Possibilities of Simulating Robot Generations in Public Education

ABONYI-TÓTH Andor

Abstract. An important part of teaching robotics is describing different robot generations, which represent the characteristic stages of technological development. These generations are also identifiable in case of educational robots. The categories refer to the tasks and problems which can be solved with a given robot. However, the development of robot generations can also be presented to students with the help of algorithmization and coding tasks which simulate the behaviour of robots in a programming environment. These activities can prepare students to work with real educational robots or complement the activities they do with them. This article summarizes the opportunities and advantages of the simulation of robot generations, and shows a concrete example of their implementation.

Keywords: Robotics, Robot Generations, Simulation, Scratch, RoboMind**,** VEXcode VR, Virtual Robotics Toolkit, MakeCode for LEGO MINDSTORMS Education EV3

1. Introduction

When introducing students to robotics, it is worth assessing the existing knowledge children have about the topic. For that, first we have to define the equipment which children perceive as robots. It can be done in a brainstorming session when we collect the robots they know of and mention some tasks those robots can perform. [1]

In these sessions, students often mention everyday objects (e.g. robot vacuum cleaners, robot lawn mowers, humanoid toy robots, industrial robots in an assembly plant, etc.) and fictional robots existing only in films and books (e.g. shape-shifting robots, doctor robots, combat robots). Of course, it is worth clarifying which robots actually exist, and which aspects of fictional robots are rooted in reality.

The robots collected by students can then be grouped into a number of categories, such as location of use, complexity of activity, size, nature (e.g., humanoid), and so on. We can come up with a diverse classification by simply focusing on the activities robots can perform. The groups may include robots performing simple or more complex tasks. At this point we can tell students that robots can be classified into different generations based on their technological level. [2]

2. Robot generations

The primary characteristics of first-generation robots is that they perform their actions according to a specific program and are unable to sense their environment. In this category are, for example, some industrial robots with pre-programmed path of movement and activities. In this case, if a workpiece is positioned incorrectly, the robot will not be able to perform its task under the changed conditions.

Second-generation robots are more advanced. They have a variety of sensors, so they can modify their movements, activities and operation using the information gained of their environment. For example, they can avoid an obstacle in their path.

Third-generation robots have even more advanced artificial intelligence, they can recognize shapes and situations, and they can solve certain tasks through machine learning. They can solve certain

tasks independently in an unfamiliar terrain; Mars rovers, for example, even find their way on an alien planet.

3. Robot generations in education

Educational robots have several generations too.

3.1 First-generation robots

First-generation robots include, for example, floor robots whose path of movement can be programmed by the physical buttons on them (forward, backward, turn right or left, wait), or by an external application. These robots travel along the specified tracks according to the pre-programmed code. They can be used with the original paths supplied with the robot or new paths programmed by the students. Various tasks and problems can be associated with the paths. Floor robots are ideal for developing students' algorithmic thinking in the lower grades of elementary school, and they can help in the playful acquisition of directions, numbers, shapes, colours, and so on. [3]

Figure 1: Blue-bot floor robot[1]

Figure 2: An example floor mat[2]

Certain floor robots also have simulation software which can be used to design and simulate floor robot activities on an interactive whiteboard. In the "Focus on Bee-Bot 3" application available for Bee-Bot robots, the robot can move on different maps and tracks, with its movement programmable with the buttons appearing on the screen. The actions of the robot can even be performed step by step, or children can follow the traversal of the path from the robot's point of view, as if a camera were attached to the robot. The software includes a number of challenges which children can solve on their own or in groups.

[1] https://bit.ly/3avl4qr

[2] https://bit.ly/2x09w0A

[3] https://www.focuseducational.com/category/item/6

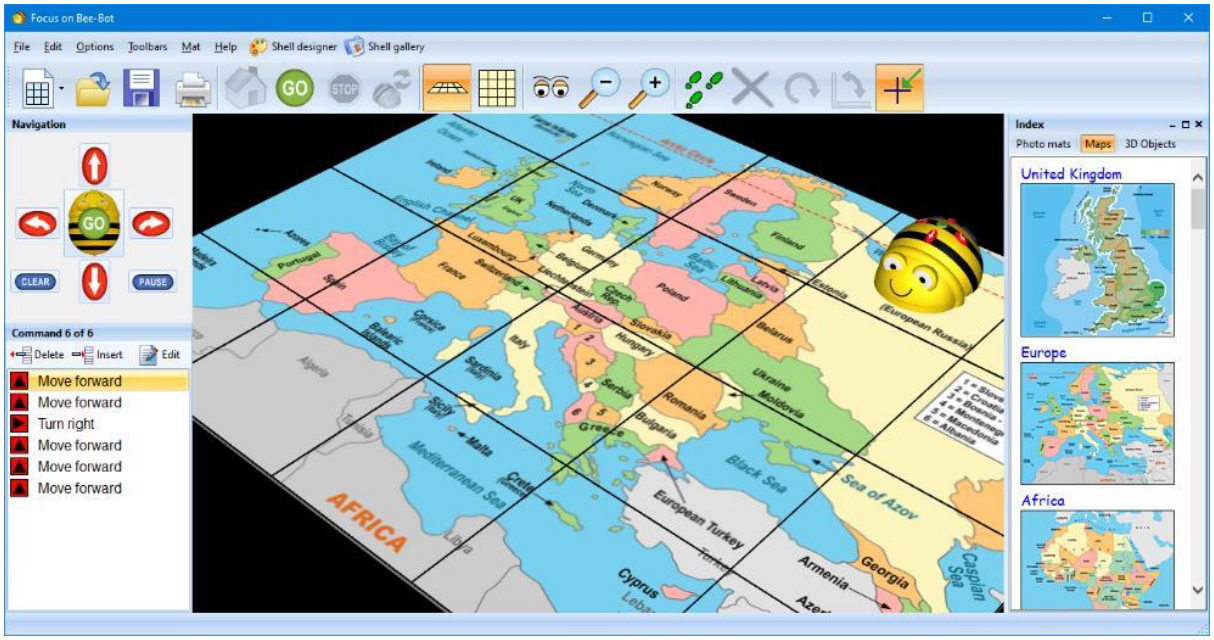

Figure 3: Focus on Bee-Bot 3 software with virtual floor robot and map

3.2 Second generation robots

Second generation robots are robots and kits which can process the data acquired by their sensors for completing the tasks. There is a wide variety of these educational robots. Which kit to choose may depend on the age group, the type and the features of the programming language, the quantity and quality of the related teaching materials, the available financial resources, and the specifics of the teaching task. [4]

If students have already become familiar with programming the deservedly popular micro:bit microcontroller during their studies, it may be worthwhile to obtain kits based on this chip. With those, students can use the familiar device and programming interface to solve robotics-related tasks.

