Received: 21 December, 2022, Accepted: 16 February, 2023, Online: 27 March, 2023
DOI: https://dx.doi.org/10.55708/js0203002

Coding: First Steps from Kindergarten up to Primary School

Elisa Benetti*, Gianluca Mazzini
Lepida ScpA, Via della Liberazione 15, 40128, Bologna, Italy
*Corresponding author: Elisa Benetti & elisa.benetti@lepida.it

ABSTRACT: Computational thinking is now featured in many school curricula around the world. It is in fact defined as the "new English", emphasizing its universally recognized indispensability. Despite this, the subject is almost never addressed until primary school where, however, hours dedicated to it are often too limited. Our first training proposal, including basic coding concepts in kindergarten, led to better results than expected in terms of children's understanding and involvement. Our field training has led to a refinement and expansion of the program in these past three years. The primary objective is to begin the study of coding at the age of three, when the foundations of logical thinking are actually already present, due to get to the writing of the first programs in pseudocode and analysis of programming languages at the end of elementary school. All methodologies used are chosen on the basis of the possibility of following a single logical trend, which gradually increases the concepts to be learned and their difficulty, but always starting from already known bases, previously addressed. This allows to optimize learning times by minimizing the necessary hours and human resources and still obtaining the desired results. In addition to not burdening the number of hours available, a further firm point was not to burden schools economically either: costs were in fact always achievable without any problems. Having no impact either on the budget, or on the number of hours, or on the required staff makes this program easily feasible for any school.

KEYWORDS: Coding, Kidergarten, Primary School

1. Introduction

The concept of computational thinking was first introduced by Seymour Papert in the book Mindstorms [1], published in 1980, where in his theory of learning, known as Constructivism and based on the LOGO language he invented, he states how the computer is an important new medium for learning. Indeed, it is seen not only as a machine with which you can process information, but a tool for building, manipulating, learning, discovering and even making mistakes. A very important point of his theory for the purposes of the work presented in this article, in fact, is that error is not seen in a negative light, but as a constructive aspect of the learning process. To err is to explore in search of alternative solutions to the problem. More recently, in 2006, this concept has been taken up by computer scientist and MIT professor Jeanette Wing, who defines it as follows, "Computational thinking is a process of formulating problems and solutions in a form that is executable by an agent who processes information." [2] Indeed, we know that computers are used to solve problems, but even before solving a problem, it is important to understand by what means it can be solved, and it is computational thinking that enables us to do this.

The revolution of this concept is to note that it is not only important to solve problems, but more importantly to understand them, in order to formulate a process that leads to its resolution. This process can be performed by an agent or executor, figures that we will see in the exercises used, who implements instructions in a mechanical and unconscious manner, replicating human thinking. Computational thinking has been used in programming for the longest time, but recently also in coding, as well as in educational robotics. Tools used are not only technological and related to these disciplines, but also in normal life situations involving the decomposition of a problem, just to emphasize its importance in any aspect of life. Every day, without even realizing it, we find ourselves deciding on the expression of a solution by instructions and the execution of those instructions: finding the shortest route to a place, executing a recipe, assembling an object, assembling constructions. All of those listed are processes that involve computational thinking and presuppose a set of precise, orderly, clear, and repeatable instructions that will enable an effective solution to be reached by whoever is executing them. These instructions represent a de facto algorithm, that will certainly lead to the solution and can be applied to another identical problem with the same result, just as is the case with the algorithms behind any type of programming: from making decisions in a video game to performing an Internet search to managing interpersonal relationships on a smartphone.

Thus, computational thinking can be described by three main stages:

1. Abstraction: formulation of the problem;

2. Automation: expression of the solution;
3. Analysis: execution of the solution and evaluation.

All this is done by starting with the ability to break down a complex problem into several parts so that it can be tackled more easily. In this, coding is similar to mathematics: it is the logic of everything that works in a programmable way as mathematics is the logic of numbers and figures.

The International Society for Technology in Education [3], too, further highlights its importance in school education, pointing out that computational thinking allows people to:

1. Represent problem data through specific models;
2. Organize problem data in a logical manner;
3. Formulate and analyze problems so that they can be solved by a performer, computer or human;
4. Segment solutions into sequences of ordered, accurately described steps, or automate them through algorithms;
5. Identify possible solutions to implement the one that is the most effective and efficient in terms of effort and resources;
6. Abstract such processes for solving similar problems;

Computational thinking therefore is not only closely related to computer science and it is essential to develop it from an early age. One of the most effective ways to do this is to cultivate it through the use of coding tools, the process of writing languages and instructions intended for machines. Coding also constitutes a practical and immediate way to apply the theory of computational thinking and its previously illustrated steps and tools, while having fun, leaving room for creativity and imagination, while learning a new language and a new way of seeing problems and situations and expressing one's ideas and solutions clearly and effectively, thus also developing one's intelligence and critical thinking.

Coding, is not the only way to develop, or apply, computational thinking, but it has proven to be particularly effective because of the immediacy, interactivity, variety, availability and versatility of available tools. For this reason, its presence in school curricula is now worldwide recognized as indispensable as also explained in [4]. However, the training solutions and proposals available to date have been designed in limited areas. In fact, they are often focused only on the specific skills of an age: a lot of work has been done to exploit cognitive skills already in the pre-school context. For example, [5] shows how a coding course has led to an objective increase in problem solving skills and cognitive abilities in 4 and 5 year old children. On the other hand, children of various elementary schools in [6] used the Code.org site showing how, after following 8 coding activities, also in this case cognitive abilities increased, and not only that: children began to spend more time to planning, with increased ability to solve standardized planning tasks and in many cases even led to an inhibition of overbearing responses. In other researches, such as [7], the age analyzed is broader but the tool used is very specific, in this case apps. Both [8] and [9] searched in several elementary schools with a wider time range but in both cases using only the Scratch application. Therefore, in the existing literature, there are no proposals that lead to a general use for a long period. Our proposal therefore has the objective of choosing some of the tools available, designed for the different abilities relating to growing ages but sufficiently similar to allow the development of a fluid and continuous coding educational path. New complexities are, in fact, gradually added along the way, following this list of concepts gradually taught, starting from 3 years of age, reaching 11 years: instruction and sequence, coding through colors and symbols, algorithm, constraints, transcoding, programming a robot, programming on tablets or devices analogues, programming a hardware device using block programming, first approaches to a real programming language.

2. Dissemination of coding at school

The spread of coding in schools around the world has been gradual. Many countries have already implemented coding as a subject in primary education programs for years, recognizing its importance, and we can see the most striking examples, summarized in table 1.

Table 1: Summary of coding dissemination in the world

Singapore	2014
France	2014
Denmark	2014
Spain	2015
UK	2015
Estonia	2015
Slovakia	2015
Philippines	2015
Australia	2015
Belgium	2016
Finland	2016
Poland	2016
Portugal	2016
United Arabian Emirates	2017
Qatar	2020
South Africa	2020
Italy	2021
Kenya	2022

Australia as early as 2015 has noticed a growing need for technology talent recognizing that the future of work is toward technology, as seen in [10]. In order to properly train its younger generation by providing the technological knowledge necessary for their future, it started early to allocate significant economic sums. To reach kindergarten and basic education on programming, the Australian government spends 64 million dollars to fund school-based STEM (Science, Technology, Engineering, and Mathematics) and early learning initiatives under the Inspiring All Australians in Digital Literacy and STEM measure. As early as 2018, coding teaching from elementary school onward is mandatory in Australia.

In Asia, on the other hand, Singapore, which adopted computer science education in 2014 as shown in [11], quickly made it a compulsory subject, as early as 2020 and giving it a lot of space in school curricula: coding, in fact, is usually done for 10 hours per week. The Singapore government, in 2017, released 3 million dollars allocated for the distribution of 100,000 coding pocket gadgets to school children before the start of compulsory coding and spent annual budget allocations for the program. Malaysia, Thailand, Vietnam, and Indonesia have already been investing economically for years in their turn for the dissemination of coding. In contrast, an Asian country that has more recently introduced programming as a subject in primary and secondary schools is the Philippines. Although the government, unlike those previously mentioned, spends much less on programming education, pupils show much interest in programming and are willing to learn programming in school. Interviews from 2015 already showed that about 97 percent of students in the Philippines were interested in learning about programming and 96 percent wanted programming to become a core subject in their schools.

