discovery.

Para la creación de la funcionalidad previamente descrita, se desarrolló una aplicación en el lenguaje de programación C# con acceso a los componentes necesarios de la STM32F4. En este caso, es preciso acceder a la pantalla táctil y la USART, que implementa la interfaz RS-232. La aplicación consiste en mostrar en pantalla un teclado numérico en el que el usuario ingresa su contraseña y cuando éste seleccione “Enviar”, la contraseña será enviada a la Raspberry Pi.

## 3.2 Raspberry Pi & Cámara

La Raspberry Pi implementa diferentes funcionalidades en el sistema. Por un lado, contiene la base de datos de contraseñas y realiza la autenticación de usuarios. Además, posee un sistema de reconocimiento facial, basado en la integración de una cámara y en el uso de las librerías OpenCV [5], que le permite detectar el rostro del usuario. La Raspberry Pi tiene un sistema de seguridad programado en C, el cual consulta en la base de datos de los usuarios registrados la validez de la información recibida. Finalmente, responde a la STM32F4 Discovery si la credencial recibida (clave+imagen) tiene acceso o no.

La Figura 3 muestra un diagrama con las funcionalidades principales implementadas por la Raspberry Pi.

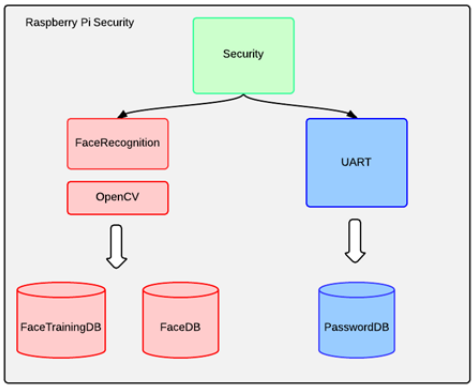


**Figura 3.** Funcionalidades realizadas por la plataforma Raspberry Pi.

Para realizar los cálculos referentes a la detección y reconocimiento facial se utilizan las librerías de código abierto OpenCV. Estas librerías son APIs sencillas, las cuales ofrecen diferentes funcionalidades como: machine learning, detección de objetos, efectos fotográficos, aceleración por GPU, y visión por computador. Ésta última es la utilizada en este proyecto. En [16] se muestran los pasos realizados para la configuración de las librerías SimpleCV y OpenCV en la Raspberry Pi. El algoritmo de reconocimiento de imagen es el *Haar Cascade Classifier*, capaz de obtener tasas de reconocimiento en torno al 95% y superiores [17,18].

### 3.3 Comunicación entre Plataformas

Para la comunicación entre las plataformas se utiliza la USART (Universal Synchronous/Asynchronous Receiver/Transmitter). Una USART es un dispositivo que permite la comunicación entre dispositivos por medio del puerto serie usando el protocolo RS-232C [4].

En el caso de la plataforma STM32F4 Discovery se utiliza la USART1, donde el pin PA10 sirve para recibir datos y el pin PA9 para enviar. Para la Rasberry Pi el pin 8 es el de envío de datos y el pin 6 es el encargado de la recepción de datos. En la Figura 4 se muestran la ubicación de los pines de la USART de la Raspberry Pi.

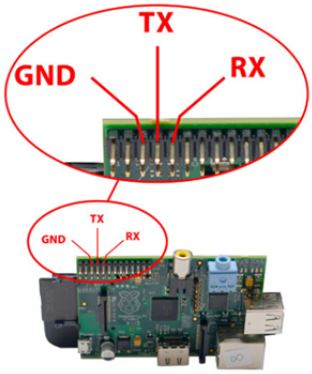


**Figura 4.** Identificación de pines en Raspberry Pi.

A continuación, debemos conectar los pines de transmisión y recepción de cada dispositivo de manera cruzada, es decir el pin USART-aRX del dispositivo A ha de conectarse al pin USART-bTX del dispositivo B (análogamente para USARTaTX USARTbRX). Además, es necesario que ambos compartan una toma de tierra (GND). La Figura 5 muestra la interconexión de ambas placas.

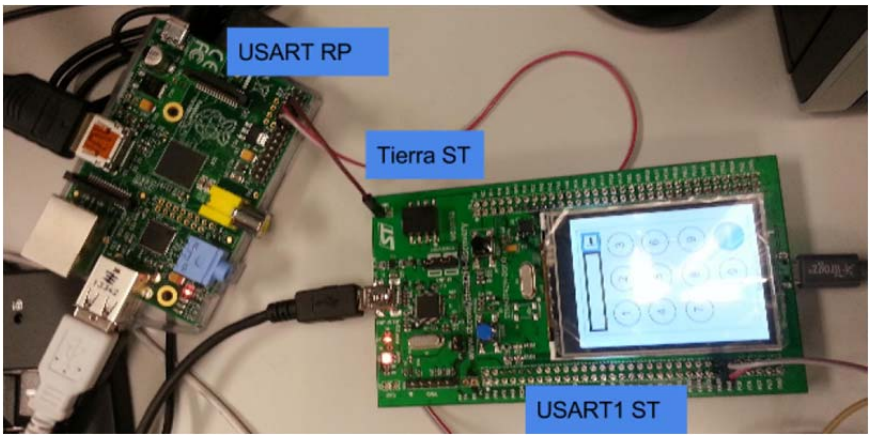


**Figura 5.** Configuración física del sistema.

## 4 Resultado

Para llegar a completar con éxito la construcción del sistema tras su modelado y configuración, se realizaron pruebas con las dos plataformas individualmente. Una vez probado que ambas plataformas funcionaban correctamente de manera individual, se procedió al ensamblaje final.

Para la realización de las pruebas se utilizó Termite [8]. Este programa es un terminal RS-232. Tiene una interfaz similar a una consola en la que se en la cual se muestra la información enviada por ambas plataformas.

A continuación se detallan en profundidad las pruebas realizadas para comprobar el funcionamiento del proyecto.

### 4.1 Análisis experimental con STM32Discovery

Para validar del correcto funcionamiento de la aplicación desarrollada sobre la plataforma STM32F4 Discovery, se conectó la placa al puerto serie de un PC y se probó la conectividad con el programa Termite. La prueba consistió en ingresar la contraseña en la placa de la plataforma STM32F4 Discovery, pulsar el botón enviar y comprobar que la información mostrada por Termite se correspondía a la ingresada en la Discovery.

### 4.2 Análisis experimental con Raspberry Pi

Al igual que las pruebas realizadas con la plataforma STM32F4 Discovery descritas en el apartado anterior, se decidió interconectar la Raspberry Pi con el puerto serie de un PC utilizando el programa Termite. Esta prueba fue realizada para comprobar la compatibilidad de la UART de RaspberryPi con una plataforma distinta.

Además, con el objetivo de comprobar el correcto funcionamiento de las librerías de visión artificial OpenCV, se realizaron pruebas inicialmente con imágenes estáticas proporcionadas por la librería de OpenCV y externas a ésta. Los resultados fueron favorables ya que el sistema reconocía exitosamente el rostro de la persona. La Figura 6 muestra un ejemplo de reconocimiento facial estático.

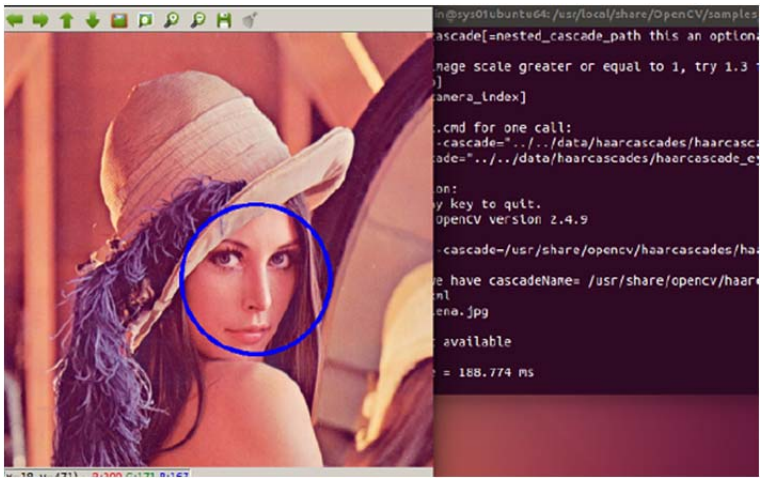


**Figura 6.** Pruebas OpenCV con imagen estática.

Posteriormente se realizaron pruebas con vídeos para validar que igualmente que con las imágenes, las librerías de OpenCV podían reconocer el rostro humano. La Figura 7 muestra un ejemplo de reconocimiento facial dinámico.

### 4.3 Análisis experimental con el sistema completo

Por último, se procedió a conectar las dos plataformas por puerto serie, como se mostró en la Figura 5. En la Sección 3.3 se encuentran los pines utilizados para la conexión entre la STM32F4 y la Raspberry Pi. Una vez conectado el sistema completo, el conjunto de pruebas realizadas en sistema digital fueron las mostradas en la Tabla I.

Además, se realizó un vídeo [13] en el que se puede observar el funcionamiento completo del sistema. La plataforma STM32F4 Discovery envía la contraseña introducida por el usuario a la Raspberry Pi. Ésta valida la contraseña en la base de datos, y si es correcta realiza el reconocimiento facial del individuo. La Rasberry Pi devuelve una variable validando o denegando el acceso de la persona.

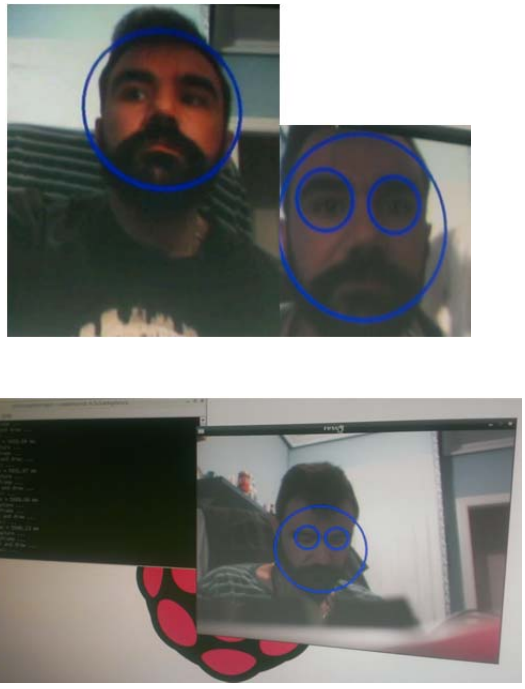

**Figura 7.** Proceso de identificación a través de cámara.

**Tabla I.** Casos probados para demostrar la funcionalidad del sistema

| # Caso | Discovery | Raspberry Pi | Resultado |
|---|---|---|---|
| 1 | Contraseña correcta | Faceprint correcto | OK |
| 2 | Contraseña correcta | Faceprint incorrecto | Fallido |
| 3 | Contraseña incorrecta | Faceprint incorrecto | Fallido |
| 4 | Contraseña incorrecta | Faceprint correcto | Fallido |

## 5 Conclusiones y Trabajo Futuro

En este artículo se ha presentado un sistema digital de seguridad. Dicho sistema trata de explotar tanto las virtudes de los sistemas empotrados distribuidos (bajo consumo, escalabilidad y tolerancia a fallos) como de la biometría. Por medio de dos placas de bajo consumo y coste, es posible implementar un sistema de alta seguridad basado en el reconocimiento de claves y el reconocimiento facial.

Como líneas futuras se podría refinar el sistema, siendo capaz de reconocer a diferentes personas y administrar distintos perfiles de seguridad para cada una de ellas. Por ejemplo, a partir de cierta hora únicamente dejar pasar a los directivos, denegar el acceso a personas despedidas, si se detecta a algún intruso comunicarse con la policía, etc.

Otra mejora interesante sería la sustitución de la comunicación serie por elementos wireless, como ZigBee, Bluetooth o incluso WiFi, y la aplicación de un protocolo de seguridad para el envío de claves, como RSA [19]. Usando una comunicación wireless, se podría escalar fácilmente el sistema y crear un sistema de cámaras distribuidas. De esta manera, podrían causar baja cualquiera de las cámaras vigilantes y el resto del sistema podría seguir funcionando, enviando una alarma por la baja causada.

## Referencias

1. Santamaría, Fernando. (2010). Una introducción a los ecosistemas digitales. Recuperado de http://fernandosantamaria.com/blog/2010/07/unaintroduccionalosecosistemasdigitales/
2. Sánchez Calle, Ángel. (2008). Aplicaciones En La Visión Artificial Y La Biometría Informática, Dykinson S.L.
3. Harry Wechsler (1998). Face Recognition: From Theory to Applications, Springer.
4. Ben Cohen (2001). Component Design by Example … A Step-by-Step Process Using VHDL with UART as Vehicle, VhdlCohen Publishing.
5. Eben Upton (2013). Raspberry Pi User Guide, Wiley, 3rd edition.
6. Simon Monk (2014). Raspberry Pi Cookbook, O'Reilly Media.

7. Horne, M. (2014). Raspberry Jam Potton Pi & Pints. Recuperado de http://www.recantha.co.uk/blog/?p=5261
8. CompuPhase. (2015). Termite: a simple RS232 terminal. Recuperado de http://www.compuphase.com/software_termite.htm
9. Máster en Ingeniería Informática de la Universidad Complutense de Madrid, http://informatica.ucm.es/estudios/2014-15/master-ingenieriainformatica
10. C. Edwards, Not-so-humble raspberry pi gets big ideas, Engineering Technology 8 (3) (2013) pp. 30-33.
11. STM32F4. (2014). Discovery kit for STM32F407/417 line: Data Brief. Recuperado de http://www.st.com/st-web-ui/static/active/en/resource/technical/document/data_brief/DM00037955.pdf
12. Raspberry Pi Foundation. Setting up your RasberryPi. Recuperado de http://www.raspberrypi.org/help/quick-start-guide/
13. Ecosistema Digital de Seguridad. Pruebas de Validación. Recuperado de https://www.youtube.com/watch?v=ZUqM24PdnIw&feature=youtu.be
14. Crisan, C. (2014). Motion Eye with Raspberry Pi. Recuperado de http://www.howtoembed.com/projects/raspberrypi/95motioneyewithraspberrypi
15. Stan Z. Li, Anil K. Jain (2011). Handbook of Face Recognition, Springer, 2nd edition.
16. Araujo, M. (2013). RASPBERRY PI + SIMPLECV + OPENCV + RASPICAM CSI CAMERA. Recuperado de http://tothinkornottothink.com/post/59305587476/raspberry-pi-simplecv-opencv-raspicam-csi-camera
17. P. I. Wilson and J. Fernandez. 2006. “Facial feature detection using Haar classifiers”. *J. Comput. Sci. Coll.* 21, 4 (April 2006), 127-133.
18. Viola, P.; Jones, M., "Rapid object detection using a boosted cascade of simple features," Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2001, pp.I-511,I-518.
19. Raúl Durán Díaz, Luis Hernández Encinas, Jaime Muñoz Masqué. “El criptosistema RSA” RA-MA S.A. Editorial y Publicaciones, 2005.

# Servicios y Seguridad, un enfoque basado en estrategias de ataque y defensa

Marcela Castro-León[1], Francesc Boixader[1], Manel Taboada[1], Dolores Rexachs[2], Emilio Luque[2]
[1]Escola Universitària d'Informàtica Tomàs Cerdà.
Sant Cugat del Vallès, Barcelona, España
{marcela.castro, francesc.boixader,manel.taboada}@eug.es
[2]Universidad Autónoma de Barcelona.
Barcelona, España
{dolores.rexachs, emilio.luque}@uab.es

**Resumen.** En este artículo se presenta el enfoque metodológico de la asignatura Servicios y Seguridad del Grado de Informática y Servicios, título oficial de la Universidad Autónoma de Barcelona que se imparte en la Escuela Universitaria de Informática Tomás Cerdá. Proponemos un enfoque basado en estrategias de ataque y defensa utilizadas en sistemas informáticos. Los modelos de estrategia constituyen el hilo conductor que permite relacionar cómo contribuyen el resto de temas, como criptografía, estándares de seguridad, metodologías de modelado de amenazas y de evaluación de riesgos en la configuración de un sistema de servicios web seguro. La parte práctica incluye sesiones de laboratorio y el desarrollo de un trabajo de *hacking*. En el laboratorio los alumnos aprenden a configurar la seguridad de un servidor de aplicaciones web, a generar certificados de servidor y de clientes, y a incluir opciones de seguridad en aplicaciones y en servicios web. Realizando el trabajo práctico los alumnos aprenden a defender mejor al sistema a través del conocimiento de las técnicas y herramientas que utilizan los atacantes para descubrir y explotar las vulnerabilidades de las infraestructuras y aplicaciones.

**Palabras Clave:** Servicios web, Seguridad de servicios, Estándares de seguridad de servicios web, WS-Security, XML-Security, *Threat-Modeling*.

**Abstract**. This article describes the methodological approach of the subject Services and Security of the Bachelor's Degree in Information Technology and Services (*Universitat Autonoma de Barcelona)*, which is taught at the Tomas Cerda Computer Science School. We propose an approach based on attack and defense strategies which are used in computer systems. Strategy models are the thread that relates how the rest of the topics as cryptography, security standards, threat modeling and risk assessment methodologies contribute in setting up a secure web service based system. The practical part includes laboratory sessions and development of a work of hacking. In the laboratory students learn to set up a web server application, to generate server and client's certificates, and to include security options into applications and web services. By doing the practical work students learn how to defend the system in a better way through the knowledge of the techniques and tools used by *hackers* to discover and exploit vulnerabilities of infrastructure and applications.

**Keywords:** web services, web-service security, web-service security standards, WS-Security, XML-Security, *Threat-Modeling*.

# 1 Introducción

El Grado de Informática y Servicios [1] título oficial de la Universidad Autónoma de Barcelona que se imparte en la Escuela Universitaria de Informática Tomás Cerdá, surge al detectar en el tejido económico de este país una demanda de profesionales en las empresas de servicios, y su propósito es acercar la informática a dicho sector.

El perfil objetivo de este grado se ajusta al de un ingeniero informático. Este profesional debe ser capaz de evaluar, diseñar e implementar los sistemas de información en los que la tecnología juega un papel clave, con el objeto de gestionar la información utilizada por las empresas en todas sus áreas de negocio.

La seguridad informática es hoy una de las competencias más requeridas de un ingeniero de servicios. Las empresas están muy interesadas en fortalecer sus sistemas a la luz de sus vulnerabilidades y de los ataques cibernéticos que se conocen día a día. La tendencia actual de utilizar plataformas de *Cloud Computing* hace que los responsables de las aplicaciones deban revisar la seguridad de las arquitecturas utilizadas para valorar si éstas están o no preparadas para asumir los riesgos de seguridad que conlleva el movimiento a la nube.

Dichas competencias se pretenden trabajar a través de la asignatura de Servicios y Seguridad que incluye el Plan de Estudios del Grado en Informática y Servicio. El temario para dicha asignatura puede ser extenso y diverso. A los temas de seguridad tradicional relacionados con tener un entorno web y una red segura, se tienen que añadir cómo se deberían programar o configurar los servicios web para intercambiar mensajes que cumplan con los objetivos de seguridad de identificación, autenticación, confiabilidad, integridad, privacidad y no repudio. Esto implica integrar una gran variedad de conceptos partiendo desde la seguridad en las redes, configuración de la infraestructura, sistemas de criptografía, vulnerabilidades de software, estándares de seguridad en servicios, etc.

Las estrategias de ataque y defensa utilizadas actualmente [2] constituyen el hilo conductor del temario de esta asignatura. Pueden ser de ataque o de defensa, y se clasifican de acuerdo a lo indicado en la **Tabla 1** - Tipos de estrategias de ataque y defensa. Son modelos abstractos de funcionamiento que abarcan todas las alternativas utilizadas. Son independientes de las metodologías y herramientas de seguridad actuales y pueden utilizarse en un futuro para relacionar tecnologías emergentes.

**Tabla 1 - Tipos de estrategias de ataque y defensa**

| **Estrategias** | Criterio | Tipos |
|---|---|---|
| **Ataque** | Según la trayectoria desde origen al destino | Directo; Progresivo; Masivos; Dirección Errónea |
| **Defensa** | Según el nivel de intromisión del atacante. | Decepción; Frustración; Resistencia; Reconocimiento y Recuperación |

El temario se desarrolla en referencia a dichas estrategias, y constituyen su marco conceptual. De esta manera, se pretende lograr una mejor cohesión y comprensión de la extensa variedad de conceptos a tratar. Así al inicio de la asignatura se explica cómo funcionan los modelos como mecanismos generales. El alumno está preparado

para comprenderlas sin conocimientos previos específicos. El resto de los temas se relacionan con sus correspondientes modelos de defensa y ataque.

A través del enfoque basado en estrategias el alumno aprende a diferenciar y a valorar el uso de las técnicas de seguridad conociendo qué aporta cada una en el cuadro global de la seguridad del sistema y de la información de la empresa.

La asignatura tiene dos tipos de actividades prácticas. Por un lado, los alumnos realizan una serie de problemas de laboratorio, en los cuales configuran la seguridad de un entorno web utilizando los diferentes estándares que se ven en la teoría. Por otro lado, desarrollan un trabajo práctico en el cual utilizan técnicas de *hacking* para detectar vulnerabilidades de sitios webs. Este trabajo les permite a los alumnos conocer cuáles son los puntos débiles de un sistema que se convierten en amenazas de mayor riesgo de ocurrencia debido a que ya existen técnicas y herramientas de uso libre en internet que las detectan y explotan. Así entonces, estos casos tienen una mayor prioridad en ser solventados lo antes posible en un sistema real.

Este artículo presenta en la Sección 2, el diseño de la asignatura. En la Sección 3, se expone la metodología y en la 4 se presentan los resultados obtenidos. En la Sección 5 se comentan otros enfoques de centros donde se estudia Seguridad orientada a servicios y por último, en la Sección 6, se presentan las conclusiones.

## 2 La asignatura de Servicios y Seguridad

### 2.1 Contexto

Seguridad y Servicios es una asignatura optativa de 6 ECTS de cuarto año del grado en Informática y Servicios en la Escuela Universitaria de Informática Tomás Cerdá. Es obligatoria para obtener la mención de "Gestión de Servicios".

La asignatura pretende que se comprendan los aspectos a tener en cuenta para disponer de un sistema de servicios seguro. Se revisan los elementos de seguridad de identificación y autentificación, autorización, integridad, confidencialidad y privacidad, desarrollando las principales recomendaciones y técnicas actuales. Se basa en las soluciones estándares indicadas por las instituciones más relevantes en este ámbito como el *National Institute of Standards and Technology* (NIST) [3], *World Wide Web Consortium* (W3C) [5], *Advanced Open Standard for the Information Society* (OASIS) [6]. El alumno trabaja los principales conceptos de seguridad de servicios web fundamentales para gestionar, diseñar y desarrollar sistemas orientados a servicios, que por su naturaleza abierta a la red, están expuestos a amenazas y a vulnerabilidades, y requieren tener un nivel de seguridad adecuado. Se enseña la metodología para elaborar un modelo de amenazas *Threat Modeling* [4,5] para detectar, documentar vulnerabilidades y realizar una evaluación de riesgos.

### 2.2 Competencias

Del conjunto de competencias asociadas al grado de Informática y Servicios, la asignatura Servicios y Seguridad cubre las siguientes competencias específicas:

1. Evaluar sistemas hardware/software en función de un criterio de calidad determinado.
2. Determinar las soluciones tecnológicas adecuadas para la seguridad informática.
3. Analizar, identificar y definir los requisitos que se han de tener en cuenta para elaborar un plan de seguridad.
4. Aplicar las diferentes normativas sobre seguridad, protección de datos, etc., en todas aquellas situaciones que lo precisen.

En relación a las competencias transversales asociadas al grado, en la asignatura se trabajan la de gestionar (planificar) el tiempo y los recursos disponibles, y el desarrollo de estrategias de aprendizaje autónomo.

### 2.3 Objetivos y Temario

A continuación se describen los objetivos de aprendizaje de la asignatura, a partir de los cuales se desarrollan las competencias, se evalúan los resultados de aprendizaje y se elabora el temario.

**Tabla 2 Objetivos por bloque temático**

| Bloque | Objetivos |
|---|---|
| **Objetivos de la seguridad** | Conocer los objetivos de la seguridad. Dimensiones. Tipos de ataques por cada capa de red OSI. |
| **Estrategias de seguridad** | Aprender los modelos de estrategia de ataque y de defensa que se utilizan actualmente para comprometer los sistemas informáticos. |
| **Criptografía simétrica y asimétrica. Malware. Estándares de seguridad.** | Reconocer los tipos de *malware*, su potencial de daño progresivo y masivo y técnicas de detección y de evitar sus daños cada vez en forma más proactiva.<br>Conocer los algoritmos de flujo y bloque actuales de la Criptografía Simétrica. Ventajas y limitaciones. Criptografía e Infraestructura de clave pública. Protocolos y estándares. Ventajas y limitaciones.<br>Aprender los principales estándares de seguridad de servicios web: SSL/TLS, XML-Security, WS-Security, etc. Modelos de funcionamiento. Herramientas que implementan estándares. |
| **Metodologías de gestión de riesgos** | Aprender metodologías de gestión de riesgos. Identificar vulnerabilidades y valorar amenazas. Realizar un modelo de arquitectura de seguridad. |

El temario se desarrolla a partir de los objetivos de cada uno de los bloques indicados en la **Tabla 2** Objetivos por bloque temático en forma secuencial. Este programa está diseñado para ser desarrollado en un semestre de 6 ECTS, que corresponden a 150 horas de trabajo del alumno. La mitad son destinadas a trabajo autónomo. El trabajo dirigido y supervisado, 64 horas, se lleva a cabo en un total de 32 sesiones de 2 horas/sesión, que se realizan a lo largo de 16 semanas.

### 2.4 Enfoque basado en estrategias de ataque y defensa

Con el propósito de realizar una formación conceptual y comprender las técnicas de seguridad, utilizamos un enfoque basado en estrategias de defensa y ataque [2]. De este modo el alumno aprende a diferenciar y a valorar el uso de las técnicas de seguridad conociendo qué aporta cada una en el cuadro global de la seguridad del sistema. Los sistemas de defensa suelen aplicar en forma sistemática los controles disponibles en cada una de las capas, dejando así activos supervisados más de una vez mientras que otros no tienen control. Una visión estratégica global facilita que se apliquen controles en forma metódica para conocer qué aspectos están cubiertos y cuáles no, logrando un sistema más fuerte y posiblemente con menores costes de control.

Si bien los ataques se pueden producir en diferentes niveles y con múltiples técnicas y pudiendo ser internos o externos a la organización, siguen uno de los siguientes modelos representados en la Figura 1 - Estrategias de ataque.

**a) Directo:** El atacante atenta contra su objetivo sin utilizar servidores intermedios excepto los normales de encaminamiento. Son ejemplos de este tipo de ataque el correo no deseado, el envenenamiento de caché del DNS, o el encaminamiento con reglas de filtrado (*blackholing*).

**b) Progresivo:** El atacante usa una serie de servidores intermedios entre el punto de lanzamiento y su objetivo. Varios ataques siguen esta estrategia ya sea contratando servidores intermedios en el mercado negro o por el compromiso manual o automático de uno de éstos utilizando técnicas de *malware*.

**c) Masivo:** el atacante compromete un grupo de servidores y utiliza todos ellos para actuar contra el servidor objetivo. Este modelo se utiliza para ataques del tipo *Distributed Denial of Service* (*DDOS*) o para captura de tráfico del servidor de destino.

**d) Dirección errónea:** Se distrae a las técnicas de defensa. Puede ser un ataque enmascarado utilizando direcciones falsas o bien que se pretenda un ataque a otros servidores para distraer a la defensa. De esta manera los atacantes tienen más posibilidades de que el verdadero ataque no se detecte.

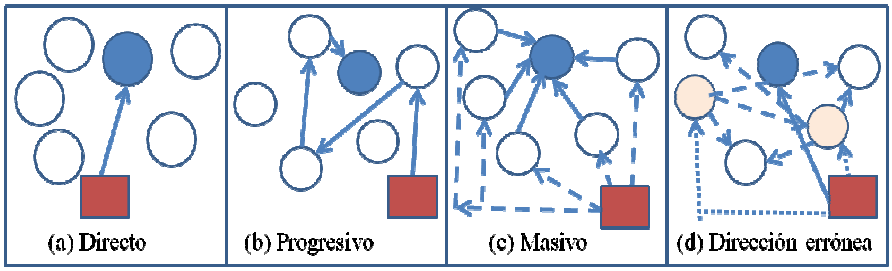


**Figura 1 - Estrategias de ataque –** Cuadro Rojo: Origen – Círculo Azul: Destino.

Las estrategias de defensa se aplican en diferentes capas de la red en forma de controles, de modo que si uno falla, pueda funcionar otro. Se basan en el arte de la guerra y se clasifican según sea el nivel de intromisión del atacante. En primer lugar se intenta evitar una incidencia inicial obstaculizando la identificación de los activos

valiosos. Luego, se trata de dificultar que el ataque avance y finalmente se intenta reconocer al adversario y recuperar el sistema. Están representadas en la Figura 2.

**a) Decepción:** Se trata que el atacante actúe contra un activo que no interese a la empresa, que no sea productivo para el ataque, que no sea un servidor crítico que tenga información confidencial. Puede utilizarse la técnica de *honeypot* utilizando máquinas virtuales con información falsa para confundir al atacante o un *tarpit* que responda a consultas de red con información incorrecta o incompleta.

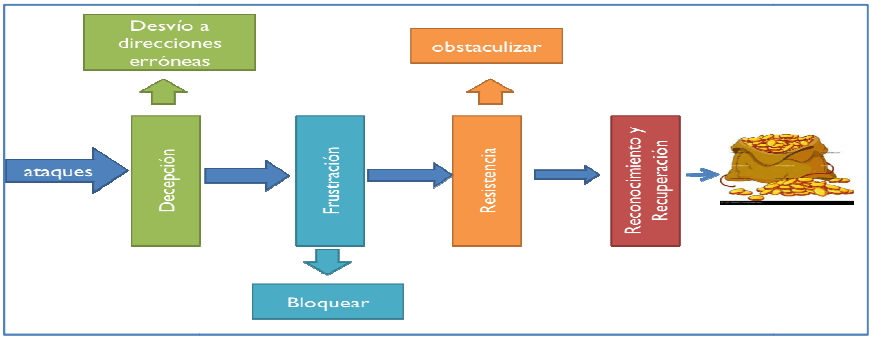


**Figura 2 - Estrategias de defensa**

**b) Frustración:** estas técnicas tienen como objetivo denegar el acceso inicial de un ataque. Se impide el ingreso a cualquier medio que pueda utilizarse. Por ejemplo, las listas de control de acceso en los enrutadores o los firewalls.

**c) Resistencia:** estas técnicas pretenden que un ataque no pueda progresar después de haber entrado al sistema. Limita la propagación de actividad en el ordenador o en la red. Requiere de una actuación más activa de los controles de seguridad, y mayor esfuerzo que las anteriores, aunque muchos de los métodos de la frustración pueden utilizarse, como control de puertos. Son ejemplos de técnicas de resistencia la encriptación o la segmentación de la red en zonas seguras.

**d) Reconocimiento y recuperación:** La idea es identificar al ataque lo antes posible y recuperar el servicio atacado. En primer lugar se ha de identificar el ataque y diagnosticarlo en términos de propósito, medios o impacto. Se pueden aplicar técnicas de comparación de patrones de tráfico o de detección de cambios de configuración. Luego, se han de recuperar los ordenadores y redes afectadas a un estado seguro. Se recomienda que la estrategia de recuperación esté relacionada con el diagnóstico, para asegurar que se guarden los datos necesarios para investigar el ataque y para minimizar los recursos perdidos durante el ataque.

# 3 Metodología docente

## 3.1 Actividades de aprendizaje

Los contenidos son trabajados en sesiones teóricas y sesiones de perfil práctico. Las sesiones teóricas buscan una participación activa del estudiante, y siguen el

temario presentado en la sección anterior. Los conceptos son expuestos en clases magistrales utilizando presentaciones que incluyen las referencias bibliográficas para que el alumno pueda profundizar más a través del trabajo autónomo. También se plantea a los alumnos la realización de actividades (algunas individuales, otras mediante trabajo cooperativo) como responder a preguntas, relacionar temas, lectura y debate de artículos de investigación, a través de las cuales el alumno reflexiona y asimila el tema tratado.

La Tabla 3 – Metodologías por bloque temático y su estrategia relacionada muestra la relación de las diferentes actividades que se realizan para conseguir los objetivos propuestos.

**Tabla 3 – Metodologías por bloque temático y su estrategia relacionada**

| Bloque temático | Metodología | Estrategias relacionadas |
|---|---|---|
| **Objetivos de la seguridad** | Clases teórico/prácticas basadas en la guía de seguridad de servidores web de NIST [3]. | - |
| **Estrategias de seguridad** | Clases teórico/prácticas de Modelos de estrategias de ataque y defensas basado en [2]. | - |
| **Criptografía simétrica y asimétrica.** | Clases teórico/prácticas basadas en el material audiovisual publicado por el proyecto IntyPedia [6] y por el libro [7]. | Defensa: frustración y resistencia |
| **Malware.** | Lectura de artículo [8] y debate grupal para tipos de *malware* e innovaciones en técnicas de detección. | Ataque |
| **Estándares de seguridad.** | Clases teórico/prácticas [4] - Lectura de artículos [9] [10] [11] [12] y debate grupal para profundizar sobre técnicas de ataques y defensa utilizadas actualmente. | Ataque/Defensa |
| **Metodologías de gestión de riesgos** | Clases teórico/practicas basadas en [4] y [5]. | Defensa |

En las sesiones prácticas, se plantean dos tipos de actividades: las sesiones de laboratorio, que se incluyen dentro de las actividades dirigidas, y el desarrollo de un trabajo práctico, una actividad supervisada que el alumno lleva a cabo de forma autónoma.

**Sesiones de laboratorio**

Se asignan cinco clases (10 horas) para realizar una serie de ejercicios de laboratorio en los cuales el alumno configura la seguridad de un servidor de aplicaciones web mediante el uso de SSL, generación y uso de certificados de servidor y de clientes y programación de aplicaciones y servicios web basada en java.

Para esta actividad es conveniente utilizar una plataforma consolidada que minimice las incidencias por errores, y que esté adoptada por un gran número de usuarios para que el alumno aprenda una herramienta usada en el ámbito profesional. Se utiliza una plataforma basada en *Netbeans*, *Glassfish* y servicios web *SOAP* y

*ReST* en Java. La configuración de servicios web en diferentes escenarios de seguridad, haciendo uso de certificados de clave pública en cliente y/o servidor se utiliza el documento publicado por *Oracle* [13].

**Trabajo Práctico**

Durante parte del tiempo asignado al aprendizaje autónomo los alumnos desarrollan un trabajo práctico (estimado en 25 horas) basado en técnicas de *hacking*, El propósito es que aprendan a proteger mejor los sistemas a través de conocer las técnicas que los atacantes utilizan para descubrir y atacar las vulnerabilidades de los sistemas web. Sabiendo qué herramientas están disponibles en internet en forma abierta, el profesional informático puede valorar y explicar mejor los riesgos de compromiso de servicio y de información si no se toman las medidas adecuadas. Como referencia para el desarrollo de este trabajo, utilizamos el libro de Web Hacking [14] entre otros recursos web de uso público que explican cómo utilizar paso a paso herramientas de detección de vulnerabilidades y como perpetuar ataques a distintos niveles: a servidores DNS, al protocolo HTTP, ataques del tipo *Cross-Site Scripting*, inyección SQL a base de datos, etc. Debido a que utilizar técnicas de *hacking* es un delito y por tanto no pueden realizarse sobre un sitio web sin tener autorización, proveemos a los alumnos de sitios web que pueden utilizarse para encontrar vulnerabilidades, de acuerdo con los administradores de dichos recursos.

### 3.2 Actividades de evaluación

Las actividades de evaluación de la asignatura consisten en:
a) Dos exámenes: que suponen un 40% de la nota final (20% cada prueba). El primero se realiza antes de la semana 8 (mitad del semestre), con el objetivo de que los alumnos estudien los conceptos básicos y así poder valorar la comprensión hasta ese momento. Al final del semestre, se realiza un examen de toda la asignatura. A través de estos exámenes se evalúan los resultados de aprendizaje relativos a las cuatro competencias de la asignatura.
b) Trabajo práctico: que supone el 30% de la nota de la asignatura, y a través del cual se evalúan los resultados de aprendizaje de las competencias 1 a 3.
c) Actividades prácticas en clases teóricas y laboratorio: que suponen el 30% de la nota de la asignatura, y a través de las cuales se evalúan los resultados asociados a las competencias 2 a 4.

## 4 Resultados Obtenidos

La presente propuesta se ha utilizado en el primer semestre del último curso académico 2014/2015. Al final del semestre, se realizó una encuesta, para saber la valoración del enfoque basado en estrategia por parte de los alumnos, en la que se preguntó ¿Cómo valoras el enfoque basado en estrategias de ataque y defensa? En una respuesta de 1 a 5, (5 es la máxima valoración), se obtuvo una media de 4.5, es decir, un 90%. En la encuesta participaron todos los alumnos. Sin embargo, los alumnos han indicado que se necesitaban más clases prácticas para llevar a cabo el trabajo de

*hacking* o bien con objetivos más concretos o actividades más acotadas. Sugirieron eliminar algunos temas o que ocuparan menos tiempo como por ejemplo el *malware*. Los trabajos prácticos presentados denotan que los alumnos han indagado mucho sobre herramientas de *hacking*, dando evidencias de que les motiva descubrir vulnerabilidades en sitios en servicio.

## 5 Servicios y Seguridad en otros centros educativos

La seguridad orientada a servicios web se estudia en otros centros nacionales. Presentamos aquí las principales similitudes y diferencias detectadas a partir de la información pública en internet. Cabe destacar que en todos los casos corresponden a asignaturas de 6 créditos y que la principal diferencia detectada es que no disponen de un enfoque basado en estrategias.

La Universidad de Valladolid en el grado de Ingeniería Informática de Servicios y Aplicaciones, imparte "Seguridad Informática" como asignatura obligatoria del 3er. curso[15]. Los contenidos son similares, incluso plantean prácticas de *hacking* para saber cómo defenderse. Sin embargo, no incluyen metodologías de gestión de riesgos.

En la Universidad de Sevilla se dicta "Seguridad en Sistemas Informáticos y en Internet" en el Grado en Ingeniería de Computadores [16], asignatura optativa de 4º curso. Si bien trata sobre protocolos de internet, en el bloque temático no se indica que se profundice en estándares de seguridad para servicios web.

En la Universidad de Zaragoza, en el Grado en Ingeniería de Tecnologías y Servicios de Telecomunicación, la asignatura de "Seguridad en redes y servicios" [17], es obligatoria del 3er. curso. Tiene un bloque temático similar, excepto que no se indica si el plan de seguridad se realiza con técnicas de *Threat Modeling*.

La Universidad Politécnica de Madrid, imparte "Seguridad en Redes y Servicios" en el Grado de Ingeniería Telemática, como obligatoria de 6 créditos en el 3er.curso [18]. En su temario dispone de un bloque que cubre la Legislación española sobre seguridad de la información y recomendaciones y auditorías. En nuestro grado, estos temas se ven en asignaturas como "Sociedad y legislación informática" y "Auditoría y Calidad de Servicios".

## 6 Conclusiones

Se ha presentado la metodología de la asignatura Seguridad y Servicios que se dicta como optativa en el cuarto curso del Grado de Informática y Servicios de la Escuela Universitaria de Informática Tomás Cerdá. La propuesta se basa en incluir un enfoque basado en estrategias de ataques y de defensa que facilite la comprensión de los temas estableciendo una relación entre estos y el correspondiente modelo en que se enmarca dicho tema. Ha tenido una buena aceptación por parte de los alumnos y hemos detectado que se han mostrado más motivados durante las clases. Siguiendo los comentarios de los alumnos, en el próximo curso, reestructuraremos el trabajo de *hacking* dando objetivos más concretos y mayor soporte.

## Referencias


1. Boixader F, Guardiola J, Albert M, Ribas J. El Grado en Informática y Servicios. Una respuesta a la nueva demanda del contexto social. In: JENUI 2010. "XVI Jornadas de Enseñanza Universitaria de la Informática." Universidade de Santiago de Compostela. Escola Técnica Superior d'Enxeñaría; 2010:130–135..
2. Timothy J. Shimeall and Jonathan M. Spring. Introduction to Information Security: A Strategic-Based Approach. In: Elsevier; 2013:382..
3. Scarfone ASTWK. Guide to Secure Web Services Recommendations of the National Institute of Standards and Technology. http://csrc.nist.gov/publications/nistpubs/800-95/SP800-95.pdf.
4. Bertino E, Martino L, Paci F, Squicciarini A, Elisa Bertino, Lorenzo Martino, Federica Paci AS. Security for Web Services and Service-Oriented Architectures. Berlin, Heidelberg: Springer Berlin Heidelberg; 2010. doi:10.1007/978-3-540-87742-4.
5. Suvda Myagmar, Adam J. Lee WY. Threat Modeling as a Basis for Security Requirements. In: Symposium on requirements engineering for information security (SREIS).Vol 2005.; :1–8.
6. Intypedia.: http://www.intypedia.com/.
7. Chwan-Hwa (John) Wu; J. David Irwin. Introduction to Computer Networks and Cybersecurity - CRC Press Book. (Press C, ed.).; 2013:1336..
8. Damshenas M, Ali Dehghantanha RM. A SURVEY ON MALWARE PROPAGATION, ANALYSIS, AND DETECTION. Int J Cyber-Security Digit Forensics. 2013;2(4):10–29..
9. Jensen M, Gruschka N, Herkenhöner R. A survey of attacks on web services. In: Computer Science - Research and Development.Vol 24.; 2009:185–197..
10. Chapman IM, Leblanc SP, Partington A. Taxonomy of cyber attacks and simulation of their effects. In: MMS '11 Proceedings of the 2011 Military Modeling & Simulation Symposium. Society for Computer Simulation International; 2011:73–80.
11. Subashini S, Kavitha V. A survey on security issues in service delivery models of cloud computing. J Netw Comput Appl. 2011;34(1):1–11.
12. Jang-Jaccard J, Nepal S. A survey of emerging threats in cybersecurity. J Comput Syst Sci. 2014;80(5):973–993..
13. The WSIT Tutorial. http://docs.oracle.com/cd/E17802_01/webservices/webservices/reference/tutorials/wsit/doc/.
14. Berta S. Web Hacking. (Manuales USERS - Creative Andina Corp, ed.). RedUsers Usershop; 2013:320.
15. Universidad de Valladolid - Ingeniería Informática de Servicios y Aplicaciones. http://www.uva.es/opencms/consultas/planesestudios/guia?menu=3&codigo_plan=413&codigo_asignatura=40823&grupo=1&ano_academico=1314.
16. Universidad de Sevilla- Ingeniería de Computadores. http://www.us.es/estudios/grados/plan_204/asignatura_2040035#contenidos.
17. Universidad de Zaragoza - Ingeniería de Tecnologías y Servicios de Telecomunicación. http://titulaciones.unizar.es/asignaturas/30353/contexto14.html.
18. Universidad Politécnica de Madrid - Ingeniería Telemática.: https://www.euitt.upm.es/estudios/grado/telematica/fichas-de-asignaturas?codasig=53&curso_academico=2014&semestre=6&titulacion=59TL.

