Sucre4Stem: A K-12 Educational Tool for Integrating Computational Thinking and Programming Across Multidisciplinary Disciplines

Sergio Trilles, Aida Monfort-Muriach, Enrique Cueto-Rubio, Carmen López-Girona, and Carlos Granell

***Abstract*—This article discusses the latest developments of the *Sucre4Stem* tool, as part of the *Sucre* initiative, which aims to promote interest in computational thinking and programming skills in K-12 students. The tool follows the Internet of Things approach and consists of two prominent components: 1) *SucreCore* and 2) *SucreCode*. *SucreCore* incorporates an advanced microcontroller packaged in a more compact design and enables wireless connectivity. *SucreCode*, the block-based visual programming tool, supports two different sets of blocks depending on the education grade, and facilitates wireless communication with *SucreCore*. At the educational level, *Sucre4Stem* fosters new group dynamics and encourages students to experiment real-world projects by promoting the "programming to learn" approach to concepts from other disciplines as opposed to the strategy widely applied in schools of "learning to program" in isolation.**

***Index Terms*—Computational thinking, Internet of Things (IoT), K-12, multidisciplinary disciplines, programming promotion.**

I. Introduction

WHEN observing today's younger generations, it is evident that Information Technology (IT) and mobile devices play a significant role in their daily lives. IT is everywhere, in and out of school. Indeed, computer science pioneers predicted that IT and programming will become even more pervasive in the future, serving as a universal means in both professional and personal settings [1]. Central to this vision is the concept of computational thinking, which is generally about "thinking" as a computer scientist or information technologist would [2]. Moreover, this way of "thinking" involves skills and competencies that provide benefits beyond the field of computing [3]. In short, computational thinking involves breaking down complex problems into smaller, more intellectually manageable parts and using logical reasoning and problem-solving techniques to solve them. It also involves recognising problem and solution patterns and using them to create computational algorithms and programs. Here, the term computational algorithm is vital to clearly highlight a key aspect of computational thinking; Denning [4], [5] emphasised that an algorithm is a series of steps that control an abstract machine that implement a computational model without human intervention. Computational algorithms are then designed to be executed partially or completely by a computing machine. Therefore, as Denning proposed, computational thinking also includes "designing the computational model, not just the steps to control it" [5, p. 33]. By developing these skills and patterns in K-12 education [6], preuniversity students can become better prepared for the challenges of university-level courses and beyond; these skills can thus be applied to various professions, such as engineering, mathematics, economics, architecture, journalism, to name a few, and be crucial in preparing the next generation of professionals for the unpredictable challenges they may face in their careers.

After a decade of implementing policies and actions to incorporate computer literacy and computational thinking into national educational curricula in many countries around the world [7], the expected impact and benefits of introducing computational thinking in schools do not make yet the above vision a reality. Although there are different strategies to deploy computational thinking education in schools, such as coding and programming, robotics [8], [9], games and puzzles, "the boundary between programming and computational thinking is somewhat blurry" [10, p. 30] when it comes to teaching and assessing computational thinking in practice. Programming is not always required to acquire computational literacy [11], but it facilitates its acquisition because it is the most visible part of computational thinking. As a result, computational thinking and programming are in practice largely considered overlapping terms in education [10].

Some critical voices try to demystify that widely accepted unique relationship between programming and computational thinking. For example, Guzdial et al. [12] conveyed the idea of integrating computing and computational thinking into other fields, rather than teaching it in isolation in computing courses. Similarly, Tissenbaum and Ottenbreit-Leftwich [13] envisioned the future of computer science education for K-12 students as a way "to understand computer science beyond simple learning to code." [p. 42]. They reinforce the idea of considering computer science more than just coding

Manuscript received 14 July 2023; revised 31 January 2024; accepted 27 June 2024. This work was supported by the Research Promotion Plan of the Universitat Jaume I under Grant UJI-B2022-64. The work of Sergio Trilles was supported in part by the Juan de la Cierva—Incorporación Postdoctoral Programme of the Ministerio de Ciencia e Innovación—Spanish Government under Grant IJC2018-035017-I funded by MCIN/AEI/10.13039/501100011033; in part by the "ERDF, a way of making Europe"; and in part by the European Union. The work of Enrique Cueto-Rubio was supported in part by the "Programa Yo Investigo" under Grant INVEST/2022/424 funded by the Generalitat Valenciana government, Spain, and in part by the European Union, NextGeneration EU. *(Corresponding author: Sergio Trilles.)*

The authors are with the Insitute of New Imaging Technologies, Universitat Jaume I, 12071 Castellón de la Plana, Spain (e-mail: strilles@uji.es).

Digital Object Identifier 10.1109/TE.2024.3422666

"to be integrated into a range of other disciplines rather than just a stand-alone subject" [13, p. 43]. Repenning and Basawapatna [10] aptly summarised this argument with the change from *learning to program* to *programming to learn*. While the former equates programming with computer science education and it is easily adopted and implemented in schools (i.e., teaching programming skills and concepts using, for instance, Scratch [14]), the latter pursues the ultimate goal of computational thinking seen as an "instrument of thought to understand powerful ideas in typical K-12 disciplines truly" [10, p. 30], which is very in line with the idea of "computational fluency" [15]. For example, versatile tools like BBC:Micro Bit [16] help students explore and experiment with multidisciplinary projects well beyond teaching them only specific computing concepts [17]. On the downside, this vision, programming to learn, "has not yet achieved a systemic impact in schools because it requires understanding how to meaningfully connect computer science and computational thinking with other disciplines" [10, p. 31]. Although attractive, the vision of programming to learn still represents an enormous challenge for K-12 educators and teachers. The *Sucre* initiative that we present here contributes precisely to the vision of programming to learn to support pre-university students to understand and explore concepts in other disciplines through computational thinking and programming.

The *Sucre* initiative (Sense yoUr Context and REact, www.programasucre.com) aimed to promote computational thinking as a problem-solving tool through the use of computational algorithms and programming together with sensors and actuators, encouraging the interest and curiosity of pre-university students in STEM disciplines (science, technology, engineering, and mathematics), and promoting their scientific vocation and commitment to science [18], [19], [20]. The initiative was built around four key pillars, including electronic devices capable of sensing or measuring the environment along with the maker movement ("do it yourself") [21], computational thinking skills and competencies (such as logic, abstraction, decomposition, creativity, evaluation, generalization, and design of computational algorithms), communication and collaboration between groups based on projects, and the scientific method. Another defining aspect behind *Sucre* was to democratise educational technologies by breaking barriers to access learning resources for computational thinking for socioeconomically disadvantaged schools. *Sucre* provides teachers and students with digital resources (e.g., online materials, videos, and tools) and physical resources (e.g., sensors and microcontrollers) that can be applied in the classroom so that students acquire computational thinking skills and know the principles of the scientific method.

