Received April 21, 2020, accepted May 7, 2020, date of publication May 12, 2020, date of current version June 2, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.2994196

Robotics vs. Game-Console-Based Platforms to Learn Computer Architecture

BEATRIZ GARCÍA FERNÁNDEZ[1], XAVIER DEL TORO[2], MARIA J. SANTOFIMIA[2], JAVIER DORADO[2], FELIX J. VILLANUEVA[2], DAVID VILLA[2], AND JUAN C. LOPEZ[2], (Member, IEEE)

[1]Science Education, Department of Pedagogy, Faculty of Education of Ciudad Real, University of Castilla-La Mancha, 13072 Castilla-La Mancha, Spain
[2]Computer Architecture and Networks Group, School of Computer Science, University of Castilla-La Mancha, 13072 Castilla-La Mancha, Spain

Corresponding author: Maria J. Santofimia (mariajose.santofimia@uclm.es)

This work was supported by the Spanish Ministry of Science, Innovation, and Universities under Grant PLATINO under Grant TEC2017-86722-C4-4-R, and in part by the Regional Government of Castilla-La Mancha under Grant SymbIoT SBPLY/17/180501/000334.

ABSTRACT The fields of robotics and game consoles offer an interesting and broad range of lab platforms with appropriate characteristics for teaching Computer Architecture concepts. This work analyzes the impact of one approach based on game consoles and another one based on robotics from a triple dimension: student motivation, acquired knowledge, and perception of the employed platform. The study has been carried out on a sample of 96 students using the Arduino-based robot and 75 students using the Nintendo-DS console. A mixed methodology is employed encompassing quantitative and qualitative approaches. Five instruments are used to measure the three aforementioned dimensions. Results show that despite both platforms performing similarly in the three considered dimensions (student motivation, acquired knowledge, and perception of the employed platform), the robotics platform does it slightly better than game console, based on the obtained average scores for the considered instruments. Despite this outperforming, motivation and perception decrease for the students using the robotics platform as result of some identified constraint. This suggests that changes are required in the organization of the lab sessions to promote teamwork skills and to overcome the lack of simulators to remove the obstacles hinting motivation and performance. However, a clear correlation between motivation and perception and acquired knowledge has not been identified on computer architecture. Implications of affordances and constraints of both platforms, types of activities, and their impact on results have been discussed.

INDEX TERMS Computer architecture, Arduino, Nintendo DS, game consoles, educational robotics.

I. INTRODUCTION

The Association for Computing Machinery (ACM) and IEEE Computer Society Joint Task Group on Computer Engineering Curricula consider the Computer Architecture and Organization area part of the body of knowledge for the curricula of Electrical Engineering, Computer Engineering, and Computer Science [1]. Despite its importance, this area has traditionally suffered from poor student performance [2]. The fact that most Computer Engineering or Computer Science students are more interested in software aspects than in those related to hardware [3] and those with hardware interests tend to choose Electrical and Electronic Engineering degrees is one of the reasons explaining the lack of engagement and the under performance in this area in the different degrees.

The implementation of the European Higher Education Area brought along new teaching methodologies and evaluation methods that are not performing as well as expected [4] what is increasing the pressure for new approaches that help improving students' motivation and performance.

Teaching Computer Architecture is therefore a challenging task. The role that the practical platform plays in the teaching process is essential, and many different approaches have been proposed to this end [5], [6] being a recurrent discussion whether simulators or virtual laboratories should be adopted instead of real platforms [7].

Any tool used for this purpose should be suitable to develop the called "computational thinking", defined as the skill needed to solve problems effectively and

The associate editor coordinating the review of this manuscript and approving it for publication was Tomás F. Pena.

efficiently in different contexts [8]. This skill is not necessarily linked to computers as is a daily life ability that every citizen needs [9], but is essential to understand and communicate computational concepts effectively [10]–[12] and programming adequately, a reason why it is a basic part of almost all engineering programs curricula [13]. Abstraction is particularly relevant in this sense to reach computational thinking at a high level [14], [15]. It is noticeable that students that follow courses of Computer Architecture are usually in the first course of the degrees, what implies they are frequently unexperienced with these fundamental skills. Many of them are not enough familiarized with suitable forms of notation, necessary to develop mental models [16], together with a lack of programming concepts and language syntax, what may lead to demotivation, hindering learning [13], so the pedagogical approaches used in teaching should be aimed at fill in these gaps. Robots and computer games reach this objective.

Traditional approaches for teaching Computer Architecture frequently leads to a poor performance of students, being far from a constructivist approach [17] in which the knowledge is actively constructed by the learner [18], [19]. In this vein, Papert and Harel in [20] coined the term constructionism, a derivative or constructivism, based on learning is much more meaningful when students create tangible or shareable artifacts, mobilizing their knowledge and skills [21]. Robots [22] and computer games [23], [24] are good examples of engaging artifacts to discover knowledge with educational interest in Computing Education.

The benefits that Nintendo DS brings as a lab platform for teaching Computer Architecture concepts have been demonstrated [25], [26] and experienced by the University of Castilla-La Mancha, pioneer in adopting this platform during the academic year 2007/2008, as a pedagogical resource. Despite being a great pedagogical asset, Nintendo DS is, at the moment, an obsolete gaming platform. Finding replacement is becoming more and more challenging. For that reason, the academic staff involved in teaching this course faced the dilemma of whether Nintendo DS should be replaced by another game console or change the approach to the currently hot-topic issue of educational robotics. The study presented here was conducted to collect evidences and make an informed decision on what approach should be implemented next: a game-console based or a robotics one.

Both robotics and game-console based platforms have demonstrated positive impact on student motivation [27]–[31]. Additionally, their hardware features are very interesting, in terms of provided resources and constrained complexity, for the purpose of teaching Computer Architecture. These two approaches therefore satisfy the Computer Engineering and Computer Science students demand of having attractive and real platforms for experiencing with Computer Architecture concepts [32]. However, to the best of our knowledge, there is a lack of works quantitatively assessing the impact that both approaches have on learning Computer Architecture. To fill this gap, this paper presents two specific platforms representing these two approaches: a fit-for-purpose Arduino-based robot and the Nintendo-DS console. Appropriate hands-on sessions have been designed to cover the content of the course Computer Structure and learning outcomes have been gathered, analyzed and compared from the perspective of the student motivation, knowledge, and perception of the employed platform.

II. EDUCATIONAL ASPECTS OF GAME-BASED AND ROBOT TEACHING APPROACHES

Despite the importance that practical training has on the learning process to give teaching a constructivist approach and promote learning, students very often find this experience compromised by the lack of appropriate equipment [33]. In this reality, the use of software-based simulators has become a common practice for undergraduate courses, especially in science and engineering disciplines. The advent of advanced computer graphics, augmented and virtual reality are leading to sophisticated virtual laboratories that provide a close-to-real experience overcoming the inconveniences associated to real laboratories (equipment failure and maintenance, associated cost, limited number, etc.) and allowing the students to focus on the most important aspects. However, these approaches are still at a very early stage and their use is commonly reserved for supporting initial steps. Still, the use of computer-based simulators has an important negative impact on the student motivation [34] who rather use traditional real equipment for more in-depth hands-on experiences [35].

It is therefore evident that a trade-off must be found between the positive impact of using real platforms and the overwhelming details that they might entail. This is what the work in [36] has referred to as the *professor's dilemma*. Students demand real platforms that provide them with a realistic picture of what is on the market, whereas on the other hand, market products are specifically designed to maximize market penetration and company profits, which make them, most of the time, inappropriate for learning purposes. An additional obstacle that must be faced when looking for a real platform for teaching Computer Architecture is the fact that students are polarized regarding their interest about software and hardware aspects. Finding a platform that fits these two polarized groups is not easy because those keener on hardware-related aspects might found as a too limited platform one that might be perceived as too cumbersome for those other more interested in software aspects [26].The solution to this dilemma consists in employing a platform with a fast learning curve but with enough challenging resources to meet the expectation of those more experienced students.

In this sense, the use of game-based approaches has different pedagogical advantages. Previous research found that a lack of initial affinity for playing computer games was not a barrier to enjoy the scientific experience of game construction [37]. These authors also highlighted game-based construction learning as a suitable approach to develop higher-order thinking and Computer Science abstraction skills, becoming an enjoyable approach for teaching Computer Science Education, even if they remark as a constraint the sophisticated level of programming skills knowledge needed.

Different approaches can be found in the literature that resort to game-based approaches for teaching Computer Architecture. The work in [38] describes how mobile educational games can be used to overcome the limitations of traditional classroom approaches, focusing in providing a fun, interactive and motivating way of discovering and experiencing with computer architecture concepts. Despite the strengths that this proposal has on how contents are delivered, limitations are identified as this is not appropriate for acquiring low-level programming skills. The work in [39] proposes a game using 16 cards, each of which has assembly instructions written in them, as a first approach for getting students familiarized with the course contents. This is a very interesting approach, but it is only intended to cover the first sessions of the course. Finally, the work in [25] focuses on the role that game-console-based project can play in learning the Input/Output subsystems. The work of Larraza-Mendiluze et al is totally in line with the approach presented here, nonetheless, the approach presented here is not only limited to the Input/Otuput subsystem, but it also covers the rest of subsystems comprising a computer.

An alternative tool for teaching Computer Architecture, which is also very appealing to students, is the use of robotics, and more precisely the so-called educational robotics. This approach has recently gained attention [40] as more evidences are being obtained demonstrating the positive results in learning [41], [42]. Due to the versatility of robot platforms [30], a wide range of applications can be pursued, turning this platform into a unique tool to target a wide audience, from primary and secondary school students up to higher education [43]. The use of robots in education has benefits as the development of thinking and social skills [44], are of high interest in all educational stages. Robots are also a powerful tool in the teaching and learning process due to their interactivity and flexibility [30], [45], and the wide variety of enjoyable hands-on experiences they provide [46].

Platforms like Arduino [47], [48], Lego Mindstorms NXT [49], [50], or Raspberry Pi [51], and programming environments such as Logo [52], [53] or Scratch [13], [54] are some of the most popular tools. These platforms and programming environments have, as their leitmotif, to provide simple, intuitive, and easy-to-use tools. The so-called maker movement or *making,Giannakos2017* is a new concept that has been recently coined to describe those activities that put the focus on the process of learning by constructing. Despite being a relatively new concept it is based in the well-known philosophy of *learning by making,Papert1991* and founded in a constructivist (and constructionist) approach of the teaching and learning proces [44].