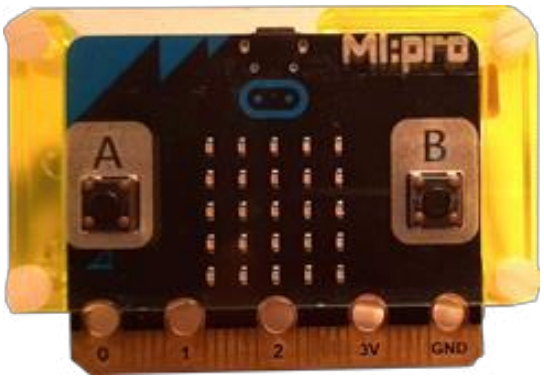

Figure 4: BBC micro:bit microcontroller

The kits differ in the range of sensors included in their basic package and available as optional accessories.

The following sensors are most commonly found in the kits: [5]

- Ultrasonic sensor: it measures the distance between objects or detect the ultrasound emitted by another source or reflected by a surface. It is great for measuring the distance between two robots (vehicles), making automatic doors, building a parking radar, etc. It is also suitable for detecting ultrasound from other sources. For example, a robot may respond to a nearby robot emitting ultrasound.
- Colour sensor: it recognizes various colours, which allows for the creation of a robot which can sort objects of different colours, follow a line or a more advanced one which solves the Rubik's cube.
- Light sensor: measures the intensity of the reflected light or ambient light. It can be used, for example, to enable the robot to stop at the edge of the table when it no longer detects the light reflected from the tabletop.
- Touch sensor: detects when a button is pressed or released. For example, we can make a robotic hand which closes when something touches it.
- Gyro sensor: The gyroscopic sensor is used to detect the angle of rotation of the robot. It can be used, for example, for building a balancing robot.

Figure 5: EV3 Touch sensor, Colour sensor, Ultrasonic sensor4

Robotic vehicles equipped with the required sensors may also be suitable for becoming familiar with second-generation robots. For example, a Bit:bot vehicle for use with a micro:bit chip includes two drive motors with rubber-coated wheels, two line tracking sensors, an analog light sensor, and an ultrasonic distance sensor at the front of the vehicle. Thus, the tool can be used to complete both line tracking and obstacle avoidance tasks.

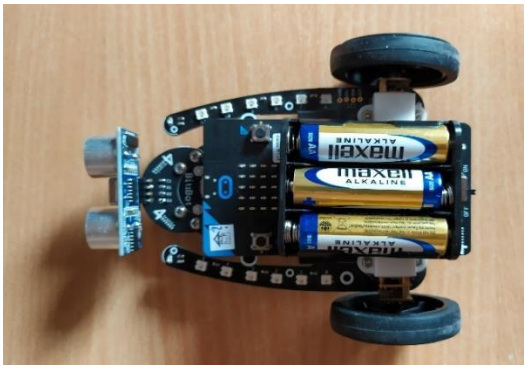

Figure 6: micro:bit controlled Bit:bot vehicle[5]

[4] http://www.legoengineering.com/ev3-sensors/

[5] Photo by Bence Gaál

3.3 Third-generation robots

Third-generation robots play a role in research and development related to artificial intelligence and therefore require high-level programming skills. They have high computing power and allow the use of sensors with advanced functionality.

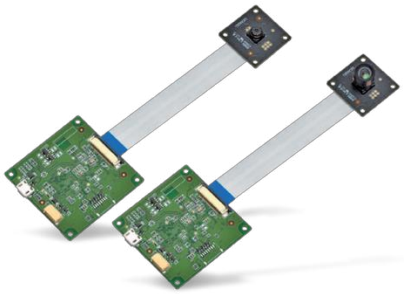

Figure 7: Image Sensor Module[6]

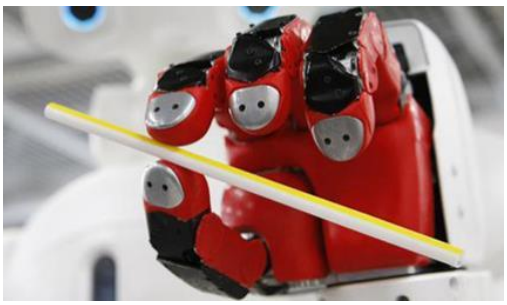

Figure 8: Tactile Sensor[7]

The development of self-driving vehicles requires good quality distance measurement, object recognition, visual and proximity sensors. The acoustic sensors allow the robot to receive voice commands or process sound effects from the environment. Third generation robots typically have a camera mounted on them, whose image must be processed. The processing tasks may include shape and emotion recognition, which can be very important in the case of robots doing social work.

These robots are typically used in institutions above the secondary school level, in tertiary and adult education. For this reason, we do not address this topic in more detail in our article. The third generation robots can also be simulated in a complex environment such as the Webots simulator.[8] The simulator can be used for modelling and programming, and the simulation of the robot's behaviour. [6]

Figure 9: Insect-shaped robot[9]

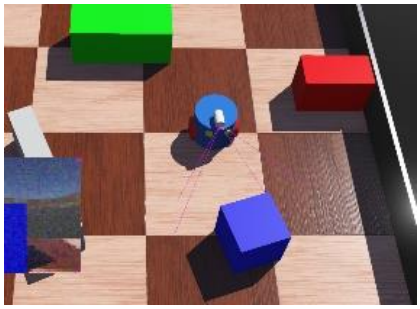

Figure 10: MyBot uses a camera to detect coloured objects[10]

[6] https://hu.mouser.com/new/omronelectronics/omron-b5t-hvc-p2-sensors/

[7] https://pressureprofile.com/nl/robotics

[8] Webots. Commercial Mobile Robot Simulation Software. https://cyberbotics.com/

[9] https://cyberbotics.com/doc/guide/samples-demos

[10] https://cyberbotics.com/doc/guide/samples-devices

4. The potential benefits of simulating robot generations in public education

The development of robot generations can even be presented to children through creating algorithms and writing code by which we can simulate the behaviour of a robot. Such activities can be good complements of working with real educational robots. Some of the activities can even be performed in an 'unplugged' way.

4.1 Unplugged activities

„*Teaching Unplugged is the term used for a teaching method and philosophy which has three primary aims: teaching through conversation, taking out external input such as course book and technology and letting the lesson content be driven by the students rather than being pre-planned by the teacher.*" [11]

Let's see an example of this method. The following task can be read in the Digital Culture textbook used in the 5th grade of Hungarian public education system: [7]

"*Go to a place with a large free area. There, mark a starting point and an end point. Form groups of 4 to 5 students.*

Each group should have a student whose eyes are blindfolded. Create an algorithm that you think will guide your partner from the starting point to the destination. Give verbal instructions to your partner according to the algorithm. What are the experiences? Did your partner actually get to the destination? Now we issue the instructions so that our partner travels exactly the way back. Did your partner get back to the starting point?

