Activity for learning computational thinking in plugged and unplugged mode

Margarida Romero, Thierry Viéville, Marie Duflot-Kremer

Activity for learning computational thinking in plugged and unplugged mode.

Margarida Romero[1] , Thierry Viéville[2],
Marie Duflot-Kremer[3]

MSc SmartEdtech Program

Research Report N°006 — October 2022 —15 pages.

Abstract: The introduction of computing into schools can drive the development of computational thinking along with associated problem-solving skills. In this context we look at the different types of activities used to learn computing, with the aim of establishing a protocol to compare plugged and unplugged activities at school, and more specifically to see how effectively unplugged activities can be used for skills transfer in order to learn computing.

Key-words: Unplugged activity, Pedagogical robotics, Computational thinkink

1 LINE laboratory – Margarida.romero@univ-cotedazur.fr

2 Inria Mnemosyne team & LINE laboratory – thierry.vieville@inria.fr

3 LORIA – marie.duflot-kremer@loria.fr

1. Introduction

There has been a major upheaval in the way computing is learned in schools in the last few decades (Baron & Bruillard, 2013). In recent years, learning computing has gained in popularity in schools, partly because of the accessibility of visual programming tools like Scratch (Resnick et al., 2009) and partly because of increasing awareness of the need to understand and demystify digital technology so that citizens can develop a critical and creative approach to digital issues. Computing has already begun to be incorporated into official curricula in France, England, Canadian provinces such as British Colombia, and many other countries (Heintz, Mannila, & Färnqvist, 2016). We can assume that to understand digital technology, you need to be familiar with some of its principles, uses, and challenges. Rather than just seeing digital as a set of technical and procedural knowledge, Wing (2006) proposes the concept of computational thinking as the ability to use computing methods and concepts to solve problems. The concept of computational thinking draws on problem-solving strategies in a number of fields. Since the concept of computational thinking was first proposed, numerous studies have been conducted to conceptualize and evaluate it in the context of different activities for learning programming (Grover & Pea, 2013). The development of computational thinking can lead to the discovery of new concepts such as those related to algorithms or the coding of information. When they set up activities for learning to program, teachers' first thought is often about what they need in terms of computer equipment (Romero & Netto, 2018). But do we really need a computer to start to develop computational thinking? The answer to this question turns out not to be straightforward; unplugged activities that transpose computing concepts into the handling of everyday objects and the movement of the learners have proved very promising. In this study, we present a protocol for studying computational thinking in a context where programming is learned using unplugged activities and using the Scratch visual programming software. Scratch was chosen because of its international popularity in primary education (Lye & Koh, 2014).

2 Learning to program using unplugged activities

2.0 Learning with unplugged computing activities.

It is often said that we should limit screen time (Saunders & Vallance, 2017), avoid digital passivity (Karsenti, 2018), and develop critical thinking about the digital world. In the context of learning to program, the use of software such as Scratch or connected objects or robots is often discussed (Misirli & Komis, 2014); however, another paradigm exists in which neither hardware nor software are required: unplugged computing (Bell, Alexander, Freeman, & Grimley, 2009; Bell, Witten, & Fellows, 1998). During unplugged or computerless activities, children learn, through play during 'unplugged' activities, the concepts at the heart of computing in general or robotics in particular. With these

activities it is possible, for example, to learn what an algorithm is, or how to encode and transmit a piece of information. They approach computational thinking (Wing, 2011) as a way of solving problems, in which computers are not the end in themselves; the learners can take a step back, embodying (Tsarava et al., 2017) their learning through activities involving movement. This approach to learning computing is also consistent with a disregard for technophilia and technophobia as opposing binary positions. Instead it aims to develop a sociocritical approach through which the learner and citizen develops a critical and creative relationship with digital technology, appropriating the concepts and processes of computational thinking with their entire mind and body. Various studies of educational robotics (Highfield, Mulligan, & Hedberg, 2008; Misirli & Komis, 2016) thus show the value of the human/robot spatial relationship as a way of developing spatial awareness.

More than twenty years ago, researchers in New Zealand (Bell et al., 1998) set up a programme for teaching the basics of computing without a computer. Their 'CS Unplugged' document (*L'informatique sans ordinateur*[4] in French) clearly explains the philosophy behind this approach and suggests a whole range of activities for pupils from primary school age upwards. The positive impact of these activities was studied by Brackmann and his colleagues (2017) on two classes with a control group; they observed a statistically significant improvement in the children's performance in computational thinking, for example, breaking down problems into subproblems, and creating algorithms.

