RoBlock – Web App for Programming Learning

https://doi.org/10.3991/ijet.v11i12.6004

P.G. Feijóo García[1] and F. De la Rosa[2]
[1] Universidad El Bosque, Bogotá, Colombia
[2] Universidad de los Andes, Bogotá, Colombia

***Abstract*—One of the major challenges related to teaching programming and algorithmics to amateur students is the time spent to explain a language's syntax. Also, students who undertake computer programming may find problems that hinder their understanding of concepts and the development of their problem-solving and programming skills. This paper presents the results of an experimental approach that evaluated the interaction of a group of Colombian students with a Web solution within the context of Mobile Robotics to learn programming and algorithmics. The designed Web App is oriented towards autodidactic learning by using Visual Blocks Programming through five interactive modules that include concepts to be learned by students such as the following: variables, sensors, conditionals, cycles, and functions. The solution is designed to present virtual scenarios for Mobile Robotics. This proposal was evaluated with middle school students from the Colombian education system and was compared to the results obtained using Scratch as a reference tool.**

***Index Terms*—Programming learning, programming teaching, student-centered learning, E-Robotics, visual programming language.**

I. INTRODUCTION

In the last decade, there have been multiple projects focused on the design of educational tools to teach programming and algorithmic skills. Several solutions propose game designing, storytelling development, and other types of approaches, offering the student the possibility to learn how to program with a blank canvas and powerful frameworks of Visual Blocks Programming. However, even if these solutions offer particular contexts (e.g. oriented for animation or videogames), they are limited in guiding the student towards the acquisition of specific programming skills, being most of them standalone Apps that depend on particular technical configurations. Most of these solutions are presented as complements to a programming course, which means that the student will also need a teacher to be able to learn.

Even though there has been global advancement in the design and elaboration of new technologies emphasized in programming and algorithmic teaching, within the Colombian educational context, such an advancement has not been a strong support to implement programming skills in classes offered in elementary or high school curriculums. With this national educational schema, the programming and algorithmic knowledge and skills are unjustifiably delegated to the technical and professional education levels, thus causing a small portion of the student population to be interested in studying careers related to Information Technologies (IT).

This issue makes interesting a proposal for a new solution oriented to teaching programming and algorithmic concepts, making possible learning them auto-didactically. Furthermore, this proposal's evaluation with Colombian school students is attractive and innovative.

This paper presents the results obtained after a group of Colombian middle school students interacted with a Web App that was developed focusing on autodidactic learning of programming and algorithmic concepts, and mainly centered in Mobile Robotics as an educational context.

This document is organized as follows: Section II presents some previous works and existing tools related to the context of interest. Section III presents the problem addressed in this study and formulates the corresponding research query that it answers. In section IV, the justification of the use of Mobile Robotics as an educational context is explained. In section V, the design of the mentioned solution is covered, and in section VI the methodology of the experimental phase is explained. Through section VII the results obtained after performing the experiment are exposed, while the final part of this paper, presents the study's conclusions and a proposal future research related to this project.

II. RELATED WORK

A. Related Works in Programming Learning

The study of Adams and Webster was focused on analyzing graphical platforms for algorithmic teaching with Visual Blocks Programming (Scratch and Alice) [1]. It focused on studying 322 projects, implemented by students who used these platforms, so as to examine the frequency of use of different concepts within these projects: variables, conditionals, loops, dialogues and events handling. The latter, in order to answer the questions "*Do students learn different concepts when creating video games compared to creating stories (Storytelling)?*" and "*What algorithmic concepts do students learn and use with this type of platforms?*". This study concludes that the context type used for learning ("Storytelling" vs video games) leads to the acquisition of different concepts in different magnitudes. In the case of Alice, which is clearly aimed at creating video stories, greater use of dialogues within projects is reported, while in Scratch (focused on music videos and video games) the use of cycles and conditional instructions is greater.

Jones *et al.* studied if an introductory computer programming course could intertwine basic programming skills and creativity, looking to demonstrate how the combination of logical thinking and creativity could enhance application development [2]. Working with Alice and its *Storytelling* proposed context, positive indicators were obtained, showing that students were generally more con-

fident, reflective and articulate with the concepts and elements taught through the course. The proposed study is an example of how the variety of pedagogical contexts can enhance the teaching of programming and algorithmics in introductory courses.

Magnenat *et al.* carried out a study using the Thymio-II (self-contained robot for young students) and the VPL (event handling graphical software development environment) with 70 young students of different ages, in order to measure if they could learn about event handling while interacting and having fun within the robotics context [3]. The results obtained followed Bloom and SOLO taxonomies, and included students who even reached an applying level for the studied concept. Only some of the youngest students could not reach a cognitive level. The study concluded that the use of robotics in an enjoyable way, could successfully teach computer science concepts to young students.

Ouahbi *et al.* conducted a study with Scratch, and Pascal as a reference language, in order to evaluate the impact of the friendly videogames development environment with high school science major students, comparing their reaction against students who followed the traditional learning process with Pascal [4]. A population of 69 students participated in the study. The results showed that the use of a friendly environment for learning programing such as Scratch, highly motivates students to pursue further studies in programming. Accordingly, 65% of the students from the sample who experienced Scratch showed an interest to continue further programming studies, while only 10.3% from the group who worked with Pascal showed some interest.

