The Robobo Project: Bringing Educational Robotics Closer to Real-World Applications

F. Bellas[1], M. Naya[1], G. Varela[2], L. Llamas[2], A. Prieto[1], J.C. Becerra[3], M. Bautista[1], A. Faiña[4], R.J. Duro[1]

[1]Integrated Group for Engineering Research, Universidade da Coruña, Spain
{francisco.bellas, martin.naya, abprieto, m.bautista, richard}@udc.es
[2]Mytech, A Coruña, Spain
gervasio.varela@mytechia.com, luis.llamas@mythechia.com
[3]MINT, A Coruña, Spain
juancarlos@mintforpeople.com
[4]IT University of Copenhagen, Denmark
anfv@itu.dk

Abstract. The Robobo Project is a STEM-based project that aims to bring educational robotics, in primary and high school, closer to real-world applications. It is based on the use of a smartphone-based robotic platform called Robobo, a very flexible programming environment, and a set of lessons to integrate them. The smartphone provides high-level hardware capabilities in terms of sensors, communications and processing capabilities that allow to create more practical and realistic lessons that exploit human-robot interaction, with a small investment. In this paper, we present the main elements of The Robobo Project in terms of hardware and software, and two illustrative educational projects that can be developed within it.

Keywords: Interactive Education, Smartphones, Computer Vision, Speech Recognition, Real-World Robotics

1 Introduction

Robotics is a key subject in STEM education [1][2], mainly due to its multi-disciplinarity and practical perspective. Thus, robotic projects involve mechanical design, electronics and programming skills which, in turn, require a background in mathematics, physics and other technological sciences. In addition, all of these topics promote the students being involved in projects with a practical objective as robots are made to solve problems in the real world. As a consequence, knowledge acquisition is much more effective due to the attention it captures [3].

Table 1 shows a summary of the main robotic platforms used in STEM education. As it can be observed, most of them have simple sensors (ultrasonic or infrared), basic communications capabilities (Bluetooth mostly) and low computing power. The main reason behind this is economical, as introducing a set of robots in a classroom is expensive. But there are other reasons like the possibility of having configurable mechanical

kits, made up of simple elements, so the students can change the robot configuration. This feature in a robotic system implies that the hardware elements must also be simple, because they can be easily damaged and repairing them should not be complex.

With such basic hardware, the projects that students can carry out are limited to a narrow range of possible applications. This was not a problem a few years ago, but nowadays and in the near future, as students acquire technological skills earlier, these simple robotic platforms will be a valid STEM tool only for primary level courses. Moreover, due to the limitations imposed by the hardware capabilities, most of the robotic lessons in elementary and high schools have traditionally been quite far from reality, that is, from being implemented in a real robot to solve a practical task [4]. For instance, a typical lesson in basic robotic courses implies developing a line-follower [5]. This is a very interesting lesson to learn basic programming skills and classical control, but students perceive robots around them every day without requiring artificial modifications of the environment to operate [6][7], so this kind of lessons seem artificial to them. And, finally, simple robotic platforms are far from being up-to-date with regards to current hardware advances, so when students are using them, they perceive that the real technology is clearly way ahead. Consequently educational centers must invest periodically to renew their robotic platforms, defeating the purpose of low cost systems.

	NAO	**LEGO EV3**	**DASH & DOT**	**MBOT RANGER**	**THYMIO II**
CPU	Atom Z530 1.6 Ghz Cpu	Arm926ej-S Core@300 Mhz	Arm Cortex-M0	Arduino Mega 2560	PIC24 32 MHz
Distance sensor	Sonar	Sonar / IR	IR	Sonar	IR
Sound	Speaker/Microphone	Speaker	Speaker /Microphone	Buzzer/Microphone	Speaker /Microphone
Camera	1280x960	No	No	No	No
Speech recognition	Yes	No	No	No	No
Communications	WIFI / USB	Bluetooth / USB	Bluetooth	Bluetooth /USB	WIFI / USB
Average Prize	7.000 €	350 €	230 €	170 €	130 €

