



Universidade Estadual de Campinas
Instituto de Computação



Laís Vasconcellos Minchillo

Towards better tools and methodologies to teach
computational thinking to children

Na direção de melhores ferramentas e metodologias
para o ensino de pensamento computacional para
crianças

CAMPINAS
2018

Laís Vasconcellos Minchillo

**Towards better tools and methodologies to teach computational
thinking to children**

**Na direção de melhores ferramentas e metodologias para o ensino
de pensamento computacional para crianças**

Dissertação apresentada ao Instituto de
Computação da Universidade Estadual de
Campinas como parte dos requisitos para a
obtenção do título de Mestre em Ciência da
Computação.

Thesis presented to the Institute of Computing
of the University of Campinas in partial
fulfillment of the requirements for the degree of
Master in Computer Science.

Supervisor/Orientadora: Profa. Dra. Juliana Freitag Borin
Co-supervisor/Coorientador: Prof. Dr. Edson Borin

Este exemplar corresponde à versão final da
Dissertação defendida por Laís Vasconcellos
Minchillo e orientada pela Profa. Dra.
Juliana Freitag Borin.

CAMPINAS
2018

Agência(s) de fomento e nº(s) de processo(s): Não se aplica.

Ficha catalográfica
Universidade Estadual de Campinas
Biblioteca do Instituto de Matemática, Estatística e Computação Científica
Ana Regina Machado - CRB 8/5467

M661t Minchillo, Laís Vasconcellos, 1992-
Towards better tools and methodologies to teach computational thinking to children / Laís Vasconcellos Minchillo. – Campinas, SP : [s.n.], 2018.

Orientador: Juliana Freitag Borin.

Coorientador: Edson Borin.

Dissertação (mestrado) – Universidade Estadual de Campinas, Instituto de Computação.

1. Pensamento computacional - Metodologia. 2. Computadores e crianças.
3. Tecnologia educacional. I. Borin, Juliana Freitag, 1978-. II. Borin, Edson, 1979-. III. Universidade Estadual de Campinas. Instituto de Computação. IV. Título.

Informações para Biblioteca Digital

Título em outro idioma: Na direção de melhores ferramentas e metodologias para o ensino de pensamento computacional para crianças

Palavras-chave em inglês:

Computational thinking - Methodology

Computers and children

Educational technology

Área de concentração: Ciência da Computação

Titulação: Mestra em Ciência da Computação

Banca examinadora:

Juliana Freitag Borin [Orientador]

Ricardo de Oliveira Anido

Joice Lee Otsuka

Data de defesa: 13-06-2018

Programa de Pós-Graduação: Ciência da Computação



Universidade Estadual de Campinas
Instituto de Computação



Laís Vasconcellos Minchillo

**Towards better tools and methodologies to teach computational
thinking to children**

**Na direção de melhores ferramentas e metodologias para o ensino
de pensamento computacional para crianças**

Banca Examinadora:

- Profa. Dra. Juliana Freitag Borin
UNICAMP
- Prof. Dr. Ricardo de Oliveira Anido
UNICAMP
- Profa. Dra. Joice Lee Otsuka
UFSCAR

A ata da defesa com as respectivas assinaturas dos membros da banca encontra-se no processo de vida acadêmica do aluno.

Campinas, 13 de junho de 2018

Acknowledgements

I would like to thank professor Juliana Freitag Borin for her support and guidance that were fundamental in allowing me to complete this work. Thanks to professor Edson Borin for all the insights, suggestions and feedback.

I would also like to express my gratitude and thanks to my parents, family, friends, professors, colleagues and everyone who in any way helped me achieve my goals.

This gratitude is extended to the teachers and other members of Prodecad, where all of our experiments took place. Thanks to the financial support from TecSinapse.

Resumo

Pensamento computacional é uma ferramenta para resolver problemas que se aplica a todas as áreas de conhecimento. Diferente de Matemática e outras ciências, o ensino de Computação, especialmente para crianças, ainda é bastante recente. Há iniciativas ao redor do mundo para ensinar pensamento computacional e programação para crianças, entretanto, ainda não há um consenso em como fazê-lo. Nosso objetivo é contribuir na direção de melhores metodologias de ensino de pensamento computacional para crianças. Para isso, realizamos um levantamento dos conceitos de pensamento computacional mais citados por importantes fontes e suas propostas de módulos de ensino. Levantamos também recursos e ferramentas que foram desenvolvidas nos últimos anos para dar suporte ao ensino de pensamento computacional. Com base nesse estudo, desenvolvemos uma série de experimentos com um protótipo de ferramenta educacional em uma escola pública que nos permitiram observar e coletar dados sobre a forma de interação das crianças com a ferramenta e com o conhecimento que estava sendo apresentado. A partir da análise desses resultados derivamos hipóteses que poderão fomentar e/ou direcionar novas pesquisas na área.

Abstract

Computational thinking is a tool to solve problems that applies to all areas of knowledge. Unlike Mathematics and other sciences, teaching Computing, especially to children, is a very recent field. Efforts around the world aim to teach children computational thinking and programming, however, there is still no consensus on how to do it. Our goal is to contribute towards better methodologies to teach computational thinking to children. To accomplish this, we made a survey on the most cited computational thinking concepts by key sources and their proposals for teaching modules. We also reviewed resources and tools that were developed in the last few years to support computational thinking teaching. Based on this study, we developed a series of experiments with an educational tool prototype in a public school that allowed us to observe and collect data on how children interact with the tool and the knowledge being presented. From the analysis of these results we derived hypothesis that can promote and/or direct new research in the field.

List of Figures

2.1	Scratch environment	19
2.2	Code.org environment	20
2.3	Cubetto: robot, board and map	20
2.4	Wonder Workshop CleverBots	21
2.5	Sphero robots	22
3.1	Main application screen	26
3.2	Free to play setup	26
3.3	Icons displaying the connection status	27
3.4	Worlds in the application	27
3.5	Levels in the application	28
3.6	One of the challenges in the application	28
3.7	Robot used in our experiments	29
3.8	A tutorial in the application	29
3.9	A quiz in the application	30
3.10	Structure of the JSON configuration file	32
4.1	The experiment's environment	35
4.2	Grade in programming tasks versus count of wrong answers in quizzes . . .	37
4.3	Grade in programming tasks versus percentage of tutorial pages read . . .	38
4.4	Count of wrong answers in quizzes versus percentage of tutorial pages read	39
4.5	Average grade for all groups by concept	40

List of Tables

2.1	Computational thinking concepts by author	15
2.2	Computational thinking concepts by school level, SBC	16
3.1	Proposed levels and concepts	25
3.2	Programming blocks	30
3.3	Application logs	31
4.1	Questions and answers in the feedback form	41

Contents

1	Introduction	11
2	Basic concepts and related work	14
2.1	Computational thinking	14
2.2	Related work	16
2.3	Tools and resources	18
2.4	Discussion	23
3	CT educational game	25
3.1	Interface	26
3.2	Bluetooth interface	28
3.3	Activities	29
3.4	Logs	31
3.5	Configuration file	31
4	Experimental setup, methodology and results	34
4.1	A change of paradigm	34
4.2	Experimental setup and methodology	34
4.3	Results	36
4.3.1	Observations	36
4.3.2	Log analysis	37
4.3.3	Children’s feedback	40
4.4	Discussion	41
5	Conclusions and future work	42
A	List of application activities	46
B	Feedback form	57
C	Consent form	59
D	Assent form	63
E	Ethics committee	66

Chapter 1

Introduction

Knowledge transmission is as old as humanity itself. Without our ability to communicate, not only would society be impossible, but we would not be here talking about our own philosophies and what they mean to our very existence. Since we began to accumulate knowledge about Biology, Mathematics, Chemistry, Physics etc, we also got aware we should be teaching future generations all we knew so they could build upon our past observations and experiences.

For centuries that has been the case, and as time passed on, newfound subjects were added to the curricula, others were merged, and education evolved as much as we did. We now find ourselves at such a point in time where we have this new science we've been exploring for the last 50 years or so about Computers and, better still, about Computation. We have been teaching this at graduate and undergraduate level for a few decades, but teaching Computation to children is still uncharted territory.

In this scenario, computational thinking (CT) emerged, a tool to solve problems that applies to all areas of knowledge and should be taught to every child. The challenge is we don't know what is the best way to do it. The teaching of more traditional subjects has been mastered and perfected along centuries, but thinking algorithmically and strictly logically is as fresh as it can be. Finding ways to have children understand and accept this subject as much as to use it like a part of their own cognition is a matter most important in our time.

Computational thinking must not be seen as a technical skill, but rather as a way to organize thoughts and solve problems, and it is natural to consider teaching it in school, either as a separate course, or as a new tool to existing disciplines [1, 2, 3, 4, 5, 6, 7, 8, 9]. The term was made popular by Wing in 2006 [1], and she defined it as solving problems, designing systems and understanding human behaviour through concepts fundamental to computer science (CS). In many countries this skill is already part of the basic curriculum [10, 11], and it is expected that new generations have a better understanding of technology and its different applications.

Teaching computational thinking to children is not a recent idea - Seymour Papert published a paper on the subject in 1972 [12]. According to him, children learn by doing and by thinking about what they do, and innovation in teaching must bring better things to do and better ways to think. At that time, the author also claimed that computing was by far the richest innovation area for teaching. He said:

The purpose of this essay is to present a grander vision of an educational system in which technology is used not in the form of machines for processing children but as something the child himself will learn to manipulate, to extend, to apply to projects, thereby gaining a greater and more articulate mastery of the world, a sense of the power of applied knowledge and a self-confidently realistic image of himself as an intellectual agent.

Papert was one of the creators of Logo, a programming language designed to provide a fun environment for children to study and learn mathematics and programming concepts [13], as well as author of the book *Mindstorms: Children, Computers and Powerful Ideas*, in which he defends the benefits of teaching computer literacy [14].

Several countries have included computational thinking in their school curricula, as well as programming and CS concepts and other related subjects (such as logical thinking, problem solving, abstraction, planning, among others). The United Kingdom's government has included CT and CS concepts in their national curriculum, stating that a high-quality computing education equips pupils to use computational thinking and creativity to understand and change the world [10]. In the United States, ex president Obama launched an initiative to include Computer Science in the K-12¹ curriculum [11]. In Brazil, the Brazilian Computing Society (SBC) is working to include Computer Science in the national curriculum [15].

Over the past decade several authors have described what CT and CS concepts to teach and how to teach them, however, there is still no consensus on how to teach CT and how to include it in the schools. Our main goal was to contribute towards better methodologies to teach computational thinking. To do that, we divided the work in three different parts. For the first part, we made a survey on CT concepts, efforts in teaching, related work and existing tools. With insights derived from this research, we had a few ideas on how to include CT teaching in the classroom, so for the second part of the work, we developed a mobile application that paired with a robot allowed us to test such ideas. The third part consisted of a series of experiments bringing the application and robot to a school and observing the way the children interacted with them. The results of the experiments were a series of observations and a set of hypothesis.

The main contributions of this dissertation are:

- a survey on computational thinking including a list of concepts accepted by key sources, recent papers with related work, existing tools;
- a mobile educational game that enables teachers and developers the ability to create different sets of activities and programming challenges;
- a report of an experiment bringing an educational game into the classroom;
- a set of hypothesis that should be tested in order to guide the development of an effective methodology to teach computational thinking to children.

¹K-12 is the school curriculum for children aged up to 12 years old.

Part of this dissertation produced the paper by the same name that was published and presented at the *26th Workshop on Education in Computer Science (WEI) - held in conjunction with Conference of the Brazilian Computing Society (CSBC)*.

The rest of this document is organized as follows: Chapter 2 presents the computational thinking concepts that are frequently discussed by key sources, existing tools developed over the last few years and national initiatives towards including CT in the Brazilian curriculum. Chapter 3 shows the educational game we developed to be paired with a physical robot in our experiments. Chapter 4 discusses the new paradigm we tested, our experimental setup and methodology, as well as our observations, experimental results and a new set of hypothesis. Finally, Chapter 6 brings our conclusions and suggestions for future work.

Chapter 2

Basic concepts and related work

This chapter presents our survey on existing work on teaching Computational Thinking, especially to children. In Section 2.1 we discuss what important sources cite as CT concepts. Section 2.2 brings a discussion of related work. In Section 2.3 we present existing tools that are related to this work and available resources to help teach CT. Finally, Section 2.4 brings a discussion on these topics.

2.1 Computational thinking

Computational thinking is not the same as programming, but rather a skill programmers use in order to solve different problems. As a consequence, using programming as a way to teach CT is common in the literature [6, 8, 9]. In this section we enumerate the main CT concepts cited by key sources: the Computer Science Teachers Association (CSTA), a membership organization that supports and promotes the teaching of computer science; Code.org, a non-profit organization dedicated to expand access to computer science in schools and increase participation by women and underrepresented minorities. They organize the annual Hour of Code campaign and provide a curriculum for K-12 computer science and they are supported by several companies including Amazon, Facebook, Google and Microsoft; SBC, Brazilian Computing Society, a non-profit organization whose goal is to encourage research and teaching in computing. We also chose three of the most cited papers in the CT topic - Barr and Stephenson [4], Grover and Pea [16] and Brennan, K. and Resnick, M. [17] - as well as one paper well cited in Brazil - França and Amaral [5] - one of the first works in the country to discuss teaching CT in schools.

The most cited concepts are:

- Sequence: a series of individual steps.
- Algorithm: a sequence of instructions to solve a task.
- Loop: the execution of the same sequence multiple times.
- Event: an external action that triggers a command sequence.
- Conditional: making decisions based on predefined conditions.

- Debugging and testing: executing an algorithm to find errors or to validate the proposed solution.
- Problem decomposition and modularization: divide a problem in smaller ones that can be solved more easily.
- Function: a sequence of instructions one can use with a given input to execute a task, possibly generating an output and modifying the original input to better suit it's purposes.
- Nested loop and conditional: a loop within a loop, or a conditional with another condition.
- Recursion: a function that calls itself.
- Parallelism: executing more than one instruction at a time, or execute more than a sequence of instructions at a time. Parallel tasks can be independent or not.

Table 2.1 shows the list of sources and concepts discussed by them.

Table 2.1: Computational thinking concepts by author

	<i>CSTA</i>	<i>Code.org</i>	<i>SBC</i>	<i>Barr and Stephenson</i>	<i>Grover and Pea</i>	<i>Brennan and Resnick</i>	<i>França and Amaral</i>
Sequence		•	•	•	•	•	•
Algorithm	•	•	•	•	•		
Loop	•	•		•		•	•
Event	•	•			•	•	•
Conditional	•	•		•	•	•	•
Debugging	•	•	•	•	•	•	•
Test	•	•	•	•	•	•	•
Decomposition, functions, modularization	•	•	•	•	•	•	•
Nested for, nested if	•	•					
Parallelism	•			•	•		•
Recursion	•		•	•	•		•

2.2 Related work

In Brazil, SBC [15] has started to define how computing should be included (or modified) in the national curriculum. There are three main categories:

- Computational thinking: understand and use models and representation to describe information, processes and techniques to build algorithmic solutions; describe solutions through algorithms that can be executed in parts or in total by machines, as well as build computational models for complex systems; analyze problems and solutions to not only find automated solutions, but to be able to evaluate their efficiency and correctness.
- Digital world: understand how information can be described and stored; understand how information is processed by computers and the relation between hardware and software; understand how digital devices communicate with each other, how the data is transmitted and how the integrity and safety of information is guaranteed.
- Digital culture: understand the impact of the digital revolution and advances in the digital world on humanity; utilize in an efficient and critical manner tools to obtain, analyze, synthesize and communicate information of different formats and with different purposes; analyze ethical and moral questions of the digital world.