South Africa was the first African country to adopt coding education at primary and secondary levels [12]. In 2020, it began by providing programs for teachers to learn how to teach programming as a first step in order to be able to pass the same knowledge on to students. On the other hand, Software Engineer turns out to be the most in-demand job in South Africa to date, and for this reason the government in South Africa is paying a lot of attention to technical education, understanding how necessary it is to equip the younger generation with relevant technical skills to adapt when it comes to future jobs. In the recent August 2022, the Kenyan government also announced the inclusion of programming as a subject in its educational curriculum for primary and secondary school pupils, becoming the second African country to adopt programming education at primary and secondary levels.

More than 90 percent of parents in the United Arabian Emirates wanted their children taught programming in schools as early as 2017, and in year 2020, about 35 percent of schools in the country have begun implementing programming courses for their students. The Arab country has immediately begun to transform the entire education system by adopting the use of e-books, robotics, and other emerging technologies in teaching and learning. Other countries have also more recently begun efforts to prepare new generations for the technological revolution that will shape future jobs, for example, Qatar since 2020 has been restructuring its education system to include programming.

In Europe, the first effort aimed at the importance of coding across a broad spectrum, is undoubtedly codeweek, [13], launched in 2013. The European Commission supports European Programming Week as part of its Digital Single Market strategy. In its Digital Education Action Plan, it especially encourages schools to join the initiative. European Programming Week is an event therefore aimed primarily at schools but not only, celebrating creativity, problem solving and collaboration through programming and other technology activities. The idea is to make programming more visible, show young people, adults and older people how to bring their ideas to life with programming, explain these skills and bring motivated people together to learn. The latest statistics regarding the event that took place in 2021 show that 4 million people from more than 80 countries around the world participated in European Programming Week. The average age of the participants was 11 years old, and 49 percent of them in 2021 were women or girls. Eighty-eight percent of European Programming Week events took place in schools, showing that efforts to strengthen teachers, during the 2021 campaign, were successful. Another experience is provided by the web site All you need is C<3DE , through which the European Coding Initiative has supported thousands of teachers across Europe in their efforts to integrate programming and coding teaching with a collection of open online courses, teaching materials, tools and lesson plans. To understand the uptake in compulsory schools it helps the JRC, which, in March 2022, published a new report [14] which examines the integration of computational thinking in compulsory schools in 29 countries, European and non-European: 18 european and 7 non-european countries have already made the teaching of coding compulsory, of the remaining 4 Denmark is carrying out a pilot initiative, while Italy, Slovenia and the Czech Republic have policies in this direction.

In fact, even Italian Parliament finally seems to have become convinced of the need and urgency, necessary requirements for law decrees, to also include coding as a basic learning skill The National Recovery and Resilience Plan(PNRR) is the plan approved in 2021 by Italy to revive its economy after the COVID-19 pandemic in order to enable the country's green and digital development. Among the measures foreseen with regard to schools can be found that, as of the school year 2025/2026, it will be mandatory in schools of all grades and levels to pursue the development of digital skills, including by fostering the learning of computer programming (coding), within the existing teachings. Albeit, quoting the text here, "with the human, instrumental and financial resources available under current legislation and in any case without or greater burdens on public finance". These resources, both human and financial, are far from substantial in Italy. In proposals regarding coding, therefore, it must also be taken into account that they should not be economically costly and should not weigh excessively on available school hours either. That is, programs must be proposed that optimize the number of hours needed with respect to the skills acquired by students, using material that is as low-cost as possible.

We have therefore seen how the importance of coding in the school curriculum is recognized worldwide and both strategic and economic plans are being implemented everywhere for its diffusion. Australia has been among the first to leave, as early as 2015. In Asia, Singapore in 2014, the Philippines in 2015 and the United Arab Emirates in 2017 started early, with a subsequent slowdown. However, since the Code for Asia project [15] was born in 2021, other countries are quickly aligning, such as Malaysia, Thailand, Vietnam and Indonesia. In Europe, the beginning was given by the codeweek in 2013, following which, between 2014 and 2017, most European countries began to promote initiatives for the diffusion of coding. Italy was among the last

to join, in 2021. The continent that moved last was Africa, where South Africa was the first to introduce coding into primary and secondary schools in 2020, followed last year by Kenya. In 2022, the AltSchool digital campus was born, with a purely technological curriculum, also attracting interest in Nigeria, Ghana, Uganda and Botswana. CodingAfrica [16] was also born in 2022, with the aim of promoting tech literacy in the rest of Africa. Our effort in trying to introduce coding already in preschool age started in 2019, with the first year of a pilot project in a kindergarten in the province of Bologna. These almost four years of proactive field experience have allowed us to already have a concrete and easily implementable solution in any school, even without any previous experience since teachers are provided with both the list of necessary materials and a series of lessons, ideas, exercises and software/hardware creations to copy or draw inspiration from. This opportunity is today very important in our territory, given the mandatory nature of coding in Italian school curricula by 2025/2026.

2.1. Gender Gap in STEM

We devote a final space for reflection to how the introduction of coding in education can also help overcome existing gender gaps.

In Iraq, for example, coding has been, since 2020, a tool used for a dual purpose: to also help the gender gap present in education. Indeed, there are still to this day both strong regional differences within the country and more widespread structural, social and cultural barriers that prevent girls from fully and equally accessing and completing their education, thus making it more difficult for them to participate in the employment sector and also in society as a whole. To address this problem, Mercy Hands for Humanitarian Aid, in partnership with Mercy Hands Europe and with support from the Canadian Fund for Local Initiatives, has implemented an innovative project to strengthen girls' technology skills in Basra through computer and coding courses, benefiting both female students and teachers. The goal of the project is to improve girls' IT and programming skills by giving them more employability while training teachers on IT and coding in public schools in Shatt al-Arab.

In Europe there is a similar gender gap related to STEM (Science, Technology, Engineering and Mathematics) subjects. In fact, as explained in [17] and [18], there is a critical gender gap in STEM areas at all levels of education and the labor market. Various research and statisticsshow that globally, women obtain 53 percent of STEM university degrees, but in the EU only 34 percent of graduates in the field are women. Moreover, in 2018, only 41 percent of EU scientists and engineers were women, and only five EU member states had more women scientists than men: Lithuania, Bulgaria, Latvia, Portugal, and Denmark. Finally, note that there is also a "gender equality paradox," whereby women are less likely to obtain STEM degrees in wealthier societies with greater gender equality, such as Finland and Sweden.

Particularly in Italy we are at the bottom of the European rankings for female participation in the digital economy and society. To make up for this, fortunately already for a number of years there have been many organizations dedicated to promoting a better image of science and technology subjects and building a real sisterhood among girls who engage in these areas. Notable among them is definitely Girls who code [19], whose mission is to change the stereotypical image of the programmer. Girls who code, instead, offers several free courses and classes on programming and women working in the technology area. Another successful initiative is Coding Girls [20], born in 2014 and supported by the U.S. Diplomatic Mission in Italy, the Ministry of Education, University and Research, Roma Capitale and Microsoft. In subsequent editions, the project has grown to shape itself as an augmented educational program to train the next generation in STEAM, but more importantly, to help young female students gain confidence in science and navigate the careers of the future. There are also Girls Code it better project clubs, which organize workshops at various secondary schools in grades I and II. Girlstart, projectCSGIRLS and Technovation girls also carry out similar operations.

3. Coding in kindergarten and primary school

There are many reasons why primary school children should be taught programming, some of which have already been discussed in the preceding paragraphs. Among first reasons is the great ability of children in being able to learn new notions quickly, which is why from an early age they are induced to learn new foreign languages. If we consider the enormous influence that technological revolution is having in the world of work, learning coding and programming language allow children to have a greater understanding, from an early age, of how computers and technology work, skills that are indispensable today in the world of work as well as in everyday life. Benefits are not only technical: an additional one comes precisely from the possibility of developing, through educational coding programs, important general social and relational skills such as working in teams. Recognizing an error in a solution process helps to understand how making mistakes can be an opportunity for improvement and collaboration to reach the solution, also in an increasingly optimized way. Specifically, on the other hand, the final learning of programming languages is a great exercise for children in learning what can be called in effect a new language. Coding has proven in our experience to be an easily applicable tool as early as kindergarten because the basics of logical thinking, already broadly understood by age 3, can be taught very simply through fun games.