# Retroinformática para la enseñanza y el aprendizaje en la Ingeniería Informática

Xavier Molero Ana Pont, Antonio Robles y Milagros Martínez

Escola Tècnica Superior d'Enginyeria Informàtica
Departament d'Informàtica de Sistemes i Computadors
Universitat Politècnica de València
{xmolero,apont,arobles,mimar}@disca.upv.es

**Resumen** En este trabajo mostramos cómo a través de la retroinformática tratamos de motivar a nuestros estudiantes en el estudio de la asignatura Estructura de Computadores. A ello contribuye el Museo de Informática de la Universitat Politècnica de València que se ha convertido durante los dos últimos cursos en un instrumento didáctico adicional para la docencia, además de cumplir con la importante misión de difundir la cultura e historia de la informática entre los estudiantes de grado de ingeniería informática. La experiencia se ha llevado a cabo durante el pasado y el actual curso con los alumnos de segundo año del mencionado grado y en el ámbito de las tareas académicas de la asignatura. El artículo explica los detalles de la articulación de las actividades en una asignatura con más de cuatrocientos alumnos matriculados y organizados en siete grupos de aula, lo que ya es de por sí un reto nada desdeñable. Parte de estas actividades se han llevado a cabo directamente por parte del profesorado y otras lo han sido de forma autodirigida mediante cuestionarios en papel y herramientas online. Por último, este trabajo muestra la manera en que se han evaluado las experiencias en ambos cursos y las conclusiones extraídas de las mismas para poder mejorarlas en futuras ediciones e incorporar nuevos retos para el futuro.

**Palabras clave:** motivación, retroinformática, materias básicas, museo de informática, informática y sociedad, historia de la informática.

**Abstract.** In this work we show how retrocomputing can be used to motivate and encourage our students to address Computer Organization subject. The Museum of Informatics of the UPV contributes to this purpose and during the last two academic years has become an additional teaching tool for the mentioned subject besides of its important mission for patrimonial and cultural dissemination among students of Computer Engineering Degree. The experience has been carried out during the last and the current academic year in this degree in the scope of the academic tasks related to the subject. This paper explains how the experience was organized and carried out for more than 400 students organized into 7 class groups, which is by itself an important challenge. Some of the related activities were directly performed by the teaching staff and others

were planned in a self-guided way, both using online tools and paper and pencil surveys. Finally, this work show how the experience has been evaluated in both years, the main conclusions got to improve it for new future editions and how to include new challenges.

**Keywords:** Motivation, retrocomputing, core subjects, Museum of Informatics, Informatics and Society, Computer History

## 1 Introducción y motivación

En la enseñanza de materias básicas de las titulaciones de informática, y en especial de aquellas que tienen como objetivo ofrecer una visión sobre los fundamentos de los computadores y los elementos básicos del hardware y software que los componen, es especialmente difícil dar una visión aplicada y cercana a la realidad de sus contenidos. Conocer los principios básicos del funcionamiento de un computador requiere de una serie de simplificaciones y abstracciones que resultan muy lejanas de la realidad tecnológica actual, pero que son necesarias para crear la base de conocimiento que permita, con posterioridad, abordar conceptos y paradigmas actuales, mucho más complejos y sofisticados.

Esta visión elemental de la máquina resulta muchas veces poco atractiva y desmotivadora para nuestros actuales estudiantes, acostumbrados a utilizar, en su día a día, dispositivos y aplicaciones basados en las últimas tecnologías difícilmente relacionables con los contenidos de la materia. Motivar a los estudiantes para que encuentren atractivas estas materias básicas deviene, por tanto, en una tarea difícil.

Por otra parte, la reducción de horas lectivas llevada a cabo en los recientes planes de estudio, así como la desaparición de las asignaturas de libre elección, hace cada vez más difícil dedicar tiempo a aspectos relacionados con la historia de la informática. Si bien estos aspectos no forman parte directamente de las competencias y habilidades específicas que se esperan conseguir de los estudios, sí son, en nuestra opinión, conocimientos básicos a reivindicar para la formación de los ingenieros informáticos.

Conocer la historia y evolución de cualquier ciencia o técnica no solo ayuda a comprender y contextualizar mejor el presente; su estudio proporciona, a pesar de la dificultad para medir objetivamente este beneficio, un bagaje cultural realmente valioso para nuestros egresados toda vez que permite mejorar la sensibilidad y compromiso con la sociedad y el medio ambiente, unos valores de carácter transversal que el diseño de los actuales planes de estudio suelen precisar y tener en cuenta.

Por todo ello, consideramos que conocer la historia de la informática no es únicamente una forma de motivar a nuestros estudiantes sino que, además, contribuye de forma clara a alcanzar las competencias de la titulación [2009]. En este sentido, el Museo de Informática de la UPV nos brinda una oportunidad única para conseguir un doble objetivo en nuestros alumnos:

1. Motivar el estudio y facilitar la comprensión de la materia Estructura y Arquitectura de Computadores.
2. Ampliar su cultura informática a través del conocimiento de su historia y evolución tecnológica.

Por ello, al inicio del curso 2013/14 se planificó la visita al Museo de Informática de la UPV como una actividad asociada a la asignatura Estructura de Computadores. Esta actividad se planteó como una tarea más a realizar dentro de la asignatura y con la misma filosofía que el resto de actividades pedagógicas, esto es, con carácter voluntario pero puntuable. A pesar de los problemas propios asociados a la primera vez que se organiza cualquier actividad, la edición del pasado curso nos permitió ver las posibilidades de la retroinformática [1] como elemento de innovación docente, por lo que en el curso actual 2014/15 la experiencia se ha llevado también a cabo incluyendo nuevas actividades.

En el resto del artículo se indica el contexto de la asignatura, se describe brevemente el Museo de Informática como marco y elemento básico de la actividad, se explica cómo ésta se ha organizado y llevado a cabo en esto dos cursos así como los contenidos de la misma y su conexión con la materia estudiada. Finalmente, se describe cómo se ha valorado el impacto de la experiencia en ambas ediciones, el alcance de los objetivos propuestos y el nivel de satisfacción mostrado por los estudiantes con la misma.

## 2 Estructura de Computadores: la asignatura

La asignatura Estructura de Computadores (EC) del plan de estudios de Grado en Ingeniería Informática que se imparte en la UPV forma parte de la materia de carácter obligatorio denominada Estructura y Arquitectura de Computadores. Esta materia se descompone en dos asignaturas, la mencionada EC, con un peso de 9 ECTS e impartida de manera anual en segundo curso, y la asignatura Arquitectura e Ingeniería de Computadores, de 6 ECTS y ubicada en el primer semestre del tercer curso. La carga anual de la asignatura EC se organiza en 6 ECTS de teoría de aula y problemas y 3 ECTS de laboratorio.

Como base informática previa, los estudiantes han cursado 24 ECTS con carácter de formación básica a través de las asignaturas Fundamentos de Compu-

[1] A lo largo de este artículo el prefijo *retro* se utiliza en castellano como abreviatura de *retrospectiva* (del latín *retrospicere*), adjetivo que, según la Real Academia Española, indica que «se considera en su desarrollo anterior», y no como el prefijo latino *retro-* que significa «hacia atrás». La abreviatura *retro*, en el primer sentido mencionado, es una manera cada vez más utilizada de referir coloquialmente aspectos de un tiempo pasado o que lo evocan. La Wikipedia en inglés (`http://en.wikipedia.org`) se hace eco ampliamente del término *retrocomputing*, que define como *the use of older computer hardware and software in modern times*. Este último es el sentido que le hemos querido dar en este trabajo, en el que nos hemos tomado la licencia de usar el término *retroinformática*, construida léxicamente igual que la palabra *rétro-informatique* usada en francés.

tadores, Introducción a la Informática y a la Programación, Tecnología de Computadores y, finalmente, Programación, todas ellas de 6 ECTS.

La competencia indispensable que se pretende alcanzar con la asignatura Estructura de Computadores se cita textualmente en la guía docente [2012] de la misma como: *capacidad de conocer, comprender y evaluar la estructura y arquitectura de los computadores, así como los componentes básicos que los conforman.*

Para satisfacer la mencionada competencia, esta asignatura comprende el estudio de las distintas unidades funcionales que integran un computador y que hacen posible la ejecución de los programas, prestando especial atención a la íntima relación entre el hardware y el software, así como a la manera en que diferentes estructuras organizativas de las unidades funcionales pueden influir sobre las prestaciones del computador. En este sentido, la asignatura aborda el estudio del procesador, la unidad aritmético-lógica, el sistema de memoria, la unidad de entrada/salida y los dispositivos periféricos. Todo ello se organiza en las siguientes unidades temáticas:

1. **El procesador**
   - Tema 1: La ruta de datos
   - Tema 2: Segmentación básica
2. **Unidad aritmético-lógica**
   - Tema 3: Unidad aritmética de enteros
   - Tema 4: Unidad aritmética de coma flotante
3. **Unidad de memoria**
   - Tema 5: El sistema de memoria
   - Tema 6: Jerarquía de memoria
4. **Unidad de entrada/salida**
   - Tema 7: Adaptadores e interfaces de entrada/salida
   - Tema 8: Sincronización de la entrada/salida
   - Tema 9: Técnicas de transferencia de la entrada/salida
5. **Los periféricos y las estructuras de interconexión**
   - Tema 10: Dispositivos periféricos
   - Tema 11: Estructuras de interconexión

Toda la materia de Estructura y Arquitectura de Computadores impartida en el grado toma el procesador MIPS R2000 como hilo conductor. Ya en la asignatura Fundamentos de Computadores de primer curso se inicia el estudio de este procesador presentando su estructura básica y los rudimentos de su lenguaje ensamblador. De manera similar a la propuesta del libro *Estructura y diseño de computadores. La interfaz hardware/software* de D.A. Patterson y J.L. Hennessy [2011], el procesador MIPS R2000 y su lenguaje ensamblador forman el sustrato sobre el que se explican las unidades funcionales del computador, su interconexión y los parámetros clave de sus prestaciones, tanto en el aula como en el laboratorio. Para una mayor información sobre la asignatura, metodología docente y evaluación, puede consultarse su correspondiente guía docente [2012].

A pesar de esta visión básica y clásica de la materia que nos ocupa, la asignatura recurre a ejemplos de la tecnología actual tanto para ilustrar los temas de

memoria, dispositivos periféricos y elementos de interconexión, como para ofrecer una visión evolutiva de los aspectos clave en el diseño de los procesadores.

En este sentido, estamos plenamente convencidos de que la exposición de la retroinformática que ofrece el Museo de Informática de la UPV puede jugar un papel clave para contextualizar los contenidos de la asignatura porque permite situar en el tiempo los avances de la tecnología, tanto en lo referente al hardware como a la programación y al desarrollo de las aplicaciones [2014]. Así, la visión retrospectiva —y no lo olvidemos, también social— que proporciona el museo acerca de la informática puede aprovecharse como un factor más de motivación para el estudio de los computadores.

## 3 El Museo de Informática

El Museo de Informática[2] de la Universitat Politècnica de València ofrece su colección al visitante como un viaje en el tiempo a través de la historia de la informática [2014]. Inaugurado en el año 2001 y, por tanto, uno de los pioneros en España, ha sido reconocido como museo oficial de su comunidad autónoma por su compromiso con la sociedad y su determinada vocación de servicio público. Por otro lado, el museo ha sido admitido recientemente como miembro de la organización ICOM (*International Council of Museums*[3]) de la UNESCO con el fin de participar activamente en la conservación y protección del patrimonio cultural ligado a la informática.

El proyecto de difusión patrimonial del museo se dirige principalmente a los jóvenes estudiantes de nuestro sistema educativo y también al público en general. Su objetivo primordial es dar a conocer la historia de la informática (orígenes, evolución tecnológica) y, dado que casi todos nosotros somos usuarios asiduos de dispositivos o aplicaciones de carácter informático (internet, redes sociales, ...), el museo pretende incitar a la reflexión crítica sobre otros aspectos menos conocidos, pero de gran importancia para nuestra sociedad, que no dejan de tener relación con la informática (basura electrónica y reciclaje, privacidad de la información, derechos humanos, ...). En el seno del museo se organiza un amplio abanico de actividades pedagógicas y culturales, entre las que destacan las visitas guiadas, los talleres de trabajo (y de juegos) con ordenadores antiguos, los ciclos de cine y los cursos de *retroprogramación* (por ejemplo, programación en lenguaje BASIC en microcomputadores de 8 bits).

La exposición permanente del museo se organiza en un conjunto de vitrinas y paneles informativos, así como dispositivos de gran tamaño, dispuestos a lo largo de las tres plantas que conforman uno de los edificios en los que se ubican las instalaciones de la Escola Tècnica Superior d'Enginyeria Informàtica. En particular, esta exposición contiene una muestra representativa de la informática de las tres últimas décadas del pasado siglo XX. El criterio expositivo utilizado es el cronológico, aunque sometido a ligeras modificaciones a causa de las restricciones impuestas por el ámbito expositivo disponible. Por otro lado, y ante

[2] `http://museo.inf.upv.es`
[3] `http://icom.museum`

la necesidad de materialidad que una exposición inevitablemente establece, los objetos (tanto ordenadores como otros dispositivos) se acompañan, siempre que ha sido posible, por los programas que usaban, materializados bien mediante las unidades de cinta o disco donde están grabados, bien mediante publicidad o listados de código que sirven de ejemplo. El contexto de cada objeto ha sido rnriquecido, además, con fotografías de personajes, manuales u otros tipos de documentos que ayudan a situar su ámbito de aplicación.

De la informática de finales de los años 70, dominada por las grandes instalaciones corporativas (*mainframes*), en la exposición se puede contemplar, entre otras, una perforadora de tarjetas, memorias de ferrita, una impresora de líneas y varias unidades de cinta y disco magnéticos de grandes dimensiones. El grueso de la exposición está determinado por la informática doméstica que se gestó en los años 80, momento histórico en el que convivieron un gran número de microcomputadores y aplicaciones de 8 bits. Se pueden ver ejemplares fabricados por empresas como Sinclair, Amstrad, Commodore o Atari; no podían faltar, en esta lista, los ordenadores MSX, el primer IBM PC o el primer Apple Macintosh. De esta época el museo también cuenta con varios minicomputadores, como el DEC PDP-11 y el Nixdorf Quattro/30, que marcan el advenimiento de sistemas informáticos de tamaño medio y precios asequibles capaces de ejecutar aplicaciones antes reservadas de manera exclusiva a los grandes computadores.

Los años 90 están representados por los dos tipos de computadores personales que se adueñaron finalmente de la casi totalidad del mercado: los compatibles con el IBM PC y los computadores fabricados por Apple, utilizados ambos a través de sistemas gráficos basados en ventanas (GUI, *Graphical User Interface*). Por otro lado, también tienen un papel destacado en el museo las estaciones de trabajo (*workstations*) aparecidas a finales de la década de los 80 y destinadas principalmente a aplicaciones científicas y técnicas con una gran componente gráfica.

Así mismo, también es posible contemplar la evolución del tamaño de los computadores personales a través de varios ejemplares de portátiles (o transportables, como fueron conocidos en un primer momento). El abanico de artefactos abarca, por otra parte, un conjunto de expositores con un amplio surtido de microprocesadores, módulos de memoria semiconductora, dispositivos de almacenamiento (tarjetas perforadas, discos, cintas) y de interconexión, así como una vitrina especial dedicada a los videojuegos, una industria muy ligada al ámbito informático que comenzó a tener importancia a finales de los años 70.

Finalmente, la exposición permanente del museo incorpora un conjunto de paneles informativos diseñados con un enfoque pedagógico que tratan diversos temas alrededor del mundo informático: la propia historia de la informática, los lenguajes de programación, el papel de la mujer en el desarrollo de la informática, algunas curiosidades sobre términos técnicos y dispositivos informáticos, la evolución de los dispositivos de almacenamiento que han ido incorporando los computadores, el concepto de mochila ecológica, los residuos electrónicos y su gestión, los materiales tóxicos presentes en los dispositivos electrónicos y el mundo de la publicidad en la informática.

## 4 Diseño de la experiencia

La visita al Museo de Informática se planificó a principio de curso para así motivar al alumno, ya desde el inicio, en el estudio de la asignatura Estructura de Computadores y hacer que fuese observada desde un punto de vista distinto al experimentado en el aula y el laboratorio. Los conocimientos acerca de la evolución de los computadores a lo largo del tiempo, tanto en algunos aspectos y tan poco en otros, facilitan al alumno la percepción de los aspectos esenciales y básicos, de la importancia de las bases que se les transmiten en la asignatura y, principalmente, del motivo por el cual estos conocimientos son importantes. ¿Cuál es el común denominador de todo lo que se ha mostrado? ¿Cómo se harán las cosas en el futuro? El intento mismo de elaborar una respuesta a estas cuestiones debería permitir el establecimiento de las bases para una nueva percepción de la asignatura.

La visita al museo se organizó en dos partes o actividades claramente diferenciadas. La primera consistió en una breve conferencia que tuvo lugar en el salón de actos, en la que se mostró a los estudiantes no solo una perspectiva histórica de la informática —desde los primeros artefactos de cálculo hasta los computadores contemporáneos—, sino que además tuvo el objetivo de despertar su sensibilidad hacia aspectos sociológicos y medioambientales implicados. La charla se complementó con la proyección de vídeos donde se mostraban los principios del desarrollo de los computadores y su programación, anécdotas de la época, anuncios publicitarios y otros elementos visuales que hacían más amena la exposición. Esta primera parte tuvo una duración aproximada de una hora.

La segunda parte fue la visita propiamente dicha a los fondos que forman parte de la exposición permanente del museo y que, como se ha indicado, se encuentran expuestos en vitrinas y paneles informativos repartidos en las tres plantas de un mismo edificio. El alumno pudo efectuar el recorrido de la visita, deteniéndose para leer y asimilar los contenidos, en más o menos otra hora.

Teniendo en cuenta estos aspectos, en los próximos apartados describimos la manera en que organizamos la experiencia durante el pasado curso 2013/14 y el actual 2014/15.

### 4.1 Curso académico 2013/14

En este curso la actividad tuvo carácter voluntario, de modo que el resultado de los cuestionarios no contribuyó en la nota final de la asignatura, aunque para incentivar la asistencia se les indicó que podría tenerse en cuenta en los casos extremos para redondear la nota al alza.

Para poder llevar a cabo tanto las experiencias de este tipo como los actos de evaluación, nuestro centro deja los lunes fuera de la planificación de horarios de los dos primeros cursos, por lo que en este caso se tiene asegurado el día adecuado en todos los grupos para llevar a cabo la actividad. Desde el punto de vista organizativo, el principal problema fue gestionar una visita para un elevado número de alumnos. Durante este curso la asignatura tuvo un total

de 423 alumnos matriculados, organizados en 7 grupos de aula, todos ellos en horario de mañana excepto uno impartido por la tarde.

Para disponer de una previsión del número de alumnos interesados en participar en la actividad, previamente se pasaron unas listas en el aula para que se apuntaran y, de este modo, poder planificar el número de sesiones a realizar. En este curso, un total de 269 alumnos mostraron su interés por asistir al evento (el 64 % de los matriculados). Dado que el aforo máximo del salón de actos es de 150 personas, se planificaron dos turnos. Para hacer un reparto sencillo se organizaron por grupos, cuatro en el primer turno y tres en el segundo, equilibrando así el número de asistentes. Las Figuras 1 y 2 recogen imágenes de esta parte.


**Figura 1.** Presentación e introducción de la actividad.

Siguiendo la filosofía general de la metodología aplicada en esta asignatura, no hay actividades de obligado cumplimiento, sino que se deja al libre albedrío del estudiante la realización o no de las mismas, eso sí, siendo conscientes de que todas son importantes para su aprendizaje e incluso se tienen en cuenta para su evaluación. A pesar de ello, o quizá por ello, a priori se esperaba un elevado número de alumnos *interesados* en realizar la visita, por lo que, si bien no había más problema que el aforo del salón de actos para la primera parte, ya se preveía imposible la realización de visitas guiadas al uso, razón por la cual fue necesario convertir esta segunda parte en visitas *autoguiadas*.

Así pues, cada uno de los turnos establecidos estaba compuesto por más de cien alumnos. Como hemos indicado, aunque esto no representaba ningún problema para la parte prevista en el salón de actos, sí hacía inviable una visita

**Figura 2.** Alumnos asistentes a la actividad.

típica de grupo con guía. Como solución se propuso la visita al museo, de forma individual o por parejas, guiándola mediante un cuestionario que los alumnos iban resolviendo a su ritmo. Esto era totalmente viable porque todas las vitrinas del museo están documentadas mediante cartelas y, además, los paneles informativos complementarios son íntegramente autocontenidos. Los cuestionarios confeccionados trataban aquellos aspectos más interesantes y/o anecdóticos de los fondos de la exposición del museo. Con estos cuestionarios se pretendía mantener el interés en la exposición y, al mismo tiempo, guiar al alumno para que realizase una visita lo más completa posible. El cuestionario se entregó a la entrada de los alumnos al salón de actos, los contestaron durante la visita y, finalmente, los entregaron al profesor al término de la misma para su posterior evaluación. Las Figuras 3 y 4 ilustran sendos momentos de esta visita autoguiada en las que se puede apreciar cómo los alumnos interactúan con las vitrinas y los panales didácticos del museo.

El diseño de los cuestionarios requirió de una cuidada elaboración por nuestra parte, ya que además de cumplir con los objetivos principales de la actividad, pretendíamos evitar que solo un pequeño grupo de alumnos realizase el trabajo y luego comunicase las respuestas al resto. Por ello se diseñaron cuatro modelos de cuestionarios distintos. Previamente se elaboró un muestrario con un total de 100 preguntas relativas a los vídeos de la exposición y a las vitrinas de cada uno de los pisos del museo, así como de todos los paneles informativos: panel de lenguajes, paneles sobre el papel de la mujer en la informática, cuestiones de gestión de residuos y otros aspectos medioambientales y curiosidades. Con todas

**Figura 3.** Visita: detalle de una de las vitrinas del museo.

**Figura 4.** Visita: detalle de los paneles informativos.

las cuestiones se confeccionaron finalmente cuatro cuestionarios diferentes, con un total de 22 preguntas de respuesta corta cada uno de ellos. Casi todas ellas se podían responder buscando la información en las cartelas y paneles, por tratarse de nombres, fechas, etc. También, en algunos casos, era preciso realizar pequeños cálculos, como por ejemplo, para comparar la capacidad de un disco duro actual con el de uno de los años 80.

A continuación se muestran las cuestiones que integran uno de los cuatro cuestionarios que se facilitaron a los alumnos para responder durante el recorrido a lo largo de la exposición permanente del museo:

- ¿Qué numeración empleaba la máquina calculadora que Blaise Pascal diseñó en el año 1642? ¿Qué operaciones aritméticas podía llevar a cabo?
- ¿Con qué tipo de tecnología diseñó Charles Babbage su Máquina Analítica? ¿Mediante qué mecanismo se introducían los programas en este artefacto?
- ¿Con qué tecnología construyó Konrad Zuse su máquina Z1 en 1938?
- ¿Quién diseño en 1945 los componentes básicos de cualquier ordenador programable?
- El miniordenador DEC PDP-11 tiene en su arquitectura un rasgo innovador respecto a los ordenadores de su época. ¿De qué innovación se trata?
- ¿Cuántos kg pesan dos unidades HP 9121D de almacenamiento basada en discos flexibles? ¿Qué tipo de formato físico tiene este dispositivo?
- ¿Cómo se llama el procesador de textos que formaba parte de los programas distribuidos con el ordenador Amstrad PCW-8256?
- ¿Quién diseñó el ordenador Apple II y en qué año lo hizo?

- ¿Qué quiere decir el término *direct* en el acrónimo DASD (*direct acces storage device*) que utiliza IBM para referirse a los discos duros magnéticos?
- ¿Cuántos platos tiene el dispositivo IBM 10SR (DASD, *direct acces storage device*)? ¿De qué tamaño son?
- ¿Para qué se usó fundamentalmente el ordenador Tatung Einstein TC-01?
- ¿Qué interfaz tienen en común los sistemas de almacenamiento de las estaciones de trabajo HP Apollo Series 700 y SGI Indigo$^2$ IMPACT?
- ¿Qué quiere decir que el IBM PC tuviera una arquitectura abierta?
- ¿En qué procesador se basó el estándar MSX?
- ¿Qué sustancias químicas se utilizan para crear la capa magnetizable de los dispositivos de almacenamiento? Indique al menos dos.
- La impresora IBM 5211, ¿por qué se denomina impresora de líneas? ¿En qué año fue introducida en el mercado?
- ¿Cuál fue el primer computador construido según el principio de programa almacenado totalmente operativo? ¿En qué año se construyó?
- ¿Por qué organización y para qué tipo de entornos fue desarrollado el lenguaje de programación ADA?
- ¿Cómo se llama la mujer que promovió la fundación Mozilla y software libre?
- ¿De qué son acrónimo las siglas SPAM? ¿Quiénes fueron los responsables de su introducción en el ámbito de la informática?
- ¿Cuántos kg de sustancias químicas hacen falta para fabricar un ordenador de tamaño medio?
- El modelo *Portable Personal Computer* fue el segundo transportable de la empresa IBM. ¿Qué conocido personaje del cine se usó para publicitarlo? ¿Cuánto pesaba?

A partir del desarrollo de esta experiencia resulta interesante, al llegar a determinados puntos del temario, relacionar los contenidos explicados con lo visto y aprendido en el museo. Por ejemplo, podemos plantear la evolución de los procesadores desde los simples modelos pedagógicos hasta las tendencias actuales con una visión histórica y contextualizada. Así mismo, las características de las arquitecturas CISC y RISC son mucho más comprensibles con la perspectiva de los años, el papel del lenguaje ensamblador en el desarrollo de las arquitecturas es mucho más cercano, la evolución de las necesidades de cálculo científico y su relación con las aplicaciones gráficas y de realidad virtual actuales permite justificar mejor el esfuerzo realizado en el diseño de algoritmos y circuitos aritméticos y, por supuesto, es una ayuda inestimable a la hora de presentar los dispositivos periféricos y de almacenamiento. También, las constantes referencias a la evolución histórica de la informática creemos que ayudan a una mejor valoración de esta ingeniería por parte de nuestros estudiantes y a comprender los esfuerzos humanos y tecnológicos que, a lo largo de su corta vida, se han realizado y se siguen haciendo para convertirla en un elemento indispensable hoy en día.

### 4.2 Curso académico 2014/15

En el curso actual la visita al museo también fue planteada como actividad voluntaria, pero en esta ocasión se decidió que la calificación obtenida en ella

fuera reflejada en la nota final del curso. Obviamente, a fin de no entrar en conflicto con la naturaleza voluntaria de la actividad, se propuso incrementar la nota final en 0.2 puntos sobre 10 y, consecuentemente, saturando en 10 la nota final. De esta manera, si un alumno no hubiera realizado la actividad, podría obtener igualmente la máxima nota participando en el resto de actos de evaluación de la asignatura.

En el presente curso nos propusimos mejorar el esquema organizativo atendiendo a la opinión expresadada por los alumnos del curso anterior, de modo que también se hicieron dos grupos para la visita, pero en este caso uno de mañana y otro de tarde. Por las mismas limitaciones de aforo —capacidad máxima del salón de actos— ambos grupos fueron dimensionados con 160 alumnos. El registro del alumnado se realizó a través de las facilidades proporcionadas por la plataforma digital PoliformaT para gestionar las asignaturas de la universidad. La visita fue realizada de un modo similar al curso anterior: una exposición introductoria y la visualización de varios vídeos divulgativos, seguida de la visita autoguiada por el cuestionario a rellenar. Teniendo presente que la nota obtenida en el cuestionario afectaría a la nota final del curso, se realizó un test online, con varias preguntas relativas a los contenidos del museo. Este test se realizó dos días después de la visita. Finalmente la nota de la actividad se obtuvo tomando la del test como punto de partida y redondeándola con la del cuestionario entregado el día de la visita.

Para poder preparar adecuadamente el test online se puso a disposición de los alumnos, a través de la página web del museo, una aplicación que planteaba baterías de diez preguntas sobre los contenidos de la exposición permanente y similares a las del test final[4]. Obviamente, el alumnos podía repetir este test las veces que considerase necesarias hasta tener la confianza suficiente para enfrentarse a la prueba evaluable. Adicionalmente se pasó una encuesta de satisfacción de la actividad en su conjunto. Esta encuesta se realizó al día siguiente. Para obtener nota en la actividad se estableció como necesario realizarla. Los alumnos fueron convenientemente avisados de estos requisitos, por correo electrónico y en la documentación aportada en la guía.

Con el propósito de motivar mejor a los estudiantes, durante la presentación oral se hizo énfasis en la importante tarea que tienen los ingenieros informáticos, y científicos en general, de contribuir al desarrollo de los computadores, haciendo hincapié en los avances tecnológicos que han provocado grandes cambios sociales durante las tres últimas décadas. Además se intentó relacionar estos aspectos con los principales contenidos de la charla, para demostrarles la utilidad de los conceptos que estudian. También se resaltó el papel, en muchas ocasiones decididamente relevante, que la mujer ha tenido en el ámbito de la infomática con algunos ejemplos fácilmente identificables, como el caso de los lenguajes de programación. Finalmente, se dio relevancia a la importancia del uso responsable de la tecnología y la conveniencia de reciclar por razones de salud y sostenibilidad medioambiental.

[4] `http://museo.inf.upv.es/telosabes`

## 5 Evaluación de la experiencia

A continuación se analizan los resultados de la experiencia, haciendo énfasis en las principales diferencias respecto al curso anterior.

En el presente curso 2014/15, el número de estudiantes de la asignatura asciende a 418. Como se ha mencionado, la actividad contó con una oferta de un total de 320 plazas distribuidas en dos grupos de 160 plazas cada uno: uno de mañana y otro de tarde. Finalmente, solamente 260 estudiantes asistieron a la visita, lo que representa el 81 % de las plazas ofertadas. En el grupo de mañana se completaron todas las plazas, mientras que en el de tarde se cubrieron únicamente 100, esto es, el 63 % de las plazas. Ello pone de manifiesto que pocos estudiantes están realmente interesados en la oferta de grupos de tarde aunque, en cambio, suelen ser bastante activos a la hora de reclamarlos. En general, se observa que poco más del 62 % del número total de estudiantes mostraron interés por la actividad, lo cual desanima bastante a los responsables de organizarla, dado el enorme trabajo que conlleva la organización y las dificultades que entraña. Sin embargo, todos los asistentes entregaron el cuestionario utilizado para la visita autoguiada al término de la misma.

Respecto al test online concebido para evaluar el grado de aprovechamiento de la actividad, solo 197 estudiantes (el 76 %) lo completó. Entre estos, 35 estudiantes consiguieron un excelente (notas entre 9 y 10 puntos), 125 consiguieron notas entre 7 y 8 puntos, 29 consiguieron notas entre 5 y 6 puntos, mientras que solo 8 obtuvieron menos de 5 puntos. En el cuestionario escrito se obtuvieron resultados similares a los anteriores, pero con un menor número de excelentes, lo cual se justifica como consecuencia de un menor nivel de concentración, ello a pesar de que ninguna nota fue inferior a 5, lo cual podría deberse al hecho de que el cuestionario se realizó en parejas.

Con el propósito de conocer la opinión de los estudiantes y compararla con los resultados obtenidos el pasado año, de modo que no permitiese identificar las fortalezas y debilidades de la experiencia, se llevó a cabo una encuesta de satisfacción de los estudiantes de forma anónima y a través de la plataforma educativa PoliformaT. La encuesta de realizó a lo largo de la misma semana de la visita al museo, de forma que los estudiantes pudieran albergar un recuerdo cercano de la experiencia. Lograr que los estudiantes respondan a cualquier tipo de encuesta es siempre complicado, especialmente si es online. Al objeto de asegurar un número suficiente de respuestas tal que nos permitiese extraer conclusiones válidas, la cumplimentación de la encuesta se impuso como condición para validar la participación de cada estudiante en la experiencia. A pesar de ello, únicamente 121 estudiantes, de los 260 que asistieron a la visita (sobre un 45 %), completaron la encuesta de satisfacción, lo cual significa que solo esos estudiantes podrán ver incrementada su nota final al cumplir todos los requisitos establecidos. Ello constituye un hecho bastante decepcionante, pues demuestra el escaso interés que los estudiantes prestan a cualquier actividad académica y lo escasamente motivados que están en general en sus estudios.

En el diseño de la encuesta se siguieron las recomendaciones acerca de encuestas de satisfacción presentadas en [2006]. En este sentido, tratamos de evaluar

**Cuadro 1.** Cuestiones tratadas en la encuesta de evaluación.

| | |
|---|---|
| | Dimensión 1: *Actividades previas a la visita* |
| Q1 | La charla previa a la visita al Museo de Informática me ha servido para descubrir aspectos sobre la historia de la informática que ignoraba |
| Q2 | El documental sobre historia de la Informática me ha resultado atractivo y divulgativo |
| Q3 | Los vídeos publicitarios me han servido para situar en su contexto las piezas del museo |
| | Dimensión 2: *Visita al museo* |
| Q4 | Las piezas y material que alberga el museo me han parecido interesantes |
| Q5 | Algunos de los componentes, dispositivos y computadores que alberga el museo han despertado en mí la curiosidad |
| Q6 | La visita me ha servido para comparar los usos de la informática actual con el pasado |
| | Dimensión 3: *Concienciación medioambiental y de género* |
| Q7 | Me parece bien que el museo se preocupe por los aspectos medioambientales relacionados con la informática |
| Q8 | Creo que es necesario dar a conocer el papel que la mujer ha tenido y tiene en el desarrollo de la tecnología informática |
| | Dimensión 4: *Organización de la actividad* |
| Q9 | La forma en que se ha organizado la visita al museo me ha parecido adecuada |
| Q10 | El cuestionario que se ha cumplimentado al término de la visita me ha servido para apreciar detalles que, de otra forma, me habrían pasado desapercibidos |
| | Dimensión 5: *Valoración global* |
| Q11 | Recomendaría a otras personas esta experiencia o actividad educativa |
| Q12 | Considero que la visita al Museo de Informática me ha servido de motivación para el estudio de la asignatura de Estructura de Computadores |

cinco aspectos clave o dimensiones relativas a la experiencia, a saber, las actividades realizadas en el salón de actos previas a la visita (conferencia, documental y anuncios publicitarios), la visita al museo propiamente dicha, la organización de la experiencia, y la valoración global de la misma. Adicionalmente, este curso se ha creído interesante apreciar el grado de concienciación de los estudiantes respecto a los temas medioambiental y de género [2012], los cuales constituyen aspectos que el museo trata de promover. Con esta finalidad, se diseñó una encuesta con 12 preguntas y 5 opciones de respuesta: totalmente de acuerdo (TDA), más bien de acuerdo (MBA), indiferente (IND), más bien en desacuerdo (MBD), totalmente en desacuerdo (TED). También se incluye la opción de no sabe/no contesta (S/C). El Cuadro 1 muestra las diferentes cuestiones de la encuesta agrupadas por cada una de las dimensiones analizadas. En su diseño se trató de reflejar el objetivo perseguido por cada una de las dimensiones a través de la formulación de cuestiones muy concretas, de modo que al estudiante se le transmitiese de forma meridianamente clara la idea que subyace tras cada uno de los mencionados objetivos.

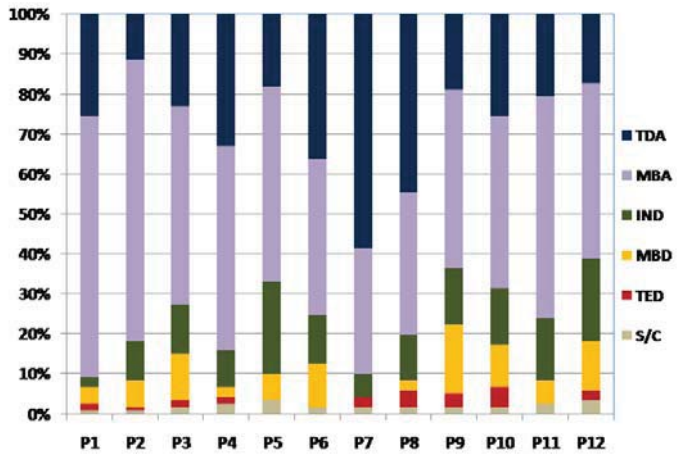


**Figura 5.** Resultado de la encuesta de evaluación.

Los resultados de la encuesta se muestran en la Figura 5. Del análisis de las respuestas se observa un incremento significativo en el porcentaje de estudiantes que reconoce que la visita le ha llevado a estudiar la materia de Estructura de Computadores con más interés (un 62 % en comparación al 50 % del curso anterior). Valoramos este resultado como muy positivo porque muestra el éxito de los esfuerzos llevados a cabo este curso para mejorar la motivación de nuestros estudiantes, lo que constituye el principal objetivo de la experiencia. El hecho de que esta actividad, por su propia naturaleza, no haya servido para motivar

a una parte de estudiantes, no significa que éstos no puedan llegar a motivarse mediante otras actividades también programadas a este efecto a lo largo del curso como, por ejemplo, la visita a un clúster de altas prestaciones. De ahí la importancia de contar con un repertorio variado de actividades capaces de cubrir el máximo de sensibilidades.

También queremos resaltar el hecho de que un mayor porcentaje de alumnos manifestó estar de acuerdo con la organización de la visita (un 65 % respecto al 60 % el curso anterior). Nuevamente, ello muestra el éxito de los cambios introducidos este año en relación con la organización de la visita, fundamentalmente enfocados a subsanar algunas deficiencias detectadas en la primera edición, como por ejemplo la ausencia de grupo de tarde, que causó cierto malestar en un porcentaje significativo de los estudiantes el curso pasado. Ello muestra una vez más las enormes dificultades de organizar cualquier clase de actividad, experiencia de aprendizaje o metodología novedosa de enseñanza cuando se dirige a centenares de alumnos. A menudo, cuando a los profesores de universidad se nos critica por la falta de innovación en nuestra actividad docente, nadie tiene en cuenta las enormes complicaciones que conlleva la organización y gestión de grandes grupos de estudiantes.

El resto de preguntas de la encuesta fueron valoradas, en general, en términos similares al curso pasado. Entre ellas, cabe subrayar el hecho de que más del 90 % de los estudiantes encontraron interesante la conferencia previa a la visita, así como que más del 76 % recomendaría la experiencia, lo cual consideramos como un dato muy positivo. Sin embargo, hay algunos resultados curiosos, tal como el decremento observado en el porcentaje de estudiantes que reconoce que los contenidos del museo había despertado su curiosidad (únicamente un 67 % en comparación al 75 % alcanzado el curso anterior). Este hecho puede explicarse como consecuencia de que algunos estudiantes repitieron la visita por segunda vez, de modo que los contenidos apenas les sorprendieron. En cualquier caso, hemos de admitir que despertar la curiosidad de los estudiantes no es tarea fácil. Por tanto, valoramos de forma muy positiva que los contenidos del museo, al menos, hayan sido atractivos para un 85 % de los estudiantes. Finalmente, cabe subrayar el hecho de que cerca de un 70 % de los estudiantes reconoció la utilidad del cuestionario que tuvieron que cumplimentar al término de la visita.

Respecto a las dos nuevas preguntas incluidas en este curso en la encuesta, relativas al medioambiente y al papel de la mujer en el ámbito informático, más del 90 % de los estudiantes valoraron positivamente la tarea de concienciación medioambiental desarrollada por el museo, mientras que un 80 % valoró positivamente que esta institución constribuyera a subrayar el papel de la mujer en el desarrollo de los computadores.

## 6 La retroinformática como perspectiva de futuro

Del desarrollo y resultados de las actividades anteriores resulta patente la dificultad del profesorado universitario para lograr la motivación de sus alumnos. En particular, dentro del ámbito informático, creemos que los museos pueden

representar, todavía, una gran oportunidad para despertar la curiosidad de una juventud posiblemente saturada de estímulos que, además, les llegan a través de las denominadas *nuevas* tecnologías y que, como sabemos, ya llevan muchos años integradas en nuestra vida cotidiana.

Dentro de la materia de la estructura de computadores, parcela docente que nos atañe en este trabajo, pensamos que la retroinformática puede seguir ayudándonos en nuestro objetivo motivador. En concreto, los primeros ordenadores personales (microordenadores) que inundaron las sociedades occidentales a principios de la década de los 80 del siglo XX, son susceptibles de convertirse en herramientas docentes con un atractivo especial. Estas máquinas de reducido tamaño, basadas en microprocesadores de 8 bits, programables en el lenguaje de programación BASIC, con su aspecto *vintage*, monitores de baja resolución —algunos de ellos monocromos—, teclados curiosos y periféricos de almacenamiento basados en cintas magnéticas y discos flexibles —con formatos y capacidades diversas—, ofrecen la oportunidad de una manipulación bastante alejada de los dispositivos actuales como móviles inteligentes y tabletas y, en consecuencia, *original* a ojos de nuestros jóvenes estudiantes.

Esta pretendida originalidad, buscada interesadamente en el pasado, puede ayudar a situar a los alumnos ante ordenadores en los que, por ejemplo, conceptos sobre representación de la información y aritmética tengan que utilizarse en el ámbito de la programación en BASIC, donde pueden combinarse órdenes sencillas y directas como `PRINT`, formatos de impresión de datos en distintas bases y funciones matemáticas como `FIX` o `ROUND`, por citar solo algunas de las más conocidas. La interacción totalmente directa e inmediata con la memoria principal, del orden de pocos KB en estas máquinas, mediante lecturas y escrituras en determinadas zonas de la misma, también ofrece un entorno ágil y sencillo en el que estudiar el funcionamiento básico de esta unidad del ordenador.

