electronics

MDPI

Article

PyBoKids: An Innovative Python-Based Educational Framework Using Real and Simulated Arduino Robots

Julio Vega *,†,‡ **and José M. Cañas** ‡

Department of Telematic Systems and Computation, Rey Juan Carlos University, 28934 Madrid, Spain

* Correspondence: julio.vega@urjc.es; Tel.: +34-914-888-755

† Current address: Department of Telematic Systems and Computation, Rey Juan Carlos University, Camino del Molino S/N, Fuenlabrada, 28934 Madrid, Spain.

‡ These authors contributed equally to this work.

Received: 28 July 2019; Accepted: 13 August 2019; Published: 14 August 2019

Abstract: In western countries, robotics is becoming increasingly common in primary and secondary education, both as a specific discipline and a tool to make science, technology, engineering, and mathematics (STEM) subjects more appealing to children. The impact of robotics on society is also growing yearly, with new robotics applications in such things as autonomous cars, vacuum cleaners, and the area of logistics. In addition, the labor market is constantly demanding more professionals with robotics skills. This paper presents the `PyBoKids` framework for teaching robotics in secondary school, where its aim is to improve pre-university robotics education. It is based on the Python programming language and robots using an Arduino microprocessor. It includes a software infrastructure and a collection of practical exercises directed at pre-university students. The software infrastructure provides support for real and simulated robots. Moreover, we describe a pilot teaching project based on this framework, which was used by more than 2000 real students over the last two years.

Keywords: science teaching; robotics framework; Python; Arduino; secondary education

1. Introduction

The field of robotics is undeniably growing in importance, as it has the power to motivate students and allows us to bring technology closer to boys and girls [1] by using robotics as a tool to present the basic concepts of science [2], technology, engineering, and mathematics (STEM) [3,4]. Thus, students learn, almost through play, notions that are difficult and complex to explain or assimilate through the classic masterclass [5,6].

The implantation of robotics in education is a fact. In the last five months, six states in the U.S. (Iowa, Nevada, Wisconsin, Washington, Idaho, and Utah) have announced plans and investments with this aim. Likewise, four countries—Canada, Ireland, New Zealand, and Romania—have recently announced similar plans, with a total investment of 300 million dollars. Japan, in its `New Robot Strategy Report` [7], highlighted that investing in robotics is fundamental for the growth of the country.

In this educational field, the teaching of robotics itself converges with other disciplines (e.g., programming) using robotics as a teaching tool [8–10].

Robotics championships for teenagers, which encourage interest in this area of technology, are another example of the increasing importance of robotics in education. At an international level, numerous championships are organized, which bring together students from all over the

world to learn, share experiences, and enjoy the development of robotic prototypes. The RoboCup Junior (http://rcj.robocup.org) [11–13] is especially worth mentioning, with tests such as rescuing or robotic soccer. There is also the First Lego League (FLL) and the *VEX robotics competitions* (https://www.vexrobotics.com/vexedr/competition). In Finland, the quintessential championship which attracts students from all over Europe [14] and has agreements with the centers of South Africa [15] is the *SciFest* (http://www.scifest.fi).

Furthermore, in the academic community, a group of congresses and conferences have emerged which emphasize the role of robotics in education, including the *Conference on Robotics in Education* (RIE), and the *Workshop on Teaching Robotics with Robot Operating System* (TRROS) within the *European robotics forum* (http://www.eu-robotics.net/robotics_forum). Special editions on education in robotics have also appeared in several scientific journals.

One of the motivations of this article was a European Erasmus+ project in which the authors have participated. Finland is a country of reference in education and, year after year, it is ranked first in the *Programme For International Student Assessment* (PISA) *Educational Annual Report* (http://www.compareyourcountry.org/pisa/country/fin). This project included interviews with numerous experts in the training of Finnish teachers, and a one-month visit to a leading center in the use of robotics in the classroom—specifically, the *Joensuu Science Society* (http://www.tiedeseura.fi). The aim was to investigate the differences between the Finnish and Spanish education system [16], and more specifically, what use was made of robotics in both countries.

Another motivation was our experience in teaching robotics with Scratch language and Lego Mindstorms platforms. This balanced combination has proven to be very effective in the early years of secondary education as an initiation into the world of programming in general, and robotics in particular. Initial learning is almost immediate, and generates great enthusiasm for implementing, designing, and programming at full speed. However, in the later years of secondary education, since students have already acquired a certain fluency in programming in previous years, this enthusiasm for doing new things diminishes considerably, and the Scratch language falls short of their needs, meaning that they become bored to a certain extent [17].

In this context, this article presents an educational framework called `PyBoKids` for pre-university students, which has been tested under a pilot project over the last two years. The framework aims to provide a complete and easy-to-use middleware for programming robots. The core elements are Arduino, as the hardware platform, or a Gazebo-simulated mBot model, developed from scratch and integrated on the `PyBoKids` framework, as well as the Python language. It was successfully implemented over the last two years within the Franciscanas de Montpellier Foundation (http://www.colegiofranciscanas.com/node/6), which has six schools distributed across Spain, as well as in two public schools.

The teaching was carried out following a constructivist methodology, inspired by concepts used in the successful Finnish educational system. The academic program followed was also inspired by the authors' previous experiences with the use of LEGO Mindstorms to teach robotics in secondary education through a constructivist methodology [18]. The proposed teaching environment has sufficient content for a full academic year, and was designed to overcome the aforementioned limitations of Scratch.

2. Teaching Robotics to Pre-University Students

Many teaching frameworks are used to teach robotics to children, ranging from those focused on primary education, to more powerful ones oriented to secondary education and high school. They are usually composed of a concrete robotic platform, that is to say, a robot which has been programmed in a certain language using software tools. Students are then required to develop different exercises, challenges, or projects (*practice activities*). These teach the basic operation of sensors, actuators, and the rudiments of programming. These frameworks are used as a tool within a specific methodology for teaching robotics classes.

We can identify four elements that characterize the most widely used frameworks and the numerous ways of teaching robotics to adolescents: hardware platform, software language and infrastructure, concrete practices, and methodology. Several illustrative examples will be described in this section, and the teaching proposal will be presented in the next.

2.1. Hardware Platforms

The robots used in pre-university education usually incorporate a limited processor, sensors, and simple actuators. Frequent use is made of infrared sensors (photodetectors), ultrasound, contact, sound, light sensors, and the like. Actuators typically include LEDs, screens, small loudspeakers, and fundamentally, motors. These motors can be of several types: DC motors, stepper motors, or servomotors. They are usually connected to the robot's processor using direct cables or simple connectors (such as RJ25).

