Overview of Technologies for Building Robots in the Classroom

Cesar Vandevelde
Dept. of Industrial System & Product design
Ghent University
Kortrijk, Belgium
cesar.vandevelde@howest.be

Jelle Saldien
Dept. of Industrial System & Product design
Ghent University
Kortrijk, Belgium
jelle.saldien@howest.be

Maria-Cristina Ciocci
Dept. of Industrial System & Product design
Ghent University
Kortrijk, Belgium
maria-cristina.ciocci@howest.be

Bram Vanderborght
Dept. of Mechanical Engineering
VUB
Brussels, Belgium
bram.vanderborght@vub.ac.be

Abstract—**This paper aims to give an overview of technologies that can be used to implement robotics within an educational context. We discuss complete robotics systems as well as projects that implement only certain elements of a robotics system, such as electronics, hardware, or software. We believe that Maker Movement and DIY trends offers many new opportunities for teaching and feel that they will become much more prominent in the future. Products and projects discussed in this paper are: Mindstorms, Vex, Arduino, Dwengo, Raspberry Pi, MakeBlock, OpenBeam, BitBeam, Scratch, Blockly and ArduBlock.**

Keywords—Do-It-Yourself; Education; Open Source; Open Hardware; Project Based Learning; Rapid Prototyping; Robotics; STEM

I. Introduction

The demand for engineers in Europe has almost tripled over the past six years, while the number of engineers who graduate from universities and colleges in that period decreased drastically. Different engineers associations, such as the VIK (Flemish Association of Engineers) assert that the drop in number of students is connected to an image problem of STEM-related knowledge domains (Science, Technology, Engineering and Mathematics) as perceived by pupils and students. Action plans focused on promoting access to the engineering profession are set up, following the highlights of the rapid developments in this field. In addition to this, new pedagogical approaches are also experimented with, e.g. [1-3].

Studies suggest that certain teaching strategies may foster the STEM participation and achievement and give some evidence that using scientific equipment and hands-on activities are related to higher science and mathematics achievement [4]. As a consequence project based education and problem-based learning are becoming the innovative approaches to engineering education and fundamental science education [5, 6]. Also, it seems that there is a precise relationship between new technologies and new pedagogical methods; even though this relationship is complex and not only instrumental, the simplest explanation is that new technologies are the instruments to realize new pedagogies. According to the model of micro worlds of Papert [7], there is a strong link between mental acquisition of knowledge and actual manipulation of the objects of knowledge. Simply put: one learns by doing. Nowadays, this pedagogical prospective meets a new frontier with the commercialization of programmable toys (e.g. Lego Mindstorms), the upcoming of affordable DIY electronics (e.g. Arduino), and the rise of FabLabs equipped with 3D-printers and laser cutters. These systems make it possible to design and construct real robots whose working is determined by a computer program. From the moment that robotics entered our houses and started influencing our everyday life it has become important for everyone to have at least a basic understanding of such technologies. To introduce robotics in schools, there is a methodological choice in the contraposition of top-down teaching and bottom-up learning.

Building robots is a popular project choice for the implementation of problem-based learning (PBL) in classrooms. The reason why it is such a popular choice can be explained by the multidisciplinary nature of the topic: robotics requires many different scientific, technical and technological skills, such as physics, electronics, mathematics and programming. It is an ideal subject because so many different courses can be linked to it [8]. Additionally, robots themselves capture the imagination of children and teenagers, providing inspiration and motivation [8].

The PBL approach in general and the use of robotics in education in particular have a number of other differences with more traditional ways of teaching. Whereas math problems typically have one, and only one correct answer, PBL emphasizes that most real world problems have many different solutions. With PBL, students learn to deal with these real world problems using creative problem solving, an important real-life skill. In addition to technical skills, the PBL approach also allows the students to learn important social skills, such as communication, leadership, planning and cooperation [9].

The relatively new spectrum of technologies that can be used to implement robotics within a standard educational curriculum can become very overwhelming for educators and their administrators who are trying to decide what is possible and what the costs will be to begin such programs. It is important that costs can be afforded without any special purpose grant. This does not mean that robotics is out of reach. Several options exist that can be leveraged to achieve very effective results.

There are two ways to implement robotics in an educational context, either by starting from an existing robotics kit or by building the entire robot from scratch. Building a robot from scratch is typically much more difficult, thus using a kit is a more popular choice, especially in projects involving younger students. These robot kits provide everything needed to make a functional robot, such as building elements, motors, sensors, instructions, a programmable microcontroller, and the software to program the robot. While this provides a great starting point, this solution is typically more expensive and less flexible than a fully customized robot.

