*electronics*

MDPI

*Article*

# PiBot: An Open Low-Cost Robotic Platform with Camera for STEM Education

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

Department of Telematic Systems and Computation, Rey Juan Carlos University, Camino del Molino S/N, 28934 Fuenlabrada, Madrid, Spain; jmplaza@gsyc.es

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

† These authors contributed equally to this work.

Received: 16 October 2018; Accepted: 10 December 2018; Published: 12 December 2018


**Abstract:** This paper presents a robotic platform, `PiBot`, which was developed to improve the teaching of robotics with vision to secondary students. Its computational core is the Raspberry Pi 3 controller board, and the greatest novelty of this prototype is the support developed for the powerful camera mounted on board, the PiCamera. An open software infrastructure written in Python language was implemented so that the student may use this camera as the main sensor of the robotic platform. Furthermore, higher-level commands were provided to enhance the learning outcome for beginners. In addition, a `PiBot` 3D printable model and the counterpart for the Gazebo simulator were also developed and fully supported. They are publicly available so that students and schools without the physical robot or that cannot afford to obtain one, can nevertheless practice, learn and teach Robotics using these open platforms: *DIY-PiBot* and/or *simulated-PiBot*.

**Keywords:** teaching robotics; science teaching; STEM; robotic tool; Python; Raspberry Pi; PiCamera; vision system

## 1. Introduction

Over the last decade, technology has become increasingly common in the majority of contexts of daily and industrial life. A machine's capacity for taking optimum decisions in real time and simultaneously handling a large quantity of data is undoubtedly far greater than that of a human being. *Industrialization 4.0* [1] involves the integration of complex robotic systems in factories, logistics and what is known as the *Internet of things*, where sophisticated automatons handle an immense quantity of data to take strategic decisions for companies.

In addition to a large computational capacity, these mobile and intelligent robots need a complex sensory system to act intelligently not only in factories but in general robot–human interaction [2]. The fixed automation of well-structured production chains is giving way to an unpredictable world and an unstructured reality, which underlines the need for a wide complementary range of sensors and actuators to attain complete autonomy [3].

Although cameras have not been the most used option in mobile robotics for some years (sonar and/or laser have been more commonly used as sensors), vision is currently the most widely used sensor and will definitely be the most commonly used in the long-term future, because of the possibilities it offers and the processing power of current computers. Cameras are low-cost devices that are potentially a rich source of information.

However, visual capacity in robots, in contrast to that of living beings, is not a simple technique. The main difficulty lies in extracting useful information from the large amount of data that a camera provides. Good algorithms are needed for this task.

Summarizing, the advance of Artificial Intelligence (AI), robotics and automation in society [4], and the future of work and industry, in particular, converge in what is already referred to as the fourth industrial revolution. According to the analysis of the University of Oxford [5] and the professional services of Deloitte [6], almost half of all jobs will be occupied by robots in the next 25 years. Furthermore, as the McKinsey Institute shows in its most recent report on the global economy [7], robots will perform the work of about 800 million employees in 2030.

It is therefore important to incorporate technology, and specifically robotics with vision systems, in the pre-university educational system since, within ten years, todays' youngest students will have to confront a labor market demanding profiles related to automation of systems [8]. From the educational point of view, robotics is a field where many areas converge: electronics, physics (Figure 1 left), mechanics (Figure 1 right), computer sciences, telecommunications, mathematics, etc.

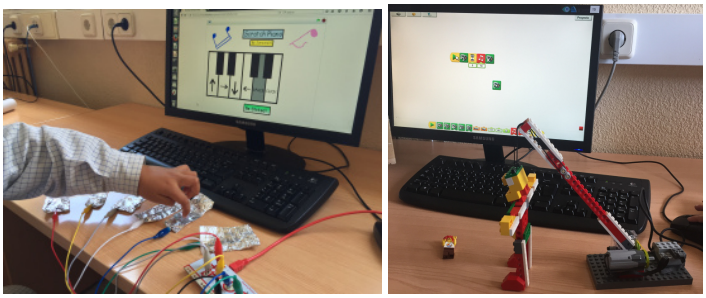

**Figure 1.** Different robotic prototypes to work in different educational areas.

Hence, robotics is growing in importance in pre-university education, either as a field of knowledge in itself or as a tool to present technology and other subjects [9,10] to young students in an attractive way. Furthermore, it has the power to motivate students, bringing technology closer to young people [11] by using robotics as a tool to present basic concepts of science [12], technology, engineering and mathematics (STEM) [13]. In an almost play-like context, students learn notions, which are difficult and complex to explain or to assimilate in the classic masterclass [14,15].

The implementation of robotics in higher education is already in place. Six states in the U.S. (Iowa, Nevada, Wisconsin, Washington, Idaho and Utah) have announced plans and investments with this aim in mind in the last five months. 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* [16], makes clear 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), where it is used as a teaching tool [17–19].

Another example of the increasing importance of robotics in education are the robotics competitions for teenagers that have appeared in recent years, which encourage interest in this technology. At 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. It is worth highlighting the RoboCup Junior (http://rcj.robocup.org) [20–22], with tests such as the rescue or the robotic soccer, as well as the First Lego League (FLL) and the VEX Robotics Competitions (https://www.vexrobotics.com/vexedr/competition). In Finland, the SciFest competition (http://www.scifest.fi) attracts students from all over Europe [23] and has agreements with educational institutions in South Africa [24].

In the academic community, several conferences also highlight the role of robotics in education, including the Conference on Robotics in Education (RIE), and the Workshop on Teaching Robotics with ROS (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.

To support this increasing presence of educational robotics, there are many teaching frameworks used to teach robotics to children, ranging from those focused on primary education to more powerful ones designed for secondary education and high school. They are usually composed of a concrete robotic platform, i.e., a robot, which is programmed in a certain language using software tools. Students engage in different exercises, challenges or projects (practice activities). They teach the basic operation of sensors, actuators and the rudiments of programming.

## 2. Robotic Platforms for STEM Education

Most robots used in commercial educational platforms are proprietary. It is worth mentioning the well-known Lego, which has featured for some years in educational robotics kits, with different versions: Mindstorms RCX, NXT, EV3 and WeDo [15,21].

Arduino boards appeared some years ago, in an effort to work around the closed-platforms limitation, providing cheaper and more adapted robotic platforms. This is a free hardware board that allows a wide variety of low-cost robotic components to be added [21,25–28]. Thus, beginning with a basic and affordable Arduino platform, teachers and students can freely adapt it to their necessities, developing an effective and low-cost robot, as described in [29–32].