Table 2.2 shows how SBC is grouping computational thinking concepts by school level.

Table 2.2: Computational thinking concepts by school level, SBC

Level	Concepts
Preschool Ages 3 to 5	Understand a problem and identify a sequence of steps to solve it. Represent these steps in an organized manner. Create steps to solve problems related to body movement and spatial trajectories.
Elementary school Ages 6 to 10	Abstraction to describe data such as lists and graphs. Identify the abstractions needed to build steps and to define algorithms that involve daily situations around the children. Use a visual language to represent algorithms. Understand problem decomposition.
Middle school Ages 11 to 14	Use visual and native languages to represent data and processes. Formalize the concepts of data structures. Use recursion to solve problems. Build new solutions by reusing solutions to problems of different context. Relate an algorithm in visual language to code in a programming language.
High school Ages 15 to 17	Work in groups designing solutions to problems integrated in other areas of the curriculum using computers, phones and other computing machines. Compare problems and reuse solutions. Analyze algorithm's cost and efficiency and justify if a solution is feasible and adequate. Argue about algorithm's correctness. Understand the limits of computing to differentiate what can or can not be automated.

Fields et al. [18] discuss the need for more empirical work in classroom environments in order to learn the best way for teachers to integrate CT into their classroom activities. They show observations on the implementation of a 6-8 week electronic textiles unit within two high school classrooms situated within the Exploring Computer Science curriculum¹. They report the ways in which teachers brought out computational thinking through students' interactions and projects, with the most prominent aspects being: 1) strategic problem solving, 2) iteration, and 3) interfacing between abstract and tangible computation.

Buitrago Flórez et al. [8] contend that CT should be taught in elementary schools and included in every university's educational curriculum. With a focus on the development of CT skills at a young age, they analyze and discuss findings from several studies measuring the impact of teaching programming, analytical thinking and CT.

Weintrop et al. [19] face the challenge of defining computational thinking and providing a theoretical grounding for what form it should take in school science and mathematics classrooms, following the decision to include CT as a core scientific practice by the Next Generation Science Standards². They propose a definition of computational thinking for mathematics and science in the form of a taxonomy consisting of four main categories: data practices, modeling and simulation practices, computational problem practices, and systems thinking practices.

Rees et al. [20] make a literature review on the teaching of coding and computational thinking to primary aged children. In their research, in addition to published peer referenced journal articles, they also included blog posts and opinions on social media. They try to answer why we are teaching coding (and whether we should be teaching it to young children at all), how we should teach it and how to best use tangible user interfaces. They also discuss if there are still gender issues to overcome. Additionally, there are further discussions on what CT is, its concepts and how to introduce it in the classroom. Some tools and programming environments are discussed as well.

Eloy et al. [9] described an experience training teachers with the goal of promoting the practice of programming and development of CT in Brazilian public schools. In their pilot project they worked on four main areas: implementation in schools, curriculum design, teacher training and monitoring and evaluation. Their first guiding material to build the curriculum was the online platform Programa³. Teachers were included in improving this curriculum through discussion sessions and questionnaires. They had over 500 students participating in the activities, and the sessions taking place in 2016 had an average of 80% student retention.

Godinho et al. [21] present a project to introduce CT and encourage children to become technology creators. The project was recognized by SBC in 2016 for bringing computing

¹The Exploring Computer Science (ECS) initiative comprises a one-year introductory computer science curriculum with a two-year professional development sequence. The curriculum consists of six units: Human-Computer Interaction, Problem-Solving, Web Design, Introduction to Programming (Scratch), Computing and Data Analysis, and Robotics (Lego Mindstorms).

²The Next Generation Science Standards is a multi-state effort within the United States to create new education standards that are rich in content and practice, arranged in a coherent manner across disciplines and grades to provide all students an internationally benchmarked science education

³<http://programae.org.br/>

to children, teenagers and people who otherwise had little contact with the area. Their encounters included mini-courses, unplugged activities or tasks in Code.org's Hour of Code, Scratch, CodeMonkey⁴, Monster Coding⁵ or App Inventor⁶. Almost 300 students participated in their activities and through feedback questionnaires approximately 90% rated the experience as excellent or great.

Aono et al. [22] use Scratch allied with an expository methodology to teach CT to elementary school students with ages 10 and 11. The children applied the concepts they learned into a project: building a "Flappy Bird" game. Every student that participated was able to build the game successfully, but all of them needed some help from the supervisors to implement the hardest parts, like the use of variables and counters.

Silva Junior and França [23] discuss the use of existing tools in the classroom in Brazil - and their effectiveness. Nine tools were analyzed for their interaction, platform, programming language and other characteristics. The tool found to be most adequate was Portugol Studio⁷, for being available in Portuguese, appropriate for beginners and featuring a user friendly interface. It also offers features to help teachers use it in classes.

2.3 Tools and resources

This section presents some existing tools and resources that support CT teaching.

2.3.1 CSTA

CSTA, or Computer Science Teachers Organization, is an organization that supports and promotes the teaching of Computer Science. Their K-12 Computer Science Standards⁸ delineate a core set of learning objectives designed to provide the foundation for a complete computer science curriculum and its implementation at the K-12 level. The CSTA Standards:

- Introduce the fundamental concepts of computer science to all students, beginning at the elementary school level.
- Present computer science at the secondary school level in a way that can fulfill a computer science, mathematics, or science graduation credit.
- Encourage schools to offer additional secondary-level computer science courses that will allow interested students to study facets of computer science in more depth and prepare them for entry into the work force or college.
- Increase the availability of rigorous computer science for all students, especially those who are members of underrepresented groups.

⁴<https://www.playcodemonkey.com/>

⁵<http://monstercoding.com/>

⁶<http://appinventor.mit.edu/>

⁷<http://lite.acad.univali.br/portugol/>

⁸<https://sites.google.com/site/cstastandards/standards>

2.3.2 Scratch

Scratch⁹ is a free programming language and online community developed and maintained by the MIT. It is designed especially for ages 8 to 16, but it is used by people of all ages in more than 150 different countries. It is available in over 40 languages.

The ability to code computer programs is an important part of literacy in today's society. When people learn to code in Scratch, they learn important strategies for solving problems, designing projects, and communicating ideas. Figure 2.1 shows its environment.

Students are learning with Scratch at all levels (from elementary school to college) and across disciplines (such as mathematics, computer science, language arts, social studies). Educators share stories, exchange resources, ask questions, and find people on the ScratchEd website¹⁰. The MIT Scratch Team and collaborators are researching how people use and learn with Scratch¹¹.

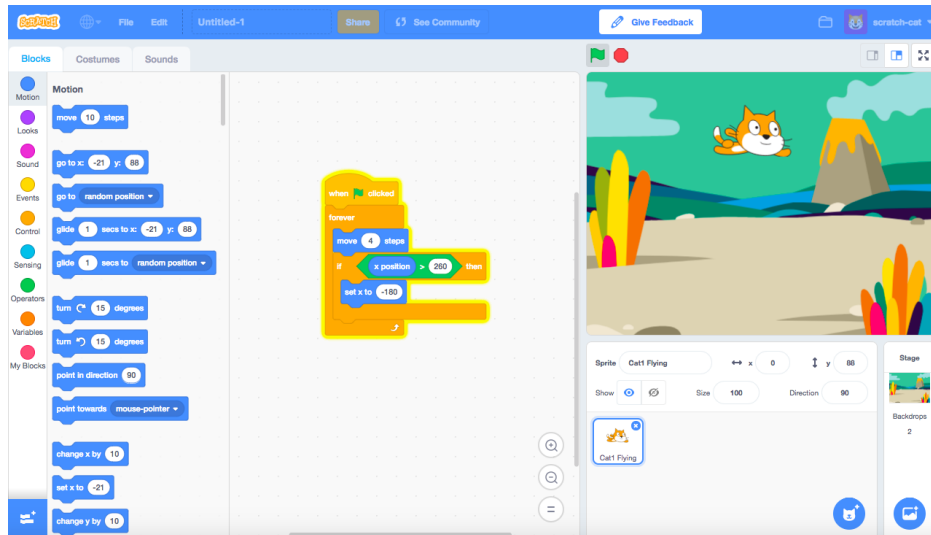


Figure 2.1: Scratch environment

2.3.3 Code.org

Code.org, as cited in the previous Section, is a nonprofit organization dedicated to expanding access to computer science in schools. They organize the Hour of Code campaign: a global movement reaching tens of millions of students in over 180 countries. In 2017 Brazil had more than 150 thousand Hour of Code events.

They also provide courses in their website for all ages, including undergraduate level, and offer block based programming, JavaScript, CSS, HTML and more. Figure 2.2 shows the environment of one activity.

⁹<https://scratch.mit.edu/>

¹⁰<http://scratched.gse.harvard.edu/>

¹¹<http://web.media.mit.edu/~mres/papers/Scratch-CACM-final.pdf>

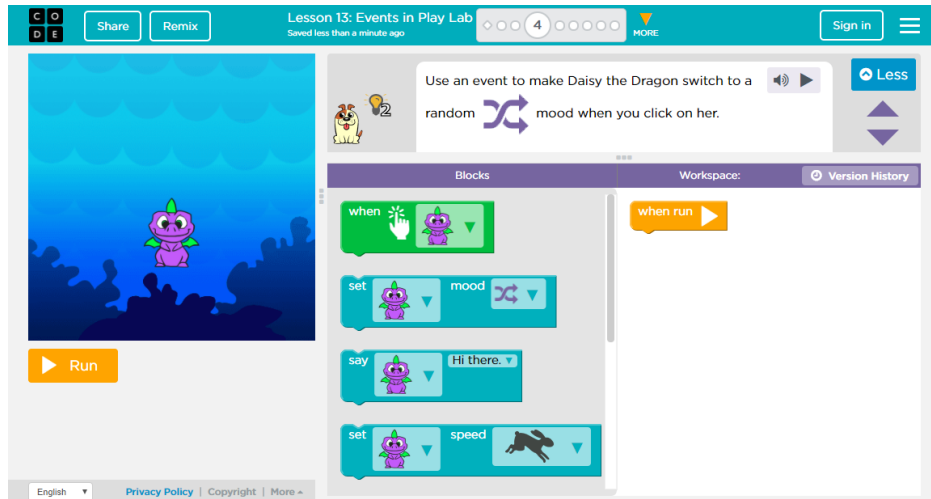


Figure 2.2: Code.org environment

2.3.4 Cubetto

Cubetto is a robot developed by Primo¹² that provides screenless coding for girls and boys aged 3 to 6. It is programmed using a board in which you can put command blocks (Figure 2.3a). They also have several different maps to be placed on a table or on the floor that show different narratives to stimulate play (Figure 2.3b shows one of such maps).

The toy introduces programming concepts, including algorithms, sequence, debugging, recursion. In their website, Primo has a bank of activities, lesson plans and other resources to help owners discover new ways to play and learn¹³.

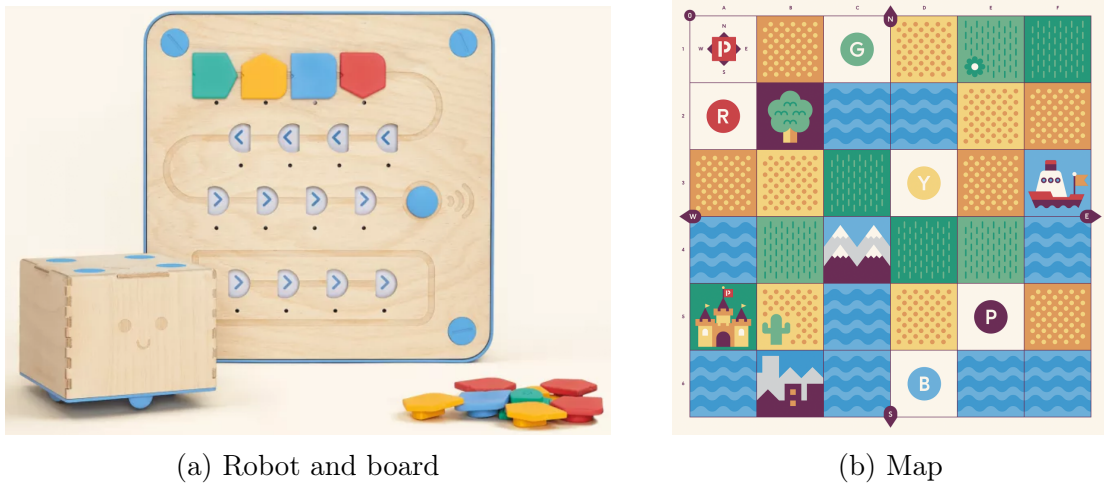


Figure 2.3: Cubetto: robot, board and map

2.3.5 Wonder Workshop CleverBots

Wonder Workshop offers three robots: each one is developed for a target age range and features different capabilities.

¹²<https://www.primotoys.com/>

¹³<https://www.primotoys.com/playroom/>



(a) Dot Creativity Kit



(b) Dash



(c) Cue

Figure 2.4: Wonder Workshop CleverBots

Dot

Dot (Figure 2.4a) is designed for children starting at age 6. It is the simplest of the three, featuring a gyroscope, voice recording and playback. Robot is programmed via block-based coding, and also has an app for pre-readers. It's creativity kit also comes with playing cards, stickers and other materials. It has hundreds of self-guided coding challenges.

Dash

Dash (Figure 2.4b) is designed to engage children of ages 6 and up. It features voice recording and playback, detects voice direction and other CleverBots, has precise motion control, has object detection and a gyroscope. It can be programmed via block-based coding, has apps for pre-readers and also has compatibility with Apple Swift Playgrounds¹⁴. It also has some optional accessories, like a xylo.

¹⁴<https://www.apple.com/swift/playgrounds/>

Cue

Cue (Figure 2.4c) is the newest Cleverbot, and is designed for children of ages 11 and up. It has a customizable personality, and offers all of Dash’s functionality and more: chatting, an accelerometer, light and volume control, faster sensors, reactive behaviors. It can also be programmed using JavaScript.

Curriculum

Wonder Workshop also provides a K-5 Curriculum to teach coding. Designed in alignment with CSTA and ISTE standards, as well as Code.org’s fundamentals, their Curriculum Pack includes several lesson plans, challenges and their solutions¹⁵.

2.3.6 Sphero

Sphero¹⁶ has a number of different robots to play with, but here we focus on the Sphero Mini (Figure 2.5a) and the SPRK+ (Figure 2.5b): they are both designed to inspire and can be used as a tool to learn to code. Sphero also offers resources for educators to use their products in the classroom¹⁷.

