An Arduino Simulator in Classroom – a Case Study

Paulo F. Gonçalves
Coimbra Polytechnic - ISEC, Portugal
a21171940@isec.pt

João Sá
Escola Secundária de Avelar Brotero, Coimbra, Portugal
joaosa@gmail.com

Anabela Coelho
Agrupamento de Escolas de Pombal, Portugal
anabela.coelho@aepombal.edu.pt

João Durães
Coimbra Polytechnic - ISEC, Portugal
jduraes@isec.pt

Abstract

The Arduino Platform is increasingly being used as a central component in introductory programming courses of the curricula in middle, high school and even higher education. Given this scenario it is pertinent to understand how the cost-effectiveness, reliability and accessibility of this central component can be improved. We propose the use of an Arduino simulator to improve usability, cost, and class efficiency, allowing for improved and even new forms of use and course benefits. This paper presents and describes an Arduino simulator that we developed for education purposes, and a case study of its use in embedded programming courses from two high-schools. We compared its use against the usual use of real hardware platform analyzing usability, student workload and time efficiency. Our results, that we present and discuss, suggest that there are no apparent drawbacks in using the simulator, and some metrics such as basic exercise-solving efficiency and global effort showed an improvement.

2012 ACM Subject Classification Applied computing → Computer-assisted instruction

Keywords and phrases Arduino, Education, Simulator

Digital Object Identifier 10.4230/OASIcs.ICPEC.2020.12

1 Introduction

Teaching programming with microcontrollers is becoming increasingly common in middle and high-schools, even outside electronics courses. Several factors promote this scenario: digital literacy is now viewed as an important aspect of enabled citizenship that should be promoted as early as possible [11, 14], the increasing pervasiveness of the *Internet of Things* (IoT), and the low cost of microcontrollers when compared to traditional desktop computers making them an interesting tool for programming courses.

One of the platforms most commonly used and best adapted to the learning with microcontrollers scenario is the Arduino Platform [2] since it combines the simplicity, yet resourcefulness, of hardware with the easiness of an integrated development environment well suited for people without significant knowledge on microcontrollers [6], and has the additional advantage of being an *open source* platform which means no licensing costs. It is then no surprise that many middle and high-schools are now including subjects of programming using Arduino [1, 12, 10].

First International Computer Programming Education Conference (ICPEC 2020).
Editors: Ricardo Queirós, Filipe Portela, Mário Pinto, and Alberto Simões; Article No. 12; pp. 12:1–12:12

OpenAccess Series in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

Despite its many positive features, the use of the Arduino platform in classroom has several aspects that may decrease the efficiency of the education effort (e.g., cost and classroom time efficiency), and may even prevent the use of some forms of learning (e.g., distance learning), as detailed in the next section. We perceive these aspects as an opportunity for improvement by introducing the use of a *virtual Arduino*, that is, an Arduino Simulator that maintains all the advantages of the real hardware, decreases or removes the aspects that decrease class performance, and open new avenues and forms of learning not present with the real Arduino.

The use of an Arduino simulator in classroom is not common. In fact, to the best of our knowledge, no school in our country uses one, and fully or at least usable Arduino simulators in teaching context are not available. We propose to contribute to this scenario by developing and making freely available a simulator that can be used in classroom allowing the development and testing of programs and small circuits in a way that is similar to the real hardware platform, and is compatible with the use of the same IDE in the same way as with the real hardware, maintaining the same procedures the developer is used to.

This paper presents the several aspects relevant to our simulator and its use: in the next section we detail the aspects that lead to the opportunity and advantages of using a simulated Arduino in classroom, leading to high-level requirements and goals for the simulator. The overall architecture and technical overview of the simulator implementation is presented in the following section. Next we describe a case study of its use in the context of two schools and analyze the results concerning its usability and advantages. We conclude the paper with a summary of our results concerning the use of the simulator in classroom.

2 Motivation and goals

Arduinos are a key component of computing resource in the classroom, providing the computing power needed to run small experiments, allowing pure programming exercises, and enabling a first contact with electronic devices. Given its advantages, the tendency to use this device in classroom will probably continue and possibly even increase. However, the use of Arduinos in classroom presents some aspects that can be improved and opportunities that can be explored to further advance the learning process. We analyze these next.