Unplugged computing activities have the advantage first and foremost of not requiring costly equipment or the skills to operate it (Curzon, Dorling, Ng, Selby, & Woollard, 2014). But this major advantage in terms of viability and accessibility is not enough unless the activities also, above all, have great educational potential as a different way of learning computing concepts and processes. In this article we analyze these activities from the point of view of embodied cognition (Wilson, 2002), a way of learning with the body in a particular context. We will finish by discussing the play experience given by unplugged learning activities.

4 https://interstices.info/upload/csunplugged/CSUnplugged_fr.pdf

2.1 Learning creative programming

The effects of programming activities on the development of computing concepts and processes need to be examined on various levels. Learning programming procedurally means learning a certain sequence of instructions, but does not guarantee the development of computational thinking (Romero, Noirpoudre, & Viéville, 2018). By aiming for more than simply teaching children to program, for example by using unplugged activities or by directing activity towards learning computational thinking, it is possible to achieve genuine positive effects at primary and early secondary level, even with newly qualified teachers (Moreno-León & Robles, 2015). What makes the difference is moving from learning programming procedurally to integrating creative programming in an interdisciplinary way (Resnick & Siegal, 2015; Romero, 2016). What counts is not simply learning programming, but instead finding ways of learning through programming that can develop computational learning. The procedural approach engages the pupil in a programming sequence where the parameters are set by the teacher, whereas the creative approach gives the pupil room for creativity in terms of both the procedure and the product created (Romero, 2016). The value of developing activities for learning programming that aim to develop computational thinking is well established in various studies (Grover & Pea, 2013), in which the authors report positive effects on, for example, problem-solving ability (as defined by Torp, 2002) and, to a lesser extent, reasoning and spatial awareness. These results were achieved with groups of university students (i.e. future teachers) and also secondary pupils. A bibliography review of various studies about learning computing is available online through the Class'Code project (Romero et al., 2018).

2.2 Unplugged computing: a range of different educational approaches

In view of the emergence and diversity of activities directed at learning computing in schools, a growing number of studies have looked at the value and effectiveness of these different approaches. Computing can be learned using computer equipment, but also through unplugged activities that use computing concepts and processes; from Bell, Witten and Fellows (1998) to Duflot (2016), there is a wide range of different unplugged activities. Some unplugged activities are very procedural: a set of instructions is provided that the pupils have to follow. Other activities take the form of a 'magic trick', typically a card trick where the explanation is based on an algorithm that is impossible to guess and can only be discovered when a 'hint' is given. Neither of these two extremes is ideal. Our experience is that it is better to offer a research activity with attainable milestones, as was done following a large-scale trial by Calmet, Hirtzig and Wilgenbus (2016). Starting with long explanations that need to be retained should also be avoided, and the participants' situation should be explained immediately: one person is the robot, stands up and gets into position, and the other is the programmer and gives the instructions; the activity is revealed as the action progresses. A low-level mode is often chosen for the

start of the activity, and then, as in video games with levels, the activity is enhanced with slightly more complex challenges. Obviously it is important not just to complete the activity, but also to take a step back, to explain the link with the concept being learned, and possibly to include a historical element as an illustration, as Viéville and Tort (2013) did. Getting the participants to talk about it, for example in a discussion or a question-and-answer session, enables further information to be gathered about the activity (Duflot et al., 2015).

Another aspect is the construction or the setting up of everyday objects for use during the activity (e.g. organizing chairs to make a maze for a robot, or building a graph on which to walk through the execution of an algorithm). Involving pupils in this (or suggesting that they then run the activity) is very worthwhile, so that they are playing an active role in their own learning, given that engagement is well known to drive learning. It is also very important for these activities to be 'infectious', in the sense that the learners of today may be the teachers of tomorrow. Making them want to be the teacher is, for some children, an important factor in engagement. We have observed this during field work, and this has been confirmed by approaches such as object-oriented learning (Hannan, Chatterjee, & Duhs, 2013).