Discuss what important lessons you learned about control during the game. How could the instructions have been made clearer? Think of similarities and differences in the control of the real robots compared to this game."

By solving the above task, children can gain many experiences. It is possible that

- the 'robot' did not perform the expected activity because the instructions were not clear;
- the 'robot' could not interpret the issued instruction (e.g. did not understand it due to noise);
- the 'robot' executed the instructions correctly, but the algorithm contained errors;
- although the 'robot' successfully reached its destination and made its way back, it did not reach the starting point;

The experience gained during the 'unplugged' activities can be used in the programming of robots simulated in real or computer environments. It may happen that the real robot does not perform the expected activities (e.g. due to incorrect code and/or algorithm), does not execute the instruction (e.g. a voice-controlled robot does not interpret the command due to noise, or the command contains a syntax error). The robot's movement may also be influenced by external factors. For this reason, the robot does not necessarily return to its starting position, even if the program for traversing the whole path is correct. This can be caused by a variety of factors, such as an uneven or slippery surface, discharging batteries, etc.

The largest collection of unplugged activities for use in computer science education can be found on the CS Unplugged website. "*CS Unplugged 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.*"[12]

[11] https://www.teachingenglish.org.uk/article/teaching-unplugged

[12] https://csunplugged.org/en/

The site also contains session descriptions for getting acquainted with robotics. An example is "Harold the robot". [8] The session is designed to program the "robot" to build a tower from the building blocks placed on the table, by executing elementary instructions. The role of the robot can be played by the teacher or a student. Students need to figure out what elementary instructions the robot responds to and how they can be used to build the tower. After completing the task, students can gather together which commands the robot has responded to and which it has not, and discuss how more complex tasks can be accomplished using a series of elementary steps.

Of course, the sessions can also be organized so that students are familiar with the basic instructions the robot can execute. The students' task is to place the instructions in the correct order to solve the given problem. The task can be done with the instructions written on pieces of paper, which can also be attached to a white board with magnets. The simulated activity can be, for example, controlling a robotic arm. [9]

Also popular are classes in which students create algorithms with which a robot can build different patterns by stacking plastic cups on top of each other (My Robotic Friends) [10]. During the session, the students take turns creating the algorithm and playing the role of a robot for the execution of the algorithm defined by symbols. In this way, they learn the relationship between the operation and the symbols, the difference between the algorithm and the program, and also gain proficiency in debugging the code.

4.2 Activities in a simulated environment

Activities in a simulated environment allow students to gain sufficient knowledge and experience in the field of programming, so that they could later program physically existing, more complex robots later.

If the school has fewer robots than would be necessary for the sessions, then while one group is programming a real robot, the other can work in the simulated environment and then they can switch.

If the school does not yet have the robots, the basic concepts of robotics can be introduced in the simulated environment. There are several initiatives available for Hungarian schools (e.g. 'wandering robots' [11], 'wandering micro:bits' [12], in which schools can borrow various devices (e.g. floor robots, micro:bits) free of charge. More advanced robotics kits are also expected to be available for borrowing in the future. As these devices are only available in a school for a limited time, students should be taught their programming in advance in simulated environments, so that they could use the kits efficiently to solve advanced tasks when they have access to them.

Simulations allow us to set homework assignments which students could not otherwise do or try out in the absence of the necessary equipment.

It is much easier and faster to define paths for a particular type of problem in a simulated environment than in a real one. The paths can be drawn by the students, so besides completing a programming task, they use their image editing skills too. They can also customise the shape of the robot if the program has that option. The students can even use an animated figure as a robot, whose creation requires creativity and a better knowledge of the equipment.

Moreover, we can define a task in which the robot's program is given, and create a path individually or in a group which meets the conditions set in the task (e.g., the robot must collect the fallen fruit, but avoid poisonous mushrooms.)

Another advantage is that we can try different algorithms and problem-solving strategies without risking causing damage to real, sometimes very expensive devices. While collisions make no difference in a simulation, they can significantly shorten the lifespan of real robots. The methods and solutions tested in the simulator can then be used for the programming of real robots.

The route travelled by the robots can be easily illustrated by the robot drawing a line after itself. Of course, this can be done in a real environment by attaching a felt-tip pen to the robot, and the mat can be made of an easily erasable material. The advantage of this method is that the routes traversed can be easily compared, illustrating that a given problem can be solved in several ways.

If the students have already gained experience in using floor robots and we want to introduce them to block programming environments, we can add introductory tasks which rely on their knowledge gained in the robotics sessions. This way we can teach them the basics of programming and using various control structures.

The simulator allows teachers to better present certain phenomena (e.g. operating model of line follower robots, traversal algorithms, obstacle avoidance algorithms) to the whole class.

If students do not have a background in robotics but have already used block programming environments (e.g. Scratch), we can set novel, playful problem-solving tasks from the field of robotics.

We can also teach functions which are not supported by the available real robots, such as picking up/laying down objects on a track, programming loading and rearrangement tasks.

Students can also be introduced to programming languages which go beyond block programming (e.g. LOGO, Robo).

Keep in mind, however, that programming physically existing and moving robots is a highly motivating activity for children, so simulations should not be used instead of that, but as complementary method. It is very important that students have hands-on experience with real robots in their studies and, if possible, solve real-life, open problems in project work. [13]

4.2.1. Simulation in Scratch 3.0 environment

The Scratch environment can be used as block programming[13] environment to simulate the activities of robots. A simulation environment should be designed from a top view of the robot and the course. The shape of the robot drawn must clearly indicate what direction it is facing.

In the Scratch environment, students can draw the shape of a robot and create animated shapes. The path that the robot must travel along can be easily drawn using either built-in or external drawing tools. The tracks and the robot images can be easily replaced.

The robots can initially be controlled with keystrokes (e.g. cursor keys) and then we can switch to writing programs which allow the creation of more advanced control structures.

The distance unit parameter must be set for each track to allow the robot to move a certain distance or turn when necessary. Parameters are crucial in the programming of real-life robots, so students should learn about them in the simulated environment as well.

Using the pen plug-in, the robot can draw a line after itself, so the users can keep track of its route.

[13] https://scratch.mit.edu/

Sensors can also be simulated. The robot can be programmed to stop if it encounters an obstacle. For that is advisable to draw the obstacle/wall with a specific colour. In the Scratch environment we can ask if our robot has encountered a line with a given colour during its movement. It allows us to simulate the movement of a second generation robot which moves forward until it detects an obstacle. A common task in teaching robotics is when the robot must be programmed to follow a line. It can be simulated in the Scratch environment, which is described in a later chapter in more detail.