In recent years, making a robot from scratch has become much easier, in part due to the many different products and platforms that implement some of the elements that are required to build a robot. They can be categorized into 3 large groups: software, electronics and hardware projects. Historically, software has been the easiest to share as Open Source because collaboration can be done easily over the internet, because development tools are readily available and because there is virtually no cost associated with copying or modifying software. Online platforms such as GitHub greatly facilitate this [10]. However, in recent years the electronics and hardware projects have taken a jump forward in a phenomenon sometimes referred to as the Maker Movement [11]. The Maker Movement is a trend that can be described as a high-tech extension of the DIY and Arts & Crafts subcultures. It is characterized by the use of CNC machines such as laser cutters, 3D printers and CNC mills for the development and replication of Open Source hardware. These CNC machines are often low-cost devices that were developed by open source communities [12, 13].

II. Challenges and Pitfalls

While PBL using robotics offers a promising alternative to the traditional teaching methods, implementing this on a large scale in education does pose several challenges. Mataric et al. [14] describe 5 big challenges:

- "Lack of teacher time"
- "Lack of teacher training"
- "Lack of age-suitable academic materials"
- "Lack of ready-for-use lesson materials"
- "Lack of a range of affordable robotics platforms"

Besides these challenges, we also detected gender issues in the context of robotics in education. Presently, robotics and other technological hobbies are usually associated with boys. Girls are often subtly discouraged and told to pursue other interests. As a result, women are underrepresented in STEM-related fields. Studies have shown that while girls will not always focus on the same aspects of building robots as boys, they show just as much interest in the topic [8]. Consequently, robotics - if properly approached - can serve as a way to increase the number of women in technical and technological fields [15].

III. Complete Robotics Kits

Historically, robotics in education is usually implemented using premade robotics kits that include everything needed to build a functional robot. While this is a great way to get up and running quickly, it does include several disadvantages.

- All-in-one kits are generally more expensive.
- It's hard to interface these kits with other components, such as standardized components, components made by third parties or off-the-shelf sensors and actuators.
- Finding or buying replacement parts is not always easy.
- Not all components are used or needed in a robot, so you are paying for components you do not need.

That being said, these systems are a great starting point as they provide a set of compatible building blocks and electronics, software for easy (graphical) programming, instructions and a community network.

A. Lego Mindstorms

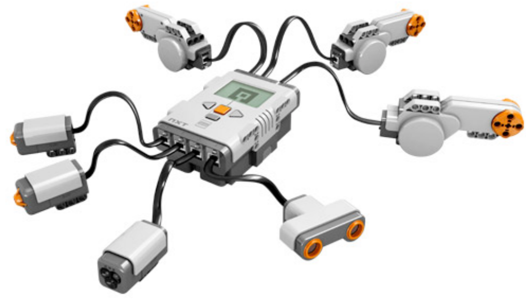

Fig. 1. Lego Mindstorms NXT [16].

Mindstorms [17, 18] is a product line by Lego that provides the necessary tools for creating simple robots using Lego bricks. Mindstorms is built around a programmable microcomputer brick that can control up to 3 motors and read up to 4 sensors. The motors and sensors can be hooked up to the control unit using snap connectors, so no special skills are required to assemble a functional set of robot electronics. The programmable brick itself can be programmed using a graphical programming language, named NXT-G, which is bundled with the sets. Alternatively, the brick can be programmed using one of the many available third-party applications, which provide support for languages such as C++ [19] or Java [20]. Lego Technic style bricks are used to build the structural parts of the robot. Because of this, building blocks from other Lego sets can be easily incorporated, expanding the potential level of intricacy of the robots.

The ease of use, the size of the Mindstorms community and the familiarity of Lego bricks make this a popular choice as a platform for robotics in education, especially when working with younger children. The popularity of Mindstorms translates to a plethora of resources available for educators, such as the many books, robot contests, online communities and workshops built around the Mindstorms ecosystem.

One of the main disadvantages of Lego Mindstorms is the cost, at a price of over €330 for the educational base set [21], providing enough robot sets for an entire class can quickly become an expensive affaire. Larger schools can alleviate this by buying enough sets for one group and then pass them around between the different class groups, but this is not always possible, especially for small schools or organizations. Another problem we've encountered frequently is that the programmable brick is limited to a maximum of 3 motors and 4 sensors. While this is typically enough to build a multitude of different robots, sooner or later students want to add another motor or sensor to their robot, only to discover they've run into the physical limit of what Mindstorms robots can do. A third point of criticism is that it's generally much harder to integrate third-party robotics components within a Mindstorms robot. Third-party sensors that can be interfaced with the Mindstorms programmable brick, such as those sold by MindSensors [22], do exist, but they typically rely on specialized circuitry for compatibility. Similarly, incorporating non-Lego hardware components into your Mindstorms robot is not always easy and typically involves modifying or otherwise damaging the Lego components.

