Enseñanza y Aprendizaje de Ingeniería de Computadores

Revista de Experiencias Docentes en Ingeniería de Computadores

Número 6, Junio 2016

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 6, Año 2016

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

Agradecemos su contribución a los autores de los nueve trabajos incluidos en este sexto número de la revista Enseñanza y Aprendizaje de Ingeniería de Computadores. Cinco de esos trabajos provienen de la Universidad de Granada (dos de ellos corresponden a contribuciones de estudiantes), uno proviene de la Universidad Complutense de Madrid, otro de la Universidad Carlos III de Madrid, y dos de la Universidad de Jaén. También agradecemos al Comité Editorial de la revista su labor de difusión de la revista y de revisión de los artículos.

Ya hace más de cinco años que empezaron a impartirse en la Universidad de Granada estudios de grado según las directrices del Espacio Europeo de Educación Superior (EEES), la denominada reforma de Bolonia. Concretamente, en el curso 2010/2011 pusieron en marcha las primeras asignaturas relacionadas con la Ingeniería de Computadores en el Grado en Ingeniería Informática, por lo que en este curso 2015/2016 se gradúa la tercera promoción de ese Grado. Es posible, por tanto, abordar ya el análisis de los resultados de la docencia en las asignaturas impartidas durante algunos cursos, para poner de manifiesto los efectos de las estrategias docentes que se están aplicando. Así, en el artículo "Reflexiones sobre el EEES tras los primeros cursos de la asignatura Arquitectura de Computadores" de J. Ortega, M. Anguita, J. González y M. Damas, se toma como referencia la asignatura de Arquitectura de Computadores de segundo del grado en Ingeniería Informática de la Universidad de Granada, que está siendo impartida por varios de los autores del artículo, para analizar algunas de las situaciones en las que se ha traducido la implementación de la reforma en nuestro país.

En números anteriores de la revista se ha hecho referencia a la importancia de utilizar referencias a la historia de la evolución de los computadores para motivar el aprendizaje de estudiantes, y se han publicado algunos artículos sobre estos tópicos. En "Explotación de la potencia de procesamiento mediante paralelismo: un recorrido histórico hasta la GPGPU" de F. Charte, A.J. Rivera, F.J. Pulgar, y M.J. del Jesús, de la Universidad de Jaén, se lleva a cabo un recorrido histórico del paralelismo implementado en el hardware disponible en cada momento, y cómo se ha buscado obtener el mayor provecho del mismo. Un trabajo muy oportuno teniendo en cuenta la tendencia hacia microprocesadores cada vez con más núcleos de procesamiento y hacia arquitecturas paralelas heterogéneas que plantean importantes retos de cara al desarrollo de códigos eficientes.

Otro tema estrella en la docencia de muchas asignaturas relacionadas con la Ingeniería de Computadores es el aprendizaje del lenguaje ensamblador y de conceptos de

microprogramación ligados con la comprensión del funcionamiento del procesador. Así, en "WepSIM: Simulador modular e interactivo de un procesador elemental para facilitar una visión integrada de la microprogramación y la programación en ensamblador" de A. Calderón, F. García y J. Prieto, de la Universidad Carlos III de Madrid, se describe una herramienta que ofrece un equilibrio entre simplicidad y el detalle de plataformas reales para la enseñanza integrada de la microprogramación y de la programación en ensamblador, permitiendo distintos juegos de instrucciones y la posibilidad de uso a través de dispositivos móviles, a través de un diseño modular que facilita modificar los elementos existentes.

En el ámbito del uso de nuevas metodologías docentes en asignaturas de Ingeniería de Computadores, el presente número incluye dos contribuciones. Por un lado, en "Gamificación en procesos de autoentrenamiento y autoevaluación. Experiencia en Arquitectura de Computadores" de M. Espinilla, A. Fernández, J. Santamaría y A.J. Rivera, se analiza la experiencia del uso de dinámicas y mecanismos de juego para incentivar el proceso de autoentrenamiento y autoevaluación del estudiante en la preparación de una prueba objetiva para la asignatura de Arquitectura de Computadores de los estudios de Ingeniería en Informática de la Universidad de Jaén. En otra línea, el artículo "Utilización de la metodología de aula invertida en una asignatura de Fundamentos de Informática" de B. del Pino, B. Prieto, A. Prieto y F. Illeras, analiza la aplicación de la metodología de aula invertida, basada en los recursos propios de un MOOC (*Massive Open Online Course*) y de una plataforma tecnológica, a la docencia de la asignatura Fundamentos de Informática de la Universidad de Granada. Se proporcionan además los resultados obtenidos de la encuesta de opinión realizada por los estudiantes y se pone de manifiesto la mejora de las calificaciones académicas con respecto a cursos anteriores.

La mecatrónica constituye un ámbito de interés creciente dentro de la Ingeniería de Computadores. Así, en "Desarrollo de sistemas mecatrónicos de bajo coste para equipamiento experimental de prácticas docentes" de G. Olivares, A. González, F. Gómez, M. Damas y A. Olivares, se describe un laboratorio realizado íntegramente por profesores y alumnos con material docente experimental constituido por maquetas y sistemas mecatrónicos de muy bajo coste. Este laboratorio forma parte del proyecto de innovación docente "Nuevas experiencias de control óptimo con sistemas mecatrónicos" y da soporte a asignaturas relacionadas con el aprendizaje de técnicas de control impartidas por el área de Ingeniería de Sistemas y Automática de la Universidad de Granada. El artículo "Aprendiendo mecatrónica mediante el diseño y construcción de una plataforma de lanzamiento de prototipos de cohetes de agua" de C. Márquez, R. Agís, A. Cañas y M. Damas, describe un trabajo de fin de máster del Máster en Ciencia de Datos e Ingeniería de Computadores de la Universidad de Granada consistente en la implementación de una plataforma de lanzamiento de prototipos de cohetes de agua, que incluye un diseño mecatrónico para controlar remotamente el sistema desde un PC. Se trata de un ejemplo práctico de algunos de los conceptos y habilidades relacionadas con la mecatrónica que se abordan en el mencionado máster. Dentro de esta línea relacionada con la docencia en sistemas de control, el artículo "Sistema Domótico Distribuido para Controlar el Riego y el Aire Acondicionado en el Hogar" de I. M. Laclaustra, J. M. Alonso, A. A. del Barrio y G. Botella, presenta el proyecto *SEDomotics* realizado dentro de la asignatura Sistemas Empotrados Distribuidos del Máster en Ingeniería Informática de la Universidad Complutense de Madrid. El proyecto implementa una plataforma de control domótico mediante varias placas

Arduino y una placa Raspberry Pi como servidor. En esta misma línea el artículo "Diseño de la Maqueta Domótica para el Aprendizaje de Sistemas de Automatización Domótica" de J.A. Holgado, presenta la experiencia que ha adquirido el grupo de investigación de Sistemas Concurrentes de la Universidad de Granada en la construcción de maquetas domóticas como herramientas docentes sobre las que realizar prácticas a diferentes niveles de abstracción en asignaturas de Ingeniería Informática e Ingeniería de Telecomunicación.

También en este número se incluyen varios artículos dentro de la sección "Desde el pupitre", que iniciamos en el número de 2014 con artículos elaborados por estudiantes (con o sin participación de profesores). Dado que dentro de los objetivos de esta revista se encuentra la difusión de la Ingeniería de Computadores entre los estudiantes universitarios, es importante disponer de artículos que reflejen de alguna manera sus inquietudes en las asignaturas de este ámbito. Junto con el artículo relacionado con la mecatrónica al que nos hemos referido más arriba, este número incluye el trabajo "Theremin DIY con Arduino UNO" de S. Moreno, M. Damas, H. Pomares y O. Baños, que describe un trabajo, propuesto por un estudiante para una asignatura, que ilustra el concepto de computación física, muy relevante en los estudios de Grado en Ingeniería Electrónica Industrial de la Universidad de Granada. También se pone de manifiesto el interés de los estudiantes en la plataforma Arduino.

Como en los cinco 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 sexta edición, se celebraron en Diciembre de 2015, centradas fundamentalmente en los nuevos Proyectos de Innovación Docente financiados que se desarrollarán en el Departamento de Arquitectura y Tecnología de Computadores durante los próximos años, y en el análisis de los resultados y la coordinación de las asignaturas de los Grados en los que imparte docencia el Departamento. Nuestro agradecimiento a todos los que han contribuido a la celebración de esas VI JDCE. En primer lugar, agradecemos al equipo de dirección de la E.T.S.I.I.T. de Granada habernos facilitado el uso de las instalaciones, y a los ponentes de las distintas sesiones sus presentaciones y su participación en las mesas redondas correspondientes. También agradecemos al Departamento de Arquitectura y Tecnología de Computadores y al Vicerrectorado de Garantía de la Calidad de la Universidad de Granada el apoyo a estas Jornadas.

El Comité Editorial

Reflexiones sobre el EEES tras los primeros cursos de la asignatura Arquitectura de Computadores

Julio Ortega, Mancia Anguita, Jesús González, Miguel Damas
Departamento de Arquitectura y Tecnología de Computadores
E.T.S.I.I.T., Universidad de Granada
{jortega, manguita, jesusgonzalez, mdamas}@ugr.es

Resumen. Han pasado ya algunos cursos académicos desde que empezaron a impartirse en la Universidad de Granada estudios de grado según las directrices del Espacio Europeo de Educación Superior (EEES). En el caso del grado en Ingeniería Informática, este curso 2015/2016 terminará la tercera promoción. En este artículo, tras resumir los objetivos de la denominada reforma de Bolonia, se describen algunas de las situaciones en las que se ha traducido la implementación de la reforma en nuestro país. Para extraer algunas de las conclusiones del artículo se ha tomado como referencia la asignatura de Arquitectura de Computadores de segundo del grado en Ingeniería Informática que está siendo impartida por varios de los autores del artículo.

Palabras clave: Arquitectura de Computadores, Especio Europeo de Educación Superior (EEES), Evaluación del aprendizaje, Mapas conceptuales.

Abstract. It has been several academic years since the beginning of study programmes at the University of Granada according to the guidelines of the European Higher Education Area. In this paper, after summarizing the main objectives of the so-called Bologna reform, some of the consequences of its implementation in Spain are described. Moreover, the subject of Computer Architecture, taught in the second course by some of the authors of the paper, is considered as a reference to extract some of the conclusions of the paper.

Keywords: Computer architecture, Conceptual maps, European Higher, Eduaction Area, Learning assessment.

1 Introducción

En este curso 2015/2016 se graduará la tercera promoción del Grado en Ingeniería Informática y la primera del Máster en Ingeniería Informática de la Universidad de Granada, ambas impartidas en la Escuela Técnica Superior de Ingenierías Informática y Telecomunicaciones (ETSIIT). Cabría preguntarse si estos nuevos planes de estudio en Ingeniería Informática, regulados bajo las directrices del Espacio Europeo de Educación Superior (E.E.E.S.), han contribuido a mejorar las calidad de la docencia

impartida y, como resultado, la formación de los titulados en Informática. Sin duda, proporcionar una respuesta científicamente concluyente a esta cuestión necesita un análisis amplio tanto de los resultados de la evaluación de los estudiantes, como del éxito con que las competencias adquiridas están dando respuesta a las demandas sociales propias de cada grado. Si bien un estudio de este calibre exige bastante más información de la que se dispone actualmente, es posible analizar ya algunos de los datos obtenidos a partir de la docencia en las asignaturas que se han impartido ya durante algunos cursos, y poner de manifiesto alguno de los efectos de las nuevas estrategias docentes que se están poniendo en práctica. En esa línea, este trabajo utiliza algunos de los resultados disponibles de la asignatura Arquitectura de Computadores del Grado en Informática, para extraer, tras los cuatro años en los que se ha impartido, algunas conclusiones respecto a la implantación de los principios del EEES y sus resultados en el aprendizaje de los contenidos de la asignatura.

Así, en la Sección 2 se revisarán los principios y los objetivos que se pretendían alcanzar con la implantación del EEES, y se confrontan con algunas de las conclusiones respecto al aprendizaje que se están extrayendo de estudios recientes en neurociencia. A continuación, la Sección 3 describe los contenidos y la metodología con la que se imparte la asignatura obligatoria Arquitectura de Computadores del segundo cuatrimestre del segundo curso del grado en Ingeniería Informática, para analizar en la Sección 4 los resultados de evaluación de los cursos en los que se viene impartiendo, en algunos de los grupos de la asignatura. Finalmente, la Sección 5 proporciona las conclusiones del trabajo y la Sección 6 contiene las referencias bibliográficas.

2 Qué es y qué ha supuesto la implantación del EEES

En [1] se hace una presentación bastante acertada de los objetivos del EEES, conocido usualmente como la reforma o proceso de Bolonia. En ese trabajo se hace hincapié fundamentalmente en las consecuencias sobre la metodología docente del nuevo marco que, en la UGR, se empezó a poner en práctica en el curso 2010/2011, y más concretamente sobre la enseñanza/aprendizaje de las asignaturas de Arquitectura y Tecnología de Computadores.

En realidad, el EEES se plantea como un proceso, iniciado en 1999, para *armonizar* los distintos sistemas universitarios de la Unión Europea, de forma que se facilitase la movilidad de los estudiantes en el marco de la globalización socio-económica desde el que se contemplaba el siglo XXI, antes de la crisis de 2008. Las reformas que se requerían en muchos países para ajustar sus sistemas a lo establecido en la Declaración de Bolonia se contempló como una oportunidad para implementar cambios en las metodologías docentes que han afectado no solo a la estructura curricular y a la movilidad de los estudiantes, sino también al proceso de enseñanza/aprendizaje y a organización del personal docente, entre otros factores. El establecimiento del sistema de créditos ECTS (*European Credit Transfer System*) en el que se tiene en cuenta la carga de trabajo del estudiante fuera de las clases

presenciales ha supuesto un cambio importante a la hora de organizar los contenidos desde una especie de contrato entre profesor y estudiante, que contribuye a la transparencia de un sistema centrado en el aprendizaje del estudiante. Así, en nuestro país, el EEES se ha contemplado no solo como un cambio en los planes de estudios sino también como una oportunidad para mejorar la calidad de la Universidad española.

En la práctica la implantación EEES ha tenido unos efectos muy concretos desde el punto de vista de la organización de las asignaturas. Así, se deben especificar las competencias que debe alcanzar el estudiante a través de la asignatura en cuestión, y elaborar guías docentes como instrumento para la transparencia, y como marco de referencia para profesores y estudiantes del proceso de enseñanza-aprendizaje. Además, el interés por las metodologías docentes ha generado un número considerable de publicaciones docentes y la financiación de proyectos de innovación docente para concretar el enfoque de aprendizaje basado en el estudiante.

Otros cambios relacionados de una u otra forma con la reforma de Bolonia incluyen la puesta en marcha de Agencias de Calidad, Vicerrectorados de Calidad, y Comisiones de Calidad de los Títulos. Uno de los resultados de la búsqueda de transparencia en los procesos Universitarios ha sido la elaboración de memorias de Verificación de los Títulos. Éstas recogen no solamente las asignaturas sino también datos de infraestructuras, profesorado que los imparte, y previsiones en cuanto a los indicadores de calidad y rendimiento relacionadas con las tasas de titulados que terminan respecto a los matriculados, etc. Así, para que pueda impartirse un título universitario se debe obtener la correspondiente acreditación sometiendo su memoria de verificación a la evaluación de la agencia nacional o de la comunidad autónoma correspondiente y, una vez aprobado y tras impartirse durante unos cursos debe nuevamente volverse a evaluar para obtener la renovación de la acreditación. En estos últimos años, también se ha dedicado una atención considerable a los procesos de evaluación del profesorado. Además de la implantación de procesos de acreditación para poder acceder a las distintas figuras docentes, las Universidades han puesto en marcha programas de evaluación de la calidad docente que incluyen encuestas de satisfacción de los estudiantes. Una de las consecuencias de las nuevas iniciativas que se han implantado es el incremento del trabajo de gestión asociado a la necesidad de generar y evaluar los documentos necesarios que aseguren la transparencia en la enseñanza Universitaria.

Como se ha dicho antes, la reforma de Bolonia también aborda cuestiones relacionadas con las metodologías docentes. De hecho, se promueve que el profesor aplique la metodología más adecuada para los contenidos de la asignatura teniendo en cuenta la relevancia que se da ahora al papel del estudiante como protagonista de su propio aprendizaje. En general, se favorece el aprendizaje basado en problemas a través del que el propio estudiante construye sus mapas de relaciones entre los conceptos de la asignatura, a partir de los elementos fundamentales presentados por el profesor, que se contempla ahora como orientador y evaluador del proceso de aprendizaje. Se pretende reducir el protagonismo de la lección magistral, que se suele identificar con el aprendizaje memorístico por parte del estudiante, pero que tiene

efectos bastante positivos si se introducen elementos que promuevan su participación activa en el desarrollo de las clases y la resolución de ejercicios y problemas con los que se va ilustrando la utilidad de los conceptos explicados. Desde nuestro punto de vista, los problemas de las clases magistrales provienen de la masificación de los grupos y de que no se ponga de manifiesto, a través de esas clases, la relación entre los conocimientos teóricos y los problemas que pueden abordarse gracias a ellos.

Uno de los resultados de las discusiones sobre dichas metodologías ha sido la instauración generalizada de la evaluación continua. La necesidad de realizar actividades de evaluación continua surge al cuestionar que mediante un examen final sea posible medir los resultados de aprendizaje que se establecen en las guías, y aun en caso de que puedan evaluarse, por la necesidad de plantear actividades que preparen al estudiante para alcanzar dichos resultados. Si se busca que los estudiantes sean capaces de resolver determinado tipo de problemas, es conveniente que se les planteen a través de relaciones de problemas y se discuta en clase la forma de abordarlos. En la mayoría de las asignaturas de las carreras científicas o técnicas ya se acostumbraba a acompañar cada tema con una relación de problemas, y se dedicaban clases a resolverlos. Además, normalmente ya se organizaba la docencia entre clases de teoría y prácticas, y en estas últimas los estudiantes se enfrentaban a problemas experimentales cuya mayor o menor aplicabilidad a casos reales estaba más bien limitada por la falta de recursos que por razones metodológicas.

Otra razón importante para que se realicen pruebas de evaluación a lo largo del curso es que los estudiantes reciban realimentación acerca de si lo que han realizado es correcto o incorrecto: la denominada evaluación formativa. En cualquier caso, la evaluación continua no significa que haya que evaluar a los estudiantes a través de pruebas parciales. De lo que se habla, más bien, es de la incorporación en la nota final de las calificaciones de las pruebas realizadas a lo largo del curso, siempre y cuando el estudiante alcance un mínimo en las pruebas presenciales, que se aconseja que sea una calificación próxima a la necesaria para aprobar la asignatura [2].

Sin embargo, la coincidencia de la impartición de un número considerable de asignaturas que implementan evaluación continua junto con la disponibilidad de apuntes, transparencias y demás material pueden afectar negativamente al aprendizaje, como veremos a continuación.

2.1 Una perspectiva del EEES desde la neurociencia

El aprovechamiento de las tecnologías de la información y las comunicaciones ha sido un elemento esencial para mejorar la calidad de la enseñanza universitaria. Plataformas docentes de fácil acceso y uso como SWAD [3] han contribuido a difundir y acceder a material docente, mejorar la comunicación profesor-estudiante, y facilitar la gestión de los grupos, entre otras facetas de la actividad docente. Por otra parte Internet y el acceso a revistas electrónicas posibilitan la actualización de material y disponer de herramientas de acceso abierto. Actualmente, es usual que los estudiantes dispongan de las transparencias de clase y soluciones a problemas propuestos. Han quedado atrás los años en que era fundamental asistir a clase para

tomar apuntes que recogiesen los contenidos de la asignatura y las explicaciones del profesor. Si se faltaba a alguna clase había que recurrir a las notas prestadas por algún compañero o a extraer la información de la bibliografía recomendada. Como se dispone de las transparencias y de gran cantidad de información muchos estudiantes no se preocupan de tomar apuntes y consideran que, en su momento, podrán asimilarla. Como si se tratase de copiar un fichero desde un directorio (en el disco) a otro (en el cerebro).

Sin embargo la neurociencia nos dice que nuestra memoria no funciona como el disco duro de nuestros computadores. De hecho, los modelos más aceptados actualmente indican que el aprendizaje se construye, y la memorización es el primer paso de un proceso de síntesis personal de lo que se ha leído, escuchado, visto, etc., que implica creatividad y juicio [4]. Los eventos que se introducen en la memoria a corto plazo se transforman en pensamientos memorizados y pasan a la memoria a largo plazo una vez transcurrido un cierto tiempo [5]. Estos dos tipos de memoria se basan en procesos biológicos diferentes y se ha puesto de manifiesto que el paso a la memoria a largo plazo no sólo cambia la concentración de neurotransmisores para reforzar ciertas conexiones, sino que también aparecen nuevas sinapsis, implicando cambios anatómicos además de bioquímicos. Por tanto, debe activarse un gen para producir dichas proteínas y responder al proceso de aprendizaje. Además, la consolidación de la memoria a largo plazo implica una cadena de interacciones relativamente larga entre el hipocampo y el córtex cerebral. La atención está en la base de la consolidación de la memoria al ocasionar que las neuronas del córtex envíen señales y se abra un camino para que la dopamina llegue al hipocampo y permita la consolidación de la memoria explícita al activar los genes que estimulan la síntesis de nuevas proteínas.

Se debería promover el estudio interrelacionado de los contenidos de las asignaturas junto con los de otras asignaturas, y plantear preguntas, prácticas y problemas que muestren la relación entre los distintos conceptos y faciliten indexar la información que tenemos en la memoria. En esta línea, sería conveniente fomentar el aprendizaje interrelacionado de distintas asignaturas. Sin embargo, la actual implementación de los Planes de Estudio de los grados se basa en asignaturas impartidas por profesores encuadrados en Departamentos. Cada Departamento es el responsable de organizar la docencia de las asignaturas de un mismo ámbito de conocimiento en los distintos estudios. Si bien los Planes de estudio ordenan en el tiempo la impartición de las distintas asignaturas en función de las dependencias entre sus contenidos, usualmente no se pone de manifiesto la interrelación que existe entre contenidos de asignaturas que se imparten simultáneamente. Es más, el número de exámenes y de actividades a realizar puede ser tan elevado, que es difícil que el estudiante profundice en los conceptos y pueda asentar lo aprendido, y se produce una fuerte competencia entre el tiempo que los estudiantes dedican a cada asignatura. Si en una asignatura se realizan exámenes parciales que permiten a los estudiantes que los superan no tener que volverse a evaluar de los contenidos del examen parcial en una prueba que abarca toda la asignatura, las demás asignaturas deberían tener en cuenta esta circunstancia dado que, como se viene observando, los estudiantes suelen concentrarse en aprobar los exámenes parciales a los que se les convoca, y les resulta más difícil mantener el

ritmo de las demás asignaturas. Si además, pueden disponer del material de la asignatura a través de una plataforma docente, la reacción usual será la de pensar que pueden preparar los exámenes, sin tener que seguir muy exhaustivamente las explicaciones de clase.

3 Arquitectura de Computadores (AC) en la UGR tras el EEES

La asignatura de Arquitectura de Computadores se imparte en el segundo curso del grado en Ingeniería Informática de la Universidad de Granada. Se trata de una asignatura obligatoria cuyos objetivos y contenidos se han descrito en [6,7]. La guía docente de la asignatura se puede encontrar en la dirección [8].

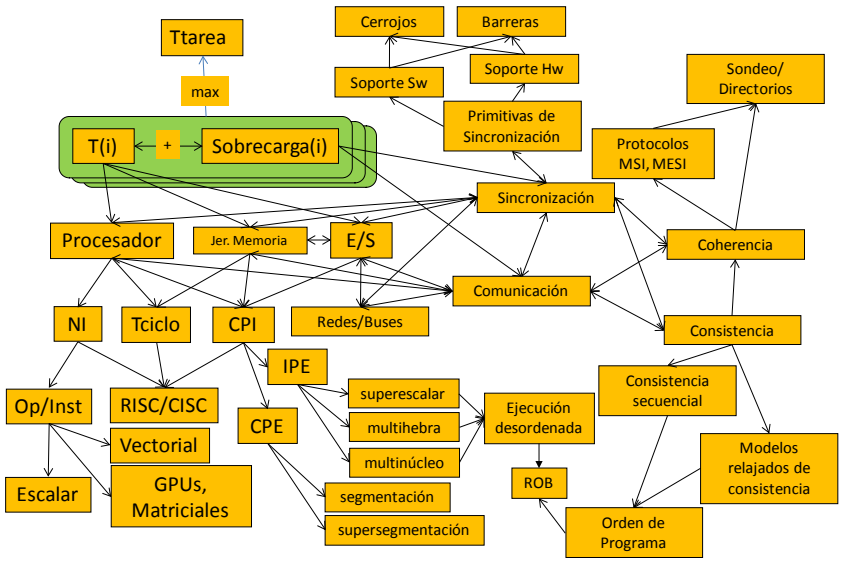


Figura 1. Un mapa conceptual para la asignatura Arquitectura de Computadores del Grado en Ingeniería Informática de la UGR [9]

Los objetivos de la asignatura proponen el estudio de la arquitectura del computador desde el punto de vista del aprovechamiento del paralelismo que implementan los computadores actuales, más que desde el punto de vista del diseño. A la vez que se introducen los conceptos fundamentales de las arquitecturas paralelas que necesita un ingeniero de computadores, esta perspectiva muestra la importancia de conocer las características de la arquitectura del computador para desarrollar aplicaciones eficientes, y justifica una asignatura obligatoria de este tipo en el segundo curso del grado en Ingeniería Informática. La Figura 1 muestra un marco conceptual de la asignatura, cuyos contenidos se estructuran en cinco temas dedicados, respectivamente a la evaluación de las prestaciones, el procesamiento paralelo, las arquitecturas con paralelismo a nivel de hebra, las arquitecturas con paralelismo entre

instrucciones, y una introducción a las arquitecturas actuales con paralelismo de datos.

La asignatura se imparte a través de grupos de teoría, cada uno de los cuales tiene asignados varios grupos de prácticas de alrededor de 25 estudiantes. La evaluación de la asignatura incluye una calificación de hasta 6 puntos de la parte de teoría y otra de hasta 4 puntos de la parte de prácticas. Para sumar las dos calificaciones es necesario que superen un 40% de la nota máxima de cada apartado (2.4 en la parte de teoría y problemas, y 1.6 en la parte de prácticas) y para aprobar la asignatura es necesario obtener una calificación total de 5.0 o más puntos. La evaluación de la parte de teoría se realiza a través de una prueba escrita de todos los contenidos de la asignatura, a la que se asigna una calificación máxima de 4 puntos, y de varias pruebas parciales (cuatro pruebas de la teoría de los diferentes temas y una prueba parcial de problemas) en las que se puede alcanzar una calificación máxima de 2.0 puntos. Las prácticas se evalúan mediante una prueba escrita de todos los contenidos, que tiene una calificación máxima asignada de 2.0 puntos, y entregas y evaluaciones de las prácticas en el aula, con una calificación máxima de 2 puntos.

Tabla 1. Descripción de los grupos analizados

Código	Grupo	Estudios (Curso Académico)	Estudiantes
12A	A	Ing. Informática (2011-12)	53
12B	B		31
12C	C		24
13A	A	Ing. Informática (2012-13)	73
13B	B		65
13IM		Ing. Informática y Matemáticas (2012-13)	23
14A	A	Ing. Informática (2013-14)	73
14B	B		72
14IM		Ing. Informática y Matemáticas (2013-14)	25
15A	A	Ing. Informática (2014-15)	88
15B	B		99
15IM		Ing. Informática y Matemáticas (2014-15)	30

4 Resultados experimentales de la docencia en AC

En esta sección se proporcionan algunos resultados de los rendimientos de los estudiantes en las distintas pruebas de teoría de la asignatura. Corresponden a los cuatro cursos académicos en los que la asignatura se ha impartido (2011/2012 a 2014/2015) y a varios grupos de teoría impartidos por uno de los autores del artículo. En la Tabla 1 se proporciona el número de estudiantes de los grupos considerados, los estudios a los que pertenecen y el curso académico al que corresponden. La evaluación en los cursos 2013-2014 y 2014-2015 se han realizado según el procedimiento de evaluación continua detallado en la guía de la asignatura [8] y descrito en la Sección 3. La Figura 2 muestra las medias de las calificaciones de teoría (examen en los cursos 2011-12 y 2012-13 y examen más pruebas de clase en los

cursos 2013-14 y 2014-15) de distintos grupos. Como se puede observar, existen cambios en los valores medios, pero no se observa una tendencia uniforme a partir de que se implanta la evaluación continua. A continuación se comprueba si las diferencias son estadísticamente significativas.

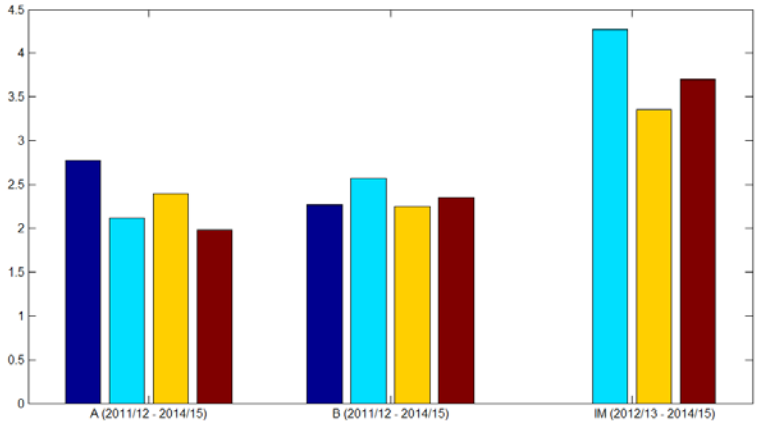


Figura 2. Medias de las calificaciones de teoría (entre 0 y 6) en los distintos cursos académicos para los distintos grupos: A y B del grado en Ingeniería Informática y grupo del grado en Ingeniería Informática y Matemáticas (IM)

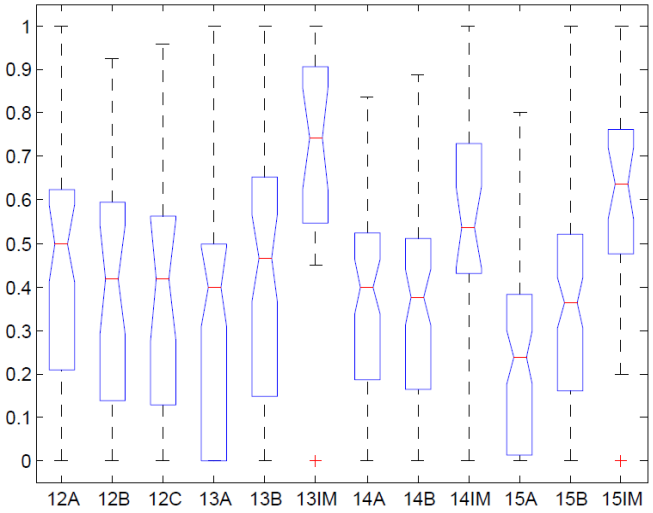


Figura 3. Medias y desviaciones de las calificaciones de teoría para cada uno de los cursos (respecto a la calificación máxima del examen)

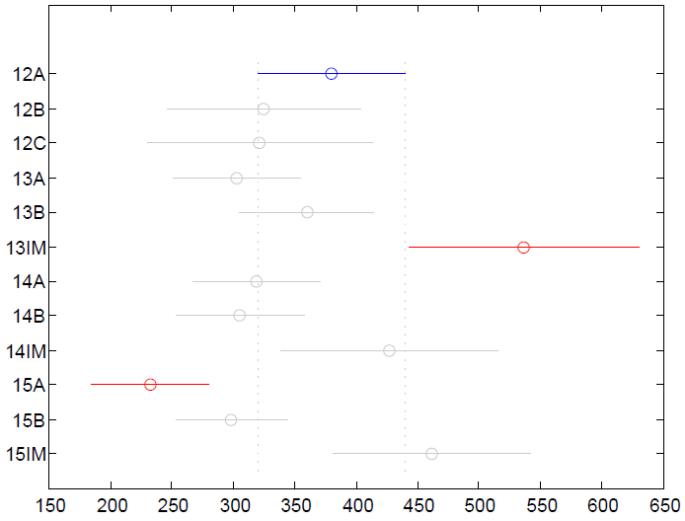


Figura 4. Análisis estadístico de la significación de las diferencias en la Figura 2 mediante el test de Kruskal-Wallis

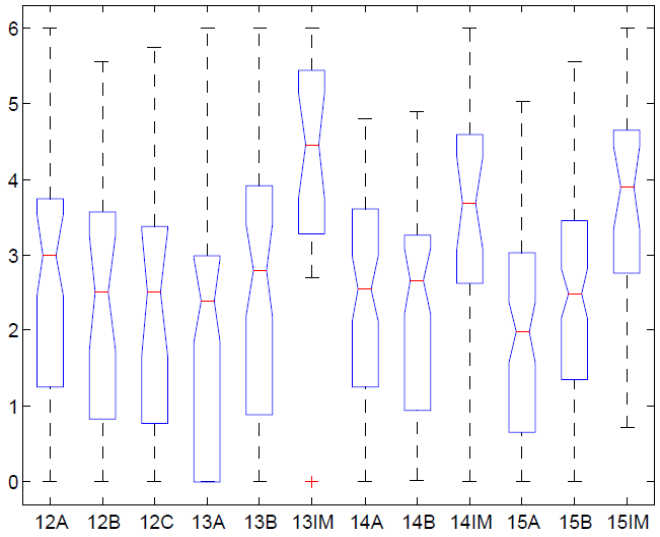


Figura 5. Medias y desviaciones de las calificaciones totales de teoría para cada uno de los cursos

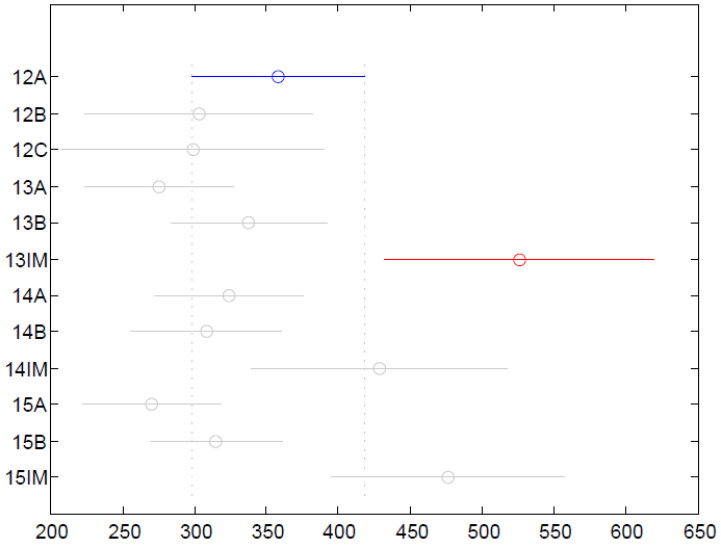


Figura 6. Análisis estadístico de la significación de las diferencias en la Figura 4 mediante el test de Kruskal-Wallis

Los resultados de las Figuras 3 y 4 comparan las fracciones de las notas máximas del examen de teoría y problemas que abarca todos los contenidos de la asignatura. En los cursos académicos 2011-2012 y 2012-2013 este examen tenía asignada una calificación máxima de seis puntos, mientras que desde la implantación de la evaluación continua la nota máxima era de 4 puntos. En la Figura 3 se muestran las medias y las desviaciones a través de los distintos grupos y años. Si se comparan por un lado los grupos del grado en ingeniería informática, y por otro los grupos del grado en ingeniería informática y matemáticas, se puede apreciar una ligera disminución en los valores medios de las calificaciones del examen de toda la asignatura en los cursos en los que se ha utilizado la evaluación continua. Se podría pensar en algún tipo de despreocupación por el estudio del examen que abarca toda la asignatura por tener algunas notas de clase de partida. Sin embargo, el análisis estadístico que se ha realizado aplicando un test de Kolmogorov-Smirnov, nos ha permitido comprobar que los datos no siguen una distribución normal en ninguno de los grupos y cursos, y el posterior test de Kruskal-Wallis nos indica (Figura 4) que las diferencias solo son estadísticamente significativas para algunos grupos en algún curso académico y no están asociadas al cambio a un sistema de evaluación continua en el curso 2013-2014.

En las Figuras 5 y 6 se comparan los resultados del examen de teoría de seis puntos en los cursos 2011-2012 y 2012-2013 con la calificación de teoría total obtenida al sumar el examen de cuatro puntos y los puntos (hasta un máximo de dos) correspondientes a las pruebas de evaluación continua. En este caso la situación sigue siendo muy similar a la que se refleja cuando se comparan los porcentajes de calificaciones de los exámenes. Existen diferencias entre los valores medios obtenidos por los distintos grupos pero no son diferencias estadísticamente significativas que se puedan asociar al cambio en el sistema de evaluación. Las diferencias significativas se producen entre

el grupo del grado en ingeniería informática y matemáticas del curso académico 2012-2013 y los demás. En los grupos de este grado suelen observarse calificaciones medias más elevadas que en los grupos del grado en ingeniería en informática, aunque también hay que tener en cuenta que existen diferencias apreciables entre las notas medias de acceso a cada una de estas titulaciones.

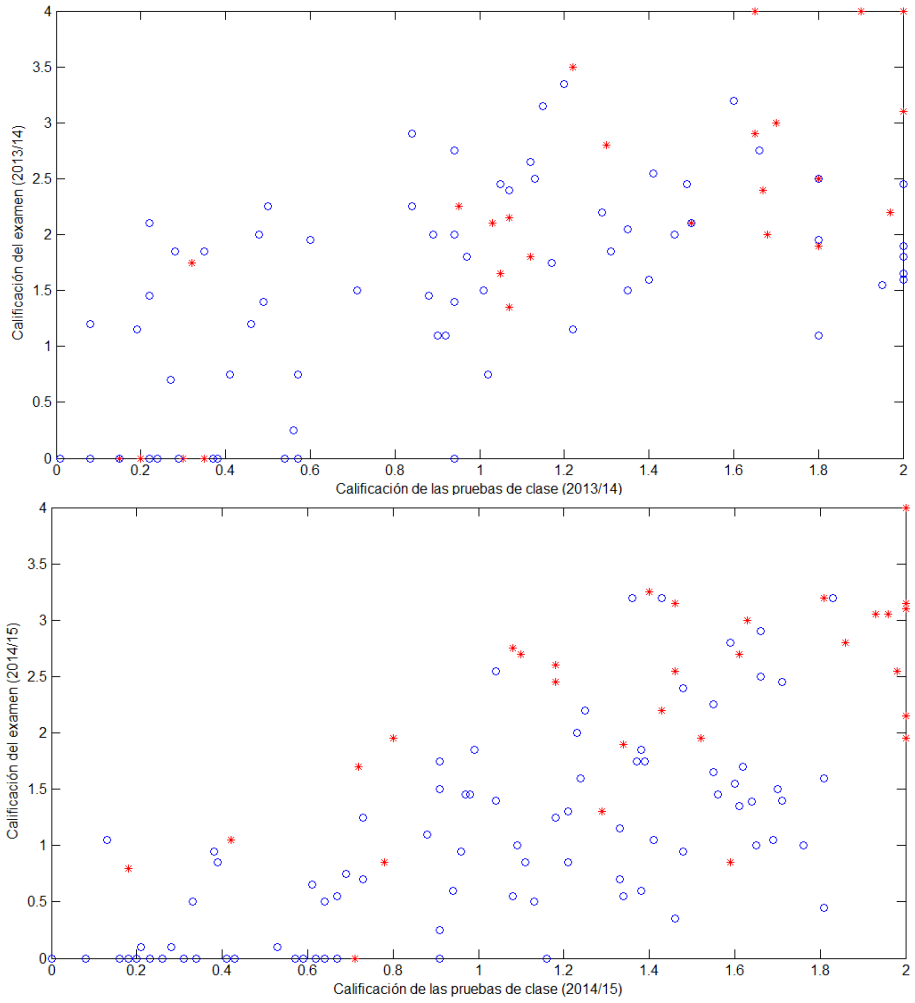


Figura 7. Relación entre calificaciones de clase y del examen de los estudiantes para el grupo A del grado en Ingeniería Informática ('o') y del grado en Ingeniería Informática y Matemáticas ('*')

En la Figura 7, cada punto corresponde a las calificaciones de las pruebas de clase y la del examen de teoría de un estudiante. Se puede ver que en general, calificaciones más elevadas en las pruebas de clase suelen coincidir con calificaciones mayores en los exámenes de toda la materia. Además, se ha comprobado que existe una

correlación significativa entre dichas calificaciones en todos los grupos y para los dos cursos académicos en los que se ha aplicado la metodología de evaluación continua: coeficientes correlación de 0.696, 0.693, y 0.669 en el curso 2013/2014 y 0.586, 0.528, y 0.808 en el curso 2014/2015, respectivamente para los grupos A y B del grado en Informática y el grupo del grado en Ingeniería Informática y Matemáticas).