Some platforms have a closed mechanical design, while others allow some flexibility using pre-built blocks that can be connected in multiple ways, or parts with sensors or actuators, where the students can decide which element to mount in each case and in what position. Other platforms have no *a priori* mechanical design. They are open-ended, and provide students with the learning materials.

Some widely used platforms are the LEGO platforms in its different models: MindStorms RCX, NXT, EV3, and WeDo [6,12].

Another widely used option, both in secondary education and high school, are plates with Arduino [12,19–22] or Raspberry Pi [23] processors to which low-cost sensors and loose servos are connected. This allows students to interact with a real robot, sensors, and real actuators at an affordable cost. It also offers many didactic possibilities, such as those described in [24–28].

Another prominent platform is that of the Thymio robot [8,29], open hardware, and the Thymio-II (IniRobot [30]). In addition, VEX robots and robotic kits, such as IQ and CORTEX models, are used with certain frequency in education [31].

Here, it is worth mentioning the Spanish manufacturer of robots, BQ Zowi, based on Arduino, as well as PrintBot evolution, based on the ATmega328P microcontroller. Also worth noting are Meet Edison's robots (https://meetedison.com)), created by an Australian company—these are small robots that allow younger children to start to manage and program a robot. Finally, we have the Makey-Makey (https://makeymakey.com) plates, which allow any electrical current—however weak—to be transformed into a signal that is interpreted and used to simulate, for example, a joystick or the keys of a piano. It is usually used in a simulated physics environment known as Flabby Physics (http://flabbyphysics.com).

Simulated robots are also used in pre-university education. For example, the TRIK-Studio environment includes a simple 2D simulator for the TRIK robot [22,32]. Another important example is the 3D simulator used in Robot Virtual Worlds (RVW) (http://www.robotvirtualworlds.com/) [33], which simulates robots from different manufacturers (VEX, LEGO, and TETRIX).

2.2. Languages and Software Environments

Typically, each robot has a software environment that enables programming in a certain language. The environment usually includes code editors, utilities to download in real robots, and, on some occasions, even simulators. Simple languages are used to facilitate programming by children. They include instructions for ordering commands to actuators to read sensor measurements, loops, and conditional and sequencing instructions.

The graphic languages of LEGO which are specific to their robots, such as the old RCX Code, RoboLab (built within LabVIEW), NXT-G, and the latest EV3 software are a useful option. All contain blocks of action, sensors, flow control, operations with data, and suchlike.

Another visual alternative is the Scratch language (https://scratch.mit.edu/) [28,34,35], or variants such as Blockly (e.g., with the robot RoBOBO [36]), Bitbloq (by BQ-Zowi), or VPL (by Thymio). All these also have graphic blocks, which typically connect in sequence in a graphic editor.

The complexity of languages such as C++, which are used successfully at university level, means they are not recommended for adolescents. However, similar languages to C without object orientation are used—an example being NXC for LEGO robots [12].

In this line is the ROBOTC (http://www.robotc.net) environment, which uses the C language and a graphical variant of it (ROBOTC-graphical) to program robots from different manufacturers (VEX IQ, VEX CORTEX, LEGO EV3, LEGO NXT, and Arduino) and simulated robots in RVW. In particular, it is used in the Carnegie Mellon robotics Academy [33] with different exercises and competitions.

Sentance [37] analyzed the use of programming languages in UK schools through a survey of 1159 technology teachers. The most widely used language was Scratch (95% in primary and secondary), followed by Python (18% in primary, 84% in secondary).

2.3. Exercises

Robotics teaching is notably practical. By its very nature, it lends itself to learning by doing. Thus, in addition to the theoretical content, emphasis is commonly placed on certain projects or exercises that students have to tackle and solve using the appropriate robot and its software environment. Performing these projects means that students encounter specific problems—and through solving such problems, they acquire a range of robotic skills.

In exploring the existing literature, we found a set of exercises that are frequently used in different teaching frameworks and academic proposals, often based on the cross-curricular approach. One of the classic projects was Behavior Follows-Lines [6,12,22,32], in which the robot has infrared (IR) sensors pointing to the ground, which are white but have a thick black line. Another is the avoidance of obstacles [12,32], where the robot has an ultrasound sensor that allows it to detect objects that interfere with the robot's movement. The student's program must then order the motors to stop and turn until they find a free space to advance once more.

Several exercises are aligned with tests within championships, such as a game of sumo between two robots, and several are related to robotic football [22]. These exercises allow a competitive playful approach that increases student motivation.

Other interesting examples are those of the robot that follows a wall [32] or that escapes from a labyrinth [22].

2.4. Methodologies

Teaching methodologies underpin the cognitive processes activated in students when they learn, and they also represent different ways of motivating students. They all seek to reach out to the students, capture their attention, and/or awaken interest in the subject [38,39]. Several stand out, such as: (a) the traditional approach, (b) constructivism, (c) project-oriented learning, (d) cooperative learning, (e) problem solving, and other derived methodologies. Rather than being mutually exclusive, they are complementary. One methodology typically makes use of one or another, in accordance with the objectives to be achieved in class. The nature of teaching robotics means the foremost approach is practical.

The traditional teaching approach is based on masterclasses, where the teaching–learning process is radically delineated: the teacher teaches, and the student receives information [40]. They usually include teaching material for theory and practical exercises, with instructions that students follow.

Constructivism considers that by providing students with the necessary tools, they can build their own procedures to solve a problem, where their ideas can be modified and learning can be continued.

Project-oriented learning also considers that students can be more responsible for their own learning. In this case, the approach focuses on applying the skills and knowledge they acquire in masterclass theory sessions to real projects [41]. A frequent project is participation in robotic competitions, where the students' robots (at group or school level) have to compete with those of other students in passing tests. This participation enhances motivation.

Cooperative learning [42] focuses on assessing the educational potential derived from the interpersonal relationships of any group, so that the work is carried out in common, thus balancing and taking advantage of the skills of the group's components.

Problem-based learning [43] is primarily based on the constructivist theory, following its fundamental principles. The importance of this methodology lies in its influence when reorganizing the information stored in the student's cognitive structure. Learning takes place within this process of modification.

Under the European Erasmus+ project, the most commonly used methodologies in the field in Finland were studied [44]. The constructivist approach and problem-based learning are mostly used in the Finnish pre-university education system. There has been a great advance in the results of the PISA report [45] of a paradigm shift in the teaching–learning process [46], whereby the teacher not only transmits knowledge, but also guides students in their self-learning. Under this framework, students can give free rein to their creativity, thanks to robotics [47].

There is a clear differentiation between formal sessions, in which the teachers deal with content that they consider necessary for the student, and non-formal ones, in which the students themselves perform the learning process by their own means [48].

In addition to specific robotics teaching, other subjects in Finland are also taught through robotics. In the 2017–18 school year, included in the secondary curriculum was the requirement that all students must be able to program a robot in a simple way, using LEGO MindStorms. Thus, for example, a series of specific mathematics skills related to this are developed from first to ninth grade.

