Using Raspberry Pi and Wyliodrin for teaching novice programmers in Secondary Education

Orfanakis Vasileios, Papadakis Stamatios
vorfan@gmail.com, stpapadakis@gmail.com
Secondary Education

Abstract

Although programming is, in general, a motivating topic, most of the students become discouraged as they perceive programming as a very difficult task. Learning to program is one of the challenges for novices. The reason is that it consists both of acquiring programming knowledge (syntax and semantics) and developing problem-solving skills. In this paper, we present an approach for teaching programming to secondary education students using the Raspberry-Pi and the visual programming environment of Wyliodrin. The main purpose of the proposed case study was to engage students, increasing their curiosity about single-board computers, and to convey a positive conception of the tiny computer and programming environment of Wyliodrin as a powerful and amusing learning tool for novice programmers. The positive learning results from this case study indicate that Raspberry-Pi and Wyliodrin can be used as tools by supporting knowledge construction and programming, learning through the design of meaningful authentic projects, learning by doing in both the virtual and physical world, facing cognitive conflicts and learning by reflection and collaboration.

Key words: Raspberry-Pi, Wyliodrin, educational robotics, novice programmers, secondary education

Introduction

Programming is considered a challenging and difficult area of learning for a significant numbers of novice programmers (Fesakis & Serafeim, 2009; Malik, 2016). Many studies (Dagdilelis, Satratzemi, & Evangelidis, 2004; Ioannou & Angeli, 2013) show that learners have misconceptions regarding the curriculum of secondary education computer science, such as, on programming languages and basic computing concepts. Thus, even if programming is in general a motivating topic, most of the students become discouraged initially as they perceive programming as a very difficult task (Papadakis, Kalogiannakis, Orfanakis & Zaranis, 2016). A main reason is the fact that students are mainly taught programming skills and not problem solving methodology (Crawford, Andujar, Jackson, Applyrs & Gilbert, 2016; Mikropoulos & Bellou, 2013). There are many proposals for teaching and learning programming (Mikropoulos & Bellou, 2013). Using robots has been shown to increase students' engagement in learning programming (Crawford et. al, 2016; Papadakis & Orfanakis, 2017). A system that integrates a visual programming language and a robotic kit such as Raspberry could serve as a novel concept that engages students to become more interested in programming.

This paper shows the use of Raspberry-Pi and the visual programming environment of Wyliodrin for the creation of a custom Web-based automation system which displays the temperature of a selected area. The goal of the present work is to leverage students' general curiosity for programming by using an open source platform as a tool for constructionist learning by giving an example of using Raspberry Pi in learning programming.

Κ. Παπανικολάου, Α. Γόγουλου, Δ. Ζυμπίδης, Α. Λαδιάς, Ι. Τζωρτζάκης, Θ. Μπράτιτσης, Χ. Παναγιωτακόπουλος (επιμ.), Πρακτικά Εργασιών 5ου Πανελλήνιου Συνεδρίου «Ένταξη και Χρήση των ΤΠΕ στην Εκπαιδευτική Διαδικασία», σ. 561-570, Ανώτατη Σχολή Παιδαγωγικής & Τεχνολογικής Εκπαίδευσης, 21-23 Απριλίου 2017.ISSN 2529-0924, ISBN 978-618-83186-0-1.

Difficulties for novice programmers

Learning to program is one of the challenges for novices. The reason is that it consists both of acquiring programming knowledge (syntax and semantics) and developing problem solving skills (Malik & Coldwell-Neilson, 2016). The unnatural and complex syntax which is characteristic of many traditional programming languages is a major stumbling block for novices (Papadakis, Kalogiannakis, Orfanakis & Zaranis, 2014; Good & Howland, 2016). Traditional approaches to teaching programming place more emphasis on the syntax and semantics of the language rather than problem solving strategies to address programming problems (Malik, 2016). As Winslow (1996, p. 17) states "novice programmers know the syntax and semantics of individual statements, but they do not know how to combine these features into valid programs" (Malik, 2016). Thus, high failure and drop-out rates from introductory programming (IP) courses are reported despite extensive research which attempts to address the issue (Malik, 2016).