The use of robotics in primary and secondary education is an emerging trend mainly due to the vast amount of information available online, thanks to the open-source community that is generally supporting some of these projects, as well as the attention that STEM (Science, Technology, Engineering, and Mathematics) subjects are receiving as enablers for skills such as the computational thinking [56]. Higher education is not unaware of this trend but educational robotics is being employed as a complimentary tool [30] because students, at this level, should have already acquired these skills. In this sense, the use of robotics has been mainly employed to promote motivation and student engagement [30], [31], [45], [57]. They have also pedagogical benefits in learning [58], as they are based on problem-solving and active thinking [59] especially in the way lessons have been designed in the present research. This process of problem-solving activates cognitive skills like representing (the external representation of a problem is transformed into an internal mental model), planning, executing and evaluating [60], besides the development of metacognitive thinking skills needed to solve problems [61].

Additionally, their hardware features are very interesting, in terms of provided resources and constrained complexity, for the purpose of teaching Computer Architecture. However, to the best of the authors' knowledge, there is no similar approach that employs educational robotics, as a lab platform, for teaching computer architecture concepts. These two approaches therefore satisfy the Computer Engineering and Computer Science students demand of having attractive and real platforms for experiencing with Computer Architecture concepts [32]. This work is also motivated by the lack of works quantitatively assessing the impact that both approaches have on learning Computer Architecture. To fill this gap, this paper presents two specific platforms representing these two approaches: a fit-for-purpose Arduino-based robot and the Nintendo-DS console. Appropriate hands-on sessions have been designed to cover the content of the course Computer Structure and learning outcomes have been gathered, analyzed and compared from the perspective of the student motivation, knowledge, and perception of the employed platform.

III. STUDY CONTEXT

Computer Structure is one of the courses that comprise the Computer Architecture area. This is taught as part of the Computer Engineering Degree program in the University of Castilla-La Mancha, Spain. This course is intended to lead students into basic computer architecture concepts such as the memory system, the instruction set architecture or the input/output system. To this end, the course is organized in 6 European Credit Transfer System (ECTS) 4 of which are devoted to theory and 2 for labs. These 6 ECTS will be taught for 14 weeks, having 3 sessions per week of 90 minutes each. One of these three weekly sessions will take place in the lab and the remaining two will be devoted to theoretical concepts. The syllabus is organized in five lessons as stated in Table 1. The first two sessions of the hands-on experiences will be intended to an introduction to the C programming language. The last 4 sessions will be devoted to work on the

TABLE 1. Course contents organized by sessions.

Lesson	Lecture sessions	Hands-on sessions
1. Introduction	2 sessions	1 session
2. Memory	5 sessions	2 sessions
3. Machine and assembly language	7 sessions	2 sessions
4. Datapath and control unit	7 sessions	0 sessions
5. Input/Output	6 sessions	3 sessions

course project in which the different contents of the course will be put into practice.

This course is taught during the second semester of the first year. Three more additional courses are taught as part of the computer architecture core knowledge, namely: Computer Technology (first semester of first year), Computer Organization (first semester of second year), and Computer Architecture (first semester of third year). The course is taught in Spanish but also in English for those students going for the bilingual mention. The Computer Structure course has been selected for this experiment mainly because it has traditionally suffered from the poorest performance, out of all courses comprising the Computer Architecture area. The failure rate of this course yearly reaches around 50%.

The learning objectives of this course (like the rest of the degree) have been certified by the National Agency for Quality Assessment and Evaluation and Accreditation from Spain and are listed below:

- Objective 1: To understand the principles of computer architecture.
- Objective 2: To know the organization of the CPU, identify the functional units, and explain their role in the execution of the instructions.
- Objective 3: To know the organization of the Input/Output subsystem and its interface with the CPU.
- Objective 4: To relate the evolution of CPU architecture and instruction sets. To identify the differences between the CISC and RISC philosophies.
- Objective 5: To program a computer at a low level.
- Objective 6: To learn, through practice, the structure and programming of a basic computer.

To address these objectives the course is organized in five lessons, as described in Table 2, addressed through lectures and hands-on experience.

The first lesson consists in an introduction to the structure of a computer. This lesson will offer a macroscopic vision of what will be studied, in more detail, in the later lessons and which makes up the so-called digital computer. The classical architecture proposed by von Neumann serves to briefly introduce the fundamental components of a computer, as well as the basic structures used in its interconnection. As this is an introductory lesson, the corresponding sessions of the laboratory were dedicated to familiarizing the student with the Arduino Zero platform or the Nintendo DS and their respective development environments.

The Nintendo-DS group faces this lesson from the perspective of the a von Neumann architecture although, as processors themselves, the ARM7 implements a von Neumann architecture whereas the ARM9 implements a Harvard one. The ARM9 has two Tighly Coupled Memory (TCM), one for data and one for instructions. The DTCM, with a size of 16 KiB, is where the stack is located. For this lesson, students are prompted to display several backgrounds. The size of the background and the representation mode employed (framebuffer, rotoscale, or tiled) determines whether the background can be properly displayed or not (run out of memory). This exercise therefore makes students aware of the different type of variables, different memories and different types or architectures. Finally, the characteristic parameters of a computer are also explored through the analysis of the platform itself.

The Arduino-based robot group, on the other hand, works with a Harvard architecture. This is however a *hybrid* Harvard architecture meaning having separeted buses for data and instructions only under specific situations. However, there is no need, at this stage to discuss whether this is a pure or hybrid Harvard architecture. Students experiment with a basic "*Hello, World!*" program in which the string (Hello, World!) is provided in different ways, each of them with different implications. This introduces students into the use of different type of variables (static or automatic) and different type of memories (one for data and one for instructions). Finally, the characteristic parameters of a computer are also explored through the analysis of the platform itself.

This lesson is therefore addressed through different graphical representation modes in Nintendo DS and, in Arduino, through different versions of a basic "*Hello, World!*". First sessions are therefore more attractive, in terms of achieved results (different graphics represented in the console), for the Nintendo DS groups. The Arduino robot-based group, on the contrary, does not experienced any interaction with the robot itself. The serial console and debugger (to explore the memory) are the only elements employed during this session.

The second lesson studies the memory system. The memory system of the Nintendo DS is probably its most valuable asset for teaching Computer Architecture. The video memory (VRAM) plays an essential role for the graphics programming which requires a deep understanding of memory organization. Framebuffer, rotoscale and tiled are the representation modes supported by Nintendo DS. Each of these involves totally different forms of dealing with the memory for graphics representation purposes. For example, the framebuffer mode maps screens to specific memory regions. Writing to this memory region has a direct effect on what it is being displayed in the screen. On the other hand, the tiled mode considers the screen as a matrix of tiles, each of which is a bitmap of 8×8 pixels. This matrix is provided as references to a memory where the tiles are stored. Students in this group are introduced to the graphics programming modes and different exercises were proposed to this end. Understanding direct and indirect addressing modes are essential for an effective use of the different graphic modes.

TABLE 2. Lesson contents and proposed activities for each lab platform.

Lesson	Content description	Activities for Nintendo DS	Activities for Arduino robot
1. Introduction	What is a computer? Functional description: von Neumann architecture. Origin and historical evolution of computers. Characteristic parameters of the computers.	Representation of different backgrounds (basic ones, like the whole screen in blue or color grading) created in the program and stored in the stack (as dynamic variables).	Implementation of different "Hello, World!" versions. The string is provided in different ways (as text, static or dynamic variabler) and depending on that, it will be located in different memory regions.
2. Memory	Memory hierarchy.	The same background is represented using different graphic modes (framebuffer, rotoscale, and tiled). Depending on the employed mode, different memory types and regions are employed (tile memory, color palette memory, framebuffer, etc.).	Implementation of different programs. The binary code, no optimized by the compiler, will be explored through the remote debugger and guided by the ABI rules.
3. Machine and assembly language	Architecture and instruction set (ISA). Addressing modes. Application Binary Interface (ABI). RISC and CISC architectures.	Specific programming exercices are provided to the students to experience both the ISA of the ARM and the ABI rules established for that architecture. The debugger and the console emulator are employed to this end.	Specific programming exercises will be proposed to students to get familiar with both the ISA and ABI of the ARM architecture. Moreover, students will programming in assembly by exploiting the asm function to embed assembly instructions into a C programm.
4. Datapath and control unit	Description of the data path. Functions of the Control Unit. Execution phases of an instruction. Micro-instructions and control signals.	This lesson is covered with the activies undertaken during the previous lesson.	
5. Input/Output	Input-output modules. Input-output modes: pooled, interruptions and direct memory access (DMA).	Implementation of a game in which different input/output strategies (pooled, interruptions, and DMA) are considered.	Implementation of a line-follower program in which different pheripheral devices are involved.

Regarding the Arduino-based robot, unlike general-purpose computers, the Arduino Zero (as a typical microcontroller) is designed to be dedicated to a single task, which is why a Harvard architecture is so convenient. This group explores the memory system resorting to the Application Binary Interface (ABI) as well as the different regions that determine the state of running program. In this sense, the status of a running program is determined, at any particular time instant, by the content of the memory and registers and, more specifically, by the content of each of the sections into which the memory is divided (text, data, .bss, stack and heap). In the case of the Arduino Zero, these sections will be mapped into specific physical memory spaces (Flash and SRAM). Students are offered a series of simple programs to explore these sections. Although the use of the debugger (KDbg [62]) may be useful, it is generally preferred that students use pointers and print memory addresses. In this manner, they also become familiar with the use of C pointers. In addition, the ABI and, more specifically, the machine's alignment rules and endianess will be used to explore the platform memory system.

Lesson 3 studies the set of instructions that a processor can execute, known as Instruction Set Architecture (ISA). The ISA is closely related to the ABI rules studied in the preceding lesson. A fundamental question for an ISA is to know the different addressing modes that are available. During this lesson different modes will be studied and we will find out which of them are available for the ARM architecture and how they are implemented in its ISA.

The machine and assembly language is experienced through the use of the KDbg debugger, for both groups (Nintendo DS and Arduino-based robot). KDbg is just a front end of the GDB debugger with support for remote debugging. It offers an easy-to-use tool for students to explore the assembly language associated to C code. From a pedagogical perspective, the use of a debugger offers a very powerful mechanism for students to analyze in detail the Application Binary Interface (ABI), the role it plays, and how it is connected to the Instruction Set Architecture (ISA). Students, for example, are prompted to verify that the binary code generated by the compiler complies the ABI rules when it comes to procedure calls. To this end, students resort to the use of a debugger like KDbg to explore not only the assembly code in a friendly manner, but also the content of the processor registers. The KDbg debugger lists the assembly code associated to a C statement in the source code. Students can therefore get familiar with the assembly code without being overwhelmed with all the details involved in an assembly file.

It is important to note that writing an entire program in assembly code has an associated complexity that is beyond the scope of the course. However, having a general understanding of the basics of assembly programming is useful for code optimization and debugging tasks. For this reason, students regardless of the employed platform are provided with a series of code fragments that they will have to analyze, using the functionality offered by the KDbg debugger.

The Arduino-based robot group explores the ISA of the Cortex M0+ in more detail. Students are prompted to write assembly sentences using the "asm" function to introduce delays (with the "nop" operation) or to move data into registers, verifying aftewards the content of the loaded register with the debugger.