The study of Maloney *et al.* was carried out to discuss the motivation of young students when choosing Scratch as their favorite programming tool. Working with a sample population between 8 and 18 years old, over an 18-month period, they analyzed a total of 536 Scratch projects, focusing on studying which were the programming concepts learned by the students [5]. The obtained results showed that the students, on their own, used commands demonstrating the concepts of user interaction, loops, conditionals, communication and synchronization. Furthermore, although not as common, they also seemed to have knowledge of variables, Boolean logic and random numbers. At the end, the authors conclude their analysis by indicating that Scratch multimedia components are attractive to engage students in programming, thus highlighting the importance of digital media as a promising pathway for new generations.

Kaucic and Asic focused on analyzing the level of satisfaction and the opinion of elementary and high school students who took a first programming course with Scratch [6]. They took into consideration 36 students, categorized into three school grades: 5th, 8th and 2nd. During five months of experimentation, students were trained in using Scratch to create projects aimed at solving specific tasks. At the end of the study, quality variables were presented in terms of satisfaction and motivation after using the tool, reporting that 5th graders presented the highest results for the proposed variables.

Vavougios and Karakasidis conducted a research in which they evaluated the result of applying an ICT (Information and Communication Technology) to teach and learn Physics [7]. Working with a population sample of 176 students, from the Pedagogical and Polytechnic Departments, they studied if the use of a common package known as MATHEMATICA®, throughout a whole semester, could help in the development of critical thinking, thus leading to a better understanding of the concepts of interest. They concluded that the use of computational technologies helped the student to reach cognitive regions, covering topics with difficult mathematic formalism, therefore leading them to state that an ICT, such as the one used in this study, is a powerful tool in the hands of a professor who aims at successfully completing instructive work in a laboratory.

Al-Imamy *et al.* present results related to the implementation of an educational system based on algorithmic "templates" [8]. The study focuses on the need to reduce the dependence on teaching syntax in introductory algorithmic courses, arguing that, by focusing the mind of the student on concepts of non-syntactic algorithms, they develop higher programming skills, compared to students whose first interaction with algorithmics includes topics which cover syntax learning.

B. *Similar Programming Tools*

Similar technological solutions are presented below:

- Scratch [9][10][11] (developed in the MIT Media Lab) proposes a Web and a standalone platform for programming and algorithmic teaching, within an event oriented paradigm. Students learn to program through the development of online games on a blank canvas and a syntax-free Visual Blocks Programming environment.
- RoboMind [12][13] (developed by Research Kitchen) offers a Robotics scenario to teach programming to students. The tool is not auto-contained, which means that the student needs a teacher or a Web course to introduce the concepts to be applied with the tool. Moreover, its free version is limited to be standalone, and its environment exposes a textual language that requires the student to learn syntax.
- Alice [2][14][15] (developed in the Carnegie Mellon University) proposes a standalone platform for programming and algorithmic teaching, which guides students to learn through the development of animated stories using a syntax-free Visual Blocks Programming environment.

III. PROBLEM FORMULATION

Although there are several approaches around the world aimed at teaching programming and algorithmics in a friendly manner, Colombia lacks alternatives that have a strong educational background in this subject. The general trend of the country's schools focuses on teaching basic computer principles and office informatics, and because teaching programming and algorithmics is not even close to being a priority in the Colombian education system nowadays, no detailed studies have been carried out to expose the impact of these new trends in the improvement of the national educational prospect.

In Colombian high school institutions, the existence of introductory programming tools is generally unknown, and if known, limitations may occur because of the poor preparation of the tutor to use such tools in the teaching process. The latter is due to the fact that the tools currently used, if taught at all, only provide an interface or syntacti-

cally friendly language, but lack a self-taught approach or guidance on a specific context. On the other hand, if they have an autodidactic approach, they do not provide an interface or syntactically user friendly language, but focus mainly on presenting and teaching existing programming languages (each of them bringing its corresponding language syntax and learning curve).

Aiming to impact the Colombian educational context, this project was carried out in order to be a pilot initiative to make programming and algorithmics accessible for children and youth in both public and private schools, thus breaking the paradigm based on the need to have a teacher for this purpose. The purpose is to motivate young students to undertake IT and/or computing careers.

Because of the latter, the proposed project RoBlock involves the design, implementation and evaluation of a platform for self-learning of programming and algorithmic concepts through a contextualized approach in Mobile Robotics. The completion of the project is expected to address the following query:

Is it possible for high school Colombian students of distinct educational schemas (public or private) to comprehend programming and algorithmic concepts, after interacting with an educational tool designed to be autodidactic, scalable and oriented towards a specific context: Mobile Robotics?

A first version of the proposed solution is presented at the end of this project, serving as an answer to the aforementioned question.

IV. MOBILE ROBOTICS PURPOSE

Mobile Robotics is a branch of knowledge that focuses on the study associated with the programming of mobile robots for the realization of tasks, having them the feature of moving spatially in two and/or three dimensions [16].

In this study, it is interesting to address the approach of teaching programming and algorithmics, because the concepts involved within the automation of activities and decision-making, and those based on the interaction of the robot with its environment, are the ones that are covered generally by programming lessons and courses.