2.1 Limitations of actual hardware use

We perceive the following limitations when using the Arduino in classroom; device wear-down, cost, skill mix-up, manual dexterity, assembly issues, time-efficiency.

Device wear-down and cost. The average number of writes of an Arduino's flash memory before failure is about 10,000 times [4]. This number is easily reached in just two years of use in classroom. If we consider the common scenario of one school having 6 classes interacting with 1 Arduino-equipped room, with 5 exercises per lesson, 10 tries per exercise (low estimation), and 15 lessons per year, we will have about 4500 writes per year. This will cause the school having to replace its Arduinos every two years.

Skill mix-up. Typical Arduino setups involve some electronic (LEDs, resistors, etc.), introducing the need for basic electronics skills, which are not typical for young students and may act against the goals of learning programming [7].

Manual dexterity. The small components used with Arduino projects require some level of fine movement control. Students with slight muscle-control disorder will find it very difficult to assemble the circuits. Schools should be inclusive and even if the number of affected students is small, this is an issue to consider.

Assembly issues. Sometimes the components' connectors are not in perfect condition and will not work well causing the project to fail by reasons outside the immediate control of the student, triggering frustration and wasting time.

Lesson time-efficiency. Arduino projects require some time to collect all the components, sorting and setting them up, and at the end of the lesson, gathering and storing. When compared to the typical lesson duration, this may add up to a considerable amount of time not being used as actual learning time.

2.2 Mitigation offered by simulation

We propose to mitigate these issues with a simulator in the following manner:

Device wear-down and cost. The components used with the simulator are virtual. There is no wear-down and no need for periodical replacement. This will immediately bring down the cost of maintaining a room equipped with Arduinos, as the computers used for the simulator typically are already available.

Skill mix-up. The electronic skill requirement can be reduced or even entirely removed if the simulator focus more on the logical aspects of the components and alleviate unnecessary details (e.g., simulated LEDs do not require an extra resistor), enabling students to focus on the central aspects of programming.

Manual dexterity and Assembly issues. Using a simulator with graphical interface and moving a pointing device is easier than handling small components. This removes some impediments to student having fine-grain movement disorders. It also solves the issues related to defective components and assembly problems.

Lesson time-efficiency. Setting up a simulated project can be as simple and fast as turning a computer on, executing a program and opening the project file. Sorting and placing components is replaced by point, click and drag components in the screen. This is faster than working with physical components, making more time available for learning.

2.3 New opportunities made available by simulation

Using a simulated Arduino opens new opportunities that can greatly improve the learning outcomes. We list those more immediately relevant:

Distance learning. Students typically don't take the hardware home. Students that cannot attend one lesson will loose that lesson; students wishing to improve skills on their own time will not be able to do so and will have to wait for the next lesson. A simulator can solve this limitation by allowing the student to use the simulator software at home, either running the simulator on his personal desktop, or by connecting to a simulator hosted on a server at school.

Project continuation support. Larger projects that cannot be concluded in one lesson are dismantled to reuse its components, preventing its continuation on the next lesson. A simulated project is just information that can be stored in a file and later reopened for continuation. This opens an entirely new possibility for larger projects.

Debugging. Debugging is not directly available on real Arduino hardware due to hardware constraints. However, debugging is important and should be encouraged. Simulators do not share the constraints of the real hardware and can offer the means to debug the code, including advanced functionality such as step-by-step execution and memory inspection.

We could identify more new uses made available by simulation. However, we see these as the most immediately relevant. We focused on these first and our simulator already supports them.

2.4 High level requirements and goals

The following are a set of high-level requirements that we identified to address the limitations and provide for the opportunities listed above. These requirements guided the simulator architecture definition and implementation choices.

Compatibility. To minimize or even remove intrusion and foreign aspects considering the typical real-Arduino development setup, the simulator must present itself as just another type of board and all it is required is to use the same IDE and simply configure this new board type. All the development and code upload to the (simulated) Arduino is then carried out in the same fashion as with real Arduinos.

Client-server architecture. To provide for easy central management, reduce operational and maintenance complexity, and allow access to users with minimum local setup, the simulator is hosted in a server where the actual simulation takes place. The user interacts with the simulation in two ways: via the usual IDE to develop the code, and via the client to interact with the circuit and components.