Other platforms are Thymio (Figure 2 left) [17,33,34], Meet Edison's or VEX robots (Figure 2 middle and right), and simulated environments such as TRIK-Studio [28,35] or Robot Virtual Worlds (RVW) [36].

**Figure 2.** Robots Thymio, VEX IQ and VEX CORTEX.

Beyond the robotic hardware platforms, different software frameworks can be found in educational robotics. Lego has its own option, EV3-software. In addition, Lego Mindstorms and WeDo can now be interfaced through Scratch language [32,37], especially version 3.0, which will be officially online from the beginning of next year. Other variants are Blockly [38], Bitbloq or VPL. All of these contain graphic blocks that typically connect in sequence in a graphic editor. Arduino platforms can be programmed with a simple, C++ based, text language using Arduino-IDE. C++ is widely used at university level, but its complexity makes it unsuitable for pre-university students.

Exploring the existing literature, we found many other works that have presented robotic platforms for educational purposes and their underlying philosophy. In [39], the authors focused on a six degrees of freedom (DOF) serial robotic arm as a robotic platform for training purposes. They derived the kinematic and dynamic models of the robot to facilitate controller design. An on-board camera to scan the arm workspace is included.

In [40], Alers and Hu presented the AdMoVeo robotic platform, which was developed for the purpose of teaching the basic skills of programming to industrial design students. This platform lets students explore their creativity through their passion for graphic and behavioral design.

In [26], Jamieson examined whether Arduino was a suitable platform for teaching computer engineers and computer scientists by means of an embedded system course. He described a project-based learning embedded system course that has been taught and identified the topics covered in it compared to the IEEE/ACM recommendations. He finally concluded by saying that students

expressed high praise for the Arduino platform and that, compared to previous years, students' final projects were of better quality and more creative.

In [41], the authors presented eBug as a low-cost, open robotics platform designed for undergraduate teaching and academic research in areas such as multimedia smart sensor networks, distributed control, mobile wireless communication algorithms and swarm robotics. This prototype used the Atmel AVR XMEGA 8/16-bit micro-controller.

Miniskybot is presented in [42] as a mobile robot for educational purposes that is 3D-printable on low cost RepRap-like machines, fully open source (including mechanics and electronics), and designed exclusively with open source tools. It is based on an 8-bit pic16f876a micro-controller.

Nevertheless, there is no system, and much less a guided one, which maintains a constant level of motivation and challenge, especially one in which vision plays an important role. In fact, the majority of these kits or robotic platforms on the market focus on doing specific tasks or are designed to arouse the interest of either the youngest students or university students in robotics, but not so that students in pre-university courses acquire correct and complete training in programming, something that is in great demand and is widespread in almost any degree. Although it is true that other kits exist that are more specialized in specific scientific fields [43], our proposed framework goes further and provides all the open tools for both students and teachers [44] required for a complete academic year. Its versatile design puts at their disposal numerous sophisticated algorithms, including vision, with a pleasant and intuitive interface.

In addition, a large gap has been identified between the level of academic training at university level in scientific and technological degrees and the official curriculum implemented at pre-university levels, specifically in science subjects at secondary education level. Thus, the present work proposes to mitigate this gap, developing a complete teaching framework for robotics with vision, which today is non-existent, integrating:

1. A RaspberryPi-based open hardware platform, economically suitable for secondary schools to satisfy the needs of a complete class, but at the same time standardized and powerful, which allows the execution of algorithms of robotics with vision.
2. An open software infrastructure that is simple and intuitive for young students to manage but at the same time is powerful and versatile. It incorporates enough resource libraries, as well as diverse examples, to provide practical exercises in programming robots with vision that are sufficient in both number and complexity to continuously motivate students [45].
3. A wide repertoire of practice activities that can be followed during a complete academic year, including sufficient and properly staggered sessions for students to correctly assimilate the content [46].

## 3. Design of the PiBot Tool for STEM Education

Following the analysis of the most important available educational robots, this section describes the design of the proposed robot. It leverages some of the new possibilities offered by different technologies and aims to overcome limitations observed in current platforms such as having no cameras or not being usable with programming languages such as Python. Using `PiBot` with Python is not intended for primary education or first year secondary education, where visual languages such as Scratch are a better starting point [47]. Instead, it is designed for students of secondary education aged over 12 years and even introductory university courses.

Better tools improve learning processes in young learners. The `PiBot` education tool follows a three-part architecture, as shown in Figure 3: the robot platform, the software drivers and the exercises. The robot and the drivers can be regarded as the infrastructure for the exercises, which can be organized in courses or levels and focus on different aspects of robotics.

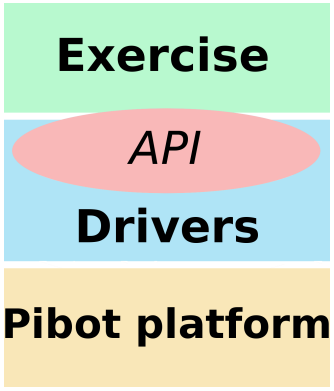


**Figure 3.** Architecture of the `PiBot` tool: hardware (platform) and software (drivers and exercise).

The creation of the `PiBot` tool followed several design principles:

1. Low cost (under 180 euros), to make it affordable for most schools and students.
2. Open: First, the robot hardware should be easy to assemble for students, who can also make the pieces using a 3D printer. Thus, the assembly of a `PiBot` can be an educational activity and interesting for the makers community. Second, drivers should be open source and publicly available.
3. Compatibility with common sensors and actuators in (Arduino-based) educational robots. In this way, if an Arduino-based robot is already available, the transition to `PiBot` is affordable; and, in any event, the acquisition of components for `PiBot` is very simple, given the wide availability of components for Arduino.
4. Including vision in an easy way. Cameras are useful sensors and this platform can give students easy and practical access to vision.
5. Supporting not only the real robot but also a simulated robot. Thus, even without a physical platform, the `PiBot` tool may be used to teach and learn robotics.
6. Python as a programming language because of its simplicity, expressive power and because it is widely used in higher levels of education and programming.

## 4. PiBot Robotic Platform

Robots are typically composed of a computer or a microprocessor, several sensors, actuators and some form of connectivity. Sensors provide information about the environment, the computer runs the robot software and actuators allow the robot to do things such as moving or performing actions in the real world.