5 Conclusiones y propuestas

Los datos de las calificaciones obtenidas por los estudiantes en la asignatura Arquitectura de Computadores desde que empezó a impartirse en el curso 2011/2012 ponen de manifiesto que no existen diferencias significativas entre las calificaciones de teoría obtenidas tras la aplicación de la evaluación continua en el curso 2013/2014. Tampoco parecen afectar a las calificaciones que los estudiantes obtienen en el examen de teoría y, además, la correlación observada entre las calificaciones de las pruebas de clase y las calificaciones del examen aconsejan la realización de estas pruebas que resultan útiles desde el punto de vista de la evaluación formativa del estudiante.

Además, tras las conclusiones acerca de las características del proceso de aprendizaje a las que se ha llegado tras los últimos avances de la neurociencia, se puede concluir que, es necesario el trabajo continuado sobre los conceptos que aborda una asignatura, pero también es importante que existan pruebas globales en las que se evalúen todos los contenidos, dada la necesidad de promover que los estudiantes creen mapas conceptuales que afiancen la asimilación de los conceptos de la asignatura y su ubicación en la memoria. Puede ser conveniente que los resultados de las pruebas que se realizan a los estudiantes se tengan en cuenta en la calificación final, pero debe prestarse la suficiente atención a las perturbaciones que se pueden originar en la marcha de otras asignaturas si la parte de la calificación que se asigna a esas pruebas es demasiado elevada.

En general, se puede afirmar que salvo la mayor transparencia asociada a la elaboración de las guías de las asignaturas y los procesos de verificación de los estudios, el EEES no ha supuesto cambios considerables en cuanto a la forma que se impartían las asignaturas en los estudios de ciencias e ingenierías donde ya se dedicaba un tiempo considerable a las clases prácticas y a la realización de problemas. La esperada mejora, relacionada con una reducción del tamaño de los grupos que permitiría una atención más personalizada del estudiante, no ha llegado. Quizá la coincidencia de la puesta en marcha de la reforma con las consecuencias de la crisis económica de 2008 ha tenido bastante que ver.

6 Referencias

1. Diaz, J; Morillas, C.; Romero, S.; Guillén, A.:"Bolonia *for dummies*". Enseñanza y Aprendizaje de Ingeniería de Computadores, No.1, pp.5-17, 2011.

2. UNED (Vicerrectorado de Espacio Europeo y Planificación Docente y Vicerrectorado de Calidad e Innovación Docente): “Evaluación Continua en el EEES” http://portal.uned.es/pls/portal/docs/PAGE/UNED_MAIN/LAUNIVERSIDAD/VICERRECTORADOS/CALIDAD_E_INTERNACIONALIZACION/INNOVACION_DOCENTE/IUED/DOCUMENTOS/EVALUACION_CONTINUA_EEES_UNED.PDF
3. SWAD (Sistema Web de Apoyo a la Docencia), swad.ugr.es, 2016.
4. Bain, K.:"Lo que hacen los mejores profesores universitarios". Publicacions Universitat de València, 2006.
5. Kandel, E.R.:"En busca de la memoria: una nueva ciencia de la mente". Katz Editores, 2007.
6. Anguita, M.; Cañas, A.; Fernández, F.J.; Ortega, J.; Rojas, I.:”El perfil de Ingeniería de Computadores y las asignaturas de Estructura y Arquitectura de Computadores en el grado de Ingeniería Informática”. Enseñanza y Aprendizaje de Ingeniería de Computadores, Nº.1, pp.89-98, 2011.
7. Ortega, J.; Anguita, M.:”Arquitectura de Computadores en seis créditos ECTS”. Enseñanza y Aprendizaje de Ingeniería de Computadores, No.2, pp.13-25, 2012.
8. Guía docente de Arquitectura de Computadores (grado en Ingeniería Informática de la UGR). http://grados.ugr.es/informatica/pages/infoacademica/guias_docentes/201516/segundo/2semestre/arquitecturadecomputadores/! 2016.
9. Ortega, J.; Anguita, M.:”Relatos, Mapas Conceptuales y Arquitectura de Computadores”. Enseñanza y Aprendizaje de Ingeniería de Computadores, Nº.4, pp.89-102, 2014.

Explotación de la potencia de procesamiento mediante paralelismo: un recorrido histórico hasta la GPGPU

Francisco Charte Ojeda[1], Antonio J. Rivera Rivas[2], Francisco J. Pulgar Rubio[2], María J. del Jesus Díaz[2]

[1] Departamento de Ciencias de la Computación e Inteligencia Artificial. E.T.S.I.I.T. Universidad de Granada.
francisco@fcharte.com
[1] Departamento de Informática. E.P.S. Universidad de Jaén.
{arivera,fjpulgar,mjjesus}@ujaen.es

Resumen. La mejora en los sistemas de fabricación de semiconductores, con escalas de integración crecientes durante décadas, ha contribuido a incrementar de forma espectacular la potencia de los sistemas de cómputo en sus diversas formas, ordenadores personales y portátiles, móviles, tabletas, consolas, etc. Esa evolución, no obstante, también ha encontrado obstáculos por el camino que, entre otros aspectos, acabaron hace varios años con la escalada en las frecuencias de reloj. En la actualidad la potencia de un procesador ya no se mide exclusivamente en GHz, sino que también influyen factores como el número de núcleos de procesamiento y el diseño de estos. En el presente artículo se lleva a cabo un recorrido histórico de cómo el paralelismo ha ido adecuándose al hardware disponible en cada momento con el objetivo de obtener el mayor provecho del mismo.

Palabras Clave: Paralelismo, tiempo compartido, núcleo, CPU, GPU.

Abstract. Due to the improvement of semiconductor manufacturing technologies, and higher integration scales in the last decades, the power of computing devices has experienced an impressive growth. However, some obstacles have been also found along the way. As a consequence, the battle for reaching higher clock frequencies almost ended a few years ago. Nowadays, the power of processors is not measured exclusively using GHz. Other factors, as the number of cores and their inner design, also have a large impact. This paper provides an historical review on how parallelism techniques have been adapted over time to overcome these changes aiming to better exploit the available hardware.

Keywords: Parallelism, time sharing, core, CPU, GPU

1 Introducción

El conocimiento que aporta al alumnado la asignatura de ingeniería de computadores es esencial para, entre otros fines, poder extraer del hardware existente en cada momento el mayor beneficio posible. Los detalles de arquitectura correspondientes a la máquina en la que se ejecutará un software, así como la identificación de los lenguajes y herramientas más apropiadas a cada caso, hacen posible el desarrollo de aplicaciones capaces de explotar toda la potencia disponible. Como sería de esperar, tanto el hardware como las soluciones diseñadas para sacar provecho del mismo evolucionan a lo largo del tiempo.

Durante décadas, las mejoras en la potencia de procesamiento de un sistema se debían, fundamentalmente, a un incremento en la frecuencia de funcionamiento de su procesador central o CPU, medida en términos de MHz/GHz. La incorporación de múltiples vías para la descodificación y ejecución de instrucciones, y en ocasiones varias unidades aritmético-lógicas, se sumaban a la citada mejora de velocidad en el reloj que actúa como marcapasos interno de la CPU. A estos siguieron el diseño de procesadores con capacidad para actuar sobre dos hilos de ejecución en paralelo y, posteriormente, los que incorporaban múltiples núcleos de ejecución independientes. La disponibilidad de la infraestructura necesaria para interconectar múltiples sistemas en una red de comunicación rápida, principalmente Ethernet, abrió en su momento una nueva vía para satisfacer la siempre creciente necesidad de procesamiento de datos, empleando múltiples máquinas como si de un único sistema de cómputo se tratase. El último escalón en este camino corresponde a las GPU (*Graphics Processing Unit*), particularmente a las tecnologías GPGPU (*General-Purpose Computation en GPU*) que permiten utilizar estos procesadores, inicialmente diseñados para acelerar el tratamiento de gráficos, como un sistema de cómputo de propósito general.

Paralelamente al propio hardware, las interfaces de programación de aplicaciones (API) ofrecidas por los sistemas operativos y los lenguajes de programación también han ido adaptándose con el objetivo de ofrecer al desarrollador los medios necesarios para explotar las nuevas capacidades. La elección de la herramienta adecuada a cada caso es un aspecto vital que complementa el conocimiento de la arquitectura del hardware.

El objeto de este artículo es ofrecer una visión global sobre cómo la explotación de paralelismo en informática ha ido evolucionando con el paso del tiempo, desde los primeros sistemas que mediante técnicas de tiempo compartido hicieron posible la multitarea hasta las actuales GPU y las diferentes API para aprovecharlas, como son CUDA, OpenCL y WebCL. En la Sección 2 se lleva a cabo un recorrido por la historia de la multitarea y el paralelismo en CPU. La Sección 3 introduce el concepto de GPGPU como evolución del uso de las GPU exclusivamente para tratamiento gráfico. Las tres API antes mencionadas, así como las herramientas asociadas a las mismas, se describen en la Sección 4. Las conclusiones finales se facilitan en la Sección 5, seguida de las correspondientes referencias.

2 Multitarea, paralelismo y computación distribuida

Actualmente resulta habitual contar con un teléfono móvil, una tableta, un ordenador personal y posiblemente también un portátil, dispositivos todos ellos que incorporan microprocesadores con una inmensa capacidad de cálculo, incluyendo la posibilidad de ejecutar múltiples tareas de manera paralela. El mundo, no obstante, no siempre ha sido así. Hace apenas medio siglo los sistemas informáticos ocupaban grandes espacios, requerían sistemas de refrigeración a medida y un gran aporte de energía eléctrica para, a cambio, obtener una fracción de la potencia que hoy ofrece un móvil. Aprovechar esa potencia, obtenida a un muy alto coste, resultaba fundamental, de ahí que algunas de las primeras técnicas de ejecución concurrente surgieran por entonces.

A la hora de planificar el grado de concurrencia de un software o algoritmo hay que diferenciar entre paralelismo SIMD (*Single Instruction Multiple Data*) y paralelismo MIMD (*Multiple Instruction Multiple Data*) [1]. Las GPU son ideales para el primer caso, en el que un mismo conjunto de sentencias se aplica simultáneamente sobre múltiples datos independientes entre sí, produciendo por lo general tantos resultados como datos de entrada existan. Es una configuración ideal para, por ejemplo, realizar operaciones sobre matrices que, dependiendo de su tamaño, pueden ser calculadas en un único ciclo de reloj. Para paralelizar tareas del segundo tipo, en las que flujos de sentencias diferentes se aplican sobre conjuntos de datos que pueden ser independientes o no, es necesario recurrir a los núcleos de la CPU, ya que cada uno de ellos puede ejecutar un programa distinto: un núcleo puede controlar la interfaz de usuario de un programa mientras otro se dedica a procesar datos introducidos en dicha interfaz, tareas paralelas pero que no tienen mucho que ver entre sí.

Los desarrollos descritos en los siguientes puntos de esta sección, relativos a la evolución de las CPU, están en su mayor parte relacionados con el modelo MIMD. En la Sección 3 y Sección 4, centradas en la programación GPGPU, el modelo de programación sería el SIMD.

2.1. Sistemas de tiempo compartido y multitarea

La velocidad a la que un operador puede escribir, a fin de comunicarse con el ordenador a través de una terminal interactiva, es muy inferior a la capacidad de un procesador para ejecutar sus instrucciones. A causa de ello la mayor parte del tiempo el sistema estaría en espera, una situación a la que en la actualidad, con nuestros ordenadores personales, no se le otorga la mayor importancia, pero que resultaba totalmente prohibitivo a mediados de la década de los 60 del pasado siglo. Por entonces los mainframes, que precedieron a la informática personal, suponían una inversión de millones de dólares y el coste de su funcionamiento y mantenimiento era también muy considerable. Es por entonces cuando surge el concepto de tiempo compartido[1] (*time-sharing*) [2] como solución para aprovechar la potencia de aquellas máquinas, ofreciendo a cada uno de los usuarios conectados a las mismas la ilusión de tener a su disposición todos los recursos.

[1] El término *time-sharing* con la acepción empleada aquí fue popularizado por John McCarthy, por entonces profesor del MIT y al que se otorgó el Premio Turing en 1971.

El tiempo compartido en un sistema informático funciona como un maestro de ajedrez jugando una partida múltiple, moviéndose en círculo y cambiando de tablero a tablero retomando la tarea donde la dejó en un momento anterior. A pesar de que el procesador en ningún caso ejecuta paralelamente varias tareas, para los usuarios así lo parece. Equipos de finales de los 60, como el GE-635/645 [3a], facilitaban la implementación de esta técnica, consiguiendo desbancar a sistemas mucho más establecidos, como el IBM System/360 [3b], en organizaciones como DARPA (*Defense Advanced Research Projects Agency*) y Bell Laboratories. En cuanto a software se refiere, Multics, el predecesor de UNIX, fue probablemente el primer sistema operativo diseñado para soportar originalmente servicios de tiempo compartido.

La técnica de tiempo compartido no solo facilitaba el acceso simultáneo de múltiples usuarios a los grandes sistemas, también fue el principio que permitió en sistemas monousuario ejecutar múltiples aplicaciones aparentemente de forma paralela. El microprocesador de los primeros ordenadores personales de IBM, el Intel 8086/8088 [4], carecía de la circuitería necesaria para facilitar el intercambio entre tareas y las unidades para ejecución paralela, pero el uso de las interrupciones hardware, generadas por el reloj interno, el teclado y ratón, hacían posible la ejecución aparentemente simultánea de varias aplicaciones. La infraestructura necesaria para efectuar el intercambio de contexto entre tareas por hardware se incorpora a esta familia de procesadores en 1986, con el procesador de 32 bits 80386 [4].

Debido a la inexistencia de estándares establecidos, construir soluciones multitarea en los 60s/70s implicaba conocer los detalles del hardware concreto en que iban a ejecutarse, así como el sistema operativo que el fabricante había desarrollado para dicho hardware. Por ejemplo, OS/360 ofrecía en 1967 la posibilidad de usar MVT (*Multiprogramming with Variable number of Tasks*) [5] como gestor de tareas con una filosofía análoga a la de los hilos de ejecución actuales.

2.2. Sistemas multiprocesador y multinúcleo

Durante décadas las CPU de los ordenadores se ajustaron fielmente al diseño establecido en la conocida como *arquitectura von Neummann* [6]. El núcleo de esta es el conocido ciclo de recuperación-decodificación-ejecución de instrucciones que, aún hoy, sigue encontrándose en el interior de los modernos microprocesadores. Se trata de una arquitectura inherentemente secuencial, cuya potencia estaba en gran medida limitada por la velocidad a que podían recuperarse las instrucciones del soporte en que se encontraban almacenadas. No obstante, dicha arquitectura ha ido adaptándose paulatinamente para aprovechar las innovaciones surgidas en cada momento, como fue el uso de memoria de tipo semiconductor, mucho más rápida que las cintas de papel y fichas perforadas, para contener las instrucciones del programa en ejecución. En 1970, como parte del proyecto de investigación ILLIAC de la Universidad de Illinois, se desarrolló el Illiac-IV [7], probablemente el primer ordenador con capacidad explícita de paralelismo. Este sistema, con hasta 256 unidades de procesamiento paralelo, fue el primer superordenador conectado a Internet, antes de que los equipos diseñados por Seymour Cray ocupasen dicho trono durante largo tiempo.

La construcción de ordenadores con múltiples zócalos de procesador se popularizó en la década de los 90, concretamente en los servidores de alta gama y las conocidas

como estaciones de trabajo (*workstations*) de alto rendimiento. Cada microprocesador aportaba su unidad de procesamiento interno, compartiendo con los demás micros, alojados en otros zócalos de la misma circuitería base, la memoria RAM y, generalmente, buses de comunicación dedicados que facilitaban las tareas de sincronización.

Una arquitectura más barata y de diseño más sencillo, en cuanto no requiere buses de comunicación entre zócalos, es el de los circuitos integrados que incorporan múltiples núcleos de procesamiento en la misma pastilla (*die*). Uno de los predecesores de esta idea fue el R65C00 de Rockwell, integrando a mediados de los 80 dos procesadores 6502[2] [8] en un mismo *chip*. Ya a principios del actual siglo tanto Intel como AMD incorporaron en sus microprocesadores dos o más núcleos, un diseño que en la actualidad resulta bastante corriente en configuraciones con 4/8 núcleos e incluso más en microprocesadores de alta gama como los Intel Xeon [4].

Fusionar varios núcleos completos en un mismo circuito integrado, compartiendo buses de comunicación con el exterior y una cierta cantidad de memoria común, resulta una solución casi obvia, que no sencilla de implementar, para mejorar el rendimiento ofrecido por un sistema de cómputo. Una vía alternativa, cuyo estudio tuvo lugar a mediados de los 90, fue la posibilidad de procesar dos o más hilos de ejecución en paralelo mediante lo que se conoce como SMT (*Simultaneous Multi-Threading*). El mítico Alpha EV8 de Digital Equipment Corporation (DEC) [9] fue el primer microprocesador en cuyo diseño se incluyó este planteamiento, muy utilizado en los productos de Intel desde hace algunos años bajo la denominación *Hyper-Threading* (HT).

En la actualidad los micros para sistemas de escritorio de Intel cuentan tanto con varios núcleos de procesamiento como con la tecnología HT, de forma que un equipo con cuatro núcleos, por ejemplo, puede ejecutar paralelamente hasta ocho hilos. También los microprocesadores móviles suelen disponer de dos, cuatro y hasta ocho núcleos, al tratarse esta de una configuración más eficiente en cuanto a consumo de energía se refiere que el aumento de la frecuencia de funcionamiento del sistema. Se trata, en consecuencia, de una de las formas de paralelismo más populares a día de hoy.

Con independencia de si el hardware a explotar consta de múltiples procesadores, procesadores con múltiples núcleos, núcleos de tipo HT o una combinación de varias de estas configuraciones, la herramienta fundamental con la que cuentan los programadores para extraer toda la potencia disponible son las librerías de tipo *pthreads* [10]. Los sistemas operativos modernos incorporan estos servicios, cuya finalidad es permitir a una aplicación crear múltiples hilos de ejecución simultánea, comunicarse con ellos, sincronizarlos, etc. En general, y a pesar de que el uso de *threads* resulta más común en lenguajes como C/C++, cualquier lenguaje de programación con acceso a los servicios del sistema puede sacar provecho de dichos servicios. El lenguaje Java, por ejemplo, pone a disposición de los programadores esa API a través de su máquina virtual (JVM).

[2] El 6502 fue un microprocesador muy popular en los 70-80 debido a su bajo precio. Fue la base de múltiples microordenadores, entre ellos los primeros modelos fabricados por Apple, Commodore y Atari.

2.3. Sistemas de computación distribuida

Paralelamente a la evolución de las CPU muchas otras tecnologías estrechamente relacionadas con la informática, en particular las relativas a sistemas de comunicación, fueron surgiendo y popularizándose. Estas hicieron posible conectar múltiples ordenadores formando redes de una manera relativamente barata, especialmente desde que Ethernet [11] pasó a ser un estándar en 1983. Interconectando ordenadores relativamente baratos es posible configurar *clústeres* con una gran potencia de cómputo, siguiendo el paradigma Beowulf [12] desarrollado en 1994 para la NASA. Esto permite, hasta cierto punto, tratar todo el *clúster* como un superordenador, con un gran número de unidades de procesamiento y mayor capacidad de memoria.

El principal obstáculo que representa la explotación de un sistema de cómputo distribuido, un *clúster* de máquinas, estriba en la distribución del trabajo entre las distintas máquinas y su sincronización. El método habitual consiste en habilitar un intercambio de mensajes entre los equipos de la red, ya sea gestionado por la propia aplicación o por un *middleware* que se ejecuta entre esta y el propio sistema operativo. Para ello, a lo largo del tiempo, se desarrollaron soluciones como RPC (*Remote Procedure Call*) [13], CORBA (*Common Object Request Broker Architecture*) [14] o Java RMI (*Remote Method Invocation*) [15]. El esquema fundamental consta de dos capas de software adicionales, el *proxy* y el *stub*, que permiten a la aplicación llamar a los métodos que la componen como si estuviesen ejecutándose de forma local, aunque en realidad lo hagan en otra máquina.

Un esfuerzo conjunto de decenas de empresas a principios de los 90 dio como fruto la especificación de MPI (*Message Passing Interface*) [16], un protocolo de intercambio de mensajes para aplicaciones distribuidas con interfaces para multitud de lenguajes de programación. En los últimos años, con el advenimiento del concepto *big data*, han ganado un considerable terreno soluciones como Apache Hadoop y Apache Spark [17]. Estas facilitan un mayor nivel de abstracción respecto a MPI, ofreciendo al desarrollador mecanismos que, siguiendo el enfoque divide y vencerás, distribuyen los datos y el trabajo entre las máquinas que forman el *clúster*.

3 De los *shaders* a la GPGPU

La evolución en el desarrollo de las CPU y los mecanismos de paralelización descritos en las anteriores secciones ha sido promovida, principalmente, por la necesidad de satisfacer la demanda que organizaciones y empresas hacían para poder ejecutar aplicaciones capaces de procesar un volumen de datos creciente y cada vez más complejo. Un ejemplo de estas son los grandes sistemas de bases de datos que, diariamente, atienden a miles de usuarios simultáneamente. Al mismo tiempo, aunque por una razón muy distinta como era la ejecución de juegos cada vez más sofisticados, el hardware de generación de gráficos experimentaba asimismo un acelerado avance. En la actualidad la industria del videojuego ha superado económicamente a la del cine [18], moviendo miles de millones de euros anualmente. Este volumen de negocio ha generado un gran beneficio para empresas con NVIDIA, revirtiendo en una inversión en investigación prácticamente sin precedentes.

Las GPU cuentan con unidades de procesamiento especializado, no de propósito general como las CPU, si bien han ido adquiriendo paulatinamente algunas de las capacidades de estas últimas. La diferencia fundamental estriba en que una GPU puede integrar miles de núcleos de procesamiento, frente a las pocas decenas de las CPU más potentes a día de hoy. Esto ha provocado que se desarrollen abundantes soluciones de procesamiento masivo de datos basados en GPU, siendo uno de los ejemplos más representativos las redes de aprendizaje profundo o *Deep Learning*. El más reciente producto de esta gama, la NVIDIA DGX-1 [19], es capaz de entrenar un modelo complejo de este tipo en 2 horas, cuando el mismo modelo requería más de 160 horas en un sistema con doble procesador Xeon con 14 núcleos/28 hilos cada uno.

El hardware de vídeo inicialmente no era programable, limitándose a convertir los datos alojados en una zona de memoria del ordenador en una señal interpretable para una pantalla. La elaboración de los gráficos era trabajo de la CPU, mientras el adaptador de vídeo leía a intervalos regulares el contenido de esa zona de memoria y actualizaba la señal de vídeo. Manteniendo un cauce (*pipeline*) gráfico no programable, poco a poco los adaptadores de vídeo fueron incorporan funciones de procesamiento gráfico parametrizables. Gracias a dichas funciones se podían aplicar sombreados, iluminación, etc., a los objetos de una escena gráfica. La llegada de las GPU programables [20] sustituyeron las funciones fijas parametrizables por funciones definidas por el usuario, conocidas habitualmente como *shaders*.

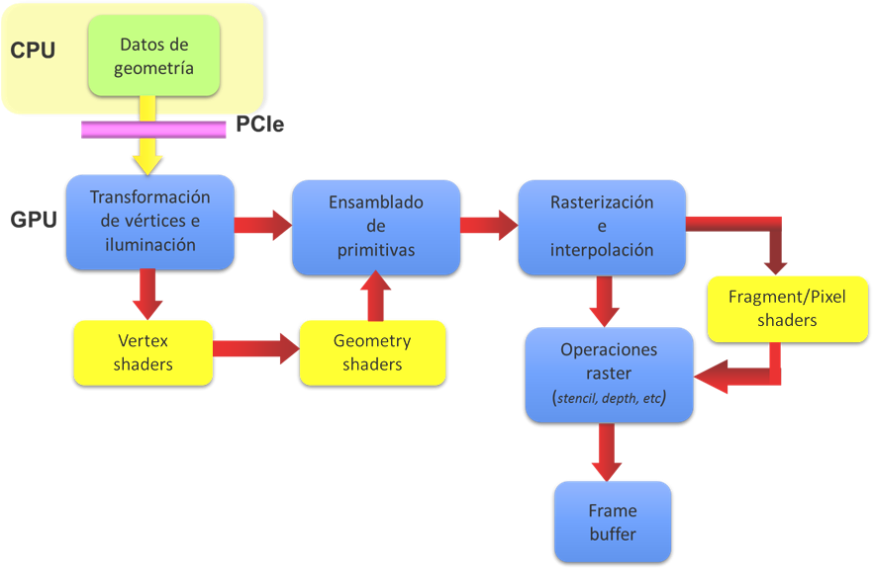


Figura 1. Estructura de bloques del *pipeline* de una GPU actual.

Los *shaders* son pequeños programas especializados en la manipulación de vértices correspondientes a los objetos gráficos, su geometría y sus píxeles, aplicando rotaciones, efectos de iluminación, texturas, etc. Existen múltiples versiones de la especificación de programación de *shaders*, denominadas *Shader Model*, y lenguajes para crearlos desarrollados por empresas como NVIDIA y Microsoft. Entre dichos lenguajes se

encuentran NVIDIA Cg (*C for graphics*), HLSL (*High-Level Shader Language*) y GLSL (*OpenGL Shading Language*). A pesar de que las operaciones realizadas por los *shaders* siempre están orientadas a la producción de gráficos, actuando sobre miles de vértices y píxeles en tiempo real, en realidad no dejan de ser funciones de tipo SIMD, potencialmente aplicables a muchos otros problemas.

El concepto GPGPU, la implementación de soluciones de propósito general mediante programación en GPU ya definida en 2004 en [21], comienza a ser accesible para los desarrolladores a partir de 2007/2008, a raíz de la introducción de tecnologías como CUDA y ATI Stream. Estas fueron las primeras herramientas de desarrollo en permitir la ejecución de código en GPU no relacionado específicamente con la generación de gráficos. CUDA, y posteriormente OpenCL y WebCL, han llevado a las técnicas GPGPU a convertirse en el principal paradigma actual de la computación de altas prestaciones.

4 Desarrollo de soluciones GPGPU

Para aprovechar la potencia de las GPU es necesario contar con herramientas de desarrollo adecuadas, capaces de explotar el alto nivel de paralelismo que ofrecen estos dispositivos. Hasta no hace mucho dichas herramientas eran bastante primitivas, ya que su objetivo era facilitar exclusivamente la programación de *shaders*. Estos son pequeños bloques de código que aplican un cierto procesamiento a los vértices de la geometría de una escena y los fragmentos resultantes de la *rasterización*. Ese código se ejecuta paralelamente en cada núcleo, lo cual permite aplicar un cierto algoritmo masivamente a miles o millones de vértices y píxeles.

Estos bloques de código tienen una longitud generalmente muy limitada y se programan en una suerte de lenguaje ensamblador a medida, por lo que difícilmente pueden aplicarse más que a la función para la que están pensados desde un principio. Como se indicó anteriormente existen diferentes versiones, denominadas *Shader Models*, que han ido evolucionando en paralelo a Microsoft DirectX y OpenGL y que tanto ATI/AMD como NVIDIA han ido implementando en su hardware. Al desarrollar una aplicación gráfica se utiliza una API, como las citadas DirectX u OpenGL, para escribir el código que se ejecutará en la CPU, usando el ensamblador del *shader model* correspondiente para escribir el código a ejecutar en la GPU. Tanto el tipo de operaciones que puede llevar a cabo ese código como la memoria a la que tiene acceso están limitados.

El desarrollo de Cg [22] por parte de NVIDIA, hace unos 15 años, fue un primer avance al facilitar la codificación de funciones a ejecutar en la GPU. En lugar de escribir el código en ensamblador se usa un lenguaje de más alto nivel, similar a C. Una función como la mostrada en el siguiente fragmento se ejecutaría una vez para cada vértice, pero no de manera secuencial sino paralelamente:

```
void processVertex(
  in float4 location : POSITION,
  out float4 locationD : POSITION,
  out float4 colorD : COLOR0,
```

```
    const uniform float4x4 ModelViewMatrix,
    const uniform float4 color)
{
    locationD = mul(location, ModelViewMatrix);
    colorD = color;
}
```

La limitación de Cg es que se trata de un lenguaje dirigido específicamente a la generación de gráficos. A medida que el número de núcleos de proceso en una GPU se fue incrementando, y ganando en rendimiento al poder operar con datos en coma flotante, se hizo cada vez más patente la necesidad de aprovechar esa potencia bruta de cálculo para propósitos alternativos, aparte de la evidente aplicación en videojuegos de última generación. Solamente se precisaban herramientas de trabajo de corte más general, con un espectro de aplicación más amplio. La primera respuesta a esta necesidad fue CUDA [23], desarrollado por NVIDIA, seguida de OpenCL [24] y WebCL [25]. En esta sección se introduce cada una de estas tecnologías.

Sobre estas tecnologías de bajo nivel se han desarrollado soluciones de más alto nivel y específicas para ciertos lenguajes. Por ejemplo se han desarrollado paquetes o librerías para programación GPU desde Matlab, Java, R, Python, etc.

4.1. CUDA

La mayoría de los lenguajes de programación no cuentan con estructuras nativas que faciliten la paralelización de procesos, entendiendo como tales partes de un algoritmo que pueden ser ejecutados de manera simultánea y no como lo que se entiende por procesos en el contexto de un sistema operativo. Es cierto que existen API y bibliotecas de funciones, como la anteriormente citada *pthreads*, que facilitan hasta cierto punto la programación paralela, pero prácticamente ninguna de ellas está pensada para ejecutar el código explotando una GPU. En la mayoría de los casos lo único que hacen es iniciar varios hilos de ejecución dejando en manos del sistema operativo el reparto de tiempo de proceso entre las unidades con que cuente la CPU. Para trasladar la aplicación a otro tipo de procesador, así como para ampliar o reducir el número de hilos en ejecución, es corriente tener que alterar, o incluso rescribir por completo, el código fuente.

La solución que ofrece CUDA (*Compute Unified Device Architecture*) es mucho más flexible y potente y, además, se basa en estándares existentes. Los programas se escriben en lenguaje C, no en el ensamblador de un cierto procesador o en un lenguaje especializado como es el caso de Cg. Esto facilita el acceso a un grupo mucho mayor de programadores. Al desarrollar una aplicación CUDA el programador escribe su código como si fuese a ejecutarse en un único hilo, sin preocuparse de crear y lanzar *threads*, controlar la sincronización, etc. Ese código será ejecutado posteriormente en un número arbitrario de hilos, asignado cada uno de ellos a un núcleo de procesamiento, de manera totalmente transparente para el programador. Este no tendrá que modificar el código fuente, ni siquiera recompilarlo, dependiendo de la arquitectura del hardware donde vaya a ejecutarse. Incluso existe la posibilidad de recompilar el código fuente, dirigido originalmente a ejecutarse sobre una GPU, para que funcione so-

bre una CPU clásica, asociando los hilos CUDA a hilos de CPU en lugar de a núcleos de ejecución de GPU. Obviamente el rendimiento será muy inferior ya que el paralelismo al nivel de CPU no es, actualmente, tan masivo como en una GPU.

Los objetivos planteados en el desarrollo de CUDA han dado como fruto un conjunto de tres componentes, disponibles gratuitamente para Windows, GNU/Linux y OS X. El controlador CUDA es el componente básico, ya que es el encargado de facilitar la ejecución de los programas y la comunicación entre CPU y GPU. Este controlador se aplica a prácticamente toda la gama hardware de NVIDIA: GeForce 8XX, 9XX y GTX 2XX y posteriores, así como a la línea de adaptadores Quadro y los procesadores Tesla. En cualquier caso, se requiere una cantidad mínima de 256 MB de memoria gráfica para poder funcionar, por lo que en adaptadores con menos memoria no es posible usar CUDA. Instalado el controlador, el siguiente componente fundamental para el desarrollo de aplicaciones es el *toolkit* CUDA, compuesto a su vez de un compilador de C llamado `nvcc`, un depurador específico para GPU, un perfilador de código y un conjunto de bibliotecas con funciones de utilidad ya predefinidas, entre ellas la implementación de la Transformada rápida de Fourier (FFT) y unas subrutinas básicas de álgebra lineal (BLAS). El tercer componente de interés es el *CUDA Developer SDK*, un paquete formado básicamente por código de ejemplo y documentación. Se ofrece más de medio centenar de proyectos en los que se muestra cómo integrar CUDA con DirectX y OpenGL, cómo paralelizar la ejecución de un algoritmo y cómo utilizar las bibliotecas FFT y BLAS para realizar diversos trabajos: generación de un histograma, aplicación de convolución a una señal, operaciones con matrices, etc. Conjuntamente estos tres componentes ponen al alcance del programador todo lo que necesita para aprender a programar una GPU con CUDA y comenzar a desarrollar sus propias soluciones, apoyándose en código probado como el de los ejemplos facilitados o el de las bibliotecas FFT y BLAS.

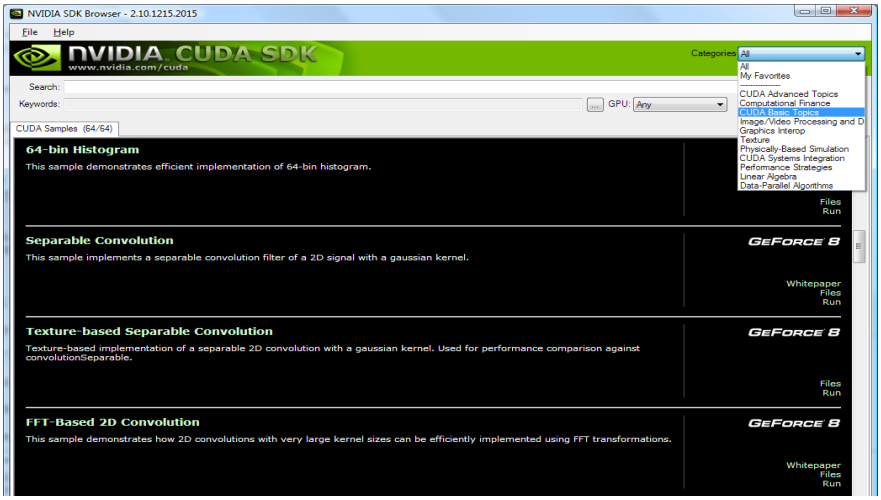


Figura 2. El NVIDIA CUDA SDK ofrece múltiples herramientas al desarollador.

El código de un programa escrito para CUDA siempre estará compuesto de dos partes: una cuya ejecución quedará en manos de la CPU y otra que se ejecutará en la

GPU. Al código de la primera parte se le denomina código para el *host* y al de la segunda código para el dispositivo. Al ser procesado por el compilador `nvcc`, el programa generará por una parte código objeto para la GPU y, por otra, código fuente u objeto para la CPU. El código objeto específico para la GPU se denomina *cubin*. El código fuente para la CPU será procesado por un compilador de C/C++ corriente, enlazando el código *cubin* como si de un recurso se tratase. La finalidad del código *host* es inicializar la aplicación, transfiriendo el código *cubin* a la GPU, reservando la memoria necesaria en el dispositivo y llevando a la GPU los datos de partida con los que se va a trabajar. Esta parte del código puede escribirse en C o en C++, lo cual permite aprovechar el paradigma de orientación a objetos si se quiere. El código a ejecutar en el dispositivo debe seguir estrictamente la sintaxis de C, contemplándose algunas extensiones de C++. Normalmente se estructurará en funciones llamadas *kernels*, cuyas sentencias se ejecutarán en paralelo según la configuración hardware del dispositivo final en el que se ponga en funcionamiento la aplicación.

Lo que hace el entorno de ejecución de CUDA, a grandes rasgos, es aprovechar el conocido como paralelismo de datos o SIMD, consistente en dividir la información de entrada, por ejemplo una gran matriz de valores, en tantos bloques como núcleos de procesamiento existan en la GPU. Cada núcleo ejecuta el mismo código, pero recibe unos parámetros distintivos que le permiten saber la parte de los datos sobre los que ha de trabajar. El sencillo ejemplo mostrado a continuación corresponde a una función *kernel* muy simple, cuyo objetivo es hallar el producto escalar de una matriz por una constante. La función solamente opera sobre un elemento de la matriz, el que le indica la variable `threadIdx.x` que identifica el hilo en que está ejecutándose el código. Esta función se ejecutaría paralelamente en todos los núcleos de la GPU, por lo que en un ciclo se obtendría el producto de una gran porción de la matriz o incluso de esta completa, dependiendo de su tamaño y el número de núcleos disponibles.

```
__global__ void ProdEscalar(
    float* M1,
    float* M2,
    float Constante)
{
   // Se obtiene el número de thread
   int i = threadIdx.x;
   // y se procesa el dato que corresponde
   M2[i] = M1[i] * Constante;
}
```

En una CPU moderna, como puede ser un Intel Core i7, el desarrollador puede dividir los datos de entrada en cuatro o seis partes, dependiendo del número de núcleos con que cuente el microprocesador, pero sin ninguna garantía de que se procesarán en paralelo salvo que se programe explícitamente el reparto trabajando a bajo nivel. Es decir, el propio programador ha de crear los hilos de ejecución independientes y asignar cada una de las particiones de datos a un hilo. En una GPU y usando CUDA, por el contrario, esos datos se dividirán en bloques mucho más pequeños, al existir 240,

512, 1024 o más núcleos de procesamiento, garantizándose la ejecución en paralelo si necesidad de recurrir a la programación en ensamblador.

4.2. OpenCL

¿Cómo podría una misma aplicación, con el objetivo de explotar toda la potencia de un ordenador actual, aprovechar tanto el paralelismo de la CPU como de la GPU? Utilizando CUDA esto implica crear hilos en la CPU para las tareas MIMD y *kernels* en la GPU para las tareas SIMD, con herramientas distintas y en ocasiones lenguajes de programación distintos, integrando los diversos componentes de la mejor manera posible. Una alternativa que cuenta con el favor de una gran parte de la industria es OpenCL, un estándar que hace posible la ejecución de código escrito en C/C++ distribuyendo las tareas entre CPU y GPU, sin que importe el fabricante de la GPU. Esa capacidad, no obstante, es actualmente teórica en un escenario en el que cada empresa intenta favorecer su oferta sobre la de la competencia. Si se cuenta con un hardware gráfico de NVIDIA, los controladores OpenCL de esta empresa solamente reconocerán como dispositivo la propia GPU, tal y como se aprecia en la ventana de la izquierda de la Figura 3. Los controladores de AMD/Intel, por el contrario, sí reconocen la CPU como dispositivo (ventana de la derecha), pero obviamente no pueden usar la GPU de NVIDIA sino las suyas propias.

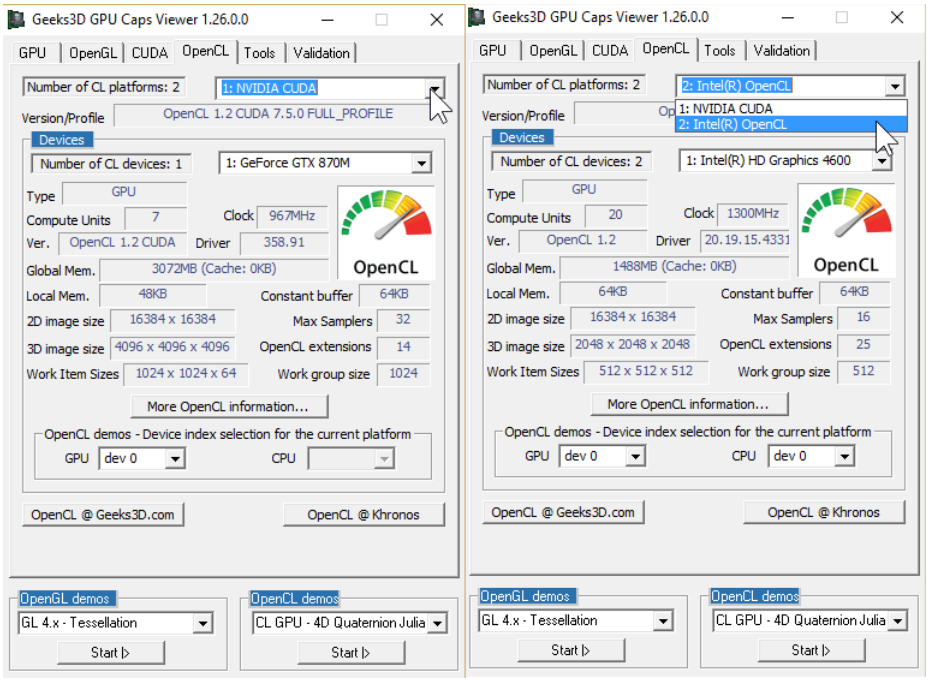


Figura 3. Configuración de controladores OpenCL de NVIDIA/Intel en un mismo sistema.