Así mismo, los videojuegos también podrían convertirse en unos buenos aliados en nuestro empeño por motivar y enseñar. Tipologías de videojuegos no faltarán, ya que recordemos que la mayoría de estos microordenadores se utilizaron con este fin. En este sentido, la parsimoniosa carga de programas desde unidades de casete, la edición de código en un entorno limitado o el manejo incipiente de gráficos y sonidos podría resultar útil para ilustrar cómo, con una arquitectura sencilla y recursos muy limitados, los informáticos de la época desarrollaron aplicaciones interesantes y competitivas. A modo de ejemplo, la manipulación de un juego para conseguir, por ejemplo, vidas infinitas, puede ser un buen acicate para insistir en conceptos propios de nuestra materia: hay que saber en qué dirección de memoria está la instrucción máquina que decrementa el contador de vidas disponibles y cómo sobreescribir esta posición (los famosos `POKE` que difundían las revistas) con una instrucción máquina que no haga nada (habría que buscar la equivalente de la instrucción `nop` del MIPS R2000 en el repertorio de instrucciones del microprocesador de que se trate).

En este recorrido que estamos diseñando para el futuro próximo, contamos con la participación del Museo de Informática a través de su denominado *Museo en vivo*, un ámbito didáctico en el que se da cabida a un variado conjunto de mi-

croordenadores de los años 80 del siglo XX todavía operativos. El inconveniente principal con que nos enfrentamos, más allá del planteamiento de la actividad didáctica en sí, es el reducido número de ordenadores disponibles, que está en torno a la decena, por lo que resultará complicado plantearla para un número elevado de alumnos. Una alternativa lógica podría venir de la mano de los emuladores disponibles de este tipo de máquinas, pero ello supondría la pérdida del contacto físico con los ordenadores reales, un valor intangible ofrecer al alumno y al que, al menos de momento, no queremos renunciar. Otra opción, de contenido mucho más ambicioso, vendría de la mano de un rediseño de los contenidos de la asignatura alrededor de la retroinformática y aplicarla en un grupo piloto con un número reducido de alumnos.

## 7 Conclusiones

La retroinformática y la evolución de los computadores nos presenta una magnífica oportunidad para motivar a nuestros alumnos en el estudio de materias básica como es el caso de la estructura de computadores y hacerles más sencillos y cercanos los contenidos. Con este fin, durante estos dos últimos académicos hemos organizado una serie de visitas y actividades alrededor del Museo de Informática de la Universitat Politècnica de València.

Uno de los principales problemas con los que se enfrenta el profesorado universitario en nuestro país a la hora de realizar actividades de innovación es el elevado número de alumnos a los que deben ir dirigidas, al que debemos añadir la escasez de medios humanos y materiales para apoyarlas. En este caso, las actividades desarrolladas han sido planteadas con carácter voluntario para más de 400 alumnos, aunque finalmente fueron alrededor de 260 los que mostraron interés en sumarse a ellas.

La experiencia se ha desarrollado en base a tres actividades básicas. La primera consiste en una charla sobre la historia de la informática, la evolución de los computadores y el software asociado, así como los aspectos sociales y medioambientales directamente relacionados. Todo ello ilustrado con la proyección de documentales cortos y anuncios publicitarios de la época. Esta primera parte se desarrolla en el salón de actos del centro, por lo que su aforo ha marcado el límite de alumnos a los que se puede ofrecer la actividad. Esta restricción nos ha obligado a planificar dos grupos para poder llevar a cabo la experiencia.

La segunda actividad consiste en una visita a la exposición permanente del Museo de Informática. Al ser tan elevado el número de asistentes no se puede plantear una visita guiada al uso, por lo que se ha recurrido a un recorrido autoguiado mediante un cuestionario específicamente diseñado para ello. Este planteamiento requiere una actitud proactiva por parte de los estudiantes y, hemos de reconocerlo, esta actitud no está presente en un elevado porcentaje de ellos, los cuales se muestran bastante apáticos y desinteresados. Esto queda patente en el bajo interés y dedicación con el que contestaron, usando lápiz y papel, las preguntas del cuestionario en ambas ediciones.

Esta apatía se ha puesto más de manifiesto este curso ya que la actividad ha sido más exigente puesto que podía afectar de forma positiva a la nota final de curso. Por ello, en la segunda edición se ha planteado una tercera parte abordable de forma autónoma, en la que debían resolver un test online y que ha servido para obtener la nota de partida de la actividad global. Menos de la mitad de los participantes llegaron a completar este test y bastantes menos aún enviaron la encuesta de satisfacción asociada, a pesar de ser el elemento validador de la nota.

Sin embargo, y desde un punto de vista general, la experiencia ha sido positivamente valorada por un importante porcentaje de alumnos y también por parte del profesorado que se ha implicado en ella. Además, algunos puntos débiles detectados en la primera edición han sido subsanados este curso. Creemos que nos hemos acercado bastante al objetivo principal de la experiencia y hemos conseguido una mayor motivación de nuestros estudiantes mediante la retroinformática y la historia de los computadores. Además, hemos contribuido a difundir la historia de esta ciencia al mismo tiempo que creemos haber sembrado la semilla de la reflexión crítica sobre las consecuencias sociales y medioambientales derivadas del uso masivo de ordenadores y otros dispositivos inteligentes.

Finalmente, para fomentar una participación más activa de nuestros estudiantes, en un futuro próximo, nos planteamos aprovechar otros recursos y actividades promovidas por el Museo de Informática, como los talleres de retroprogramación, manejo y manipulación de videojuegos, y la participación en ciclos temáticos de cine.

## Referencias

[2006] *Guía para la realización de estudios de análisis de la demanda y encuestas de satisfacción*, Ministerio de Administraciones Públicas. `http://www.aeval.es/es/difusion_y_comunicacion/publicaciones/Guias/Guias_Marco_General_Mejora_Calidad/guia_analisis.html` (2006)

[2009] *Memoria para la solicitud de verificación del título de Grado en Ingeniería Informática de la UPV*, `http://gradoinf.webs.upv.es/pdf/Memoria_grado_II.pdf` (2009)

[2011] David A. Patterson, John L. Hennessy. *Estructura y diseño de computadores. La interfaz hardware/software*, Reverté, segunda edición (2011)

[2012] J. Abbate. *Recoding gender: women's changing participation in computing.* MIT Press, Cambridge, MA (2012)

[2012] *Guía docente de la asignatura Estructura de Computadores*, ETS d'Enginyeria Informàtica, 2012. `http://www.upv.es/pls/oalu/sic_gdoc.get_content?P_ASI=11552&P_IDIOMA=c&P_VISTA=&P_TIT=156&P_CACA=2013` (2012)

[2014] Ana Pont Sanjuán, Antonio Robles Martínez, Xavier Molero Prieto, Milagros Martínez Díaz. *The Museum of Computer History - Teaching Support for Computer Organization Subjects* 8th IT STAR Workshop on History of Computing, Hungría (2014)

[2014] Xavier Molero. *El proyecto de difusión patrimonial del Museo de Informática de la UPV: el inicio de una nueva andadura*, Congreso Internacional de Museos Universitarios, Madrid (2014). Pendiente de publicación.

# Una experiencia para fortalecer los procesos de enseñanza de la programación mediante el uso de entornos virtuales de aprendizaje

Aguirre, Jesús Francisco, Viano, Hugo josé, García, Berta
Departamento de Informática - Fac. de Cs. Físico Matemáticas y Naturales
Universidad Nacional de San Luis - San Luis, Argentina
{jaguirre@unsl.edu.ar}{hviano, bertae.garcia}@gmail.com

**Resumen.** Este trabajo describe y presenta los primeros resultados de una experiencia desarrollada en el contexto de la asignatura Programación, de la carrera Ingeniería Electrónica con Orientación en Sistemas Digitales. A partir de un diagnóstico inicial se identificaron algunas problemáticas comunes a los estudiantes tales como: falta de hábitos de estudio, escaso nivel de motivación y dificultad para cumplir con los horarios establecidos. Con el propósito de lograr mayor participación y proveer acceso permanente a los materiales de estudio se creó un aula virtual. Se propusieron actividades colaborativas y se diagramó un práctico de máquina en forma grupal para resolver en un entorno de desarrollo integrado de software libre.
Puede concluirse que los recursos y estrategias utilizadas impactaron positivamente en los procesos de aprendizaje, favorecieron la colaboración y tutorización, proporcionaron información valiosa para el seguimiento de los estudiantes y permitieron aprender más allá de los límites del aula presencial.

**Palabras Clave:** entornos de aprendizaje virtual, software libre, entornos de desarrollo integrado, aprendizaje colaborativo.

**Abstract.** This work describes and presents the first results of an experience conducted in the context of the subject Programming, which is part of the academic program Electronic Engineering with an Orientation to Digital Systems. An initial diagnosis allowed for the identification of some problems typical of students, such as: lack of study habits, poor motivation level, and difficulty to meet scheduled times. A virtual classroom was created with the purpose of achieving greater participation on the part of students and providing permanent access to study materials. Collaborative activities were put forward, and a practical group work was planned to be solved in an integrated development environment of free software. Results suggest that the resources and strategies used had a positive impact on the learning processes, favored collaboration and mentoring, provided useful information for student follow-up, and allowed for learning beyond the limits of a conventional classroom.

**Keywords:** virtual learning environments, free software, integrated development environments, collaborative learning

# 1 Introducción

El impacto generado en nuestra sociedad debido a una evolución constante de la informática, permitió la aparición de nuevas Tecnologías de la Información y las Comunicaciones a las que comúnmente se las denomina TIC. El surgimiento del movimiento de Software Libre (SL) propone un nuevo modelo de trabajo sobre las libertades de los usuarios y la propiedad intelectual del software desarrollado. En la actualidad, ha demostrado tener viabilidad tanto técnica como económica y las estadísticas reflejan un crecimiento sostenido de su uso.

Del análisis realizado por la cátedra sobre los resultados obtenidos en los últimos años de la asignatura "Programación", se identificaron diferentes factores que influyeron en forma negativa en los aprendizajes. A partir de este diagnóstico, se propusieron estrategias para promover el compromiso, la motivación y el interés de los alumnos.

La elección de un Entorno Virtual de Aprendizaje (EVA), y en este caso particular del entorno "Aulas virtuales" implementado sobre la plataforma Moodle [1], responde a la necesidad de incrementar la participación de los alumnos que plantean conflictos para realizar las actividades presenciales. El aula virtual creada para la asignatura aporta un espacio de participación flexible como complemento al aula presencial. [2,3] El uso de un entorno de desarrollo integrado para resolver grupalmente un problema computacional responde a la necesidad de promover aprendizajes significativos a partir de la interacción con otros estudiantes.

Este trabajo relata una experiencia desarrollada en el marco de la enseñanza de la programación a alumnos de segundo año de la carrera de Ingeniería Electrónica con orientación en Sistemas Digitales, usando herramientas de SL y un aula en un EVA. El mismo se organiza de la siguiente manera: se sintetiza el marco teórico y el contexto del trabajo; luego se describen el diagnóstico inicial y las estrategias propuestas; posteriormente se explica la forma de recolección de los datos que permitieron evaluar la propuesta, obtener conclusiones y delinear futuras líneas de trabajo.

# 2 Marco teórico

La aparición de nuevas formas de organización económica, social, política y cultural trae consigo nuevas maneras de trabajar, comunicarnos, aprender, pensar y vivir. A este nuevo orden social se lo llama Sociedad de la Información [4]. Las TIC ofrecen la posibilidad de reconstruir y reinterpretar las posibilidades de enseñar, de acuerdo con el marco socio-educativo-cultural de referencia.

El desafío para la educación superior consiste en encontrar modelos más abiertos y flexibles, donde el estudiante pueda hacerse responsable de la construcción de su propio aprendizaje. Para ello, la enseñanza debe adoptar metodologías más activas. En este sentido, el uso de entornos virtuales amplía las posibilidades de aprendizaje personalizado y seguimiento de los alumnos por parte de los equipos de cátedra.

Diseñar un espacio virtual como apoyo a la clase presencial promueve la construcción de conocimiento manteniendo cierta independencia del espacio y el

tiempo pedagógicos. En estas propuestas el docente asume un rol de guía, centrado en el proceso más que en los resultados, organizando y secuenciando el material didáctico de acuerdo a las características de los estudiantes destinatarios, el contexto en el que se produce el proceso, la organización de la información y la tutorización.

Promover aprendizajes significativos en los estudiantes implica una participación más activa de quienes aprenden y da lugar al aprendizaje colaborativo.

El aprendizaje colaborativo es definido por Johnson [5] como: "el conjunto de métodos de instrucción o entrenamiento para su uso en grupos, así como de estrategias para propiciar el desarrollo de habilidades mixtas: aprendizaje y desarrollo personal y social. El aprendizaje colaborativo virtual también es entendido como un proceso de construcción social de conocimiento. A partir del trabajo conjunto y las metas comunes, se da una "reciprocidad entre un conjunto de individuos que saben diferenciar y contrastar sus puntos de vista de tal manera que llegan a generar un proceso de construcción de conocimiento. Es un proceso en el que cada individuo aprende más de lo que aprendería por sí solo, fruto de la interacción de los integrantes del equipo" [6]. En el desarrollo de un grupo, por tanto, la interacción se convierte en un elemento clave, teniendo en cuenta que es el proceso esencial de reunir las contribuciones de los participantes en la co-creación de conocimiento [7].

Si bien el uso de tecnologías no implica nuevas formas de aprendizaje, la presencia de computadoras, equipos portátiles, teléfonos inteligentes, instrumentos de geolocalización, por nombrar sólo algunos, se ha convertido en parte integral de nuestra vida social, trabajo y aprendizaje. Los dispositivos son cada vez más pequeños, económicos, portátiles e interconectados. Si bien es posible usar las nuevas tecnologías para recrear pedagogías tradicionales, centradas en la transmisión de conocimientos, los contextos de aprendizaje se han modificado constituyéndose en posibilitadores de nuevos aprendizajes. Como expresa Nicolás Burbulles, "las formas radicalmente nuevas en que las personas interactúan con la tecnología en el presente también proporcionan una valiosa oportunidad para que los docentes reformulen su trabajo y su función". [8]

Bill Cope y Mary Kalantzis [9] se refieren al aprendizaje ubicuo y lo definen en forma general como sigue: "el aprendizaje ubicuo representa un nuevo paradigma educativo que en buena parte es posible gracias a los nuevos medios digitales". La relación entre ubicuidad del aprendizaje y tecnologías no es biunívoca, se debe basar en la computación ubicua, pero requiere opciones metodológicas, tales como la colaboración, para hacer posible su aplicación práctica.

En síntesis, el aprendizaje colaborativo y ubicuo permite adquirir competencias para el trabajo en equipo y desarrollar habilidades requeridas en el ámbito profesional y laboral, habilidades deseables para los futuros ingenieros. Enriquecer puntos de vista desde perspectivas diferentes puede contribuir a una formación ciudadana basada en el respeto y la participación constructiva para la solución de problemas sociales de interés común. Citando nuevamente a Cope y Klantzis [9]: "Las tecnologías son el producto de las necesidades sociales. Cuando trabajan para nosotros, sus beneficios suelen ser más revolucionarios que sus especificaciones técnicas".

## 3 Contexto del trabajo

La asignatura "Programación" forma parte del plan de estudios de la carrera Ingeniería Electrónica con orientación en Sistemas Digitales. Dicha asignatura, que se ubica en el segundo año de la Carrera, tiene como objetivo proporcionar al alumno herramientas que le permitan resolver problemas de cálculo numérico. Es la segunda asignatura con contenidos sobre programación que posee dicha carrera. Está estructurada en dos partes: la primera de ellas tiene como objetivo el aprendizaje del sistema operativo GNU/Linux: distribuciones, comandos, consola, interfaz gráfica y aplicaciones. La segunda parte tiene como objetivo profundizar el uso del lenguaje C para la resolución de problemas reales, trabajando con variables punteros, estructuras de datos y archivos. A partir de la formulación de problemas, los alumnos deben pensar soluciones de una forma creativa y expresarlas de una manera precisa, utilizando las herramientas adecuadas y buenas prácticas de programación.

La metodología utilizada por la cátedra consistía en clases teóricas de tipo expositivas, a cargo del profesor responsable, y posteriormente actividades de aula y laboratorio en forma individual, con la asistencia de un jefe de trabajos prácticos y auxiliares de docencia. Todas estas actividades se realizaban en forma presencial. El material de estudio podía descargarse del sitio web de la materia.

A partir del año 2008 el Centro de Informática Educativa (CIE) dependiente del área de Informática Educativa del Departamento de Informática pone a disposición de docentes y alumnos de la Universidad Nacional de San Luis (UNSL) el EVA "Aulas Virtuales" implementado sobre la plataforma Moodle. Con la creación del aula virtual de la materia, en el año 2013, surgen nuevas estrategias metodológicas pensadas para complementar la clase presencial. La siguiente tabla muestra la nueva diagramación de la materia (Tabla 1). Es oportuno señalar que, además de dejar todo el material disponible en el aula virtual, se proponen actividades para resolver en forma colaborativa, por ejemplo foros de preguntas y respuestas y/o foros generales, sobre cuestiones teóricas importantes.

| Conte-nidos | Variables Punteros | Pasaje de parámetros con Punteros | GNU/Linux | | Tipo *struct* | Práctico de Máquina | Entrada y Salida no estándar | | Prog. Orientada a Objetos | Práctico de Máquina | Defensa Prác. de Máq. |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Práctico N° | **1** | **2** | **3** | **4** | **5** | **Parte A** | **6** | **7** | **8** | **Parte B** | |
| Tipo de Actividad | **Individual** | | | | | **Grupal** | **Individual** | | | **Grupal** | |
| Tipo de Aula | Prácticas en aula | Prácticas en Laboratorio de computadoras | | | | | | | | | |

**Tabla 1:** Cronograma de la materia "Programación"

El práctico de máquina debe resolverse en forma grupal, la colaboración se puede llevar adelante tanto en las clases presenciales, como en los foros dispuestos para tal fin. La incorporación en la asignatura de un aula virtual usando este EVA, posibilita un marco educativo más flexible y complementa la enseñanza presencial, permite al equipo de cátedra incorporar estrategias y desarrollar habilidades para el trabajo en grupo y el aprendizaje colaborativo. [10]

## 4 Diagnóstico inicial

Teniendo en cuenta la experiencia recuperada a partir del seguimiento realizado durante 3 años consecutivos de dictado de la asignatura (período 2011-2013), se detectó un alto porcentaje de deserción y un bajo rendimiento académico de los alumnos.

Este seguimiento fue realizado por los docentes de la cátedra considerando resultados de parciales, asistencia a clases prácticas y teóricas, corrección de trabajos prácticos, muestra y resolución de parciales. Indica falta de hábitos de estudio, falencias en la comprensión de textos (puesta en evidencia en la dificultad para la resolución de problemas) e inconvenientes para cumplir con los horarios establecidos.

## 5 Estrategias metodológicas

A partir de la incorporación de herramientas de SL para facilitar el aprendizaje ubicuo fue necesario realizar modificaciones en la metodología utilizada. Se decidió utilizar un entorno integrado de desarrollo (IDE) denominado CODE::BLOCKS [11]. Esta herramienta permitió trabajar en un ambiente con una interfaz común para poder editar, compilar, depurar y ejecutar códigos en lenguaje C, facilitando la tarea de programación.

Si bien esta herramienta no funciona en forma integrada con el aula virtual, la propuesta didáctica para el práctico grupal de máquina se llevó a cabo a través de un foro de discusión creado dentro del EVA.

Utilizando las actividades y recursos disponibles en el aula virtual, se desarrollaron como complemento a los trabajos prácticos de aula y las evaluaciones parciales: lectura de materiales de apoyo desarrollado por la cátedra, discusión en foros, actividades colaborativas para resolver ejercicios propuestos, visualización de videos tutoriales, cuestionarios y entrega de tareas planificadas (códigos).

Es importante mencionar que con el objetivo de desarrollar habilidades de colaboración y favorecer el compromiso con el propio proceso de aprendizaje, se propone el práctico de máquina en forma grupal. Los estudiantes eligen libremente los grupos de trabajo, conformando equipos de dos integrantes. Cada grupo cuenta con la guía de un tutor que promueve la colaboración y el intercambio de conocimiento entre pares. El práctico de máquina consiste en resolver un problema real e implementar una solución en lenguaje C , utilizando el IDE CODE::BLOCKS bajo el sistema operativo GNU/Linux.

## 6 Recolección de datos

Con el propósito de analizar resultados y modificar las prácticas de enseñanza, se obtuvo información a través de diferentes instrumentos: el Diagnostico Inicial explicado en el punto 4, la información de seguimiento provista por el EVA y la encuesta de satisfacción, diseñada por la cátedra, para conocer la opinión de los alumnos.

La encuesta de satisfacción consta de 11 preguntas sobre los siguientes aspectos: desempeño docente, estrategias y materiales elaborados por la cátedra y un espacio especial para que el alumno especifique sugerencias y/o comentarios. (Anexo I).

El cuestionario elaborado se aplicó a todos los alumnos matriculados en la asignatura, en los horarios y aulas habituales que utiliza la cátedra. Una vez completado en forma anónima se procedió a la interpretación de la información para obtener conclusiones.

La Figura 1 refleja que únicamente un 14% de los alumnos asisten a las clases presenciales según lo recomendado por la cátedra y un porcentaje mayor al 80% no puede asistir. Esto confirma la necesidad de proponer estrategias alternativas.

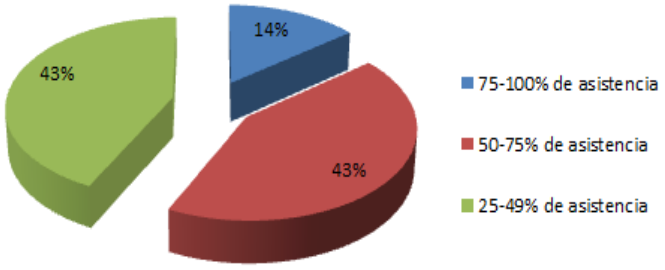


**Figura 1:** Asistencia a las clases presenciales

La Figura 2 refleja que la mitad de los alumnos tienen inconvenientes de asistencia por cuestiones laborales. Por otra parte, un 22% de los alumnos presentan incompatibilidad horaria debido a que cursan materias de diferentes años de su plan de estudio.

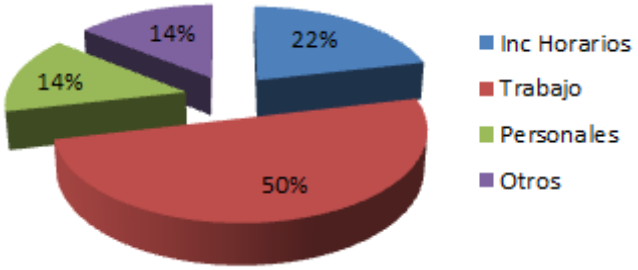


**Figura 2:** Motivos de inasistencias

La Figura 3 indica que la mayoría de los alumnos consideran adecuado el uso de la plataforma "Aulas virtuales" como complemento a las actividades presenciales.

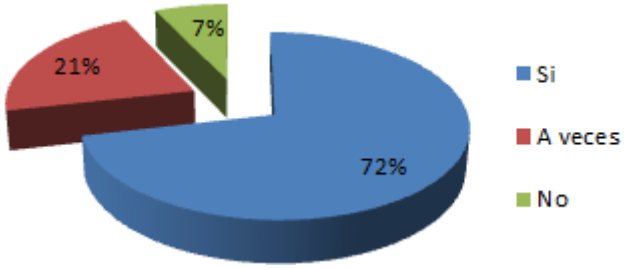


**Figura 3:** Utilización de "Aulas virtuales"

La Figura 4 señala que únicamente el 7% de los alumnos encuentra dificultades con la metodología utilizada por la cátedra para la realización de esta tarea.

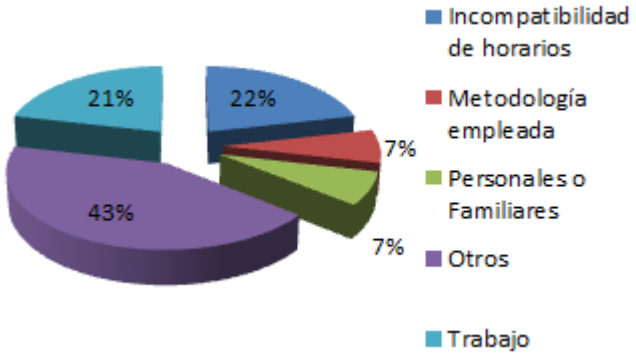


**Figura 4:** Problemas en la realización del práctico de máquina

La Figura 5 indica que los alumnos consideran adecuadas las estrategias y recursos utilizados por la cátedra.

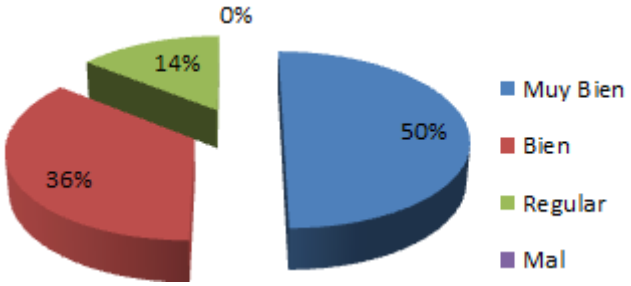


**Figura 5:** Uso de métodos y recursos didácticos adecuados

A los resultados anteriores se suman los datos del rendimiento académico del período 2011-2014 para cotejarlos.

La Figura 6 muestra una comparativa de resultados académicos para las condiciones: Libre por Faltas, Libre por Parcial, Regular y Promoción.

En cuanto a la condición de Libre por Faltas: si bien en el año 2012 existe una disminución importante, en el año 2013 aumenta nuevamente, pero en el año 2014 se registran los porcentajes más bajos de deserción del período analizado.

Se puede observar una disminución progresiva en porcentaje de los alumnos Libres por Parcial y un aumento significativo de aquellos que aprueban la materia, Regular y Promoción.

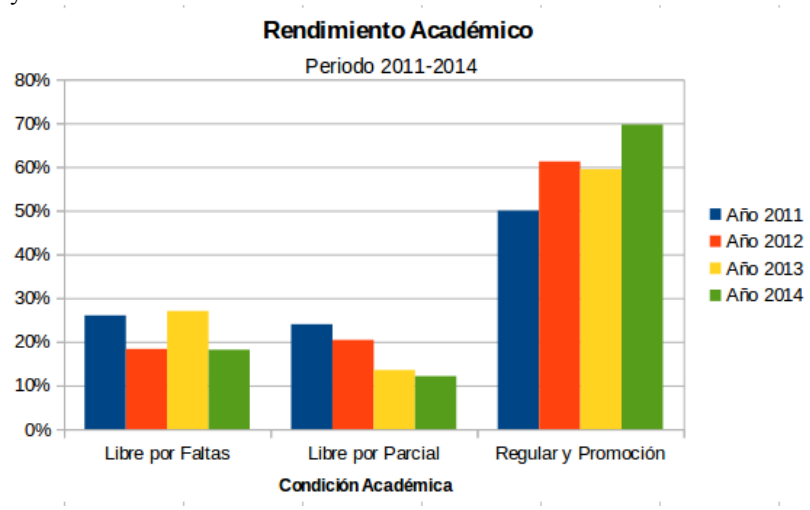


**Figura 6:** Análisis comparativo períodos 2011-2014

## 7- Conclusiones

A partir de 2014 se fortaleció el uso del aula virtual en el desarrollo de actividades colaborativas, lo que impactó positivamente en el rendimiento académico. Desde el punto de vista del docente, el uso de este modelo implicó modificar significativamente el rol, en lo que respecta a la elaboración de materiales pedagógicos, creación y mantenimiento del aula virtual, seguimiento del proceso de aprendizaje y comunicación con los alumnos. Por otra parte, posibilitó el seguimiento continuo de los aprendizajes mediante la lectura de los foros y los informes estadísticos provistos por el EVA, para ofrecer la ayuda necesaria en el momento oportuno. El uso del EVA "Aulas virtuales" y el IDE CODE::BLOCKS permitió a los alumnos trabajar colaborativamente, con independencia de horarios y espacio. Ofreció vías de comunicación más ágiles y contribuye a una formación más flexible durante el proceso de enseñanza y aprendizaje, mediante la realización de actividades no presenciales. Además, brindó respuesta a las demandas expresadas en la encuesta de satisfacción en cuanto a las actividades grupales.

En síntesis, los resultados académicos obtenidos mediante el uso del EVA "Aulas Virtuales" como complemento a la clase presencial y el IDE CODE::BLOCKS, avalan la continuidad de esta línea de trabajo. Surge como alternativa evaluar la posibilidad de utilizar un módulo para programación integrado en el EVA. Esto permitiría realizar un mejor seguimiento del estado de resolución del práctico de máquina, aprovechando los beneficios de usar una interfaz única.

## Referencias


1 Sitio oficial de Moodle. Documentación en línea. Disponible en: www.moodle.org
2 Aguirre Jesús, Viano Hugo, García Berta: Entornos virtuales y herramientas de software libre como apoyo a la enseñanza de programación. Modalidad Poster. 2° Congreso Nacional de Ingeniería Informática-CONAIISI. San Luis. (2014).
3 Viano, Hugo & Aguirrre, Jesús: Uso de aulas virtuales como apoyo a la enseñanza de Programación. En: Chiarani, Daza y Allende (Coords). Cultura digital en la Universidad Nacional de San Luis. pp. 51 a 53 Nueva Editorial Universitaria. ISBN: 978-987-1852-97-0 (2014)
4 Coll, C. & Monereo, C. Educación y aprendizaje en el siglo XXI: Nuevas herramientas, nuevos escenarios, nuevas finalidades. En C. Coll y C. Monereo (Eds.) Psicología de la Educación Virtual (pp.19-53). Madrid: Ediciones Morata. (2008)
5 Johnson, D.W. Johnson, R.T.,& Holubec, E.J.: El aprendizaje cooperativo en el aula. Barcelona: Paidós. (1999).
6 Guitert, M.; Giménez, F.: El trabajo cooperativo en entornos virtuales de aprendizaje. En: Duart, J.M.; Sangra, A. (Ed.) Aprender en la virtualidad, pp. 113 -134. Barcelona: Gedisa. (2000)
7 Gunawardena, Ch., Lowe, C. & Anderson, T.: Analysis of a global online debate and the development of an interaction analysis model for examining social construction of knowledge in computer conferencing. J. Educational Computing Research, vol. 17, núm. 4, pp. 395-429. (1997).
8 Nicholas C. Burbules: El aprendizaje ubicuo y el futuro de la enseñanza, Encounters/Encuentros/Rencontres on Education, Vol. 13, 2012, pág 3 a 14. (2012)
9 Cope Bill y Kalantzis, M.– Aprendizaje ubicuo – Grupo nodos ELE. (2009)
10 Mondéjar, J., Mondéjar, J. A. & Vargas, M.: Implantación de la metodología e- learning en la docencia universitaria: una experiencia a través del proyecto Campus Virtual. Revista Latinoamericana de Tecnología Educativa, 5 (1), 59-71. (2006). [http://www.unex.es/didactica/RELATEC/sumario_5_1.htm]
11 CODE::BLOCKS Sitio oficial: http://www.codeblocks.org/
12 Area, M.y Adell, J.: eLearning: Enseñar y aprender en espacios virtuales. En J. De Pablos (Coord): Tecnología Educativa. La formación del profesorado en la era de Internet. Aljibe, Málaga, pags. 391-424. (2009)

# Anexo I: Encuesta de satisfacción a Estudiantes - 2 Cuat. de 2014

1- ¿Cuántas veces te has matriculado en esta asignatura?

A. 1
B. 2
C. 3 o más

2- ¿Cuál ha sido tu asistencia a las clases de esta asignatura?

A. 0- 25%
B. 25-49 %
C. 50-75%
D. 75% - 100%

3- ¿Cuál es el grado de dificultad de esta asignatura con respecto a otras de la titulación?

A. Muy fácil
B. Fácil
C. Normal
D. Difícil
E. Muy Difícil

4- ¿Cuáles son los motivos de las faltas de asistencia?

A. Trabajo
B. Incompatibilidad de horarios
C. Personales
D. Metodologia empleada
E. Otros

5- ¿Cómo considera que fue su participación en los foros de la plataforma "Aulas Virtuales" durante la cursada?

A. Muy Pobre
B. Pobre
C. Activa
D. Muy Activa

6- Considera que el uso de la plataforma "Aula Virtual" mejoró sus posibilidades de aprender?

A. Si
B. No
C. A veces

7- Cuáles son los principales problemas en la realización del trabajo práctico de máquina:

A. Trabajo
B. Incompatibilidad de horarios
C. Personales
D. Metodologia empleada
E. Otros

8- Considera necesario incluir otras herramientas para favorecer el aprendizaje de la asignatura como:

A. Redes sociales: Facebook, Twitter, etc.
B. La nube: Dropbox, Skydrive, etc.
C. Otros

9- La Cátedra usa métodos y recursos didácticos adecuados para favorecer el aprendizaje de la asignatura (pizarra, transparencias, recursos audiovisuales, etc):

A. Regular
B. Bien
C. Muy Bien

10- La actividad grupal del trabajo práctico de máquina de la asignatura es:

A. Muy fácil
B. Fácil
C. Normal
D. Difícil
E. Muy difícil

11- ¿Cuáles son las actividades que considera necesaria incorporar a la asignatura?

Cuestionarios

A. Encuestas
B. Trabajos grupales
C. Ejemplos
D. Otros

# Diseño e implementación de un simulador software basado en el procesador MIPS32

Manuel Rivas Pérez[1], Manuel Domínguez Morales[1], Francisco Gómez Rodríguez[1], Alejandro Linares Barranco[1], Gabriel Jiménez Moreno[1], Antón Civit Balcells[1]

[1] Departamento de Arquitectura y Tecnología de Computadores
Escuela Técnica Superior de Ingeniería Informática
Universidad de Sevilla
Avenida Reina Mercedes s/n
Sevilla, España
{mrivas, mdominguez, gomezroz, alinares, gaji, civit}@atc.us.es

**Resumen.** La arquitectura de computadores es una asignatura de gran importancia actualmente en las titulaciones de Informática. Pero en muchas ocasiones, los estudiantes tienen problemas para comprender la materia debido a la falta de herramientas que muestren el funcionamiento de los componentes internos de la arquitectura de los computadores de manera fácil e intuitiva. En este trabajo se expone un simulador del procesador MIPS32 desarrollado en .NET que puede ser controlado a través de línea de comandos o desde una interfaz gráfica versátil e intuitiva para facilitar a los alumnos el estudio de la arquitectura de los procesadores segmentados. La interfaz gráfica ofrece un entorno de desarrollo integrado en el que editar y ensamblar los programas, así como mostrar el funcionamiento del procesador a través de sus registros, memoria, pipeline y el cronograma de ejecución. En este trabajo se expondrá un simulador como producto que responde a las necesidades de los alumnos en asignaturas relacionadas con el estudio de la arquitectura de los computadores. En primer lugar se expondrá una comparativa de simuladores MIPS, posteriormente se mostrarán las características del procesador que se simula, se describirá la implementación del ensamblador y del propio simulador y finalmente se mostrará su funcionamiento a través de la interfaz gráfica desarrollada denominada VisualMips32.

**Palabras Clave:** MIPS32, simulador, segmentación, .NET, arquitectura de computadores.

**Abstract.** Nowadays, computer architecture is a very important subject in Computer Science degrees. But often, students have problems understanding the topic due to the lack of tools to show the behavior of the internal computer architecture components in an easy and intuitive way. This paper presents a MIPS32 processor simulator developed in .NET that can be controlled via command line orders or using a versatile and intuitive graphical interface that makes the study of segmented processor architecture easier. The GUI (graphical user interface) offers an integrated development environment, where assembly programs can be assembled and run, in addition to being able to watch a step-

by-step execution through its registers, memory, pipeline and execution chronogram. This paper presents a simulator as a software tool developed to meet the students' needs in subjects related to the computer architecture. First, a comparative of several MIPS simulators is shown. After that, the implementation of the assembler and the simulator itself will be described; and, finally, its operation is displayed through the developed graphical interface, called VisualMips32.

**Keywords:** MIPS32, simulator, segmentation, .NET, computer architecture.

# 1 Introducción

La electrónica digital ha evolucionado rápidamente en muy poco tiempo llegando a integrar más de mil millonesde transistores en un mismo chip. La arquitectura también ha evolucionado con nuevas estructuras como la segmentación, el uso de cachés, procesamiento multi-hilo o el multiprocesamiento ayudados por este aumento de la integración de los transistores. El estudio de la arquitectura de los computadores es de gran importancia en las titulaciones universitarias relacionadas con la informática. Estas arquitecturas, fruto de su evolución, pueden resultar complejas lo que provoca que muchos estudiantes encuentren dificultades para comprender algunos aspectos de la asignatura de arquitectura de computadores, como la segmentación de cauce (*pipelining*) o el cálculo del rendimiento del procesador. La segmentación es una técnica muy importante en arquitectura de computadores en la que varias instrucciones pueden ejecutarse simultáneamente manteniendo cada una de ellas en una etapa diferente de la ruta de datos (*datapath*). El uso de procesadores ejemplo ayuda al alumno a comprender los conceptos clave de la arquitectura y las mejoras de los procesadores. Este es el caso del procesador MIPS descrito en los libros de Henessy y Patterson [1] [2] ampliamente conocidos por la comunidad universitaria, que han convertido a este procesador de ejemplo en uno de los más utilizados para enseñar la asignatura de arquitectura de computadores en las universidades [3].

Pero aun así, estos contenidos son difíciles de asimilar si se exponen únicamente haciendo uso de la pizarra o con lápiz y papel. Es por ello que muchos los profesores emplean simuladores gráficos como herramienta para mostrar los conceptos de forma intuitiva e interactiva simplificando la forma en la que los profesores enseñan y los alumnos aprenden la arquitectura de computadores. Hay numerosos artículos que tratan sobre el aprendizaje de la arquitectura de computadores [4–7] y la mayoría defienden la importancia de los simuladores para el aprendizaje. De hecho estos simuladores se han convertido en una herramienta indispensable para que los alumnos puedan entender las complejas arquitecturas difíciles de asimilar sin la interfaz gráfica que los simuladores actuales pueden ofrecer [8].

Hay una gran variedad de simuladores actualmente, desde los más simples hasta complejos simuladores utilizados en investigación y que son analizados en [9]. Algunos dirigidos a simular los componentes hardware y otros la arquitectura del juego de instrucciones cada uno de ellos con distinto nivel de detalle desde simplemente modelar el comportamiento externo a mostrar las interacciones de los componentes internos. A pesar de haber gran variedad de ellos, algunos no son

adecuados para el aprendizaje ya que son difíciles de usar y comprender, o bien son demasiado específicos [6].

En este trabajo se presenta un software que simula y visualiza el procesamiento de instrucciones del procesador segmentado MIPS para mejorar la calidad de la enseñanza y dar a los estudiantes un entorno en el que experimentar. Este trabajo se organiza de la siguiente forma: En la sección 2 se exponen los requerimientos necesarios para desarrollar el simulador académico adecuado. La sección 3 describe muchos de los simuladores que existen y se especifican los aspectos más relevantes de cada uno de ellos. La sección 4 muestra las características del procesador MIPS que se ha simulado. Posteriormente se describe la implementación del ensamblador y del simulador en la sección 5, el funcionamiento del entorno visual VisualMIPS32 en la sección 6 y, terminando el presente trabajo, con las mejoras futuras y las conclusiones.

## 2 Objetivos

Para conseguir el objetivo principal de obtener una herramienta que facilite la enseñanza y el aprendizaje de la arquitectura de computadores, el desarrollo de este software se basa en los siguientes requerimientos:

- Permitir reconfigurar y parametrizar el procesador.
- Realizar ejecución paso a paso y control de puntos de interrupción (*Breakpoints*).
- Simular la versión monociclo, multiciclo y segmentada del procesador.
- Ofrecer una interfaz gráfica intuitiva, clara y fácil de usar.
- Integrar un editor de ensamblador en el entorno de desarrollo.
- Incluir un amplio número de instrucciones del repertorio tanto de enteros como de coma flotante además de pseudo-instrucciones.
- Mostrar información clara y detallada sobre la codificación de cada instrucción, la segmentaciónde las instrucciones, los tipos de bloqueos que se producen, la activación de desvíos (*bypasses*) y las estadísticas sobre el rendimiento obtenido.

## 3 Otros simuladores

Como se indicó anteriormente, existen actualmente muchas herramientas para simular el funcionamiento de un procesador y de muy diversos tipos, algunas de ellas poco adecuadas para la enseñanza debido a su dificultad o por ser muy específicas [7]. A continuación se describen los simuladores orientados a la enseñanza más conocidos especificando algunos puntos a favor o en contra de ellas:

- *QtSpim* [10, 11]. Es un simulador de MIPS32 programado en C++ y Qt que fue muy utilizado tanto en educación como en la industria [12]. Este

simulador implementa un amplio repertorio de instrucciones incluyendo algunas en coma flotante [13]. Es una buena herramienta de depuración y es intuitiva pero sólo simula la versión del procesador monociclo.