3. Teaching Framework, PyBoKids

The developed teaching framework (http://jderobot.org/PyBoKids) includes a hardware platform (Section 3.1), a software infrastructure (Section 3.2), as well as an educational program (Section 3.3) for a full academic year, and a suggested specific pedagogical methodology (Section 4). The central pillars of the design are: robots with free hardware processors (Arduino), the Python programming language, and a collection of practice activities of progressive complexity.

3.1. Hardware Platform

An Arduino-based robot, the MakeBlock `mBot` (https://makeblock.es) (Figure 1 left), was chosen as the main reference hardware platform. The `mBot`, with its Arduino Uno processor, can be connected to the sensors and motors commonly used in educational robotics. It has different models, depending on its connectivity: USB, 2.4 G, and Bluetooth. It can be connected through the USB cable to the computer to download programs. It is affordable, mechanically compact, and extensible. Kits of mechanical parts, such as sensors or actuators, are economically priced.

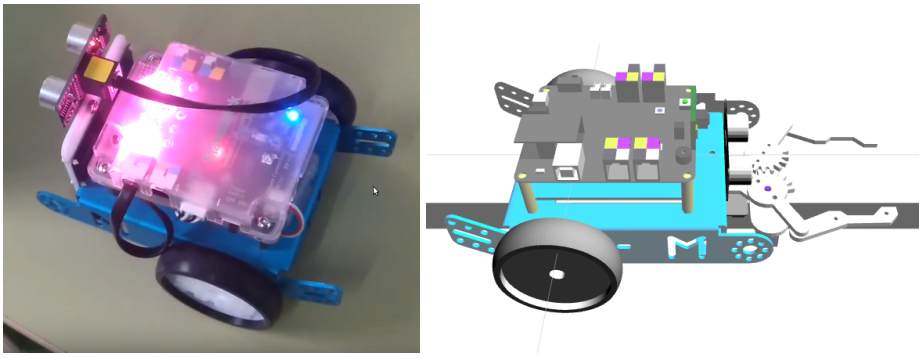

Figure 1. Robot mBot, real and simulated, in Gazebo.

In addition, it has good support for programming in the mBlock graphic language, which is based on Scratch 2.0, and in the Arduino language, which has an extensive community of users all over the world and proven software tools.

In addition to the real robot, the counterpart for the Gazebo simulator (Figure 1 right) was also programmed in `PyBoKids`. Gazebo is a free software 3D simulator that incorporates several physical engines for realistic simulations, and is a de facto standard in the robotics research community, with more powerful robot models [49]. Specifically, the graphic, mechanical model, as well as a C++ plugin that runs within the simulator and is able to communicate with external programs was developed for `mBot`. This plugin allows the students' programs to collect readings from the virtual IR and ultrasound sensors, as well as to send movement commands to the emulated motors. That is, it allows the behavior of the robot in the simulated world to be controlled.

The initial motive for giving support to the simulated robot was that students and educational centers, without the physical robot, could nevertheless practice and learn or teach robotics with `PyBoKids`. In addition, this support mitigates the common problems of economic costs and hardware maintenance that arise when introducing robotic artifacts in a classroom.

Likewise, a homemade robot was also built by connecting sensors and actuators to a protoboard mounted on an Arduino and assembling them in a mechanical chassis. This shows the versatility of the teaching framework, which is valid for different platforms provided they incorporate the Arduino microprocessor.

3.2. Language and Software Infrastructure

Arduino is normally programmed by Arduino IDE, or by Scratch (or some of its variants, such as mBlock of `mBot`). In `PyBoKids`, Python was chosen as the programming language because of its simplicity, its expressive power, and because it is widely used in higher levels of education and programming. It is a text language that is interpreted and object-oriented. This language is easier to learn than Arduino (very similar to C/C++) and, simultaneously, has great power. It is also used in university education, together with more powerful libraries.

Two questions arise with this approach: (a) the difficulty of learning to program in a programming language that is not visual, and (b) the high economic and logistical cost of acquiring a considerable amount of robotic equipment for a high-school class, which usually has around thirty students.

As the Python language is not supported by the manufacturer of the `mBot`, an entire infrastructure was created in `PyBoKids`. The Arduino microprocessor is too limited to run an on-board Python interpreter. Therefore, a module for the real robot, called `realMBot` was implemented and programmed as a Python library that runs on the computer and communicates continuously (via USB or via Bluetooth 2.4G) with the physical robot `mBot` using the Firmata protocol (https://github.com/firmata/protocol), in which an intermediary program is executed on the native Arduino firmware (Figure 2 left). The chosen design is shown in Figure 3. It prioritizes simplicity of use, which required making the underlying infrastructure quite sophisticated.

A specific library was developed to provide the programming interface (API), `PyBoKids.py` (https://gitlab.etsit.urjc.es/jmvega/PyBoKids/blob/master/PyBoKids.py). This simple and natural interface includes methods to read the measurements from the sensors, as well as methods to give commands to the actuators of the `mBot` (Table 1).

The use is as simple as programming an application in Python (Figure 4) to use these methods to control the robot using any file editor (such as *Gedit*). In this way, students concentrate on the algorithm they are developing, avoiding low-level details such as ports or connectivity with the robot, which are stored in the library configuration file. Then, the Python code is executed as usual, commonly run from a command line (*Terminal*) with the following command: `python myPyBoKids.py`. The error messages are displayed below.

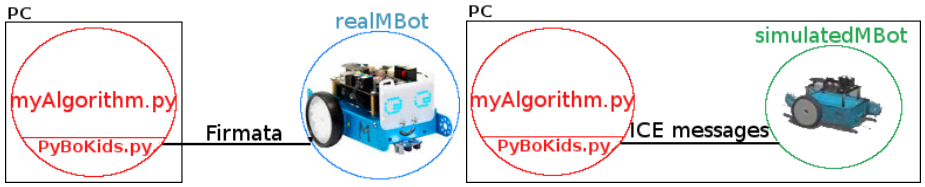

Figure 2. Connection of the PyBoKids.py library with the real mBot and the simulated mBot.