As we explained earlier, coding in schools started to spread as early as 8 years old, and in recent years many tools have been proposed that can be used in programs to be carried out in school and in specific events. Those chosen for this project were decided on the basis of two basic components, which are derived from the previously quoted sentence of the PNRR:

1. Possibly they must be no cost, where not possible they must be limited cost and allow with limited purchases ample opportunity for use.

2. They must fit the relative abilities of the age of the children to whom they are proposed while trying to

create a uniform path with a smooth transition from one tool to the next more advanced one, all the way from age 3 to 11, thus minimizing the hours needed to learn coding.

We then go on to illustrate first for kindergarten and then for primary school, all proposed exercises and tools chosen for this pathway.

3.1. Constraints and choices for preschool

The most obvious constraint in preschool is that children are unable to read and write. Tools used must therefore rely on colors and symbols. The need to make lessons playful is high, especially in the early approach. Finally, although devices such as smartphones and tablets are now used from the earliest years of life, the concepts of programming and computer science as they are understood in the working world are far from the minds of children of this age. It therefore becomes essential to demonstrate how logical thinking is applicable, and often unconsciously already applied, in what they do every day. In addition, in order to minimize the hours needed for coding, the proposed exercises are integrated with other activities and topics covered in the school year.

3.2. Sequences and encodings

The first approach with 3 and 4 year old children is done using two types of cards: codyfeet free and codycolor. As explained in [21] this decision was made in order to separate the concepts of sequencing and coding so that they can be learned gradually before combining them at a later stage. Interaction with children is crucial to maintain attention and the introduction is an example of this, starting by asking them if they want to guess what coding is, then suggesting that it is related to the word code. Already this first stimulus to reasoning has always brought out the link to passwords and secret codes. Asking what secret codes they know leads to the unlock code on their parents' tablet or smartphone. This already leads to the next level where teachers notice together with children that the codes are of several different types: some parents use numbers, some use signs, some use biometric data such as fingerprint or face recognition. Also when asked what they ask to unlock these devices for, the main reasons are: viewing videos on social channels or video games. This leads to the explanation that behind both there is a code that is called programming language, which programmers use. Programming is not only needed for games and applications but it can be found everywhere and the theacher can look with the children in what they do during the day or what they would like to do when they grow up, where the programming or coding is found. The amazement and interest in seeing that it exists in everything increases their interest: if they want to fly a plane they will use programmed controls, if they want to be police officers they will certainly know the coding of road signs, when they watch television it is a program that follow one cartoon to another at the same times but with different episodes, when they are in the car with mom and dad it is the programming of a "strange card called a control unit" that makes a sound to remind parents to add fuel or put on the seat belt. Once we get their attention, we return to coding related to computers and explain the first concept, that of sequence. Computer, in fact, executes a series of instructions that are given to it one after another. We must therefore as a first step decide on the right sequence of instructions to get to what we want.

To make children understand that instruction is nothing more than an action that you want them to perform, ask them to perform some elementary actions, e.g. 'Raise your hand' 'Clap your hands' ' Do no with your head'. Finally ask 'come to me jumping like a kangaroo' then having them analyze what movements they did: they stood up then performed N kangaroo jumps forward. This is a sequence. Often, without our realizing it, we are asked to do something that requires a sequence of instructions in order to do it, and we then illustrate the cody-feet free tiles that will be used to demonstrate this. The tiles are of 3 types, as shown in the Figure 1: beginning/instruction/end, and are recognizable by the way they can fit together like puzzle pieces, forming a long line of tiles, that is, a sequence. On top of the instruction tiles is a piece of velcro and it is explained that it is used to have fun playing different games depending on what you stick on it.

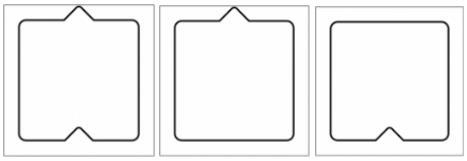

Figure 1: CodeFeetFree Cards

We can then play the first game with these tiles to explain the importance of putting the instructions in the right order in a sequence. We tell the children that their parents woke up particularly sleepy and have to get them dressed for school. As we tell in what order the clothes are put on, we create the sequence by sticking the clothes on the instruction velcro. The sequence will obviously be wrong, and we show at the end a drawing of how the parents would have sent them to school: with the underwear over the pants,the tank top over the vest, and the socks over the shoes! What should the correct sequence look like? By reasoning aloud with the group of children, they independently manage to dress correctly.

The second proposed game, on the other hand, combines coding and movement: each tile represents a movement to do, as shown in the Figure 2. Education tiles with the movements on them are made available to the children. They take turns choosing their favorite movement and, putting them in sequence, create a dance to try all together. This type of exercise can also be proposed during motor skills hours by changing the symbols as desired to create motor pathways for the children.

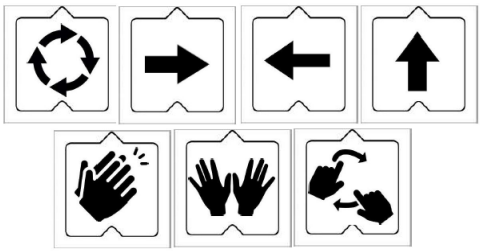

Figure 2: Dance with CodyFeetFree Cards Example

All the material used so far is paper-based and can safely be printed by the school at little cost. Not only that: exercises can also be suggested to be done at home with the relevant material. In fact, at the end of the illustrated lesson, suggestions of games to be played together at home were sent to parents, complete with cards, obviously of a small size, that can be printed, cut out and used. First suggested exercises were:

1. Practice discovering the sequences of instructions hidden in what we do: try to think together with the children about a job you do often and what is the right sequence of actions. For example to set the table, brush teeth, cook pasta. You can draw the actions together on instruction tiles and put them in sequence, then have fun shuffling them around to see how much the end result changes if you don't perform them in the correct sequence: if you drained the pasta before you turned on the stove, what would happen?

2. Dance battle: each family member proposes his or her own dance using the tiles already known from the lesson at school. The others will vote to see who is the best choreographer in the family.

3. Storytelling Inventory: tiles with good characters, bad characters, places and objects are also provided. The game is to create a sequence and invent a fairy tale by following it. A similar exercise can also be offered at school in storytelling hours, preparing tiles concerning the fairy tale that will be read and asking children to put them in the right sequence to recreate the story.

In a second lesson the term coding is introduced, starting with the explanation that it is nothing more than a simple way to explain an instruction to be executed. In the previous lesson, symbols were used, as they are also used for example on devices and remote controls: square for stop and trinagle for play are already known at age 3. It is not only symbols that are used for coding of course, and one of the most frequently used ways is certainly the use of colors, which can easily be explained to them by showing them a drawing of a traffic light. Color coding will also be used in later exercises, through the new codycolor tiles, explained in [22]: these have only one color but no interlocks. Tiles are large so they can walk on them. On gray they will step forward, on red they will turn to the right and step forward, and on yellow they will turn to the left and step forward. Since the concept of left and right is not yet clear to all children, 2 yellow and red bracelets are used to help them i understand which hand to turn toward. A first volunteer child is then sought to put the bracelets on and an object is placed on the floor: the other children in turn will have to choose the right tiles for the child to reach for the object. The teacher places the first one and the child stands on it, then it is decided where to spot in the room to go further, and which tile needs to be placed in front of the child's feet to reach the goal. The paths to take can be guided by imagination or by stories read at school, for example, one can imagine that the dragon is coming and a princess-doll needs to be collected and then placed safely in a container-castle. A variation may be, instead, to place an object in the center of the room, arrange some obstacles on the floor and form two teams that, starting from opposite corners of the room, will have to try to reach the object first.

The third and final lesson introduces the chessboard into the games. A large 5x5 chessboard is placed on the floor and they try to create a path together like the ones on the floor, but this time putting the tiles on the squares of the board. The first tile is placed by the teacher. When the path comes out of the chessboard then the teacher introduces the last two new tiles to the children, namely the start and end tiles of the path by putting the triangle before the first tile of the path and the circle at the end. Start and end now are put in retrospect, the following year they will instead become constraints to be respected: they will be set at the beginning and the children will have to make sure to create a path between the two. All the necessary information and tools for the next level have then been given.