- *MARS* [12, 14]. Está escrito en java y se utiliza de muchas universidades de todo el mundo. Simula la ejecución del programa ensamblador paso a paso y muestra el resultado de los registros y de la memoria. Su entorno de desarrollo integra un editor capaz de mostrar la sintaxis en una ventana emergente durante la programación. Es una buena herramienta de simulación y depuración aunque, al igual que QtSpim, sólo es capaz de simular la versión monociclo del MIPS. Una de las características más destacable de este simulador es que admite la inclusión de plug-ins lo que permite ampliar su funcionalidad. Buen ejemplo de ello es el MIPS X-Ray [15], un plug-in que representa gráficamente la ruta de datos del procesador.
- *ProcessorSim* [16]. Es una herramienta desarrollada en java que simula el MIPS R2000 monociclo. Este simulador incluye varias rutas de datos diferentes, e incluso el usuario puede crear más, que se muestran gráficamente en forma de animación. Respecto a la animación, un problema del que adolece es que sólo un componente puede enviar un mensaje en cada instante de tiempo cuando en realidad todos los componentes deberían trabajar concurrentemente [17]. Otras de sus debilidades son que no muestra ruta de datos de la versión segmentada, que su editor es muy simple y que el juego de instrucciones que implementa es muy reducido.
- *MIPS-Datapath* [18]. Es un simulador desarrollado en C++ que muestra la ruta de datos gráficamente tanto de la versión monociclo como de la segmentada, con y sin adelantamientos. Permite la ejecución paso a paso, pero posee un juego de instrucciones muy reducido y no soporta el bloqueo del cauce.
- *WebMIPS* [19]. Este simulador fue programado en ASP y simula el MIPS segmentado con detección de riesgos. Es un simulador educativo interesante ya que se ejecuta desde un explorador web por lo que no precisa instalación. Es capaz de ejecutarse paso a paso mostrando la ruta de datos gráficamente. Es un buen simulador educativo pero no implementa la versión monociclo, su editor es muy simple y la representación de la ruta de datos es estática.
- *EduMIPS64* [20, 21]. Esta herramienta es una re-implementación del simulador WinMips64 [22][23]. Simula el procesador MIPS64 segmentado, posee una interfaz muy intuitiva, posee un amplio repertorio de instrucciones que incluye operaciones en punto flotante, llamadas al sistema e implementa la detección de bloqueos. Por el contrario, no posee un editor integrado, no soporta simulación monociclo ni muestra la ruta de datos.
- *MiniMIPS* [24]. Es un simulador implementado en C para Unix. Muestra la ruta de datos durante la simulación pero no es a nivel RT ni es animada y el único dato de rendimiento que ofrece es el número de ciclos.
- *DrMips* [25]. Es un simulador de MIPS32 implementado en java que visualiza paso a paso la ruta de datos incluyendo la latencia de los componentes cuando usa el modo performance. Muestra varias rutas de datos basadas en libros de Henessy y Patterson [1, 2] incluyendo la ruta de datos monociclo y segmentada. Posee un editor que muestra la sintaxis en un menú

emergente. Su característica más destacable es que posee una versión para la plataforma Android.

En [19] se muestra una tabla que compara varios simuladores MIPS teniendo en cuenta si implementan el MIPS de Hennessy y Patterson [1, 2] a nivel RT, si muestran datos estadísticos del rendimiento, si realizan animaciones en tiempo de simulación que muestren cómo se transfieren las señales o si emplean una plataforma independiente que los haga más portables. En [25] también se expone una tabla comparativa muy completa de muchos de los simuladores anteriormente citados teniendo en cuenta muchos aspectos como, si integra un editor, si implementa instrucciones en coma flotante, si la ruta de datos es configurable, si es segmentado, o si permite la edición de datos durante la simulación.

## 4 Características del procesador simulado: MIPS32

MIPS (acrónimo de Microprocessor without Interlocked Pipeline Stages) es un procesador con un juego reducido de instrucciones (RISC) desarrollado por MIPS Computer Systems, que pasó a llamarse MIPS technologies y que ha sido vendida a Imagination technologies.

Los procesadores MIPS forman parte de una gran familia de procesadores RISC desde su primer procesador R2000 en el 1986 cuyo juego de instrucciones ha evolucionado desde la versión original denominada MIPS I hasta las actuales MIPS32 y MIPS64.

Gracias a la sencillez de su arquitectura y al hecho de que actualmente sea bastante utilizado en sistemas embebidos, lo ha convertido en un procesador muy adecuado para el estudio de la arquitectura de computadores en las universidades.

El simulador presentado en este trabajo está basado en el procesador MIPS32 cuya arquitectura ISA puede consultarse en [26, 27, 28]. Las características principales de este procesador son:

- Procesador RISC segmentado de 32 bits y direcciones de 32 bits
- Instrucciones de tamaño fijo para una rápida decodificación de las instrucciones.
- Almacenamiento en memoria en *big-endian* o *little-endian*.
- 32 registros de propósito general de 32 bits para enteros.
- Unidades funcionales independientes para el producto y la división. Estas unidades funcionales contienen además los registros HI y LO utilizados para almacenar el resultado de operaciones (véase Figura 1).
- Coprocesador de coma flotante. Este coprocesador es identificado como coprocesador 1 y posee un banco de 32 registros de 32 bits (Figura 1) que pueden emplearse para almacenar números en coma flotante de simple precisión (32 bits) o bien pueden agruparse por pares para almacenar números en coma flotante de doble precisión (64 bits).
- Unidad de detección y control de riesgos.
- Implementa adelantamientos (*forwarding*) para reducir los bloqueos de datos.
- Saltos retardaos para reducir los bloqueos de control.

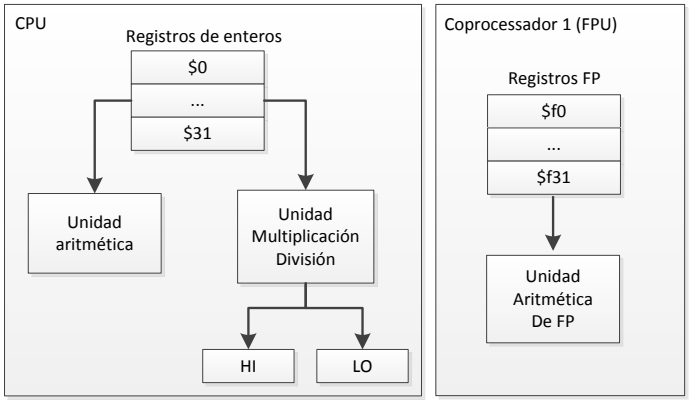


**Figura 1.** Unidades de procesamiento del MIPS32

Todas estas características han sido implementadas en el procesador siendo configurables la utilización de adelantamientos, los saltos retardados, el número de ciclos que duran las etapas y cuáles de ellas son segmentadas (ver Figuras 11-12).

### 4.1 Modos de direccionamiento

Las arquitecturas RISC admiten un conjunto muy reducido de modos de direccionamiento y todos ellos han sido implementados en el simulador. Los modos de direccionamiento son:

- Registro. El operando está en un registro del banco de registros de enteros o de coma flotante según el tipo de instrucción. Para especificar los registros de enteros en ensamblador pueden utilizarse indistintamente dos nomenclaturas. La nomenclatura empleada para cada registro aparece en la Tabla 1 y también se muestra el uso al que va destinado cada registro. Una nomenclatura consiste en emplear el símbolo '$' seguido del número del registro (por ejemplo *add $1, $2, $3*) y otra utiliza el símbolo '$' seguido de una palabra que define su utilización, por ejemplo *add $at, $v0, $v1*.

**Tabla 1.** Nomenclatura y uso de los registros de enteros del procesador.

| Nombre | Registro | Uso |
|---|---|---|
| **$zero** | 0 | Constante 0 |
| **$at** | 1 | Reservado para el ensamblador |
| **$v0-$v1** | 2-3 | Valores de retorno |
| **$a0-$a3** | 4-7 | Argumentos |
| **$t0-$t7** | 8-15 | Temporales |
| **$s0-$s7** | 16-23 | Valores Guardados |
| **$t8-$t9** | 24-25 | Temporales |
| **$k0-$k1** | 16-27 | Reservados para el kernel |
| **$gp** | 28 | Puntero Global |
| **$sp** | 29 | Puntero de Pila |
| **$fp** | 30 | Puntero de Marco |
| **$ra** | 31 | Dirección de retorno |

- Inmediato. El operando es una constante de 16 bits almacenada en la propia instrucción, por ejemplo *addi $1,$2, 2000*.
- Registro base más desplazamiento. Direccionamiento a memoria en el que la dirección efectiva viene determinada por el contenido de un registro denominado registro base más una constante que actúa de desplazamiento. Este modo de direccionamiento sólo se usa en instrucciones de acceso a memoria (cargas y almacenamientos). Su sintaxis es **inm(reg)**, siendo *reg* el registro base e *inm* el desplazamiento, por ejemplo *lw $1, 24($2)*.
- Relativo al Contador de Programa. Se emplea exclusivamente en los saltos condicionales y la dirección destino de salto se obtiene sumando una constante de 16 bits en complemento a 2 al registro contador de programa. Este modo de direccionamiento queda oculto en el ensamblador gracias al uso de etiquetas. Un ejemplo de este modo sería *beqz $1, $2, etiqsigue*.
- Pseudodirecto: Utiliza una constante para especificar los 26 bits menos significativos de la dirección. Los 6 bits más significativos que faltan, de los 32 bits que forman la dirección, se toman del registro PC. Es por ello por lo que su nombre contiene el prefijo pseudo. Este modo se emplea exclusivamente para indicar la dirección de salto en las instrucciones de salto incondicional sin registro (instrucciones *j* y *jal*). Al igual que en el modo de direccionamiento anterior, el editor oculta este detalle mediante etiquetas (por ejemplo *j sigue*).

### 4.2 Repertorio de instrucciones

De todo el repertorio de instrucciones de la arquitectura MIPS32, formado por algo más de 200 instrucciones incluyendo las que operan en coma flotante, este simulador implementa 80 de ellas que son las que comúnmente se encuentran en la mayoría de los programas en ensamblador.

Las tablas 2a-2d muestran el repertorio de las instrucciones implementadas en el simulador clasificadas según el tipo de operación que realizan.

**Tabla 2a.** Instrucciones de carga y almacenamiento.

| Instrucción | Descripción |
|---|---|
| lb reg,inm16(reg) | Carga byte con signo en reg. entero desde mem (extiende signo). |
| lbu reg,inm16(reg) | Carga byte sin signo en reg. entero desde mem (extiende ceros). |
| lh reg,inm16(reg) | Carga 16 bits con signo en reg. entero desde mem. (extiende signo) |
| lhu reg,inm16(reg) | Carga 16 bits sin signo en reg. entero desde mem. (extiende ceros) |
| lw reg,inm16(reg) | Carga 32 bits en reg. entero desde mem. |
| sb reg,inm16(reg) | Guarda byte en mem. desde reg. entero |
| sh reg,inm16(reg) | Guarda 16 bits en mem. desde reg. entero |
| sw reg,inm16(reg) | Guarda 32 bits en mem. desde reg. entero |
| lwc1 freg,inm16(reg) | Carga 32 bits en reg. FP simple prec. desde mem. |
| ldc1 freg,inm16(reg) | Carga 64 bits en reg. FP doble prec. desde mem. |
| swc1 freg,inm16(reg) | Guarda 32 bits en mem. desde reg. FP |
| sdc1 freg,inm16(reg) | Guarda 64 bits en mem. desde reg FP doble prec. |
| mfc1 reg,freg | Carga 32 bits en reg. entero desde reg. FP |

| | |
|---|---|
| mtc1 reg,freg | Carga 32 bits en reg. FP desde reg. entero |
| mflo reg | Carga 32 bits en reg. entero desde reg. LO |
| mfhi reg | Carga 32 bits en reg. entero desde reg. HI |
| lui reg,inm16 | Carga inm de 16 bits en parte alta de reg. entero |
| li reg,imm32 | Carga inm de 32 bits en reg. entero (pseudo-instrucción). |
| la reg,etiq | Carga en reg. entero la dirección de una etiqueta |

**Tabla 2b.** Instrucciones de operaciones aritméticas y lógicas con enteros.

| Instrucción | Descripción |
|---|---|
| add reg,reg,reg | Suma entre registros de enteros |
| addu reg,reg,reg | Suma entre registros de enteros sin desbordamiento |
| addi reg,reg,inm16 | Suma de enteros con inmediato |
| addiu reg,reg,inm16 | Suma de enteros con inmediato sin desbordamiento |
| sub reg,reg,reg | Resta entre registros de enteros |
| subu reg,reg,reg | Resta entre registros de enteros sin desbordamiento |
| mul reg,reg,reg | Multiplicación de enteros con signo. |
| mult reg,reg | Multiplicación de enteros con signo de 32 bits. |
| multu reg,reg | Multiplicación de enteros sin signo de 32 bits. |
| div reg,reg | División de enteros con signo. |
| divu reg,reg,reg | División de enteros sin signo. |
| and reg,reg,reg | Operación AND entre registros |
| andi reg,reg,inm16 | Operación AND entre registro e inmediato |
| nor reg,reg,reg | Operación NOR entre registros. |
| or reg,reg,reg | Operación OR entre registros. |
| ori reg,reg,inm16 | Operación OR entre registro e inmediato |
| xor reg,reg,reg | Operación OR exclusiva entre registros. |
| xori reg,reg,inm16 | Operación OR exclusiva entre registro e inmediato |
| sll reg,reg,inm16 | Desplazamiento lógico a la izquierda |
| srl reg,reg,inm16 | Desplazamiento lógico a la derecha |
| sra reg,reg,inm16 | Desplazamiento aritmético a la derecha |
| sllv reg,reg,reg | Desplazamiento lógico a la izquierda variable |
| srlv reg,reg,reg | Desplazamiento lógico a la derecha variable |
| srav reg,reg,reg | Desplazamiento aritmético a la derecha variable |
| rol reg,reg,reg | Rotación a la izquierda variable |
| ror reg,reg,reg | Rotación a la derecha variable |
| slt reg,reg,reg | Comparación “menor que” entre reg. con signo |
| sltu reg,reg,reg | Comparación “menor que” entre reg. sin signo |
| slti reg,reg,inm16 | Comparación “menor que” con inm y con signo |
| sltiu reg,reg,inm16 | Comparación “menor que” con inm sin signo |
| nop | No operación. |

**Tabla 2c.** Instrucciones de operaciones aritméticas y lógicas en coma flotante.

| Instrucción | Descripción |
|---|---|
| add.d freg,freg,freg | Suma en FP de simple precisión |
| sub.d freg,freg,freg | Resta en FP de simple precisión |
| mul.d freg,freg,freg | Multiplicación en FP de simple precisión |
| div.d freg,freg,freg | División en FP de simple precisión |
| abs.d freg,freg | Valor absoluto en FP de simple precisión |
| add.d freg,freg,freg | Suma en FP de doble precisión |
| sub.d freg,freg,freg | Resta en FP de doble precisión |
| mul.d freg,freg,freg | Multiplicación en FP de doble precisión |

| | |
|---|---|
| div.d freg,freg,freg | División en FP de doble precisión |
| abs.d freg,freg | Valor absoluto en FP de doble precisión |

**Tabla 2d.** Instrucciones de control.

| Instrucción | Descripción |
|---|---|
| beq reg,reg,etiq | Salto rel. a PC si ambos registros son iguales |
| bne reg,reg,etiq | Salto rel. a PC si ambos registros son distintos |
| beqz reg,etiq | Salto rel. a PC si el registro es igual a "0" |
| bnez reg,etiq | Salto rel. a PC si el registro es distinto de "0" |
| bltz reg,etiq | Salto rel. a PC si el registro es menor que"0" |
| blez reg,etiq | Salto rel. a PC si el registro es menor o igual a "0" |
| bgtz reg,etiq | Salto rel. a PC si el registro es mayor que "0" |
| bgez reg,etiq | Salto rel. a PC si el registro es mayor o igual a "0" |
| bgt reg,reg,etiq | Salto rel. a PC si es mayor con signo |
| bge reg,reg,etiq | Salto rel. a PC si es mayor o igual con signo |
| blt reg,reg,etiq | Salto rel. a PC si es menor con signo |
| ble reg,reg,etiq | Salto rel. a PC si es menor o igual con signo |
| bgtu reg,reg,etiq | Salto rel. a PC si es mayor sin signo |
| bgeu reg,reg,etiq | Salto rel. a PC si es mayor o igual sin signo |
| bltu reg,reg,etiq | Salto rel. a PC si es menor sin signo |
| bleu reg,reg,etiq | Salto rel. a PC si es menor o igual sin signo |
| j etiq | Salto incondicional a dirección de 26 bits (etiq) |
| jal etiq | Salto incondicional a dir. de 26 bits con retorno. |
| jr reg | Salto incondicional a dir. dada por registro |
| jalr reg | Salto incondicional a dir. por registro con retorno. |

## 5 Implementación

Esta herramienta se ha implementado en C# .NET y consta fundamentalmente de los siguientes elementos:

- *Ensamblador* Se ha desarrollado una librería dinámica (.dll) encargada de analizar el código ensamblador de las 80 instrucciones enumeradas en las Tablas 2a-2d y generar el código máquina incluyendo la información del analizador.
- *Simulador MIPS32*. Se ha implementado el simulador del MIPS32 en otra librería dll independiente incluyendo la memoria principal. Básicamente se encarga de simular paso a paso el procesamiento de las instrucciones e informar de las etapas del pipeline. Esta librería utiliza directamente el componente anterior por lo que abarca todo el proceso desde el ensamblado del programa hasta la simulación.
- *ConsolaMips32*. Es una interfaz que muestra por pantalla el cronograma de ejecución en modo texto.
- *VisualMips32*. Interfaz gráfica del simulador que integra el editor y que muestra el estado del procesador desde distintos puntos de vista como es el pipeline, cronograma, registros, memoria, etc. durante la simulación de forma gráfica.

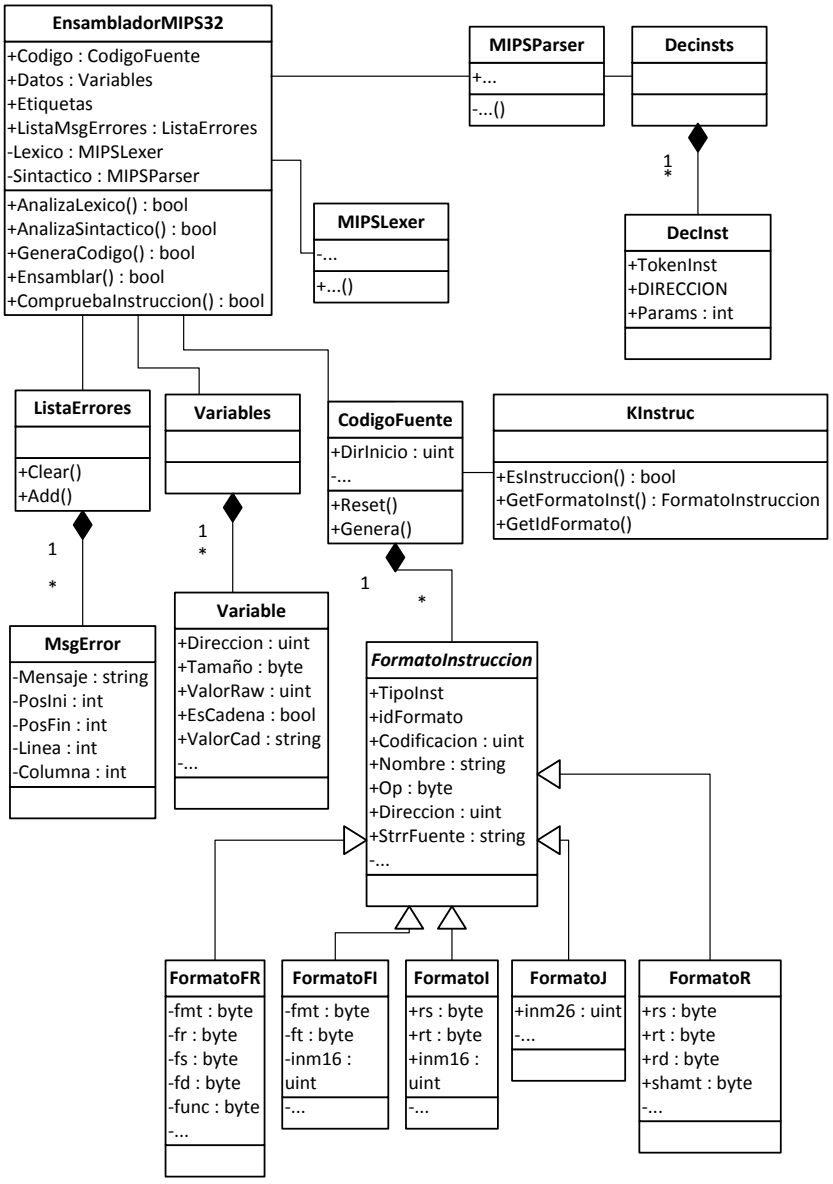


**Figura 2.** Diagrama de clases básico del ensamblador

### 5.1 Desarrollo del ensamblador

Los analizadores léxico y sintáctico, así como el generador de código, se encuentran implementados en la librería *AnalizadorMIPS32_ANTLR3.dll* cuyo diagrama de clases se muestra de forma muy simplificada en la figura 2. Este simulador verifica la sintaxis de las instrucciones basándose en el repertorio del instrucciones (ISA) descrito en [26] [27]. En la Figura 2 destaca la clase *EnsambladorMIPS* encargada de gestionar el análisis, el control de errores y la generación de código. A partir del programa en ensamblador, la clase *MIPSLexer* es la responsable de realizar el análisis léxico y generar la lista de *tokens*. Posteriormente, el analizador sintáctico *MIPSParser* realiza el parsing de los *tokens* del analizador léxico y genera una lista de declaraciones de instrucciones (clase *DecInsts*). Cada objeto de la clase *DecInst* guarda el *token* que identifica la operación de la instrucción, una lista de sus parámetros y la dirección que le correspondería en memoria. Durante la fase de generación de código, se crea un objeto de la clase *CodigoFuente* el cual genera una lista de objetos *FormatoInstruccion* a partir de la lista de declaración de instrucciones obtenida por el analizador sintáctico y la clase estática *Kinstruc*. La clase abstracta *FormatoInstruccion* es implementada por cada uno de los formatos de instrucción que posee el MIPS: *FormatoR (registro)*, *FormatoI (inmediato)*, *FormatoJ (salto incondicional) y FormatoFR (operaciones en punto flotante)*. *FormatoInstruccion* representa a una instrucción máquina a simular y mantiene, además del código de la instrucción, toda la información relacionada con el programa fuente, los campos de cada formato, dirección de memoria, mnemónico, etc. Aunque podría haberse generado gran parte del código (salvo la resolución de etiquetas) durante el análisis sintáctico, realizar la generación de código en un paso posterior reduce el tiempo del análisis sintáctico y por consiguiente el de la detección de errores del programa.

### 5.2 Desarrollo del simulador

La implementación del simulador está subdividida básicamente en 3 elementos: La memoria, los bancos de registros y el procesador. Éste último se encarga de la parte más compleja (implementar cada etapa del pipeline, decodificar instrucciones, implementar los desvíos para los adelantamientos y registrar el estado del pipeline en cada ciclo), permitiendo crear un historial de acontecimientos con el fin de simplificar el desarrollo de interfaces de usuario. Un diagrama de clases simplificado del simulador se muestra en la Figura 3. En ella sólo se han representado los atributos, métodos y relaciones entre clases más importantes para mantener la claridad del diagrama.

El sistema simula una memoria de 4GB, esto es, la capacidad máxima admisible para el procesador MIPS32. Puesto que implementar todas las posiciones de la memoria sería muy costoso, se ha optado por diseñar un método que almacene sólo los rangos de posiciones de memoria que contengan valores distintos de cero. Con este método, al escribir datos en memoria fuera de los rangos ya definidos, se crea un nuevo rango de memoria o se amplía un rango ya existente en función de lo alejados que se encuentren los nuevos datos de dicho rango.

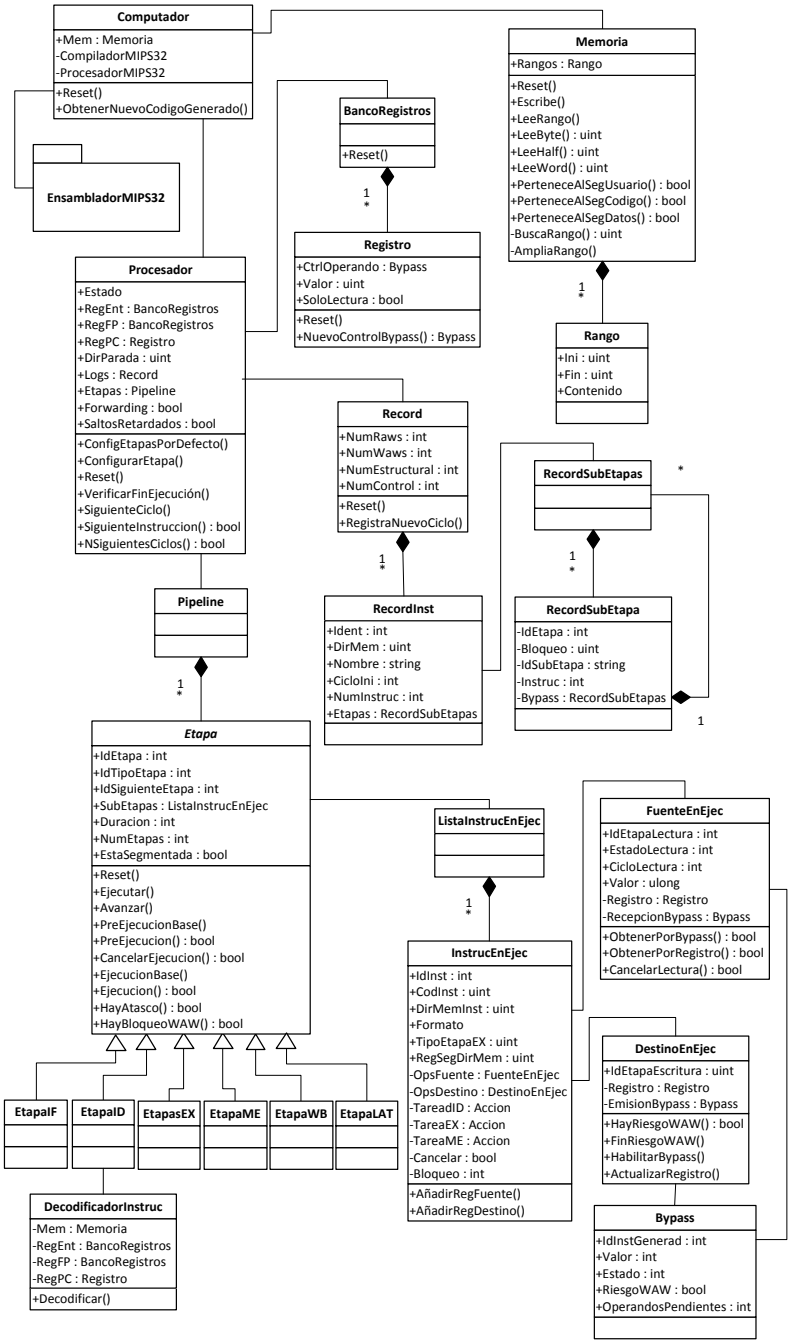


**Figura 3.** Diagrama de clases básico del simulador. Nota: EtapaLAT no es del diseño MIPS, pero es necesaria en el simulador (detallado más adelante).

La clase *Procesador* es la responsable de controlar la ejecución paso a paso. Para la simulación de un ciclo del reloj, el procesador realiza un proceso de 3 pasos con cada etapa del pipeline empezando desde la última hasta la primera etapa. Tales pasos se realizan de forma intercalada, esto es, se realiza el paso 1 en todas las etapas antes de continuar con el paso 2.

Cada paso consiste en lo siguiente:

- *Paso 1*. *Avanzar*: el procesador llama a la función *Avanzar()* de cada etapa encargada de transferir la instrucción en ejecución (implementada en la clase InstrucEnEjec) a la siguiente etapa o subetapa (en caso de ser una etapa segmentada, ver Figuras 11-12) si se encuentra libre.
- *Paso 2*. *Pre-ejecución*: se llama a la función *PreEjecucionBase()* de la superclase Etapa encargada de verificar si la instrucción alojada en la etapa cumple las condiciones para esa etapa. Para tener en cuenta las consideraciones propias de la etapa, la función *PreEjecucionBase()* llama a su vez a *PreEjecucion()* que es propia para cada subclase de la superclase Etapa. Este último método se encarga habitualmente de verificar si los operandos fuente están disponibles pues son necesarios para la ejecución de la etapa permitiendo así detectar bloqueos RAW. Para realizar la verificación el método deberá inspeccionar el atributo *OpsFuente* de la clase *InstruccEnEjec*. De no cumplir alguna de las condiciones, la ejecución de la etapa será cancelada hasta el siguiente ciclo, momento en el que volverán a ser reevaluadas las condiciones. En este paso también podrían llevarse a cabo las acciones propias del primer semiperiodo de un ciclo, como es el caso de la lectura de los bancos de registros o la decodificación de la instrucción en la etapa ID. El proceso de decodificación de la instrucción se realiza a través de la clase DecodificadorInstruc.
- *Paso 3*. *Ejecución*. Del último paso se encarga la función *EjecucionBase()* de la superclase Etapa. Ella será la responsable de verificar si se cumplen las condiciones relacionadas con los operandos destino, esto es, verificar si hay bloqueos WAW; y también debe realizar las tareas específicas de esa etapa. Para ejecutar tales operaciones, *EjecucionBase()* llama a la función *Ejecución()* de la subclase de Etapa correspondiente. Esta última función verifica las condiciones particulares de la etapa y lanza la tarea que la instrucción en ejecución *InstrucEnEjec* tiene asociada a esa etapa. Concretamente los atributos *TareaId*, *TareaEx* y *TareaME* de *InstrucEnEjec* contienen las tareas a realizar mediante delegados.

Para la implementación de los desvíos cuando hay adelantamientos (forwarding), se utiliza la clase Bypass que es mantenida por *Pipeline* a través de las clases *Registro*, *FuenteEnEjec* y *DestinoEnEjec*.

La estrategia básica para controlar los desvíos se basa en las dos reglas siguientes:

- Cada registro destino de la instrucción (atributo *OpsDestino* de *InstrucEnEjec*) creará un nuevo objeto Bypass que se asociará al registro correspondiente del banco de registros durante la decodificación de la instrucción.
- Cada registro fuente de la instrucción (atributo *OpsFuente* de *InstrucEnEjec*) se enlazará con el objeto Bypass perteneciente al registro fuente del banco de

registro cuando la instrucción es decodificada. Este proceso se conoce como suscripción al bypass.

- En la etapa en la que se calcule el resultado de la operación deberá habilitarse el bypass del registro destino para notificar que se encuentra disponible para todos los operandos fuente relacionados con dicho bypass. Ello se consigue llamando a la función *HabilitarBypass()* del objeto *DestinoEnEjec*.
- En la etapa en la que la instrucción precise un operando fuente, lo tomará por bypass siempre que éste se encuentre disponible. En caso de tomarlo, deberá notificarlo para actualizar convenientemente el número de envíos que el bypass tiene pendiente.
- En cuanto se actualice el resultado de la operación en el registro destino correspondiente, deberá notificarse que el desvío ya no será necesario para nuevos operandos fuente aunque los operandos ya suscritos sí seguirán tomando el operando a través del desvío.
- A pesar de haber ejecutado todas las etapas hasta WB, una instrucción no podrá abandonar el pipeline mientras tenga envíos por bypass pendientes, permaneciendo mientras tanto en la *EtapaLAT* diseñada para tal fin.

La Figura 4 muestra un ejemplo del mecanismo descrito anteriormente indicándose el instante en el que se crean los desvíos, se suscribe a ellos, se habilitan, se envían operandos por bypass y se actualiza el registro. Por ejemplo, el ciclo de vida del bypass B1:

- *Ciclo 2 (ID de I1)*: se crea el bypass B1 asociado al registro de destino $1 de la instrucción 1.
- *Ciclo 3 (EX de I1)*: el bypass B1 es habilitado por lo que el valor de $1 se encuentra disponible por bypass.
- *Ciclo 3 (ID de I2)*: el operando fuente $1 de I2 se suscribe a B1 para tomar el valor desde éste cuando la instrucción I2 lo precise.
- *Ciclo 4 (EX de I2)*: la instrucción I2 recibe el dato correspondiente por bypass gracias a la suscripción que se realizó a B1.
- *Ciclo 4 (ID de I3)*: el operando destino de la instrucción I3 es $1 por lo que se crea un nuevo bypass (bypass B3) que remplaza a B1. A partir de este momento, las nuevas suscripciones a $1 irán dirigidas a B3.
- *Ciclo 5 (WB de I1)*: se actualiza el valor del bypass B1 en el registro.

| | **Ciclo** | **1** | **2** | **3** | **4** | **5** | **6** | **7** |
|---|---|---|---|---|---|---|---|---|
| **I1** | **addi $1,$0,1** | IF | **ID** | **EX** | ME | WB | | |
| **I2** | **addi $5,$1,3** | | IF | **ID** | **EX** | ME | WB | |
| **I3** | **lw $1,0($5)** | | | IF | **ID** | EX | ME | WB |
| **I4** | **add $9,$1,$7** | | | | IF | ID | -- | EX |
| Bypass creado | | | $1→B1 | $5→B2 | $1→B3 | $9→B4 | | |
| Bypass suscrito | | | | B1@I2 | B2@I3 | B3@I4 | | |
| Bypass habilitado | | | | B1 | B2 | | B3 | B4 |
| Bypass enviado | | | | | B1→I2 | B2→I3 | | B3→I4 |
| Bypass actualizado | | | | | | B1 | B2 | B3 |

**Figura 4.** Ejemplo de funcionamiento de los desvíos

El simulador implementa un modelo para registrar los sucesos que se van produciendo en cada etapa para cada ciclo de reloj durante la simulación, como son los bloqueos o los desvíos que se activan. El objetivo de mantener este historial de acontecimientos es ofrecer un mayor soporte a las interfaces que podrían necesitar estos datos, por ejemplo cuando el usuario navegue por el cronograma. Las clases que implementan esta funcionalidad se muestran en la Figura 3.

La clase Record mantiene además de la estadística de rendimiento, una lista de las instrucciones que se han ejecutado. Cada instrucción ejecutada se registra en un objeto de la clase *RecordInst* que incluye toda la información que pudiera ser relevante posteriormente, como es el ciclo en el que comenzó la ejecución, un identificador de la línea con la que se corresponde en el programa ensamblador, la dirección de memoria y una lista de lo ocurrido en cada ciclo de reloj mientras estuvo la instrucción en ejecución. Esta lista de sucesos está implementada en la clase *RecordSubEtapas*, la cual contiene objetos de *RecordSubEtapa* que son los responsables de guardar la información del estado de la instrucción en cada ciclo, como es el tipo de bloqueo que se produjo o desde o hacia dónde se activó el desvío.

## 6 Funcionamiento del entorno VisualMips32

La interfaz gráfica VisualMips32 para Windows ofrece un entorno de simulación integrado con las herramientas necesarias para la edición, depuración y simulación de un código ensamblador para MIPS. El objetivo principal de este entorno de desarrollo es simplificar la interacción con el simulador y visualizar de forma clara e intuitiva el estado del procesador en cualquier momento a través del cronograma de ejecución de las instrucciones, el contenido de la memoria, los registros del procesador y la representación del pipeline.

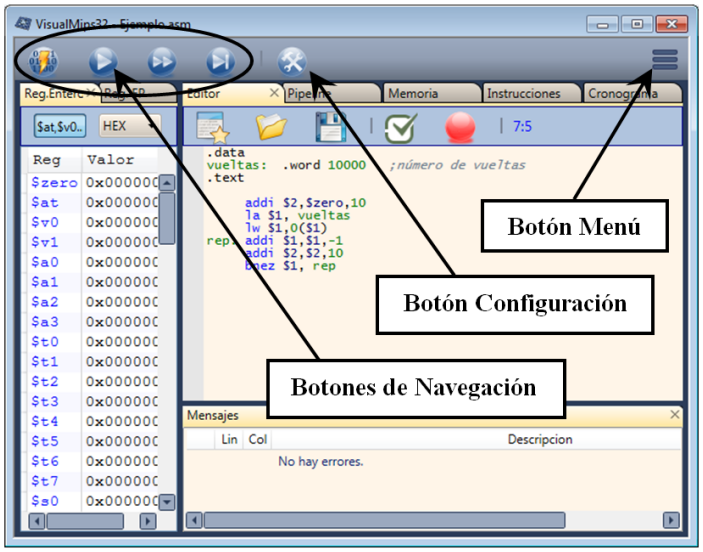


**Figura 5.** Visión general del entorno, la ventana principal

El entorno se compone fundamentalmente de siete ventanas que el usuario puede acoplar a la ventana principal en forma de pestañas o subdividiendo el área de otras ya acopladas. Al iniciar la aplicación las ventanas se encuentran distribuidas en la ventana principal como muestra la Figura 5.

La ventana principal del entorno, que sirve de contenedor para el resto de ventanas, posee los siguientes botones:

- *Botón de Menú*. para acceder a todas las funciones del entorno.
- *Botón de Configuración*. Muestra la ventana de configuración del procesador y simulador.
- *Botones de navegación*. Para reiniciar y avanzar en la simulación ciclo a ciclo, instrucción a instrucción o hasta el final.

### 6.1 Ventana Editor

Esta venta se utiliza principalmente para cargar, editar y ensamblar el código ensamblador a simular. Los programas se guardan con la extensión *asm*.

Como se aprecia en la Figura 6, además de los botones propios de cualquier editor de texto, incluye el botón Ensamblar encargado de detectar los errores del código y mostrarlos por la ventana de mensajes. Si la opción *AutoEnsamblar* de la ventana de configuración está activa los errores se mostrarán automáticamente durante la edición del programa.

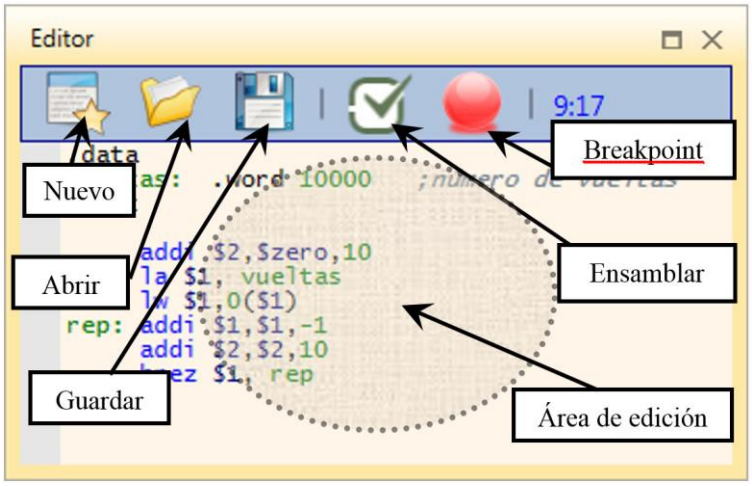


**Figura 6.** Ventana Editor

El entorno tiene soporte para gestionar puntos de interrupción (*breakpoints*) durante la simulación. Desde la ventana Editor se pueden insertar estos puntos de interrupción bien a través del botón *Breakpoint* mostrado en la Figura 6 o haciendo doble clic con el ratón en el margen izquierdo del área de edición. Al comenzar la simulación, los puntos de interrupción que se hayan definido en el editor se transfieren a la ventana Instrucciones.

### 6.2 Ventana Mensajes

Esta ventana se encarga de detallar los errores de ensamblado que presenta el código durante la fase de edición o bien las excepciones lanzadas por el procesador al ejecutar las instrucciones durante la simulación.

### 6.3 Ventana Memoria

Como se aprecia en la Figura 7, esta ventana representa el contenido de la memoria en hexadecimal y permite agrupar las posiciones de memoria formando bytes, medias palabras, palabras o dobles palabras. Tanto el contenido del segmento de código como el del segmento de datos puede ser también modificado directamente por el usuario. Cuando se realiza una escritura en memoria, bien sea por el usuario o por el procesador, la ventana muestra en rojo el último dato que ha sido modificado. A continuación se describen los elementos de la ventana memoria:

- *Selector Dirección de inicio*: permite al usuario acceder rápidamente a una zona determinada de la memoria. En ella puede indicarse, una dirección de memoria en hexadecimal (por ejemplo 0x00400000), o el nombre de un segmento (como puede ser *.text* para ir al comienzo del segmento de código) o bien el nombre de una variable definida en el programa.
- *Selector Bytes por dato*: especifica en cuántos bytes se agrupan los datos contenidos en la memoria. Pueden agruparse por bytes, medias palabras (16 bits), palabras (32 bits) o dobles palabras (64 bits).

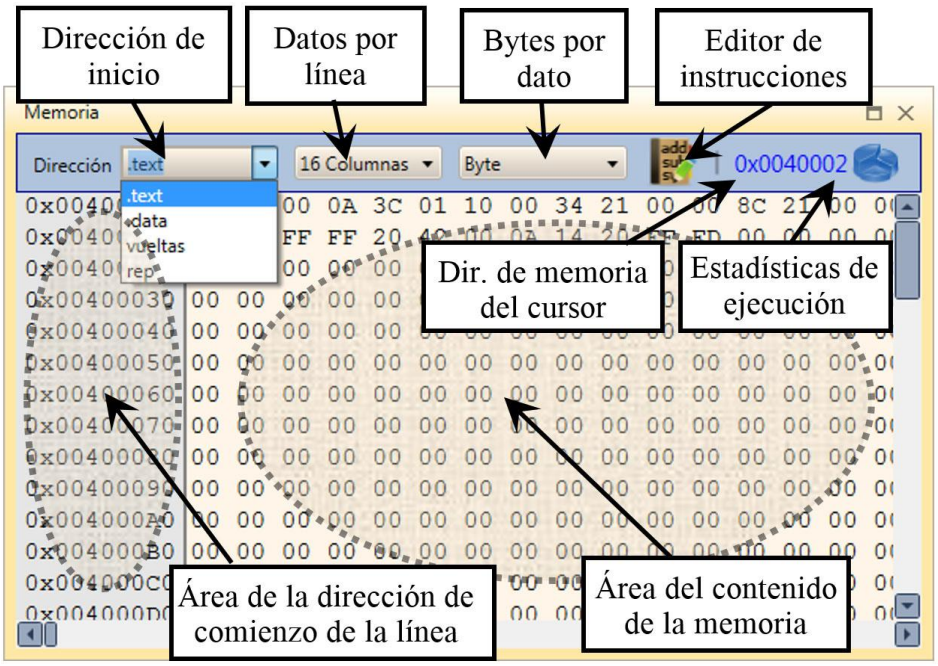


**Figura 7.** Ventana Memoria

- *Selector Datos por línea*: establece cuantos datos se muestran en cada línea del área del contenido de la memoria, siendo el tamaño de cada dato especificado por el selector anteriormente descrito.
- 
- *Botón editor de instrucciones*: al pulsar este botón cuando el cursor se encuentra en el segmento de código, el entorno toma los datos de la memoria sobre los que se encuentra el cursor y los interpreta como el código de una instrucción. A continuación se abre la ventana *Editor de Instrucciones* (Figura 8) que permite modificar el contenido de la memoria en función de la instrucción que el usuario defina.

### 6.4 Ventana Editor de instrucciones

La función principal del editor de instrucciones es codificar instrucciones de forma sencilla con el fin de modificar el programa cargado en memoria durante su ejecución. Por otro lado, proporciona una herramienta didáctica de interés para estudiar los formatos de instrucción del procesador MIPS.

Una vez abierto el editor, el usuario puede modificar directamente los bits del código de la instrucción, especificar el mnemónico de una nueva instrucción o incluso modificar cada uno de los campos que contiene el formato de la instrucción. En función del código de operación que se especifique los campos que forman el formato de instrucción cambiarán automáticamente. Cada una de las alternativas anteriores puede especificarse en base decimal, binaria o hexadecimal.