2.2 The potential of unplugged computing activities in education

The educational benefits of unplugged activities are discussed, for example, in Wohl, Porter and Clinch (2015) and Brackmann et al. (2017). The study by Wolf and colleagues tested how well the skills of understanding the concept of an algorithm (measured by the ability to describe a procedure), of logical prediction and of debugging were learned by children aged 5 to 7 years, and showed that these concepts were learned, particularly with unplugged activities (though there was no explicit comparative study). The second study, concerning similar skills (problem decomposition, structure recognition, algorithm design, abstraction of a process from one context to another) with children aged 10 to 12 years, establishes that unplugged activities make a significant contribution to learning, by comparison to a control group. We take into consideration the didactic aspects and we use the reference system developed by Curzon, Dorling, Ng, Selby and Woollard (2014) and the earlier educational research of Calmet, Hirtzig and Wilgenbus (2016). The work done by Bell, Alexander,

Freeman and Grimley (2009) led to the development of an unplugged computing curriculum (https://csunplugged.org) and to a precise definition of unplugged computing, which we summarize here. Unplugged computing refers to the discovery or acquisition of computing concepts without the use of digital tools. Learning computing using unplugged methods is not restricted to a basic level of understanding of a simple algorithm. It includes, for example in the case of Duflot (2016), understanding a programmable machine and, in the case of Calmet, Hirtzig and Wilgenbus (2016), concepts associated with data and data representation, networks and robotics. A study should be conducted on each of these aspects. Unplugged computing activities thus rely on interactions between the pupil and their spatial environment during an activity that should be meaningful in terms of the activity's development (Shelton, 2016). There is currently little research into the effects of unplugged computing, but what there is provides valuable insights. The research by Faber, Wierdsma, Doornbos, van der Ven and de Vette (2017) looked at lesson design for unplugged programming activities. They make recommendations such as taking account of differences in pupils' skill levels when designing unplugged activities so that the activities on offer have varying degrees of complexity. They also recommend clearly explaining, after an unplugged activity, how the concept will be used when the pupils work with computers. From a teaching perspective, unplugged activities also breed confidence among teachers in their own computing abilities and an understanding of the concepts of computational thinking. Teachers also learn teaching techniques for the introduction of computational thinking, which they can include in their own practice (Curzon et al., 2014). Teachers' confidence about computing is particularly important at a time when computing is starting to appear on official curricula. The work done by Wohl, Porter and Clinch (2015) most closely resembles our research objectives. They compared the effectiveness of unplugged activities, tangible programming with Cubelets, and programming with a digital interface such as Scratch. They state that the pupils' engagement was greatest with tangible programming, but that it was through the unplugged activities that the pupils developed the best understanding of the concepts of algorithm, data and data representation, logical prediction and debugging, according to the reference system developed by (Curzon et al., 2014).

2.3 The advantages of unplugged computing

Here, we present two differentiating aspects of unplugged computing compared to computing with visual programming tools on a digital device.

1) *The cognitive load of using a machine*. During plugged activities, the machine itself requires not an insignificant amount of technical learning and involves a considerable cognitive load. An unplugged computing activity is less startling for pupils and teachers because it is a type of game played with other people. This makes it easier to work as a group or a whole class, and avoids minor technical issues unrelated to the concepts being studied. An unplugged computing activity is more

familiar for pupils and teachers, because this type of game is also used for other mathematics-related subjects or for cross-disciplinary purposes such as learning self-control. Using machines imposes a cognitive load (Sweller, 1994) that can limit reflection on the general principles. In practice, some pupils also have difficulty listening to instructions or interacting with one another when working on a computer because so much of their attention is focused on the screen. This was observed in experiments set up during the production of the '1,2,3 codez' manual (Calmet et al., 2016). Also, with unplugged computing it is easier to distinguish between understanding of concepts, and learning to use a technological tool. Unplugged computing can also make it easier to work as a group or a whole class, and avoids minor technical issues unrelated to the concepts being studied.

2) *Embodied cognition*. Playing using the body and learning through movement and action is a physically and cognitively engaging activity because it uses procedural and episodic memory (the activity often consists of a series of scenarios) and also semantic memory, with interaction between them. This is a common finding confirmed by studies such as Owen and colleagues (2016) in respect of procedural memory; this type of link between episodic and semantic memory is well established (Tulving, 1972).

3) *Tangible analogy*. The main purpose seems to us to be constructing a tangible analogy for the abstract concepts encountered in computing. This notion of 'metaphor' means creating a concrete situation that appropriates the mechanisms to be used as the basis for constructing a representation of the concept to be learned. Using something to represent something else with which it shares some essential quality means giving an opportunity to take a step back and look at the subject being learned from a different perspective. The moment when the metaphor reaches its limits is also important: as soon as the learner says "it's not the same", the objective is met because the learner has begun to think about the subject. For example, how the TCP/IP transfer protocol works can be explained using a game consisting of sending messages around the class using Post-it notes, giving a physical experience of the concepts of addressing and connectivity, and the need for acknowledgement of receipt and retransmission on timeout: eventually the learners realize that datagrams circulating on the internet have to have additional functionalities. This aspect needs to be examined in more detail, for example using research such as that of Sander (2000) as the starting point.

