The Use of Video-Gaming Devices as a Motivation for Learning Embedded Systems Programming

Jesús González, Héctor Pomares, Miguel Damas, Pablo García-Sánchez, Manuel Rodríguez-Álvarez and José M. Palomares, *Senior Member, IEEE*

***Abstract*—As embedded systems are becoming prevalent in everyday life, many universities are incorporating embedded systems-related courses in their undergraduate curricula. However, it is not easy to motivate students in such courses, since they conceive of embedded systems as bizarre computing elements, different to the personal computers with which they are familiar.**

This problem has been overcome at the University of Granada, Spain, by taking advantage of the connection many students have with video games.

***Index Terms*—Embedded systems teaching, firmware development, mobile and personal devices, open-source development tools, student motivation**

I. Introduction

EMBEDDED computing systems are omnipresent, being found, for example, in consumer electronics such as cell phones, digital cameras or set-top boxes; home appliances like microwave ovens, washing machines or air conditioners; in office automation appliances such as faxes, card readers, scanners, or in automobiles, in transmission control, fuel injection or antilock brakes. This ubiquity means that more computing devices are embedded into larger electronic devices than the total of Personal Computers (PCs) and servers currently in use. Specifically, 10 billion embedded processors were sold in 2008 [1], while the number of PCs sold that year was "only" 302.2 million units, according to [2]. That is, 33 embedded processors were sold for each PC sold in 2008. Additionally, according to market research [3], this gap tends to increase each year.

On the other hand, embedded systems usually share certain characteristics that distinguish them from other computing systems. These kinds of systems usually run a fixed application (or group of applications), are tightly constrained to reach a suitable performance-cost-power relation with the application for which they were designed, and this application usually needs to be executed in real-time. These features, as well as the aforementioned large sales volume, have led to the recent incorporation of embedded systems-related courses into Electrical Engineering, Computer Engineering or Computer Science undergraduate degree programs in most universities and technical Schools worldwide [4]–[6]. Many books covering the various aspects of embedded systems, from their design [7]–[9] to their software and firmware development [10]–[13], have also appeared in recent years.

Even though most courses use development platforms based on typical embedded processors (ARM, MIPS, AVR, x86, etc.) and FPGAs in their labs [14]–[16], this paper describes an approach in use since 2004 at the University of Granada, Spain, and awarded with an Innovative Teaching Honorable Mention in 2008, where portable video game consoles are used as the platform to teach embedded systems programming. Besides being based on embedded processors and custom hardware (video processors, media codecs, etc.) and incorporating more devices than average development boards at a lower price, the most important feature of these consumer electronics is the connection that many students feel with them, having spent pleasant times on them with family and friends. Another advantage of using a video game console as the development platform is that because of the high sales volume of the video games market, many students will probably have already purchased or would like to purchase one for recreational purposes, or have easy access to one via a relative or friend. As an illustration of the scale of this, 409.9 million video games were sold across the world in 2008, according to [17].

The rest of the paper is organized as follows. Section II presents some related work also aimed to motivate students to learn about embedded systems. The selection of an adequate video game console for the course labs is described in Section III. Section IV justifies the use of open-source tools to develop the embedded software for the consoles. Section V summarizes the structure and learning outcomes of the embedded systems course at the University of Granada. Section VI details the lab sessions, providing the basic information needed to set up a similar lab at any university. Section VII presents the students' response to this course. Finally, Section VIII concludes this paper.

J. González, H. Pomares, M. Damas, P. García-Sánchez and M. Rodríguez-Álvarez are with the Department of Computer Architecture and Computer Technology, University of Granada, E.18071 Granada, SPAIN.

J. M. Palomares is with the Department of Computer Architecture and Electronics, University of Córdoba, E.14005 Córdoba, SPAIN.

The final authenticated version is available online at https://doi.org/10.1109/TE.2012.2208194.

II. Related Work

Motivating students to learn about embedded systems is not a new topic. This section discusses various approaches to this in the literature by means of labs based on other types of platforms attractive to students, and emphasizes their differences with the work presented in this paper.

For example, [18] describes how the use of LEGO Mindstorms, programmable toys consisting of a Robotic Command Explorer (RCX) based on an ARM7 processor, two motors, one light sensor, two touch sensors, wires, and LEGO blocks, excites a high level of student interest for embedded systems programming. However, this experience suffers from a serious drawback. Although RCX modules are based on the same ARM core as the portable video game consoles proposed in this paper, the development tools and the runtime environment for applications are both very specific to RCX modules. This makes the skills acquired by students far less applicable to software development for embedded systems in general, one of the main objectives of this work.

Another attempt at student motivation is presented in [19], which proposes the use of a low-cost System-on-Chip computer, specifically, the eBox 2300 MSJK embedded computer, which is PC compatible, supports most of the popular embedded operating systems, and has a price level close to the cost of an academic textbook, so that students have the option of purchasing their own development platform. This proposal has a couple of disadvantages. On the one hand, as the execution platform is PC compatible, students do not work with typical embedded systems architectures. In fact, the proposed exercises could be performed on a PC. On the other hand, both development tools and the operating system are proprietary, which may be discouraging for students due to their usually high license fees. Indeed, another goal of the work presented in this paper is to use open source and multi-platform tools to increase student' motivation (who thus have no need to buy licenses) and let them acquire development skills easily applicable to any embedded platform.

In [20] a project-based laboratory is described where students must build a line-following robot based on a PIC microcontroller. Although this approach seems quite interesting, it relies on the support of the Taiwanese Microchip company, whose headquarters is at nearby Lunghwa University, and who supports this laboratory by offering compilers and microcontrollers at no cost.