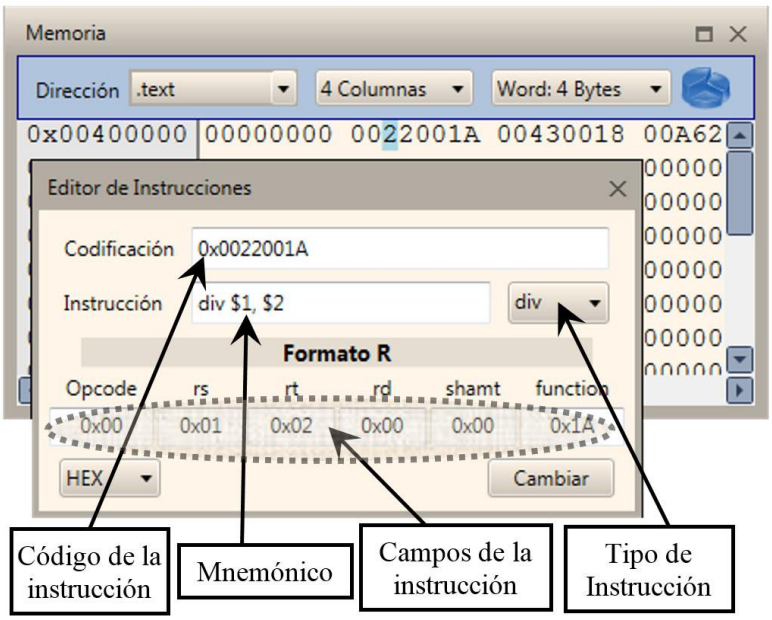


**Figura 8.** Ventana Editor de Instrucciones

### 6.5 Ventanas de edición de registros

Consisten en dos ventanas que visualizan y permiten modificar el contenido de los registros del banco de registros de enteros y de coma flotante durante la simulación.

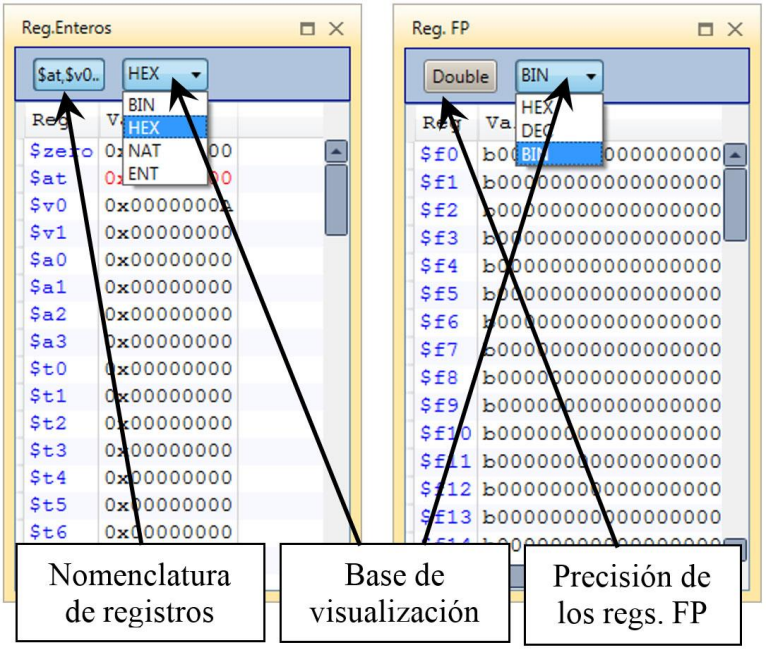


**Figura 9.** (Izquierda) Ventana del banco de registros de enteros; (Derecha) Ventana del banco de registros de coma flotante.

En estas ventanas se encuentran los siguientes elementos:

- *Selector de la base de visualización*: especifica la base en la que se mostrarán los registros: binario, hexadecimal, entero o natural (entero sin signo). Independientemente de cuál sea la base de visualización seleccionada, el usuario podrá modificar el contenido de los registros en cualquier base simplemente especificando el prefijo 0x para referirse a la base hexadecimal, el prefijo b para describir que se trata de base binaria o ninguno de los anteriores para tratar el valor introducido como base decimal.
- *Botón Precisión de los registros en FP*: Cada registro de FP de los 32 que posee el banco puede utilizarse para almacenar un número en coma flotante de simple precisión, o bien pueden agruparse dos a dos para formar registros de 64 bits con el fin de almacenar números en coma flotante de doble precisión. El botón Precisión conmutará entre la visualización del banco como 32 registros de simple precisión o 16 registros de doble precisión.
- *Botón Nomenclatura de registros*: en MIPS32 pueden utilizarse dos nomenclaturas distintas para referirse a los registros, una consistente en especificar el símbolo $ seguido del número que ocupa en el banco de registros (por ejemplo $1), o bien emplear el símbolo $ junto al nombre que

denota la función del registro en el procesador (por ejemplo $t2). La Tabla 1 recoge ambas nomenclaturas, su equivalencia y uso.

### 6.6 Ventana Instrucciones

Lista las instrucciones que se encuentran almacenadas en el segmento de código y las decodifica.

Para cada instrucción de la lista se especifican los siguientes campos:

- *Dirección de memoria donde comienza la instrucción.*
- *Codificación de la instrucción expresada en hexadecimal.*
- *Nemónico de la instrucción.*
- *Texto de la instrucción según aparece en el editor del código de usuario.*
- *Etapa en la que se encuentra cada instrucción durante la simulación.*

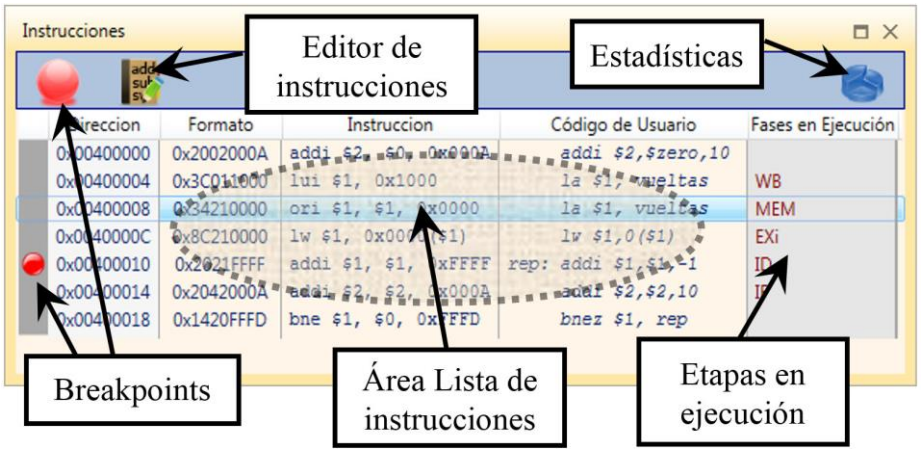


**Figura 10.** Ventana Instrucciones

Esta ventana permite además editar las instrucciones y establecer puntos de interrupción durante la ejecución.

### 6.7 Ventana Pipeline

La arquitectura MIPS32 dispone de varias unidades funcionales de cálculo, unas integradas en el procesador y otras alojadas en un coprocesador independiente (Figura 1), cada una de ellas dedicada a realizar ciertos tipos de operaciones. A continuación se describen las unidades funcionales que posee el MIPS32 y que han sido implementadas en el simulador:

- *Unidad de enteros principal (EXi)*: encargada de las cargas, almacenamientos, saltos y las operaciones ALU enteras excepto las de multiplicación y división. Por ejemplo la instrucción add $1, $2,$3 se realizaría en Exi.

- *Unidad para multiplicación de enteros (EXm)*.
- *Unidad para la división de enteros (EXd)*.
- *Unidad de FP principal (EXfp)*: realiza todas las operaciones de coma flotante a excepción de la multiplicación y la división. Ejemplo *add.s $f2, $f4, $f5*.
- *Unidad para la multiplicación en coma flotante (EXfpm)*.
- *Unidad para la división en coma flotante (EXfpd)*.

Teniendo en cuenta estas unidades funcionales, el simulador representa un pipeline según se muestra en la Figura 11.

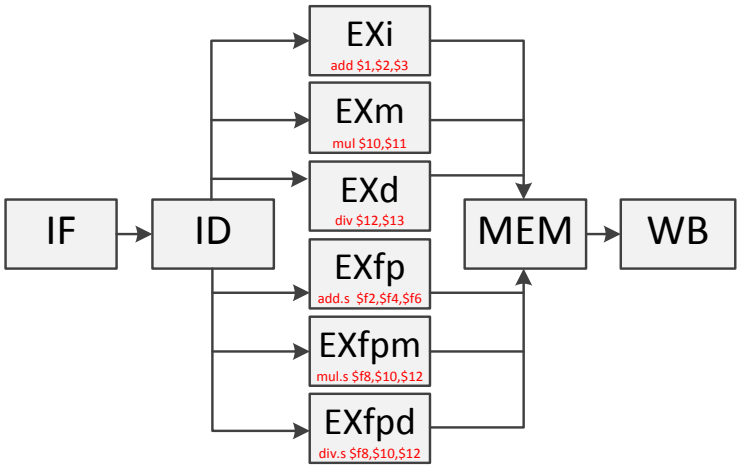


**Figura 11.** Pipeline con las unidades funcionales independientes

Salvo la unidad funcional *EXi*, estas unidades realizan operaciones complejas por lo que suelen precisar de varios ciclos de reloj para realizar el cálculo, así que la etapa EX de tales instrucciones dura más de un ciclo. Además, algunas de estas unidades de cálculo multiciclo pueden estar a su vez segmentadas, esto es, que el cálculo de la operación puede estar dividido en varias subetapas. La ventaja de las unidades segmentadas es que permiten comenzar una nueva operación en cada ciclo de reloj aunque otras anteriores no hayan terminado.

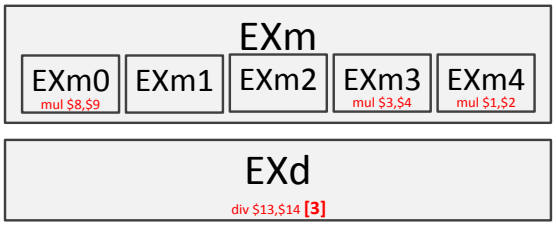


**Figura 12.** (Arriba) Ejemplo de unidad segmentada multiciclo; (Abajo) Ejemplo de unidad no segmentada multiciclo

La Figura 12 muestra una posible forma de representar las unidades segmentadas y no segmentadas multiciclo. El ejemplo de la Figura 12a corresponde a una unidad de la multiplicación segmentada con 5 etapas y que contiene 3 instrucciones en ejecución cada una en subetapas distintas. La Figura 12b muestra una unidad funcional de la división configurada como no segmentada en la que hay una instrucción que se encuentra en el ciclo 3 de su ejecución.

Un ejemplo de la ventana Pipeline se encuentra en la Figura 13. En ella se representa el estado del pipeline mediante cajas siguiendo un esquema similar al mostrado en la Figura 11. Para especificar el estado de cada etapa, el borde del recuadro que representa a la etapa se colorea según el resultado de la ejecución de la instrucción alojada en ella. Un borde de color negro especifica que la ejecución se ha realizado con éxito y otro color representa el tipo del bloqueo según la configuración de colores especificada en la configuración del simulador.

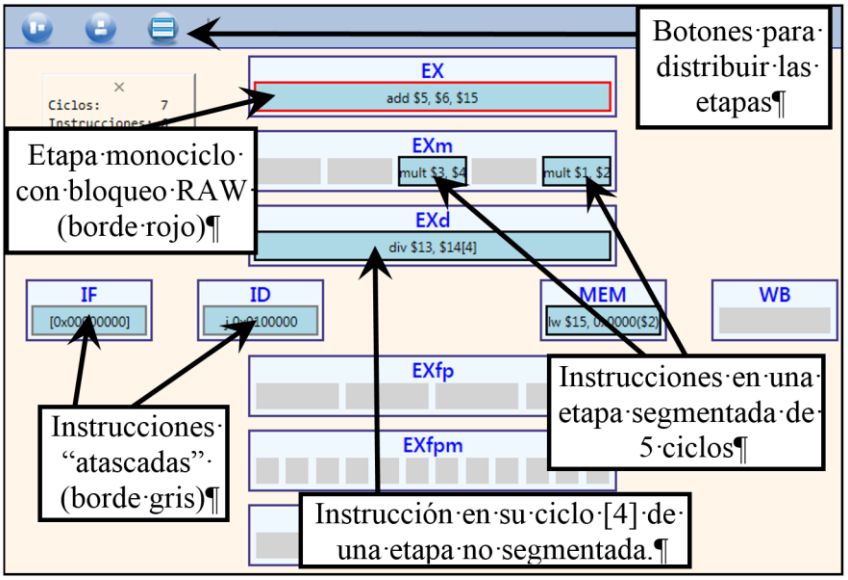


**Figura 13.** Ventana Pipeline

Cada caja que representa a una etapa se puede mover y cambiar de tamaño para adaptarse a las necesidades del usuario. También dispone de botones que simplifican la distribución de las etapas en el espacio de trabajo y reorganizan las subetapas de cada etapa horizontal o verticalmente.

### 6.8 Ventana Cronograma

Una de las ventanas que aporta más información al simular el procesamiento segmentado es la ventana Cronograma. Como puede apreciarse en la Figura 14, en esta ventana se representa el diagrama de tiempos del estado de cada etapa en cada ciclo de reloj durante la ejecución de las instrucciones del código en ensamblador.

Como suele ser lo habitual, el eje de ordenadas indica las instrucciones y las abscisas el ciclo de ejecución. Para cada instrucción se especifica un número de secuencia, que ocupa en la ejecución de la instrucción, la dirección de memoria y su representación en ensamblador. Para cada intersección instrucción-ciclo se muestra el nombre de la etapa y el estado en el que se encuentra. Cuando la etapa ha sido bloqueada se utiliza un código de colores configurable que identifica el tipo de bloqueo. Para una mejor identificación del bloqueo, es posible opcionalmente tachar el nombre de la etapa y añadir un carácter especial junto al nombre. El significado de estos caracteres especiales es el siguiente:

- *Asterisco como prefijo (*)*: denota que la etapa posee un bloqueo tipo lectura tras escritura (RAW). Por defecto se representan en rojo. Ejemplo ~~*EXi~~
- *Asterisco como sufijo (*)*: especifica que la etapa posee un bloqueo tipo escritura tras escritura (WAW) y se representan en verde. Por ejemplo ~~EX*~~.
- *Acento circunflejo (^)*: representa a los bloqueos estructurales y suelen aparecer en color amarillo. Sirva de ejemplo ~~EX^~~.
- *Barra ascendente (/)*. Indica un bloqueo de control y es dibujado en color azul. Por ejemplo ~~IF/~~.

Cuando la instrucción no puede avanzar de etapa debido a la detención del cauce no se utiliza ningún carácter especial, tan sólo se muestra la etapa en gris por defecto y aparece tachada.

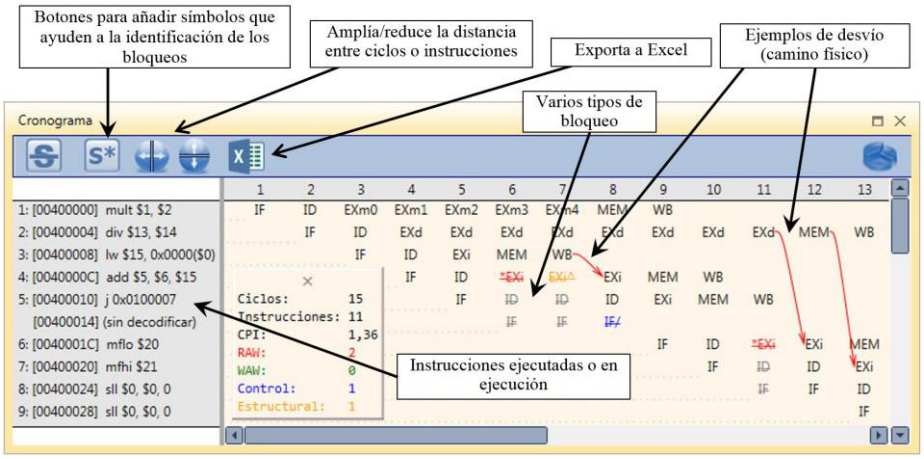


**Figura 14.** Ventana Cronograma

En una etapa multiciclo, la instrucción deberá permanecer en la unidad funcional durante varios ciclos. Si la etapa está segmentada, el cronograma muestra junto al nombre de la etapa el número de la subetapa en la que se encuentra la instrucción. Véase como ejemplo la instrucción 1 que aparece en la Figura 14; en ella puede verse que la instrucción *mult* realizó su fase EX en la unidad de multiplicación de enteros EXm, que está segmentada y que dura 5 ciclos. Dicha instrucción accedió a la subetapa 0 de EXm en el ciclo 3 y fue avanzando a las siguientes subetapas hasta alcanzar la última subetapa en el ciclo 7. En cambio las etapas configuradas como

multiciclo no segmentadas no añaden ningún identificador de la subetapa en la que se encuentran ya que en realidad la subetapa es única, sólo que dura varios ciclos. La Figura 14 muestra un ejemplo de ello en la instrucción 2, la cual entra en la subetapa que posee EXd en el ciclo 4 y permanece en ella hasta pasar a la etapa de MEM.

Para finalizar este artículo, se exponen a continuación los posibles trabajos futuros y las conclusiones.

# 7 Trabajos futuros

Aunque el simulador presentado en este trabajo ya ha alcanzado el fin de su fase de desarrollo según los requerimientos establecidos inicialmente, son innumerables las mejoras que se le están aplicando para conferirle mayor estabilidad, fiabilidad y versatilidad. Además de estas continuas mejoras, claves para conseguir un software de calidad, se han añadido a la lista de tareas por hacer nuevas características y herramientas. Algunas de ellas encaminadas a ampliar el sistema que se simula con nuevos componentes como cachés y BTB; algunas orientadas a representar el sistema con mayor detalle, como es mostrar la ruta de datos a nivel RT, pero sin sacrificar su facilidad de uso.

Por último, en aspectos relacionados con la docencia, este simulador abarca un contenido íntimamente relacionado con la asignatura *Arquitectura de Computadores* de segundo curso de los grados de Ingeniería Informática impartidos por la Universidad de Sevilla. Por consiguiente, se pretende utilizarlo este curso académico en las sesiones prácticas relacionadas con esta materia, así como dejarlo a libre disposición del alumnado para su descarga gratuita y uso para prácticas en casa.. De esta manera, se pretende realizar un análisis de la mejora del rendimiento académico en el alumnado mediante dos mecanismos diferenciados:

- *Encuesta anónima al alumnado*: en la que se evaluarán aspectos más subjetivos de la utilidad del simulador ampliamente detallado en las secciones anteriores. De esta manera se pueden obtener unas conclusiones rápidas sobre la utilidad y fidelidad del simulador, de cara a realizar las mejoras oportunas para el siguiente curso académico.
- *Evaluación de resultados académicos*: este mecanismo pretende aportar unas métricas objetivas acerca de la mejoría de comprensión del procesador MIPS mediante una comparativa entre las calificaciones obtenidas por el alumnado en cursos anteriores y las obtenidas en el curso académico actual.

Mediante la evaluación de las métricas anteriores se espera obtener suficiente información para mejorar aquellos aspectos del simulador que más problemas ocasionen al aprendizaje del alumnado. De esta forma, el simulador será sometido a un ciclo de mejora continua; obteniéndose una versión definitiva del simulador en pocos cursos académicos.

## 8 Conclusiones

El estudio de la arquitectura de computadores puede convertirse en una ardua tarea para profesores y alumnos si no se dispone de un simulador adecuado. En este trabajo se ha descrito algunos de los simuladores académicos más utilizados y se ha presentado un nuevo software de simulación del procesador MIPS32 segmentado orientado a la enseñanza y al aprendizaje de la arquitectura de computadores que se imparte en las universidades. Este simulador implementa más de 80 instrucciones y simula el comportamiento del procesador con capacidad para detectar y resolver los riesgos en la segmentación. Implementa algunas mejoras del procesador MIPS como son los adelantamientos (forwarding) y los saltos retardados. También se ha propuesto una interfaz gráfica muy intuitiva y versátil con el propósito de hacer más atractivo el estudio del procesador a los estudiantes. El simulador está implementado en una librería independiente de la interfaz por lo que podría diseñarse nuevas interfaces de usuario adaptadas a las necesidades de los alumnos.

## Referencias


1. D. A. Patterson and J. L. Hennessy, “Computer Organization and Design - The Hardware/Software Interface”, 3rd ed. Morgan Kaufmann, 2005
2. John L. Hennessy and David A. Patterson, “Computer Architecture: A Quantitative Approach”. San Francisco, CA: Morgan Kaufman, 2007.
3. J. L. S. C. Pereira, “Educational package based on the MIPS architecture for FPGA platforms”, Master Thesis, Faculdade de Engenharia da Universidade do Porto, June 2009. http://repositorio-aberto.up.pt/bitstream/10216/59975/1/000135086.pdf
4. A. Clements, “The undergraduate curriculum in computer architecture”, IEEE Micro, vol. 20, no. 3, pp. 13–21, 2000
5. J. Djordjevic, B. Nikolic, T. Borozan, and A. Milenkovie, “CAL2: Computer aided learning in computer architecture laboratory”, Comput. Appl. Eng. Educ., vol. 16, pp. 172–188, 2008
6. B. Nikolic, Z. Radivojevic, J. Djordjevic and V. Milutinovic, "A Survey and Evaluation of Simulators Suitable for Teaching Courses in Computer Architecture and Organization", Education, IEEE Transactions on , vo1.52, no.4, pp.449-458, 2009
7. H. Oztekin, F. Temurtas, and A. Gulbag, “BZK.SAU: Implementing a hardware and software-based computer architecture simulator for educational purpose”, Proc. 2nd Int. Conf. Comput. Design Appl., pp. 490–497, 2010
8. B. Mustafa, "Modem computer architecture teaching and learning support: An experience in evaluation", International Conference on Information Society (i-Society), pp.411-416, 27-29, 2011
9. W. Yurcik, G. S. Wolffe and M. A. Holiday, “A Survey of Simulators used in Computer Organization/Architecture Courses”, Procs. of Summer Conf. on Computer Simulation, pp. 524-529, Orlando, Florida, 2001.
10. Página web del simulador QtSpim: http://spimsimulator.sourceforge.net

11. Página web del simulator SPIM: http://cs.wisc.edu/~larus/spim.html
12. D. K. Vollmar and D. P. Sanderson, "MARS: An Education-Oriented MIPS Assembly Language Simulator", Procs. of 37th SIGCSE technical symposium on computer science education, 2006.
13. J. Larus, "SPIM S20: A MIPS R2000 Simulator", Computer Sciences Department, University of Wisconsin, Tech. Rep., 1990. http://pages.cs.wisc.edu/~larus/SPIM/spim_documentation.pdf
14. Página web del simulador MARS: http://mars-sim.sourceforge.net/
15. G. C. R. Sales, M. R. D. Araujo, and F. L. C. Padua, "MIPS X-Ray: A Plug-in to MARS Simulator for Datapath Visualization", 2nd Intern. Conf. on Education Technology and Computer (ICETC), 2010
16. Página web del simulador ProcessorSim: http://jamesgart.com/procsim.
17. H. Sarjoughian, Y. Chen, & K. Burger, (2008). "A component-based visual simulator for MIPS32 processors". In Proceedings – 38th Frontiers in Education Conference (FIE), 2008.
18. Página web del simulador MIPS-Datapath: https://github.com/acecil/MIPS-Datapath
19. I. Branovic, R. Giorgi, and E. Martinelli, "WebMIPS: A New WebBased MIPS Simulation Environment for Computer Architecture Education", Workshop on Computer Architecture Education, 31st International Symposium on Computer Architecture, 2004.
20. Página web del simulador EduMIPS64: http://www.edumips.org.
21. Página web del cuestionario sobre impacto de EduMIPS64 en los estudiantes: http://www.diit.unict.it/users/spadaccini/edumips64-survey.html
22. Página web del simulador WinMIPS64: http:// http://indigo.ie/~mscott/
23. P. S. Coe, F. W. Howell , R. N. Ibbett and L. M. Williams, "Technical Note: A hierarchical computer architecture design and simulation environment". ACM Trans. Model. Com. Simul., pp. 431–446, 1998.
24. E. Z. Bem and L. Petelczyc, "MiniMIPS: a simulation project for the computer architecture laboratory". Procs. of the 34th SIGCSE technical symposium on Computer science education, pp. 64–68, 2003
25. B. Nova, J.C. Ferreira and A. Araujo. "Tool to support computer architecture teaching and learning". 1st International Conference of the Portuguese Society for Engineering Education (CISPEE), 2013
26. MIPS32® Instruction Set Quick Reference, Revision 1.01. http://www.imgtec.com/downloads/factsheets/MD00565-2B-MIPS32-QRC-01.01.pdf
27. MIPS® Architecture for Programmers Volume II-A: The MIPS32® Instruction Set, Revision 5.04. http://www.imgtec.com/mips/architectures/mips32.asp
28. MIPS® Architecture for Programmers Volume I-A: Introduction to the MIPS32® Architecture. Revision v6.01. http://www.imgtec.com/mips/architectures/mips32.asp

# Simuladores de Planificadores de Sistemas en Tiempo Real

Francisco J. Aliaga García, Isabel M. Aliaga García,
Joaquín Olivares Bueno[1], Juan C. Gámez Granados[1], José M. Palomares Muñoz[1]

[1] Dpto. de Arquitectura de Computadores, Electrónica y Tecnología Electrónica de la Universidad de Córdoba
Córdoba, España
{i72algaf, i52algai, olivares, jcgamez, jmpalomares}@uco.es

**Resumen.** En este artículo se presenta un simulador desarrollado que permite ejecutar diferentes planificadores de Tiempo Real, como el algoritmo de planificación cíclica, Algoritmo de la Razón Monótona (RMA) y EDF (Earliest Deadline First) para un conjunto de procesos con unos datos dados y muestra los resultados obtenidos. Mediante este simulador se facilita a los alumnos el aprendizaje de los algoritmos de planificación.

**Palabras Clave:** Planificadores, simuladores, sistemas en tiempo real

**Abstract.** This paper presents a simulator that has been developed to allow the execution of scheduling algorithms such as the Cyclic Non-preemptive Executive, Rate-monotonic scheduling (RMS) and Earliest Deadline First (EDF) for a given set of processes with different values and the simulator displays the results. With this simulator, students are able to learn about scheduling algorithms.

**Keywords:** Schedulers, simulator, Real-Time systems, algorithms

## 1 Introducción

Existen múltiples aplicaciones y sistemas que necesitan obtener una respuesta antes de un determinado tiempo límite (denominado comúnmente, *deadline*). A dichos sistemas se les denomina Sistemas con restricciones de Tiempo Real o simplemente Sistemas en Tiempo Real. En estos sistemas se admite que el resultado numérico no sea el óptimo, siempre que el resultado sea suficientemente aceptable y se proporcione a tiempo.

En general, estos sistemas se dividen en un conjunto de tareas, cada una de ellas encargada de dar respuesta a variaciones en el entorno o en el propio estado lógico del sistema. Existe un límite en el número de tareas que se puede ejecutar simultáneamente (dicho límite es igual al número de procesadores o núcleos de procesamiento existentes en el sistema). Cuando se supera dicho máximo ya no es posible la ejecución simultánea real y por tanto, es necesario organizar de alguna manera las tareas, para que se pueda establecer el orden de ejecución de cada una de ellas en cada momento entre los diferentes procesadores. Las reglas que permiten hacer dicho ordenamiento constituyen los algoritmos de planificación.

El sistema operativo y, en particular, el planificador, es quizá el componente más importante de un Sistema de Tiempo Real [1]. Los algoritmos de planificación establecen prioridades a las tareas que permiten al sistema operativo escoger las tareas más relevantes en cada instante. Dichas prioridades pueden ser fijas y no variar a lo largo de la ejecución, o pueden ir variando conforme se van ejecutando las tareas.

Dada la importancia del procesamiento en tiempo real, es materia de estudio en algunas asignaturas. En particular, se puede destacar la asignatura Sistemas en Tiempo Real, perteneciente a los planes de estudios de la titulación de Ingeniería en Informática, Ingeniería en Automática e Ingeniería Industrial y Graduado en Ingeniería Informática de la Universidad de Córdoba (España).

El resto del artículo está estructurado como sigue. En el apartado 2 se hace una breve introducción a la planificación en tiempo real. Continúa el apartado 3 mostrando otros planificadores que podemos encontrar en la literatura. Seguidamente, el apartado 4 nos muestra la aplicación desarrollada junto con sus principales características para finalmente mostrar las conclusiones del trabajo en el apartado 5.

## 2 Planificación de Tiempo Real

En el momento en que hay más de dos tareas que se ejecutan en un determinado sistema hay que plantear algún mecanismo para poder intercambiar las tareas. El planificador es un módulo del S.O. que se encarga de controlar las tareas existentes en el sistema y proporcionarles tiempo de CPU en un determinado instante, gestionando el estado de ejecución en el que se encuentra y decidiendo en cada caso qué tarea debe ejecutarse en cada momento [2].

La planificación en un sistema operativo de propósito general intenta garantizar un uso extensivo de la CPU, equidad en la distribución de acceso al tiempo de ejecución y bajo *overhead*. La planificación de un conjunto de tareas es viable si todas las tareas comienzan su ejecución después de su instante de lanzamiento y todas terminan antes de su *deadline*. La planificación de las tareas tiene dos mecanismos principales:

- **Planificación cíclica no apropiativa**: Las tareas se ejecutan repetitivamente con un determinado periodo. El sistema operativo permite ejecutar cada tarea sin interrupción por parte de las otras tareas hasta que finalice, tras lo cual se ejecuta otra tarea.
- **Planificación apropiativa**: El sistema operativo asigna la CPU a una tarea hasta que la interrumpe y le proporciona la CPU a otra tarea distinta. Se dice que es apropiativa (o expropiativa) porque el S.O. se apropia de la CPU, que estaba en posesión de una tarea. La asignación de una tarea a una CPU puede realizarse por un cierto tiempo conocido apriori o hasta que suceda un bloqueo o llamada al sistema operativo.

La asignación de prioridades a las tareas se puede realizar con planificaciones apropiativas y con no apropiativas, aunque lo más habitual es que se realice con planificaciones apropiativas. Las prioridades pueden asignarse de manera estática (Algoritmo de la Razón Monótona, RMA) o de manera dinámica (EDF).

### 2.1 Planificación cíclica no apropiativa

La planificación cíclica sin prioridad es un mecanismo de planificación sencillo de implementar y calculable apriori. Requiere un conocimiento muy concreto de los tiempos de ejecución de las rutinas y es muy eficiente en términos de bajo *overhead* pero no maneja bien eventos esporádicos ni interrupciones. Su desventaja es que es muy restrictivo, ya que todas las tareas deben tener tiempos de ejecución similares.

Esta planificación se utiliza principalmente en pequeños sistemas empotrados donde se conocen completamente las tareas que requiere el sistema. Para desarrollar un planificador basta con diseñar un bucle que vaya ejecutando una tras otra las tareas.

Hay que definir un ciclo mayor, que coincide con el periodo de repetición de toda la planificación (y es el mínimo común múltiplo de todos los periodos) y un ciclo menor, que es el máximo común divisor de los periodos de todas las tareas.

### 2.2 RMA (Algoritmo de la Razón Monótona)

El Algoritmo de la Razón Monótona (RMA) es uno de los algoritmos de asignación de prioridad más estudiados y es el más utilizado en la práctica. En RMA todas las tareas son periódicas y el *deadline* (relativo, es decir, en cada iteración de la tarea) de cada tarea es igual a su periodo. Las tareas de mayor prioridad pueden interrumpir a las tareas de menor prioridad.

Este algoritmo asigna prioridades ($P_i$) a las tareas inversamente proporcionales a su periodo ($T_i$). La prioridad de cada tarea se calcula de la siguiente forma:

$$\text{Prioridad}_i = \frac{100}{T_i}$$

El esquema de prioridades de RMA es óptimo, en tanto que si un conjunto de tareas es planificable mediante cualquier mecanismo de prioridades fijas, entonces lo será mediante RMA.

**Test de planificabilidad**

El algoritmo de la Razón Monótona posee un mecanismo matemático que permite determinar si un conjunto de tareas es viable utilizando RMA. En 1973, Liu y Layland demostraron que un conjunto de tareas es planificable utilizando RMA si:

Para N procesos, se cumple que:

$$\sum_{i=1}^{N}\left(\frac{C_i}{T_i}\right) < N\cdot\left(2^{\frac{1}{N}}-1\right)$$

Donde Ci es el tiempo de ejecución en el peor caso del proceso i.
Ti es el periodo de repetición del proceso (llamado periodo de lanzamiento).
La sumatoria representa el grado de utilización total de la CPU por el conjunto de procesos.

Este test es solo una condición suficiente y no necesaria. Si se cumple el test, entonces se cumplirán todos los *deadline*. Si el test falla, los *deadline* podrían cumplirse o no, por lo que se deberá hacer el gráfico de *timeline* para comprobar el comportamiento de las tareas y observar si son planificables o no.

### 2.3 EDF (Earliest Deadline First)

El Algoritmo EDF (Earliest Deadline First) es un algoritmo de asignación dinámica de prioridades. Las prioridades de las tareas van modificándose, proporcionando mayor prioridad a las tareas que tienen más cercano su *deadline*. EDF es un algoritmo óptimo en uniprocesadores: si un conjunto de tareas no es viable con EDF, no existe ningún otro algoritmo de planificación que lo haga viable.

Con EDF no es necesario realizar la suposición de que las tareas son periódicas. El resto de suposiciones realizadas para RMA se toman para EDF.

La ventaja de las prioridades dinámicas es que aumentan la flexibilidad, permiten aumentar la prioridad de una tarea en una situación de alerta. Sin embargo, también

tiene algunas desventajas como que el cambio dinámico de prioridades es peligroso porque el comportamiento del sistema es mucho más complejo de predecir y, además, existe el riesgo de bloquear algunas tareas durante demasiado tiempo. [3]

El test de planificación para EDF cuando todas las tareas son periódicas y sus *deadlines* coinciden con sus periodos, es simple: Si la utilización total del conjunto de tareas es menor que 1, el conjunto de tareas se puede planificar en un procesador usando EDF. Sin embargo, si no se cumplen que las tareas sean periódicas o que los *deadlines* no coinciden con los periodos, no existe ningún mecanismo para comprobar la planificabilidad: hay que ejecutarlo y comprobarlo experimentalmente.

## 3 Antecedentes

Existen diversas herramientas de apoyo al análisis de planificabilidad en este tipo de sistemas, destinadas a usuarios profesionales. Las hay tanto comerciales como de libre distribución. A continuación se hará un breve resumen de las principales:

**PerfoRMAx**: Esta herramienta facilita tanto el análisis de planificabilidad como la simulación del comportamiento del planificador. El sistema se especifica de manera tabular y se emplea el modelo de eventos/acciones/recursos para describir las tareas. Estas se especifican mediante su prioridad, tiempo de ejecución, periodo y los recursos compartidos que requieren.

**Rapid RMA** [4]: Conocida originalmente como PERTS, proporciona uno de los entornos de análisis de planificabilidad más completos. Incluye técnicas que admiten políticas del tipo Deadline Monotonic Analysis (DMA), Earliest Deadline First (EDF) o ejecutivos cíclicos, permite el análisis End-to-End monoprocesador y distribuido, Off-line scheduling para CORBA de tiempo real y da soporte para el protocolo de acceso a recursos comunes orientado a objetos DASPCP.

**TimeWiz**: Es una herramienta bastante completa basada en el modelo de eventos/acciones/recursos. Admite entrada de datos tabular y parcialmente gráfica y dispone de una variedad de protocolos para el acceso mutuamente exclusivo a recursos compartidos. Proporciona un diagrama de eventos de salida sobre una línea de tiempos basada en eventos de entrada o en los nodos del sistema.

**TIMES**: Se presenta como una herramienta para el análisis simbólico de planificabilidad y la generación de código ejecutable para sistemas de tiempo real. Se basa en la utilización de un autómata temporizado de comportamiento temporal predecible como bloque básico para la construcción de sistemas de tiempo real. Permite generar el código C ejecutable para el sistema.

**MAST** (Modeling and Analysis Suite for Real-Time Applications): Se trata de un conjunto de herramientas de libre distribución y código abierto, que facilitan análisis de planificabilidad mono-procesador y distribuido, cálculo de holguras, asignación óptima de prioridades, simulación de eventos discretos, etc.

En general, todas las herramientas presentan problemas para su utilización dentro del ámbito docente. Son complicadas de usar, no están adaptados para la comprensión de los alumnos, los entornos de trabajo son complejos, muy pesados computacionalmente y no están adaptados a su uso remoto. Muchos de ellos son de pago y por tanto, no están disponibles de manera gratuita a los alumnos, lo que limita en gran manera su uso docente.

Para un uso educacional es necesario que la herramienta sea sencilla de utilizar, gratuita, con pocos requerimientos computacionales, con posibilidad de uso remoto y adaptada a los contenidos de los cursos de Tiempo Real.

# 4 Simulador

En esta sección se presenta el simulador de planificadores de Sistemas en Tiempo Real que se ha desarrollado, que tiene como principal objetivo facilitar el aprendizaje de los principales algoritmos de planificación. Mediante este simulador, el alumno puede ejecutar el algoritmo de planificación cíclica no apropiativa, RMA y EDF, así como crear conjuntos de procesos, editar sus datos y observar los resultados obtenidos tras aplicar los distintos algoritmos. También tiene la posibilidad de importar y guardar en sus propios archivos los datos relativos a los procesos y dispone de una opción para exportar los resultados obtenidos para un análisis posterior de los mismos.

## 4.1 Características

Mediante el simulador se modelará el comportamiento parametrizado de planificadores tanto apropiativos como no apropiativos habituales en Sistemas en Tiempo Real. Consta de una aplicación web [5] y una aplicación móvil bajo plataforma Android [6]. Sus principales características son las siguientes:

- Su funcionamiento no presenta dificultad en sí mismo, ya que persigue la máxima simplicidad y facilidad de uso. La interfaz es accesible y amigable.
- Ofrece la posibilidad de crear conjuntos de procesos y realizar operaciones con sus datos, como su exportación e importación y, además, permite exportar los resultados obtenidos tras ejecutar un algoritmo (esta opción no está disponible para Android).
- Realiza el cálculo de los tests de planificabilidad.

- Para el algoritmo EDF incluye una opción que permite visualizar las prioridades de los procesos en cada instante de tiempo.
- Permite visualizar el contenido de la página web en dos idiomas: inglés y español.
- El programa es fácilmente ampliable, por lo que se pueden introducir nuevos planificadores en el futuro.
- Permite consultar contenido teórico acerca de los algoritmos de planificación y ejemplos ilustrativos de cada uno de ellos.
- La aplicación web no requiere conexión a Internet, ya que se ha codificado en Javascript como aplicación con ejecución local.

### 4.2 Ejemplo de ejecución

En este ejemplo se muestran los resultados que se han obtenido al ejecutar el algoritmo de planificación RMA en el simulador. En la Tabla 1 se presentan los datos que se han considerado para los procesos.

**Tabla 1.** Datos de los procesos

| Proceso | Inicio | Periodo (T) | T. ejecución (C) | Prioridad (P) | Utilización (U=C/T) |
|---|---|---|---|---|---|
| A | 0 | 5 | 1 | 20 | 0,2 |
| B | 0 | 10 | 3 | 10 | 0,3 |
| C | 0 | 15 | 2 | 6,67 | 0,13 |
| D | 0 | 10 | 2 | 10 | 0,2 |
| **Utilización global de la CPU** | | | | | 0,833 |

#### Test de planificabilidad

Se ha obtenido que el grado de utilización global de la CPU por el conjunto de procesos es de 0,833, mientras que $N(2^{1/N} - 1)$ es $4(2^{1/4} - 1) = 0,757$; por lo que no se cumple el test de planificabilidad. Si el test falla, los *deadline* podrían cumplirse o no. En este caso se debe realizar el gráfico de *timeline* para comprobar el comportamiento de las tareas y ver si son planificables o no.

En la Figura 1 se muestra la interfaz de la aplicación web, que es muy similar a la de la aplicación móvil. En la parte superior se sitúa la barra de menús, que permite acceder a las diferentes secciones y cambiar el idioma, y debajo se localiza el área central de la página, donde aparece el contenido de la sección a la que se haya accedido.

El simulador es la página de inicio y la parte más importante de la aplicación web, donde el usuario puede interactuar con el sistema y obtener unos resultados a partir de los datos que introduce. En primer lugar, se selecciona el algoritmo que se desea ejecutar. Una vez elegido, se desplegará debajo el segundo paso, en el que se mostrará

una tabla con los datos de los procesos. Debajo de la tabla se incluyen distintos botones para importar o exportar los datos de los procesos, así como borrar toda la tabla y ejecutar el algoritmo seleccionado.


**Figura 1.** Interfaz del simulador de la aplicación web.

En la Figura 2 se muestra el tercer paso, que aparece al ejecutar el algoritmo. Para RMA se obtiene una tabla con la prioridad de cada proceso y el grado de utilización de la CPU, la explicación teórica y el resultado del test de planificabilidad, además de la línea temporal obtenida. También es posible exportar los resultados de la ejecución.

Para este ejemplo, el test de planificabilidad no se cumple, por lo que el *deadline* de los procesos podría cumplirse o no, pero al realizar el gráfico de *timeline* se puede ver que el conjunto de procesos es planificable y así se indica en el resultado.