### *4.1. Hardware Design*

The block diagram of the `PiBot` hardware is shown in Figure 4. The main computer is a Raspberry Pi 3 controller board (Figure 5 middle). It is more powerful than Arduino processors, maintains low costs, and runs a functional operating system based on Linux; specifically, the Raspbian Stretch distribution. It allows the use of standard development tools on the Linux community and the use of the PiCamera.

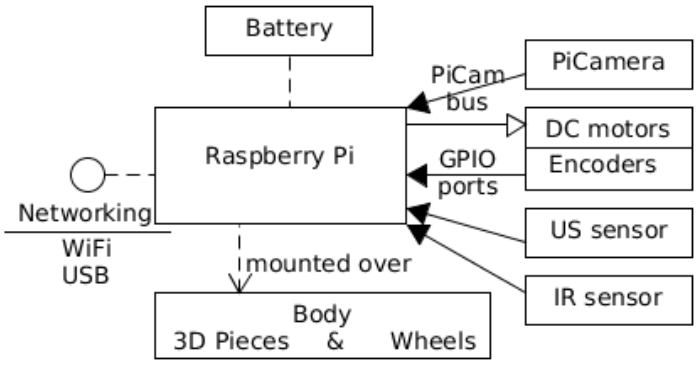


**Figure 4.** Hardware design of the `PiBot` robot.

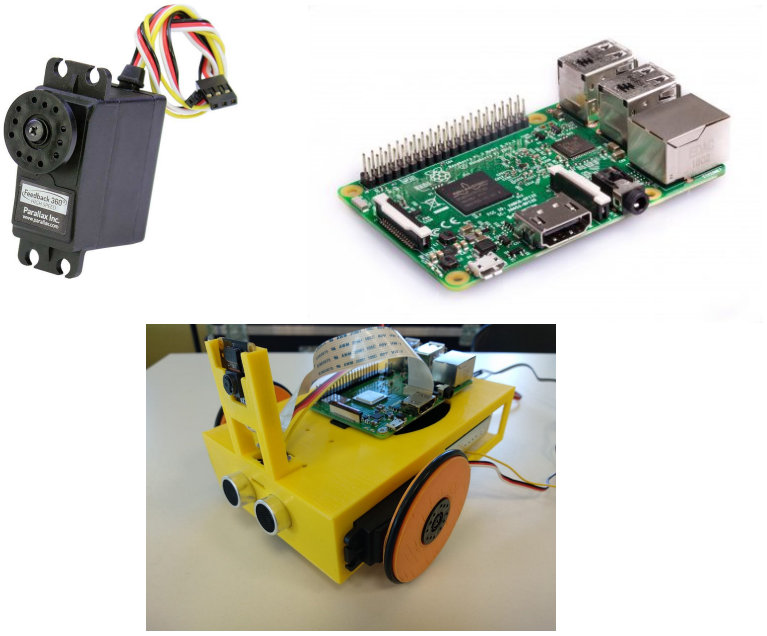

**Figure 5.** Motors, Raspberry Pi board and `PiBot` made with 3D printable pieces.

The sensors mounted onboard `PiBot` are:

- An ultrasound sensor model HC-SR04 (Figure 6 left)
- Infrared sensors
- Motor encoders
- Raspberry PiCamera (Figure 6 right). This is connected to the computer using a dedicated data bus. Its technical details are included in Table 1.

The ultrasonic (US), infrared (IR) and encoder sensors are connected to the Raspberry Pi board through several GPIO ports (General Purpose Input/Output). This protocol allows the connection and control of several devices at the same time and requires a configuration on each port to serve as input and output of data [48].

The actuators mounted onboard `PiBot` are two DC motors (Parallax Feedback 360° High Speed Servo (Figure 5 left)). They provide movement and differential drive to the `PiBot`. The motors include encoders and are connected to the main processor through GPIO bus.

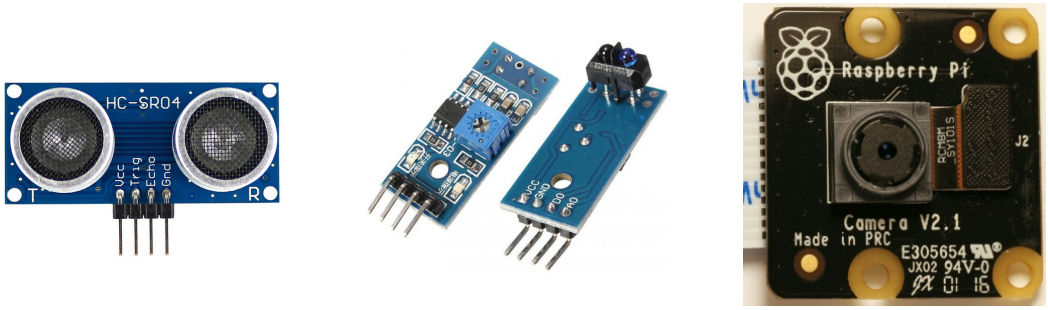


**Figure 6.** Ultrasonic sensor model HC-SR04, IR sensors and Raspberry PiCamera.

**Table 1.** PiCamera (v2.1 board) technical intrinsic parameters.

| PiCamera Params. | Values |
|---|---|
| Sensor type | Sony CMOS 8-Mpx |
| Sensor size | 3.6 × 2.7 mm (1/4" format) |
| Pixel count | 3280 × 2464 (active px.) |
| Pixel size | 1.12 × 1.12 um |
| Lens | f = 3.04 mm, f/2.0 |
| Angle of view | 62.2 × 48.8 degrees |
| SLR lens equivalent | 29 mm |

All these components are assembled into a body made of 3D printable pieces. The 3D printable models of all the chassis pieces are publicly available on the web (https://github.com/JdeRobot/JdeRobot/tree/master/assets/PiBot). The body also allocates a battery of 10,000 mAh, which provides power to all the electronic onboard devices. An official list of components and some tentative providers are also available at the same webpage so that anyone can buy the components, print the pieces and build a `PiBot`.

### *4.2. Simulated Robot*

The Gazebo simulator (http://gazebosim.org) was selected for simulation of the `PiBot` platform. This is an open source robotic simulator powered by Open Robotics Foundation and the de facto standard in the robotics scientific community. It provides a physical engine so collisions and realistic movements are provided.

The students can program an exercise and run their code seamlessly both on the physical `PiBot` or on the simulated `PiBot` inside Gazebo, at will. The student code lies on top of the `PiBot` API (Application Programming Interface), which is used to obtain sensor readings and to command actuator orders. The API is exactly the same in both cases. In the first one, drivers will be used to connect to the physical devices. In the second one, other drivers will exchange messages with the simulator to implement the same functions.