OpenCL ofrece al desarrollador una interfaz de más alto nivel para la programación de tareas que han de ejecutarse en paralelo, abstrayéndole de las diferencias entre GPU y CPU y los distintos hardware gráficos. Se trata, por tanto, de una herramienta que facilita el aprovechamiento del paralelismo sobre plataformas heterogéneas, incluyendo CPU, GPU y potencialmente otro tipo de procesadores especializados como las FPGA.

Uno de los obstáculos que habitualmente encontramos al iniciarnos en el uso de una nueva tecnología, en este caso concreto la programación en GPGPU con CUDA u OpenCL, estriba en que habitualmente se requiere el conocimiento de un lenguaje relativamente complejo, como puede ser C/C++, y el conjunto de herramientas asociado para compilación, depuración, etc. En los últimos años este proceso se ha visto simplificado gracias a lenguajes como Python que, a través de los paquetes `pycuda` y `pyopencl`, hacen posible el acceso interactivo a las funciones ofrecidas por los respectivos paquetes de desarrollo. La información relativa a las plataformas disponibles, anteriormente mostrada en la Figura 1 y obtenida con una herramienta específica, puede obtenerse desde Python previa instalación del paquete `pyopencl` con las siguientes sentencias:

```
import pyopencl as cl

cl.VERSION
cl.get_platforms()
cl.get_platforms()[0].get_devices()
```

4.3. WebCL

Al tratar el tema del paralelismo siempre se asume que el objetivo es emplearlo en programas que serán instalados y ejecutados en un ordenador de forma nativa, ya sea a través de compiladores específicos para un hardware concreto o el uso de máquinas virtuales que se ocupan de los detalles de más bajo nivel. En cualquier caso, son aplicaciones dirigidas a funcionar bajo una cierta configuración: microprocesador, GPU, sistema operativo, etc. De un tiempo a esta parte, sin embargo, la web se ha convertido en una plataforma más para la ejecución de aplicaciones superando esas especificidades, no precisando más que un navegador que se ajuste a los estándares: HTML5, CSS3 y Javascript.

El código Javascript de una aplicación web puede ejecutar código en múltiples hilos en la CPU gracias a los *Web Workers*, pero hasta hace poco no existía un método que permitiese aprovechar la gran potencia con la que cuentan las GPU actuales con independencia del hardware, sistema operativo o navegador que el usuario emplee para acceder a la aplicación web. Por suerte es una situación que ha cambiado gracias a WebCL.

WebCL es a las aplicaciones web lo que OpenCL a las aplicaciones nativas: una capa de abstracción que permite ejecutar código en paralelo tanto en CPU como en GPU, sin importar el fabricante del hardware ni el sistema operativo empleado. Es un estándar regido por el Khronos Group y que siguen múltiples fabricantes de hardware, entre ellos AMD, NVIDIA e Intel. En realidad, WebCL es en esencia un enlace o

binding para poder acceder a OpenCL desde Javascript. Al tratarse de un estándar en desarrollo ningún navegador lo incluye actualmente. Para poder probarlo es necesario instalar un complemento en el navegador. Completado ese paso, es posible crear desde Javascript un objeto `WebCLComputeContext` (en la implementación de Samsung) y usarlo para obtener información sobre el hardware disponible, preparar el código a ejecutar paralelamente y enviarlo a la CPU/GPU.

El grupo de trabajo encargado de este estándar dentro del Khronos Group fue creado en 2011 y, en principio, su objetivo es facilitar una guía de implementación para fabricantes conservando la esencia de OpenCL y poniendo especial énfasis en el tema de la seguridad, ya que el código se ejecutaría en el ordenador de los usuarios al acceder a una aplicación desde su navegador, sin necesidad de instalar ni ejecutar explícitamente un programa externo.

5 Conclusiones

La potencia de los ordenadores se ha incrementado varios órdenes de magnitud en las últimas décadas, haciendo realidad el desempeño de tareas que parecían prácticamente imposibles hace no muchos años. El aprovechamiento de esa potencia, sin embargo, ha demandado el desarrollo de nuevas arquitecturas hardware y tecnologías software a lo largo de este tiempo. Si en los primeros años de la informática personal, en los 70-80, el incremento en la frecuencia de reloj de un microprocesador se traducía automáticamente en un mayor rendimiento del software, desde principios de este siglo esa mejora no se obtiene de manera tan directa, siendo preciso recurrir a técnicas de paralelización basadas en los principios SIMD/MIMD.

En este trabajo se ha resumido la evolución de las técnicas de paralelización desde casi los albores de los primeros ordenadores, y las técnicas de tiempo compartido, hasta las modernas tecnologías GPGPU que hacen posible explotar la enorme potencia del hardware gráfico actual. El desafío futuro probablemente estribará en cómo facilitar el aprovechamiento del cada vez mayor número de procesadores alojados en un ordenador, CPU, GPU y otros integrados especializados, ofreciendo una infraestructura de desarrollo homogénea sobre un hardware cada más potente pero también más heterogéneo.

Referencias

1. Ganesan, R., Govindarajan, K., & Wu, M.: Comparing SIMD and MIMD Programming Modes. J. Parallel Distrib. Comput., 35, 91-96 (1996).
2. Ritchie, D., & Thompson, K.: The UNIX Time-Sharing System (Reprint). Commun. ACM, 26, 84-89 (1983).

3a. GE-635 System manual. General Electric. 1964.

3b. Emerson W. Pugh, Lyle R. Johnson, John H. Palmer, IBM's 360 and Early 370 Systems, 360-363, MIT Press 2003.

4. Intel® 64 and IA-32 Architectures Software Developer Manuals. `http://www.intel.com/content/www/us/en/processors/architectures-software-developer-manuals.html`

5. IBM System/360 Operating System: MVT Guide, OS Release 21.7. Sixth Edition. IBM. August 1974. GC28-6720-5.
6. Neumann, J.V.: First draft of a report on the EDVAC (1945). IEEE Annals of the History of Computing, 15, 27-75 (1993).
7. Adve, S.V. et al.: Parallel Computing Research at Illinois the Upcrc Agenda Parallel@illinois: Pioneering and Promoting Parallel Computing (2008).
8. James, G., Silverman, B., & Silverman, B.: Visualizing a classic CPU in action: the 6502. SIGGRAPH (2010).
9. Supnik, R.M.: Digital's Alpha Project. Commun. ACM, 36, 30-32 (1993).
10. Mueller, F.: A Library Implementation of POSIX Threads under UNIX. USENIX (1993).
11. Metcalfe, R.: Ethernet: Distributed Packet Switching for Local Computer Networks. BERKELEY (1976).
12. Becker, D.J., Dorband, J.E., Packer, C.V., Ranawake, U.A., Sterling, T.L., & Savarese, D.: BEOWULF: A Parallel Workstation for Scientific Computation. ICPP (1995).
13. Birrell, A., & Nelson, B.J.: Implementing Remote Procedure Calls. ACM Trans. Comput. Syst., 2, 39-59 (1984).
14. Beeharry, A., Bouguettaya, A., & Delis, A.: Managing Persistent Objects in a Distributed Environment. ADC (1987).
15. Domajnko, T., Hericko, M., Juric, M.B., Krisper, M., Rozman, I., & Zivkovic, A.: Java and Distributed Object Models: An Analysis. SIGPLAN Notices, 33, 57-65 (1998).
16. Bruck, J., Dolev, D., Ho, C., Rosu, M., & Strong, H.R.: Efficient Message Passing Interface (MPI) for Parallel Computing on Clusters of Workstations. SPAA (1995).
17. Ingram, J.E., Monteith, J.Y., & McGregor, J.D.: Hadoop and its Evolving Ecosystem. ICSOB (2013).
18. Chatfield, T. Videogames now outperform Hollywood movies. The Guardian, September, 27 (2009).
19. http://www.NVIDIA.com/object/deep-learning-system.html.
20. Kilgard, M.J., Lindholm, E., & Moreton, H.: A User-programmable Vertex Engine. NVIDIA Corporation (2001).
21. Buck, I., Govindaraju, N.K., Harris, M.J., Krüger, J., Luebke, D., Lefohn, A.E., Purcell, T.J., & Woolley, C.: GPGPU: general purpose computation on graphics hardware. SIGGRAPH (2004).
22. Akeley, K., Glanville, R.S., Kilgard, M.J., & Mark, W.R.: Cg: a system for programming graphics hardware in a C-like language. ACM Trans. Graph., 22, 896-907 (2003).
23. Buck, I.: GPU computing with NVIDIA CUDA. SIGGRAPH (2007).
24. Gohara, D., Stone, J.E., & Shi, G.: OpenCL: A Parallel Programming Standard for Heterogeneous Computing Systems. Computing in Science and Engineering, 12, 66-73 (2010).
25. Foley-Bourgon, V., Hendren, L.J., Khan, F., Kathrotia, S., & Lavoie, E.: Using JavaScript and WebCL for numerical computations: a comparative study of native and web technologies. DLS (2014).

WepSIM: Simulador modular e interactivo de un procesador elemental para facilitar una visión integrada de la microprogramación y la programación en ensamblador

Alejandro Calderón Mateos, Félix García Carballeira, Javier Prieto Cepeda

Computer Architecture Group, Computer Science and Engineering Department
Universidad Carlos III de Madrid, Leganés, Madrid, Spain.
acaldero,fgcarbal@inf.uc3m.es, javpriet@pa.uc3m.es

Resumen Lo que diferencia WepSIM[1] de otros simuladores usados en la enseñanza de Estructura de Computadores está en tres aspectos importantes. Primero, ofrece una visión integrada de la microprogramación y de la programación en ensamblador, dando la posibilidad de trabajar con distintos juegos de instrucciones. Segundo, permite al estudiante una mayor movilidad al poder usarse también en dispositivos móviles.
Tercero, tiene un diseño modular, es posible añadir, quitar o modificar los elementos existentes. Busca un equilibrio entre simplicidad, para facilitar la enseñanza, y el detalle, para imitar la realidad. Una de las grandes ventajas del simulador WepSIM es que no está limitado a un juego de instrucciones concreto, permitiendo definir un amplio juego de instrucciones de procesadores reales o inventados.
Este artículo describe WepSIM y los resultados en la primera experiencia de su uso en la asignatura de Estructura de Computadores impartida en el Grado en Ingeniería Informática de la Universidad Carlos III de Madrid.

Palabras Clave: Estructura de Computadores, Procesador Elemental, Microprogramación, Programación en ensamblador

Abstract There are three differences between WepSIM[1] and other simulation tools used in Computer Structure Teaching. First one, it offers an integrated vision of microprogramming and assembly programming where different instructions sets can be defined. Second one, it provides more flexibility to students by using mobile devices. Third one, it was created with a modular design. It is possible to add, remove or modify the existing elements, and it is simple enough to be used for teaching but with enough details to mimic the reality.
This paper introduces WepSIM, and the results obtained in the first experience of usage in the Computer Structure course from the Bachelor's

[1] http://www.arcos.inf.uc3m.es/~ec-2ed

Degree in Computer Science and Engineering in the Universidad Carlos III de Madrid.

Keywords: Computer Structure, Elemental Processor, Microprogramming, Assembly programming

1 Introducción

Hay distintos simuladores que se pueden utilizar para trabajar con los principales aspectos que se tratan en las asignaturas de Estructura y Arquitectura de Computadores: ensamblador, caché, etc. Aunque la idea de usar distintos simuladores cae dentro de la estrategia de "divide y vencerás", hay dos principales problemas con estos simuladores: cuanto más realistas son más compleja se hace la enseñanza (tanto del simulador como de la tarea simulada), y cuantos más simuladores se usan más se pierde la visión de conjunto.

Hay otro problema no menos importante: la mayoría de los simuladores están pensados para PC. Uno de los objetivos que nos planteamos con WepSIM es que pudiera ser utilizado en dispositivos móviles (*smartphones* o *tablets*), para ofrecer al estudiante una mayor flexibilidad en su uso.

Además de tener un simulador portable a distintas plataformas, el simulador ha de ser lo más autocontenido posible de manera que integre la ayuda principal para su uso (no como un documento separado que sirva de manual de uso para ser impreso).

Por todo ello, nos hemos planteado cómo ofrecer un simulador que sea simple y modular, y que permita integrar la enseñanza de la microprogramación con la programación en ensamblador. En concreto, puede utilizarse para microprogramar un juego de instrucciones y ver el funcionamiento básico de un procesador, y para crear programas en ensamblador basados en el ensamblador definido por el anterior microcódigo. Esto es de gran ayuda, por ejemplo, para la programación de sistemas dado que es posible ver cómo interactúa el software en ensamblador con el hardware en el tratamiento de interrupciones. La idea es ofrecer un simulador que ofrezca una visión global de lo que pasa en hardware y software.

En este artículo vamos a introducir el simulador WepSIM (*Web Elementary Processor SIMulator*) que hemos diseñado e implementado buscando alcanzar los anteriores objetivos. El simulador WepSIM está evolucionando con la experiencia de su uso, y queremos compartir los pilares del mismo, las ideas que hay detrás de la propuesta y los primeros resultados alcanzados.

El resto del documento se organiza como sigue: la segunda sección del documento introduce la arquitectura de WepSIM y el modelado de hardware que hemos propuesto. La tercera sección introduce el procesador elemental que se simula usando la arquitectura anteriormente presentada. También se describe la forma de definir un determinado juego de instrucciones así como el microcódigo asociado al mismo. Con este formato los estudiantes pueden diseñar e implementar código fuente en cualquier ensamblador previamente definido. La cuarta

sección resume los principales aspectos de su implementación. La quinta sección repasa el estado del arte, y finalmente la sexta sección muestra las conclusiones y trabajos futuros.

2 La arquitectura de WepSIM y el modelo hardware

La arquitectura de la solución presentada (véase la Figura 1) en este trabajo tiene tres elementos principales: (1) un modelo hardware que permite definir el hardware a usar, (2) un modelo software que permite definir el microcódigo/-lenguaje máquina a utilizar y (3) un *motor* que simula el funcionamiento del hardware ejecutando el microcódigo/lenguaje máquina definido anteriormente.

El modelo hardware permite definir los distintos elementos típicos de un computador (memoria principal, procesador, etc.) de una forma modular. La forma de definir estos elementos equilibra dos objetivos contrapuestos: es suficientemente completa como para imitar los principales aspectos de la realidad, pero es lo suficientemente mínima para facilitar su uso. Ante todo se persigue que sea una herramienta didáctica.

El modelo software permite definir el microcódigo y el ensamblador basado en este microcódigo de la forma tan intuitiva posible. El ensamblador a usar viene dado por un conjunto de instrucciones que puede ser definido por el profesor/alumno e intenta ser lo suficientemente flexible como para poder definir diferentes tipos y juegos de instrucciones, como por ejemplo MIPS o ARM.

El tercer elemento de la arquitectura propuesta es un *motor* que toma como entrada el modelo hardware descrito y el modelo software de trabajo, y se encarga de mostrar el funcionamiento del hardware con el software dado.

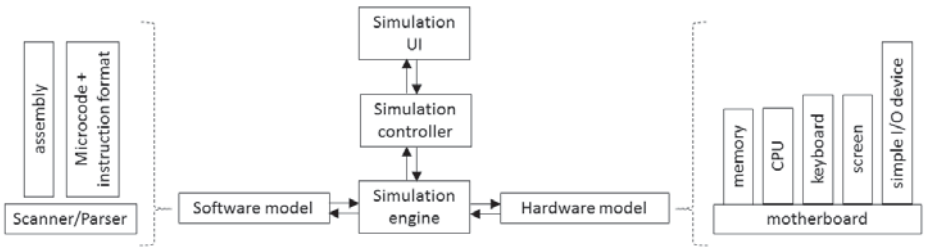


Figura 1. Arquitectura del simulador WepSIM.

La figura 1 resume la arquitectura de WepSIM. El punto de inicio es el modelo hardware que describe el procesador a ser simulado. Ello incluye el procesador, la memoria y algunos dispositivos de E/S: teclado, pantalla y un dispositivo de E/S simple que genera interrupciones. El modelo hardware describe el estado global del procesador. A partir del estado global del procesador, el motor de simulación actualiza el estado en cada ciclo de reloj.

La unidad de control simulada almacena las señales de control de cada ciclo en una memoria de control. La memoria de control tiene todos los microprogramas para las instrucciones con las que trabaja el procesador, y el *fetch* para leer la instrucción de memoria y decodificarla.

El microcódigo (el contenido de la memoria de control) junto con el formato de cada instrucción (campos de la instrucción y su longitud) se describe en un fichero de texto. El modelo software lee este fichero, lo traduce a binario y lo carga en el procesador. La definición del lenguaje ensamblador a utilizar se describe junto con el microcódigo, y el modelo software permite traducir a binario programas escritos en dicho ensamblador.

El motor de simulación pregunta al subsistema del modelo software por el microcódigo definido, la descripción del formato de instrucción y el contenido de la memoria principal. Los binarios se cargan en los elementos del modelo hardware, y a continuación el motor de simulación actualiza el estado global en cada ciclo de reloj.

WepSIM dispone de un controlador de simulación que se encarga de actualizar el ciclo de reloj y mostrar el estado global. El subsistema de interfaz de simulación actualiza la interfaz de usuario. Cuando el usuario usa la interfaz de usuario para solicitar una operación, el subsistema de interfaz de simulación traslada la petición al controlador de simulación. Como se puede ver, se usa un Modelo-Vista-Controlador (MVC) básico para la arquitectura de WepSIM.

2.1 El modelo hardware en WepSIM

El modelo hardware que usa WepSIM permite definir los distintos elementos típicos de un computador (memoria principal, procesador, etc.) de una forma modular y de manera que sea posible añadir, quitar o modificar estos elementos.

La figura 2 introduce el modelo propuesto. Cada elemento del circuito se describe como una caja negra con posibles entradas, posibles salidas y señales de control (que controlan las posibles transformaciones de las entradas a las salidas). El subsistema del modelo hardware transforma esta caja negra en dos conjuntos de objetos: estados y señales. Un estado tiene un identificador (el nombre), el valor (un valor entero) y un valor inicial (el valor por defecto). Los valores que puede tomar son valores naturales dentro de un rango, dado por el número de bits con los que se representa el estado. Una señal es un estado especial que controla el valor de otros estados o señales. Hay dos atributos asociados a las señales (y no a los estados): el tipo de señal (por nivel o por flanco) y su comportamiento. Para cada valor de señal, una cadena de caracteres describe en un Lenguaje Simple lo que la señal mueve o transforma. Este Lenguaje Simple se compone principalmente de instrucciones que representan las operaciones elementales.

Las figuras 3 y 4 muestran dos ejemplos: un triestado y un registro.

El triestado controla dos estados: el estado del bus al que se conecta BUS_IB y el estado del registro de entrada, REG_RT1 en este caso. Ambos representan el valor a la salida de la puerta (BUS_IB) y el valor del registro RT1 (REG_RT1). La señal T4 se encarga de indicar cuándo el valor del registro RT1 se envía a la salida. Esta señal T4 es una señal por nivel (tipo: L), con valor cero no

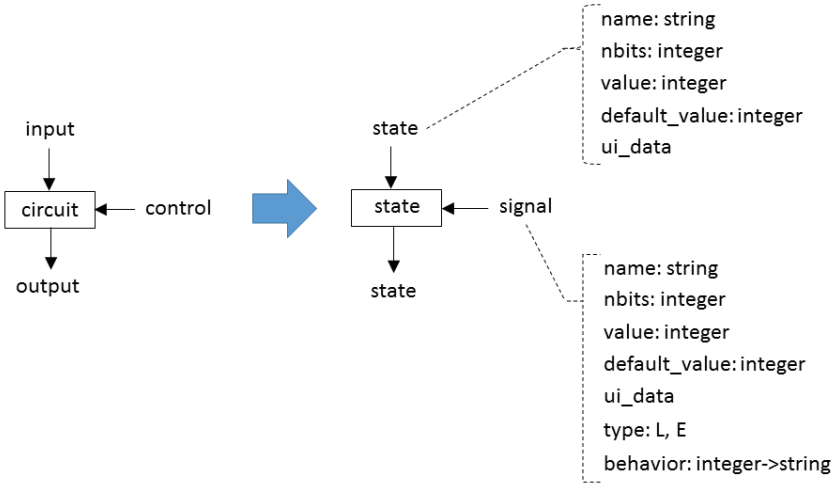


Figura 2. Modelado del hardware.

tiene efecto (comportamiento "NOP"). Cuando el valor de la señal es uno entonces el comportamiento es el de copiar el valor del registro RT1 a la salida (comportamiento "MV BUS_IB REG_RT1").

El ejemplo con el registro (véase la figura 4) es similar. En este caso trabaja con dos estados: el contenido del registro RT1 y el contenido situado a la entrada (BUS_IB). La señal C4 controla cuándo se almacena en el registro RT1 el valor que hay en la entrada. La diferencia está en el tipo de señal: C4 es una señal por flanco de bajada (tipo: E), por lo que al final del ciclo de reloj (pasa de uno a cero) si la señal vale uno entonces el comportamiento es el de copiar el valor situado a la entrada al registro (comportamiento "MV REG_RT1 BUS_IB").

El Lenguaje Simple usado para definir los comportamientos añade a las operaciones elementales otras operaciones necesarias. Por ejemplo disparar una señal ("FIRE C4") que ayuda a propagar el efecto de una señal al re-evaluar la señal inmediata que podría verse afectada. Otro ejemplo lo encontramos en dos operaciones que pueden ser muy útiles a la hora de depurar: imprimir el valor de un estado ("PRINT_E BUS_IB") e imprimir el valor de una señal ("PRINT_S C4").

3 El procesador elemental WepSIM

Usando el modelo simplificado introducido anteriormente es posible definir todos los elementos de nuestro procesador elemental. La figura 5 muestra la estructura de Procesador Elemental WepSIM.

Los elementos del procesador se describen con 57 estados y 62 señales, de forma similar a los ejemplos comentados anteriormente (de hecho T4 y C4 están

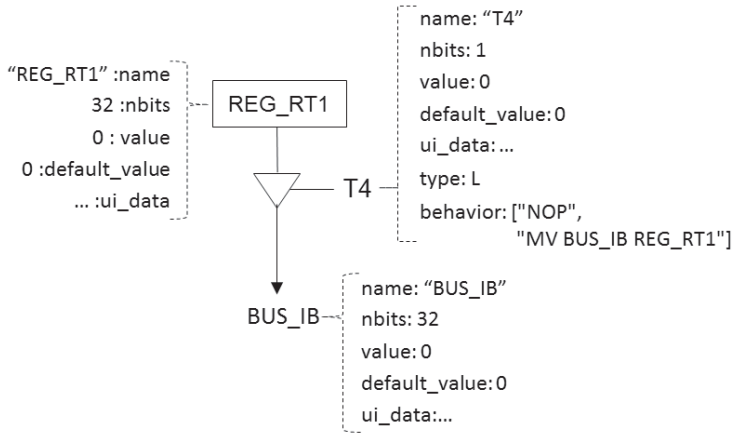


Figura 3. Ejemplo de modelado de una puerta triestado.

incluidas entre las 62 señales, así como REG_RT1 y BUS_IB (bus interno) están incluidos entre los 57 estados). El lenguaje que describe los comportamientos incluye 46 operaciones. La ALU precisa de 15 de ellas, y la memoria y dispositivos de E/S precisan alrededor de 15 operaciones más. WepSIM tiene un módulo de memoria, un dispositivo teclado, un dispositivo pantalla y un dispositivo de E/S genérico que se usa para trabajar con interrupciones. También incluye un procesador que está etiquetado en la figura 5 como "Processor".

WepSIM es un procesador de 32 bits que direcciona la memoria por bytes, que tiene un banco de registros (con 32 registros) y dos registros adicionales (RT1 y RT2) que no son visibles para el programador de ensamblador. Desde los registros es posible enviar los valores para operar en una ALU con 15 operaciones aritmético-lógicas, que incluyen las más comunes. El registro PC tiene su propio operador de sumar cuatro, de forma que la ALU no es necesaria para esta operación. El resultado se puede almacenar en un registro temporal (RT3) transparente también para el programador de ensamblador, o se puede enviar directamente al bus interno a través del triestado correspondiente.

El registro de estado (SR) se puede actualizar con los flags resultantes de la última operación de la ALU (O, N y Z). Para ello SELEC/SelP representa un bloque de circuitos (de más alto nivel que un multiplexor, demultiplexor, etc.) que permite indicar qué parte del registro de estado (SR) actualizar. A la derecha de SELEC/SelP llega los bits del registro de estado SR como entrada (Input=O N Z I U) y SelP permite seleccionar qué grupo de estos bits se actualizará en el registro de estado: los bits O, N y Z con los valores procedentes de la ALU, el bit I con el valor indicado o el bit U con el valor indicado para el mismo.

El registro de instrucción (IR) tiene asociado un módulo selector (circuito de más alto nivel que un multiplexor, etc.) que permite seleccionar una porción del valor binario almacenado en el registro de instrucción que pasará hacia T3.

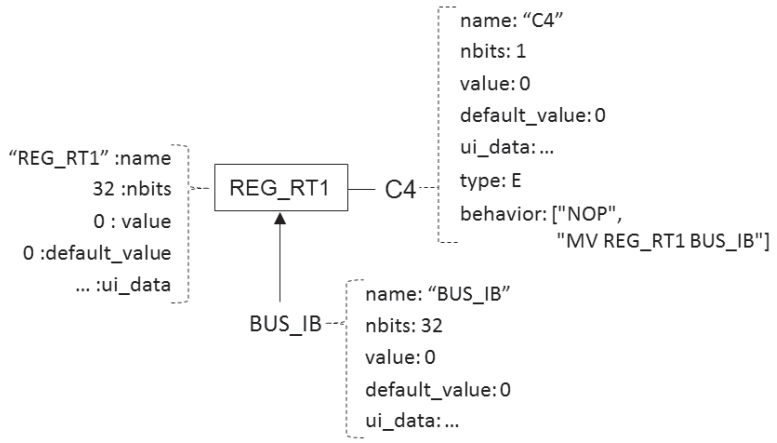


Figura 4. Ejemplo de modelado de un registro.

En concreto, se indica la posición (*Offset*, donde 0 represente el bit menos significativo del registro IR) inicial y el número de bits (*Size*) a tomar a partir de dicha posición inicial, así como si se desea hacer extensión de signo (*SE*) antes de pasar el valor a la entrada de T3.

Los registros MAR y MBR se usan para almacenar la dirección y el contenido asociado a esta dirección en las operaciones de lectura/escritura con la memoria. La memoria está diseñada para un funcionamiento síncrono o asíncrono. Actualmente funciona de forma síncrona, pero dispone de la señal MRdy para en un futuro trabajar de forma asíncrona. El circuito de selección permite indicar qué porción de la palabra de memoria es la que se desea (un byte, dos bytes o una palabra completa de cuatro bytes).

Se dispone también de tres dispositivos de E/S: un teclado, una pantalla y un dispositivo genérico que se puede configurar para generar diversos tipos de interrupciones.

Finalmente, la Unidad de Control genera las señales de control para cada ciclo de reloj.

La figura 6 muestra la Unidad de Control en más detalle. Se trata de una unidad de control microprogramada con secuenciamiento implícito. Las señales de control para el ciclo de reloj actual se almacenan en el registro de microinstrucción (aquel con los campos A0, B, C, SelA, etc.). El contenido de este registro proviene de la memoria de control, concretamente del contenido en la posición a la que apunta el registro de microdirección ($\mu Addr$). La microdirección almacenada en este registro puede modificarse usando el multiplexor "MUX A". Hay cuatro opciones: la microdirección actual más uno, una microdirección indicada en la propia microinstrucción (que se solapa con SelA, SelB y parcialmente con SelE), la primera microdirección asociada al campo de código de operación de la instrucción del registro IR, y finalmente el valor cero, que es la dirección de la

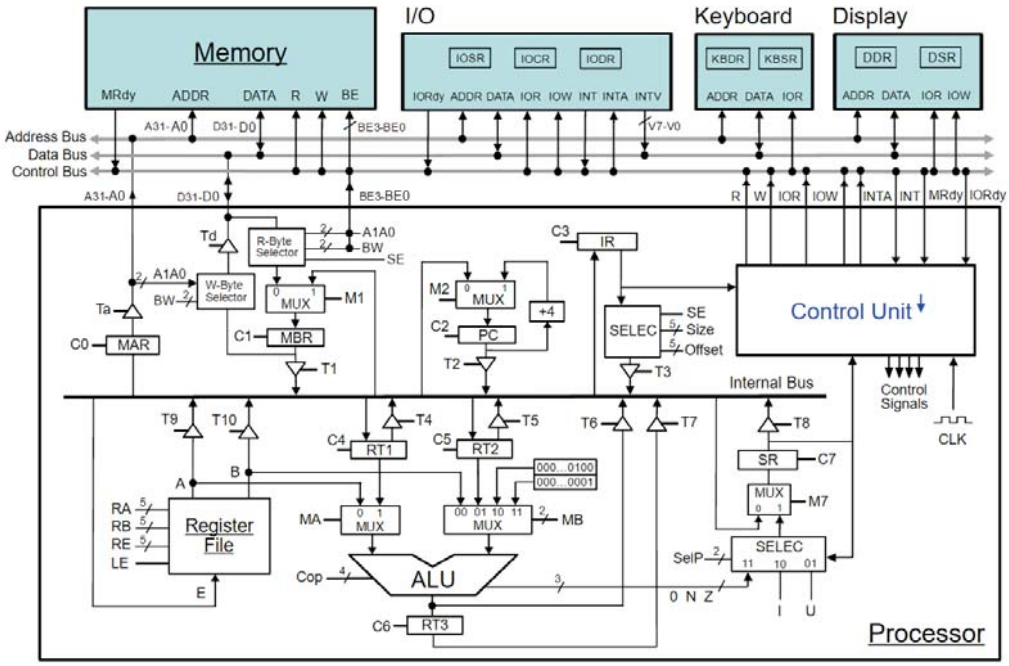


Figura 5. Estructura del Procesador Elemental WepSIM.

memoria de control donde se almacena la microrrutina correspondiente al *fetch*. La microdirección se puede seleccionar de forma condicional, para ello se usa el multiplexor "MUX C"que permite seleccionar los bits del registro de estado (SR) o valores de las señales de control de E/S.

Para generar los valores correspondientes a las señales selectoras del banco de registros RA, RB y RE se usan los circuitos selectores SelRA, SelRB y SelRE. Estos selectores toman como entrada los 32 bits del registro de instrucción (IR) por un lado y el campo SelA, SelB y SelE por otro, de forma que toman SelX como el desplazamiento dentro del registro de instrucción (de 0 a 32) desde donde tomar los siguientes 5 bits correspondientes a las señales RA, RB o RE. Permiten por tanto seleccionar 5 bits consecutivos de los 32 bits del registro de instrucción.

El multiplexor MR permite indicar si las señales RA, RB y RE serán literalmente los valores almacenados en SelA, SelB y SelE (MR=1) o bien si SelA, SelB y SelE indican el desplazamiento dentro de la instrucción donde están los valores a utilizar para RA, RB y RE. Esto último permite que en la instrucción se pueda indicar los registros a usar como operandos en el banco de registros, en lugar de indicarlos desde la microinstrucción.

Para la señal Cop (código de operación en la ALU) la señal MC permite tomar el valor SelCop de la microinstrucción (MC=1) o bien los 4 bits menos significativos del registro de instrucción (MC=0), es decir IR3-IR0.

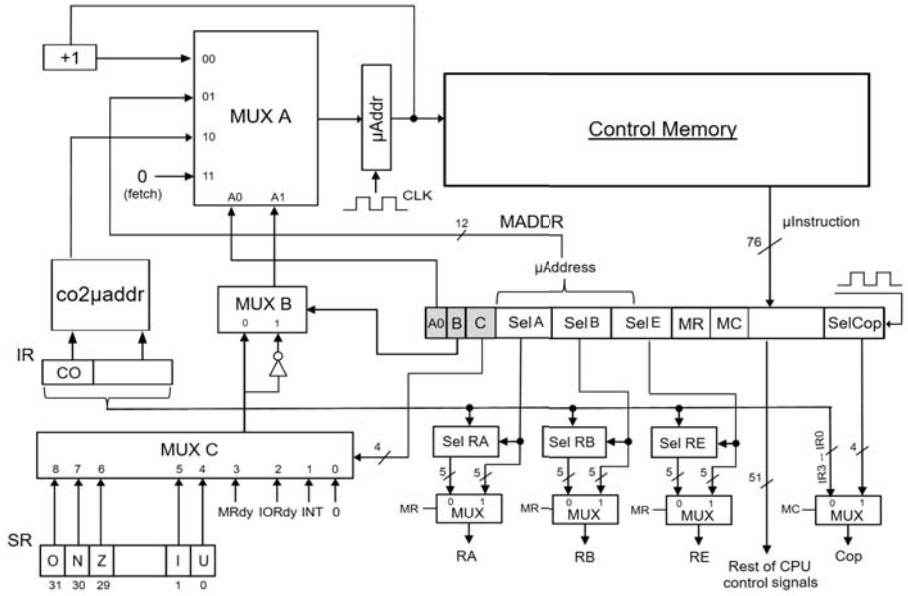


Figura 6. Unidad de control del procesador WepSIM.

En [1] se puede encontrar una descripción más detallada del procesador descrito anteriormente.

3.1 Definición del microcódigo y del conjunto de instrucciones a usar en WepSIM

Una vez definido el procesador elemental usando el modelo hardware propuesto, toca describir el conjunto de instrucciones que es capaz de ejecutar así como el microcódigo que lo orquesta. En un fichero de texto se define el formato de las instrucciones máquina junto con el cronograma asociado a la ejecución de cada una de las instrucciones máquina. El listado 1.1 muestra un ejemplo de definición para la instrucción **li** (*load inmmediate*), que almacena un valor inmediato en un registro.

El fichero con el cronograma de fetch y todos los cronogramas de las instrucciones define el microcódigo para la plataforma WepSIM. El simulador permite la definición de diferentes juegos y formatos de instrucciones. Inicialmente hemos implementado un subconjunto de las instrucciones del MIPS, pero es posible definir instrucciones de otros conjuntos de forma similar. En este fichero se pueden asignar códigos simbólicos a los registros del banco de registros, lo que permite que en los programas escritos en ensamblador se puedan usar dichos símbolos (por ejemplo, registro `$t3` en el listado 1.2).

El campo "co" identifica el código de instrucción máquina, que es un número binario de 6 bits. Esto permite definir hasta 64 instrucciones distintas. Dado

```
li reg val
{
       co=000010,
       nwords=1,
       reg=reg(25,21),
       val=inm(15,0),
       {
          (SE=0, OFFSET=0, SIZE=10000, SE=1, T3=1,
           LE=1, MR=0, SELE=10101, A0=1, B=1, C=0)
       }
}
```

Listado 1.1. Ejemplo de formato de instrucción y su cronograma asociado.

que los últimos 4 bits de la instrucción pueden usarse para seleccionar la operación en la ALU, es posible seleccionar hasta 16 operaciones aritmético-lógicas con un mismo código de instrucción, por lo que se podrían tener 79 (63+16) instrucciones en total.

Cuando WepSIM carga el microcódigo, cada código de instrucción tiene asociado una dirección de comienzo en la memoria de control donde se almacena el cronograma asociado. Esta tabla con dos columnas (el código de instrucción y su dirección de comienzo asociada en la memoria de control) se carga en la ROM $co2\mu Addr$ mostrada en la figura 6.

El campo "nwords" define cuantas palabras precisa la instrucción para su definición y carga en memoria. Una palabra en WepSIM son 4 bytes.

Para cada campo de la instrucción (*reg* y *val* del ejemplo del listado 1.1) se define el bit inicial, el bit final (ambos incluidos) y el tipo de campo (registro, valor inmediato, dirección absoluta y dirección relativa a PC). Una vez definido el formato, se definen todas las microinstrucciones que necesita la instrucción máquina definida para su ejecución. Todas las microinstrucciones se encuentran encerradas entre llaves y cada microinstrucción está formada por una lista de tuplas (señal, valor) encerradas entre paréntesis. Para la instrucción definida en el listado 1.1 se precisa de una sola microinstrucción, en la que se indican qué señales de activan durante un ciclo de reloj. Para las señales no indicadas se asume que su valor es 0 durante el ciclo de reloj correspondiente.

Una vez cargado el microcódigo en WepSIM, es posible cargar cualquier fichero ensamblador que haya sido codificado usando las instrucciones máquina definidas anteriormente en el microcódigo.

En el listado 1.2 se muestra un ejemplo de código fuente en ensamblador que se puede usar en WepSIM. Este ejemplo en particular (listado 1.2) muestra un código estilo MIPS. Para que un programa en ensamblador pueda utilizar la instrucción de carga inmediata li (load inmmediate) y de suma add (addition),

```
.text
main:  li  $t3 8
       li  $t5 10
       add $t6 $t3 $t5
```

Listado 1.2. Ejemplo de código fuente en ensamblador.

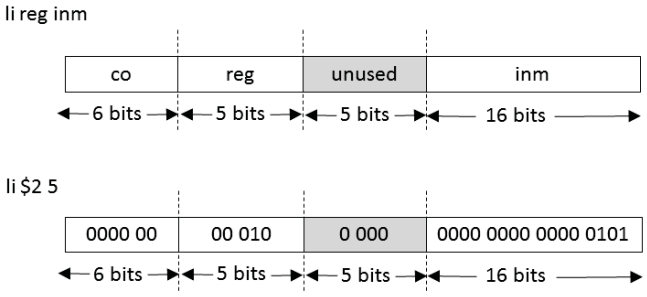


Figura 7. Formato de instrucción descrito en el microcódigo y ejemplo de su traducción en binario.

deben haber sido definidas previamente en el microcódigo. WepSIM puede comprobar los errores de sintaxis y construir el binario mediante el rellenado de los campos descritos en la definición del microcódigo correspondiente a la instrucción. La figura 7 muestra un ejemplo de traducción a binario para la instrucción `li $2 5` en función del formato definido en el listado 1.1. También se debe haber definido en el fichero de microcódigo el valor del registro asociado a la etiqueta `$2` (00100 en este caso).

Una de las grandes ventajas del simulador WepSIM es que no está limitado a un conjunto de instrucciones concreto. Se puede definir un amplio conjunto de instrucciones de procesadores reales o inventados. Se puede usar para añadir, por ejemplo, a un conjunto de instrucciones MIPS, otras instrucciones diferentes no incluidas en dicho conjunto de instrucciones.

4 El diseño y la implementación de WepSIM

Para acercar la docencia al alumno se ha tenido que buscar una forma de implementar y desplegar el simulador de forma que pueda ejecutarse en el mayor número de plataformas posibles, priorizando las plataformas móviles de forma que, por ejemplo, cuando un estudiante viaje en transporte público pueda usar el simulador desde un dispositivo móvil (con pantalla de cinco o más pulgadas) sin

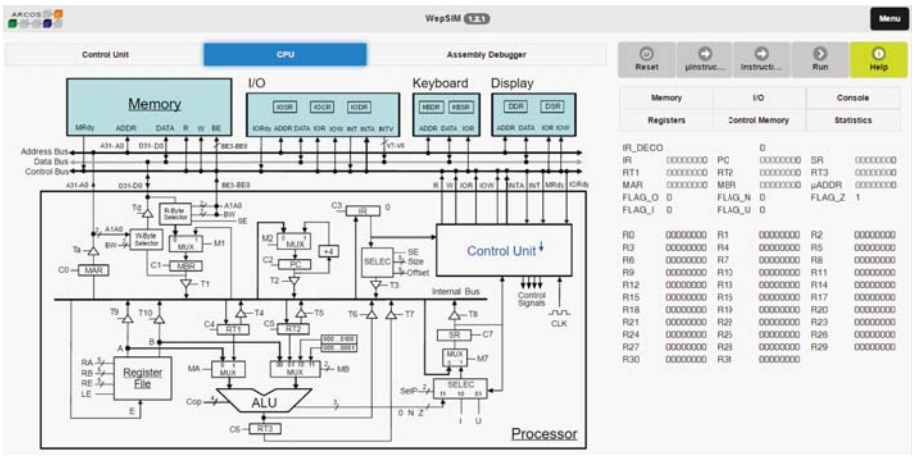


Figura 8. Pantalla principal de WepSIM.

necesitar recursos adicionales (apuntes, papel, etc.). Este acercamiento también implica de forma implícita que hay que ofrecer una ayuda en línea, que sea lo más autocontenida posible

El prototipo presentado en este artículo se implementó en HTM5 (HTML + JavaScript + CSS) de forma que es posible ejecutarlo en cualquier plataforma (*smartphones*, *tablet*, portátil, PC, etc.) en que pueda ejecutarse Mozilla Firefox o Google Chrome. El prototipo inicial se compone de 13 ficheros, con cerca de 5 000 líneas de código disponible con licencia LGPL 3.0 en GitHub: `https://github.com/wepsim/wepsim`. El prototipo depende de los siguiente frameworks/bibliotecas bien conocidos: JQuery, JQueryUI, JQuery Mobile, Knockout y BootStrap.