The experience with 3 and 4-year-olds has been carried out with 3 different classes of children year after year. Every year it has been confirmed that through continuous interaction, playful exercises, and the presence of movement, the children's attention and interest always manages to remain high, often with requests from them to extend the lesson. After the first trial year, in which many parents asked for information following their children's enthusiastic stories, explanations and exercises that could be done at home were introduced. Obviously interest is subjective, but over the next two years 30 percent of parents shared with us fairy tales and dances invented together with their children or fun times when dad shaved his beard following the sequence specially scrambled by his own child. We initially expected this feedback to come from families in which at least one of both parents worked in the computer field or similar, and had clear concepts related to coding. Instead, we feel it is important to make explicit that the feedback was always related solely to the child's interest, and the parents who participated often asked for additional guidance because, completely distanced from the concept of coding and programming in their own work, they found themselves intrigued in turn.

3.3. Consolidation of acquired knowledge and introduction of Constraints and cycles

This second part is offered to 5 and 6-year-olds. In the first lesson we start immediately with the chessboard but using tiles that combine the two concepts of sequence and coding, cody-feet cards. As explained in [21] and [23] these have both point and wedge shapes to fit together like a puzzle,

and color, with the same coding as the codycolor tiles. The main change from the previous level of difficulty is that you do not create the path before, but during. The start and end tiles are placed a priori, then 3 children proceed at a time, each with a specific task: one of them will be the executor, that is, he will only follow the instructions given to him; one will be the analyst, who will decide on the sequence of instructions to achieve the goal; and one will be the programmer, who, starting from the solution identified by the analyst, translates it into code to be executed. As much as this process has always been followed easily and naturally by children, we can see how complex it is: it involves doing teamwork while managing to keep each person in his or her specific role.

The first challenge to propose is to try to get the children to create an increasingly shorter path, until they achieve the shortest possible one. This explains the importance in achieving a goal with as few instructions as possible: it is achieved in less time and the performer gets less tired. The link to scheduling in the business world is becoming more and more evident, but there are still generalizable motivations: for example, planning better for a long journey allows one to arrive at the destination sooner, less tired and spending less money on fuel. Some planning concepts can then be explained through simple games, such as creating a tree of depth 5 by keeping the same starting point but, by choosing tiles from a predefined set, reaching different end points. By drawing each path on an A4 sheet of paper with the 5x5 checkerboard reproduced and going over it on tissue paper, the various paths found, when overlaid, will show just such a tree. Again, is sent a document to their parents, with an explanation of what was done, together with tiles and chessboards that they can print out to play with their children on A4 sheets. Some challenges proposed for home, and shown in Figure 3,are a first approach to the concept of constraint, which will later be explained at school. Using a 3x3 chessboard, how could it be filled using any tiles? How could it be filled if one had no yellow tiles? What if one had only two grays?

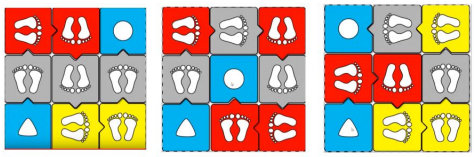

Figure 3: First Approach to Constraints Excercises

The concept of constraint in class is introduced with a google maps satellite map where they can see their school from above. Different groups of children are asked to draw the route they would take to reach a house nearby where they are having a big party. They may choose different routes, but instinctively they will follow the constraint of only being able to drive a car on one road: none of them go through a farmer's field or a park, even though doing so might take a shorter route! During exercises on the chessboard, constraints will be obstacles of various kinds to be avoided in order to reach the goal. Again, they develop concepts through themes they are dealing with in school or by linking them to a particular time of year. In this three-year project, for example, a game was proposed where the child had to program a robot to make it clean a room of garbage by managing a separate collection of items: paper, plastic and organic. As shown in the Figure 4 in order to clean up everything it will be necessary to start with one type of objects and then move on to the others.

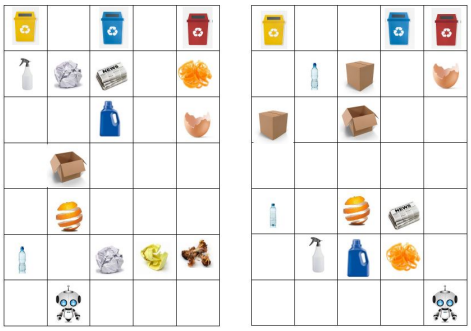

Figure 4: Garbage Collection Excercise

Both at school and at home, collection challenges have been proposed: the Easter Bunny's Easter egg collection, shown in Figure 5 where the child has to impersonate the rabbit and return to his burrow trying to put as many eggs in his basket as possible. Similarly, for the holiday season, a tale of a blizzard has been proposed at home, that dropped a lot of presents from Santa's sleigh, which he must now try to retrieve while being careful not to fly into high buildings.

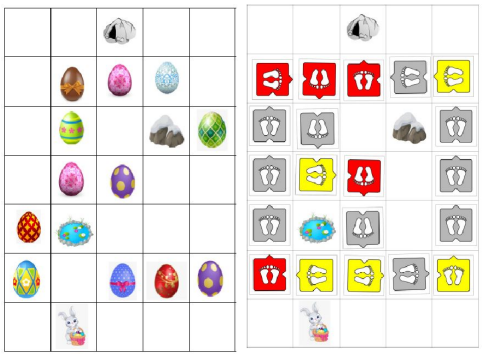

Figure 5: Easter Game Example with Solution

During the third year of the piloting of this program, the 2021/2022 school year, one section of the school was following a parallel experiment in which the section teacher, from the morning reception until the end of lunch, communicated with children only in english language. We then took advantage of this for a joint, more in-depth lesson on exercises related to one of the cycles of programming. The english teacher previously explained the key words IF THEN ELSE to the children. Next, taking a cue from the story of a bee that served as a thread for the various exercises on the board, children are divided into 3 groups. In the first group the bee, which cannot fly because it has injured a wing, will have to reach the hive while avoiding lakes and stones. In the second group the bee will be given a bulldozer and will be able to go over the boxes with stones. In the third group the bee with the boat will be able to cross the lakes. At the end of each of the three paths, the tiles

used on the small checkerboards on the A4s are redrawn and overlaid on tissue paper. This time overlaying the 3 tissue papers will show the decision tree. The first real pseudocode is then made for the children to write. An A4 has already written IF <drawing bulldozer> then <free space> ELSE IF <drawing boat> THEN <free space> ELSE <free space>. In the three free spaces, the children will draw the sequence of colors they used in the three paths. This simple pseudocode got them so excited thinking they were becoming real programmers that they wanted to do it again, with the results shown in the Figure 6.

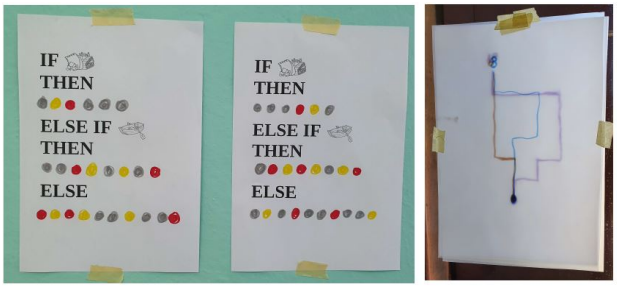

Figure 6: If Then Else Pseudocoding Example

This second phase of teaching tended to have less feedback at home: 20 percent of parents reported examples of exercises done with the children. In contrast, there was a noticeable increase during lockdown periods when schools were not open: coding was gladly exploited at home to teach in a playful way. During these periods as many as 60 percent of parents shared paths, challenges, and even proposed new ideas followed by other parents in the school. The simplicity of these tools that can be used to stimulate computational thinking was certainly confirmed by showing how it is accessible to everyone and easily transformed into fun games that can be proposed even to very young children.

4. First approaches to technology

One of the main tools of coding is visual or block programming: this type of programming offers an intuitive approach, reducing syntactic rules to simple interlocking between blocks of complementary shape. In short: the program code does not have to be typed. Even for children as young as 5 or 6, who still need to learn to read and write, visual programming thus allows them to immediately experience the effect produced by the colored blocks on the characters, called sprites, that animate the story or game being created. As children play and invent stories, they have to work hard to figure out which colored blocks to choose and fit together to recreate what they have in mind. As they do this, they unknowingly write lines of computer code. Block programming is shown as a first approach to technology.