4) Unplugged activities can then be followed up by plugged activities, because enlightened use of computers is still one of the main reasons for learning computing. It could even be frustrating to engage in activities presented as computing activities without ever using a computer; it is therefore

necessary to ensure activities initially carried out unplugged can be transposed to plugged activities.

5) Another aspect is the construction or the setting up of everyday objects for use during the activity (e.g. organizing chairs to make a maze for a robot, or building a graph on which to walk through the execution of an algorithm). Involving pupils in this (or suggesting that they then run the activity) is very worthwhile, so that they are playing an active role in their own learning, given that engagement is well known to drive learning. 'Low-tech' working makes it easier to use these educational drivers. This links to the idea of questioning where and when the use of computers is valuable (or not) in our everyday lives.

3. Analysis of some practices.

Unplugged activities do not require any computer equipment, but our qualitative observations during dozens of activities trialed as part of Class'Code show that they have much greater value than simply making up for a lack of equipment. During this experiment, despite a major effort to give the programming activity more structure between the initial testing of the protocol and the second trial being run, the pupils still had certain difficulties understanding the activity. When they were doing the movement activity on Scratch, the pupils wondered whether they could add blocks of a similar type to those already selected by the research team. Problems understanding the activity also emerged regarding how to assemble the blocks to create the code sequences. After the second iteration, we added some notes explaining that the pupils can use the same blocks more than once. We also added a sample function to help them understand how to create functions. To keep the artifacts produced with Scratch, an Educator account was created following the second iteration.

Analysis of the 'robot game' activity

The purpose of the 'robot game' unplugged activity is to introduce learners to the concept of instructions, which form the basis of computer programs. This activity introduces them to algorithms and to programming instructions without using any computer technology. With this type of activity, the pupil plays the role of a robot, which must follow the movement instructions given by two other pupils verbally and with pictograms corresponding to movement programming instructions (go forwards, go backwards, turn right, turn left). With this type of activity, the educational concepts of programming and robotics can be introduced while using skills associated with the pupils' spatial awareness (Romero & Vallerand, 2016). In her research, Duflot[5] identifies the robot game as offering the potential to work with programming concepts and processes, algorithms, numbers, spatial awareness, movement (absolute and relative) on a grid, and differentiation of left and right.

5 https://members.loria.fr/MDuflot/

Figure 1. Secondary pupils playing the robot game

Playing the robot game or the place of play in unplugged computing

In the activity proposed by Duflot (2016)[6], before starting to program, the facilitators (teachers or science mediators) set up the environment in which the pupil playing the robot will move. Using a sheet can make it easier to set out a landscape or scene because it enables real obstacles (chairs, tables, etc.) to be added or representations of obstacles to be drawn on the sheet. Another advantage of this method is that the precise distance corresponding to one step can be specified (using dots or a grid) and it enables very precise quarter turns to be made.

6 https://pixees.fr/dis-maman-ou-papa-cest-quoi-un-algorithme-dans-ce-monde-numerique-%E2%80%A8/

Figure 3. Robot game on a landscape created with a sheet.

In this environment, the places where the robot can and cannot walk are clearly explained, along with the commands it can execute. To begin with, absolute movements can be made by fixing the direction in which the pupil playing the robot is facing and making the pupil do translations but not rotations.

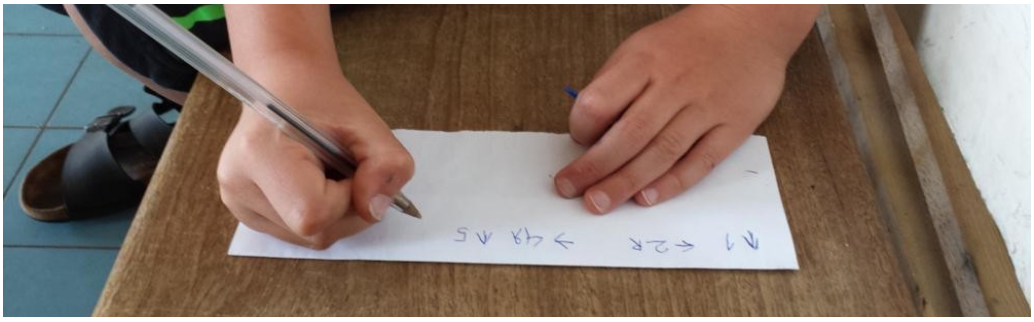

Figure 4. Sequence of movements shown with arrows.