In this sense, a few years ago we started the development of an educational robotics project called “The Robobo Project” [8], which aims to bring robotics closer to practical implementations for elementary and high schools. The Robobo Project is based on a hardware platform called Robobo, displayed in Fig. 1, and a set of STEM lessons supported on this platform, which are realized through a programming environment. The Robobo hardware is made up of a wheeled base that transports a smartphone, which provides Robobo with high level sensing, communications and processing capabilities, and that controls the base actuators. The smartphone allows the development of lessons using more complex sensing modalities, like computer vision or speech recognition,

together with high processing capabilities to execute them on-board. This permits proposing projects closer to the real applications that robotics demands nowadays, with a high degree of human-robot interaction (HRI) and using more realistic sensors, so students start to deal with them early on. On the other hand, the cost of Robobo is reduced, as compared to robots with similar hardware capabilities, as it can be observed in Table 1, where the only platform that supports computer vision or speech recognition is the most expensive one. In this sense, two more considerations must be taken into account. First, Robobo is a high-value product with a low-cost for the educational centers, because the main cost is in the smartphone, which can be provided by the student. Second, the lifespan is much longer than typical robotic platforms because it can be updated just by replacing the smartphone with a better one, and, as the phone in use is the student's, there is no cost for the school. Finally, as the student is using her own phone, she can work on the projects any time. As it is well-known, the smartphone market is highly competitive, so the new models are continuously updating sensors and features and they are always at the edge of technology. This implies that, when new sensing modalities appear, the Robobo will be able to handle immediately.

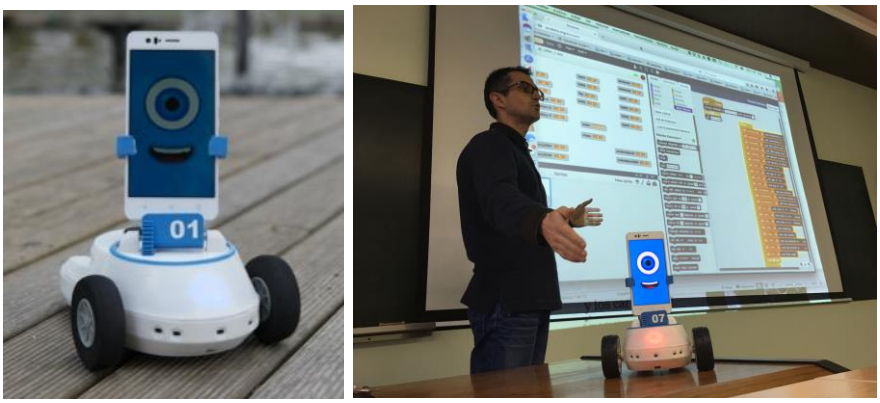


Fig. 1. Robobo hardware (left) and its use in a class (right)

In this paper, we present The Robobo Project's main features, that is, a description of the wheeled platform in section 2, the programming environment in section 3 and two illustrative educational projects supported by them in section 4. Finally, section 5 is devoted to some general remarks and conclusions.

2 Robobo Hardware

The Robobo robot is made up of two main parts: a mobile platform, called the ROB, that includes a series of low level sensors, the mechanical and electronic components for all the actuators, and a communications channel to connect to a smartphone, and the smartphone itself, the OBO.