4.1. Block coding Example

Harking back to the previous lesson, we begin by showing, on old smartphones lent by parents and given to various groups of children, a game in which, using exactly the pseudocode written with them in lesson three, the bee makes the three different paths. The mblock platform was used for this purpose, as it easily allows one to create ad-hoc blocks and visually reproduce exactly the sequence that the children see on the sheets they filled in. In figure 7 we can see on the upper part the game and on the lower the code, with the color of the instructions that the children had chosen in the different paths.

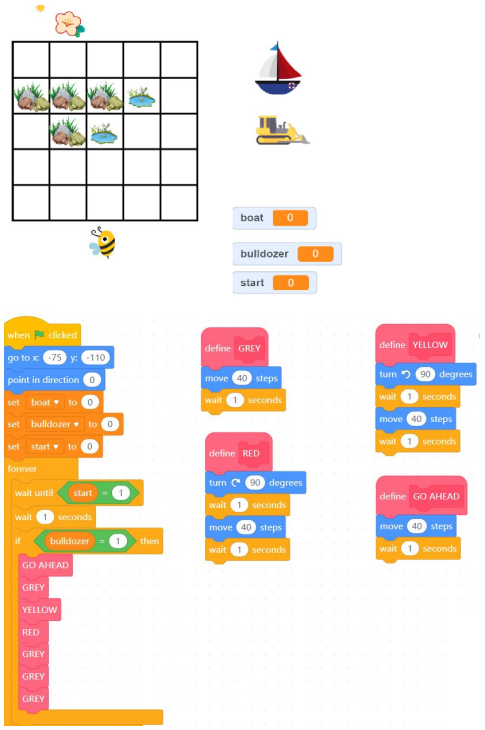

Figure 7: If Then Else Implementation with mBlock

4.2. Programming a robot

Certainly finishing kindergarten by programming a real robot was the winning choice. As explained extensively in [21], it is explained that depending on the object we want to program the coding to be used changes. In fact there are so many programming languages and often a programmer is faced with the need to transcode one of his sequences. Transcoding is necessary to transform the paths taken on the large chessboard into a sequence of keys to be pressed on a robot to make it take the same path on a chessboard of a suitable size for the length of the robot's step. The focal point of teaching at this last stage is actually the error. Easily children can make mistakes in the sequence on the robot, and at the first mistake you will cheer up the child by letting him know when the mistake is important to understand in order to correct and improve the path-it is nothing more than the debugging that every programmer does for a very high percentage of his working time! It is important to look very carefully at each step performed by the robot: when does it not do what we expect? That is exactly where there is the error that needs to be fixed!

It is also very interesting to introduce voluntary errors at 3 different points in the same sequence: at the beginning, towards the middle and almost at the end. The earlier we

introduce an error the farther it will take us from the desired result.

Fortunately, also the robot was chosen not only for its easy transcoding but also for its decidedly low cost: this not only helped the school but also the parents. In fact, each year 25 to 30 percent of the seniors were so proud of the programming achievement that they asked their parents for the robot they used at school as an end-of-school gift.

5. Primary School

By primary school, manual dexterity has markedly improved, and all children are already familiar with smarthphones, tablets, and interactive whiteboards at school. Until they are yet able to read and write, a less paper-based and more technological approach to coding can still be given through online applications.

Despite this, a first approach that summarized the previously assimilated concepts proved to be effective. We had further confirmation of the easy adaptability of the tools chosen in this case as well: a specific school, for example, had chosen a ship of little pirates as the theme that would also act as a leitmotif in the textbooks. During the first year it was easy to organize courses related to this area, for example:

- drive the ship between rocks and sea monsters making it arrive at the treasure island
- follow a map and reach the treasure by passing through marked key points

Adding motor skills is still important especially in the first year, therefore both courses of this type and exercises via applications on tablets or interactive whiteboards have been proposed, in parallel.

Scratch is a programming language developed by MIT (Massachusetts Institute of Technology) and made freely available. It is a block programming environment used for coding that aids in logical reasoning. In this environment there is no need to type any lines of code, but simply drag and drop blocks. The block system allows the implementation of a series of commands by simply arranging the blocks in a particular order. Each block corresponds to a command and they are executed in the order in which they were placed, from top to bottom. These features make Scratch undoubtedly one of the most popular programming languages for children.

Scratchjr is a declination dedicated to younger children and can therefore be approached in first grade. In this case blocks do not have written description of the related instruction but explain it with an intuitive picture. Available blocks are divided into 6 groups according to their functions:

1. the yellow group contains all the possibilities for starting a sequence: when you press start, when you touch a character, when a message comes to the character, and so on;
2. the blue group contains all the movements the character can do;
3. the purple group allows you to perform certain actions such as making the character talk, making him zoom in or out, making him disappear or appear;
4. the orange group contains cycles and timings;
5. the red group only indicates whether the sequence at the end should be repeated in a loop or not.

During the first year ScratchJR is then used to learn how to use blocks in sequence.

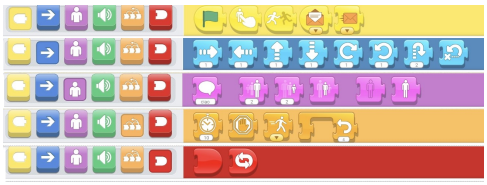

Figure 8: ScratchJR Blocks

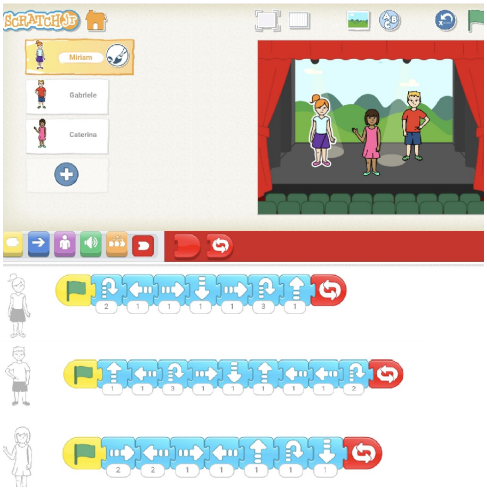

Figure 9: School Dance with ScratchJR

Exercises are aimed at recognizing the groups of blocks and using them creatively. The first one is, in fact, dedicated to the blue group that will lead the babies to organize the school dance. Each child chooses his character and his 10 moves. An example of an implementation performed by 3 children can be seen in Figure 9. Then, exploring the orange block, students can play a game of guessing who comes first among the chosen animals. One child in turn will choose 3 different animals, decide the speed of each one to the other children will decide which animal to bet on. In the example proposed in Figure 10 the child had the heavy elephant come first, the fast zebra second and the lazy piglet third. The level of difficulty is increased in subsequent lessons by proposing the creation of real stories. For example, this school year's class decided to fulfill the math and science teacher's dream by arranging for her to travel to the moon. As can be seen from the Figure 11 sequence, movement, action and timing blocks were used.

As a last exercise they are stimulated in thinking of a more complex story with at least three characters interacting with each other. For example, one class proposed the following story: the child comes out of school and his mother takes

him to his best friend's house to play together. Together with the teacher, the children first have to identify interactions, that is, when one character's action starts another character's sequence. In this case, the following interactions were identified:

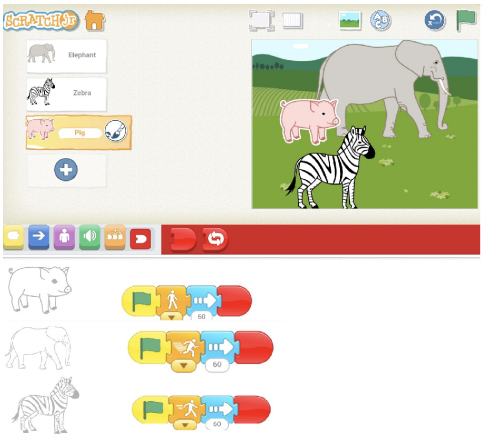

Figure 10: Race with ScratchJR

Figure 11: Travel to the Moon with ScratchJR

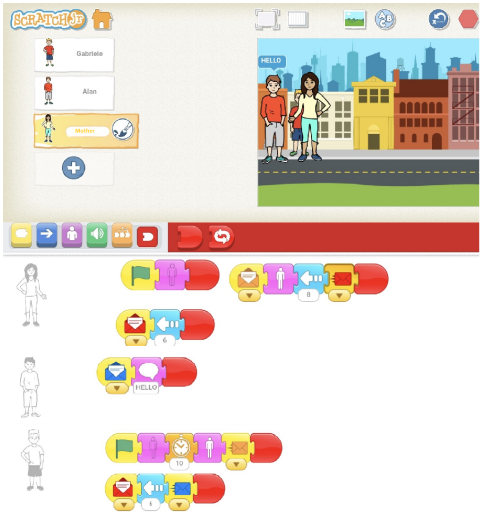

Figure 12: More Complex Interactions with ScratchJR

1. when the child appears outside school mom walks from home to school.
2. when mom comes to the child they go together to his friend's house, that is both child and mother start walking.
3. when they arrive the friend immediately says "hello".