In the current climate of "computation for all" (Papadakis, 2016; Papadakis, Kalogiannakis & Zaranis, 2016; 2017) there has been a renewed focus on determining the "best" language for novices (Good & Howland, 2016). In this comparison, there is an implicit assumption that blocks-based languages are a better solution for novice programmers compared to more "serious" text-based languages (Orfanakis & Papadakis, 2014) a view shared by the learners themselves (Good & Howland, 2016). During the last decade, mobile devices and robotics has attracted the high interest of teachers and researchers as a valuable tool to develop cognitive and social skills for students from pre-school to high school and to support learning in science, mathematics, technology, informatics and other school subjects or interdisciplinary learning activities (Alimisis, 2013; Zaranis, Kalogiannakis & Papadakis, 2013). During the whole process that is the construction of the robotic systems, writing, downloading and executing the appropriate program, students think about the problem under study, design their own meaningful projects, create things and manipulate objects, reflect, and collaborate (Mikropoulos & Bellou, 2013). This makes it easier for the students to overcome certain difficulties when working with programming. Educational robotics contributes to the understanding of the notional machine (that describes the role of the machine to the programming) and its relation to the physical machine. Working with robots, students also shorten or even eliminate the distance between the "objects of the world" and the "computational objects" such as variables (Mikropoulos & Bellou, 2013). As the price of credit card-sized single-board computers continues to drop, the feasibility of using these devices as a novel tool in the classroom (Orfanakis et al., 2016; Papadakis & Kalogiannakis, 2017) for the creation of cognitively controlled virtual environments becomes more realistic.

About Raspberry Pi

Raspberry Pi is a small, powerful, cheap, hackable and education-oriented computer board introduced in 2012 (Figure 1). This credit card-sized computer with many functions is affordable at €30-40 and is the perfect platform for interfacing with many devices.

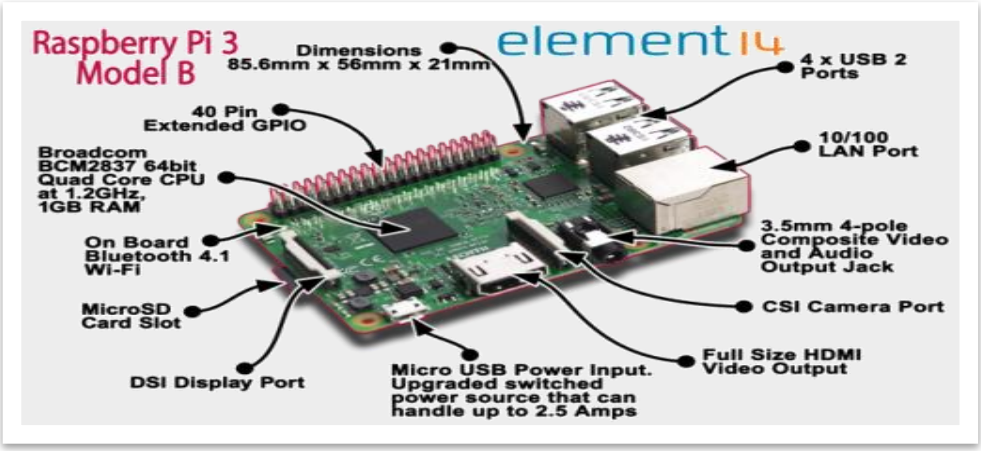

Figure 1. Raspberry Pi 3 (Source: https://www.element14.com)

Raspberry Pi 3 is the third generation of Raspberry Pi. Raspberry Pi provides a wide range of educational usage. This includes but is not limited to, the use of GPIO (General Purpose Input/Output) which allows automated data acquisition and producing simple digital control systems in a school laboratory setting (Jain, Vaibhav & Goyal, 2014). Other projects of Raspberry Pi usage with sensors, automation, displays and motors are given by Bell (2014), Dennis (2013), Goodwin (2013), Monk (2014), Warner (2013).