También recientemente hemos creado una aplicación móvil para Android y Windows Phone gracias al proyecto Apache Cordova. Para el próximo curso el simulador podrá ser ejecutado desde un navegador Web y desde la aplicación móvil correspondiente. La segunda opción es interesante para poder trabajar sin conexión a Internet.

4.1 La interfaz de usuario

La figura 8 muestra la pantalla principal del simulador. Hay tres áreas de trabajo bien diferenciadas: la de ejecución, la pantalla de trabajo con el microcódigo y la pantalla de trabajo con el ensamblador.

La pantalla de trabajo con el microcódigo (ver figura 9) permite indicar el microcódigo que tendrá la memoria de control y que controla el comportamiento del procesador. Se puede cargar un microcódigo de ejemplo (botón *Example*), se puede modificar el contenido con el editor, guardar con *Save* y cargar con *Load*.

Una vez disponible la especificación del microcódigo deseado, se microensambla con el botón *µassemble* y es posible ver el contenido de la memoria de control

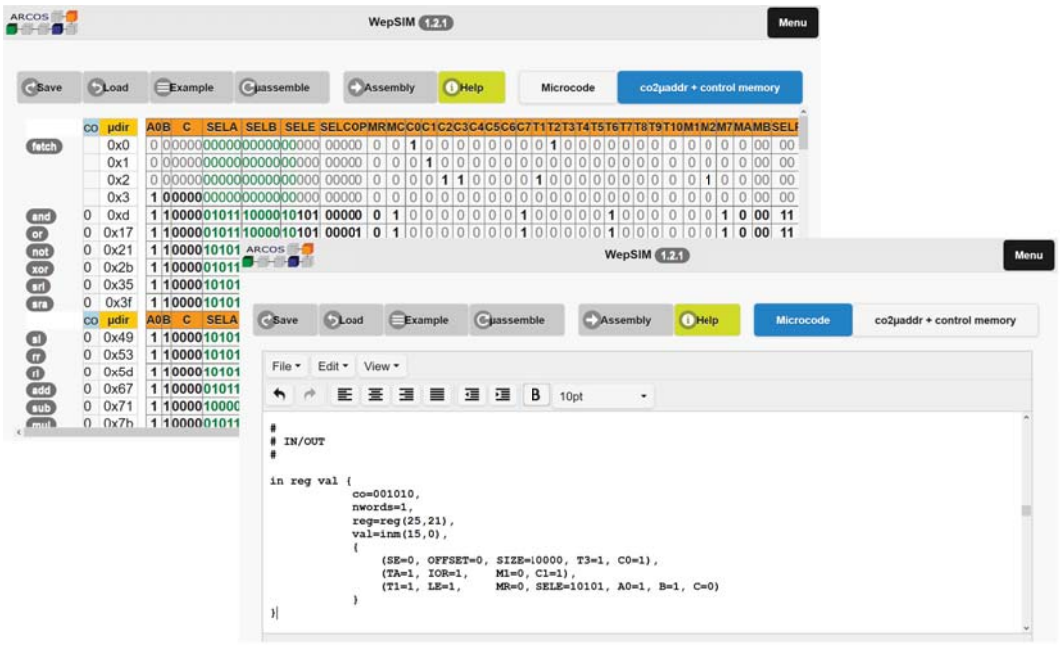


Figura 9. Pantallas de carga de microcódigo y visualización en binario.

y de la ROM de traducción (*co2µaddr + control memory*) en binario, resultado del microensamblado.

El microcódigo define, al menos, el tratamiento para el *fetch* de cada instrucción así como las señales a generar para poder ejecutar cada instrucción de ensamblador. Es posible definir instrucciones de distintos tipos de procesador (MIPS, ARM, Intel, Z80, etc.) lo que facilita posteriormente cargar un programa que use el conjunto de instrucciones definido.

La pantalla de trabajo con ensamblador (ver figura 10) permite cargar en memoria un programa en ensamblador, de acuerdo a la sintaxis definida en la definición del microcódigo. De igual forma que antes, se puede cargar un programa de ejemplo (botón *Example*), puede editarse, guardar con *Save* y cargar posteriormente con *Load*.

A partir de un determinado programa escrito en ensamblador, se puede ensamblar (botón *Assemble*) y es posible ver el contenido de la memoria principal en binario (*Main Memory*), resultado del proceso de ensamblado del código ensamblador dado, usando el ensamblador definido en el microcódigo.

Una vez cargado el microcódigo y el programa en ensamblador, es posible ejecutar dicho programa instrucción a instrucción de ensamblador o bien ciclo a ciclo de cada microprograma correspondiente a cada instrucción. Esta integración permite a los estudiantes ver cómo se integran ambos niveles (véase la Figura 11).

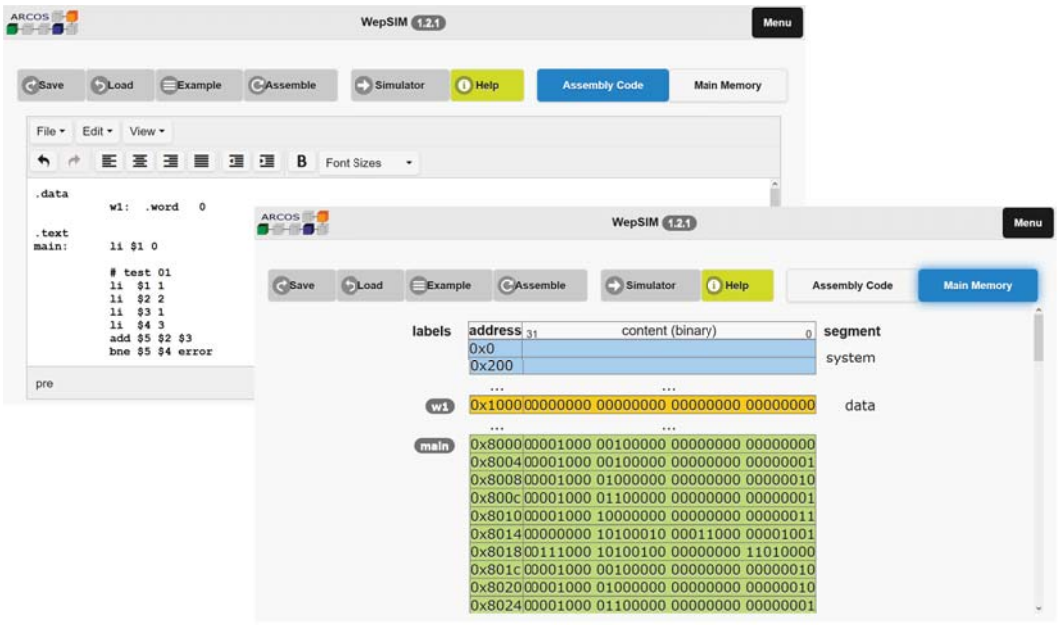


Figura 10. Pantallas de carga de código ensamblador y visualización en binario.

El diagrama del procesador elemental muestra con diferente color las señales activas (rojo), así como los caminos de datos activos (azul). Una señal está activa si aparece en la definición de la microinstrucción, y el camino de datos que dicha señal selecciona es el camino de datos activo asociado. De esta forma es más fácil e intuitivo seguir la pista de lo que está pasando en el procesador ciclo a ciclo. Destacar que el simulador es interactivo por defecto (es una opción configurable) de manera que si se quiere cambiar el valor de una señal sobre la marcha, solo se precisa hacer clic con el botón izquierdo del ratón en el nombre de la señal en el diagrama del procesador. Como se muestra en la figura 12, se puede seleccionar el valor (cero y uno) y se puede pedir una breve ayuda sobre la señal.

En el caso de ejecución instrucción a instrucción en ensamblador es posible usar la pestaña *Assembly Debugger* tal y como se muestra en la figura 8. Dicha pestaña permite ver el código en ensamblador, señalando la instrucción en curso y permitiendo el uso de puntos de ruptura (*Breakpoints*).

La implementación se ha realizado de forma que el simulador pueda ejecutarse sin mostrar el diagrama del procesador elemental. Ello permite que pueda utilizarse para una ejecución por lotes de pruebas consecutivas, útil por ejemplo para la corrección de distintos microcódigos.

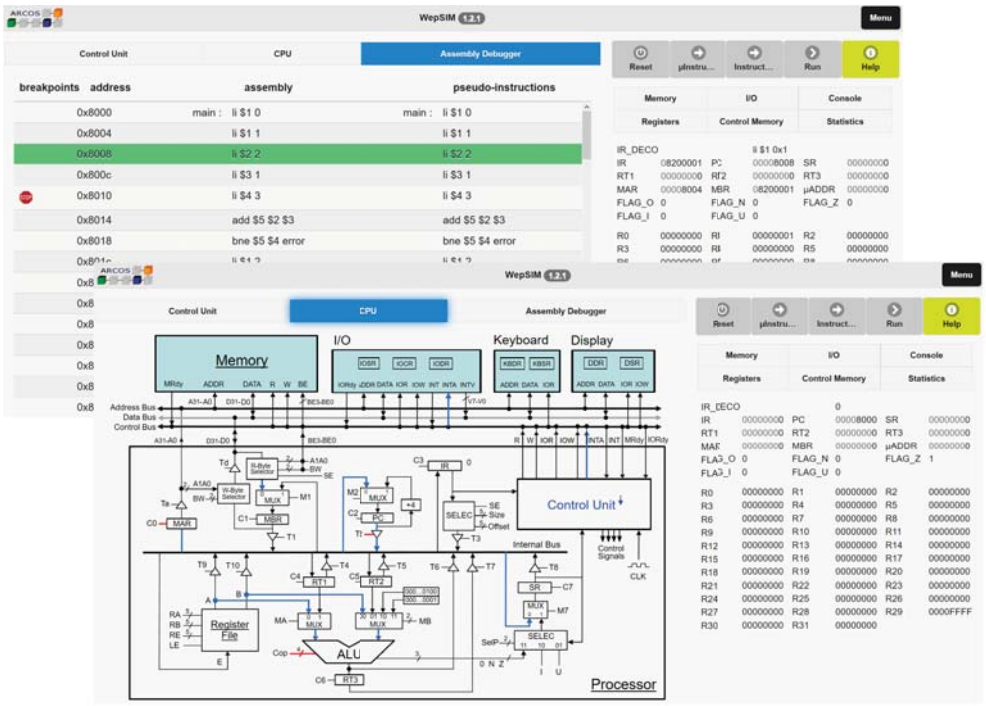


Figura 11. Pantallas que muestran la ejecución del simulador.

5 Estudio del uso del simulador WepSIM

Hemos analizado los registros del servidor Web desde donde se accede al simulador a través de Web para estudiar los dispositivos que acceden al simulador (ordenadores cliente). Durante la última semana de la entrega de la actividad docente el simulador WepSIM ha sido usado un 7 % con direcciones IP de la Universidad y un 93 % con direcciones IP de fuera de la Universidad. Estos resultados demuestran por tanto que hemos facilitado el uso de nuestra herramienta remotamente.

El segundo análisis realizado sobre los registros del servidor Web estudia el sistema operativo usado en los ordenadores cliente (tal y como se identifican ellos mismos). Queremos con ello estudiar si nuestro simulador WepSIM facilita a los estudiantes usar distintas plataformas de trabajo. Los resultados se muestran en la Figura 13, donde el eje y representa el número de accesos y el eje x la plataforma desde la cual provienen los accesos. El eje y usa una escala logarítmica. El grupo a la izquierda del eje x son los sistemas operativos de escritorio (Linux, Windows-NT y MacOS) mientras que el grupo de la derecha son los sistemas operativos para plataforma móviles (Android, iOS y Windows Phone). Estos resultados muestran la diversidad de los sistemas operativos usados, y por tanto que los estudiantes han podido usar distintas plataformas.

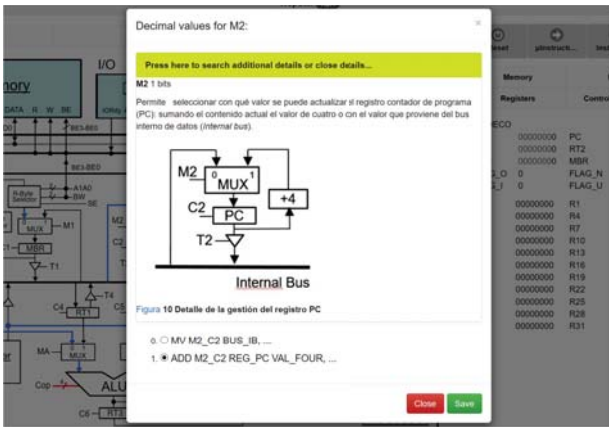


Figura 12. Detalle de la ventana emergente para cambiar interactivamente el valor de una señal. Incluye una breve ayuda de aprendizaje sobre la señal.

Para estudiar en qué horas se pide acceder al simulador, hemos analizado las horas de acceso en los registros del servidor Web. Los resultados se muestran en la Figura 14, donde el eje x representa la línea temporal del día sobre la que se señalan los accesos en cada instante del día producidos en la última semana de entrega de la práctica.

Salvo de la una de la noche hasta las ocho de la mañana donde los accesos son mínimos, el resto del día se realizaron accesos a WepSIM. Estos resultados nuevamente demuestran que hemos facilitado el uso de nuestra herramienta en las distintas horas del día, y desde distintas plataformas.

6 Estado del arte

Hay distintas herramientas muy útiles para labores docentes relacionadas con la asignatura de Estructura de Computadores, aunque como se comentó en la introducción, no hemos encontrado una herramienta como la presentada en este trabajo, especialmente en los siguientes aspectos:

- Permite a los estudiantes ofrecerles una visión interrelacionada de la microprogramación y la programación en ensamblador.
- Permite una flexibilidad en la plataforma usada (pensando en dispositivos móviles).
- Es lo más autocontenida posible (incorpora ayuda en línea y asociada en lo posible al contexto presentado).
- Presenta un modelo hardware que puede modificarse o ser ampliado si se desea.
- Permite definir un amplio conjunto de instrucciones máquina.
- Se puede usar como herramienta de microprogramación o de programación en ensamblador.

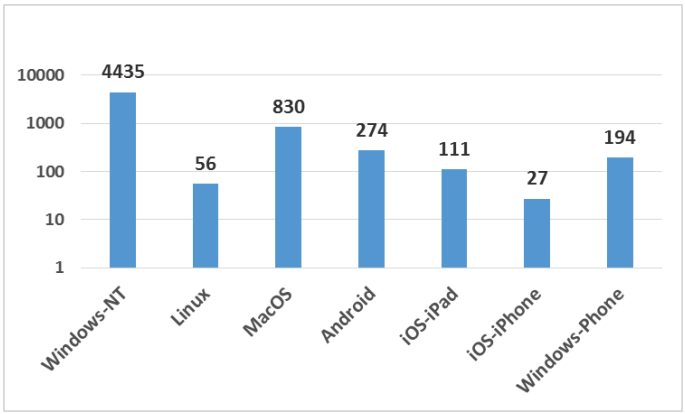


Figura 13. Sistema operativo de los ordenadores cliente que han usado WepSIM.

De entre los simuladores más conocidos para labores docentes, desde el punto de vista de programación en ensamblador, está SPIM y Mars. SPIM [7] es un simulador de un procesador MIPS de 32 bits que permite ejecutar (y depurar) programas en ensamblador para esta arquitectura. Este simulador creado por James R. Larus tiene versiones compiladas para Windows, Mac OS X y Unix/-Linux e incluso tiene una versión básica para Android [8]. Desafortunadamente no integra la visión para microprogramación, no está pensado su diseño para dispositivos móviles (la versión existente busca parecerse a la versión para ordenador) y se basa en el conjunto de instrucciones de MIPS32 (con la extensión de la instrucción syscall).

Mars [12] es también un simulador de un procesador MIPS de 32 bits similar a SPIM desarrollado en Java. Sus autores destacan la interfaz gráfica, la edición integrada y la reciente incorporación de una utilidad para acceder al hardware desde ensamblador. De igual forma que antes, no ofrece una visión integrada, no se encuentra versión alguna para dispositivos móviles y se centra en MIPS32.

Otro simulador conocido en entornos docentes es el em88110 descrito en [10], y como en casos anteriores, se centra en una arquitectura específica (MC88110) y no ofrece una ayuda en la movilidad del estudiante. Aunque integra el efecto de un procesador superescalar, no integra la microprogramación. Detrás de este simulador hay un trabajo muy interesante descrito en [5], donde se muestra la metodología detrás del simulador pc88110 para ser usado en prácticas. Ello incluye la descripción del entorno de prácticas, el sistema de entrega, el corrector automático, la detección de copias, etc.

Figura 14. Instantes en el tiempo del día en los que se ha pedido usar WepSIM.

Fuera del ámbito docente hay simuladores/emuladores como por ejemplo OVPsim [13] o GXemul [6] que permiten trabajar con distintas arquitecturas como por ejemplo ARM, MIPS, PowerPC, etc. En el caso de OVPsim, se ha usado para la investigación de plataformas de computación paralelas [11], co-diseño de hardware/software [9], etc. y aunque es posible su uso para labores docentes, por su nivel de detalle es posible que no sea la mejor herramienta con la que empezar a aprender.

Respecto a simuladores para labores docentes en microprogramación destacamos P8080E [3] desarrollado en el DATSI de la Facultad de Informática de la Universidad Politécnica de Madrid. A diferencia del P8080E nuestra propuesta (a) presenta una interfaz gráfica portable e interactiva, y (b) el ensamblador es definido en el microcódigo de forma que luego es posible ensamblarlo y generar el binario asociado. En el P8080E esta última labor se hace a mano, en la definición de las instrucciones no se incluye su formato y no está pensado para poder usarse para enseñar tanto microprogramación como ensamblador con la misma herramienta. Otros simuladores para microprogramación son el UT1000 [2] de 1989 y MicMac [4] de 1987. MicMac está pensado para usar su propio código máquina denominado Mac1 (no tanto diseñar nuevos), y UT1000 es descrito por sus autores como un simulador de una CPU de 16 bits con un secuenciador de AMD2910. Desafortunadamente UT1000 tras más de 25 años, es un proyecto a día de hoy no accesible.

7 Conclusiones y trabajos futuros

Este trabajo presenta un nuevo simulador que es intuitivo, portable y extensible. Permite definir diferentes juegos de instrucciones, y ejecutar y depurar código fuente que use el conjunto de instrucciones definido. Se basa en un modelo simple pero suficientemente capaz de mimetizar el funcionamiento básico de un procesador. También permite definir el comportamiento del procesador mediante microprogramación.

WepSIM permite a los estudiantes entender cómo funciona un procesador elemental de una forma fácil. Puede usarse desde un *smartphone*, una *tablet*, un portátil o un ordenador de escritorio con un navegador Web moderno, sin necesidad de instalación. Los estudiantes pueden interactuar con el simulador y aprender de esta forma cómo funciona un procesador elemental, incluyendo los mecanismos de interacción con el software de sistema. Integra tanto la programación en ensamblador como la microprogramación.

Hay varias líneas de trabajos futuros con las que ya estamos trabajando:

- Completar con una herramienta de pruebas basada en el motor de WepSIM que permita ejecutar un microcódigo con varios programas en ensamblador y un programa de ensamblador con varios microcódigos.
- Estamos trabajando en una aplicación móvil basada en el proyecto Apache Cordova para iOS, de forma que sea capaz de trabajar incluso sin conexión a Internet.

Hay más líneas de trabajos futuros que están siendo consideradas:

- Queremos estudiar el ensamblador de MIPS/ARM generado con gcc/clang de forma que pueda ser usado directamente en WepSIM.
- Introducir más elementos hardware, como una memoria caché.

Referencias

1. Carballeira, F.G., Pérez, J.C., Sánchez, J.D.G., Singh, D.E.: Problemas resueltos de estructura de computadores, segunda edición, vol. 1, pp. 1–307. Ediciones Paraninfo (2015)
2. Cornett, F.: The ut1000 microprogramming simulator: An educational tool. SIGARCH Comput. Archit. News 17(4), 111–118 (Jun 1989), http://doi.acm.org/10.1145/71317.71325
3. DATSI.FI.UPM.ES: P8080E (Apr 2016), http://www.datsi.fi.upm.es/docencia/Estructura/U_Control/
4. Donaldson, J.L.: Micmac: A microprogram simulator for courses in computer organization. SIGCSE Bull. 19(1), 428–431 (Feb 1987), http://doi.acm.org/10.1145/31726.31800
5. Dopico, A.G., de la Fuente, S.R., García, F.J.R.: Automatización de prácticas en entornos masificados. In: Actas de las IX Jornadas de Enseñanza universitaria de la Informática. pp. 119–126. Jenui 2003, Thomson-Paraninfo, Spain (2003), https://books.google.es/books?id=U07SAAAACAAJ
6. Gavare, A.: GXemul (Mar 2016), http://gxemul.sourceforge.net/
7. Larus, J.R.: SPIM (Feb 2016), http://spimsimulator.sourceforge.net/
8. Matak, J.: Assembly Emulator (Feb 2016), https://play.google.com/store/apps/details?id=gr.ntua.ece.assembly.emulator
9. Nita, I., Lazarescu, V., Constantinescu, R.: A new hw/sw co-design method for multiprocessor system on chip applications. In: Proceedings of the 5th IEEE/ACM International Conference on Hardware/Software Codesign and system Synthesis (ISSCS). pp. 1–4. IEEE Computer Society (2009)
10. Pérez Villadeamigo, J.M., de la Fuente, S.R., Cavanillas, R.M., García Clemente, M.I.: The em88110: Emulating a superscalar processor. SIGCSE Bull. 29(4), 45–50 (Dec 1997), http://doi.acm.org/10.1145/271125.271153
11. Pinto, C., Raghav, S., Marongiu, A., Ruggiero, M., Atienza, D., Benini, L.: Gpgpu-accelerated parallel and fast simulation of thousand-core platforms. In: The 11th International Symposium on Cluster, Cloud and Grid Computing (CCGRID). pp. 53–62. IEEE Computer Society (2011), http://dblp.uni-trier.de/db/conf/ccgrid/ccgrid2011.html#PintoRMRAB11
12. Sanderson, P., Vollmar, K.: MARS (Feb 2016), http://courses.missouristate.edu/kenvollmar/mars/
13. Software, I.: Open Virtual Platforms simulator (Mar 2016), http://www.ovpworld.org/

Gamificación en procesos de autoentrenamiento y autoevaluación. Experiencia en la asignatura de Arquitectura de Computadores

M. Espinilla, A. Fernández, J. Santamaría y A. Rivera

Departamento de Informática,
Universidad de Jaén, España
{mestevez,ahilario,jslopez,arivera}@ujaen.es

Resumen En la educación, la gamificación de procesos está suponiendo una excelente solución para aumentar la motivación del alumnado para desempeñar las actividades de su aprendizaje. Así, en dicho entorno, la gamificación se traduce en brindar a los estudiantes una motivación adicional inmediata que les permita alcanzar una tarea a largo plazo. Nuestro interés en este trabajo se enfoca en compartir la experiencia del uso de dinámicas y mecanismos de juego para incentivar el proceso de aprendizaje autónomo del alumno a través de los procesos de autoentrenamiento y autoevaluación en la preparación de una prueba objetiva para la asignatura de Arquitectura de Computadores en los estudios de Ingeniería en Informática de la Universidad de Jaén.

Palabras clave: Gamificación, autoentrenamiento, autoevaluación, aprendizaje autónomo, arquitectura de computadores.

Resumen Gamification is based on the application of concepts and techniques of games in other environments with the aim of encouraging certain behaviors or concrete actions. In education, the gamification is assuming an excellent solution to increase student motivation to perform self-learning activities. Therefore, the gamification translates to provide students with an immediate extra motivation that allows achieving a long-term task. Our interest in this paper focuses on sharing the experience of the use of dynamic and mechanics of games to improve the process of autonomous learning by means of the processes of self-training and self-assessment in the preparation of an choice test in the subject of Computer Architecture at the University of Jaén.

Keywords: Gamification, self-training, self-assessment, autonomous learning, computer architecture.

1 Introducción

La gamificación es una de las diez tendencias relacionadas con la tecnología más destacadas a nivel mundial. Consiste en la aplicación de conceptos y técnicas de juegos en otros entornos que previamente no son tan entretenidos o motivadores. El objetivo final consiste en incentivar determinados comportamientos o recompensar acciones

concretas [1]. La gamificación no se refiere estrictamente a la creación de juegos sino al empleo de mecanismos propios de juegos en entornos y aplicaciones no lúdicas. De este modo, tal y como hemos indicado previamente, se consigue potenciar la motivación, la concentración, el esfuerzo o la fidelización [2].

Entre todas sus posibles aplicaciones, la gamificación está suponiendo una excelente solución en el campo de la educación. Se ha observado que, en efecto, la motivación del alumnado para desempeñar las actividades de aprendizaje se ve beneficiada por el uso de este tipo de técnicas [3]. En dicho entorno la gamificación se traduce en brindar a los estudiantes una motivación adicional inmediata que permite conseguir alcanzar una tarea de aprendizaje a largo plazo. Además, dicho incentivo permite que la realización de una actividad por parte de los alumnos sea más gratificante [4].

El uso de la gamificación está demostrando un gran crecimiento y unos excelentes resultados en el ámbito educativo. El eje vertebrador de este auge se fundamenta en que los alumnos consiguen realizar una actividad por el disfrute del trabajo en sí mismo [5], el cual implica los siguientes beneficios:

- Aumenta la implicación y la motivación del alumno.
- Refuerza el sentido del trabajo desempeñado.
- Establece un compromiso con el logro de los resultados.
- Ofrece un refuerzo positivo.
- Proporciona una retroalimentación inmediata.

Un sencillo proceso de gamificación en el ámbito educativo superior es mostrar un ranking competitivo que permita comprobar a los alumnos cómo están desempeñando sus actividades respecto al resto de compañeros de la clase en una asignatura. El reconocimiento que supone ocupar los primeros puestos dentro del ranking supondría una motivación adicional.

Esta contribución se enfoca a la asignatura de Arquitectura de Computadores que se imparte en los estudios de Ingeniería en Informática y donde parte de ella se evalúa mediante una prueba objetiva de preguntas tipo test. La preparación de pruebas objetivas es vital para superar un cuantioso número de materias en la educación universitaria, ya que hoy en día hablar de evaluación, en muchas ocasiones, implica hablar de pruebas objetivas.

Son muchos los factores que marcan la equidad entre evaluación y pruebas objetivas, entre todos ellos destacan los siguientes:

- Cubrir un área mayor de conocimiento si están bien confeccionadas.
- Propiciar que el alumno se concentre exclusivamente en el contenido de la materia y no en aspectos como la redacción o la ortografía.
- Facilitar la corrección en un tiempo breve, sobre todo en el caso de grandes grupos.
- Eliminar el juicio subjetivo del docente.

Para la superación de una prueba objetiva, como se ha señalado previamente, un factor esencial es una buena preparación mediante la realización de pruebas de este tipo. A medida que el alumnado lleva a cabo una prueba, puede adquirir nuevo conocimiento, ya que se le proporciona una retroalimentación en tiempo real, es decir, lo que se denomina “autoentrenamiento” [6,7]. Conjuntamente, una vez finalizada la prueba,

el alumno mide el grado de adquisición de sus conocimientos, al que se le denomina "autoevaluación" [8].

Los procesos de autoentrenamiento y autoevaluación forman parte del aprendizaje autónomo del alumno, demandado en la educación universitaria por el actual Espacio Europeo de Educación Superior. Por tanto, una pieza clave en el aprendizaje autónomo es el conjunto de recursos de aprendizaje que el docente pone a disposición del alumnado. Dichos recursos deben de estar correctamente diseñados para que el alumnado pueda aprender de ellos y con ellos, sin la necesidad de la presencia o ayuda del docente [9].

Nuestro interés en este trabajo se enfoca en compartir la experiencia del uso de dinámicas y mecanismos de juego para incentivar el proceso de aprendizaje autónomo del alumno, a través de los procesos de autoentrenamiento y autoevaluación, y a la hora de preparar la prueba objetiva de la asignatura de Arquitectura de Computadores en los estudios de Ingeniería en Informática de la Universidad de Jaén. Además se presenta la opinión de los alumnos sobre este proceso que ha sido recabada a través de una encuesta de satisfacción. Los comentarios vertidos muestran una opinión, en general, muy favorable al uso de técnicas y dinámicas de juegos en la asignaturas con algunos puntos a mejorar que también son señalados.

Para llevar a cabo este trabajo, en primer lugar se describe el contexto de la asignatura donde se ha llevado a cabo la experiencia. Posteriormente, se describen las técnicas de gamificación que se han utilizado para los procesos de autoentrenamiento y autoevaluación junto con una evaluación por parte del alumnado implicado en la experiencia. Finalmente, se incluye una sección con las conclusiones y los trabajos futuros.

2 Las asignatura de Arquitectura de Computadores en la Universidad de Jaén

La asignatura donde se ha llevado a cabo la experiencia en el uso de técnicas de gamificación es la asignatura de Arquitectura de Computadores del Grado de Ingeniería en Informática impartida en la Escuela Politécnica Superior de Jaén en la Universidad de Jaén.

Dicha asignatura, junto a la asignatura de Programación y Administración de Redes, conforma la materia de Arquitectura de Computadores y Redes, la cual es común a la rama de informática. Arquitectura de Computadores se imparte en primer lugar, estando ubicada en tercer cuatrimestre, es decir, en el primer cuatrimestre del segundo curso. Mientras, la asignatura de Programación y Administración de redes se imparte en el cuarto cuatrimestre, es decir, en el segundo cuatrimestre del segundo curso. La carga total lectiva de la asignatura de Arquitectura de Computadores es de 6 créditos ECTS, distribuidos en 2,5 créditos ECTS para clases expositivas, 3 créditos ECTS para clases en pequeño grupo y, finalmente, 0,5 créditos ECTS para tutorías colectivas.

La importancia de la asignatura en el contexto de la titulación viene determinada por el deber de un graduado en Ingeniería en Informática de conocer los diferentes elementos de un computador, así como las distintas arquitecturas avanzadas que actualmente se utilizan. De este modo, se dota al graduado en Ingeniería en Informática de un punto de vista práctico esencial, que le permite conocer a fondo las características

y el funcionamiento del hardware, saber cómo mejorar el rendimiento de un sistema, y saber comparar dos arquitecturas distintas con el fin de identificar y escoger la mejor en función de las necesidades.

Las competencias de la asignatura de Arquitectura de Computadores son las siguientes:

- Conocimiento y aplicación de las características, funcionalidades y estructura de los Sistemas Distribuidos, las Redes de Computadores e Internet, y diseño e implementación de aplicaciones basadas en ellas.
- Conocimiento y aplicación de los principios fundamentales y técnicas básicas de la programación paralela, concurrente, distribuida y de tiempo real.
- Conocimiento, administración y mantenimiento de sistemas, servicios y aplicaciones informáticas.
- Capacidad de conocer, comprender y evaluar la estructura y arquitectura de los computadores, así como los componentes básicos que los conforman.

Así, los resultados de aprendizaje de la asignatura son los siguientes:

- Conocer y aplicar las características, funcionalidades y estructura de los Sistemas Distribuidos, las Redes de Computadores e Internet. Diseñar e implementar aplicaciones basadas en ellas.
- Conocer y aplicar los principios fundamentales y técnicas básicas de la programación paralela, concurrente, distribuida y de tiempo real.
- Ser capaz de conocer, administrar y mantener sistemas, servicios y aplicaciones informáticas.
- Conocer, comprender y evaluar la estructura y arquitectura de los computadores, así como los componentes básicos que los conforman.

Los contenidos básicos de la asignatura versan sobre la clasificación de arquitecturas paralelas, la evaluación de prestaciones, paralelismo a nivel de instrucción (procesadores segmentados y superescalares), computadores paralelos (multiprocesadores, multicores, procesadores multihebra y multicomputadores), optimización de código y, finalmente, programación paralela.

Respecto a los contenidos teóricos, en los cuales se ha llevado a cabo la experiencia del uso técnicas de gamificación, están desglosados en los siguientes módulos:

- Módulo 1: Introducción.
- Módulo 2: Segmentación de cauce.
- Módulo 3: Procesadores superescalares.
- Módulo 4: Procesadores VLIW.
- Módulo 5: Procesadores vectoriales.
- Módulo 6: Computadores paralelos, programación paralela, prestaciones y redes de interconexión.

Sobre las metodologías y actividades que son llevadas a cabo en la asignatura se incluyen:

- Clases magistrales expositivas donde se intercalan exposiciones de teoría y ejemplos generales de los contenidos de la asignatura. Además, se incluyen en las clases magistrales expositivas en gran grupo otras metodologías como debates y resolución de ejercicios.
- Clases en pequeño grupo realizadas en laboratorios y en aulas de informática del departamento de informática de la Universidad de Jaén donde se llevan a cabo resoluciones de ejercicios sobre los contenidos teóricos y actividades prácticas con las aplicaciones informáticas.
- Tutorías colectivas consistentes en la aclaración de dudas sobre los contenidos de la asignatura, tanto teóricos como prácticos, en las que se fomenta el debate entre los asistentes para la aclaración de las dudas, bajo la dirección del profesor.

La evaluación de la asignatura se sustenta en el siguiente sistema consistente en tres pilares:

- Dominio de los conocimientos teóricos y operativos de la materia, el cual es evaluado mediante un examen teórico consistente en una prueba objetiva de tipo test, cuestiones cortas teórico-prácticas y la resolución de ejercicios.
- Dominio sobre un tema relacionado con arquitectura de computadores, el cual es evaluado mediante la elaboración de un informe y su defensa.
- Dominio de los conocimientos prácticos de la materia, el cual es evaluado a través de la elaboración de guiones consistentes en cuestiones prácticas, desarrollo de programas, informe de resultados y su posterior defensa.

En esta contribución se expone el uso de técnicas de gamificación para incentivar la preparación de la prueba objetiva de la asignatura a través del proceso de aprendizaje autónomo del alumno mediante los procesos de autoentrenamiento y autoevaluación.

3 Gamificación en procesos de autoentrenamiento y autoevaluación. Experiencia en la asignatura de arquitectura de computadores

Durante el curso 2015/2016 en la asignatura de Arquitectura de Computadores de la Universidad de Jaén se han utilizado técnicas y dinámicas de juegos enfocadas a incentivar al alumnado en la preparación de la prueba objetiva del examen teórico.

En esta sección, se describe la experiencia. Para ello, primero se describe el análisis de recursos TIC que se ha realizado para llevar a cabo el proceso de gamificación (Sección 3.1). A continuación, se indican las técnicas que se han utilizado, las cuales hemos denominado recurso gamificador (Sección 3.2). Por último, se indica la evaluación del recurso gamificador por parte del alumnado implicado en la experiencia (Sección 3.3).

3.1 Elección del recurso TIC para llevar a cabo la experiencia

Al inicio de la experiencia, se realizó un estudio de las técnicas y dinámicas de gamificación más adecuadas en la educación superior, llevándose a cabo un análisis de plataformas y recursos de gamificación existentes.

En dicho análisis se estudió tanto plataformas gamificadoras existentes en la red como la posibilidad de adaptar las técnicas y dinámicas de gamificación a la plataforma de docencia virtual que brinda la Universidad de Jaén, ILIAS[1].

Sobre las plataformas analizadas, se estudiaron con mayor profundidad las siguientes, debido a su relevancia:

- **Gameonlab**[2], el cual contiene un conjunto de servicios sobre cómo utilizar el pensamiento de juego para aumentar la participación y compromiso en una organización, pudiéndose adaptar al proceso de autoevaluación y autoentrenamiento de un alumno.
- **Socrative**[3], se trata de un sistema de respuesta inteligente con el que el profesor puede lanzar preguntas, quizzes y juegos, a los que los alumnos pueden responder en tiempo real desde sus dispositivos, ya que funciona desde un móvil, desde una tableta, desde un PC o un portátil con conexión a Internet.
- **Captainup**[4], se utiliza con el objetivo de aprender sobre gamificación jugando. Este sitio permite tener una experiencia de gamificación desde el punto de vista del individuo gamificado.
- **Openbadges**[5], es una iniciativa gratuita de la Fundación Mozilla. Su propósito es reconocer mediante insignias o medallas digitales los aprendizajes adquiridos por una persona en distintos momentos y situaciones tanto formales como informales.

Fruto de dicho análisis se detectó que la capacidad de exportación de preguntas tipo de test desde la plataforma que brinda la Universidad de Jaén, ILIAS, a las plataformas de gamificación existentes era muy limitada, siendo necesario crear nuevamente la base de datos de preguntas tipo de test en la plataforma de destino de gamificación, en el caso de que existiese esa posibilidad.

Debido a este gran obstáculo, se decidió adaptar los recursos de los que dispone ILIAS para la integración de técnicas y dinámicas de gamificación en la asignatura. Este hecho, además, brinda la posibilidad de integrar en un único espacio todos los materiales de la asignatura.

Es de destacar que la revisión realizada sobre las plataformas estudiadas nos permitió adquirir conocimientos sobre cómo aplicar las técnicas y dinámicas de gamificación en los procesos docentes de la asignatura.

3.2 Recurso Gamificador. Técnicas y mecanismos de dinámicas de juegos

Para llevar a cabo la experiencia, se creo un espacio denominado recurso gamificador integrado en la plataforma de docencia virtual ILIAS, la cual implementa las técnicas y dinámicas de gamificación.

La gamificación persigue el concepto de refuerzo positivo e implica que a partir de las reacciones a un determinado comportamiento, aumente la posibilidad de que el

[1] http://www.ilias.de
[2] http://www.gameonlab.es/canvas/
[3] http://www.socrative.com/
[4] https://captainup.com/
[5] http://openbadges.org/

comportamiento vuelva a ocurrir en el futuro. Para ello, en el recurso gamificador se han creado objetivos y recompensas personalizadas que implican que dicho comportamiento sea la motivación para alcanzar el objetivo.

En nuestro caso, el recurso gamificador cuenta con pruebas tipo test que se realizan a través de la plataforma y se nutren de un repositorio de preguntas tipo test de cada uno de los módulos que integran la asignatura. Dicho repositorio está creado desde el curso académico 2010/2011, actualizándose con nuevas preguntas cada curso académico.

El recurso gamificador con las técnicas y dinámicas de gamificación consiste en la realización de pruebas tipo test a través de la plataforma ILIAS a lo largo de la impartición de la asignatura, utilizando la base datos de preguntas, la cual está desglosada por módulos. Dicho recurso está basado en un sistema de puntuación, mecanismo muy popular y habitual, donde la participación o la finalización adecuada de pruebas tipo test implica obtener puntos, bonificaciones y reconocimientos en el escalafón de la asignatura. Para ello, de forma periódica en la asignatura se han llevado a cabo pruebas de tipo test competitivas que tiene una puntuación de 20 puntos cada una. La puntuación total obtenida puede verse incrementada por diferentes bonificaciones.

Para lograr el proceso de autoentrenamiento y autoevaluación, durante un intervalo de tiempo anterior a la prueba competitiva, están disponibles para el alumnado las pruebas tipo test en la plataforma ILIAS con el objetivo de que los alumnos puedan realizar dichas pruebas tantas veces como deseen, favoreciendo el proceso de autoentrenamiento. Dichas pruebas no puntúan y serán generadas aleatoriamente a partir del repositorio de preguntas tipo test contenido en el espacio virtual de la asignatura en ILIAS. Una vez finalizada cada una de las pruebas de entrenamiento, al alumno se le muestra la respuesta que ha seleccionado en cada pregunta y la respuesta correcta con el fin de que pueda realizar su autoevaluación.

En las pruebas competitivas de tipo test, se aplican logros y recompensas que consisten en mecanismos de juego en los que si se alcanza cierto nivel se pueden obtener bonificaciones que se traducen en la acumulación de puntos extra. En nuestro caso, los logros y recompensas que se establecen son los siguientes:

- Los alumnos que obtengan la mayor puntuación en la prueba competitiva obtendrán 200 puntos, a repartir entre los alumnos que obtengan dicha puntuación.
- Los alumnos que obtengan una puntuación mayor que 5 puntos en la prueba competitiva, obtendrán una puntuación extra de 15 puntos.
- Los alumnos que obtengan una puntuación mayor que 7 puntos en la prueba competitiva, obtendrán una puntuación extra de 30 puntos.
- Los alumnos que obtengan una puntuación mayor que 9 puntos en la prueba competitiva, obtendrán una puntuación extra de 50 puntos.

