Sobre la utilización de sistemas embebidos para la enseñanza de la programación en una carrera de Ingeniería Electrónica

José L. Simón, Nora Blet, Cristina Bender, Rodolfo Recanzone, José I. Sosa, Andrea Torres
Facultad de Ciencias Exactas, Ingeniería y Agrimensura, Universidad Nacional de Rosario
{jlsimon, nblet, bender, mikerrr, jisosa, atorres}@fceia.unr.edu.ar

Resumen

Como los sistemas embebidos continúan incrementando su número y complejidad, la currícula de una carrera de Ingeniería Electrónica debe ocuparse de las destrezas exigidas por la programación embebida y necesaria para sus graduados. Hoy día se reconoce que el manejo explícito de la dimensional temporal en el software es un concepto fundamental que debe introducirse en forma temprana en dicho plan de estudios. Como parte de la Reforma Curricular de una carrera de Ingeniería Electrónica, la cátedra de una asignatura introductoria a la informática que utiliza lenguajes de programación C y C++, está intentando introducir algunos cambios principalmente en torno a la idea de utilizar un microcontrolador como plataforma experimental. En particular, se está considerando la plataforma *open source* Arduino la cual, aparte de su bajo costo expone a los alumnos novatos a la suficiente complejidad y desafíos propios de la programación embebida. Este trabajo describe los primeros pasos dados en este proceso de transformación.

Palabras clave: Sistemas Embebidos, Programación Introductoria, Arduino, Ingeniería Electrónica, Educación.

Motivación

La adquisición de competencias en el uso de lenguajes de programación se transformó en un requisito excluyente para la formación de grado de los futuros ingenieros electrónicos, impulsada fundamentalmente por la omnipresencia de dispositivos programables basados en microprocesadores y micro-controladores en el campo de la Ingeniería Electrónica actual. Así lo refleja la mayoría de los planes de estudio de la carrera, en universidades de América, Asia y Europa [1], [22], en los cuales se hace especial hincapié en los lenguajes de programación C y C++, el uso de entornos integrados de desarrollo de software y, metodologías de diseño, en particular la Orientación a Objetos [23].

En Argentina, el Consejo Federal de Decanos de Ingeniería (CONFEDI) ha establecido en su documento "Desarrollo de Competencias Genéricas" de 2006 [2] la importancia de la inclusión de los temas mencionados para la formación del ingeniero, y la Resolución 1232/01 del Ministerio de Educación, Ciencia y Tecnología [24] establece contenidos mínimos indispensables para los fundamentos de informática.

En la carrera de Ingeniería Electrónica de la Facultad de Ciencias Exactas, Ingeniería y Agrimensura (FCEIA) de la Universidad Nacional de Rosario, desde 1996 se implementó en el 2º cuatrimestre del ciclo básico el dictado de una asignatura de informática orientada a las necesidades específicas de la carrera (Informática II), con fuerte acento en el aprendizaje del diseño, la codificación, compilación y depuración de software utilizando los lenguajes de programación C y C++. El enfoque tradicional de la enseñanza en los primeros años se basa en el uso de herramientas de desarrollo del tipo IDE (por *Integrated Development Environment*) y la práctica de resolución de problemas sencillos, de temática general, a fin de desarrollar habilidades en la aplicación de algoritmos

básicos y el conocimiento de las herramientas disponibles en un IDE moderno. Esta asignatura representa la primera experiencia técnica en la currícula de Ingeniería Electrónica [3] y, completa la formación obligatoria en informática de un estudiante de dicha carrera que comienza con la materia Informática I, común a todas las carreras de ingeniería de la FCEIA, dictada en el primer cuatrimestre de las mismas.