About Wyliodrin

Wyliodrin (https://www.wyliodrin.com) comes from the old Gallic words "wylio" and "drin", which summarise what the platform does: it monitors and handles the treatment/control of Raspberry Pi. One of the basic problems faced by students who want to engage with electronics creation and programming is the use of a generic-purpose programming language like C which is hard to understand and use. The combination of Raspberry Pi and Wyliodrin seems to deal with this difficulty, bringing novice programmers closer to engineering and to building things. Wyliodrin implemented pins, LEDs and button blocks, to allow users to run several applications. With Wyliodrin, a user can program the Raspberry Pi computer board using a web browser (recommended browsers: Chrome, Firefox or Safari). Thus, Raspberry Pi programming is possible regardless of its geographical location as the necessary code is stored on the Wyliodrin servers (Tataru & Culic, 2014).

Wyliodrin provides a visual, drag-and-drop programming language which is based on Google Blockly, offering an easy way for interaction with the real world, as well as for the development of intuitive human – machine interfaces. Blockly, is an open source development kit that can be used to create new block-based visual programming languages. Due to its intuitive and easy-to-follow block interface Blockly is usually more accessible to novices than standard programming languages (Crawford et. al, 2016). In such visual environments, learners can assemble functioning programs using only a mouse by snapping together instructions and receiving visual feedback, informing the user if a given construction is valid (Weintrop & Wilensky, 2015). The Wyliodrin visual programming

interface is accessible via a web browser or by using Wyliodrin STUDIO (Figure 2) which is a Chrome based IDE.

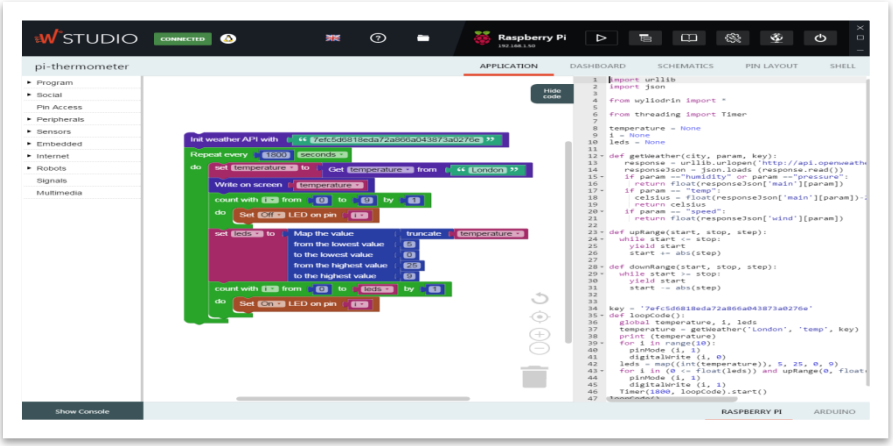

Figure 2. Wyliodrin STUDIO

The didactic approach

In the design of this didactic approach, instead of using a "black box" approach (i.e. the robot has been constructed or programmed in advance and is introduced in the learning activity as an end or a passive tool) we followed the "white-box" approach. In this approach users, can construct and deconstruct objects, can program robots from scratch and have a deep structural access to the artefacts themselves rather than just consume ready-made technological products (Alimisis, 2013). The goal is to make students reflect on the strong relation between the theoretical contents and a direct real experience, to lead them to think about the world in an informed and scientific way. The cost-effective quality of Raspberry Pi was particularly appreciated in this case to provide more kits during the lessons, one for every two or three students.

This study followed the Problem-based Learning (PBL) approach as a programming teaching method. In PBL, the students face real world problems which help them to enhance their "disciplinary knowledge, higher order thinking and practical skills" (Malic, 2016). For the PBL implementation we followed the seven steps as proposed by Nuutila (2005).