Respecto a las puntuaciones finales, los alumnos que hayan obtenido una puntuación media de las pruebas en los siguientes intervalos recibirán las siguientes bonificaciones, siempre que aprueben todas las pruebas competitivas:

- $[5,7)$: bonificación de 100 puntos.
- $[7,9)$: bonificación de 200 puntos.
- $[9,10)$: bonificación de 300 puntos.
- 10: bonificación de 400 puntos.

Después de cada prueba competitiva, se muestra una clasificación que promueve la competición entre los alumnos para incentivar su participación donde siempre se cuente con la satisfacción de verse en la zona alta de una tabla o, en el caso opuesto, evitar el último puesto de la tabla. Así, al finalizar cada prueba se publica dentro del recurso gamificador en ILIAS un archivo con el ranking de participantes con la puntuación obtenida en la última prueba competitiva realizada, la puntuación global (suma acumulativa de las diferentes pruebas competitivas más las bonificaciones que haya podido lograr), junto con la media de puntuaciones de las pruebas competitivas.

Con el objetivo de incentivar aún más el proceso de autoentrenamiento y autoevaluación, se establecieron como premios dispositivos electrónicos a los alumnos que obtuvieran una mayor puntuación final en la asignatura.

Además, se planificó sortear premios de consolación a los alumnos que completen todas las pruebas tipo test y la encuesta de evaluación del recurso gamificador. De este modo, los alumnos que ocupan la zona baja del ranking no se ven desmotivados al tener un aliciente adicional para continuar con el proceso de autoentrenamiento y autoevaluación. Dichos premios fueron financiados a través del proyecto de innovación docente de la Universidad de Jaén titulado "Gamificación para incentivar los procesos de auto-entrenamiento y auto-evaluación".

Cabe notar que el recurso gamificador está desarrollado únicamente para llevar a cabo los procesos de autoentrenamiento y autoevaluación, por lo que las pruebas tipo test competitivas realizadas en la plataforma no puntúan para la evaluación de la asignatura. Como se ha comentado, el recurso gamificador está enfocado para motivar el proceso de aprendizaje autónomo del alumno.

Para finalizar, se indica la distribución temporal de las pruebas tipo test competitivas que se llevaron a cabo. Cada una de las pruebas se realizó al menos una semana después de impartir el módulo correspondiente en las clases magistrales expositivas. Las pruebas se realizaron siempre a las 22 horas con el fin de que las pruebas no se solaparan con docencia reglada del grado de Ingeniería Informática, ya que el turno de tarde en la Universidad de Jaén finaliza a las 21:30 horas. Las pruebas se llevaron a cabo en los siguientes días:

- Módulo 1: 5 de octubre de 2015
- Módulo 2: 18 de noviembre de 2015
- Módulo 3 y 4: 15 de diciembre de 2015
- Módulo 5 y 6: 21 de diciembre de 2015

3.3 Evaluación del recurso gamificador por parte del alumnado

Al finalizar todos las pruebas tipo test competitivas, se llevó a cabo una encuesta de satisfacción consistente en 8 preguntas donde cada alumno debía indicar su grado de acuerdo o desacuerdo en la siguiente escala: *Totalmente en desacuerdo*-(1), *Desacuerdo*-(2), *Ni acuerdo ni desacuerdo*-(3), *De acuerdo*-(4) y *Totalmente de acuerdo*-(5).

Las 8 preguntas contenidas en la encuesta fueron las siguientes:

1. La gamificación te ha servido para comprender la materia.
2. La gamificación te ha servido para preparar la prueba objetiva del examen de teoría.

3. La gamificación te ha servido para aumentar el interés por la materia.
4. La existencia de premios consistentes en dispositivos electrónicos en la gamificación ha motivado tu participación en las pruebas competitivas.
5. La existencia de un ranking entre compañeros ha motivado tu participación.
6. La gamificación es un buen recurso didáctico.
7. La distribución de preguntas por módulos te ha parecido adecuada
8. La hora de las pruebas competitivas te ha parecido apropiada.

La encuesta fue completada por un total de 40 alumnos, obteniéndose los resultados de la Tabla 1.

Tabla 1. Resultados de la encuesta de satisfacción

Pregunta	1	2	3	4	5	Mediana
1	n=0 (0,00%)	n=0 (0,00%)	n=4 (9,76%)	n=28 (68,29%)	n=9 (21,95%)	4
2	n=0 (0,00%)	n=0 (0,00%)	n=5 (12,50%)	n=22 (55,00%)	n=13 (32,50%)	4
3	n=0 (0,00%)	n=5 (12,50%)	n=14 (35,00%)	n=13 (32,50%)	n=8 (20,00%)	3
4	n=2 (5,00%)	n=3 (7,50%)	n=10 (25,00%)	n=11 (27,50%)	n=14 (35,00%)	4
5	n=0 (0,00%)	n=3 (7,50%)	n=9 (22,50%)	n=16 (40,00%)	n=12 (30,00%)	4
6	n=1 (2,50%)	n=0 (0,00%)	n=3 (7,50%)	n=15 (37,50%)	n=21 (52,50%)	5
7	n=1 (2,50%)	n=1 (2,50%)	n=7 (17,50%)	n=17 (42,50%)	n=14 (35,00%)	4
8	n=10 (25,00%)	n=7 (17,50%)	n=5 (12,50%)	n=4 (10,00%)	n=14 (35,00%)	3

Además, en la encuesta a los alumnos se les brindó la posibilidad de realizar observaciones sobre la experiencia en el uso de técnicas de gamificación. Entre los comentarios vertidos, destacan los siguientes comentarios positivos de los alumnos sobre la experiencia:

- Ayuda realmente a llevar al día la asignatura, sobre todo en los aspectos teóricos de la misma, pues en la gamificación también se fomenta la competitividad entre los compañeros de clase motivando al alumnado para realizarla.
- La gamificación sí sirve para preparar el examen de teoría, pues en cierta medida te obliga a estudiar periódicamente el temario a modo de evaluación continua.
- Daros a los docentes de la asignatura la enhorabuena por la iniciativa a pesar de que haya que seguir puliendo dicha idea.
- Me parece una iniciativa genial y nos prepara bastante para el examen final viendo preguntas tipo sobre la materia.
- Me ha parecido bien organizado y bastante útil.
- Los pruebas competitivas han ayudado a ir estudiando día a día para poder participar en la competición. Es una muy buena forma de hacer que llevemos la asignatura más al día.
- Ha estado muy *guay* competir con los compañeros.

Entre los puntos a mejorar señalados por el alumnado destaca la hora de realización de la prueba tipo test competitiva que puntúa para el ranking, ya que se ha manifestado

un desacuerdo sobre la hora para llevar a cabo dicha prueba. En próximos cursos, a inicio de curso se consensuará con los alumnos la hora para llevar a cabo la prueba competitiva que puntúa para el ranking.

Otros comentarios relativos a la gamificación que podrán ser tenidos en cuenta en los próximos cursos académicos son los siguientes:

- Debería facilitarse un listado con las preguntas del repositorio para facilitar el estudio y así no tener que acceder a *ILIAS*.
- Cada prueba competitiva debería versar sobre un único módulo de teoría.
- Se deberían incluir más preguntas tipo test en cada prueba.

Como se ha comentado previamente, para incentivar aún más el proceso de autoentrenamiento y autoevaluación, se establecieron como premios dispositivos electrónicos a los alumnos que obtuvieran una mayor puntuación final en las pruebas competitivas, sorteándose, además, premios de consolación a los alumnos que habían llevado a cabo todas las pruebas tipo test competitivas y que habían completado la encuesta de satisfacción del recurso gamificador.

La Figura 1 muestra una fotografía del día que se hizo entrega de los premios a los galardonados y a los alumnos ganadores en el sorteo.

Figura 1. Alumnos galardonados en la gamificación con la profesora del grupo de teoría de la mañana.

4 Conclusiones y trabajos futuros

En esta contribución hemos presentado la experiencia de utilización de técnicas de gamificación en el entorno académico universitario. El fin último fue aumentar la motivación del alumnado para realizar el proceso de autoentrenamiento y autoevaluación para la preparación de la prueba objetiva en la asignatura de Arquitectura de Computadores del grado de Ingeniería en Informática de la Universidad de Jaén.

Durante el primer cuatrimestre del curso académico 2015/2016 se llevó a cabo con éxito la puesta en funcionamiento de las técnicas de gamificación. La evaluación de la utilización de técnicas de gamificación por parte del alumnado que ha cursado la asignatura ha sido muy satisfactoria. Nuestros trabajos futuros se encaminan a la incorporación de nuevas técnicas para aumentar el nivel de competitividad, considerando además la retroalimentación recibida por el alumnado durante el curso académico en que se puso en marcha la prueba piloto.

Agradecimientos

Esta publicación ha sido financiada por el proyecto de innovación docente de la Universidad de Jaén con código PID24_201416 titulado "Gamificación para incentivar los procesos de auto-entrenamiento y auto-evaluación".

Referencias

1. S. Deterding, D. Dixon, R. Khaled, and L. Nacke, "From game design elements to gamefulness: Defining gamification," pp. 9–15, 2011.
2. S. Deterding, K. O'Hara, M. Sicart, D. Dixon, and L. Nacke, "Gamification: Using game design elements in non-gaming contexts," pp. 2425–2428, 2011.
3. D. Dicheva, C. Dichev, G. Agre, and G. Angelova, "Gamification in education: A systematic mapping study," *Educational Technology and Society*, vol. 18, no. 3, pp. 75–88, 2015.
4. C. Nevin, A. Westfall, J. Martin Rodriguez, D. Dempsey, A. Cherrington, B. Roy, M. Patel, and J. Willig, "Gamification as a tool for enhancing graduate medical education," *Postgraduate Medical Journal*, vol. 90, no. 1070, pp. 685–693, 2014.
5. L. Moccozet, C. Tardy, W. Opprecht, and M. Leonard, "Gamification-based assessment of group work," pp. 171–179, 2013.
6. E. Martin, "Conclusiones: Un currículo para desarrollar la autonomía del estudiante," *La universidad ante la nueva cultura educativa. Enseñar y aprender para la autonomía*, pp. 285–292, 2003.
7. J. Pozo and C. Monereo, "Introducción. un currículo para aprender. profesores, alumnos y contenidos ante el aprendizaje estratégico," *El aprendizaje estratégico. Enseñar a aprender desde el currículo*, pp. 11–25, 2002.
8. A. Garcia-Beltran, R. Martinez, J. Jaen, and S. Tapia, "Self-assessment in virtual teaching and learning environments," *Revista de Educación a Distancia*, 2006.
9. G. McNamara and J. O'Hara, "The importance of the concept of self-evaluation in the changing landscape of education policy," *Studies in Educational Evaluation*, vol. 34, no. 3, pp. 173–179, 2008.

Utilización de la metodología de aula invertida en una asignatura de Fundamentos de Informática

Begoña del Pino, Beatriz Prieto, Alberto Prieto, Francisco Illeras

Departamento de Arquitectura y Tecnología de Computadores
Universidad de Granada
{bpino, beap, aprieto, filleras}@ugr.es

Resumen. En este trabajo presentamos la experiencia llevada a cabo durante el primer año de desarrollo de un proyecto de innovación docente financiado por la Universidad de Granada, en el que se ha aplicado la metodología de aula invertida utilizando los recursos propios de un MOOC (*Massive Open Online Course*), tales como videoclases, cuestionarios de auto-evaluación y foros de debate. Tras describir brevemente la plataforma tecnológica y la metodología docente, mostramos los principales resultados obtenidos de la encuesta de opinión realizada por los estudiantes y la mejora alcanzada en las calificaciones académicas con respecto a cursos anteriores. Finalmente, exponemos algunas conclusiones con la finalidad de seguir mejorando, no sólo los resultados académicos sino también el grado de satisfacción de estudiantes y profesores al utilizar este método.

Palabras Clave: Aula invertida, Clase invertida, Aprendizaje invertido, Lecciones aprendidas, Docencia aprendida, Enseñanza semi-presencial, MOOC, MOODLE, Fundamentos de Informática.

Abstract. In this paper, we present the experience carried out during the first year of a teaching innovation project supported by the University of Granada. The aim of the project is to apply the Flipped Classroom methodology taking advantage of the learning resources included in a MOOC (Massive Open Online Course), such as video-lectures, self-assessment tests and discussion forums,. After a brief description of both the technological platform and the learning methodology, we show the main results obtained from a survey of students and the academic outcomes. Finally, we draw several conclusions in order to continue improving not only the academic results, but also the satisfaction degree of students and teachers with the application of the method.

Keywords: Flipped Classroom, Flipped Learning, Blended Learning, Lessons Learned, MOOC, MOODLE, Fundamentals of Computer Systems.

1 Introducción

El método de aula invertida (*Flipped Classroom*) es un sistema de aprendizaje en que el estudiante debe haber estudiado la materia correspondiente con antelación a la clase presencial mediante vídeos en los que se exponen los diferentes conceptos. Después asiste a la clase para aclarar dudas, relacionar y reforzar conceptos, y realizar ejercicios prácticos [1,2,3]. Se invierte, por tanto, la dinámica respecto a la metodología más tradicional en la que el profesor expone la materia en clase, y con posterioridad el estudiante estudia el contenido y realiza ejercicios y tareas en casa como complemento a su estudio.

La metodología del aula invertida es una de las técnicas de enseñanza y aprendizaje con mayor proyección de futuro [4, 5]. Se relaciona con los conceptos de docencia aprendida (*Lessons Learned)* y enseñanza semi-presencial (*Blending learning* [6]). El estudiante debe haber estudiado previamente la materia, y lo hace por medio de materiales disponibles en línea.

El éxito de esta metodología depende obviamente de la calidad del material disponible en internet, y también de la apropiada organización de las clases presenciales en las que el profesor debe detectar las dificultades en el aprendizaje previo de los estudiantes, y utilizar los recursos idóneos para corregir los errores de comprensión que se hayan producido y, en definitiva, guiar el aprendizaje de un grupo de estudiantes que puede ser muy numeroso y heterogéneo, fomentando la interacción estudiante-profesor y procurando una atención personalizada [7]. No obstante, hay otros factores que también influyen en el rendimiento de los estudiantes durante el curso, como por ejemplo su grado de aceptación hacia una metodología que exige una dedicación constante para poder participar y seguir con aprovechamiento las clases.

Durante este curso académico 2015-2016 se ha aplicado esta metodología en la asignatura "Fundamentos de Informática" (Tabla 1) del primer curso de las Titulaciones de Grado de Ingeniería en Tecnologías de Telecomunicación (105 estudiantes matriculados) y Grado de Ingeniería Electrónica Industrial (74 estudiantes) de la Universidad de Granada. Los estudiantes de estas asignaturas han utilizado en línea el material disponible en un MOOC de la plataforma *abiertaUGR* [8], en el que además han participado de forma no presencial otros 116 estudiantes de diferente procedencia.

2 Antecedentes

Con objeto de mejorar los resultados académicos, en el curso pasado 2014-2015 se realizaron vídeos a partir de las presentaciones que se venían utilizando en clases presenciales expositivas más tradicionales, con la idea de que este medio resultase más atractivo a los estudiantes, acostumbrados a desenvolverse cotidianamente en un mundo digital. Se les propuso visionar las lecciones con antelación a las clases presenciales, con el propósito de disponer también de más tiempo para realizar problemas en el aula y resolver dudas. Además, fue posible realizar 5 tests de evaluación al finalizar los diferentes temas teóricos durante las propias clases

presenciales (en cursos anteriores se realizaba un único test de teoría al final del cuatrimestre).

En enero de 2015 se habían publicado 37 videoclases en Youtube, que en el momento de redactar este artículo han superado las 70.000 visualizaciones con una duración total de 477.692 minutos. Si bien el 57% de las visitas proceden de España, el 43% restante procede de otros 84 países.

Con esta experiencia mejoraron las calificaciones obtenidas por los estudiantes con respecto a cursos anteriores, y además, ante el interés suscitado por los vídeos del curso en la web, surgió la idea de diseñar un MOOC complementando las videoclases con otros materiales docentes para el aprendizaje, la auto-evaluación y la tutorización de los participantes.

Tabla 1. Descripción resumida de la asignatura "Fundamentos de Informática" de acuerdo con la guía docente.

Breve descripción de contenidos (según memoria de verificación de grado):
Estructura funcional de los ordenadores. Concepto y uso de Sistema Operativo. Concepto y uso de Base de Datos. Elementos de programación. Herramientas informáticas con aplicación en Ingeniería.Trabajo aislado de los alumnos.
Objetivos (expresados como resultados esperables de la enseñanza):
• Entender el significado global de la Informática. • Comprender cómo se representa la información en el interior de un computador. • Conocer la estructura funcional de un computador. • Comprender el concepto de programación y enumerar sus principales características. • Comprender el funcionamiento de un computador a nivel de lenguaje máquina y lenguaje ensamblador. • Conocer los fundamentos de los traductores: compiladores e intérpretes. • Analizar la funcionalidad de un sistema operativo en cuanto a la gestión de procesos, gestión de memoria, gestión de entradas/salidas y gestión de archivos. • Manejar adecuadamente los sistemas operativos más comunes en la actualidad. • Entender el concepto de base de datos. • Diseñar bases de datos relacionales sencillas e implementarlas en un sistema gestor de bases de datos. • Conocer y aplicar herramientas informáticas específicas de las ingenierías, como aplicaciones para cálculo matemático, representación científica de información y simulación de sistemas.

3 Plataforma tecnológica

Los recursos docentes que utilizan los estudiantes de las asignaturas de "Fundamentos de Informática" se encuentran en un MOOC (descrito con detalle en

[9]) en la plataforma de formación abierta online de nuestra universidad. Se proporcionan diversos materiales docentes estructurados por temas de teoría, encuestas de opinión y estrategias de gamificación basadas en la obtención de insignias para incentivar la participación y seguimiento durante el cuatrimestre.

Para cada tema (Figura 1) se dispone de las videoclases junto con las correspondientes presentaciones en texto, relaciones de problemas propuestos y resueltos, algunas tareas complementarias destinadas a la búsqueda de información actual en relación con cuestiones de particular interés en la asignatura, tests de autoevaluación y un foro de debate, en el los estudiantes plantean y resuelven sus propias dudas bajo la supervisión de los profesores.

Asimismo, se han realizado tres encuestas sobre la situación y conocimientos previos del estudiante, el grado de satisfacción con la metodología docente seguida en la asignatura, y sobre la labor docente del profesorado.

Se han concedido insignias parciales ligadas fundamentalmente a la realización de actividades de cada tema (exigiendo que se hayan superado los tests de autoevaluación con una calificación mínima de 8 puntos sobre 10) y una insignia final el seguimiento del curso completo, que nos proponemos registrar en el sistema *Mozilla Open Badges*.[1]

4 Metodología docente

La metodología seguida en la asignatura se basa en las siguientes acciones formativas:

- Videoclases expositivas de la materia
- Clases presenciales de debate y resolución de problemas
- Clases prácticas y seminarios
- Tutorías

Los contenidos teóricos correspondientes a cada tema se transmiten a los estudiantes de forma sistemática mediante la visualización de las lecciones en vídeo, fomentando de este modo, su capacidad para estudiar y aprender nuevos conceptos de forma autónoma. Después, en las clases presenciales el profesor plantea ciertas cuestiones a través de las cuales descubre las posibles dudas y dificultades, refuerza e interrelaciona los conceptos principales y propone la realización de problemas aplicados a casos reales. Se busca interpelar al estudiante, proponiéndole nuevos puntos de vista para que se forme en una mentalidad crítica.

En las clases prácticas y seminarios los estudiantes aplican los conocimientos teóricos mediante la instalación y uso de una distribución GNU-Linux, el diseño de bases de datos relacionales sencillas con un sistema gestor de bases de datos, el uso de herramientas informáticas con aplicación en la ingeniería (MATLAB), y la realización de ejercicios básicos de programación con lenguajes de alto nivel (MATLAB o C) y de programación en código máquina y ensamblador para un procesador didáctico elemental. Estas actividades se realizan de forma presencial en grupos reducidos (de 20-25 estudiantes, aproximadamente).

[1] https://support.mozilla.org/es/products/open-badges

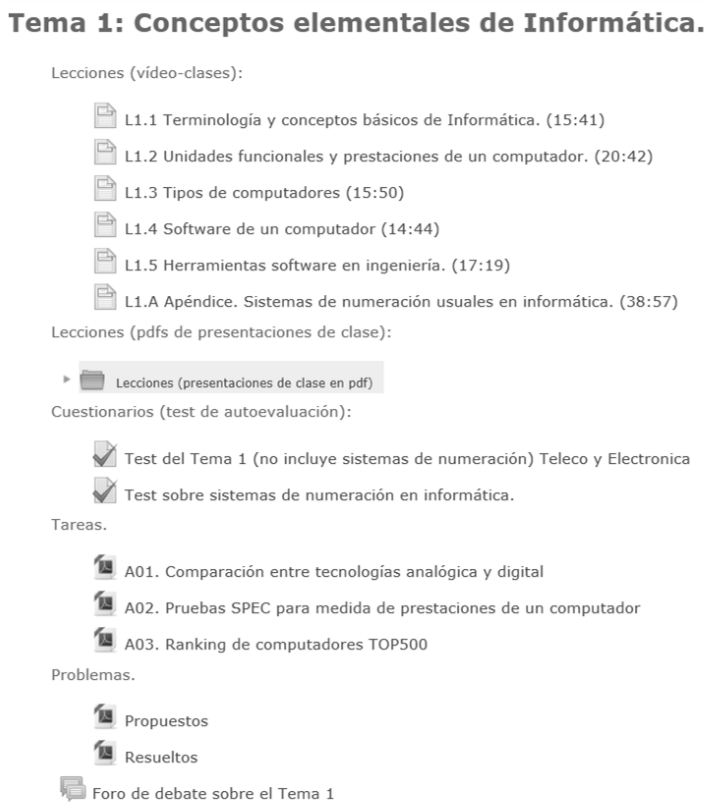


Figura 1. Ejemplo de materiales docentes proporcionados en el MOOC en relación con el Tema 1.

La tutorización se realiza de forma presencial en el despacho de los profesores en un horario establecido para atención a los estudiantes y también de forma no presencial, no sólo respondiendo dudas planteadas directamente al profesor, sino también supervisando los debates que se suscitan en los foros.

En este curso académico, el método de aula invertida sólo se ha empleado para las actividades relacionadas con los contenidos teóricos, no para las prácticas y seminarios. Cada semana el profesor indica por internet a los estudiantes de su grupo las videoclases que debe estudiar con antelación, junto con las referencias bibliográficas que puede consultar (fundamentalmente [10] y [11]) y algunos problemas propuestos en la plataforma que puedan estar relacionados. Las actividades se planifican para que el tiempo semanal de trabajo autónomo del estudiante no supere las 4 horas, de acuerdo con los créditos asignados a la asignatura en la guía docente.

Al finalizar los temas de teoría, se propone a los estudiantes que realicen los cuestionarios de test disponibles en la plataforma web para su auto-evaluación (de entre un total de 305 preguntas, se generan cuestionarios de 10 preguntas de respuesta múltiple ordenadas aleatoriamente), y se determina con antelación suficiente la fecha en que se realiza un cuestionario de test durante la clase presencial. Las preguntas en los tests presenciales tienen una dificultad similar a la de los tests de autoevaluación.

5 Resultados

A continuación resumimos los principales resultados en cuanto a la valoración de la metodología docente por parte de los estudiantes (según la encuesta de opinión disponible en la plataforma) y en cuanto a las calificaciones académicas obtenidas en la convocatoria ordinaria de este curso con respecto a cursos anteriores.

5.1 Seguimiento por parte de los estudiantes

Aunque desde el principio del cuatrimestre se hizo hincapié en que se debían dedicar al trabajo autónomo 4 horas semanales, tan sólo el 15% afirma haber dedicado en media entre 3 y 4 horas semanales al estudio de la asignatura. El 69% le ha dedicado entre 1 y 3 horas por semana, y un 6% más de 4 horas.

Por otro lado, un 15% manifiesta haber preparado entre el 60% y el 80% de las actividades propuestas semanalmente, y sólo un 12% afirma haber preparado más del 80% de las actividades.

5.2 Valoración de la metodología del aula invertida

El 63% considera que esta metodología le ha sido "bastante" o "muy" útil para comprender los conceptos, y el 59% opina que le ha sido "bastante" o "muy" útil para aprender a solucionar los ejercicios propuestos. El 74% afirma que la metodología fomenta el auto-aprendizaje "mucho" o "bastante".

Un porcentaje del 59% prefiere en general la metodología del aula invertida frente a las clases presenciales expositivas tradicionales. Al preguntar sobre las principales ventajas e inconvenientes del método del aula invertida se observa que los estudiantes aprecian la ventaja de poder acceder a los vídeos por internet en cualquier momento, y tantas veces como sea necesario repitiendo el visionado de las partes de mayor dificultad. Sin embargo, aparecen opiniones muy diversas en cuanto al papel de las clases presenciales. La mayor parte de las respuestas valoran positivamente que las clases presenciales sean más dinámicas y que se puedan realizar más ejercicios; otros en cambio cuestionan el sentido de asistir a clase: unos por considerar que ya se ha estudiado previamente la materia; y otros porque no pueden aprovechar el tiempo si por el contrario no se han preparado las tareas con antelación.

5.3 Valoración de los materiales y tecnologías disponibles en la plataforma online

Los recursos más utilizados y mejor valorados han sido los tests de autoevaluación (un 97% los ha utilizado "bastante" o "mucho", y el 91% los considera "bastante" o "muy" útiles), seguidos de las videoclases (utilizadas "mucho" o "bastante" por el 80%, y valoradas como "muy" o "bastante" útiles por el 69%). Cabe destacar que sólo el 52% dice haber utilizado "mucho" o "bastante" los foros de debate, mientras que el 65% los considera "bastante" o "muy" útiles para su aprendizaje.

5.4 Valoración de las clases presenciales

Las clases presenciales son consideradas "bastante" o "muy" útiles para comprender y afianzar los conceptos presentados en el material docente por el 65%, y para aprender a realizar ejercicios por el 71%.

5.5 Valoración del curso en general

El porcentaje de estudiantes que indican un grado de satisfacción "bastante" o "mucho" con la metodología del aula invertida es del 66%; con el material docente del 83%; con las clases presenciales del 66%; y con el curso en general del 79%.

5.6 Mejora en los resultados académicos

En los últimos tres cursos académicos el porcentaje de estudiantes que aprueba la asignatura (con respecto al total de matriculados) ha pasado del 57,25% al 82,70%, y la nota media final ha pasado de 5,32±1,56 a 6,74±1,59. La nota final se ha obtenido como la suma ponderada de las notas obtenidas en los cuestionarios de test presenciales (30%), en el examen de problemas realizado al final del cuatrimestre (50%) y de las actividades propuestas en prácticas y seminarios (20%).

6 Conclusiones

Durante el presente curso académico 2015-2016 se ha aplicado la metodología de aula invertida en la asignatura de "Fundamentos de Informática" de las Titulaciones de Grado en Ingeniería Electrónica Industrial e Ingeniería en Tecnologías de Telecomunicación utilizando los recursos propios de un MOOC disponible en la plataforma de formación abierta online de la Universidad de Granada. Esta experiencia se ha realizado en el marco de un proyecto de innovación docente financiado por nuestra Universidad de dos años de duración. Presentamos a continuación las principales conclusiones a las que hemos llegado a la vista de los resultados obtenidos, con el fin de tratar de mejorar la experiencia de cara al próximo curso:

No se ha conseguido en general que, de forma constante durante el cuatrimestre, los estudiantes dediquen un tiempo de estudio semanal suficiente para realizar las tareas asignadas. Aparte de la capacidad de motivación en las clases presenciales, otros factores externos pueden contribuir a explicar esta situación.

En cada curso de las dos titulaciones involucradas en la experiencia, esta es la única asignatura que emplea el método del aula invertida. Por otro lado, los estudiantes son de primer curso y primer cuatrimestre por lo que aún están en fase de adaptación a sus nuevos estudios de educación superior. Además, se ha observado la interferencia de otras asignaturas en un descenso en la asistencia a clase o falta de preparación de la misma, coincidiendo con semanas en las que los estudiantes estaban convocados para realizar otras pruebas o exámenes en el marco de la evaluación continua.

El método resulta muy exigente tanto para el estudiante como para el profesor, que en ocasiones se encuentra en la disyuntiva de "presentar los conceptos principales" para contextualizar a quienes no han podido realizar las tareas semanales (a costa de aburrir a quienes han realizado un cumplimiento adecuado) o "respetar" lo establecido asumiendo que una parte de los estudiantes no van a poder participar adecuadamente en las actividades propuestas para la clase presencial. Una coordinación más estrecha entre asignaturas de un mismo curso podría ayudar a planificar las actividades semanales (para antes y durante la clase presencial) de una forma más flexible.

Aunque parece que las clases presenciales han cumplido en general con el objetivo de reforzar conceptos y ayudar a resolver ejercicios, será conveniente explorar nuevas técnicas que fomenten la participación interactiva de los estudiantes para aumentar su motivación y tratar así de que se sientan más comprometidos con el seguimiento continuado de la metodología.

En general, los recursos docentes que se han utilizado han sido bien valorados, lo que nos reafirma en que el uso de los recursos del MOOC ha sido un acierto, si bien en la pregunta de respuesta libre sobre inconvenientes de las videoclases, la mayoría coincide en señalar su excesiva duración, por lo que será necesario acortarlas para próximas ediciones.

A la vista de la notable mejora que se ha producido en las calificaciones obtenidas por los estudiantes al realizar pruebas de evaluación semejantes en estructura, contenido y dificultad a las de cursos académicos anteriores, podemos afirmar que utilizar el método de aula invertida junto con los recursos de un MOOC permite conjugar de forma muy positiva la enseñanza a distancia y la enseñanza presencial.

Agradecimientos. La experiencia descrita se desarrolla en el marco del Proyecto de Innovación Docente 15-82 financiado por la Unidad de Calidad, Innovación y Prospectiva de la Universidad de Granada. Nuestro agradecimiento también al Centro de Enseñanzas Virtuales de esta universidad (CEVUG) por su apoyo y asesoramiento.

Referencias

1. Alvarez, A.; Flipping the Classroom: Homework in Class, Lessons at Home. Education Digest: Essential Readings Condensed for Quick Review 77(8) 18-21, (2012).

1. Bergmann, J.; Overmyer, J.; Wilie, B.: The flipped class: Myths vs. reality. The Daily Riff, 1-4. (2011)
2. Tucker, T.: The flipped classroom. Education Next 12 (1), 2012.
3. Pearson. Ideas que inspiran. Cinco técnicas de enseñanza que están revolucionando las aulas. http://ideasqueinspiran.com/2015/10/08/cinco-tecnicas-de-ensenanza-que-estan-revolucionando-las-aulas/ 8 octubre (2015).
4. Rosenberg, T.: Turning Education Upside Down. New York Times.9 Octubre (2013).
5. Garrison, D.R.; Heather Kanuka. Blended learning: Uncovering its transformative potential in higher education. The internet and higher education 7.2: 95-105, (2004).
6. Kim, M.K..; Kim, S.M.; Khera, O.; Getman, J.: T The experience of three flipped classrooms in an urban university: an exploration of design principles. The Internet and Higher Education, Vol. 22, pp. 37-50, July (2014).
7. Curso MOOC "Fundamentos de Informática" en la plataforma *abiertaUGR*: https://abierta.ugr.es/course/view.php?id=10. (2016)
8. Prieto, A.; Prieto, B.; del Pino, B.; Illeras, F.: Sinergias entre MOOC y Flipped Classroom: una experiencia. Experiencia MOOC: Un enfoque hacia el aprendizaje digital, la creación de contenidos docentes y comunidades online; M. Gea (Edt.); Editorial Universidad de Granada; pp.133-154, (2016).
9. Prieto, A.; Beatriz Prieto, B.: Conceptos de Informática. Serie Schaum, McGraw-Hill (2005).
10. Prieto, A.; Lloris, A.; Torres, J.C.: Introducción a la Informática, 4ª Edición. McGraw-Hill. (2006).

Desarrollo de sistemas mecatrónicos de bajo coste para equipamiento experimental de prácticas docentes

G. Olivares, A.González, F. Gómez, M. Damas, A.Olivares.

Departamento de Arquitectura y Tecnología de Computadores. ETSI Informática y de Telecomunicación. Universidad de Granada
Granada, España
{gonzalo, mdamas, frgomez, aolivares}@ugr.es

Resumen. Este trabajo describe los objetivos y el estado de desarrollo actual de un conjunto de maquetas y sistemas mecatrónicos de muy bajo coste destinados al aprendizaje de las principales técnicas de control. Se está construyendo un laboratorio con material docente experimental realizado íntegramente por profesores y alumnos, y sin duda será muy útil para dar soporte a la mayoría de las asignaturas relacionadas con sistemas de control impartidas por el Área de Ingeniería de Sistemas y Automática y el Departamento de Arquitectura y Tecnología de Computadores de la Universidad de Granada. El trabajo realizado forma parte del proyecto de innovación docente: "Nuevas experiencias de control óptimo con sistemas mecatrónicos".

Palabras Claves: Control, Mecatrónica, Bajo Coste, Docencia Experimental.

Abstract. This paper describes the objectives and the current state of development of a set of low cost models and mechatronics systems for learning the main control techniques. The control laboratory is being making entirely by teachers and students, and will certainly be very useful to support most of the subjects related control systems, teached by the Engineering Systems and Automation Area and by the Department of Architecture and Technology Computer at the University of Granada. The work is part of teaching innovation project: "New experiences optimal control with mechatronic systems."

Keywords: Control, Mechatronics, Low-cost, Experimental Learning.

1 Introducción

Con motivo del proceso de implantación de asignaturas del Área de Ingeniería de Sistemas y Automática en los nuevos grados de ingeniería que se imparten en la Escuela Técnica Superior de Ingeniería Informática y de Telecomunicación y en la Facultad de Ciencias de la Universidad de Granada, se han venido preparando en los últimos años un conjunto de prácticas, basadas principalmente en herramientas gráficas de simulación por software. En ese sentido, se han realizando un conjunto de maquetas virtuales para prácticas de control secuencial con PLC (Controladores Lógicos Programables), con magníficos resultados docentes y gran aceptación por

parte de los alumnos [1,2,3]. Se optó por la simulación virtual principalmente por la facilidad de uso, el bajo mantenimiento y la robustez de la solución, pero además también por la falta de presupuesto y la necesidad de repartir la escasa dotación económica anual entre todas las asignaturas, que hubo que implantar de forma casi simultánea.

Además de la realización de prácticas virtuales simuladas que reducen la necesidad de disponer de un mayor número de maquetas "reales", se considera que es también muy importante disponer de una mayor variedad de dichas maquetas para poder realizar un mayor número de prácticas diferentes. A pesar de las dificultades presupuestarias, cada año se podrían haber adquirido uno o dos productos de empresas especializadas en material de prácticas. Sin embargo, se ha preferido ir preparando con los alumnos, mediante trabajos en grupos y trabajos de fin de Grado y de Master, el diseño inicial de un conjunto de sistemas de control de maquetas mecánicas con control y regulación digital, e ir construyendo poco a poco todo el equipamiento del laboratorio de Ingeniería de Sistemas, Automática y Mecatrónica.

En la mayoría de los centros docentes universitarios se establecen diversos puestos de prácticas diferentes e individuales, donde los alumnos tienen que ir rotando. Estas prácticas se realizan de forma secuencial, independientemente de la materia teórica que en ese momento se esté impartiendo. Esta metodología de organización del laboratorio tiene un coste inferior, ya que en este caso basta con comprar una unidad del material de cada práctica; sin embargo, sólo podría ser verdaderamente efectiva si todas las prácticas se impartiesen una vez finalizadas las clases teóricas, y no de forma simultánea, tal y como se plantea en la mayoría de las planificaciones docentes universitarias (al menos así ocurre en los centros de nuestra universidad donde impartimos docencia).

Por otro lado, sucede que casi siempre los trabajos experimentales propuestos por los fabricantes de material didáctico emplean nomenclaturas y material formativo distintos a los utilizados por el profesor en clases de teoría, y en la mayoría de las ocasiones emplean software propietario de simulación, adquisición de datos y de control. A veces es más complicado estudiar las herramientas y los manuales que suministran, que asimilar los propios fundamentos teóricos que se desean transmitir, lo que requiere también un trabajo extra de traducción y readaptación de los manuales y guiones de prácticas.

Nuestro enfoque es distinto. Pensamos que lo mejor es que el alumno aplique inmediatamente los conceptos que se le transmiten cada semana en la teoría. Para ello, si, por ejemplo, los grupos de prácticas son de 20 alumnos, será necesario disponer al menos de 10 unidades idénticas de material experimental para cada sesión. Evidentemente, la única forma de implantar este sistema, con las dificultades presupuestarias actuales, es abaratando al máximo los costes de las maquetas de control a utilizar, y realizando entre profesores y alumnos la totalidad del trabajo de preparación de material, herramientas de control y guiones de prácticas, para no tener que pagar ese valor añadido a empresas de material didáctico. Es un esfuerzo grande, requiere tiempo, pero pensamos que merece la pena, ya que además si los alumnos

intervienen directamente en el diseño y construcción de las maquetas que van a utilizar en sus prácticas, se van a involucrar totalmente y aprenderán mucho más.

El laboratorio que se está construyendo, está ya dando servicio a las siguientes asignaturas impartidas en la Universidad de Granada:

- Ingeniería de Sistemas (Grado en Ingeniería Electrónica Industrial)
- Mecatrónica y Sistemas Aeroespaciales (Master de Ciencia de Datos e Ingeniería de Computadores).
- Sistemas de Control (Grado en Ingeniería de Tecnologías de Telecomunicación)
- Automática (Grado en Ingeniería Electrónica Industrial)
- Informática Industrial (Grado en Ingeniería Informática)
- Controladores Lógicos Programables (Grado en Ingeniería Informática)

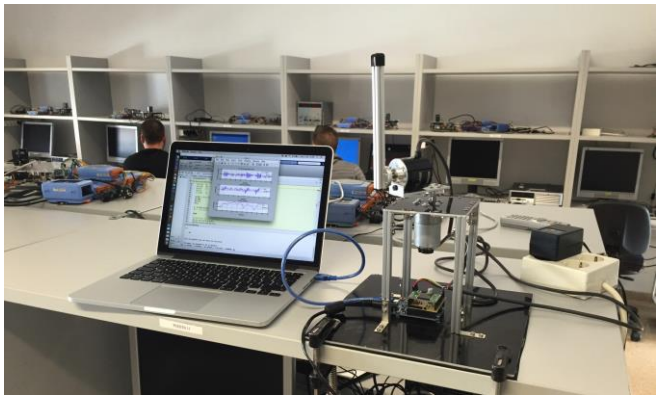


Figura 1. Fotografía parcial del Laboratorio de Control

2 Objetivos de implementación y de aprendizaje

A continuación se muestra una descripción de los sistemas que se están desarrollando y los objetivos de aprendizaje a conseguir con la utilización de los mismos.

2.1 Objetivos de implementación

En un principio nos hemos planteado la realización de un conjunto de maquetas de control, ya clásicas y habitualmente conocidas, cuyos modelos físicos (en forma de ecuaciones diferenciales, funciones de transferencia o en su representación en el espacio de estados) son fáciles de calcular, y de hecho dichos desarrollos se llevan a cabo en las clases teóricas. Presentamos a continuación una lista inicial:

- Control de posición (servo) y de velocidad angular de motor de corriente continua.
- Bola en barra.
- Péndulo invertido rotatorio (péndulo de Furuta).
- Péndulo invertido lineal en carrito.
- Mini-Segway.
- Bola en plano.

En la Figura 2 presentamos fotografías de algunos de los primeros prototipos ya realizados:

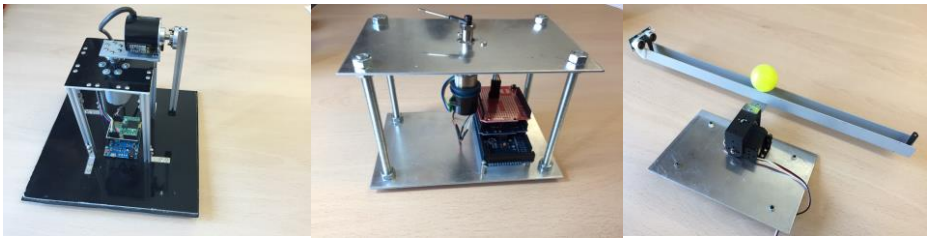


Figura 2. Fotografías de algunas de las maquetas realizadas (péndulo rotativo, servo-motor y bola en barra).

