Learning with Robots in CS and STEM Education: A Case Study with ISEP-R0B0

Patrick Wang[1], Ilaria Renna[1], Frédéric Amiel[1], Xun Zhang[1]

1 Institut Supérieur d'Électronique de Paris, 75006 Paris, France

*** patrick.wang@isep.fr**

Abstract

ISEP-R0B0 is a project which combines a small programmable robot and a visual programming language. Its goal is to provide a full-fledged system at a very low cost, targeting schools but also informal learning situations such as after-class activities. Through the programming and observation of the behavior of the robot, students can learn notions either related to Computer Science or Science, Technology, Engineering, and Mathematics. Since ISEP-R0B0 is still at an early stage, this article focuses on introducing the design of the system and two case studies we plan on conducting shortly.

Introduction and Context

Research in Computer Science Education (CSE) has long tried to introduce robots in programming courses. Oftentimes, the objective is to foster students' interest and creativity through "the design of tangible and interactive object using programmable hardware" [14], also known as physical computing. In this regard, results indicate that students experience an increase in motivation [7, 17] and that underrepresented populations in Computer Science (CS) courses feel empowered [16]. However, learning outcomes can vary depending on the context and course taught [2, 4].

Two aspects of programming often cause difficulties to beginners [3,9]: the complexity of a programming language and the structural instructions of a programming algorithm. By focusing on the design of algorithms rather than on the code implementation itself, Visual Programming Languages (VPL) can alleviate these two issues. We can mention the cases of Scratch [12] and App Inventor [6] which were both used to teach fundamental CS principles. A more recent example is the BBC micro:bit project, from which the design of ISEP-R0B0 is inspired. The micro:bit is a "pocket-sized, codeable computing device" which is programmable with an online block-based VPL [1]. Though useful as a tool to foster creativity and increase motivation [17], we could not find any publication identifying the effects of using the micro:bit in acquiring CS and programming knowledge.

ISEP-R0B0 is composed of a programmable robot and a VPL. We expect learners to program ISEP-R0B0, and to see the robot itself as a playful tool for the embodiment of CS or Science, Technology, Engineering, and Mathematics (STEM) concepts. Other systems using block-based VPLs were previously designed with similar intents: .NET Gadgeteer [7], LEGO Mindstorms [2], Thymio-II [10], and more recently micro:bit [1]. However, distinctions can be drawn on the granularity of each VPL and on the cost of each product. Indeed, ISEP-R0B0 provides *low-level instruction* blocks of code to program the robot (in opposition to LEGO Mindstorms or .NET Gadgeteer which use *high-level action* blocks). While Thymio-II can translate blocks into a dedicated event-based programming language (in this case ASEBA), ISEP-R0B0 and micro:bit produce more standard programming languages (respectively Python and Javascript). Finally, we wish to design an affordable robot (with a retail price similar to the micro:bit) while providing features present in higher-end robots.

The ISEP-R0B0 project is still at an early stage. That is why, in this article, we mainly address the design of ISEP-R0B0 and two case studies. In the first one we want to identify the differences in learning outcomes when a novice programmer (e.g., K-12 students) observes the results of her code (which concerns fundamental programming notions such as variables, conditional structures, and loops) on a tangible object instead of only in a web-based simulation. In the second one we wish to explore the use of ISEP-R0B0 in education through the design and enactment of learning scenarios in STEM by programming the robot. In conclusion, we discuss on the results we expect to achieve from these studies and describe our next steps.

Design of ISEP-R0B0

A key consideration during the design of ISEP-R0B0 is usability. Since we expect a broad range of users (basically, anyone whom is interested in being introduced to CS and programming), the system must (1) allow for a variety of scenarios, (2) be easily programmable, and (3) offer a seamless experience from programming to flashing the program on the robot.

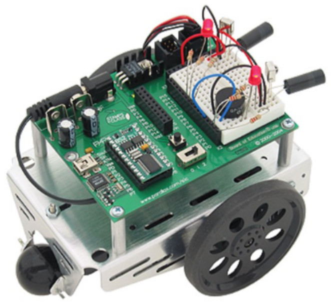
Figure 1. A first prototype of ISEP-R0B0. Board design and microcontroller unit will be subject to changes.

The illustration in Fig. 1 depicts a prototypical version of ISEP-R0B0. It comprises two servomotors, LEDs, an accelerometer, an electromagnet, a microphone, and IR and light sensors. With these components, the robot would be able to move around its surroundings, detect and move objects, and track and move along lines. The board design and microcontroller unit (MCU) will be subject to changes to allow an easier integration of new components and programming of the robot. In particular, we are considering options which use MicroPython[1].

The second specification takes advantages of VPLs to design programs, as evidenced by studies using block-based programming languages [6, 12, 15]. In the case of ISEP-R0B0, we built a VPL based on Blockly[2] and embedded it in a web application (see Fig. 2). A decisive factor in favor of Blockly is its ability to easily design new blocks and to generate their equivalent Python codes.