Since this inception in 2016, the *Sucre* initiative has undergone significant changes, resulting in two complementary tools –*Sucre4Kids* and *Sucre4Stem*– that serve different educational stages and learning needs. *Sucre4Kids* is aimed at school children aged 5–10 years, covering the last year of preschool and the first four years of primary school. *Sucre4Stem* is designed for secondary school students (10–16) as well as intermediate vocational education and training (VET) courses. The boundary between *Sucre4Kids* and *Sucre4Stem* is somewhat blurry, especially for students in the last two years of primary school (10–12 years old), who can explore the advanced features of *Sucre4Kids* while simultaneously engaging with the basics of *Sucre4Stem*.

Our focus here is on the *Sucre4Stem* tool. To describe its design principles and technical features, and, most importantly, how *Sucre4Stem* can serve as a tool to realise the vision of *programming to learn*. So teachers and students can design and develop collaborative projects that bridge the K-12 disciplines through computational thinking and programming. In what follows, we briefly review the evolution of *Sucre* up to the present day, and then delve into the technical building blocks of the *Sucre4Stem* tool. Subsequently, we discuss a series of interventions among researchers and teachers to encourage the use of *Sucre4Stem* as a tool to promote computational thinking and programming for learning multidisciplinary concepts.

II. Overview of *Sucre*

When designing educational activities, it is essential to consider how individuals learn. Over time, various educational theories have been developed to explore and understand how people learn best in different contexts. A theory that interests us specially is constructionism, which emphasises the importance of creation and experimentation in the learning process [22]. Constructionism suggests that students learn best when they actively construct or explore real-world objects, especially if they are meaningful to them. By tapping into their interests and passions, students are more likely to enjoy and be motivated by the learning process. At *Sucre*, our initial goal was to design a tool to encourage students to learn by doing through experimental and participatory sessions [23] that fostered critical thinking, collective work, and social interaction. By creating important projects for students, they can develop a greater interest and passion for learning coding and computational concepts [15], [24].

Already in the preliminary prototypes developed in the realm of the *Sucre* initiative [25], we considered tangible computing to spark student interest in programming and computing concepts. Relevant initiatives in the market at that time, such as Cubelets[1] or LittleBits[2] prioritised the tangible aspect of programming by allowing students to touch and connect with sensing technology to code. *Sucre* also uses microcontrollers and electronic components to develop computational projects that prioritise tangible interaction. Indeed, the initial approach consisted of a convenient briefcase containing various electronic components (Fig. 1), paired with a visual block-based programming tool [20]. The briefcase contained all the necessary elements to replicate the minimum operation of a computer: a core and input/output devices. The core was an Arduino UNO mounted on a shield called Grove that simplified the connection of I/O devices to the core. The sensors served as input devices, while the actuators as output devices. The briefcase included four sensors for measuring air and soil humidity, sound, light, and proximity, as well as two actuators, a multicolored LED, and an LED bar. It

[1] https://modrobotics.com
[2] https://bit.ly/3CRwa9L

Fig. 1. Screenshot of the first version of the *Sucre* briefcase (2016–2017).

Fig. 2. Screenshot of the *SucreKit* for *Sucre4Kids* version.

also had four badges with specific roles for students to assign for collaborative projects, nine informative cards about the included sensors and actuators, and a type-B USB cable.

In the course of development, several activities were carried out to improve the methodology, target audience, and characteristics of the prototype being developed, reaching almost 400 high school students per year (see details in [18] and [20]). Due to a recent change in the education law in Spain [26], in which computational thinking is an integral part of the educational curricula, we have recently held training sessions for teachers to jointly explore the concept of computational thinking to take advantage of the *Sucre* initiative as a tool for programming to learn, as we describe in Section V.

Based on the experience gained with past interventions, along with the advancement of IT, efforts were made to tailor the solution to different educational levels. It is important to keep in mind that programming concepts cannot be taught the same way to infant, primary, and secondary school students. Effective communication that is adapted to the audience is crucial when introducing technical concepts. In this sense, Mercer-Mapstone and Kuchel [27] identified essential aspects to consider to convey scientific concepts to a nonscientific audience effectively. These include understanding the audience's prior knowledge about the subject, distinguishing between essential and nonessential concepts, using appropriate language and communication methods, and considering the social, cultural, and physical context of the activity. The format of an intervention is also significantly influenced by the daily work and environment of a nursery/primary classroom. This often involves working without screens, arranging groups at shared tables, using cards to develop literacy and recognise phonemes and letters, and using sensory and tangible learning techniques to promote good mobility in using routine utensils, such as pencils and scissors.

Therefore, we decided to divide the *Sucre* initiative into two tools, *Sucre4Kids* and *Sucre4Stem*, to serve different educational levels and needs. Technologically speaking, one of the challenges was to determine which parts of *Sucre4Stem* could be adapted for *Sucre4Kids* and what aspects had to be designed from scratch, given the prior knowledge and context of primary school students. As a result, both tools share the *SucreKit*, consisting of sensors, actuators, and a microcontroller, which is essential for students to assemble the components. However, while many components of the briefcase (Fig. 2) were retained, changes were made to how primary school students interact with them, such as using pictograms to pair sensors with corresponding pins and reducing the assembly complexity to one sensor and actuator at a time. The most significant change was in the programming part; *Sucre4Kids* employs a set of tangible cards for programming instead of the visual block-based programming tool in *Sucre4Stem*. Each card (called *SucreCards*) has an NFC tag on the back that encodes the type of programming element it represents. Working with cards is more familiar to younger students, as the cards are designed to be visually appealing and easily understandable for them, featuring pictograms, large format, and flexible puzzle pieces that can be combined to form the word "sucre." As the cards include visual cues, such as outer frames or dashed lines to help match them, students find it doable to arrange the cards in sequence on the table to specify the program's logic. After establishing the algorithm, students pass each card sequentially through the *SucreCore* reader to read the logic associated with a card. Finally, the program is executed using the special "Execute" card.

Regarding the *Sucre4Stem* tool, we updated it technologically to bring more functions and overcome the identified limitations detected over past interventions. This was achieved by, including the Internet of Things (IoT) paradigm. By amplifying the communication and network capabilities of *SucreCore* and developing a new comprehensive visual programming tool named *SucreCode* to support this paradigm. These advancements related to *Sucre4Stem* are explained in more detail below.