Lesson four is not experienced first hand with any of the platforms. On the contrary the content of this lesson is approached from the point of view of knowing the different elements that make up the datapath (ALU operations, registers, stack, buses, etc.) and, therefore, the activities developed along the previous sessions have already introduced all these concepts. This lesson, as such, does not involve concrete hands-on experiences but we consider them part of the previous activities.

The fifth lesson deals with the input/output system, which along with the processor and memory, is one of the essential parts of a computer. The input/output system will, in essence, enable the communication of the computer with the external world (people and other devices). Both platforms have an interesting list of peripheral devices. For example, the Nintendo DS has two 2D graphics engines, a 3D graphics engine, a touchscreen, 16 sound channels, stereo speakers and microphone, timers, etc., while the Arduino-based robot has a list of sensors and actuators such as ultrasound, infrared, light, buzzer, servo or mini-servo.

Nintendo DS employs a memory-mapped input/output system, meaning that handling the peripheral devices of the Nintendo DS basically consists in reading and writing from certain memory positions. The input/output registers are therefore mapped into specific memory positions that students are prompted to identify. To this end, the library header files are shown as the resource from which this type of information is provided.

Finally, the three input/output modes (programmed, interrupt and DMA) are presented through different examples. In this sense, the Nintendo DS is an excellent hardware to experiment with these three modes. Students are requested to write a simple game in which the three modes are employed. The programmed mode is experienced through the use of keys, the interrupt through the use of timers and graphics scrolls whereas the DMA is employed for loading graphics from memory into the screens. The benefits of using DMA is clearly experienced when loading backgrounds in the Nintendo DS screens. The use of DMA leads to instant loading of graphics whereas the use of the processor (not the DMA) leads to a gradual load of the graphics that it is easily perceived by the human eye.

The Arduino-based robot group has a first contact with each of the sensors and actuators of the platform. After the isolated experimentation, students are requested to integrate as many of them as possible into a line-follower robot. Advanced functionalities are considered to extend the line follower. For example, the robot can be guided by following a source of light or it can play different sound frequencies according to certain events. Students are prompted to extend the functionality as desired, employing as many peripheral as possible. Students soon realize that dealing with different devices call for more advanced management mechanisms. This leads them to consider an interrupted input/output, as an alternative to the programmed one, when tasks involving several devices cannot be attended simultaneously.

Table 3 summarizes the main affordances and constraints of the two platforms employed in this experiment: the Arduino-based robot and the Nintendo DS. These affordances and constraints are analyzed in the light of the content of the lessons comprising the course.

In the overall, these lessons are covered during the fourteen lab sessions available all over the course. The three first lessons are employed to introduce students into the C programming language. There is one lab session per week plus two in-class lectures.

IV. METHODOLOGY

The main aim of this study was to evaluate the convenience, from a pedagogical and motivational perspective, of using either robotics or game-consoles based platforms for teaching Computer Architecture. Provided the lack of studies evaluating and comparing these two strategies, this paper pursues the following specific research goals:

- To compare both strategies in terms of student motivation, acquired knowledge, and perceived usefulness.
- To identify the pedagogical strengths and weaknesses of both approaches.

A. SAMPLE DESCRIPTION

The sample has been selected by intentional non-probabilistic sampling. First-year students of the Degree in Computer Engineering of the University of Castilla-La Mancha, Spain compose the sample. It is divided into two cohorts (1 and 2) both for the same academic year (2016/2017). Each of them uses a different lab platform for the development of hands-on experiences, working in groups of 2 to 4 people. Cohort 1 (using Arduino-based robot) consists of 96 subjects (82 males and 14 females; mean age: 19.20 years, SD $=$ 1.63 years) and the 2 cohort (using Nintendo DS console) of 75 subjects (67 males and 8 females; mean age: 20.25 years, SD $=$ 1.69 years).

B. STUDY DESIGN

The methodology used is based on the mixed paradigm, since the research has used qualitative and quantitative instruments to collect data. Two of the instruments were applied both at the beginning and at the end of the course (pre-test and post-test) and the platform perception instruments (Arduino and Nintendo DS) and the inquiry instrument that were only applied at the end of the course.

C. INSTRUMENTS

The data collection process has been automated using forms. One form has been created for each of the instruments that comprise the study. The Microsoft Forms platform has been employed for being this platform integrated into

TABLE 3. Affordances and constraints of both platforms.

Lesson	Affordances		Constraints	
	Nintendo DS	Arduino-based robot	Nintendo DS	Arduino-based robot
1. Introduction	The Nintendo DS processor suffices to introduce the notion of von Neumman and Harvard architecture since its processor is comprised of two other processors (ARM7 and ARM9). ARM7 is von Neumann architecture and ARM9 is Harvard (throuhg its TCM) with one cache memory (DTCM) for data and another for instructions (ITCM). Moreover, the stack is located in the DTCM so it is easy to combine graphic representation hosted in the stack (dynamic variables) to run out space and fail to represent the graphic background.	It is a Harvard architecture with a simple memory system. Basic programming exercises are enough to experience the theoretical foundation behind Harvard architectures (data and instructions are located in different memories).	Students need to be introduced to the different graphic representation modes (framebuffer, rotoscale, and tiled) so this introduces an additional complexity that it is not directly related to the course content.	The robot is not employed for the proposed exercises. Only the Arduino Zero board is required.
2. Memory	The Nintendo DS memory system is very complete and the memory map is not done by default so students need to be aware of the memory need they have, enable the required banks and map these banks to specific memory position ranges	Simple programming exercises are enough to experience the ABI rules regarding memory. The different memory regions of a program in execution are straigh forward verificable.	The memory system can be complex for first-year students. The process of memory bank enabling and mapping is done by code and it is easy that they just follow the steps without understanding the real meaning of what they are doing.	Again, students only work with the Arduino Zero board. The robot is not yet involved in any activity. They feel these activities are decoupled from their lab platform.
3. Machine and assembly language	The ISA of the ARM is a good example to introduce students to assembly language. The reduced set of instructions is also very useful for pedagogical purposes. The ABI for embedded C can be easily verified in both platforms (forcing the ARM compiler to not to optimized the binary code).		The remote debugging is done over a Nintendo DS simulator, not the real console. The simulator is very close to the real hardware but there are some inconsistences (for example, the stack size is not considered and there is not problem in loading any amount of data in the DTCM).	The configuration and execution of remote debugging is not trivial. Students needs to understand the role of the OpenOCD as the server supporting the communication between the running program being debug and the debugger.
4. Datapath and control unit	The main aspects elements of the datapath (general purpose or specific registers) can be explored through the debugger.	The datapath of the Cortex M0+ is public and therefore well known. The debugger is useful to experience with specific purpose registers like the PC, SP or LR.	The Nintendo DS processor is known from the information published by the homebrew community. This platform is not appropriate for experiencing aspects related to the control unit or the ALU.	This platform is not appropriate for experiencing aspects related to the control unit or the ALU.
5. Input/Output	The Nintendo DS offers a rich and varied set of peripheral devices that can be employed in the implementation of a game. The memory-map input/output is easy to understand and to verify. The role of the DMA is also easy to experience through the representation of different graphics.	Most of the peripheral devices involve an interaction with the environment. Therefore, they are affected by external conditions (light, surface, etc.) what lead students to be aware of how the interaction takes place. Some of the robot sensors and actuactors need to be properly set up so students adjust the robot peripherals.	The implementation of a game might deviate students from the end goal which is to employ different input/output strategies and peripheral devices. Efforts should not therefore be addressed to achieve a nice graphics or complex game logics. On the contrary the focus should be put on the course contents.	The use of real sensors and actuators might sometimes impose a burden that some students find hard to handle. For example, the light intensity under wich a robot is working might affect on the line-follower logic behaviour. Sensors should be calibrated accordingly and servors desadjust easily.

the corporative network. This has simplified the process of student identification as well as data collection and processing stage.

Students were informed, before accessing the questionnaires, about the aim of the study. Only the professors directly involved in the study have has access to personal data (name, gender, age, etc.) which have been anonymized for their statistical analysis by assigning an ID number to each subject. Students were also told that the anonymization process would take place prior to the statistical analysis of the data. This was intended to minimize bias in the responses of students who believed that the outcome of their responses might affect their final grade. Additionally, students were assured before the start of a questionnaire that none of the answers provided would affect their final grade.

The ordinal nature of the variables in the Likert scale instruments determined the use of polychorical correlation matrix for validation purposes. Thus, the reliability of the Likert scales instruments have been tested through the ordinal Chronbach's alpha. All the Likert scale instruments were also tested in terms of content and construct validity.

Motivation and acquired knowledge have been considered as it seems that the first one has an impact in the second [63], specially when computer technologies are involved [64], [65].

1) INSTRUMENT 1: STUDENT MOTIVATION

The work in [48] analyses the convenience of state-of-the-art instruments for measuring the students' attitude towards introductory programming. The same conclusions apply to our study since learning low-level programming is one of the main competences of the Computer Structure course. We have therefore put the focus on low-level programming.

Among the considered instruments the TAM model [66] is considered to most appropriate one because, apart from being successfully used in [48], it was originally envisioned to estimate the acceptance of innovative information systems. Similarly to the work in [48] a translation was conducted precising *low-level programming* when a reference to programming was made.

The main constructs of the TAM model are: the perception of usefulness, the perception of ease of use and the behavioral intention to use. This model was therefore employed to inspired this instrument that measures the student motivation towards the low-level programming skill about to be acquired during the course. The questionnaire, listed in Table 4 is comprised of 10 questions which use a Likert scale with 5 possible answers, as known: Totally agree (5), agree (4), neither agree nor disagree (3), disagree (2), totally disagree (1).

Despite being based on a validated questionnaire [48], the validity and reliability parameters have been calculated for our study. The content validity was tested by a panel of ten experts on Computer Architecture, and they agreed that the items were suitable for the research, well written and easy to understand. The construct validity was tested by an Exploratory Factor Analysis (EFA). The Kaiser-Meyer-Olkin sample suitability test (0.758) and the Bartlett's sphericity test ($p<0.01$) confirmed the pertinence of an EFA. The EFA revealed a structure in which items are grouped into three factors, according to the theoretical construct considered for its development: Factor 1: the perception of usefulness (items 1, 2, 3, 4), Factor 2: the perception of ease (items 5, 6, 7), and Factor 3: the behavioral intention to use (items 8, 9, 10). The reliability of the scale was tested by the ordinal Chronbach's alpha. The results indicated a very good reliability of the scale [67] as a whole (0.78) and by factors (0.77 for factor 1, 0.72 for factor 2 and 0.76 for factor 3).

