INVESTIGACIÓN e INNOVACIÓN en INGENIERÍAS
ISSN2344-8652

Comprehensive Visual Development Environment for Learning Robotics Programming

Ambiente visual integrado de desarrollo para el aprendizaje de programación en robótica

Carlos Alejandro Ruíz Ramírez **Jovani Alberto Jiménez-Builes**

Universidad Nacional de Colombia

Diana María Montoya Quintero
Instituto Tecnológico Metropolitano

OPEN ACCESS

Recibido: 28/07/2020
Aceptado: 27/11/2020
Publicado: 14/12/2020

Correspondencia de autores:
caaruizra@unal.edu.co

Abstract

Objective: This work seeks to review some positive aspects and characteristics of using robotics in academic environments and to describe some features of programming languages used in robotics implementation. **Methodology:** This work explains how a development environment implemented through cloud computing features can be used to teach robotics at K-12 and college levels. The programming environment combines visual programming using Business Process Management (BPM) notation and text-based programming. **Results:** The obtained results reveal that a gradual transition from visual languages to text-based programming languages is feasible and present the advantages of teaching standards aimed at developing process modeling skills. **Conclusions:** This study demonstrates the potential exhibited by educational programming environments for teaching robotics in a cloud computing environment, thus reducing the gap between visual programming and text-based programming using multiple robotic devices within the same development scenario.

Keywords: Educational Robotics, Visual Programming, STEM, Fourth Industrial Revolution, Artificial Intelligence.

Resumen

Objetivo: Revisar algunos aspectos y características positivas del uso de la robótica en ambientes educativos y describir algunas de las características de los lenguajes de programación utilizados en su implementación. **Metodología:** se explica como un ambiente de desarrollo implementado mediante el uso de algunas características de computación en la nube, puede ser usado para aplicar robótica en escuelas, colegios y universidades. El entorno de programación combina la programación gráfica utilizando notación de procesos de trabajo (BPM) y programación textual. **Resultados:** se demostró que se puede hacer una transición gradual de lenguajes visuales a lenguajes de programación textuales, así como las ventajas de aprender estándares que permiten el potenciar el aprendizaje de habilidades en el modelado de procesos. **Conclusiones:** se comprobó la capacidad que tienen los entornos de programación diseñados con propósitos educativos para trabajar la robótica en un ambiente cloud computing, reduciendo la brecha que permite pasar de una programación gráfica a una programación textual usando múltiples dispositivos robóticos, en un único escenario de desarrollo.

Palabras clave: robótica educativa, programación visual, STEM, cuarta revolución industrial, inteligencia artificial.

Como citar (IEEE): C. Ruíz-Ramírez., D. Montoya-Quintero., y J. Jiménez-Builes. "Ambiente visual integrado de desarrollo para el aprendizaje de programación en robótica". Investigación e Innovación en Ingenierías, vol. 9, n°1, 7-13, 2021. DOI: https://doi.org/10.17081/invinno.9.1.3957

Introducción

This study discusses part of the results from the master's thesis titled "Comprehensive Development Environment for Visual/Textual Hybrid Programming Integrated with Robotic Artifacts for STEM Education via Computer Programming," for which a development environment was created based on the particular characteristics of Colombia [1].

Artificial intelligence (AI) has evolved in tandem with computing advancements. In fact, as early as 1956, J. McCarthy had already used this term to refer to the ability demonstrated by machines to make decisions commonly associated with human cognitive capabilities. AI is broadly defined as the study of simulating human intelligence behavior in electronic devices and machines [2]. Even though computer sciences are typically considered a subset of AI, the AI field is ultimately multidisciplinary and depends on multiple human knowledge areas for its proper application, such as logic, biology, electronics, physics, and language studies.

This study focuses on AI's educational possibilities, such as its capacity to adapt academic content, make learning strategies more flexible, and give autonomy to schools, colleges, and universities. This is true even when AI is used in various applications, such as image recognition, data processing, trend forecasting, business decision-making, autonomous car driving, and robotics. AI has also been widely used in the construction and configuration of different academic robotic devices [3] for several purposes [4, 5], thereby clearly distinguishing between two different approaches: robotics aimed at education and educational robotics.

Prensky [6, 7] compared the technological viewpoints of new generations who grew up with technology to adults who had adapted to the use of digital tools at the start of the twenty-first century, identifying the need to incorporate technologies into the classroom environment. Furthermore, several authors [8, 9] claimed that new generations pose a natural disposition and facility for using information and communication technology (ICT). In fact, ICTs conveniently generate student interest and strengthen science, technology, engineering, and mathematics (STEM) learning.