In Mobile Robotics, the following concepts are involved harmonically, which is why it is a viable teaching context for algorithmic and programming problem solving:

- Variables: When programming a robot, the student can work with different types of variables: spatial positions, angles and Boolean variables to ask if a sensor is active or not.
- Sensors: A Robot incorporates different types of sensors that allow its interaction with its environment.
- Conditionals: Deciding whether to go ahead, if the orientation is changed or if the robot goes back, are some of the questions that are asked when putting a mobile robot within a particular scenario.
- Iterations and cycles: In Mobile Robotics, the automation of activities is fundamental. The teaching of cycles and iterations through robot programming is made easy because, there is a need to repeat steps in paths, as well as to perform iterations until some goal or challenge is achieved.
- Functions: When using robotics as a teaching context, various possibilities for teaching functions are provided. This happens because it is common to do the following in this context: use search algorithms, define behaviors by using priorities, percentages and/or probabilities.

V. ROBLOCK: DESIGNED SOLUTION

The designed solution consists of a Web App oriented towards teaching programming and algorithmic concepts to Colombian students (App with Spanish texts), using Mobile Robotics as an educational context. This App includes six modules: Five of them are focused on one concept to be taught specifically through the programming of virtual Robots (Fig. 1) and the last one offers the possibility to interact with a remote robotics laboratory with physical scenarios (prototyped but not tested in this study).

Figure 1. Main menu screenshot: Available modules.

Each module (except for the first) has a previous module as a prerequisite, and the advancement between them is defined by means of a scheme of points that is evaluated using the following criteria:

- C1: The conducted solution responds to the issues that were raised (issues with a unique response).
- C2: The student solved the problem within a stipulated time.
- C3: The student solved the problem using the blocks created specifically for the module.

Moreover, the student has the possibility of returning to previous module, while being able to advance to further ones. Every new student starts always at the first module and is able to see the global score obtained for every module that is achieved, with the possibility to reset any of them in particular, or even the complete account if desired. Given that the target population is Colombian, the Web App was designed in Spanish, considering that it is oriented towards Colombia's different educational schemas and that most of the students in the national territory are not bilingual. The modules included in RoBlock are the following (Fig. 1): Variables (M1), Sensors (M2),

Conditionals (M3), Cycles and iterations (M4), Functions (M5), and Remote mobile lab (M6).

RoBlock provides two user profiles and gives each of them access to different functionalities. The first profile is for the student, which interacts with the Web Interface and the proposed exercises (Fig. 2). The second one is designed for the system administrator, who is responsible for creating modules and exercises for the students.

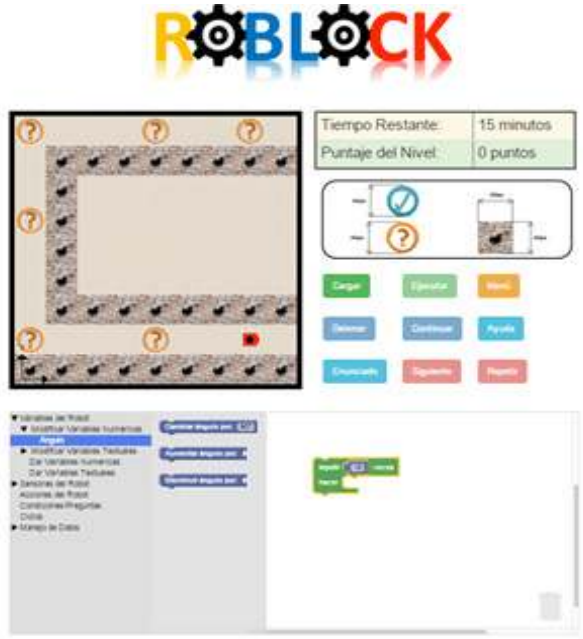

Figure 2. Exercise page: Visual blocks programming.

The system is structured as a multi-layer architecture, delegating tasks and activities to specific software components. It was deployed in two servers, which are described below:

- The first server exists in the cloud and uses Heroku as its Cloud platform. In this server, the components dedicated to the virtual environment, the communication with the remote laboratory server and the Web visualization were deployed.
- The second server exists locally in the remote laboratory used for this study. The components in charge of the Robot programming and the communication with the Cloud server where deployed here.

The front-end interface is user-friendly and has a design that aims to facilitate usability and interaction. It provides all the features expected from a Web App in terms of authentication, registration and user administration; whilst, on the other hand, providing the student with different modules that expose video-tutorials for self-learning.

VI. EXPERIMENTAL PHASE

A. Tools and Technologies

The following tools were used to carry out this study:

- RoBlock v1.0: Educational software specifically designed and developed for the needs of this project. This tool is provided as a mechanism to assess and validate the previously stated research query.
- Scratch v2.0: Educational software used as a reference App for this study. By using it, RoBlock is comparatively evaluated as an educational software for teaching programming and algorithmic concepts. The features of this tool were described in Section II.

Scratch was selected as the main comparative technology, because it is one of the most attractive and proposed tools for learning algorithmics in Colombian schools nowadays. RoBlock is compared to it in order to evaluate its impact on the target population, and to quantify if it is a tool that, as Scratch, could be used in Colombian schools to achieve a successful autodidactic learning of programming and algorithmics. The main purpose of this comparison is not to replace Scratch or any other technology used in schools, but to evaluate RoBlock as a possible complementary software for programming and algorithmic learning.

B. Target Population and Studied Groups

The population of students who participated in this project was chosen based on the following criteria:

- Educational Level: Currently in 6th, 7th, or 8th middle school grades, in accordance with the Colombian education system.
- Age Range: 11 to 14 years old.
- Gender: Students of both genders.
- Psychological Condition: The student does not have characteristics related to learning disabilities or attention deficit.
- Algorithmic Knowledge: None or Minimal.

