Development of a Hardware Educational Tool for Teaching Computational Thinking with Scratch®

Ing. Martha Lucia Cano, Pontificia Universidad Javeriana, Bogotá, Columbia

Professor at the Faculty of Engineering at Pontificia Universidad Javeriana in Bogotá and current Ph.D student in Engineering Education at Rowan University. Received the B.S. degree in electronics engineering from Pontificia Universidad Javeriana, Bogotá, Colombia, and the M.S. degree in critical systems and networks from Université Paul Sabatier, Toulouse, France, in 2006. She has worked as professor at Pontificia Universidad Javeriana since 2006.

Alejandro Castro Martinez
Prof. Jairo Alberto Hurtado JAH, Pontificia Universidad Javeriana, Bogotá, Columbia

Associate professor at Pontificia Universidad Javeriana Bogota, Colombia, at Electronics Department. He was Chair of Electronics Engineering Program and he has been working in different projects to get a better process learning in his students

Eduardo Rodriguez Mejia, Pontificia Universidad Javeriana, Bogotá, Columbia

Hi, my name is Eduardo, I am a Rover Scout and professional Electronic Engineer with a Masters degree in Electronic Engineer. I am pursuing my PhD in Engineering with a Concentration in Engineering Education within the ExEEd department. I am interested in new teaching methodologies that involve a hands on experience that let students see, smell, and feel the things that they are learning about.

Development of a Hardware Educational Tool for Teaching Computational Thinking with Scratch®.

Abstract. In “The Future of Jobs Report 2020”, the world economic forum (WEF) built a list of ten skills that will be most required in jobs by 2025, one of them being “technology design and programming”. In response to the above, in recent years, many projects have been launched to increase programming knowledge for different audiences and in different parts of the world. One of these projects was developed through a collaboration between a university in Colombia and a worldwide company from the technology industry. The purpose of the project was to teach programming skills to the underserved population in Colombia. For this, different programming languages were taught, one of them was the visual programming language Scratch®, in conjunction with a hardware platform to help students understand the interaction between hardware and software.

The hardware platform provided for the project had some limitations, like not working with the last version of Scratch®; research for other platforms showed that few existed, some were too expensive, or were no longer available on the market. These problems led the authors of this article to design and develop a new platform that interacts with the latest version of Scratch® and included more sensors and actuators that help students to develop their computational thinking.

For the conception, design, and implementation of the hardware platform, we followed a top-down methodology, starting from a high-level idea, and increasing the detail of each module until reaching a minimum viable product that was tested following a predefined test protocol. We used Finite States Machine as a theory of computation to model the behavior of the platform and to develop the embedded code. To secure the operation of the platform, we added new blocks to the source code of the latest version of Scratch®, this was a difficult challenge because it implied modifying complex programming sources, for this, a structured design strategy was followed. Finally, a function-oriented design was used to develop a software bridge in Python to assure communication between the above two components.

This project resulted in three main components being developed and fully tested. First a shield for an Arduino Uno board, with sensors and actuators compatible with Scratch’s functions, second a standalone modified version of Scratch, that can run from a free access webpage or can be downloaded to run locally, and finally a software bridge that allows the communication between the hardware and the software. The resulting solution works properly with Scratch® codes that use the built-in blocks as well as the ones developed in the project.

The project attained its objectives and made a contribution to the development of educational tools for teaching computational thinking. The entire solution will be used in summer camp training to teach programming skills to a young audience in Colombia. New projects have derived from the results, like the development of instructional guides for practices that use the solution, and the development of enhanced versions that can reduce the costs of production and introduce wireless communication.