In recent decades, different ways of adapting ICT in the classroom have been proposed. For example, through online courses (e-learning, b-learning, t-learning, u-learning) or the application of AI techniques, such as smart tutoring environments and robotics, to enhance teaching and learning support

One of the first experiences using AI for academic purposes was Logo, a high-level and easy-to-learn language used to teach basic programming to children and adolescents. These early studies demonstrated that teachers could use computer programming to develop new ways of understanding, thinking, and learning [10]. Additionally, these studies demonstrate that using computers is important and necessary for acquiring new knowledge as they facilitate learning processes in students.

In [11], the authors studied the pedagogical challenges of the 21st century. They demonstrated the radical changes to traditional teaching that are required to incorporate a technological component because technology has essentially changed how people understand the world around them, interact with others and learn [12].

In [11], related works were reviewed, highlighting the constructivist characteristics of educational robotics and its approach to solving problems with social activities [13]. Based on the foregoing, the authors compared a group of robots used in education, considering their modularity, reusability, versatility, and price. Similarly, robots were classified into versatile robots containing small parts that can be assembled according to

different needs and non-versatile platforms with a factory-defined structure. Finally, they discussed the mechanisms that control robotic platforms so that they can perform the previously programmed tasks. They noted that there are different robot control paradigms, such as wireless commands, visual language programming, and textual language programming.

Different pedagogical approaches may be used in the classroom according to the characteristics of each robotic device. The most basic robotic devices lack the ability to execute user-programmable instructions, and they are limited to performing the restricted tasks for which they were designed (e.g., bristlebot, solar cars, quadruped robots) [14]. Basic robotic devices are regularly used at basic academic levels because they help students develop an interest in electronics and robotics at an early age. However, robotic devices are programmable and modular at the most advanced levels and may execute multiple programmed activities simultaneously with high precision. For example, Ozobot, Lego EV3, Arduino, Raspberry PI, and Bioloid Kit. Among these advanced robotic devices, three types can be identified: commercial robotic kits (Lego EV3, Bioloid), single-board computers (Arduino, Intel Galileo, Raspberry PI, Asus Tinker Board S, and BeagleBoard), and custom devices built by researchers using microcontrollers.

In [15], the authors highlighted different artifacts used in robotics according to the desired complexity and academic level (for example, Lego NXT, Vernier, VMCU, and HomeLab Kit). HomeLab and DistanceLab were proposed as blended learning environments because they provide a context that facilitates direct programming and remotely controls devices through a web programming environment. According to their study, educational robotics had already been implemented in several educational environments in different countries. They [15] further explained the adoption process within the Estonian educational system, from basic education to the university level, using blended learning methodologies with microcontrollers and electronic devices [16].

This study describes the design and construction of an integrated development platform that uses a hybrid programming environment (graphic-textual and cloud computing-based) and serves as a supporting tool for learning sequential robot programming of robot applications in basic, middle, and higher education. This paper contains the following sections: the materials and methods, emphasizing the review of the programming languages used in educational robotics; the methodology used to build the programming environment; the results obtained from this study; the conclusions and references used.

Materials and Methods: Programming Languages used in Educational Robotics

The robotic artifacts programming requires programming languages specially adapted to the specific hardware, which, due to their possible complexity, are often used at higher academic levels [17]. These programming languages can be divided between textual programming languages [18], in which instructions follow specified text-based syntax, and graphic or visual programming languages [19], in which instructions are inputted in an interconnected visual block form. Some of the most representative graphic or visual languages are Scratch, LabView, and Lego. Some common textual programming languages are C#, JavaScript, Java, C/C++, Python, Pascal, and Lisp.

One of the most representative visual programming languages used in education is Scratch [20], which is intended for children and teens between the ages of 8 and 16. This language facilitates the autonomous learning of basic programming concepts. In Scratch, routines are built like a puzzle by interlinking blocks representing instructions. The edge of these blocks represents how they can be combined with other

blocks, making it easier to prevent syntax errors. Scratch has no user-defined functions and no additional data structures that can be used to store information.

Alternatively, the programming environment was included with the Lego EV3 device, which was developed jointly using Lego and LabVIEW. This is a fast-learning language; however, it is intended for primary education students [15]. The EV3 environment includes a number of interconnectable boxes that perform specific tasks. In this environment, green blocks represent action blocks, orange blocks let users control a program's flow, yellow blocks operate sensors and read data, red blocks operate on data, blue blocks perform advanced activities, and cyan blocks represent user-defined functions. Each box has a unique image that identifies the action performed and its configuration parameters.