III. *Sucre4Stem*: Technical Building Blocks

As noted above, *Sucre4Stem* consists of two parts: 1) *SucreKit*, a compact briefcase that includes all the essential hardware components needed to build assemblies and 2) *SucreCode*, a Web-based tool for visual block-based programming across various devices. This section presents a

Fig. 3. *SucreKit* version of *Sucre4Stem* tool: Sensors (blue), actuators (green), and the *SucreCore*.

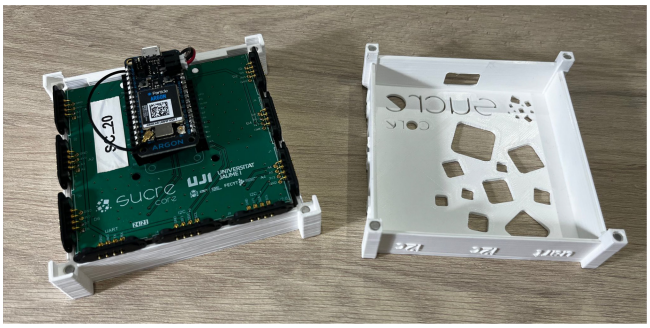

Fig. 4. *SucreCore*: microcontroller mounted on the custom shield (left) enclosed in the white 3D-printed box.

comprehensive overview of both components and their latest features.

A. SucreKit

The *SucreKit* briefcase (Fig. 3) contains sensors and actuators that can be wired to the microcontroller *SucreCore* (Fig. 4). The sensors included in the *SucreKit* are: a button, a distance sensor, a light sensor, a temperature/humidity sensor, a rotation angle sensor, and a noise sensor. It also has actuators, such as a simple LED, a multicolor LED, a buzzer, and a segment display. Magnetic connectors are built into each sensor/actuator to help wire them to the *SucreCore* (Fig. 3), facilitating assembly.

SucreCore is powered by the Particle Argon microcontroller [28], which has a 32-bit 64 MHz ARM Cortex-M4F CPU, 1 MB of flash memory, 256 KB of RAM, and a 20-pin GPIO mixed signal (6 × analogue, 8 × PWM), UART, I2C, and SPI. The microcontroller is connected to a custom-made shield with a suitable pin-out to facilitate the addition of magnetic connectors. The *SucreCore* also includes a battery (located below the shield, not visible in Fig. 4) with a power switch, providing students with greater ease of use and independence from external power. All the components are enclosed in a 3D-printed box made of PLA designed by the authors.

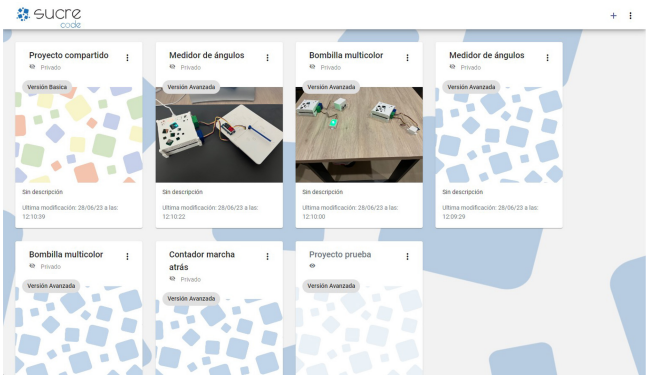

Fig. 5. Screenshot of the main interface of the SucreCode app.

SucreCore differs from similar solutions by incorporating IP connectivity, which allows it to function as an IoT device. This is done by using Wi-Fi for IP/Internet protocol connectivity and bluetooth LE (BLE) for additional connection options. This advanced connectivity gives users with a variety of cutting-edge capabilities, such as the ability to initiate over-the-air (OTA) updates with new codes or sketches without the need for a wired computer connection.

B. SucreCode

While the *SucreKit* allows students to build an assembly, the *SucreCode* allows them to define the logic to make the assembly operational. *SucreCode* (Fig. 5) is a Web-based (Angular) application for students to develop their coding skills. It simplifies programming tasks using the same principles as Scratch/Blockly: drag and drop blocks to visually create a program by combining them. This application is compatible with both desktop computers and touchscreen tablets commonly used in classrooms as it follows mobile-first principles using HTML5 responsive libraries.

The main view of *SucreCode*, where the user's projects are listed, is shown in Fig. 5. Users can tag their projects with keywords for better categorization and, for example, use tags to identify student groups or classrooms. *SucreCode* supports user management with varying privilege levels. The *SuperAdmin* has permission to assign devices to users, view all users, and change their roles or delete them. *CenterAdmins*, typically teachers, can manage users in their high school, add or delete users, change device assignments, and create predefined tag sets for users. Finally, *RegularUsers*, reserved for students, are assigned a *SucreCore* and can start using the platform immediately without any additional configuration. They can share their projects publicly and use predefined tags to categorise them. These improvements were implemented using different Angular views and Firebase for user management and project persistence.

SucreCode is aimed at students from 10 to 18 years old, including the last years of primary school (10–12 years old). To meet the needs of this broad group of students, two sets of blocks, i.e., instructions, have been designed. The "Basic" set (Fig. 6) simplifies programming and avoids the use of constructs, such as variable definition, which can be

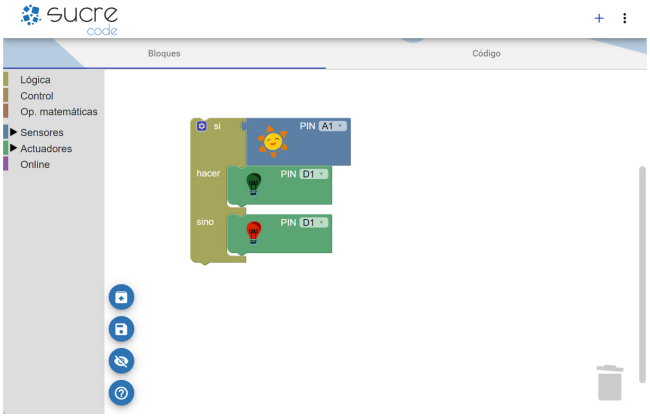

Fig. 6. Screenshot of the *SucreCode* application, displaying a program built using the basic block set.