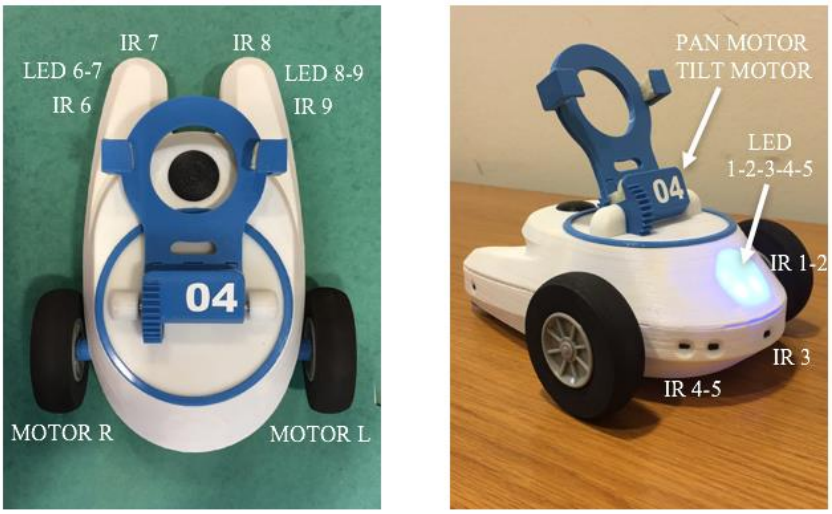


Fig. 2. The Robobo platform and its main sensors and actuators

Mechanically, the ROB part is divided into two elements that can be observed in Fig. 2: the main platform and the pan-tilt unit that supports the smartphone. The main platform contains the circuit boards, infrared sensors, batteries, docks for add-ons and the pan-tilt unit. The platform is driven by two wheels which are powered by two geared CC motors that allow the Robobo to achieve a velocity of 1m/s with 40Ncm of torque. This is quite enough for snappy performance and climbing small ramps. The two wheels can move independently either backwards or forward, allowing the robot to turn.

There is a pan-tilt unit to hold the smartphone. This is a key feature in its HRI capabilities, as it allows the smartphone to perform actions such as nodding or shaking its "head". When the pan-tilt motions are combined with images displayed on the smartphone screen, the sounds it produces, the displacement of the main platform and the different colored LEDs, the robot can express and transmit emotions, feelings, etc in a way that is very natural and understandable for humans. Thus, it provides a mechanism for the robot to actively interact with children, influencing their mood, predisposition to learn, or even making them more sociable. Obviously, in addition to the interaction possibilities, the pant-tilt support allows for almost omnidirectional semi-spherical vision all around the robot from the smartphone cameras, providing a lot of information about the environment in which the robot operates.

The ROB platform electronics are structured around a PIC32 microcontroller for real time operation and responsiveness. It is in charge of controlling the ROB's sensors, actuators and routine programs as well as the communications with the OBO. 9 RGB LEDs, with a color resolution of 12bits per LED, for a total of 212 = 4096 different colors on each RGB LED, are arranged around the ROB platform (see Fig. 2). These LEDs can be used to provide any type of information to the user. For instance, by default they are associated to each one of the infrared sensors, allowing the user to assert what the sensor is sensing in a very simple and intuitive manner. They are in charge of obstacle detection and fall avoidance as well as ambient light sensing. Regarding obstacle detection, 7 of the 9 infrared sensors are placed around the ROB to sense the

distance to the different objects, providing obstacle range detection up to 10-15 cm depending on the color of the obstacle, and thus, Robobo is able to carry out collision avoidance tasks when required.

Two frontal sensors are dedicated exclusively to the detection of "holes" (in the sense of lack of floor). These sensors are tilted towards the ground at a 70° angle to optimize hole detection and its range, so that the robot has time to stop before falling. Two of the obstacle detection sensors in the back of the robot share the fall avoidance functionality. It is important to note that, even though the sensors are configured to work as infrared sensors, they can be reconfigured as ambient light sensors if the user so desires. In this configuration, the sensor returns the light intensity value.

The ROB platform communicates to the OBO part of the system by means of a Bluetooth module together with a series of firmware programs whose function is to abstract the low-level control into an API that the OBO can use. That is, the software developed on the smartphone just sees this abstraction of the ROBs functions. The choice of a wireless communications protocol over a wired one, such as USB, had to do with increasing the versatility and reducing the complexity of the platform due to the diversity of connectors and connector positions in different smartphones. Thus, the ROB platform can be used with any smartphone just as long as it has Bluetooth connectivity. As a summary, Table 2 shows the sensors, actuators and communication capabilities of the Robobo robot, both those provided by the ROB and those of the OBO. It must be pointed out that, although the processor of the ROB has low computing capabilities, the one in the smartphone is, in general, much more powerful than any other robot in educational robotics if a mid-range smartphone is considered.