B. Vex

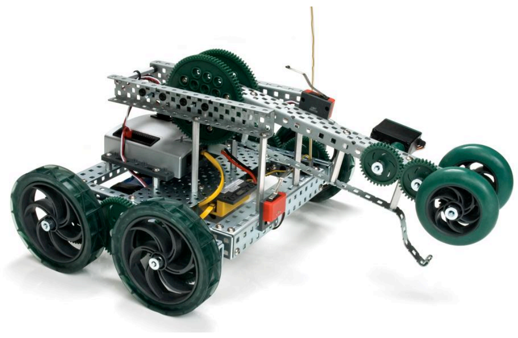

Fig. 2. Protobot Robot Kit [23].

The Vex Robotics system [24] is similar to Lego Mindstorms in that it is a platform that supplies all the necessary elements needed to build a functional robot: structural components, electronics, software and instructions. The main difference is that while Mindstorms tends to be more toy-like, suited for younger children, Vex instead opts for a more serious approach to robotics, targeting older students and adults. This is evident in the way the kit works; instead of plastic, perforated metal beams are used as structural elements and connections are made using nuts and screws, instead of snap and friction connections. Vex offers 2 different microcontroller options for use with its products, the PIC microcontroller and the Cortex microcontroller, both of which can be programmed using written code, as opposed to the graphical programming language used in Mindstorms. Vex allows for much choice in its product line, whereas Mindstorms aims to provide everything needed in one box, Vex allows for much more granular choice by splitting everything up in separate kits.

The Vex platform is more in line with "real" engineering practices than Lego Mindstorms and it offers a large degree of flexibility. This flexibility does make it more complex to use, meaning that this platform is better suited for older students and adults. At around €356 for a programmable starter kit [25], the cost of Vex seems similar to Mindstorms, however this not include programming software (which costs another €70) or the various expansion kits needed to complete a specific kind of robot. Mindstorms, in contrast, does include programming software and the components supplied in the base kit will typically be enough for many different types of robot projects. The Vex system is well documented, offering detailed instruction manuals, CAD models of the different parts, video tutorials and teacher materials. In our opinion, the main downsides of this platform are the high cost of the products and the degree of complexity, which makes it less suited for young children.

IV. Electronics

One of the core requirements for a functional robot is an electronics system that can read sensors, process information and control outputs. Many options of electronics are available when building a robot from scratch, making it possible to tune the electronics to the specific needs of your robot.

A. Arduino

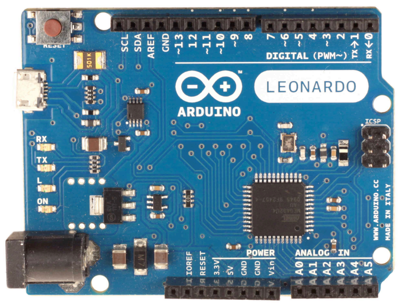

Fig. 3. Arduino Leonardo [26].

First developed at the Ivrea Institute for Interaction Design in 2005, the Arduino platform is the combination of a microcontroller board, a set of C++ libraries and an Integrated Development Environment (IDE) aimed at making microcontrollers accessible to artists, designers and hobbyists [27, 28]. The board itself is based around the AVR series of microcontrollers, made by Atmel. It provides all the necessary circuitry needed to make the microcontroller functional and breaks out the microcontroller's pins to easily accessible headers. On the software side, a multiplatform IDE bundles a code editor, a compiler, a linker and a programming utility into one preconfigured package. By preloading a boot loader onto the Arduino boards, it is possible to program the boards using a standard USB cable, without the need for an external hardware programmer. Writing code for Arduino boards is done in C++, but special libraries are provided which abstract the intricacies of programming microcontrollers into easy to use functions and classes. The combination of a low cost (€18 for an Arduino Leonardo board [29]) and the ease of use have made Arduino a very popular platform, especially among hobbyists.

The Arduino boards are not specifically designed for use in robotics, but this functionality can be added through the use of daughter boards, called shields. These shields can be attached on top of an Arduino board and provide the board with extra functionality, such as a display or circuitry to control DC motors directly. Many of the Arduino and Arduino compatible boards use the same physical pin layout; because of this the shield form factor has become a de facto standard. Many different shields exist, providing a plethora of possible functionality, made by either the Arduino organization or, more often, by third party vendors.

