Shield Arduino de bajo coste para la enseñanza de asignaturas de Ingeniería

Andres Gongora, Juan-Antonio Fernández-Madrigal, Ana Cruz-Martín, Vicente Arevalo, Cipriano Galindo, Carlos Sánchez-Garrido, Javier Monroy, Javier Fernández-Cañete[1]

Resumen—**En este artículo presentamos el diseño, implantación y evaluación de una novedosa circuitería que extiende la popular placa Arduino UNO, para facilitar la realización de múltiples actividades educativas en diferentes cursos de ingeniería. Los principales objetivos de nuestro trabajo son producir un *shield* (extensión PCB para Arduino) de bajo coste, y además proporcionar un gran número de experimentos educativos con él. Ambos se han logrado mediante la cuidadosa elección de componentes electrónicos y una óptima combinación de los mismos. El diseño resultante puede fabricarse a un coste significativamente menor que otros dispositivos similares (alrededor de 50€ sin considerar la mano de obra - soldadura de componentes y diagnósticos básicos), y tanto los tests simulados como los reales han demostrado que la funcionalidad resultante se ajusta con exactitud a nuestros requerimientos de diseño si se usan métodos sencillos de calibración. Nuestra intención es utilizar la placa en asignaturas muy diversas en los próximos años; hasta la fecha ya se ha incluido en una de control de sistemas (grado) y en otra de tiempo real para sistemas empotrados (máster). Aquí presentamos las posibilidades de nuestro *shield* para éstas y otras asignaturas, y también algunos resultados educativos obtenidos durante el presente curso.**

Palabras clave—**Placa educativa con microcontrolador, placa de entrenamiento en ingeniería, sistemas empotrados de tiempo real, ingeniería de sistemas de control.**

I. INTRODUCCIÓN

EL trabajo de laboratorio es un aspecto fundamental en la formación de los estudiantes de ingeniería, ya que sólo trabajando con dispositivos reales pueden relacionar teoría y práctica adecuadamente, reforzando y mejorando la comprensión de ambas; es también ahí donde comienzan a conocer los problemas a los que se enfrentarán en su futuro laboral.

Hay una gran diversidad de dispositivos hardware dedicados a facilitar las prácticas de laboratorio en cursos de ingeniería; sin embargo, a menudo son propietarios, caros y muy orientados a un campo de estudio concreto (por ejemplo, programación de microcontroladores, ingeniería de sistemas de control, ingeniería mecánica, robótica, etc.). En el caso de dispositivos electrónicos para formación en ingeniería, podemos distinguir las siguientes clases principales:

Kits de formación en ingeniería — Aquí se engloban todos los dispositivos sin un microcontrolador o un microprocesador que pueda ser programado por el estudiante. Proporcionan una colección de experiencias predefinidas para formación en electrónica [1],[2], sistemas de control [3], y otros campos de la ingeniería. Debido a que no ofrecen posibilidades de programación, no son adecuadas para otras áreas destacadas de la ingeniería, como sistemas en tiempo real, programación de microcontroladores, adquisición de datos, etc., y son difíciles de adaptar a ejercicios diferentes de los que traen por defecto.

Placas de entrenamiento con microcontroladores — Se trata de placas de circuito impreso (*Printed Circuit Boards*, *PCB*) que permiten al usuario conectar un microcontrolador y proporcionan una serie de experiencias básicas para aplicaciones empotradas [4],[5],[6],[7]. Hasta donde conocemos, estas soluciones no incluyen experiencias para ingeniería de sistemas de control o adquisición de datos, ni permiten conexiones a distintos sistemas alimentados externamente, como sí hace nuestra propuesta; están enfocadas a interfaces digitales y a la programación de la MCU (*Micro Controller Unit*) en sí, y suelen ser más caras.

Placas de extensión de microcontroladores — Estas placas son más baratas que las anteriores, puesto que extienden un microcontrolador ya existente (posiblemente también integrado en una pequeña placa que alimente la circuitería, como en el caso de Arduino [8]), para dotar a la placa anfitriona de alguna funcionalidad muy específica (por ejemplo, extender sus comunicaciones). Sus posibilidades formativas de extensión son, por tanto, más específicas que en el caso previo [9],[10],[11], y definitivamente más limitadas que la propuesta en este artículo.

Placas de formación de microprocesadores— Análogas a sus equivalentes con microcontroladores, pero usando procesadores mucho más potentes (como la popular Raspberry Pi [12]), por lo que son más caras que las anteriores. Son útiles fundamentalmente para aprender a programar dichas CPUs, o para la formación en sistemas operativos y, a lo sumo, ofrecen interfaces digitales con el mundo exterior, limitando por tanto la clase de experimentos que podrían llevarse a cabo en un laboratorio de ingeniería y dejando de lado cuestiones de

[1] Todos los autores pertenecen al Dpto. de Ingeniería de Sistemas y Automática, e-mail: anacm@ctima.uma.es

aprendizaje relacionadas con sistemas empotrados [13],[14].

En el mercado podemos encontrar muchos más dispositivos que los que han sido patentados. La mayoría de patentes corresponden a kits para entrenamiento en ingeniería [15],[16],[17],[18],[19],[20],[21]; sólo se han encontrado unos pocos que podrían clasificarse como placas de entrenamiento o de extensión de microcontroladores [22],[23],[24]. Una de las posibles causas es el fuerte movimiento *open-hardware* u *open-source hardware* desarrollado en la última década, que anima al usuario a construir sus propios dispositivos a partir de diseños que están disponibles públicamente [25].

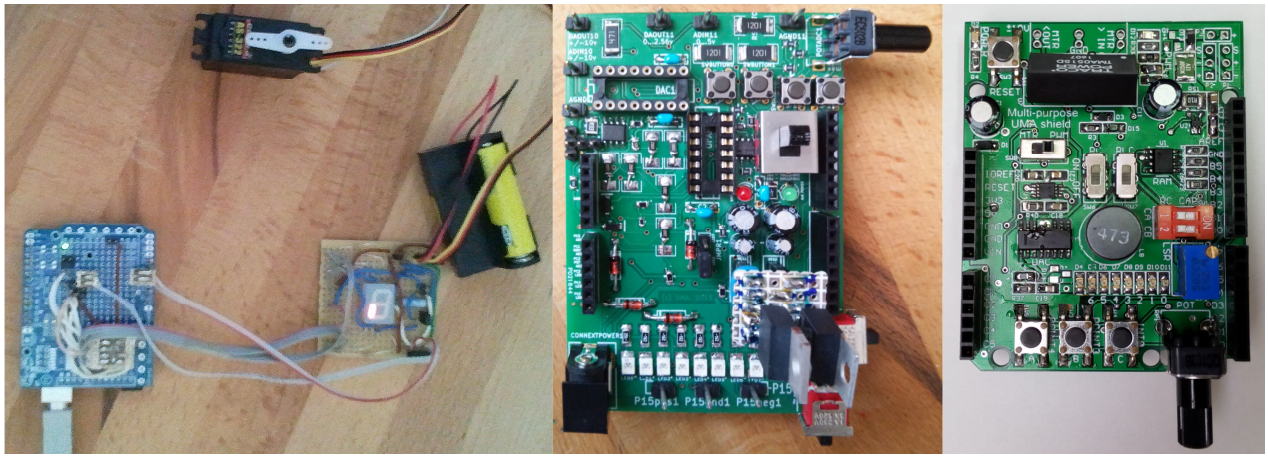

Fig. 1. Evolución del diseño de nuestra placa. a-izquierda) desde el curso académico 2010/2011 al 2014/2015; b-centro) curso académico 2015/2016; c-derecha) diseño actual