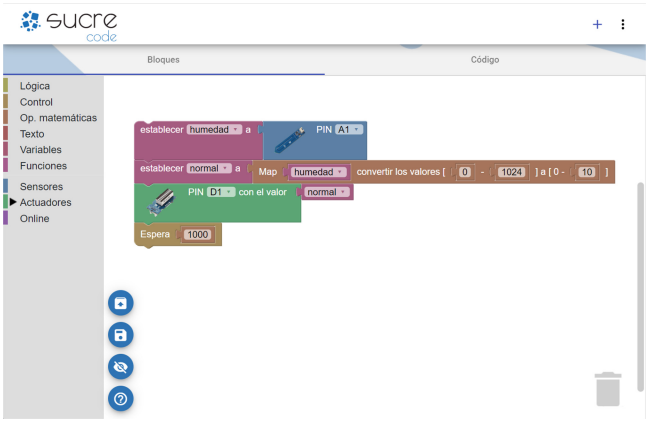

Fig. 7. Screenshot of the *SucreCode* application, demonstrating a program created using the advanced block set.

more difficult to understand for students who are exposed to programming for the first time. Blocks add functionality to represent sensor and actuator states using more easily understood pictograms. Blocks that reference sensors do not require threshold values to be entered in conditions; the threshold value and the conditional statement are embedded in the block, so the block only returns a True/False value whether or not the condition is met. For example, the "Sun" block (Fig. 6) returns True if the light sensor detects more light than the predefined threshold value, and False otherwise. In this way, the use of such aggregate blocks simplifies the programming process by focusing students' attention on high-level constructs, which is especially useful for beginners who are first exposed to coding. The actuator blocks have also been simplified in the same way as the sensor blocks.

When users log into the *SucreCode* application, they select which set of blocks to use for their projects before starting a new project. When the "Advanced" option is selected (Fig. 7), students are shown the full set of blocks in greater detail. Technically, the basic blocks (high-level of abstraction) are made up of advanced blocks (low-level of abstraction), hiding the inner details of the computing instructions. Yet, by exposing advanced students to the full range of advanced blocks or instructions, they can create more nuance and complex programs, have full control of the algorithms they want to develop, and access to sophisticated constructs, such as functions and definitions of shared variables (see below).

Enabling Internet access for *SucreCore* offers multiple possibilities. The most important is the ability to remotely update the code to be executed by a *SucreCore* without needing a wired connection. *SucreCode* generates a new program (text code) based on the combination of visual blocks and packages it with the corresponding sensor/actuator libraries. This code package and the microcontroller identifier are sent through the cloud service's update operation to compile and release the code update on the corresponding *SucreCore*. The *SucreCode* Web interface includes an icon bar in the program creation view, which facilitates one-click updates (top blue icon in Figs. 6 and 7). This bar provides additional actions, such as accessing online documentation, sharing a project between users of the same school or classroom, and saving a project to resume work later. Additionally, a status bar (not shown in Figs. 6 and 7) displays status messages for update and save actions. Combined with the ability to support different client devices, *SucreCode* becomes a versatile, flexible, and user-friendly tool for different learning environments, such as homes, regular classrooms, and computer labs.

As part of the advanced block set, student groups can share sensed data with each other through shared variables. Data captured by a sensor connected to a *SucreCode* can be shared with multiple *SucreCore*s. A couple of special blocks allow students to convert a local variable into a shared variable via the publish/subscribe paradigm [29]. For example, consider the simplest case where two *SucreCore*s are involved, one assuming the role of publisher or producer of data, and the other the subscriber or consumer of data. To make a previously defined variable shareable in a *SucreCore* publisher, the "Publish" block is attached to the block representing a local variable. This block specifies a name used for other *SucreCore*s to subscribe to that variable. This means that unique name defines a type of communication channel shared between *SucreCore*s through an MQTT connection, the chosen technology for developing this functionality. Subscribers only need to use the "Subscribe" block and indicate such a name in the block's configuration. Like the "Publish" block, the "Subscribe" block is also attached to a local variable to store the received values. Therefore, this feature turns isolated *SucreCore*s into collaborative IoT-powered devices. Groups of students, whether located in the same room or apart, can cooperate to create a larger project as the sum of the parts, which is one of the drivers of computational thinking.

Once a collaborative project with shared variables collects and shares data, these values are not permanently saved by default. Once publishing stops, shared values are no longer accessible. To enable explicit data persistence, both for local or shared variables, another special block was created that stores values persistently in a database. This block requires students to specify the saving frequency (e.g., every few minutes) to avoid excessive data storage and server overload. For example, depending on the intended use, a temperature observation can be safely saved once per hour and still have a complete time

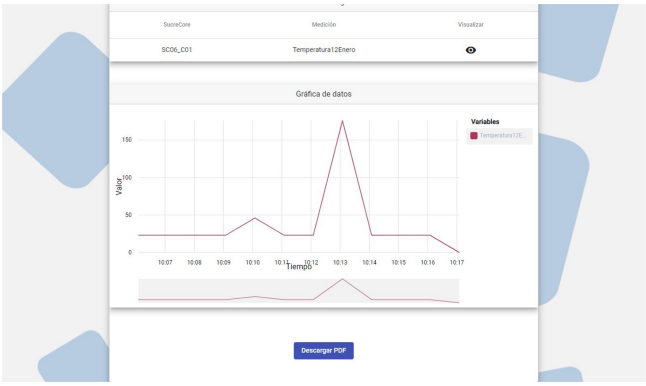

Fig. 8. Screenshot of the *SucreCode* app showing the chart view of values saved using the persistence block.

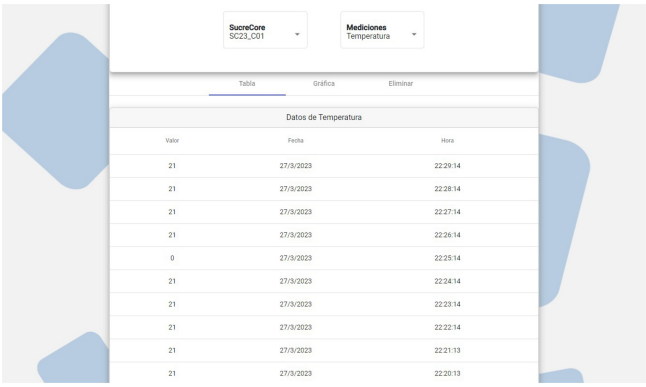

Fig. 9. Screenshot of the *SucreCode* application's tabular view which displays the values stored through the persistence block.

series of that phenomenon. Therefore, the "Persistence" block allows students to store data for later visualisation or analysis. To this end, *SucreCode* provides two views to display stored data in graphical or tabular format (Figs. 8 and 9). Stored data can also be downloaded as a comma-separated values (CSVs) file for analysis using other tools. The InfluxDB database and the Node-red flow editor were used to develop the services required for data persistence.