I. Introduction

In “The Future of Jobs Report 2020” [1], the world economic forum (WEF) built a list of ten skills that will be most required in jobs by 2025, one of them being “technology design and programming”. Having technological skills is becoming crucial to find better job opportunities in different domains, but that poses a problem for people that cannot access higher education and therefore cannot access formal technological training. In response to the above in 2019 a project was launched by a worldwide company from the technology industry, to train young people of disadvantaged backgrounds (e.g., low income and no access to higher education) with the skills needed by industry, such as problem-solving, computational thinking and programming [2]. In Colombia, the project was developed through a collaboration with the *Pontificia Universidad Javeriana* in Bogota. Due to the COVID-19 pandemic, the project was delayed to the second semester of 2020 and was adjusted to be virtual and include participants all around the country, which in the end allowed it to reach more underprivileged populations and increase the overall number of participants. The first cohort was successful, prompting a second cohort to open within two months of the first one. A third cohort was offered the next year, certifying over 1,000 participants in total over two years. The targeted population was young women and men, from 15 to 24 years of low-income households and with no higher education.

The project included teaching computational thinking, different programming languages, and programming Arduino boards. It also included a closed-source hardware platform that interacted with Scratch. However, the hardware platform had limitations that posed problems during the training program. First, it did not work with the last Web version of Scratch®, which implied working with the limited and less appealing desktop version. Another limitation was the stock and availability of the platform. Because it was closed-source, there was no possibility that they could be fabricated by a third party. A market search was done to look for other platforms that connected with Scratch, but few existed, some were too expensive, and others were no longer available for sale. These problems led the authors of this article to design and develop a new platform that interacted with the latest version of Scratch® that could be used for future training programs. We included actuators, which were not included in the original platform, and more sensors to improve the students’ learning of hardware-software interaction.

In this article, we present the deployment of the hardware-software solution that allows the interaction between Scratch® and an embedded system with electronic sensors and actuators. In the article we present background about computational thinking and Scratch, then we explain the general scheme of the solution, the design and implementation of the embedded system, a brief description of the software interface implemented to bridge the embedded firmware and Scratch, and the adaptation of Scratch’s source code. Finally, we present the results of the evaluation of the overall system.

II. Background

A. Computational thinking

Computational thinking (CT) is widely related to computer scientists and engineers but is a critical skill that can help everybody, even in daily life tasks. It “involves solving problems, designing systems, and understanding human behavior” [3], all part of daily routine like trying to figure out how to dress up or what to do when the water pipe in the bathroom breaks. These

examples involve identifying the problem and thinking of the best possible solution, breaking the original problem into smaller tasks, reformulating the problem in a way it reminds us of something we are familiar with, planning, learning, and developing an action plan in the presence of uncertainty [3].

Wing [4] comments that one of the most important aspects of CT is the concept of abstraction to define patterns previously identified, generalize, and do a parametrization. The capacity to abstract is precisely what will allow someone that develops CT to apply it to every aspect of life and across all professions. Abstraction is the core element that helps identify risks, opportunities, viable solutions, steps, and critical thinking in general [4]. Because of the importance of CT, there has been a growing interest in teaching it through all levels of education, from K-12 to undergraduate level, with sponsors that include industry, government, academia, and industry and academia partnerships.

B. Scratch® and Computational Thinking

Scratch®, aimed at beginner programmers, is a media-rich, free, and open-source programming language that uses blocks to create code [5]. Scratch was developed initially by the Massachusetts Institute of Technology (MIT) and it is hosted in the MIT web domain. Scratch's blocks interconnect with each other and allow users to create graphics, animations, stories, and games with a basic point-and-click interface. Additionally, the platform has numerous media libraries composed of images, sprites, and sounds that can be edited and used as part of any program. It also allows users to import their images and sprites, as well as record or import new sounds [5].

The platform allows for the integration of programming blocks or extensions developed by third parties. This option extends the scope of the editor and makes it possible to work with external hardware elements like makey-makey and micro:bits, information sources like Google Translate, and advanced functionalities such as video sensing and music instruments [6].