On the other hand, an experimental laboratory designed in [21] based on the Cisco-Linksys WRT54GL wireless router, which is an inexpensive commodity that can be already found in homes and small businesses. Although this approach shares several characteristics with that proposed in this paper, such as the use of a fairly familiar device and open source development tools, the main processor of the router is an MIPS 32, which is not as widely used as the ARM cores.

The use of video games to motivate students is also a well-known topic. Due to the interest that many students have in games and consoles [22]–[24], a new trend that tries to exploit this fact with educational purposes is emerging, giving rise to the so-called educational games or serious games [25]–[27]. Some examples of these serious games in the field of engineering- and computing-related courses are [28]–[30]. The basic difference between this trend and the approach presented in this work is the use that students make of the consoles. Rather than playing a video game on the console whose subject is how to develop embedded software, students use the console as a familiar platform that helps them to learn the basic skills about embedded software development.

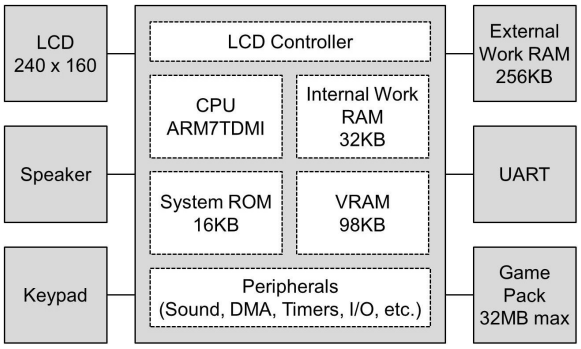


Fig. 1. GameBoy Advance block diagram

III. Development Platform Description and Evolution

As the video game console market has evolved since the introduction of these devices in the course labs, this section describes how the initial platform was selected and how this selection has been updated with the passage of time.

A. Initial Platform

From a study of the trends in the embedded systems market in 2004, evidence emerged that many embedded systems were using 32-bit ARM processors. These processors, at least in their basic incarnation, also had a reasonably simple and straightforward instruction set, free from many of the idiosyncrasies of more limited microcontrollers, and relatively easy for students to use [31].

After examining the video game consoles available at that time, the *Nintendo GameBoy Advance* (GBA), based on an ARM7TDMI processor, was chosen. As well as incorporating this processor, that even nowadays is one of the most used ARM cores, this console also incorporates special-purpose hardware to speed up the most commonly used functions in video games, the typical controllers in a computer-based system, and a diverse set of peripherals, all at a significantly lower price than a typical development board. Fig. 1 shows the architecture of the proposed platform.

Another advantage of the chosen platform is the availability of the open-source *VBA-M* simulator [1], which

[1] http://sourceforge.net/projects/vbam

implements the console hardware to such a level of accuracy that it can even execute commercial games originally designed for the console. This simulator can also act as a *gdb* server, allowing the debugging of a piece of code in exactly the same way as if it were running on a real embedded development platform. As a result, even though the lab exercises were always carried out on real consoles in the laboratory, there was no actual need for the students to attend lab sessions or even to buy the console in order to learn how to develop and debug embedded software for the GBA.

B. Present Platform

In 2008, Nintendo discontinued GBA consoles in favor of their successor, Nintendo DS (NDS); it thus no longer made sense to use the GBA consoles as the development platform, since they were not currently available. However, taking NDS as the new development platform presented several drawbacks. First, as NDS is a multiprocessor platform, its architecture is significantly more complex than that of the GBA, and is thus unsuitable for an introductory course on embedded systems. Additionally, NDS simulators are not as advanced and robust as VBA-M, and can cause random failures under certain runtime conditions that could frustrate students. Nevertheless, as NDS is fully binary compatible with its predecessor, it was decided to keep using VBA-M to develop and debug embedded software based on the GBA architecture, and to execute this software on the NDS. This approach provides stable development tools in the lab sessions, keeps a quite simple architecture for the embedded system, and allows students to execute the embedded programs on one of the most currently popular video game consoles, which reinforces their motivation to learn about embedded systems.

IV. Development Tools

Given that open-source development tools are currently widely accepted in academic environments, and that the GNU toolchain is able to generate a high quality code for ARM platforms, all the lab sessions were designed to use open-source tools. In order to develop embedded software for the GBA, it is necessary to use the GNU compiler collection *gcc* to generate the object code, the GNU *binutils* to manage the target binary files, the *newlib* C library for embedded systems, and the *insight* debugger.

The embedded software developed in the lab sessions can be executed in the VBA-M simulator, which also implements a *gdb* server; thus, the students will be able to practice remote debugging in the same way as is currently done with any other embedded development platform. As all these tools are open-source, students can build them on their own computers and carry out some of the proposed exercises in the lab sessions at home [2].

On the other hand, as the software developed in lab assignments can be run on any GBA console, and also in NDS and NDS Lite, all lab assignments are evaluated by running the programs on a real console, which also increases students' motivation. To perform the evaluation, the lab is equipped with two GBA and two NDS platforms, and four SuperCard cartridges to load the software on the consoles. Additionally, students can choose to use their own devices instead of the consoles available in the lab.

[2]A script to construct these tools from their sources can be found at http://atc.ugr.es/~jesus/esTedu/.