	Platform	**Smartphone**
Sensors	✓ 9 IR sensors (front, back and floor) ✓ 4 Odometric sensors (motor encoders)	✓ Vision: front and back high-resolution cameras ✓ Ambient: proximity, light, temperature ✓ IMU: gyroscopes, accelerometers, magnetometers ✓ GPS ✓ Sound ✓ Touch on screen
Communications	✓ Bluetooth	✓ 3G-4G ✓ WI-FI ✓ USB
Actuation	✓ Two motors on the wheels ✓ One motor for PAN ✓ One motor for TILT ✓ 9 LED lights	✓ High-resolution LCD Screen ✓ Speaker

3 Robobo software

One of the main goals of the Robobo robot is to serve as an educational and entertainment tool for teaching programming, robotics, and STEM disciplines across a wide

range of ages. This goal has been the main idea driving the design of the software architecture of the robot, and the result is a completely modular architecture that:

- On the one hand, allows the easy extension of the robot with new features;
- And, on the other hand, allows users to program the robot using very different approaches that suit a variety of educational levels, like block programming, Java language or the Robot Operating System (ROS) [9].

In this section, we provide a detailed description of the Robobo software architecture and the different programming paradigms supported by it.

3.1 Software Architecture

As previously introduced, the main requirement of the Robobo software architecture was to support the programming of the robot using very different programming paradigms. The goal was to be able to use the robot for teaching STEM and robotics in very different levels of education, from primary schools to universities, through secondary schools or even for research purposes. With this idea in mind, we have designed the completely modular software architecture shown in Fig. 3. It is based around the concept of Robobo module and a very lightweight library, called the Robobo Framework, which provides the essential mechanisms to manage and run those modules on an Android smartphone. On top of this library, two different kinds of modules can be loaded:

- Functionality modules (the orange ones in Fig. 3): implemented in Java using the Android standard API, they provide different functionalities to the robot like speech recognition, face detection, environment sensing or physical movement. Furthermore, these modules complement the native API of the robot, which can be directly used for programming it using Java for Android, and thus creating Robobo Apps that can be run in any Android smartphone.
- A series of proxy modules (ros-proxy and remote-proxy modules in Fig. 3), also implemented in Java, which provide particular interfaces to other programming environments, like ROS, Scratch or the native Robobo Educational IDE using Blockly. These interfaces provide a translation between an external API or protocol, and the native Robobo API in Java.

Robobo comes by default with a particular set of these modules but, as they are completely decoupled between them and the framework, advanced users can customize the set of modules, and even implement new modules to support new robot functionalities, or even new programming paradigms through proxy modules.

Finally, it is important to note that there exists a module for connecting the Robobo framework and its modules to the Robobo robotic base (ROB). The rob-interface module, shown in pink in Fig. 3, implements the Bluetooth communications protocol of the ROB and provides a control API for other modules to use. This module is not special at all, it is implemented and loaded as any other module of the framework, but we have decided to present it in a different color because it is an essential module to control the physical part of the robot, the ROB platform.

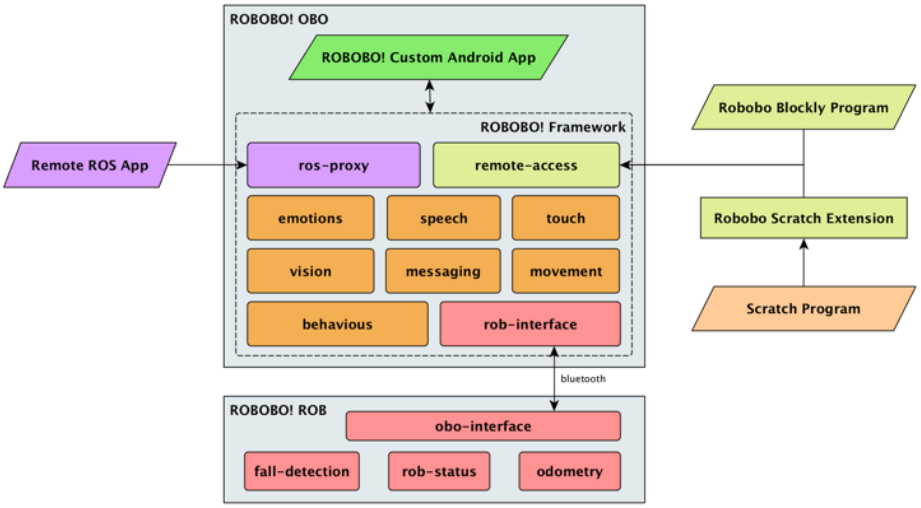


Fig. 3. Block diagram showing the Robobo software architecture and the different programming paradigms supported by it