There was a total of 46 students (9 boys, 37 girls) of an average age of 12.2 years, who came from private and public educational institutions: Two private schools and one public educational organization. Neither of the institutions had used RoBlock nor Scratch in their courses of computer programming or informatics. Based on this, we assumed that the tools were unknown for the students who participated in the study.

C. Experimental Methodology

At the beginning of the study, every student presented a 20-question preliminary test to evaluate his/her prior knowledge of programming and algorithmic concepts. This test aimed to measure the progress and learning achieved during every module proposed in the study.

Subsequently, each student worked for five hours, divided into two or four sessions (decided by the participant institutions), using one of the two educational tools for learning programming and algorithmic concepts: Scratch or RoBlock. Before starting the working sessions, all the students were divided randomly into two sub-groups, each one working particularly with one of the two proposed educational tools.

At the end of every proposed module, each student had to take a specific five-question test focused on evaluating his/her knowledge of programming and algorithmic concepts that should have been acquired during the course. The intention was to measure the progress and knowledge gained in comparison with the initial knowledge evidenced in the preliminary evaluation mentioned before. Similarly, and in parallel, the student answered a number of questions focused on measuring how much he/she con-

sidered he/she had learned, and how interesting the exercises proposed in the module were.

Finally, a quality assessment was carried out exclusively with the students who participated using RoBlock, with the purpose of obtaining feedback in order to improve the first version of the designed Web App. No quality assessment was considered for the Scratch group, because this qualitative approach is specifically proposed for RoBlock's improvement as a tool.

To make the analysis of the results obtained from the experimental phase, three sets of variables and indexes are described in the following section. Therefore, the conclusions required to answer the research query raised at the beginning were obtained.

The groups that were evaluated consisted of 27 students working with RoBlock, and 19 students working with Scratch. This distribution was obtained because the number of students per educational institution was different in the experimental phase, consequently, some groups could not be exactly divided by 2. The students had to work in four modules:

- Module 1: This module was designed to learn about variables. The student was focused on working with position variables, having the need to position the robot (RoBlock) or the main actor (Scratch) around a scene.
- Module 2: This module was designed to learn about sensors and conditionals. The student was focused on working with the displacement sensors of the robot for RoBlock, or of the main actor for Scratch. The proposed exercises guided the student in the use of conditionals in harmony with the available sensors.
- Module 3: This module was designed to learn about cycles and iterations. The problems included in this module followed a maze style, through which the student had to work with cycles in order to move through it, and solve the proposed exercises within the indicated time.
- Module 4: This module was designed to learn about functions. Through it the student had to declare and call his/her own functions, in order to solve the exercises here presented. For this last module, the student had to use all the concepts previously learnt in order to solve the problems exposed.

D. Variables and Indexes of Interest

To answer the research question posed in section III, three sets of indexes were specifically designed and measured for each student *i* per module *m,* and for each tool *t* that was used (RoBlock and Scratch).

The first set of indexes is related to the students' performance after solving the exercises proposed in each tool:

- Performance Index per Student per Module ($PIS_{i,m,t}$): This variable intends to measure the performance of student *i* when solving the exercises of module *m* by using tool *t*. It was defined as the relation (%) between the number of exercises finished by student *i* ($nfe_{i,m,t}$) and the exercises proposed in module *m* for tool *t* ($npe_{m,t}$).

$$PIS_{i,m,t} = \frac{nfe_{i,m,t}}{npe_{m,t}} \tag{1}$$

- Performance Index per Module ($PIM_{m,t}$): This is defined as the average of students' performance in module *m*, when working with the same tool *t.*

$$PIM_{m,t} = \frac{\sum_{i=1}^{S_t} PIS_{i,m,t}}{S_t} \tag{2}$$

S_t represents the number of students working with tool *t*.

- Performance Index per Tool (PIT_t): This is defined as the average of the proven performance in the different modules available in tool *t.*

$$PIT_t = \frac{\sum_{m=1}^{M_t} PIM_{m,t}}{M_t} \tag{3}$$

M_t represents the number of modules solved by the students that used tool *t*.

Two new variables were defined to estimate the knowledge of the students regarding the concepts related to computer programming, which were measured in the different modules using the proposed tools:

- $nfq_{i,m,t}$: Number of right answers obtained by student *i* in the final evaluation in module *m* after working with tool *t*.
- $npq_{i,m}$: Number of right answers obtained by student *i* in the preliminary test in module *m* (i.e. before participating in the learning experiment).

Based on the aforementioned variables, a second set of indexes was designed to measure the improvement of the students as the percentage (%) between the results of the final test versus the results of the preliminary test. These indexes reflect the improvement/gain in knowledge relative to the results obtained in the preliminary test:

- Improvement Index per Student per Module ($IIS_{i,m,t}$): It assesses the level of progress of student *i* in the concepts related to module *m* when the student worked with tool *t*.

$$IIS_{i,m,t} = \begin{cases} \dfrac{nfq_{i,m,t} - npq_{i,m}}{npq_{i,m}} & \text{if } npq_{i,m} \neq 0 \\ nfq_{i,m,t} & \text{if } npq_{i,m} = 0 \end{cases} \tag{4}$$

- Improvement Index per Module ($IIM_{m,t}$): It exists for the purpose of representing the overall number of students who worked with tool *t* throughout each module *m*.

$$IIM_{m,t} = \frac{\sum_{i=1}^{S_t} IIS_{i,m,t}}{S_t} \tag{5}$$

- Improvement Index per Tool (IIT_t): It exists for the purpose of representing the overall number of students who worked with tool *t*.

$$IIT_t = \frac{\sum_{m=1}^{M_t} IIM_{m,t}}{M_t} \tag{6}$$

Finally, we defined a third set of indexes to observe the global learning of the students with respect to the final test.