To support this new robot, a 3D model of the robot was developed (Figure 7). In addition, several plugins were also integrated for the simulation of the onboard camera, the distance sensor (sonar) and IR sensors. IR support was implemented using small cameras. Each IR consists of a 4 × 4 pixel camera and an additional code that computes the virtual IR measurement from the values of these pixels. The movement was also supported with the corresponding Gazebo plugin, which also provides a 2D position sensor (as encoder).

The 3D `PiBot` model and all the developed plugins are publicly available on the web (https://github.com/JdeRobot/JdeRobot/tree/master/assets/gazebo; https://github.com/JdeRobot/JdeRobot/tree/master/src/drivers/gazebo/plugins/pibot).

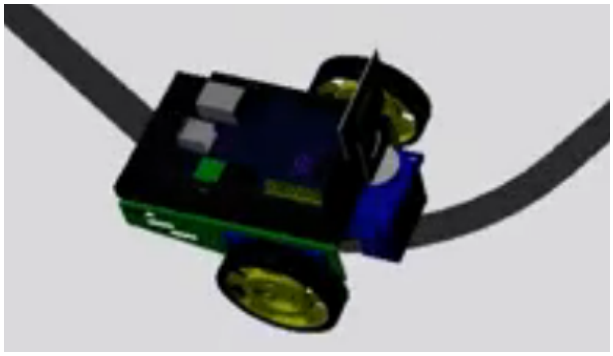

**Figure 7.** PiBot robot simulated in Gazebo.

## 5. Software Infrastructure

Python was chosen as a programming language to support `PiBot` because of its simplicity, its expressive power and because it is widely used in higher levels of education and many industries (in conjunction with powerful libraries). It is a text language, interpretative and object oriented. It is easier to learn than other also widely used programming languages, such as C/C++ or Java, and at the same time it is highly powerful. It is a real world language but accessible for pre-university students.

Using the proposed educational tool, the students program their exercises in Python by writing the file *exercise.py*, for example, with a text editor. This program uses the `PiBot` Application Programming Interface (API) to control the robot, which contains a set of natural methods to read the measurements from the robot sensors (US, IR, and camera) and methods to give commands to the robot actuators (DC motors). The most important API methods are detailed in Table 2.

Two different libraries were developed to support this API. One runs onboard the `PiBot` Raspberry Pi and a second one communicates with the simulated robot inside Gazebo. As the programming interface is the same in both cases, the student application works interchangeably on the physical platform and the simulated one. The final robot in each case is selected by specifying it in the configuration file.

Using this API, students concentrate on the algorithm they are developing, on the robot's intelligence, avoiding the low level details such as ports, connectivity with the robot, etc., which are stored in the library configuration file.

**Table 2.** Application Programming Interface (API).

| | **Actuators** | **Sensors** |
|---|---|---|
| **Low level methods** | RightMotor(V)<br>LeftMotor(V) | readUltrasound<br>readInfrared<br>getImage |
| **High level methods** | move(V, W) | getColoredObject(color)<br>getDistancesFromVision<br>getRobotPosition |

The API methods can be divided into low and high level methods. The low level methods provide access to a single device, such as readUltrasound, readInfrared or getImage. RightMotor(V) controls the single right motor commanding desired speed, as does LeftMotor(V) for the other motor. The high level methods provide a simpler and more compact way to control the whole robot or two vision functions to get useful information from the image in an easy way. These are described below.

### *5.1. Drivers for the Real PiBot*

To support the real `PiBot`, two modules were programmed, as shown in Figure 8. One includes the management of the PiCamera and the other deals with GPIO devices (US sensor, IR sensor and motors).

They were programmed in Python using standard libraries available in the Python community. It is publicly available on the web (https://github.com/JdeRobot/JdeRobot/tree/master/src/drivers/PiBot/real). The image processing functionality also relies on OpenCV.

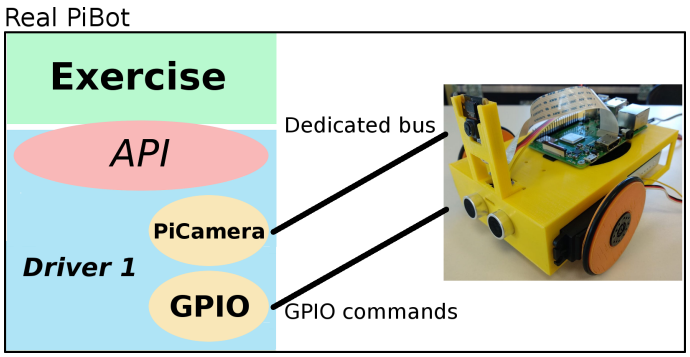


**Figure 8.** Connection of the library with the real `PiBot`.

### *5.2. Drivers for the Simulated PiBot*

To support the simulated `PiBot` on Gazebo, a specific library was developed, which connects with the previously mentioned plugins (Figure 9), exchanging messages through the ICE communication layer. It achieves sensor readings and camera images through network interfaces built in the JdeRobot project (https://jderobot.org). It is also publicly available on the web (https://github.com/JdeRobot/JdeRobot/tree/master/src/drivers/PiBot/Gazebo).

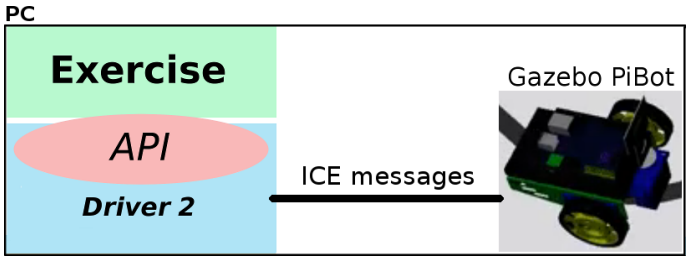


**Figure 9.** Connection of the library with the simulated `PiBot`.

### *5.3. Movement Control*

Regarding motors, beyond the low level methods RightMotor(V) and LeftMotor(V), a new high level method is provided for simpler control of the robot movements: Move(V,W). As parameters, this method accepts the desired linear speed $V$ and the desired rotation speed $W$, internally translating them into commands to the left and right motors so that the whole robot moves in accordance with $V$ and $W$. It takes into account the geometry of the `PiBot` and its wheels.