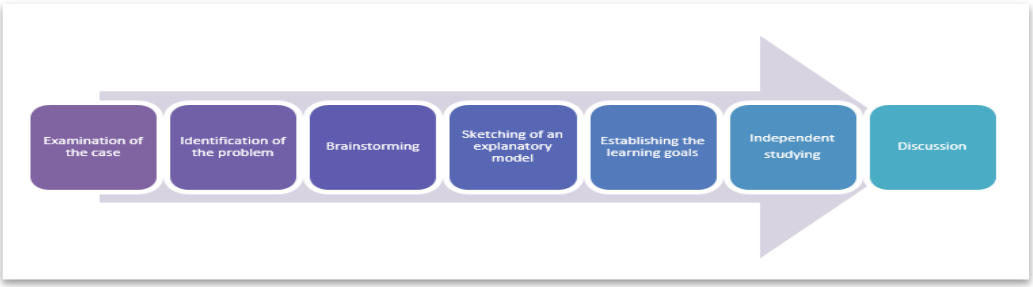

Figure 3. The seven steps of a PBL implementation (adopted by Nuutila, 2005)

Regarding the students, the goals of a didactic approach of this type should be (Agatolio & Moro, 2017):

- to convey the concepts of sensor, actuator and microcontroller,
- to learn how these material interact with each other,
- to make them glimpse some practical applications of the theoretical knowledge,
- to inform the students of the existence of affordable robotic tools that can be further explored out of school,
- to learn basic programming concepts.

The main theories behind educational robotics are constructivism and constructionism. In these methodologies, the educator does not act as a teacher—an authority that transfers ready knowledge to students—but rather acts as an organizer, coordinator and facilitator of learning for students (Agatolio & Moro, 2017). During the didactic approach the educator's role was to offered opportunities for children to engage in hands-on explorations and to provide tools for children to construct knowledge in the classroom environment (Alimisis, 2013).

The course

The course was 2 hours, once a week for ten weeks. During the first five weeks, students were given simple worksheets to learn about Raspberry Pi and its features. Then, some hours were devoted to becoming familiar with creating simple assemblies using LEDs, buttons, resistors etc. which were used to respond to specific stimuli. The students were given assignments each week that had to be turned in during the lab. The assignments focused on how students could use basic programming concepts and Wyliodrin features, such as, using conditional statements, loops, variables, saving a project to reuse it in another file, etc. At the sixth week, the students were asked to create a project appropriate to a device resembling a thermometer with the ability to display the temperature of any region in the world using an API from the website openweathermap.org. Figure 4 and 5 show abstracts of two worksheets aimed at preparing students both in the use of materials and the use of the visual programming environment. Both sheets provide the schematics and the code students are going to use. In the first worksheet (Figure 4) students became accustomed with the Raspberry Pi by using the GPIO pins to make an LED blink.

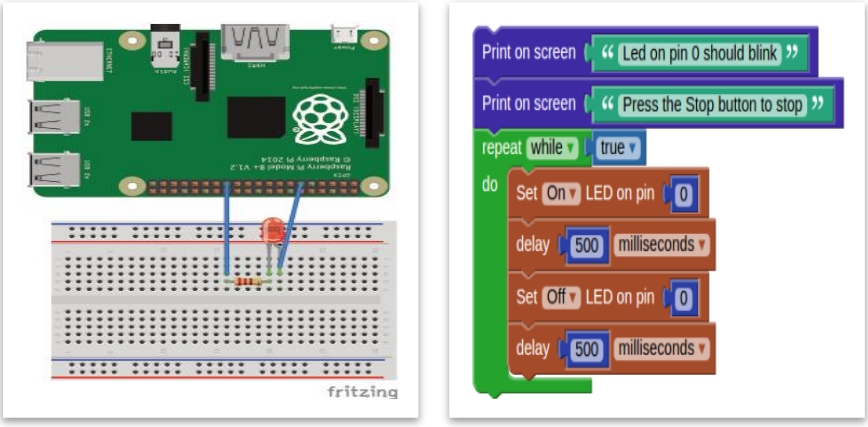

Figure 4. Worksheet 1: LED blink schematics and code

In the second worksheet (Figure 5), students learnt how to signal an S.O.S. message using an LED or a buzzer.