En este artículo presentamos un dispositivo que ofrece mayor funcionalidad, y más diversa, que los previamente descritos. Nuestra placa contiene conversores AD y DA, entradas/salidas digitales, medidores de consumo de servomotores, SRAM para adquisición de datos, dispositivos SPI para prácticas de comunicaciones empotradas, varios sistemas lineales e invariantes en el tiempo para ingeniería de control, y conexiones para dispositivos analógicos alimentados externamente a +/-10v; además, todo esto se logra a un coste significativamente bajo gracias a un proceso muy cuidado de diseño, selección de componentes y combinación de los mismos. Nuestro dispositivo es un *shield* para la popular placa de hardware abierto Arduino (se clasificaría como una placa de extensión de microcontrolador); por tanto, aprovecha la alimentación de Arduino, su microcontrolador empotrado (ATmega328P de 8 bits [26] en el caso del Arduino UNO [27]) y su ecosistema software, todo disponible a un precio reducido.

Desde 2010 hemos usado en asignaturas de ingeniería versiones primitivas de la placa mucho más limitadas que nuestro diseño actual, y hemos recogido numerosos experimentos y propuestas de mejora tanto por parte de alumnos como de profesores (ver figura 1). Esta experiencia acumulada se ha traducido este curso 2016/2017, en el contexto de un proyecto de innovación educativa financiado por la Universidad de Málaga (cuya información se incluye en la sección de Agradecimientos), en llevar a producción el diseño actual y fabricar 10 placas de calidad profesional, y también en usarlas intensivamente en dos cursos de ingeniería en niveles de grado y máster relativos a sistemas de ingeniería de control y sistemas empotrados de tiempo real, respectivamente; además, otras asignaturas también incluidas en titulaciones de ingeniería están utilizando la placa en sus prácticas de laboratorio este mismo curso. Esto nos ha permitido ofrecer a los estudiantes una serie de ejercicios que han cumplido nuestros requisitos y expectativas iniciales.

La estructura del artículo es la siguiente: la sección II describe la placa y sus funcionalidades; la sección III explica cómo se ha utilizado este curso en dos asignaturas diferentes de ingeniería; finalmente, en la sección IV se resumen los resultados obtenidos y se perfilan los trabajos futuros. A lo largo del texto nos referiremos al último modelo de la placa y curso académico (figura 1-c)

II. DESCRIPCIÓN DE LA PLACA

La versión actual de la placa y sus funcionalidades se muestran en la figura 2, con su esquemático correspondiente en la figura 3. Ésta consta de:

Subsistema de alimentación — Aunque nuestra placa se alimenta del Arduino (que, a su vez, puede alimentarse del PC o de un adaptador AC/DC), proporcionamos capacidad y regulación adicionales. En particular, filtrado de la alimentación, protección contra cortocircuitos causados por usuarios no experimentados, y gestión de energía para un servomotor y otros dispositivos que pueden obtener su alimentación desde nuestro *shield*.

Indicadores luminosos — Se trata de un subsistema simple de salida digital compuesto de 8 ledes de bajo consumo, fisicamente dispuestos en una línea para imitar los bits de un byte (más un led adicional para temporización y depuración). Está especialmente indicado para el aprendizaje de la programación de manipulación de bits, una necesidad fundamental en sistemas empotrados.

Botones — Hemos incluido tres botones independientes como entradas digitales de usuario. Están conectadas a señales de interrupción para el aprendizaje de sistemas de tiempo real empotrados asíncronos.

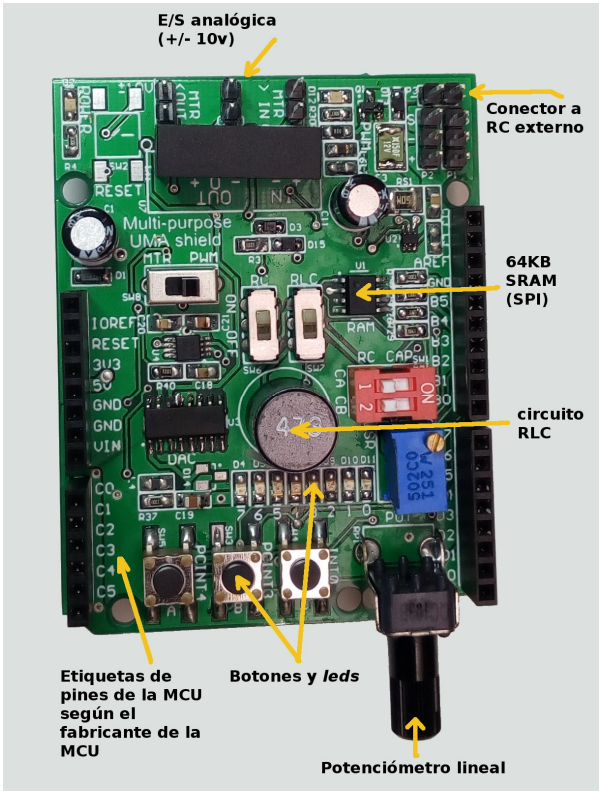

Fig. 2. Principales funcionalidades de la versión actual del *shield*.

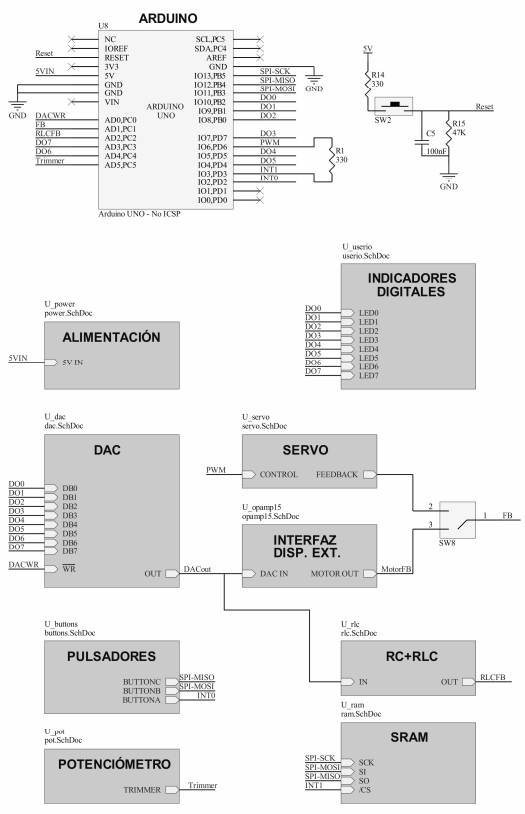

Fig. 3. Esquemático de la versión actual del *shield*.