This function provides general 2D movement control: the `PiBot` may rotate without displacement (setting $V = 0$ and using $W$) both left or right (depending on the sign of $W$), may advance in a straight line (setting $W = 0$ and using $V$) both backwards and forward (depending on the sign of $V$), and may move in generic arcs advancing and rotating at the same time.

This is a useful speed control when programming reactive behaviors, which is better than position-based control.

### *5.4. Vision Support*

One advantage of the `PiBot` educational tool is its support for the camera. This allows many new exercises with vision and vision-based behaviors. It also introduces the students to computer vision in a simple and natural way. Two functions (getColoredObject(color) and getDistancesFromVision()) have been included so far in the `PiBot` API to easily get useful information from images, because the low

level method getImage() and the pixels processing are too complex for high school students. They were implemented and included in a vision library which performs complex image processing, hides all the complexity inside and is simple and intuitive to use. It internally employs OpenCV library, a standard in the Computer Vision community.

First, the high level method getColoredObject(color) accepts the desired *color* as input parameter and filters all the pixels within a range of that color (some of which are already predefined in the library: orange, red or blue) in the current camera image. It delivers as output the position of the colored object inside the image (its mean X and Y value) and its size (the number of detected pixels of that color). It works with single objects, as can be seen in Figure 10.

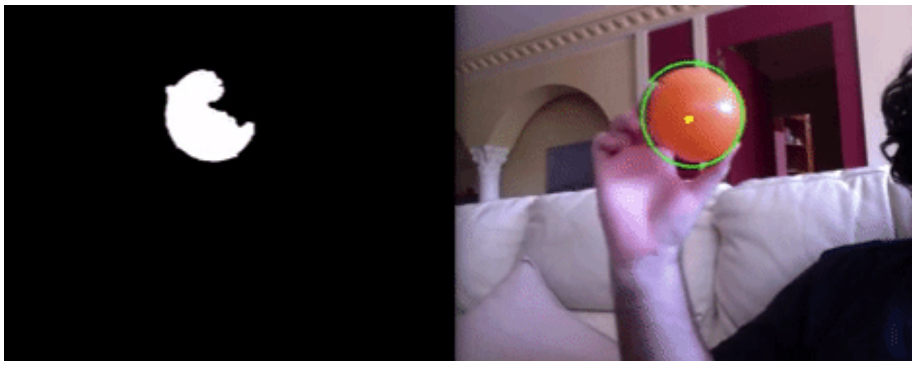

**Figure 10.** GetColoredObject function for orange color with empirical predefined ($H_{min}$, $H_{max}$, $S_{min}$, $S_{max}$, $V_{min}$, and $V_{max}$) ranges.

It uses HSV color space and OpenCV filtering methods. This function on `PiBot` API allows for exercises such as Object-Following, which is described in the next section.

Second, the high level method getDistancesFromVision() computes the distance to obstacles in front of the `PiBot` and provides a depth map from the robot to the surrounding objects. Typically the sonar sensor measures the distances in one direction. Using the camera for the same operation, the angular scope is extended to the camera field of view (around 60 degrees).

The vision library developed contains an abstract model of the camera (pin-hole) and several projective geometry algorithms. The camera parameters are known (K matrix and relative position inside the robot). As the `PiBot` only has a single camera, no stereo technique can be used for depth estimation. Instead, the implementation of the getDistancesFromVision() method assumes that all objects lie on the floor and the floor surface has a uniform color (ground hypothesis). It sweeps all the columns of the current image from its bottom. When the first edge pixel is found on a column, it is backprojected into 3D space, using ray tracing and the pin-hole camera model. The intersection of this ray with the floor plane is the estimated position of this edge in 3D space, and its distance to the robot is computed. In this way, the 3D point corresponding to each bottom pixel of the obstacle in the image can be obtained (Figure 11).

For instance, the left-hand side of Figure 12 shows the image coming from the camera, with the white floor (the battery was safely ignored as only green pixels were taken into account for explanatory purposes in this test). On the right-hand side, the estimated depths for the green object are displayed as red points and the field of view is also shown as a white trapezoid. The estimated distances are regularly consistent and correct.

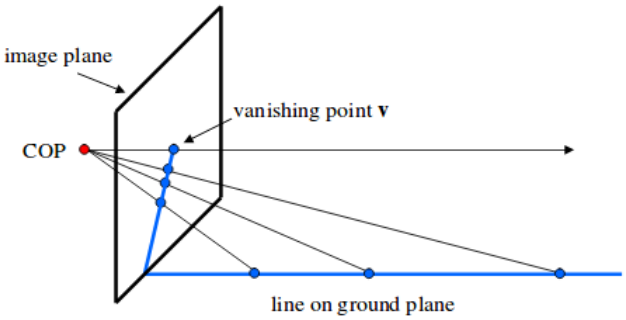


**Figure 11.** Ground Hypothesis assumes all objects are on the floor.

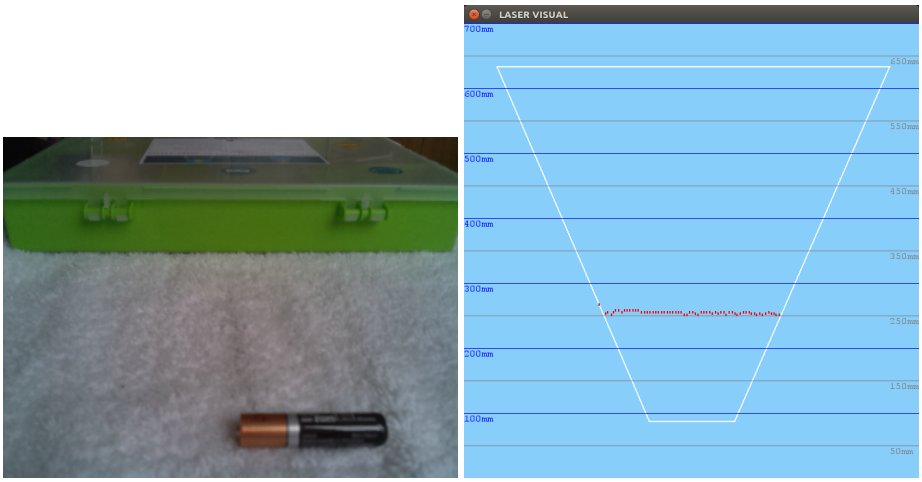


**Figure 12.** Example of visual sonar reading with 25 cm object shown using 3D scene simulator.

This getDistancesFromVision() function on `PiBot` API allows for exercises such as robot navigation with obstacles. An example is the vision-based obstacle avoidance which is described in the next section.