3.2 Robot Programming

As briefly introduced in the previous subsection, the robot can be programmed using many different programming languages or paradigms and, thanks to the modular design of our solution, it can even be expanded to support new paradigms. Currently, Robobo supports four different programming languages or paradigms:

- Java programming for creating Android Apps.
- Scratch programming with ScratchX [10].
- Block programming using our own IDE based on Blockly [11].
- ROS programming for universities and scientific usages.

Java programming is directly supported by the native API provided by the different modules. Using the Robobo framework users can create custom Android Apps that control the behavior of the robot. These apps use the programming interfaces of the Robobo modules to access the different functionalities of the robot and build new robot behaviors or solve new tasks.

For block programming, we currently support two different approaches. Scratch, and our own block-based IDE that uses Blockly. As can be seen in Fig. 3, both approaches are connected to the framework using the same proxy, the Robobo Remote Access Protocol (RRAP). This is a simple JSON based protocol that allows remote access to the Robobo modules' APIs.

Scratch is supported by a Scratch extension, implemented in Javascript, that translates between Scratch blocks and operations of the Robobo modules exported through the RRAP. Our Blockly IDE follows a similar approach, where blocks are directly translated to Javascript code that uses the RRAP to control the robot remotely from the user's browser.

Finally, Robobo can also be programmed using the Robot Operating System (ROS), commonly used by the scientific community and for teaching robotics at many universities. ROS is supported by a ros-proxy module than translates between the native APIs of the modules and ROS messages and topics.

4 Robobo Interactive Lessons

To illustrate the type of interactive lessons that can be developed within The Robobo Project and the potentiality of using more advanced hardware in educational robotics, in this section we are going to present some examples of learning projects that have been designed, and that use the Robobo hardware and software commented in the previous two sections. These lessons are adequate for high-school students that have some basic notions of block-based programming, and they can be implemented with the ScratchX blocks already implemented in the Robobo software.

All the classical robotic lessons that can be developed using the typical sensors present in many other robotic platforms like infrared, IMU, simple sounds, ambient light or ground color, are directly applicable to Robobo. Consequently, we are going to focus our attention in the sensorial modalities that are rarely used in STEM robotics: computer vision, advanced sound processing, speech recognition and touch interaction.

All of these lessons start from the following didactical premises:

1. The students use their *own smartphone* in classes: this is a motivational aspect because they exploit a familiar element that, in addition, can also be used at home.
2. The Robobo can be *programmed* using a *tablet or a PC*: depending on their programming skills, preferences or educational center policy
3. The Robobo software simply requires that the *Robobo* and the *smartphone* are on the *same WI-FI* network (a dedicated one can be created in the classroom)
4. All the Robobo lessons can be *applied in the real world*

4.1 Simon Says Musical

In this first example, the project that students must solve is creating an interactive game consisting on a musical version of the classical Simon Says game using ScratchX blocks. Two elements are required in this case, a Robobo and a piano (see Fig 4a for an example of configuration). They must be close and in a low noise environment. Specifically, the students must develop a game with the following steps:

1. The robot operation starts with a clap that "wakes up" Robobo, which says "Ready"
2. With a second clap the game starts, and the robot tries to challenge the user saying something like "follow me if you can"
3. The robot produces a single musical note
4. The facial expression in the LCD screen changes to one that seems to pay attention to the user (Fig 4b shows a possible expression) and the Robobo starts recording the musical notes the user creates

5. The original note produced by Robobo must be compared with the one sensed by it:
   5.1. If they are the same, the robot changes its expression to a happy one (Fig 4c shows an example), the pan-tilt unit moves up and down (like saying YES with the head) and it emits a congratulating sound. The program returns to step 3 but now playing one more note.
   5.2. If they are different, the expression changes to a sad one (Fig 4d), the pan-tilt unit moves from side to side (like saying NO with the head) and a failure sound is emitted. The program returns to step 1.