Más adelante continuaremos con más ideas de mecanismos de regulación y control (seguimiento de vehículos eléctricos, *segway* completo, micro-cuadricópteros, control de mini-trenes, etc.)

2.1 Objetivos de aprendizaje

Con este conjunto inicial de sistemas mecatrónicos didácticos, pensamos que es suficiente para que el estudiante comprenda y pueda utilizar las principales técnicas de modelado y de control básico [4] , tales como:

- Técnicas de obtención de modelos lineales de sistemas físicos mediante funciones de transferencia y representación en el espacio de estados.
- Control PID (Proporcional-Integral-Diferencial)
- Control digital por ubicación de polos y realimentación en el espacio de estados:
  - Configuración con ganancia previa
  - Configuración “servo”
  - Configuración con integrador
- Controlador-Observador.
- Control LQR (Regulador Cuadrático Lineal).
- Control LQG (Regulador Cuadrático Gaussiano).

Por otro lado, transversalmente el alumno adquiere conocimientos relacionados con:

- Procedimientos de medida y de cálculo numérico de multiples magnitudes (posición lineal, velocidad lineal, posición angular, velocidad angular, aceleración, detección de posición,…), utilizando sensores de bajo coste (codificadores (*encoders*) incrementales, ultrasonidos, hilo resistivo, sensores de infrarrojos, cámaras de video, acelerómetros, giróscopos, magnetómetros, pantallas resistivas, etc.) (ver Figura 3).
- Control sobre actuadores (motores de corriente continua y servos).
- Configuración, diseño y desarrollo de sistemas basados en microcontroladores.
- Programación en Matlab-Simulink .
- Programación en tiempo real en lenguaje C.
- Procesamiento digital de señales.
- Comunicaciones.
- Diseño CAD.

Figura 3. Algunos de los sensores, actuadores y tarjetas de control empleadas.

3 Metodología de desarrollo empleada

Considerando que el objetivo primordial es conseguir que el estudiante comprenda mejor los fundamentos de control mediante la práctica experimental, y teniendo en cuenta que antes de acudir al laboratorio, se supone que ya domina Matlab-Simulink como herramienta de simulación, pensamos que es ideal que pueda seguir usando la misma herramienta también para realizar el control en tiempo real de los sistemas mecatrónicos desarrollados. Afortunadamente Mathworks suministra un conjunto de modelos Simulink [5] para Arduino [6] (y también para Raspberry Pi [7]). Este grupo de modelos incluye: lectura y escritura digital y analógica, PWM y comunicaciones por puertos serie, Ethernet y Wifi. Para nuestras aplicaciones didácticas, necesitamos disponer también de bloques de lectura de sensores de distancia mediante

ultrasonidos, de *encoders* y de unidades inerciales (IMUs); para ello se deberán usar funciones-S específicas.

Para bajas frecuencias de muestreo y ejemplos sencillos se puede usar el modo "*External*", que permite la visualización de resultados en tiempo real mediante módulos tales como *Scope* y *Display*; pero en nuestro caso, se requieren periodos de muestreo del orden de T=0.01 seg, y esta opción no se puede utilizar. Es necesario usar el modo "*Deploy to hardware*" que transfiere totalmente el firmware a la tarjeta de control. Para visualizar los gráficos dinámicos con las variables (estados, salidas y señal de control) hay que incluir bloques de comunicaciones, y realizar después una aplicación Matlab para la lectura y visualización.

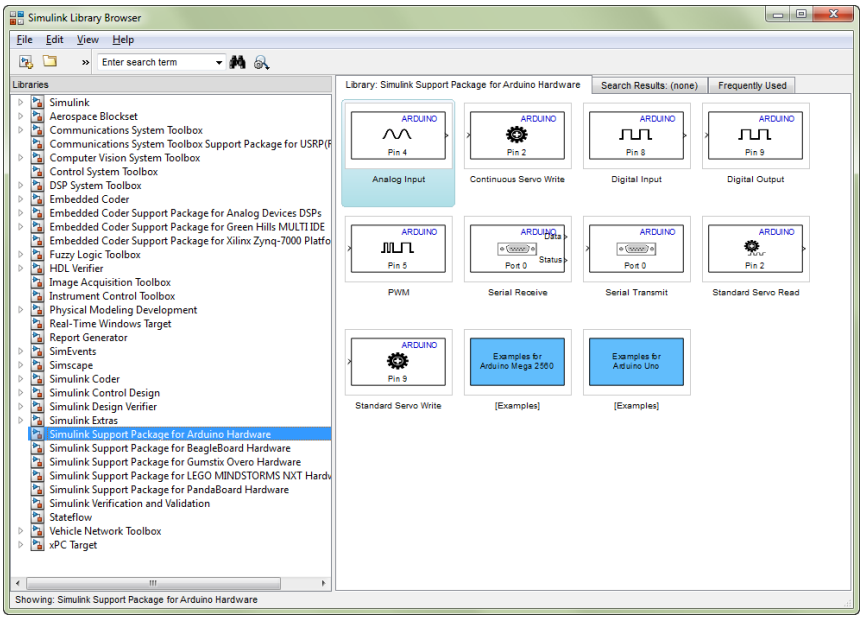


Fig. 4. Módulos de la Biblioteca Arduino para Simulink

El desarrollo de aplicaciones de control con este sistema es muy rápido; sin embargo, el alumno puede pensar que sigue realizando algo muy parecido a la simulación (aunque vea funcionar la maqueta) y es por ello, que pensamos que, alternativamente, es también conveniente que realice por sí mismo todo el software de control.

Se desarrolla también la aplicación directamente en lenguaje C, para que el alumno pueda ver también cómo se estructura una aplicación firmware en tiempo real, y aprenda también a manejar las interrupciones, realizar una derivada discreta en forma precisa (para obtener una velocidad angular a partir de la salida de un *encoder*, por ejemplo), a leer los pulsos de dicho codificador bajo interrupciones, a realizar un filtro digital con ecuaciones en diferencias, a linealizar la respuesta de control de un motor

DC (mediante una tabla de "*look up*"), a manejar la comunicación serie multivariable, etc. Es también muy interesante analizar el firmware en C que Simulink obtiene cuando se compila una aplicación con sus módulos gráficos. El estudiante puede así compararlo con el programa en C realizado por él.

Se resumen a continuación las dos configuraciones empleadas.

3.1 Configuraciones de control

En un principio proponemos estas dos configuraciones de diseño, control y visualización de resultados:

1. Desarrollo en C sobre Arduino o Raspberry Pi del software de medida y control, con comunicación (vía puerto serie USB, Bluetooth, Ethernet o Wifi) con aplicación realizada en Matlab. Se simulan con Matlab los resultados previstos para un diseño de control específico, se visualizan las variables simuladas de estado y de salida, se reciben en tiempo real las magnitudes de cada maqueta y se representan conjuntamente (ver Figura 5).

2. Desarrollo completo en Simulink del firmware de control, que se compila y transfiere totalmente al firmware de las tarjetas de control (ver Figura 6). Para ello se están utilizando la biblioteca de modelos ofrecida por Mathworks, tanto para Arduino, como para Raspberry Pi antes comentada.

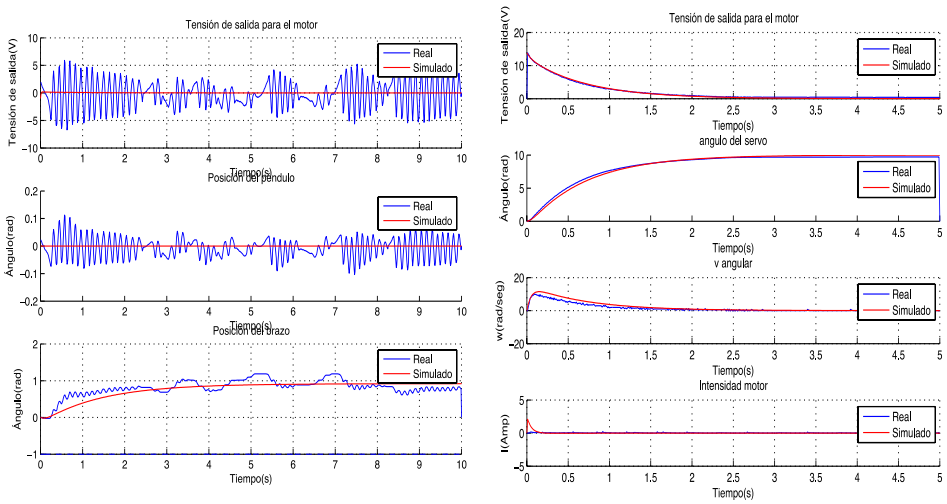


Figura 5. Resultados experimentales y simulados para péndulo invertido y servomotor, con configuración Matlab + firmware en C sobre tarjeta Arduino Mega.

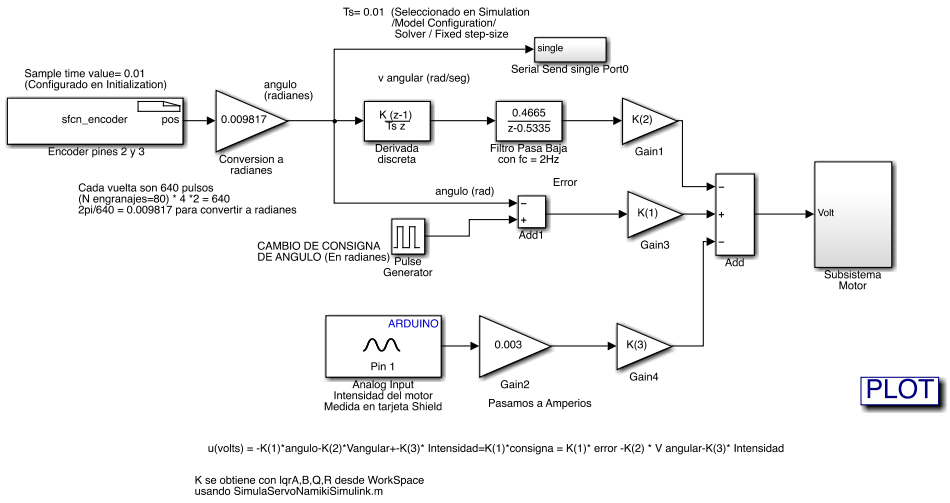


Figura 6. Diseño del control LQR de un servo motor, utilizando la biblioteca de Arduino para Simulink.

Más adelante se tiene previsto utilizar también Labview (National Instruments)[8] con tarjetas RIO, y comunicar los sistemas de control experimental con aplicaciones de móviles tanto para Android como para iOS. Esta solución permite usar los recursos de Labview para visualización en tiempo real, sin necesidad de usar módulos de comunicación a medida; sin embargo, en este caso el coste de las tarjetas de control es mucho mayor. No obstante, también puede conectarse Labview a tarjetas Arduino y Raspberry, y National Instrument permite que los alumnos utilicen Labview durante un año, sin coste alguno.

Actualmente estamos empezando a usar Raspberry Pi para proyectos de control que requieren procesamiento de imágenes, tal como el dispositivo “Bola en plano”. La posición de la bola en plano se puede medir en tiempo real con una pantalla resistiva, pero también se puede realizar con una sencilla cámara web, procesando digitalmente la imagen. Este procesamiento de puede realizar fácilmente utilizando los modelos apropiados de la biblioteca de procesamiento de imágenes de Simulink.

Hasta ahora, ninguna de las maquetas realizadas tiene un coste de material superior a 150 €, y la mayoría rondan los 50 €, sumando todos los elementos (sensores, actuadores, módulos de potencia, tarjetas de control, mecanización), siempre que ese material se compre directamente vía web a distribuidores de bajo coste, por lo que, sin duda pensamos que los objetivos iniciales se están cumpliendo.

4 Conclusiones

Se ha presentado en este artículo el procedimiento de desarrollo de un laboratorio de control de bajo coste. Las dos configuraciones empleadas: firmware en C con Matlab frente a Firmware con Simulink son complementarias. Está claro que es más fácil y rápido trabajar en un entorno gráfico, donde gran parte de los modelos ya están disponibles, pero si queremos que el alumno sepa exactamente como se realizan los algoritmos de control en tiempo real, es necesario al menos mostrar las aplicaciones realizadas en C, con la ventaja añadida de poder comparar los resultados simulados con los experimentales. El bajo coste de los materiales utilizados permite la fabricación a mayor escala de las maquetas de prácticas implementadas, por lo que, sin duda, a corto plazo los alumnos podrán disponer del material adecuado para simultanear las prácticas con la teoría.

Agradecimientos. Al Secretariado de Innovación Docente de la Universidad de Granada por los Proyectos de Innovación Docente concedidos que nos han permitido financiar la adquisición de algunas de las maquetas de laboratorio utilizadas en prácticas.

Referencias

1. M.Damas, G.Olivares, H.Pomares, A.Olivares: “Laboratorio Virtual basado en Web para el ciclo completo de desarrollo de sistemas de control industriales”. Workshop en Informática Industrial (WIIND), ISBN: 84-692-2381-9, Universidad Rey Juan Carlos, Madrid, 2009.
2. M.Damas, H.Pomares, G.Olivares: “Innovación en la docencia práctica de asignaturas relacionadas con la informática industrial”. 1ª Jornadas Andaluzas de Innovación Docente Universitaria. ISBN: 978-84-692-7263-3, Córdoba, 2009.
3. M.Damas, O.Baños, G.Olivares, F.Gómez: “Complementos para Informática Industrial del perfil de Ingeniería de Computadores del Grado de Informática de la UGR: Controladores Lógicos Programables”. Enseñanza y Aprendizaje de Ingeniería de Computadores. Revista de Experiencias Docentes en Ingeniería de Computadores. ISSN: 2173-8688, Nº3, pp.107-120, 2013.
4. Ogata,K. “Ingeniería de Control Moderna”. Ed.Prentice Hall. ISBN: 9788483226605, 2010.
5. Arduino support from Simulink. http://es.mathworks.com/hardware-support/arduino-simulink.html. Accedido el 4/04/2016.
6. Arduino. https://www.arduino.cc. Accedido el 11/04/2016.
7. Raspberry Pi. https://www.raspberrypi.org/. Accedido el 11/04/2016.
8. National Instruments. https://www.ni.com. Accedido el 11/04/2016.

Sistema Domótico Distribuido para Controlar el Riego y el Aire Acondicionado en el Hogar

Iván Manuel Laclaustra+, Jesús Martín Alonso+, Alberto A. del Barrio+, Guillermo Botella+

Departamento de Arquitectura de Computadores y Automática, Facultad de Informática de la Universidad Complutense de Madrid
Madrid, España
+ {ilaclaus, jesusm01, abarriog, gbotella} @ ucm.es

Resumen. Este trabajo presenta el proyecto *SEDomotics*, realizado en la asignatura Sistemas Empotrados Distribuidos, perteneciente a la titulación del Máster en Ingeniería Informática de la Universidad Complutense de Madrid. En este trabajo se describe e implementa una plataforma de control domótico de los sistemas de riego y aire acondicionado en el hogar, utilizando para ello dos placas Arduino y una Raspberry Pi como servidor. Además de capturar los datos en tiempo real, el sistema es capaz de almacenar un histórico con dichos datos.

Palabras Clave: Sistemas Empotrados Distribuidos, Internet of Things, Arduino, Raspberry Pi, Domótica, Sistema de riego.

Abstract. This work presents the *SEDomotics* project, developed in the Distributed Embedded Systems subject, allocated within the Computer Science Master, which is taught in the Universidad Complutense de Madrid. This work describes and implements a distributed domotic system for the irrigation and air conditioner at home, using for this two Arduino boards and a Raspberry Pi as a server. In addition to capturing real time data, the system is capable of storing a record with that data.

Keywords: Distributed Embbeded Systems, Internet of Things, Arduino, Raspberry Pi, Domotic, Irrigation System.

1 Introdución

Desde la llegada de Internet a nuestras vidas, vivimos en un mundo cada vez más interconectado. La tecnología nos ha proporcionado la capacidad de comunicarnos salvando grandes distancias y a una gran velocidad. El rápido desarrollo de las tecnologías móviles durante los últimos años ha hecho que las posibilidades aumenten

exponencialmente, y ha cambiado por completo nuestra forma de ver el mundo. Además, el hecho de que los costes de producción se estén reduciendo paulatinamente ha traído al frente un nuevo paradigma, el llamado Internet de las Cosas (IoT) [12-13], que está llamado a hacerse realidad en los años venideros.

En el paradigma del IoT, todos los objetos de uso común están conectados a Internet de forma distribuida [15,17], de manera que pueden proporcionar o recibir información relevante para su tarea. Podemos encontrar un ejemplo reciente en los SmartWatch que, además de ofrecer lo que un reloj convencional, pueden mostrarnos mensajes de nuestro móvil, ritmo cardíaco [14,16], llamadas entrantes, etc. Otra ventaja de este enfoque, es que los objetos podrían comunicarse y coordinarse entre sí. Aunque aún estamos muy lejos de un uso regularizado, desde hace algunos años se está trabajando en la *domótica* [20-21]. La domótica se encarga del control del hogar a distancia. Mediante una conexión a Internet, podríamos encender o apagar luces, televisores, altavoces, encender un hornillo en la cocina, etc. Estas ideas sobre IoT, domótica, smart devices, entre otras, son las que se estudian en la asignatura *Sistemas Empotrados Distribuidos* (SED), materia ubicada en el Máster de Ingeniería Informática de la Universidad Complutense de Madrid (UCM) [22], y en la cual se realizó este trabajo.

Una de las funcionalidades domóticas más extendidas es la referente al control del riego del jardín. Dado que el riego supone uno de los mayores gastos de agua en un hogar que lo posea, es importante mantener un uso óptimo del mismo, y para ello la domótica es ideal. Con ella, podríamos hacer cosas como programar un riego automático, abrir y cerrar válvulas o comprobar el estado del césped, con la ventaja de poder hacerlo a distancia. Sin embargo, el principal problema es que se hace necesaria la instalación de un equipamiento especial, que servirá de interfaz para que los objetos correspondientes se comuniquen satisfactoriamente. Esto hace que el precio de implantación se dispare. En el caso de los sistemas de riego, suelen rondar los 200€ [1] y el precio aumenta en el caso de querer un sistema domótico completo [2]. En dichos sistemas se ofrecen otras funcionalidades, como es el caso del control automatizado del aire acondicionado.

Debido al alto coste de los sistemas comerciales tradicionales, una buena solución es el uso de placas de bajo coste para la implementación. Por ejemplo, las placas Arduino [3] tienen un microcontrolador integrado con varios pines y otros componentes, todos programables mediante un lenguaje de alto nivel. Además, es posible conectar fácilmente multitud de sensores y otros sistemas. El lenguaje de programación utilizado es de uso común (C). Tanto las placas (a partir de 20€ las oficiales) como los sensores (unos 3€) tienen un precio asequible.

Por tanto, en este artículo se presenta un sistema domótico distribuido para controlar tanto el sistema de riego de una casa como automatizar el uso inteligente del aire acondicionado. En las siguientes secciones, mostraremos una implementación basada en Arduino [3] y Raspberry Pi [4].

El resto del artículo se organiza de la siguiente forma: la Sección 2 describe tanto el diseño del sistema como los detalles de la implementación del mismo. En la Sección 3 se muestran los resultados obtenidos y se describen los experimentos realizados para validar el sistema y finalmente en la Sección 4 se presentan nuestras conclusiones.

2 Propuesta

El sistema descrito en este artículo representa el control domótico de una casa, como muestra la Figura 1. La plataforma propuesta será capaz de controlar el sistema de riego y el del aire acondicionado, de forma que puedan encenderse/apagarse a distancia. Además incluirá un modo automático, de manera que los sistemas conectados se enciendan o apaguen en función del estado exterior (humedad/temperatura).

Por tanto, en el sistema propuesto se diferencian dos componentes principales:

- ***Subsistema exterior.*** Está conectado al sistema de riego de la vivienda. Se encarga de monitorizar los datos de temperatura exterior y de la humedad del suelo y en función de ellos activa/desactiva el sistema de riego según convenga.
- ***Subsistema interior.*** Está conectado al sistema de aire acondicionado de la vivienda. Se encarga de monitorizar los datos de la temperatura y humedad interior y en función de ellos activa/desactiva el sistema de aire acondicionado según convenga.

Además se plantea la creación de un centro de control, donde el usuario pueda ver en tiempo real los datos registrados por ambos subsistemas, a la vez que controlar los elementos que integran los subsistemas manualmente y consultar un histórico de los datos registrados por los sensores.

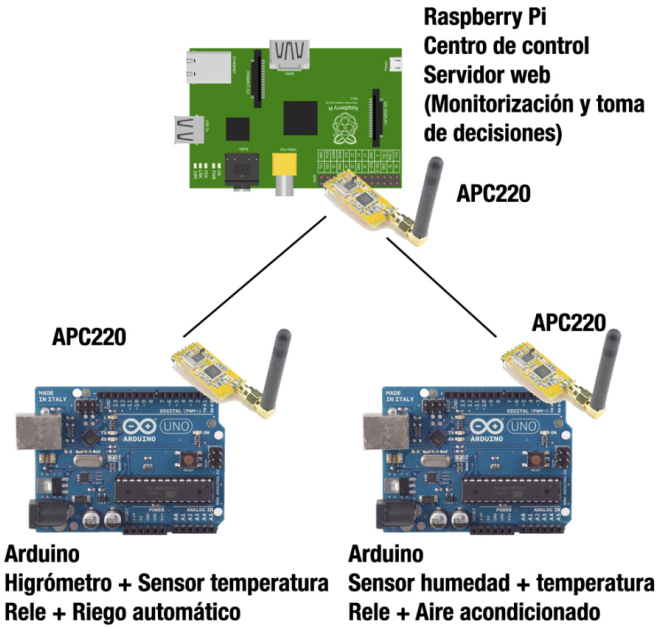


Figura 1. Esquema general del sistema domótico de control del riego y el aire acondicionado

2.1 Hardware empleado

Como muestra la Figura 1, ambos *subsistemas* se implementaron con placas Arduino Uno [3], ya que no requieren una alta potencia computacional. Por el contrario, para el '*centro de control'* se escogió una Raspberry Pi 1 Model B [4], que tiene un microprocesador lo suficientemente potente como para alojar el servidor web y la base de datos. Para completar el proyecto se escogieron los siguientes módulos *'plug & play'* para Arduino:

- Para la comunicación se escogieron módulos de radio APC220, por el bajo precio y su alcance (hasta 1000 metros).
- Un sensor de humedad y temperatura *DHT-11 Keyes* [24]. Este sensor se ha utilizado en el *subsistema interior* para monitorizar la temperatura y el grado de humedad, de tal forma que los datos sensados ayuden a tomar decisiones sobre el sistema de aire acondicionado de la casa.
- Un sensor de temperatura *Keyes DS18B20* [25]. Dicho sensor se ha utilizado en el *subsistema exterior* para monitorizar la temperatura.
- Un sensor de humedad del suelo *YL-69* [26]. Este sensor se ha utilizado en el *subsistema exterior* para monitorizar la humedad del suelo. A partir de los datos obtenidos se tomarán decisiones sobre el sistema de riego.
- Dos módulos de relés *Keyes* [27]. Los relés se han utilizado en ambos *subsistemas* para activar/desactivar los elementos asociados a ellos (aire acondicionado y riego).

2.2 Comunicación entre los componentes del Sistema

La comunicación por cable parece poco adecuada para un sistema domótico, ya que las distancias pueden ser elevadas y la instalación bastante costosa. De esta manera, nos decantamos por elegir comunicación inalámbrica. En concreto utilizamos los módulos de radiofrecuencia APC220 [5], ya indicados anteriormente. Decidimos utilizar radiofrecuencia debido a la simplicidad de uso (utilizando el puerto serie), a su bajo coste y al alto alcance que tienen los módulos APC220.

Estos módulos permiten la comunicación hasta una distancia de 1000 m (sin obstáculos) y pueden configurar su frecuencia (418-455 MHz), así como otras opciones.

La configuración de los módulos se realizó mediante el programa *RFMagic*, de acuerdo a los parámetros mostrados por la Tabla 1.

Tabla 1. Parametros de configuración de los módulos RF APC220, realizada mediante el programa 'RFMagic'

Frecuencia	433 MHz
Tasa de transmisión de datos	9600 bps
Bit de paridad	Desactivado

Estos módulos se conectan a los puertos serie de las placas Arduino, así como de la Raspberry Pi.

La Raspberry ejerce el control sobre los subsistemas, enviando las órdenes necesarias. De esta manera, cuando se necesitan los datos de los sensores o de los sistemas, la Raspberry envía una orden y las placas Arduino Uno la reciben y contestan. Por motivos de complejidad, se trata de un protocolo sin comprobación de errores. Es decir, no se comprueba que los datos lleguen correctamente o si las placas están apagadas. Por último, al ser un medio de difusión, en cada placa se comprueba si la trama recibida va destinada a ella. A continuación se detalla el formato de las tramas intercambiadas:

Tabla 2. Resumen del formato de las tramas intercambiadas

Trama	**Campos**
Trama de envío de información del sistema	*Identificador de placa*
	Temperatura leída por el sensor
	Humedad leída por el sensor
	Estado del relé conectado a la placa: *'on' / 'off'*
Trama de encendido/apagado de sistemas	*"relay"*
	Identificador de placa
	Estado del relé conectado a la placa:

	'on' / 'off'
Trama de petición de datos	*getData*
Trama de puesta en modo manual/automático	*manual*
	Modo al que se quiere cambiar *'ON' para activar modo manual* *'OFF' para activar modo automático*

- **Trama de envío de información del sistema**. Contiene la información del sistema. En el caso del subsistema interior enviará la temperatura y la humedad ambiente y en del subsistema exterior enviará la temperatura y la humedad del suelo. Consiste en una línea (terminada en \n) con campos separados por el carácter ';', como muestra la Tabla 2. El formato es el siguiente:

  1 ; temperatura ; humedadAmbiente/Suelo ; estadoSistema

  Ejemplo de trama:
  1;25;30;off

- **Trama de encendido/apagado de sistemas (Centro de control y subsistema interior/exterior).** El centro de control (Raspberry) envía esta trama cuando el usuario quiere encender/apagar los sistemas de forma manual. A su vez, la trama es enviada por parte de los subsistemas cuando se produce un cambio en el relé, bien debido a una orden del centro de control, o bien por decisión de la placa, en caso de que se encuentre en modo automático. Consiste en una línea (terminada en \n) con campos separados por el carácter ';', como puede observarse en la Tabla 2. El formato es el siguiente:

  relay ; identPlaca ; estadoSistema

  Ejemplo de trama:
  relay;2;on

- **Trama de petición de datos (Centro de control).** El centro de control (Raspberry) envía esta trama cuando necesita obtener los datos de los sensores y sistemas conectados a los Arduinos. Consiste en una línea (terminada en \n):

  getData

  La respuesta a esta trama es una trama de envío de información del sistema, descrita anteriormente.

- **Trama de puesta en modo manual/automático (Subsistema interior/exterior y Centro de control).** El centro de control (Raspberry) envía esta trama cuando el usuario cambia de modo automático a manual o viceversa. Por su parte, los subsistemas, al encenderse, envían esta petición (manual;ON), para que el resto (Raspberry y Arduinos restantes) se configuren en modo manual. De esta manera se evitan inconsistencias en el sistema. Consiste en una línea (terminada en \n) con campos separados por el carácter ';', tal y como muestra la Tabla 2. El formato es el siguiente:

  manual;orden

  Ejemplo de trama:
  manual;ON *(Activa el modo manual)*

2.3 Desarrollo de los subsistemas

Como se comenta en la Sección 2.2, la implementación de los subsistemas se realiza mediante el uso de placas Arduino Uno. A continuación se detalla su funcionamiento.

Al encenderse un subsistema, envía una petición para que el sistema entre en modo manual. De esta forma, conseguimos que no haya incoherencias entre las distintas placas. Por defecto, los relés están apagados. Para finalizar, tras configurar los sensores propios de cada subsistema, se envía una trama de información al controlador, para que éste actualice los datos en la página de control.

Una vez realizadas las configuraciones iniciales, ambas placas esperan la llegada de mensajes a través del puerto serie. Cuando se recibe un mensaje, cada placa comprueba si va dirigido a ella, y responde en consecuencia. Por otro lado, en caso de que el subsistema se encuentre en modo automático, llevará a cabo comprobaciones sobre la información de sus sensores, actuando en consecuencia. Por ejemplo, en el caso del subsistema exterior, el riego se activará automáticamente siempre que

estemos en modo automático y la humedad baje del 35%, y se desactivará si sube de 70%.

Para facilitar la implementación de los subsistemas, se usaron librerías externas en algunos de los sensores:

- **OneWire [6].** Esta librería sirve para comunicarse con dispositivos que utilicen el protocolo *1-wire*. En nuestro caso, el sensor de temperatura del subsistema exterior.
- **DallasTemperature [7].** Esta librería hace uso de la OneWire indicada anteriormente. Se utiliza para simplificar el uso de dicha librería, ya que es relativamente complejo.
- **DHT [8].** Esta es la librería proporcionada por el fabricante para utilizar el sensor de humedad+temperatura del subsistema interior.

2.4 Desarrollo del centro de control

Mediante el centro de control el usuario podrá ver toda la información de los subsistemas (sensores y estado de los sistemas) en tiempo real mediante una página web. Además podrá controlar los subsistemas manualmente desde la misma página.

Se decidió utilizar como servidor *Node.js* [9] debido a su gran variedad de módulos y su facilidad de programación (javascript). Para construir la página web se utilizó Express [10], un framework que facilita la construcción de webs en Node.js. Para la comunicación serie se instaló el módulo *serialport*, que permite utilizar el puerto serie. Para resolver el problema de la visualización en tiempo real se utilizó *Socket.io* [11], que permite enviar datos desde el servidor (Raspberry) al cliente (ordenador de usuario), de manera que los cambios se reflejen de manera inmediata. Por último para guardar los datos, y poder mostrar un histórico se desarrolló una base de datos MySQL [23].

Para mostrar un aspecto visual atractivo se utilizaron los siguientes librerías/componentes:

- Bootstrap para el aspecto visual de la web.
- Highcharts para las gráficas.
- Bootstrap switch para el modo manual.
- JustGage para los marcadores de temperatura/humedad.

El funcionamiento general del servidor es el siguiente:

- Cada 10 minutos se solicitan datos a los subsistemas para almacenarlos en la base de datos y construir el histórico.
- Cuando el usuario está en la página de monitorización se solicitan datos con un intervalo de 1 segundo, para que se refleje cualquier cambio de manera inmediata.
- Cuando el usuario está en modo manual y apaga/enciende un subsistema, se envía una trama al Arduino correspondiente para que actúe en consecuencia.
- Si recibimos una trama “manual;ON”, activamos el modo manual.
- Cuando el usuario está en la página histórico, se realizan consultas a la base de datos para construir gráficas.

3 Resultados y experimentos

En esta sección se proporciona el resultado de la implementación del sistema.

3.1 Implementación hardware

En la Figura 2 se puede apreciar el resultado del montaje del prototipo del subsistema exterior. En la Figura 3 se observa el resultado del montaje del prototipo del subsistema interior y finalmente, en la Figura 4, se observa el prototipo hardware del centro de control. Dichas figuras presentan los elementos anteriormente descritos.

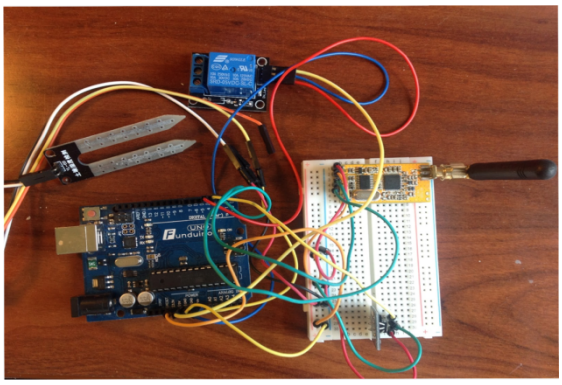


Figura 2. Prototipo del subsistema exterior

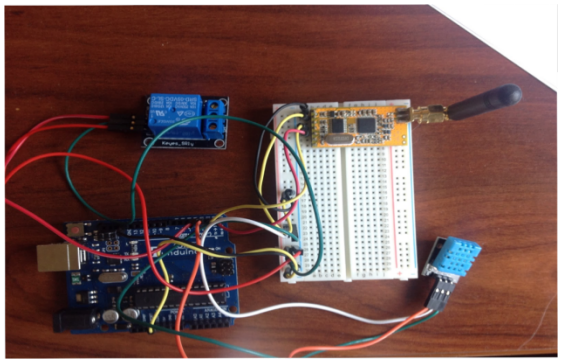


Figura 3. Prototipo del subsistema interior

Figura 4. Prototipo del centro de control

3.2 Implementación Software

En la Figura 5 se puede apreciar la página web que muestra la información de los sensores y el estado de los subsistemas (interior y exterior). En la Figura 6 se observa la página web que proporciona el histórico de los datos recogidos por los sensores. En dicha página se muestran cuatro gráficas: temperatura exterior, humedad del suelo, temperatura interior y humedad Interior. Por defecto, se muestran los datos recogidos en el último día. Sin embargo, el usuario puede elegir para cada gráfica mostrar los datos de la última semana, último mes, o todos los datos registrados.

Figura 5. Vista de la página web que muestra los datos correspondientes a los subsistemas en tiempo real

Figura 6. Vista de la página web que muestra el histórico de los datos recogidos por los sensores

3.3 Experimentos

Para validar el sistema se realizaron los siguientes tests:

- Comprobación del funcionamiento modo automático y de monitorización en tiempo real.

Como se ha indicado anteriormente, el sistema posee un *modo automático* en el que los subsistemas activan/desactivan los elementos que controlan en función de los datos recogidos por los sensores. Para comprobar el correcto funcionamiento se realizaron los siguientes pasos:

1. Cambiar sistema a modo automático.
2. Entrar en el apartado monitorización del servidor.
3. Comprobar subsistema interior. Se incrementó tanto la temperatura como la humedad y se comprobó que los valores recogidos cambiaban automáticamente en la página web, y de manera consistente. Además, al llegar al umbral de temperatura establecido se vio cómo se activaba automáticamente el relé que controla el sistema de aire acondicionado.
4. Comprobar subsistema exterior. Con el sensor de humedad de suelo introducido en tierra húmeda, se procedió a extraerlo poco a poco. Se pudo comprobar cómo los valores relativos a la humedad del suelo van

disminuyendo en la página web. De la misma manera, se observó cómo al sacar por completo el sensor de humedad del suelo, se activaba automáticamente el relé que controla el sistema de riego.

- Comprobación del modo manual.

El modo manual, permite controlar (activar/desactivar) el sistema de riego y el sistema de aire acondicionado manualmente, a través de la opción *control manual,* presente en la página web. Para comprobar su correcto funcionamiento se tomaron los siguientes pasos:

1. Entrar en la sección del servidor *control manual*, a través de la página web.
2. Cambiar el sistema a *modo manual*.
3. Activar y desactivar varias veces el sistema de riego, comprobando que se activaba/desactivaba el relé correspondiente de forma correcta.
4. Activar y desactivar varias veces el sistema de aire acondicionado, comprobando que se activaba/desactivaba el relé correspondiente de forma correcta.
5. Volver a poner el sistema en *modo automático*.

- Comprobación del histórico.

El sistema ofrece la posibilidad de observar un histórico de los datos recogidos por los sensores de los subsistemas mediante gráficas. Para comprobar el correcto funcionamiento de esta característica se realizaron los siguientes pasos:

1. Se dejaron activados los sistemas durante varias horas.
2. Se accedió al apartado *histórico* de la página web
3. Se comprobó en las gráficas que se habían registrado datos durante el periodo de tiempo en el que el sistema había estado activo.

4 Conclusiones

En este artículo se ha presentado un sistema domótico basado en una plataforma distribuida. Dicho sistema se realizó como proyecto para la asignatura de Sistemas Empotrados Distribuidos del Máster de Ingeniería Informática de la Universidad Complutense de Madrid.

Mediante el uso de plataformas de hardware como Arduino y Raspberry Pi, se implementó un sistema domótico inalámbrico y distribuido que permite el control de los sistemas de riego y de aire acondicionado del hogar.

De cara al futuro, se plantean varios puntos, como la elaboración de un segundo prototipo más avanzado que incluya alimentación por batería, la posibilidad de configurar los parámetros del sistema o la elaboración de una aplicación móvil que complemente a la página web elaborada.

Como conclusión final se ha de decir que este proyecto nos ha dado la oportunidad de aplicar en un caso real los conceptos vistos en clase, lo que incrementado nuestra motivación en el ámbito de los Sistemas Empotrados Distribuidos.

5 Trabajo futuro

Debido al limitado tiempo del que disponíamos, sólo pudimos realizar un prototipo limitado del sistema, que nos sirvió como prueba de concepto para sistemas mayores. Por ello, a continuación citamos algunas mejoras que nos hubiera gustado incluir de haber tenido más tiempo:

5.1 Ampliación de dispositivos

Además de los dispositivos de control tradicionales para riego y aire acondicionado, sería interesante añadir otros dispositivos, como por ejemplo:

- Estaciones meteorológicas: podrían monitorizar el tiempo para actuar automáticamente sobre los controladores de los demás dispositivos
- Cerrojos para puertas y ventanas: control a distancia de los cerrojos de la casa.
- Detectores de presencia: combinado con alarmas, podría cerrar todos los cerrojos de la casa en caso de saltar la alarma.

5.2 Reestructuración en módulos

Aunque nuestra implementación destaca por su sencillez, es necesario modificar el código en diferentes puntos a la hora de añadir más elementos. Esto hace que su ampliación o modificación sea un trabajo arduo para usuarios ajenos al sistema. Por ello, sería interesante reestructurar el sistema para hacerlo más modular, de forma que la adición de nuevos módulos sea más directa.

5.3 Configurabilidad

Actualmente, los dispositivos incluidos en el sistema tienen un funcionamiento muy concreto, que vino dado por nuestras especificaciones. Esto no tiene por qué satisfacer a todos los usuarios, por lo que sería interesante poder cambiar los parámetros del sistema (como porcentaje de humedad a partir del cual el riego para).

Otra mejora sería tener un mapa con los módulos que tenemos instalados en nuestro sistema desde la página web, pudiendo acceder a la configuración de cada uno

de ellos fácilmente, o desconectarlos por completo (junto con todos sus componentes).

5.4 Aplicaciones móviles

Dado que no se nos exigía portabilidad para dispositivos móviles, nuestro sistema no está preparado para ello. Debido a que hoy en día las tecnologías móviles están en pleno crecimiento, sería provechoso implementar una versión para móviles de nuestra aplicación, con todas las funcionalidades disponibles actualmente.

Referencias

1. Control de riego GreenIQ Smart Garden Hub, https://domboo.es/producto/control-de-riego-greeniq-smart-garden-hub/. Accedido el 14/03/2016.
2. Sistema domótico Control4, http://www.control4.com/o/new-entertainment-and-automation . Accedido el 14/03/2016.
3. "Arduino". https://www.arduino.cc/ . Accedido el 14/03/2016.
4. "Raspberry Pi". https://www.raspberrypi.org/ . Accedido el 14/03/2016.
5. "APC220 Radio Data Module wiki". http://www.dfrobot.com/wiki/index.php?title=APC220_Radio_Data_Module%28SKU:TEL0005%29 . Accedido el 14/03/2016.
6. "Documentación librería OneWire". http://playground.arduino.cc/Learning/OneWire . Accedido el 14/03/2016.
7. "Documentación librería Dallas Temperature". https://milesburton.com/Dallas_Temperature_Control_Library . Accedido el 14/04/2016
8. "Midiendo temperatura y humedad con Arduino y el sensor DHT11", http://www.internetdelascosas.cl/2014/07/08/midiendo-temperatura-y-humedad-con-arduino-y-el-sensor-dht11/ . Accedido el 14/03/2016.
9. "Node.js" https://nodejs.org/en/ .Accedido el 14/03/2016.
10. "Express" http://expressjs.com/es/ . Accedido el 14/03/2016.
11. "Socket.io" http://socket.io/ . Accedido el 14/03/2016.
12. O. Vermesan and P. Friess "Internet of Things - Global Technological and Societal Trends From Smart Environments and Spaces to Green ICT", 2011, River Publishers.
13. H. Chaouchi, "The Internet of Things: Connecting Objects", 2013, John Wiley & Sons.
14. R. Braojos *et al*., "Ultra-low power design of wearable cardiac monitoring systems," *Design Automation Conference (DAC), 2014 51st ACM/EDAC/IEEE*, San Francisco, CA, 2014, pp. 1-6.
15. A. A. Del Barrio, J. Cong and R. Hermida, "A Distributed Clustered Architecture to Tackle Delay Variations in Datapath Synthesis," in *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 35, no. 3, pp. 419-432, March 2016.
16. A. Dias Junior, S. Murali, F. Rincon and D. Atienza, "Estimation of Blood Pressure and Pulse Transit Time Using Your Smartphone," *Digital System Design (DSD), 2015 Euromicro Conference on*, Funchal, 2015, pp. 173-180.
17. A. A. Del Barrio, S. O. Memik, M. C. Molina, J. M. Mendias and R. Hermida, "A Distributed Controller for Managing Speculative Functional Units in High Level Synthesis," in *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 30, no. 3, pp. 350-363, March 2011.