Conversor Digital-Analógico — Este subsistema proporciona una salida analógica, un elemento que se echa de menos en Arduino, desde un bus digital compuesto de los mismos 8 bits conectados a los indicadores luminosos.

Extensión de memoria — En nuestro diseño hemos incluido 64KB de SRAM que puede emplearse para adquisición de datos (la RAM de la MCU es bastante limitada para tareas de ese tipo) y también para prácticas de comunicaciones empotradas, al estar conectada a la MCU mediante un bus SPI.

Potenciómetro lineal — Un pequeño potenciómetro lineal rotacional se ha añadido al *shield* para lecturas analógicas simples. Está directamente conectado a una entrada analógica de la placa Arduino.

Subsistema RC+RLC — Éste es quizás el subsistema del *shield* más innovador y potente: consiste en un circuito eléctrico de dos mallas (una malla RC, es decir, resistencia+condensador, y otra malla RLC, resistencia+bobina+condensador) que, mediante una pareja de interruptores, puede configurarse como un sistema lineal SISO (*single-input, single-output*) de primer, segundo o tercer orden. Sus componentes eléctricos se han escogido para permitir que la MCU de Arduino muestree todas las características de la respuesta correctamente. Este subsistema está conectado al conversor digital-analógico, que le proporciona la entrada, y a una entrada analógica de Arduino, que recibe su salida. Con él pueden realizarse ejercicios de adquisición de datos, modelado de sistemas y control automático.

Interfaz analógico externo — Este subsistema transforma el rango [0,5]v de las señales analógicas del *shield* (es decir, aquéllas producidas por el conversor digital-analógico) hacia y desde un rango de [-10,+10]v, por lo que es útil tanto para adquisición como para el control de dispositivos externos (que pueden ser alimentados externamente), como motores DC.

Interfaz de servomotor externo — Este subsistema permite al usuario conectar servomotores RC estándar o dispositivos controlables mediante señales PWM generadas por Arduino. Además, se han insertado una resistencia *shunt* y un amplificador en la pista de alimentación para poder medir el consumo de dichos dispositivos en tiempo real; estas medidas quedan disponibles en una de las entradas analógicas de Arduino.

Todas las líneas de la MCU que se han usado para los propósitos arriba enumerados también están disponibles en conectores que imitan a los de Arduino, por lo que pueden usarse para otras aplicaciones. Muchas de estas líneas están compartidas entre subsistemas, para así reducir el coste de fabricación y proporcionar interconexiones diversas. Como ya se ha dicho, nuestro diseño está basado en la placa Arduino UNO, es decir, hereda las capacidades y limitaciones de una MCU ATmega328P; aunque el *shield* podría usarse con otros modelos de Arduino gracias a la compatibilidad en la

distribución de sus patillas, hay que tener en cuenta lo siguiente:

- Nuestro *shield* requiere señales de 5v y una corriente mínima de 500mA (con 1A se logran mejores resultados).
- Los componentes electrónicos del subsistema RC+RLC se han escogido para que el ATmega328P pueda muestrearlos correctamente con un reloj de 16MHz.
- Hemos replicado en nuestro *shield* el etiquetado original de la distribución de patillas de esa MCU, para facilitar el aprendizaje de la programación de ese microcontrolador (algo que no existe en la placa Arduino).

La Tabla I enumera los principales campos de ingeniería para los que el *shield* ofrece experimentos de laboratorio, y cómo pueden montarse éstos combinando uno o más de los subsistemas descritos anteriormente.

TABLA I. CAMPOS DE INGENIERÍA A LAS QUE SE DESTINA EL *SHIELD*.

Campo de ingeniería	*Subsistema*
Programación de sistemas empotrados	-Indicadores luminosos -Botones -Potenciómetro -Servomotor
Sistemas empotrados de tiempo-real, Automatización	-Indicadores luminosos -Botones -Memoria -Conversor D/A -Servomotor -Int. Analógico
Adquisición de datos, Ingeniería de sistemas de control, Modelado de sistemas	-Botones -Memoria -Potenciómetro -Conversor D/A -RC+RLC -Int. Analógico
Ingeniería eléctrica	-Botones -Memoria -Potenciómetro -Conversor D/A -RC+RLC -Int. Analógico -Servomotor
Robótica	-Conversor D/A -Servomotor -Int. Analógico

Puesto que describir con detalle todos los subsistemas de nuestro *shield* requeriría un artículo mucho más extenso, en lo que resta de sección nos vamos a centrar en cómo hemos diseñado e implementado la aportación más compleja e innovadora de la placa: el subsistema RC+RLC, que además es la parte que puede usarse en más áreas de ingeniería. En la sección III explicaremos los ejercicios prácticos que hemos implantado con este y otros subsistemas para dos asignaturas de ingeniería concretas.

La figura 4 muestra un diagrama de bloques detallado del subsistema RC+RLC. Nuestro objetivo ha sido incluir varios sistemas continuos SISO lineales e invariantes en el tiempo (*linear and time-invariant, LTI*) que permitan preparar diferentes ejercicios de laboratorio en asignaturas de sistemas empotrados de tiempo real, adquisición de datos, modelado de sistemas y sistemas (empotrados) de control. Hasta donde sabemos, no existe una extensión de microcontrolador como esta en el mercado de la electrónica educativa.

La entrada de cualquiera de los sistemas LTI será la señal analógica producida por el conversor D/A del *shield*, que estará en el rango [0,5]v, tras almacenarse temporalmente en un amplificador operacional que proporciona al menos 100mA. Esto se hace así para evitar el drenado excesivo de corriente de otras partes del *shield* distintas del subsistema de potencia. Por otro lado, la salida está conectada a la patilla de entrada analógica AD2 del ATmega328P.

En la mayoría de cursos de modelado e ingeniería de control, los estudiantes deben estudiar sistemas LTI de primer orden, segundo orden y órdenes superiores. Puesto que no se abordan simultáneamente, hemos diseñado un subsistema reconfigurable con el que se puede escoger sólo una de esas opciones (mediante los interruptores "Selector_1" y "Selector_2" mostrados en la figura 4). Más concretamente, nuestra circuitería tiene dos mallas eléctricas: una de primer orden y otra de segundo orden; los selectores permiten activar la primera, la segunda, o las dos en serie (obteniéndose así un sistema de tercer orden). También sería posible una cuarta configuración (cortocircuito), que permitiría muestrear directamente la señal con la que se estimula el circuito.

