Teaching the first and only logic control course with HOME I/O and Scratch 2.0

B. Riera, A. Philippot, D Annebicque

Teaching the first and only logic control course with HOME I/O and Scratch 2.0

B. RIERA*, A. PHILIPPOT* and D. ANNEBICQUE*

** CReSTIC, UFR Sciences Exactes et Naturelles, University of Reims Champagne Ardenne, Moulin de la Housse, BP 1039, 51687 Reims - France (bernard.riera@univ-reims.fr).*

Abstract: This paper deals with teaching the first and only logic control course for 2nd year students (after the high school diploma) in computer science by using HOME I/O, the virtual house for control and STEM education and Scratch 2.0. HOME I/O is a fruitful collaboration between CReSTIC lab from the University of Reims Champagne-Ardenne and Real Games, a Portuguese company. In this paper, it is shown how HOME I/O combined with Scratch 2.0 has been used to present the main concepts of logic control. From a pedagogical point of view, an original approach based on learning from errors have been developed. After 2 years of experience, feedback from students and teachers have been very positive.

Keywords*:* logic control education, learning from errors, serious games.

1. INTRODUCTION

Since 2017, HOME I/O presented during the ACE conference in 2016 (Riera et al., 2016), has been able to be used together with Scratch 2 through scratch extensions. Even if Scratch is seen as a way to initiate young people to programming, in this paper we present how HOME I/O combined with Scratch 2 has been used to teaching the first and only logic control course in non-control engineering programs in order to introduce the main concepts of sequential logic.

The first part of the paper presents the curriculum of the course named "Control of automated processes" of 24 hours developed for 16 students in 2nd year of license in "informatics and data processing" at IUT ("Institute of Technology") at Reims Champagne Ardenne University (URCA). The main objective of this module for students is to give them basic knowledge and know-how to be able to develop a logic controller whatever the programming language. A learning from errors pedagogy has been used to have students felt and understood the specificities of control logic. The proposed control problems are simple, like the programming language used (Scratch) but without formal specification and methods to implement, the work is not as simple as it seems.

The second and third parts of the paper deals with HOME I/O and Scratch 2 extensions. The last part of the paper presents 2 introductory problems used to propose a teaching based on learning from errors.

The paper concludes with some feed-backs about this experience and gives some perspectives.

2. "LOGIC CONTROL OF AUTOMATED PROCESSES" COURSE FOR COMPUTER SCIENCE STUDENT IN IUT

In France, after a high school diploma, Institutes of Technology at the University (IUT) have the primary vocation to prepare students in two years for a university degree in 24 technology fields like computer science.

2.1 IUT computer science department

The 28 departments of computer science of the IUT in France train students to computing and to team work so as to prepare them to their integration in working life. It therefore offers a teaching supervised by available teachers, and professionals who teach students a practical and concrete experience of the computing line of work. The part dedicated to computer science of the teaching is based on 3 axes, each of them requiring specific qualities and skills:

- Tools and methods for software engineering that deals with specifications of customer requirements;
- Algorithmic and programming;
- Architecture system and network.

To those teaching objectives, other general academic disciplines are added for half of the teaching learning time (Mathematics, Economy, Communication...), that provide a multidisciplinary aspect to the training. Besides, the assessment is based on theory but also on practical situations, via supervised projects and a work placement in a company enabling to validate the knowledge and skills acquired during the training. At URCA, second year computer science students at IUT can follow a 24 hours course named "Logic control of automated processes".