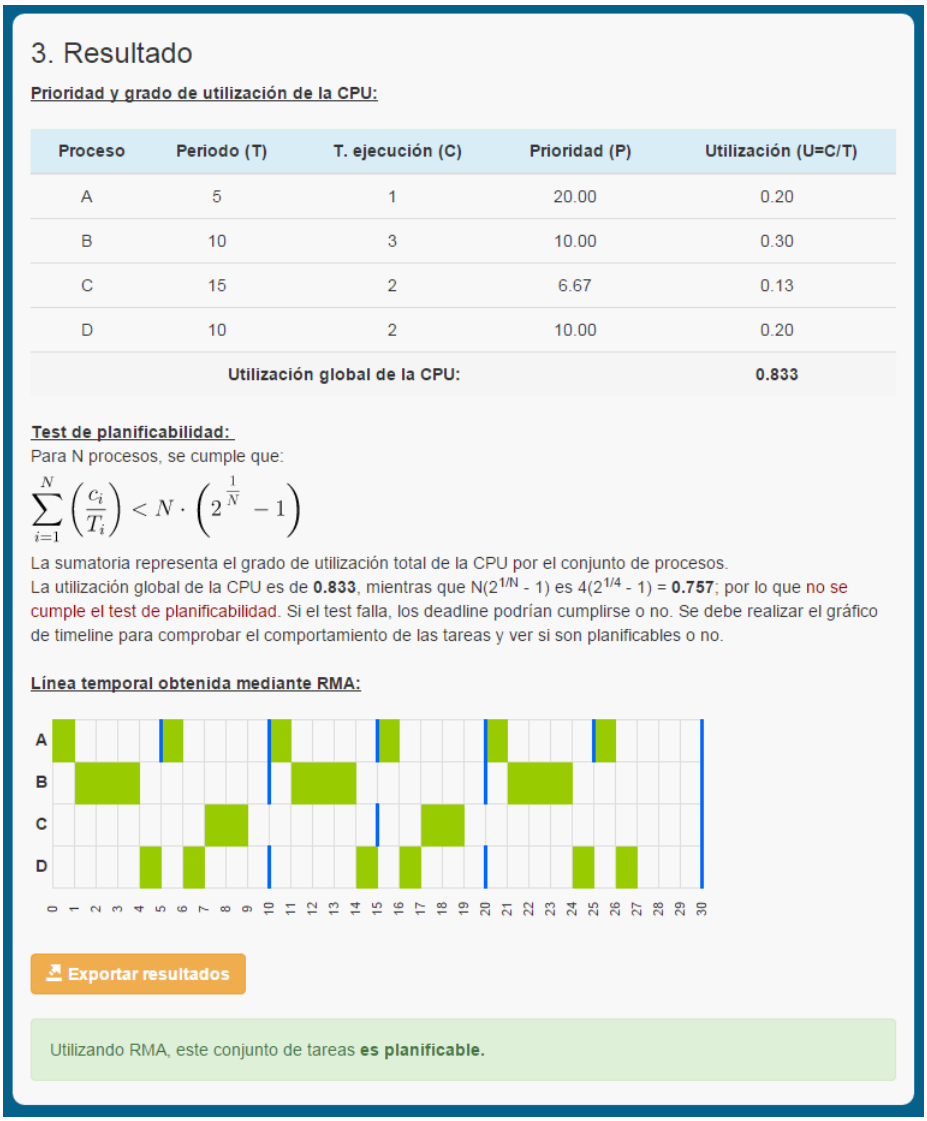


**Figura 2.** Visualización de resultados al ejecutar un algoritmo.

## 5 Conclusiones

En este artículo se han descrito las características novedosas que presenta el simulador desarrollado, que lo diferencian de otras herramientas que se pueden encontrar en el mercado. Esta herramienta es idónea para ser utilizada en el ámbito

académico, ya que facilita al profesor su labor docente, los alumnos pueden usarla para corregir sus propios ejercicios y los resultados obtenidos se pueden interpretar fácilmente.

La aplicación web y la aplicación Android son similares en cuanto a resultados aunque tienen pequeñas variaciones. La versión web, al estar codificada como una aplicación con ejecución local mediante lenguaje Javascript, permite su ejecución en múltiples entornos sin necesidad de conexión con Internet.

La exportación de la simulación permite un análisis más pormenorizado de los resultados utilizando otros mecanismos externos (paquetes estadísticos, hojas de cálculo, etc.) que consiguen realizar el cálculo de diferentes parámetros de interés, como puede ser el *jitter* de las tareas entre repeticiones, holgura de los periodos para interrupciones o eventos, etc.

Dentro de la línea seguida en el desarrollo, se puede contemplar la introducción de nuevos planificadores en el futuro, ya que se ha tenido en cuenta que la actualización y el mantenimiento del software no resulte difícil. También se ha estimado interesante la creación de componentes adicionales que permitan analizar los resultados obtenidos, así como incluir distintos estilos que modifiquen mediante CSS el diseño de la web para mejorar su accesibilidad.

## Referencias

1. Stallings, W. Sistemas Operativos. Traducido por Dodero Beardo, J.; Torres Franco, E; Katrib Mora, M. Revisión Técnica de Joyanes Aguilar, L. 2ª ed. Madrid. Prentice-Hall. 1997. 732 p. ISBN: 84-89660-22-0.
2. Palomares Muñoz, J.M. Tema 6: Planificación de Tiempo Real. Sistemas en Tiempo Real. Universidad de Córdoba.
3. Aldea Rivas, M. Planificación de Tareas en Sistemas Operativos de Tiempo Real Estricto para Aplicaciones Empotradas. Universidad de Cantabria [en línea]. [Última consulta: 4 de abril 2015]. Disponible en: http://marte.unican.es/documentation/tesis-mario.pdf
4. Tri-Pacific Software Inc. Rapid RMA [sitio web]. [Última consulta: 4 de abril 2015]. Disponible en: http://www.tripac.com/rapid-rma
5. Aliaga García, F.J.; Aliaga García, I.M.; Palomares Muñoz, J.M.; Gámez Granados, J.C. Simulador de Planificadores de Sistemas en Tiempo Real [sitio web]. [Última consulta: 6 de abril 2015]. Disponible en: http://www.uco.es/~el2pamuj/simuladorSTR/
6. Aliaga García, F.J.; Aliaga García, I.M.; Palomares Muñoz, J.M.; Gámez Granados, J.C. App Android de Simulador de Planificadores de Sistemas en Tiempo Real [App Android]. [Última consulta: 6 de abril 2015]. Disponible en: http://www.uco.es/~el2pamuj/simuladorSTR/SimuladorSTR.apk

# Una Orquesta Sinfónica como Ejemplo de Aplicación de un Sistema Empotrado Distribuido

Franklin Parrales*, Alberto A. Del Barrio[+], Guillermo Botella[+]

Departamento de Arquitectura de Computadores y Automática, Facultad de Informática de la Universidad Complutense de Madrid
*fparrale@gmail.com, +{abarriog, gbotella}@ucm.es

**Resumen.** El presente artículo trata sobre el diseño e implementación de una orquesta sinfónica distribuida haciendo uso del paquete de Lego Mindstorms, como proyecto final enmarcado dentro de la asignatura Sistemas Empotrados Distribuidos. En esta contribución se aplican los conocimientos obtenidos en dicha asignatura, en la que se fomenta la aplicación de los mismos para la realización de proyectos novedosos. En este artículo se describen el diseño, las diversas tecnologías evaluadas y la implementación final.

**Palabras Clave:** Sistemas Empotrados Distribuidos, Proyectos, Mindstorms, lejOS, RNDIs.

**Abstract.** This paper discusses the design and implementation of a distributed symphony orchestra using the Lego Mindstorms package, as a final project belonging to the Distributed Embedded Systems subject. In this contribution, the knowledge achieved during the subject is applied. It must be noted that the application of the studied contents to create novel projects is greatly encouraged. In this paper the design, the evaluation of several technologies, as well as the final implementation, are presented.

**Keywords:** Distributed Embedded Systems, Projects, Mindstorms, lejOS, RNDIs.

# 1 Introducción

La asignatura Sistemas Empotrados Distribuidos (SED) es una materia que se ubica en el Máster de Ingeniería Informática de la Universidad Complutense de Madrid (UCM) [1], implantado en el curso 2013/2014. Dicho máster está orientado a dotar a los estudiantes de conocimientos adicionales que les haga más competitivos en el mundo laboral. Con el fin de obtener estas habilidades, las asignaturas del máster, y en concreto SED, basan gran parte de su evaluación en la realización de múltiples prácticas y proyectos que permitan a los alumnos mejorar sus capacidades. Si bien las prácticas se realizan sobre la placa de desarrollo S3CEV40 [2], el proyecto queda totalmente abierto a la decisión del alumno. De esta manera, los estudiantes pueden investigar otras plataformas que les permitan desarrollar conocimientos en el área de la *Computación Ubícua* o *Pervasive Computing* [3-5].

**Tabla 1.** Competencias dentro de la asignatura Sistemas Empotrados Distribuidos.

| Tipo | Código | Competencia |
|---|---|---|
| Generales | MCG8 | Capacidad para la aplicación de los conocimientos adquiridos y de resolver problemas en entornos nuevos |
| Específicas | MCETI8 | Capacidad de diseñar y desarrollar sistemas, aplicaciones y servicios informáticos en sistemas empotrados y ubicuos |
| Básicas | MCB6 | Poseer y comprender conocimientos que aporten una base u oportunidad de ser originales en el desarrollo y/o aplicación de ideas |
| Transversales | MCT4 | Capacidad de razonamiento crítico como vía para mejorar la generación y desarrollo de ideas en un contexto profesional |

La Tabla 1 muestra un resumen de algunas de las competencias adquiridas en SED. En esta tabla puede observarse que dos de los objetivos principales son potenciar la aplicación de los conocimientos adquiridos así como la originalidad del alumno. Como muestra de realización de dichos objetivos, en este artículo se presenta una plataforma capaz de implementar una orquesta de forma distribuida, basándose en el hardware comercial LEGO Mindstorms EV3 [6]. Gracias a este proyecto, sería posible reforzar el proceso de aprendizaje de varios conceptos musicales compás y tiempo.

La música, a lo largo del tiempo, ha tenido grandes aportaciones. Grandes compositores han escrito obras como “El Mesías” de Handel, las sinfonías de Beethoven, o la “Marcha Turca” de Mozart. Muchas de estas creaciones se presentan como obras para un conjunto de instrumentos y voces. Para entender la expresión musical, entonación, matiz, y otros factores que el compositor deja apuntado en sus partituras se requieren estudios de varios años. Gracias a este proyecto, cualquier aficionado a la música puede ver cómo interactúan los diversos elementos de una orquesta en tiempo real. Además, el usuario puede incluso interactuar con los componentes de la orquesta por medio de órdenes sencillas como silenciar o subir/ bajar volumen.

La literatura ofrece varios trabajos en este campo, como una banda para entonar diversos instrumentos musicales, cuyo proyecto se llama “the NeXT blues”[7], o un proyecto del California Institute of Arts llamado “CalArts KarmetiK Machine

Orchestra"[8]. Ambos casos están netamente orientados a ejecutar sonido en instrumentos reales, y cuentan con un presupuesto bastante alto, mientras que en este artículo se pretende ofrecer una solución con un coste más asumible.

El resto del artículo se organiza de la siguiente manera: la Sección 2 presenta un estudio de las plataformas hardware disponibles para los propósitos del proyecto; en la Sección 3 se presenta el diseño del proyecto y en la 4 detalles más concretos de implementación; la Sección 5 ofrece nuestras conclusiones y posibles líneas de trabajo futuro.

## 2 Herramientas de hardware disponibles

De acuerdo a lo estudiado en la asignatura de Sistemas Empotrados y Distribuidos (SED), una de las ventajas de diseñar estos sistemas es que cada elemento posee su procesamiento autónomo y puede cooperar con otros elementos en un propósito común [9]. Este modo de comportamiento es muy similar a lo que ocurre en el entorno de orquesta musical planteado anteriormente. Por tanto, es preciso determinar qué elementos compondrán el sistema final. Es por ello que previamente al desarrollo se hizo un estudio de posibles componentes hardware:

- Arduino [10]:
  - Es la opción más barata, el kit básico cuesta la cuarta parte del precio de un kit de Lego Mindstorms EV3 [11]. Se pueden adquirir componentes extras a precios muy baratos.
  - Arduino es una plataforma open-source tanto hardware como software. Además, se programa con el lenguaje C, lo cual facilita su programación.
  - Sin embargo, la desventaja es que sería preciso comprar placas de expansión y varios componentes compatibles para poder realizar el proyecto. Por tanto, el tiempo de búsqueda, montaje y configuración es un inconveniente muy importante. Además, la compra de estos componentes elevaría el coste final de la plataforma, convirtiéndose en una opción no tan económica con respecto al kit de Lego. Problemas similares se experimentarían con otras placas como Raspberry Pi [12].

- Lego Mindstorms EV3 [6]:
  - Dispone de un equipo completo para realizar infinidad de proyectos de forma sencilla. Este proceso se basa en la unión de varios bloques de plástico (Lego) y la programación del bloque principal o *ladrillo EV3*. Este ladrillo contiene un ARM9 de 300MHz, modelo Sitara AM1808 (Texas Instruments), que hay que programar.
  - A la facilidad de montar prototipos hardware integrando componentes tales como sensores y servomotores, se añade la facilidad de programación, ya que existen varios entornos de desarrollo gráficos.
  - Además de los entornos de programación, existen varias *Application Programming Interfaces* (APIs) que dan soporte a las funcionalidades básicas

de los elementos del sistema, por ejemplo la rotación de un servomotor o la captación de la información por medio de un sensor.

Finalmente, evaluando los recursos disponibles y el tiempo máximo de desarrollo del proyecto (2 meses), se optó por la implementación basada en Mindstorms EV3.

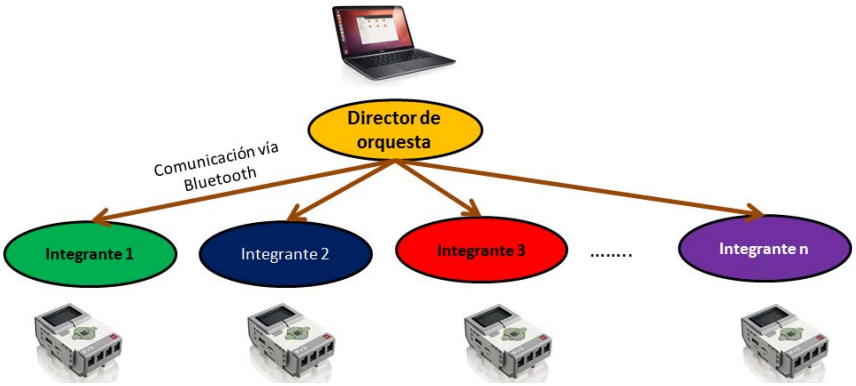


**Figura 1**. Diagrama de asignación de roles en la orquesta

## 3 Diseño del sistema

En esta sección se hablará del diseño de la orquesta. Primeramente introduciremos a los actores, o integrantes de la orquesta, y en segundo lugar ofreceremos el modelado del funcionamiento del sistema.

### 3.1 Integrantes de la orquesta

Distinguimos dos roles en la orquesta:

- Director. Es quien se encarga de coordinar al resto de componentes. Será implementado en el computador.
- Integrante o músico. Todos los músicos se comunican con el director. Cada músico, implementado solamente por un ladrillo EV3, se encarga de tocar una partitura enviada por el director. Su propósito es simular la ejecución de un instrumento musical dentro de la orquesta. Para mejorar la vistosidad del sistema, además se ha dotado de movimiento a los músicos, gracias a piezas y componentes del Mindstorms.

El reparto de funcionalidades queda expresado en la Figura 1, donde se aprecia que el rol de director de orquesta lo ejecutará el computador, quien se comunicará vía Bluetooth con cada ladrillo EV3 para enviarle las órdenes.

### 3.2 Modelado de la orquesta

El diagrama de estados de la Figura 2 muestra el modelo distribuido de la orquesta. Éste es un ejemplo claro de aplicación de las metodologías de diseño aprendidas en la asignatura Sistemas Empotrados Distribuidos.

Como puede observarse, el autómata se compone de los siguientes estados:

- Estado inicial: cada componente de la orquesta permanece en este estado hasta que el director no envía la señal de inicio de ejecución y un tiempo de reloj para sincronizarse. Si la conexión se establece, se pasa al estado de conexión. Previamente ha sido cargada la partitura (secuencia de notas musicales) a cada integrante de la orquesta.
- Conexión con Integrante: el integrante ya ha recibido la partitura y el tiempo de reloj enviados por el director, y está atento a la recepción de la señal de inicio de ejecución de la obra. Si se recibe la señal se pasa al estado de ejecución.
- Ejecución: el integrante de la orquesta procede a ejecutar cada nota y compás de la partitura enviada en el primer estado. Siempre está atento a silenciar su ejecución cuando el director así se lo ordene. Si esto llegase a ocurrir pasa al estado de silencio, si no, continúa hasta la finalización de la obra, en cuyo caso envía señal a director de que ha culminado su ejecución con éxito y se pasa al estado final.
- Silencio: el director envía señal de silencio si él lo desea, y el integrante sigue la ejecución de la partitura pero sin volumen. Una vez el director vuelve a enviar señal de fin de silencio, pasa al estado ejecución.
- Fin de la obra: el director recibe las señales de fin de la obra de cada uno de sus integrantes.

En el diseño se ha considerado el envío de señal de reloj por parte del director para que cada integrante esté sincronizado con todos los demás. Por otra parte, el envío de la partitura solo se hace en el estado inicial, dado que no sería una buena opción enviar compás por compás, ya que podrían saturarse las comunicaciones con el director.

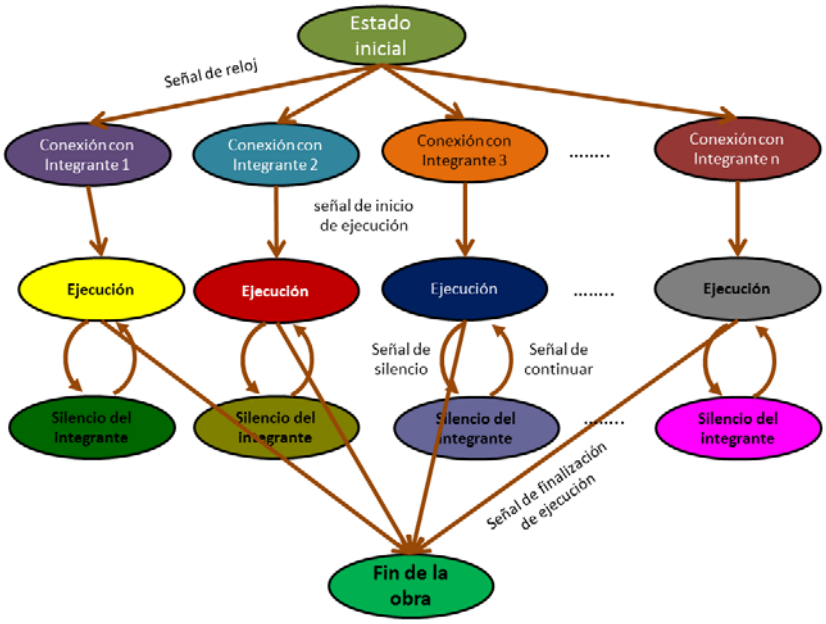


**Figura 2**. Diagrama de estados de la orquesta

## 4 Implementación

En esta sección se describirán los detalles concretos de la implementación, comentando en profundidad el entorno de trabajo utilizado, la configuración y la detección de los integrantes de la orquesta.

El mayor inconveniente que presenta el lenguaje de programación del kit, es la falta de funciones de alto nivel que permitan ejecutar acciones complejas. Por ello es necesario elegir alguna librería y entorno de programación (se ha elegido eclipse con Java y el API de LejOS), pues a diferencia del software que viene por defecto, permite escalar la solución propuesta fácilmente, y combinarla con implementaciones de algoritmos de visión por computadora u otros que se puedan usar en un futuro.

### 4.1 Librerías

Entre las librerías que permiten programar el kit, se han evaluado las siguientes opciones:

1. BrickOS [13]: es un sistema operativo alternativo que da soporte a los bloques RCX de Lego Mindstorms. Gracias a las librerías proporcionadas por BrickOS, los bloques podrían programarse en C, C++ y ensamblador. BrickOS está

soportado en la mayoría de las distribuciones de Linux y en Windows (por CYGWIN), usando los compiladores gcc o gcc++.
2. leJOS[14]: a diferencia de BrickOS, leJOS no requiere instalar un sistema operativo que reemplaza el firmware de los bloques Lego RCX, NXT o EV3, sino que instala una máquina virtual de Java, lo que permite programar los bloques en este lenguaje.
3. Not Quite C [15]: es un lenguaje con una sintaxis parecida a C, que posee diversas librerías para dar soporte a los bloques RCX de Lego. La principal desventaja es su incompatibilidad con los bloques EV3, y la necesidad de de un entorno de desarrollo propio, diferente al del kit del EV3. Not Quite C está disponible para Mac OS y Windows, y utiliza como lenguaje de programación una versión propia de C.
4. EV3 API para WinRT, Windows Phone 8 y Windows desktop [16]: provee de mecanismos de comunicación y control entre los ordenadores y el Mindstorms. Es una solución disponible para aplicaciones .net y permite controlar motores y obtener información de sensores del robot. Se puede usar acompañado de SignalR [17] para manejar desde un web Service a nuestro dispositivo EV3.

De entre estas opciones previamente descritas se ha seleccionado leJOS, dado que tanto BrickOS como Not Quite C todavía no tienen soporte para la nueva versión del EV3. El API de EV3 para .net hace necesario el uso de dicho framework privativo. Además, dado que lejOS está basado en la máquina virtual de Java, hereda los beneficios de este lenguaje, como son: la programación orientada a objetos, multihilos, recursión, sincronización, excepciones, arrays multidimensionales.

### 4.2 Configuración del entorno

Para configurar el entorno de desarrollo, se ha instalado la versión embebida de Java “SE Embedded 8” [18-19] en el ladrillo del EV3, a través de una tarjeta de memoria microSD insertada en el propio bloque. Además, se han instalado las librerías de leJOS en el computador en el que posteriormente se han desarrollado las aplicaciones que implementarán los componentes de la orquesta.

Por último, ha de notarse que para implementar la comunicación inalámbrica entre los ladrillos EV3 (músicos de la orquesta) y el ordenador (director de la orquesta) se ha utilizado Bluetooth, mediante la instalación del driver RNDIS [20].

### 4.3 Encontrando los dispositivos existentes en la red local

Para realizar el descubrimiento de los dispositivos activos dentro de una red, y que formarán parte de la orquesta, se hace uso de la clase *BrickFinder*. Dicha clase posee métodos para encontrar los ladrillos mediante descubrimiento de vecinos en la red.

El método *discover()* se puede utilizar para obtener una lista de todos los ladrillos EV3 remotos que se encuentran actualmente visibles. Por ejemplo, el siguiente código realiza el descubrimiento de vecinos en la red local y para cada ladrillo EV3 ordena emitir un pitido.

```
BrickInfo[] bricks = BrickFinder.discover();
for(BrickInfo info: bricks) {
   Brick brick = new RemoteEV3(info.getIPAddress());
   brick.getAudio().systemSound(0);
}
```

**Figura 2**.Código para encontrar integrantes de la orquesta.


**Figura 3**. Sistema implementado con dos integrantes de la orquesta.

### 4.4 Implementación final

La orquesta Lego puede observarse en la Figura 4. Aunque el diseño se generalizó para N integrantes de la orquesta, el proyecto presentado consta de dos ladrillos EV3 y de un computador, que ejercerá de director. El director utiliza un sistema operativo Windows 8, el cual es compatible con la versión del driver RNDIS, anteriormente comentado.

A fin de dotar de una mayor vistosidad y trabajar con más elementos del kit Lego, los músicos disponen de movilidad en línea recta, como puede observarse en la Figura 4. El video demostrativo de la implementación se puede visualizar en [22]. Ha de notarse que el funcionamiento descrito en este artículo es solo aplicable al ladrillo EV3. El movimiento adicional que se observa en el vídeo se ha conseguido mediante piezas y componentes del kit Mindstorms. Dicho movimiento se implementó con el fin de trabajar con más elementos del kit, aunque no era la propuesta principal en este trabajo.

## 5 Conclusiones

En este artículo se ha descrito el diseño e implementación de una orquesta sinfónica distribuida. Este proyecto se realizó durante la asignatura de Sistemas Empotrados Distribuidos, dentro del Máster de Ingeniería Informática ofertado por la Universidad Complutense de Madrid. En concreto, se hizo uso del paquete de robótica Mindstorms de Lego con la librería LejOS de Java. Por tanto, este proyecto ilustra cómo aplicar las competencias adquiridas con la asignatura, que fomentan el aprendizaje práctico del alumno así como el desarrollo de sus propias ideas.

De cara al futuro, el sistema podría mejorarse al hacer uso de la conexión WiFi en lugar de Bluetooth, dado que se puede lograr una autonomía de señal de hasta 300 metros, mucho mayor que los 10 metros que conseguimos con Bluetooth. Además, el sistema podría complementarse añadiendo la capacidad de reconocer notas musicales de forma visual, al leerlas de una partitura, o permitiendo la interacción de los componentes con el director de la orquesta por medio de la voz humana.

## Agradecimientos

Queremos agradecer al grupo de investigación GreenDISC [21] por el préstamo de uno de los ladrillos EV3 para la realización del proyecto.

## Referencias

1. Máster en Ingeniería Informática de la Universidad Complutense de Madrid, http://informatica.ucm.es/estudios/2014-15/master-ingenieriainformatica
2. Embest S3CEV40 EVB User Guide, http://www.vas.co.kr/products/support/S3CEV40_UserGuide.pdf
3. Chtourou, S.; Kharrat, M.; Ben Amor, N.; Jallouli, M.; ABID, M., "Evolution of robot programming, towards the Ubiquitous Computing era," *Individual and Collective Behaviors in Robotics (ICBR), 2013 International Conference on* , vol., no., pp.44,48, 15-17 Dec. 2013
4. Daoqing Sun, "Researches to the trusted ubiquitous computing," *Electronics, Communications and Control (ICECC), 2011 International Conference on* , vol., no., pp.433,437, 9-11 Sept. 2011
5. Kortuem, G.; Kawsar, F.; Fitton, D.; Sundramoorthy, V., "Smart objects as building blocks for the Internet of things," *Internet Computing, IEEE* , vol.14, no.1, pp.44,51, Jan.-Feb. 2010
6. Rollins, Mark. Beginning Lego Mindstorms Ev3. Apress, (2014).
7. "The NeXT blues" on LEGO MindStorms NXTs, Disponible en youtube a través de www.youtube.com/watch?v=6bEk0bkaOAY. Accedido: 18/02/2015.
8. Kapur, Ajay, Michael Darling, Dimitri Diakopoulos, Jim W. Murphy, Jordan Hochenbaum, Owen Vallis, and Curtis Bahn. "The machine orchestra: An ensemble of human laptop performers and robotic musical instruments." Computer Music Journal 35, no. 4: 49-63. (2011).
9. A. Burns, A. J. Wellings, and A. Burns, Real-time systems and programming languages: Ada 95, real-time Java, and real-time POSIX. London; New York: Addison-Wesley, 2001.
10. “Arduino”: http://www.arduino.cc/. Accedido: 18/02/2015.

11. E. Martínez de Carvajal, 150 proyectos con LEGO Mindstorms: tecnología, instrumentación, robótica., Premiá de Dalt, Barcelona: E. Martínez, (2014).
12. C. Severance, Eben Upton: Raspberry Pi, Computer 46 (10) (2013) 14-16.
13. Hundersmarck, Christopher, Charles Mancinelli, and Michael Martelli. "Viva la brickOS." Journal of Computing Sciences in Colleges 19, no. 5: 305-307. (2004).
14. “LeJOS, Java for Lego Mindstorms”, http://www.lejos.org/. Accedido: 18/02/2015.
15. “Not Quite C”, http://bricxcc.sourceforge.net/nqc/. Accedido: 18/02/2015.
16. “Lego mindstorms EV3 api”, http://legoev3.codeplex.com/documentation#GettingStarted. Accedido: 18/02/2015.
17. “SignalR”, http://www.asp.net/signalr/overview/getting-started. Accedido: 18/02/2015.
18. “Installing leJOS”, http://sourceforge.net/p/lejos/wiki/Installing%20leJOS/. Accedido: 18/02/2015.
19. “Java for LEGO® Mindstorms® EV3”, http://www.oracle.com/technetwork/java/embedded/downloads/javase/javaseemeddedev3-1982511.html. Accedido: 18/02/2015.
20. Microsoft, “RNDIS”, http://msdn.microsoft.com/enus/library/windows/hardware/ff569967(v=vs.85).aspx. Accedido: 18/02/2015.
21. GreenDISC, http://greendisc.dacya.ucm.es/. Accedido: 18/02/2015.
22. Vídeo demostrativo de la orquesta distribuida, https://www.youtube.com/watch?v=c00BBB-Gwao

**DESDE EL PUPITRE**
(Experiencias de estudiantes)

# Aplicación de las materias de Ingeniería de Computadores en la mejora de los Algoritmos Meméticos y Metaheurísticas en general

Fernando Palacios López, Jorge Chamorro Padial

Estudiantes del Grado en Ingeniería Informática

{ferpalacios, jorgechp}@correo.ugr.es

**Resumen.** En el presente artículo analizamos la aplicación de los conocimientos adquiridos en las diferentes asignaturas relacionadas con la Ingeniería de Computadores que componen el Grado en Ingeniería Informática con el uso de metaheurísticas que se imparten en la especialidad de Computación y Sistemas Inteligentes, haciendo especial enfoque en los Algoritmos Evolutivos. Nos centraremos en los Algoritmos Meméticos, comparando su rendimiento frente a los Algoritmos Genéticos. Finalmente, se comprobará la mejora del rendimiento de una metaheurística cuando se aplica Programación Paralela.

**Palabras Clave:** computación, metaheurísticas, algoritmos meméticos

**Abstract.** In this paper, we analyze how to apply the content of many subjects of Computer Engineering to those metaheuristic that are imparted in Computing and Intelligent Systems mention of Computing Engineering degree. The purpose of this analysis is to make emphasis on the importance of applying Computer Engineering techniques as a transversal knowledge of the degree. We focus on Evolutionary Algorithms, analyzing Memetic Algorithms and comparing their perfomance against Genetic Algorithms.

**Keywords:** computing, metaheuristics, memetic algorithms

## 1 Introducción

Metaheurísticas es una asignatura del tercer curso del grado Grado en Ingeniería Informática, enmarcada en la especialidad de Computación y Sistemas Inteligentes, siendo obligatoria para obtener la especialidad.

La asignatura, como su propio nombre indica, estudia diferentes metaheurísticas utilizadas en el ámbito de la Inteligencia Artificial para resolver problemas de diversa índole. El temario se estructura por temas, y en cada uno de estos temas se estudian diferentes algoritmos, según se expone a continuación [1].

- **Algoritmos de Búsqueda Local Básicos [2]:** Basan su estrategia en un estudio local del espacio de búsqueda, es decir, analizan las soluciones incluidas en el entorno de la solución actual. Dichas soluciones son denominadas soluciones vecinas.
  Estos algoritmos presentan el inconveniente de ofrecer como solución óptimos locales, que pueden llegar a estar muy alejados del óptimo global del problema.

- **Enfriamiento Simulado [3]:** Basados en la termodinámica, los algoritmos de Enfriamiento Simulado realizan una búsqueda por entornos en la que permiten movimientos a soluciones peores, con el fin de evitar finalizar en un óptimo local. Estos "movimientos de empeoramiento" se realizan de un modo controlado según un criterio probabilístico.

- **Búsqueda Tabú [4]:** Los algoritmos de Búsqueda Tabú generan entornos restringidos para evitar la exploración de zonas del espacio de búsqueda previamente visitadas. Además, implementan mecanismos de reinicialización para intensificar la búsqueda sobre zonas prometedoras y diversificar la búsqueda, visitando nuevas zonas. También permiten movimientos de empeoramiento para escapar de óptimos locales.

- **Métodos Basados en Trayectorias Múltiples [5]:** El funcionamiento de los algoritmos de este bloque consiste en la generación de una solución inicial y posteriormente en una etapa de mejora de dicha solución. Los algoritmos principales son GRASP, ILS y VNS.

- **Metaheurísticas Basadas en Poblaciones:** Estas metaheurísticas se encuentran dentro de la rama de la computación evolutiva, que recrea los mecanismos de la evolución biológica para obtener la solución a los problemas. Podemos destacar en esta clase los Algoritmos Genéticos, los Algoritmos Evolutivos [6] para optimización contínua, la Evolución Diferencial y la Programación Genética [7].

- **Metaheurísticas Basadas en Adaptación Social [8]:** Simulan el comportamiento colectivo de una sociedad para la resolución de problemas. Destacan los Algoritmos de Optimización Basados en Colonias de Hormigas y la Optimización Basada en Nubes de Partículas.

- **Metaheurísticas Híbridas:** Como su nombre indica, estas metaheurísticas combinan varios tipos de algoritmos. Podemos encontrar los Algoritmos Meméticos [9], que combinan Algoritmos Evolutivos con algoritmos de Búsqueda Local, y los algoritmos de Búsqueda Dispersa [10], un método evolutivo que también suele utilizar la Búsqueda Local como método de mejora de la solución actual.

- **Metaheurísticas Paralelas [11]:** Dado que las Metaheurísticas suelen realizar un cálculo intensivo, llega a ser necesario utilizar paralelismo para reducir el tiempo de ejecución. Este tema está centrado en el uso del paralelismo aplicado a metaheurísticas previamente estudiadas.

En este artículo, los autores nos proponemos aplicar el aprendizaje adquirido a lo largo de nuestra formación en las diferentes asignaturas de Ingeniería de Computadores (Arquitectura de Computadores, Ingeniería de Servidores, Estructura de Computadores y Tecnología y Organización de Computadores) a la implementación de metaheurísticas estudiadas en Computación y Sistemas Inteligentes, nuestra especialidad.

Para este fin, nos hemos decantado por la implementación y el análisis de un algoritmo memético y su aplicación a un problema de optimización.

La elección de un algoritmo memético en este artículo se debe a la cantidad de algoritmos que, de forma indirecta, van incluidos en él. Nos referimos a algoritmos de búsqueda local y Basados en Poblaciones. Entendemos que esta es la forma más efectiva de abarcar el mayor contenido de la asignatura Metaheurísticas en este artículo.

## 2 Características de los algoritmos meméticos

Los algoritmos meméticos son algoritmos basados en poblaciones y derivan de los genéticos. Estos algoritmos se basan en el concepto de *meme*. En la Antropología, un meme es la unidad mínima de información cultural que un individuo puede transferir a otro permitiendo así la preservación cultural entre generaciones.

El concepto de meme va asociado al de información. Los algoritmos meméticos tratan de utilizar el Dominio de conocimiento de un problema y preservarlo durante la búsqueda de una solución aceptable.

En los algoritmos genéticos, se generaban nuevas poblaciones a partir de las actuales, utilizando diferentes operadores (cruce, recombinación…) y posteriormente aplicando una mutación con una determinada probabilidad. Estas transformaciones no utilizan información del dominio de conocimiento para generar nuevos individuos.

Los algoritmos meméticos, en cambio, utilizan operadores que trabajan con información del dominio del problema. Estos operadores reciben el nombre de *híbridos* y traspasan atributos entre generaciones de individuos de una población.

A continuación, aclararemos el concepto de *traspaso de información* del dominio del problema entre individuos. En primer lugar, los algoritmos evolutivos son algoritmos que se han mostrado eficientes a la hora de explorar la diversidad en un espacio de soluciones. Esto nos permite tener algoritmos que hayan explorado diferentes zonas del

universo de soluciones. Sin embargo, los algoritmos evolutivos no se comportan adecuadamente a la hora de explotar un espacio de soluciones, esto es, dado una zona del espacio de soluciones, encontrar la mejor solución posible dentro de esa zona del espacio.

Por otro lado tenemos los algoritmos de búsqueda local, son algoritmos eficientes que son capaces de explotar una zona del espacio hasta encontrar la mejor solución posible. Sin embargo, son malos exploradores, es decir, no son algoritmos adecuados para buscar en diferentes zonas del espacio de soluciones.

Tenemos entonces un compromiso entre explotación y exploración. Existen algoritmos, como el de trayectorias múltiples, que comienzan a explotar desde diferentes zonas del espacio. Pero si lo que nos interesa es preservar las bondades que nos pueda ofrecer un algoritmo evolutivo a la hora de resolver nuestro problema, entonces podemos recurrir a los algoritmos meméticos.

Aún queda una cuestión pendiente, ¿nos aporta verdaderamente ventaja un algoritmo de Búsqueda Local?: Podemos considerar que un algoritmo de Búsqueda Local es un algoritmo experto en el dominio, que conoce, dadas un conjunto de soluciones, cuál es la mejor de todas. Wolpert y Macready enunciaron en 1997 el teorema "No Free Lunch"[12] por el cual, sabemos que para cada algoritmo, la ganancia en su rendimiento para un problema concreto es el coste que se paga en su rendimiento en otra clase de problema. Aplicar un algoritmo de Búsqueda Local sobre un problema determinado nos va a permitir aumentar los resultados del algoritmo en ese problema, a costa de perder eficacia en los otros.

La estructura general de un algoritmo poblacional es la siguiente [13]:

```
BÚSQUEDA POBLACIONAL
begin
        POBL <- GenerarPoblacionInicial()
        repeat
                NUEVAPOBL <- GenerarPoblacion(POBL)
                POBL <- ActualizarPobl(POBL,NUEVAPOBL)
                si POBL converge
                        POBLS <- reinicia(POBL)
                finsi
        until criterioParada
end
```

Podemos definir un algoritmo memético como "una población de agentes que alterna periodos de automejora (vía búsqueda local) con periodos de cooperación (vía recombinación), y competición (vía selección)" [14]

En los algoritmos meméticos es común hablar de agentes y no de individuos, ya que los primeros son considerados una extensión de los segundos.

Se pueden encontrar dos tipos de procesos de relación entre los agentes, los competitivos y los cooperativos. Entre los primeros tenemos la selección y la actualización, o reemplazo, mientras que entre los segundos encontramos la reproducción.

Existen dos operadores básicos de reproducción: la recombinación y la mutación. La recombinación se encarga de crear agentes nuevos haciendo uso de la información extraída de agentes recombinados, mientras que en la mutación se crean agentes nuevos mediante la modificación de otros existentes insertándoles información externa.

## 3 Análisis del algoritmo

En este artículo no queremos centrarnos en la construcción del algoritmo, ni en analizar cada una de sus partes. Tampoco vamos a implementar un algoritmo memético. Nos queremos centrar en aplicar paralelismo a un algoritmo memético como forma de casar el contenido de ambas especialidades del Grado.

Vamos a utilizar para las pruebas una implementación de un algoritmo memético realizada por Quang Huy Nguyen, de la Universidad Tecnológica de Nanyang. El algoritmo está publicado bajo una licencia GNU GPL v3 [15]. Es una implementación muy eficiente pero no aprovecha paralelismo. Utilizando OpenMP, añadiremos directivas necesarias para convertir el algoritmo en una solución paralela y posteriormente, compararemos resultados.

El algoritmo se ejecutará resolviendo un problema de ortogonalización utilizando los siguientes algoritmos. Como batería de pruebas, los datos de cada problema a procesar se generan de forma aleatoria en cada una de las comparaciones que se realizarán:

- Algoritmo de ortogonalización de Gram-Schmidt en un problema bidimensional [16]
- Algoritmo Genético utilizando la Función de Rastrigin [17]
- Algoritmo memético.

Tanto en el caso del algoritmo genético como en el memético, ambos estarán limitados a 50 generaciones. Gram-Schmidt no tiene restricciones y hallará un resultado óptimo.

Las baterías de pruebas anteriormente mencionadas se ejecutarán sobre dos máquinas cuyas características se muestran a continuación:

**Tabla 1.** Información de las máquinas utilizadas para la ejecución de las pruebas

| Datos | Máquina 1 | Máquina 2 |
|---|---|---|
| **Procesador** | Intel Core 2 Quad CPU Q8400 | AMD Athlon(tm) II X4 630 |
| **Cache** | 4Mb | 1Mb |
| **Frecuencia** | 2.66 GHz | 2.8 GHz |
| **Nº de núcleos** | 4 | 4 |
| **Memoria** | 4 GB | 2 GB |

El código de la implementación está escrito en C++, habiendo sido compilado utilizando g++ con los parámetros:

*-g -Wall -O3 -D__LINUX__ -DEBUG -Wall -I. -c -w*

En primer lugar, se hace una ejecución del algoritmo tal cual ha sido implementado por su autor (es decir, sin aprovechar paralelismo), obteniendo los siguientes resultados:

**Tabla 2.** Medición de tiempos sin paralelismo (en milisegundos)

| Medición | Genético | Memético | Gram-Schmidt |
|---|---|---|---|
| **MÁQUINA 1** | | | |
| **1** | 3950 | 3071 | 43425 |
| **2** | 4948 | 3036 | 56614 |
| **3** | 4705 | 3047 | 57225 |
| **4** | 5590 | 3241 | 49362 |
| **5** | 2426 | 4338 | 47424 |
| **6** | 2424 | 3780 | 50589 |
| **7** | 4420 | 3046 | 57809 |
| **Media** | **4066** | **3366** | **51778** |
| **Desv. Tipica** | **1227** | **505** | **55591** |
| **MÁQUINA 2** | | | |
| **1** | 1934 | 3110 | 339312 |
| **2** | 1930 | 3110 | 346738 |
| **3** | 1927 | 3256 | 329656 |
| **4** | 1942 | 3141 | 334336 |
| **5** | 1929 | 3099 | 337174 |
| **6** | 1939 | 3130 | 332181 |
| **7** | 1946 | 3110 | 340268 |
| **Media** | **1935** | **3136** | **337095** |
| **Desv. Tipica** | **7,3** | **54,42** | **5701,98** |

Hasta aquí hemos obtenido los resultados de un algoritmo que no aprovecha el paralelismo. Es decir, estamos utilizando el algoritmo de la misma forma que se explica en la asignatura Metaheurísticas. Ahora es el momento de ir más allá y de aprovechar el contenido impartido en asignaturas como Arquitectura de Computadores, estableciendo una forma rápida de paralelizar el algoritmo.

Haremos uso de OpenMP ( http://openmp.org/wp/ ), una API que facilita la aplicación de concurrencia en C/C++. OpenMP divide en hebras de ejecución diferentes tareas. La granularidad de la división de tareas puede dejarse a la libre elección de OpenMP o bien puede ser especificada por el programador a nivel de funciones, bucles u otros bloques de código. OpenMP funciona por medio de directivas de compilación, lo que minimiza la invasión del código fuente. OpenMP se utiliza en las prácticas de Arquitectura de Computadores.

Hay muchas estrategias para paralelizar diferentes metaheurísticas. Reiteramos que no es nuestra intención hacer un análisis del algoritmo, por lo tanto, los tiempos se han medido utilizando los parámetros por defecto que establece el autor y tampoco hablaremos de los resultados obtenidos por el algoritmo (*fitness*). El objeto de este artículo no es mejorar la eficiencia de los algoritmos meméticos superando lo que se ha conseguido hasta la fecha, sino el de crear sinergias entre los contenidos de asignaturas de diferente especialidad dentro del Grado en Ingeniería Informática. Enrique Alba propone diferentes enfoques para paralelizar algoritmos evolutivos [18]. Haremos una paralelización básica y analizaremos la mejora de la eficiencia de ese algoritmo utilizando el conocimiento que hemos adquirido en Ingeniería de Servidores (ISE).