The availability of the large number of boards and shields, the ease of use, especially compared to other microcontrollers, and the low cost have all contributed to the creation of an Arduino ecosystem. Consequently, there is a large body of documentation and support available, in the form of books, tutorials and online communities. Arduino makes building a robot from scratch easier, in a low cost and flexible way. This low cost and flexibility does have an impact on the ease of use. While Arduino makes the use of microcontrollers easier, building robots using Arduino still requires a good working knowledge of both electronics and programming. For this reason, building robots with Arduino is significantly more complex than using an all-in-one solution, such as Mindstorms. Another problem that educators face when choosing Arduino is that the sheer amount of boards and shields can be overwhelming, so that the starting point is not always clear.

B. *Dwengo*

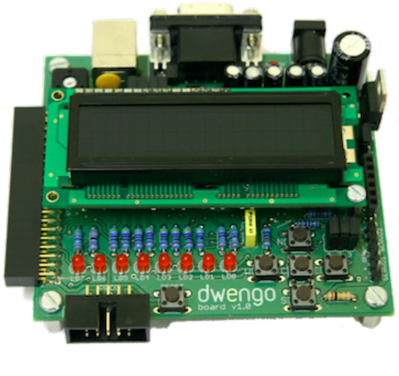

Fig. 4. Dwengo board [30].

The Dwengo project aims to provide an easy to use platform for getting started with microcontrollers, electronics and programming [31]. Originally started in 2009 at Ghent University as a microcontroller experimentation board for internal use, it was developed further when its potential benefits for education became clear. This board is built around the PIC series of microcontrollers, made by MicroChip. Unlike Arduino boards, which typically aim to provide a very low cost bare-bones board, the Dwengo board includes several features to facilitate building robots. The board includes a 16x2 character LCD, input buttons, 2 servo connectors and a 2 channel motor driver. In addition to the electronics board, the Dwengo project also provides a set of C libraries to facilitate programming, a set of tutorials which teach how to use board's features in a step-by-step manner and an experimental web application to program the board using a drag & drop interface. Like with the Arduino, building robots with Dwengo is definitely more complex than using a complete system such as Lego Mindstorms, but this in turn allows for more flexibility and gives the students better insight concerning how and why things work. While Dwengo has some large benefits over Arduino, such as the built in peripherals for making robots, Dwengo does not have the benefit the same large ecosystem.

C. *Raspberry Pi*

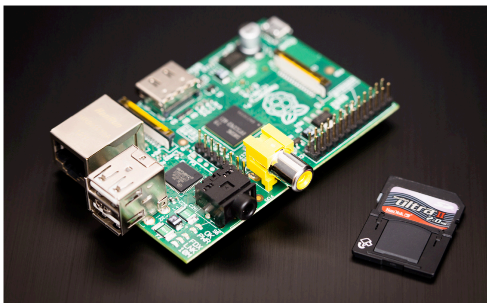

Fig. 5. Raspberry Pi [32].

The Raspberry Pi is a low cost single board computer, developed to promote the teaching of programming and computer science in education [33, 34]. The boards can be bought for $25 (model A) or $35 (model B) and provide the hardware required for a simple Linux system. In addition to 2 USB ports, an HDMI port, an SD card reader and an Ethernet port, the board also contains a pin header that gives access to low-level peripherals. The low cost combined with the access to these low-level peripherals make the board popular with hobbyists. One important thing to note is the difference between a microcontroller board, such as an Arduino, and a single-board computer, such as a Raspberry Pi. While a single-board computer is generally much more powerful, the overhead caused by the operating system (OS) makes it less suitable for applications that require precise timing, a task at which a dedicated microcontroller excels. On the other hand, the high-level nature of a Raspberry Pi makes it possible to program using high-level languages, to interface with more complex peripherals, such as webcams, and to connect to the internet.

The Raspberry Pi lacks many of the features that are required to build a robot, such as the ability to control DC motors. Luckily, this functionality can be added using daughter boards, much like Arduino shields. While the Raspberry Pi may not be able to provide the same level of real-time control as a dedicated microcontroller, it does offer many advantages compared to more traditional solutions. The board is designed to run Linux; this makes it possible to make a robot using more powerful programming languages. Another advantage is the ability to easily interface with USB devices, such as webcams and Wi-Fi dongles, which allows for advanced robots with internet connectivity and image recognition capabilities.

V. Hardware

Hobbyist robot makers tend to rely on a wide variety of techniques when it comes to building the physical embodiment of their robots. Some repurpose old toy components, some make their robot out of cardboard, glue and duct tape, some even build their own metal chassis using advanced CNC machines. A few projects aim to facilitate this process by providing a standardized building system.