Publicaciones especializadas [4] muestran la gran incidencia del desarrollo de software en el ejercicio de la profesión en particular en la temática de sistemas embebidos. Investigaciones de los autores del presente artículo, muestran evidencia del trabajo de una proporción significativa de egresados de la carrera de Ingeniería Electrónica de la FCEIA en este aspecto de la profesión y, la opinión prevalente en ellos acerca de la insuficiente formación recibida en el tema durante el curso de la carrera de grado. Ricks *et al.* [4] comparten esta opinión y señalan particularmente, las deficiencias en los enfoques actuales utilizados en la enseñanza de la programación, desde la perspectiva de las destrezas requeridas en el desarrollo de software para sistemas embebidos. Los autores remarcan que, normalmente en una asignatura típica de programación de propósito general en computadoras personales (PC) en una carrera de Ingeniería Electrónica, determinados temas muy específicos para sistemas embebidos no se tratan puesto que, no se encuentran razones válidas para presentarlos:

- Normalmente en un curso de programación de propósito general que utilice lenguaje C, se tiene muy poca motivación para presentar la aritmética de punteros fuera del contexto de estructuras de datos particulares; tema crítico para acceder y manipular registros en un sistema embebido.
- Asignaciones de punteros a direcciones específicas de memoria, operaciones a nivel de bits y calificadores de variables tales como `const` y `volatile` son tópicos que muy frecuentemente no se cubren en profundidad en un curso de programación en lenguaje C de propósito general. Las variables representan simplemente los datos necesarios para escribir un programa almacenados “en algún lugar de la memoria”. Las herramientas de traslación utilizadas, tales como compiladores y *linkers*, colaboran en abstraer aún más estos detalles. El programador debe usar explícitamente el prefijo “&” para determinar esta información en caso de ser necesario.
- A pesar que el uso de funciones, código `inline`, enfoque de programación modular, elección de variables globales versus variables locales y distintas formas de paso de argumentos a una función, sean todos temas presentados en un curso de programación de propósito general en lenguaje C, las motivaciones e impacto de su utilización en el desempeño de un sistema, no quedan completamente claros para los estudiantes mientras que, necesitan ser perfectamente entendidos por un desarrollador de software embebido. Debería educarse al alumno para poder realizar la mejor elección en cada aplicación particular, en lo referido a los aspectos estructurales mencionados en lugar de utilizar aquélla alternativa que considere más fácil. Adicionalmente, las herramientas de desarrollo de alto nivel utilizadas normalmente, abstraen los detalles estructurales durante el proceso de traslación del código fuente, construyendo ejecutables en forma automática; el alumno raramente es consciente de los mismos, aspectos que un programador de sistemas embebidos no puede desconocer.
- Debido a que los sistemas embebidos típicamente tienen recursos más limitados que aquéllos de propósito general, la necesidad de enseñar a manejarlos cuidadosamente es más imperiosa en el caso de los primeros. Cuando se incorpora al curso la orientación a objetos, el paradigma oculta resueltamente al programador los detalles de implementación de los recursos utilizados. Objetos tales como una instancia de alguna clase

compleja, una lista enlazada o un arreglo de estructuras en C, serían imposibles de alojar en la memoria reducida de un sistema embebido. La elección de tipos de datos más eficientes como así también el análisis de los requerimientos de memoria para cada variable declarada, son factores importantes en el caso de un sistema embebido, no así en un programa escrito para ejecutarse en una computadora personal. El uso de estructuras de datos más eficientes en cuanto a la utilización del espacio de memoria, tales como uniones y campos de bits, requeriría entender las distintas formas en que el compilador almacena los bits (*endianess*); detalles que muy raramente se tratan en los cursos de programación de propósito general.

- Muchos sistemas embebidos deben operar bajo restricciones temporales significativas, difícilmente consideradas en un curso de programación de propósito general; como consecuencia no se pone suficiente énfasis en la importancia de una codificación eficiente.
- En un sistema embebido otro recurso limitado e interrelacionado con los dos anteriores es la potencia. Normalmente este obstáculo incrementa la dificultad de cumplir con las restricciones temporales del sistema. Un programador de sistemas embebidos debe ser capaz de balancear el *trade-off* entre velocidad de ejecución, espacio de memoria y potencia disponible como así también conocer estrategias para conseguirlo.