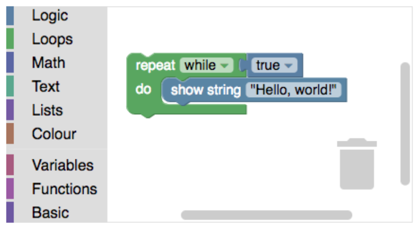

Figure 2. The interface displaying blocks to program the robot. The equivalent Python code can be generated, but is not illustrated due to space restrictions.

To comply with the third requirement, we allowed users to generate the Python code from the program designed with blocks. By implementing a script on our server, we can create the equivalent binary file which can be downloaded on the user's laptop[3]. By simply dragging and dropping this binary file into the memory storage of the robot, the user can update the firmware of the robot, and thus visualize the results of her program directly from the behavior of the robot.

[1]See `https://micropython.org/` for more information.
[2]See `https://developers.google.com/blockly/` for more details on Blockly.
[3]See `https://uflash.readthedocs.io/en/latest/_modules/uflash.html#hexlify` for this step.

Finally, we have planned the design of ISEP-R0B0 to make it possible to add new components, though not doable by any user. The process follows three steps: (1) the soldering of the new component onto the board, (2) the design of the necessary custom block with Blockly, and (3) the implementation of a new method to control the component in the MicroPython library.

Case Studies

This section introduces two separate case studies. The first one aims towards identifying the benefits of using the robot and/or its simulation for students learning the elementary concepts of programming (i.e., variables, conditional structures, and loops). The second one involves the use of ISEP-R0B0 for STEM education. In particular, the goal is to see whether the use of an easily programmable robot in specific learning scenarios helps learners to understand STEM related notions. In both studies we will consider three participant age groups: primary (6-10 years old), middle (11-15 years old), and high school (16-19 years old). This will let us compare the benefits of using a robot and/or a simulation depending on age and customize STEM experience depending on age.

Tangible objects and simulations in CSE

Research in CSE which introduces robots shows that students are motivated and eager to use the systems described [2, 4, 5, 17]. However, these projects lack comparison between situations where learners use a tangible robot or a virtual one, sometimes merely because there is no simulation tool embedded into the programming interface. This consideration is for instance highlighted by Fagin and Merkle [4], as they observed the difficulties faced by their students because of the lack of simulation to facilitate the process of developing, running, and debugging their programs.

In the case of ISEP-R0B0, we wish to explore the differences in learning outcomes when programming with a tangible object and/or with an equivalent simulation. In the end, the objective would be to create a system in which the tangible object and the simulation can complement themselves in programming learning activities. By enacting learning scenarios which take advantage of both tools and the implemented VPL, we hypothesize that learners can acquire a more complete understanding of key notions of programming. This approach of using both a material and a digital artifact has been for example studied in Mathematics [11].

As a first step, we plan on conducting an experiment which target is to identify the learning outcomes specific to programming with the robot and/or its simulation. The experimental protocol relies on dividing students into three groups: a group which uses the robot, a group which uses the simulation, and a control group which uses both tools. By analyzing the students' interaction with the VPL and their answers to a pre-test and post-test evaluation, we hope to identify distinct learning benefits stemming from the use of the robot or the simulation. The experimentation will focus on two aspects: (1) the level of comprehension of what variables, conditional statements, and loops are, and (2) how long and to which degree this knowledge is retained. At the end of this experiment, our goal is to design scenarios where students would benefit from using both tools in programming learning activities.

As a complementary study, it would be interesting to ask students whether or not they have witnessed differences in behaviors between the robot and its simulation. For instance, if a student programs the robot to move straight forward, the simulation would keep the same direction indefinitely while the real robot might deviate from its initial trajectory due to internal (e.g., wheels might not be exactly parallel) or external (e.g., plane might not be perfectly horizontal) factors. The task here would be for students to realize that their programs are not wrong, but rather that the differences might originate from a faulty hardware design or from the surroundings of the robot influencing its behavior.

ISEP-R0B0 for STEM education

The second case study revolves around the use of ISEP-R0B0 in STEM fields, and more particularly in Mathematics and Physics. Based on the constructionist theory of learning [13], such situations are designed to allow students to create and alter the environment in which the robot is moving. Lots of systems already exist for this specific goal and research projects seem to indicate positive results in the students' learning and motivation (see [8] for a review).

We designed a learning scenario based on a classic problem in kinematics: the inclined plane. The main objective for learners would be to calculate the amount of power to give to the motors so that the robot can climb up a slope. Students would be divided into two groups: one group with the robot, the other solving the same problem on a paper. By initializing differently some set-up parameters (such as the inclination of the plane and/or the material composing the surface of the plane), students would have to gather their knowledge on movement, gravity, and static friction and analyze their effects on the robot.

We expect to see results suggesting that, through the observation and use of the robot, students have a better understanding of the principles in play in this scenario compared to students working on the same problem on paper.

A follow-up study could be to explore the effects of using a VPL for programming a robot in a STEM learning activity. Indeed, out of the 26 systems identified in [8], only five presented a visual programming language. Our concern is that the secondary task of writing lines of code with a complex programming language might interfere with the primary task of solving a problem in Mathematics or Physics. By spending more time analyzing and modeling the problem instead of implementing a technical solution, learners should get a better understanding of the concepts in play.