V. Course Description

Embedded Systems is a 60 contact-hour elective course offered in the last semester of the Computer Science degree program at the University of Granada. As each semester is 15 weeks long, the course is scheduled into four hours each week: two lecture hours, and two lab hours. Typically, students have had prior coursework in programming in C/C++, digital logic, computer architecture, and operating systems. Students have also been taught the basic embedded programming topics, such as the use of timers, AD/DA converters, interrupts and general purpose I/O, in the prerequisite Micro-controllers course. Thus, this course was focused on how to develop embedded software for complex embedded systems based on 32 bit processors. While most micro-controller-based courses are typically focused on how to interact with I/O devices to control robots and the like, using a Board Support Package (BSP) and a toolset already provided by the microcontroller manufacturer, this course is rather focused on how to generate a working toolchain from open source tools for a given platform, the design and development of the BSP (Board Support Package) for that platform from scratch, and the implementation of the C library system calls based on this BSP. The course was designed for the students to achieve the following learning outcomes:

1) Identify the features which distinguish embedded systems from general purpose computing systems.
2) Select, configure, and use embedded systems debugging and development tools.
3) Develop firmware for simple embedded systems.
4) Develop peripheral drivers at different levels of abstraction.
5) Optimize embedded software to maximize its performance and minimize its power consumption.

The theoretical basis and general principles of this course are explained in the lecture hours, leaving the lab sessions for the students to achieve the necessary skills to develop embedded software. In order to pass the course, students must solve all the lab homework successfully, and optionally, they can improve their final grade by means of a short-answer final examination.

The following section details the laboratory assignments designed for the course, showing how to make the most of GBA to motivate students to learn the development of embedded software for bare-metal ARM systems using an open-source toolchain.

VI. Laboratory Assignments

All of the lab sessions described below are split into two parts. The first part consists of a guided tutorial that

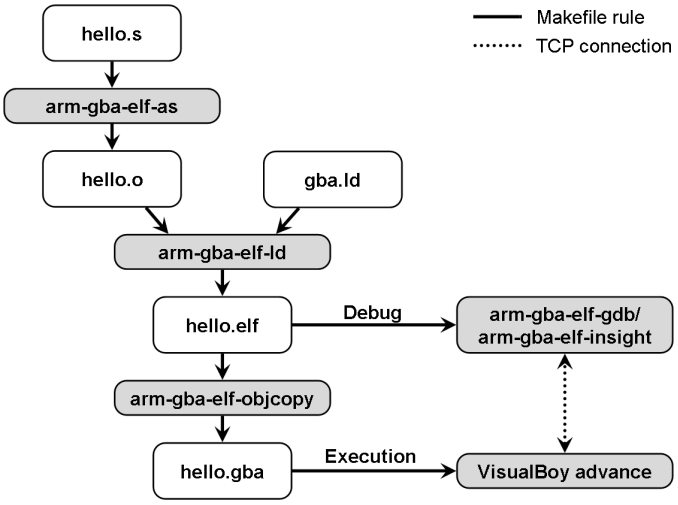


Fig. 2. Toolchain usage to build the GBA version of the *hello world* program

presents the basic objectives of the session and how to attain them. Once students finish the tutorial, they can begin with the second part, which proposes some exercises to develop the contents introduced in the tutorial [3]. Whereas students are free to follow the tutorials, or even to solve the exercises at home, the final assessment of the exercises is performed in the laboratory, on real consoles.

Once students have completed all four lab assignments, the remaining lab hours consist in the application of all what they have learned during the course in a final project, a full application that must run on the console.

A. Introduction to Platform and Toolset

This four-hour lab is intended to introduce students to the toolchain and the development platform. The objectives are for the students to learn how to build and use the open-source development tools to generate code for ARM-based platforms, and also, how to debug embedded software running in the console, covering learning outcomes 1 and 2 (Section V). To achieve these goals, first, students are given a script to build the toolchain from its source code, so that they can construct their own development tools. Once the tools are ready, they can be used to build and debug a quite simple assembler GBA-adapted version of the *hello world* program, which simply flickers the central pixel of the GBA screen within an infinite loop. Fig. 2 shows how to build the program from its assembler code (`hello.s`) and the default linker script for the console (`gba.ld`).

This lab session also introduces the need to keep the embedded system memory map in mind in order to allocate the program instructions and data to the right memory regions. Thus, the students are also guided to generate a linker script, which details all the memory regions of the embedded system and the mapping between each program section and its corresponding memory region.

[3]All the lab assignments, along with some source code examples and additional documentation can be downloaded from http://atc.ugr.es/~jesus/esTedu/.

Homework

Once the students know how to build and use the toolchain, they are instructed to modify the *hello world* program to alternately flicker two pixels. The objective of this homework is for students to learn the basics of assembly programming for ARM cores, and also to become acquainted with the development tools which will be used along the course.

B. Development of a Basic C Runtime Environment

Although assembly language is necessary for an efficient implementation of some low level system functions in every embedded system, most of the system code is usually written in a high level language, such as C or C++, in order to speed up the system development. Taking this into account, this four-hour lab shows the students how to design, in assembly code, a basic loader and runtime environment able to execute basic C programs on an embedded system [32]–[34], covering learning outcome 3 (Section V). Fig. 3 shows the basic steps performed by the startup code.

Homework

To check that students have understood the tutorial, and also that they have implemented a correct C runtime environment for the GBA, they are asked to develop a simple C application that swings a pixel alternately between the left-hand and the right-hand sides of the GBA screen. This application must make at least one function call, and must also use both global and local variables in order to test that the C runtime works correctly.

C. Device Handling

At this point, even though the students know how to build C programs for the embedded platform, their programs have a quite limited functionality, as the details of the GBA architecture [35] have been deliberately omitted to make the two previous lab sessions more general. This tutorial shows them how to design data structures that fit into the device control registers, how to allocate these structures to the right memory addresses using the linker script, and how to design a basic set of primitive functions to manage these structures, covering the low level part of learning outcome 4 (Section V). All these topics are illustrated with examples that show how to manage the GBA video controller.