Later on, the language can be changed to include rotations with the movements. This means that the movements become relative and the perspective is therefore changed. For the pupils playing the programmers, the programme can be made as a series of arrows, making it accessible even to children who have not yet learned to read. Once the pupils have understood the principle of writing a sequence of instructions and transmitting them to the pupil playing the robot, additional challenges can be introduced (crossing the river, getting to the woods, etc.). To vary the programs, objects can be placed at certain points in the landscape and instructions to pick up all the objects at the robot's feet can be added.

Figure 5. Designing programs for the robot.

To extend the learning objectives, bugs can be slipped into the programs. To begin with, pupils have to describe the program executed by the robot, then later on they have to correct the bugs in the program to achieve the objective. It is also interesting to translate a program from one language into another by switching from absolute movements to relative movements, and vice versa.

6 Conclusion

Learning computing is a necessity in the digital era if people are to switch from being digital consumers to critical, creative citizens (Romero et al., 2017). Computing can be taught in a highly procedural way or in ways that are more engaging and creative for the learners (Romero, 2016). Our experience is that teachers and science mediators are now using enlightened practices, their actions on the ground are assessed among their target audiences (satisfaction and measurement of what has been learned) and those involved in this scientific mediation reflect together on their practices and on how they can be improved.

However, it would be worth giving more sustained consideration to setting up research in the educational sciences on the development of computational thinking using different types of plugged and unplugged activities.

References

Baron, G.-L., & Bruillard, É. (2013). École et dispositifs technologiques: points de vue croisés. *Carrefours de l'éducation*, (2), 117–129.

Bell, T., Alexander, J., Freeman, I., & Grimley, M. (2009). Computer science unplugged: School students doing real computing without computers. *The New Zealand Journal of Applied Computing and Information Technology*, *13*(1), 20–29.

Bell, T., Witten, I. H., & Fellows, M. (1998). *Computer Science Unplugged: Off-line activities and games for all ages*. Computer Science Unplugged.

Calmet, C., Hirtzig, M., & Wilgenbus, D. (2016). *1,2,3 Codez*. Editions le Pommier.

Canellas, C., De La Higuera, C., Peinchaud, É., & Roche, M. (2016). Class' Code a un an... et c'est un commencement. *1024 – Bulletin de la société informatique de France*, (9), 19-24.

Curzon, P., Dorling, M., Ng, T., Selby, C., & Woollard, J. (2014). Developing computational thinking in the classroom: a framework.

Desgagné, S., Bednarz, N., Lebuis, P., Poirier, L., & Couture, C. (2001). L'approche collaborative de recherche en éducation: un rapport nouveau à établir entre recherche et formation. *Revue des sciences de l'éducation*, *27*(1), 33–64.

Duflot, M. (2016). Jouer à «robot-idiot» pour s'initier aux algorithmes. Pixees. Available at https://pixees.fr/dis-maman-ou-papa-cest-quoi-un-algorithme-dans-ce-monde-numerique-%E2%80%A8/

Duflot, M., Quinson, M., Masseglia, F., Roy, D., Vaubourg, J., & Viéville, T. (2015). When sharing computer science with everyone also helps avoiding digital prejudices. In *Scratch2015AMS*. Amsterdam, Netherlands. Available at https://hal.inria.fr/hal-01154767

Faber, H. H., Wierdsma, M. D., Doornbos, R. P., van der Ven, J. S., & de Vette, K. (2017). Teaching Computational Thinking to Primary School Students via Unplugged Programming Lessons. *Journal of the European Teacher Education Network*, *12*, 13–24.

Grover, S., & Pea, R. (2013). Computational Thinking in K–12 A Review of the State of the Field. *Educational Researcher*, *42*(1), 38-43. https://doi.org/10.3102/0013189X12463051

Hannan, L., Chatterjee, H., & Duhs, R. (2013). Object Based Learning: A powerful pedagogy for higher education. *Museums and Higher Education Working Together: Challenges and Opportunities. ed./Anne Boddington*, 159–168.