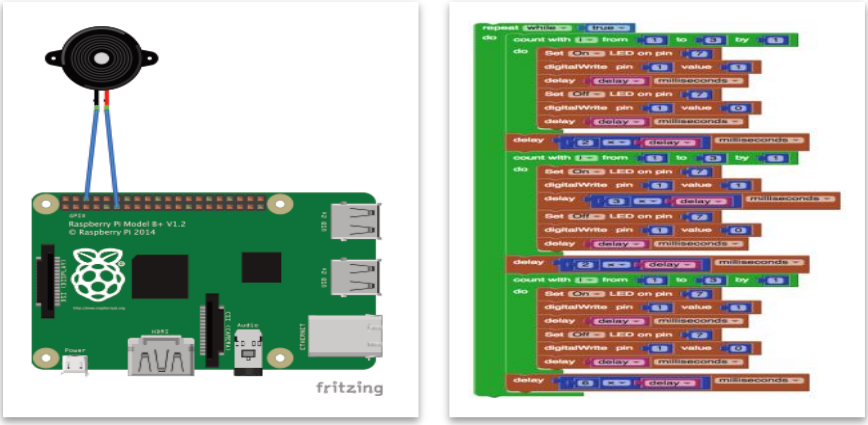

Figure 5. Worksheet 2: Buzzer SOS schematics and code

In both examples, students had the ability to see the same code in the Python programming language (Figure 6). The Wyliodrin environment automatically generated the Python code and novices were able: a) to realize the benefits of the visual programming interface and b) to become familiar with a real programming language.

```
from wyliodrin import *
from time import *

pinMode (0, 1)

print('Led on pin 0 should blink')
print('Press the Stop button to stop')
while True:
  digitalWrite (0, 1)
  sleep ((500)/1000.0)
  digitalWrite (0, 0)
  sleep ((500)/1000.0)
```

```
delay = None
i = None

pinMode (0, 1)
delay = 300
digitalWrite (0, 0)
sleep ((delay)/1000.0)
while True:
  for i in range(1, 4):
    digitalWrite (0, 1)
    sleep ((delay)/1000.0)
    digitalWrite (0, 0)
    sleep ((delay)/1000.0)
  sleep ((2 * delay)/1000.0)
  for i in range(1, 4):
    digitalWrite (0, 1)
    sleep ((3 * delay)/1000.0)
    digitalWrite (0, 0)
    sleep ((delay)/1000.0)
  sleep ((2 * delay)/1000.0)
  for i in range(1, 4):
    digitalWrite (0, 1)
    sleep ((delay)/1000.0)
    digitalWrite (0, 0)
    sleep ((delay)/1000.0)
  sleep ((6 * delay)/1000.0)
```

Figure 6. Python code for LED blink and Buzzer SOS applications

For the project implementation, the students used a set of different colored LEDs, a set of 220-ohm resistors as well as a breadboard and jumper wires. The cost of these materials was 45 euros. The duration of the project was 5 weeks. Each week the students had to complete a specified task using worksheets. In the first week, students had to correctly set up the Raspberry Pi. In the second week, they had to create the thermometer application using Wyliodrin. In the third week, students had to build the thermometer, by correctly

connecting the LEDs to the GPIO pins of the Pi and visualizing the pins in Wyliodrin. In the fourth week, the students learned how to retrieve data from the Web using an API (openweathermap.org/appid). The openweathermap.org website offers an API that allows someone to get weather details from cities around the world. Finally, in the fifth week the students completed and tested their application. The application read the weather from the city of their choice and mapped the value to the number of LEDs. The device read the temperature from the Internet every 30 minutes and updated accordingly (Figure 7 and 8).