Homework

Once the students are able to draw pixels on the GBA display, they must develop an application to draw a bitmap on the console screen. To achieve this task, apart from understanding how to manage the GBA video controller, they also need to develop a custom tool, `bmp2gba`, to

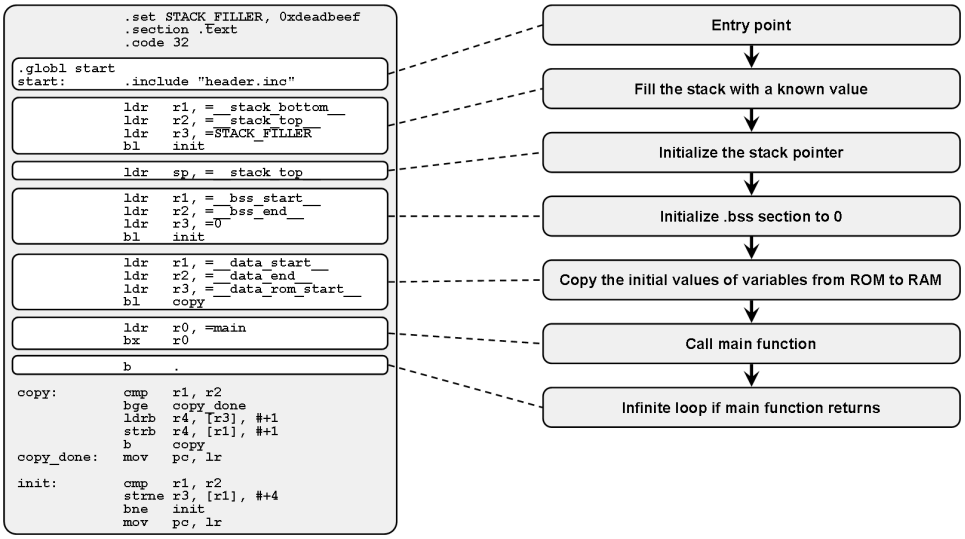


Fig. 3. Startup program for the console.

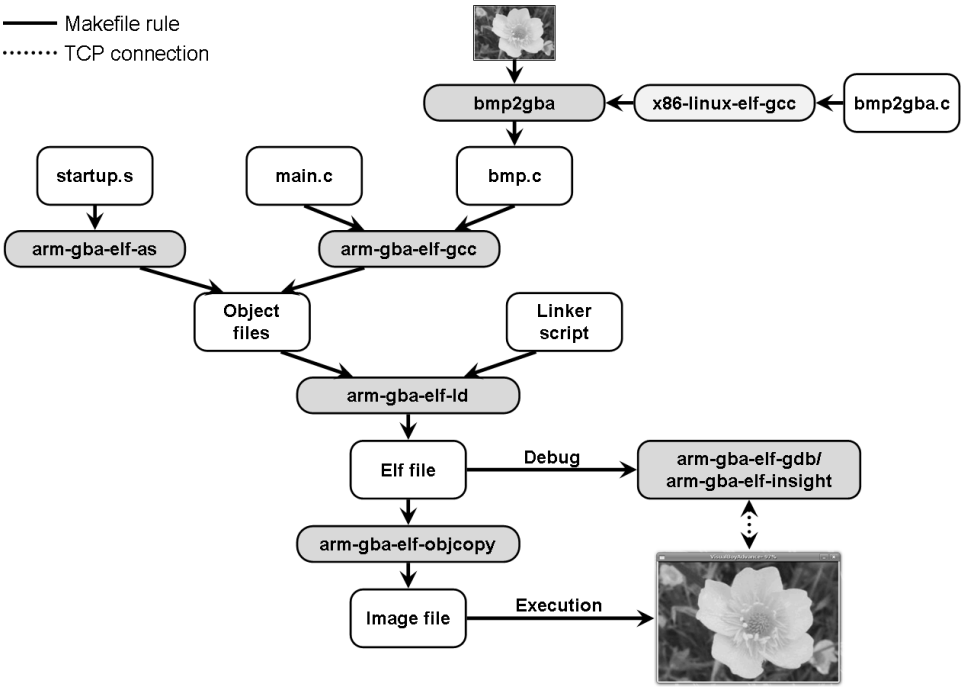


Fig. 4. Tools usage to draw a bitmap image on the GBA screen.

translate a bitmap file into a C source code containing an array with all the image data. These data must be in the correct format to be directly stored in the GBA video buffer. Fig. 4 shows a diagram of the whole process.

D. System Calls

There are some functions in the *libc* library that need some kind of operating system support. This six-hour lab chooses one of them, the `printf` function. As `printf` is designed to output a text stream to the standard output file `STDOUT`, the target platform should support standard input/output, i.e., standard file handling system calls should be implemented for the GBA console. As *newlib*, the *libc* implementation selected for the toolchain, approaches this problem by implementing all system calls via stub functions which call platform dependent functions, this lab shows how to design these stub functions according to the target resources [36], [37], covering the high level part of learning outcome 4 (Section V).

Homework

This homework consists in porting the standard input/output system calls to the GBA. To carry out this task, students must design a text font for the GBA, and also write functions for setting the cursor on a position on the screen and drawing a character. Once these functions are available, the students must design an implementation for the *newlib* standard file handling stubs that outputs

text to the GBA screen any time that it is written to the standard output file `STDOUT`.

E. Final Project

After the four lab assignments described above, students must develop a final project applying everything they have learned in the course, both in previous lab sessions and in theoretical classes. The objective of this final project is for students to demonstrate they are able to deploy a complete application in an embedded system from scratch, covering all the learning outcomes described in Section V. To carry out this task, they will also need to apply knowledge gathered from previous courses (in microcontrollers, programming, and so on).

Although students are allowed to choose their project topic freely, because the final execution platform is a console students usually end up developing a more or less sophisticated video game, depending on their skills and their degree of motivation with respect to the course. Classic games such as Tetris, Packman or Space Invaders are a recurrent topic every year.