IV. *Sucre4Stem*: Community Feedback

During January and February 2023, the authors visited three secondary schools in the region (IES Miralcamp and IES Professor Broch i Lluc, Vila-real, Spain; IES Serra d'Irta, Alcalà de Xisvert, Spain). A program consisting of 7 sessions of 3 h each was deployed to train the science and technology teachers of each school. The training focused on the possibilities of the *Sucre4Stem* tool to design creative learning situations in classroom that integrate computational thinking at its core. Added to these visits, the authors organised a 20-h training course at the university premises from March to May 2023. Ten teachers from different schools in the region actively participated in the 5 sessions of the training course. In total, 23 teachers with different profiles (mathematics, physics, chemistry, biology, technology, etc.) participated in the different training activities.

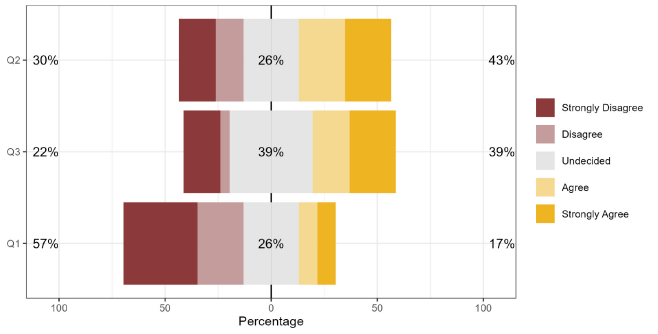

Fig. 10. Previous knowledge on computational thinking practices ($N = 23$).

Before starting, all participants took a simple survey with the following three questions to get an idea of their initial knowledge and any misconceptions related to computational thinking: (Q1) does being able to use technological devices mean having developed computational thinking skills?; (Q2) can computational thinking skills be developed in your classroom now?; and (Q3) do you feel prepared to develop computational thinking with your students?. Moreover, the survey included a free-form question "from your point of view, computational thinking is. . .." Fig. 10 illustrates a Likert-scale plot for the previous survey questions. Almost 60% of respondents disagree with Q1, which equates computational thinking with generic digital skills, such as the use of digital tools. However, regarding the free-form question, most respondents defined computational thinking as programming or learning to program, which, as we noted earlier, is a narrow view of what computational thinking really encompasses. Since teachers' perceptions regarding computational thinking were strongly connected to mere programming, this made us doubt the results of questions Q2 and Q3, since the respondents probably did not see computational thinking as a multiskilled approach (Section I), including programming skills, to learn multidisciplinary concepts.

Given this widespread misconception, after the initial survey, the next topic of discussion was to find a common understanding of the meaning and multiskilled characterisation of computational thinking linked to the vision of "programming to learn," so that participants could reflect on their previous misconceptions about what computational thinking actually covers and recognise the potential of *Sucre4Stem* to achieve the vision of "programming to learn." Subsequent sessions were intentionally informal, which encouraged a lively discussion between participating teachers and researchers on how to use and combine *Sucre4Stem* in classroom. Many ideas for integrated and multidisciplinary projects emerged, especially for secondary education where the subjects of mathematics, biology and technology are integrated into the same scientific-technical area in the Spanish secondary education system [30]. Table I summarises the most relevant project ideas raised during the training sessions between researchers and teachers. As Repenning and Basawapatna [10] argue, devising activities that adequately mix computational thinking and programming on the one hand with concepts from other disciplines, on the other hand, was not an easy exercise.

TABLE I
SUMMARY OF MULTIDISCIPLINARY PROJECTS

Project name	**Programming concepts**	**Hardware elements**	**To learn concepts of...**
Music creation	Sequence, loops	Buzzer	Music
Heartbeat simulation	Variables, functions, sequence	Rotary angle, LED and 4-Digit Display	Biology, maths
Carbon dioxide impact on greenhouse effect	Variables, shared variables, loops, data persistence	Temperature sensor	Environmental science, maths
Geometry	Maths operations and variables	Button, ultrasonic distance sensor, and 4-Digit Display	Maths
Motion detection alarm	Variables, shared variables, conditionals, loops	Button, ultrasonic distance sensor, buzzer and chainable LED	Technology, IoT-powered devices

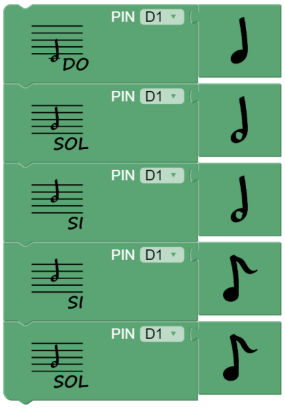

Fig. 11. Programming blocks for the music creation project.

Algorithm 1 Heartbeat Simulation Pseudocode

1: **while** True **do**
2: *RotaryAngleInput* ← Get rotary angle input.
3: $ShortTime = \frac{60.000}{10 \cdot RotaryAngleInput}$
4: $LongTime = 7 \cdot ShortTime$
5: Power on LED for *ShortTime* milliseconds (First pumping).
6: Turn LED off for *ShortTime* milliseconds.
7: Power on LED for *ShortTime* milliseconds (Second pumping).
8: Turn off LED for *LongTime* milliseconds (Time between beats).
9: **end while**= 0

Indeed, the activities proposed in the following section did not arise spontaneously but were the result of a reflection and continuous exchange of ideas by all participants, i.e., researchers and teachers.

V. *Sucre4Stem* AS TOOL FOR PROGRAMMING TO LEARN

A. Music Creation

The first proposal for a transversal project was to explore the potential of the *Sucre4Stem*'s set of basic blocks (see Section III-B) in musical notation and melody generation. This project involves using the basic blocks (see Fig. 11) that encode various musical notes and tones. The process for generating melodies is straightforward, with a focus on working with the available blocks and sequencing. Additionally, loops can be used to create choruses while retaining the number of blocks required. Assembling the project basically involves the buzzer actuator to produce the desired tones.

B. Heartbeat Simulation

The second project is a heartbeat simulator, which uses *Sucre4Stem*'s advanced blocks like the other projects below. This simulator can be adjusted to simulate a faster or slower heartbeat, depending on whether we want to simulate physical activity or rest. To build the simulator, concepts of maths and biology are required. The heart beats in two phases, which can be heard with a stethoscope as "lup-DUP," followed by a brief silence until the next beat. Heart rate is the number of beats that occur in one minute. Generally, the heart rate ranges from 50 to 100 beats per minute (bpm) under resting conditions or everyday activities, depending on physical condition and age. During physical activity or stress periods, the heart rate can range between 160 and 220 bpm, depending on age and physical condition.