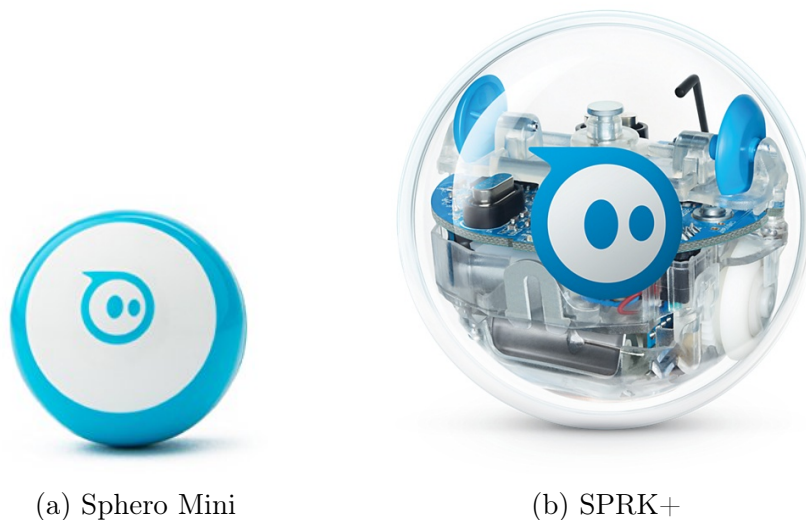


Figure 2.5: Sphero robots

Sphero Mini

The Sphero Mini packs a huge experience into a tiny color-changing robot. It features motor encoders, gyroscope and accelerometer sensors, Bluetooth LE and is compatible with Sphero Edu (iOS, Android, Kindle, Chrome) and Sphero Mini (iOS, Android) applications, that feature block-based programming.

¹⁵<https://education.makewonder.com/curriculum>

¹⁶<https://www.sphero.com/>

¹⁷<https://www.sphero.com/education>

SPRK+

This is Sphero Mini's big brother: larger, more durable and waterproof. It's designed to inspire curiosity, creativity and invention through connected play and coding. This robot offers all the features Sphero Mini does, and it is also compatible with Apple Swift Playgrounds.

2.3.7 CS Unplugged

CS Unplugged¹⁸ is a collection of free teaching material that teaches Computer Science through engaging games and puzzles that use cards, string, crayons and lots of running around. The website contains activities on 5 different topics: binary numbers, error detection and correction, kidbots, searching algorithms and sorting networks. There is also a list on curriculum integrations, relating activities to curriculum areas, such as arts, mathematics, literacy and science.

2.3.8 Exploring Computational Thinking

Exploring Computational Thinking (ECT)¹⁹, part of Google for Education, is a curated collection of lesson plans, videos, and other resources on computational thinking (CT). The site was created to provide a better understanding of CT for educators and administrators, and to support those who want to integrate CT into their own classroom content, teaching practice, and learning.

2.3.9 Tackle 3 Coding

Tackle 3 Coding²⁰ is a project that supports Primary School and teachers who want to teach computing. They provide knowledge and materials in a website of ideas and resources, as well as training and other development events.

2.4 Discussion

This survey on existing work was fundamental for our research. Several authors have described which CT and CS concepts to teach and how to teach them, and as we discussed in the previous sections, in some countries this has already been included in the national school curriculum offering every child the opportunity to learn and benefit from computational thinking.

By looking at how several important sources define computational thinking and their concepts, as well as how they were divided in teaching modules, we were able to derive our own set of concepts and decide the ones to be included in our experiments.

Moreover, we were also able to see that many countries have already started working on a way to include these concepts in their school curriculum, not as a separate subject,

¹⁸<https://csunplugged.org/>

¹⁹<https://edu.google.com/resources/programs/exploring-computational-thinking/>

²⁰<http://www.tackle3.eu/>

but as an integrated part of the already existing ones. Not only is teaching computational thinking a growing concern, it is also an unresolved task. The fact that there is no consensus on how to teach CT and how to include it in the schools, especially in Brazil, has been part of our motivation for this work.

Back in 2015 when this work started, none of the tools discussed in Section 2.3 was so robust as today. Existing robots such as the ones mentioned in this Chapter were costly and lacked a proper integration with teaching. They were therefore, very expensive toys that didn't add much. But just as we saw an opportunity to bring these physical devices to the classroom, so did these companies: today they all offer lesson plans, guided activities or even their own curriculum.

We aimed to have a low cost physical device as a motivating factor, paired with a free mobile application and a methodology to use it to teach CT concepts to children.

Chapter 3

CT educational game

We designed and implemented an Android application and a physical low cost robot to support our experiments¹. The application communicates with the robot via Bluetooth and provides a visual block-based programming interface through Google's Blockly library².

Our application is a prototype of an educational game that has six levels, each one introducing new concepts and having multiple activities within it. Table 3.1 presents the levels and the CT concepts used in each one of them. The CT concepts were selected based on the age of the children and the amount of time we would have with them.

Table 3.1: Proposed levels and concepts

Levels	Concepts
Move the robot	Sequence
Have the robot make a path of a certain shape (e.g. square)	Sequence, algorithm
Find mistakes in given algorithms	Sequence, algorithm, testing, debugging
Have the robot repeat the same tasks multiple times	Sequence, algorithm, loop, problem decomposition
Have the robot decide on what action to take based on external conditions	Sequence, algorithm, conditional
Combine repetition and conditional scenarios	Sequence, algorithm, conditional, loop

¹The robot was developed by another student in this research group.

²<https://developers.google.com/blockly/>

3.1 Interface

The application was designed to look like a regular game. The first screen has three main action buttons, as shown in Figure 3.1. The first button (*play*) takes the user to the activities. The second button (*free*) takes the user to a sandbox mode where all commands are available and there is no predefined task. It is shown in Figure 3.2. Finally, the third button (*configuration*) takes the user to the settings screen.

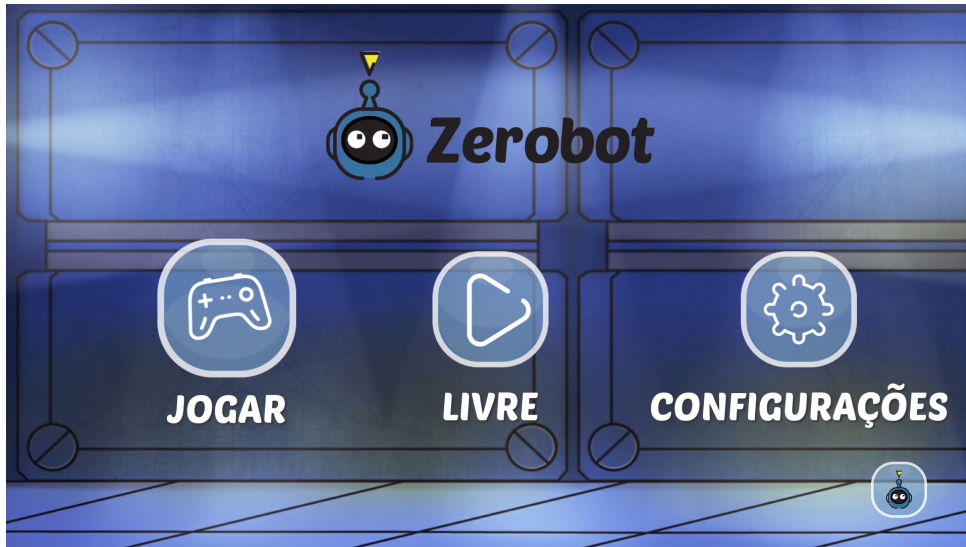


Figure 3.1: Main application screen

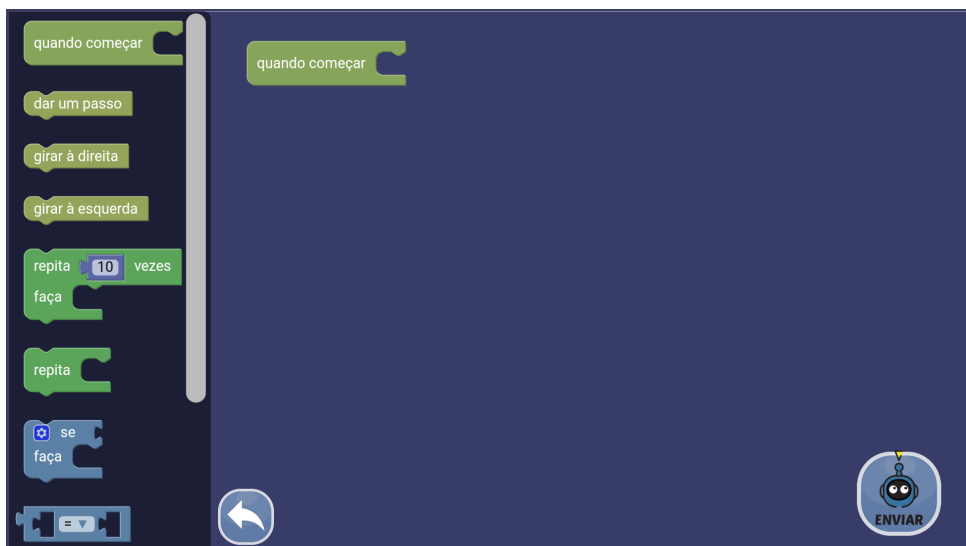


Figure 3.2: Free to play setup

In every application screen there is an icon in the lower right corner indicating the connection status with the robot: there are three different icon for the states disconnected, connecting and connected. This is shown in Figure 3.3.

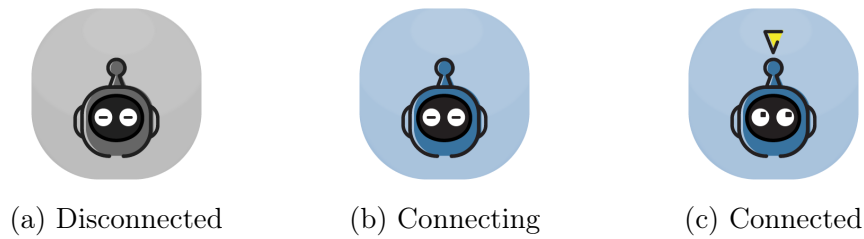


Figure 3.3: Icons displaying the connection status

Figures 3.4 and 3.5 show the level selection screens. Every world has multiple activities in it, and earlier activities must be completed in order to unlock the following ones. Completed activities have a blue background icon and have a tick mark next to it. Available activities have a blue background, and unavailable activities are grayed out. This is a common feature in games, to stimulate the user to complete activities if he wants to move on to more advanced levels.



Figure 3.4: Worlds in the application

Figure 3.6 shows the interface of a programming activity. The title and subtitle display the current activity name, number and the task to be completed. In the lower left corner there is a back button to go to the previous screen. In the lower right corner there are two buttons: a check mark you must tap when you completed the activity (you will be prompted to confirm this action) and a robot icon labeled *send*, to send the current blocks in the workspace for the robot to execute. To the right there are two buttons: a hint button (represented by a lamp) that will give the user some help in completing the current activity and a reset button that will reset the workspace to the starting one. The canvas provided by Blockly is divided in two sections: to the left are the available blocks and to the right the workspace.

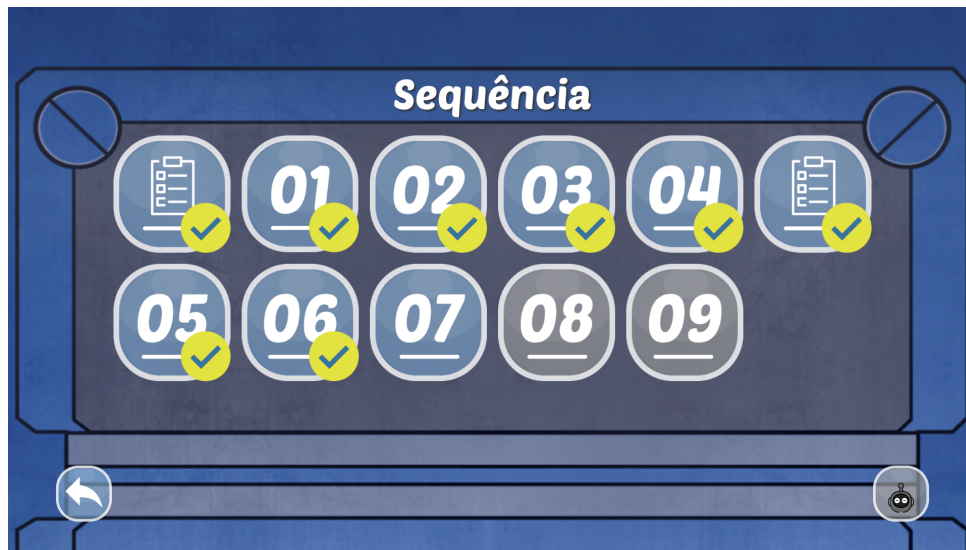


Figure 3.5: Levels in the application

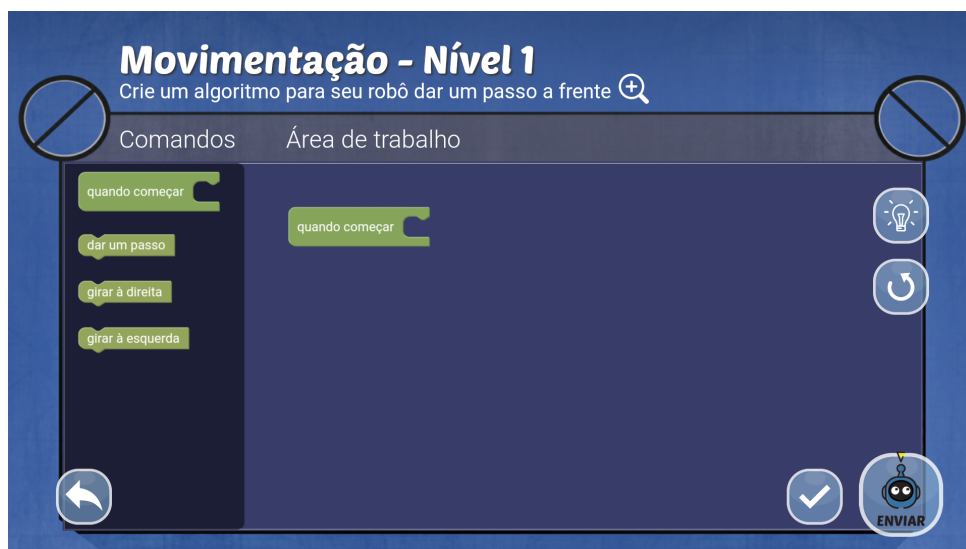


Figure 3.6: One of the challenges in the application

3.2 Bluetooth interface

The application communicates with the robot (see Figure 3.7) via Bluetooth. When someone taps play or free mode in the main screen, the application prompts them to connect to a robot. After the pairing is done, the application regularly pings the robot to make sure the connection was not lost.

When the user hits the send button, Blockly translates the blocks to a binary code. The code is then divided in 20 byte blocks and finally these blocks are sent to the robot in sequence.



Figure 3.7: Robot used in our experiments. The robot has two motor encoders, a Bluetooth adapter, an ultrasonic sensor, two LEDs and a button. It also has a hole children can put a pen in to use the robot for drawing.

3.3 Activities

There were three types of activities available: **tutorials** (Figure 3.8) that explained what concepts the following activities would involve, and hints of how the user could solve the problems that would be presented next; **quizzes** (Figure 3.9) to test the user knowledge, each one comprised by a question and three possible answers - of which only one was correct; **programming challenges** (Figure 3.6) that involved either creating a new algorithm to solve a problem or to fix an existing algorithm.