VII. Assessment

Although the use of video game consoles to motivate students to enroll on the course could, *a priori*, look like a good idea, the only way to ascertain whether this is true is by means of an opinion poll about the course lab sessions. Therefore, a survey has been administered annually since 2004. Fig. 5 shows the questions along with the responses gathered in recent years. It can be appreciated that questions Q1 and Q2 were introduced in the survey in 2006 to estimate, more directly, the motivating effect of video game consoles on the students in the course.

As a result of the student responses, the following conclusions were drawn:

- When the students chose the course, they knew that the lab sessions were based on portable video game consoles (Q1), probably because some of the students who had taken the course before had recommended it to them (Q6).
- As for the lab sessions, students are quite pleased with their contents (Q4) and with the fact that the development platform is a video game console (Q2). Additionally, the ability to do the lab exercises at home, with the aid of open-source tools, is highly appreciated by the students (Q3).
- Finally, students think that the course assessment is adequate, mainly in recent years (Q4), thus it is hardly surprising that most of them recommend the course to other students (Q6).

This last conclusion may be related to the increment in the number of students who have taken the course in the last years. As Fig. 6 shows, the course enrollment has been increasing year on year, going from around 20 students before the introduction of the consoles in the labs in 2004 to nearly 40 in the last three years. This increment is independent of external factors such as the overall enrollment in the CS program, or the changes in elective courses offered over that period, since the number of new places per year and the elective courses in the curriculum were both fixed, prior to the experience related in this work, by the Andalusian and Spanish governments, respectively. With this scenario, the only new factor in this period has been the change of the laboratory, thus all the appreciated improvements should be due to the introduction of consoles in the lab.

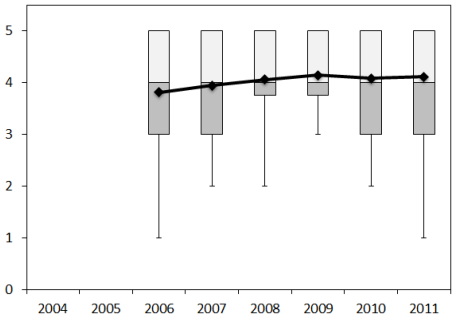
TABLE I
Distribution of the final score between laboratory assignments and learning goals. The short-answer final examination is optional, and can be used to increase the final score up to 30% over the lab score.

Lab assignment	ESD	FD	CO	DIO	Total
Introduction	10%				**10%**
C runtime		10%			**10%**
Device Handling		10%		10%	**20%**
System calls		10%		10%	**20%**
Final project	10%	10%	10%	10%	**40%**
Total	**20%**	**40%**	**10%**	**30%**	**100%**
Final exam (optional)	10%	10%		10%	**30%**

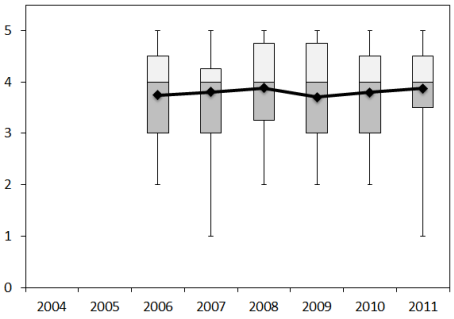
The effectiveness of using the GBA in the course can also be assessed by the students' final score in recent years. As Fig. 6 shows, the ratio of students achieving a final grade higher than E has increased significantly. Consequently, the average score has also increased from a value of around 6.5 (C) before the console was introduced as the labs' target platform, reaching a stable value of around 7.2 (B), as shown in Fig. 7.

To gain a deeper understanding of these results, all the students' lab exercises have been classified according to the following learning goals, which are in accordance with the main objectives of the course: embedded systems design (ESD), firmware development (FD), code optimization (CO), and drivers and I/O (DIO). The final short answer examination contributes optionally, as it is generally used by some students to increase their final score by up to 30%. It is important to remark that this evaluation method was also used before the introduction of the proposed platform in labs, thus the scores analysis presented below is significant. Table I shows the distribution of the final score between laboratory assignments, learning goals and the optional final exam. This yields a set of scores associated with each of these goals for every year since 2002. Now, a one-way ANOVA test can be run for every learning goal where the academic year is the factor under analysis.

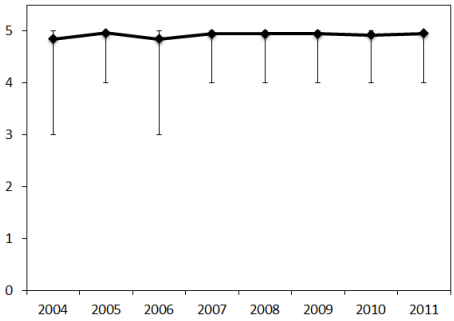
The results obtained are shown in Table II, where it can be seen that, with a 5% significance level, the data corresponding to ESD and CO learning goals do not reject the null hypothesis that states that the means are statistically equal, and thus they can be considered independent of the factor 'academic year'. In other words, it cannot be stated that the lab course sessions introduced in this paper have had a statistical influence on the students' learning concerning ESD and CO. This really makes sense because the design of embedded systems is only covered by the



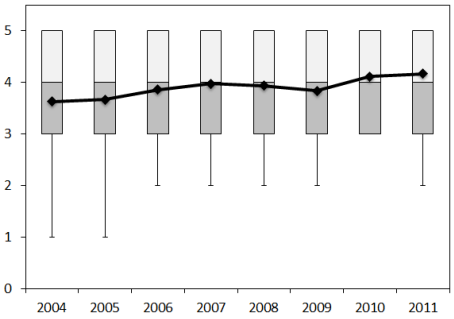
Q1 Did you take this course because you like video-game-consoles?