Web-compatibility. We decided that interaction with the simulator should be made via a web-browser to maximize usability. By using a web-based interface and protocols, we also gain the accessibility provided by the web.

3 Context and related work

3.1 Simulator implementations

There are several simulators available. We analysed their characteristics considering our goals (see summary in Table 1):

Binary-level code compatibility. This is important and needed to maintain the program loading process the same as with the read hardware. Our survey shows that more than half of the existing simulators are compatible only at the API level, meaning that they only simulate a fixed known basic functions of the Arduino, simulating their operation but not the code execution itself. This may prevent many existing libraries for Arduino from running in the simulator.

IDE compatibility. It is also important to minimize changes in the environment the developer is used to. None of the simulators we analysed is compatible with the Arduino IDE. This is contrary to the notion that in a teaching scenario, students should learn using the same tools that they will later use.

Web accessibility. We previously identified web-based interface as one of the requirements desirable for the simulator. However, none of the simulators that are binary-level compatible provide a web interface.

Debug ability. Although several of the simulators that are binary compatible allow debugging, all except one require an external tool, and the exception to this is not freely available. This either increases complexity and removes compatibility with the IDE, or causes extra costs.

3.2 Simulation/virtualization techniques

There are three main virtualization techniques: interpretation, compiled simulation and dynamic translation. The latter transforms instructions from the target architecture (Arduino microcontroller) to the host architecture (of the machine running the simulator), in practice, rewriting the code. This type of transformation may insert undesired effects in timing and

Table 1 Arduino simulators comparison.

Software	Free	Open-source	Cross-platform	Web interface	binary compatible	Maintained	Allows Debug	Arduino IDE
Proteus (https://www.labcenter.com/)	No	No	Yes	No	Binary	Yes	Yes	No
Virtronics Simulator for Arduino (https://virtronics.com.au/Simulator-for-Arduino.html)	No	No	No	No	API	No	Yes	No
VBB4Arduino (http://www.virtualbreadboard.com/)	No	No	No	No	API	Yes	Yes	No
123D Circuits (https://123d.circuits.io/)	Yes	No	Yes	Yes	API	No	–	No
ArduinoDebugger (https://github.com/Paulware/ArduinoDebugger)	Yes	Yes	No	No	API	No	Yes	No
CodeBlocks Arduino IDE (http://arduinodev.com/codeblocks/)	Yes	Yes	No	No	API	Yes	–	No
Simuino (http://web.simuino.com/home-1)	Yes	Yes	Yes	Yes	API	No	Yes	No
Emulino (https://github.com/ghewgill/emulino)	Yes	Yes	Yes	No	Binary	No	No	No
Atmel Studio 7 (https://www.microchip.com/mplab/avr-support/atmel-studio-7)	Yes	No	Yes	No	Binary	Yes	Yes[1]	No
Emulare (http://emulare.sourceforge.net/)	Yes	Yes	Yes	No	Binary	No	Yes[2]	No
SimAVR (https://github.com/buserror/simavr)	Yes	Yes	Yes	No	Binary	Yes	Yes[2]	No

controllabillity that may affect the intended behavior of the original code and prevents the debugging ability. Interpretation technique simulates each instruction, one by one, while compiled simulation builds an entire program in the host architecture with the instructions needed to simulate the complete sequence of the simulated instructions. Interpretation offers more controllability to support debugging, but in theory is slower than compiled simulation. We conducted a study to compare the performance of these two techniques [8] and concluded that, in our case, using Java, interpretation is faster. Thus, we opted for the interpretation technique to implement the simulator.

3.3 Arduino platform

Arduino is a platform composed by both hardware and software that can be used to control many types of electronic components and projects. The hardware is a printed circuit board with an AVR microcontroller [3], a power supply, a serial interface for programming, input and output connections, and a bootloader to program the device. The software component includes an API and libraries to manipulate the hardware and components, and a self-contained integrated development environment (IDE) to develop and upload code to the device.

4 Architecture and Implementation

The simulator is organized as a typical web-based client/server system. The server hosts the simulation logic and mechanisms and can serve multiple independent simulations at the same time, depending on the computing power. The client handles all the user and IDE interaction. The user interface is based on common web technology and compatible with common web-browsers. Figure 1 depicts the modules composing the simulator, which are described next.