Integrar y reforzar todos estos conceptos asociados a la programación de sistemas embebidos, en un curso de propósito general, no requiere un completo rediseño [4]; pueden cubrirse estas deficiencias modificando las asignaciones. Normalmente, en este tipo de asignaturas no se utilizan las plataformas de hardware requeridas para el desarrollo de sistemas embebidos; el hardware embebido puede simularse usando restricciones artificiales de espacio de memoria, temporales y direccionamiento de registros.

A partir de todos estos factores comienzan a analizarse las posibles vías de introducción de la temática en etapas tempranas de la carrera [5], encontrando evidencias de la factibilidad de implementar pruebas básicas de ensayo del ciclo de desarrollo de software para sistemas embebidos, manteniendo la complejidad en el nivel esencial desarrollado en la asignatura. La aparición de plataformas de microcontroladores sencillos y kits de desarrollo didácticos de bajo costo, como por ejemplo Arduino [14] o Raspberry Pi [25], hace más atractiva esta opción [6]. La Reforma Curricular de la carrera de Ingeniería Electrónica que comenzará a implementarse a partir del 2014, permitirá desde el 2015 desplazar el dictado de la asignatura Informática II (que pasará a llamarse Informática Aplicada) al 4º cuatrimestre de la misma, alentando aún más la implementación de esta propuesta.

La introducción temprana de los sistemas embebidos en la formación de grado permitiría poner de manifiesto la dimensión temporal del software, que tiene gran incidencia en el campo de la profesión por la presencia de sistemas de tiempo real. Como subraya Edward Lee [7]: “La falta de consideración de restricciones temporales en la abstracción esencial (de las ciencias de la computación) es un error, desde la perspectiva del software embebido”. O, como afirman Vahid y Givargis [8]: “El manejo explícito de las restricciones temporales propias de un sistema embebido es un concepto fundamental que debería enseñarse en las primeras etapas de una carrera de Ingeniería Electrónica”.

En [8] se remarca que, el aprendizaje de la programación orientada a datos en etapas tempranas de una carrera de ingeniería conduce a una perspectiva y conjunto de hábitos que, posteriormente cuando el alumno se enfrenta en etapas superiores de la misma a la programación de sistemas embebidos, son muy difíciles de desterrar. Los autores afirman que un enfoque donde, inicialmente se introduzcan métodos disciplinados orientados a la programación de sistemas embebidos (en particular la llamada por ellos, programación estructurada orientada al tiempo real) y, en

etapas avanzadas de la carrera los detalles de bajo nivel de estos sistemas, conduce a mejores desarrollos. Proponen la utilización en un curso introductorio de:

- una vista simplificada del micro-controlador (al que denominan micro-controlador virtual) que provee una plataforma bien definida para exponer e introducir los detalles de bajo nivel de un sistema embebido, además de abstrac-ciones de más alto nivel representando la estrecha relación entre hardware y software en este tipo de sistemas.
- Y, puesto que, la funcionalidad explícita de un sistema embebido no sólo involucra la noción de tiempo real sino también la de eventos, las máquinas de estado sincrónicas permiten una descripción sencilla del procesamiento de estos últimos, además de proveer una base elegante para especificar las restricciones temporales.

Los autores [8] afirman que los modelos computacionales son mucho más importantes que los lenguajes de programación y que, proveer a los alumnos con las abstracciones adecuadas facilita el entendimiento de los conceptos asociados, obteniendo como resultado, una sólida comprensión de cómo escribir una aplicación para un sistema embebido con o sin sistemas operativos de tiempo real.