In Figure 12 we see the implementation on scratchJR of the story. All interactions are expressed by sending colored letters, the receipt of which is the start of another sequence.

5.1. Deepening of block programming

With reading and writing skills established, the mBlock platform can be used in third and fourth grade. Makeblock, or mBlock [24], starts with scratch 3.0 and expands its opportunities by tying it to different types of hardware devices and the language C code hidden behind each block. Children can then approach this transitional version between the block world and real programming code. As on Scratch, blocks on mBlock also have the description of the corresponding instruction written in text. The first exercises proposed are similar to those done on scratch,for example implementing again the trip to the moon with mBlock. If the school where this program is being followed is in the same territory as the preschool and therefore there are many children who have followed the previous path, it is interesting to analyze with the newly acquired skills, the coding that the teacher had written for the exercise on the if,then,else cycle. The groups of available blocks, compared to scratchJR, obviously have many more instructions in them, and groups of operators and sound actions are added. The connection with programming languages is evident in the group of loops in which we find all the most frequently used ones.

5.2. Exercises with advanced circuits

Fourth grade concludes with an actual project, including hardware. The harware used can be an Arduino UNO or Elegoo UNO, there are kits for both including LEDs, sensors, displays and many other devices that can satisfy the children's creativity, at a very low cost, such that a school can buy enough of them for children to work on in small groups. The project involves developing a kit that lights up the Italian flag and plays the anthem at the push of a button and was explained specifically in [25].

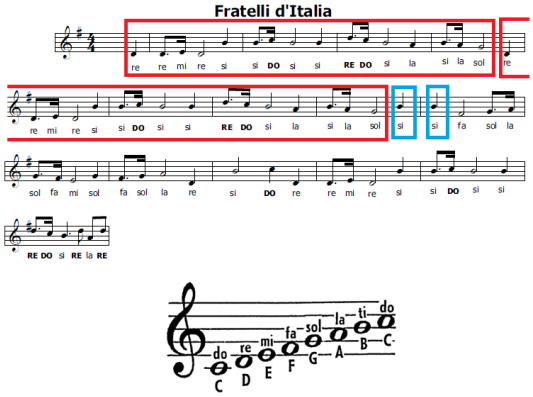

Figure 13: Looking for Repetitions and Notation Transformation

It first involves the music teacher, with whom the score of the anthem is analyzed looking for repetitions of groups of notes, as shown in Figure 13. Next comes a thorough understanding of the notes and their encoding in Anglo-Saxon notation, necessary for programming, through the transformation table in the same figure.

In Figure 14 we see the final real implementation of the project, thoroughly explained in [25]. Programming the hardware is done precisely through mBlock, already widely used by children. The Arduino UNO device is simply added to the software, and by connecting the Arduino to the PC, the program written through the blocks is loaded to it.

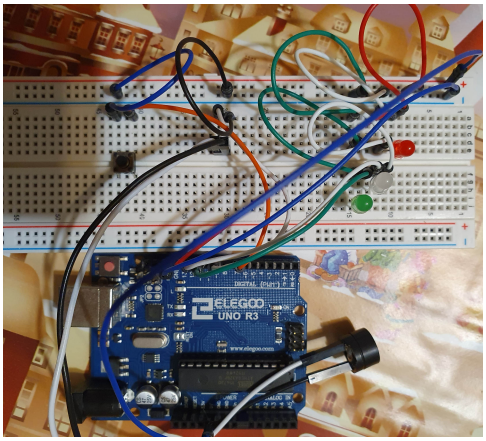

Figure 14: Real Implementation of the Anthem Project

Fifth grade is dedicated to creativity. The goal is to collaborate on the creation of an interactive landscape. The first part is devoted to analyzing what kits provided to the school include, getting inspired. The various proposals are then evaluated together, both from the point of view of feasibility and difficulty, deciding the number of people of each team that will develop the chosen ones. This project is still developing, enthusiasm is high, and many ideas have emerged, from which some choices have been made according to feasibility.

The simplest projects made by groups of two children will be as follows: The starry sky: several white and yellow LEDs with different on and off timings will be placed behind a blue veil, giving the idea of stars shining. A light sensor will be placed upstream so that the led circuit will be activated only if the sensor is in the dark or is dimmed. Design drawing and first draft of programming are respectively on the left and right side in Figure 15

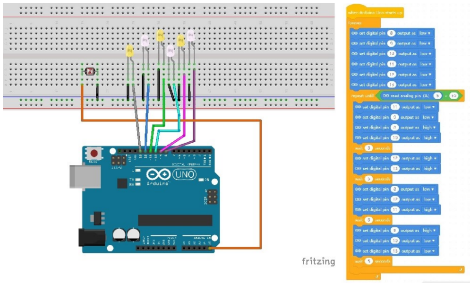

Figure 15: Starry Sky Project

Traffic management at an intersection: programming two traffic lights by reasoning about the timings needed to operate the 'intersection properly without causing accidents. The design drawing is in the left side of Figure 16, and, in the draft scheduling on the right part of the same figure, we see the timings of the two traffic lights studied through some simulations to see if they were suitable for cars to pass through without accidents.

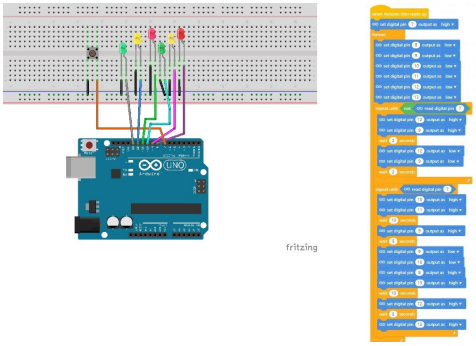

Figure 16: Traffic Lights Project

More complicated projects were then chosen to be carried out by larger groups. A group of three children will design an advertising panel in which LEDs are turned on in different ways to show different figures every minute. The main work was to understand the matrix of the LEDs and, by reproducing the designs by coloring squares on a checkerboard the size of the display used, translate them into block code.In addition, it is necessary to find the correct extension for managing the led matrix through blocks. On the left side of Figure 17 we see the design and, in Figure 18 on the right two examples of programming to make the image of a heart and an up arrow.

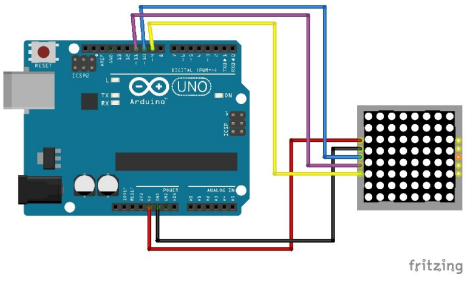

Figure 17: Advertising Panel Project

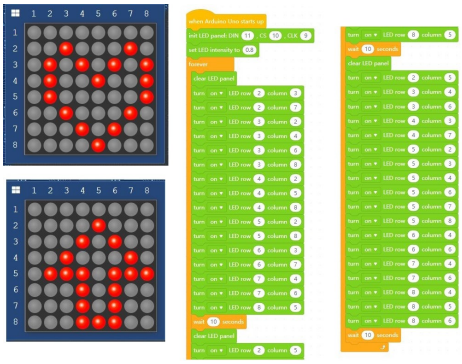

Figure 18: Matrix Programming Examples

Groups of four children will work on projects concerning the school that will be included in the landscape. The first is an automatic cooling system based on a temperature sensor that when a given threshold is exceeded activates a fan. In addition to finding the right extension to have suitable blocks to handle the sensor, this project was assigned to the children who had shown more interest in the electronic part. In fact, of course with great help from the teacher, it was necessary to include other elements: a resistor, a diode and a transistor. In Figure 19 on the left we see the design and on the right the block programming. The second is a pad with security code for school entry. For ease of resolution, the code is a single digit. If the correct digit is pressed, the green LED lights up, otherwise the red one.