The interface provided by the Scratch® software is a suitable and easy environment to learn CT concepts, good programming practices, and shifts in perspectives to interact and go through life [7]. Brennan & Resnick [7] went further to propose a new approach to study and assess CT, and it is widely used as a framework when learning CT through Scratch [8]. They identified seven concepts (sequences, loops, parallelism, events, conditionals, operators, and data), four practices (being incremental and iterative, testing and debugging, reusing and remixing, and abstracting and modularizing), and three shifts in participants' perspectives (expressing with the aid of digital technology, connecting with other programmers, and questioning the thing around and make sense of the world) [7]. All of them can be learned using the Scratch platform and scaffolding each of the concepts that eventually will allow learners to migrate to any other programming language.

C. Hardware for computational thinking with Scratch

Arduino hardware is one of the main platforms that has been used to interface with Scratch software; tools like Scratch4Arduino and mBlock allow programming the Arduino platform using Scratch blocks [9]. Other hardware tools, like makey-makey and micro:bits, allow for bidirectional interactions in which individuals can program actions to take place in the hardware and/or input commands to the software. These tools have a price range from 30 to 100 dollars and there is a trend in pushing forward with the technological advances to develop tools that will be

available for less than 10 dollars to make them more accessible to a larger populations around the world [5]

These hardware tools allow for the physical and digital worlds to work together, helping with physical demonstrations of an algorithm, parallel design effort, and validation for software and hardware solutions [10]. Kafai & Burke [5] describe these interactions as a tangible dimension for learning and understanding CT. They also state that, “The mind-and-hand merger of the digital and physical repositions youth as active creators rather than just consumers of knowledge.” [5, p 109] as a response to the physical side of computing and digital technology that we use in our everyday life.

III. General scheme of the deployed solution

The hardware-software solution, presented in Figure 1, is composed of two hardware, two software, and one firmware component. The hardware components are an Arduino Uno and a hardware platform that was designed to fit as an Arduino shield. The hardware platform contains sensors, actuators, and signal conditioning circuits. The firmware component is the code developed to manage the sensors and actuators and communicate with the software bridge. The Arduino, and the firmware are referred to as the Embedded System; its design and implementation are explained in Section IV.

The software is composed of an adapted version of Scratch and a Python software bridge that interprets the communication between the Embedded System and Scratch. The adapted version of Scratch is required to communicate with embedded external systems. MIT legally allows third parties to deploy their Scratch version with modifications, if the Scratch name is preserved. The modified version that we developed includes Scratch’s standard blocks, third-party extensions on the MIT version, and our extension with the blocks that interact with the sensors and actuators of the embedded system.

To create a reliable way to transfer data from Scratch to the embedded system and vice versa, the use of an intermediate Python Bridge is required.

The adapted version of Scratch and the Python bridge will be explained in Section V.

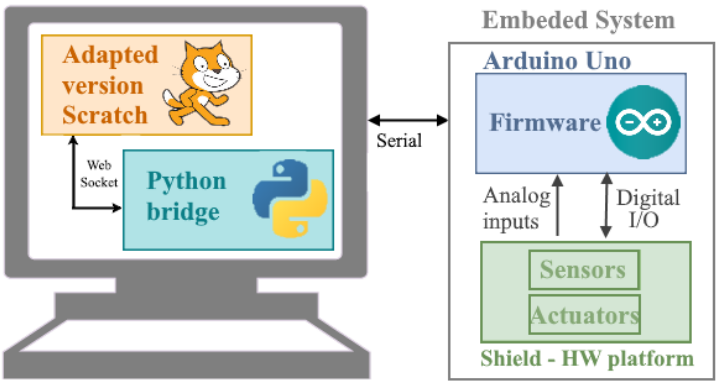


Figure 1: Block diagram of the hardware-software solution.

IV. Design and implementation of the Embedded System

A. Shield design and implementation

The first step in the design process was the selection of the sensors and actuators that were going to be included in the Arduino shield. Table 1 lists the chosen components and presents a brief explanation of the blocks included in the Scratch extension for each of them.

Table 1: Components used in the Arduino shield.