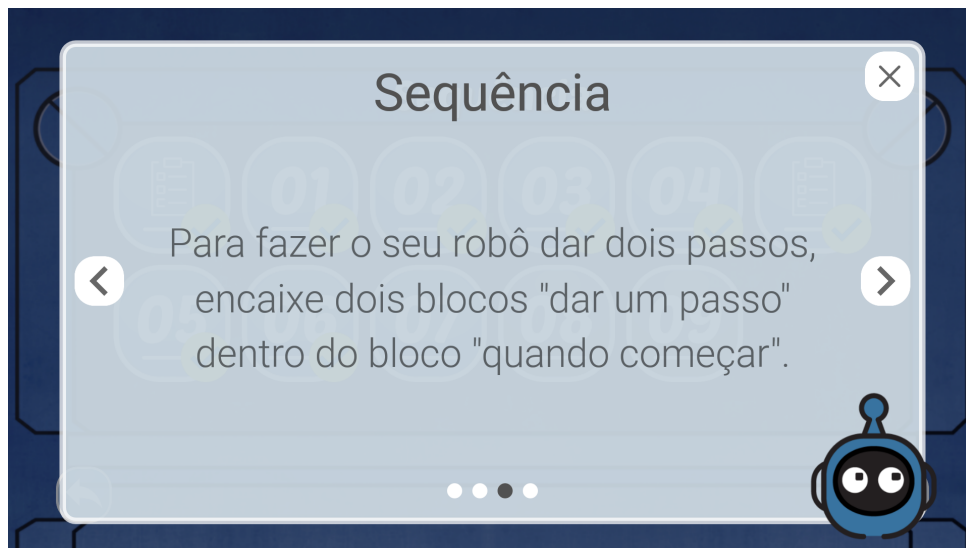


Figure 3.8: A tutorial in the application

In the later activity type, the user is required to solve a problem by programming the robot using a block-based visual programming language. Blockly provides a canvas in which the users can drag programming blocks and connect them to compose their program (see Figure 3.6). Blocks are shaped so that only connections that make sense are allowed. For example, the user may add a "Step forward" and a "Turn right" block and

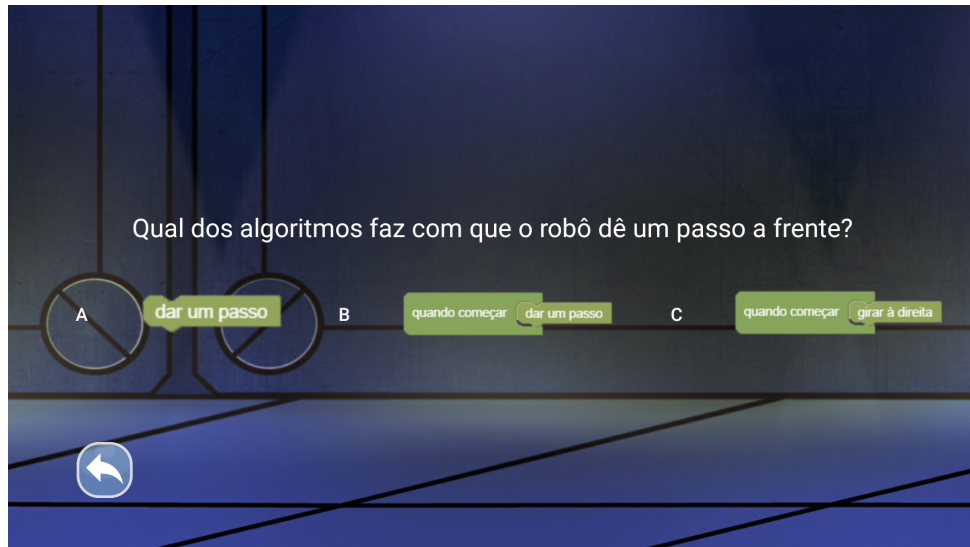


Figure 3.9: A quiz in the application

connect them to express a sequence of commands, however, a "Number" block may not be connected to a "Step forward" block. This feature minimizes issues associated with programming syntax, which are common in text based programming languages. Table 3.2 lists the programming blocks available in the application. See Appendix A for a list of all the activities.

Table 3.2: Programming blocks

Step forward		Turn left	
Turn right		Number	
Repeat		Repeat a number of times	
Number comparison		Distance in front of the robot	
If		If else	

3.4 Logs

In order to help our evaluation we had the application log some of the actions, as listed in Table 3.3, as well as the timestamps and the current activity identification number. From these logs we were able to extract information such as how many attempts each pair had in each quiz activity or the algorithms that were marked as correct for a specific coding activity. This data, along with our notes and observations, allowed us to reach the conclusions presented in Chapter 4.3.

Table 3.3: Application logs

TUTORIAL-OPENED	User opened the tutorial
TUTORIAL-NEXT	Next tutorial page
TUTORIAL-PREV	Previous tutorial page
TUTORIAL-CLOSE	User closed the tutorial
QUIZ-OPEN	User opened the quiz
QUIZ-CORRECT	Correct quiz answer
QUIZ-INCORRECT	Incorrect quiz answer
QUIZ-CLOSE	User closed the quiz
ACTIVITY-OPEN	User opened the (programming) activity
ACTIVITY-HINT	User clicked hint
ACTIVITY-SEND	User sent the algorithm to the robot
ACTIVITY-CORRECT	User marked the current solution as correct
ACTIVITY-CLOSE	User closed the activity

3.5 Configuration file

All of the levels in the application are generated automatically from a JSON file. The reason behind this is to have the content of the tasks be completely separate from the application itself. The structure of the JSON is as follows (see Figure 3.10): the two main objects are **version**, an integer showed in Line 2, and **levels**, an array of arrays showed in Line 3. The objects in these arrays are the descriptions of the activities. Every activity must have an **id** number, showed in Lines 6, 15 and 30. Every activity must have a **type** that must be one of **tutorial**, **coding** or **quiz**. Every activity must also have a **prerequisites** array, an array of integers representing the activities that must have been completed for it to be available.

```

1  {
2      "version": 1,
3      "levels": [
4          [
5              {
6                  "id": 0,
7                  "type": "tutorial",
8                  "title": "Some Lesson",
9                  "text": [
10                     "Text1", "Text2"
11                 ],
12                 "prerequisites": []
13             },
14             {
15                 "id": 1,
16                 "type": "quiz",
17                 "title": "Quiz name",
18                 "text": "Question",
19                 "image": "image_0",
20                 "answers": [
21                     "image_1", "image_2", "image_3"
22                 ],
23                 "correct": "image_2",
24                 "prerequisites": [0]
25             }
26         ],
27         [
28             {
29                 "id": 2,
30                 "type": "coding",
31                 "title": "Title",
32                 "text": "Do something",
33                 "hint": "Hint",
34                 "blocks": "'event_onstart','step_forward','turn_right','turn_left'",
35                 "workspace": "",
36                 "prerequisites": [1]
37             }
38         ]
39     ]
40 }

```

Figure 3.10: Structure of the JSON configuration file

Tutorial activities must also have these attributes: a string **title**, with its title and an array **text**, with strings representing the content of the tutorial.

Quiz activities must also have the following attributes: a string **title**, with its title; a string **text**, with the question being asked in this quiz; a string **image**, with the image name or empty if this question has no associated image; a string array **answers**, with the names of the images to be each of the answers; a string **correct**, with the correct image name.

Coding activities must also have these attributes: a string **title**, with its title; a string **text**, with the task that should be completed; a string **blocks**, containing the names of the blocks available for this activity or empty for all blocks, a string **workspace**,

containing a string representation of the blocks that must be in the canvas when the activity starts (or empty for no blocks).

Although for our experiments we used a single JSON file with a single set of activities, different files can be used in order to present different tutorials, quizzes or challenges. This can be better explored by modifying the application to let the user choose which one he wants to open. In schools, the application could be programmed to open a specific set of activities depending on the schedule of classes, so different teachers could have activities more suited to the current content. A web application or similar can be developed to help create these JSON files - what is now a very long and dull task.

Chapter 4

Experimental setup, methodology and results

4.1 A change of paradigm

The usual teaching paradigm follows a very strict scheme: the teacher will first introduce a new subject, explaining its importance, its content and so on. After that, some sort of assessment will take place to make sure the students learned what they were supposed to. This can be in form of a test, an essay, a group project, among others. And last but certainly not least, the students then receive some kind of feedback that will tell them if they are on the right track. This is the classic teaching method, created hundreds of years ago and still the most used today.

In this work we tried to teach CT thinking to the participants through a different paradigm: the experiment was not to be seen by the children as a regular class environment, but rather as a time they would be playing instead of studying. No teacher would interfere and the children themselves would be able to set their own pace as they passed through the application's activities. This by itself is not a big change, but the lack of feedback is: not having a teacher determine the rhythm also meant they would have to decide whether or not the algorithms they developed in the application were correct. The main reason for using this teaching method was that the experiments took place in the student's free period - and not part of the regular classes at school.

4.2 Experimental setup and methodology

We designed experiments with students from a local public school. In total there were twenty five children aged 9 to 11. The children were asked to solve problems by controlling a physical robot using a block-based visual programming language. The problems and the tools were designed to motivate the children to learn computational thinking skills.

Figure 4.1 shows a picture of the experiment's environment, including the robot and a tablet running the application.

In total we had three sets of experiments - the first one with fourteen students, the second one with six students, and the final, and third one, with five students. In total

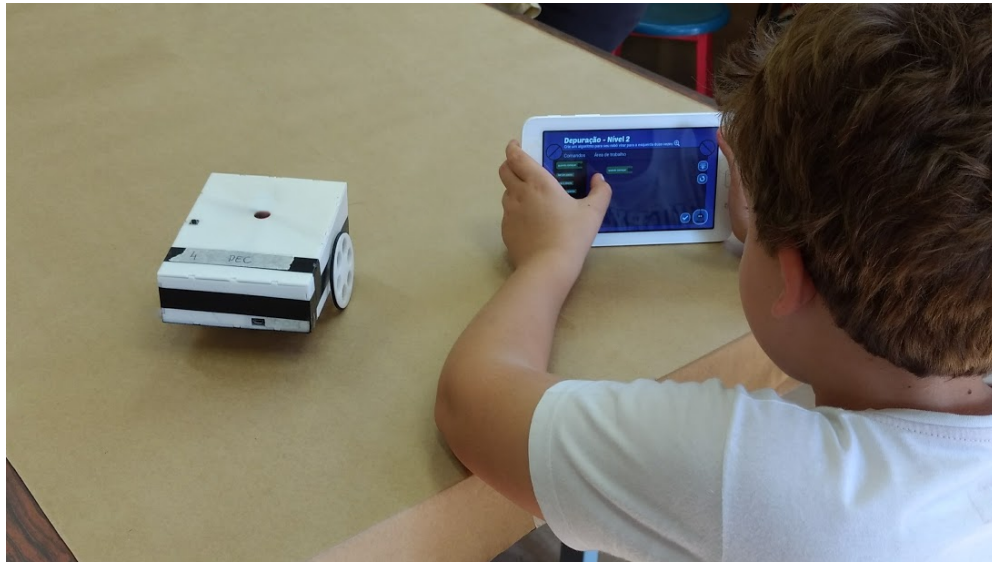


Figure 4.1: The experiment’s environment, including both the robot and the application

there were ten girls and fifteen boys. One of our ideas was that children should work in pairs so that one could help the other. In the first experiment the children were divided in pairs by their teachers. This proved to be a poor strategy since some of the children were very uncomfortable with the person they were paired to. For this reason, in the next two experiments children were allowed to choose their pair and generally this proved to be a better approach.

For each experiment we had around seven encounters, one for introduction and the others for the actual activities. In the last session of each experiment we also asked the participants to answer a short feedback form (see Appendix B) - twenty two students filled this form out.

In the introduction session we provided a brief explanation of the research and interested students received a consent form (see Appendix C) to be signed by their parents or legal tutors as well as an assent form (see Appendix D) to be signed by them. Also, as part of the introduction, each group was asked to name its robot.

After the children split in pairs and we handed over the robots and tablets, they were free to explore the application. The application itself is not capable of evaluating the user’s solutions to programming challenges - there are several solutions to each of the proposed challenges, so simply having one correct algorithm and testing that it matches user’s input would not suffice. To evaluate user’s algorithms we would have needed a much more sophisticated setup so the application could have the robot’s step-by-step information to only then determine if the desired solution was reached. We decided to let the students evaluate their own solutions by watching the robot to see if it behaved the way it was expected to. By doing so we could also observe whether the students had the ability to decide if their solution was correct.

We aimed at creating a playful and spontaneous environment, having as few evaluations and interventions as possible throughout the experiment, hoping it would encourage the children to *want to play*, rather than make them feel they *had to*. We could observe from the very beginning that competition was a much stronger motivating factor than

collaboration. For that reason, in some occasions we proposed having the robots compete in a race with obstacles or an arena where the last robot standing would be the winner. The rules of the arena were simple: the robots had to keep moving and if the robot hit a wall or another robot, it was out for the round. Our goal was to combine the concepts of conditional (only move forward if there is no obstacle detected by the ultrasonic distance sensor, otherwise turn left or right) and repetition (never stop moving). In these competition environments it was clear that the students tried several solutions, reaching for the best one possible.

4.3 Results

4.3.1 Observations

During the introduction session we learned that most children were familiar with the use of smartphones, tablets and computers. They seemed very excited to work with the robots - particularly the boys. Asking each group to name its robot worked even better than anticipated because they developed a personal connection with it throughout the experiment.

We also aimed to intervene as little as possible while also being available to clear any doubts or to help if the children got stuck in any task. We observed that a very loose environment compromises their ability to focus on the proposed activities; however, it had a very positive effect on their will. This was confirmed by the teachers, who stated that the children were very excited to be participating in the experiments, especially considering it took place in their free period, when they could choose the activity they liked best.

One thing that stood out in their behavior was that they clearly preferred competition to collaboration. The pairs that were supposed to be working together divided the activities in a round robin fashion, and instead of helping each other they rushed their colleague so they could get to play with the robot faster. However, when an environment of team competition occurred - like the robot races - they would collaborate with their teammate to reach a better solution and try to win. In the regular activities, the pairs were not so motivated to keep trying different solutions when the first one didn't work, and would often lose attention in the current task and go see what other children were doing. Because of this behavior we expect that the children would reach their best when working individually or in a competition setting.

Another thing that stood out was that boys and girls showed a very distinct behavior: boys wanted to grab both the robot and the tablet straight away and go play with it - although not necessarily play within the scope of the proposed activities. Rather than that, most boys wanted to complete the tasks as fast as they could to either reach other groups that were in more advanced stages or to be the first to get to those stages. Girls on the other hand showed a much higher interest in reading the tutorials and completing the activities successfully. The children, in general, were very interested in discovering what the parts of the robot could do - like the "eyes" (an ultrasonic sensor) - and how to make it move or turn on the LEDs.

4.3.2 Log analysis

As stated previously, the application was unable to evaluate the user's solution, and therefore it was expected that the children themselves would figure out whether they had successfully completed the given task or not. In many cases, they marked an activity as done even when they didn't program the robot as expected. There are some factors to consider:

- First, they could have thought that their solution was correct even if it wasn't, so it was an honest mistake.
- Being very familiar with other games and applications that can evaluate the solutions, they thought the application would only let them mark an activity as done if their solution was correct, so that's also an honest mistake.
- The third (and possibly worse) case is when the children mark the activity as done, when they knew it wasn't, so they could pass on to the next levels - to either reach their colleagues or to be the first to get to the next levels.
- There is also another extreme case: when the solution is correct but they aren't sure it is, so they keep trying to alter the code in order to see a better result in the robot.

Perhaps all of these issues can be solved by having a teacher or tutor check the child's solution before they can move on, and this certainly looks like a good solution to handle the aforementioned problems. However, this could potentially make the application lose the fun appeal and discourage children to play with it.