Finally, National Instruments' LabVIEW is an example of a graphical programming environment. This is a visual development environment where interconnecting parameter-driven blocks conduct actions. Applications built in LabVIEW use a graphical user interface (GUI) to generate dashboards through which they can manage and monitor events. LabVIEW necessitates technical expertise because it allows for the construction of highly complex and versatile programs [21]. However, its high cost limits its use in industrial environments.

Nonetheless, several general-purpose programming languages, such as Java, C, C++, and Python, have been adapted to specific robotics hardware [22]. These programming languages generally do not involve additional costs since their software applications can be implemented for free on different hardware devices (See Table 1)..

Table 1. List of Some Programming Languages used in Educational Environments for the Teaching of Robotics

Language	Input	Programming Paradigm	Compatible Robotic Devices and Implementation
C	Textual	Imperative	Lego EV3 (ev3dev)
			Lego NXT (RobotC)
			Arduino
			PIC (PICC)
			Raspberry Pi (GCC)
C++	Textual	Object Oriented	Lego EV3 and NXT (C4Ev3)
			Raspberry Pi (G++)
JAVA	Textual	Object Oriented	Lego EV3 and NXT (Lejos)
			Raspberry Pi (OpenJDK)
Python	Textual	Multiparadigm	Lego EV3 (Ev3dev)
			LegoNXT (PyNXC)
Scratch	Visual	Imperative/Event-Driven	Lego NXT (Enchanting)
			Raspberry Pi
LabView	Graphical	Data Flow	Lego EV3
			Raspberry Pi
Lego Mindstorm	Graphical	Data Flow	Lego EV3 and NXT
Open Roberta	Graphical	Imperative/Event-Driven	Lego EV3

Fuente: [1, 14, 20].

Despite the numerous programming languages available on the market, there are still gaps in transitioning from a visual language to a visual/graphical programming language that students must work through gradually.

Methodology

The construction of a new visual programming environment and its corresponding components was proposed to create an environment that enables a progressive transition between visual and textual programming languages. A cascade model was used to construct this new environment [23]. The construction phases defined for this development environment are as follows:

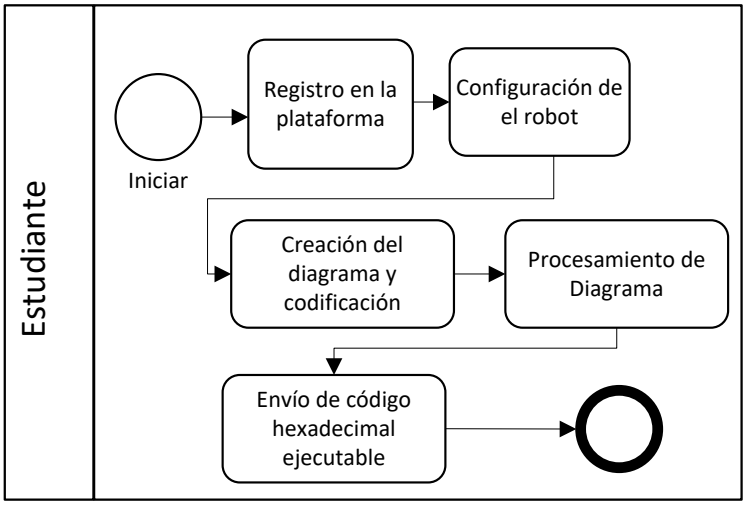
Definition of the Functional Behavior of the Platform

Students will be able to perform the following online operations through a web browser:

- Register on the platform using a username and password.
- Manage robotic devices on a platform where each robot is assigned a unique robot code and access key.
- Synchronize robotic devices with the programming environment.
- Execute remote robot tasks.
- Build programming routines.
- Compile routines into executable machine code.
- Send this executable code to the robotic devices.

El flujo de estas actividades se encuentra en la Figura 1.

Figure. 1. Robotics Platform Student Workflow.

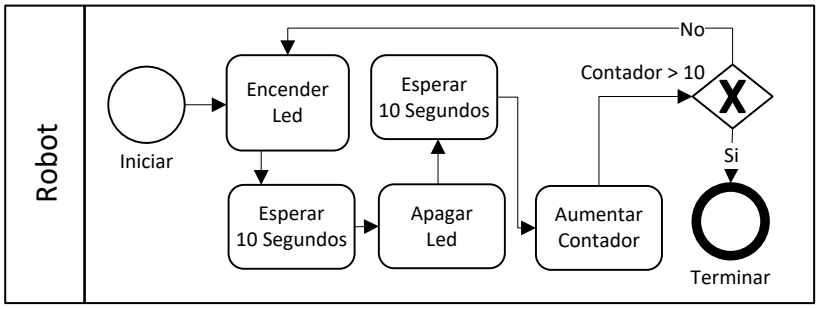


Source: Prepared by the Authors

Business Process Model and Notation (BPMN) [24] were used to represent the flow of activities executed by the robot. The routine consisted of a series of rounded boxes representing activities, pools, lanes, and control structures, such as logic gates, start and finish events. The boxes were interconnected using arrows representing the corresponding execution order (See Figure 2).