18. D. Lora et al., “Sistema de Seguridad Basado en una Plataforma Heterogénea Distribuida“, Enseñanza y Aprendizaje de Ingeniería de Computadores, 5: 29-38 (2015).
19. F. Parrales et al. “Una Orquesta Sinfónica como Ejemplo de Aplicación de un Sistema Empotrado Distribuido”, Enseñanza y Aprendizaje de Ingeniería de Computadores, 5: 115-124 (2015).
20. A. Varma, ”Domotics: Smart Technology, Smarter Homes”, 2009, Icfai University Press.
21. R. Carbou et al., “Digital Home Networking”, 2013, John Wiley & Sons.
22. Máster en Ingeniería Informática de la Universidad Complutense de Madrid. http://informatica.ucm.es/estudios/master-ingenieriainformatica. Accedido el 10/05/2016
23. “MySQL”. https://www.mysql.com/. Accedido el 14/03/2016.
24. “DHT11 Humidity & Temperature Sensor” http://www.micropik.com/PDF/dht11.pdf. Accedido el 11/04/2016
25. “Keyes DS18b20 Sensor Module”, http://misclab.umeoce.maine.edu/boss/Arduino/bensguides/KeyesDs18b20_Digital_Temperature.pdf . Accedido el 11/04/2016
26. “Módulo YL-69 y YL38”. https://es.scribd.com/doc/233114065/Modulo-YL69-y-YL38. Accedido el 11/04/2016
27. “Keyes 5V Relay Module” http://tinkbox.ph/sites/tinkbox.ph/files/downloads/KEYES%205V%20Relay%20Module%20KY-019.pdf. Accedido el 11/04/2016

Diseño de la Maqueta Domótica para el Aprendizaje de Sistemas de Automatización Domótica

Juan A. Holgado-Terriza

Universidad de Granada
C/ Periodista Daniel Saucedo Aranda s/n 18071,
Granada, España,
jholgado@ugr.es

Resumen. La utilización de recursos didácticos que permitan a los alumnos disponer de herramientas que fomenten la innovación y la aplicación de los conocimientos adquiridos durante su formación, es una forma muy interesante de potenciar tanto su interés como su participación proactiva en el proceso de aprendizaje. En este trabajo se presenta la experiencia que ha adquirido el grupo de investigación de Sistemas Concurrentes de la Universidad de Granada en la construcción de maquetas domóticas como herramientas docentes que pueden favorecer el proceso de aprendizaje a los alumnos en carreras de carácter técnico, como son la ingeniería informática o la electrónica.

Palabras Clave: Docencia, domótica, maqueta, enseñanza, informática, electrónica.

Abstract. The use of teaching tools that allow students to have available tools which encourage innovation and the application of the knowledge acquired during their training, is a very interesting way to promote both their interests and their proactive participations in the learning process. In this work, we present the experience acquired by the research group of "Concurrent Systems" at the University of Granada in the construction of home automation scale models as teaching tools that can facilitate the active learning process for students in technical careers such as computer engineering or electronics.

Keywords: Teaching, home automation, scale model, computer engineering, electronics

1 Introducción

El aprendizaje en campos como la automática, control de procesos o la programación de sistemas requiere contar con herramientas docentes que faciliten la elaboración de diversos tipos de actividades que motiven y acerquen a los alumnos a problemas reales, haciéndoles protagonista del proceso de aprendizaje para que puedan encontrar sus posibles soluciones. Si además las soluciones pueden probarse sobre plataformas reales, esto permite a los alumnos comprender mejor la utilidad de los conocimientos adquiridos, y a su vez, adquirir competencias para su futuro profesional.

La domótica ofrece en la actualidad un campo de aplicación particularmente interesante desde el punto de vista tecnológico e industrial, ya que se están produciendo avances significativos desde los tradicionales sistemas de automatización del hogar en el que conviven dispositivos domóticos con servicios automáticos como los climatizadores, los detectores o los sistemas de vigilancia, a los más avanzados sistemas de inteligencia ambiental capaces de adaptarse y anticiparse a las necesidades del usuario de forma autónoma sin necesidad de ser programados; sistemas que, además son cada vez más accesibles desde cualquier tipo de terminal fijo, portátil o móvil.

En este contexto resulta particularmente propicio para la formación de los alumnos poder conjugar de forma óptima aspectos como el control industrial, las arquitecturas de telecomunicaciones, el diseño de nuevos dispositivos domóticos, la programación de sistemas o la aplicación de técnicas de inteligencia artificial. Por este motivo, es importante encontrar nuevas herramientas docentes donde pueda reproducirse en una escala manejable para los alumnos el entorno de una vivienda, y posibilitar el desarrollo de aplicaciones tal y como se podrían realizar sobre una instalación real.

Habitualmente se suelen emplear dos tipos de herramientas docentes como son los simuladores y las maquetas didácticas. Los simuladores proporcionan un entorno de trabajo controlado de la realidad en el que se pueden establecer distintos tipos de escenarios como casas adaptadas con el conjunto de dispositivos, sensores, actuadores o controladores específicos. Son sistemas fácilmente configurables y completamente accesibles al ejecutarse en el contexto de una aplicación que puede ejecutarse sobre diferentes tipos de plataformas software. Sin embargo, sólo son capaces de proporcionar modelos parciales de la realidad, no ofreciendo el grado de realismo suficiente para que la interacción con el alumno pueda ser rica y completa.

En cambio, las maquetas físicas didácticas a diferencia de los simuladores permiten desarrollar experiencias docentes más amplias y ricas para todos los actores del proceso de enseñanza/aprendizaje, fomentando el aprendizaje activo del alumno. En este sentido la maqueta domótica da la oportunidad de enfrentarse a los problemas prácticos de manejo de dispositivos reales, los problemas derivados del diseño, implementación y depuración de un sistema completo integrando los aspectos hardware y software, lo que proporciona una experiencia docente mucho más completa que aumenta la competencia de los alumnos para afrontar proyectos completos tal y como se encontrarán en su futuro laboral.

En este trabajo se presenta la experiencia que ha adquirido el grupo de investigación de Sistemas Concurrentes (TIC-157) del Departamento de Lenguajes y Sistemas Informáticos de la Universidad de Granada en la construcción de maquetas domóticas con fines docentes desde el año 2004 hasta hoy en día. Durante estos años se han construido hasta tres maquetas completas y se han realizado distintos tipos de adaptaciones tanto a nivel electrónico, físico, arquitectónico y lógico para adaptarse a distintos tipos de experiencias docentes. Hemos propuesto lo que denominamos la *Maqueta Domótica Didáctica* (MDD) como un modelo de diseño de construcción de maquetas domótica a escala, que determina la estructura de la maqueta tanto a nivel físico como a nivel lógico, y los pasos a realizar en el proceso de construcción.

Dado que el propósito de la maqueta es proporcionar un entorno donde el alumno pueda probar el desarrollo de aplicaciones a distintos niveles de abstracción proponemos la utilización de un lenguaje de alto nivel común como Java. Se ha

seleccionado Java como lenguaje de programación principal por las características específicas que tiene el lenguaje como la portabilidad, la capacidad de abstracción en objetos, la modularidad, el encapsulamiento y las facilidades para la programación como la ligadura dinámica, la carga dinámica de clases o el recolector de basura. Además permite utilizar el mismo lenguaje en cualquier ámbito de desarrollo, tanto a bajo nivel en el diseño de un driver como a alto nivel en el diseño de un servicio web. Por otra parte, contamos con herramientas de desarrollo muy estandarizadas que facilitan la implementación de las aplicaciones, su depuración y despliegue sobre diferentes tipos de entornos (microcontroladores, sistemas web, middleware, etc.) en el que se cargará un entorno de ejecución basado en una JVM (Java Virtual Machine) para la ejecución directa de las aplicaciones Java implementadas.

2 Diseño de la maqueta domótica didáctica (MDD)

En este trabajo proponemos un modelo de maqueta domótica aplicable al ámbito docente a partir de la experiencia que se ha ido adquiriendo en la construcción de maquetas. Dicha maqueta la denominaremos *maqueta domótica didáctica* (MDD), y consiste en un modelo a escala de una casa física real sobre la que se puede realizar diferentes tipos de actividades prácticas relacionadas con el área de la domótica. La maqueta proporciona una solución hardware-software que facilita al alumno la implementación, despliegue y depuración de aplicaciones que hacen uso de forma extensiva de los dispositivos domóticos presentes en una vivienda.

La MDD tiene características similares a una casa real para poder reproducir a una escala menor las condiciones del entorno real. Por tanto, al igual que en una instalación real es necesario distribuir una gran cantidad de componentes electrónicos por toda la casa para su correcto funcionamiento, así como la interconexión de dichos componentes electrónicos y el control de automatismos dentro del entorno de la casa. En nuestro caso, hemos incluido diferentes tipos de sensores, actuadores y dispositivos a escala para situarlos dentro de la MDD [1]. En la Tabla I se presenta algunos de los dispositivos que se incluyen en la MDD.

Desde un punto de vista topológico podemos distinguir tres niveles en la construcción de las maquetas, tal y como se muestra en la figura 1. En el nivel inferior se encuentra la capa de dispositivos que contiene todos los sensores, actuadores, reguladores y dispositivos que permiten controlar todos los automatismos de la casa. Se encuentran dispersos a lo largo de toda la maqueta, pero organizados en torno a un bus que facilita la identificación física de cada dispositivo. En este nivel, los alumnos pueden diseñar nuevas adaptaciones de dispositivos domóticos como, por ejemplo, nuevos controladores de persianas o un subsistema completo como un sistema de riego automático.

En el segundo nivel, se sitúa el controlador principal de la maqueta encargado del control hardware-software de todos los dispositivos instalados en la maqueta. Todos los elementos físicos se encuentran cableados y conectados al controlador principal a través de un bus. El controlador principal también dispone de los drivers y el código necesario para poder trabajar con los distintos dispositivos conectados. En este nivel los alumnos pueden desarrollar pequeños controladores o reguladores lógicos combinando la información procedente de uno o varios sensores con actuadores; por

ejemplo, se puede construir un sistema de seguridad pasiva activando todos los sensores de presencia de la maqueta, o un sistema de regulación día-noche para mantener el nivel de luz de una habitación.

Tabla 1. Dispositivos utilizados en las MDD construidas.

Funcionalidad	Tipo	Dispositivos
Iluminación	Actuador	Conjunto de bombillas de nivel de luz variable dispersas en cada habitación de la maqueta, así como en la parte exterior de la maqueta. Todas controladas independientemente.
Detector de iluminación	Sensor	Conjunto de sensores de iluminación en cada habitación, así como en el exterior para la detección día/noche.
Control de iluminación	Regulador	Se puede controlar la iluminación combinando los sensores de iluminación y la activación de las bombillas.
Detector ambientales	Sensor	Dispositivos que facilitan la detección de la temperatura interior/exterior, humedad, presión hidrostática, entre otros.
Control de temperatura	Regulador	Se combina un detector de temperatura con un climatizador para controlar la temperatura de una habitación. Se utiliza un sistema basado en calefactor o refrigerador, o una célula peltier y ventiladores cambiando la cara fría/caliente.
Detector de gas y humo	Sensor	Se incluyen varios sensores capaces de detectar la presencia de gases en la cocina para evitar posibles escapes de gas.
Control de persianas	Controlador	Las persianas pueden subirse o bajarse mediante un motor o servomotor, y uno o más sensores para controlar el estado de apertura de la persiana, o para saber si está completamente abierta o cerrada.
Alarma	Actuador	Se disponen un conjunto de zumbadores y sirenas que pueden activarse cuando sea necesario.
Control de presencia	Sensor	Se incluye sensores de presencia por infrarrojo distribuidos por las habitaciones para controlar si se pasa por una habitación u otra.
Apertura de Garaje	Controlador	Se incluye un dispositivo para controlar la apertura y cierre del garaje, y del estado de apertura del mismo.
Control energético	Controlador	Se incluye un dispositivo capaz de determinar la cantidad de corriente manejada. También se ha construido un sistema eficiente energéticamente mediante un sistema de batería para la acumulación de energía y un sistema de captación de energía solar.

Por último, en el tercer nivel se encuentran las aplicaciones interactivas que hacen uso de la maqueta, bien para interactuar con los dispositivos de la maqueta, o bien para recibir las notificaciones de los sucesos o eventos que están sucediendo en el interior o exterior de la maqueta. Dichas aplicaciones se ejecutan con un esquema cliente-servidor utilizando algún protocolo de comunicación, generalmente basados en TCP/IP como Ethernet y Wifi, aunque también es posible hacerlo sobre Bluetooth o Zigbee (o IEEE 802.15.4). Las aplicaciones interactivas desarrolladas pueden ser nativas basadas en las características de los clientes (Windows, Android, IOS, ...) o bien basadas en web utilizando navegadores web.

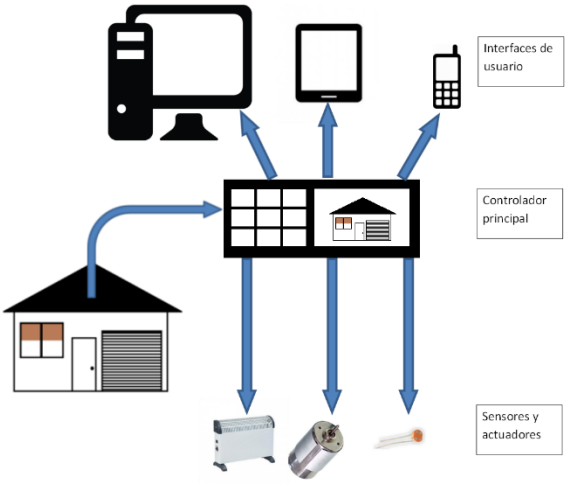


Figura 1. Representación topológica de la MDD

Para el diseño de la MDD se han seguido varios enfoques que dependen de la necesidad de construcción de toda la MDD, así como el conexionado de los distintos elementos. Concretamente, podemos distinguir tres alternativas principalmente:

a) Diseño de la MDD sobre un demostrador o prototipo.
b) Diseño de la MDD sobre una infraestructura parcialmente construida
c) Diseño de la MDD completa.

En el primer caso el diseño de la MDD se basa en la utilización de un demostrador o prototipo que incluye toda la infraestructura necesaria para desplegar los dispositivos domóticos así como el cableado necesario para interconectarlos entre sí. En este caso, el diseño de la MDD consistiría en construir una adaptación del entorno domótico si es posible, establecer el conexionado de los dispositivos con un microcontrolador o pc donde ejecutar las aplicaciones, y el desarrollo del software necesario. El demostrador o prototipo puede consistir en un panel con los controladores y los dispositivos empotrados en el panel, o bien un sistema de construcción en base a piezas individuales tipo LEGO que facilite la construcción del entorno y su sistema de conexionado.

En el segundo enfoque el diseño de la MDD se realiza en base a una infraestructura parcialmente construida. En este caso, podríamos utilizar, por ejemplo, una casa de muñecas o una maqueta de una casa, y adaptarla a las necesidades de la MDD añadiendo los dispositivos necesarios, controladores, sensores, actuadores y el conexionado necesario [2]. En la Figura 2(b) se puede observar el montaje que hemos realizado sobre la infraestructura de una casa de muñecas denominado CasaDomo.

Con el tercer enfoque se puede construir una MDD orientada a un propósito específico, por ejemplo, para cubrir la seguridad perimetral de una vivienda, inundando la maqueta de sensores de detección de presencia y cámaras, o trabajar más el aspecto de automática mediante la inclusión de numerosos dispositivos con un conjunto de reguladores [3]. En este caso, es necesario planificar el proceso de construcción de la infraestructura de la casa, el conjunto de dispositivos que se va a considerar, el sistema de conexionado, y los elementos que conforman la topología.

(a) Maqueta HAL9013 (b) CasaDomo

(c) DomoLab (d) DomoSOS

Figura 2. Diversas maquetas diseñadas siguiendo el enfoque MDD.

Siguiendo los enfoques mencionados anteriormente hemos construido y desarrollado cuatro tipos de maquetas:

- **Maqueta HAL9013** (2004-2005). Siguiendo el enfoque 3 se construyeron varias habitaciones equipadas con sistemas de control de iluminación, control de presencia, detectores de gas y humo, control de persianas y control de temperatura, entre otros [3]. El diseño primaba la facilidad para interconectar los distintos dispositivos, y por este motivo se situó el microcontrolador y el sistema de cableado en la parte superior de la maqueta. Sin embargo, era un sistema pesado y difícil de transportar para las prácticas.
- **Maqueta CasaDomo** (2008-2010). En este caso se buscó un diseño más flexible, fácil de transportar, y sobre el que utilizando un mismo tipo de conector se pudieran conectar diferentes tipos de dispositivos. Se utilizó un enfoque 2 para su construcción utilizando una casa de muñecas con paredes más flexibles y se centralizó toda la electrónica en la parte lateral de la maqueta [2].
- **Maqueta DomoLab** (2010-2012). Basándose en CasaDomo, la maqueta DomoLab plantea la posibilidad de acceder y controlar de forma remota a los dispositivos de la maqueta desde cualquier ordenador conectado a Internet mediante el desarrollo de un laboratorio remoto. Para ello, se tuvo que idear un sistema de carga de aplicaciones Java a distancia para poder ejecutar las aplicaciones que implementaban los alumnos en sus ordenadores, así como la sincronización de un conjunto de cámaras de video para mejorar la experiencia de interacción del alumno sobre la maqueta [4,5].

a) HAL9013

(b) DomoSOS

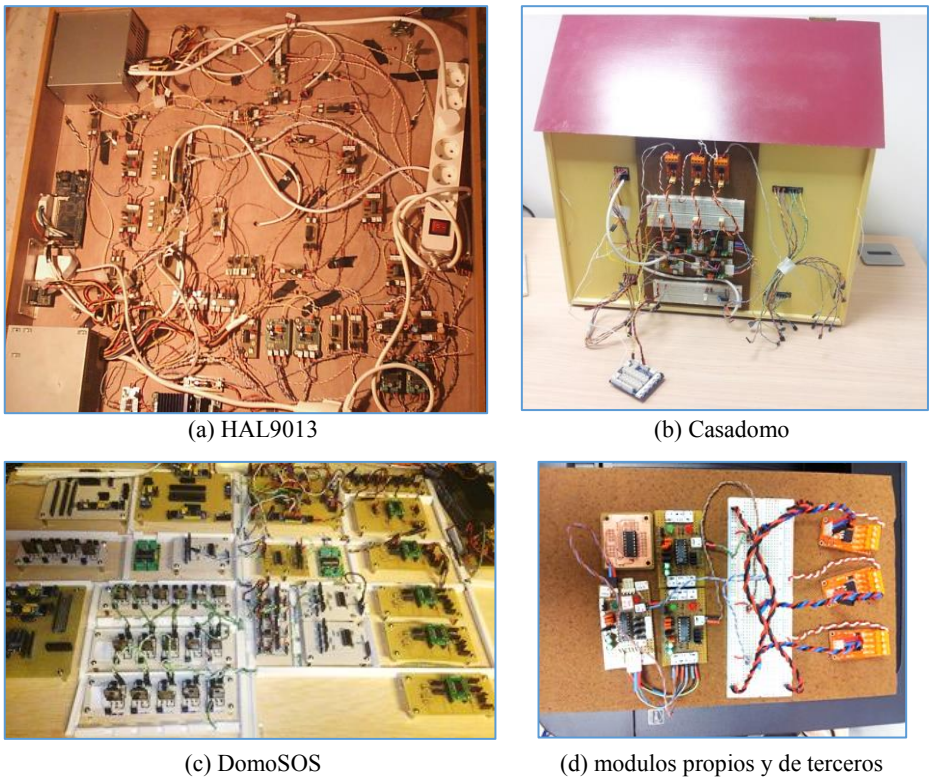
Figura 3. Estructuración de habitaciones en (a) HAL9013 y (b) DomoSOS.

- **Maqueta DomoSOS** (Sistema domótico sostenible) (2012-2013). En este caso, se propuso la construcción de una maqueta sostenible en dos sentidos. Por una parte, la maqueta incluye habitaciones reciclables que pueden cambiar en forma y extensión y, por otra parte, incluye distintos tipos de dispositivos domóticos intercambiables "plug and play" según las necesidades de la práctica. También se diseñó un sistema de gestión energética de todo el sistema, que además puede funcionar con energía solar [6].

3.1. Construcción de la maqueta y preparación de la electrónica.

La construcción de las distintas maquetas se ha realizado teniendo en cuenta los elementos del sistema de control domótico que queríamos automatizar en la casa, con la intención de obtener un modelo físico final ampliable y/o adaptable a nuevos usos futuros de modo que pueda ser utilizado como herramienta didáctica para fines docentes. Para ello se ha elaborado una maqueta fácilmente desmontable para facilitar las labores de mantenimiento y reparación, ocultando en la mayor medida de lo posible el cableado que pasa por las habitaciones y proporcionando la suficiente movilidad para que pueda ser desplazada sin excesivos problemas. En la Figura 3 se pueden ver algunas imágenes de los prototipos de maqueta HAL9013 y DomoSOS. En ambos casos se pueden extraer las paredes y adaptarlas según las necesidades, aunque en el caso de DomoSOS incluso las paredes de la parte exterior de la casa son reutilizables.

Del mismo modo, la electrónica y el cableado se han diseñado con cuidado realizando una planificación de todo el sistema eléctrico y electrónico de la maqueta (ver Figura 4), probando cada dispositivo electrónico (chip, sensor, actuador) antes de ser instalado en la maqueta. Para ello, se ha realizado una selección de los dispositivos electrónicos teniendo en cuenta el tamaño de los mismos, el consumo, el tipo de alimentación, o el tipo de control (si es analógico o digital, etc.), y la construcción de módulos genéricos de adaptación de niveles de corriente e intensidad o circuitos que aíslen de posibles interferencias generadas por actuadores. También se ha tenido en cuenta un esquema de colores para distinguir los cables, tipo de conectores específicos, serigrafía de circuitos desarrollados, leds para indicar el estado de las placas adaptadoras, y circuitos para separar la alimentación de los dispositivos de la lógica.



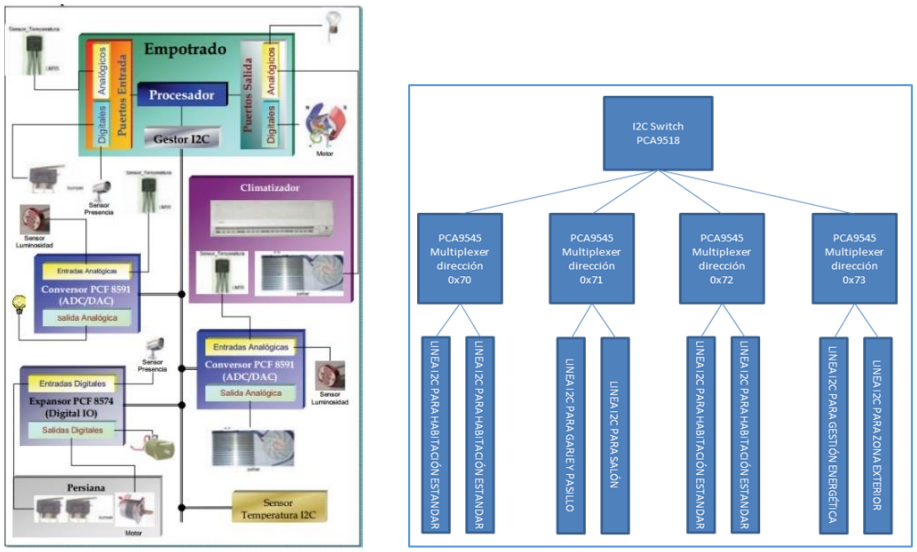
(a) HAL9013 (b) Casadomo

(c) DomoSOS (d) modulos propios y de terceros

Figura 4. El cableado se ha organizado de forma diferente en cada maqueta: (a) arriba en HAL9013, (b) lateral en CasaDomo/DomoLab y (c) abajo en DomoSOS. En (d) se muestra algunos modulos diseñados por el equipo y de tercero.

3.2. Selección del controlador principal.

El controlador principal de la MDD se encarga de organizar todos los dispositivos, sensores y actuadores disponibles en la MDD para que estén accesibles tanto para las aplicaciones clientes que tienen que interaccionar con los dispositivos domóticos como para cargar las aplicaciones que desarrollen los alumnos. Se implementa sobre un microcontrolador o un sistema empotrado conectado a todos los dispositivos domóticos de la MDD, que se encarga de generar las señales eléctricas precisas para leer el estado de los sensores, lo que nos permite monitorizar el entorno físico del hogar, y por otra parte genera las señales de control adecuadas para accionar los actuadores que interaccionan sobre el entorno físico del hogar. Dado que los sistemas empotrados cuentan con un número escaso o inexistente de entradas/salidas analógicas y digitales, la incorporación del bus basado en I^2C (ver Figura 5) facilita la extensión de las capacidades hardware del sistema agregando, por ejemplo, expansores digitales de E/S o conversores ADC.



(a) Bus I²C en SCD o DomoLab (b) Bus I²C jerarquico en DomoSos

Figura 5. Esquema de conexionado

El bus de comunicaciones I²C [7] permite controlar los dispositivos y expansores de E/S con un paradigma maestro-esclavo utilizando distintas velocidades (100 KHz a 5 MHz). Sin embargo, la velocidad está limitada por el dispositivo más lento y la estabilidad de la señal depende de que la capacitancia total del bus no supere los 400 pF. En la maqueta DomoSOS se ha ido un paso más alla, y se ha diseñado un bus del sistema plug-and-play jerárquico basado en el bus de comunicaciones I²C que facilita la incorporación de nuevos circuitos de adaptación, periféricos de E/S y expansores de E/S sin necesidad de resetear el sistema [8].

Para la selección del microcontrolador o empotrado se han valorado varios aspectos como la capacidad de conectividad, la capacidad de acceder a los dispositivos domóticos esclavos del bus I²C, la presencia de sistema operativo, y la posibilidad de ejecutar aplicaciones en el lenguaje de programación Java. En la tabla 2 se indica las características de los empotrados utilizados para cada maqueta.

Tabla 2. Lista de Empotrados utilizados en la MDD

Empotrado	SmartControl SC210	IM3910	Arduino Uno	Raspberry Pi B
Fabricante	Snijder	Imsys	Arduino	Raspberry
Características	Cirrus Logic EP7312, 64 MB RAM, 16 MB Flash	IM3910 (167 Mhz), 32 Mb RAM, 8 Mb Flash	ATmega328P (16 Mhz), 2 Kb SRAM, 32 KB Flash	ARM1176JZF-S (700 MHz), 512 MB RAM, SD Flash
Conectividad	50 GPIO, 3 ADC, I²C, LCD 5,7" color	8 I/O digital, I²C, SPI	13 E/S digital, I²C, 4 Ent. analó, Shield Ethernet	8 GPIO, SPI, I²C, UART
Maqueta	SCD	CasaDomo	DomoLab	DomoSos
Sist. Operativo	S.O. Tiempo Real	S.O. Tiempo Real	-	Linux
Programación	Java	Java	Processing	Java

4. Programación con la MDD

La programación con la MDD se puede realizar en varios niveles de abstracción basados en el lenguaje de programación Java, según las necesidades de la práctica o del sistema que se quiera probar sobre la maqueta. La MDD contiene la infraestructura básica que facilita la ejecución de aplicaciones Java sobre un entorno de ejecución JVM basado en JavaSE o JavaME, según el tipo de empotrado utilizado. Esto es, contiene tanto el sistema de carga de aplicaciones con el que se pueden desplegar aplicaciones sobre el empotrado como un agente de depuración que facilita la depuración remota de las aplicaciones sobre dicho empotrado. De este modo, la MDD proporciona un entorno completo de programación, compilación, despliegue y depuración de aplicaciones.

Si bien es factible desarrollar aplicaciones Java individuales que trabajen sobre un conjunto de dispositivos domóticos teniendo en cuenta las características específicas del entorno JVM instalado en el empotrado, se han desarrollado distintos tipos de tecnologías encuadradas en proyectos de innovación docente y en trabajos fin de carrera o de máster que han permitido extraer todo el potencial que pueden tener estas herramientas didácticas. Una lista de trabajos se muestra en la Tabla 3.

Tabla 3. Trabajos realizados sobre la MDD

Trabajos Fin de Carrera (TFC) / Trabajo Fin de Máster (TFM)
Viúdez J., Montes J.M. Sistema de Control Domótico HAL9013. TFC (2005)
Francisco J. González, Pablo Polo y Francisco J. Viceira. Desarrollo de un sistema domótico "inteligente" utilizando Jini. TFC (2006)
Mª Dolores Serrano y Sandra Rodríguez. Control domótico del hogar utilizando JXTA. TFC (2007).
Jaime Viúdez Aivar. Desarrollo de una plataforma para el diseño de sistemas de control con requerimientos en tiempo real en entornos empotrados basado en Java. TFM (2007)
Jesús Luis Muros Cobos. Framework con soporte de video para laboratorios remotos y virtual: aplicación al laboratorio remoto Domolab. TFM (2011)
Rafael Hidalgo. DomoWeb: Automatización domótica basado en servicios web. TFC (2012)
Francisco J. Novo y Juan M. Leal. Sistema de Teleasistencia DOMOSMART. TFC (2012)
Jonatan Sánchez Agulló. Diseño y Control de una Centralita Domótica accesible desde Dispositivo Android: Aplicación a la Maqueta Domótica. TFM (2012)
Manuel José Baena Toquero. DomoSOS: Sistema domótico sostenible basado en Raspberry Pi y Android. TFC (2013)

Como no es posible incluir todas las tecnologías desarrolladas, se comentan algunos ejemplos a continuación:

- **JavaES** (Java for Embedded Systems) [9] es un framework basado en Java que ofrece una plataforma adaptable, flexible y robusta para el desarrollo de aplicaciones sobre sistemas empotrados de 8, 16 y 32 bits. Facilita el diseño de aplicaciones de control (incluso con características de tiempo real) sobre dispositivos que se conecten a los puertos de E/S del entorno empotrado, o a través de buses digitales I^2C, SPI y 1-Wire. Para ello, proporciona mecanismos y facilidades de alto nivel para el acceso y manejo del hardware subyacente independientemente de la arquitectura hardware del sistema empotrado.

```
package example.buses;
import JavaES.digitalBus.*;
import JavaES.sys.*;

public class pruebaLM92{
    public static void main(String args [ ]) {
        JSYS.initialize(); // To Initiate any of supported embedded system.
        DigitalBus bus = JSYS.getDigitalBus(0,DigitalBus.I2C,"Bus_I2C");
        TemperatureSlave sensor = new LM92_I2CSlave(0,"LM92_sensor", bus);
        while(true){
            try{
              System.out.println("-Temperatura: "+sensor.getTemperature());
              Thread.sleep(1000);
            }catch(Exception e){ e.printStackTrace(); }
        }
    }
}
```

Figura 6. Ejemplo de un programa en Java que accede a un sensor de Temperatura a través de I^2C utilizando la tecnología JavaES sobre la maqueta Domolab.

- **DOHA** (Dynamic Open Home Automation) es una plataforma de servicios con una arquitectura orientada a servicios, diseñada para proporcionar soporte a sistemas distribuidos, ubicuos y dinámicos, así como para facilitar el acceso, control y gestión de espacios de interacción ubicuos como el hogar desde cualquier dispositivo de cómputo como ordenadores portátiles, teléfonos móviles o plataformas empotradas. Proporciona un modelo de programación basado en servicios que facilita la integración y colaboración de servicios que pueden ejecutarse en el mismo dispositivo o en diferentes dispositivos, ocultando la complejidad en las comunicaciones o en el acceso al hardware.

5. Conclusiones

La experiencia recogida en la construcción de maquetas nos ha permitido crear diferentes tipos de maquetas domóticas que nos permite ofrecer diferentes tipos de opciones a la hora de plantear actividades prácticas con fines docentes. Como consecuencia de los proyectos de innovación docente se han desarrollado guiones de prácticas para su aplicación en asignaturas de Ingeniería Informática e Ingeniería de Telecomunicación sobre diferentes niveles de abstracción. Así, por ejemplo, se han elaborado guiones de prácticas que enseñan a crear un driver de tiempo real de un dispositivo domótico, y otros guiones que enseñan a construir aplicaciones distribuidas basadas en la composición de servicios web como el diseño de un sistema de seguridad perimetral o de teleasistencia para personas mayores.

Sin embargo, la aplicación de la maqueta domótica al Aula ha tenido un éxito relativo por varios motivos. La utilización de la maqueta plantea algunas dificultades relacionadas con el transporte y puesta en marcha del sistema en el aula de prácticas, dificultades con el uso compartido de la maqueta cuando varios estudiantes tienen que trabajar con el mismo conjunto de dispositivos domótica, y una limitación de tiempo en el uso de la herramienta en ocasiones impuesta por el propio tipo de prácticas que

hacen complicada la finalización y prueba del sistema final. Aunque una solución fue la implantación del laboratorio remoto DomoLab, esto planteó otros problemas de fiabilidad cuando el estudiante realizaba una carga de aplicación no válida o incorrecta.

Por otra parte, debemos tener en cuenta que el abaratamiento del hardware y la aparición de los sistemas de open hardware como Arduino o Raspberry Pi han permitido plantear diseños parciales de una MDD basados en enfoques de primer tipo, que proporcionan acceso a un mercado de dispositivos (entre ellos dispositivos domóticos) que puede ser programados y controlados a través de microcontroladores o sistemas empotrados con recursos limitados. No sólo se incluyen facilidades para la programación, sino que se proporcionan conjuntos de dispositivos sobre la base de kit de desarrollo o "starter kit". Esto permite plantear la realización de prácticas directamente sobre el microcontrolador de una forma mucho más directa en lugar de plantearlo sobre toda la casa. Sin embargo, en este caso la aplicación a prácticas de mayor nivel de abstracción requiere el desarrollo de soluciones ad-hoc específicas.

En líneas generales la experiencia ha sido positiva, especialmente para la puesta en marcha y prueba de nuevas tecnologías como las que se han desarrollado en algunos de los trabajos fin de carrera o trabajos fin de máster.

Agradecimientos. La construcción de las maquetas no habría sido posible sin la colaboración de muchas personas que han contribuido con su aportación como Jaime Viudez Aivar, José Miguel Gutiérrez Guerrero, Jesús Muros Cobos, Manuel Baena Toquero, Francisco Novo Sánchez, Antonio Gutiérrez López, Sandra Sheila Rodríguez Valenzuela, Jaime Viudez Aivar, José Antonio Molina López, Jonathan Sánchez Agulló, entre otros.

Referencias

1. Viúdez J., Holgado Juan A. Diseño y construcción de una maqueta domótica controlable a través de microcontroladores Java. V Jornadas de Enseñanza a través de Internet/Web de la Ingeniería de Sistemas y Automática (EIWISA'07), pp. 47-52. Thomson. ISBN: 978-84-9732-603-2 (2007)
2. Holgado, Juan A. Proyecto de Innovación Docente: Maqueta Domótica para la Programación de Sistemas (número 08-181). Universidad de Granada (2008)
3. Holgado-Terriza, Juan A., Viúdez-Aivar Jaime, Capel-Tuñon Manuel I., Montes J.M. Diseño de un Sistema de Control Domótico basado en Java. Actas de las XXVII Jornadas de Automática (CEA), páginas 1401-1408. ISBN: 84-689-9417-0. Almería. (2006)
4. Holgado, Juan A. Proyecto de Innovación Docente: Maqueta Domótica para la Programación de Sistemas: Laboratorio Remoto DomoLAB (número 10-192). Universidad de Granada (2010)
5. Muros-Cobos, Jesus and Holgado-Terriza Juan A. A Componentizable Server-Side Framework for Building Remote and Virtual Laboratories.. International Journal of Online Engineering (iJOE), Vol (8), Num 4, pag. 43-51 (2012)
6. Manuel José Baena Toquero. DomoSOS: Sistema domótico sostenible basado en Raspberry Pi y Android. Trabajo Fin de Carrera. Universidad de Granada (2013)
7. Bus digital I^2C (Inter-Integrated Circuit). Circuitos Inter-integrados. http://www.nxp.com

8. Gutierrez-Guerrero, José M.; Holgado-Terriza, Juan A.; Muros-Cobos, Jesús; Pomboza-Junes, Gonzalo. Redes de Sensores I2C Inteligentes. Actas de las XVI Jornadas de Concurrencia y Sistemas Distribuidos, pag. 77-91. ISBN: 978-84-16478-90-3 (2016)
9. Holgado-Terriza Juan A., Viúdez-Aivar Jaime. JavaES, a flexible Java Framework for Embedded Systems. Distributed, Embedded and Real-time Java Systems, pp. 323-355. Springer (2012)

DESDE EL PUPITRE
(Experiencias de estudiantes)

Theremin DIY con Arduino UNO

Salvador Moreno[1], Miguel Damas[1], Hector Pomares[1], and Oresti Banos[2]

[1] Departamento de Arquitectura y Tecnología de Computadores
Universidad de Granada
[2] Telemedicine Group, University of Twente
smoreno94@correo.ugr.es, mdamas@ugr.es, hector@ugr.es,
o.banoslegran@utwente.nl,

Resumen. El objetivo que se persigue con este artículo es mostrar cómo mediante un trabajo de una asignatura propuesto por un alumno (autor principal del artículo) se puede asimilar el concepto de computación física, que es de suma relevancia en los estudios de Grado en Ingeniería Electrónica Industrial de la Universidad de Granada. Concretamente el proyecto implementa en una placa de Arduino UNO un *Theremin*. Mediante un sensor ultrasonidos HC-SR04 de bajo coste se obtiene la distancia del *Theremin* a la mano, y a partir de ésta se calcula una nota a emitir dentro de las escalas musicales disponibles en el diseño. La etapa de sonido analógica está conformada por con un amplificador de sonido de baja potencia PAM8403 y un altavoz de 4 Ω. Por último, el control del *Theremin* está facilitado por un LCD de 16x2 caracteres, conectado vía I2C, y tres pulsadores.

Palabras clave: Arduino, audio, electrónica, ultrasonidos, Procesadores Integrados

Abstract. The main objective of this paper is to expose how some valuable concepts related to physical computation, which are very valuable in the Bachellor's Dregree of Industrial Electronic Engineering, can be tackled through a project proposed by a student (the main author of the paper). This work consists of a Theremin implemented in an Arduino UNO board. A low-cost ultrasonic sensor HC-SR04 measures the distance to the controlling hand, calculating then a single note to emit within the available scales in the design. The analogic sound stage is conformed by a low-power sound amplifier PAM8403 and a 4 Ω speaker. Finally, the Theremin's control is provided by a 16x2 LCD, connected via I2C, and three buttons.