Problemática

La utilización de una plataforma distinta a una computadora personal tradicional plantea todo un desafío [5] puesto que, implica ciertas dificultades fundamentalmente centradas en la introducción del concepto de "compilación cruzada", la cual demanda utilizar grupos de herramientas (*toolchains*) que deben incor-porarse al IDE. Al respecto hay abundante literatura en especial sobre la integración de estos conjuntos de herramientas al entorno o*pen source* Eclipse, experimentadas en forma directa por los autores del presente trabajo, durante la construcción de servicios web para sistemas embebidos.

Otra problemática a tener en cuenta es la falta de conocimiento detallado de los alumnos de primer año sobre la arquitectura de microprocesadores, la noción de espacio de entrada/salida y los conceptos de puertos y registros, que deben introducirse en etapas iniciales del cursado, particularmente cuando se desarrolla la arquitectura de Von Neumann. El concepto de "tiempo real" debe introducirse también, teniendo en cuenta que la formación previa recibida por nuestros estudiantes prescinde de la dimensión temporal en la construcción de programas. Podrían utilizarse las recomendaciones mencionadas en [8].
Finalmente, deben desarrollarse los conceptos de simulación y depuración "*on circuit*" para atender a la necesidad de los alumnos de comprender cabalmente el ciclo de vida de estos proyectos, en particular los aspectos específicos de depuración del software en entornos de compilación cruzada.

Experiencias

En etapas avanzadas de la carrera se introdujo con éxito la temática de programación de microcontroladores en lenguaje C, utilizando diversas plataformas, entornos de desarrollo y kits para el dictado de una asignatura electiva del noveno semestre de la carrera de grado, con excelentes resultados: no solo se completan las prácticas planteadas sino que, los alumnos proponen y ejecutan proyectos innovadores basados en la tecnología en cuestión [9]. Desde 2009 la asignatura Informática Electrónica viene dictándose ininterrumpidamente y, al presente han cursado y aprobado casi un centenar de alumnos.
Asimismo en otra materia electiva de la carrera que se dicta desde el 2003, Informática III, se introducen conceptos de diseño de software de tiempo real para sistemas embebidos, temática directamente relacionada. La participación de sistemas operativos en tiempo real se aborda desde el punto de vista de la concurrencia y las restricciones temporales, asunto que excede notoriamente al alcance planteado para Informática II aunque

existen puntos de contacto con ésta; en particular la utilización de servicios del sistema operativo y/o entorno en tiempo de ejecución, por parte de las aplicaciones.

Futuras direcciones

Una de las tareas pendientes es la correcta elección de la plataforma a utilizar para desarrollar los temas aquí presentados, módulo clave para las innovaciones previstas [10]. La evaluación actual orientó la búsqueda hacia una plataforma con buen grado de transparencia, es decir, abstracción del hardware subyacente, fundamentalmente porque los alumnos carecen de conceptos claves al respecto, evitando a su vez una abstracción completa que devolvería al estudiante al escenario tradicional de la enseñanza centrada en la computadora personal [8], [11].

El costo es también una variable muy importante, dadas las restricciones presupuestarias. Así, se dirigió la selección hacia una plataforma que brinde la posibilidad de correr un sistema operativo/entorno de *runtime* que, aunque mínimo, asegure manejo de entrada-salida, interrupciones y memoria. La abstracción ofrecida por este sistema operativo no debe ser completa, a fin que los alumnos puedan experimentar una exposición básica a la problemática de la interacción con el hardware y, al mismo tiempo tomar contacto con las restricciones temporales citadas más arriba.

El tamaño y tipo de alimentación de la placa son fundamentales para la elección, así como la conectividad, el tipo y dimensión de memoria y los dispositivos de entrada y salida disponibles. Es deseable la utilización, por ejemplo, de una plataforma con interface USB, que, además de conectar a una computadora permita alimentar la placa.