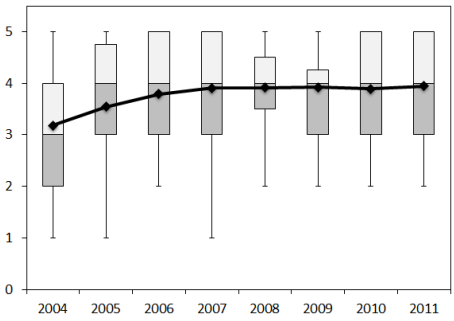
Q2 Is the GBA an efficient tool to learn about embedded systems?



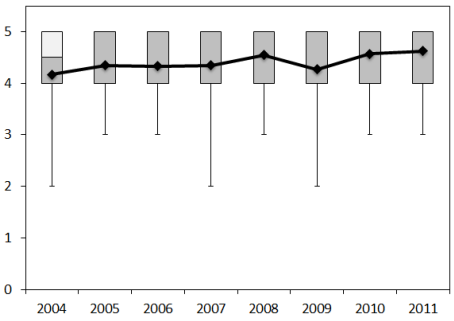
Q3 Do you think the possibility of solving lab exercises at home is interesting?



Q4 Are the laboratory assignments worthwhile?



Q5 Is the assessment method adequate?



Q6 Would you recommend the course to other students?

Fig. 5. Survey responses from 2004–11 (mean, sample minimum, lower quartile, median, upper quartile and sample maximum). Each question can be answered with a Likert scale from 1 to 5, for which 1 = strongly disagree, 2 = disagree, 3 = unsure, 4 = agree, 5 = strongly agree.

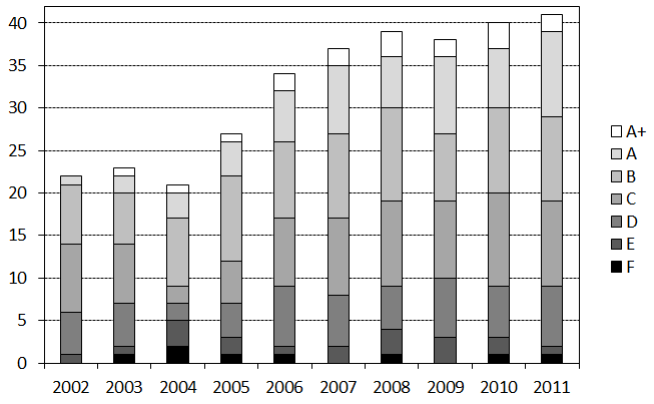


Fig. 6. Course enrollment 2002-11. Each bar is broken down according to grades obtained by students.

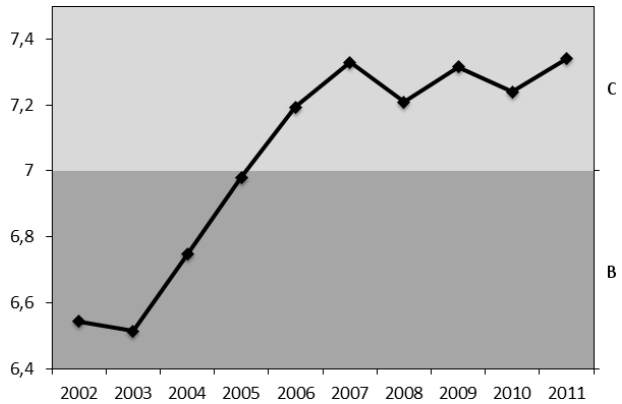


Fig. 7. Average score evolution 2002-11.

TABLE II
ONE-WAY ANOVA TEST RESULTS FOR THE INFLUENCE OF THE ACADEMIC YEAR ON THE STUDENTS' LEARNING GAINS.

Learning goal		Sum of squares	df	Mean square	F	Sig.
ESD	Between Groups	105.555	8	13.194	1.391	0.200
	Within Groups	2579.992	272	9.485		
	Total	2685.547	280			
FD	Between Groups	251.326	8	31.416	3.379	0.001
	Within Groups	2529.062	272	9.298		
	Total	2780.388	280			
CO	Between Groups	30.210	8	3.776	0.355	0.943
	Within Groups	2896.797	272	10.650		
	Total	2927.007	280			
DIO	Between Groups	168.108	8	21.013	2.620	0.009
	Within Groups	2181.464	272	8.020		
	Total	2349.572	280			

lectures, so obviously it should not be affected by the introduction of the console in the laboratory, and as far as code optimization is concerned, despite the theoretical concepts are explained in the lectures, its practical implementation depends on how the students have applied these concepts in their final project (usually the development of a video game). Since most of these video games are quite simple, the console's resources seem to be enough to make the game run without the need of a deeper code optimization. Nevertheless, after several years it can be observed that the mean of the scores shows a slight upward trend, although without statistical significance.

However, the scores registered for FD and DIO learning goals do reflect a remarkable statistical dependency on the academic year. In order to assess the nature of this dependency more precisely, a contrast test was run between the scores obtained before and after the introduction of the consoles in the lab. The results, presented in Table III, demonstrate that if it is assumed that the variances of the different groups are not equal (and indeed they were not, according to the Levene's Test with a 5% significance level), the difference between the means of each group is statistically significant. Representing these scores with respect to the academic year (Figs. 8.a-b), this difference can be appreciated very clearly. It is apparent how, from 2004 on (the year the console started to be used in the lab), the scores have significantly improved for both learning goals; as these two goals represent 70% of the students' final score, it can be concluded that the introduction of the new platform in 2004 has positively influenced the students' final score.