- Learning Index per Student per Module ($LIS_{i,m,t}$): It defines the average of right answers of student i in the final test, related with module m which was completed using tool t:

$$LIS_{i,m,t} = \frac{nfq_{i,m,t}}{Q_m} \quad (7)$$

Q_m represents the number of questions in the final test that are about the topics included in module m.

- Learning Index per Module ($LIM_{m,t}$): It defines the learning average for module m, considering all the students who worked with tool t:

$$LIM_{m,t} = \frac{\sum_{i=1}^{S_t} LIS_{i,m,t}}{S_t} \quad (8)$$

- Learning Index per Tool (LIT_t): It defines the overall learning average for the students who worked with tool t during the learning experience:

$$LIT_t = \frac{\sum_{i=m}^{M_t} LIM_{m,t}}{M_t} \quad (9)$$

VII. RESULTS

After implementing the proposed experiment, the following results were obtained:

A. Performance Index per Module ($PIM_{m,t}$) and Performance Index per Tool (PIT_t)

The results of this index present a gap between the students that worked with RoBlock, versus the group of students who participated using Scratch. As shown in Figure 3, the group that worked with Scratch obtained a higher result in every module (i.e. solved more exercises), in comparison to the group that worked with RoBlock. The only module in which the gap was narrower is the second one (Sensors and Conditionals).

It is important to highlight that with RoBlock, modules 3 and 4 had the lowest $PIM_{m,t}$ values. The students who worked with RoBlock could solve about 52% of the exercises proposed in each module (Fig. 3).

Compared to the group that worked with Scratch, the module 2 with lowest value of $PIM_{m,t}$ was greater in comparison to the values obtained with RoBlock. The average of exercises per module that were solved using Scratch is estimated to be 81% (Fig. 3).

In each module, for both tools, all students solved at least 33% of the proposed exercises, although some students managed to solve a 100% (Table I).

B. Improvement Index per Module ($IIM_{m,t}$) and Improvement Index per Tool (IIT_t)

The knowledge acquired by using each tool, with respect to the initial knowledge (measured in the preliminary test) is shown in Figure 4. With RoBlock, the students' improvement was an average of at least 70% for

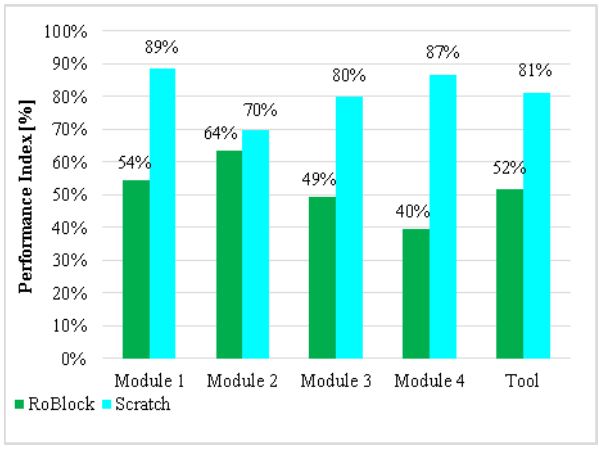

Figure 3. Performance Index per Module ($PIM_{m,t}$), and Performance Index per Tool (PIT_t).

TABLE I.
PERFORMANCE INDEX PER MODULE FOR EACH TOOL

	t = RoBlock			*t* = Scratch		
m	**Minimum [%]**	**PIM [%]**	**Maximum [%]**	**Minimum [%]**	**PIM [%]**	**Maximum [%]**
Mod. 1	33%	54%	100%	45%	89%	100%
Mod. 2	33%	64%	100%	50%	70%	93%
Mod. 3	33%	49%	100%	36%	80%	100%
Mod. 4	33%	40%	67%	70%	87%	100%
Tool		52%			81%	

module 4 and of up to 106% for module 1. With Scratch, the students' improvement was an average of at least 58% for module 3 and of up to 131% for module 4. In general, both tools helped students to achieve a significant learning during the experiment; students evidenced improvements regarding their concepts and their knowledge about programming with an average of 91%.

In addition, the learning improvement was more significant in the last two modules for both tools (Fig. 5). This may be because the students were more familiar with the tools towards the end of the learning experience and/or because the students had learnt more programming concepts with the help of the tools. Therefore, students could solve significantly more questions from the final test than those when they had taken the preliminary test. Additionally, in both tools, there was a similar percentage of students that obtained over the average results.

C. Learning Index per Module ($LIM_{m,t}$) and Learning Index per Tool (LIT_t)

The results of the learning index (Fig. 6) show that there are two modules with better indexes for RoBlock and two other modules with better indexes for Scratch. Nevertheless, the learning index is similar for both tools (38%), which is not considered low when taking into account the five-hour long experimental phase of the study.