[1] with extra hardware
[2] with external debugger

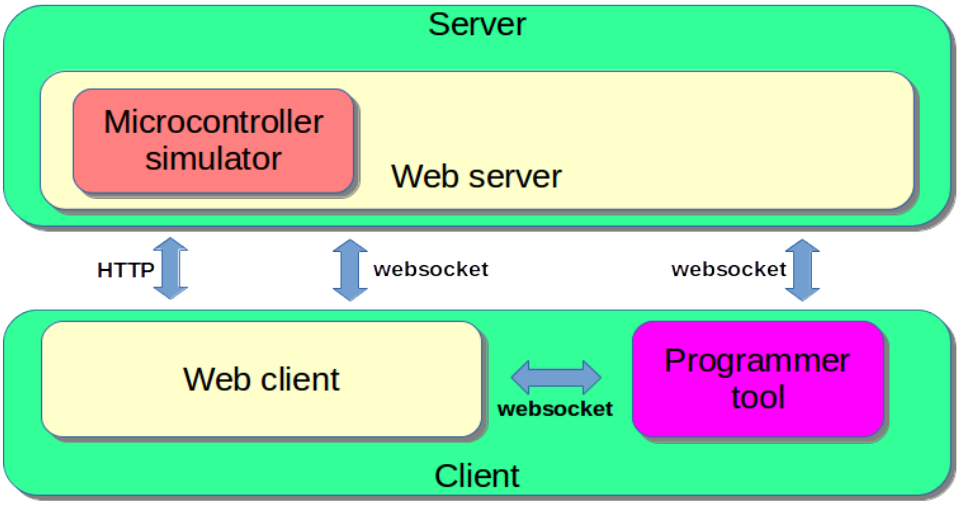

Figure 1 Architecture.

Microcontroller simulator. It is in this module that all the features of the microcontroller are implemented, namely the AVR Instruction Set, the microcontroller peripherals, FLASH and SRAM. This module executes the microcontroller code, exposes methods to change pins values and throws events when their state is changed. Its internal structure is very modular and each part can be easily replaced by other to allow simulating other Atmel microcontrolers (the one now simulated is the ATmega328P [4]) maintaining the integration with the parts responsible for simulating the ISA, FLASH, SRAM and the peripherals.

Web server. This module manages the users, maintains simulation instances, and links simulations to their respective user client and *programmer tool*. It is also responsible for storing user-created projects and all their related data.

Web client. This is where the user creates circuits to simulate. There is a drawing area available where the user places and connects the Arduino and various electronic components. All the simulator functionality can be accessed with the client: start/pause the simulation, execute step-by-step, inspect FLASH and SRAM memory contents, manage breakpoints, etc. Establishing a relationship between a web-client, the IDE/*programmer tool* and the web server cannot be done directly given the way browser *sandboxing* works. The client periodically sends information to the *programmer tool* to establish this relationship.

Programmer tool. This module corresponds to a board driver that is installed in the Arduino IDE replacing the device programming program (avrdude [5]) so that when uploading the binary code, instead of programming a real device the binary is sent to the simulator in the web server.

The web client interface can be seen on Figure 2, with the toolbar, components palette, drawing area with an example circuit and the inspecting windows for Source Code, FLASH and SRAM opened.

The usage of the Arduino IDE is the same as programming a real Arduino with the exception of selecting the new device installed by the board driver. In Figure 3 is possible to see the IDE with the same program loaded in the simulator.