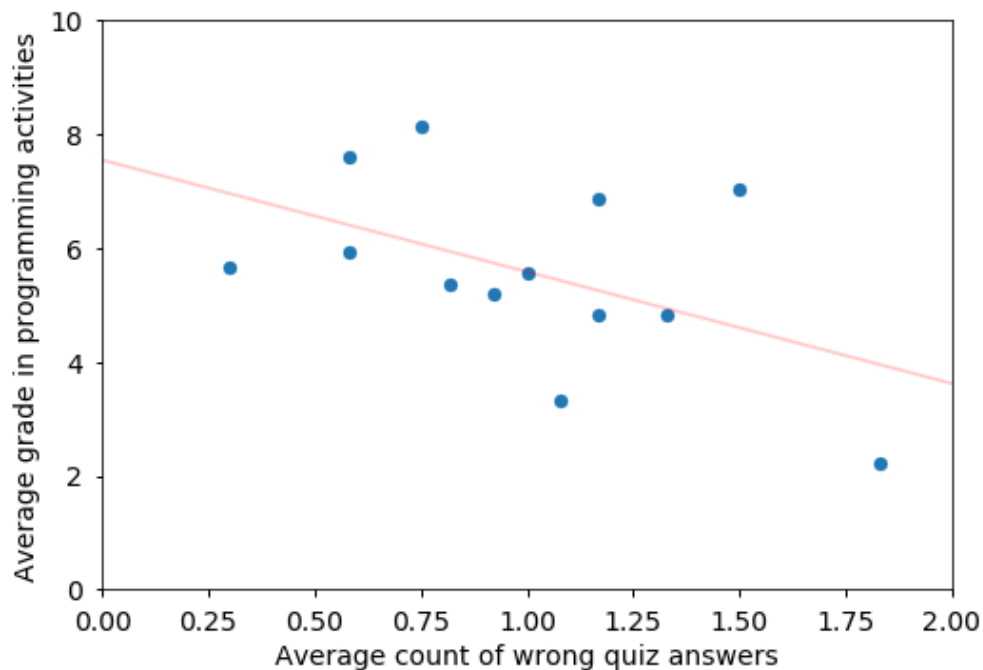


Figure 4.2: Grade in programming tasks versus count of wrong answers in quizzes

We analyzed the application's logs in order to look for some correlations in the statistics. All graphs presented in this section have thirteen points of data - since we had twenty five students, there were a total of twelve pairs and one child working individually. In total there were eleven tutorial activities, twelve quiz activities and twenty six programming activities.

- For the tutorial activities we calculated the average percentage of tutorial read per pair: each tutorial had a certain number of pages, and we checked the number of open pages and time spent in each page. Some children only opened the first page and already quit - meaning they never read the following pages.
- For the quiz activities we calculated the average number of incorrect answers, also per pair. We counted the wrong answers because the children had to complete the quiz in order to move forward, therefore a statistic to know how well they performed on these activities was to see how many mistakes they made.
- Finally, we graded the programming tasks using the following scale: 10 - completed the activity, 5 - partially completed the activity, and 0 - didn't complete the activity. For every pair we calculated the average grade across all programming activities.

We then compared these statistic two by two. The first result is presented in Figure 4.2, comparing average grade in programming activities versus average count of wrong quiz answers. The graph contains a red line showing a linear fit of the data. Notice that this line suggests that there is a negative correlation between these two metrics, i.e., the better the students scored in programming activities the less they select wrong quiz answers.

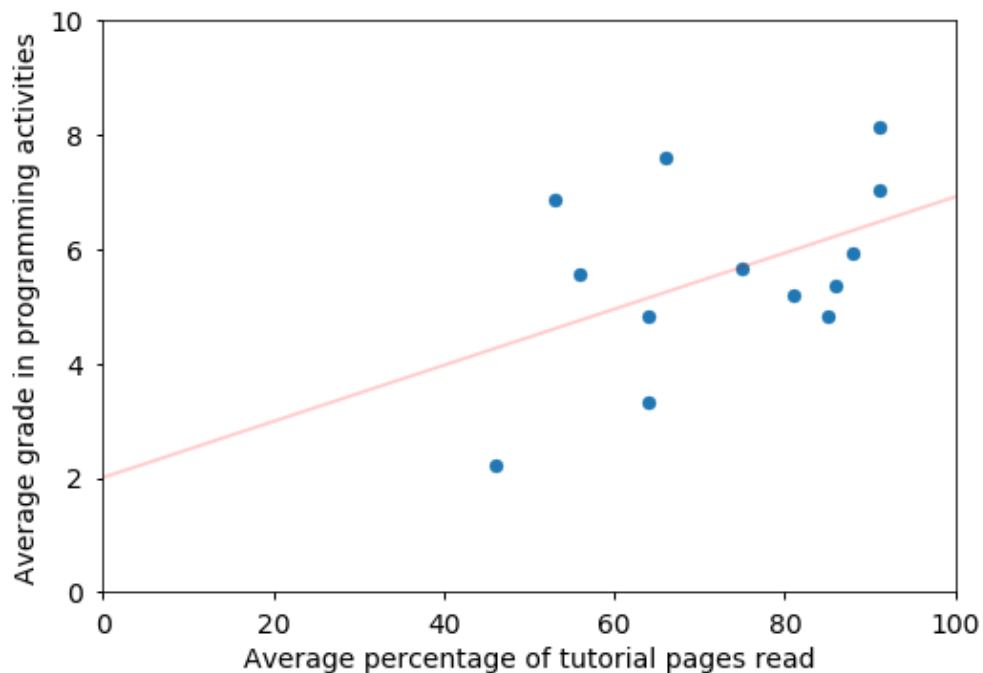


Figure 4.3: Grade in programming tasks versus percentage of tutorial pages read

Prior to the experiments, we anticipated that the children who read the tutorials would have a better performance in the quizzes and programming activities. As Figure

4.3 indicates, there seems to be a positive correlation between reading the tutorial pages and scoring higher on programming activities.

Even though the linear regression indicates that there is a positive correlation between grades in programming activities and number of tutorial pages read, we noticed that different groups with very similar grades had read different amounts of tutorials pages. One of the possible reasons this happened is due to the different children's backgrounds. Some of them could already have more knowledge of similar games and activities, and therefore even not reading the tutorials - or reading less of them - had a better comprehension of the activities and how to solve them.

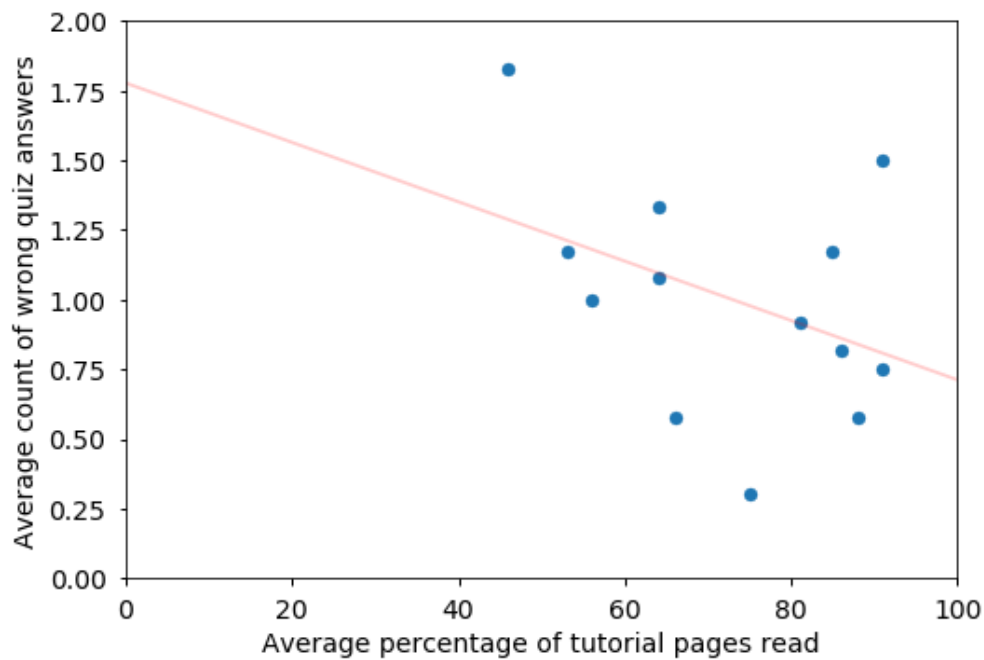


Figure 4.4: Count of wrong answers in quizzes versus percentage of tutorial pages read

Figure 4.4 shows the average count of wrong quiz answers against the percentage of tutorial pages read. Again, as indicated by the linear regression, there seems to be a negative correlation between the percentage of tutorial pages read by the students and the amount of incorrect quiz answers. In general, these results indicate that children who read more tutorials achieve higher grades in programming tasks and select fewer incorrect answers in quiz activities.

We also evaluated how well the children understood each concept by looking at the average grade that was obtained for the programming tasks involving that concept. Figure 4.5 shows a graph of average grades per concept. Activities involving the first concepts - movement, sequence and debugging - were completed with success by most groups, indicating that the students either learned the concepts or had previous knowledge of them. The activities that involved the loop concept were only completed partially, with a much lower grade than the previous ones. Finally, no child was able to achieve the expected results for the conditional concept. It is still unclear whether children at that age can't understand this concept well or if the way it was presented in our experiment was too complicated.

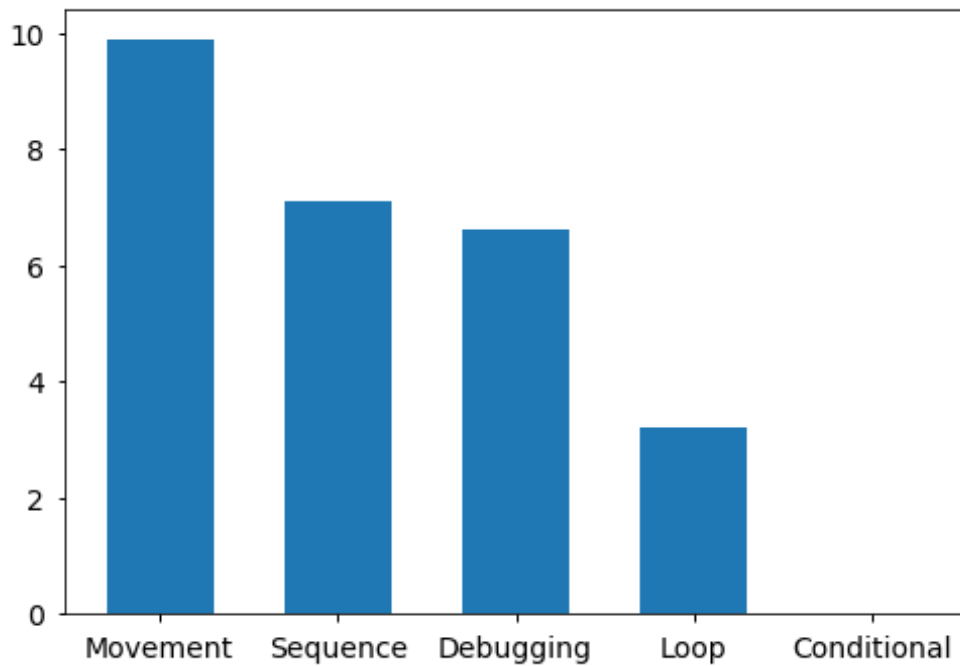


Figure 4.5: Average grade for all groups by concept

4.3.3 Children's feedback

Table 4.1 shows the answers to the feedback form - twenty two of the twenty five participants filled this form. Each question had three possible answers: yes, partially, and no. Overall, the feedback provided by the children was very positive - we only had 5.5% of negative answers across all questions, against 25.5% neutral and 69% positive. The only question that did not have a significant positive outcome was "Is the application easy to use?", indicating that we need to evaluate what's the best way to present the programming interface. The teachers also gave us very positive feedback, stating that the students were very excited to be participating and that they felt this was an excellent way to keep their attention.

The form also included space for comments and suggestions. Many of the children wrote that they would like to see other types of robots and other features: "*My suggestion is to create other types of robot and present it to other schools.*", "*I wish it [the robot] could talk and that it had a laser.*", "*My suggestions are that the robot should have arms and legs like other toys and be able to speak.*", "*A suggestion is to make the robot faster.*".

About working in pairs, the children were conflicted: some liked it ("*I liked working in pairs. It's fun that every level is different and has a tutorial.*") and other did not ("*I liked the idea of working in pairs but I didn't like my pair.*").

Finally, many of the feedback was related to having more activities and levels: "*I think it would be cool if you made more activities and different things to do.*", "*I wish there were more worlds.*", "*More levels. It would be fun and come back next year so more people can enjoy this project.*".

Table 4.1: Questions and answers in the feedback form

Question	Positive answers	Neutral answers	Negative answers
Would you use the application and robot again?	16 (72.7%)	5 (22.7%)	1 (4.5%)
Would you recommend the application and robot to a friend?	17 (77.3%)	4 (18.2%)	1 (4.5%)
Is the application easy to use?	7 (31.8%)	14 (63.6%)	1 (4.5%)
Is the application content fun and interesting?	19 (86.4%)	2 (9.1%)	1 (4.5%)
Did you like working in pairs?	17 (77.3%)	3 (13.6%)	2 (9.1%)
Average	15.2 (69%)	5.6 (25.5%)	1.2 (5.5%)

4.4 Discussion

These experiments were made in order to support the development of a methodology to teach CT to children through programming. The interaction with the children helped us design the following hypothesis:

- having a physical instrument to interact with is a motivating factor, whether it is a robot, a board, a set of command pieces, etc;
- evaluation in the form of tests or exams can have a negative impact on the way children see the CT education project;
- leaving the children free to use the application and play with the robot make them more comfortable, but it does not necessarily mean a positive impact on their learning;
- having one robot per child or projects that require collaboration between the children to be completed improves learning experience and children's engagement in the activities;
- competition improves children's engagement in the activities;
- have teachers suggest activities that complement the regular disciplines may improve learning and engagement in the classroom.

Future experiments with a higher number of children and for a longer period are needed in order to test these hypothesis. We believe that such a study would greatly contribute to the development of methodologies to teach CT and CS to children.

Chapter 5

Conclusions and future work

Computing is still a relatively young science, especially when compared to Mathematics and Physics, both hundreds (if not thousands) of years old. However, it has developed so rapidly that education was not able to keep up. Teachers are unable to prepare children to see computers for their full potential. It is imperative today that everyone is able to think logically and algorithmically, and so computational thinking is more necessary than ever. Over the past decade several authors discussed the importance of and methods to teach CT and CS concepts to children, nonetheless, there is still no consensus on the best teaching methodology.

We expected the children to be engaged and to use their critical thinking in order to judge their own work. Our expectations were only half met: the children were indeed excited, but failed in deciding if their solutions were good enough to move on. Without this feedback going on to more advanced lessons means nothing: it's like trying to learn how to multiply before you know how to add.

In the previous Chapters we described our experiments teaching computational thinking concepts to children between 9 and 11 years old using a robot and application we developed. We now share our insights on how to support the development of effective tools and methodologies to teach CT to children. Our results indicate that:

- children were very excited to interact with the robot;
- by giving a name to the robot, children established a personal connection with it improving their engagement in the experiment;
- competition is a motivating factor and encourages teamwork;
- the children did not enjoy working in pairs when a task was too trivial, therefore we think they should either have their own robot or more complex tasks should be given;
- most of the children did not have any difficulty with building sequences or debugging, however, they had a hard time applying loop and conditional concepts;
- the children were unable to assess their solutions by themselves.