2.2 Initiation to "Logic control of automated processes"

The main objective is to give to students, who know nothing about control systems and Programmable Logic Controller (PLC), the fundamentals about control of industrial systems, discrete event systems and logic controllers. Indeed, today the massive use of digital and information technologies like Cyber-Physical Systems (i.e. network of interacting elements with physical input and output instead of as standalone devices), Industrial Internet of Things (IIoT), Machine to Machine (M2M) communication, Big Data and Cloud Computing represent what it is called the fourth industrial revolution (Industry 4.0). One of the main challenges is to propose methodologies and tools matching the two worlds: IT (Information Technology) and OT Operational Technology). At least, computer science technicians and

engineers will have in the future to combine IT and OT know-how. The need for multiple hard and soft skills will become more and more important day after day. Employees will have to possess greater flexibility to adapt to new roles and work environments, and get accustomed to continual interdisciplinary IT and OT learning. It is why the area of computer science and control education, training and outreach have to evolve in order to be adapted to the requirements of the Factory of the Future and more generally to our society (Lamnabhi-Lagarrigue et al., 2017). This course has been developed having in mind this objective. The idea is to enable the students to feel and to understand some specificities of logic controller program development. In this course, voluntarily, we do not focus on PLC programming and the languages of the standard IEC 61131-3 which defines five programming languages for programmable control systems: function block diagram (FBD), ladder diagram (LD), structured text (ST), instruction list (IL; similar to assembly language), and sequential function chart (SFC). These languages, except for ST, are quite far from advanced languages learned and used by computer scientists. It is why in this course, we focuse on 2 major requirements:

- A first stage to formalize specification before coding the controller.
- A second stage of implementation, to code from formal specification a controller, whatever the programming language and whatever hardware (ARDUINO, PLC…) or software (soft PLC, Scratch…).

In the last 12 hours of the course, students should develop a complete soft PLC in C#. In order to enable computer science students to feel and understand characteristics of logic control of dynamic system, a "learning from errors" based approach has been carried out.

2.3 Learning from errors

Learning from errors is not new. For early educational reformer John Dewey (1859-1952): "Failure is instructive. The person who really thinks learns quite as much from his failures as from his successes". Failure is an opportunity for students to receive feedback on their strengths as well as their areas of improvement. When reframed as a good, constructive, and essential part of learning, failure is a master teacher. More, making errors is a natural part of the learning process. A lot of errors do not occur randomly, but originate from misconceptions (Nesher, 1987). Nevertheless, although errors should be eradicated, different theoretical approaches on learning see them as beneficial for learners when used in a constructive way (Wernecke et al., 2018). This pedagogical approach presents several advantages (Pappas, 2016):

1. The pedagogy of error promotes the personal development of learners who take more risks. Errors provide students with the opportunity to take risks that they would not have taken in a normal context. In the real world, they would not be as innovative or creative when approaching a problem, simply because they would be too afraid of the outcome or the consequences. Experiencing by making mistakes can help them in all areas of their lives.
2. The pedagogy of error promotes problem solving and critical thinking. When help manuals or tutorials are used, students have the opportunity to develop skills. Rather than simply following instructions, they must use their problem solving and thinking skills to come up with a practical and effective solution on their own.
3. The pedagogy of error reinforces the retention and understanding of knowledge. When students come to a conclusion or formulate a solution on their own, they are more likely to absorb this information and retain it in their long-term memory. Indeed, if teachers give them the answer directly, they will remember it for one or two minutes, then move on to the next module. On the other hand, if they have to manage to find a solution and do some research to obtain a correct answer, they will remember it thanks to the circumstances used to arrive at the solution. The educational experience itself becomes more memorable.
4. The pedagogy of error removes the limits created by the fear of failure. By developing courses based on the pedagogy of error, failure could be always an option for students. Being in a healthy learning environment where error is not a blemish, students will not be afraid of failure or the consequences of their mistakes. Mistakes are seen as full teaching tools rather than obstacles in the learning process. Students will be motivated to make efforts and answer questions, even if they are unsure of the correct answer. Thus, they can learn from their mistakes and thus rectify the situation, pushing them to broaden their knowledge base in a deeper and more active way.
5. Learning from errors allows learners to be more confident in general in all aspects of their lives. Mistakes provide students with confidence and self-esteem. They feel empowered to find their own solutions. With the support system in place when needed, the fear of error is dispelled.
6. The pedagogy of error helps students to make connections between ideas and concepts. Giving an incorrect answer is better than getting a good answer without doing anything. Indeed, students are more likely to remember "wrong" choices. Because of this, they are invited to continue to find the right solution by themselves. This creates a connection in their minds between the learning experience in itself and the idea or concept. The experience becomes even more dynamic and real.