With the Video sensing plugin available in Scratch, we can present creating applications with the help of a webcam image, in which we can control participants with our movement or detect the colours appearing in the video image. This could be a transition to simulating third-generation robots.

4.2.2. Simulation in the RoboMind environment

ROBO is an imperative/procedural programming language. With the help of this language, you will gain an insight into areas such as robotics and artificial intelligence. The language consist of basic instructions to control the robot, repetition loops, conditional if ... then ... else statements, the possibility to define instructions yourself by creating procedures.

In Robo, we can program in a development environment called RoboMind.

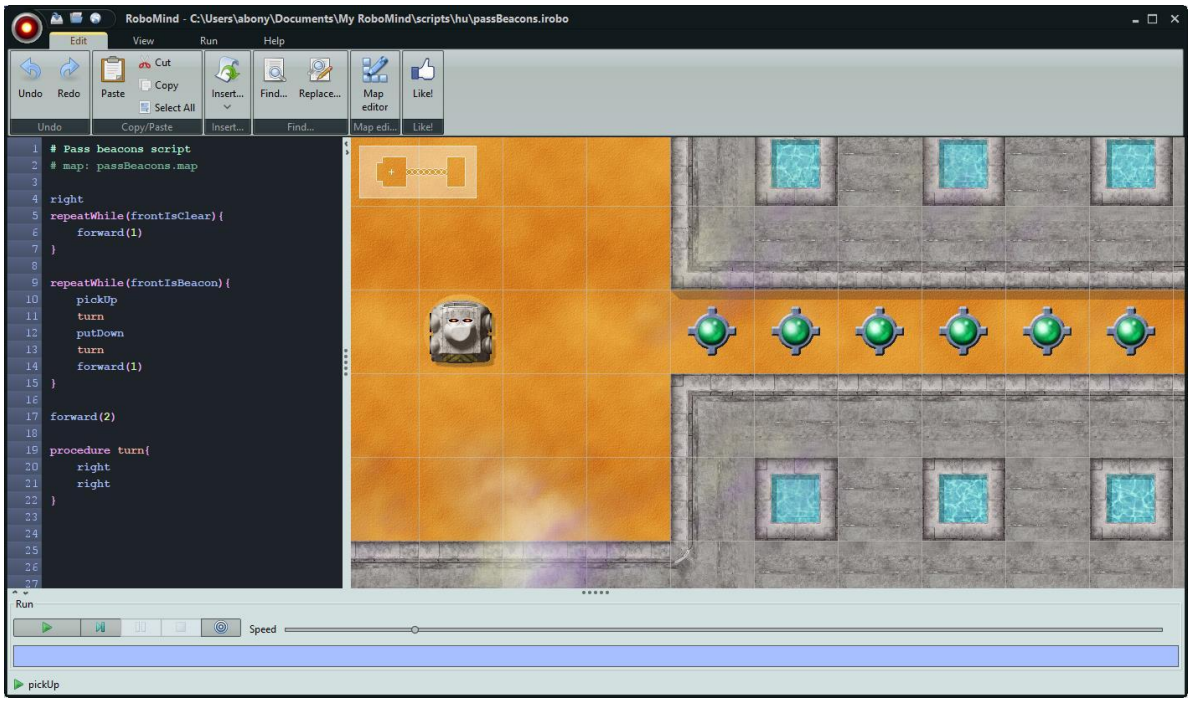

Figure 11: The RoboMind development environment

The virtual robot can navigate in a square grid environment by moving forward, backward, and turning left and right by 90 degrees. The directions can also be set according to the four cardinal points.

We cannot draw the robot in this environment, but we can choose from several existing shapes. The course can be easily created with the help of the built-in map editor. On the course we can

place a wall, various landmarks (crates, plants, water surface), a beacon, and routes marked in white or black.

The advantage of the environment is that the simulated robot can also be controlled with a remote control displayed on the screen (forward, backward, right, left), so that tasks that may be important for younger children can be solved initially, without programming.

The robot can mark the traversed route in white or black. In addition, it can pick up / put down objects (beacons) on the track or "eat" them, which means that it deletes the object after picking it up.

The robot can detect whether there is an obstacle to the left, right or in front of it, whether the path is clear, whether there is a beacon, and whether the road is painted white or black.

The environment is suitable for setting up tasks typical of first-generation robots. Examples are route and area exploration tasks (knowledge of the course, pre-programmed), rearrangement tasks (rearrangement objects of known location and number). The behaviour of second-generation robots equipped with sensors can also be simulated through various tasks. Examples are path finding tasks (the path to be followed is unknown), getting out of a maze, search tasks (finding and collecting objects on an open course or one containing few obstacles), rearrangement tasks (with the position and/or number of objects previously unknown), traversal tasks (comparison of random and systematic traverses). [14]

4.2.3. Simulation in the VEXcode VR environment

VEX Robotics released the VEXcode VR simulation environment in April 2020. It can be used in a browser, so there is no need to install special software. The website allows virtual use of the VEXcode environment used to program VEX 123, GO, IQ and V5 robots. The interface supports both block programming and programming in Python, so it is versatile for use in public education.