Component	Blocks included in Scratch
Buzzer	One block to turn it on/off
Vibrator motor	One block to turn it on/off
RGB led	Three blocks to turn on/off each led (Red, Green, Blue). One block with a scrolling list to select a specific color to be displayed
DC motor	One block to turn it on/off. One block to define the turning direction to the right. One block to define the turning direction to the left.
Joystick	One block that returns the actual values of X and Y positions. One block to determine if the joystick is pressed
Potentiometer	One block that returns its value.
Microphone	One block that returns the instantaneous value.
Press Button	One block to determine if the button is pressed
LCD screen	One block to write any ASCII character inside the display One block to clean what has been written on the LCD screen

The next step was to identify which pins of the Arduino the components were going to be connected. Figure 2 shows the identified location for each component.

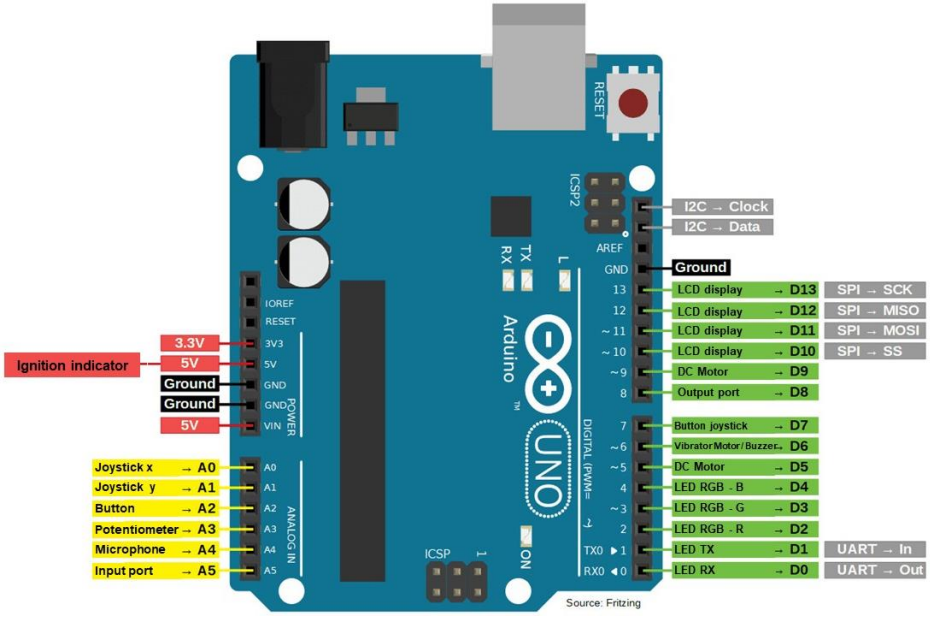
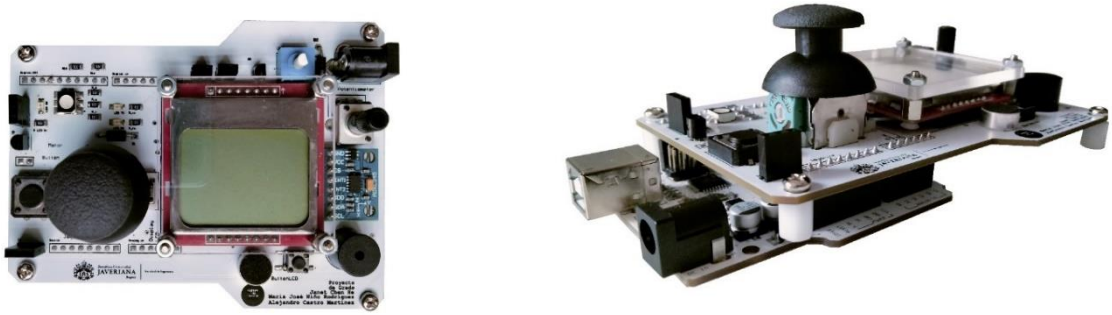


Figure 2: Connections from the shield to the Arduino UNO board.