Since the notation does not offer a means to represent the "For" or "While" loops from imperative programming, if required, logical gates that "given a condition return to the desired start task" must be used. As opposed to a custom notation, this standardized notation makes it easier for users to familiarize themselves with a notation that may allow them to represent and provide solutions to industrial processes in the future. Additionally, develop procedural and systematic thinking.

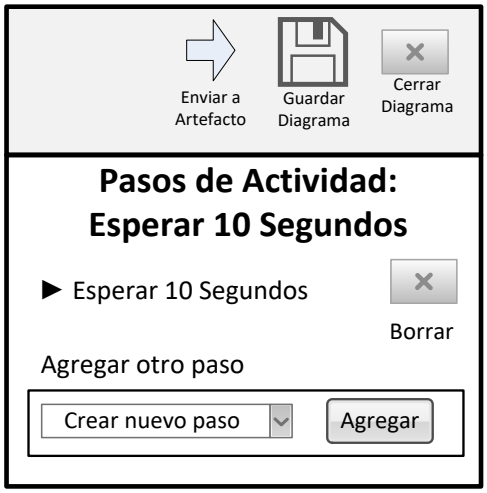
Figure 2. LED Flashing Diagram



Source: Prepared by the Authors

Cada actividad, representada mediante cajas redondeados, es integrada por una serie ordenada de pasos (Ver Figura 3). Un paso representa un fragmento de código textual ejecutable que puede ser interpretado como una acción por el robot. Los pasos pueden ser funcionalidades precargadas o fragmentos de código que el usuario hubiera desarrollado con anterioridad.

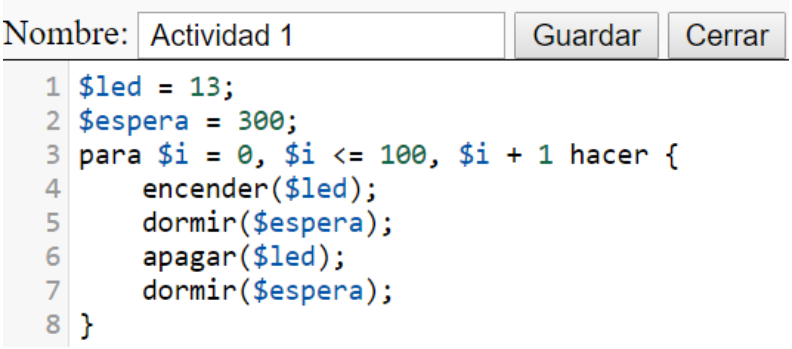
Figure 3. Activity Steps



Source: Prepared by the Authors

A textual programming language with syntax similar to the PHP language was developed for step coding. This language provides control statements for loops (while, for) and conditionals (if, elseif, else) and supports user-defined functions and the declaration of data storage variables. (See Figure 4 below).

Figura 4. Ejemplo de código fuente en un paso.



Source: Prepared by the Authors

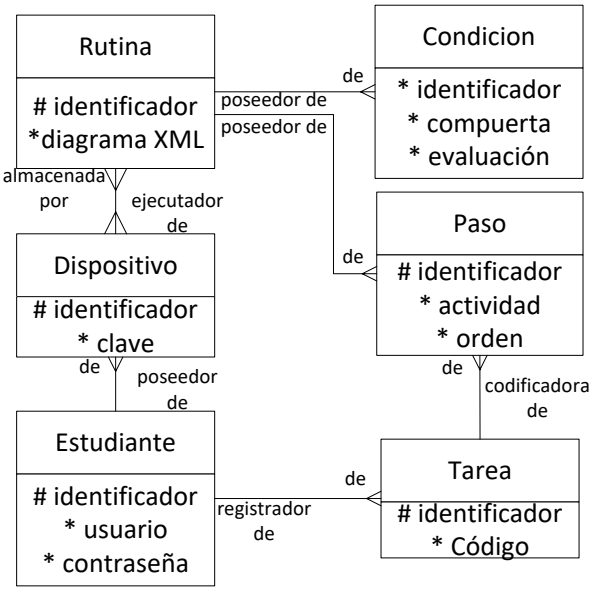
After creating the routine and coding the steps, students can send the routines to the robots for execution to confirm whether the desired objective is being reached. The routine was compiled and transformed into machine code during the transmitting procedure.