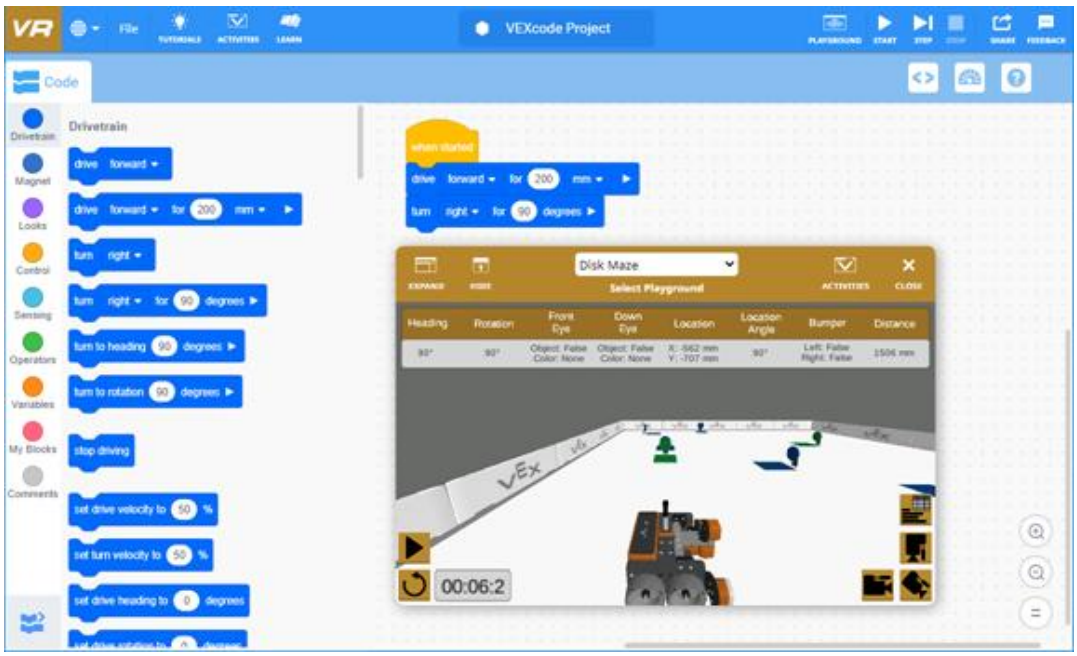

Figure 12: VEXcode VR environment

The virtual robot is placed in a playground where it executes the specified code. There are several types of playgrounds: a grid, a maze, a castle floor plan, a coral reef, etc., which are associated with different types of problem-solving tasks. The tasks are also accessible on the website. The track and the robot can be displayed in 2D (top view) or in 3D, with several camera views are available for the latter.

The environment can be used to simulate the activities of both first and second generation robots. Sensors include a rotation sensor, left and right impact sensors, a distance sensor, front and bottom colour sensors/brightness sensors, and a location sensor.

4.2.4. Simulation in the Virtual Robotics Toolkit environment

The Virtual Robotics Toolkit is a development environment for LEGO® MINDSTORMS® robots that can be installed on Windows/Mac operating systems. It can be used with a subscription.

The virtual robot can be programmed in the LEGO® MINDSTORMS® EV3 environment. The physical parameters can be modified in the simulation, so it is possible to simulate how the robot would operate in a state of weightlessness. 3D models can also be imported in the environment.

The following sensors can be used in the simulated environment: MINDSTORMS EV3 – Ultrasonic sensor, Color sensor, Touch sensor, IR sensor, Gyro Sensor, HiTechnic – Infrared sensor, Compass sensor.

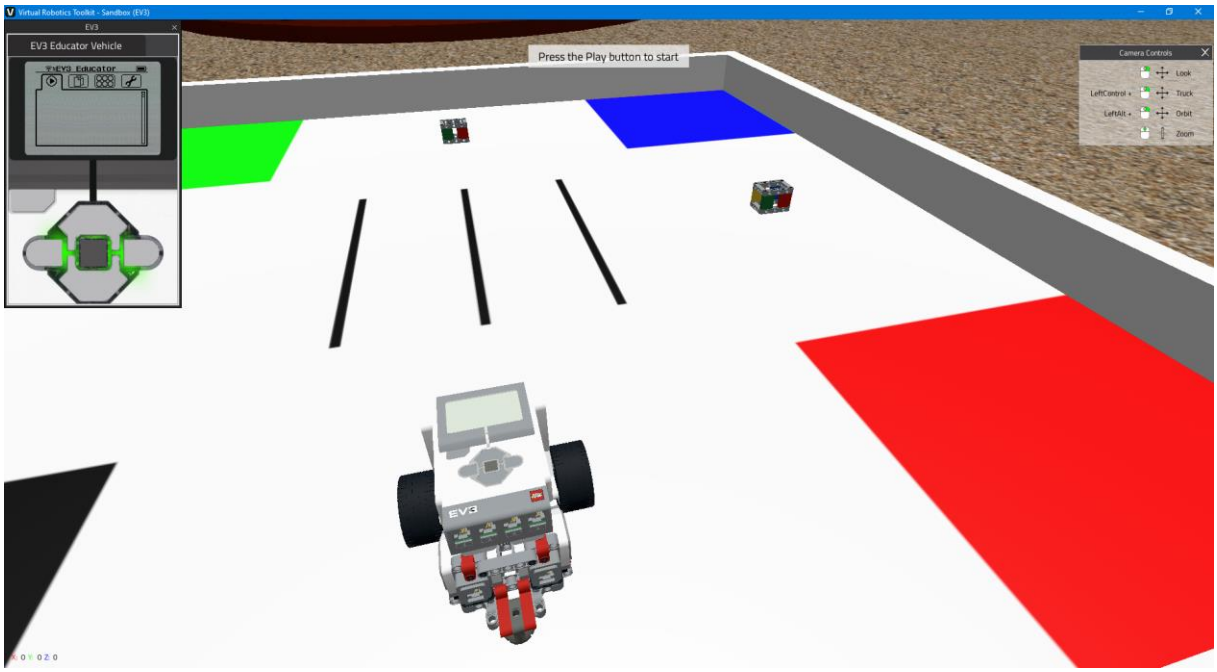

Figure 13: Virtual Robotics Toolkit environment

4.2.5. Simulation in MakeCode for LEGO® MINDSTORMS® Education EV3 environment

LEGO® MINDSTORMS® EV3 robots can be programmed using an online, free environment supporting[14] both block programming and JavaScript programming.

The interface has a simulation option, but this is different from the ones seen before. It is not a robot moving in a virtual space, but an EV4 brick (the brain of the device) to which different components can be connected, e.g. touch sensors, colour sensors, ultrasonic distance sensors, gyro sensors, infrared sensors, as well as large and medium-sized motors.

In the simulator, we can set the values measured by each sensor, so we can test whether our program is working properly (in principle), and we can also see when a motor is switched on/off, and check its rotation direction and power.