Maths are also required to calculate and adjust beat times to the simulated heart rate, which are operations that can be coded. To keep the example simple, we assume that the time between heartbeats (*LongTime*) is approximately seven times the length of each heartbeat (*ShortTime*). To simulate this, we use an LED that will turn on and off in a time proportional to the simulated heart rate, according to the following pseudocode. (Algorithm 1).

The assembly is made up of the rotary angle, an LED, and the four-digit display, all wired to the *SucreCore*. Classical programming constructs can be used to code the above algorithm. Beyond this, computational thinking patterns, such as modularization, decomposition, and abstraction can enrich the learning experience through, for example, the definition of functions (see Fig. 12). The function *GetFrequency* calculates the simulated frequency using the rotary angle sensor and displays it on the four-digit screen, separating that logic (line 2, Algorithm 1) from the heartbeat simulation (lines 3–8, Algorithm 1). The other function *CalculateTime* computes the two waiting times, *ShortTime* and *LongTime*.

C. Carbon Dioxide Impact on Greenhouse Effect

Another experiment discussed among teachers and researchers was to model the impact of carbon dioxide on the greenhouse effect (inspired by [31]) as a way to integrate computer science and environmental science. The idea of the experiment was to allow students to evaluate carbon

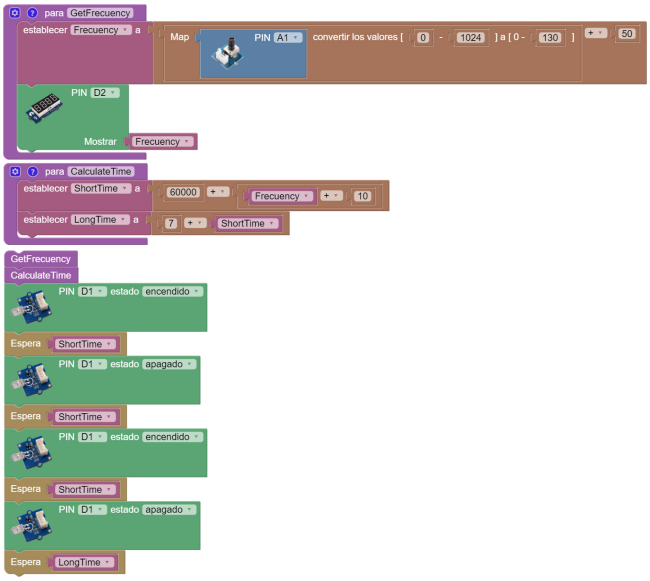

Fig. 12. Programming blocks for the heartbeat simulation project.

Fig. 13. Assembly of the carbon dioxide impact on greenhouse effect project.

dioxide's contribution to air warming by recreating three different situations (samples). Each sample consisted of a plastic bottle with a different object inside: nothing (control sample), a plant, and a carbonated drink can (Fig. 13). The experiment consisted of taking the temperature inside the bottle at regular intervals for 2 h to evaluate in which sample the temperature increases the most. Here, *Sucre4Stem* was used to construct an assembly composed of a temperature sensor to periodically take (every 5 min, for example) the temperature inside each bottle and persist the collected values so that students can analyse and compare later (for example, via graphs and basic descriptive statistics) the temperature measurements of the three samples and discuss which sample affected the most to the heating of the air inside each bottle. Abstraction, programming constructs, data collection, persistence, and analysis are computational thinking skills and patterns embedded in this greenhouse effect model to simulate

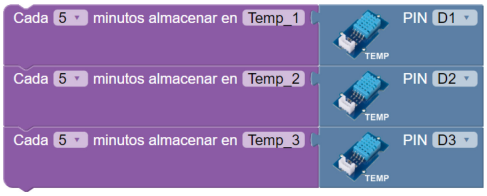

Fig. 14. Programming blocks for the carbon dioxide impact on greenhouse effect project.

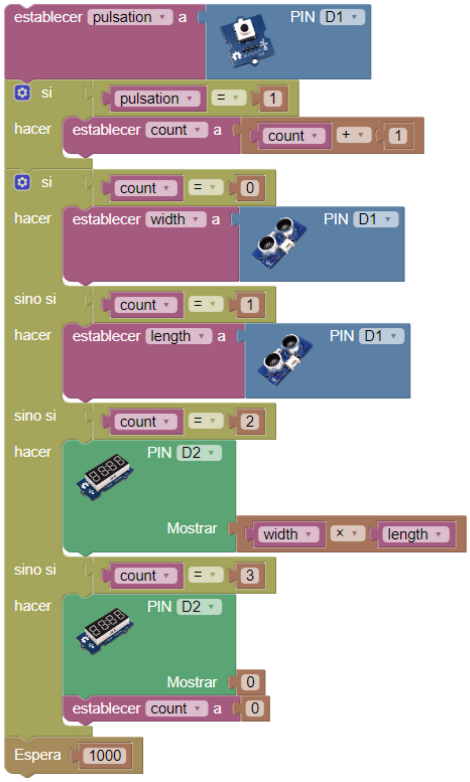

Fig. 15. Programming blocks for the geometry project.

that phenomenon under different circumstances. Fig. 14 shows the programming blocks used to develop this project.

D. Geometry

The fourth project idea focused on mathematics, specifically in the branch of geometry. The goal was to use a distance sensor, button, and 4-digit display to calculate areas or volumes. Length and width distances (and height for volumes) are captured using the ultrasonic distance sensor and stored when pressing the button. In particular, students measure the width of their classroom, for example, and press the button to store the value. They do the same to capture the length. After the second press, the area is calculated and displayed on the 4-digit screen. Fig. 15 shows the corresponding block-based program. This project can be expanded to volumes by adding a third value –height– and computing the volume accordingly. It can also be carried out collaboratively, obtaining the distances from distinct *SucreCore*s and having another *SucreCore* to compute the area or volume.

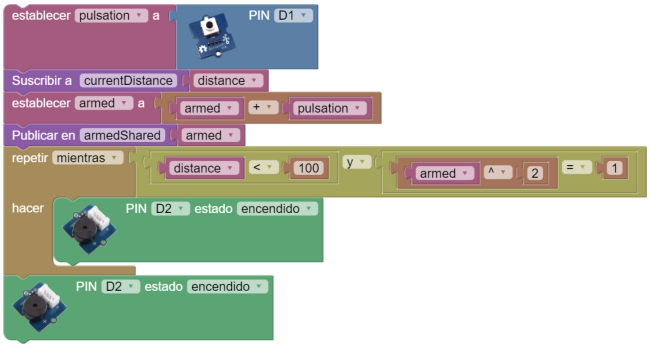

Fig. 16. Programming blocks for the motion detection alarm project: *SucreCore* that monitors distance.