A. *MakeBlock*

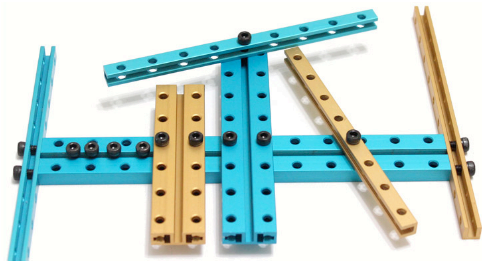

Fig. 6. MakeBlock [35].

MakeBlock [36] is a commercial building system specifically aimed at building robots. The MakeBlock system is built around several different types of aluminium beams which are joined together using standard machine screws. These screw connections can be made using either the beam's threaded slot, the grid of evenly spaced holes or the tapped holes at the end of a beam. In addition to these beams, several accessories are available, such as DC motor mounts, servo mounts, angle brackets and joining plates.

The MakeBlock system is a great way to make a very rigid robot and the threaded slot makes it easy to connect third-party components to the frame. The system uses the same hole spacing as Lego Technic bricks, further improving compatibility. In our opinion, the main downsides of MakeBlock are the relatively high costs and the limited availability.

B. *OpenBeam*

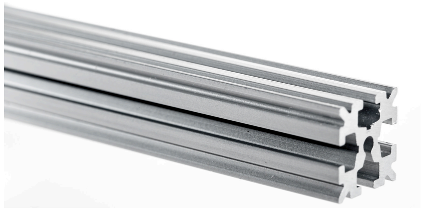

Fig. 7. OpenBeam [37].

OpenBeam [38] is a small scale version of the well known industrial T-slot aluminium profile systems. OpenBeam was conceived as an Open Hardware project and was realized though the Kickstarter crowdsourcing platform [39]. While regular T-slot profiles typically use their own proprietary nuts and bolts, OpenBeam was specifically designed to make use of standard M3 nuts and bolts, which are generally much cheaper and more readily available. These beams can be combined using angle brackets to build three-dimensional structures. Because OpenBeam uses standard hardware and because any arbitrary hole spacing can be used, it is easy to connect third-party components to the system. OpenBeam was not made specifically with robotics in mind, but because other components can be connected so easily, it does make it a viable way of building robots. Still, the OpenBeam system focuses on providing strong static connections, but offers much less in terms of building moving mechanisms.

C. *BitBeam*

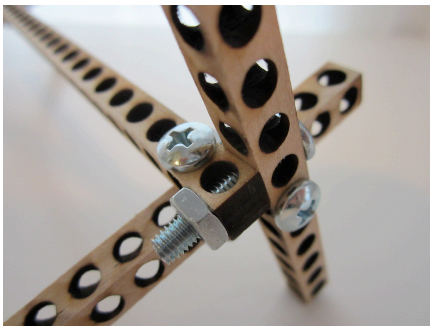

Fig. 8. BitBeam [40].

BitBeam [41] is a miniaturized version of GridBeam, a building system that uses 1.5 inch square beams with regularly spaced holes as a construction material to build large objects such as furniture [42, 43]. The specific geometry of the beams allows for a technique called a trijoint, shown in figure 8, in which 3 beams are joined in a corner using only 3 bolts and nuts. The BitBeam variant of the GridBeam system uses 8 mm square beams with holes spaced at 8 mm intervals, making it much more suited for building small scale robots. The 8 mm distance was not chosen arbitrarily, the hole spacing matches that of Lego Technic bricks, making integration with Lego bricks trivial.

The big advantage of BitBeam over other systems is its low cost and the fact that it can be made from a variety of materials. The beams shown in the examples are made by laser cutting holes in 5/16 inch square wooden beams, but compatible beams can also be made using a 3D printer or even by manually drilling holes using a drill press. In our experience, wooden beams are not as durable as similar components made from plastic or metal, especially because the beams are severely weakened by the 2 directions of holes. We have tried making BitBeams out of 8 mm acrylic, and while they are much more durable, we ran into the problem that longer beams started to warp significantly due to the heat caused by the laser cutting process.

VI. Programming

Often, robots are programmed using a low-level, textual programming language, such as C. These textual languages can be quite daunting for people with no prior experience. Not only is it hard to translate human language concepts to algorithms a computer can understand, one has to take care that the syntax of the program is correct. Graphical programming languages can alleviate the latter problem, while also providing an interface that is more appealing to children.

A. *Scratch*

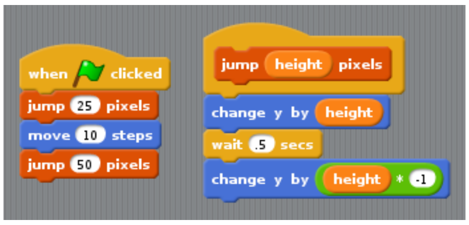

Fig. 9. Scratch [44].