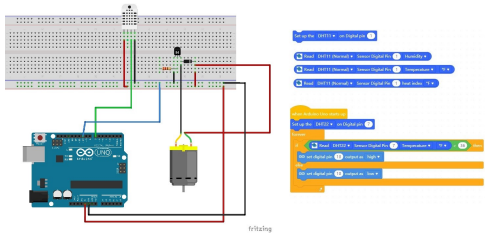

Figure 19: Cooling System Project

The main difficulty is connecting the pad and initializing it on mBlock, as well as having to find the correct extension to handle the pad here as well. Again we can see in Figure 20 on the left the design and on the right the programming.

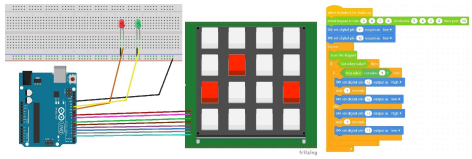

Figure 20: Security Code Pad Example

The low-cost kit proposed to the school contains many other devices not used here but which can lead to further projects that can be implemented in a primary school but with a great final functional impact. For example, it could be possible to automatically turn on the external lights of the school that is part of the project, using a proximity sensor. The cardboard model representing the school is equipped with some LEDs connected to a proximity sensor and an ultrasonic sensor is placed on the ground. When you place a hand at a predefined distance or less, all LEDs will light up for a certain number of seconds. The circuit diagram is shown in Figure 21. In Figure 22 we see a last example of a level crossing. In an infinite loop, for example every 5 minutes, the servomotor is activated so that the rod attached to it, colored to remember the level crossing, moves 90 degrees centigrade to close going into a horizontal position, lighting up the red LED parallel to the start of the movement. After 1 minute the servomotor will move 90 degrees in the opposite direction, so as to reopen returning to the vertical position, and finally the LED will turn off.

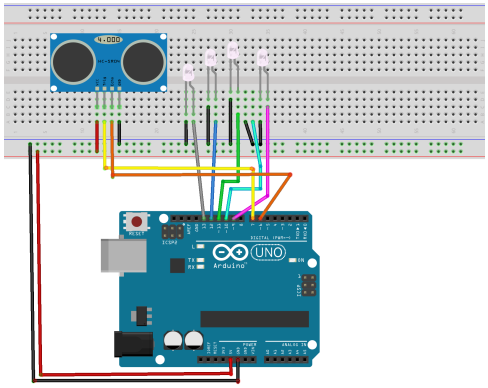

Figure 21: Switching on of Lights with Proximity Example

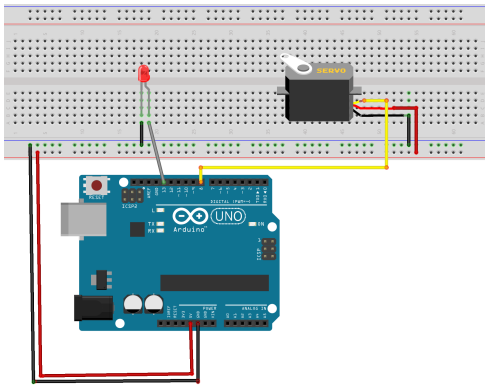

Figure 22: Level Crossing Example

The final step will be to show the programming code related to the blocks they used on mBlock and try to analyze it together, so as to have a first approach to the actual programming. The code in C language related to the hymn implementation blocks, being very simple and repetitive, lends itself well to being a first approach in which the structure of a program written in code is studied. As can be seen from the extract of Figure 23, the following can be highlighted: the inclusions of external libraries in the head of the file, the subdivision into functions, the initial setting of the variables, the main while and for loops,several calls to functions which will give an output relative to the variables that are passed as input. A more advanced analysis can instead be carried out thanks to the code relating to the exercise on the security access code. In Figure 24 students could, first of all, understand how the Pad matrix and its configuration on the hardware side are instantiated, so as to be able to uniquely encode each key pressed. The final challenge that can be proposed is to analyze the code to find the right point, shown in Figure 25, which must be changed to change the security code that should be pressed on the Pad.

Arduino C

```
#include <Arduino.h>
#include <Wire.h>
#include <SoftwareSerial.h>

float button = 0;

void _delay(float seconds) {
  long endTime = millis() + seconds * 1000;
  while(millis() < endTime) _loop();
}

void setup() {
  pinMode(7,OUTPUT);
  pinMode(7,INPUT);
  pinMode(11,OUTPUT);
  pinMode(12,OUTPUT);
  pinMode(13,OUTPUT);
  pinMode(9,OUTPUT);
  while(1) {
      digitalWrite(7,1);
      while(!(!(digitalRead(7))))
      {
        _loop();
      }
      digitalWrite(11,1);
      digitalWrite(12,1);
      digitalWrite(13,1);
      for(int count=0;count<1;count++){
          tone(9,294,0.5*1000);
          delay(0.5*1000);
```

Figure 23: Analysis of the C Language of the Hymn

```
const byte ROWS = 4; //four rows
const byte COLS = 4; //four columns
char keys[ROWS][COLS] = {
  {'1','2','3','A'},
  {'4','5','6','B'},
  {'7','8','9','C'},
  {'*','0','#','D'}
};
byte rowPins[ROWS] = { 9, 8, 7, 6}; //connect to the row pinouts of the keypad
byte colPins[COLS] = { 5, 4, 3, 2}; //connect to the column pinouts of the keypad

Keypad keypad = Keypad( makeKeymap(keys), rowPins, colPins, ROWS, COLS );
```

Figure 24: Analysis of the Pad implementation in C

```
while(1) {
    key_value = keypad.getKey();
    if(key_value != NO_KEY){
        if(String(key_value).indexOf(String("5")) > -1){
            digitalWrite(12,1);
            _delay(1);
            digitalWrite(12,0);

        }else{
            digitalWrite(11,1);
            _delay(1);
            digitalWrite(11,0);

        }
    }
```

Figure 25: Modification of the Security Code

Each group will have its own degree of interest and understanding of the programming code, so it will be decided from time to time what type of analysis to propose in terms of difficulty and depth.

6. Conclusion

Economic investments and initiatives aimed at introducing coding into schools are present, as we have seen, in all continents: the first started in 2014, the last in 2020 but the diffusion is now widespread and is slowly becoming even mandatory in most part of the countries. In Europe, Italy is one of the last to have confirmed its willingness to invest in this topic at a political level and our aim was to move proactively with the first pilot projects to arrive at a complete and structured proposal, easily applicable in every school, in the moment in which the obligation would also arrive in our country. To our great satisfaction we have reached our goal early: compulsory education in Italy will start from the 2025/2026 school year. This work, which continues to be ongoing and expanding to this day, has always aimed to define a project that can be used in schools to incorporate coding into the school curriculum. The proposed solution respects the need to have neither additional funding nor impacts on current hours and human resources, constraints imposed by the Italian government. The choices made, in fact, took into account many key aspects that we summarize here. First of all, continuity: the main idea is a program that starts in kindergarten with 3-year-old children and continues until the end of compulsory schooling. It can be seen that all tools chosen for teaching are: on the one hand optimized for the age-appropriate skills of the students, and on the other they have a common thread that leads naturally to the next level of learning, which always has points in common with what was used previously.

The concepts of instruction, its coding through colors or symbols, the algorithm seen as a sequence of instructions that allows to reach an objective are proposed first to 3 and 4 year olds. The computational complexity of proposed exercises already increases in children aged 5 and over by adding constraints, transcoding and using symbols for programming a robot. The change of school at the age of 6 starts with a recovery of the concepts seen, through exercises on the chessboard, however combining them with their implementation of increasing complexity, on applications for touch devices, thus introducing block programming. This same block programming is finally linked to the hardware, to give life to ever-changing projects, which start from the ideas of the children themselves, and lead to the creation of something functional and interactive. The discovery of the C language code hidden behind the blocks, its analysis and modification allows students to reach the end of primary school with a suitable preparation for learning real programming languages.

Although this work is focused on kindergarten and elementary school, the same tools can be used to continue teaching in the following years, moving to the actual programming language, analyzing it, testing it, fixing it and gradually abandoning the use of the block language.