VIII. Conclusion

This paper has described work carried out at the University of Granada, Spain, where the lab sessions of an embedded systems programming course were designed to use portable video game consoles and open-source tools to motivate students.

As the student responses have shown, the experience was quite successful. In general, students think that the use of a video game console to learn how to develop embedded software is quite appealing, and that the course lab sessions contents are worthwhile. The use of open source-tools is highly appreciated as well, since students are accustomed to using them, and also because they can be built to solve the lab exercises at home. These subjective results are somewhat confirmed by the remarkable enrollment increase, which has nearly doubled in the last four years, and by the students' average final score improvement, which has risen from C to B in the last few years.

As the students' opinion may be quite subjective, the final scores were further analyzed according to the students' achievement of the course's learning outcomes. As described in Section VII, it can be seen that students have significantly improved on firmware and driver development for embedded systems, which represents 70% of the final score, since consoles were introduced in the lab sessions, suggesting that the motivational character of this platform has influenced these results.

Acknowledgment

This work was partially supported by the Spanish CICYT Project SAF2010-20558 and by the University of Granada Innovative Teaching Project 04-03-08, awarded with an Innovative Teaching Honorable Mention in the 2008 University of Granada Innovative Teaching Awards.

References

[1] VDC Research. (2009) Embedded Processors & Chipsets: Global Market Demand Analysis. [Online]. Available: http://www.vdcresearch.com

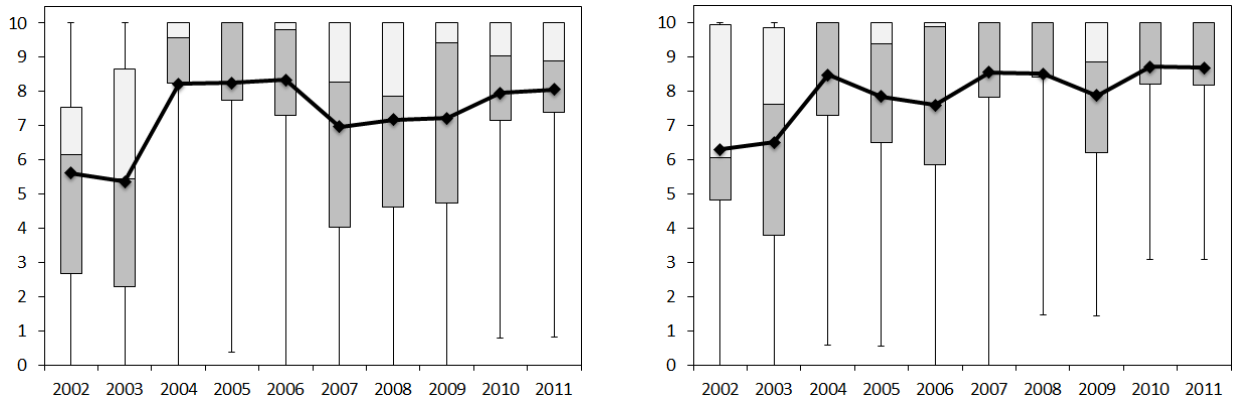
[2] Gartner, Inc. (2009) Dataquest Alert: Preliminary PC Market Results, Worldwide, 4Q08 and Full-Year 2008. [Online]. Available: http://www.gartner.com

[3] Semicast Research. (2009) Semiconductor Market Forecasts: 32/64-bit Microcontrollers, Embedded Microprocessors & DSPs. [Online]. Available: http://semicast.net

[4] A. L. Sangiovanni-Vicentelli and A. Pinto, "An Overview of Embedded System Design Education at Berkeley," *ACM Transactions on Embedded Computing Systems*, vol. 4, no. 3, pp. 472–499, August 2005.

[5] D. T. Rover, R. A. Mercado, Z. Zhang, M. C. Shelley, and D. C. Helvick, "Reflections on Teaching and Learning in an Advanced Undergraduate Course in Embedded Systems," *IEEE Transactions on Education*, vol. 51, no. 3, pp. 400–412, August 2008.

[6] L. Jing, Z. Cheng, J. Wang, and Y. Zhou, "A Spiral Step-by-Step Educational Method for Cultivating Competent Embedded System Engineers to Meet Industry Demands," *IEEE Transactions on Education*, vol. 54, no. 3, pp. 356–365, August 2011.



(a) Firmware development scores.

(b) Drivers and I/O scores.

Fig. 8. Learning outcomes improved by the proposed lab assignments (mean, sample minimum, lower quartile, median, upper quartile and sample maximum).

TABLE III
CONTRAST TEST TABLE BETWEEN THE SCORES BEFORE AND AFTER THE CONSOLE.

Learning goal	Assumption	Value of contrast	Std. Error	t	df	Sig. (2-tailed)
FD	Variances are not equal	2.252	0.551	4.087	55.894	0.000
DIO	Variances are not equal	1.823	0.555	3.282	53.294	0.002

[7] F. Vahid and T. Givargis, *Embedded System Design. A Unified Hardware/Software Introduction*. Danvers, MA: John Wiley & Sons, Inc., 2002.

[8] W. Wolf, *Computers as Components. Principles of Embedded Computing System Design*, 2nd ed. Burlington, MA: Morgan Kaufmann, 2008.

[9] J. Ganssle, *The Art of Designing Embedded Systems*, 2nd ed. Burlington, MA: Newnes, 2008.

[10] J. Labrosse, J. Ganssle, T. Noergaard, R. Oshana, C. Walls, K. Curtis, J. Andrews, D. J. Katz, R. Gentile, K. Hyder, and B. Perrin, *Embedded Software. Know it all*. Burlington, MA: Newnes, 2008.

[11] C. Walls, *Embedded Software. The Works*. Burlington, MA: Newnes, 2006.