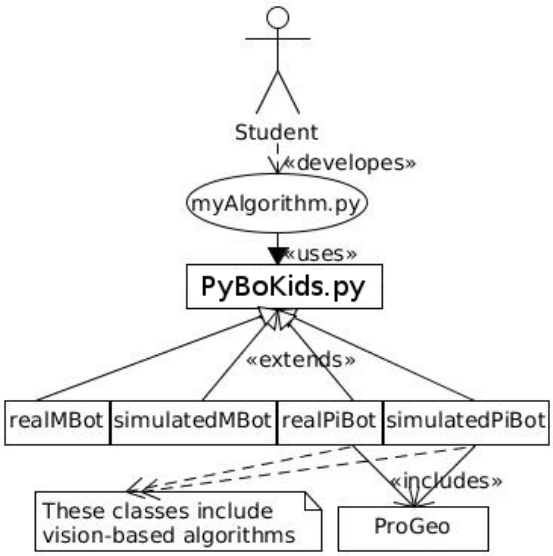

Figure 3. The student uses the PyBoKids.py library in his program.

```
from PyBoKids import MBotReal, MBotGazebo

if __name__ == "__main__":
  # Loading config file:
  ic = EasyIce.initialize(sys.argv)
  # Loading real or simulated robot:
  myPyBoKids = ic.getProperty("Robot")

  # TODO: type your code here! Example:
  myPyBoKids.forward(200)
  time.sleep(2)
  myPyBoKids.stop()
```

Figure 4. PyBoKids Python code example program.

The most important API methods for `PyBoKids.py` are detailed in Table 1, where `V` is linear velocity and `W` is angular velocity—the functionality is quite clear, following the nomenclature of the different functions. These allow access to each of the usual sensors, such as the US sensor, IR sensors, and light sensors. Each of the motors can be governed individually (raw methods). Movement orders for the whole robot (cooked methods), which are simpler to use, can also be sent. In this case, the library translates the desired combined movement into the orders for each of the two motors that carry it out.

The students program their exercises in Python by writing the file `myAlgorithm.py`, for example, with a text editor. From this program, all the methods provided by this library may be used. `PyBoKids.py` includes two different modules that perform exactly the same API functions. One module implements the interface for the management of the real robot, and another for the simulated robot in Gazebo. The final robot in each case is selected by specifying it on the library configuration file. As the

programming interface is the same in both cases, the application is identical and works interchangeably on both the physical and simulated platforms.

The orders issued from the student's application arrive at the library, which transmits them—following the Firmata protocol—to the intermediary program on board the robot, written in Arduino language, which executes them on the motors. The read requests of sensors from the application arrive at the library, which takes the last readings received from the intermediary program on board and delivers them to the application.

Table 1. PyBoKids.py programming interface.

Acting	Sensory
move (V, W)	readLightLevel
forward (V)	readUltrasound
backward (V)	readSoundLevel
left (W)	readPotentiometer
right (W)	readRemoteIR
stop	
moveServo (θ)	
ledOn	
ledOff	
writeText (T)	
playTone (Fs)	

The second module which was developed, the `simulatedMBot`, allows access to the simulated robot inside *Gazebo* (Figures 2 right and 3). In this case, the methods of the API `PyBoKids.py` are translated to send messages to the Gazebo-developed plugin in C ++, which controls the sensors and actuators emulated in Gazebo. These messages are implemented with the ICE communications middleware (https://zeroc.com), the library of which provides services for networked applications. The simulator natively works on Linux computers, and on MS-Windows or MacOS computers using *docker* containers.

3.3. Academic Program

A plan of activities for the subject of *programming, robotics, and technology* was designed and implemented in different years of secondary education and for a course of extracurricular activities. Since the students in these year-groups have no notion of computer programming, they have to start from a basic level until they are ultimately able to develop a complex project consisting of a standard robotics task.

The academic program was divided into four phases of progressive learning:

1. 14 sessions: Basic notions of programming using the visual language, Scratch: loops, conditions, variables, etc.
2. 10 sessions: Introduction to Python language, with basic practice activities using loops, conditions, variables, functions, etc.
3. 20 sessions: robotics programming practice with sensors and actuators individually.
4. 10 sessions: Programming behaviors in a robot. Final project encompassing all the above.

Each session lasts one hour. Each phase is described below, indicating the content and practical tasks that students develop as the academic program progresses.

3.3.1. Basics of Programming

In the first part of the course, basic notions of computer programming are acquired. In this way, students understand the way a computer works internally, and thus the reason for the use of variables or functions. Furthermore, concepts such as loops or conditionals are totally new to them. Hence,

this first contact with the subject is very important. Depending on the students, this usually lasts about four sessions.

After that, another five sessions are dedicated to implementing the basic notions learned in a language suitable for young students—an intuitive language, such as the visual language, Scratch. Here, some aspects of syntax are presented in broad strokes, as well as novel concepts that continue to appear, such as that of a counter, and the use of sets or vectors. They also understand why a variable must be defined, as well as other minor topics. This is an important phase where students internalize the structure, organization, and restrictions of a programming language.

The practical exercises that the students carry out in Scratch to achieve the aforementioned objectives, which usually take about ten sessions, are the following:

1. Introduction to Scratch. Designing an interactive character so that, when clicked on, there will be visual effects, a movement, a sound, and a change of appearance.
2. Use of variables. Developing a game in which the previously designed character picks up objects distributed around the scene.
3. Dynamic objects (loops). Adding to the game objects that move constantly in a cyclic movement. If they touch the character, it will lose a life; if it reaches 0, the game ends.
4. Final project. Continuing the game with different screens (phases) through which the character progresses and which will be accessed through passages, pipes, and secret access.

3.3.2. Introduction to the Python Language

In this second phase, the basic notions of the Python language are specified. The focus is on understanding keywords, as well as purely syntactic issues that are typical of Python. Here, the students are already prepared to perform standard, initial programming exercises, such as printing the sum of the first 100 numbers and suchlike. Performing, correcting, and explaining these exercises take ten sessions.

3.3.3. Robotic Practice Activities: Handling of Sensors and Actuators

In this third phase, the students carry out ten activities directly related to robotics. They begin assembling different components on an Arduino board (in the case of the homemade prototype by pieces) and review some basic concepts of electronics so that they will have no problems connecting the different devices. Step by step, they begin by installing simple components on a protoboard mounted on Arduino, such as a buzzer, or LEDs, and their corresponding software developments, to move on to more complex ones such as light, infrared, or ultrasound sensors. Once they have mastered the electronics of these components, they tackle the use of the sensors and actuators already pre-installed in a `mBot`. Finally, they begin to use the camera as a sensor and the treatment of images that it provides. All this covers about twenty sessions, as follows:

1. Use of push-button with LEDs (Figure 5 left) and microphone (Figure 5 right).
2. Sensor readings: light, infrared, and ultrasonic.
3. Control of servos.
4. Using LED array (Figure 6).
5. Gripper control on pan-tilt.