It is interesting to compare the learning index (Fig. 6) with the performance index (Fig. 4) in this study. Module after module, the learning index becomes lower, with the exception of module 4 when using RoBlock. There are

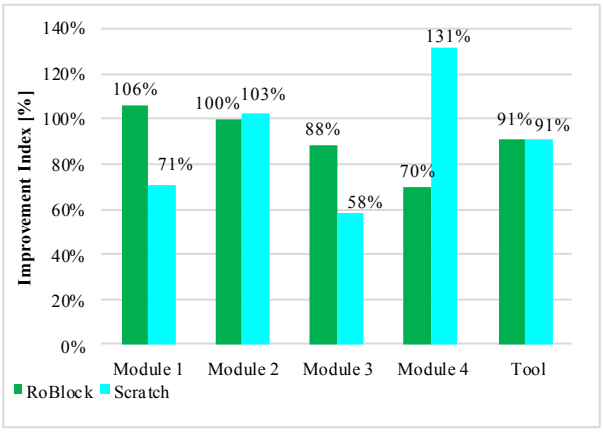

Figure 4. Improvement Index per Module ($IIM_{m,t}$), and Improvement Index per Tool (IIT_t).

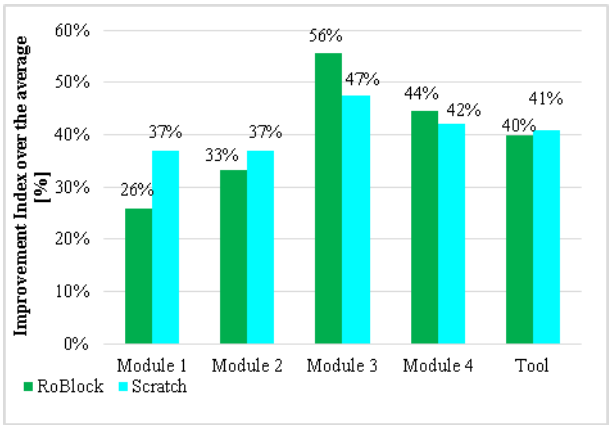

Figure 5. Improvement Index: Percentage of students over the average.

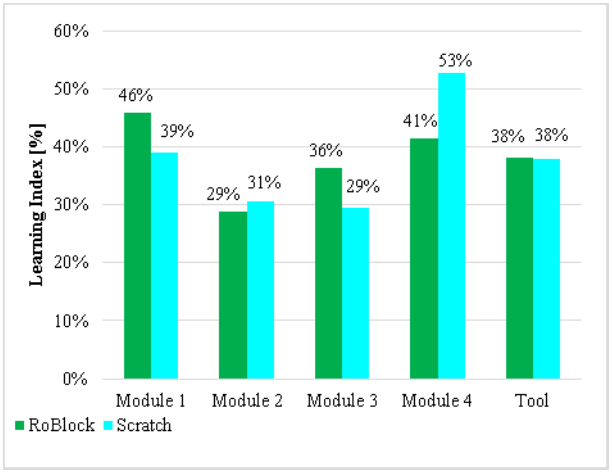

Figure 6. Learning Index per Module ($LIM_{m,t}$) and Learning Index per Tool (LIT_t).

two values to highlight from the group that worked withRoBlock: module 2, which had the highest performance index, also scored the lowest learning index; however, the performance and learning indexes are similar for module 4. On the other hand, for the group of Scratch, the learning indexes are lower than the performance indexes, although they maintain the same behavior.

In order to better understand the results of the learning and performance indexes for RoBlock and Scratch, the

TABLE II. COMPARISON BETWEEN LEARNING INDEX AND PERFORMANCE INDEX FOR STUDENTS WORKING WITH ROBLOCK

		$PIS_{i,t}$						
	Level	**1**	**2**	**3**	**4**	**5**	**Total**	**Stud**
$LIS_{i,t}$	**1**	-	7,4%	3,7%	7,4%	-	18,5%	5
	2	-	18,5%	22,2%	3,7%	-	44,4%	12
	3	-	-	18,5%	-	-	18,5%	5
	4	-	-	7,4%	7,4%	3,7%	18,5%	5
	5	-	-	-	-	-	0%	0
	Total	0%	25,9%	51,9%	18,5%	3,7%	100%	
	Stud	0	7	14	5	1		27

TABLE III. COMPARISON BETWEEN LEARNING INDEX AND PERFORMANCE INDEX FOR STUDENTS WORKING WITH SCRATCH

		$PIS_{i,t}$						
	Level	**1**	**2**	**3**	**4**	**5**	**Total**	**Stud**
$LIS_{i,t}$	**1**	-	-	-	15,8%	-	15,8%	3
	2	-	-	5,3%	31,6%	5,3%	42,1%	8
	3	-	-	-	5,3%	31,6%	36,8%	7
	4	-	-	-	-	5,3%	5,3%	1
	5	-	-	-	-	-	0%	
	Total	0%	0,0%	5,3%	52,6%	42,1%	100,0%	
	Stud	0	0	1	10	8		19

levels obtained by each student for the exercises solved with the tool and for the questions solved in the final test were normalized. For each index, we defined five levels from 1 (lowest) to 5 (highest). Each student was classified into one cell according to his/her learning index ($LIS_{i,t}$) and his/her performance index ($PIS_{i,t}$). The distribution of the group that used RoBlock is detailed in Table II. 44% of the students (biggest group) reached level 2 in the $LIS_{i,t}$, whilst 18% reached level 4, the highest level. Furthermore, 51% (biggest group) obtained level 3 in the $PIS_{i,t}$, and only 3.7% obtained level 5. Additionally, 52% of the students had a $LIS_{i,t}$ equal or higher than their $PIS_{i,t}$; therefore, these students confirmed to have learned with the support of RoBlock. 48% of the students had their final results lower than these obtained with RoBlock.