2) INSTRUMENT 2: ACQUIRED KNOWLEDGE

This instrument consists in 10 multiple-choice questions (with four answers) designed *ad-hoc* for the present study, as listed in Table 5. This instrument was addressed to measure the achievement of the learning objectives pursued by the course, listed in Section III: objective 1 (questions 1, 2, 3, 4), objective 2 (questions 2 and 3), objective 3 (questions 3 and 9), objective 4 (questions 2 and 10), objective 5 (questions 5, 6, 7), objective 6 (8, 9, 10).

TABLE 4. Questionnaire employed to study the student motivation.

Perceived Usefulness of programming
Item 1. Knowing low-level programming will help me find a job
Item 2. It will be easier to finish my studies if I know low-level programming
Item 3. During my studies it will be useful for me to know low-level programming
Item 4. Knowing low-level programming will be useful for my work
Perceived Ease of programming
Item 5. It is easy for me to learn low-level programming
Item 6. It is easy to make the low-level program do what I want to
Item 7. Low-level programming is easy
Intention to program
Item 8. I intend to use low-level programming during my studies
Item 9. In my job I will code low-level programs that will be helpful for me
Item 10. Low-level programming will form part of my profession

The validity of this instrument was carried out by a panel of 10 experts (8 professors of the Computer Architecture and Technology area, 1 undergraduate student, and 1 postgraduate student). Following the first revision, the proposed amendments were undertook and a second version of the questionnaire was resubmitted to this panel of expert who finally consider it valid. This version was applied through a pilot test of 10 students of the Degree in Computer Engineering, proving that there were no doubts in the understanding of any of the items. Due to the dychotomical character of these variables, the reliability of the instrument was calculated using the ordinal Chronbach's alpha. The result (0.67) indicated a good internal consistency of the instrument and makes it suitable for the research.

Additionally, this questionnaire also collects information about students' previous experience. To this end four questions will formulated: 1) Do you have previous experience in programming microcontrollers?; 2) If so, which ones?; 3) Do you have previous experience in computer programming?; 4) If so, what languages have you used?

3) INSTRUMENT 3: STUDENT PERCEPTION OF THE ARDUINO-BASED ROBOT PLATFORM

Students' perception of the computing platforms used in teaching provides important information for assessing the teaching practice [68]. This instrument is intended to evaluate the student perception of the Arduino-based robot platform. The questionnaire proposed in [48] inspires the current instrument to evaluate the use of the Arduino board in the context of low-level programming.

The instrument is comprised of 10 items which use a Likert scale with 5 possible answers, as known: Totally agree (5), agree (4), neither agree nor disagree (3), disagree (2), totally disagree (1). This questionnaire, listed in Table 6, has been inspired in [48].

TABLE 5. Questionnaire employed to measure the learning outcomes with correct answers in bold.

	Option a	Option b	Option c	Option d
1. What kind of electronic circuits make up the basic structure of a digital computer?	Analogue circuits	Combined logic circuits	Sequential logic circuits	**Combined and sequential logic circuits**
2. On which part of a computer are the calculations performed?	**In the Central Processing Unit (CPU)**	In the memory	In the registers	In the CPU and memory
3. Which are the basic elements of a computer?	The processor and the memory	**CPU, memory and input/output**	The ALU, the registers and the communication buses	The peripheral devices and the processor
4. What characterizes a micro-controller?	The micro-controller integrates in a single chip a CPU, volatile and non-volatile memory and specific input and output peripherals	Micro-controllers are usually employed in specific applications where a single program is run	Micro-controllers usually have a reduced cost and consumption	**All the answers are correct**
5. The size of a memory is determined by:	**The word size and word number**	The number of available addresses	The number of available memory chips	The number of memory modules that comprise it
6. A memory of 1KiB:	It has 1024 memory positions of 1 byte each position	It has 2^10 memory positions of 1 byte each	It can be addressed using 10 address lines	**All the answers are correct**
7. Comparing a RAM memory and a Flash memory:	Both memories are volatile	RAM memory is non-volatile and Flash memory is volatile	Both memories are non-volatile	**RAM memory is volatile and Flash memory is non-volatile**
8. Regarding the time required to access the memory:	They are always very fast and do not affect the execution of a program	**It depends on the position of each memory type in the memory hierarchy.**	They do not have any influence on the execution time of an instruction	They are not related to the cost and size of the memory
9. Regarding the computer buses:	These are the external wiring of the computer	**These are the lines of communication between the different functional units that make up the computer**	They are only used to communicate with peripherals	None of the above answers is correct
10. The instructions that a computer executes:	They are stored in the main memory	They always have the same duration in number of clock cycles	There can be different types: arithmetic, logic, jump, data transfer,...	**The first and third options are correct**

TABLE 6. Questionnaire employed to study the student perception of the Arduino-based robot platform.

Perceived Usefulness of the Arduino-based robot
Item 1. When low-level programming for Arduino-based robot I learn more
Item 2. With the Arduino-based robot I learn better low-level programming
Item 3. The Arduino-based robot helps me understand low-level programming
Item 4. I learn more quickly when working with the Arduino-based robot
Perceived Ease of Use of the Arduino-based robot
Item 5. It is easy to work with Arduino-based robot
Item 6. The Arduino-base robot is easy to use.
Item 7. It is easy to low-level program the Arduino-based platform
Perceived Enjoyment when using the Arduino-based robot
Item 8. Lab sessions with the Arduino-based robot are more entertaining
Item 9. I have fun working with the Arduino-based robot
Item 10. Working with the Arduino-based robot is more enjoyable

This instrument is based in a previous one, already validated, but it was necessary to validate it in the context of the present research. Firstly, the content validity was tested by a panel of experts on Computers Architecture, who agreed in the suitability of the items for the research purposes, written expression and ease of understanding. Secondly, the construct validity was tested by an EFA. The pertinence of carrying out an EFA was confirmed by the Kaiser-Meyer-Olkin sample suitability test (0.874) and the Bartlett sphericity test ($p<0.01$) results. The EFA revealed a construct structure of three factors, according to the theoretical construct considered for its development: Factor 1: Perceived usefulness of the Arduino board (items 1, 2, 3, 4), Factor 2: Perceived Ease of Use of the Arduino board (items 5, 6, 7), and Factor 3 (items 8, 9, 10). The reliability was tested by the ordinal Chronbach's alpha, due to the ordinal nature of the variables. The results indicated an excellent reliability of the scale [67] as a whole (0.9) and by factors (0.88 for factor 1, 0.85 for factor 2 and 0.90 for factor 3).

4) INSTRUMENT 4: STUDENT PERCEPTION OF THE NINTENDO-DS PLATFORM

This instrument is intended to evaluate the student perception of the Nintendo-DS platform. The questionnaire proposed in [48] has been adapted to refer to Nintendo DS instead of the Arduino board, as in its original version. This questionnaire is listed in Table 7.

The instrument is comprised of 10 items which use a Likert scale with 5 possible answers, as known: Totally agree (5), agree (4), neither agree nor disagree (3), disagree (2), totally disagree (1).

The instrument was validated in terms of validity and reliability for this research case study. The content validity was tested, as in the previous cases, by a panel of ten experts on computers education. They agreed that all the items were pertinent and well written, easy to understand and to reply. The construct validity was tested with an EFA. The Kaiser-Meyer-Olkin sample suitability test (0.841) and the Bartlett sphericity test ($p<0.01$) confirmed the pertinence of carrying out an EFA. The results confirmed that the items were grouped in three factors, according to the theoretical construct considered in the design: Factor 1: Perceived usefulness of the Nintendo DS platform (items 1, 2,3, 4), Factor 2: Perceived Ease of Use of the Nintendo DS platform (items 5, 6, 7), and Factor 3: Perceived Enjoyment when using the Nintendo DS platform (items 8, 9, 10). The reliability of the scale was confirmed by the ordinal Chronbach's alpha, due to the ordinal nature of the variables. The results indicated a very good reliability of the scale [67] as a whole (0.81) and by

TABLE 7. **Questionnaire employed to study the student perception of the Nintendo DS platform.**

Perceived Usefulness of Nintendo DS
Item 1. When low-level programming for Nintendo DS I learn more
Item 2. With Nintendo DS I learn better low-level programming
Item 3. Nintendo DS helps me understand low-level programming
Item 4. I learn more quickly when working with Nintendo DS
Perceived Ease of Use of the Nintendo DS
Item 5. It is easy to work with Nintendo DS
Item 6. Nintendo DS is easy to use.
Item 7. It is easy to low-level program the Nintendo DS platform
Perceived Enjoyment when using the Nintendo DS
Item 8. Lab sessions with Nintendo DS are more entertaining
Item 9. I have fun working with Nintendo DS
Item 10. Working with Nintendo DS is more enjoyable

factors (0.75 for factor 1, 0.70 for factor 2 and 0.74 for factor 3).

5) INSTRUMENT 5: QUALITATIVE EVALUATION OF THE EMPLOYED PLATFORM

This instrument consists of the following open question: "*Briefly tell us your impressions, assessment, or suggestions for the future about the course and the hands-on experience with the Arduino-based robot/Nintendo DS*". This instrument was applied to a panel of 10 experts composed of undergraduate and postgraduate students and professors from the area of Computer Architecture and Technology. The instrument was applied in a pilot test with 10 students.

D. DATA ANALYSIS PROCEDURE

The SPSS v. 23, R and Factor software were used for data analysis. The Kolmogorov-Smirnov test (as the sample is larger than 50 subjects) was applied to the variables corresponding to the different instruments based on the Likert scale. Since these variables do not meet the normality criterion ($p<0.05$), non-parametric statistics have been used [69]: Mann Withney's U test for independent samples and Wilcoxon test for related samples. The analysis was complemented with the effect size with r, due to the non-normality of the variables, but also the mean and standard deviation as indicative descriptors.

V. RESULTS

A. STUDENT MOTIVATION

When the inter-group analysis is done using the Wilcoxon test for related samples (see Table 8), in the Arduino group it turns out that in 4 of the 10 items there are statistically significant differences between the pre and post test (items 3, 5, 9, 10). In these four items they score more in the pre-test than in the post-test (they are demotivated in these four aspects). However, in the Nintendo-DS group, statistically significant differences appear in 3 of the 10 items (1, 2, 6), and in all three, this group scores more in the post test than in the pre test.

The results listed in Table 9 of Mann Withney's U test (comparing independent samples) reveal that in the pre test both cohorts had statistically non-significant differences in scores on 6 of the 10 items (items 1, 2, 3, 4, 6, 7). In the first four and 7, the students in the Arduino cohort showed greater motivation, and in 6, less than those in Nintendo-DS cohort. In the post test, no statistically significant differences were found in any of the items of this instrument.