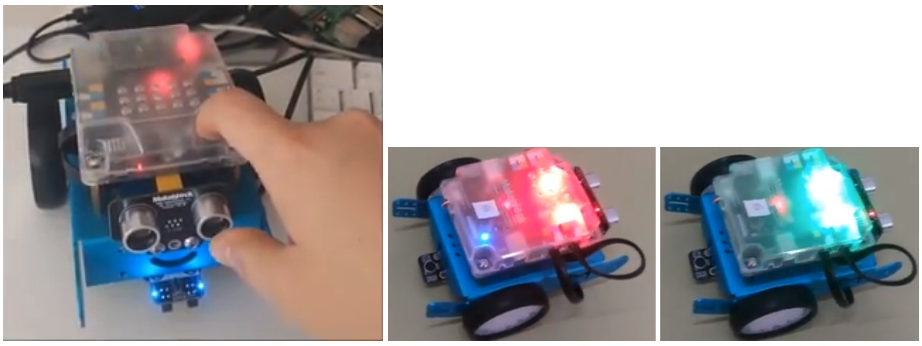

Figure 5. Practice tasks with mBot to operate the push button and LED, and the microphone with LED to recognize sounds.

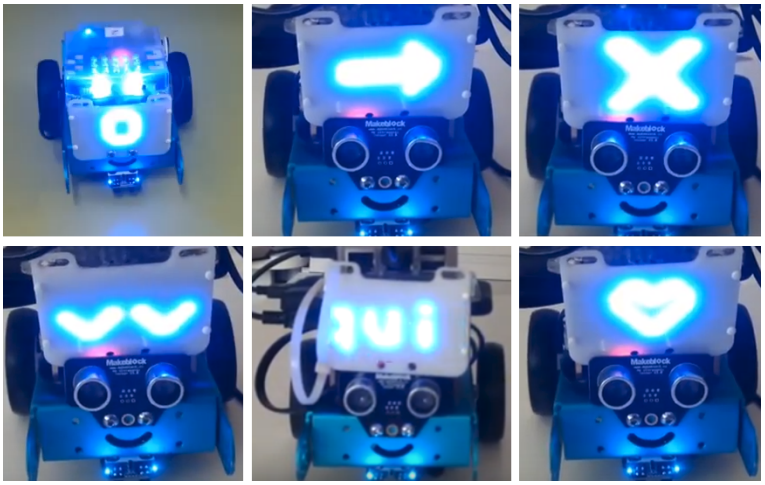

Figure 6. Practice task with the LED matrix actuator.

3.3.4. Robotic Practice Activities: Autonomous Behaviors

The last step of this learning pyramid consists of a complete robotics project where students combine all the things they have previously learnt. Projects developed include:

1. Navigation following a line (Figure 7).
2. Navigation avoiding obstacles by means of ultrasounds (Figure 8).
3. Navigation following the light projected by the flash of a mobile phone.
4. Sumo fight between two mBots.
5. Navigation by clapping.
6. Stone, paper or scissors game, using the LED array.

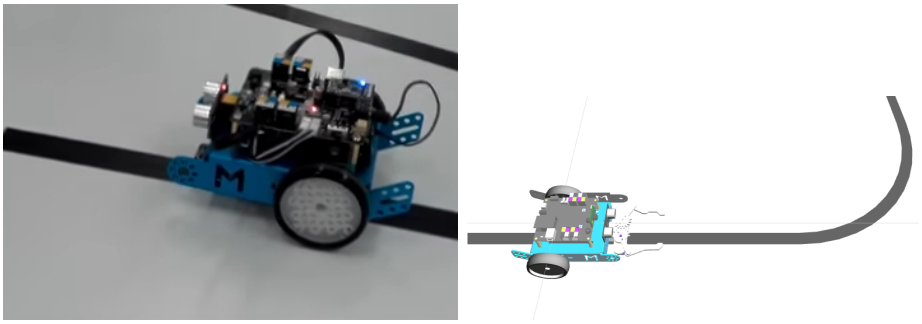

Figure 7. Practice line-tracking task in a real and simulated mBot robot.

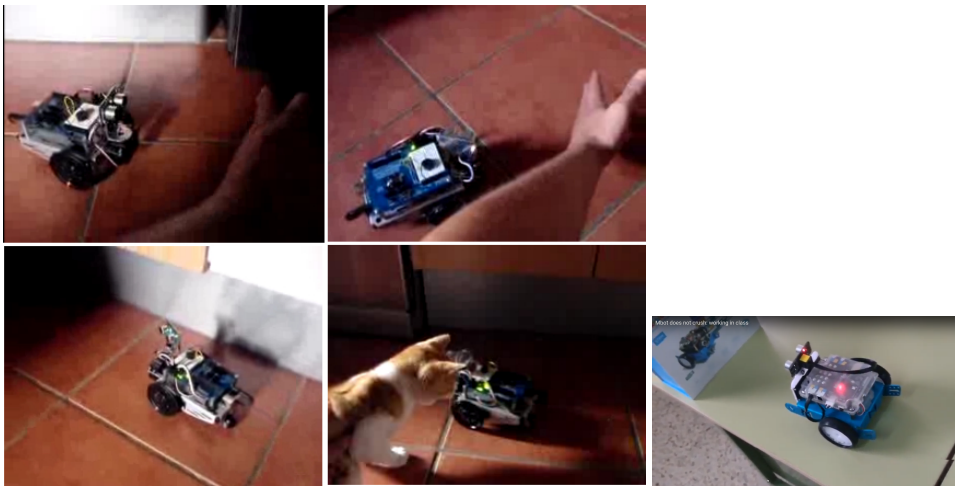

Figure 8. Navigation practice avoiding obstacles through US in Arduino and in mBot.

A further ten sessions are required to finish this final project.

4. Constructivist Methodology in Robotics

Our own experience teaching robotics to pre-university students for several years, the analysis of teaching methodologies carried out in Section 2.4, and especially the on-site study of how to implement this teaching in Finland within the European Erasmus+ project mentioned in the introduction have served to refine the proposed teaching methodology.

We recommend using the `PyBoKids` teaching framework within the constructivist methodology described in this section. This is based on the premise that knowledge is within the participants, and that these participants—who could be called *thinking subjects*—have no alternative but to build their own procedures or learning paths based on what their own experience dictates.

The pioneer of this approach was Ernst von Glasersfeld [50,51]. This theory posits that students learn more when they are given the opportunity to explore and create knowledge that is of personal interest to them [52]. This fits perfectly with the teaching of robotics, since students can experiment with a physical device, make mistakes, and learn from them while working, thus building their own knowledge [18].

In the sessions described in the academic program, there is no differentiation between theoretical and practical sessions. At the beginning of each class, the content learned in the previous session is revisited, concepts that will be seen in the current session are mentioned, and the objectives to be reached by the end of the session are explained, with all of this being contextualized in a challenge students have to pursue. The above takes between five and ten minutes of class time. Subsequently, students are given full freedom to access all the available tools (computers, robots, and components) so that they can decide how to distribute the time and what to do first. They can be corrected or advised if they stray from the path that will lead them to reach the proposed objectives.