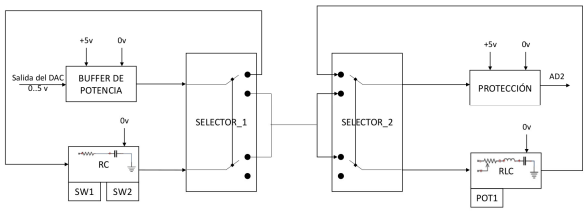

Fig. 4. Diagrama de bloques del subsistema RC+RLC.

Los principales problemas al diseñar esta parte del *shield* han sido: i) encontrar una topología para cada malla que minimice el consumo de energía, ii) elegir componentes lo más baratos posible pero que se ajusten al ATmega328P @16MHz, cuyo período mínimo de muestreo es de 104 μs, para que así pueda muestrear adecuadamente la salida de cualquiera de los sistemas LTI, y iii) proporcionar flexibilidad suficiente para que cada LTI pueda variar su comportamiento dentro de ciertos límites (por ejemplo, cambiar la localización de los polos en el sistema de tercer orden para que se asemeje o diferencie del comportamiento de un sistema de segundo orden).

Respecto a i), la malla de primer orden podría ser un circuito LR o RC; el primero presenta mayores consumos en estado estacionario, por lo que hemos escogido la segunda opción. La malla de segundo orden requiere necesariamente combinar un condensador y una inductancia para lograr dos polos, por lo que hemos elegido un circuito RLC clásico.

El proceso más costoso en tiempo ha sido ii), es decir, seleccionar los componentes para las dos mallas. Nótese que en éstas tanto el condensador como la inductancia tienen resistencias internas que deben tenerse en cuenta,

y que los componentes disponibles en el mercado no tienen valores arbitrarios y se fabrican con tolerancias, disipaciones de energía y formas y tamaños que tienen un impacto significativo en nuestra placa.

Hemos tratado este problema primero en simulación, sin considerar los aspectos prácticos mencionados. Para ello, hemos establecido las siguientes restricciones: los sistemas de segundo y tercer orden deben mostrar un comportamiento subamortiguado con un tiempo de pico lo suficientemente largo como para permitir a la MCU muestrearlo un número mínimo de veces; también deben mostrar un cierto número de oscilaciones claramente distinguibles con la resolución del conversor analógico/digital de la MCU; el sistema de primer orden debe tener una constante de tiempo lo suficientemente grande para que la MCU lo pueda muestrear un mínimo número de veces; finalmente, el pico de corriente (o sea, el consumo) de todos los sistemas debe mantenerse bajo un cierto umbral.

La idea general es encontrar, en la región del espacio (R_1,C_1,R_2,C_2,L_2) que satisface todas estas restricciones, los puntos óptimos respecto a las limitaciones prácticas del mercado. Esto puede hacerse usando una serie de métodos y herramientas de búsqueda óptima con restricciones; sin embargo, puesto que nuestras dos mallas son desacopladas, hemos realizado una búsqueda exhaustiva más simple: primero en el sistema de segundo orden (R_2,C_2,L_2), y luego en los de primer (R_1,C_1) y tercer orden. El espacio de valores de componentes se ha rastreado con una resolución compatible con la de los componentes comerciales.

Tras esta búsqueda, hemos estrechado el margen de elección con una búsqueda (manual) del mercado para componentes similares a los teóricos, siendo la más restrictiva la inductancia L_2, debido al espacio disponible en la placa y a su necesidad de apantallamiento[2]. Nuestra elección final ha sido la siguiente:

- R_1=100Ω, 1%, ¼W;
- C_1={**2µF**, **4.7µF**, 6.7µF, **15µF**, 17µF, 21.7µF}, 10% (en negrita, los componentes instalados en paralelo en la placa; cualquiera de los demás valores puede obtenerse usando los interruptores en la placa);
- R_2= [0,250]Ω, 10%, ½W (resistencia variable);
- L_2=47mH, 5%, 52Ω, 13mA, apantallada;
- C_2=1µF, 2%.

El problema iii) de permitir cierta variación en los comportamientos del primer, segundo y tercer orden se ha resuelto mediante dos componentes variables: C_1 y R_2. El sistema de primer orden consigue distintas constantes de tiempo gracias a la variación de C_1 (mayor conforme C_1 aumenta); el sistema de segundo orden cambia su factor de amortiguación cuando R_2 varía (mayor conforme R_2 aumenta); finalmente, el sistema de tercer orden se asemeja más a un sistema de segundo orden cuando C_1 disminuye.

[2] En realidad, el componente que hemos escogido impide apilar más *shields* sobre el nuestro, como suele ocurrir con otras placas de extensión de Arduino.

III. UTILIZANDO EL *SHIELD* EN ASIGNATURAS DE INGENIERÍA: IMPLANTACIÓN Y RESULTADOS

Como se mencionó en la introducción, durante el curso 2016/2017 hemos usado intensivamente nuestra placa en dos asignaturas de ingeniería del primer cuatrimestre; actualmente, también se está utilizando en otras cinco asignaturas, pero estas aún no han sido evaluadas (sus exámenes son en junio), por lo que en este artículo nos centramos en las primeras. La lista de asignaturas en las que la placa ha sido usada o se está usando, junto con el número de alumnos matriculados, créditos ECTS y otros datos, se recopila en la tabla II.

TABLA II. ASIGNATURAS DE INGENIERÍA QUE UTILIZAN EL *SHIELD* (* – YA EVALUADAS; † – OBLIGATORIA EN SU PLAN DE ESTUDIOS).

Asignatura	*Curso / ECTS / Nº alumnos*	*Titulación*
Tiempo Real para Sistemas Mecatrónicos * †	1º / 5 / 29	Máster en Ingeniería Mecatrónica
Control por Computador * †	4º / 6 / 18	Grado en Ingeniería de Computadores
Sistemas de Tiempo Real †	3º / 6 / 24	Grado en Ingeniería de Computadores
Programación de Robots	3º-4º / 6 / 21	Varios grados de Ing. Informática
Sistemas de Información para la Industria	4º / 6 / 14	Grado e Ingeniería Informática
Automática †	2º / 6 / 73	Grado en Ingeniería en Tecnologías Industriales
Control Automático †	2º / 6 / 68	Grado en Ingeniería de la Salud

A. Caso de estudio: Tiempo Real para Sistemas Mecatrónicos

El temario de esta asignatura está enfocado en su mayor parte a sistemas empotrados de tiempo real [28], es decir, a la programación de tiempo real para microcontroladores y al diseño hardware/software e implantación de sistemas de tiempo real de pequeña escala. Hemos usado 10 placas para esta asignatura, lo que implica que los estudiantes trabajan en pareja, y algunos en tríos. Para el próximo curso queremos producir más placas para reducir el tamaño de grupo a dos a lo sumo.