Integrated Development Visual Environment Architecture

The development environment was built in four layers using the HTTP protocol. These layers are as follows:

- Front-End: Built using HTML, Javascript, and CSS, the Front-end layer manages student interaction with the development environment. This layer includes the GUI application.
- Back-End: This layer manages web service requests and task execution. It was built following a model-view-controller (MVC) pattern [25].
- ORM: This is a link that maps objects to existing relationships within a database.
- Database: A relational database for storing and querying data. This layer also manages student information, robotics devices, and routines. (See Figure 5 below).

Figure 5. Development Environment Data Model



Source: Prepared by the Authors

Device selection for Integration with a Visual Development Environment

Three robotics platforms (custom-built based on microcontrollers, a development card, and a commercial educational robotics kit) were connected and used in the visual development environment. These platforms were chosen based on local market availability. The robotics device requirements for project implementation are as follows:

Code: The device must be programmable. Additionally, this programming must be conducted through a wireless network, a serial connection, USB, or other mechanisms.

Storage and Processing: The device must support memory storage and run a virtual machine capable of interpreting the code sent by the students. The processor used by the robotics device and its RAM, ROM or FLASH, and EEPROM memory is assessed.

Sensor Integration. The possibility of modifying sensors using analog and digital inputs to connect equipment that can enhance training, such as light source detectors or infrared/ultrasound obstacle detectors. The number of analog and digital inputs is also assessed.

New Devices. The possibility to connect other devices using I2C or SPI communication buses.

Costs. Available in the local Colombian market at a commercial price under $500.

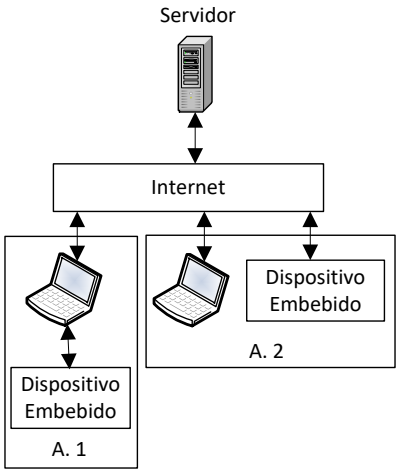
Robot Communication – Development Environment

To support the data transmission between the robot and the development environment, the communication capabilities of the robotic devices must be considered. For this reason, the following mechanisms were defined through REST service requests (See Figure 6):

Intermediate Device Connection. Since the robotics device lacks the ability to connect directly to the server where the development environment is deployed, an intermediary device with an internet connection is required for data transfers.

Direct Connection. The robotics device includes mechanisms to establish a direct connection to the server through its network interface.

Figura 6. Diagrama de comunicación robot – servidor remoto



Source: Prepared by the Authors

Preprocessing and Code Compilation

A preprocessing process is required to associate the XML file containing the routine diagram, step information, and the various gates with the routine compilation process. The preprocessing process follows the following phases (See Figure 7 below):

Create an association array. Database reading and XML diagram representation as an association array.

Array Representation. Two association arrays are generated in which the activity information and the existing logic gates are included in the routine diagram. The unique identifier of the element is used as the index of the association array.

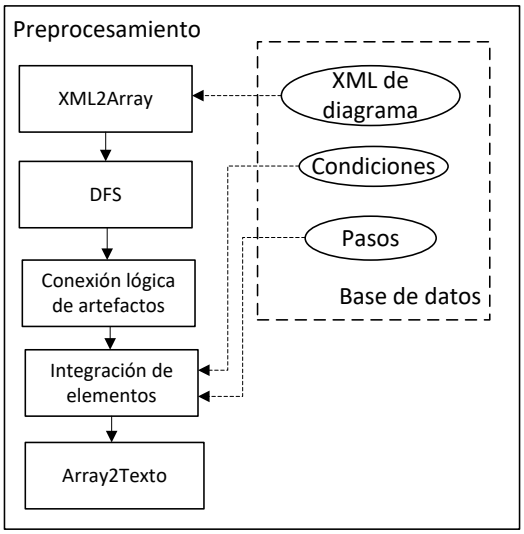
Graph Path. The depth-first search (DFS) algorithm is used to traverse the entire BPM diagram. The scan will determine code order in the flat text file.

Element Recognition. The elements extracted from the diagram are navigated, and their links are found using association array indexes.