To solve the lesson, students must apply the following ScratchX blocks: clapDetector, speechProducer, noteProducer, noteDetector, soundProducer, emotionProducer and motorCommand, as well as other ScratchX native blocks to create the algorithm. The configuration of the blocks to achieve a fluid and reliable interactive game is part of the students work because, as highlighted in the introduction, the goal is to solve problems that can be used in the real-world.

As it can be observed, this lesson does not require any movement of the robotic platform, so one could think that a robot is not necessary in this case. But the approach behind The Robobo Project is that the human-robot interaction is more than just movement, and developing this type of static interaction is also very important. Moreover, the pan-tilt unit must be moved in this lesson so, from a didactical point of view, the students must use the Robobo motor commands.

(a) (b) (c) (d)

Fig. 4. a) The Robobo and the piano b) Waiting for the user notes c) User success d) User failure

4.2 Robobo Pet

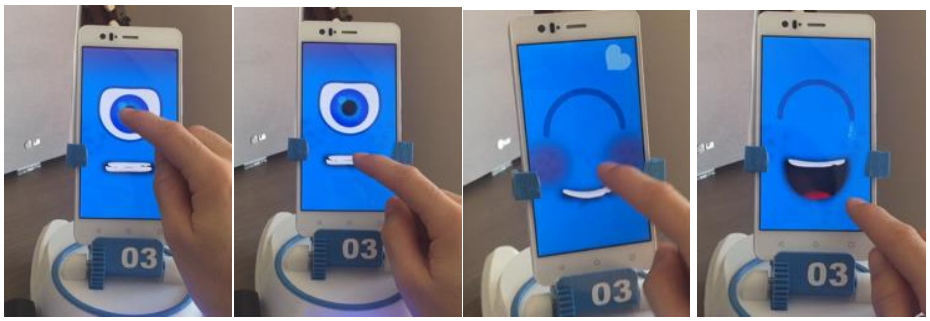
The second example project is focused on programming the Robobo to act as a pet. That is, the robot will ask the user for food, attention and “affection”. The modules required to solve this project use all the interactive modalities of Robobo, that is, computer vision, speech recognition, sound production and touch. In this case, the specifications provided to the students start with the premise that only a Robobo and a user are required in a controlled environment, in terms of other elements that can interfere

between them. Moreover, the program that controls the robot does not follow a sequential scheme in this case, but an event-based execution that is fired by the user with its interaction. The pet behavior must work as follows:

- *Basic instincts*: Robobo must store two variables that represent thirst and hunger levels. They start at a predefined value and they decrease proportionally to the robot activity (basically, motor movements). If these levels are lower than a predefined threshold, the robot must say "I am hungry" or "I am thirsty" until they are refilled as explained below.
- *Movement*: Robobo must move in order to capture user attention if no event has been executed for a predefined time-lapse. For instance, it can spin or emit a sound.
- *Touch*: two different behaviors must be implemented depending on the type of screen touch modality:
  - *Tap*: if the user touches the screen with a single tap, Robobo must react differently depending on the part of the "face" that is touched. If it is in the "eye" or in the "mouth", it will move the tilt motor backwards and show an angry expression (see Fig. 5a and 5b for examples). If it is in any other point, it will show a laughing face and emit a sound saying "tickles" (see Fig. 5c and 5d for examples)
  - *Flip*: if the user slides a finger over the screen, the pan-tilt unit moves accordingly. The possible sllide directions must be discretized into four pan-tilt movements: tilt backwards or forwards and tilt rightwards or leftwards.
- *Voice*: Robobo must react to the following predefined phrases:
  - *Here comes the food:* the robot prepares to receive food
  - *Here comes the drink:* the robot prepares to receive drink
  - *Hello:* the robot answers by saying "Hello"
  - *How are you?:* the robot will respond by saying its thirst and hunger levels
- *Vision*:
  - *Color:* Robobo must detect 2 different colors, green for food and blue for drink, but only after the corresponding voice command has been detected (shown in the right image of Fig. 6). In both cases, it must say whether the color is correct or not. For instance, if the user says *Here comes the drink*, the robot must look for a blue color area of a predefined size in its field of view (the left image of Fig. 6 shows example), and after a time-lapse, it must say "Thank you for the drink" or "I don't see the drink".
  - *Face:* Robobo must detect the user face. If it is below a threshold, the robot will move backwards