Scratch, developed at MIT Media Lab in 2006, is a graphical programming language that aims to teach children the principles programming through the creation of simple games and interactive movies [45, 46]. Scratch features a display area, onto which different sprites can be placed, and a programming area, onto which puzzle-like programming blocks can be placed. These puzzle blocks can represent simple commands (e.g. “move 10 steps”), or more complicated control structures, such as loop statements (e.g. “repeat … until”) or conditional statements (e.g. “if … then …”). The blocks can be snapped together to create a logical sequence of actions, akin to the sequence of statements one would find in traditional code. Because of the different shapes of the different types of puzzle-like blocks, they can only be combined in a way that makes sense, making syntax errors impossible.

While Scratch is very much oriented at computer-centered use, there are some options for making Scratch interact with the outside world, such as Enchanting [47] and PicoBoard [48]. Enchanting is a variant of the Scratch application that can compile Scratch programs into programs that can be run on a Lego Mindstorms NXT intelligent brick. PicoBoard is a board featuring a light sensor, a sound sensor, a button, a slider and 4 extra ports to which external sensors can be connected. While there are some options to make Scratch interact with the outside world, these options are limited and Scratch is better suited for computer use only. In our opinion, Scratch still provides a great way of learning the fundamentals of programming in a colorful, attractive environment.

B. *Blockly*

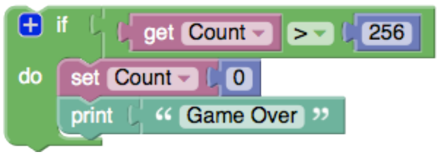

Fig. 10. Blockly [49].

Google Blockly [50], a programming language influenced by the aforementioned Scratch, is different from other graphical programming languages in that it’s not intended for direct use. Instead, Blockly can be seen as a set of libraries that greatly facilitates the development of a Scratch-like programming language. Blockly is written in Javascript and is intended to run inside a web browser environment. Using the Blockly Application Programming Interface (API), one can easily define its own set of blocks in order to create a fully customized graphical programming language. One of the defining features of Blockly is that it can automatically translate a Blockly program to readable, written code. Language generators for Javascript and Python already exist, but custom generators can also be made using the API. Blockly cannot be used to program robots directly (and is not meant to do so), but it does provide a very convenient way to design a language that can be used for that purpose.

C. *ArduBlock*

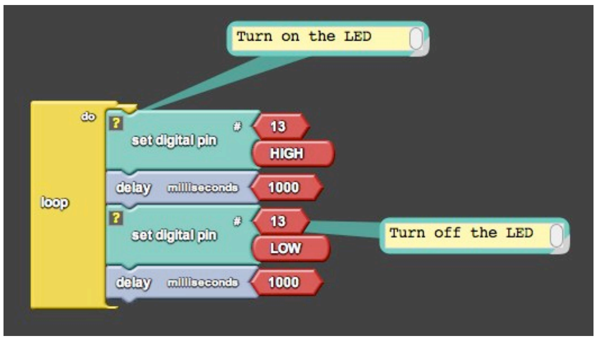

Fig. 11. Raspberry Pi [51].

ArduBlock [52] is a plugin application for the Arduino IDE. It provides an integrated tool that makes it possible to write Arduino programs using the same style of graphical blocks as Scratch and Blockly. In addition to blocks that are literal translations of the functions in the Arduino library, it also provides some predefined blocks for working with third-party Arduino components, such as a relay or a joystick. When programming an Arduino board using ArduBlock, the graphical program is translated to regular Arduino code, not unlike Blockly’s language generators. This facilitates the transition between using graphical blocks for programming and using written C++ code, which is very helpful for novice programmers. While ArduBlocks is a great introductory tool for the Arduino platform, it is our opinion that there is still much room for improvement. It does not yet have the same level of attention to visual details as Blockly or Scratch, not all Arduino functions or features are available and some editing functionality (such as deleting blocks) works counter intuitively. Still, it is a great tool for use in education, especially when students have already been introduced to a similar language, such as Scratch.

VII. DIY AS A NEW WAY OF TEACHING

A DIY methodology promises to transform education from student observation and listening to active engagement through ingenious interactive hands-on lessons guided by instructors and augmented by examples and equipment that can be fabricated only when needed allowing for personalizable and customizable classes and individual learning within common frameworks. The feasibility of a robotics-enhanced problem-based curriculum depends on the access to versatile robotics tools and well-organized tutorials. We screened the available solutions and we carried out a series of interviews with educators to pin point the needs in the field when implementing a new curriculum. For this first round of informative interviews we focused on Flanders (Belgium) and a few primary and secondary school teachers in Italy and Switzerland.