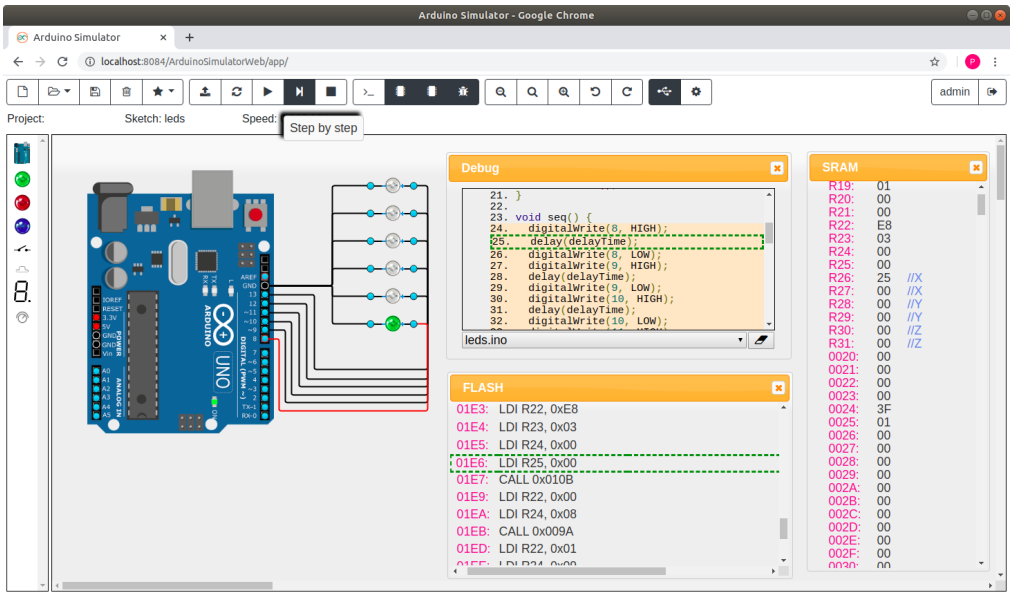

Figure 2 Simulator user interface.

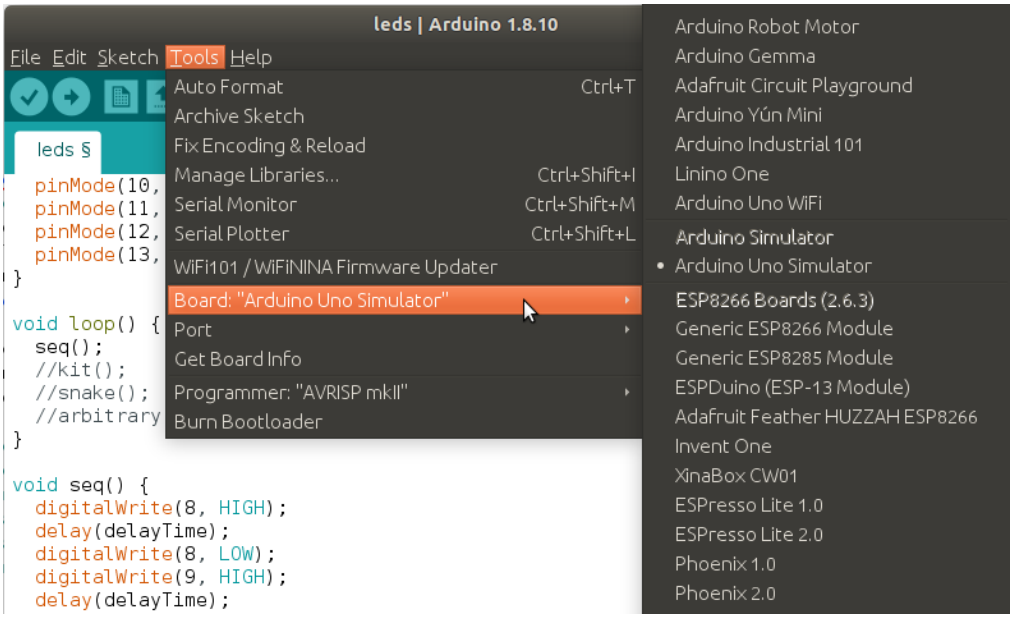

Figure 3 Arduino IDE.

5 Experimental use in real scenario

We conducted a real-use case study in two high-schools of the region to assess the usability and advantages of using the simulator in the classroom. The case study comprises 5 classes, 3 from *Escola Secundária de Avelar Brotero* (Coimbra), and 2 from *Agrupamento de Escolas de Pombal* (Pombal). In both schools students were aged between 16 and 18 years old. Some of the classes belonged to courses of technological area, while other belonged to health. The

Table 2 Characterization of students.