Condition Extractions. Information about the conditionals used by the logic gates and activities is retrieved from the database.

Finally, a single-intermediate code file is built that serves as the compiler input.

Figure 7. Routine Preprocessing Process



Source: Prepared by the Authors

In the compilation process, the statements and instructions are translated into the operational code used by the virtual machine. The compiler includes the following:

A constant definition table. Used in the translation of instructions to a numeric code.

A lexical analyzer. Tokenizes the source code received as an argument characterized by language elements.

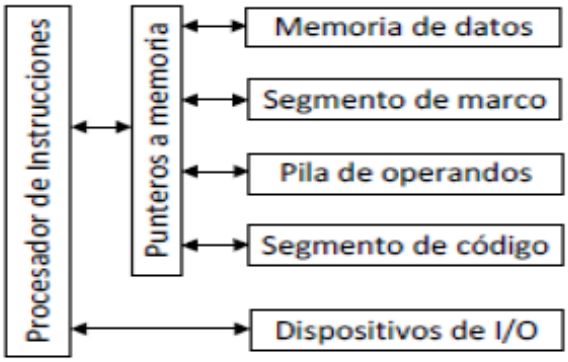
A syntactic analyzer. Checks and analyzes the lexical analyzer output for semantics. The machine code defines the syntactic language and the construction rules used.

A symbol table. It stores the names of the variables and functions and the scope of their declaration.

The robot downloads the resulting code through REST. The robot processes and executes the machine code through a virtual machine.

Virtual Machine

Figure 8. Virtual Machine Architecture



Source: Prepared by the Authors

The virtual machine runs on a robotics device based on Harvard architecture. That is, data and instructions are located in separate memory segments. It is also composed of a code segment, data memory, an operator stack, input and output devices, a framework segment, an instruction processor, and a series of pointers. The virtual memory handles the execution of the executable hexadecimal code generated at the compilation stage. Using a virtual machine facilitates running the same executable code on different robotic devices.

Validation and Verification

All produced components are subjected to a black box validation to find and correct any potential programming flaws. In the usability tests, five users performed a set of predefined tasks while their mouse actions were recorded and timed. This stage concludes with a survey that determines usability ratings from the user's viewpoint..

Results and Discussion

Selection of Alternative Robotics Platforms

For this study, a different robotics device was selected from the three specified categories: educational robotics kit, single-board computer, and custom design board (See Table 2).

Table 2: Single-Board Computer Feature Comparison

	Raspberry PI 2	Arduino Uno R3	BeagleBone Black
Clock speed	Quad-core ARM at 900 MHz	20 MHz	1 GHz ARM Crotes A8
Flash	32 GB SD Card	32 KB	4 GB on-board Flash Storage
EEPROM	0 B	1024 B	0 B
RAM	1024 MB	2048 B	512 MB
I/O Ports	26	19	63
PWM Channels	1	6	8
Analog Channels	0	6	7
Analog Port Resolution	Not Applicable	10-bit	12-bit

Communications	Ethernet, USB, I2C, SPI	SPI, Serial USART, I2C	UART, PWM, LCD, GPMC, MMC1, SPI, I2C, CAN BUS
Price	35 USD	23 USD	50 USD
USB Module	Integrated	External	Integrated

Source: [1]

The characteristics of the Arduino UNO R3, Raspberry PI 2, and BeagleBone Black were evaluated to select a single-board computer. After evaluating the different options, the Raspberry PI was selected due to its sufficient computing capabilities, large number of input and output ports, possible direct connection to the internet, internal storage, large community, and worldwide availability. The PIC 18F4550 microcontroller built before the project was used on the development card.

The Lego Minstorm EV3 and the Bioloid Premium Kit were evaluated for the educational robotics kit. Here, Lego Minstorm EV3 was selected because the Bioloid Premium Kit was too expensive.

Platform Implementation

LAMP stack (Linux, Apache, MySQL, PHP) was used to build the development environment, and a framework based on the MVC architecture and the "Don't Repeat Yourself" (DRY) philosophy was created for it. The preprocessor and the compiler were developed as separate libraries from the framework to facilitate their reuse. Virtual machines were built and adapted to each robotic platform's capabilities and communication interfaces. The CCS C compiler software was used for the microcontroller-based custom development card, and a communication interface was built in Microsoft Windows using Python (See Figure 9), which serves as a communications bridge between the microcontroller and the remote system. The virtual machine and the communication interface for the Lego EV3 and Raspberry PI devices were built on Python..