These 3 years of experimentation have brought various results:

- student interest has always been high throughout the process;
- a minimization of hours needed has been obtained, since all the necessary basic skills are already present at each change in difficulty level and teaching is dedicated only to novelty in the strict sense;
- even in the primaries in which the project was not implemented, there was evidence of how the basic notions of coding already received in kindergarten had made it possible to immediately move on to more advanced coding;
- the appropriateness of the increasing difficulty pro-

posed in parallel with increasing age was confirmed by the students' understanding;

- paper instruments or electronic devices chosen, have always been tools with a negligible cost and easily accessible by public schools;
- it was possible to consolidate a list of already tested materials to offer to schools, accompanied by lessons, course examples, application exercises and examples of Arduino hardware that can be replicated or used only as a starting point.

LepidaScpA, among all its objectives for the citizens of the Emilia Romagna region, has always had computer literacy, aimed at schools but not only, and ensuring all schools broadband connectivity to better exploit the current media technologies. The channel of communication towards schools and citizens is therefore already open on these topics and this has led us to think, as a work for the near future, of providing a portal for the exchange of information on the topic of coding. Sharing is thought of at different levels:

- publicity of training and information events by the municipalities;
- sharing initiatives in schools;
- exchange of ideas, material and examples between the teachers of the schools themselves, making the material deriving from our training proposal available first;
- exchange of information and clarifications with the students' parents, trying to involve them in the process;
- exchange of ideas, projects and collaborations between students, both from the same school and from different schools.

The basic concept of the portal is that the exchange of opinions and ideas, in this first phase of the approach that Italy is having towards coding, leads to an increase in interest with a consequent natural generalized enrichment among participants, making it a facilitating tool in the diffusion of teaching coding in schools of Emilia Romagna Region. In the hope that it will be an inspiration for the whole national territory.

Conflict of Interest All authors declare that they have no conflicts of interest.

Acknowledgment Author Elisa Benetti thanks Lepida ScpA and Gianluca Mazzini, as general manager of the company, for giving her the opportunity to teach firsthand the coding program proposed here in a pilot school. The authors thank the Primi Giochi preschool in San Pietro Capofiume, Bologna, for their willingness during the first three years of testing the program.

References

[1] S. Papert,*Children, Computers and powerful ideas*. New York: Basic Books, 1990, 10: 1095592.

[2] J. M. Wing, Computational thinking. *Communications of the ACM*, 2006, 49.3: 33-35.

[3] S. Boss, J. Kraus. Reinventing project-based learning: Your field guide to real-world projects in the digital age. *International Society for Technology in Education*, 2022.

[4] S. Bocconi, et al. Developing computational thinking in compulsory education. *European Commission, JRC Science for Policy Report*, 2016, 68.

[5] C. S. Iftci, A. Bildiren. The effect of coding courses on the cognitive abilities and problem-solving skills of preschool children. *Computer science education*, 2020, 30.1: 3-21.

[6] B. Arfè, T.Vardanega, L. Ronconi. The effects of coding on children's planning and inhibition skills. *Computers And Education*, 2020, 148: 103807.

[7] S. Papadakis. Apps to Promote Computational Thinking Concepts and Coding Skills in Children of Preschool and Pre-Primary School Age. In: *Mobile Learning Applications in Early Childhood Education. IGI Global*, 2020. p. 101-121.

[8] J.M. Sáez-López, M. Román-González, E. Vázquez-Cano. Visual programming languages integrated across the curriculum in elementary school: A two year case study using Scratch in five schools. *Computers and Education*, 2016, 97: 129-141.

[9] A. Wilson, T. Hainey, T. Connoly. Evaluation of computer games developed by primary school children to gauge understanding of programming concepts. In: *European Conference on Games Based Learning. Academic Conferences InternationalLimited*, 2012. p. 549.

[10] F. Heintz, L. Mannila, T. Farnqvist. A review of models for introducing computational thinking, computer science and computing in K-12 education. In: *2016 IEEE Frontiers in Education conference (FIE). IEEE*, 2016. p. 1-9.

[11] P. Seow, et al. Educational policy and implementation of computational thinking and programming: Case study of Singapore. In: *Computational thinking education. Springer, Singapore*, 2019. p. 345-361.

[12] V. Lin, O. Shaer. Beyond the lab: Using technology toys to engage South African youth in computational thinking. In: *Proceedings of the 2016 CHI Conference Extended Abstracts on Human Factors in Computing Systems*. 2016. p. 655-661.

[13] Codeweek site, https://codeweek.eu/ Accessed: 2022-12-21

[14] S. Bocconi, et al. *Reviewing Computational Thinking in Compulsory Education. Joint Research Centre (Seville site)*, 2022.

[15] Code for Asia Site, https://www.codefor.asia/ Accessed: 2023-02-07

[16] Coding Africa Site, https://www.codingafrica.org/ Accessed: 2023-02-07

[17] A. Garcìa-Holgado, et al. Trends in studies developed in Europe focused on the gender gap in STEM. In: *Proceedings of the XX International Conference on Human Computer Interaction*. 2019. p. 1-8.

[18] A. Garcìa-Holgado, et al. Gender equality in STEM programs: a proposal to analyse the situation of a university about the gender gap. In: *2020 IEEE Global Engineering Education Conference (EDUCON). IEEE*, 2020. p. 1824-1830.

[19] Girls who Code Site, https://girlswhocode.com/ Accessed: 2022-12-21

[20] Coding Girls Site, https://www.coding-girls.com/ Accessed: 2022-12-21

[21] E. Benetti, G. Mazzini. Coding Training Proposal for Kindergarten. In: *2020 International Conference on Software, Telecommunications and Computer Networks (SoftCOM). IEEE*, 2020. p. 1-5.

[22] L. C. Klopfestein, et al. CodyColor: Design of a Massively Multiplayer Online Game to Develop Computational Thinking Skills. In: *Extended Abstracts of the Annual Symposium on Computer-Human Interaction in Play Companion Extended Abstracts*. 2019. p. 453-458

[23] A. Bogliolo. 2015. Unplugged language-neutral card games as an inclusive instrument to develop computational thinking skills. In *Proceedings of the 9th International Technology, Education and Development Conference (9th International Technology, Education and Development Conference), IATED*, Madrid, Spain, 7609–7615

[24] mBlock Site, https://mblock.makeblock.com/en-us/ Accessed: 2022-12-21

[25] E. Benetti, G. Mazzini. Coding Training Proposal from Kindergarten to High School. In: *2021 International Conference on Software, Telecommunications and Computer Networks (SoftCOM). IEEE*, 2021. p. 1-5.

ELISA BENETTI was born in Portomaggiore (FE) in 1981. In October 2006 she obtained a bachelor's degree in Computer and Automation Engineering from the University of Ferrara. In October 2008 she obtains her bachelor's degree at the same university with a thesis entitled "Efficient Systems with Authentication for Digital TV." The study of MHP services led her, from November 2008, to a one-year collaboration contract at Lepida SpA, within the LepidaTV project where she deepened her studies in the field of Digital Terrestrial and Audio/Video. In January 2009, he began his PhD in Engineering Sciences at the University of Ferrara, receiving his degree in January 2012 with a thesis entitled "Architectures, Services and Multimedia in New Media." From 2009 to the present, she has authored and coauthored some 20 publications in the areas of telecommunications, DataCenter infrastructure and coding dissemination in public schools. In January 2010 she is hired in LepidaSpA in the Ideation and Prototypes Area. Since 2014 she also started working in the Design and Development - DataCenter and Cloud area where to date she works full-time as a systems engineer.

GIANLUCA MAZZINI was born in 1968. He received his Master Degree and PhD from the University of Bologna in 1992 and 1996, respectively. Since 1996 he had a permanent position at the University of Ferrara. Since 1995 he taught courses on Telecommunications. He served as a tutor for 150 theses and supervised 15 PhD students. He published more than 280 international and peer-reviewed papers. He served has IEEE Associate Editor for 9 years. He managed 40 research projects, with the role of coordinator for 20. He served in 60 Technical Program Committee. He is an IEEE Fellow. He was member in the Scientific Council of: CNIT, GARR, Cento Technopole, Telematic Strategy for the Emilia-Romagna Region, ASTER, Iperbole Wireless for the Municipality of Bologna, Guglielmo Marconi Foundation. He served as R&D Director in LepidaSpA, as CEO in Cup2000ScpA, as member of the Board of Directors in LepidaSpA and as member ofthe Board of Directors in CNIT. Since 2009 he is CEO of LepidaSpA, reformed as LepidaScpA in 2019. Since 2019 he is a member of the Board of Directors of the Guglielmo Marconi Foundation.