To solve this second project, students must apply, and configure, the following ScratchX blocks: speechProducer, speechRecognition, faceRecognition, colorDetection, touchDetection, emotionProducer and motorCommand, as well as other ScratchX native blocks to create the algorithm. The main feature of this project is that it can become as complex as the student wants, because achieving a fluid response of the pet is not simple. Furthermore, its upgrading possibilities are huge, and this is very im-

portant to promote student creativity. Thus, the teacher could propose several improvements. For instance, the robot could dance following the rhythm of a musical piece, it could follow the user voice or face while he/she is moving, or it could serve as a movable alarm clock.



(a) (b) (c) (d)

Fig. 5. Possible expressions in the touching behavior

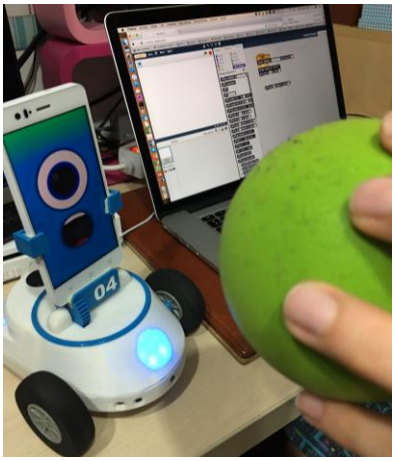


Fig. 6. The user showing a green ball that represents "food"

5 Conclusions

The Robobo Project aims to be a starting point in a new didactical methodology for educational robotics, which is based on more realistic projects to be solved by students. It is supported by a smartphone-based robotic platform, which exploits the high end technology that is included in current mobile phones without implying a large investment for the educational centers. Moreover, it uses a very flexible programming environment that allows students to program in different languages depending on their skills, so the Robobo lifespan is large. In addition to presenting the main features of

hardware and software architectures, two illustrative lessons that can be developed with Robobo have been described. As it can be observed in both examples, a more natural and realistic interaction can be obtained with the high-level sensing capabilities provided by the robot, allowing for more realistic applications to be developed by students.

Acknowledgements

This work has been partially funded by the EU's H2020 research and innovation programme under grant agreement No 640891 (DREAM project) and by the Xunta de Galicia and redTEIC network (ED341D R2016/012).

References

1. Alimisis, D., Moro, M. (2016). Special issue on educational robotics. Robotics and Autonomous Systems, Vol 77, pp 74-75
2. Barker, B. S., Nugent, G., Grandgenett, N., & Adamchuk, V. I. (2012). Robots in K-12 Education: A New Technology for Learning, IGI Global, pp 1-402
3. Kandlhofer, M., Steinbauer, G. (2016) Evaluating the impact of educational robotics on pupils' technical-and social-skills and science related attitudes. Robotics and Autonomous Systems. Vol 75, Part B, pp 679-685
4. Danahy, E., Wang, E., Brockman, J., Carberry, A., Shapiro, B., & Rogers, C. B. (2014). Lego-based robotics in higher education: 15 years of student creativity. International Journal of Advanced Robotic Systems, 11(2), 27
5. Khalid, A., Zahir, E. (2015) Optimizing the Turning Velocity in a Line Follower Robot. International Journal of Computer Applications, Vol 117-3
6. Jones, J.L. (2006) Robots at the tipping point: the road to iRobot Roomba, IEEE Robotics & Automation Magazine, vol. 13, no. 1, pp. 76-78
7. Pepper Robot web page. Aldebaran Robotics: https://www.ald.softbankrobotics.com/en/cool-robots/pepper
8. The Robobo Project web page: http://therobobоproject.com
9. ROS web page: http://www.ros.org
10. ScratchX web page: http://scratchx.org
11. Blocky web page: https://developers.google.com/blockly/