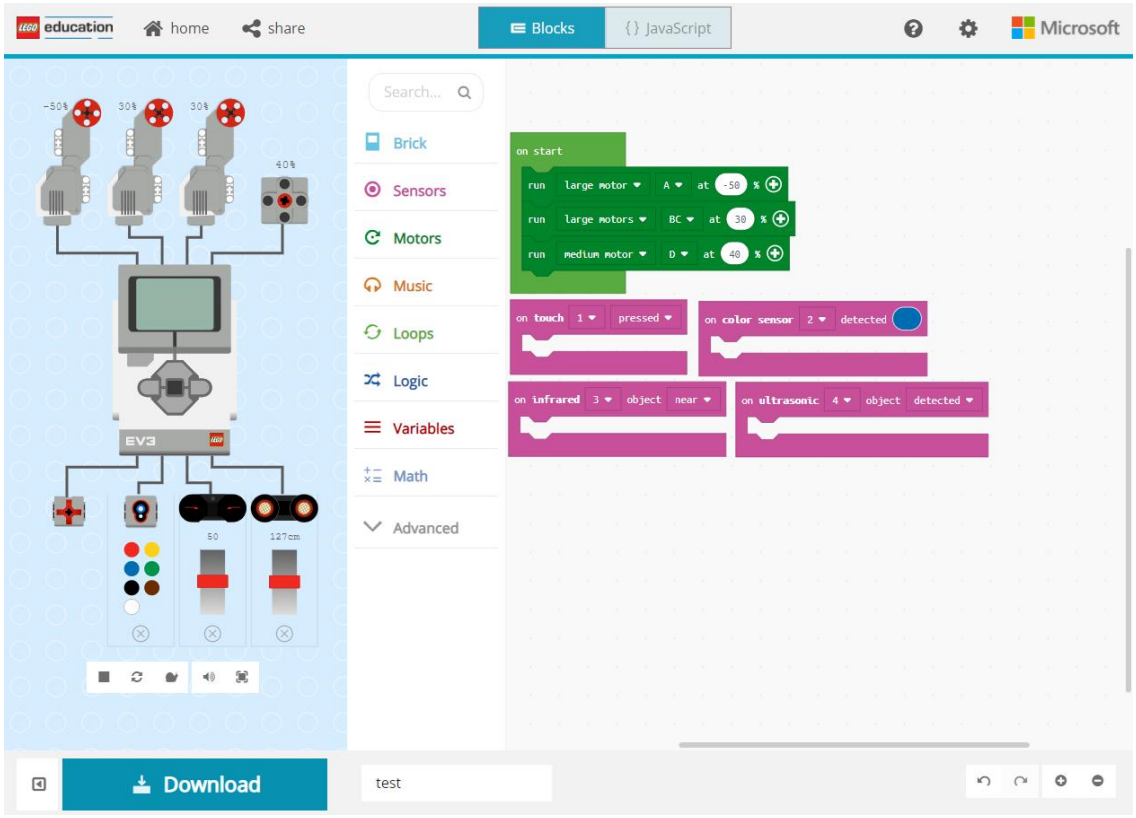

Figure 14: MakeCode for LEGO® MINDSTORMS® Education EV3 environment

This environment is therefore recommended for the development of a program for the operation of more advanced, individually assembled robots.

[14] https://makecode.mindstorms.com/

4.3 Summary of the possibilities of the environments

The following table summarizes the functions provided by each simulation environment.

	Scratch 3.0	**RoboMind**	**VEXcode VR**	**Virtual Robotics Toolkit**	**MakeCode for LEGO® MINDSTORMS® Education EV3**
Is it free to use?	☑	☑ v. 6.0 ☒ v. 7.0	☑	☒	☑
Can it be used in a browser?	☑	☒	☑	☒	☑
Supported operating system (offline version)	macOS 10.13 or later Windows 10 or later	v 6.0: Windows v7.0: Linux, macOS, Windows	-	Windows, macOS	Windows, macOS
Does it support block programming?	☑	☒	☑	☑	☑
Supported programming languages	☒	ROBO	Python	☒	JavaScript
What type of robot can we simulate?	General	General	VEX 123, GO, IQ and V5	LEGO MINDSTORMS NXT/EV3	LEGO MINDSTORMS EV3
Simulation environment (2D/3D)	2D	2D	3D	3D	2D
Suitable for simulating the movement of a robot vehicle	☑	☑	☑	☑	☒
Does it support creating custom tracks?	☑	☑	☒	☑	☒

It is visible that there are significant differences between the individual environments based on the examined aspects.

If you are looking for a browser-based environment which supports block-level programming, and it is not very important to see the robot's operation in 3D, then the Scratch environment may be the right choice for the simulations.

If you want to simulate the operation of a robot vehicle in 3D and are satisfied with the tracks offered by the manufacturer, we recommend the VEXcode VR environment. If the environment or the tasks need to be customised, or if your students go on to work with LEGO MINDSTORMS NXT/EV3 robots later, the Virtual Robotics Toolkit may be the right choice.

If you want to introduce students to the basics of robotics with an easy-to-use, imperative/procedural programming language, the RoboMind environment is the perfect choice. The easy modification of the tracks and the variety of basic instructions allow you to design a wide range of activities.

In the MakeCode for LEGO® MINDSTORMS® Education EV3 environment, the simulation is less spectacular, but it provides an opportunity to simulate the operation of a custom-built EV3 robot. Therefore, we recommend using this environment if it is also part of the task for students to build a robot on their own to solve a problem.

5. Case study

Let us examine a line following task in detail. In real life, the route is usually marked by a duct tape glued to the floor or a table, which allows for its simple modification. One solution may be to mount sensors on the front, left and right sides of the line tracking robot which detects either the intensity of the reflected light or its colour. The goal is to keep the centre of the robot on the line as it moves. If the left sensor detects the line, it means that the line is turning left, so the robot must also turn left by a certain degree. The same applies to turning right.

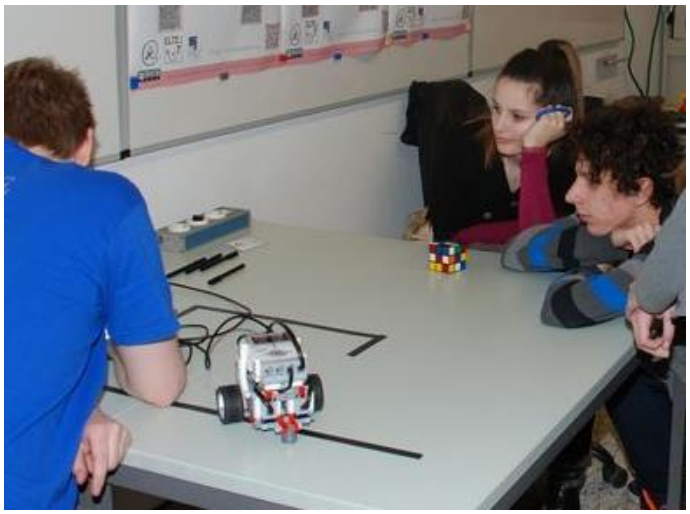

Figure 15: Robotics demonstration with a line-following robot in ELTE University's T@T Kuckó (T@T hub)[15]

Another solution may be a colour or light sensor mounted on the bottom centre of the robot. The simplest algorithm in this case is that when the vehicle moves away from the left edge of the strip,

[15] http://tet.inf.elte.hu/tetkucko/galeria/

it has to turn right until it finds its way back to the strip and then turn left afterwards. That is, the vehicle will perform a zigzag movement. In this implementation the strip must be relatively wide, as the vehicle can easily pass over a thin line.

Let us see an example how a line following task can be simulated in different simulation environments.