Resumiendo, los requerimientos son:

- Costo acotado
- Disponible en nuestro país
- Tamaño reducido
- Alimentada por USB
- Conectividad USB
- Dispositivos de entrada: potenciómetros, botones
- Sensores (acelerómetros, giróscopos, etc.)
- Display: deseable, no imprescindible
- Leds para mostrar salidas digitales

Desde este punto de vista tiene sentido la elección de Arduino [12], [13], fundamentalmente debido a la sencillez del lenguaje de programación y su portabilidad a distintas ediciones de kits de desarrollo y del mismo microcontrolador, siempre teniendo en cuenta como factor primordial para la selección el costo. Otras plataformas consideradas [12] son: Pic32 Starter Kit, Atmel STK500, AVR Butterfly, Parallax Stamp, Freescale Tower y Cypress PSoC.

Arduino es una plataforma de hardware libre [14], basada en una placa con microcontrolador y entorno de desarrollo, diseñada para facilitar el uso de la electrónica en proyectos multidisciplinarios. La comunidad Arduino no sólo está compuesta por ingenieros y científicos sino también por gran cantidad de artistas y aficionados a la electrónica [15] y, es parte de una creciente tendencia hacia el desarrollo de hardware/software *open source* por parte de revolucionarios tecnológicos privados conjuntamente con empresas que comercializan kits de electrónica.

El hardware consiste en una placa con un microcontrolador de 8 bits Atmel AVR y puertos de entrada/salida [14]. Los microcontroladores más usados son el Atmega168, Atmega328, Atmega1280, ATmega8; que por su sencillez y bajo costo permiten el desarrollo de múltiples diseños. Por otro lado el software consiste en un entorno de desarrollo que implementa el lenguaje de programación Processing/Wiring, variación simplificada del lenguaje C, y el cargador de arranque (*bootloader*) que corre en la placa. Igualmente puede programarse en otros lenguajes, incluyendo C/C++.

Desde octubre de 2012 [14], Arduino se usa también con microcontroladores ARM Cortex M3 de 32 bits, que coexistirán con las más limitadas, pero también económicas AVR de 8 bits. ARM y AVR no son plataformas compatibles a nivel binario, pero pueden programarse con el mismo IDE de Arduino y

hacer programas que compilen sin cambios en las dos plataformas. Arduino puede utilizarse para desarrollar objetos interactivos autónomos como prototipos o, interaccionar con software instalado en el ordenador. Las placas pueden montarse a mano o adquirirse. El entorno de desarrollo integrado libre puede descargarse gratuitamente. Al ser *open-hardware*, tanto su diseño como su distribución son libres. Es decir, puede utilizarse libremente para el desarrollo de cualquier tipo de proyecto sin haber adquirido ninguna licencia. El proyecto Arduino recibió una mención honorífica en la categoría de Comunidades Digital en el PrixArs Electrónica de 2006.

El Arduino UNO (a un costo de 30 dólares) [16] es una buena elección para aplicaciones académicas, con un excelente desempeño técnico considerando su bajo costo.

Figura 1: Arduino UNO

Permite con muy poco esfuerzo, sin requerir el soldado de componentes externos ni tener conocimientos de circuitos complejos, determinar el estado de llaves y otros sensores y controlar motores, luces, etc. El Arduino UNO, que se muestra en Figura 1, es una placa electrónica basada en el ATmega328. Sus componentes A/D permiten leer hasta 7 canales de datos a velocidades suficientes para un amplio rango de aplicaciones. Tiene 14 pines de entradas/salidas digitales (de los cuales 6 pueden utilizarse como salidas PWM emulando una salida analógica de nivel variable), 6 entradas analógicas, un oscilador de cristal de 16 MHz, una conexión USB. El ATmega328, microcontrolador de la compañía Atmel, cuenta con 32KB de memoria *flash*, 2KB de memoria RAM y 1KB de memoria EEPROM, es decir, tiene capacidad de memoria suficiente para desarrollar aplicaciones complejas de adquisición de datos y control. Arduino UNO contiene todo lo necesario para funcionar, sólo tiene que conectarse a una PC con un cable USB o, alimentarse con un adaptador AC-DC o batería para comenzar a trabajar.