However, learning from errors could be difficult in automatic control teaching. Hence, the use of real equipment is mandatory. However, it is not possible to allow students to make errors in the controller. Indeed, real equipment requires considerable room and regular maintenance, which has important costs and requires qualified people; last but not least, most applications of interest tend to be risky for the inexperienced students who are trying to making them work. Software simulation can help a lot in this subject. In fact, real time computer based simulation can be a risk free, affordable and easy to replicate training platform. In the specific case of industrial control and automation education and training, software simulations recreating industrial plants can replace physical target systems (Callaghan et al., 2009). This "synthetic" approach makes possible to get an inexpensive training environment that does not present any risk of injury

to man or damage to machines. Modern technology, most derived from computer games, is making them very realistic, low cost and increasingly easy to use and integrate with external devices. Video games can be a great tool, really adapted to control and STEM education (Mayo, 2009) (Arango et al., 2008) (Riera et al., 2009). It is with this objective that CReSTIC lab from the University of Reims Champagne-Ardenne and Real Games, a Portuguese company, performed in a 3-year R&D project (2011-2014) bringing a complete "virtual" house, called HOME I/O, into the classroom for control and STEM education (Riera et al., 2016). Hence, it is possible to propose healthy learning environment where error is possible and not a blemish.

In the course proposed to second year computer science students at IUT, we have used learning from errors at the beginning. These students knows very well how to code and programming languages. So it is proposed to the students to develop in Scratch 2.0, 2 simple programs to control one light and the garage door of HOME I/O. It is explained that HOME I/O and scratch 2.0 are used in high schools to learn computer programming. So, theoretically, second year computer science students at the university would not have any problems to design that piece of Scratch 2.0 code.

Before presenting the specifications of the 2 problems and the results obtained, the following of the paper gives some information about HOME I/O and its connection with Scratch 2.0.

3. HOME I/O: A VIRTUAL HOUSE FOR CONTROL AND STEM EDUCATION

HOME I/O (https://realgames.co/home-io/) is the result of «DOMUS» (2011-2014) a 3-year research and development project between the CReSTIC lab from the University of Reims Champagne-Ardennes and Real Games, a Portuguese company, which was partially founded by the French Ministry of National Education. HOME I/O is real time FPS simulation software (figure 1) of a smart house and its surrounding environment.

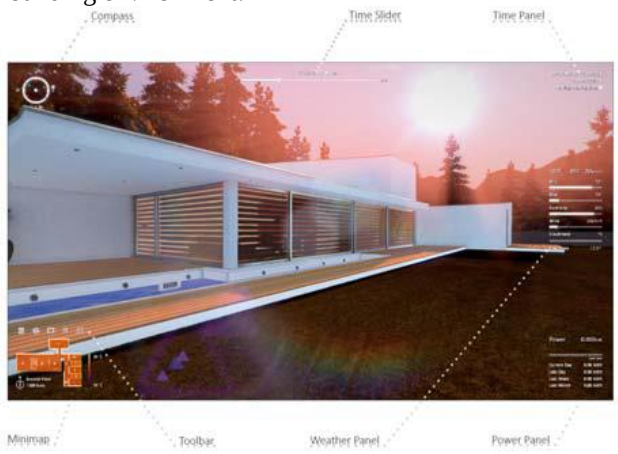

Fig. 1. HOME I/O

This software was built to cover a large spread of educational applications in technology and engineering sciences (Riera et al., 2016) (Philippot et al., 2017). More than a simulator, HOME I/O is a learning, experimenting and project development environment dedicated to the student from Middle schools, High schools and Universities. HOME I/O can be easily interfaced with third party utilities, software (LabView, Matlab Scratch 2.0…) or hardware (microcontrollers, programmable logic controllers…). In 2017, a native connection to Scratch 2.0, using a web server, has been implemented in HOME I/O. This update has extended the pedagogical possibilities of HOME I/O, particularly for high schools students.