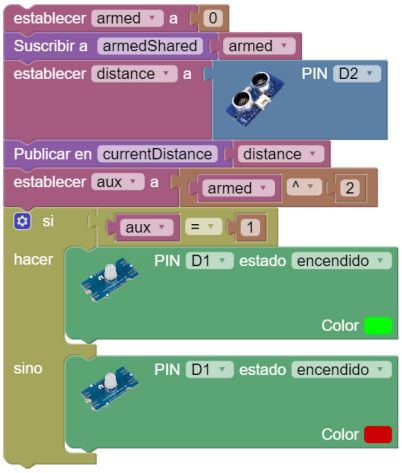

Fig. 17. Programming blocks for the for the motion detection alarm project: *SucreCore* that manages alerts.

E. Motion Detection Alarm

This example recreates a remote motion detection alarm, which works similar to a real burglar alarm installed in homes. The novelty here is the *remote* aspect, which implies two *SucreCore* devices communicate each other through shared variables (see Section III-B). This example adheres to Tissembaum and Ottenbreit-Leftwich [13]'s vision that computer science education must go beyond the classroom to meet the world which "allows students to see how computer science operates in the world as a means to solve real-world problems" [13, p. 43].

As the assembly comprises two IoT devices, each device is configured with a set of sensors/actuators for the task it performs. One device models the motion detection by installing a *SucreCore* near, for example, the access door to a room that it monitors. In particular, this *SucreCore* is equipped with a distance sensor that continuously measures the distance (in cm) between the *SucreCore* and the door, and with a multicolor led actuator to indicate whether the motion is authorised (green) or not (red). When someone enters or passes through the door, the observed distance value will be lower than usual (without presence), which will indicate motion detected (see Fig. 16).

The captured distance value is published as a shared variable so that the second *SucreCore* can consumes it. This second *Sucrecore* manages the received alerts and it is wired with a button sensor and a buzzer actuator (see Fig. 17). When the received value is lesser than a certain threshold, the buzzer is activated, emitting a continuous beep to signal a motion alert. As such, the button controls the activation/deactivation of the alarm, whose value is transmitted as a shared variable to the first *SucreCode* so that it activates the multicolor LED accordingly. Therefore, in this example students explore real-time, bi-directional communication of data between two remote IoT-powered devices to recreate a real situation.

VI. Conclusion

This article provides an overview of the current status of the *Sucre* initiative. More specifically, we detail the *Sucre4Stem* tool, designed to promote computational thinking and programming among pre-university students. *Sucre4Stem*, through *SucreCore*, is based on the IoT paradigm, which simplifies microcontroller programming (using OTA) and enables remote communication between devices. Its design also facilitates collaborative projects, promoting interactive group learning. The Web-based tool *SucreCode* uses *SucreCore*'s connectivity features to send code to microcontrollers wirelessly. Additionally, it provides advanced features, such as user authentication, password protection, project saving, and shared variable generation.

Sucre4Stem has been adapted to serve a broader audience with the creation of two sets of blocks. The basic set generates simpler and more direct statements, eliminating the need to create variables and simplifying programming for beginners. The advanced set offers full support for traditional programming statements, such as declarations of variables, functions, etc. The advanced mode has also been enhanced to enable shared variables between *SucreCore*s and the storage of sensor data values.

It should be noted that the *Sucre4Stem* tool is aligned with the vision of programming to learn, in which computational thinking and programming concepts are integrated into multidisciplinary projects to allow students to experiment and learn key concepts from other domains. Through a series of training sessions, we put this vision into practice with high school teachers who initially struggled to integrate programming into their disciplines. However, through discussion and collective reflection, a series of multidisciplinary projects progressively emerged to explore disciplines as diverse as music, biology, mathematics, and environmental science through programming, thus taking advantage of the potential of computational thinking in the classroom beyond its classic role in computer science and programming education in isolation.

References

[1] V. G. Cerf, "Computer science in the curriculum," *Commun. ACM*, vol. 59, no. 3, pp. 7–7, 2016.

[2] J. M. Wing, "Computational thinking," *Commun. ACM*, vol. 49, no. 3, pp. 33–35, Mar. 2006. [Online]. Available: https://doi.org/10.1145/1118178.1118215