5.1 Line following simulation in RoboMind environment

Below is the track drawn in the RoboMind environment and a possible solution of the task. In the task, a white line on a square grid defines the path the robot should follow. The target where the robot should stop is marked at the end of the white line. The path should be such that the white line is not in the immediate vicinity of the robot, but falls into its path if it starts to move in the set direction. Let us see how this problem can be solved in the two environments.

The solution is that we advance 1 step until the field in front of us is white. We then go through the line with a loop which checks whether the target is in front of us (the execution of the loop stops), whether there is a white field in front of us (we move forward), or whether there is a white route on the left or right, as then we have to turn in the right direction.

Figure 16: The path drawn in the map editor

```
repeatWhile(not frontIsWhite){
     forward(1)
}
repeat
{
 if(frontIsBeacon){end}

elseif(frontIsWhite){forward(1)}
     else if(rightIsWhite){right}
     else if(leftIsWhite){left}
}
```

5.2 Line following simulation in Scratch environment

The solution is a little more complex in the Scratch environment, but that is an advantage in this case, as it shows in more detail how the task would be implemented in the real world. There are several types of sensors which can be used in the Scratch environment. As we mentioned, real robots often complete the task by colour perception, so we can try to do that in a similar way in Scratch.

A good way to do it is to draw a robot shape which has two antennas on the left and the right side, with ends marked with different colours. Those colours should also be different from the colour

of the path. If the colour of the sensor on the right antenna touches the colour of the route, the robot should turn right, and go on. The same applies for turning left.

In this case, a large black circle marks the target on the course. The left antenna of the robot is blue and the right is red.

The implementation is similar to the solution created in Robo, but in this case it must be ensured that if the left or right sensor detects the colour of the route, the robot should not turn immediately, but take so many steps forward that its middle be approximately above the centre of the line. That distance depends on the shape of the robot as well as the unit used on the route. The other difference compared to the previous solution is that the two colour sensors can be used to better illustrate the implementation with real robots, thus making better use of the knowledge gained for the construction of real robots.

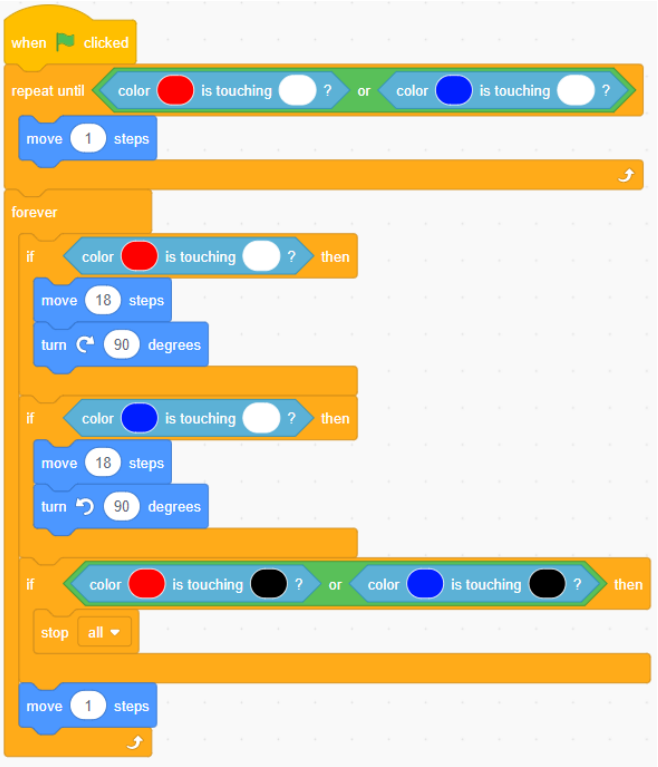

Figure 17: The Scratch code

Figure 18: The drawn robot

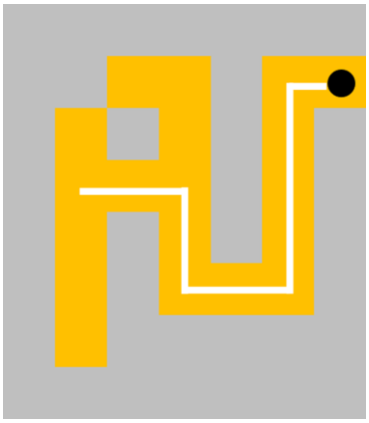

Figure 19: The path

5.3 Line following simulation in VEXcode VR environment

Unfortunately, we have not been able to solve the task in this environment. The playground in which the robot travels is pre-programmed and it cannot be modified (at least at the moment), so it is difficult to draw the track containing the line. The robot can draw with a pen, but it should get back to the starting point with the pen raised. However, even though we drew the line, the robot's sensor did not detect it. The colour recognition function only works for objects placed on the track. We hope that the possibilities of the VR environment will be expanded in the future, so that more advanced tasks can be solved in the environment.

5.4 Line following simulation in Virtual Robotics Toolkit environment

In this simulation environment we found a pre-designed track (Maze (EV3)) in which we could solve the line tracking problem.

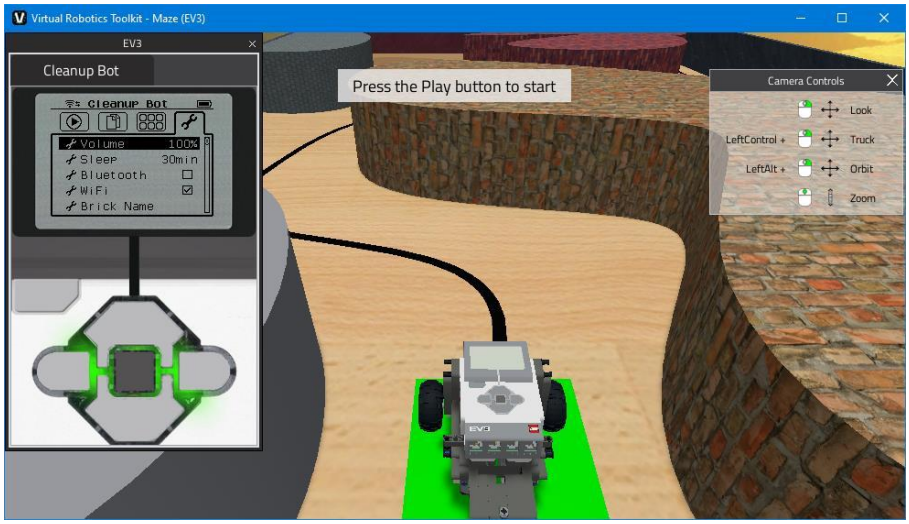

Figure 20: A built-in track for line tracking tasks

When the track is loaded, the robot's starting position is in a green area, while its colour sensor detects the black path. So, a program must be generated which follows the black path. You can see that the route does not turn at right angles as in previous cases, so the solution will be different.