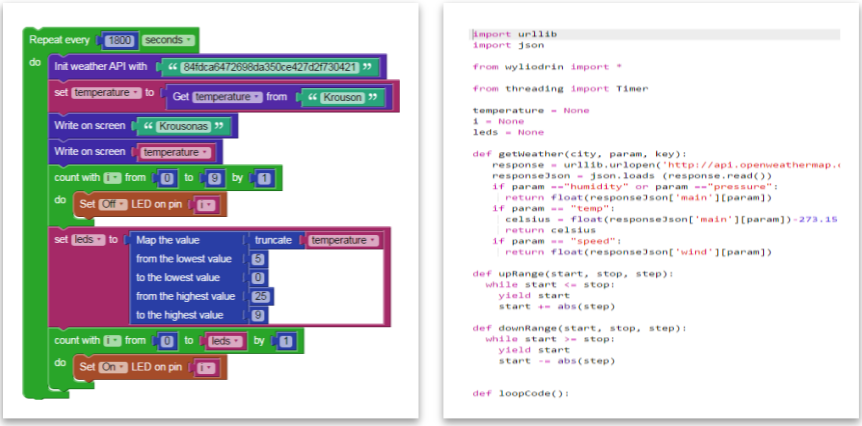

Figure 7. Blocks and Python code for final project

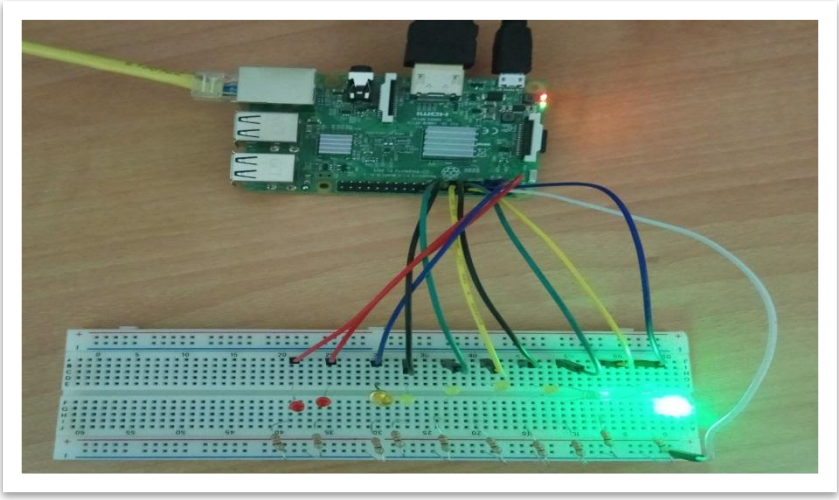

Figure 8. Final project implementation

Evaluation of the educational activity

The teaching proposal that we present was conducted during the 2015-2016 school year in a General Lyceum (Upper Secondary School) in Heraklion, Crete, with the involvement of first-year students who attended the subject entitled "Computer Applications". 23 students (15 boys and 8 girls) between the ages of 15 and 16 participated in the activity. All students were taught some introductory lessons in the LOGO programming language in the third grade of Gymnasium. For the evaluation of the activity's effectiveness regarding students attitude towards programming as well as their knowledge acquisition in basic concepts, data was collected through semi-structured interviews prior to and after the teaching intervention. The development of the interview questionnaire was based on the prior literature review. The questionnaire was tested in four pilot interviews with students who were not included in the final sample and minor revisions were made to clarify some of the questions. All the interviews were performed by the same researcher in the time of student's free activities. Each participant was interviewed individually and the data collection time for each patient lasted for almost 15 to 30 minutes. Follow-up questions were both closed and open-ended. Students provided their views concerning issues related to the teaching objectives. Answers to open questions were categorized and analyzed quantitatively. The analysis of the children's answers concerning the programming concepts showed that the didactic approach contributed to understanding and correct use of the basic programming concepts such as input, output processes, variables, procedural flow, loops and conditions. Similarly, their knowledge about the data types revealed an impressive degree of improvement. The answers to the closed questions were analyzed using IBM SPSS 21.0 statistical software, and revealed no statistically significant differences between the different types of students' conceptualizations in relation to gender.

Concerning students' attitudes towards programming, in their majority they noted that they had enjoyed the activities. The students positively commented that they had obtained information in both different contexts outside the "formal" classroom environment and that the information was not limited to standard teaching material only. In general, and despite some technical problems and obstacles (such as slow internet connections and time), all students were very enthusiastic about the activity. They also said that they hoped to repeat the activity in the next school year, as all of them noted that they had acquired new knowledge in a pleasant and creative way. Some of the students' answers are presented in Figure 9.