Keywords: Arduino, audio, electronics, ultrasonic, Integrated Processors

1. Introducción

Arduino [4] se ha implantado como una de las herramientas más útiles para aquellos que buscan en la electrónica una manera de llevar a cabo sus proyectos

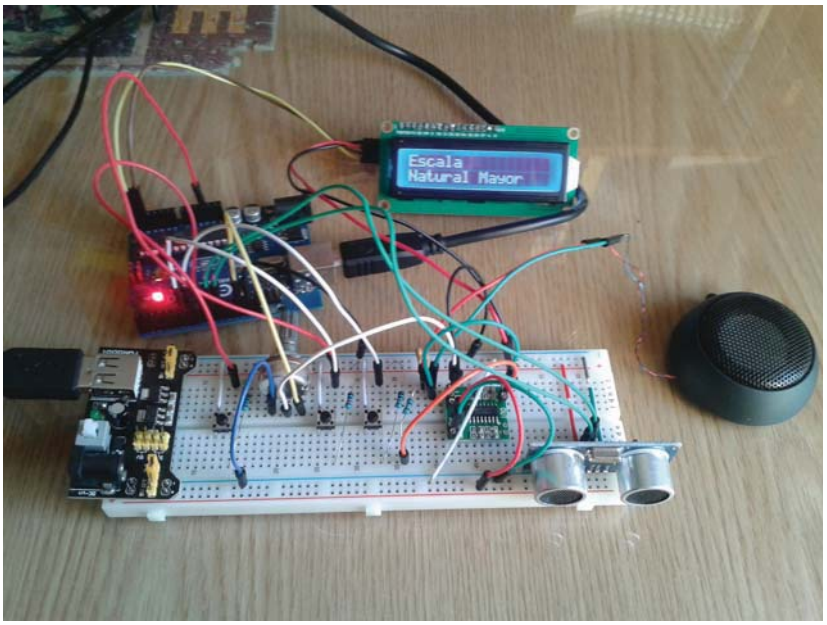


Figura 1. Montaje del dispositivo

y materializar su espíritu creativo. Este proyecto nace como respuesta a una propuesta abierta de trabajo de la titulación de Grado en Ingeniería Electrónica Industrial, concretamente de la asignatura Procesadores Integrados – y de las inquietudes musicales del principal autor del trabajo. Esta asignatura se centra en el estudio de microprocesadores, microcontroladores y procesadores de señales digitales (DSP). Se evalúan las distintas posibilidades que ofrece cada uno de estos componentes, estudiando así diferentes tipos de microcontroladores y aprendiendo a valorar las características necesarias para optimizar nuestros diseños. Además, se analizan los diversos buses y métodos de conexión que se encuentran en la mayoría de los sistemas implantados a día de hoy. Se adquiere así una base con la que el estudiante puede comenzar en relativamente poco tiempo a diseñar sus sistemas, conociendo lo más esencial de cada una de las partes que lo conforman.

El *Theremin* [1] es uno de los primeros instrumentos musicales electrónicos, desarrollado en 1920. Se compone de dos antenas que, determinando la distancia a la que se encuentra la mano del músico, controlan cada una un oscilador y con él la frecuencia del sonido (que determina la nota musical) y el volumen. Ha sido utilizado por multitud de músicos a lo largo de la historia, de compositores como Percy Graigner o Jean Michael Jarre, llegando incluso a la música popular de manos de grupos como *Led Zeppelin* o *Pink Floyd*. La electrónica de este dispositivo es capaz de fascinar a cualquier electrónico, además de lo atractivo

que se hace como proyecto debido a su índole musical y a que cualquier persona puede interactuar con él.

Se plantea en este trabajo una versión mucho más asequible a nivel de complejidad, así como de presupuesto, como se puede ver en la Figura 1. Buscaremos esa detección de la posición desde la mano hasta nuestro *Theremin* mediante sensores ultrasonidos de bajo coste como el HC-SR04, adaptando las características del sistema a las limitaciones inherentes al mismo. Se prescinde así de la electrónica compleja de las antenas y osciladores para verse sustituida por un tratamiento de datos y de las señales óptimo, concedidas por la unión del sensor de ultrasonidos con el Arduino UNO. En la bibliografía [2, 3] se pueden encontrar implementaciones más sofisticadas que se siguen sirviendo de Arduino para recoger la señal de los osciladores.

En el marco que proporciona la asignatura de Procesadores Integrados se busca con este proyecto adquirir competencias específicas relacionadas con el uso de microcontroladores (ATMega328P incluido en Arduino UNO), la programación en C++, el procesamiento de señales recogidas con un sensor o transductor (sensor ultrasonidos HC-SR04), la conexión entre elementos del sistema y el mostrar resultados (a través de un LCD de 16x2 conectado por I2C [7]).

2. Esquema y descripción del Theremin en Arduino

Entre la variedad disponible de productos Arduino se escoge el UNO ya que buscamos la máxima simplicidad en nuestro Theremín, y que las características del ATMega328P que lo gobierna son óptimas para el control de las señales de ultrasonidos, la emisión de sonido y el funcionamiento del LCD. Las interrupciones del microcontrolador serán las que comanden el sonido y la recepción de ultrasonidos de manera precisa.

2.1. Medida de distancia con el sensor HC-SR04

El sensor ultrasonidos HC-SR04 se trata de un medidor de distancia de bajo coste y de funcionalidad limitada. Para su implementación se ha utilizado la librería NewPing [5] (mejorando la oficial Ping), que, sirviéndose del Timer 2 realiza una mejor medida. Además se ha implementado una media móvil de 5 muestras para suavizar la respuesta del dispositivo, mientras se mantiene una sensibilidad adecuada a los cambios de la posición de las manos. Cabe destacar que las capacidades de este sensor están muy restringidas. Aunque aseguran una precisión de hasta 3 mm, las pruebas no otorgan una resolución más allá de 1 cm, en los mejores casos. Estas medidas se realizaron para variaciones de posición relativamente rápidas, siempre utilizando superficies planas para la detección en lugar de la mano. Esto ha impedido implementar un rango de frecuencias de sonido continuo como hace un auténtico *Theremin*. En su defecto, se ha optado por la discretización de las notas, cada una con su frecuencia asignada, en distintas escalas como solución más plausible, facilitando así la utilización musical del mismo. Se han implementado cinco octavas consecutivas

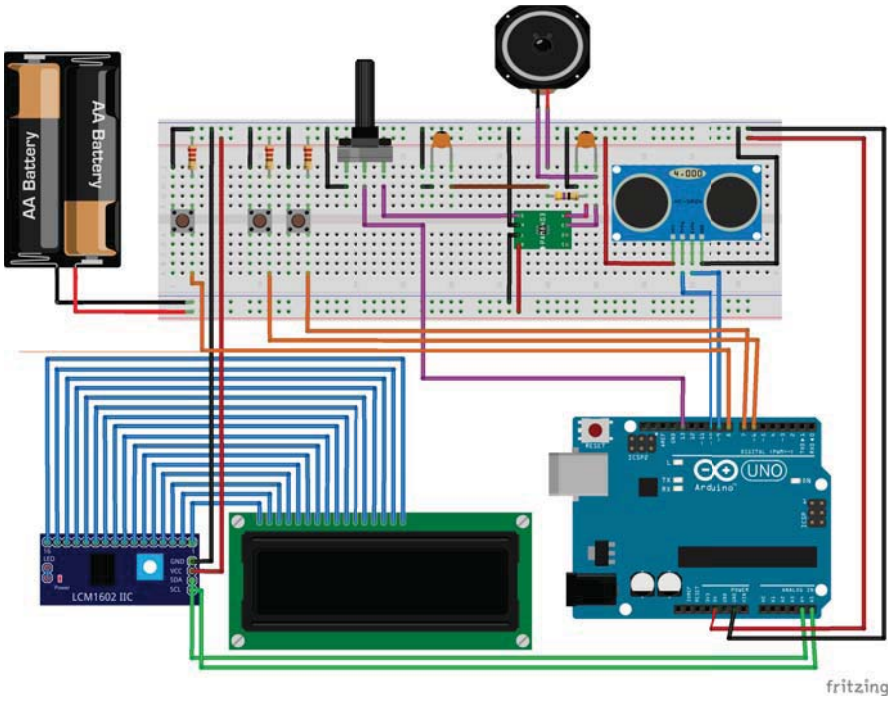


Figura 2. Diagrama de conexión

en el intervalo 3—7 con nota base Do. Las escalas musicales son la cromática, natural mayor y menor, y pentatónicas mayor y menor.

2.2. Control del Theremín

Se disponen tres pulsadores en configuración pull-up con resistencias de 10 $k\Omega$ para control del Theremín:

- *Intro* - ON/OFF
- Más (+)
- Menos (-)

Nos servirán para movernos por el menú de selección de escala. Una vez seleccionada, el primer botón (*Intro*) nos servirá para iniciar el sonido del *Theremin*. Mientras el dispositivo esté sonando, podremos aumentar o disminuir la octava en las que el programa selecciona las notas con los botones (+) y (-). Pulsando de nuevo el botón ON/OFF paramos la reproducción de sonido.

2.3. Etapa de audio con amplificador PAM8403

Para emitir nuestro sonido utilizamos el amplificador de audio PAM8403 y así conectar nuestro altavoz de 4Ω. La conexión de esta etapa viene implementada

Figura 3. Sensor ultrasonidos HC-SR04

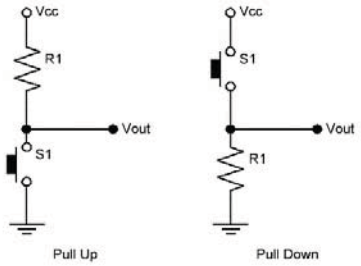


Figura 4. Pulsador en pull-up y en pull-down

para facilitar la adaptación de un altavoz de mayor potencia (hasta 3 W) e incluso una posible adición de sonido estéreo. Para este caso hemos utilizado únicamente un canal (mono). Añadimos también 2 condensadores cerámicos 0.1 uF para filtro de ruido de la etapa por alimentación y una resistencia de 47 Ω en serie con el altavoz para protegerlo, ya que está diseñado para funcionar a una potencia inferior a 3 W.

Un potenciómetro logarítmico de 10 $k\Omega$ realiza el control de volumen. Se ha prescindido de la utilización de otro ultrasonidos para el control de volumen por motivos inherentes a la gestión de interrupciones por parte de Arduino UNO, insuficientes para controlar a la vez dos HC-SR04 y la emisión de sonido.

Para la alimentación de esta etapa ha sido necesario un módulo de carga a 5V/3.3V. A efectos de test puede utilizarse un altavoz piezoeléctrico en lugar de toda la etapa analógica de amplificación. El sonido se envía a la etapa mediante la NewTone [6](mejorando la librería Tone por defecto) que nos permite seleccionar la frecuencia de exacta del tono que queremos utilizando el Timer 1.

2.4. Monitorización

Display. Se utiliza un LCD de 16x2 para visualizar resultados junto con su módulo LCD1602 para controlarlo vía I2C. El control ha sido realizado cómodamente con la librería LiquidCrystal_I2C. La utilización de un módulo I2C está justificada por la gran reducción de número de pines a utilizar. El LCD requiere 6 pines de control más 2 de alimentación, y añadiendo el controlador I2C lo reducimos a 2 pines de control más 2 de alimentación. Ésto es útil sobre todo para una implementación en Arduinos de pequeño tamaño como el UNO.

Watchdog. El watchdog está programado para reiniciar la placa de Arduino en caso de que exista un bloqueo de más de un segundo. El tiempo asignado es alto ya que las pulsaciones vienen acompañadas de delay(100) para evitar rebotes. Luego, puesto que realizamos una media de las últimas 5 muestras del

ultrasonidos, éstas nos ocupan unos 5x33ms, es decir, 165ms como mínimo de interacción con el HC-SR04. Esto sumado a otros procesos del programa hacen que 500ms o menos no sean suficientes para la programación del watchdog.

3. Materiales y Presupuesto

Los componentes necesarios vienen dispuestos en la Tabla 1. El potenciómetro es logarítmico porque el sonido también es percibido de la misma manera (en dB). Las resistencias escogidas son de 1/6 Watt de potencia para asegurar su correcto funcionamiento según la potencia del diseño. El presupuesto total ha sido de 58,35 €.

Componente	Proveedor	Ud	Precio/Ud [€/Ud]	Coste [€]
Potenciómetro – Rotary Pot 10k Ohm, Log	sparkfun.com	1	2,35 €	2,35 €
Sensor de distancia ultrasonidos – HC-SR04	dx.com	1	2,05 €	2,05 €
Pulsador – Mini Pushbutton Switch	sparkfun.com	3	0,29 €	0,87 €
Resistor 1k Ohm 1/6 Watt PTH	sparkfun.com	3	0,21 €	0,63 €
Resistor 47 Ohm 1/6 Watt PTH	sparkfun.com	1	0,21 €	0,21 €
PAM8403 DC 5V	dx.com	1	1,20 €	1,20 €
Condensador cerámico 0.1uF	sparkfun.com	2	0,21 €	0,42 €
Altavoz 4-8 Ohm 0.5W	sparkfun.com	1	1,62 €	1,62 €
Arduino UNO	sparkfun.com	1	20,79 €	20,79 €
Módulo LCD I2C – LCD1602	dx.com	1	1,67 €	1,67 €
Basic LCD 16x2	sparkfun.com	1	12,46	12,46 €
Altavoz piezoeléctrico	sparkfun.com	1	1,62	1,62 €
Breadboard Power Supply Stick 5V/3.3V	sparkfun.com	1	12,46	12,46 €
COSTE TOTAL				**58,35 €**

Tabla 1. Presupuesto

4. Conclusiones

En este trabajo se ha desarrollado un instrumento musical basado en los principios del *Theremin* e implementado en Arduino UNO con un bajo presupuesto. Se han adquirido y ampliado competencias muy valiosas dentro del campo de la ingeniería electrónica, como sobre tratado y procesamiento de señales de un sensor ultrasonidos HC-SR04, del que se obtiene, a partir de la distancia medida una nota dentro de una multitud de escalas y octavas disponibles, y conocimientos entre partes de un sistema, como la comunicación I2C entre Arduino y el LCD. I2C es además un estándar de comunicación que sigue vigente a día de hoy en multitud de sistemas industriales y de consumo.

Existen multitud de mejoras a implementar, como sustituir el potenciómetro del volumen por otro sensor ultrasonidos (para lo cual haría falta actualizar la versión de Arduino Mega), cambiar el modelo de sensor ultrasonidos por otro

modelo comercial con una mayor resolución, inserción de un modo metrónomo o su implementación como shield de Arduino UNO.

Se ha realizado así un trabajo completo dentro de las competencias de la asignatura Procesadores Integrados del Grado en Ingeniería Electrónica Industrial, donde además de los conocimientos específicos referentes al desarrollo técnico, se han desarrollado una serie de habilidades generales útiles a la hora de realizar el planteamiento, la planificación y la gestión de un proyecto.

Bibliografía

1. *Theremin World*. Accedido 14 de abril de 2016. http://www.thereminworld.com/
2. Liu, Tsung-Ching, Shu-Hui Chang y Che-Yi Hsiao. *A modified Quad-Theremin for interactive computer music control.* International Conference on Multimedia Technology (ICMT), 6179-82, 2011.
3. Gomes, A., D. Albuquerque, G. Campos, y J. Vieira. *TheremUS: The Ultrasonic Theremin*, 2:779-85, 2009.
4. *Arduino - Home*. Accedido 14 de abril de 2016. http://www.arduino.cc/.
5. *Arduino Playground - NewPing Library*. Accedido 14 de abril de 2016. http://playground.arduino.cc/Code/NewPing.
6. *Arduino New Tone*. Accedido 14 de abril de 2016. https://bitbucket.org/teckel12/arduino-new-tone/wiki/Home.
7. *I2C Bus*. Accedido 14 de abril de 2016. http://www.i2c-bus.org/.

Aprendiendo mecatrónica mediante el diseño y construcción de una plataforma de lanzamiento de prototipos de cohetes de agua

Cristian Márquez, Rodrigo Agis, Antonio Cañas, Miguel Damas

Departamento de Arquitectura y Tecnología de Computadores
Universidad de Granada
Granada, España
crsmarqz@correo.ugr.es, (ragis, acanas, mdamas)@ugr.es

Resumen. Este artículo describe un TFM del Máster en Ciencia de Datos e Ingeniería de Computadores de la Universidad de Granada, consistente en la implementación de una plataforma de lanzamiento de prototipos de cohetes de agua, donde se ha realizado un diseño mecatrónico que permite el control del sistema de forma remota desde el PC. El objetivo que se persigue con este trabajo es mostrar mediante un ejemplo práctico algunos de los conceptos y habilidades adquiridas en dicho Master relacionadas con la mecatrónica. El diseño parte del modelado 3D de cada pieza que interviene en el mecanismo, luego se seleccionan los dispositivos electrónicos que se utilizan para el control de la plataforma, y finalmente se desarrolla el sistema virtual que permite el control desde el PC. El mecanizado de piezas e integración de todos los sistemas se realizó en el laboratorio de robótica y mecatrónica del CITIC-UGR.

Palabras Clave: Mecatrónica, Plataforma de lanzamiento, modelado 3D, mecanizado, cohete de agua.

Abstract. This paper describes a TFM of the Master in Data Science and Computer Engineering at the University of Granada, which is implementing of a launching pad prototype rocket water, where there has been a mechatronic design that allows control system remotely from the PC. The objective sought to be achieved with this work is to show through a practical example some of the concepts and skills acquired in this Master related to mechatronics. The design of the 3D modeling of each piece involved in the mechanism, then the electronic devices used to control the platform, and finally the virtual system that allows control from the PC develops are selected. Machining parts and integration of all systems was conducted in the laboratory of robotics and mechatronics CITIC-UGR.

Keywords: Mechatronics, launch pad, 3D design, machining, water rocket.

1 Introducción

Este proyecto se desarrolla en el marco de la iniciativa UGRASP (UGR Aero Space Program), impulsada por el Departamento de Arquitectura y Tecnología de Computadores de la Universidad de Granada para ayudar a los estudiantes a desarrollar sus habilidades de ingeniería (telecomunicaciones, mecánica, control, etc.) a través de misiones espaciales simuladas tales como vehículos voladores, robots móviles o plataformas de lanzamientos, entre otras.

Una plataforma de lanzamiento son las instalaciones y el área donde despegan los cohetes y las naves espaciales. Una base espacial (o lugar de despegue de cohetes) puede contener una o varias plataformas. Una plataforma de lanzamiento típica consiste básicamente en estructuras de servicio y líneas de abastecimiento [1].

En cuanto a las plataformas de lanzamientos de cohetes de agua, existen varios modelos y diseños, desarrollados por aficionados o grupos de estudiantes de escuelas secundarias que realizan proyectos de cohetes de agua (Fig. 1) [2].

Debido a esto, es escasa la información que se puede encontrar acerca de plataformas como las que se describe en este trabajo, donde se realizan tareas multidisciplinares en las que se conjugan varias ramas de la ingeniería, con el objetivo de lograr la integración de todos los elementos mecánicos y de control electrónico para un sistema de control y monitorización remoto. Entre los modelos que se encuentran en la web los más comunes son los de elaboración casera y accionamiento manual [3].

Figura 1. Plataforma de lanzamiento de cohetes de agua, de elaboración casera

Dentro de la propuesta de UGRASP, éste trabajo se plantea con el objeto de desarrollar capacidades de ingeniería multidisciplinar, aprendiendo de experiencias de grandes referentes en el área como la NASA y ESA, para motivar a los nuevos alumnos en su formación universitaria.

Para el desarrollo de la plataforma se ha realizado un diseño que simplifica la complejidad del control, incorporando elementos de hardware libre, actualmente con ilimitada información en la web sobre su uso y funcionamiento. En cuanto a la

programación y control se ha diseñado una plataforma que permite la modificación o expansión del proyecto de forma fácil, donde con una base pequeña de conocimientos de programación se podría modificar parámetros de la plataforma.

El resto del artículo se divide en las siguientes secciones: en la sección 2 se hace una revisión del diseño mecatrónico del modelado de las piezas, la configuración de dispositivos electrónicos y el desarrollo del software que permite el control desde el ordenador. En la sección 3 se detalla el proceso de fabricación e implementación de la plataforma, desde el mecanizado de piezas hasta la integración de todos los sistemas. En la sección 4 se aportan los resultados obtenidos de esta experiencia, y por último en la sección 5 se presentan las conclusiones.

2. Diseño del sistema mecatrónico

El diseño del sistema incluye todo lo correspondiente a los esquemas desarrollados tanto en la parte mecánica así como también un pequeño diseño de la parte electrónica para el control.

Para el desarrollo del diseño mecánico se obtuvieron ideas y sugerencias del proyecto de la plataforma de lanzamiento de cohetes desarrollado en la asignatura de Mecatrónica y Sistemas Aeroespaciales del Máster de Ciencia de Datos e Ingeniería de Computadores, periodo 2014 - 2015. [6]

El diseño mecánico consta de dos partes, la primera determina toda la infraestructura donde se van a montar todas las piezas fijas y móviles además de los dispositivos que ayudan al control del proyecto, y la segunda parte es el sistema que permite el llenado y presurización del cohete (Fig. 2).

Para una visualización conceptual se desarrolló una vista en 3D (Fig. 3), la cual permite depurar y corregir de manera más detallada el diseño de la estructura previo al mecanizado de las piezas con la fresadora CNC del laboratorio.

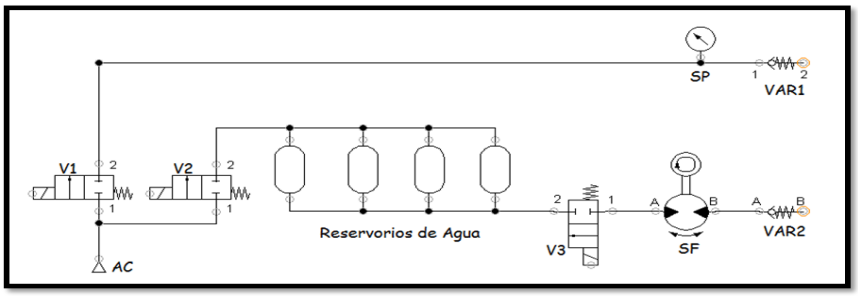


Figura 2. Diseño del sistema de llenado y presurizado del cohete

Donde:
V1, V2, V3: Válvulas Solenoides (12 Vdc).
AC: Aire Comprimido (Compresor).
SP: Sensor de Presión de Aire.
SF: Sensor de Flujo de Agua.
VAR1, VAR2: Válvulas anti retorno.

Para desarrollar las piezas definitivas se realizaron varios modelos previos en 3D, los mismos que permitieron pulir detalles para conseguir el diseño final, siguiendo un proceso de desarrollo y optimización.

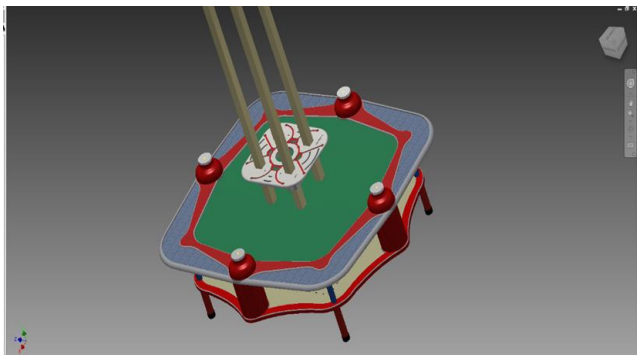


Figura 3. Diseño 3D de Ensamble de piezas

Para el diseño electrónico se pensó en varias soluciones para la elección de la plataforma de control, quedando como la más óptima la placa Arduino Uno. La comunidad Arduino ofrece muchas ventajas para el desarrollo de prototipos, además de disponer de una amplia gama de sensores e interfaces que permiten el manejo de periféricos de entrada y salida, digitales y analógicos.

Para una visualización previa al montaje en la placa de prototipo, se realizaron pruebas en el software de diseño Proteus 8 Profesional (Fig. 4).

Las pruebas realizadas con dicho software permiten simular los circuitos en condiciones ideales, sin tomar en cuenta la fuente de alimentación y la comunicación mediante bluetooth. Es importante resaltar estos detalles ya que durante la implementación e instalación se realizó una selección rigurosa que permitió un funcionamiento adecuado, muy similar al que se pudo observar durante las simulaciones.

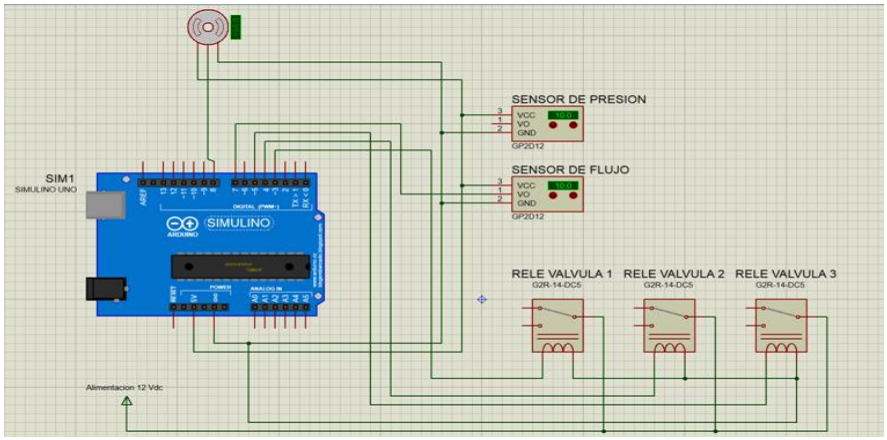


Figura 4: Esquema del circuito electrónico con placa de desarrollo Arduino

Para el diseño del software partimos de que los dispositivos que se seleccionaron, en este caso la tarjeta Arduino, nos permiten comunicar con el software que queremos escoger, en este caso LabView. Para poder establecer una comunicación entre Arduino y LabView, se requieren de dos elementos muy importantes, uno es el ToolBox de Arduino para LabView que se puede encontrar en el *package manager* de LabView, y la segunda herramienta importante es el firmware que ha de cargarse en la placa Arduino, que permite comunicar dicha placa con el ToolBox previamente instalado en LabView [5].

Una vez desarrollado el circuito que permite el control, se estableció un esquema de funcionamiento representado mediante etapas y transiciones (Fig. 5), que sigue la secuencia de lanzamiento del cohete de agua. Es decir, que las entradas del circuito (sensores, pulsadores, temporizadores, etc.), a medida que vayan cumpliendo su función irán activando las salidas (válvulas, mecanismo de despegue) en concordancia con la secuencia de lanzamiento.

La secuencia comienza con el accionamiento del botón inicio desde el ordenador, para dar paso al accionamiento de la válvula de control de aire comprimido de todo el circuito, donde pasado un tiempo, se activa la válvula llenado de agua del cohete hasta que el sensor de caudal lo indique, para que luego se de paso a la etapa de presurización del cohete de la misma forma hasta que el sensor lo indique. A partir de la siguiente etapa se espera el pulso desde el ordenador, que permite accionar el mecanismo de despegue del cohete. Es importante mencionar que los estados de los actuadores y el valor de los sensores se pueden visualizar desde la plataforma de control en el ordenador.

3. Fabricación de la plataforma de lanzamiento “SATURNIA”

Para el proceso de fabricación de la plataforma de lanzamiento partimos del modelado y diseño de las piezas en AutoCAD [8], software que permite el desarrollo de planos de construcción en 2D (Fig. 6). Una vez que se han adecuado los planos de cada pieza, se procede con la exportación al software VCarve [10] que configura el programa de mecanizado de la Maquina CNC, la cual corta, desgasta y taladra el material de forma automática en base al diseño cargado. El material que se seleccionó para la fabricación de las piezas fueron chapas de aluminio de 8 y 5 mm de espesor, tubos cuadrados de aluminio de 800 mm de largo, 50 mm de ancho y 2 mm de espesor.

Es importante mencionar que para el proceso de fabricación es necesario tener en consideración varios criterios del método constructivo para el mecanizado de piezas, además de algunos criterios de manejo y selección de las herramientas que permiten una adecuada sujeción de las piezas en construcción.

Mediante la configuración del software VCarve, el diseño se somete a un proceso de mecanizado virtual, donde se establece rutas que seguirá la máquina CNC (Fig. 7), para perforar, cajear (o desgastar parcialmente el material) y cortar. Es necesario saber el diámetro de las fresas que se disponen para establecer puntos de partida a conveniencia, con el fin de obtener las medidas deseadas.

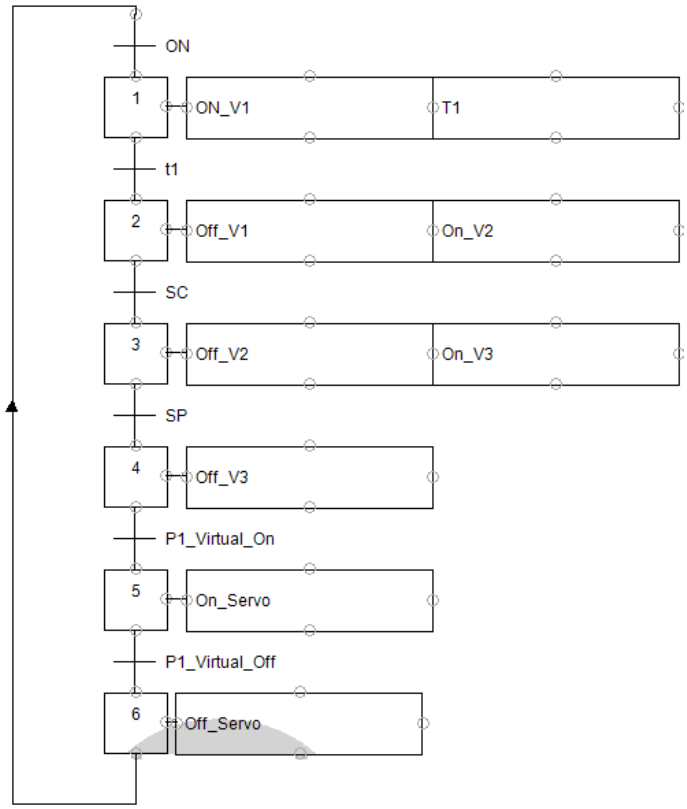


Figura 5. Diagrama de estados y transiciones de la plataforma de lanzamiento para prototipos de cohetes de agua

Donde:
V1, V2, V3: Válvulas de Solenoide 1, 2, 3
T1: Tiempo de transición 1
SC: Sensor de Caudal
SP: Sensor de Presión
P1_Virtual: Pulso Virtual para activar o desactivar servomecanismo de lanzamiento

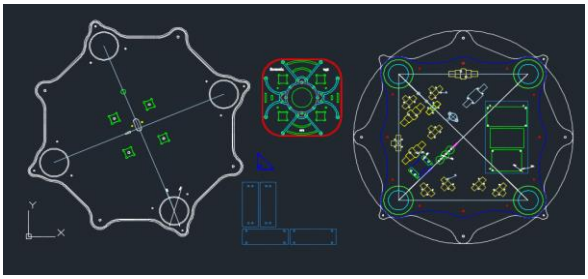


Figura 6. Diseño de piezas en 2D, en AutoCAD

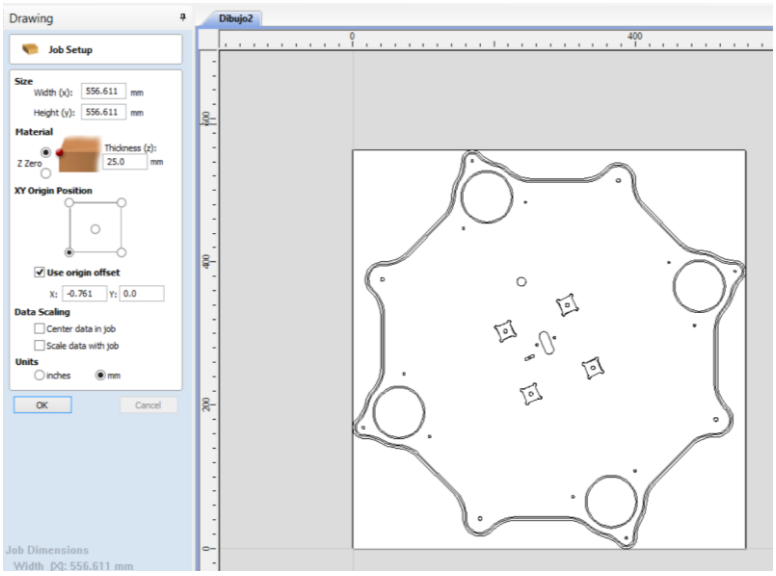


Figura 7. Proceso de cajeado y perforado en 2D previo al mecanizado mediante software VCarve.

Para el desarrollo de la etapa de control se seleccionaron los dispositivos electrónicos necesarios que deben intervenir en el control de la plataforma Saturnia, los mismos que fueron conectados de acuerdo al diseño simulado anteriormente (Fig. 8). Una vez implementado el circuito y con el firmware necesario se realizaron las primeras pruebas de funcionamiento del circuito, la lectura de sensores y la correcta activación de los actuadores.

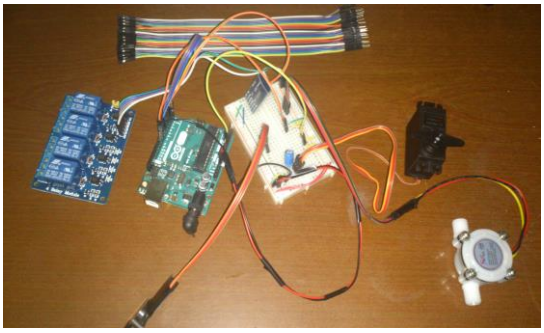


Figura 8. Circuito de control de la plataforma de lanzamiento de cohetes de agua, SATURNIA

Una vez realizada las pruebas de conexiones de la tarjeta Arduino se procedió con pruebas de conexión y transmisión de datos por bluetooth (Fig. 9), estableciendo comunicación entre el ordenador que alberga a LabView con la tarjeta Arduino, mediante la ToolBox de LabView para control de Arduino.

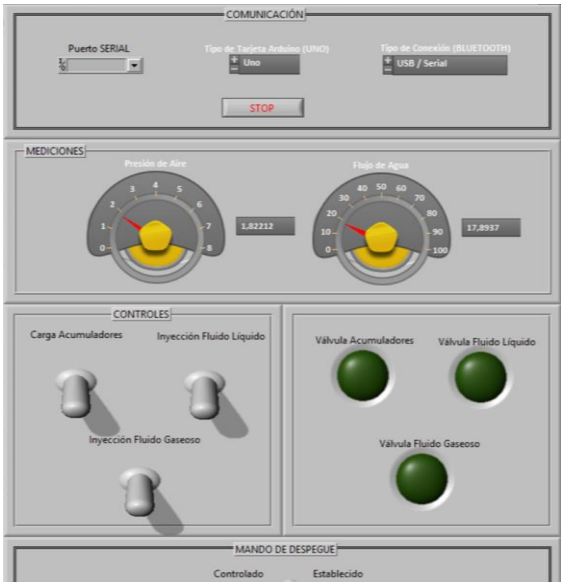


Figura 9. Panel de comunicación, control y visualización con LabView

La secuencia de funcionamiento se implementó en la plataforma LabView (Fig. 10), donde se realiza la comunicación bluetooth, la lectura de los sensores y el control de las salidas que intervienen en la plataforma de lanzamiento.

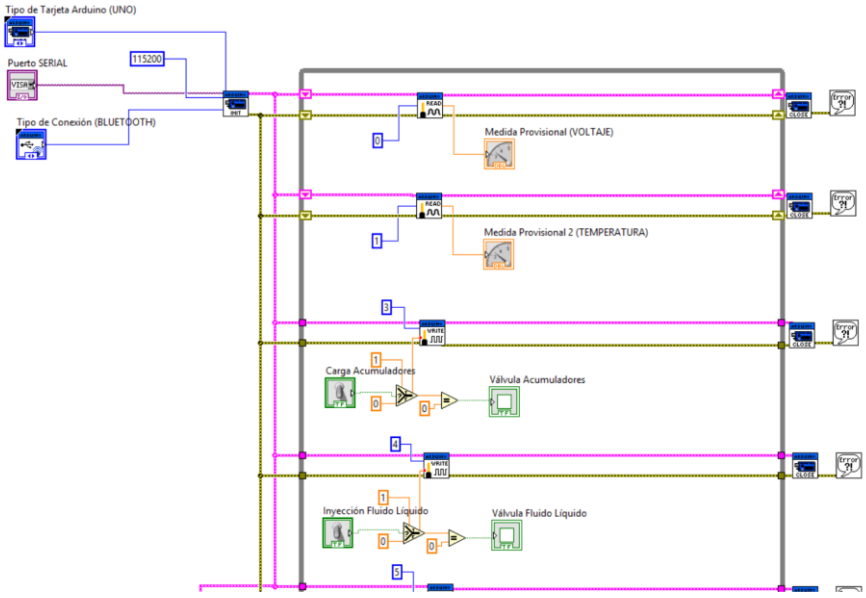


Figura 10. Programa de control y comunicación implementado en LabView

4. Resultados

Una vez terminado con el armado y sujeción cuidadosa de las piezas fabricadas y de los dispositivos que fueron adquiridos para el control, se realizaron las pruebas de conexión eléctrica y neumática para la alimentación de los circuitos (Fig. 11).

Durante las pruebas de llenado se observaron pérdidas de agua en la conexione del cohete debido a la presión que generaba el aire comprimido, lo cual se pudo solucionar con una válvula anti-retorno.

Las lecturas de los sensores fueron calibradas utilizando el manómetro provisto en el compresor de aire y el volumen máximo que podía albergar cada cohete. Las válvulas anti-retorno que se fijaron a continuación de cada sensor permitieron obtener mejores medidas.

La comunicación bluetooth establecida entre Arduino y LabView permitió controlar la plataforma de lanzamiento en espacio abierto con un rango de distancia de hasta 20 metros aproximadamente.

La toma de conexión de los cohetes con la plataforma se diseñó de tal forma que pueda ser considerada de forma universal, es decir que los cohetes fabricados con cualquier tipo de botella de plástico puedan ser utilizados.

El mecanismo de despegue en principio tuvo unos pocos inconvenientes para permitir la liberación debido a la fricción entre los materiales, lo cual se solucionó realizando una pequeña modificación en la sujeción del mecanismo de accionamiento.

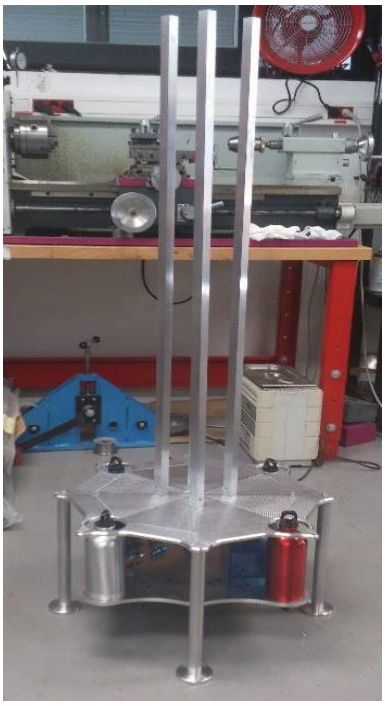


Figura 11. Plataforma ensamblada

4. Conclusiones

En éste trabajo se ha mostrado el desarrollo de un proyecto para la construcción de una plataforma de lanzamiento de prototipos de cohetes de agua, para lo que se ha tenido que recurrir a varias disciplinas de la ingeniería, es decir a la mecatrónica, para por ejemplo, la construcción mecánica y el diseño de la electrónica de control, o el desarrollo de una aplicación para su monitorización de forma remota.

Durante el diseño y la construcción de la plataforma de lanzamiento para cohetes de agua fue necesario contar con un adecuado esquipo de herramientas virtuales y físicas que permitieron la fabricación de las piezas.

Durante las pruebas de funcionamiento de la plataforma se comprobó que el diseño de llenado y presurización del cohete fue el más óptimo ya que permitió un adecuado despegue del cohete.

El control de la plataforma por medio de una comunicación bluetooth ha permitido la conexión instantánea del ordenador con la plataforma tanto para leer como para controlar los dispositivos conectados.

La integración de varios sistemas requiere de una adecuada planificación que parte de un tiempo adecuado para el diseño y la fabricación, y luego la implementación y las pruebas que se realicen.

Finalmente señalar que debido al presupuesto limitado varias ideas se han quedado fuera, pero se espera que en un futuro se pueda contar con algo más de presupuesto para las mejoras del proyecto Saturnia, tales como la implementación del control para sistemas móviles o añadir un mecanismo de lanzamiento angular para dar trayectorias curvas al cohete, entre otras.

Referencias

1. Water Rocket Simulator. http://www.grc.nasa.gov/WWW/K12/rocket/BottleRocket/sim.htm
2. Experciencia, http://www.experciencia.com/festa-de-la-ciencia-en-barcelona/
3. Cohetes Propulsados por agua, Artusa, Juan Ignacio. http://www.astroeduc.com.ar/ COHETES%20PROPULSADOS%20POR%20AGUA%20ISFN.pdf
4. Rocket Activity Water Rocket Launcher. http://www.nasa.gov/pdf/ 153405main_Rockets Water_Rocket_Launcher.pdf
5. VI Package Manager http://jki.net/vipm
6. Goddard, Robert (2002), Rockets, New York: Dover Publications, ISBN 978-0-486-42537-5
7. Cohetes de agua: C. J. Gommes, A more thorough analysis of water rockets: moist adiabats, transient flows, and inertials forces in a soda bottle. https://es.wikipedia.org/wiki/ Cohete_de_agua
8. Autocad. http://www.autodesk.es/products/autocad/overview (Autocad)
9. Inventor. http://www.autodesk.es/products/inventor/overview (Inventor)
10. Vcarve. http://www.vectric.com/products/vcarve.htm
11. Labview. http://www.ni.com/labview/esa/
12. FluidSIM. http://www.fluidsim.de/fluidsim/index5_e.htm

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

June 2016, Number 6