The analysis of the results measuring the student motivation towards the use of low-level programming shows a decrease in the perception of the usefulness for those using the Arduino-based robot. This decrease affects both to the use of low-level programming during their studies and professional career. Particularly interesting is the decrease experienced in item 5, that regards with the easiness of learning low-level programming. This same item does not suffer significant differences for the Nintendo DS group, although the post test score was lower, as an average score, than that obtained for the Arduino group (2.99 for the Arduino group versus 2.96 for the Nintendo DS group). For items 9 and 10, that regards with the perceived utility of low-level programming for the future professional career, a similar situation occurs as described for item 5: from a pre test with higher values in the Arduino group there is a statistically significant decrease despite the fact that the final scores are still higher (on average) than those obtained by the Nintendo DS group, in which there are no statistically significant changes between the pre test and post test.

Note that r gets low values (see Table 10) for both Nintendo DS and Arduino and therefore the effect of the intervention is not relevant in terms of motivation.

B. ACQUIRED KNOWLEDGE

The results of Mann Withney's U test show that there are no statistically significant differences between the two cohorts in either the pre test ($p = 0.153$) or the post test ($p = 0.319$).

Table 11 summarizes the learning outcomes comparing the score in the pre and post test, with scores in the range of 0 to 10. Although the differences found between groups are not statistically significant, in the pre test, the Nintendo DS group scores slightly better (median = 7.0, mean = 7.16, SD = 1.83) than the Arduino group (median = 7.0, mean = 6.84, SD = 1.80). In the post test, it is the Arduino group that scores better (median = 9.0, mean = 8.06, SD = 1.61) than the Nintendo DS group (median = 8.0, mean = 7.85, SD = 1.60).

However, when the inter-group analysis is done for related samples using the Wilcoxon test, it is evident that there are significant differences between the pre test and post test for both the Arduino group (p=0.000) and the Nintendo DS group (p=0.009). To deepen these results, the size of the effect has been calculated using the r coefficient for non-normal distribution variables [69]. The result shows a greater effect of the intervention in the Arduino group (0,32) than in the Nintendo DS group (0,58).

A comparison has been also carried out considering independent questions (see Table 12). Results of the Chi-Square test show that initially both cohorts showed non-significant

TABLE 8. Descriptive statistics and Wilcoxon Test results for motivation test.

	Arduino-based robot								Nintendo DS							
	Pre test			Post test					Pre test			Post test				
	Median	Mean	SD	Median	Mean	SD	Z	Asymptotic significance (2-tailed)	Median	Mean	SD	Median	Mean	SD	Z	Asymptotic significance (2-tailed)
Item 1	4	3,78	0,89	4	3,82	0,73	-0,170	0,865	4	3,59	1,00	4	3,92	0,78	-2,819	0,005
Item 2	4	3,63	0,95	4	3,85	0,78	-1,925	0,054	4	3,45	0,98	4	3,81	0,81	-2,611	0,009
Item 3	4	3,74	0,84	4	3,55	0,77	-2,101	0,036	4	3,56	0,83	4	3,55	0,83	-0,187	0,852
Item 4	4	3,61	0,91	4	3,47	0,85	-1,356	0,175	4	3,43	0,89	3	3,26	0,97	-1,575	0,115
Item 5	3	3,32	0,91	3,5	2,99	1,03	-2,916	0,004	3	2,93	1,06	3	2,96	1,01	-0,036	0,972
Item 6	3	3,31	0,85	3	3,36	1,03	-0,616	0,538	4	3,46	1,00	4	3,63	0,93	-2,132	0,033
Item 7	3	3,00	0,94	4	2,81	1,00	-1,742	0,081	3	2,76	0,93	3	2,79	0,90	-0,051	0,959
Item 8	4	3,68	0,85	3	3,49	0,87	-1,620	0,105	3,5	3,36	0,77	3	3,27	0,92	-0,926	0,355
Item 9	4	3,56	0,73	4	3,35	0,92	-2,052	0,040	3	3,09	0,83	3	3,15	0,95	-0,387	0,699
Item 10	4	3,59	0,92	3	3,30	0,88	-2,730	0,006	3	3,08	0,85	3	3,08	0,93	-0,023	0,981

TABLE 9. Mann Withney's *U* Test results for motivation test.

	Pre test		Post-test	
	Mann-Whitney's U	Asymptotic significance (2-tailed)	*Mann-Whitney's U*	Asymptotic significance (2-tailed)
Item 1. Knowing low-level programming will help me find a job	3181.500	.191	3304.000	.296
Item 2. It will be easier to finish my studies if I know low-level programming	3144.500	.149	3412.000	.812
Item 3. During my studies it will be useful for me to know low-level programming	3145.000	.141	3586.000	.963
Item 4. Knowing low-level programming will be useful for my work	3156.000	.215	3164.500	.196
Item 5. It is easy for me to learn low-level programming	2855.000	.020	3539.500	.844
Item 6. It is easy to make the low-level program do what I want to	3128.500	.192	3132.500	.120
Item 7. Low-level programming is easy	3034.500	.103	3522.000	.799
Item 8. I intend to use low-level programming during my studies	2791.500	.018	3016.500	.109
Item 9. In my job I will code low-level programs that will be helpful for me	2448.000	.000	3213.000	.201
Item 10. Low-level programming will form part of my profession	2465.500	.000	3193.000	.177

TABLE 10. Results of the effect size using the r for motivation.

	Arduino-based robot	Nintendo DS
	Effect size r	Effect size r
Item 1. Knowing low-level programming will help me find a job	-0.02	-0,33
Item 2. It will be easier to finish my studies if I know low-level programming	-0,20	-0,30
Item 3. During my studies it will be useful for me to know low-level programming	-0,21	-0,02
Item 4. Knowing low-level programming will be useful for my work	-0,14	-0,18
Item 5. It is easy for me to learn low-level programming	-0,30	0,00
Item 6. It is easy to make the low-level program do what I want to	-0,06	-0,25
Item 7. Low-level programming is easy	-0,18	-0,01
Item 8. I intend to use low-level programming during my studies	-0,17	-0.11
Item 9. In my job I will code low-level programs that will be helpful for me	-0,21	-0,04
Item 10. Low-level programming will form part of my profession	-0,28	0.00

TABLE 11. Learning outcomes (scores in the range of 0 to 10).

	Pre test			Post test		
	Median	Mean	SD	Median	Mean	SD
Arduino-based robot	7,0000	6,84375	1,8025	9,0000	8,0625	1,6144
Nintendo DS	7,0000	7,16	1,830	8,0000	7,8533	1,5997

differences in 6 of the 10 questions that composed the questionnaire, and after the interventions, non-significant differences were found in all the items. The questions with lower scores after the interventions were those related to micro controllers, memory, and instructions.

It is worth noting that students' previous experience in both cohorts is very similar. 28% of the students in the Arduino-based robot group have previous experience in programming microcontrollers (all of them reported having used Arduino or Arduino-compatible board) and 19% have previous experience with the C programming language, whereas for the Nintendo DS cohort, 22% have previous experience with microcontrollers like Arduino or similar and 17% have previously programmed in C.

C. STUDENT PERCEPTION OF ARDUINO PLATFORM

This instrument was applied to the Arduino group at the end of the intervention. A descriptive statistical analysis has therefore been carried out. The results are shown in Table 13. The item that obtained the lowest score regards the perception of the utility of knowing low-level programming during their studies (item 3). On the contrary, the one that obtained the highest score regards the perceived utility of knowing low-level programming for their future jobs (item 9).

D. STUDENT PERCEPTION OF NINTENDO-DS LABS

This instrument was applied to the Nintendo DS group at the end of the intervention. A descriptive statistical analysis has therefore been carried out. The results are shown in Table 14.

TABLE 12. Descriptive and inference analysis by questions of the knowledge questionnaire. Fisher test was carried out as Chi-square requirements were not reached.

	Prestest					Postest				
		NDS		Arduino			NDS		Arduino	
Question	Chi-square p-value	n	% (n=75)	n	% (n=96)	Chi-square p-value	n	% (n=75)	n	% (n=96)
1	0.233	57	76	80	83.33	0.375	62	82.67	84	87.5
2	.018*	71	94.67	79	82.29	0.393	71	94.67	87	90.63
3	0.032	61	81.33	64	66.67	0.081	65	86.67	73	76.04
4	0.429	35	46.67	39	40.63	0.739	45	60	60	62.5
5	0.021	43	57.33	38	39.58	0.12	56	74.67	61	63.54
6	0.914	40	53.33	52	54.17	0.062	61	81.33	66	68.75
7	0.802	51	68	67	69.79	0.507	57	76	77	80.21
8	0.264	65	86.67	77	80.21	0.59	69	92	86	89.58
9	0.03	71	94.67	80	83.33	0.912	70	93.33	90	93.75
10	0.367	57	76	67	69.79	0.504	52	69.33	71	73.96

TABLE 13. Statistical results for the student perception of the Arduino platform.

	Item 1	Item 2	Item 3	Item 4	Item 5	Item 6	Item 7	Item 8	Item 9	Item 10
Median	4	3	4	3	3	4	3	4	4	3
Mean	3,4896	3,2500	3,4362	3,2813	3,4000	3,4896	3,3723	3,5104	3,6000	3,3229
SD	1,0760	1,0761	0,9898	1,0926	1,1051	1,0563	0,9944	1,2896	1,1522	1,1002

TABLE 14. Results of the student perception of the Nintendo-DS platform.

	Item 1	Item 2	Item 3	Item 4	Item 5	Item 6	Item 7	Item 8	Item 9	Item 10
Median	4	4	4	3	3	3	3	4	4	3
Mean	3,4133	3,3200	3,3200	3,2133	2,8800	3,0133	3,0000	3,4933	3,5467	3,2667
SD	0,9739	0,9028	0,8724	0,8589	0,9147	0,9078	0,9153	0,8601	0,8268	0,8110

TABLE 15. Results of the categorized answers.

	Arduino		NDS	
	N (Total number of answers)	Percentage (%) with respect to the total number (N=96) of subjects in that cohort	N (Total number of answers)	Percentage (%) with respect to the total number (N=75) of subjects in that cohort
Issue 1. Issues regarding the lab platform	11	11,46	1	1,33
Issue 2. Issues regarding the platform programming	1	1,04	7	9,33
Issue 3. Problems understanding the course content	3	3,13	6	8
Issue 4. Regretting the replacement of the NDS platform	0	0	26	34,67
Issue 5. Issues leading to student frustration	20	20,83	6	8
Issue 6. Gap or lack of connection between the theory and lab	2	2,08	4	5,33
Issue 7. More time required for the final project	30	31,25	0	0

The item that obtained the lowest score regards the perceived simplicity of learning low-level programming (item 5). The one that obtained the highest score regards the perceived utility of knowing low-level programming for their future jobs (item 9).

E. QUALITATIVE EVALUATION OF THE EMPLOYED PLATFORM

This tool was applied to both groups. Out of the 171 answers, 54 were discarded because they were left blank or did not know the answer. 67 answers of the cohort 1 (Arduino group) and 50 of the cohort 2 (Nintendo-DS group) were analyzed. Table 15 shows the results of the categorized answers, organized by cohorts.