There are many possible extensions to this dissertation. Some of them are:

- the most obvious suggestion is to design a new set of experiments with a larger number of participants and for a longer period to test the hypothesis stated in Section 4.4 as well as observe long term effects of learning computational thinking;
- the application was built to generate the levels automatically from a JSON file, as discussed in Chapter 3. This could be better explored in the future, for example in building a web application where teachers can create or modify tasks and deploy them to the whole classroom;
- the application could be modified to generate reports of the usage and deliver this data to the teachers or parents;
- new challenges can be developed to take advantage of all the available hardware in the robot, like the button and the LEDs;
- the robot itself can be improved for better speed, better accuracy in its movement, capability to turn in other angles than 90 degrees. Other entirely new sensors could be installed, like motion sensors, microphones, etc;
- both the robot and application could be improved to allow an automatic evaluation of the proposed tasks;
- teachers could be invited to participate in a session of the experiments so they could provide valuable feedback on how to improve and what new features to develop;
- teachers could be invited to collaborate in creating new tasks to complement the current content being taught in the classroom as defined by the national curriculum;
- work is also needed in training today's teachers to be able to teach these concepts they are not familiar with.

Bibliography

- [1] Wing, J. M. (2006). Computational thinking. *Communications of the ACM*, 49(3), 33-35.
- [2] Wing, J. M. (2008). Computational thinking and thinking about computing. *Philosophical transactions of the royal society of London A: mathematical, physical and engineering sciences*, 366(1881), 3717-3725.
- [3] Barcelos, T. S. & Silveira, I. F. (2012). Pensamento computacional e educação matemática: Relações para o ensino de computação na educação básica. In *XX Workshop sobre Educação em Computação, Curitiba. Anais do XXXII CSBC* (Vol. 2, p. 23).
- [4] Barr, V. & Stephenson, C. (2011) Bringing computational thinking to K-12: what is Involved and what is the role of the computer science education community?. *Acm Inroads*, 2(1), 48-54.
- [5] França, R. S. de, & Amaral, H. J. C. do. (2013) Proposta Metodológica de Ensino e Avaliação para o Desenvolvimento do Pensamento Computacional com o Uso do Scratch. In *Anais do Workshop de Informática na Escola* (Vol. 1, No. 1, p. 179).
- [6] Lye, S. Y., & Koh, J. H. L. (2014). Review on teaching and learning of computational thinking through programming: What is next for K-12?. *Computers in Human Behavior*, 41, 51-61.
- [7] CSTA. (2017) CSTA K-12 Computer Science Standards, Revised 2017, <https://sites.google.com/site/cstastandards/standards>.
- [8] Buitrago Flórez, F., Casallas, R., Hernández, M., Reyes, A., Restrepo, S., & Danies, G. (2017). Changing a Generation's Way of Thinking: Teaching Computational Thinking Through Programming. *Review of Educational Research*, 87(4), 834-860.
- [9] Eloy, A. A. D. S., Martins, A. R. Q., Pazinato, A. M., Lukjanenko, M. D. F. S. P., & Lopes, R. D. D. (2017, June). Programming Literacy: Computational Thinking in Brazilian Public Schools. In *Proceedings of the 2017 Conference on Interaction Design and Children* (pp. 439-444). ACM.
- [10] UK Department for Education. (2013) National curriculum in England: computing programmes of study, <https://www.gov.uk/government/publications/national-curriculum-in-england-computing-programmes-of-study>.

- [11] Smith, Megan. (2016) Computer Science For All, The White House, <https://www.whitehouse.gov/blog/2016/01/30/computer-science-all>.
- [12] Papert, Seymour. (1972) Teaching Children Thinking, *Programmed Learning and Educational Technology*, 9(5), 245-255.
- [13] Logo Foundation. (1991) Logo, <http://el.media.mit.edu/logo-foundation/>.
- [14] Papert, S. (1980) *Mindstorms: Children, computers, and powerful ideas*. Basic Books, Inc.
- [15] Sociedade Brasileira de Computação. (2017) Referenciais de Formação em Computação: Educação Básica, <http://www.sbc.org.br/noticias/10-slideshow-noticias/1996-referenciais-de-formacao-em-computacao-educacao-basica>.
- [16] Grover, S. and Pea, R. (2013). Computational thinking in K–12: A review of the state of the field. *Educational Researcher*, 42(1), 38-43.
- [17] Brennan, K. & Resnick, M. (2012) New frameworks for studying and assessing the development of computational thinking. In *Proceedings of the 2012 annual meeting of the American Educational Research Association*, Vancouver, Canada (pp. 1-25).
- [18] Fields, D. A., Lui, D., & Kafai, Y. B. (2017). Teaching computational thinking with electronic textiles: High school teachers' contextualizing strategies in Exploring Computer Science. In *Conference Proceedings of International Conference on Computational Thinking Education* (pp. 67-72).
- [19] Weintrop, D., Beheshti, E., Horn, M., Orton, K., Jona, K., Trouille, L., & Wilensky, U. (2016). Defining computational thinking for mathematics and science classrooms. *Journal of Science Education and Technology*, 25(1), 127-147.
- [20] Rees, A., García-Peñalvo, F. J., Jormanainen, I., Tuul, M., & Reimann, D. (2016). An overview of the most relevant literature on coding and computational thinking with emphasis on the relevant issues for teachers.
- [21] Godinho, J., Torres, K., Batista, G., Andrade, E., & Gomide, J. (2017) Projeto Aprenda a Programar Jogando: Divulgando a Programação de Computadores para Crianças e Jovens. XXV Workshop sobre Educação em Computação, Anais do XXXVII CSBC (p. 2140).
- [22] Aono, A. H., Rody, H. V. S., Musa, D. L., Pereira, V. A., & Almeida, J. (2017) A Utilização do Scratch como Ferramenta no Ensino de Pensamento Computacional para Crianças. XXV Workshop sobre Educação em Computação, Anais do XXXVII CSBC (p. 2169).
- [23] Silva Junior, S. M. da, & França, S. V. A. (2017) Programação para todos: Análise Comparativa de Ferramentas Utilizadas no Ensino de Programação. XXV Workshop sobre Educação em Computação, Anais do XXXVII CSBC (p. 2199).

Appendix A

List of application activities

A.1 Introduction to algorithms

A.1.1 Tutorial

An algorithm is a sequence of commands. An example of a common algorithm is a cake recipe: it has a series of steps to be followed in order to make a cake. These are some of the commands in a recipe:

- Preheat the oven
- Mix the ingredients until smooth

Our robot can also follow commands to do a task. The available commands are:

- Step forward
- Turn left
- Turn right

A.1.2 Quiz

1. Preheat the oven to 180C.
2. Beat together the eggs, flour, caster sugar, butter, baking powder and cocoa until smooth in a large mixing bowl.
3. Turn into the prepared tins and bake in the preheated oven for 25 mins.
4. Leave to cool in the tin, then turn on to a serving plate.

What's the first step to make a chocolate cake according to this recipe?

Preheat the oven to 180C.

(a) Correct answer

Beat together the eggs, flour, caster sugar, butter, baking powder and cocoa until smooth in a large mixing bowl.

(b) Incorrect answer

Turn into the prepared tins and bake in the preheated oven for 25 mins.

(c) Incorrect answer

A.1.3 Quiz

1. Preheat the oven to 180C.
2. Beat together the eggs, flour, caster sugar, butter, baking powder and cocoa until smooth in a large mixing bowl.
3. Turn into the prepared tins and bake in the preheated oven for 25 mins.
4. Leave to cool in the tin, then turn on to a serving plate.

What's the first step to make a chocolate cake according to this recipe?

Turn into the prepared tins and bake in the preheated oven for 25 mins.

(a) Correct answer

Preheat the oven to 180C.

(b) Incorrect answer

Leave to cool in the tin, then turn on to a serving plate.

(c) Incorrect answer

A.2 Introduction to movement

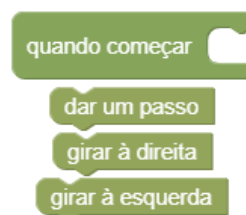
A.2.1 Tutorial

A command can be sent to your robot so that it can perform a task. We saw that the available commands are:

- Step forward
- Turn left
- Turn right

To send a command, we need the block on start (*quando começar*). The robot will execute the command that is inside this block. To make your robot step forward, simply drag the block step forward (*dar um passo*) to fit inside the on start block, then hit send.

Blocks available after this activity:



A.2.2 Programming activity

Objective: Design an algorithm to make your robot step forward.

Hint: Fit the movement block inside the on start block and hit send.

A.2.3 Programming activity

Objective: Design an algorithm to make your robot turn left.

Hint: Fit the movement block inside the on start block and hit send.

A.2.4 Programming activity

Objective: Design an algorithm to make your robot turn right.

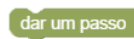
Hint: Fit the movement block inside the on start block and hit send.

A.2.5 Quiz

Which algorithm makes the robot step forward?



(a) Correct answer



(b) Incorrect answer



(c) Incorrect answer

A.2.6 Quiz

Which algorithm makes the robot turn left?



(a) Correct answer



(b) Incorrect answer



(c) Incorrect answer

A.2.7 Quiz

Which algorithm makes the robot turn right?



(a) Correct answer



(b) Incorrect answer



(c) Incorrect answer

A.3 Sequence

A.3.1 Tutorial

In the previous activities, we learned how to send the robot a command. But is the robot capable of following more than one command? The answer is yes. To create a sequence of commands, drag more blocks inside the on start block. To make your robot take two steps forward, use two step forward blocks. Let's do it!

A.3.2 Programming activity

Objective: Design an algorithm to make your robot take two steps forward.

Hint: Put more than one movement block in the on start block and hit send.

A.3.3 Programming activity

Objective: Design an algorithm to make your robot turn left twice.

Hint: Put more than one movement block in the on start block and hit send.

A.3.4 Programming activity

Objective: Design an algorithm to make your robot turn right twice.

Hint: Put more than one movement block in the on start block and hit send.

A.3.5 Programming activity

Objective: Design an algorithm to make your robot take five steps forward.

Hint: Put more than one movement block in the on start block and hit send.

A.3.6 Tutorial

We already know how to create a sequence so the robot follows a command more than once. We can also make the robot follow different types of commands in the same sequence. You just have to put different commands inside the on start block. You can put as many commands as you want, and the robot will execute them one at a time in the given order.

A.3.7 Programming activity

Objective: Design an algorithm for your robot to make an L shaped path.

Hint: Put different movement blocks in the on start block in the order you want them to be executed.

A.3.8 Programming activity

Objective: Design an algorithm for your robot to make a U shaped path.

Hint: Put different movement blocks in the on start block in the order you want them to be executed.

A.3.9 Programming activity

Objective: Design an algorithm for your robot to make a square shaped path.

Hint: Put different movement blocks in the on start block in the order you want them to be executed.

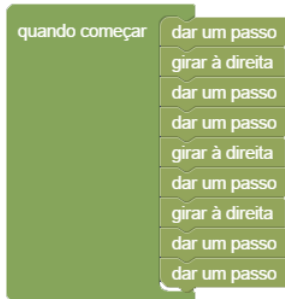
A.3.10 Programming activity

Objective: Design an algorithm for your robot to dodge an obstacle.

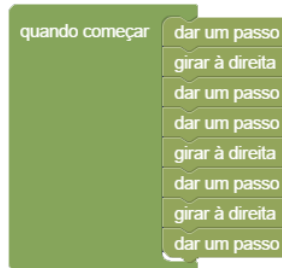
Hint: Put different movement blocks in the on start block in the order you want them to be executed.

A.3.11 Quiz

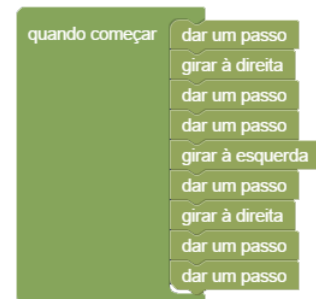
Which of the algorithms makes the robot walk in a rectangle shaped path?



(a) Correct answer



(b) Incorrect answer



(c) Incorrect answer

A.4 Debugging

A.4.1 Tutorial

It's very common for an algorithm to have an error. How can we find the error and fix it? This is what debugging is about - trying to find which step is incorrect. A lot of the times the algorithm is on the right track, but incomplete. This means we have to add more commands in order to have it do what we expect. Let's practice this now.

A.4.2 Programming activity

Objective: Complete the algorithm for the robot to take 4 steps forward.

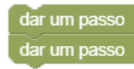


Hint: Add one or more blocks to the current algorithm.

A.4.3 Quiz



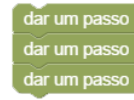
What blocks should we add to this algorithm to make the robot take 5 steps forward?



(a) Correct answer



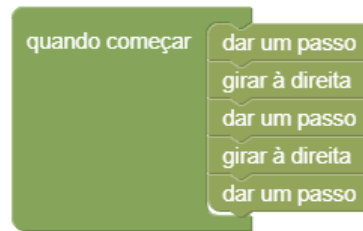
(b) Incorrect answer



(c) Incorrect answer

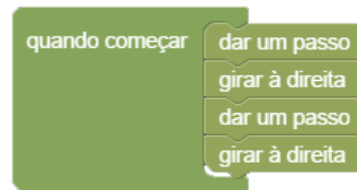
A.4.4 Programming activity

Objective: Complete the algorithm for the robot to make a squared shaped path.

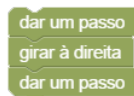


Hint: Add one or more blocks to the current algorithm.

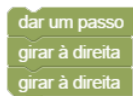
A.4.5 Quiz



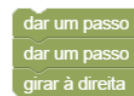
What blocks should we add to this algorithm for the robot to make a squared shaped path?



(a) Correct answer



(b) Incorrect answer



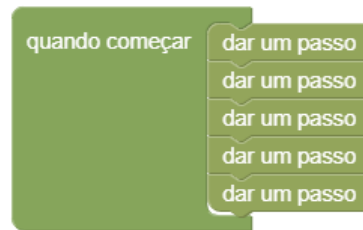
(c) Incorrect answer

A.4.6 Tutorial

We already learned how to fix an incomplete algorithm: add commands to make it behave as expected. What if our algorithm has too many commands? In that case we need to remove one or more blocks. We are practicing this in the next activities.

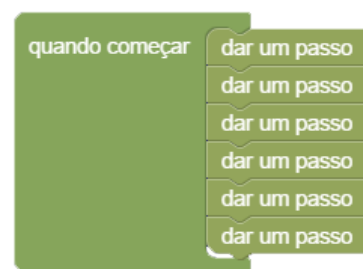
A.4.7 Programming activity

Objective: Remove one or more blocks from the current algorithm to make the robot take 4 steps forward.

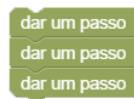


Hint: Remove one or more blocks from the current algorithm.

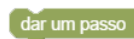
A.4.8 Quiz



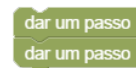
Which blocks should we remove to make the robot take 3 steps forward?



(a) Correct answer



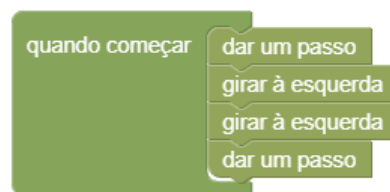
(b) Incorrect answer



(c) Incorrect answer

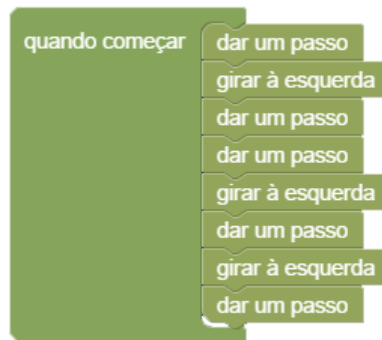
A.4.9 Programming activity

Objective: Remove one or more blocks from the current algorithm for the robot to make an L shaped path.