With the connections identified, we proceeded to carry out the design of the shield's printed circuit board (PCB). The PCB was designed considering aspects such as size, cost, and resistance to improper handling.

Figure 3 (a) presents the final shield implemented and figure 3(b) shows it coupled with the Arduino UNO board.



(a) (b)

Figure 3: (a) Shield implemented (b) shield coupled with an Arduino UNO

B. *Arduino Firmware*

The firmware that controls the sensors and actuators and communicates with the PC was designed using a Finite State Machine (FSM). Figure 4 presents the state diagram of the FSM

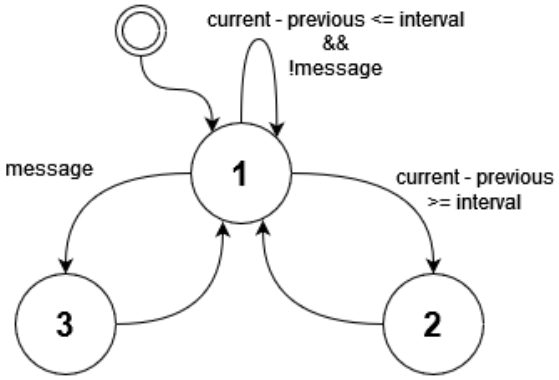


Figure 4: FSM of the implemented firmware

State 1 is a waiting state, in which the machine will stay until there is a timeout equal to "interval" or when it receives a message from the PC. If the timeout occurs, the FSM goes to state 2 in which the data from sensors is acquired and sent to the PC where Scratch is running, using a Firmata serial protocol. After completing the transmission, it returns to state 1. State 3 is executed when the microcontroller receives a message from the PC. In this state, control actions are sent to the actuators according to the instructions received in the message. The standard input output Arduino's libraries were used to read the sensors' values and control the actuators. The complete firmware was developed in Arduino's Integrated development environment (IDE).

V. Software adaptation and development

A. Scratch's source code adaptation

We first defined the blocks needed to interact with the sensors and actuators, then each block was coded and included in the base version of Scratch available in a public GitHub repository. This process is called coding an extension.

The process to code a new block inside Scratch includes the modification of two separate projects, Scratch-vm [11] and scratch-gui [12]. The following steps describe how to create it:

- Find the src/extensions folder inside the scratch-vm project.
- Inside that folder, create a new folder with a comprehensive name, such as *scratch3_newblocks*. *newblocks* can be replaced with another word more consistent with the blocks' function.
- Inside the new folder, create a file named *index.js*. It's important to not change the file name. This file will contain the creation and definition of the functions that will be executed by the new blocks that will be added to Scratch.
- The structure of this file must follow the structure proposed by Scratch®.

In the getInfo function, the following terms are used:

- id: A unique internal name of the extension. It cannot match any other extension name within the deployment.
- name: This will be the name that will appear when selecting the extension.
- blocks: This contains the objects/blocks that will be created in the extension.
- opcode: This is the method that will be called when executing the block.
- blockType: Describes the type of block that is created.
- text: Contains the description of the block, that is, what appears in the block within the web page.
- arguments: This is an object that contains the fields for the arguments defined in the text.
- menus (optional): This field is used for the definition of drop-down menus for arguments of the created blocks.

For more information regarding the creation of extensions for Scratch, the scratch-vm repository has a file explaining some optional fields within the previous definition [13].

After creating the new extension, a reference to it needs to be added to the scratch_gui project [12] following the next steps:

- Create a folder with the same name as the extension in the path *src/lib/libraries/extensions*
- In the new folder save the images that will appear in the "Add extension" option in Scratch. One image for the cover and one for the thumbnail.
- Find the file *index.jsx* in the path *src/lib/libraries/extensions*. In this file, several lines of code must be added to include the reference to the images and the reference to the extension.