Figure 9. Windows Communication Interface for Microcontroller Communications

Fuente: Elaboración propia

User Experience in the Programming Interface

Seeking to evaluate possible existing weaknesses, a population of 105 IT professionals were asked to conduct four tasks on the platform: (1) user registration, (2) adding a robotics device, (3) creating a routine, and (4) adding two steps to an activity. Activity times were recorded, and participants were asked three post-execution questions. Each task took an average of (1) 2 min 57 seconds, (2) 1 min 31 seconds, (3) 36 seconds, and (4) 3 min 11 seconds, respectively (See Table 3)..

Table 3. Survey Results

Question	Yes	No	Maybe
Is this system easy to use?	97	2	6
Is this system easy to learn?	94	5	6
Would you recommend this application to another user?	95	2	8

Source: Prepared by the Authors

Conclusions

For decades, it has been evident that convenient mechanisms must be sought to integrate technology into educational environments to facilitate and provide greater flexibility to learning processes, given the characteristics of today's students. The application of educational robotics, framed within constructivist practices and classroom didactics, helps students solve problems by applying their knowledge through dynamic, participatory, and interactive processes.

Several robotics kits are available in the market, including custom programming environments. These kits are usually marketed to countries with advanced economies. However, the Colombian educational model cannot accept them due to their high expenses. Additionally, their methods do not entirely meet the needs identified by the Colombian students.

This study summarizes some of the advantages of educational robotics within this context. It also discusses some programming language characteristics used in educational robotics. Furthermore, a programming environment built on cloud computing and intended for robotics application educational settings is presented. This environment bridges the gaps by allowing students to move easily from visual programming to textual programming using multiple robotics devices in the same development environment. After evaluating the major paradigms in the construction of computational environment representations, it was decided to use the occupation cell mapping technique for the mathematical models of the robot, thus validating the potential of intuitive methods in controller design.

The system was tested in indoor environments such as hallways, offices, laboratories, and classrooms. The models obtained were adjusted both to the morphological characteristics of the test environments and their characteristic metrics.

Referencias bibliográficas

1. C. A. Ruiz, "Ambiente de desarrollo integrado de programación híbrida visual/textual integrado con artefactos robóticos para el aprendizaje la enseñanza de áreas STEM a través de programación de computadores". Tesis de Maestría en Ingeniería, Universidad Nacional de Colombia, 2019.
2. G. A. Wiggins y A. Smaill, Musical Knowledge: what can Artificial Intelligence bring to the musicians? 2000.
3. J. Prentzas, "Artificial intelligence methods in early childhood education", en Artificial Intelligence, Evolutionary Computing and Metaheuristics, *Springer, 2013*, pp. 169–199. DOI: https://doi.org/10.1007/978-3-642-29694-9_8
4. Y.-W. Cheng, P.-C. Sun, y N.-S. Chen, "The essential applications of educational robot: Requirement analysis from the perspectives of experts, researchers and instructors", *Comput. Educ.*, vol. 126, pp. 399-416, nov. 2018. DOI: https://doi.org/10.1016/j.compedu.2018.07.020.

5. C. Angeli y N. Valanides, "Developing young children's computational thinking with educational robotics: An interaction effect between gender and scaffolding strategy", *Comput. Hum. Behav.*, vol. 105, p. 105954, abr. 2020. DOI: https://doi.org/10.1016/j.chb.2019.03.018.

6. M. Prensky, "Digital natives, digital immigrants part 1", *Horiz.*, vol. 9, no 5, pp. 1–6, 2001. DOI: https://doi.org/10.1108/10748120110424816.

7. M. Prensky y B. D. Berry, "Do they really think differently", *Horiz.*, vol. 9, no 6, pp. 1–9, 2001. DOI: https://doi.org/10.1108/10748120110424843

8. B. De Wever, P. Mechant, P. Veevaete, y L. Hauttekeete, "E-Learning 2.0: social software for educational use", *Multimedia Workshops 2007 ISMW'07. Ninth IEEE International Symposium on*, 2007, pp. 511–516. DOI: https://doi.org/10.1109/ISM.Workshops.2007.91

9. F. Fovet, "Impact of the use of Facebook amongst students of high school age with Social, Emotional and Behavioural Difficulties (SEBD)", *2009 39th IEEE Frontiers in Education Conference, 2009*, pp. 1–6. DOI: https://doi.org/10.1109/FIE.2009.5350786

10. Papert, Mindstorms: Children, Computers, and Powerful Ideas, Edición: New Ed. New York: The Perseus Books Group, 1993. DOI: https://doi.org/10.1145/1045071.1045074