Hint: Remove one or more blocks from the current algorithm.

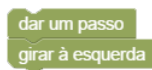
A.4.10 Quiz



Which blocks should we remove for the robot to make a squared shaped path?



(a) Correct answer



(b) Incorrect answer



(c) Incorrect answer

A.4.11 Tutorial

We have already learned to add missing blocks or to remove extra blocks to fix an algorithm. Knowing this we can identify errors on many algorithms and fix them. But what if we have a command that was supposed to be another one? In that case we need two steps to fix the error: removing the incorrect block and replacing it with the correct one. Let's try it.

A.4.12 Programming activity

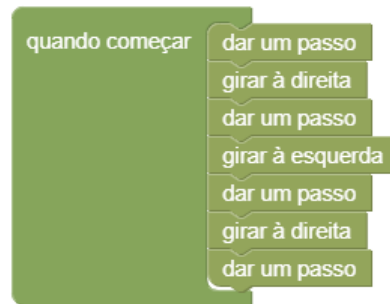
Objective: Replace one of the blocks for the robot to make an L shaped path.



Hint: Replace a block with a different command.

A.4.13 Programming activity

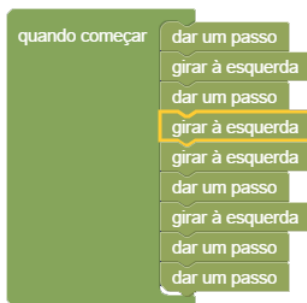
Objective: Replace one or more of the blocks for the robot to make a square shaped path.



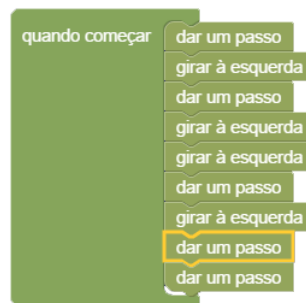
Hint: Replace one or more blocks with different commands.

A.4.14 Quiz

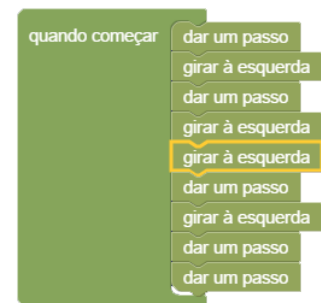
Which block should we replace for the robot to make a rectangle shaped path?



(a) Correct answer



(b) Incorrect answer



(c) Incorrect answer

A.5 Loop

A.5.1 Tutorial

We already know how to create a sequence of commands. What if we need the robot to execute the same sequence multiple times? Is the best way to do that just copy the blocks again? Not really: we can use the repeat block to avoid having to duplicate all the comands. To repeat a sequence ten times, use the repeat N times block (*repita N vezes*), write 10 as your number and put your sequence inside it.

New block available after this activity:



A.5.2 Programming activity

Objective: Design an algorithm for your robot to walk around an obstacle.

Hint: Use the movement blocks in a sequence.

A.5.3 Programming activity

Objective: Design an algorithm for your robot to walk around an obstacle twice.

Hint: Put the algorithm for the robot to walk around an obstacle inside the repeat 2 times block.

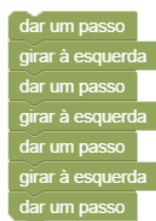
A.5.4 Programming activity

Objective: Design an algorithm for your robot to walk around an obstacle five times.

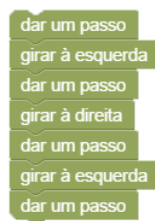
Hint: Put the algorithm for the robot to walk around an obstacle inside the repeat 5 times block.

A.5.5 Quiz

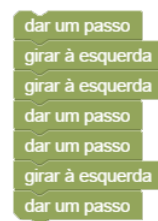
Which algorithm should be repeated for the robot to make a square shaped path three times?



(a) Correct answer



(b) Incorrect answer



(c) Incorrect answer

A.5.6 Tutorial

We already know how to make the robot repeat a task for a number of times. What if we need the robot to execute an algorithm forever? You should use the repeat block (*repita*) for that. The robot will continue executing the algorithm inside this block until it is turned off.

New block available after this activity:



A.5.7 Programming activity

Objective: Design an algorithm for your robot to step forward (nonstop).

Hint: Put movement blocks inside the repeat block.

A.5.8 Programming activity

Objective: Design an algorithm for your robot to zig zag (nonstop).

Hint: Put movement blocks inside the repeat block.

A.5.9 Programming activity

Objective: Design an algorithm for your robot to walk around an obstacle (nonstop).

Hint: Put movement blocks inside the repeat block.

A.6 If

A.6.1 Tutorial

We already know how to make your robot execute a sequence once or many times. Can we ask the robot to make a decision? We can use the if block (*se*) to do that. If the condition following the block is true, the robot will execute the commands inside it.

New blocks available after this activity:



A.6.2 Programming activity

Objective: Design an algorithm for your robot to step forward if the distance ahead of it is greater than 20.

Hint: Use the if block and put the step forward block inside it. As a condition, use $\text{distance} > 20$.

A.6.3 Tutorial

We know how to make the robot execute a command if the condition is true. What if it's false? In that case, use the else block (*senão*). If the condition is false, the robot will execute these commands instead.

A.6.4 Programming activity

Objective: Design an algorithm for your robot to step forward if the distance ahead of it is greater than 20, if not then turn right.

Hint: Use the if block and put the step forward block inside it. As a condition, use $\text{distance} > 20$. Put a turn right block in the else block.

A.6.5 Programming activity

Objective: Design an algorithm for your robot to walk around without hitting any obstacles.

Hint: Use the repeat block with a if else block inside it.

Appendix B

Feedback form

This was the final feedback form given to the children that participated in the experiments.

Qual a sua opinião sobre o aplicativo e robô?

Para cada uma das perguntas, selecione uma resposta.

Você usaria o aplicativo e robô novamente?		
 Sim	 Talvez	 Não

Você recomendaria o aplicativo e robô para um amigo?		
 Sim	 Talvez	 Não

O aplicativo é fácil de usar?		
 Sim	 Um pouco	 Não

O aplicativo traz desafios divertidos e interessantes?		
 Sim	 Um pouco	 Não

Você gostou de ter trabalhado em dupla?		
 Sim	 Um pouco	 Não

Você tem algum outro comentário ou sugestão?

Appendix C

Consent form

Termo de Consentimento Livre e Esclarecido

Oficina de aprendizagem de conceitos de computação
com um robô controlado por um aplicativo

Pesquisadora: Laís Vasconcellos Minchillo

Orientadora: Dra. Juliana Freitag Borin

Você ou o menor de idade sob sua responsabilidade legal está sendo convidado a participar como voluntário de um estudo. Este documento, chamado Termo de Consentimento Livre e Esclarecido (TCLE), visa assegurar os direitos e deveres de cada participante e é elaborado em duas vias, uma que deverá ficar com você e outra com o pesquisador.

Por favor, leia com atenção e calma, aproveitando para esclarecer suas dúvidas. Se houver perguntas antes ou mesmo depois de assiná-lo, você poderá esclarecê-las com o pesquisador. Se preferir, pode levar para casa e consultar seus familiares ou outras pessoas antes de decidir participar. Se você ou o menor de idade sob sua responsabilidade legal não quiser participar ou deseja retirar sua autorização a qualquer momento, não haverá nenhum tipo de penalização ou prejuízo.

Justificativa e objetivos:

Pensamento computacional é uma habilidade útil para resolução de problemas em todas as áreas de conhecimento. Iniciativas no mundo todo têm como objetivo o ensino desta habilidade para crianças, e em alguns países isto já é parte do currículo escolar básico. Nos Estados Unidos, o presidente Obama lançou no início do ano uma iniciativa para o ensino de Ciência da Computação no currículo escolar. No Brasil, a SBC (Sociedade Brasileira de Computação) também está trabalhando para incluir o ensino de Computação no currículo escolar brasileiro.

Este trabalho propõe o uso da programação como ferramenta de ensino do pensamento computacional, contando com um robô físico de baixo custo, controlado por um aplicativo para tablets ou smartphones, que traz um ambiente lúdico e motivador para as crianças. Nosso objetivo é que os alunos exercitem conceitos de programação enquanto brincam com o robô e também avaliar os conhecimentos adquiridos por eles.

Procedimentos:

Você está sendo convidado para participar de uma série de oficinas (com duração aproximada de duas horas cada). Nessas oficinas serão oferecidas atividades para a programação de um robô, que é controlado por um aplicativo em um tablet. Os alunos serão divididos em grupos de dois ou três membros, e cada grupo vai explorar as lições disponíveis de forma independente. Nosso objetivo é observar o tempo gasto em cada atividade, o número de tentativas e se ela foi concluída com sucesso.

Desconfortos e riscos:

Não há riscos previsíveis para os participantes do estudo, uma vez que serão utilizados dispositivos não invasivos com os quais os participantes já estão familiarizados, tais como tablets e

smartphones. Os robôs também foram construídos de forma a não oferecer riscos como cortes ou choques.

Benefícios:

O possível benefício direto para as crianças é adquirir conhecimentos em lógica, matemática, resolução de problemas e conceitos de programação.

Acompanhamento e assistência:

Durante as oficinas, os pesquisadores estarão disponíveis para ajudar a responder perguntas na utilização ou desenvolvimento de qualquer atividade. Não há necessidade de assistência fora das oficinas.

Sigilo e privacidade:

Você tem a garantia de que sua identidade, ou a do menor de idade sob sua responsabilidade legal, será mantida em sigilo e nenhuma informação será dada a outras pessoas que não façam parte da equipe de pesquisadores. Na divulgação dos resultados desse estudo, seu nome não será citado. A gravação das oficinas é apenas para garantir que nenhum detalhe, importante ou não, seja omitido.

Ressarcimento:

Os pesquisadores se deslocarão até a escola para a realização das oficinas, durante a rotina dos alunos. Por este motivo, não haverá valor de ressarcimento.

Indenização:

Os participantes da pesquisa que vierem a sofrer qualquer tipo de dano resultante de sua participação na pesquisa, previsto ou não neste termo, têm direito à indenização, por parte do pesquisador, patrocinador e das instituições envolvidas.

Contato:

Em caso de dúvidas sobre o estudo, você poderá entrar em contato com:

Laís Vasconcellos Minchillo
Instituto de Computação, UNICAMP
Av. Albert Einstein, 1251 - Cidade
Universitária, CEP 13083-852,
Campinas/SP Brasil

Juliana Freitag Borin
Instituto de Computação, UNICAMP
Av. Albert Einstein, 1251 - Cidade
Universitária, CEP 13083-852,
Campinas/SP Brasil

Em caso de denúncias ou reclamações sobre sua participação no estudo, você pode entrar em contato com a secretaria do Comitê de Ética em Pesquisa (CEP): Rua: Tessália Vieira de Camargo, 126; CEP 13083-887 Campinas/SP; telefone (19) 3521-8936; fax (19) 3521-7187; e-mail: cep@fcm.unicamp.br

Consentimento livre e esclarecido:

Após ter sido esclarecido sobre a natureza da pesquisa, seus objetivos, métodos, benefícios previstos, potenciais riscos e o incômodo que esta possa acarretar, aceito participar, ou concordo com a participação do menor de idade sob minha responsabilidade legal.

Nome do(a) participante: _____

_____ Data: ____/____/____

(Assinatura do participante ou nome e assinatura do responsável)

Responsabilidade do Pesquisador:

Asseguro ter cumprido as exigências da resolução 466/2012 CNS/MS e complementares na elaboração do protocolo e na obtenção deste Termo de Consentimento Livre e Esclarecido. Asseguro, também, ter explicado e fornecido uma cópia deste documento ao participante. Informo que o estudo foi aprovado pelo CEP perante o qual o projeto foi apresentado. Comprometo-me a utilizar o material e os dados obtidos nesta pesquisa exclusivamente para as finalidades previstas neste documento ou conforme o consentimento dado pelo participante.

_____ Data: ____/____/____

(Assinatura do pesquisador)

Appendix D

Assent form

Termo de Assentimento Livre e Esclarecido

Oficina de aprendizagem de conceitos de computação
com um robô controlado por um aplicativo

Pesquisadora: Laís Vasconcellos Minchillo

Orientadora: Dra. Juliana Freitag Borin

Você está sendo convidado a participar como voluntário da pesquisa "Oficina de aprendizagem de conceitos de computação com um robô controlado por um aplicativo".

O motivo que nos leva a estudar esse assunto é: pensamento computacional é uma habilidade útil para resolução de problemas, e seu ensino já é parte do currículo escolar básico em muitos países. Neste trabalho propomos o ensino de pensamento computacional através da programação de um robô físico.

Adotaremos os seguintes procedimentos: durante cada uma das sessões (com duração aproximada de duas horas), os voluntários serão divididos em grupos de dois ou três integrantes, cada grupo receberá um robô e um tablet que é responsável por sua programação. Cada grupo vai explorar os desafios propostos de forma independente. Durante as oficinas, os pesquisadores estarão disponíveis para ajudar a responder perguntas na utilização ou desenvolvimento de qualquer atividade.

A sua participação é voluntária e a recusa em participar não acarretará qualquer penalidade ou modificação na forma em que é atendido. Para você participar desta pesquisa, seu responsável legal deverá autorizar sua participação e assinar um Termo de Consentimento. Você será esclarecido em qualquer aspecto que desejar e estará livre para participar ou recusar-se. O seu responsável legal poderá retirar o consentimento ou interromper a sua participação a qualquer momento.

Você não terá nenhum custo, nem receberá qualquer vantagem financeira. Caso sejam identificados e comprovados danos provenientes dessa pesquisa, você tem direito à indenização.

Não há riscos previsíveis na participação da pesquisa, uma vez que serão utilizados dispositivos não invasivos com os quais os participantes já estão familiarizados, tais como tablets e smartphones. Os robôs também foram construídos de forma a não oferecer riscos como cortes ou choques.

A pesquisa poderá contribuir para você adquirir conhecimentos em lógica, matemática, resolução de problemas e conceitos de programação.

Você tem a garantia de que sua identidade será mantida em sigilo e nenhuma informação será dada a outras pessoas que não façam parte da equipe de pesquisadores. Na divulgação dos resultados desse estudo, seu nome não será citado. Os resultados estarão à sua disposição quando finalizada.

Este termo de consentimento encontra-se impresso em duas vias originais: sendo que uma será arquivada pelo pesquisador responsável, e a outra será fornecida a você.

Em caso de dúvidas sobre o estudo, você poderá entrar em contato com:

Laís Vasconcellos Minchillo
Instituto de Computação, UNICAMP
Av. Albert Einstein, 1251 - Cidade
Universitária, CEP 13083-852,
Campinas/SP Brasil

Juliana Freitag Borin
Instituto de Computação, UNICAMP
Av. Albert Einstein, 1251 - Cidade
Universitária, CEP 13083-852,
Campinas/SP Brasil

Em caso de denúncias ou reclamações sobre sua participação no estudo, você pode entrar em contato com a secretaria do Comitê de Ética em Pesquisa (CEP): Rua: Tessália Vieira de Camargo, 126; CEP 13083-887 Campinas/SP; telefone (19) 3521-8936; fax (19) 3521-7187; e-mail: cep@fcm.unicamp.br

Assentimento livre e esclarecido:

Aceito participar desta pesquisa após ter sido esclarecido sobre sua natureza, seus objetivos, métodos, benefícios previstos, potenciais riscos e o incômodo que esta possa acarretar.