La sección de código seleccionada para ser paralelizada ha sido la función “evolve” de la clase “MemeticAlgorithm”. Esta función es la encargada de, como su nombre indica, evolucionar la población actual a un número de generación que se determina como parámetro. Por ello, es la función sobre la que recae la mayor parte de la carga computacional. Particularmente, se ha paralelizado el bucle más interno para que cada hebra se encargue de una porción del bucle:

```
#pragma omp parallel num_threads(nthreads)
{
        #pragma omp for
        for(unsigned j=0; j<v.size(); j++)
        {
                //cout << -gs->pop[v[j]]->fitness << " --> ";
                vector<double> ch = gs->pop[v[j]]->toDoubleVector();
                //cout << ls->search(ch) << " == ";
                ls->search(ch);

                // if lamarckian learning then copy back
                if (maLearningStrategy == maLSLamarckian) gs->pop[v[j]]->fromDoubleVector(ch);

                gs->pop[v[j]]->fitness = gs->evaluate(ch);

                //cout << -gs->pop[v[j]]->fitness << endl;
        }
}
```

**Figura 1.** Código paralelizado con directivas OpenMP

Tras compilar el algoritmo con OpenMP, obtenemos los siguientes resultados:

**Tabla 3.** Medición de tiempos con paralelismo (en milisegundos)

| Medición | 1 núcleo | 2 núcleos | 4 núcleos |
|---|---|---|---|
| **MÁQUINA 1** | | | |
| **1** | 3071 | 2348 | 3590 |
| **2** | 3036 | 2503 | 3714 |
| **3** | 3047 | 2377 | 1964 |
| **4** | 3241 | 5114 | 3715 |
| **5** | 4338 | 2376 | 1862 |
| **6** | 3780 | 4528 | 2876 |
| **7** | 3046 | 2635 | 1782 |
| **Media** | **3366** | **3126** | **2786** |
| **Desv. Tipica** | **505** | **1174** | **905** |
| **MÁQUINA 2** | | | |
| **1** | 3110 | 2992 | 3011 |
| **2** | 3110 | 2962 | 2953 |
| **3** | 3256 | 2985 | 2950 |
| **4** | 3141 | 3039 | 3000 |
| **5** | 3099 | 3017 | 2965 |
| **6** | 3130 | 2993 | 2990 |
| **7** | 3110 | 3004 | 2925 |
| **Media** | **3136** | **2999** | **2971** |
| **Desv. Tipica** | **54,42** | **24,49** | **30,87** |

Podemos observar la mejora de la eficiencia del algoritmo cuando se realiza el reparto de tareas entre los diferentes núcleos.

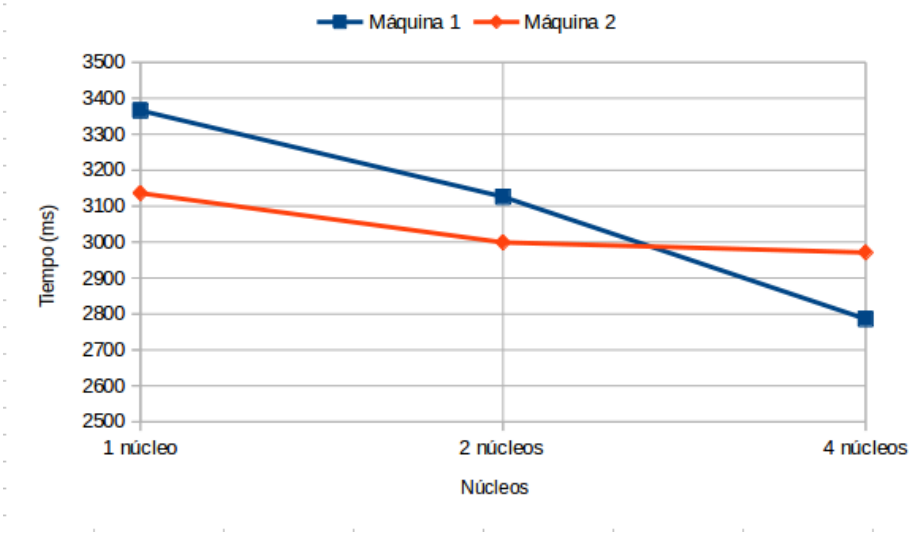


**Figura 2.** Evolución de tiempos de ejecución en ambas máquinas.

De la primera máquina, no podemos hablar de una mejora significativa cuando se paraleliza con dos núcleos, de acuerdo a la prueba t de student, obtenemos un p-valor de 0.6288 lo que no nos permite rechazar la hipótesis nula (es decir, que no haya diferencias significativas). Un panorama totalmente diferente nos ofrece la versión paralela de 4 núcleos, que muestra diferencias significativas con un p-valor de 0.1714, que nos permite rechazar la hipótesis nula.

Con respecto a la versión lineal, hemos obtenido un tiempo $\frac{3366}{3126} = 1.07$ veces mejor para dos núcleos y $\frac{3366}{2786} = 1.2$ en el caso de los cuatro núcleos.

En cuanto a la segunda máquina, podemos considerar que se ha producido una mejora de rendimiento respecto a la versión para cuatro núcleos rechazando la hipótesis nula con un p-valor de 0.2251.

Obtenemos un tiempo $\frac{3136}{2999} = 1.05$ veces mejor para dos núcleos y $\frac{3136}{2971} = 1.06$ en el caso de los cuatro núcleos.

## 4 Conclusiones

En la especialidad de Computación y Sistemas Inteligentes (CSI) se estudia de forma teórica gran cantidad de metaheurísticas, heurísticas y algoritmos y otras técnicas de gran utilidad en diferentes campos de las Tecnologías de la Información y la Comunicación (criptografía, visión por computador, aprendizaje automático, robótica, análisis de señales…). Pero, a la hora de enfrentarnos a un problema real, estos conocimientos son insuficientes sin el apoyo del conocimiento que proporcionan asignaturas como TOC, EC, AC o ISE y que, de un modo u otro, siempre están y han de estar presentes. Creemos indispensable profundizar en el temario de las mismas y que se transmita la importancia de estos contenidos a la hora de desarrollar soluciones de calidad.

En este artículo hemos comprobado la evolución en el rendimiento de un algoritmo memético cuya ejecución se ha paralelizado. Los resultados, aún sin haber realizado un análisis en detalle sobre la mejor configuración de sus parámetros o la mejor estrategia de paralización del algoritmo, muestran que hemos conseguido mejoras sustanciales.

Estos resultados reflejan la necesidad de transmitir a los estudiantes del grado la importancia, ya no solo de que sean impartidos, sino también de incidir en aquellos contenidos que se han de adquirir por primera vez durante el segundo año de carrera, pero que deben formar parte de la formación del estudiante en etapas posteriores.

**Agradecimientos.** Nos gustaría mostrar nuestros agradecimientos a Roberto Cogolludo Ayuso, graduado en Traducción e Interpretación por la Universidad de Granada, por su labor como revisor de este artículo y a Francisco Herrera Triguero, profesor de la asignatura de Metaheurísticas, por su labor como profesor y por la información, de gran utilidad, ofrecida para redactar este artículo.

## Referencias

1. Material de la asignatura Metaheurísticas proporcionado por Francisco Herrera Triguero. 3º de Grado de Ingeniería Informática, perfil de Computación y Sistemas Inteligentes, 2013-2014. http://sci2s.ugr.es/docencia/metah/ en línea 30 de marzo de 2015.
2. E.-G. Talbi. Metaheuristics. From design to implementation. Wiley, 2009.
3. S. Kirkpatrick, C. D. Gelatt, M. P. Vecchi, Optimization by Simulated Annealing, Science 220:4598 (1983) 671-680.
4. F. Glover, M. Laguna. Tabu Search. Kluwer Academic, 1997
5. F. Glover, G.A. Kochenberber. Handbook of Metaheuristics. Kluwer Acad., 2003. Cap 12: Multi-start Methods. (Rafael Marti), 355-368. Cap 8: Greedy Randomized Adaptive Search Procedure. (M.G.C. Resende, C.S. Ribeiro), 219-251. Cap 6: Variable Neighborhood Search, P.hansen, N.Mladenovic, 145-184, Cap: 11, Iterated Local Search, H.R. Lourenço, O.C. Martin, T.Stützle, 321-353.

6. D.B. Fogel (Ed.). Evolutionary Computation. The Fossil Record. (Selected Readings on the History of Evolutionary Computation). IEEE Press 1998.
7. J. R. Koza, Genetic Programming. MI Press, 1992.
8. E. Bonabeau, M. Dorigo, G. Theraulaz. Swarm Intelligence. From Nature to Artificial Systems. Oxford University Press, 1999.
9. P. Moscato, C. Cotta, "A Gentle Introduction to Memetic Algorithms". In: F. Glover, G.A. Kochenberber, (Eds.). Handbook of Metaheuristics. Kluwer Academics. (2003) 105-144, Kluwer, Boston MA, 2003.
10. M. Laguna, R.Martí. Scatter Search. Kluwer, 2002.
11. E. Alba (ed.), "Parallel Metaheuristics", John Wiley & Sons, 2005.
12. Wolpert, D.h., y W.g. Macready. "No Free Lunch Theorems for Optimization." IEEE Transactions on Evolutionary Computation (1997): 67-82.
13. Morales, Eduardo F. "Algoritmos Meméticos." Instituto Nacional de Astrofísica, Óptica y Electrónica (INAOE), México, 2 Nov. 2004. Web. 22 Mar. 2015. <http://ccc.inaoep.mx/~emorales/Cursos/Busqueda/node117.html >
14. Moscato, P.A. (1989). On Evolution, Search, Optimization, Genetic Algorithms and Martial Arts: Towards Memetic Algorithms. Caltech Concurrent Computation Program Report 826, Catech, Pasadena, California.
15. Huy Nguyen, Quang. "Memetic Algorithm Package". 1 Ene. 2010. Web. 24 Mar. 2015. <https://code.google.com/p/memetic-algorithm/ >
16. I.V. Proskuryakov "*Encyclopedia of Mathematics*". Orthogonalization (ISBN: 1402006098) Web. 7 May. 2015. <http://www.encyclopediaofmath.org/index.php/Orthogonalization >
17. H. Mühlenbein, D. Schomisch y J. Born. "The Parallel Genetic Algorithm as Function Optimizer ". Parallel Computing, 17, páginas 619–632, 1991.
18. Alba, Enrique y Tomassini, Marco. "Parallelism and evolutionary algorithms" IEEE Transactions on Evolutionary Computation (2010): 443-462

# Convolución paralela con GPU

F. J. Mesa Hidalgo, M. G. Arenas

Departamento de Arquitectura y Tecnología de Computadores.
ETSI Informática y de Telecomunicación. Universidad de Granada
Granada, España

fjmh89@icloud.com, mgarenas@ugr.es

**Abstract.** Este trabajo presenta un proyecto de fin de carrera realizado en la Escuela Superior de Ingeniería Informática y Telecomunicación de la Universidad de Granada donde se pone de manifiesto el potencial de la computación paralela en la GPU mediante el uso de dos tecnologías concretas: CUDA y OpenCL, tomando para ello, como ejemplo real, el algoritmo de la convolución para imágenes. Se analizaron las diversas implementaciones propuestas para cada una de las técnicas y se realizó un estudio sobre los resultados obtenidos tras la ejecución de éstos en función de las imágenes y filtros de entrada. Así mismo, se tratará el presente documento como la elaboración de un pequeño manual que pueda servir de apoyo al uso que dicho proyecto de fin de carrera tiene en el actual curso académico para la asignatura Arquitectura y Computación de Altas Prestaciones.

**Keywords:** Computación Paralela, gpu, cuda, opencl, convolución

**Abstract.** This paper presents a thesis undertaken in the ETSIIT of the University of Granada where the potential of the parallel computing on the GPU is illustrated by using two different kinds of technology: CUDA and OpenCL, taking as a real example the Convolution algorithm for images. Several proposals of algorithm will be analyzed as well as the possible explanations about the obtained outcomes based upon various input images and filters. Additionally, the present document is intended to be conceived as a manual for starters, which could potentially become a support during the current academic year for the course “Arquitectura y Computación de Altas Prestaciones”.

**Keywords:** Parallel programming, gpu, cuda, opencl, convolution

## 1 Introducción

Este artículo presenta otra vertiente de los proyectos Fin de Carrera (PFC) desarrollados dentro del departamento de Arquitectura y Tecnología de Computadores de a

adfa, p. 1, 2011.

Universidad de Granada. En este caso, se trata de un proyecto con un claro carácter investigador, pero cuya aplicación está claramente orientada a docencia, por lo que se pone de manifiesto que no siempre un proyecto investigador tiene por qué tener una aplicación al ámbito investigador, sino que se le pueden buscar otros usos.
El proyecto que se presenta está relacionado con la Computación paralela [1], tanto desde un punto de vista teórico como práctico. Este punto de vista se pone de manifiesto en las titulaciones que imparte este departamento a través de varias asignaturas que están englobadas tanto en titulaciones de Grado como en titulaciones de Ingeniería, ya en extinción.

Concretamente conceptos relacionados con la Computación Paralela se introducen dentro del Grado en Informática en asignaturas como Arquitectura de Computadores, impartida por este departamento en el cuarto semestre de la titulación. Otras asignaturas como Arquitectura y Computación de Altas Prestaciones impartida dentro de la especialidad de Ingeniería de Computadores en el sexto semestre del Grado de Informática o la asignatura Programación Paralela impartida por el departamento de Lenguajes y Sistemas de la Universidad de Granada durante el sexto semestre del Grado en Informática. Y dentro de titulaciones recién extinguidas se impartían en asignaturas como Arquitectura de Computadores I y II (quinto curso de la Ingeniería informática) o Programación Distribuida y Paralela que impartía el departamento de Lenguajes y Sistemas Informáticos de la Universidad de Granada dentro de la Ingeniería Informática (cuarto curso).

En todas estas asignaturas se abordan conceptos como los diferentes modelos de programación paralela, Paso de Mensajes, Variables Compartidas, diferentes herramientas para llevarla a cabo, como OpenMP, MPI o los diferentes niveles de paralelismo, como el paralelismo a nivel de instrucción, ILP, o a nivel funcional. Este proyecto está centrado en la utilización de una tecnología novedosa para la Computación Paralela como es CUDA [2].

CUDA es una tecnología introducida por NVIDIA para la computación de propósito general con dispositivos de procesamiento gráfico, GPU (*Graphics Processing Unit).* Este tipo de tecnología proporciona una herramienta para utilizar las GPU para acelerar programas no necesariamente gráficos, sino de ámbito general, naciendo así lo que se ha dado a llamar GPGPU *Computing (General Purpose Graphics ProcessingUnits).*

El objetivo de este proyecto era realizar un estudio de la escalabilidad que se podía obtener en una GPU utilizando diferentes tecnologías: CUDA y OpenCL [3]; realizando para ello un caso de estudio con varias tarjetas gráficas y un algoritmo determinado. Concretamente el algoritmo utilizado es una Convolución de imágenes [4]. La aplicabilidad de este proyecto ayudará a los alumnos de la asignatura Arquitectura y Computación de Altas Prestaciones a entender el un concepto importante dentro de lo que es la Computación Paralela como es la adaptación o *mapeo* de la cantidad de trabajo que un algoritmo debe realizar con la máquina o arquitectura paralela que va a llevarla a cabo.

Para ello, se detallará el proyecto realizado incluyendo detalles de las diferentes versiones de Convolución que se han desarrollado para ejecutarlas dentro de una GPU y la propuesta de práctica que se va a realizar para que los alumnos de la especialidad

de Ingeniería de Computadores lleguen a entender el concepto de *mapeo* realizando ellos nuevas versiones de los algoritmos propuestos donde la unidad o cantidad de trabajo que cada una de las hebras del algoritmo ejecuta sea variable y poder así estudiar el impacto que el *mapeo* de una tarea puede tener sobre la ejecución de un algoritmo, concepto muy relacionado con la granularidad de dicho algoritmo.

Este artículo está compuesto por 10 secciones que incluyen un pequeño estado del arte, una relación de las herramientas existentes para abordar GPGPU *Computing*, características de la arquitectura de una GPU a tener en cuenta para desarrollar aplicaciones para ellas, detalles de las diferentes versiones del algoritmo desarrollado dentro el proyecto Fin de Carrera, la practica propuesta y unas conclusiones que en cierto modo recopilan el beneficio esperado por esta iniciativa.

## 2 Estado del Arte

La computación acelerada por GPU se basa en el uso de una unidad de procesamiento de gráficos (GPU) junto con una CPU para mejorar significativamente el rendimiento de aplicaciones científicas [5], analíticas [6], de ingeniería [7] y empresariales [8].
En 2006 nace la primera GPU cuya arquitectura aunaba la capacidad de renderizar imágenes en 3D en tiempo real y la posibilidad de ejecutar programas escritos en C mediante el modelo de programación CUDA.

En 2007, la conocida compañía de tarjetas gráficas NVIDIA estuvo al frente en el desarrollo de las GPU que ahora potencian centros de datos con eficiencia energética en laboratorios gubernamentales [9], universidades [10], grandes compañías y pequeñas y medianas empresas de todo el mundo [11]. Las GPU están acelerando las aplicaciones en plataformas en automóviles [12], teléfonos móviles y tablets [13], aviones no tripulados y robots [14].

Las GPU han evolucionado hasta el punto de disponer de un extenso número de núcleos de ejecución para fines no solo gráficos, sino para la ejecución de programas de diversa índole [15].

A partir de ese momento surge una tendencia que, a día de hoy, sigue en auge [16]. Investigadores y desarrolladores han acogido con entusiasmo el concepto de computación en GPU para un gran rango de aplicaciones. Su buena relación coste-rendimiento ha permitido que su adopción continúe expandiendo el número de autores y universidades interesados en enseñar CUDA, como ha ocurrido en esta misma Universidad. Los propios desarrolladores contribuyen en esta tarea de expansión mediante la creación de librerías y utilidades y la cooperación vía foros de discusión [17].

## 3 Arquitectura GPU

A continuación, se va a realizar una presentación de las primeras nociones sobre paralelismo en la GPU con CUDA. Se presentará el modelo lógico, el modelo físico y la conexión entre ambos.

En primer lugar, se presenta el concepto de hebra o *Thread*, la cual constituye una unidad de ejecución en el dispositivo o device. Del mismo modo, se presentan los conceptos de rejilla/retícula o *Grid* y bloque o *Block*.

A nivel lógico, CUDA establece un modelo de ejecución que comprende las diversas estructuras que la *Figura 1* representa. Estas estructuras están relacionadas con la forma en la que se asignan los recursos para una determinada ejecución.

Cada función que se ejecuta en el device se denomina *kernel*. Los *kernel*s se ejecutan organizados en un Grid. Este puede estar organizado a lo largo de hasta 2 dimensiones que albergan una serie limitada de bloques. Cada bloque posee el mismo número de hebras, las cuales se distribuyen a lo largo de hasta 3 dimensiones (altura, anchura, profundidad). Cada hebra ejecuta una instancia del *kernel*, es decir, el código escrito en la función.

El número de bloques por *Grid*, así como de hebras por bloque, es limitado y depende de la generación del *device* (limitación hardware).

Además, la unidad de ejecución física se denomina *warp* y está formado por 32 hebras donde la ejecución es la misma en todas ellas.

A nivel hardware, toda tarjeta gráfica NVIDIA compatible con CUDA dispone de una serie de *Streaming Multiprocessors* o SM, donde se organizan los recursos (memoria, procesadores, contador de programa, etc). Cada SM dispone de una serie de SP que se encargan del cómputo de una hebra.

La conexión entre la parte lógica y física de CUDA se llevaba a cabo mediante la asignación de bloques al SM y la organización de los bloques en *warps*. La cantidad de bloques asignados en un momento dado dependerá de la cantidad de hebras que el SM puede ejecutar concurrentemente.

Por ejemplo, la NVIDIA GeForce GT200, es un dispositivo que aceptaba un total de 1024 hebras en ejecución. Ello condiciona la cantidad de bloques que podrán coexistir en el SM a la vez. Si el bloque contiene 16x16=256 hebras cabrán 1024/256=4 bloques que se ejecutarán al mismo tiempo. Cuando un bloque es asignado a un SM, este bloque permanecerá activo hasta que todas las hebras hayan finalizado.

Actualmente, el número de SP de los que consta cada SM suele ser 32, uno por cada hebra de un *warp*. Las ganancias que se obtienen gracias a la GPU residen, en su mayor parte, en la forma en la que se ocultan las latencias al acceder a memoria. Esto se debe a que la unidad de ejecución hardware es el *warp*. En el momento en el que alguna hebra del *warp* debe esperar una latencia por haber accedido a memoria, ese *warp* sale de ejecución. En su lugar, se elige de forma inmediata otro *warp* cuyas hebras estén listas. Esta característica conduce a una nueva situación a propiciar: maximizar el número de *warp*s por bloque. Gracias a ello, se dispone de un rango más amplio de *warps* entre los que alternar, ocultando siempre los tiempos de acceso a memoria (*latencyhiding*).

Otro factor a destacar, es el número de bloques que físicamente el dispositivo puede ejecutar o *block-slots*. De forma que si la GT200 tiene una limitación de 8 slots, un tamaño de bloque de 8x8=64 provocaría un desperdicio de los recursos. Esto es debido a que el número de hebras que la GT200 puede manejar concurrentemente es 1024. 64 hebras por bloque originaría 1024/64=16 bloques, pero ya que solo es capaz de

albergar 8, otros 8 se quedarían sin ejecutarse, desperdiciándose así 8x64=512 hebras, la mitad de las que el SM soporta.

Además, cada SM cuenta con un determinado número de registros que pueden usarse durante la ejecución de un *kernel*. Como se ha mencionado antes, cada hebra ejecuta una instancia del *kernel*, que emplea en su código un determinado número de registros locales. El programador debe velar porque el número de registros total correspondiente a las hebras que van a residir en el SM concurrentemente no supere la limitación impuesta por el hardware. Si las 1024 hebras que la GT200 puede mantener van a ser usadas y cada SM de la GT200 posee una limitación de 32786 registros, quiere decirse que cada hebra no puede usar más de 32768/1024=32 registros. Si este número es mayor habrá ciertos bloques que no podrán entrar al *device* por falta de registros, con el correspondiente desperdicio de recursos asociado.

Del mismo modo, cada *device* dispone de una cantidad limitada de memoria global, memoria constante y memoria compartida, cada una con sus propias peculiaridades (ver Figura 1). La memoria global será la memoria de propósito general que albergue los datos de un algoritmo que se esté ejecutando en la GPU. Sus ratios de transferencia son superiores a los que la memoria constante o compartida pueden aportar. Se usará memoria constante para acceder a datos de memoria con una frecuencia mucho mayor, ya que provee menores tasas de transferencia. Por último, la más veloz de los 3 tipos es la memoria compartida, que es la que usan todas las hebras de un mismo bloque.

Esta memoria compartida necesita una inicialización previa desde memoria global que tiene lugar durante la ejecución del *kernel*. La cantidad está ligada al *device*, por lo que será necesario dotar a nuestros algoritmos CUDA de cierta adaptabilidad para que puedan ser ejecutados en las diferentes tarjetas NVIDIA del mercado.

Como se ha especificado anteriormente, cuando un *warp* se ejecuta en la GPU, si el algoritmo requiere datos de entrada, éstos han de pasar por memoria global, pues no existe un acceso directo desde el device a memoria principal de la CPU durante la ejecución del *kernel*. En el caso concreto del algoritmo de la convolución, será necesario el envío de la imagen a tratar al device, organizando sus *pixels* de forma secuencial formando un array unidimensional de tamaño anchuraxalturaxprofundidad de la imagen. Será necesario establecer un convenio que permita saber en todo momento la correspondencia de cualquier pixel que se exprese de la forma Imagen[0][0][0] (matriz tridimensional) al de una matriz unidimensional.

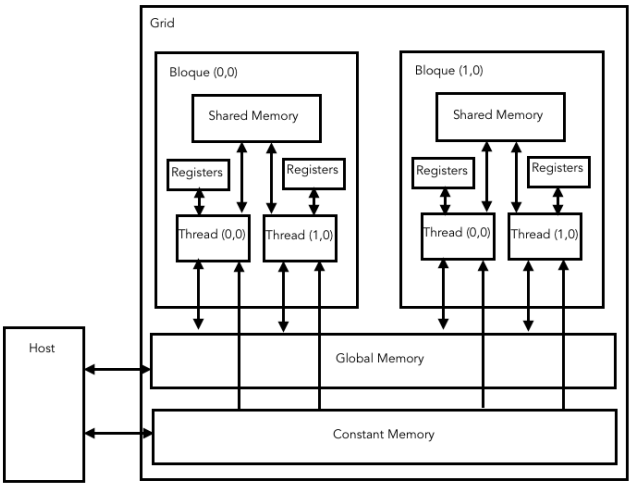


**Fig. 1.** Organización de la memoria en el *device*

## 4 Proyecto: convolución 2D en imágenes mediante la GPU

Este proyecto se centra en la computación en GPU para realizar una convolución 2D en imágenes a color.

Una convolución [4] es un operador matemático [18] que transforma dos funciones [19] f y g en una tercera función que, en cierto sentido, representa la magnitud en la que se superponen f y una versión trasladada e invertida de g. En concreto, este proyecto trata un determinado tipo de convolución: la convolución discreta bidimensional.

El proceso de convolución aplica una ventana rectangular (de un tamaño generalmente muy inferior a las dimensiones de la imagen) a la imagen, centrando la ventana en cada uno de los *pixels* de la imagen consecutivamente. Es decir, tomamos un pixel de la imagen y nos quedamos con todos los *pixels* que lo rodean. Llamaremos a este bloque $IMG_{xy}[i][j]$, donde:

- *x, y* son las coordenadas del pixel que se esta procesando en este momento.
- i, j son los índices horizontal y vertical dentro de la matriz de los elementos que rodean al pixel que estamos procesando. Así, IMGxy[1][1] es el valor del pixel actual. IMGxy[1][0] es el valor del pixel encima del actual, etc.

Además, tenemos una matriz de Coeficientes *coef[i][j]*, que para nuestros ejemplos serán matrices 3x3. Con estas definiciones, la operación a realizar para cada pixel de la imagen (es decir, para cada *y* en la dimensión vertical y cada *x* en la dimensión horizontal) será:

```
nuevo_pixel=IMG_xy[0][0]*coef[0][0] + IMG_xy[0][1]*coef[0][1]
+ ... + IMG_xy[2][2]*coef[2][2];
```

Será una condición necesaria que los valores obtenidos para un pixel determinado no excedan los valores [0-255], truncando el valor si fuera necesario. En concreto, el algoritmo secuencial en nuestro caso tendría el siguiente aspecto:

```
for( ) //Rojo, Verde, Azul
for( ) //filas
for(){  //columnas
for() //filas máscara
for() //columnas máscara
if( dentro de los limites )
suma parcial
sustituir pixel por la suma
}
```

Lo que interesa comprender de este código es que está usando 3 bucles para poder iterar por todos los elementos del buffer realizando las indexaciones necesarias. Además, para cada uno de ellos se necesitan dos bucles para poder iterar sobre la máscara o *kernel*(vecindario). Como tratamiento de bordes se ha optado por simular un marco negro del mismo grosor que la máscara. Es decir, para los elementos de la máscara que no se correspondan a ningún pixel en la imagen por haber excedido los límites de ésta no se tendrán en cuenta, o lo que es lo mismo, sumaremos 0 (negro).
La convolución, combinada con la máscara adecuada, puede originar resultados muy diferentes [20]. En función de la máscara, podrá usarse para la eliminación del ruido de las imágenes o el suavizado del mismo, difuminar rostros, identificar los elementos de una imagen o enfocar imágenes desenfocadas entre otras. Es un algoritmo que interviene en numerosos proyectos cuyos resultados nos rodean a menudo (efectos fotográficos, películas, etc) y es aplicado directamente por una gran cantidad de programas informáticos; desde la suite de Adobe (*Premiere, Photoshop*) [21] o GIMP [20] hasta MATLAB [22] o *Mathematica* [23].

# 5 Paralelización del problema

## 5.1 Paralelización CUDA I: Distribución unidimensional

El enfoque a seguir consiste en que cada hebra aplique el algoritmo de la convolución a un pixel en concreto (ver Figura 2). Esto quiere decir que trabajará sobre 3 elementos de la matriz (en realidad, como se ha descrito antes, es un array, pero por simplicidad, se llamará matriz en las ocasiones que ayuden a un mejor entendimiento). Es decir, para el pixel que se corresponda a la hebra actual se aplica el algoritmo de la convolución accediendo a cada uno de los elementos de su vecindario y de los elementos de la máscara. Esta operación se repite para el resto de elementos que consti-

tuyen el pixel, es decir, para el elemento que forma la parte roja del pixel, la verde y la azul.

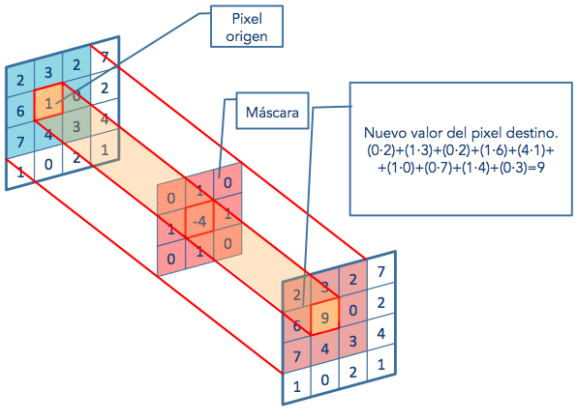


**Fig. 2.** Convolución 2D en imágenes.

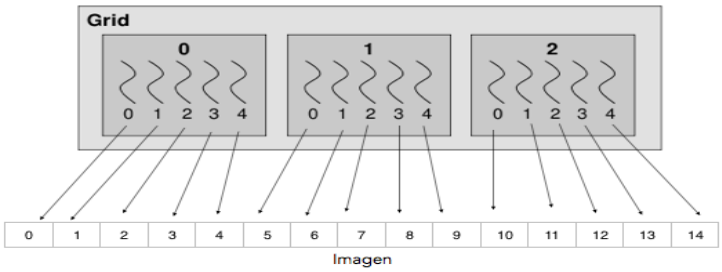


**Fig. 3.** Asignación de trabajo por hebra.

En una primera implementación de este algoritmo se propuso una organización unidimensional tanto de bloques como de hebras. Esta versión crea tantas hebras como *pixels* contiene la imagen. Posteriormente se crean tantos bloques como fueran necesarios para albergar las hebras tendiendo en cuenta las limitaciones:

Además, esta correspondencia entre hebra y pixel trata a cada pixel como un simple elemento de trabajo y no como un elemento cuya ubicación en la matriz está estrechamente relacionada con su vecindario. Esta asignación se puede apreciar en la Figura 3. Esto implica un mejor aprovechamiento de las hebras de cada bloque porque todos los bloques contienen hebras útiles (hebras que se corresponden a elementos de la imagen, *warps* que aprovecharán todos los SP del SM) salvo, en el peor de los casos, el último bloque si el número de *pixels* total no fuera divisible entre el tamaño de bloque decidido .

No obstante, esta primera versión desestimó el uso de memoria compartida y las condiciones de organización que ésta necesita para su uso.

### 5.2 Paralelización CUDA II: distribución bidimensional

El enfoque sobre el trabajo que cada hebra va a realizar se mantiene: cada hebra trabajará sobre los 3 elementos que constituyen un pixel.

La diferencia en esta ocasión estriba en que no se trata cada elemento de la imagen de forma independiente desligándolo de la semántica del problema. En cambio, en esta versión del algoritmo se aplica una disposición bidimensional a nivel de bloque y de hebra donde se podrá apreciar una correspondencia más intuitiva entre los bloques de CUDA y la imagen, donde cada uno trabaja sobre una región de la imagen de idénticas proporciones. Esta nueva organización puede apreciarse en la Figura 4.

En este enfoque existen ocasiones donde la última fila y/o columna de bloques no aprovechen todas sus hebras debido a que no haya una división exacta entre la anchura del bloque con la anchura de la imagen y/o entre la altura del bloque con la altura de la imagen. Concretamente, en este ejemplo se muestra una imagen de tamaño 5x5 donde cada bloque se compone de 4 hebras que trabajan sobre 4 *pixels*. Los últimos bloques solamente trabajan sobre 2 hebras (1 en el caso de la esquina inferior derecha) ya que no queda más imagen a tratar.

En esta ocasión, cada hebra se vale también de la componente *y* de sus coordenadas como hebra dentro del bloque y de sus coordenadas como bloque dentro del Grid, pues ahora la ubicación del pixel se realiza bidimensionalmente.


**Fig. 4.** En esta versión los bloques son bidimensionales y trabajan sobre áreas también bidimensionales de la imagen.

Así pues, tenemos que, para localizar el pixel que se corresponde concretamente a una hebra, en general sus filas y columnas son:

```
fila= blockIdx.y x altura_bloque  +  threadIdx.y;
columna = blockIdx.x x anchura_bloque  +  threadIdx.x;
```

Donde `altura_bloque` y `anchura_bloque` determinan las dimensiones de cada bloque. Estas dimensiones se discutirán en breve. De esta manera, tenemos que, para acceder a un pixel en concreto en el buffer secuencial, bastaría con hacerlo de forma: `buffer [fila x anchura_imagen + columna`. Teniendo en cuenta estas modificaciones, el algoritmo CUDA equivalente al secuencial sería el siguiente:

```
__global__ void convolve(  . . .  ) {

for( ){   //Rojo, Verde, Azul
for( )   //filas máscara
for()   //columnas máscara
suma parcial
Sustituir pixel por la suma
   sum = 0;
}
```

En resumen, el algoritmo aplica el algoritmo de la convolución para los 3 elementos (uno por cada color) de un pixel en concreto. La correspondencia hebra-pixel nos lleva siempre al primero de estos (el rojo según la representación interna que hemos seguido) y posteriormente se tratarán el verde y el azul.

Se puede apreciar cómo los dos bucles anidados que iteraban la matriz global han desaparecido, pues ahora no existe una iteración como tal sino que, en función de la hebra que esté ejecutándose, se halla una correspondencia directa con un pixel en concreto de la matriz de la imagen. Mediante la creación de un número de hebras mayor o igual al número de *pixels* nos aseguramos de que todos los *pixels* de la imagen van a ser tratados.

Respecto a la forma del bloque se ha optado siempre por respetar las proporciones cuadradas ya que es la opción que permite un mayor rendimiento a la hora de cargar *pixels* en la memoria compartida.

En cuanto al tamaño, se seguirán las mismas restricciones que las impuestas en la versión paralela anterior, pues el hecho de que el bloque sea unidimensional o bidimensional no cambia el número de hebras que lo compone y, por tanto, su repercusión en el SM.

Como se ha mencionado en los fundamentos teóricos de CUDA y OpenCL, la cantidad de memoria, ya sea global, compartida o constante, depende del dispositivo sobre el que estamos trabajando, de modo que se trata de un factor que requiere de una consulta en tiempo de ejecución. En problemas que requieren trabajar sobre unos datos de entrada, como en el caso de la convolución, la cantidad de memoria global libre determinará el tamaño máximo del problema, pues será en ella donde se aloje la imagen de entrada y de salida. Además, es un parámetro que puede variar a lo largo de una misma ejecución, pues, concurrentemente, nuestro ordenador puede estar ejecutando varios procesos que requieran también de la GPU. En nuestro caso, son 3 las reservas necesarias para albergar, respectivamente, la imagen principal, la imagen de salida y la máscara.

La memoria constante es una memoria de reducido tamaño (normalmente 64 KB), de solo lectura que proporciona un ancho de banda mayor que el de la memoria global. Es por esto y por el elevado número de accesos por hebra que es la candidata perfecta para hospedar la máscara, pues ésta no excede los 64. Esta memoria se deberá declarar como una variable global, fuera de cualquier ámbito, y será inicializada antes de la llamada al *kernel*.

Respecto a la memoria global, queda por discutir las reservas de la imagen de entrada y la imagen de salida. Como se ha establecido, la cantidad de memoria disponible es un factor que limita el tamaño de nuestro problema. Ya que si usamos la computación GPU es, precisamente, para obtener ganancias considerables con problemas de gran tamaño, no podemos permitir que la cantidad de memoria libre nos limite el rendimiento en este tipo de problemas. Se crea un algoritmo que, efectivamente, compruebe la cantidad de espacio libre del que disponemos. En función de éste, la imagen se divide (presumiblemente por filas) en piezas que puedan caber en el device. El procesamiento se lleva a cabo mediante sucesivas llamadas al *kernel* que procesan cada una de las partes de la imagen independientemente. Cada resultado se incorporaría a una matriz global donde se irían acoplando el resto de las partes. Este hecho plantea un problema que no existe en el algoritmo secuencial respecto al tratamiento de bordes. Ya que el algoritmo secuencial trabaja sobre una misma imagen sin dividir debido a que ésta se encuentra en memoria principal, los *pixels* que necesitaban de un vecindario que se escapaba de los límites de la imagen no formaban parte de la suma (es decir, se sumaba 0, el color negro). En esta versión del algoritmo, si vamos a dividir la imagen en piezas tenemos que tener en cuenta que, si lo hacemos en filas, los bordes horizontales no pueden ser tratados de la misma manera. Es decir, no podemos fingir la presencia de un marco negro cuando en realidad ese marco son *pixels* de la parte anterior y/o siguiente de la imagen.

Las imágenes de mayor dimensión no serán aptas para todos los dispositivos ya que no podrán almacenarse en su totalidad en éste. En estos casos, la imagen que será procesada en la GPU será una de las particiones de la imagen original a la que se le han añadido los bordes necesarios para llevar a cabo la convolución de forma coherente.

### 5.3 Paralelización CUDA III: distribución bidimensional

La motivación por la implementación de un algoritmo que use memoria compartida reside en los resultados que podemos obtener debido a las bajas latencias que ésta posee respecto a memoria global. Sin más preámbulos, se procede a mostrar la propuesta de algoritmo:

```
__global__ void convolve ( ** Md imagen . . .  ) {
      extern __shared__ unsigned char Mds[];
      for( ){   //Rojo, Verde, Azul
      //Loading the pixels
      Mds[pixelLocal] = Md[pixelActual];
      <cargar los pixels de los bordes>
```

```
__syncthreads();
for( )   //filas máscara
for( )   //columnas máscara
suma parcial accediendo a Mds
Sustituir pixel por la suma
__syncthreads();
}
```

En él se resaltan en rojo las modificaciones aplicadas al algoritmo CUDA anterior:

- Primeramente se declara el buffer compartido Md. La palabra reservada "*extern*" permitirá asignarle un tamaño dinámicamente en función de la limitación de memoria compartida del device. Este parámetro se asignará en el momento de invocar al *warp*.
- Posteriormente se procede al trasladado del pixel desde la imagen en memoria global a memoria compartida.
- A continuación, si la hebra actual representa un pixel de los bordes se traerían los bordes implicados a memoria compartida.
- Tras ejecutar estos puntos se establece una barrera de sincronización, ya que hasta que todos los *pixels* estén cargados no se procederá a su cómputo, pues hasta ese momento la memoria contendrá datos sin especificar.
- Tras su cómputo es necesario establecer otra barrera de sincronización. De lo contrario, las hebras que hayan finalizado este cómputo comenzarán a cargar el elemento del siguiente color, perjudicando cómputo del color anterior de sus hebras vecinas.

## 6 Experimentos

Se han elegido 4 tamaños de imagen diferentes, siendo éstas de proporciones cuadradas. Sus tamaños son, en orden ascendente: 2800x2800px, 5600x5600px, 11200x11200px y 22400x22400px.

Además, se ha contado con 3 tipos de máscara diferentes que apliquen el algoritmo de la convolución de tamaños 3x3, 5x5 y 7x7.

Cada una de las posibles combinaciones entre imagen y filtro van a ser ejecutadas en las diferentes versiones del algoritmo: algoritmo secuencial, algoritmo CUDA, algoritmo CUDA que usa memoria compartida y un algoritmo en OpenCL que no se ha detallado en este artículo pero sí se utiliza para comparar.

En cuanto al hardware empleado, se trata de un PC con la distribución Ubuntu 12.04 (Linux 3.5.0-36-generic) cuyas características CPU más trascendentes son [24]:

- Nombre de modelo: Intel® Core™ 2 Quad CPU Q6600
- Número de procesadores: 1
- Núcleos CPU: 4
- CPU MHz: 2400

En cuanto a las características de la GPU, podemos destacar:

- Nombre: GeForce GTX 660Ti
- Generación: 3.0
- Nº máximo de hebras por SM: 2048
- Nº de SM: 7
- Memoria global: 2GB
- Nº máximo de registros: 65536

A diferencia de CUDA, OpenCL no nos proporciona mecanismos para conocer detalles concretos sobre la implementación hardware de la GPU que permitan una asignación óptima de los elementos y grupos de trabajo. Por ejemplo, a través de OpenCL no podemos saber el número de elementos de trabajo que cada SM puede permitir simultáneamente, lo cual es un factor determinante para definir la composición de los grupos de trabajo y estimar cuántos de éstos podrán caber en la máquina. Es por esto que el tamaño del grupo de trabajo elegido en OpenCL será el tamaño de bloque optimizado por el algoritmo CUDA, pues el hardware sigue siendo el mismo. Se deduce que en otros dispositivos (Intel, AMD, etc) no podremos valernos de la API de NVIDIA para optimizar el bloque, por lo que el programador deberá encontrar empíricamente el tamaño de grupo de trabajo apropiado para algoritmos OpenCL.

La medición de los tiempos se llevó a cabo mediante la ejecución de dos scripts escritos en *tcsh*, donde cada ejecución se lanza 5 veces para poder computar un promedio que normalice las ejecuciones críticas.

## 7 Resultados

Los tiempos obtenidos, quedan reflejados en la Tabla 1, donde la primera columna indica el tamaño del filtro aplicado y la segunda expresa el ancho y alto de la imagen.