4. HOME I/O WITH SCRATCH 2.0

Scratch is a well-known visual programming environment (figure 2) that allows users (primarily ages 8 to 16) to learn computer programming while working on personally meaningful projects such as animated stories and games. A key design goal of Scratch is to support self-directed learning through tinkering and collaboration with peers (Maloney et al., 2010). The Scratch project began in 2003, and the Scratch software and first web site were publicly launched in 2007. Scratch is free, available in nearly 70 languages, and more than 30 million users all over the world. In addition, Scratch software is often redistributed by school systems and educational organizations.

Scratch allows children to learn coding concepts and create interactive projects without needing to learn a text-based programming language. This means they won't be slowed down by their keyboard skills or the ability to remember complex code. Scratch is dynamic, it allows to modify the code of the program in progress execution. Multimedia oriented for teaching to the computer world children, Scratch treats with great ease the basic concepts of programming such as loops, tests, variable assignments, and especially the manipulation of objects, such as sounds and videos. Scratch is visual, all the code is directly written in the language kindergarten in the form of colored bricks (for example yellow controls, variables in red, movements in blue) placed inside scripts. Scratch is free and allows the teacher to spread his pedagogy through an almost playful interactivity of the objects manipulated by these software bricks. Key features of Scratch are liveness and tinkerability, making execution visible, no errors messages, making data concrete and minimizing the command set (Maloney et al., 2010).

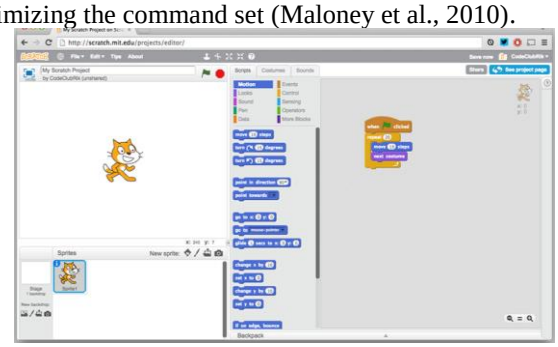

Fig. 2. Scratch 2.0 editor

Scratch uses a broadcast mechanism to support inter-scripts communication and synchronization. Any script can broadcast a message (an arbitrary string). A broadcast triggers all scripts in all scripts that begin with a matching "when I receive <msg>" trigger block (Maloney et al., 2010). Scratch lacks the explicit concurrency control mechanisms often found in other programming languages, such as semaphores, locks, or monitors. Instead, Scratch builds concurrency control into its threading model in a way that avoids most race conditions, so that users do not need to

think about these issues. This is done by constraining where thread switches can occur. In the Scratch model, a thread switch can occur in only two places: (1) on a command that waits explicitly (e.g., "wait 1 second") or (2) at the end of a loop. A thread switch cannot occur in the middle of a sequence of non-waiting statements, or between the test of an "if" command and its body. Although the Scratch threading model avoids most race conditions, it does not eliminate all concurrency issues (Maloney et al., 2010).

Scratch can either be used online in a web browser, or downloaded and used offline. Scratch 2.0 can be extended to control external devices (e.g. robotics kits, musical instruments) and to access data from external sensor hardware (e.g. sensor boards). A Scratch 2.0 extension extends Scratch with a collection of command and reporter blocks that can be used to interact with a particular device. When an extension is enabled, its blocks appear in the "More Blocks" palette. Due to browser security restrictions, Scratch 2.0 cannot interact with hardware devices directly. Instead, hardware extensions come with a helper app, a separate application that the user must install and run on their computer. Scratch 2.0, only with the offline editor, can communicate with a helper app via HTTP requests, and the helper app talks to the hardware. Scratch 2.0 sends commands to the helper app and the helper app sends sensor values and status information back to Scratch 2.0 via HTTP GET requests. Since the protocol is standard HTTP, any browser can be used to test and debug helper apps. Scratch 2.0 retrieve sensor values and status information from the helper app by sending a poll command. In response to a poll command, the helper app returns a list of (sensor name, value) pairs, one pair per line. Scratch 2.0 sends poll commands roughly 30 times per second. HOME I/O can be used together with Scratch 2.0 through scratch extensions. Data exchange between Scratch 2.0 and HOME I/O is done through a built-in web server in HOME I/O which is listening on port 9797. User may disable the web server or define a different listening port. A Scratch 2.0 template file includes the necessary HOME I/O extension blocks. After opening this file in Scratch, more blocks are available. In order to use HOME I/O devices in Scratch 2.0, they must be set to external mode first. A successful connection between Scratch and HOME I/O is indicated by a green light next to the HOME I/O (en) extension (cf. figure 3).