[12] M. Barr and A. Massa, *Programming Embedded Systems: With C and GNU Development Tools*, 2nd ed. Sebastopol, CA: O'Reilly, 2006.

[13] J. Ganssle, *The firmware Handbook*. Burlington, MA: Newnes, 2004.

[14] T. Mitra, "Challenges in Designing Embedded Systems Courses," in *Proceedings of the 2006 Workshop on Embedded Systems Education*, Seoul, South Korea, October 2006, pp. 18–21.

[15] J. S. Chenard, Z. Zilic, and M. Prokic, "A Laboratory Setup and Teaching Methodology for Wireless and Mobile Embedded Systems," *IEEE Transactions on Education*, vol. 51, no. 3, pp. 378–384, August 2008.

[16] A. Stollenwerk, C. Jongdee, and S. Kowalewski, "An Undergraduate Embedded Software Laboratory for the Masses," in *Proceedings of the 2009 Workshop on Embedded Systems Education*, Grenoble, France, October 2006, pp. 34–41.

[17] NPD Group. (2009, February) 2008 Video-game Software Sales Across Top Global Markets Experience Double-Digit Growth. [Online]. Available: http://www.npd.com/press/releases/press_090202.html

[18] S. H. Kim and J. W. Jeon, "Introduction for Freshmen to Embedded Systems Using LEGO Mindstorms," *IEEE Transactions on Education*, vol. 52, no. 1, pp. 99–108, February 2009.

[19] J. O. Hamblen, "Using A Low-Cost SoC Computer and Commercial RTOS in an Embedded Systems Design Course," *IEEE Transactions on Education*, vol. 51, no. 3, pp. 356–363, August 2008.

[20] C. S. Lee, J. H. Su, K. E. Lin, J. H. Chang, and G. H. Lin, "A Project-Based Laboratory for Learning Embedded System Design with Industry Support," *IEEE Transactions on Education*, vol. 53, no. 2, pp. 173–181, May 2010.

[21] D. Brylow and B. Ramamurthy, "Nexos: A Next Generation Embedded Systems Laboratory," *ACM SIGBED Review*, vol. 6, no. 1, pp. 7:1–7:10, January 2009.

[22] D. Williams, N. Yee, and S. E. Caplan, "Who plays, how much, and why? Debunking the stereotypical gamer profile," *Journal of Computer-Mediated Communication*, vol. 13, no. 4, pp. 993–1018, July 2008.

[23] A. Voida and S. Greenberg, "Wii All Play: The Console Game as a Computational Meeting Place," in *Proceedings of the 27th Annual Chi Conference on Human Factors in Computing Systems*, April 2009, pp. 1559–1568.

[24] E. A. Boyle, T. M. Connolly, T. Hainey, and J. M. Boyle, "Engagement in Digital Entertainment Games: A Systematic Review," *Computers in Human Behavior*, vol. 28, no. 3, pp. 771–780, May 2012.

[25] L. A. Annetta, "Video Games in Education: Why they should be used and how they are being used," *Theory into Practice*, vol. 47, no. 3, pp. 229–239, 2008.

[26] C. Dennis, "From Edutainment to Serious Games: A Change in the Use of Game Characteristics," *Games and Culture*, vol. 5, no. 2, pp. 177–198, April 2010.

[27] D. Gouveia, D. Lopes, and C. V. de Carvalho, "Serious Gaming for Experiential Learning ," in *Proceedings of the 41st Annual ASEE/IEEE Frontiers in Education Conference*, October 2011.

[28] B. D. Coller and M. J. Scott, "Effectiveness of Using a Video Game to Teach a Course in Mechanical Engineering," *Computers & Education*, vol. 53, no. 3, pp. 900–912, November 2009.

[29] C. G. von Wangenheim, R. Savi, A. F. Borgatto, and A. Ferreti, "DELIVER! - An Educational Game for Teaching Earned Value Management in Computing Courses," *Information and Software Technology*, vol. 54, no. 3, pp. 286–298, March 2012.

[30] M. Arevalillo-Herraez, R. Moran-Gomez, and J. M. Claver, "Conquer the Net: An Educational Computer Game to Learn the Basic Configuration of Networking Components," *Computer Applications in Engineering Education*, vol. 20, no. 1, pp. 72–77, March 2012.

[31] S. Nooshabadi and J. Garside, "Modernization of Teaching in Embedded Systems Design – An International Collaborative Project," *IEEE Transactions on Education*, vol. 49, no. 2, pp. 254–262, May 2006.

[32] M. Samek. (2007, June) Building Bare-Metal ARM Systems with GNU: Part 1 – Getting Started. Embedded.com. [On-

line]. Available: http://www.eetimes.com/design/embedded/4007119/
[33] ——. (2007, July) Building Bare-Metal ARM Systems with GNU: Part 2 – Startup Code and Low-level Initialization. Embedded.com. [Online]. Available: http://www.eetimes.com/design/embedded/4026075/
[34] ——. (2007, July) Building Bare-Metal ARM Systems with GNU: Part 3 – The Linker Script. Embedded.com. [Online]. Available: http://www.eetimes.com/design/embedded/4026080/
[35] Nintendo of America Inc., *AGB Programming Manual*, 2001.
[36] B. Gatliff, "Embedding with GNU: Newlib," *Embedded Systems Programming*, vol. 15, no. 1, January 2002. [Online]. Available: http://www.eetimes.com/discussion/other/4024637/Embedding-with-GNU-newlib
[37] ——, "Embedding with GNU: Newlib part 2," *Embedded Systems Programming*, vol. 15, no. 1, January 2002. [Online]. Available: http://www.eetimes.com/discussion/guest-editor/4023922/Embedding-GNU-Newlib-Part-2