Hemos desarrollado 2 ejercicios prácticos con el *shield* destinados a ser realizados en el laboratorio al final de la parte de la asignatura relativa a sistemas de tiempo real de pequeña escala (aproximadamente, a los 2/3 del semestre completo). En ambos ejercicios los estudiantes deben resolver el mismo problema: en el primero, a través de *polling* activo de señales, es decir, sin interrupciones; en el segundo, mediante un diseño

asíncrono guiado por eventos que debe proveer garantías de tiempo real estricto.

El problema en sí consiste en controlar un servomotor RC con una señal PWM para que alcance cierta posición angular (y la mantenga), y después muestrear su consumo de energía periódicamente en los períodos de nivel bajo de dicha PWM (cuando la circuitería interna del motor se alimenta para moverse), especialmente en el caso de que algún obstáculo fuerce la detención del servo, disparando su consumo. Este ejercicio exige: generar las señales PWM apropiadas; detectar eventos (flanco de bajada de la señal para comenzar a muestrear; fin de las conversiones analógico-digitales para lanzar el próximo muestreo; etc.); adquirir y procesar datos analógicos (el consumo de energía del servo); satisfacer requisitos de tiempo real para hacer estas tareas; mostrar resultados y depurar usando el hardware disponible.

Para programar la MCU de la placa Arduino los estudiantes utilizan la *suite* Atmel Studio [29] y el lenguaje de programación C [30]; este IDE también incluye un simulador muy potente del microcontrolador ATmega328P y puede mostrar el código ensamblador producido por el compilador, etiquetado de forma apropiada con las instrucciones C.

De todo el *shield*, los alumnos utilizan el subsistema de alimentación (para alimentar el servo), los indicadores luminosos (para mostrar el resultado del procesamiento del consumo de energía, depurar el programa, y verificar algunos requisitos de tiempo real), así como el interfaz externo del servomotor (para proporcionar al servo la señal PWM). El procesamiento de las muestras del consumo de energía es sencillo en este ejercicio: simplemente tienen que obtener el valor máximo y mostrarlo con los ledes; por tanto, no es necesario utilizar el subsistema de extensión de memoria. Sin embargo, planeamos usar la conexión SPI a ese subsistema en la asignatura de Sistemas en Tiempo Real que se está impartiendo actualmente.

Todos los requisitos de estos ejercicios podrían cumplirse con una placa Arduino y una *protoboard* en lugar de nuestro *shield*, pero también supondría montar toda la circuitería externa a mano: leds, resistencias y, lo que es más importante, la *shunt* y el amplificador operacional para la monitorización de consumo. Con nuestro *shield* los estudiantes tienen todo esto conectado y funcionando, y por tanto, pueden dedicarse a diseñar, implantar, simular y depurar la aplicación. Además, los indicadores luminosos facilitan la visualización de números digitales para comprobar si los resultados del procesamiento de las lecturas de consumo de energía; en años anteriores usábamos un módulo led de 7 segmentos, mucho más difícil de interpretar para los alumnos que un array de ledes a la hora de representar un número binario de 8 bits. Hemos comprobado que estos indicadores son muy útiles también para que los alumnos aprendan la manipulación de bits en C, ya que los ledes están mapeados en patillas digitales no adyacentes de la MCU. Finalmente, nuestro *shield* tiene una conexión eléctrica entre la patilla donde se genera la PWM y la patilla correspondiente a la interrupción INT1 de la MCU, por lo que los estudiantes pueden escoger detectar el flanco de bajada de la PWM bien por software o por hardware.

Considerando todos estos aspectos, nuestra experiencia ha sido muy satisfactoria al usar el *shield* para resolver estos ejercicios: los estudiantes están más motivados, tienen que aplicar de forma práctica, en un entorno hardware/software amigable, los conceptos enseñados en las clases teóricas y, en general, obtienen mejores resultados en la parte práctica de la asignatura. La figura 5 muestra la evolución de las calificaciones obtenidas en estas prácticas desde que aplicamos por primera vez una versión muy preliminar de una circuitería de extensión de Arduino (año académico 2010/2011, ver figura 1a) hasta el curso actual, todo en una escala 0,0-10,0. Nótese que en el curso 2015/2016 usamos un prototipo que ya incluía la resistencia *shunt* y otras partes del *shield* actual (figura 1b). También es reseñable que, debido a las elevadas varianzas en algunos años y el relativamente reducido número de estudiantes, no es posible realizar un ANOVA para evaluar estadísticamente las diferencias observables entre medias; sin embargo, visualmente resulta clara la progresiva mejora en las notas medias en los ejercicios descritos en esta sección con el uso de sucesivas versiones mejoradas del *shield*.

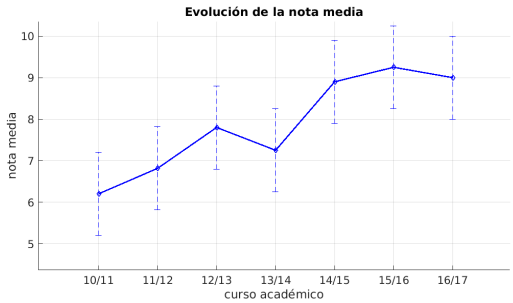

Fig. 5. Evolución de las calificaciones de los estudiantes en los ejercicios relativos al *shield* en la asignatura Tiempo Real para Sistemas Mecatrónicos.

Aparte del análisis de las calificaciones, hemos recabado la opinión de los alumnos antes y después de la realización de las prácticas en el laboratorio con el *shield*. La figura 6 muestra el resultado de algunas de las preguntas que hemos planteado. El primer histograma muestra la evolución percibida por los alumnos sobre sus conocimientos de Arduino que, como se observa, es positiva; el segundo histograma recoge la evolución que han percibido en sus conocimientos de lenguaje C, que para ellos han mejorado; finalmente, la última gráfica muestra la opinión sobre el material que se les ha proporcionado, la atención del profesorado a la hora de resolverles cuestiones relativas a las prácticas con el *shield*, su grado de satisfacción con las prácticas y su influencia a la hora de comprender mejor la teoría de la asignatura.