School	Class	Area	Curricular Year	# students	# Sim.	# Real
Avelar Brotero	Class 1	Sciences	12^{th}	29	12	17
Avelar Brotero	Class 2	Health	12^{th}	29	12	17
Avelar Brotero	Class 3	Mixed	12^{th}	20	12	8
Pombal	Class 4	Electronics	11^{th}	11	6	5
Pombal	Class 5	Electronics	12^{th}	12	7	5
			Total:	101	49	52

duration of the lessons was not the same for all tests and to compensate, the number of exercises also varied. The participation in the experiment was optional and we did not notice any reservation from any student. In each class, half the students used real Arduinos and the other half used the simulator. Table 2 summarizes the characteristics of the classes and students. Our methodology is described next.

5.1 Methodology

We separated each class into two groups of students: one used a real Arduino Uno and the other used the simulator. We balanced the groups in terms of experience and knowledge both in programming and in the Arduino Platform. This separation counted on the help of the respective teachers. The exercises presented to the students consisted on programming challenges involving simple circuits and were the same for both groups. These exercises were the usual for those classes, were prepared by the teachers and had no influence or change related to the simulator.

Due to the size of the classes and the lack of computers for all students at the school *ESAB* the exercises were performed in groups of 2 students. This was already the common scenario for those classes and it had nothing to do with the simulator.

We measured the efficiency of the simulator as a learning tool by observing the time students took to solve the exercises, comparing real Arduino with the simulator, the number of exercises completed, and their final result (correct/incorrect). We also used a questionnaire to evaluate the perception of the students about the use of the simulator.

5.2 Exercises

We used three exercises in each test. These exercises were defined by the teachers following their usual plan for the classes and there was no influence in the exercise definition related the Simulator. The exercises had incremental difficulty. All the exercises involved both programming and building a simple circuit. The circuit was assembled on a breadboard or in the simulator client; the code was written with the IDE in all cases.

The exercises were as follows:

1. Blink a LED with one second on and one second off.
2. Make 3 LEDs light up in sequence, ensuring that only one is lit at a time, and with a half-second interval.
3. Flash a set of 3 LEDs intermittently (all at the same time) only when a push button connected to the Arduino is pressed.

5.3 Questionaires

We defined a questionnaire adapted from the NASA Task Load Index (NASA-TLX) [9] which are questionnaires created by the Human Performance Group of the National Aeronautics and Space Administration to assess the workload when accomplishing a given task. This has the dual advantage of considering the point of view of the subjects and including subjective aspects such as discomfort or stress.

NASA-TLX questionnaires consist of 2 parts. In the first, 6 subjective parameters are assessed: *Mental Demand, Physical Demand, Temporal Demand, Performance, Effort* and *Frustration*. Subjects grade each parameter using a scale of 1 (very low) to 20 (very high). This grading is related to the execution of one task and thus, it is repeated for each task. The second part assesses the importance each subject assigns to each parameter and it is given only once at the end of the test. In this part, parameters are compared in pairs and for each pair subjects are asked to identify the parameter most relevant to them. This allows assigning weights to each parameter (for each subject) and compute an overall workload index experienced by each individual. We decided not to use the *effort* parameter since in our context it can be captured individually by the the *physical demand* and *mental demand*. We also adapted the scale from 1-20 to 1-6 to avoid pressuring the students with excessive accuracy when classifying each of the parameters. We introduced additional questions to understand the students background and later assess any possible correlation with the performance shown when using the simulator (Table 5 in next subsection).

5.4 Results and Discussion

After a first assessment of the questionnaires, we noticed that not all students answered all questions, either in the first part or in the second part of the questionnaire and we excluded those from the results. This resulted in a total of 189 valid exercises, 89 of which were performed in the real environment and 100 in the simulator (Table 3).

Table 3 Valid and invalid inquiries.

	Total	Invalid	Valid
Students	101	3	98
Exercises	212	23	189

It should be noted that not all students performed the 3 exercises proposed in class because teachers did not impose a time limit for the exercises (and we did not want to change their usual process) and students only moved on to the next exercise when they finished the previous one. Table 4 shows the number of students that executed each exercise.

In class 1 the students performed all the exercises in a row, having only counted the total time and only responded once to the first part of the questionnaire. However, this happened to both real Arduino and simulator students and the comparison between them remains possible. This was not planned and happened due to insufficient initial communication between the parts involved and was corrected in the following tests.