Conclusion and Future Work

In this article, we presented ISEP-R0B0 which is composed of two main components: a programmable robot and a web application allowing users to program the robots with a VPL. By interacting with ISEP-R0B0, we expect novice programmers to learn key concepts of CS and/or STEM. The robot adopts a voluntarily simple design in order to comply with our objectives: to be affordable and to diffuse in schools.

Concerning STEM activities, the design of ISEP-R0B0 allows for the enactment of learning scenarios. In addition to the one we presented about kinematics, we plan to test other learning scenarios: an objective could be to require students to work with angles in geometry in order to program the robot to make it move out of a maze.

From the presented case studies, we expect to draw conclusions on (1) the learning benefits specific from the use of a tangible object or an equivalent simulation, and (2) the effects of using a robot in a STEM learning activity (and the effects of VPLs in such situations). Results from these experiments would also shed light on learning design specific to the use of robots in CS or STEM education.

The design and development of ISEP-R0B0 is still on-going, and our next steps consist in finalizing the integration between the robot and the web platform allowing for its programming. We also leave space for new components; in such cases, complementary work must be done to create a custom block for this specific feature within the VPL and to generate the equivalent code lines in Python.

Acknowledgments

This project is partially funded by the Fondation Orange.

References

1. T. Ball, J. Protzenko, J. Bishop, M. Moskal, J. de Halleux, M. Braun, S. Hodges, and C. Riley. Microsoft touch develop and the bbc micro: bit. In *Software Engineering Companion (ICSE-C), IEEE/ACM International Conference on*, pages 637–640. IEEE, 2016.

2. D. C. Cliburn. Experiences with the LEGO Mindstorms throughout the Undergraduate Computer Science Curriculum. In *Proceedings. Frontiers in Education. 36th Annual Conference*, pages 1–6, 2006.

3. B. Du Boulay. Some Difficulties of Learning to Program. *Journal of Educational Computing Research*, 2(1):57–73, 1986.

4. B. Fagin and L. Merkle. Measuring the Effectiveness of Robots in Teaching Computer Science. In *ACM SIGCSE Bulletin*, volume 35, pages 307–311. ACM, 2003.

5. S. Gibson and P. Bradley. A Study of Northern Ireland Key Stage 2 Pupils' Perceptions of Using the BBC micro:bit in STEM Education. *The STeP Journal*, 4(1):15–41, 2017.

6. J. Gray, H. Abelson, D. Wolber, and M. Friend. Teaching CS Principles with App Inventor. In *Proceedings of the 50th Annual Southeast Regional Conference*, pages 405–406. ACM, 2012.

7. S. Hodges, J. Scott, S. Sentance, C. Miller, N. Villar, S. Schwiderski-Grosche, K. Hammil, and S. Johnston. .NET Gadgeteer: A New Platform for K-12 Computer Science Education. In *Proceeding of the 44th ACM Technical Symposium on Computer Science Education*, pages 391–396. ACM, 2013.

8. M. E. Karim, S. Lemaignan, and F. Mondada. A Review: Can Robots Reshape K-12 STEM Education? In *Advanced Robotics and its Social Impacts (ARSO), 2015 IEEE International Workshop on*, pages 1–8. IEEE, 2015.

9. E. Lahtinen, K. Ala-Mutka, and H.-M. Järvinen. A Study of the Difficulties of Novice Programmers. In *Acm Sigcse Bulletin*, volume 37, pages 14–18. ACM, 2005.

10. S. Magnenat, J. Shin, F. Riedo, R. Siegwart, and M. Ben-Ari. Teaching a core cs concept through robotics. In *Proceedings of the 2014 conference on Innovation & technology in computer science education*, pages 315–320. ACM, 2014.

11. M. Maschietto and S. Soury-Lavergne. Designing a Duo of Material and Digital Artifacts: The Pascaline and Cabri Elem e-Books in Primary School Mathematics. *ZDM*, 45(7):959–971, 2013.

12. O. Meerbaum-Salant, M. Armoni, and M. Ben-Ari. Learning Computer Science Concepts with Scratch. *Computer Science Education*, 23(3):239–264, 2013.

13. S. Papert and I. Harel. Situating Constructionism. *Constructionism*, 36(2):1–11, 1991.

14. M. Przybylla and R. Romeike. Key Competences with Physical Computing. *KEYCIT 2014: Key Competencies in Informatics and ICT*, 7:351, 2015.

15. A. Schmidt. Increasing Computer Literacy with the BBC micro:bit. *IEEE Pervasive Computing*, 15(2):5–7, 2016.

16. S. Sentance and S. Schwiderski-Grosche. Challenge and Creativity: Using .NET Gadgeteer in Schools. In *Proceedings of the 7th Workshop in Primary and Secondary Computing Education*, pages 90–100. ACM, 2012.

17. S. Sentance, J. Waite, S. Hodges, E. MacLeod, and L. Yeomans. Creating Cool Stuff: Pupils' Experience of the BBC micro:bit. In *Proceedings of the 2017 ACM SIGCSE Technical Symposium on Computer Science Education*, pages 531–536. ACM, 2017.