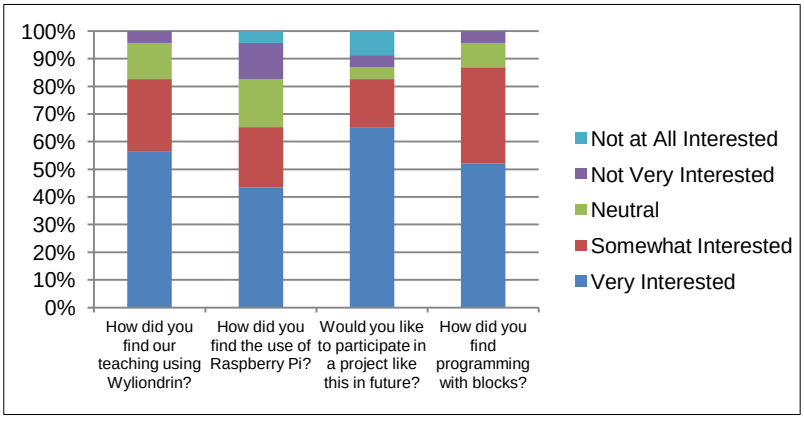

Figure 9. Interest graphs

Conclusion - Discussion

Instructors of introductory programming classes are faced with the challenge of helping novice programmers learn to design, build, and debug computer programs (Al-Linjawi & Al-Nuaim, 2010). Robotics has much potential to offer in education, however, the benefits in learning are not guaranteed for students just by the simple introduction of robotics in the classroom (Alimisis, 2013).

The purpose of the study was to present a didactic approach which supports the teaching and learning process for novice programmers and the same time encourages students to focus on all aspects of learning to program adequately. The characteristic of the didactic approach is that it has a "low floor and high ceiling": specialist skills are not required to start using the toolset while, at the same time, the toolset allows users to produce projects of substantial complexity.

The current study suggests that, in general, the outcomes of the intervention are positive. Both quantitative and qualitative analysis showed that the learning goals were reached, the children became interested in science and technology and developed significantly better cognitive and social skills. Additionally, on students' attitudes, our perception is that they enjoyed being part of the project. Moreover, some of the students seemed to be interested in the possibility of carrying on autonomously with the activity outside the school. Students understood that they can build programs without being professionals, and at the same time they recognized their own potential to use technology in any professional path of their choice. Undoubtedly, a more complete evaluation of the didactic approach need to be done. While robots have positive educational potential, they are not panacea. A drawback of this study is that the presented results are dependent on teacher or student perceptions rather than rigorous research designs based on student achievement data.

It is a good premise to convince schools and single students to invest in robotics and thus to promote its wider diffusion. Our experience has also already showed its flexibility: Raspberry Pi is easily adaptable to different levels of competence and school stages; it can be used both to introduce the fundamentals of robotics/electronics and to develop more complex projects.

References

Agatolio, F., & Moro, M. (2017). A Workshop to Promote Arduino-Based Robots as Wide Spectrum Learning Support Tools. In *Robotics in Education* (pp. 113-125). Springer International Publishing.

Alimisis, D. (2013). Educational Robotics: new challenges and trends. *Themes in Science and Technology Education*, 6(1), 63-71.

Al-Linjawi, A. A., & Al-Nuaim, H. A. (2010). Using Alice to Teach Novice Programmers OOP Concepts. *Journal of King Abdulaziz University: Science*, *22*(1), 59-68.

Bell, C. (2014). *Beginning sensor networks with Arduino and Raspberry Pi*. Apress.

Crawford, C. S., Andujar, M., Jackson, F., Applyrs, I., & Gilbert, J. E. (2016). Using a Visual Programing Language to Interact with Visualizations of Electroencephalogram Signals.

Dagdilelis, V., Satratzemi, M., & Evangelidis, G. (2004). Introducing secondary education students to algorithms and programming. *Education and Information Technologies*, *9*(2), 159-173.

Dennis, A. K. (2013). *Raspberry Pi home automation with Arduino*. Packt Publishing Ltd.

Fesakis, G., & Serafeim, K. (2009). Influence of the familiarization with scratch on future teachers' opinions and attitudes about programming and ICT in education. In *ACM SIGCSE Bulletin* (Vol. 41, No. 3, pp. 258-262). ACM.

Good, J., & Howland, K. (2016). Programming language, natural language? Supporting the diverse computational activities of novice programmers. *Journal of Visual Languages & Computing*. http://dx.doi.org/10.1016/j.jvlc.2016.10.008

Goodwin, S. (2013). *Smart Home Automation with Linux and Raspberry Pi*, 2nd Edition, Apress Media.