Figure 4 presents for both real and simulated Arduino the average and median workload index experienced by the students when performing the exercises, and the average and median time they took to solve the exercises.

As we can see, the workload index appears to be approximately the same for both real and simulated Arduino, suggesting that the use of the simulator does not greatly interfere with the overall effort experienced by the students, although when using the simulator it is about 8% lower, which is a positive result towards the use of the simulator.

Table 4 Distribution of exercises across classes.

	Exercise 1		Exercise 2		Exercise 3	
Class	Sim.	Real	Sim.	Real	Sim.	Real
1	8	13	-	-	-	-
2	12	16	12	12	4	3
3	11	8	10	8	8	2
4	5	4	5	4	5	4
5	7	5	7	5	6	5

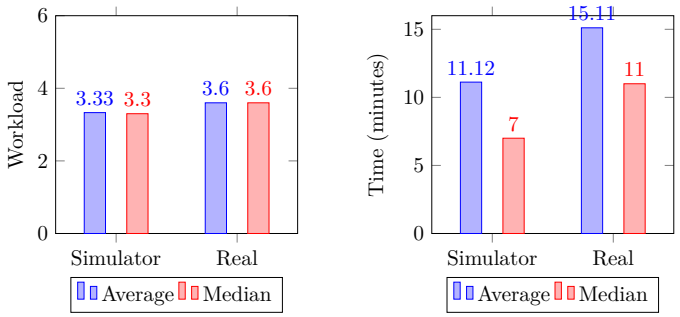

Figure 4 Comparison of workload and exercise time.

Concerning the time parameter, we noticed that students using the simulator take significantly less time to solve the exercises: 26% on average and 36% median. Combined with the lower Physical Demand, this may indicate that using the simulator is more intuitive than making electrical connections on a breadboard. This result suggests that using the simulator is beneficial considering the number of exercises possible to execute during the lesson.

Regarding the Physical Demand (Figure 5), we also observed a significant improvement (one third of the physical demand). We expected an improvement as it is easier to move a mouse than handling small components. The fact that the improvements are so significant is a very encouraging result suggesting that the use of the simulator positively impacts the learning process. Considering Mental Demand and Performance (exercise completion), the results are the same for both groups of students (Figure 5). This was also expected: there was no time limit, so completion depends mostly on the exercise itself; mental performance should also not vary much as the IDE and the programming effort is the same in both cases. This actually is in accordance to the goals of not introducing intrusiveness in the development process.

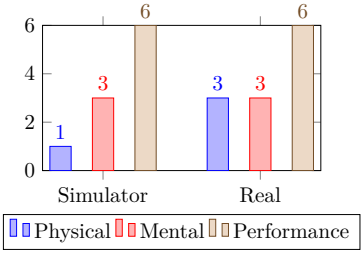

Figure 5 Comparison of the median of Physical Demand, Mental Demand and Performance.

To confirm that the results obtained were not influenced by the students background and previous experience, we analyzed the correlation between the answers about previous background (in the questionnaire) with the workload and the time taken to solve the exercises. We computed the Point Biserial Correlation [13] between the "Yes" answer to the three questions in Table 5 and the average of time to solve the exercises and the average of workload during the exercises.

In the case of previous experience using drawing software and previous experience with Arduino, the correlation is very low, suggesting that these two are not related to the tests results. Concerning the previous experience in programming, the correlation is a little higher but also not significant.

Table 5 Correlation between "Yes" answer, Time and Workload.

Question	Time	Workload
Do you use drawing programs?	−0.099	0.030
Had you already done programming before this school year?	−0.131	−0.158
Have you done programs for Arduino before this course?	−0.047	−0.025

5.5 Teacher point of view

The teacher's point of view is very relevant to our analysis. The teachers of the classes of the case study were involved in all preparation steps and in their opinion, the use of the simulator did not introduced any extra class-management work, and dispensing the handling of components alleviated the beginning and ending of the lesson. There is the need of one initial explanation to the students about the simulator, but that is just for the first lesson using it. So far it seems that the simulator does not involve extra workload to the teachers.