[3] A. V. Aho, "Computation and computational thinking," *Comput. J.*, vol. 55, no. 7, pp. 832–835, 2012.
[4] P. J. Denning, "Beyond computational thinking," *Commun. ACM*, vol. 52, no. 6, pp. 28–30, Jun. 2009. [Online]. Available: http://dx.doi.org/10.1145/1516046.1516054
[5] P. J. Denning, "Remaining trouble spots with computational thinking," *Commun. ACM*, vol. 60, no. 6, pp. 33–39, May 2017. [Online]. Available: http://dx.doi.org/10.1145/2998438
[6] C. Wang, J. Shen, and J. Chao, "Integrating computational thinking in STEM education: A literature review," *Int. J. Sci. Math. Educ.*, vol. 20, no. 8, pp. 1949–1972, 2022.
[7] M. U. Bers, "Coding and computational thinking in early childhood: The impact of ScratchJr in Europe," *Eur. J. STEM Educ.*, vol. 3, no. 3, p. 8, 2018.
[8] Y. H. Ching and Y. C. Hsu, "Educational robotics for developing computational thinking in young learners: A systematic review," *TechTrends*, vol. 68, pp. 423–434, May 2024.
[9] M. Chevalier, C. Giang, A. Piatti, and F. Mondada, "Fostering computational thinking through educational robotics: A model for creative computational problem solving," *Int. J. STEM Educ.*, vol. 7, p. 39, Dec. 2020.
[10] A. Repenning and A. Basawapatna, "Explicative programming," *Commun. ACM*, vol. 64, no. 11, pp. 30–33, Nov. 2021. [Online]. Available: https://dl.acm.org/doi/10.1145/3486642
[11] A. G. Chakarov, M. Recker, J. Jacobs, K. Van Horne, and T. Sumner, "Designing a middle school science curriculum that integrates computational thinking and sensor technology," in *Proc. 50th ACM Tech. Symp. Comput. Sci. Educ.*, 2019, pp. 818–824.
[12] M. Guzdial, A. Kay, C. Norris, and E. Soloway, "Computational thinking should just be good thinking," *Commun. ACM*, vol. 62, no. 11, pp. 28–30, Oct. 2019. [Online]. Available: https://dl.acm.org/doi/10.1145/3363181
[13] M. Tissenbaum and A. Ottenbreit-Leftwich, "A vision of K-12 computer science education for 2030," *Commun. ACM*, vol. 63, no. 5, pp. 42–44, Apr. 2020. [Online]. Available: https://dl.acm.org/doi/10.1145/3386910
[14] M. Resnick et al., "Scratch: Programming for all," *Commun. ACM*, vol. 52, no. 11, pp. 60–67, 2009.
[15] M. Resnick and N. Rusk, "Coding at a crossroads," *Commun. ACM*, vol. 63, no. 11, pp. 120–127, 2020.
[16] N. Kvaššayová, M. Cápay, S. Petrík, M. Bellayová, and E. Klimeková, "Experience with using BBC micro:bit and perceived professional efficacy of informatics teachers," *Electronics*, vol. 11, no. 23, p. 3963, Nov. 2022. [Online]. Available: http://dx.doi.org/10.3390/electronics11233963
[17] A.-M. Cederqvist, "An exploratory study of technological knowledge when pupils are designing a programmed technological solution using BBC micro: Bit," *Int. J. Technol. Design Educ.*, vol. 32, no. 1, pp. 355–381, 2022.
[18] S. Trilles and C. Granell, "SUCRE4Kids: El fomento del pensamiento computacional a través de la interacción social y tangible," *Actas de las Jornadas Sobre Enseñanza Universitaria de la Informática*, vol. 3, p. 14, Jul. 2018.
[19] S. Trilles, D. Tortosa, and C. Granell, "La evolución del proyecto Sucre4Kids mediante el paradigma del Internet de las Cosas," *Actas de las Jornadas sobre Enseñanza Universitaria de la Informática*, vol. 5, pp. 53–60, Jul. 2020.
[20] S. Trilles and C. Granell, "Advancing preuniversity students' computational thinking skills through an educational project based on tangible elements and virtual block-based programming," *Comput. Appl. Eng. Educ.*, vol. 28, no. 6, pp. 1490–1502, 2020.
[21] C. Anderson, *Makers*. Amsterdam, The Netherlands: Nieuw, 2013.
[22] E. Ackermann, "Piaget's constructivism, Papert's constructionism: What's the difference," *Future Learn. Group Publ.*, vol. 5, no. 3, p. 438, 2001.
[23] P. J. Sotorrio-Ruiz, D. Trujillo-Aguilera, C. García-Berdonés, F. J. Sanchez-Pacheco et al., "Aproximación a la técnica 'Aprender Haciendo' para la docencia en microprocesadores," in *Proc. Congreso Universitario de Innovación Educativa en las Enseñanzas Técnicas*, 2017, p. 25.
[24] M. Tissenbaum, J. Sheldon, and H. Abelson, "From computational thinking to computational action," *Commun. ACM*, vol. 62, no. 3, pp. 34–36, Feb. 2019. [Online]. Available: https://doi.org/10.1145/3265747
[25] S. Trilles, C. Granell, and E. Aguilar, "Sucre4kids: Tres años de experiencia en la incentivación del pensamiento computacional en edades preuniversitarias," in *Proc. TICAI TICs Para El Aprendizaje de la Ingeniería*, 2019, pp. 49–56.
[26] "Ley Orgánica 3/2020, de 29 de diciembre, por la que se modifica la Ley Orgánica 2/2006, de 3 de mayo, de Educación [National education act]," *Boletín Oficial del Estado*, vol. 340, pp. 122868–122953, Dec. 2020. [Online]. Available: https://www.boe.es/eli/es/lo/2020/12/29/3
[27] L. Mercer-Mapstone and L. Kuchel, "Core skills for effective science communication: A teaching resource for undergraduate science education," *Int. J. Sci. Educ. B*, vol. 7, no. 2, pp. 181–201, 2017. [Online]. Available: https://doi.org/10.1080/21548455.2015.1113573
[28] "Particle argon," Accessed: May 1, 2020. Online]. Available: https://bit.ly/31xX4RW
[29] U. Hunkeler, H. L. Truong, and A. Stanford-Clark, "MQTT-S—A publish/subscribe protocol for wireless sensor networks," in *Proc. 3rd Int. Conf. Commun. Syst. Softw. Middleware Workshops (COMSWARE)*, 2008, pp. 791–798.
[30] "Real Decreto 217/2022, de 29 de marzo, por el que se establece la ordenación y las enseñanzas mínimas de la Educación Secundaria Obligatoria [Curricular organization of compulsory secondary education]," *Boletín Oficial del Estado*, vol. 76, pp. 41571–41789, Mar. 2022. [Online]. Available: https://www.boe.es/eli/es/rd/2022/03/29/217
[31] A. M. Ortiz Espín, "Existe el cambio climático? [is there climate change?]" Accessed: Jul. 13, 2023. [Online]. Available: https://descargas.intef.es/recursos_educativos/ODES_SGOA/ESO/BG/3A.2_-_Cambio_climtico_3ESO/index.html

Sergio Trilles received the Doctoral degree in geospatial information integration from Universitat Jaume I (UJI), Castellón de la Plana, Spain, in 2015.

After several postdoctoral fellowships at different levels, he is currently a Juan de la Cierva-Incorporation Postdoctoral Researcher with the GEOTEC Group, UJI.

Aida Monfort-Muriach received the master's degree in geospatial technologies in a joint programme run by the Universitat Jaume I (UJI), Castellón de la Plana, Spain, Westfälische Wilhelms-Universität Münster, Münster, Germany, and Universidade Nova de Lisboa, Lisbon, Portugal, in 2015.

She is a Computer Engineer. Since 2016, she has been an Associate Researcher with the GEOTEC Group, UJI.

Enrique Cueto-Rubio received the B.S. degree in computer engineering from Universitat Jaume I (UJI), Castellón de la Plana, Spain, in 2022, where he is currently pursuing the master's degree in intelligent systems.

He is a member of the GEOTEC Research Group, UJI, where he directly worked on the *Sucre4Stem* development in 2021. He rejoined the group in 2022 to work with data science and GIS technologies with a "Programa Yo Investigo" grant.

Carmen López-Girona is currently pursuing the B.S. degree in computer science with Universitat Jaume I (UJI), Castellón de la Plana, Spain, where she completed an internship with UBIK Geospatial Solutions to work on *Sucre4Stem* development.

Her research interests revolve around teaching strategies and methodologies for secondary education.

Carlos Granell received the degree in computer engineering in 2000, and the Ph.D. degree in computer science from the Universitat Jaume I (UJI), Castellón de la Plana, Spain, in 2006.

He is an Associate Professor with UJI, and a member of the GEOTEC Group. His research interests lie in the multidisciplinary application of geographic information science, spatial analysis and visualisation of new forms of spatial data, and in the search for synergies between education and science dissemination projects.