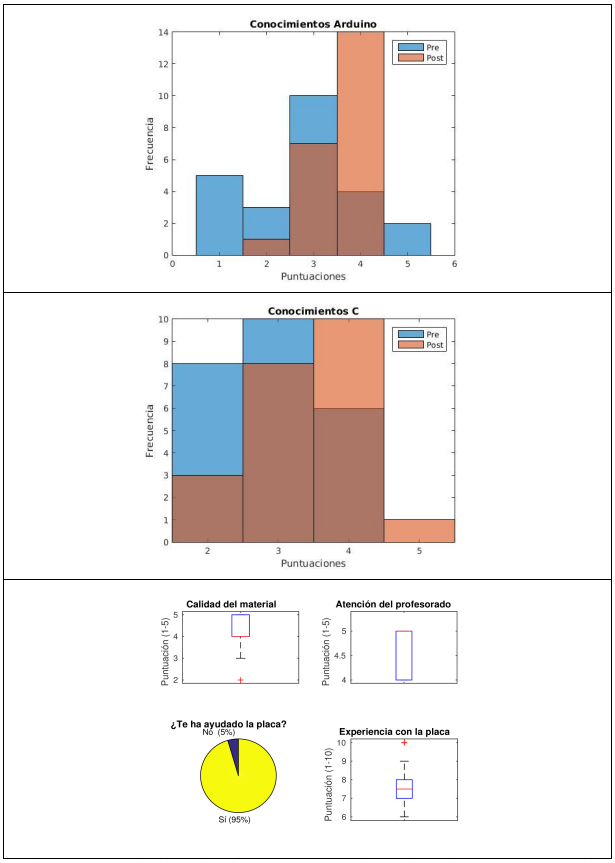

Fig. 6. Resultados de las encuestas de valoración pasadas a los alumnos. a) arriba: evolución de la percepción de sus conocimientos de Arduino; b) centro: evolución de la percepción de sus conocimientos de C; c) abajo: valoración de diferentes aspectos de las prácticas con el *shield*.

B. Caso de estudio: Control por Computador

Esta asignatura ha sido la que más intensivamente ha usado el *shield* hasta ahora, tanto en el número de ejercicios prácticos como en las partes del *shield* implicadas en ellos. La asignatura se centra en sistemas LTI, con una primera parte dedicada a su modelado e identificación y una segunda parte donde se introducen métodos básicos y aspectos importantes del control directo por computador [31]. Este año hemos empleado el *shield* sólo en la primera parte (para la segunda empleamos robots Lego Mindstorms NXT [32]). Nuestro objetivo es extender el uso del *shield* a la segunda parte el año próximo.

Hemos desarrollado 6 ejercicios prácticos para la primera parte de esta asignatura, que brevemente describimos a continuación:

Introducción a la adquisición de datos — El primer ejercicio es una introducción tanto a la programación de Arduino como a la adquisición de datos con nuestro *shield*. Los estudiantes tienen que adquirir datos en tiempo real desde el potenciómetro, almacenarlos en la SRAM, y posteriormente volcarlos a través de la conexión serie entre el Arduino y Matlab en un formato de texto válido para ser leído como una matriz para su dibujo y análisis. Salvo el RC+RLC y el subsistema Digital-Analógico, que se usan en ejercicios posteriores, los estudiantes deben aprender a usar diferentes partes del *shield*, incluyendo los botones (para lanzar la adquisición de datos) y los indicadores luminosos (para señalar el estado interno del programa). Este ejercicio sería válido también para otras asignaturas de sistemas en tiempo real o adquisición de datos.

Sistemas LTI eléctricos — En esta práctica los alumnos deben usar el subsistema Digital-Analógico para proporcionar al subsistema RC+RLC un voltaje de entrada aceptable y leer de dicho subsistema el voltaje de salida resultante. En primer lugar, deben calibrar linealmente estos componentes de entrada y de salida usando un multímetro, a través del *bypass* directo de las mallas RC+RLC (se activa mediante los selectores 1 y 2 mostrados en la figura 4). Debe hacerse así porque, debido a las tolerancias de los fabricantes, las calibraciones pueden variar de un *shield* a otro. Esto, por sí mismo, constituye un interesante ejercicio para asignaturas de adquisición de datos; tras nuestra experiencia este curso, consideramos que es mejor dar esta parte resuelta a los alumnos de ingeniería de control, ahorrándoles un tiempo valioso para centrarse en el modelado eléctrico en sí. De cualquier forma, tras la calibración, el ejercicio continua pidiendo a los estudiantes que adquieran los datos del RC+RLC configurado como un sistema de tercer orden al que se le proporciona una entrada escalón unitaria. Los datos se trasvasan a Matlab como en el ejercicio introductorio y los estudiantes deben comparar su comportamiento con el teórico dado por un modelo matemático del sistema.

Sistemas LTI electro-mecánicos — Los objetivos de este ejercicio son análogos a los del sistema eléctrico, pero usando un motor DC. Para ello, usaremos el interfaz analógico externo del *shield*, que se conecta a los motores DC PHYWE disponibles en el laboratorio [3]. Éstos trabajan en el rango +-10v de señales de entrada/salida; también incluyen sensores de velocidad y posición. Como en el ejercicio previo, primero los alumnos deben calibrar el interfaz analógico externo, tanto para entradas como para salidas, de sus *shields* (para el año próximo les proporcionaremos esta parte ya resuelta). Después, escriben un programa para el Arduino que envía un escalón de 5v al motor y lee el sensor de velocidad. También deben diseñar un modelo teórico en Matlab para ese sistema electro-mecánic, y comparar ambos comportamientos. La figura 7 muestra la configuración de laboratorio para esta práctica.

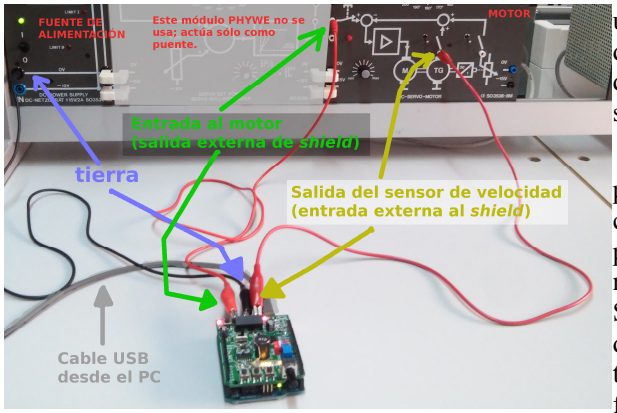

Fig. 7. Montaje en laboratorio para el ejercicio electro-mecánico de la asignatura Control por Computador. Arriba, el motor DC PHYWE, alimentación y sensores; abajo, el *shield* conectado a un Arduino y al PHYWE (a través del interfaz analógico externo).

Respuesta temporal de sistemas de primer orden — En este ejercicio los estudiantes leen los datos de nuevo desde el sistema eléctrico RC+RLC del *shield*. El objetivo ahora es activar únicamente el RC, la malla de primer orden, e identificar sus parámetros teóricos a partir de la respuesta temporal. Los alumnos deben proponer variaciones en los valores nominales de los componentes del circuito que producen ciertos efectos en esa respuesta.