One situation we have encountered frequently is that educators choose an all-in-one robotics platform because they do not know of any alternatives or because they cannot find a clear starting point for alternative platforms. This choice is often further motivated because their regional colleagues tend to use the same platform, so there is a certain form of a support network. In our experience, classes that use a complete robotics platform, such as Mindstorms, tend to outnumber classes that build their own robots from scratch by a large margin. And while robots built with an all-in-one kit may perform better, students that build their own robot either completely from scratch, or by combining elements from the different systems as described above, tend to gain a much deeper understanding of technology, engineering and creative problem solving. Class groups that use a complete robotics platform also tend to stay within that platform, adding a third-party or a home-made component does not happen frequently. Robots built using a complete platform tend to have more functionality and are usually easier to build, while building robots from scratch tends to require more experience but also give the students more insight.

We believe that the DIY techniques of the Maker Movement provide a good way to make building robots from scratch much easier, while also bridging the gap between all-in-one platforms and DIY robots. The Maker Movement makes Rapid Prototyping technologies, such as laser cutting, CNC milling and 3D printing much more accessible to the general public. Whereas the use of 3D printing used to be limited to large organizations and businesses, this movement has created comparable machines, such as the many types of RepRap machines, which can be made for under €1000 [53]. In addition to the option of building your own rapid prototyping machine, which many not always be financially or practically possible for schools, other options have also become available, such as the use of online rapid prototyping services or the use of machines at a local FabLab. FabLabs are publically accessible local workspaces that offer access to these rapid prototyping machines. These FabLabs, if they are locally available, can be a great benefit to robotics projects in schools [54]. They offer a suitable space to work in, access to machines to manufacture parts with, and a community of like-minded people who can share their knowledge and experience.

This DIY way of thinking has already gained some foothold within the context of small-scale robotics. Some of the projects we described above, such as Arduino, OpenBeam, BitBeam, Scratch, Blockly and ArduBlock are Open Source and/or Open Hardware. However, this phenomenon is not limited to these projects. Thingiverse [55], an online repository for design files of physical objects, lists many different types of DIY robot projects, ranging from very simple parts (e.g. a mounting plate to connect an Arduino to Lego bricks [56]) to small wheeled robots (e.g. MiniSkybot [57]) to very complex robots (e.g. humanoid robot InMoov [58]). Another great example of this DIY way of thinking being applied in education is Arvind Gupta's Toys from Trash project [59, 60], which is a website that lists a plethora of small scientific experiments, all of which can be made with very cheap materials.

VIII. Conclusion

In this paper, we have presented a summary of products and projects that can be used as tools for enabling robotics projects in education. The categories we discussed are complete, all-in-one robotics platforms, electronics, hardware, and software. Generally speaking, there are 2 ways to build a robot: either by using a complete robotics platform, or by constructing a robot from scratch. Complete systems are easier to use, allow for quicker results and are better suited for young students. The downside is that they are generally more expensive and less flexible. Building a robot from scratch, in contrast, is much harder and is more suited for older students, but gives much better insight in the technology, is more flexible and can be much cheaper. In recent years, building a robot from scratch has become much easier due to numerous projects and products that implement certain aspects of a robot, such as hardware, software or electronics. These product and projects can often be linked to the recent DIY and Maker Movement trends. These trends are characterized by the use of CNC machines and the collaboration over the internet to create physical hardware projects. We believe that the DIY and Maker subculture can have a valuable impact on education, as it not only encourages young people's interest in STEM-related fields, it also fosters creativity and technological fluency. All of these skills will undoubtedly be vital in the society of tomorrow.

References

[1] W. Sunthonkanokpong, "Future Global Visions of Engineering Education," *Procedia Engineering,* vol. 8, pp. 160-164, 2011.

[2] J. Perkins, "Education in process systems engineering: past, present and future," *Computers & chemical engineering,* vol. 26, pp. 283-293, 2002.

[3] M. Molzahn, "Chemical Engineering Education in Europe," *Chemical Engineering Research and Design,* vol. 82, pp. 1525-1532, 2008.

[4] P. B. Campbell, E. Jolly, L. Hoey, and L. K. Perlman, "Upping the Numbers: Using Research-Based Decision Making To Increase Diversity in the Quantitative Disciplines," 2002.

[5] I. Ríos, A. Cazorla, J. M. Díaz-Puente, and J. L. Yagüe, "Project–based learning in engineering higher education: two decades of teaching competences in real environments," *Procedia-Social and Behavioral Sciences,* vol. 2, pp. 1368-1378, 2010.