### *5.5. Downloading Software for Each Exercise into the Real Robot*

Since Raspberry Pi is the brain of the `PiBot`, and unlike current educational robots with an Arduino processor, this board is a fully functional PC (Personal Computer) running a Raspbian Linux OS (Operating System), there are many ways to transfer the Python program to the real robot.

One of the Pi's USB (Universal Serial Bus) ports can be used to insert a memory stick which includes the program to be executed, but the easiest way to transfer it is by connecting the Raspberry Pi to the Internet via either Wi-Fi or Ethernet. When it is connected, its IP (Internet Protocol) address can be found to remotely transfer the program from a PC—which can be running over Windows, Linux or MacOS operating systems—using a graphical SCP client, such as Filezilla, or typing and executing text based commands on a terminal.

Another method to transfer files from the PC to the Raspberry is using Virtual Network Computing (VNC) software. It is only necessary to have a VNC server previously installed on the board and a VNC viewer installed on the PC.

Nevertheless, not only can students remotely transfer their codes but they can also remotely execute them using the SSH protocol both graphically, with tools such as PuTTY, or using commands on a terminal. In addition, because the Raspberry is a full PC, students can even develop the exercises directly on the Raspberry board. It is only necessary to connect it to a screen through the HDMI port, and plug in a keyboard/mouse through the USB ports.

## 6. Exercises for Students Using PiBot

The program of activities for an introductory course on robotics was prepared. It includes basic exercises with a real robot (learning to deal with its isolated sensors and actuators), basic behaviors (combining both sensors and actuators on a single program) and vision-based behaviors.

It covers the syllabus of the subject of *Programming, Robotics and Technology* in higher education in Madrid (Spain). It also covers the typical topics of any robotics introductory course.

### *6.1. Basic Exercises*

Students can begin assembling different components on the `PiBot` and review some basic concepts of electronics so they have no problems when connecting the different components, such as infrared or ultrasound sensors (Figure 13).

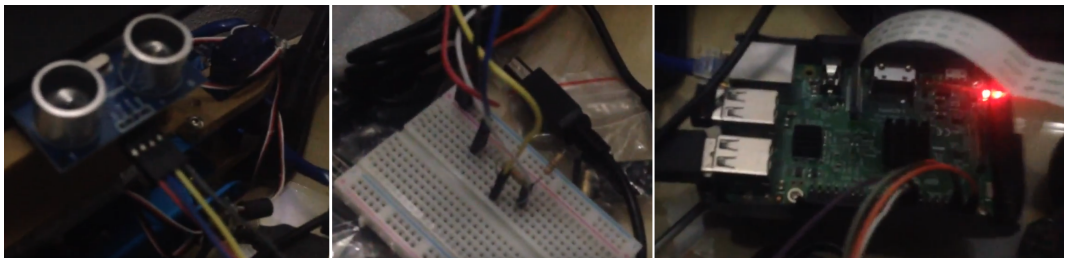

**Figure 13.** Practice activity with `PiBot` to handle an ultrasonic sensor.

The exercises here were designed to achieve the objective of *Disassembling objects, analyze their parts and their functions* from the unit *The Technological Process* and all the objectives included in the units of *Electricity*, *Materials for Technical Use* and *Programming*.

### *6.2. Basic Behaviors*

This second step consists of a complete robotics project where students combine everything they have previously learnt. Exploring the existing literature, we found a set of exercises that are frequently used in different teaching frameworks and academic proposals, in an almost cross-curricular manner. One of the classic projects is the follow-line behavior [15,21,28,35], as shown in Figure 14. In this project, the robot has IR sensors pointing to the ground, which is white but has a thick black line. Another is the avoidance of obstacles [21,35], as shown in Figure 15, where the robot has an ultrasound sensor that allows it to detect objects that interfere with its movement. The student's program must then order the motors to stop and turn until they find a free space to go forward again.

These exercises were designed to achieve the objectives included in typical *Hardware and Software* and *Programming* units.

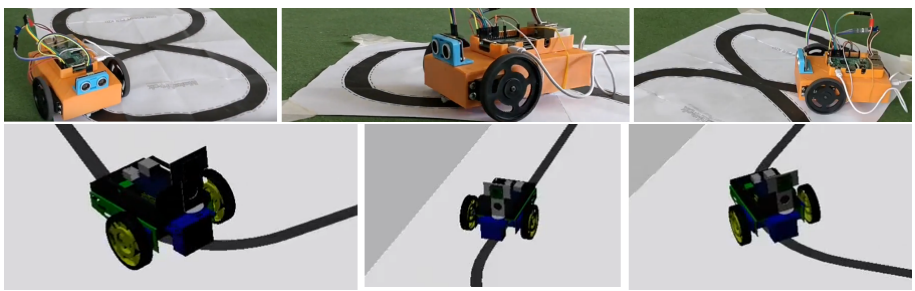

**Figure 14.** Practice line-tracking task in both real and simulated `PiBot` platforms using IR sensor.

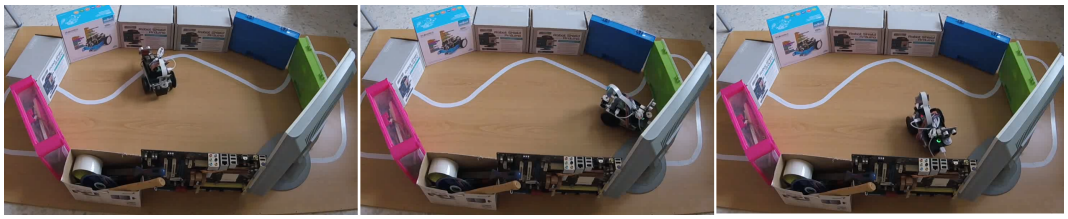

**Figure 15.** Navigation practice avoiding obstacles through US in `PiBot`.

### *6.3. Vision-Based Behaviors*

In this group of exercises, the students can solve the previously described exercises but now using vision as the main sensor. Some projects developed are: following a colored object (Figure 16), line tracking or bump-and-go.