Debido a la familiaridad de los alumnos con Visual Studio podría adoptarse como IDE el Atmel Studio 6 [17] basado en el mismo y que, permite desarrollar/depurar código en C/C++ y *assembler*. Este IDE provee más de 1000 ejemplos de código fuente.

Los programas en Arduino se denominan *sketch* por que el IDE proviene de Processing y, en este lenguaje de programación enfocado al mundo gráfico cada código es considerado un boceto, en inglés "*sketch*". La estructura de un sketch consta de dos funciones: *setup* y *loop*. Al momento de comenzar la ejecución se realiza la fase de *setup* o inicialización y, luego de completarse la misma se ejecuta la de *loop* (lazo infinito) correspondiente a las tareas principales de una aplicación escrita en lenguaje C; esto se repite hasta que la placa sea apagada o reseteada.

El "Hola Mundo" de Arduino, cuyo código se reproduce en Figura 2, es un *sketch* que, trata de hacer que un led incorporado en la placa o uno externo, se encienda y apague a intervalos de tiempo definidos en el código, primera aproximación del alumno a un sistema que incluye explícitamente la noción de tiempo real. En [8] se muestra un modelo computacional asociado a este código, representado por una sencilla máquina de estados sincrónica.

Si en lugar de utilizar este led se añade uno RGB (rojo/verde/azul) externo, conjuntamente con un potenciómetro o pulsador; a partir de un programa pre escrito similar al "Hola Mundo" pueden realizarse una gran cantidad de modificaciones al mismo: hacer titilar el led a distintas frecuencias o siguiendo una secuencia determinada de colores, modificar el estado del mismo según se apriete o no un pulsador, cambiar su nivel de brillo o el

porcentaje de colores primarios, según el ángulo de rotación del cursor de un potenciómetro [12]. El simple agregado de un pulsador que controle el apagado y encendido del led introduce al alumno a la noción de "evento": acción externa que puede ocurrir en cualquier instante de tiempo y, ante la cual el sistema embebido debe responder de alguna forma predeterminada.

```
/
*-----------------------------------
  Hola Mundo
 -----------------------------------
 Enciende un LED por un segundo y lo
apaga por el mismo tiempo
 */

/* Función principal:
Se ejecuta cada vez que el Arduino
se inicia */

void setup(){
/* Inicializa el pin 13 como una
salida */
   pinMode(13,OUTPUT);
} // Tener un LED en el pin 13

/*Funcion cíclica:
Esta función se mantiene ejecutando
mientras esté energizado el
Arduino*/
void loop(){
// Encientde el LED
  digitalWrite(13,HIGH);
// Temporiza 1 s. (1s = 1000ms)
  delay(1000);
// Apaga el LED
  digitalWrite(13,LOW);
// Temporiza 1 s. (1s = 1000ms)
  delay(1000);
}// Fin del programa
```

Figura 2: *Sketch* del Hola Mundo en Arduino

Albrecht *et al.* [6] aconsejan que la primera interacción del alumno con la plataforma Arduino consista en abordar un problema simple de entrada/salida, que les permita obtener resultados tangibles y evidentes muy fácilmente. Una introducción sencilla al uso de la plataforma puede encontrarse en el libro escrito por un cofundador de la misma [18]; algunas de las asignaciones sugeridas en dicho texto para un curso introductorio son:

1. Controlar el apagado/encendido y frecuencia de intermitencia de un led.
2. Leer entradas de distintos tipos de dispositivos digitales.