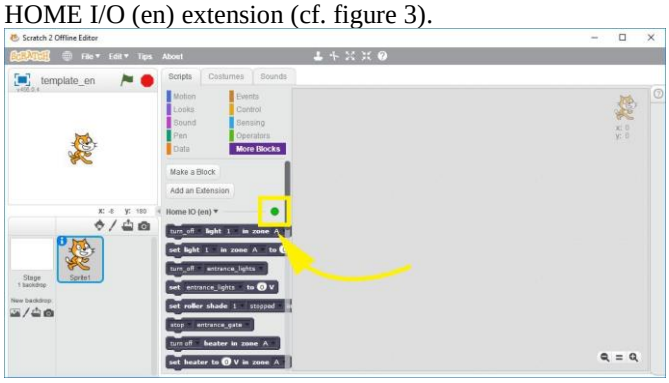

Fig. 3. HOME I/O blocks in Scratch 2 offline

For experts in logic controllers, this approach using helper app via HTTP requests, even if it is simple from a user point of view, raises several questions from a theoretical and pedagogical point of views. First, a sensor can change anytime, and a program is sequential. So, that can involve bad calculation and it is why an image memory is used in a PLC in order to work with constant values of I/O during a PLC scan time. In addition, the way in which communication is varied out between helper app and Scratch can be a source of problem and can involve a Scratch crash if variables are exchanged all the time (figure 4).

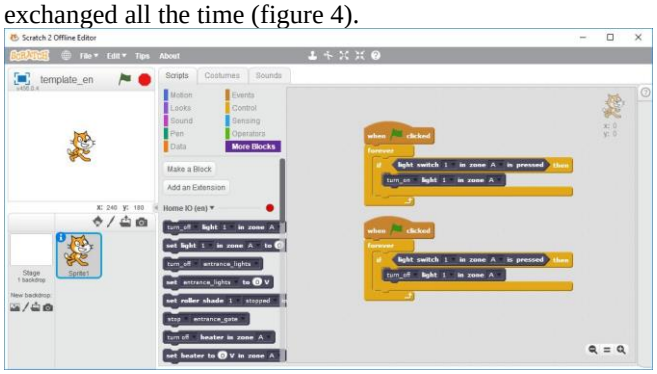

Fig. 4. Example of Scratch 2 crash

All the HOME I/O devices can be controlled with Scratch 2.0 blocks and all the data coming from HOME I/O sensors can be used in Scratch 2.0 scripts in a very easy way. For instance, the block (figure 5) enables to turn on the light number 2 in room D.

Fig. 5. Example of HOME I/O block in Scratch 2.0 offline

Hence, it is not possible to design a robust and efficient controller, if the behavior of the control part and the operative part is not well known. It is quite strange from a pedagogical point of view that these problems are not at all developed in the bibliography even if Scratch 2.0 extensions are popular to communicate with external hardware and software. It is why HOME I/O with Scratch 2.0 can be very useful to understand and learn logic control system.

5. LEARNING FROM ERRORS TO START WITH LOGIC CONTROL

Two simple control problems have been proposed to second year computer science students at IUT: light and garage door (figure 6).

1st Problem: one click on the remote control button 1 has to switch on the garage light if the light is off and to switch off is the light is on. Initially, the light is off.