F. CORRELATION BETWEEN ACQUIRED KNOWLEDGE, MOTIVATION, AND PERCEPTION OF THE PLATFORMS

When motivation and knowledge are correlated, inconclusively results are found. Before the intervention, non-significant correlations have been found between these two variables considering all the sample (Spearman $Rho = 0.075$, $p = 0.330$), the Arduino group (Spearman $Rho = 0.121$, $p = 0.240$) and the NDS group (Spearman $Rho = 0.082$, $p = 0.484$) separately. Nevertheless, after the intervention, significant correlations have been found for all the sample (Spearman $Rho = 0.191$, $p = 0.012$) and Arduino group (Spearman $Rho = 0.247$, $p = 0.015$), and non-significant correlation for NDS group (Spearman $Rho = 0.115$, $p = 0.326$). Thus, it seems that motivation is not determinant to acquired knowledge, but further research is needed to deep in these results.

Besides, non-significant correlations have been found between the perception of the Arduino platform and the acquired knowledge (Spearman $Rho = 0.168, p = 0.102$), and in the perception of Nintendo DS platform and the acquired knowledge (Spearman $Rho = 0.052, p = 0.661$) of the respective groups after the intervention.

VI. DISCUSSION

This study compares two approaches to teach Computer Architecture: Arduino, a robot-based approach widely used from primary to higher education, and Nintendo DS, a game-based one. The results show that, despite the a priori advantages of Arduino as a more engageable tool [30] and pedagogical affordances related to easiness of low-level programming and machine and assembly language, the differences among both approaches at the end of the interventions are not conclusive in terms of motivation and learning outcomes. These results are discussed in this section regarding the different types of learning activities, pedagogical affordances and constraints, and attitudes developed by the students in both didactic sequences.

Students' motivation towards programming was measured as previous literature revealed it has an impact on motivation to learn and acquired knowledge [63], especially when robots are used for teaching [30]. It is stunning how motivation in the Arduino group decreases after the intervention, on the contrary to what happens with the Nintendo DS group, even when the Arduino group was expected to be more motivated according to previous literature [30], [63] given the characteristics of the tool that facilitates programming and provides more interaction with hardware (calibration of sensors, servo, robot pieces, etc.). The Arduino group was indeed initially more motivated than the Nintendo DS one, probably because the use of robots is usually more motivating to learn programming [63]. But at the end, differences in motivation between both groups were not significant. Particularly, the Nintendo DS cohort increased their motivation towards low-level programming, and they perceive that after the intervention it was easier than initially expected. This can be explained by the fact that the game-development process has a positive impact on motivation and harder work when solving problems [13], [70]. Further explanations might be found in different aspects. The work in [71] points out that the use of robots does not necessarily increases the relevance, confidence or satisfaction when programming. The results of our study support these ones, as other aspects play a role in the teaching and learning process that should not be overlooked. Thus, regarding the differences in the low-level programming teaching approaches, the Nintendo DS group resorts to the use of different graphic representation modes (frame-buffer, rotoscale, and tiles), while Arduino group employs simple programs to be analyzed with a remote debugger. The use of graphic modes has an inherent complexity, especially for those facing for the first time computer graphics. On the other hand, first sessions of the Arduino group spin around the use of simple programs which will softly guide them through the different lesson contents. This implies that the robot-approach entails less complexity in terms of low-level programming competences than that for console-approach, taking into account that the Nintendo DS activities need a basic understanding of how graphics and memory work. This different level of complexity in the first sessions is a pedagogical advantage of Arduino over Nintendo DS. However, the fact that first activities of the Arduino group are carried out on the Arduino board (without involving the robot until lesson 5), may lead students to perceive these programming activities as decoupled from the rest of the sessions (implementation of the robot functionality). This is a constraint that directly impact on the learning process for the Memory-lesson concepts. The Nintendo DS cohort performs better that the Arduino one on questions regarding the memory system design. All these issues have determined that the Nintendo DS group increases it motivation towards the usefulness of low-level programming, counterbalancing initial motivation differences among groups.

The different characteristics of Nintendo DS and Arduino that determine the learning activities can also explain the differences on perception of both platforms, after the lessons, regarding the *easiness to program*. Arduino cohort has a better perception of the platform than the Nintendo DS one. Arduino, through its Arduino IDE (programming environment), provides a simple and easy-to-use environment to program the board, what has turned it into a widely used platform from primary to higher education. This is not the case for the Nintendo DS having to resort to command-line tools which might overwhelmed students unfamiliar with the GNU/Linux like systems. This result is of interest as it points out to Arduino, over Nintendo DS, for those students unfamiliar with GNU/Linux systems. This requires further research as this question was not initially part of the research and there were no statistically significant differences between the two cohorts in terms of acquired knowledge either for the pre-test or the post-test.

The differences found between both approaches can also be explained by the way teaching activities have been organized. Arduino activities needed from robots, and were carried out during the lab sessions, in groups. Nintendo DS activities based on simulators and real consoles, enable students to work at home, in an individual manner, preventing some teamwork issues from occurring, even when both approaches were based on teamwork. The higher load of teamwork is a strong point of the Arduino learning activities, as it helps students to develop social skills and work in an environment that is closer to their future work environment and very valuable skills in Computer Science [72]. However, teamwork has inherent difficulties to deal with, as those derived from the conflicts that may emerge among peers, as the answers of the students to the open question show. This result is especially valuable because, even when previous literature has focused on the benefits of developing teamwork skills of game-based approaches [73], for robotics further research was claimed [74], [75], and this work contributes to give light to this issue. Nevertheless, aspects specifically derived from teamwork have not been specifically assessed, further research is needed to evaluate conflicts and their reasons, and the way they assume responsibilities, and how these issues affect the results of the interventions. Nonetheless, the work in [76] recommends working in groups of two people rather than in larger groups because in this way the benefits of pair programming are enhanced and the exchange of leadership roles is facilitated.

The design of both teaching approaches was also different in terms of the introduction of the console and the robot. The Nintendo DS group combine both the console and its emulator counterpart from the first session, what enables a seamless transit to the real platform for less experienced students [35]. On the other hand, in the Arduino group, the robot was not introduced until the Input/output lesson (last lesson of the syllabus). Initially, it was unclear whether this later use of the platform could have an impact on acquired knowledge. The Nintendo DS group started from lower scores than the Arduino one in the knowledge questionnaire (pre test), but in the post test there were no statistically significant differences among both cohorts, being scores around 8 out of 10 in both cases. These high scores support the evidences found by other authors [25], [30], [32], [37], [44], [48], [74], [77] about the convenience of both strategies, robotics and game-based approaches, for teaching Computer Architecture. That implies that both approaches are equally suitable in terms of learning outcomes, and valid to teach Computer Architecture with similar learning results. Besides, these learning outcomes seem not to be correlated to motivation, in contrast to previous research that suggest that motivation has an impact on acquired knowledge [63], especially when computer technologies are involved [64], [65]. In this case, both approaches seem to homogenize the final perception, independently of the employed platform. It is also outstanding how more than a third of the Nintendo DS cohort regrets the possibility of new students not having the opportunity of programming a game for the console, what aim at the acceptance that the platform has despite its obsolescence.

Despite the lack of statistically significant differences both in the acquired knowledge and at the level of the single questions comprising the questionnaire, it can be noticed that there are three questions (questions number 4, 5 and 10) whose results are under the 75% although over 60%. This open new and future lines of research to explore how these issues could be improved (new activities or improved ones) with the considered platforms.

VII. CONCLUSION

The results of this study show the suitability of the robot-based (Arduino) and the game-based (Nintendo DS) approaches to teach Computer Architecture in terms of learning outcomes. Nevertheless, the different affordances and constraints had also an impact on results due to their differences from a pedagogical perspective.

Arduino is revealed as a powerful approach to teach low-level programming to poorly experienced students due to the facilities that the tool presents. It is also useful to teach input/output concepts since it gives students the opportunities to program different behavior in the robot (line follower, obstacle avoidance, light follower, etc.) and deal with external conditions that affect the robot sensors or other peripheral devices. On the contrary, Nintendo DS is especially interesting because of the different graphic modes it support and the different peripheral devices (keys, touch screen, timers, etc.). The design and implementation of a game, including the representation of graphics and its logic involve dealing with different type of memories and input/output techniques. Students can experience the three types of input/output techniques: pooled to manage the console keys, interrupt to use timers, and DMA to display different backgrounds (copying backgrounds to the video memory).

However, besides these pedagogical approaches, teachers must consider other constraints and affordances in their teaching, as the results of this research reveal. The impact that external conditions have on the robot sensors means that it is necessary to calibrate them whenever conditions change. This process is time-consuming, especially in the beginning when students are not familiar with the process. It is therefore desirable to make robots available in addition to the lab hours to minimize frustration that might demotivating them. This is not necessary for game-based approaches, as a simulator is available. This enables students to work individually at home. In this sense, teamwork should not be overlooked, as even when it resemble a real working context, it may trigger conflicts causing delays that could eventually lead to increase the time needed to complete the proposed tasks. It seems also advisable to provide students with the opportunity to work with robots platform from the beginning of the course, avoiding this way the perception of lab sessions being decoupled from input/output lesson. This might be responsible for a low perception of the usefulness of low-level programming. Regarding game-based approaches, teachers should pay attention at initial knowledge requirements on low-level programming to prevent students from stucking from the beginning.

Despite the obsolescence of Nintendo DS, the results of this work are interesting for teachers aimed to design teaching plans based on other game platforms, and of course, to design their robot-based lessons, even using other platforms than Arduino. Further research in other contexts as in vocational training computer programs and in Secondary Education are proposed as future lines of research. Finally, as this is a case study and results cannot be extrapolated, further research is needed to deep in these results.

REFERENCES

[1] J. Impagliazzo, S. Conry, J. Hughes, J. Junlin, A. McGettrick, E. Durant, H. Lam, R. Reese, and L. Herger, *Curriculum Guidelines for Undergraduate Degree Programs in Computer Engineering*. New York, NY, USA: Association for Computing Machinery, 2016.

[2] L. Porter, S. Garcia, H.-W. Tseng, and D. Zingaro, "Evaluating student understanding of core concepts in computer architecture," in *Proc. 18th ACM Conf. Innov. Technol. Comput. Sci. Edu. ITiCSE*, New York, NY, USA, 2013, pp. 279–284.

[3] J. M. Lopez-Guede, I. Soto, A. M. F. D. Leceta, and J. M. Larrañaga, "Educational innovation in the computer architecture area," *Procedia Social Behav. Sci.*, vol. 186, pp. 388–394, May 2015.

[4] J. C. Cuevas-Martinez, A. J. Yuste-Delgado, J. M. Perez-Lorenzo, and A. Trivino-Cabrera, "Jump to the next level: A four-year gamification experiment in information technology engineering," *IEEE Access*, vol. 7, pp. 118125–118134, 2019.