La plataforma Arduino es lo suficientemente sencilla como para completar un trabajo en una o dos clases. Se plantea por tanto, la necesidad de desarrollar nuevo material de lectura, ejercicios de laboratorio, elaboración de directivas de manipulación adecuada del hardware y búsquedas de sitios en Internet donde los alumnos puedan encontrar material de apoyo extra. Actualmente ya se ha incorporado al material de la cátedra, algunos trozos de código ejemplos de los temas vistos en las clases teóricas [19], extraídos de hojas de datos de microcontroladores como así también, algunos ejercicios simples y apuntes de la electiva Informática Electrónica, con respecto a los temas de más bajo nivel del lenguaje C. Durante la transición, hasta que efectivamente se implemente la asignatura Informática Aplicada, se prevé reforzar esta tarea.

En [8] publicado en 2008, se indica que la enseñanza de programación de sistemas embebidos ha progresado sólo moderadamente en las últimas dos décadas, careciendo por tanto de una sólida disciplina. Además Recktenwald y Hall [20] y Jamieson [15] señalan que prácticamente no existe material exhaustivo donde se utilice Arduino como plataforma para un curso introductorio de programación. Si bien existen muchos recursos en Internet (tutoriales, videos de YouTube y algunos libros de texto) que resultan valiosos para la cátedra, no proveen una instrucción más sistemática para utilizar la plataforma como herramienta pedagógica en un curso de programación introductorio. Los autores recuerdan además que no existe solución sencilla para el difícil aprendizaje de C, como primer lenguaje de programación, dada su lacónica e implacable sintaxis, o sea, hay que ser realistas en el uso de cualquier tipo de solución pedagógica al respecto.

Plan de acción

En el transcurso del 2013 y primer semestre del 2014 se prevé realizar la compra de la plataforma Arduino para la preparación del material necesario, centrándose principalmente en la selección del IDE más apropiado para un alumno principiante, con la mínima cantidad de *toolchains* necesarias. Durante el segundo semestre del 2014 y primero del 2015, en ocasión de re dictados de la materia Informática II, se propondrá a los estudiantes la realización de un trabajo práctico de programación sencilla sobre la plataforma Arduino, trabajando en grupos reducidos. Esta asignación que será de ejecución opcional, voluntaria y sin efectos promocionales, permitirá evaluar la recepción del tema entre los alumnos, las posibles dificultades y los resultados obtenidos, con miras a su implementación definitiva en el segundo semestre del 2015 durante el primer dictado de la nueva asignatura Informática Aplicada.

Reflexiones finales

La introducción de un lenguaje de programación muy similar a C pero diferente en términos de utilización de librerías y modelo de desarrollo [21] [10] plantea dudas en cuanto a la dificultad de extrapolar la experiencia a plataformas comerciales programables en C/C++. Como lo remarcan Recktenwald y Hall [20], la popularidad de Arduino entre hobbistas y profesionales no significa que los estudiantes responderán positivamente y que, en su experiencia no todos los alumnos se mostraron interesados o inspirados por el uso de esta tecnología. Finalmente, en [15] se advierte que la comunidad Arduino está más enfocada en hacer que las aplicaciones funcionen antes que, en elegir la forma óptima de implementarlas por tanto, recomiendan que en cursos avanzados centrados en optimizaciones de bajo nivel, se escojan otras plataformas. Esta recomendación se aplica desde el 2009 en la electiva Informática Electrónica.

Bibliografía

[1] Grimheden, M., Törngren, M., How should embedded systems be taught?: experiences and snapshots from Swedish higher engineering education. SIGBED Review 2(4): 34-39, 2005.

[2] CONFEDI, Desarrollo de Competencias en la Enseñanza de la Ingeniería Argentina, Villa Carlos Paz - Argentina. 2006.

[3] Mealy, B. J., Work in progress - embedded system-based introductory programming course for computer and electrical engineering students. 38th ASEE/IEEE Conference on Frontiers in Education - FECS, pp. F3E-17-F3E-18, 2008.

[4] Ricks, K. G., Jackson, D. J. and Stapleton, W. A., Incorporating embedded programming skills into an ECE curriculum. ACM SIGBED Review Special Issue on the Second Workshop on Embedded Systems Education (WESE 2006), vol. 4, no. 1, pp. 17-26, 2007.