Nome do(a) participante: _____

_____ Data: ____/____/____

(Assinatura do participante)

Responsabilidade do Pesquisador:

Asseguro ter cumprido as exigências da resolução 466/2012 CNS/MS e complementares na elaboração do protocolo e na obtenção deste Termo de Consentimento Livre e Esclarecido. Asseguro, também, ter explicado e fornecido uma cópia deste documento ao participante. Informo que o estudo foi aprovado pelo CEP perante o qual o projeto foi apresentado. Comprometo-me a utilizar o material e os dados obtidos nesta pesquisa exclusivamente para as finalidades previstas neste documento ou conforme o consentimento dado pelo participante.

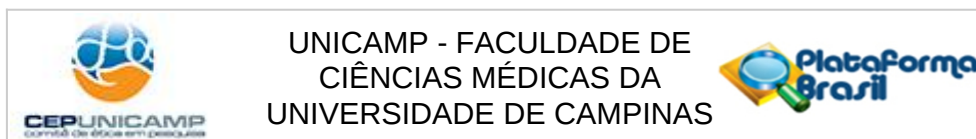
_____ Data: ____/____/____

(Assinatura do pesquisador)

Appendix E

Ethics committee

This project has been approved by the Universidade Estadual de Campinas (UNICAMP) Ethics Committee (*Comitê de Ética em Pesquisa - CEP*).



PARECER CONSUBSTANCIADO DO CEP

DADOS DO PROJETO DE PESQUISA

Título da Pesquisa: Programação como ferramenta de ensino de pensamento computacional

Pesquisador: LAIS VASCONCELLOS MINCHILLO

Área Temática:

Versão: 3

CAAE: 61934616.8.0000.5404

Instituição Proponente: Instituto de Computação

Patrocinador Principal: TECSINAPSE TECNOLOGIA DA INFORMACAO LTDA.

DADOS DO PARECER

Número do Parecer: 1.890.508

Apresentação do Projeto:

Pensamento computacional é uma habilidade útil para resolução de problemas em todas as áreas de conhecimento. Iniciativas no mundo todo têm como objetivo o ensino desta habilidade para crianças, e em alguns países isto já é parte do currículo escolar básico. Propomos o uso da programação como ferramenta de ensino de pensamento computacional, com o desenvolvimento de um aplicativo para plataformas móveis. Esta tarefa traz diversos desafios: definição de conceitos de programação a ensinar, métodos de ensino, escolha da faixa etária alvo, criação de uma metodologia de testes e avaliação final. Esperamos constatar ao final deste trabalho que o ensino de programação de fato traz o aprendizado de pensamento computacional.

Objetivo da Pesquisa:

Objetivo Primário:

Testar o uso de um aplicativo que exercita conceitos de programação. Avaliar se os usuários adquiriram conhecimento após o uso do aplicativo.

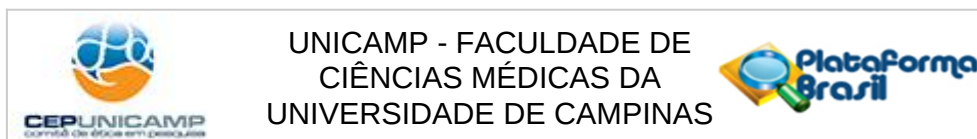
Objetivo Secundário:

Verificar se o robô é um agente motivador para o uso do aplicativo.

Avaliação dos Riscos e Benefícios:

Riscos:

Endereço: Rua Tessália Vieira de Camargo, 126
Bairro: Barão Geraldo **CEP:** 13.083-887
UF: SP **Município:** CAMPINAS
Telefone: (19)3521-8936 **Fax:** (19)3521-7187 **E-mail:** cep@fcm.unicamp.br



Continuação do Parecer: 1.890.508

Os robôs que serão utilizados nas oficinas foram construídos de forma a evitar qualquer perigo durante seu uso, como cortes ou choques. Os tablets também oferecem pouco risco.

Benefícios:

Os usuários possivelmente vão adquirir conhecimentos em lógica, matemática, resolução de problemas e conceitos de programação.

Comentários e Considerações sobre a Pesquisa:

Trata-se de um projeto de mestrado do IC-UNICAMP que propõe a utilização de um robô programado por um tablete no ensino de programação em uma escola (DEDIC). O projeto está dividido em 5 fases e na sessão 5 será aplicado um questionário (n=30).

A pesquisa é pertinente e embasada na literatura. Não há riscos previsíveis e o possível benefício direto aos participantes da pesquisa será adquirir conhecimentos em lógica, matemática, resolução de problemas e conceitos de programação.

Considerações sobre os Termos de apresentação obrigatória:

Todos os documentos foram apresentados:

- 1) Folha de rosto, devidamente assinada pelo diretor associado da FT-UNICAMP.
- 2) Projeto de Pesquisa gerado pela Plataforma Brasil, com o cronograma e orçamento adequados.
- 3) Projeto de pesquisa detalhado e questionário, devidamente escritos e referenciados.
- 4) TCLE e termo de assentimento devidamente redigidos.

Recomendações:

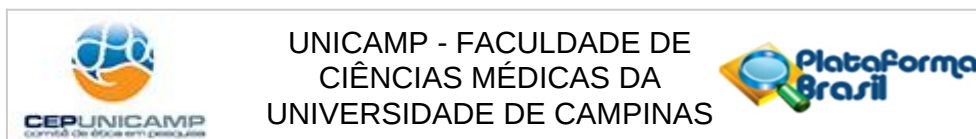
Conclusões ou Pendências e Lista de Inadequações:

Uma vez que todas as pendências foram atendidas e o projeto é pertinente e embasado na literatura, recomento a aprovação.

Pendências e respostas da segunda análise:

- 1) Em pesquisas cujos convidados sejam adolescentes, é necessário à anuência do participante da pesquisa através do termo de assentimento livre esclarecido, sem prejuízo do consentimento de seus responsáveis legais. Tais participantes devem ser esclarecidos sobre a natureza da pesquisa,

Endereço: Rua Tessália Vieira de Camargo, 126
Bairro: Barão Geraldo **CEP:** 13.083-887
UF: SP **Município:** CAMPINAS
Telefone: (19)3521-8936 **Fax:** (19)3521-7187 **E-mail:** cep@fcm.unicamp.br



Continuação do Parecer: 1.890.508

seus objetivos, métodos, benefícios previstos, potenciais riscos e o incômodo que esta possa lhes acarretar, na medida de sua compreensão e respeitados em suas singularidades.

Resposta: O público alvo da pesquisa são crianças entre 10 e 12 anos. Explicaremos a natureza da pesquisa, assim como seus objetivos, métodos, benefícios previstos e riscos potenciais. Os alunos que desejarem participar deverão assinar um Termo de Assentimento, e será solicitada a assinatura do Termo de Consentimento por um responsável legal.

Situação: O termo de assentimento foi incluso.

Conclusão: Pendência atendida.

Pendências e respostas da primeira análise:

Uma vez que a carta resposta está no formato de imagem não foi possível transcrevê-la.

1) TCLE:

1.1) Numerar as páginas de forma a saber a quantidade total de páginas em cada página, por exemplo, 1/2 e 2/2. Readequar.

Situação: O TCLE foi modificado.

Conclusão: Pendência atendida.

1.2) O texto como foi descrito no TCLE não garante indenização por danos decorrentes da pesquisa. A Resolução 466/12 (item IV.3) define que "os participantes da pesquisa que vierem a sofrer qualquer tipo de dano resultante de sua participação na pesquisa, previsto ou não no TCLE, têm direito à indenização, por parte do pesquisador, patrocinador e das instituições envolvidas". Cabe enfatizar que a questão da indenização não é prerrogativa da Resolução 466/12, estando prevista no código civil. Portanto, solicitamos que seja assegurado, de forma clara e afirmativa, que o participante de pesquisa tem direito à indenização em casos de danos decorrentes da pesquisa. Readequar.

Situação: O TCLE foi modificado.

Conclusão: Pendência atendida.

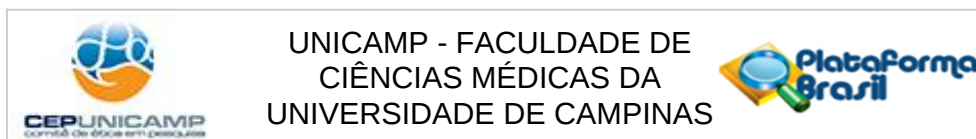
1.3) Colocar o benefício como um possível benefício. Readequar.

Situação: O TCLE foi modificado.

Conclusão: Pendência atendida.

2) Em pesquisas cujos convidados sejam adolescentes, é necessário à anuência do participante da pesquisa através do termo de assentimento livre esclarecido, sem prejuízo do consentimento de seus responsáveis legais. Tais participantes devem ser esclarecidos sobre a natureza da pesquisa, seus objetivos, métodos, benefícios previstos, potenciais riscos e o incômodo que esta possa lhes acarretar, na medida de sua compreensão e respeitados em suas singularidades.

Endereço: Rua Tessália Vieira de Camargo, 126
Bairro: Barão Geraldo **CEP:** 13.083-887
UF: SP **Município:** CAMPINAS
Telefone: (19)3521-8936 **Fax:** (19)3521-7187 **E-mail:** cep@fcm.unicamp.br



Continuação do Parecer: 1.890.508

Situação: O assentimento não foi incluído. Apesar dos pesquisadores dizerem, na carta resposta, que falarão sobre a natureza do projeto aos adolescentes.

Conclusão: Pendência não atendida.

Considerações Finais a critério do CEP:

- O sujeito de pesquisa deve receber uma via do Termo de Consentimento Livre e Esclarecido, na íntegra, por ele assinado (quando aplicável).

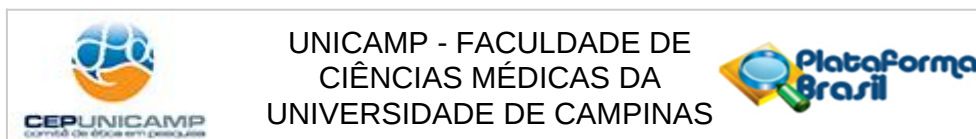
- O sujeito da pesquisa tem a liberdade de recusar-se a participar ou de retirar seu consentimento em qualquer fase da pesquisa, sem penalização alguma e sem prejuízo ao seu cuidado (quando aplicável).

- O pesquisador deve desenvolver a pesquisa conforme delineada no protocolo aprovado. Se o pesquisador considerar a descontinuação do estudo, esta deve ser justificada e somente ser realizada após análise das razões da descontinuidade pelo CEP que o aprovou. O pesquisador deve aguardar o parecer do CEP quanto à descontinuação, exceto quando perceber risco ou dano não previsto ao sujeito participante ou quando constatar a superioridade de uma estratégia diagnóstica ou terapêutica oferecida a um dos grupos da pesquisa, isto é, somente em caso de necessidade de ação imediata com intuito de proteger os participantes.

- O CEP deve ser informado de todos os efeitos adversos ou fatos relevantes que alterem o curso normal do estudo. É papel do pesquisador assegurar medidas imediatas adequadas frente a evento adverso grave ocorrido (mesmo que tenha sido em outro centro) e enviar notificação ao CEP e à Agência Nacional de Vigilância Sanitária – ANVISA – junto com seu posicionamento.

- Eventuais modificações ou emendas ao protocolo devem ser apresentadas ao CEP de forma clara e sucinta, identificando a parte do protocolo a ser modificada e suas justificativas e aguardando a aprovação do CEP para continuidade da pesquisa. Em caso de projetos do Grupo I ou II apresentados anteriormente à ANVISA, o pesquisador ou patrocinador deve enviá-las também à mesma, junto com o parecer aprovatório do CEP, para serem juntadas ao protocolo inicial.

Endereço: Rua Tessália Vieira de Camargo, 126
Bairro: Barão Geraldo **CEP:** 13.083-887
UF: SP **Município:** CAMPINAS
Telefone: (19)3521-8936 **Fax:** (19)3521-7187 **E-mail:** cep@fcm.unicamp.br



Continuação do Parecer: 1.890.508

- Relatórios parciais e final devem ser apresentados ao CEP, inicialmente seis meses após a data deste parecer de aprovação e ao término do estudo.

-Lembramos que segundo a Resolução 466/2012 , item XI.2 letra e, "cabe ao pesquisador apresentar dados solicitados pelo CEP ou pela CONEP a qualquer momento".

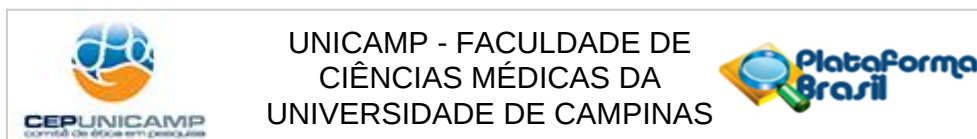
Este parecer foi elaborado baseado nos documentos abaixo relacionados:

Tipo Documento	Arquivo	Postagem	Autor	Situação
Informações Básicas do Projeto	PB_INFORMAÇÕES_BÁSICAS_DO_PROJETO_821298.pdf	19/12/2016 18:06:23		Aceito
Outros	CartaResposta2.pdf	19/12/2016 18:06:03	LAIS VASCONCELLOS MINCHILLO	Aceito
TCLE / Termos de Assentimento / Justificativa de Ausência	TermodeAssentimentoLivreeEsclarecido.pdf	19/12/2016 18:05:44	LAIS VASCONCELLOS MINCHILLO	Aceito
Outros	carta_resposta_pendencias.pdf	04/12/2016 20:07:47	LAIS VASCONCELLOS MINCHILLO	Aceito
TCLE / Termos de Assentimento / Justificativa de Ausência	TCLE_v2.pdf	04/12/2016 20:06:47	LAIS VASCONCELLOS MINCHILLO	Aceito
Projeto Detalhado / Brochura Investigador	projeto_completo.pdf	10/11/2016 17:17:43	LAIS VASCONCELLOS MINCHILLO	Aceito
Outros	questionario_avaliacao.pdf	10/11/2016 12:59:46	LAIS VASCONCELLOS MINCHILLO	Aceito
Declaração de Instituição e Infraestrutura	declaracao_instituicao_dedic_unicamp.pdf	09/11/2016 21:09:56	LAIS VASCONCELLOS MINCHILLO	Aceito
Folha de Rosto	folha_de_rostoassinada.pdf	09/11/2016 21:09:31	LAIS VASCONCELLOS MINCHILLO	Aceito

Situação do Parecer:

Aprovado

Endereço: Rua Tessália Vieira de Camargo, 126
Bairro: Barão Geraldo **CEP:** 13.083-887
UF: SP **Município:** CAMPINAS
Telefone: (19)3521-8936 **Fax:** (19)3521-7187 **E-mail:** cep@fcm.unicamp.br



Continuação do Parecer: 1.890.508

Necessita Apreciação da CONEP:

Não

CAMPINAS, 13 de Janeiro de 2017

Assinado por:
Renata Maria dos Santos Celeghini
(Coordenador)

Endereço: Rua Tessália Vieira de Camargo, 126
Bairro: Barão Geraldo **CEP:** 13.083-887
UF: SP **Município:** CAMPINAS
Telefone: (19)3521-8936 **Fax:** (19)3521-7187 **E-mail:** cep@fcm.unicamp.br