Of course, for students on computer science, to understand and to use Scratch 2.0 is very easy. In addition, all of them are gamers, and the use of HOME I/O is natural. The main interests of this first simple example is to have the students understood different aspects of sequential systems:

- concept of Inputs (sensors) and Outputs (actuators) of a controller;
- gap between textual specifications and programming a controller;
- "forever" loop is necessary (to mimic the PLC scan cycle);

- differences between events and signals (rising edge or falling edge) and the necessity to use "repeat until" block instead of "if" in Scratch 2.0.

Fig. 6. 2 simple control problems with HOME I/O

This year, only 2 of the students performed to do a correct controller in less than 30 minutes (figure 7). With this example, students learned a lot from their errors and can by themselves understood fundamentals of logic control without talking about PLC. One can notice that none of them has proposed a solution based on communication messages between several scripts.

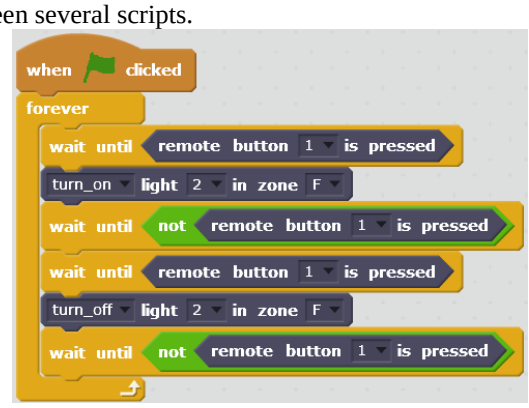

Fig. 7. Scratch 2.0 script for the light control

The second problem concerns the garage door and is more complex.

2nd problem: pressing button 1 on the remote control opens the garage door, after a delay of 5 seconds in the open position, the door closes. When closing, press button 1 again or, if the infrared sensor at the garage door detects a passage, the door opens again. This cycle is repeated as long as the garage door is not closed.

The main interests of this second problem is to have the students understood from their errors that:

- need of methods to formalize specification, like GRAFCET (IEC 60848, 2002);
- gap between formal specifications and programming and the difficulty to validate a controller.

This year, only 1 student has proposed a working "empiric" solution in 45 minutes.

After this introduction, students because of the encountered difficulties, are really interested in the content of the course. The second part of the course deals with the specification language GRAFCET (IEC 60848, 2002) used a lot in France. This standard is mainly for all people (design engineers, realization engineers, maintenance engineers...) who need to specify the behavior of a system (control-command of automatic machine, safety component...). This specification language should also serve as a communication means between designers and users of automated systems. The implementation of a specification described by GRAFCET isn't included in the scope of IEC 60848 standard. GRAFCET (the acronym of GRAphe Fonctionnel de Commande Etape/Transition or, in English, Step Transition Function Charts) is a graphical method for specifying industrial automation. Simple syntax, graphical representation, powerful and concise commands these are what make GRAFCET easy to learn and use. A good presentation of the main features of GRAFCET standard can be found in (David, 1995). The specification language described in the IEC 60848 standard differs from the SFC (Sequential Function Chart) proposed by the IEC 61131-3 standard (IEC, 2003), even if both are often named SFC in English and if models in these two languages may look similar; the differences stand both in syntax and semantics (Provost et al., 2011). The basic notions of GRAFCET are only presented in the course: terms and definitions, graphical representation of the elements (steps, transitions and links), graphical representation of sequential basic structures (cycle of single sequence, selection of sequences, activation of parallel sequences and synchronization of sequences), general principles (syntax and evolution rules). We particularly insist on transient evolution and exclusive activation of a selected sequence which is not guaranteed from the structure.

The third part of the course deals with the implementation. First, we discuss about the different ways to implement a Grafcet with Scratch 2.0. A method based on "messages", not presented in this paper, is proposed to convert a Grafcet in Scratch Scripts. This method is applied to the garage door problem (figure 8). A second solution, based on the calculation of transitions, step variables and actions is proposed with constant I/O values during a cycle time. In fact, students implement it in Scratch 2.0 and program the behavior of a PLC: scan inputs, execute program logic and update outputs (figure 9). This solution can be used whatever the controller hardware or software.