Finally, under the scratch-vm project [11], in the file named *extension-manager.js* in the path *src/extension-support/*, the reference to the new extension must be included in the code.

An adapted version of Scratch, with the same functionalities as the official version and the extension created in this work, was deployed. The deployment was made in Github pages, a website hosting service provided by Github free of charge. The adapted version can be found at the following link: https://hardware-for-education.github.io/Scratch_For_Education/

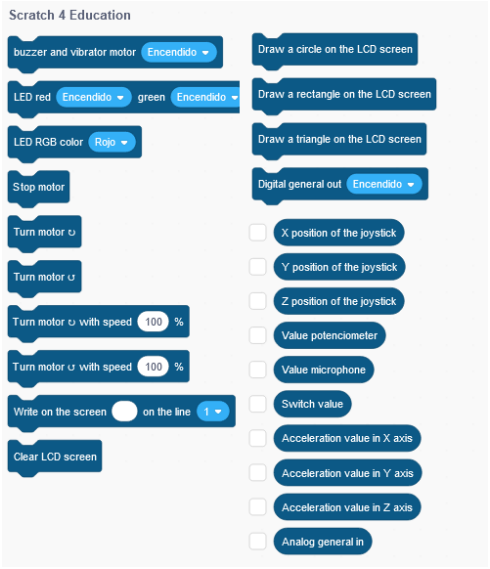


Figure 5: Blocks of the *Hardware para educación* extension.

After selecting the *Hardware para educación* extension, the blocks will appear in the main interface and may be included in a Scratch code. Figure 5 presents the resulting blocks for the extension.

B. Python Bridge

The program developed in Python was built from the base provided by Alan Yorinks in his s3-extend project. This is an open-source project licensed under the GNU Affero General Public License v3.0 which allows the code to be modified giving credit to its original creator. Yorinks' project aims to create a connection with external hardware through Scratch® [14].

The Python bridge uses several files to create a communication channel between Scratch and external hardware, each with a specific task to execute. Figure 6 shows a block diagram of the files and the connections with Scratch and the Arduino.

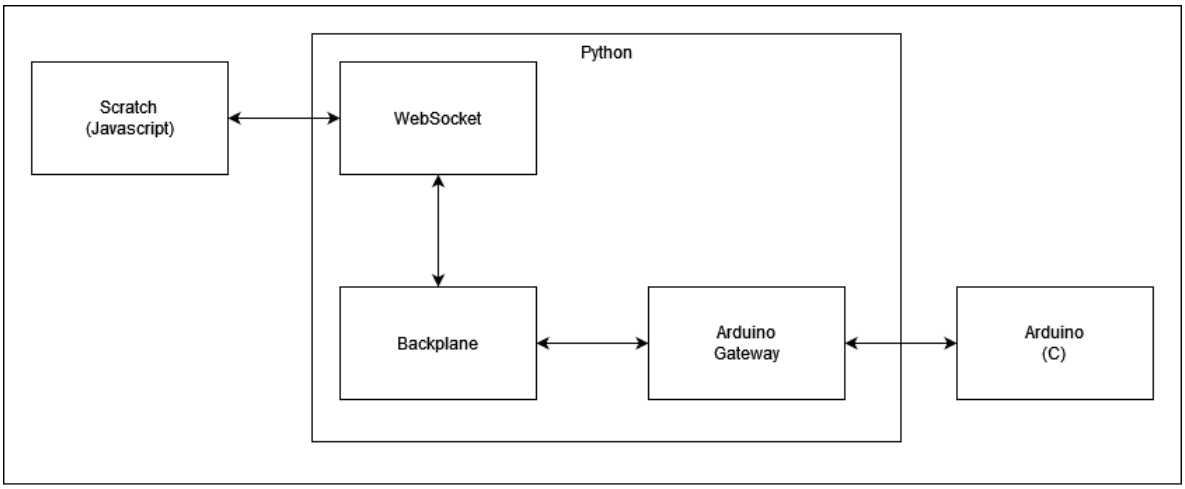


Figure 6: Structure of the Python bridge coded.