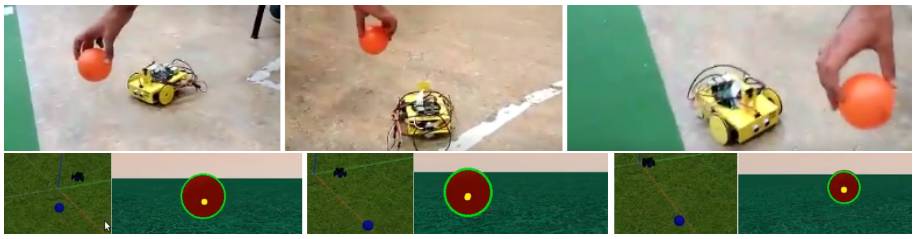

**Figure 16.** Navigation exercise of following a colored object using vision in both real and simulated `PiBot` platforms.

In July 2018, a Robotics workshop was taught to ten teachers at the Campus of Fuenlabrada of the Rey Juan Carlos University (Madrid) (Figure 17), training them to use the developed framework with `PiBot` as a robotic platform.


**Figure 17.** Workshop at Rey Juan Carlos University to train teachers for teaching with JdeRobot-Kids framework using `PiBot`.

## 7. Conclusions

This research focused on incorporating robotics and robots with vision in the classroom to train pre-university students, satisfying the demands imposed by society in the Digital Age and the

motivational needs detected in students, who study in a system not currently adapted to the so-called Industrial Revolution 4.0.

Although there are numerous educational robotics kits on the market, most are aimed at very young students. Many robotics kits are based on building robotic platforms with their own programming environments, and do not employ standardized programming languages. A standard language across several robotic platforms is Scratch 3.0, but it is also oriented to elementary level. In addition, typical robotics kits usually have somewhat limited capabilities, which means they tend to generate little mid-term motivation in students (for instance, in students who have already followed an introductory robotics course). Furthermore, given the complexity involved in the processing of images, cameras are not usually included in the educational robotic frameworks despite their great versatility and extensive use in real life applications.

After studying the current market in existing robotics educational kits and conducting an in-depth analysis of what the near to mid-term future holds in terms of the demands of the labor market, the authors (one of whom is an experienced Secondary Education teacher) detected a gap between the level of academic training at university level in scientific and technological degrees and the official curriculum implemented at pre-university levels, specifically in science subjects at secondary education level. Therefore, a complete new educational tool was developed, which includes:

- A robotic platform, the `PiBot`, based on the free hardware controller board Raspberry Pi 3. This board is mounted over a chassis designed as a 3D printable model, which lets anyone, not only schools, build their own robots at a low cost, following the Do It Yourself (DIY) philosophy. The Raspberry was chosen as the `PiBot` processor for several reasons: firstly, the inclusion of a camera with its own data bus, the PiCamera, which allows the teaching of Artificial Vision algorithms; secondly, it maintains a low cost robotic platform but with high computational power; thirdly, the inclusion of the GPIO ports on the board, thanks to which various sensors and actuators were connected; and, finally, it is a standardized versatile board.
- A simulated robot for `PiBot` under Gazebo simulator. Thus, students have the possibility of using both a real and a simulated robot. This provides a valuable learning process through which students can appreciate the differences between the real and the simulated world.
- A software infrastructure developed in Python language, which includes all the appropriate drivers. This facilitated students' programming of the robot, with simple and intuitive functions to handle the different sensors and actuators. At the same time, this infrastructure has great potential due to its handling of a camera as a sensor.
- A wide-ranging set of exercises that serve as a support to students for their progression in learning to program robots with vision. The real or simulated robot is programmed in a simple and powerful language, Python, which is not widely used in educational robotics due to the limitations of the processors managed to date.

Regarding future lines of research, one intended improvement in the short term is to extend the vision support: (a) developing new practical sessions with vision such as the detection and monitoring of people's faces, and materializing a visual memory in the `PiBot`; (b) seating the camera on a servo so the current vision range can be extended to a wider field of view, thanks to the movement of the camera.

It is also intended to develop the support for the encoders of the `PiBot` motors, which should allow more position-based sophisticated navigation to be developed.

In addition, in the 2018–2019 academic year, although use of this new tool has already been successful under a pilot plan, with 25 students (15 using the simulated platform from Ceuta, and 10 using the real robot in a workshop), the impact on the educational effectiveness of this new tool will be measured by means of an experiment with actual students and a control group in several schools.

Finally, the authors are also working to support `PiBot` programming with the popular visual Scratch language, so that younger students can start simple programming of this robot. With the same

PiBot platform, they can start learning robotics with Scratch and subsequently move up to Python and engage in more motivating exercises.

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

**Funding:** This work was partially funded by the Community of Madrid through the RoboCity2030-III project (S2013/MIT-2748) and by the Spanish Ministry of Economy and Competitiveness through the RETOGAR project (TIN2016-76515-R). The APC was funded by the RoboCity2030-III project (S2013/MIT-2748).

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

## References

1. Schwab, K. *The Fourth Industrial Revolution*; World Economic Forum: Cologny, Switzerland, 2016.
2. Vega, J.; Cañas, J. Sistema de atención visual para la interacción persona-robot. In Proceedigns of the Workshop on Interacción Persona-Robot, Robocity 2030: UNED, Madrid, Spain, 29 September 2009; pp. 91–110, ISBN: 978-84-692-5987-0.
3. Arbel, T.; Ferrie, F. Entropy-based gaze planning. *Image Vis. Comput.* **2001**, *19*, 779–786. [CrossRef]
4. Solove, D. *The Digital Person: Technology and Privacy in the Information Age*; NYU Press: New York, NY, USA, 2004.
5. Frey, C.; Osborne, M. *The Future of Employment: How Susceptible Are Jobs to Computerisation*; University of Oxford: Oxford, UK, 2013.
6. Deloitte. *From Brawn to Brains: The Impact of Technology on Jobs in the UK*; Deloitte LLP: London, UK, 2015.
7. Institute, M. *Jobs Lost, Jobs Gained: Workforce Transitions in a Time of Automation*; McKinsey Global Institute: New York, NY, USA, 2017.
8. UK-RAS White Papers. *Manufacturing Robotics: The Next Robotic Industrial Revolution*; UK-RAS: London, UK, 2016.
9. Rubinacci, F.; Ponticorvo, M.; Passariello, R.; Miglino, O. Robotics for soft skills training. *Res. Educ. Media* **2017**. [CrossRef]
10. Rubinacci, F.; Ponticorvo, M.; Passariello, R.; Miglino, O. Breeding Robots to Learn How to Rule Complex Systems. *Robot. Educ.* **2017**._13. [CrossRef]
11. Rodger, S.H.; Walker, E.L. Activities to attract high school girls to computer science. In Proceedings of the twenty-seventh SIGCSE technical symposium on Computer science education, Philadelphia, PA, USA, 15–17 February 1996.
12. Altin, H.; Pedaste, M. Learning approaches to applying robotics in science education. *J. Balt. Sci. Educ.* **2013**, *12*, 365–377.
13. Mubin, O.; Stevens, C.J.; Shahid, S. A review of the applicability of robots in Education. *Technol. Educ. Learn.* **2013**. [CrossRef]
14. 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, doi:10.1016/j.riai.2015.04.005. [CrossRef]
15. 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]
16. Japan-Economic. *New Robot Strategy*; The Headquarters for Japan's Economic Revitalization: Tokyo, Japan, 2015.
17. 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.
18. Merkouris, A.; Chorianopoulos, K.; Kameas, A. Teaching programming in secondary education through embodied computing platforms: Robotics and wearables. *ACM Trans. Comput. Educ.* **2017**, *17*, 9. [CrossRef]
19. Kubilinskiene, S.; Zilinskiene, I.; Dagiene, V.; Sinkevièius, V. Applying Robotics in School Education: A Systematic Review. *Balt. J. Mod. Comput.* **2017**, *5*, 50. [CrossRef]
20. Eguchi, A. RoboCupJunior for promoting STEM education, 21st century skills, and technological advancement through robotics competition. *Robot. Auton. Syst.* **2016**, *75*, 692–699. [CrossRef]