Table III shows the results for the group that worked with Scratch. The biggest group in $LIS_{i,t}$ has 42% of the students categorized in level 2, where level 4 is the highest level for $LIS_{i,t}$, which has 5% of the students. On the other hand, 52% obtained level 4 in $PIS_{i,t}$, and 42% obtained level 5. It is important to highlight that with this tool, the learning index results were lower than the performance index results.

As a common fact, in both tools, there were no students who got level 1 in the performance index nor level 5 in the learning index.

D. Level of Interest reported per Module

As shown in Figure 7, students answered favorably to the question: "*Do you consider the exercises proposed for this module interesting?*"

In both groups the average interest for the exercises was higher than 92%, which shows that the students were motivated throughout the study, and that the exercises were considered friendly and interesting.

E. *Quality Assessment*

The group that worked with RoBlock was asked to answer a quality assessment to evaluate the experience with the first version of the designed App.

The first question was intended to determine if RoBlock was interesting for the participating students (Fig. 8). The results show that 74% of the students considered it very interesting, and 22% interesting, which means its debut was a success within the target population.

Figure 9 shows the results for the second question in this assessment. As can be seen, 48% of the students considered RoBlock an appropriate autodidactic tool, while 52% disagreed. Nevertheless, taking into account that the study comprised a five-hour working session, the population of students that considered RoBlock autodidactic was favorable. This, also considering that the educational schema to which they are used to is highly dependent on the presence of a tutor or a teacher.

VIII. CONCLUSIONS AND FUTURE WORK

A first conclusion is that RoBlock complies with the main objective of being autodidactic, scalable and with a contextualized approach to Mobile Robotics. This, based on the results of the interaction between the students and the tool, which goes beyond the experimental results detailed in the previous section.

On the other hand, this study answers the research query raised at the beginning of this article affirmatively. It showed that it is possible for middle and high school students to learn programming and algorithmic concepts by using an autodidactic and scalable tool, oriented in a specific context of interest (Mobile Robotics). Such a conclusion arises based on the results obtained for the improvement index, in which the interval presented for the students who worked with RoBlock has a gain between 70% and 106%, with an average of 91%, for a lapse of five hours of interaction with the tool (duration of the study). Additionally, there are interesting results showing the relation between the performance of the students working with RoBlock and their learning of programming concepts.

Moreover, another conclusion is that the first version of RoBlock was a success, and that it is a nice, interesting and friendly tool, which was well received by the target population to which it was directed.

Overall, the study provided satisfactory results, and this pilot project opens multiple possibilities for it to be expanded and evolve. Projects like RoBlock and Scratch are necessary to break an educational scheme as rigid as the Colombian one, where tools like these offer young students the possibility to learn concepts on subjects that have been unjustifiably delegated in technical and higher education institutions.

Following this study, and having designed and implemented RoBlock's first version, a wide horizon for future work involving the tool is raised.

This study evaluated only the virtual schema RoBlock offers through its modules. Evaluating the use of physical and remote scenarios with students is worthwhile, in order to measure the remote interaction of students with physical Robots in pre-configured scenarios.

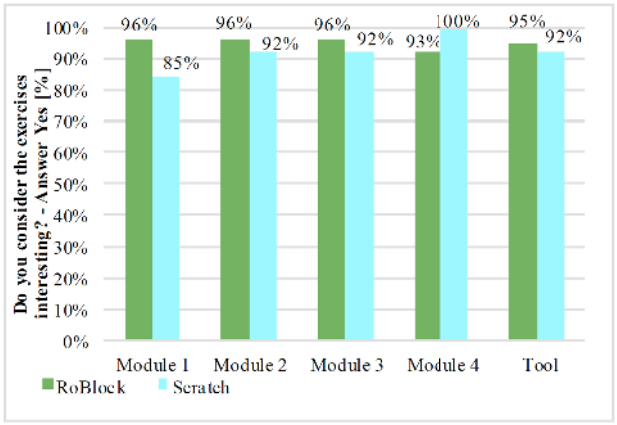

Figure 7. Level of interest reported per module, and per tool.

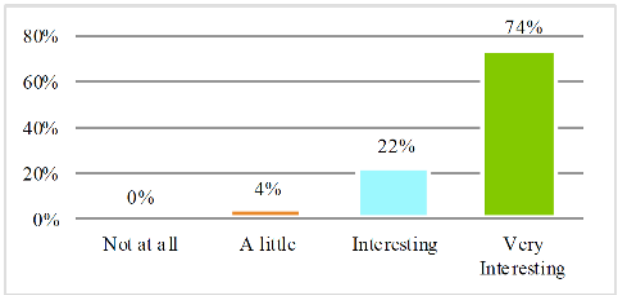

Figure 8. Quality Assessment – Do you consider RoBlock an interesting tool?.

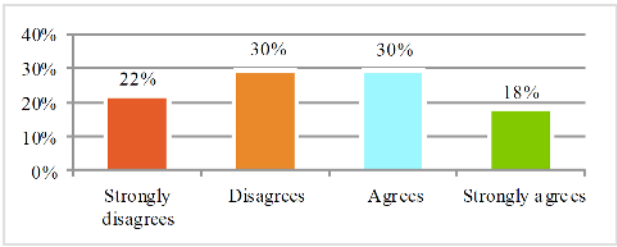

Figure 9. Quality Assessment – Do you believe RoBlock is appropriate for autodidactic learning?.