In this way, the teacher becomes a guide, rather than a strict setter of norms, guidelines, and knowledge to be assimilated. Moreover, by averaging the philosophy of cooperative learning, students always work on robotics in groups, because in this way they can help each other and are not frustrated by failures—some group members will always be sure of what to do.

Fifteen minutes before finishing the class, they are notified of the time left to finish the session. They then know they have five more minutes to finish, or to save the work they are doing, since the last ten minutes are always reserved for reflection on how each group has learned and what each student is learning individually. This final moment is suitable for clarifying issues and introducing (if necessary) certain detailed and theoretical concepts. In this way, students acquire useful notions which are then fixed in their memory, since they have used them to solve a specific difficulty they have

actually faced. Thus, in addition to the teacher ensuring they have a solid base of knowledge, the teacher becomes a *learning supervisor*.

Following this line of constructivist learning, regular assessments of knowledge lack meaning. In a certain way, evaluation is reversed, since the students assess themselves daily by giving a grade as a group, on how they consider they have been able to tackle the problem and solve it (in their case), as well as individually evaluating their contribution to the group. The teacher combines this student self-evaluation with their own assessment, based on the observation of the class both at a group and individual level, taking into account each student's potential and the effort made.

At the end of each topic a session is dedicated to reviewing what has been presented, what they have learned, what difficulties have been encountered, and how they have been resolved. Likewise, the teacher comments on both the group and individual work and, consequently, each student's grade for that unit. Thus, students are always aware of their strengths and weaknesses so they can try to balance them in the following units.

5. Deployment and Results

In the 2016/2017 academic year, the proposed academic program was implemented with the `PyBoKids` teaching framework at the Franciscanas de Montpellier Foundation, which has six schools spread across Spain. In addition, it was also used as an extracurricular subject at the Ntra. Sra. Sagrado Corazón School in Madrid and the Villa de Móstoles School. In the 2017/2018 academic year, the program was continued in the six schools of the Foundation and at the Rihondo School in Alcorcón.

The results were measured through surveys administered to both teachers and students over the last two academic years (2016/2017 and 2017/2018). Specifically, 2050 students from the six schools of the Franciscan Foundation of Montpellier, the Villa de Móstoles School and the Rihondo School in Alcorcón were surveyed. All of these were in secondary education, distributed across curricular subjects (53.2%), extracurricular activities (36.2%), and a small percentage of specific events (10.6%) commonly organized in the schools, such as Open Days and Family Days. In total, nine teachers were responsible for delivering this content (six from the Foundation and three for extracurricular activities), who were also surveyed.

5.1. Student Surveys

On the question of whether it was easy to learn, more than 54% of students gave scores of 8–10, while a little fewer than 26% gave scores of 5–7. Taking into account that their initial level was very low or zero, and that the objectives of the educational proposal were quite ambitious, the results are more than positive: the framework was easy to learn.

More than 70% reported finding robotics very interesting (scoring between 8–10). The materials received, the `PyBoKids` manual slides used in each session, and the brief introduction to the topic of such session were scored as 8–10 by more than 60%, while slightly fewer than 40% of students rated them between 5–7. More than 70% found the practice activities performed, that is, the exercises, very interesting (8–10).

Taking all the above into account, Figure 9 shows the overall assessment given to the course with `PyBoKids`.

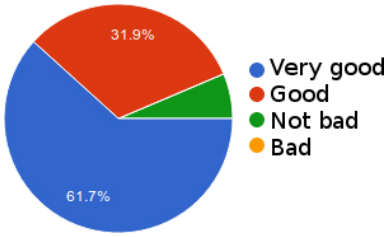

Figure 9. General assessment of PyBoKids by students.

5.2. Teacher Surveys

The evaluation of the teaching staff regarding the deployment of our educational proposal is also very positive. The overall assessment of the educational proposal is shown in Figure 10.

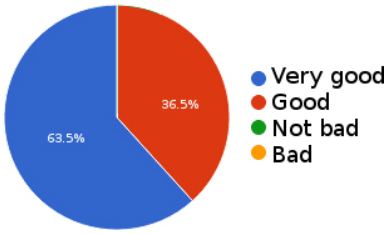

Figure 10. General assessment of the educational proposal by teaching staff.

Eight teachers scored 4/4 on the question of whether the students followed the classes easily; another scored it 3/4.

In all cases, they considered that the academic performance of their students improved—half of the respondents considered that the academic performance of their students improved, rating it 4/4, since the average grade of the class improved by 2 points, while the other half rated it 3/4, given that in their cases, the average grade improved by 1 point. As described above, the grades are a combination of student self-evaluation and teacher assessment.

5.3. Discussion

The results were satisfactory. However, the surveys show slightly different ratings among students in the curricular and extracurricular classes. This is arguably because, in the first case, the students had more limited time and usually showed high interest in the classes, while in the extracurricular classes, they had more time but tended to be less interested.

In two of the schools, there had been no previous use of robotics; little use in two others; in another one, moderate use; and, finally, considerable use in two others. Another positive indication is that after the deployment of our educational proposal, all the schools, without exception, have embraced robotics with great enthusiasm and held various competitions and workshops throughout the academic year.

6. Conclusions

New technologies are changing the way people around the world are having an ever-increasing impact on society. Educational institutions are promoting reforms that take this revolution into account, both to take advantage of the educational possibilities opened up by these technologies, and to train students in using them. Not only is the content transmitted changing, but also the way the classes themselves are focused. However, much is still to be done. On the one hand, teaching staff must be willing and/or prepared to change the classic teaching–learning paradigms, while on the other, educational institutions have to refine and develop more solid, validated educational proposals.

Based on this analysis and our experience in Spanish schools, an educational proposal on how to introduce robotics into pre-university education has been designed and presented. The proposal was formed by the teaching environment `PyBoKids`, which serves as a tool and includes concrete academic content, as well as by a constructivist methodology to teach the classes.

The environment developed uses robotic platforms based on an Arduino processor, and allows students to program the robot in Python language. It also allows practice with simulated robots. In addition, it includes an academic program organized into four phases of progressive complexity, ranging from the introduction to programming and the Python language to programming with loose sensors and actuators, and the programming of behaviors or tasks in robots.

This educational proposal has been implemented in several schools and followed by 2000 students in the last two years. Its impact has been measured through surveys, and the results have been very satisfactory. Students and teachers showed high levels of acceptance and satisfaction with the program. In addition, the robotic projects carried out by the students demonstrate a high level of assimilation of concepts, while the class dynamics were always pleasing.

This teaching environment and its accompanying methodology are expected to contribute to a long-term improvement in educational indicators in Spain, reducing the gap in educational quality with other countries such as Finland.