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

Joffredo-Le Brun, S., Morellato, M., Sensevy, G., & Quilio, S. (2017). Cooperative engineering as a joint action. *European Educational Research Journal*, 147490411769000. https://doi.org/10.1177/1474904117690006

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.

Moreno-León, J., & Robles, G. (2015). Dr. Scratch: a Web Tool to Automatically Evaluate Scratch Projects. In *Proceedings of the Workshop in Primary and Secondary Computing Education* (p. 132–

133). ACM.

Owen, K. B., Parker, P. D., Van Zanden, B., MacMillan, F., Astell-Burt, T., & Lonsdale, C. (2016). Physical activity and school engagement in youth: a systematic review and meta-analysis. *Educational Psychologist*, *51*(2), 129–145.

Resnick, M., Maloney, J., Monroy-Hernández, A., Rusk, N., Eastmond, E., Brennan, K., … others. (2009). Scratch: programming for all. *Communications of the ACM*, *52*(11), 60–67.

Resnick, M., & Siegel, D. (2015, November 10). A Different Approach to Coding. Available at https://medium.com/bright/a-different-approach-to-coding-d679b06d83a#.b9jcw2js5

Román-González, M., Pérez-González, J.-C., & Jiménez-Fernández, C. (2017). Which cognitive abilities underlie computational thinking? Criterion validity of the Computational Thinking Test. *Computers in Human Behavior*, *72*, 678-691.

Romero, M, Noirpoudre, S., & Viéville, T. (2018). Apprentissage du code ? Que disent les sciences de l'éducation. *Pixees*. Available at https://pixees.fr/apprentissage-du-code-que-disent-les-sciences-de-leducation/

Romero, M. (2016). De l'apprentissage procédural de la programmation à l'intégration interdisciplinaire de la programmation créative. *Formation et profession*, *24*(1), 87-89. https://doi.org/10.18162/fp.2016.a92

Romero, M. Lille, B., & Patino, A. (Éd.). (2017). *Usages créatifs du numérique pour l'apprentissage au XXIe siècle* (Vol. 1). Quebec: Presses de l'Université du Québec.

Romero, M.., & Loufane. (2016). *ViBot, le robot*. Quebec, QC: Publications du Québec.

Romero, M., & Netto, S. (2018). *Séminaire de recherche organisé par l'équipe pédagogique du master IME et le laboratoire TECHNÉ de l'université de Poitiers « Apprendre à programmer à l'école : représentations sociales et pratiques des enseignants »* [Video]. Available at https://www.youtube.com/watch?v=dTEKcW2bA7M

Romero, M., & Vallerand, V. (2016). *Guide d'activités technocréatives pour les enfants du 21e siècle* (Vol. 1). Quebec, QC: Livres en ligne du CRIRES. Available at http://lel.crires.ulaval.ca/public/guidev1._guide_dactivites_technocreatives-romero-vallerand-2016.pdf

Sander, E. (2000). *L'analogie, du naïf au créatif: analogie et catégorisation*. Editions L'Harmattan.

Sensevy, G., Forest, D., Quilio, S., & Morales, G. (2013). Cooperative engineering as a specific design-based research. *ZDM*, *45*(7), 1031–1043.

Shelton, C. (2016). Time to plug back in? The role of "unplugged" computing in primary schools. *ITTE*

Newsletter, (2016).
Sweller, J. (1994). Cognitive load theory, learning difficulty, and instructional design. *Learning and instruction*, *4*(4), 295–312.
Torp, L. (2002). *Problems as possibilities: Problem-based learning for K-16 education*. ASCD.
Tulving, E. (1972). Episodic and semantic memory. *Organization of memory*, *1*, 381–403.
Viéville, T., & Tort, F. (2013). Donner du sens aux éléments de technologie : l'exemple des URI. In B. Drot-Delange, G.-L. Baron, & E. Bruillard (Éd.), *Didapro5 - DidaSTIC : Didactique de l'informatique et des STIC en milieu éducatif*. Clermont-Ferrand, France: Université Blaise Pascal, Université Paris V, ENS Cachan/IFé. Available at https://hal.inria.fr/hal-00843963
Wohl, B., Porter, B., & Clinch, S. (2015). Teaching Computer Science to 5-7 year-olds: An initial study with Scratch, Cubelets and unplugged computing. In *Proceedings of the Workshop in Primary and Secondary Computing Education* (p. 55-60). ACM.