Respuesta temporal de sistemas de segundo orden — Ejercicio análogo al anterior; en este caso los estudiantes trabajan con el circuito RLC, la malla de segundo orden. También deben modificar la posición de la resistencia variable y estudiar los efectos sobre el factor de amortiguamiento del sistema.

Respuesta temporal de sistemas de orden superior — En esta prácticas los estudiantes configuran el sistema RC+RLC para obtener un comportamiento de tercer orden (es decir, activan ambas mallas en serie). Deben variar el valor del condensador de la primera malla y comprobar cómo puede llegar a aproximarse el sistema resultante a un segundo orden, tanto en la realidad como con un modelo matemático teórico que han debido desarrollar para esa aproximación.

Como se observa, todos los elementos del *shield* se usan intensivamente en estas prácticas. En el futuro, planeamos llevar a cabo ejercicios de control directo con el subsistema RC+RLC, lo que no requiere ningún componente adicional. Como ocurría con la asignatura de sistemas en tiempo real, estos ejercicios implican programar la placa Arduino en lenguaje C, lo que hacemos a través del IDE ATmel Studio; sin embargo, en este caso proporcionamos a los estudiantes algunas rutinas básicas (en forma de cabeceras en C) para la entrada/salida digital -aquí no buscamos que se centren en los entresijos de la manipulación de bits-, y para acceder la SRAM, -los objetivos de la asignatura tampoco incluyen el manejo de la comunicación SPI-. Resumiendo, ninguno de los ejercicios anteriores podría implantarse fácilmente en esta asignatura sin el *shield*; en años anteriores hemos realizado ejercicios similares únicamente con Matlab, mediante una librería de desarrollada por nosotros para una tarjeta de adquisición de datos profesional, como se describe en [33]. Esa solución es mucho más cara que el uso del *shield*.

En esta asignatura disponemos de menos información para evaluar la mejora cuando se usa el *shield* que en el caso de estudio de tiempo real. Esto es debido a que pertenece a un grado de Ingeniería de Computadores que no existía antes del Espacio Europeo de Educación Superior; más concretamente, esta asignatura se oferta desde el curso académico 2013/2014, y algunos años ha tenido muy pocos estudiantes. Aún así, mostramos en la figura 8 un diagrama de cajas de las calificaciones obtenidas por los estudiantes en los ejercicios prácticos explicados anteriormente (excepto el primero, que no estaba disponible antes del *shield*). No hay suficientes datos estadísticos y hay demasiada varianza como para aplicar un test t-student, pero cualitativamente es fácil ver la mejora en calificaciones altas; la nota media en el curso 2016/2017 (en el que se ha usado el *shield*) es también ligeramente mayor: 7,5 frente a 6,8 del año anterior (en una escala de 0,0 a 10,0).

La figura 9 muestra el resultado de encuestas semejantes a las comentadas para la asignatura de Tiempo Real para Sistemas Mecatrónicos. Los histogramas relativos a la evolución de conocimientos de Arduino y C vuelven a mostrar una tendencia positiva. Respecto al conjunto de preguntas de la gráfica inferior, vemos que los resultados son un poco más bajos. En general, los estudiantes están satisfechos con la nueva placa este año, pero han señalado algunos problemas que se deben fundamentalmente a nuestro diseño de las prácticas: sobre todo, la falta de tiempo debido a la necesidad de calibrar la placa para cada grupo; también, debido a que los componentes los hemos montado nosotros mismos en las placas, hemos detectado una serie de fallos en algunos subsistemas que han supuesto un tiempo extra para arreglarlos y también han producido frustración en los alumnos al intentar hacer los ejercicios.

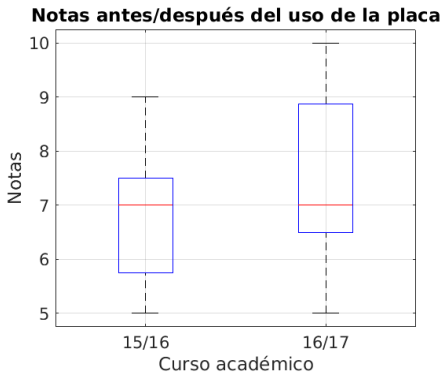

Fig. 8. Calificaciones de los estudiantes en los ejercicios relativos al *shield* en la asignatura Control por Computador.

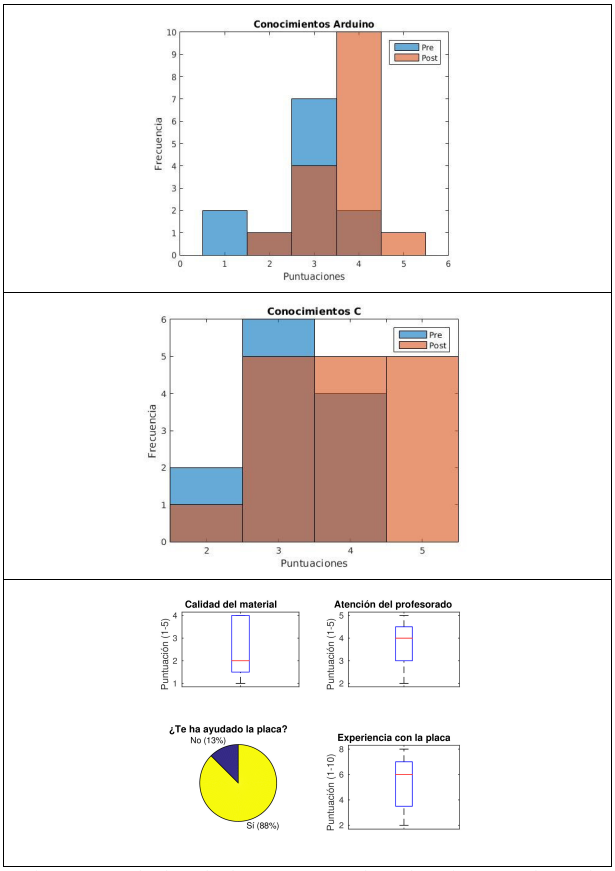

Fig. 9. Resultados de las encuestas de valoración pasadas a los alumnos. a) arriba: evolución de la percepción de sus conocimientos de Arduino; b) centro: evolución de la percepción de sus conocimientos de C; c) abajo: valoración de diferentes aspectos de las prácticas con el *shield*.

IV. CONCLUSIONES Y TRABAJOS FUTUROS

En este artículo hemos presentado una tarjeta de expansión para microcontroladores novedosa que, con un bajo coste de fabricación, proporciona un número de funcionalidades para formación mucho más numeroso que otros dispositivos existentes. Hemos incluido una breve descripción de la placa, detalles concretos de su subsistema más complejo y potente, y resultados de utilizarla en dos asignaturas diferentes de ingeniería.