As future lines of research, firstly, a new physical platform called `PiBot` is being built, adding a RaspberryPi-3 on top of a chassis with the usual sensors and actuators. The aim is to increase the processing capacity on board and add the Pi-Cam camera, so that it is possible to introduce new activities with simple artificial vision.

Secondly, the use of Jupyter booklets (*Notebooks*) for Python is being explored, with the idea of using the web browser itself as the editor of the student's program.

Author Contributions: Conceptualization, J.V. and J.M.C.; methodology, J.V. and J.M.C.; software, J.V.; validation, J.V. and J.M.C.; formal analysis, J.M.C.; investigation, J.V.; resources, J.M.C.; data curation, J.V.; writing-original draft preparation, J.V. and J.M.C.; writing-review and editing, J.V. and J.M.C.; visualization, J.V. and J.M.C.; supervision, J.M.C.; project administration, J.M.C.; funding acquisition, J.M.C.

Funding: This work was partially funded by the Community of Madrid through the RoboCity2030-DIH-CM (S2018/NMT-4331) and by the Spanish Ministry of Economy and Competitiveness through the RETOGAR project (TIN2016-76515-R). The APC was funded by the RoboCity2030-DIH-CM (S2018/NMT-4331).

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Rodger, S.H.; Walker, E.L. Activities to attract high school girls to computer science. In Proceedings of the 27th SIGCSE Technical Symposium on Computer Science Education, Philadelphia, PA, USA, 15–17 February 1996.
2. Altin, H.; Pedaste, M. Learning approaches to applying robotics in science education. *J. Balt. Sci. Educ.* **2013**, *12*, 365–377.
3. Mubin, O.; Stevens, C.J.; Shahid, S. A review of the applicability of robots in Education. *Technol. Educ. Learn.* **2013**, *1*, 13. [CrossRef]
4. Vega, J. Educational Framework Using Robots with Vision for Constructivist Teaching robotics to Pre-University Students. Ph.D. Thesis, University of Alicante, Alicante, Spain, 2018.
5. Cerezo, F.; Sastrón, F. Laboratorios Virtuales y Docencia de la Automática en la Formación Tecnológica de Base de Alumnos Preuniversitarios. *Rev. Iberoam. Autom. Inform. Ind. RIAI* **2015**, *12*, 419–431. [CrossRef]
6. Jiménez, E.; Bravo, E.; Bacca, E. Tool for experimenting with concepts of mobile robotics as applied to children education. *IEEE Trans. Educ.* **2010**, *53*, 88–95. [CrossRef]
7. Japan-Economic. New Robot Strategy, 2015. Available online: https://www.meti.go.jp/english/press/2015/pdf/0123_01b.pdf (accessed on 15 February 2019).
8. Magnenat, S.; Shin, J.; Riedo, F.; Siegwart, R.; Ben-Ari, M. Teaching a core CS concept through robotics. In Proceedings of the 2014 Conference on Innovation & Technology in Computer Science Education, Uppsala, Sweden, 23–25 June 2014; pp. 315–320.
9. Merkouris, A.; Chorianopoulos, K.; Kameas, A. Teaching programming in secondary education through embodied computing platforms: Robotics and wearables. *ACM Trans. Comput. Educ. (TOCE)* **2017**, *17*, 9. [CrossRef]
10. Kubilinskiene, S.; Zilinskiene, I.; Dagiene, V.; Sinkevièius, V. Applying robotics in School Education: A Systematic Review. *Balt. J. Mod. Comput.* **2017**, *5*, 50. [CrossRef]
11. Eguchi, A. RoboCupJunior for promoting STEM education, 21st century skills, and technological advancement through robotics competition. *Robot. Auton. Syst.* **2016**, *75*, 692–699. [CrossRef]
12. Navarrete, P.; Nettle, C.J.; Oliva, C.; Solis, M.A. Fostering Science and Technology Interest in Chilean Children with Educational Robot Kits. In Proceedings of the XIII Latin American robotics Symposium and IV Brazilian robotics Symposium (LARS/SBR), Recife, Brazil, 8–12 October 2016; pp. 121–126.

13. Kandlhofer, M.; Steinbauer, G. Evaluating the impact of robotics in education on pupils' skills and attitudes. In Proceedings of 4th International Workshop Teaching robotics and 5th International Conference robotics in Education, Padova, Italy, 18 July 2014; pp. 101–109.
14. Jormanainen, I.; Korhonen, P. Science Festivals on Computer Science Recruitment. In Proceedings of the 10th Koli Calling International Conference on Computing Education Research (Koli Calling '10), Koli, Finland, 28–31 October 2010; ACM: New York, NY, USA, 2010; pp. 72–73. [CrossRef]
15. Graven, M.; Stott, D. Exploring online numeracy games for primary learners: Sharing experiences of a Scifest Africa Workshop. *Learn. Teach. Math.* **2011**, *2011*, 10–15.
16. Puelles Benitez, M.D. La educación secundaria en la España democrática: Antecedentes, problemas y perspectivas. *Cuad. Pesqui.* **2011**, *41*, 710–731. [CrossRef]
17. Vega, J.; Cañas, J. Entorno docente con Arduino y Python para Educación Robótica en Secundaria. In *JITICE 5th Workshop, Educational Innovation and ICT*; Rey Juan Carlos University: Madrid, Spain, 2016.
18. Vega, J.; Cañas, J. Curso de Robótica en Educación Secundaria usando constructivismo pedagógico. In *JITICE 4th Workshop, Educational Innovation and ICT*; Rey Juan Carlos University: Madrid, Spain, 2014.
19. Araujo, A.; Portugal, D.; Couceiro, M.S.; Rocha, R.P. Integrating Arduino-Based Educational Mobile Robots in ROS. *J. Intell. Robot. Syst.* **2015**, *77*, 281–298. [CrossRef]
20. Jamieson, P. Arduino for Teaching Embedded Systems. Are Computer Scientists and Engineering Educators Missing the Boat? In Proceedings of the International Conference on Frontiers in Education: Computer Science and Computer Engineering, Las Vegas, NV, USA, 18–21 July 2011.
21. Chaudhary, V.; Agrawal, V.; Sureka, P.; Sureka, A. An experience report on teaching programming and computational thinking to elementary level children using lego robotics education kit. In Proceedings of the IEEE Eighth International Conference on Technology for Education (T4E), Mumbai, India, 2–4 December 2016; pp. 38–41.
22. Filippov, S.; Ten, N.; Shirokolobov, I.; Fradkov, A. Teaching robotics in secondary school. *IFAC-PapersOnLine* **2017**, *50*, 12155–12160. [CrossRef]
23. Vega, J.; Cañas, J. PiBot: An Open Low-Cost Robotic Platform with Camera for STEM Education. *Electronics* **2018**, *7*, 430. [CrossRef]
24. Junior, L.A.; Neto, O.T.; Hernandez, M.F.; Martins, P.S.; Roger, L.L.; Guerra, F.A. A low-cost and simple arduino-based educational robotics kit. *Cyber J. Multidiscip. J. Sci. Technol. J. Sel. Areas Robot. Control (JSRC)* **2013**, *3*, 1–7.
25. Plaza, P.; Sancristobal, E.; Fernandez, G.; Castro, M.; Pérez, C. Collaborative robotic educational tool based on programmable logic and Arduino. In Proceedings of the Technologies Applied to Electronics Teaching (TAEE), Seville, Spain, 22–24 June 2016; pp. 1–8.
26. Balogh, R. Educational robotic platform based on arduino. In Proceedings of the 1st International Conference on robotics in Education, RiE2010, Bratislava, Slovakia, 16–17 September 2010; pp. 119–122.
27. Afari, E.; Khine, M. robotics as an educational tool: Impact of LEGO mindstorms. *IJIET* **2017**, *7*, 437–442. [CrossRef]
28. Beyers, R.N.; van der Merwe, L. Initiating a pipeline for the computer industry: Using Scratch and LEGO robotics. In Proceedings of the Conference on Information Communication Technology and Society (ICTAS), Durban, South Africa, 8–10 March 2017; pp. 1–7.
29. Mondada, F.; Bonani, M.; Riedo, F.; Briod, M.; Pereyre, L.; Rétornaz, P.; Magnenat, S. Bringing robotics to formal education: The thymio open-source hardware robot. *IEEE Robot. Autom. Mag.* **2017**, *24*, 77–85. [CrossRef]
30. Roy, D.; Gerber, G.; Magnenat, S.; Riedo, F.; Chevalier, M.; Oudeyer, P.Y.; Mondada, F. IniRobot: A pedagogical kit to initiate children to concepts of robotics and computer science. In Proceedings of the 6th International Conference on robotics in Education (RIE 2015), Yverdon, Switzerland, 20–22 May 2015.
31. Demetriou, G.A. Mobile robotics in education and research. In *Mobile Robots-Current Trends*; InTech: London, UK, 2011.
32. Stone, A.; Farkhatdinov, I. robotics Education for Children at Secondary School Level and Above. In Proceedings of the Conference Towards Autonomous Robotic Systems, Guildford, UK, 19–21 July 2017; Springer: Cham, Switzerland, 2017; pp. 576–585.