[5] E. Larraza-Mendiluze and N. Garay-Vitoria, "Approaches and tools used to teach the computer input/output subsystem: A survey," *IEEE Trans. Educ.*, vol. 58, no. 1, pp. 1–6, Feb. 2015.

[6] B. Nikolic, Z. Radivojevic, J. Djordjevic, and V. Milutinovic, "A survey and evaluation of simulators suitable for teaching courses in computer architecture and organization," *IEEE Trans. Educ.*, vol. 52, no. 4, pp. 449–458, Nov. 2009.

[7] J. R. Brinson, "Learning outcome achievement in non-traditional (virtual and remote) versus traditional (hands-on) laboratories: A review of the empirical research," *Comput. Edu.*, vol. 87, pp. 218–237, Sep. 2015. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0360131515300087

[8] V. J. Shute, C. Sun, and J. Asbell-Clarke, "Demystifying computational thinking," *Educ. Res. Rev.*, vol. 22, pp. 142–158, Nov. 2017. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1747938X17300350

[9] H.-T. Hung, J. C. Yang, G.-J. Hwang, H.-C. Chu, and C.-C. Wang, "A scoping review of research on digital game-based language learning," *Comput. Edu.*, vol. 126, pp. 89–104, Nov. 2018, doi: 10.1016/j.compedu.2018.07.001.

[10] J. M. Wing, "Computational thinking," *Commun. ACM*, vol. 49, no. 3, pp. 33–35, 2006, doi: 10.1145/1118178.1118215.

[11] S. Grover and R. Pea, "Computational thinking in K–12: A review of the state of the field," *Educ. Researcher*, vol. 42, no. 1, pp. 38–43, Jan. 2013.

[12] K. Howland and J. Good, "Learning to communicate computationally with flip: A bi-modal programming language for game creation," *Comput. Edu.*, vol. 80, pp. 224–240, Jan. 2015.

[13] D. Topalli and N. E. Cagiltay, "Improving programming skills in engineering education through problem-based game projects with scratch," *Comput. Edu.*, vol. 120, pp. 64–74, May 2018. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0360131518300113

[14] J. Kramer, "Is abstraction the key to computing?" *Commun. ACM*, vol. 50, no. 4, pp. 36–42, Apr. 2007.

[15] J. M. Wing, "Computational thinking and thinking about computing," *Philos. Trans. Roy. Soc. London A, Math. Phys. Sci.*, vol. 366, no. 1881, pp. 3717–3725, 2008.

[16] J. J. Lu and G. H. Fletcher, "Thinking about computational thinking," in *Proc. ACM SIGCSE Bull.*, vol. 41, no. 1, 2009, pp. 260–264.

[17] L. Moreno, C. Gonzalez, I. Castilla, E. Gonzalez, and J. Sigut, "Applying a constructivist and collaborative methodological approach in engineering education," *Comput. Edu.*, vol. 49, no. 3, pp. 891–915, Nov. 2007.

[18] L. S. Vygotsky, *Mind in Society: The Development of Higher Psychological Processes*. Cambridge, MA, USA: Harvard Univ. Press, 1978.

[19] J. G. Greeno, A. M. Collins, and L. B. Resnick, "Cognition and learning," *Handbook Educ. Psychol.*, vol. 77, pp. 15–46, Jan. 1996.

[20] S. Papert and I. Harel, "Situating constructionism," *Constructionism*, vol. 36, no. 2, pp. 1–11, 1991.

[21] Ü. Çakiroğlu, M. Yildiz, E. Mazlum, E. T. Güntepe, and Ş. Aydin, "Exploring collaboration in learning by design via weblogs," *J. Comput. Higher Edu.*, vol. 29, no. 2, pp. 309–330, Aug. 2017.

[22] J. Leonard, M. Mitchell, J. Barnes-Johnson, A. Unertl, J. Outka-Hill, R. Robinson, and C. Hester-Croff, "Preparing teachers to engage rural students in computational thinking through robotics, game design, and culturally responsive teaching," *J. Teacher Edu.*, vol. 69, no. 4, pp. 386–407, Sep. 2018.

[23] M. Carbonaro, M. Cutumisu, H. Duff, S. Gillis, C. Onuczko, J. Siegel, J. Schaeffer, A. Schumacher, D. Szafron, and K. Waugh, "Interactive story authoring: A viable form of creative expression for the classroom," *Comput. Edu.*, vol. 51, no. 2, pp. 687–707, Sep. 2008.

[24] G.-J. Hwang, C.-M. Hung, and N.-S. Chen, "Improving learning achievements, motivations and problem-solving skills through a peer assessment-based game development approach," *Educ. Technol. Res. Develop.*, vol. 62, no. 2, pp. 129–145, Apr. 2014, doi: 10.1007/s11423-013-9320-7.

[25] E. Larraza-Mendiluze, N. Garay-Vitoria, J. I. Martin, J. Muguerza, T. Ruiz-Vazquez, I. Soraluze, J. F. Lukas, and K. Santiago, "Game-Console-Based projects for learning the computer input/output subsystem," *IEEE Trans. Educ.*, vol. 56, no. 4, pp. 453–458, Nov. 2013.

[26] M. J. Santofimia and F. Moya, "Nintendo DS: A pedagogical approach to teach computer architecture," in *Proc. Int. Conf. Embedded Syst. Appl., (ESA)*, H. R. Arabnia and A. M. G. Solo, Eds. Las Vegas NV, USA: CSREA Press, Jul. 2009, pp. 269–273.

[27] D. Kumar, "Curriculum descant: Pedagogical dimensions of game playing," *Intelligence*, vol. 10, no. 1, pp. 9–10, Mar. 1999.

[28] S. Bergin and R. Reilly, "The influence of motivation and comfort-level on learning to program," in *Proc. 17th Workshop Psychol. Program. Interest Group*. Brighton, U.K.: University of Sussex, 2005, pp. 293–304.

[29] B. D. Coller and M. J. Scott, "Effectiveness of using a video game to teach a course in mechanical engineering," *Comput. Edu.*, vol. 53, no. 3, pp. 900–912, Nov. 2009.

[30] N. Spolaôr and F. B. V. Benitti, "Robotics applications grounded in learning theories on tertiary education: A systematic review," *Comput. Edu.*, vol. 112, pp. 97–107, Sep. 2017.

[31] D. Bazylev, A. Margun, K. Zimenko, A. Kremlev, and E. Rukujzha, "Participation in robotics competition as motivation for Learning1," *Procedia Social Behav. Sci.*, vol. 152, pp. 835–840, Oct. 2014. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S187704281405397X

[32] E. Brunvand,"Games as motivation in computer design courses: I/O is the key," in *Proc. 42nd ACM Tech. Symp. Comput. Sci. Edu. SIGCSE*, 2011, pp. 33–38.

[33] P. Tiernan, "Enhancing the learning experience of undergraduate technology students with LabVIEW software," *Comput. Edu.*, vol. 55, no. 4, pp. 1579–1588, 2010.

[34] C. A. Canizares and Z. T. Faur, "Advantages and disadvantages of using various computer tools in electrical engineering courses," *IEEE Trans. Educ.*, vol. 40, no. 3, pp. 166–171, Aug. 1997.

[35] V. Potkonjak, M. Gardner, V. Callaghan, P. Mattila, C. Guetl, V. M. Petrović, and K. Jovanović, "Virtual laboratories for education in science, technology, and engineering: A review," *Comput. Edu.*, vol. 95, pp. 309–327, Apr. 2016. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0360131516300227

[36] A. Clements, "ARMs for the poor: Selecting a processor for teaching computer architecture," in *Proc. IEEE Frontiers Edu. Conf. (FIE)*, Oct. 2010, pp. T3E–1–T3E–6.

[37] M. Carbonaro, D. Szafron, M. Cutumisu, and J. Schaeffer, "Computer-game construction: A gender-neutral attractor to computing science," *Comput. Edu.*, vol. 55, no. 3, pp. 1098–1111, Nov. 2010. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0360131510001399

[38] A. Tlili, F. Essalmi, and M. Jemni, "A mobile educational game for teaching computer architecture," in *Proc. IEEE 15th Int. Conf. Adv. Learn. Technol.*, Jul. 2015, pp. 161–163.

[39] S. Tabik and L. F. Romero, "16 cards to get into computer organization," *Enseñanza y Aprendizaje de Ingeniería de Computadores*, vol. 2014, no. 4, pp. 5–14, 2014.

[40] S. Papavlasopoulou, M. N. Giannakos, and L. Jaccheri, "Empirical studies on the maker movement, a promising approach to learning: A literature review," *Entertainment Comput.*, vol. 18, pp. 57–78, Jan. 2017, doi: 10.1016/j.entcom.2016.09.002.

[41] G. Chen, J. Shen, L. Barth-Cohen, S. Jiang, X. Huang, and M. Eltoukhy, "Assessing elementary students' computational thinking in everyday reasoning and robotics programming," *Comput. Edu.*, vol. 109, pp. 162–175, Jun. 2017, doi: 10.1016/j.compedu.2017.03.001.

[42] M. U. Bers, L. Flannery, E. R. Kazakoff, and A. Sullivan, ''Computational thinking and tinkering: Exploration of an early childhood robotics curriculum,'' *Comput. Edu.*, vol. 72, pp. 145–157, Mar. 2014, doi: 10.1016/j.compedu.2013.10.020.

[43] M. Ruzzenente, M. Koo, K. Nielsen, L. Grespan, and P. Fiorini, ''A review of robotics kits for tertiary education,'' in *Proc. Int. Workshop Teach. Robot. Teach. Robot., Integrating Robot. School Curriculum*, Riva del Garda, Italy, 2012, pp. 153–162.

[44] Y.-W. Cheng, P.-C. Sun, and N.-S. Chen, ''The essential applications of educational robot: Requirement analysis from the perspectives of experts, researchers and instructors,'' *Comput. Edu.*, vol. 126, pp. 399–416, Nov. 2018. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0360131518302033

[45] C.-W. Chang, J.-H. Lee, P.-Y. Chao, C.-Y. Wang, and G.-D. Chen, ''Exploring the possibility of using humanoid robots as instructional tools for teaching a second language in primary school,'' *J. Educ. Technol. Soc.*, vol. 13, no. 2, pp. 13–24, 2010.

[46] D. Alimisis, ''Educational robotics: Open questions and new challenges,'' *Themes Sci., Technol. Educ.*, vol. 6, no. 1, pp. 63–71, 2013.

[47] T. S. Sklirou, ''Programming in secondary education: Applications, new trends and challenges,'' in *Proc. IEEE Global Eng. Edu. Conf. (EDUCON)*, Apr. 2017, pp. 580–585.

[48] M. A. Rubio, R. Romero-Zaliz, C. Mañoso, and A. P. de Madrid, ''Closing the gender gap in an introductory programming course,'' *Comput. Edu.*, vol. 82, pp. 409–420, Mar. 2015.