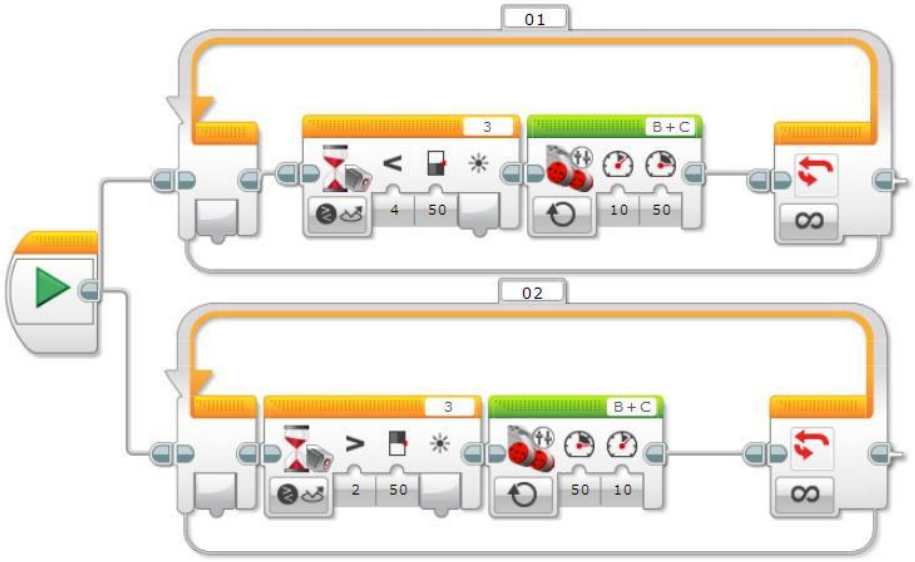

Figure 21: The completed program

In the above program, we examine whether the intensity of the reflected light is less than or greater than a threshold (in this case 50), and depending on this, we turn either left or right with the robot, so that it follows the edge of the line. Turning is done by changing the power of the left and right engine, respectively. When the power of the left engine is 10 and that of the right engine is 50, it means that the vehicle is turning left. The greater the difference in power between the two engines, the smaller the curve.

6. Conclusion

This article outlines the possibilities and benefits of simulating robot generations in public education, highlighting the fact that these activities can well complement (but not completely replace) the work with real robots.

We have demonstrated that students can gain important experience in controlling and programming robots, even by unplugged activities, which they can later use during practical activities.

Several computer environments are available to simulate the activities of robots. Some of these simulate the operation of specific robot types, while in others we have the ability to simulate general (or fictitious) robots. Important aspect when choosing an environment may be the programming languages in supports, the software's ease of use, the customizability of the environment and the tracks, the option to simulate the movement of robotic vehicles, and whether it uses a 2D or a 3D visual representation. The main features of the simulation environments are also summarized in a table, which helps in selecting the appropriate environment.

In the case study, we showed how a line tracking task can be solved in different simulation environments. During the implementation of the specific task, we found that there are very different options in each environment for setting and customizing the properties of the environment, which greatly affects the extent to which we can set our own, unique tasks for students.

Bibliography

1. Pluhár Zsuzsa: Robotikáról tanároknak. [About robotics for teachers.] 2019. [on-line] ⟨https://bit.ly/3aAW697⟩
2. Dr. Kodácsy János, Dr. Pintér József (2011): Szerszámgépek és gyártórendszerek. [Power tools and production systems] [on-line] ⟨https://bit.ly/2zfRRml⟩
3. Lénárd András (ed.) Az algoritmikus gondolkodás fejlesztése padlórobotok segítségével. [Development of algorithmic thinking with the help of floor robots.] 2018. Budapest, Hungary: Stiefel Kft.
4. Gaál Bence: Comparative analysis of sets used in robotics education. In: Proceeding of Didmattech 2020 Conference. [on-line] ⟨http://didmattech.inf.elte.hu/proceedings-2020/⟩
5. John Burfoot: EV3 Sensors – LEGO Engineering, 2018. [on-line] ⟨http://www.legoengineering.com/ev3-sensors/⟩
6. Michel, Olivier: WebotsTM: Professional Mobile Robot Simulation. International Journal of Advanced Robotic Systems. 1. DOI: 10.5772/5618. 2004. [on-line] ⟨https://bit.ly/2xRP96i⟩
7. Pintér Gergely (ed.) Digitális kultúra 5. tankönyv [Digital culture 5. school book] 2020-Budapest, Hungary. Oktatási Hivatal. ⟨https://www.tankonyvkatalogus.hu/pdf/OH-DIG05TA__teljes.pdf⟩
8. Richard Nelson, Jason Clutterbuck, Sebastian Höhna, Stefan Marks and Wilson Siringoringo: Harold the Robot. [on-line] ⟨https://classic.csunplugged.org/harold-the-robot-2⟩

9. Miller, Blanca & Kirn, Adam & Anderson, Mercedes & Major, Justin & Feil-Seifer, David & Jurkiewicz, Melissa. (2018). Unplugged Robotics to Increase K-12 Students' Engineering Interest and Attitudes. 1-5. DOI: 10.1109/FIE.2018.8658959.

10. CS Fundamentals Unplugged | Code.org [on-line] ⟨https://code.org/curriculum/unplugged⟩

11. Péter Fehér, Dóra Orsolya Aknai: Wandering Robots in Hungarian Primary Schools: a Case Study. ECEr 2019 Conference. [on-line] ⟨https://bit.ly/2XXOZEV ⟩

12. Abonyi-Tóth Andor, Pluhár Zsuzsa: Wandering microbits in the public education of Hungary- LECTURE NOTES IN COMPUTER SCIENCE 11913 pp. 189-199., 11 p. 2019.

13. Jonassen, D. H.: Toward a design theory of problem solving. Education Technology Research and Development, 48 (4), 63--85. (2000). DOI: 10.1007/BF02300500.

14. Bernát Péter: Robotika az általános iskolában és a RoboMind programozási környezet Paper:3 [Robotics in primary school and the RoboMind programming environment Paper:3] In: Péter Szlávi; Zsakó, László (ed.) INFODIDACT 2015, Budapest, Hungary: Webdidaktika Alapítvány, (2015)

Authors

ABONYI-TÓTH Andor

Eötvös Loránd University, Faculty of Informatics, Department of Media and Educational Informatics, Hungary,
e-mail: abonyita@inf.elte.hu

About this document

Published in:

CENTRAL-EUROPEAN JOURNAL OF NEW TECHNOLOGIES IN RESEARCH, EDUCATION AND PRACTICE

Volume 3, Number 2. 2021

ISSN: 2676-9425 (online)

DOI:

10.36427/CEJNTREP.3.2.1469

License