**Table 1.** Tiempo medio (ms) de las 5 ejecuciones lanzadas por cada versión del algoritmo, tamaño de máscara y tamaño de la imagen.

| | | Sec. | CUDA (II) | CUDA (III) | OpenCL | hebras | Tamaño bloque |
|---|---|---|---|---|---|---|---|
| | 2800 | 6194 | 2,13 | 8,83 | 2,14 | 7840000 | 16x16 |
| | 5600 | 25182 | 9,03 | 55,50 | 9,15 | 31360000 | 32x32 |
| | 11200 | 102996 | 36,06 | 202,54 | 30,84 | 125440000 | 32x32 |
| | 22400 | 414130 | 144,33 | 776,14 | 144,84 | 265932800 | 32x32 |
| | 2800 | 16524 | 5,25 | 15,23 | 4,27 | 7840000 | 16x16 |
| | 5600 | 67248 | 21,13 | 93,91 | 18,06 | 31360000 | 32x32 |
| | 11200 | 274914 | 84,51 | 332,51 | 60,89 | 125440000 | 32x32 |
| | 22400 | 1339042 | 300,16 | 1301,62 | 274,84 | 265932800 | 32x32 |
| | 2800 | 32936 | 9,93 | 25,43 | 7,30 | 7840000 | 16x16 |
| | 5600 | 169982 | 39,13 | 143,13 | 25,91 | 31360000 | 32x32 |
| | 11200 | 745736 | 146,37 | 537,32 | 110,38 | 125440000 | 32x32 |
| | 22400 | 3211858 | 553,62 | 1435,05 | 406,02 | 265574400 | 16x16 |

**Table 2.** Desviaciones estándar de cada uno de los tiempos.

| | | Sec. | CUDA (II) | CUDA (III) | OpenCL | hebras | Tamaño bloque |
|---|---|---|---|---|---|---|---|
| 3x3 | 2800 | 26,08 | 0,01 | 0,01 | 0,00 | 7840000 | 16x16 |
| | 5600 | 4,47 | 0,01 | 0,01 | 0,00 | 31360000 | 32x32 |
| | 1200 | 16,73 | 0,00 | 16,00 | 0,00 | 125440000 | 32x32 |
| | 22400 | 113,36 | 0,01 | 27,51 | 3,66 | 265932800 | 32x32 |
| 5x5 | 2800 | 5,48 | 0,01 | 0,01 | 0,00 | 7840000 | 16x16 |
| | 5600 | 8,37 | 0,01 | 0,01 | 0,00 | 31360000 | 32x32 |
| | 11200 | 53,67 | 0,01 | 19,51 | 0,00 | 125440000 | 32x32 |
| | 22400 | 267670,62 | 14,72 | 24,02 | 18,39 | 265932800 | 32x32 |
| 7x7 | 2800 | 1740,99 | 0,01 | 0,02 | 0,00 | 7840000 | 16x16 |
| | 5600 | 34495,12 | 0,01 | 7,84 | 0,00 | 31360000 | 32x32 |
| | 11200 | 118934,51 | 13,39 | 4,76 | 7,89 | 125440000 | 32x32 |
| | 22400 | 1558,24 | 23,38 | 22,34 | 18,64 | 265574400 | 16x16 |

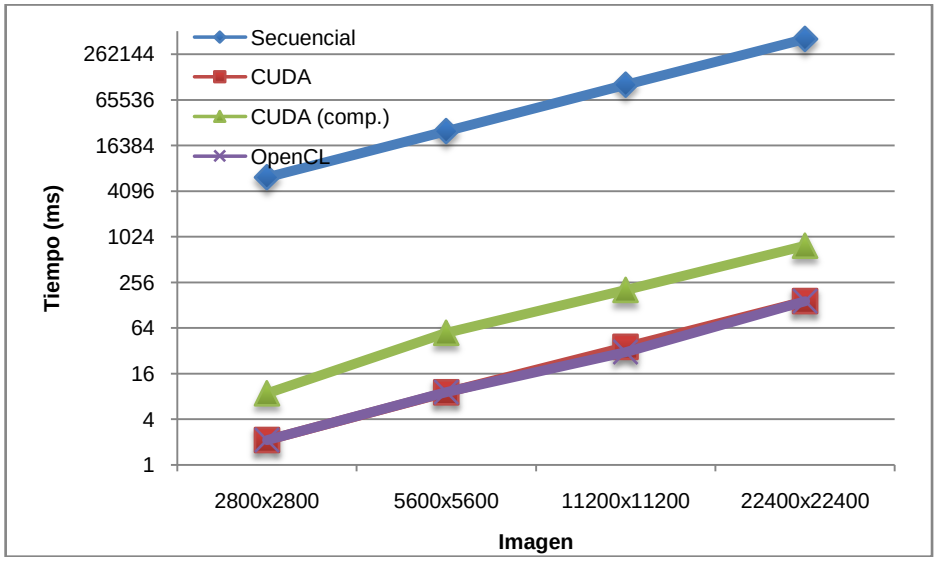


**Fig. 5.** Tiempos para diversas imágenes con una máscara de 3x3. (Escala logarítmica base 2.)

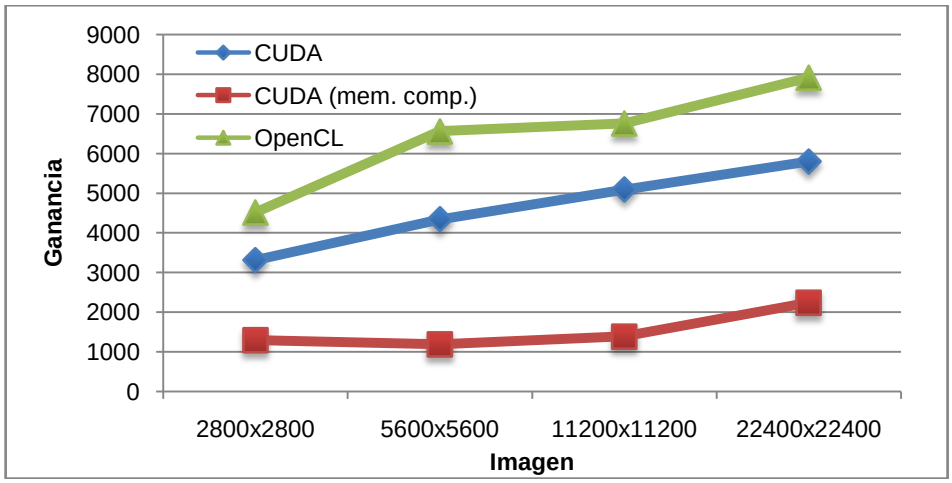


**Fig. 6.** Ganancias de los distintos algoritmos paralelos respecto al secuencial. Se ha tomado el filtro de dimensiones 7x7.

En la Figura 5 se puede apreciar los diferentes tiempos de ejecución obtenidos para las diferentes versiones del algoritmo. Puede observarse que la escala es logarítmica, por lo que los tiempos se mejoran de forma exponencial conforme aumenta el tamaño de la imagen a tratar. También queda de manifiesto que los tiempos obtenidos con la versión de CUDA que utiliza memoria compartida no es todo lo buena que se esperaba, resultados que se explican debido al tipo de dato utilizado en cada pixel de la imagen, puesto que se ha representado la imagen utilizando bytes para maximizar la capacidad de la GPU y esto provoca colisiones en el acceso a la memoria del *device*. Por otra parte, el coste en tiempo que supone traer los datos desde memoria global hasta memoria local no se compensa con las operaciones que se realizan con cada dato, por lo que los tiempos de la versión de memoria compartida son peores. Respecto a los tiempos de la versión en OpenCL y la versión CUDA son prácticamente similares, por lo que ambas tecnologías son capaces de proporcionar un rendimiento óptimo de la GPU.

Utilizando estos tiempos (Figura 5) hemos calculado la ganancia obtenida para cada tipo de imagen y cada versión del algoritmo (ver Figura 6). Como se puede apreciar las ganancias obtenidas son superlineales para todos los casos.

## 8 Práctica propuesta. Beneficios esperados.

Durante el desarrollo del PFC se tuvo en mente siempre que era un trabajo de investigación para ver hasta donde se podía explotar una GPU para realizar una tarea de tratamiento de imágenes altamente parametrizable y paralelizable. Sin embargo, a la terminación del mismo, se pensó que quizás todo ese trabajo debería ser aprovechado por otros alumnos, añadiéndole así un grado más de utilidad al proyecto para que éste pudiera ser reutilizado por otros estudiantes.

Por otra parte, los alumnos no siempre se sienten implicados o motivados a realizar PFC o Proyectos Fin de Grado, PFG, relacionados con la Computación Paralela, puesto que en la mayoría de los casos les atraen más otros proyectos o simplemente les parece una tarea de alta complejidad el abordar un trabajo de programación y optimización de código paralelo. Sin embargo creemos que plantearles una práctica que puedan realizar utilizando un código desarrollado en un PFC anterior, podría ser algo positivo que dota a la práctica de un plus para la motivación del alumnado puesto que podrán ver que compañeros como ellos son capaces de realizar un código optimizado y adaptado a una máquina tan bueno como cualquier otro desarrollado por profesores, editores de bibliografía o investigadores especializados en esta tarea y que, además, ellos son capaces de modificar, optimizar y utilizar sin problemas.

Por último, esta práctica está planteada para que manejen y apliquen conceptos implicados en Computación Paralela y que afectan claramente a la escalabilidad, eficiencia y rendimiento del algoritmo y que no en todos los casos son capaces de asimilar. Concretamente nos referimos a los conceptos de granularidad y mapeo de una tarea que debe llevar a cabo un algoritmo en una máquina concreta de ejecución.

El PFC explicado en la sección anterior está planteado asumiendo que el tamaño de una tarea básica es el procesamiento de un pixel de la imagen, es decir, cada hebra del algoritmo procesa la convolución de un pixel produciendo así un algoritmo de granularidad fina y este principio se ha mantenido en todas las versiones del algoritmo. Es decir, existirán tantas tareas independientes como *pixels* tenga la imagen a procesar y cada una de estas tareas será ejecutada por una hebra diferente. Pero, qué pasaría si nuestra tarea básica no fuera el procesamiento de un pixel, sino que este valor fuera algo variable del algoritmo. Este segundo enfoque puede apreciarse en la imagen que representa la Figura 7.

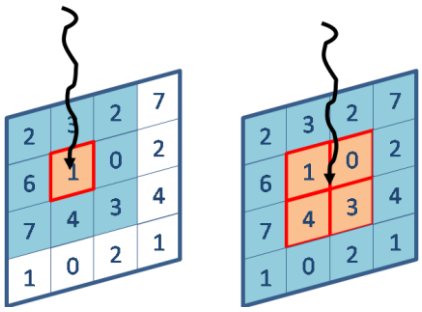


**Fig. 7.** En esta figura se aprecia la diferencia entre considerar una tarea básica el procesamiento de un sólo pixel, parte izquierda de la figura, frente a considerar que una tarea básica está formada por el procesamiento de 4 *pixels*, representada en la parte derecha de la imagen. En ambos casos, la tarea básica se procesa por una sola hebra de procesamiento, representado en la figura por una flecha vertical ondulada.

La práctica consiste en que el alumno sea capaz, partiendo ya de algunas de las versiones del algoritmo, modificar la granularidad del mismo, viendo cómo al cambiar la granularidad, es posible cambiar los tiempos de ejecución, tanto en un sentido, disminuyéndolos, como en otro, aumentándolos. Cuando el tamaño de la tarea básica aumenta, lo normal es que los tiempos de ejecución disminuyan, puesto que cada hebra

creada tendrá más trabajo a procesar y por lo tanto el tiempo que se ha empleado en su creación, será compensado por la cantidad de trabajo que es capaz de llevar a cabo en paralelo con las demás hebras. Sin embargo, esta disminución del tiempo compensará sólo en el caso de que el trabajo realizado por todas las hebras en paralelo no provoque otro tipo de sobrecarga adicional que ocasiona otros retrasos. Estos otros retrasos surgen, por ejemplo, cuando varias hebras necesitan acceder al mismo conjunto de *pixels* o cuando el número de hebras necesarias para procesar la imagen disminuye de tal manera, al dividirse el trabajo a realizar entre ellas, que la GPU no puede ser aprovechada de forma completa y simultánea.

El trabajo pedido a los alumnos consistirá en la parametrización del tamaño de la tarea básica y en el estudio de la influencia de este parámetro en una de las versiones del algoritmo, concretamente en la versión de CUDA que utiliza memoria global.

Los resultados esperados con esta práctica se pueden ver de manifiesto en el siguiente ejemplo que se ha llevado a cabo utilizando una imagen de 3000x4000 *pixels* en formato RGB junto con una máscara pequeña de tamaño 3x3. Los resultados se pueden ver en la figura 11, donde se puede apreciar que al ir modificando el tamaño de la cantidad de trabajo que cada hebra debe realizar se obtiene un tiempo medio de unos 183 ms, conforme aumentamos la granularidad, el tiempo de ejecución disminuye hasta valores medios de 179 ms, unos 4 milisegundos menos en media, optimización que teniendo en cuenta que la unidad de medida son los milisegundos, es apreciable. A partir de este valor, al aumentar ya el tamaño de las tareas y pasar a tamaños de granularidad superiores a 4, es decir que cada hebra procese un cuadrado de la imagen de tamaño 5x5 o 6x6 los tiempos de ejecución vuelven a empezar a subir. Es en este punto donde podemos afirmar que ya un aumento de la granularidad no compensa en términos de tiempo de ejecución. La gráfica que representa dichos resultados se puede apreciar en la gráfica siguiente.

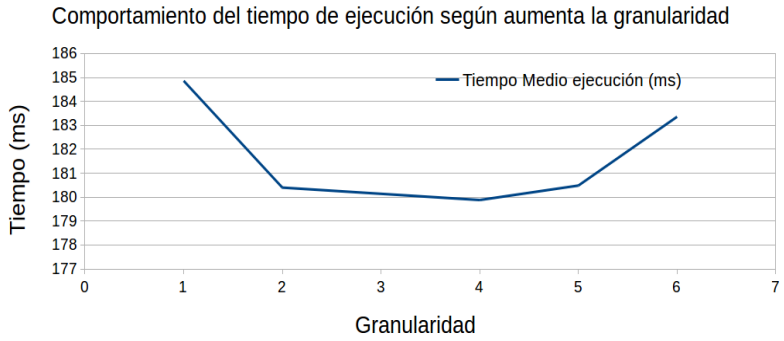


**Fig. 8.** Ilustración de cómo el tiempo de ejecución disminuye al aumentar la granularidad de 1 a 4. A partir de este punto, el aumentar la granularidad no mejora el tiempo de ejecución, sino que lo empieza a empeorar, puesto que el número de hebras es menor y seguramente ya no son suficientes para ocultar la latencia de los accesos a memoria.

## 9 Conclusiones

En el estudio realizado se puede apreciar como una GPU es capaz de alcanzar ganancia superlineales para el procesamiento de cualquier imagen, sea del tamaño que sea, teniendo en cuenta que se trata de un algoritmo muy paralelizable y que se puede adaptar fácilmente a la arquitectura paralela de una GPU. Concretamente, los resultados incluyen ganancias de más de 7000x en imágenes de gran tamaño para la versión realizada con OpenCL y de más de 7000x en el caso de la versión realizada con CUDA.

Esto pone de manifiesto que no hay que realizar ninguna presunción sobre la paralelización de algoritmos en arquitecturas tan específicas como una GPU y que cualquier límite puede ser superado sin problemas por un algoritmo bien organizado y bien estructurado adecuándolo de la mejor forma a la arquitectura que subyace en el plano de la ejecución.

Por otra parte, se plantea con este proyecto su reutilización para una de las asignaturas de la especialidad de Ingeniería de Computadores, concretamente para la asignatura "Arquitectura y Computación de Altas Prestaciones". En dicha práctica se plantea que los alumnos cambien la granularidad de una de las versiones del algoritmo paralelo que se describe en este trabajo para que realicen un estudio de la influencia de esta característica en los tiempos de ejecución. De este modo se propone una forma de abordar, por una parte, el entendimiento del concepto de granularidad y, por otra, la puesta en práctica de técnicas de programación paralela impartidas en la parte teórica de la asignatura como es la agrupación de tareas de igual tipo y tamaño, disminuyendo así el número total de tareas a realizar por la GPU y por lo tanto del número de hebras necesarias para la conclusión del trabajo de procesamiento de una imagen.

Este trabajo persigue a su vez, una reutilización de un PFC, que en la mayoría de los casos, es trabajo que no vuelve a utilizarse una vez superada dicha materia y que por lo general, el alumno deja de mantener y mejorar para dedicarse al mundo laboral o simplemente porque no se ha planteado en el diseño del proyecto que pudiera llegar a ser un producto útil, cosa que sólo ocurre en unos pocos casos de todos los proyectos fin de carrera desarrollados.

## Referencias

[1] Yan Yong and Zhang Xiaodong, "Profit-effective parallel computing," *Concurrency*, vol. 7, no. 2, April 1999.

[2] NVIDIA Corporation. (2015, January) nvidia.com. *URL* http://www.nvidia.com/object/what-is-gpu-computing.html

[3] Khronos Group. (2014, December) khronos.org. *URL* https://www.khronos.org/opencl/

[4] Wikipedia Inc. (2014, September) wikipedia.org. *URL*

http://es.wikipedia.org/wiki/Convolución

[5] Javier Delgado, Joao Gazolla, Esteban Clua, and S. Masoud Sadjadi, "A Case Study on Porting Scientific Applications to GPU/CUDA," Miami, U.S.A., 2010.

[6] Ren Wu, Bin Zhang, and Hsu Meichun, "GPU-Accelerated Large Scale Analytics," HP Laboratories, 2009.

[7] Nesrin Aydin Atasoy, Baha Sen, and Burhan Selcuk, "Using gauss - Jordan elimination method with CUDA for linear circuit equation systems," Karabuk University, Engineering Faculty, Karabuk,TURKEY, 2012.

[8] Douglas C.C and Hyoseop Lee, "Basket Option Pricing Using GP-GPU Hardware Acceleration," Univ. of Wyoming, Laramie, WY, USA, 2010.

[9] R. Ammendola et al., "The GAP project - GPU for realtime applications in high energy physics and medical imaging," in *Nuclear Science Symposium and Medical Imaging Conference (NSS/MIC), 2013 IEEE*, Seoul, 2013.

[10] G. Poli, J.H. Saito, J.F. Mari., and M.R. Zorzan, "Processing Neocognitron of Face Recognition on High Performance Environment Based on GPU with CUDA Architecture," in *Computer Architecture and High Performance Computing, 2008. SBAC-PAD '08. 20th International Symposium on*, Campo Grande, MS, 2008.

[11] Inc Tabor Communications. HPC wire. *URL* http://www.hpcwire.com/2011/12/15/emerging_companies_ride_wave_of_gpu_computing/

[12] Andreas Reich and Jen-Hsun Huang. Youtube. *URL* http://youtu.be/wNSWWOf6-Hw

[13] Migel Bordallo López, Henri Nykanen, Jari Hannuksela, Olli Silvén, and Markku Vehvil¨ainen, "Accelerating image recognition on mobile devices using GPGPU," University of Oulu, Oulu, Oulu, Finland, 2011.

[14] O. Cetin and G. Yilmaz, "GPGPU accelerated real-time potential field based formation control for Unmanned Aerial Vehicles," in *Unmanned Aircraft Systems (ICUAS), 2014 International Conference on*, Orlando, FL, 2014.

[15] NVIDIA Corporation. (2014, December) NVIDIA. *URL* http://www.nvidia.es/object/corporate-timeline-es.html

[16] John Nickolls and William J. Dally, "THE GPU COMPUTING ERA," NVIDIA, Santa Clara, CA, 2010.

[17] Stack Exchange inc. Stack Overflow. *URL* http://stackoverflow.com/search?q=CUDA

[18] Wikipedia Inc. (2014, November) wikipedia.org. *URL* http://es.wikipedia.org/wiki/Operador

[19] Wikipedia Inc. (2014, November) wikipedia.org. *URL* http://es.wikipedia.org/wiki/Función_matemática

[20] GIMP. (2013) docs.gimp.org. *URL* http://docs.gimp.org/en/plug-in-convmatrix.html

[21] Adobe Systems Incorporated. (2014, December) http://helpx.adobe.com. *URL* http://helpx.adobe.com/premiere-pro/using/video-effects-transitions.html

[22] The Mathworks Inc. (2014, December) es.mathworks.com/. *URL* http://es.mathworks.com/help/matlab/math/convolution.html

[23] Wolfram. (2014, December) reference.wolfram.com/. *URL* https://reference.wolfram.com/language/ref/Convolve.html

[24] Stack Exchange Inc. (2011, January) unix.stackexchange.com. *URL* http://unix.stackexchange.com/questions/6345/how-can-i-get-distribution-name-and-version-number-in-a-simple-shell-script

# Proyecto Prácticas Procesadores Integrados: Self-Balancing Robot basado en Arduino

Carlos Bailón[1] and Antonio F. Díaz[1]

Departamento de Arquitectura y Tecnología de Computadores.
Universidad de Granada
cbailon37@correo.ugr.es, afdiaz@ugr.es

**Resumen** Procesadores Integrados es una asignatura que se imparte en el Grado en Ingeniería Electrónica Industrial donde los estudiantes adquieren la capacidad de desarrollar pequeños proyectos basados en microcontrolador. En este artículo se presenta uno de estos proyectos: un robot auto-balanceado basado en Arduino. Se describen los elementos principales a considerar en el diseño hardware y software. La ventaja de esta configuración es que permite evaluar un sistema de control PID en tiempo real a partir de la información obtenida por un acelerómetro y un giróscopo implementados en un sensor MEMS.

**Palabras clave:** Control PID, microcontrolador, Arduino, Sensor MEMS.

**Abstract** Integrated Processors is a course in the Electronic Engineering Degree where students acquire the ability to develop small proyects based on microcontrollers. In this paper we present one such projects: a self-balanced robot based on Arduino. The main elements to consider in the hardware and software design is described. The advantage of this configuration is that allows evaluating a PID control system in real time from information obtained by a accelerometer and a gyroscope implemented on a MEMS sensor.

**Keywords:** PID control, microcontroller, Arduino, MEMS sensor.

## 1. Introdución

La asignatura Procesadores Integrados se imparte en 3° del Grado en Ingeniería Electrónica Industrial de la Universidad de Granada y se centra en el estudio de microprocesadores, microcontroladores y procesadores de señales digitales (DSP). En la introducción se muestran las características principales de éstos y a continuación se empieza con el bloque relacionado con los microcontroladores, evaluando las posibilidades que ofrecen. Se estudian diversos microcontroladores para definir criterios de selección óptimos en función de las características del diseño y se analiza la información que ofrecen los fabricantes de microcontroladores de sus productos. También se analizan las

etapas para el diseño y se revisan buses y distintos métodos de interconexión tanto con otros sistemas como con dispositivos, evaluando las posibilidades de conexión de elementos de entrada y salida así como las tecnologías de memorias empleadas con microcontroladores. La ventaja de comenzar con el bloque de microcontroladores es que el estudiante adquiere la base suficiente para empezar a realizar prácticas en un corto de periodo de tiempo debido a la rápida curva de aprendizaje, por lo que el resto de la asignatura puede comprenderse desde un punto de vista más práctico.

Además, en la asignatura se estudian: Los elementos internos de los procesadores para incrementar sus prestaciones basadas en el paralelismo interno del procesador, las características de los procesadores segmentados y superescalares, cómo el tratamiento de las dependencias influye en el diseño del cauce de instrucciones, cómo la emisión de instrucciones desordenada en los procesadores superescalares reduce el número de ciclos que como media requiere una instrucción para su ejecución, técnicas de ejecución especulativa e identificar ciertos elementos necesarios en la implementación de los procesadores modernos, como por ejemplo las estaciones de reserva, o el buffer de reordenación, las limitaciones en cuanto al rendimiento de los procesadores segmentados y superescalares y elementos que penalizan su rendimiento, las extensiones multimedia incorporadas en los procesadores actuales, por qué los procesadores multihebra mejoran las prestaciones, qué elementos internos incorporan los procesadores para dar soporte al sistema operativo, la diferencia entre paralelismo de datos y paralelismo funcional.

También se estudian otros elementos directamente relacionados como son: cómo se implementan los buses en los sistemas basados en microprocesadores y el uso de los chipsets, diferencias entre los distintos buses y conocer sus prestaciones y forma de funcionamiento, buses de alto rendimiento para el procesador, la necesidad de disponer de una jerarquía de memoria así como el concepto de memoria virtual como solución al problema de la capacidad de almacenamiento, distintos tipos de memoria empleados para la memoria principal así como los principios básicos de las memorias caché, la importancia del diseño de las E/S en las prestaciones globales del sistema, la estructura y principios de funcionamiento de los controladores de E/S, el impacto de las interrupciones y como se gestionan, las distinta técnicas de E/S.

Finalmente se analiza la arquitectura interna de los DSPs y cómo está orientada al tratamiento de señales, las posibilidades de aplicación de los DSPs así como las diferencias de arquitectura frente a los procesadores de propósito general.

Partiendo de estos conocimientos, en la parte práctica los estudiantes deben desarrollar como proyecto un sistema electrónico basado en microcontrolador. En este sentido las plataformas basadas en Arduino permiten que los estudiantes adquieran rápidamente dicha capacidad ya que existen gran cantidad de documentación y recursos. En este artículo se presenta uno de los trabajos desarrollados para la evaluación de la parte práctica de la asignatura. En particular el desarrollo de un robot autobalanceado basado en Arduino.

Básicamente, este robot se mantiene en equilibrio sobre dos ruedas. El microcontrolador monitoriza su posición vertical mediante un acelerómetro de 3 ejes y un giróscopo de 3 ejes. Dicha información es la que se utiliza en el PID que controla los motores en tiempo real, permitiendo mantenerse vertical. En la Figura 1 se muestra el aspecto final que tiene el sistema propuesto.

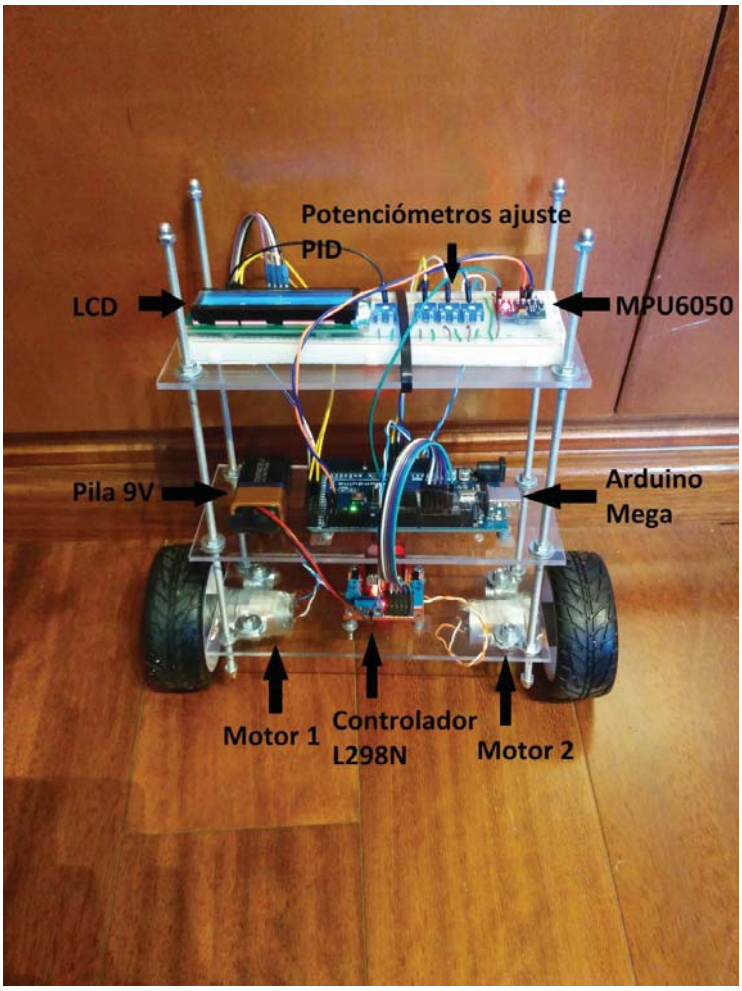


**Figura 1.** Aspecto del robot auto-balanceado.

## 2. Descripción hardware del sistema

Los elementos principales del sistema son:

- Microcontrolador Arduino Mega 2560
- Sensor MEMS MPU6050
- Motores y puente H
- Pantalla y calibración

*La plataforma Arduino*

Para este diseño se ha utilizado el Arduino Mega 2560 que está basado en el microcontrolador ATmega2560 con una frecuencia de 16 MHz, aunque se podía haber utilizado otras placas de Arduino más reducidas.

*MPU6050*

El MPU6050 es un circuito MEMS (Micro-Electro-Mechanical System) que combina un acelerómetro de 3 ejes y un giróscopo de 3 ejes para medir las inclinaciones y rotaciones e incluye un DMP (Digital Motion Processor). Esta unidad permite realizar complejos cálculos con las medidas captadas por el chip, ahorrando tiempo y trabajo al microcontrolador. Este elemento se encarga de la captación de datos así como de procesar información de posibles magnetómetros externos. La Figura 3 muestra una pequeña placa con el MPU6050.

El chip va orientado de forma que su eje Y sea perpendicular al robot, de modo que sólo es necesaria medir la inclinación en ese eje. La conexión se realiza mediante I2C, los pines SDA (datos) y SCL (reloj), más los de alimentación y un pin de interrupción (INT0). Si se desea modificar la dirección I2C del dispositivo se dispone de otro pin (AD0), que modifica el último bit de dicha dirección, B110100X, donde X indica el estado de este pin.

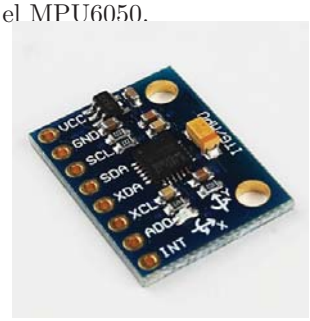


**Figura 2.** MPU6050

En nuestro sistema, captamos los datos de entrada del acelerómetro en unas coordenadas conocidas como Yaw-Pitch-Roll (figura 3), que son usadas para medir rotaciones en aeronáutica. De este modo, solo utilizamos la segunda coordenada (eje Y), que será nuestra señal de entrada. Este componente se encuentra en el nivel superior del robot, insertado en una protoboard con más componentes, de manera que se reduce el cableado ya que la alimentación a 5V es común.

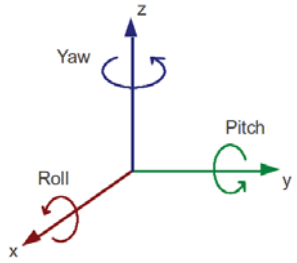


**Figura 3.** Sistema de coordenadas.

*Motores*

Permiten el movimiento de las ruedas del robot para su equilibrio y se controlan mediante salidas del microcontrolador. En este diseño se utilizan dos motores con sistema de reducción incorporado. Trabajan entre 3

y 9 V y dan una velocidad de salida nominal de 281 rpm (a 6V). Es fundamental que estos motores ofrezcan un gran torque, ya que nuestro objetivo es que el robot ofrezca una respuesta rápida y robusta ante cambios en la inclinación, para lo que es necesaria bastante fuerza. Las ruedas utilizadas son de 8 cm de diámetro.

*Puente H*

Para el control de los motores se utiliza una placa con el circuito integrado L298N (Figura 4). Este chip contiene puente H, que mediante una estructura simétrica de pares darlington y diodos permite el control de la velocidad y el sentido de giro de los motores. La placa necesita alimentación externa acorde con la necesaria para los motores, a través de la cual se extrae también la alimentación de 5V para toda la electrónica (mediante un regulador linear LM7805). Podemos controlar ambos motores mediante 3 pines cada uno, 1 para la regulación de velocidad (se utiliza el PWM de Arduino) y 2 para el control del sentido de giro (ambos conectados, el motor se enclava, ninguno el motor se mueve libremente en punto muerto, y uno solo determina el sentido de giro). Podemos utilizar esta alimentación para alimentar la placa de Arduino. Los motores y el controlador están en el nivel más bajo del robot.

**Figura 4.** Puente H basado en L298N.

*Alimentación*

Para la alimentación se utiliza una pila de 9V que se conecta directamente a la placa controladora de motores, de la cual se extrae la alimentación regulada a 5V para el resto de la electrónica del robot. La pila está situada en el nivel medio junto a la placa de Arduino.

La Tabla 1 muestra el presupuesto de los componentes utilizados en el proyecto.

## 3. Descripción software del sistema

*Recursos utilizados*

El MPU6050 se comunica por I2C, y su utilización es compleja, por lo que se han usado dos bibliotecas, *I2Cdevlib* y *MPU6050_6Axis_MotionApps20*, desarrolladas por Jeff Rowberg, que facilitan enormemente la captura de datos válidos para procesamiento. La biblioteca para I2C *Wire* también es necesaria para utilizar las dos anteriores.

Por otro lado, para la monitorización del proceso se envían datos por puerto serie, tanto el ángulo de inclinación del robot como las constantes del control PID mientras se está calibrando, siempre y cuando se seleccione la opción correspondiente al inicio del código. Además, cada 5 segundos se guardan los

| Elemento | Cantidad | Precio unidad (€) | Precio total (€) |
|---|---|---|---|
| ***Chasis y elementos mecánicos*** | | | |
| Ruedas (Ø 55 mm) | 2 | 4,90 | 9,80 |
| Ejes ruedas (5 cm) | 2 | 0,15 | 0,30 |
| Juntas ejes motor - rueda | 2 | 0,30 | 0,60 |
| Motor CC con reductora 6V 281 rpm | 2 | 6,39 | 12,78 |
| Abrazaderas fijación motores | 2 | 0,32 | 0,64 |
| Planchas metacrilato (180x80x3 mm) | 3 | 1,58 | 4,74 |
| Varilla roscada (20 cm) | 4 | 0,32 | 1,28 |
| Tuercas varillas roscadas | 24 | 0,03 | 0,72 |
| Tornillos fijación placas | 8 | 0,03 | 0,24 |
| Topes varillas roscadas | 4 | 0,06 | 0,24 |
| **Total chasis** | | | **31,34** |
| ***Electrónica*** | | | |
| Arduino Mega 2560 Rev3 | 1 | 35,00 | 35,00 |
| Acelerómetro MPU6050 | 1 | 4,20 | 4,20 |
| Motor Driver Board L298N | 1 | 5,70 | 5,70 |
| Protoboard MB-102 | 1 | 3,09 | 3,09 |
| Cables macho-hembra | 15 | 0,03 | 0,45 |
| Pila 9V | 1 | 3,28 | 3,28 |
| **Total electrónica** | | | **51,72** |
| **Precio Final** | | | **83,06** |

**Tabla 1.** Presupuesto del proyecto.

datos de inclinación del robot en la memoria EEPROM externa, por si es necesario hacer una gráfica a largo plazo de los niveles de inclinación. Para ello se han utilizado interrupciones, mediante el Timer 3 de Arduino Mega (evitando así bloquear funcionalidades de otro Timer). Se han utilizado las bibliotecas *EEPROM* y *TimerThree*.

Para el control de motores se ha creado una biblioteca propia, que se ha llamado *LMotorController*. Esta dispone de más funcionalidades que las que usa el robot (ya que éste sólo hace un movimiento rectilíneo, y la biblioteca permite girar y mover ambas ruedas con velocidades diferentes), pero está pensada para poder usarla de forma más genérica. Por último, se ha incorporado la configuración del watchdog, para lo que se utiliza la biblioteca *wdt*, incorporada en la IDE de Arduino, y que con simples órdenes permite configurar el watchdog del sistema. En este caso se ha configurado para lanzar un reset si no se reinicia en 8 segundos.

*Control PID*

La placa de Arduino Mega 2560 se encuentra en el segundo nivel del robot, y es la encargada de hacer los cálculos para el procesamiento de la señal de entrada.

Este procesamiento se hace mediante un control PID, para lo que recurrimos a la biblioteca *PIDv1*, que se encuentra en la página oficial de Arduino. Para ajustar el control, se utilizan 3 potenciómetros de 10k (uno para cada una de las

constantes), situados en la placa de prototipado. Se monitorizan los valores de las constantes por puerto serie, de forma que se pueda conocer cuál es el valor óptimo.

Para realizar el primer ajuste, se ponen a cero los potenciómetros, por lo que el robot no hace nada (constantes cero, no hay control PID), y se va aumentando el que corresponde a la constante proporcional *Kp*, hasta que alcanzamos un valor para el cual los motores del robot responden a los datos del acelerómetro. En nuestro caso ha sido $Kp = 70$. Una vez fijada, pasamos a variar el potenciómetro correspondiente a la constante integral *Ki*. Ésta se encarga de reducir el error en estado estacionario, por lo que la aumentamos hasta el valor para el cual la oscilación de los motores es pequeña. En nuestro caso es $Ki = 240$. Finalmente ajustamos la constante diferencial *Kd*, que reduce el sobredisparo, hasta el punto en el que el movimiento del robot es suave y no hay picos de fuerza. $Kd = 1.9$. Una vez ajustadas manualmente se introducen en el código las constantes obtenidas para simplificar la ejecución del programa. En cada bucle se llama a un método que implementa el algoritmo PID y nos da la señal de salida.

*Descripción del código*

Se comienza incluyendo las bibliotecas necesarias y definiendo una serie de variables que nos permiten hacer una monitorización del sistema. Son 3 variables que dependiendo de si valen 1 ó 0 habilitan ciertas partes del código (para ver los valores del acelerómetro por puerto serie, para ver las constantes PID si se están sintonizando de forma manual y para el propio ajuste manual). A continuación se declaran los objetos *mpu*, *pid* y *controlMotores* para las bibliotecas correspondientes, y se declaran las variables necesarias para el funcionamiento del sistema. Respecto al MPU6050, se incluyen variables para detectar interrupciones mediante el pin de interrupción asignado, otras para comprobar el estado del dispositivo tras cada operación y otras más para trabajar con la memoria FIFO que incorpora el sensor, la cual nos permite hacer un almacenamiento de los datos para su lectura. Se ha configurado también un búfer para almacenar los datos. El resto de variables corresponden a los vectores necesarios para obtener los datos y procesarlos.

Después se definen las variables utilizadas para el PID y el controlador de motores y una variable de temporización para almacenar valores de tiempo y ejecutar la captación de datos para sintonización manual en ciertos intervalos (si ha sido seleccionada previamente). En concreto compara constantemente el tiempo mediante *millis()* y ejecuta la rutina cada segundo.

En el *setup*, lo primero que se hace es iniciar el bus I2C y el puerto serie, y después se realizan todas las inicializaciones y conexiones necesarias en el sensor MPU6050. También se ha incluido una rutina para detectar errores en el inicio, con su correspondiente indicación en el monitor serie. Después se configuran el control PID, el Timer para el envío de datos a la EEPROM y el watchdog. En el *loop* lo primero que se hace es el cálculo de los valores de salida del control PID y la actuación de los motores, así como el bucle correspondiente a la variable de temporización señalada anteriormente. Tras ello se efectúa la obtención de los datos, con una rutina para el desbordamiento de la memoria

FIFO (lo cual no debería ocurrir). Para obtener los datos, primero se obtienen en *cuaterniones*, que son los valores primarios (conocidos como *raw values*) que obtiene el sensor. Estos valores son un tipo de coordenadas complejas para medidas de rotaciones, previas a la invención de otras medidas como los ángulos de Euler o las coordenadas Yaw-Pitch-Roll.

Por último se transforman a coordenadas YPR y se muestran en tiempo real por el monitor serie si se ha seleccionado dicha opción al comienzo del código. Tras ello se actualiza el valor de entrada del control PID y se reinicia el watchdog. Una vez fuera del *loop*, se define el método *envíoDatos*, que se ejecuta en cada interrupción, y que cada 5 s guarda el valor del ángulo en la memoria EEPROM. El hecho de que se configure cada 5 s es porque la EEPROM tiene un número limitado de escrituras por celda, que no queremos rebasar. También se declara el método *sintonizacionManual* (correspondiente al bucle temporizado anterior, y solo se ejecuta si hemos puesto a 1 la variable correspondiente al principio), que nos permite captar los datos de las constantes PID a través de los potenciómetros en cada iteración del bucle.

## 4. Conclusiones

En este artículo se ha presentado un trabajo de prácticas de la asignatura Procesadores Integrados. Este trabajo consiste en un robot autobalanceado basado en Arduino. Uno de los principales problemas que plantea este diseño es la posición del acelerómetro. Al estar insertado en una placa de prototipado, la fijación no es tan exacta como si estuviera atornillado, por lo que hay que asegurarse de que no se varía la inclinación del mismo si el robot sufre alguna caída, ya que cualquier variación de la inclinación supone el fin del equilibrio. En general, el comportamiento del robot una vez calibrado es bastante estable.

## Referencias

1. Asignatura Procesadores Integrados. Grado Ingeniería Electrónica Industrial. Universidad de Granada. https://swad.ugr.es/es?crs=7267
2. Ogata K. Ingeniera de control moderna. Pearson-Prentice Hall, 5a ed. (2010)
3. Arduino: http://www.arduino.cc/
4. Jeff Rowberg.*MPU6050_6Axis_MotionApps20*. https://github.com/jrowberg/i2cdevlib/tree/master/Arduino/MPU6050
5. Biblioteca PID Arduino. http://playground.arduino.cc/Code/PIDLibrary

# Instrucciones para Autores

*Enseñanza y Aprendizaje de Ingeniería de Computadores (Teaching and Learning Computer Engineering)* es una revista de Experiencias Docentes en Ingeniería de Computadores que edita el Departamento de Arquitectura y Tecnología de Computadores de la Universidad de Granada, se publica anualmente, y se difunde tanto en papel como electrónicamente, a través del repositorio institucional de la Universidad de Granada (http://digibug.ugr.es/).

Los artículos remitidos para su evaluación pueden estar escritos en castellano o inglés, incluyendo un resumen y palabras clave en inglés en caso de que estén escritos en castellano, y deben seguir el formato descrito en la dirección web:

http://atc.ugr.es/pages/actividades_extension/

El correspondiente fichero .pdf debe enviarse a la dirección de correo electrónico jortega@ugr.es o mdamas@ugr.es

Los artículos deben abordar, tanto contenidos relacionados con la docencia universitaria en general, como con la docencia de asignaturas específicas impartidas por las áreas de conocimiento involucradas en estudios relacionados con la Ingeniería de Computadores, y también pueden aspectos relativos a las competencias profesionales y la incidencia de estos estudios en el tejido socio-económico de nuestro entorno.

En particular, se anima a antiguos alumnos de los estudios de Informática y a estudiantes de grado y posgrado a que envíen colaboraciones relacionadas con sus experiencias al cursar asignaturas relacionadas con la Ingeniería de Computadores, sugerencias, propuestas de mejora, etc.

# Teaching and Learning Computer Engineering

Journal of Educational Experiences on Computer Engineering

May 2015, Number 5