Ioannou, I., & Angeli, C. (2013). Teaching computer science in secondary education: A technological pedagogical content knowledge perspective. In *Proceedings of the 8th Workshop in Primary and Secondary Computing Education* (pp. 1-7). ACM.

Jain, S., Vaibhav, A., & Goyal, L. (2014). Raspberry Pi based interactive home automation system through E-mail. In *Optimization, Reliabilty, and Information Technology (ICROIT), 2014 International Conference on* (pp. 277-280). IEEE.

Malik, S. I. (2016). *Role of ADRI model in teaching and assessing novice programmers* (No. PhD). Deakin University.

Malik, S. I., & Coldwell-Neilson, J. (2016). A model for teaching an introductory programming course using ADRI. *Education and Information Technologies*, 1-32.

Mikropoulos, T. A., & Bellou, I. (2013). Educational robotics as mindtools. *Themes in Science and Technology Education*, *6*(1), 5-14.

Monk, S. (2014). *Raspberry Pi Cookbook*. O'Reilly Media, Inc.

Nuutila, E., Törmä, S., & Malmi, L. (2005). PBL and computer programming—the seven steps method with adaptations. *Computer Science Education*, *15*(2), 123-142.

Orfanakis, V., & Papadakis, S. (2014). A new programming environment for teaching programming. A first acquaintance with Enchanting. *The 2nd international virtual Scientific Conference - Scieconf 2014* (pp. 268-273). EDIS - University of Zilina, Slovakia.

Orfanakis, V., Papadakis, St., Kalogiannakis, M., Ampartzaki, M., & Vassilakis, K. (2016). Digital Student Conference Platform Implementation: The case study of the "Research Project" course. *Open Education*, *12*(2), 5-23.

Papadakis, S. (2016). Creativity and innovation in European education. 10 years eTwinning. Past, present and the future. *International Journal of Technology Enhanced Learning*, *8, 3/4,* 279 - 296.

Papadakis S., & Orfanakis V. (2017). The Combined Use of Lego Mindstorms NXT and App Inventor for Teaching Novice Programmers. In: Alimisis D., Moro M., Menegatti E. (Eds.), *Educational Robotics in the Makers Era*. *Edurobotics* 2016. *Advances in Intelligent Systems and Computing*, vol 560, pp.193-204. Springer, Cham.

Papadakis, S., Kalogiannakis, M., & Zaranis, N. (2016). Developing fundamental programming concepts and computational thinking with ScratchJr in preschool education: a case study. *International Journal of Mobile Learning and Organisation*, *10*(3), 187-202.

Papadakis, S., Kalogiannakis, M., & Zaranis, N. (2017). Designing and creating an educational app rubric for preschool teachers. *Education and Information Technologies*, 1-19.

Papadakis, S., Kalogiannakis, M., Orfanakis, V., & Zaranis, N. (2014). Novice Programming Environments. Scratch & App Inventor: a first comparison. In *Proceedings of the 2014 Workshop on Interaction Design in Educational Environments* (p. 1). ACM.

Papadakis, S., Kalogiannakis, M., Orfanakis, V., & Zaranis, N. (2016). Using Scratch and App Inventor for teaching introductory programming in Secondary Education. A case study. *International Journal of Technology Enhanced Learning,* 8, (3/4), 217 - 233

Papadakis, St., & Kalogiannakis, M. (2017). Combining mobile technologies in environmental education: A Greek case study. *International Journal of Mobile Learning and Organisation*, (Forthcoming article).

Tatarou, M., & Culic, I. (2014). Programming the Raspberry Pi from a web browser using a visual language, The MagPI, Issue 22, Apr. 2014.

Warner, T. L. (2013). *Hacking Raspberry Pi*. Que Publishing.

Weintrop, D., & Wilensky, U. (2015). To block or not to block, that is the question: students' perceptions of blocks-based programming. In *Proceedings of the 14th International Conference on Interaction Design and Children* (pp. 199-208). ACM.

Zaranis, N., Kalogiannakis, M., & Papadakis, S. (2013). Using mobile devices for teaching realistic mathematics in kindergarten education. *Creative Education*, *4*, 1-10.