The block presented in the top left corner refers to the modified Scratch web editor, this program sends and receives data to the Python bridge. This connection is made using the file called WebSocket, which has the task is to create a communication socket between the modified Scratch web editor and the Backplane.

The Backplane is the file responsible for keeping the bridge open. This is done by verifying recurrently if the WebSocket and the Arduino Gateway programs are being executed. Additionally, this program is responsible for transmitting messages between WebSocket and the Arduino Gateway files.

The last program is the Arduino Gateway file. This is responsible for opening a communication channel between the Arduino and the computer and sending data to the Arduino concurrently.

VI. Results

This section presents the results for the different stages of the solution. A video about the general functioning can be viewed in the following link: https://youtu.be/zCNuMKegE2E

A. Software

Three different software developments were implemented for the adequate functioning of the hardware tool. First, an Arduino firmware to do the setup of the required peropherals. Second, a Python program to act as the bridge between the Arduino and the Scratch platform. Finally, the Scratch blocks for the interaction with the hardware tool elements (e.g., LEDs, LCD screen, buzzer). A GitHub repository was created to keep all three projects, as well as the corresponding documentation and read-me information for running the software.

A. Hardware

The Printed Circuit Board (PCB) was designed, fabricated, and checked to see that it functioned with an external power supply. All electronic components were checked to ensure signal integrity with test input signals. Next, a test was run with the Arduino platform to verify the integration of the electronic devices that act as input elements. Lastly, the output elements were tested with each one of the developed Scratch blocks.

A small batch of 10 circuit boards were fabricated at a production cost of $45 USD each.

B. Educational Tool

Pilot tests were made with the developed web version of Scratch 3.0. For these tests, 13 participants of different backgrounds were presented with the Board and were asked to interact with it.

Participants' ages were between 17 and 23 years. Of the participants, 8 out of 13 had worked with Scratch while the other 5 had no previous experience with the program. Only 6 out of the 13 participants knew Arduino previously. All of them were asked to interact with the embedded system, as well as go through a simple example with Scratch that used the blocks of the new extension. Foreseeing that some of the participants may not know how to use any of the programs, a protocol document was provided to guide them through the steps. After the interaction with the Educational Tool, participants were presented with a Likert scale survey to assess their experience on a scale from 1 to 5. The survey included a final section to leave comments about their experience and assessment. Table 2 presents the survey questions and the Likert scales.

The average values of the survey response indicate that the overall experience was satisfactory. The Scratch blocks obtained a score of 4.5 and the hardware system obtained 4.0. From the comments section, 8 out of the 13 the participant highlighted that the buzzer's volume knob was too difficult to turn, and that the buzzer did not function well.

Table 2: Components used in the Arduino shield.

Questions	Scale options
1. Age	Open question.
2. Bachelors degree	Engineering, Other, NA
3. Do you know or have previously worked with Scratch?	Yes, No.
4. Do you know or have previously worked with Arduino?	Yes, No
5. Do you understand the functionality of the Scratch blocks?	1 through 5 with 1 as completely in disagree and 5 completely agree.
6. Is was easy to interact with all the elements in the board while you do the exercise?	1 through 5 with 1 as completely in disagree and 5 completely agree.
7. Which of the board elements position would you change, based on the level of interaction while you went through the exercise?	Button, joystick, RGB LED, LCD screen, LCD screen button, microphone, buzzer, buzzer volume nob, potentiometer, none.
8. Comments, suggestions, and recommendations.	Open question.

VII. Conclusions and future work

The collaboration between a university in Colombia and a worldwide company for the technology industry led to the creation of a training program to teach digital skills to underserved population. The training program included the use of a hardware platform that allowed participants to interact with the codes they developed with Scratch. This platform presented limitations with its availability, which ended up hindering the participants' progress and limitations in its use, working only with an older version of Scratch. Additionally, the industry partner had over costs with the platform delivery, because all of them were imported. Despite the difficulties mentioned above, the program was a success, and end up being held for another year for two additional cohorts.