In the last part of the course, which is not described in this paper, students in 12 hours develop a full controller in C# for HOME I/O, managing lights, garage door and entrance gate after proposing a GRAFCET specification.

5. CONCLUSION

In this paper, it has been shown how a first and only course in logic control for 2nd year computer science at IUT is performed by using HOME I/O and Scratch 2.0. A learning from errors pedagogy has been used to have students felt and understood the specificities of control logic, without dealing with PLC. The proposed control problems are simple, like the programming language (Scratch 2.0), but without formal specification (GRAFCET) and methods to implement, the work is not so easy at all (Pichard et al., 2018). The use of HOME I/O as a pedagogic digital twin is really adapted to a learning from errors pedagogy. This course has been a success for 2 years, students are interested and motivated as proved by questionnaires that will be presented during the IFAC ACE 2019 conference. However, teachers have to show to students that controller design based only on errors corrections is not the right way to work, at all.

ACKNOWLEDGMENT

Parts of the work presented in this paper are carried out in the context of the HUMANISM No ANR-17-CE10-0009 research program, funded by the French ANR "Agence Nationale de la Recherche".

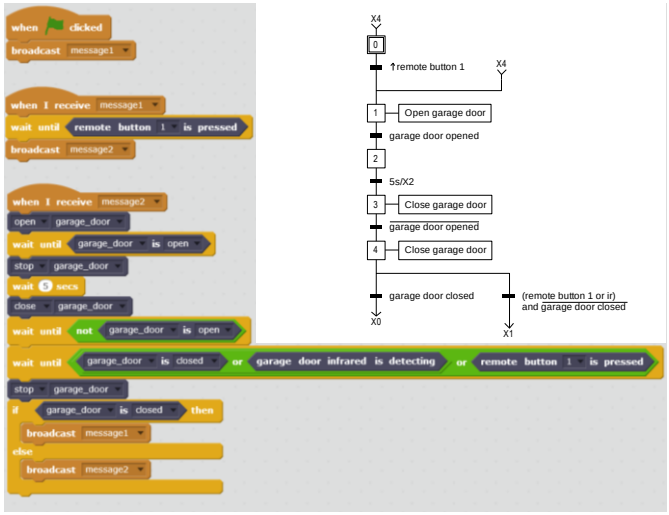

Fig. 8. Grafcet specification and Scratch 2.0 script (1st solution) for the garage door

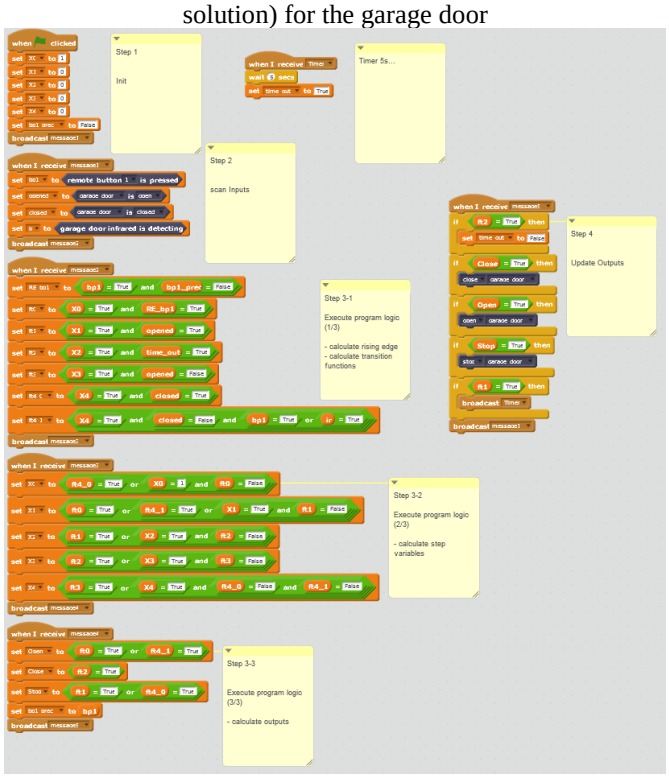

Fig. 9. Scratch 2.0 script (2nd solution) for the garage door

REFERENCES

Arango, F, Aziz, E.-S., Esche, S.K., Chassapis, C. (2008). A review of applications of computer games in education and training. Frontiers in Edu. Conf.. Pp. T4A.1-4A.6.