Actualmente estamos explorando todas las posibilidades de la placa en otras asignaturas de ingeniería, cubriendo un amplio rango de campos: adquisición de datos, control automático, sistemas en tiempo real, robótica, etc. También hemos recopilado una serie de problemas menores que hay que resolver en el diseño y la fabricación. Nuestra intención es mejorar el diseño (por ejemplo, haciendo los botones independientes del bus SPI, ya que comparten esas líneas) y fabricar una nueva revisión de las placas. También estamos trabajando en la escritura en C y Matlab de un software de diagnóstico completo que, a través de la conexión de la placa al PC, nos permita detectar más rápida y ajustadamente posibles defectos en su fabricación o uso. Finalmente, estamos desarrollando librerías para el uso del *shield* desde el popular IDE de Arduino.

En el futuro queremos alcanzar un estado estacionario en el uso de las placas en las asignaturas que ya están utilizándola, añadir nuevos ejercicios prácticos, y extender su uso a más asignaturas. Igualmente, se realizará un análisis más exhaustivo de los datos, de manera que podamos detectar elementos que influyan en las calificaciones además del uso del shield, como el número de alumnos en el laboratorio, la formación inicial de los mismos, etc.

AGRADECIMIENTOS

Este trabajo ha sido financiado por la Universidad de Málaga (España) mediante el proyecto de innovación educativa PIE-15-093 "*Innovación en el trabajo en laboratorio de una diversidad de asignaturas de ingeniería mediante el diseño y aplicación de una extensión de la plataforma de hardware abierto Arduino*", perteneciente a la Convocatoria 2015-2017.

Los autores también desean dar las gracias a todos los alumnos que han intervenido a lo largo de diferentes cursos académicos en esta experiencia. Sus ideas, aportaciones, sugerencias y críticas nos han sido de gran utilidad.

Finalmente, agradecemos la labor y los comentarios de los revisores anónimos, que han permitido mejorar el artículo final.

REFERENCIAS

[1] SparkFun (2014). SparkFun inventor kit. https://www.sparkfun.com/products/12060. Consulted 20th sept. 2016.
[2] Snapcircuits (2016). Circuits for training in electronics. http://www.snapcircuits.net/. Consulted 20th sept. 2016.
[3] Lucas-Nuelle (2014). Lucas-Nuelle company page. http://www.lucas-nuelle.com. Consulted 16th sept. 2014.
[4] Reach Electronics (Kickstarter) (2014). Portable Dual Arduino (TM) Micro XPlorerBoard. https://www.kickstarter.com/projects/1576747460/portable-dual-arduino-micro-xplorerboard?lang=es. Consulted 20th sept. 2016.
[5] Parallax Inc. (2014). Board of Education Shield. https://www.parallax.com/product/35000. Consulted 20th sept. 2016.
[6] Velleman Kits (2003). PIC Programmer & Experiment Kit K8048. http://www.apogeekits.com/pic_programmer_k8048.htm. Consulted 20th sept. 2016.
[7] Parallax Inc. (2014). Basic STAMP Discovery Kit. https://www.parallax.com/product/27807. Consulted 20th. sept. 2016.
[8] Arduino (2016). Official website of Arduino/Genuino, https://www.arduino.cc/. Consulted 12th sept. 2016.
[9] Dilligent Inc. (2014). Analog Shield. http://store.digilentinc.com/analog-shield-high-performance-add-on-board-for-the-arduino-uno/. Consulted 27th sept. 2016.
[10] Visgence Inc. (2013). Power DAC Shield. https://www.tindie.com/products/visgence/power-dac-shield/. Consulted 20th sept. 2016.
[11] Numato Labs. (2015). Digital and Analog IO Expander Shield. http://numato.com/digital-and-analog-io-expander-shield/. Consulted 20th sept. 2016.
[12] Raspberry Pi Foundation (2016). Official website of Raspberry Pi. https://www.raspberrypi.org/. Consulted 20th sept. 2016.
[13] Gertboard (2012). Gertboard for Raspberry Pi. http://es.farnell.com/gertboard/gertboard/assembled-gertboard-for-raspberry/dp/2250034. Consulted 20th sept. 2016.
[14] Piface (2013). Piface I/O board for Raspberry Pi. http://es.farnell.com/piface/piface-digital/i-o-expansion-board-for-raspberry/dp/2218566. Consulted 20th sept. 2016.
[15] Sony Corp. (1985). Interactive teaching apparatus. Code US4812125A.
[16] NIDA Corp. (1978). Electronic teaching and testing device. Code US4213253A.
[17] K. Jensen (1982). Simulator systems for interactive simulation of complex dynamic systems. Code US4464120A.

[18] R. M. Kuczewski (1996). Cooperative/interactive learning system for logic instruction. Code US5868575A.
[19] E. Z. Gabriel (1979). Educational analog computer laboratory. Code US4315320A.
[20] E. Z. Gabriel (1974). Electronic analog computers. Code US3996457A.
[21] C. Durukan (2014). Training and experiment system supported by an animation based full simulation method. Code WO2015112103A1.
[22] Ma-Wang (2013). Electronic technology experimental box for classroom teaching. Code CN104424836A.
[23] Dan (2010). Open and modular singlechip teaching and learning experimental device. Code CN 201765731 U.
[24] Z. Yufu (2014). Embedded single-chip microcomputer application technology project training system. Code CN 103646585 A.
[25] J. M. Pearce (2013). *Open-Source Lab: How to Build Your Own Hardware and Reduce Research Costs*. Elsevier, ISBN 9780124104624.
[26] Atmel (2016). Official website of the ATmega328P microcontroller, http://www.atmel.com/devices/atmega328p.aspx. Consulted 12th sept. 2016.
[27] Arduino (2016). Official website of the Arduino/Genuino UNO, https://www.arduino.cc/en/Main/ArduinoBoardUno. Consulted 12th sept. 2016.
[28] Laplante P.A. (2004), *Real-Time Systems Design and Analysis,* IEEE Press 0-471-22855-9.
[29] Atmel (2016). Official website of the Atmel Studio, http://www.atmel.com/tools/atmelstudio.aspx. Consulted 12th sept. 2016.
[30] S. Prata (2004). *C Primer Plus*. Sams, ISBN 0-672-32696-5.
[31] N. S. Nise (2011). *Control Systems Engineering*. International Student Version, Sixth Edition, Wiley, ISBN 978-0-470-64612-0.
[32] A. Cruz-Martín, J.A. Fernández-Madrigal, C. Galindo, J. Gonzalez-Jimenez, C. Stockmans-Daou, J.L. Blanco, *"A Lego Mindstorms NXT approach for teaching at data acquisition, control systems engineering and real-time systems undergraduate courses"*, Computers & Education, vol. 59, no. 3, 2012.
[33] J. A. Fernández-Madrigal (2014), *"Using Matlab as a tool for focusing the lab work in engineering courses"*, 7th International Conference of Education, Research and Innovation (ICERI, 2014), Seville (Spain).