11. M. Ruzzenente, M. Koo, K. Nielsen, L. Grespan, y P. Fiorini, "A review of robotics kits for tertiary education", *Proceedings of International Workshop Teaching Robotics Teaching with Robotics: Integrating Robotics in School Curriculum*, 2012, pp. 153–162. DOI: https://doi.org/10.1007/s10798-012-9210-z

12. A. Larcher., F. Turri, J. Collins, I. Derweesh, A. Volpe, J. Kaouk, R. Koon, "Definition of a structured training curriculum for robot-assisted partial nephrectomy: A Delphi-consensus study from the ERUS Educational Board", *Eur. Urol*. Suppl., vol. 17, n.o 2, pp. e678-e682, mar. 2018. DOI: https://doi.org/10.1016/S1569-9056(18)31310-1.

13. N. Reich-Stiebert, F. Eyssel, y C. Hohnemann, "Involve the user! Changing attitudes toward robots by user participation in a robot prototyping process", Comput. Hum. Behav., vol. 91, pp. 290-296, feb. 2019. DOI: https://doi.org/10.1016/j.chb.2018.09.041.

14. B. S. Blais, "Using Python to Program LEGO Mindstorms Robots: The PyNXC Project", *Python Pap.*, 2010.

15. S. Seiler, R. Sell, y D. Ptasik, "Embedded System and Robotic Education in a Blended Learning Environment Utilizing Remote and Virtual Labs in the Cloud, Accompanied by 'Robotic HomeLab Kit'", *Int. J. Emerg. Technol. Learn. IJET,* vol. 7, no 4, dic. 2012. DOI: https://doi.org/10.3991/ijet.v7i4.2245

16. X. Xie, C.-C. Huang, Y. Chen, y F. Hao, "Intelligent robots and rural children", *Child. Youth Serv.* Rev., vol. 100, pp. 283-290, may 2019. DOI: https://doi.org/10.1016/j.childyouth.2019.03.004.

17. C. Fernández-Llamas, M. A. Conde, F. J. Rodríguez-Lera, F. J. Rodríguez-Sedano, y F. García, "May I teach you? Students' behavior when lectured by robotic vs. human teachers", *Comput. Hum. Behav.*, vol. 80, pp. 460-469, mar. 2018. DOI: https://doi.org/10.1016/j.chb.2017.09.028.

18. M. Campusano, J. Fabry, y A. Bergel, "Live programming in practice: A controlled experiment on state machines for robotic behaviors", *Inf. Softw. Technol.*, vol. 108, pp. 99-114, abr. 2019. DOI: https://doi.org/10.1016/j.infsof.2018.12.008.

19. B. Jost, M. Ketterl, R. Budde, y T. Leimbach, "Graphical Programming Environments for Educational Robots: Open Roberta - Yet Another One?", *2014 IEEE International Symposium on Multimedia*, 2014, pp. 381–386. DOI: https://doi.org/10.1109/ISM.2014.24

20. J. Maloney, M. Resnick, N. Rusk, B. Silverman, y E. Eastmond, “The Scratch Programming Language and Environment”, *ACM Trans. Comput. Educ.*, vol. 10, no 4, pp. 1–15, nov. 2010. DOI: https://doi.org/10.1145/1868358.1868363.

21. J. M. Gomez-de-Gabriel, A. Mandow, J. Fernandez-Lozano, y A. J. Garcia-Cerezo, “Using LEGO NXT Mobile Robots With LabVIEW for Undergraduate Courses on Mechatronics”, *IEEE Trans. Educ.*, vol. 54, no 1, pp. 41–47, feb. 2011. DOI: https://doi.org/10.1109/TE.2010.2043359

22. R. U. Pedersen, J. Nørbjerg, y M. P. Scholz, “Embedded Programming Education with Lego Mindstorms NXT Using Java (leJOS), Eclipse (XPairtise), and Python (PyMite)”, *Proceedings of the 2009 Workshop on Embedded Systems Education, New York, NY, USA*, 2009, pp. 50–55. DOI: https://doi.org/10.1145/1719010.1719019

23. W. W. Royce y others, “Managing the development of large software systems”, en proceedings of IEEE WESCON, 1970, vol. 26, pp. 1–9. DOI: https://doi.org/10.4236/cus.2018.62015 1,006

24. S. A. White, “Introduction to BPMN”, *IBM Coop.*, vol. 2, no 0, p. 0, 2004.

25. G. E. Krasner y S. T. Pope, “A Cookbook for Using the Model-view Controller User Interface Paradigm in Smalltalk-80”, *J Object Oriented Program*, vol. 1, no 3, pp. 26–49, ago. 1988.