6 Conclusions

Given the increasing use of the Arduino Platform as a key learning tool it is pertinent to address the aspects where this type of use can be improved. We identified a set of aspects where the use of Arduino in classroom can benefit from the use of an Arduino simulator, including new opportunities that can be explored to the mutual benefit of teachers and students. We presented the planning and development of an Arduino simulator that although it can be used for general purpose, it is specifically aimed at its use in education context as its goals and requirements were based on the needs we identified for classroom use. The simulator was implemented in Java and can be run in the typical computers usually found in schools. It has a web-based client-server architecture allowing it to be centrally managed and remotely used, enabling distance learning scenarios. Most importantly, it is compatible with the usual IDE for Arduino and has no impact on the developer usual procedures.

We tested and validated the use of the simulator in classroom in a case-study involving two high-schools comprising five classes using the same exercises already planned by the teachers in regular context. We collected metrics regarding mental and physical workload experienced by the students, and also performance-related metrics such as time spent and exercise completeness. We concluded that the use of the simulator did not have any negative impact on the students or class management, and observed a significant improvement on the physical workload and in the time needed to solve the exercises. This improvement can have a very positive impact on the efficiency of the lesson time, making possible more exercises per

lesson. Concerning the point of view of the teachers, our feedback is that no negative aspects were introduced, class management is easier and all that is needed is one initial explanation to students concerning the use of the simulator.

We analysed the correlation of the background of the students with the results obtained. We did not find any significant correlation and we assume that the performance improvements we observed are indeed related to the use of the simulator, suggesting that its use in classroom is beneficial for learning goals.

We identify several avenues for future work: the continued enhancement of the simulator to pursue further functionality and enable new potential, the continuation of tests in more classroom-related scenarios, and the opening of the simulator as an open-source project [1] to increase its visibility and use. We believe that this simulator is a positive contribution to promote early and broadened digital-literacy and we will look for and pursue any opportunities of partnership with education officials and state-sponsored projects to disseminate the simulator in schools.

References

1 Francesca Agatolio and Michele Moro. A workshop to promote arduino-based robots as wide spectrum learning support tools. In *Robotics in Education*, pages 113–125. Springer, 2017.

2 Arduino.cc. Arduino - home, 2018. URL: `https://www.arduino.cc/`.

3 Atmel. AVR Instruction Set Manual, 2016.

4 Atmel. ATmega328/P Datasheet, 2018.

5 Avrdude. Avr downloader/uploader, 2019. URL: `https://www.nongnu.org/avrdude/`.

6 Hernando Barragán. Wiring: Prototyping physical interaction design. *Interaction Design Institute, Ivrea, Italy*, 2004.

7 Edward Currie and Carl James-Reynolds. The use of physical artefacts in undergraduate computer science teaching. In *E-Learning, E-Education, and Online Training*, pages 119–124. Springer, 2017.

8 Paulo F. Gonçalves, Jorge Bernardino, and João Durães. Virtualization technologies for arduino simulation. In *2019 14th Iberian Conference on Information Systems and Technologies (CISTI)*. IEEE, June 2019. `doi:10.23919/cisti.2019.8760727`.

9 Sandra G Hart and Lowell E Staveland. Development of nasa-tlx (task load index): Results of empirical and theoretical research. In *Advances in psychology*, volume 52, pages 139–183. Elsevier, 1988.

10 Peter Jamieson. Arduino for teaching embedded systems. are computer scientists and engineering educators missing the boat? In *Proceedings of the International Conference on Frontiers in Education: Computer Science and Computer Engineering (FECS)*, page 1. The Steering Committee of The World Congress in Computer Science, Computer Engineering and Applied Computing (WorldComp), 2011.

11 Marc Prensky. Programming is the new literacy. *Edutopia magazine*, 2008.

12 John Sarik and Ioannis Kymissis. Lab kits using the arduino prototyping platform. In *2010 IEEE Frontiers in Education Conference (FIE)*, pages T3C–1. IEEE, 2010.

13 Robert F Tate. Correlation between a discrete and a continuous variable. point-biserial correlation. *The Annals of mathematical statistics*, 25(3):603–607, 1954.

14 Annette Vee. Understanding computer programming as a literacy. *Literacy in Composition Studies*, 1(2):42–64, 2013. `doi:10.21623/1.1.2.4`.

[1] `https://github.com/pafgoncalves/ArduinoSimulator/`