Similarly, a future study could include the evaluation of the module's exercises with different pedagogical schemas. This, in order to quantify the success of various educational approaches, seeking to find the type of exercises that would be able to optimize the performance, improvement, and learning indexes proposed in this study. Likewise, a future study could even repeat this same study extending the experimentation time, aiming to find the minimum period required to internalize and learn the desired concepts completely.

Furthermore, this study could be carried out again with different exercises for the tools used: Scratch and RoBlock. This could be done in order to standardize the amount of exercises proposed per module, thus achieving the minimum influence of this variable on future results.

It is evident that there are several possibilities to extend the project, because the current pilot study successfully shows that RoBlock fulfills the purpose for which it was designed. The tool can be extended, as well as the methodology used, by modifying and improving the schema that was proposed and implemented in this first step.

ACKNOWLEDGMENT

Our sincere gratitude to the educational institutions that helped us with the study. Colegio Provinma – Providencia Inmaculada (Bogotá, Colombia), Centro Educativo y Cultural Reyes Católicos (Bogotá, Colombia), and Programa Progresa Fenicia (Universidad de los Andes, Bogotá, Colombia) were essential for this study to succeed, since they efficiently collaborated with their students and with all the necessary computational resources.

REFERENCES

[1] J. C. Adams, and A. R. Webster, What do students learn about programming from game, music video, and storytelling projects?, Proceedings of the 43rd ACM Technical Symposium on Computer Science Education - SIGCSE, 2012, pp. 643-648. https://doi.org/10.1145/2157136.2157319

[2] M. E. Jones, M. Kisthardt, and M. A. Cooper, Interdisciplinary teaching: Introductory programming via creative writing, Proceedings of the 42nd ACM Technical Symposium on Computer Science Education SIGCSE, 2011, pp. 523-528. https://doi.org/10.1145/1953163.1953313

[3] S. Magnenat, J. Shin, F. Riedo, R. Siegwart, and M. Ben-Ari, Teaching a core CS concept through robotics, Proceedings of the 2014 conference on Innovation & Technology in Computer Science Education – ITiCSE, 2014, pp. 315-320. https://doi.org/10.1145/2591708.2591714

[4] I. Ouahbi, F. Kaddari, H. Darhmaoui, A. Elachqar, and S. Lahmine, Learning basic programming concepts by creating games with Scratch programming environment, Procedia - Social and Behavioral Sciences, Vol. 191, 2 June 2015, pp. 1479–1482. https://doi.org/10.1016/j.sbspro.2015.04.224

[5] J. Maloney, K. Peppler, Y. Kafai, M. Resnick, and N. Rusk, Programming by choice: Urban youth learning programming with Scratch, ACM SIGCSE Bulletin-SIGCSE 08, Volume 40, Issue 1, March 2008, pp. 367-371.

[6] B. Kaucic, and T. Asic, Improving introductory programming with Scratch?. 2011 Proceedings of the 34th International Convention MIPRO, 2011, pp. 1095–1100.

[7] D. Vavougios, and T. Karakasidis, Application of ICT technology in physics education: Teaching and learning elementary oscillations with the aid of simulation software, International Journal of Emerging Technologies in Learning (iJET), Vol. 3, No. 2, June 2008, pp. 53-58.

[8] S. Al-Imamy, J. Alizadeh, and M. A. Nour, On the Development of a Programming Teaching Tool : The Effect of Teaching by Templates on the Learning Process, Journal of Information Technology Education: Research, Vol. 5, No. 1, 2006, pp. 271–283. Retrieved from http://www.editlib.org/p/111545/

[9] M. Resnick, J. Maloney, A. Monroy-Hernández, N. Rusk, E. Eastmond, K. Brennan, A. Millner, E. Rosenbaum, J. Silver, B. Silverman, and Y. Kafai, Scratch: programming for all, Communications of the ACM, Volume 52 Issue 11, November 2009, pp. 60-67. https://doi.org/10.1145/1592761.1592779

[10] J. Maloney, M. Resnick, N. Rusk, B. Silverman, and E. Eastmond, The Scratch Programming Language and Environment, ACM Transactions on Computing Education (TOCE), Vol. 10, No. 4, Article 16, 2010.

[11] MIT Media Lab & Lifelong Kindergarten Group, Scratch, 2006. Retrieved from http://scratch.mit.edu

[12] P. Marshman, Engage your pupils with RoboMind, Computing at School NewsLetter, Spring 2011. Retrieved from http://www.robomind.net/en/publications.html

[13] Research Kitchen, RoboMind, 2005. Retrieved from http://www.robomind.ne

[14] S. Cooper, The Design of Alice, ACM Transactions on Computing Education (TOCE), Vol. 10, No. 4, Article 15, 2010.

[15] Carnegie Mellon University, Alice, 1998. Retrieved from http://www.alice.org

[16] M. Mataric, Move it! In The Robotics Primer (Vol. 1), Cambridge, Massachusetts: The MIT Press, 2007.

AUTHORS

Pedro Guillermo Feijóo García, Assistant Professor of the Program of Systems Engineering, Universidad El Bosque, Bogotá, Colombia (e-mail: pfeijoo@unbosque.edu.co).

Fernando De la Rosa, Associate Professor in the Systems and Computing Engineering Department, Universidad de los Andes, Bogotá, Colombia (e-mail: fde@uniandes.edu.co).

Submitted, 2 July 2016. Published as resubmitted by the authors on 25 August 2016.