33. Witherspoon, E.B.; Higashi, R.M.; Schunn, C.D.; Baehr, E.C.; Shoop, R. Developing computational thinking through a virtual robotics programming curriculum. *ACM Trans. Comput. Educ. (TOCE)* **2017**, *18*, 4. [CrossRef]
34. Olabe, J.; Olabe, M.; Basogain, X.; Castaño, C. Programming and robotics with Scratch in primary education. In *Education in a Technological World: Communicating Current and Emerging Research and Technological Efforts*; Formatex: Badajoz, Spain, 2011; pp. 356–363.
35. Plaza, P.; Sancristobal, E.; Carro, G.; Castro, M.; Blázquez, M.; Muñoz, J.; Álvarez, M. Scratch as Educational Tool to Introduce robotics. In Proceedings of the International Conference on Interactive Collaborative Learning, Budapest, Hungary, 27–29 September 2017; Springer: Cham, Switzerland, 2017; pp. 3–14.
36. Naya, M.; Varela, G.; Llamas, L.; Bautista, M.; Becerra, J.C.; Bellas, F.; Prieto, A.; Deibe, A.; Duro, R.J. A versatile robotic platform for educational interaction. In Proceedings of the 9th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications (IDAACS), Bucharest, Romania, 21–23 September 2017; Volume 1, pp. 138–144.
37. Sentance, S. *Annual National Survey 2015: Results*; Technical Report; Computing At School: Swindon, UK, 2015. Available online: https://community.computingatschool.org.uk/files/6098/original.pdf (accessed on 23 January 2019).
38. Bain, K. *Lo que Hacen los Mejores Profesores Universitarios*; PUV: Valencia, Spain, 2007.
39. Vega, J. El Humor en el Aula de Matemáticas. Master's Thesis, Rey Juan Carlos University, Madrid, Spain, 2011.
40. Dewey, J. *Experience and Education*; Simon and Schuster: New York, NY, USA, 2007.
41. Morón, C. La Mejora de la Práctica Docente a Través de la Metodología de Proyectos de Investigación. Ph.D. Thesis, University of Malaga, Malaga, Spain, 2015.
42. Schul, J. Revisiting an Old Friend: The Practice and Promise of Cooperative Learning for the Twenty-First Century. *Soc. Stud.* **2011**, *102*, 88–93. [CrossRef]
43. Acosta-Nassar, C.A. El uso de una estrategia híbrida entre aprendizaje basado en problemas y clases magistrales para mejorar aprendizajes. *Educare Electron. J.* **2014**, *18*, 143–158. [CrossRef]
44. Opoku, E.; Lehtinen, E. *Responding to Changing Student Demographics in Finland: A Study of Teachers' Developing Cultural Competence*; University of Turku: Turku, Finland, 2015; ISBN 978-951-29-6336-2.
45. OECD. *Results in Focus*; OECD: Paris, France, 2015.
46. Tirri, K. The last 40 years in Finnish teacher education. *J. Educ. Teach.* **2014**, *4*, 600–607. [CrossRef]
47. Jormanainen, I.; Erkki, S. *Supporting Teachers in Unpredictable Robotics Learning Environments*; University of Eastern Finland: Kuopio, Finland, 2013; ISBN 978-952-61-1230-5.
48. Tirri, K.; Kuusisto, E. How Finland Serves Gifted and Talented Pupils. *J. Educ. Gifted* **2013**, *36*, 84–96. [CrossRef]
49. Cañas, J.; Martín, L.; Vega, J. Innovating in robotics education with Gazebo simulator and JdeRobot framework. In *XXII Congreso Universitario de Innovación Educativa en Enseñanzas Técnicas, CUIEET*; University of Castilla-La Mancha: Ciudad Real, Spain, 2014; Volume 2, pp. 1483–1496.
50. Von Glasersfeld, E. *Radical Constructivism: A Way of Knowing and Learning. Studies in Mathematics Education Series: 6*; Falmer Press: Bristol, PA, USA, 1995.
51. Steffe, L.; Gale, J. *Constructivism in Education*; Lawrence Erlbaum: Hillsdale, NJ, USA, 1995.
52. Lefoe, G. Creating constructivist learning environments on the web: the challenge in higher education. In Proceedings of the ASCILITE Conference Proceedings, Wollongong, Australia, 14–16 December 1998.