21. 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 2016 XIII Latin American Robotics Symposium and IV Brazilian Robotics Symposium (LARS/SBR), Recife, Brazil, 8–12 October 2016; pp. 121–126.
22. Kandlhofer, M.; Steinbauer, G. Evaluating the impact of robotics in education on pupils' skills and attitudes. In Proceedings of the 4th International Workshop Teaching Robotics, and 5th International Conference Robotics in Education, Padova, Italy, 18 July 2014; pp. 101–109.
23. 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; pp. 72–73.
24. 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.
25. 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]
26. Jamieson, P. *Arduino for Teaching Embedded Systems. Are Computer Scientists and Engineering Educators Missing the Boat*; Miami University: Oxford, OH, USA, 2012.
27. 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 2016 IEEE Eighth International Conference on Technology for Education (T4E), Mumbai, India, 2–4 December 2016; pp. 38–41.
28. Filippov, S.; Ten, N.; Shirokolobov, I.; Fradkov, A. Teaching robotics in secondary school. *IFAC-PapersOnLine* **2017**, *50*, 12155–12160. [CrossRef]
29. 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.* **2013**, *3*, 1–7.
30. 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 2016), Seville, Spain, 22–24 June 2016; pp. 1–8.
31. Afari, E.; Khine, M. Robotics as an educational tool: Impact of LEGO mindstorms. *IJIET* **2017**, *7*, 437–442. [CrossRef]
32. 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), Umhlanga, South Africa, 8–10 March 2017; pp. 1–7.
33. 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]
34. 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 RIE 2015, Yverdon-les-Bains, Switzerland, 20–22 May 2015.
35. Stone, A.; Farkhatdinov, I. Robotics Education for Children at Secondary School Level and Above. In Proceedigns of the Conference Towards Autonomous Robotic Systems, Guildford, UK, 19–21 July 2017; pp. 576–585.
36. 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.* **2017**, *18*, 4. [CrossRef]
37. Plaza, P.; Sancristobal, E.; Carro, G.; Castro, M.; Blázquez, M.; Muñoz, J.; Álvarez, M. Scratch as Educational Tool to Introduce Robotics. In Proceedigns of the International Conference on Interactive Collaborative Learning, Budapest, Hungary, 27–29 September 2017; pp. 3–14.
38. 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 Proceedigns of the 2017 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.
39. Manzoor, S.; Islam, R.U.; Khalid, A.; Samad, A.; Iqbal, J. An open-source multi-DOF articulated robotic educational platform for autonomous object manipulation. *Robot. Comput.-Integr. Manuf.* **2014**, *30*, 351–362. [CrossRef]

40. Alers, S.; Hu, J. AdMoVeo: A Robotic Platform for Teaching Creative Programming to Designers. In *Learning by Playing. Game-based Education System Design and Development*; Springer: Berlin/Heidelberg, Germany, 2009; pp. 410–421.
41. Dademo, N.; Lik, W.; Ho, W.; Drummond, T. eBug—An Open Robotics Platform for Teaching and Research. In Proceedings of Australasian Conference on Robotics and Automation, Melbourne, Australia, 7–9 December 2011.
42. Gonzalez, J.; Valero, A.; Prieto, A.; Abderrahim, M. A New Open Source 3D-printable Mobile Robotic Platform for Education. In *Advances in Autonomous Mini Robots*; Springer: Berlin/Heidelberg, Germmany, 2012.
43. Schweikardt, E.; Gross, M.D. roBlocks: A Robotic Construction Kit for Mathematics and Science Education. In Proceedings of the 8th International Conference on Multimodal Interfaces, ICMI '06, Banff, AB, Canada, 2–4 November 2006.
44. Bers, M.U.; Ponte, I.; Juelich, C.; Viera, A.; Schenker, J. Teachers as Designers: Integrating Robotics in Early Childhood Education. *Inf. Technol. Child. Educ. Annu.* **2002**, *2002*, 123–145.
45. Benitti, F. Exploring the educational potential of robotics in schools: A systematic review. *Comput. Educ.* **2012**, *58*, 978–988. [CrossRef]
46. Ainley, J.; Enger, L.; Searle, D. Students in a Digital Age: Implications of ICT for Teaching and Learning. In *International Handbook of Information Technology in Primary and Secondary Education*; Voogt, J., Knezek, G., Eds.; Springer: Boston, MA, USA, 2008; pp. 63–80.
47. Papadakis, S.; Kalogiannakis, M.; Orfanakis, V.; Zaranis, N. Novice Programming Environments. Scratch and App Inventor: A first comparison. In Proceedings of the 2014 Workshop on Interaction Design in Educational Environments, Albacete, Spain, 9 June 2014; pp. 1–7.
48. Balachandran, S. *General Purpose Input Output (GPIO)*; Technical Report ECE 480 Design Team 3; Available on the College of Engineering, Michigan State University Website. Available online: https://www.egr.msu.edu/classes/ece480/capstone/fall09/group03/AN_balachandran.pdf (accessed on 10 August 2018).