[5] Benson, B., Arfaee, A., Kim, Ch., Kastner, R., Gupta, R.K., Integrating Embedded Computing Systems Into High School and Early Undergraduate Education. IEEE Trans. Education 54(2): 197-202, 2011.

[6] Albrecht, W., Bender P. and Kussmann, K., Integrating microcontrollers in undergraduate curriculum, Journal of Computing Sciences in Colleges, Volume 27 Issue 4, pp. 45-52, 2012.

[7] Lee, E.A., Building Unreliable Systems out of Reliable Components: The Real Time Story, University of California at Berkeley, Technical Report Nº UCB/EECS-2005-5, 2005.

[8] Vahid, F. and Givargis, T., Timing is everything-Embedded systems demand early teaching of structured time-oriented programming. Proc. Workshop Embedded Syst. Educ., pp. 1 -9, 2008.

[9] Proyectos Informática Electrónica: http://www.dsi.fceia.unr.edu.ar/index.php?option=com_content&view=article&id=393:proyectos-destacados-2009&catid=53:informatica-electronica&Itemid=80

[10] Li, X., Li, X., Gou H. and Tang, X., Innovation for the Course of C Language to Embedded Platform, International Conference on Education Technology and Computer - ICETC, vol. 2, pp. V2-529 - V2-532, 2010.

[11] Parikh, Ch., Dunne, B., Adkins, B., Embedded Systems in Introductory Programming Course. ASEE North Central Section Conference, 2009.

[12] Furman, B. Wertz, E., A first course in computer programming for mechanical engineers. Mechatronics and Embedded Systems and Applications (MESA), 2010 IEEE/ASME International Conference on, pp. 70 – 75, 2010.

[13] Sarik, J., Kymissis, I., Lab kits using the Arduino prototyping platform, Frontiers in Education Conference (FIE), 2010 IEEE, pp. T3C-1 - T3C-5, 2010.

[14] Arduino, es.wikipedia.org/wiki/Arduino.

[15] Jamieson, P., Arduino for Teaching Embedded Systems. Are Computer Scientists and Engineering Educators Missing the Boat?, International Conference on Frontiers in Education: Computer Science and Computer Engineering (FECS'11), 2011.

[16] Arduino UNO, disponible en: arduino.cc/en/Main/arduinoBoardUno.

[17] Atmel Studio 6, disponible en: www.atmel.com/microsite/atmel_studio6/.

[18] Banzi, M., Getting Started with Arduino. O'Reilly Media Inc., Sebastopol, 2009.

[19] Material cátedra Informática II: www.dsi.fceia.unr.edu.ar/downloads/informatica/info_II/Estructuras.pdf

[20] Recktenwald G.W. and Hall, D.E., Using Arduino as a platform for programming, design and measurement in a freshman engineering course, 118th ASEE Annual Conference and Exposition, Vancouver, 2011.

[21] Lee, E.A., What's Ahead for Embedded Software, Vol. 33, Issue 9, IEEE Computer, 2000.

[22] Joint Task Force on Computer Engineering Curricula, IEEE Computer Society, Association for Computing Machinery, Computer Engineering 2004: Curriculum Guidelines for Undergraduate Degree Programs in Computer Engineering, December 12, pp. A.43 – A.45, 2004. Disponible en:

www.computer.org/education/cc2001/CCCE-FinalReport-2004Dec12-Final.pdf.

[23] Vahid, F., Time-Oriented Programming, 2008. Disponible en:

www.ee.washington.edu/research/nsl/aar-cps/FrankVahid-20081021192932.pdf.

[24] Resolución 1232/01 del Ministerio de Educación, Ciencia y Tecnología. Disponible en:

www.confedi.org.ar/sites/files/privado/Acreditacion-13terminales.pdf.

[25] Raspberry Pi, Wikipedia: es.wikipedia.org/wiki/Raspberry_Pi.