[49] A. Cruz-Martín, J. A. Fernández-Madrigal, C. Galindo, J. González-Jiménez, C. Stockmans-Daou, and J. L. Blanco-Claraco, ''A LEGO mindstorms NXT approach for teaching at data acquisition, control systems engineering and real-time systems undergraduate courses,'' *Comput. Edu.*, vol. 59, no. 3, pp. 974–988, Nov. 2012. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0360131512000838

[50] I. Calvo, I. Cabanes, J. Quesada, and O. Barambones, ''A multidisciplinary PBL approach for teaching industrial informatics and robotics in engineering,'' *IEEE Trans. Educ.*, vol. 61, no. 1, pp. 21–28, Feb. 2018.

[51] N. S. Yamanoor and S. Yamanoor, ''High quality, low cost education with the raspberry pi,'' in *Proc. IEEE Global Humanitarian Technol. Conf. (GHTC)*, Oct. 2017, pp. 1–5.

[52] P. Molins-Ruano, C. Gonzalez-Sacristan, and C. Garcia-Saura, ''Phogo: A low cost, free and 'maker' revisit to logo,'' *Comput. Hum. Behav.*, vol. 80, pp. 428–440, Mar. 2018. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0747563217305551

[53] I. Paliokas, C. Arapidis, and M. Mpimpitsos, ''PlayLOGO 3D: A 3D interactive video game for early programming education: Let LOGO be a game,'' in *Proc. 3rd Int. Conf. Games Virtual Worlds Serious Appl.*, May 2011, pp. 24–31.

[54] O. Erol and A. A. Kurt, ''The effects of teaching programming with scratch on pre-service information technology teachers' motivation and achievement,'' *Comput. Hum. Behav.*, vol. 77, pp. 11–18, Dec. 2017.

[55] M. N. Giannakos, M. Divitini, and O. S. Iversen, ''Entertainment, engagement, and education: Foundations and developments in digital and physical spaces to support learning through making,'' *Entertainment Comput.*, vol. 21, pp. 77–81, Jun. 2017. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1875952117300307

[56] D. Weintrop, , E. Beheshti, M. Horn, K. Orton, K. Jona, L. Trouille, and U. Wilensky, ''Defining computational thinking for mathematics and science classrooms,'' *J. Sci. Edu. Technol.*, vol. 25, no. 1, pp. 127–147, 2016.

[57] R. Mitnik, M. Nussbaum, and M. Recabarren, ''Developing cognition with collaborative robotic activities,'' *J. Educ. Technol. Soc.*, vol. 12, no. 4, pp. 317–330, 2009.

[58] A. Felicia and S. Sharif, ''A review on educational robotics as assistive tools for learning mathematics and science,'' *Int. J. Comput. Sci. Trends Technol*, vol. 2, no. 2, pp. 62–84, 2014.

[59] M. Barak and M. Assal, ''Robotics and STEM learning: Students' achievements in assignments according to the p_3 task taxonomy—Practice, problem solving, and projects,'' *Int. J. Technol. Design Edu.*, vol. 28, no. 1, pp. 121–144, 2018.

[60] D. H. Jonassen, *Learning to Solve Problems: An Instructional Design Guide*, vol. 6. Hoboken, NJ, USA: Wiley, 2004.

[61] M. Akcaoglu and M. J. Koehler, ''Cognitive outcomes from the game-design and learning (GDL) after-school program,'' *Comput. Edu.*, vol. 75, pp. 72–81, Jun. 2014.

[62] J. Sixt. (Jan. 1, 2020). *KDGB: A Graphical Debugger Interface*. KDBG.org. Accessed: May 19, 2020. [Online]. Available: http://KDBG.org

[63] A. Tella, ''The impact of motivation on Student's academic achievement and learning outcomes in mathematics among secondary school students in nigeria,'' *EURASIA J. Math., Sci. Technol. Edu.*, vol. 3, no. 2, pp. 149–156, Jun. 2007.

[64] T.-Y. Liu and Y.-L. Chu, ''Using ubiquitous games in an english listening and speaking course: Impact on learning outcomes and motivation,'' *Comput. Edu.*, vol. 55, no. 2, pp. 630–643, Sep. 2010.

[65] E. A. Gomez, D. Wu, and K. Passerini, ''Computer-supported team-based learning: The impact of motivation, enjoyment and team contributions on learning outcomes,'' *Comput. Edu.*, vol. 55, no. 1, pp. 378–390, Aug. 2010.

[66] F. D. Davis, ''User acceptance of information technology: System characteristics, user perceptions and behavioral impacts,'' *Int. J. Man-Mach. Stud.*, vol. 38, no. 3, pp. 475–487, Mar. 1993.

[67] J. R. A. Santos, ''Cronbach's Alpha: A tool for assessing the reliability of scales,'' *Extension Inf. Technol.*, vol. 37, no. 2, pp. 1–5, Apr. 1999. [Online]. Available: http://www.joe.org/joe/1999april/tt3.php

[68] T. Luo and Q. Xie, ''Using Twitter as a pedagogical tool in two classrooms: A comparative case study between an education and a communication class,'' *J. Comput. Higher Edu.*, vol. 31, no. 1, pp. 81–104, Apr. 2019.

[69] A. Field, *Discovering Statistics Using IBM SPSS Statistics*, 4th ed. Thousand Oaks, CA, USA: Sage Publications, 2013.

[70] D. Ozoran, N. Cagiltay, and D. Topalli, ''Using scratch in introduction to programming course for engineering students,'' in *Proc. 2nd Int. Eng. Edu. Conf. (IEEC)*, 2012, pp. 125–132.

[71] M. M. McGill, ''Learning to program with personal robots: Influences on student motivation,'' *ACM Trans. Comput. Edu.*, vol. 12, no. 1, pp. 1–32, Mar. 2012, doi: 10.1145/2133797.2133801.

[72] R. Vivian, K. Falkner, N. Falkner, and H. Tarmazdi, ''A method to analyze computer science Students' teamwork in online collaborative learning environments,'' *ACM Trans. Comput. Edu.*, vol. 16, no. 2, pp. 1–28, Mar. 2016.

[73] T. Hainey, W. Westera, T. M. Connolly, L. Boyle, G. Baxter, R. B. Beeby, and M. Soflano, ''Students' attitudes toward playing games and using games in education: Comparing scotland and The Netherlands,'' *Comput. Edu.*, vol. 69, pp. 474–484, Nov. 2013. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0360131513001905

[74] F. B. V. Benitti, ''Exploring the educational potential of robotics in schools: A systematic review,'' *Comput. Edu.*, vol. 58, no. 3, pp. 978–988, Apr. 2012, doi: 10.1016/j.compedu.2011.10.006.

[75] G. Nugent, B. Barker, N. Grandgenett, and V. Adamchuk, ''The use of digital manipulatives in k-12: Robotics, gps/gis and programming,'' in *Proc. 39th IEEE Frontiers Edu. Conf.*, Oct. 2009, pp. 1–6.

[76] S. Lopez-Pernas, A. Gordillo, E. Barra, and J. Quemada, ''Examining the use of an educational escape room for teaching programming in a higher education setting,'' *IEEE Access*, vol. 7, pp. 31723–31737, 2019.

[77] J. Gonzalez, H. Pomares, and I. Rojas, ''Use of the sony playstation 2 to motivate University students into learning embedded systems,'' in *Proc. IASK Int. Conf. Teach. Learn. (TL)*, At Aveiro, Portugal, 2008, pp. 923–927.

BEATRIZ GARCÍA FERNÁNDEZ received the Ph.D. degree in civil engineering from the University of Castilla-La Mancha, Spain, in 2010. She holds a teaching position at the Faculty of Education of Ciudad Real in the field of Science Education, since 2011. She has been a Visiting Professor with the Pontifical University of Chile, Odisee University, Belgium, the Polytechnic Institute of Lisbon, and the University of Algarve, Portugal. She has participated in various research projects in the fields of civil engineering and education, publishing more than fifty journal articles and book chapters

XAVIER DEL TORO received the degrees of Technical Engineer in Industrial Electronics and Engineer in Automatic Control and Industrial Electronics from the Universitat Politècnica de Catalunya, Spain, in 1999 and 2002, respectively, and the Ph.D. degree from the University of Glamorgan, U.K., in 2008. From September 2005 to October 2006, he was a Marie Curie Research Fellow with Politecnico di Bari, Italy. Since 2008, he has been working as a Postdoctoral Researcher with the University of Castilla-La Mancha, Spain, holding a Juan de la Cierva Fellowship, from 2012 to 2015. He is currently as a Faculty Member with the School of Computer Science. His research interests include power electronics, energy harvesting, the Internet of Things, and computer science education.

MARIA J. SANTOFIMIA received the B.S. degree in computer engineering from the University of Cordoba, in 2004, the M.Sc. degree in computer system security from the University of South Wales, U.K., in 2003, and the Ph.D. degree in computer engineering from the University of Castilla-La Mancha, in 2011. During half semester, in 2009, she was a Visiting Researcher with the Language Technology Institute, Carnegie Mellon University, USA. Her main research interest includes common-sense reasoning systems. Since 2007, she holds a teaching position at the University of Castilla-La Mancha.

JAVIER DORADO received the degree in computer science and M.Sc. degree in teacher training from the University of Castilla-La Mancha, in 2015 and 2017, respectively, where he is currently pursuing the Ph.D. degree. He is also a Teaching Assistant with the University of Castilla-La Mancha. His research interests include cyber physical systems, learning analytics systems, and health systems.

FELIX J. VILLANUEVA received the Diploma degree in computer engineering and Ph.D. degree from the University of Castilla-La Mancha (UCLM), in 2001 and 2009, respectively. He is currently working as an Assistant Professor with UCLM. His research interests include wireless sensor networks, ambient intelligence, and embedded systems.

DAVID VILLA received the degree in computer engineering from the University of Castilla-La Mancha (UCLM), in 2002, and the Ph.D. degree in computer engineering, in 2009. Since then, he works as a Lecturer of computer architecture and technologies area with UCLM, on topics around network computers, distributed systems, and the IoT. His current research interests include heterogeneous distributed systems, distributed embedded systems design, virtual network routing protocols, and intelligent systems.

JUAN C. LOPEZ (Member, IEEE) received the M.S. and Ph.D. degrees in telecommunication engineering from the Technical University of Madrid, in 1985 and 1989, respectively. From September 1990 to August 1992, he was a Visiting Scientist with the Department of Electrical and Computer Engineering, Carnegie Mellon University, Pittsburgh, PA, USA. From 1989 to 1999, he was an Associate Professor with the Department of Electrical Engineering, Technical University of Madrid. He is currently a Professor of computer architecture with UCLM, where he served as the Dean of the School of Computer Science, from 2000 to 2008. His research activities center on embedded system design, distributed computing, and advanced communication services.