Callaghan, M. J. McCusker, K. Losada, J. L. Harkin, J. G. and Wilson, S. (2009) "Teaching Engineering Education Using Virtual Worlds and Virtual Learning Environments," in 2009 International Conference on Advances in Computing, Control, and Telecommunication Technologies, 2009, pp. 295–299.

David, R. "Grafcet: a powerful tool for specification of logic controllers," IEEE Transaction on Control Systems Technology, vol. 3, no. 3, pp. 253–268, 1995.

Lamnabhi-Lagarrigue, F., Annaswamy, A., Engell, S., Isaksson, A., Khargonekar, P., Murray, R., Nijmeijer, H., Samad, T., Tilbury, D., Van den Hof, P. (2017) "Systems & Control for the future of humanity, research agenda: Current and future roles, impact and grand challenges". Annual Reviews in Control 43 (2017) 1-64.

Maloney, J., Resnick, M., Rusk, N., Silverman, B. and Eastmond, E. (2010). "The scratch programming language and environment". Trans. Comput. Educ.,10:16:1–16:15, November 2010.

Mayo, M. J. (2009).Video Games: A Route to Large-Scale STEM Education? Science. 2009 Jan 2; 323(5910):79-82. doi: 10.1126/science.1166900.

Nesher, P. (1987). Towards an Instructional Theory: The Role of Student`s Misconceptions. For the Learning of Mathematics 7,3: pp 33-40, Publishing Association Montreal, Quebec, Canada.

Philippot, A. Riera, B., Koza, M., Pichard, R. Saddem, R., Gellot F., Annebicque, D. and Emprin, F. HOME I/O and FACTORY I/O - 2 Pieces of innovative PO simulation software for automation education. The 27th European Association for Education in Electrical and Information Engineering Annual Conference (EAEEIE 2017), Grenoble, France, June 2017.

Pichard, R., Philippot, A., Saddem, R., Riera, B. Safety of Manufacturing Systems Controllers by Logical Constraints With Safety Filter. IEEE Transactions on Control Systems Technology:1 - 9, 2018. 10.1109/TCST.2018.2827329.

Riera, B. Marangé, P., Gellot, F., Nocent, O., Magalhães, A., Vigário, B. (2009). Complementary usage of real and virtual manufacturing systems for safe PLC training. 8th IFAC Symposium on Advances in Control Education (ACE09). Kumamoto, Japon, October.

Riera, B., Emprin, E. Annebicque, D., Colas, C., Vigario, B. (2016). "HOME I/O: a virtual house for control and STEM education from middle schools to Universities". 11th IFAC Symposium on Advances in Control Education ACE 2016, Bratislava (Slovakia), 1-3 June 2016

IEC INTERNATIONAL STANDARD 61131-3 (2003). Second edition 2003-01, Programmable controllers – Part 3: Programming languages. Reference number CEI/IEC 61131-3: 2003.

IEC INTERNATIONAL STANDARD 60848 (2002). Second edition 2002-02, GRAFCET specification language for sequential function charts. Reference number CEI/IEC 60848: 2002.

Pappas, C. (2016). Les 7 avantages de la pédagogie de l'erreur en eLearning, https://elearningindustry.fr/pedagogie-de-lerreur-avantages

Provost, J., Roussel, J.-M., Faure, J.-M. (2011). A formal semantics for Grafcet specifications. 7th IEEE Conference on Automation Science and Engineering (IEEE CASE 2011), Aug 2011, Trieste, Italy. pp.488-494, 2011.

Wernecke, U., Schütte, K., Schwanewedel, J., Harms, U. "Enhancing Conceptual Knowledge of Energy in Biology with Incorrect Representations". CBE Life Sci Educ. 2018 Spring;17(1). pii: ar5. doi: 10.1187/cbe.17-07-0133.