After the partnership ended up, the University wanted to continue the training with children and young adults. Therefore, professors at the institution started a project to design and build a new hardware platform, improving the limitations of the previous one, and with the possibility of being manufactured locally at a lower cost. The solution included the design and implementation of hardware, firmware, and software components. The final solution is functional, and the adapted Scratch version is available on a public website. Ten samples of the educational tool were manufactured and distributed among private university professors in Colombia. The overall comments were positive, and they praise the electronic design. One of the professors manifested his interest in using the tool for a summer camp training that would take place during the year 2023 with children between 6 and 9 years old. This training will serve as the first pilot test for users' validation regarding ergonomic design, ease of use, and effectiveness to facilitate the learning process of CT with Scratch. For this assessment, we will use the framework proposed by Brennan & Resnick [6].

Right now, this version of the hardware tool inspired two undergraduate engineering projects that will be developed during 2023. One of the projects will do a second version of the platform to reduce even more production costs. The second project will design a stand-alone platform that doesn't use the Arduino UNO board. The purpose is to create a version that is more user-friendly and designed for a younger population.

In addition to the development of the hardware, two projects were started in parallel to design guides for the training of both teachers and students who would take programming courses using these cards, for the development of computational thinking.

VIII. References

[1] World Economic Forum, "The Future of Jobs Report 2020," Oct. 2020. [Online]. Available: https://www3.weforum.org/docs/WEF_Future_of_Jobs_2020.pdf

[2] A. Benešová and J. Tupa, "Requirements for Education and Qualification of People in Industry 4.0," Procedia Manufacturing, vol. 11, pp. 2195-2202, 2017, doi: 10.1016/j.promfg.2017.07.366.

[3] J. M. Wing, "Computational thinking," Communications of the ACM, vol. 49, no. 3, pp. 33-35, 2006, doi: 10.1145/1118178.1118215.

[4] Wing, J. Research notebook: Computational thinking—What and why. The link magazine, 6, 20-23.

[5] Y. B. Kafai, Q. Burke, and M. Resnick, Connected Code: Why Children Need to Learn Programming, 1st ed. (The John D. and Catherine T. MacArthur Foundation Series on Digital Media and Learning Ser.). Cambridge: MIT Press, 2014.

[6] "Extension.". https://en.scratch-wiki.info/wiki/Extension

[7] Karen Brennan and M. Resnick, "New frameworks for studying and assessing the development of computational thinking," in 2012 annual meeting of the American educational research association, Vancouver, Canada, 2012, vol. 1, p. 25.

[8] L. Zhang and J. Nouri, "A systematic review of learning computational thinking through Scratch in K-9," Computers & Education, vol. 141, 2019, doi: 10.1016/j.compedu.2019.103607.

[9] A. Fidai, M. M. Capraro, and R. M. Capraro, "“Scratch”-ing computational thinking with Arduino: A meta-analysis," Thinking Skills and Creativity, vol. 38, 2020, doi: 10.1016/j.tsc.2020.100726.

[10] I. Fronza, L. Corral, and C. Pahl, "Combining Block-Based Programming and Hardware Prototyping to Foster Computational Thinking," presented at the Proceedings of the 20th Annual SIG Conference on Information Technology Education, 2019.

[11] MIT. (2023) "scratch-vm." (Version 1.3.26). https://github.com/llk/scratch-vm

[12] MIT. (2023)"scratch-gui." (Version 1.3.9). https://github.com/llk/scratch-gui

[13] Karishma Chadha, Eric Rosenbaum, Christopher Willis-Ford, and u9g. "Scratch 3.0 Extensions.". https://github.com/LLK/scratch-vm/blob/develop/docs/extensions.md

[14] A. Yorinks. "python_banyan.". https://github.com/MrYsLab/python_banyan