[6] O. Ates and A. Eryilmaz, "Factors affecting performance of tutors during problem-based learning implementations," *Procedia-Social and Behavioral Sciences,* vol. 2, pp. 2325-2329, 2010.

[7] S. Papert, *Mindstorms: Children, computers, and powerful ideas*: Basic Books, Inc., 1980.

[8] J. Johnson, "Children, robotics, and education," *Artificial Life and Robotics,* vol. 7, pp. 16-21, 2003.

[9] S. Bell, "Project-based learning for the 21st century: Skills for the future," *The Clearing House,* vol. 83, pp. 39-43, 2010.

[10] L. Dabbish, C. Stuart, J. Tsay, and J. Herbsleb, "Social coding in github: transparency and collaboration in an open software repository," in *Proceedings of the ACM 2012 conference on Computer Supported Cooperative Work*, 2012, pp. 1277-1286.

[11] D. Dougherty, "The Maker Movement," *Innovations: Technology, Governance, Globalization,* vol. 7, pp. 11-14, 2012.

[12] R. Jones, P. Haufe, E. Sells, P. Iravani, V. Olliver, C. Palmer, *et al.*, "RepRap–the replicating rapid prototyper," *Robotica,* vol. 29, pp. 177-191, 2011.

[13] E. Ford. (2013). *ShapeOko*. Available: http://www.shapeoko.com/

[14] M. J. Mataric, N. Koenig, and D. Feil-Seifer, "Materials for enabling hands-on robotics and STEM education," 2007.

[15] S. Hartmann, H. Wiesner, and A. Wiesner-Steiner, "Robotics and gender: The use of robotics for the empowerment of girls in the classroom," in *Gender Designs IT*, ed: Springer, 2007, pp. 175-188.

[16] Lego, "The NXT Technology," http://cache.lego.com/upload/contentTemplating/Mindstorms2History/images/picB2F61C4781F076001DA7E229CE695D81.jpg, Ed., ed, 2013.

[17] Lego. (2013). *LEGO.com MINDSTORMS: Home*. Available: http://mindstorms.lego.com

[18] G. J. Ferrer, "Using Lego Mindstorms NXT in the classroom," *Journal of Computing Sciences in Colleges,* vol. 23, pp. 153-153, 2008.

[19] T. Chikamasa. (2013). *nxtOSEK: Index*. Available: http://lejos-osek.sourceforge.net/

[20] J. Solórzano. (2013). *LeJOS, Java for Lego Mindstorms / NXJ*. Available: http://lejos.sourceforge.net/nxj.php

[21] Rato. (2013). *LEGO Education*. Available: http://www.mindstormsnxt.be/

[22] Mindsensors. (2013). *Mindsensors*. Available: http://www.mindsensors.com/

[23] VEX, "VEX Protobot Robot Kit," http://www.vexrobotics.com/media/catalog/product/cache/103/image/5e06319eda06f020e43594a9c230972d/P/r/Protobot.jpg, Ed., ed, 2013.

[24] InnovationFirstInternational. (2013). *VEX - VEX Robotics*. Available: http://www.vexrobotics.com/vex

[25] InnovationFirstInternational. (2013). *Programming Control Starter Kit - VEX Robotics*. Available: http://www.vexrobotics.com/276-2750.html

[26] Arduino, "ArduinoBoardLeonardo," http://arduino.cc/en/uploads/Main/ArduinoLeonardoFront_2.jpg, Ed., ed, 2013.

[27] Arduino. (2013). *Arduino*. Available: http://arduino.cc

[28] D. Mellis, M. Banzi, D. Cuartielles, and T. Igoe, "Arduino: An open electronic prototyping platform," in *Proc. CHI* vol. 2007, ed, 2007.

[29] Arduino. (2013). *Arduino Leonardo (+headers) [A000057] - €18.00 : Arduino Store - community and electronics*. Available: http://store.arduino.cc/eu/index.php?main_page=product_info&cPath=11_12&products_id=226

[30] Dwengo, "Overview functionality of the Dwengo board," http://www.dwengo.org/sites/default/files/overzicht_dwengo-board_display_0.png, Ed., ed, 2013.

[31] Dwengo. (2013). *Dwengo | Gets you started with microcontrollers*. Available: http://www.dwengo.org/

[32] RaspberryPi, "Raspberry Pi Model B," http://www.raspberrypi.org/wp-content/uploads/2011/07/7513051848_9a6ef2feb8_o-1024x682.jpg, Ed., ed, 2013.

[33] RaspberryPi. (2013). *Raspberry Pi | An ARM GNU/Linux box for $25. Take a byte!* Available: http://www.raspberrypi.org/

[34] G. Mitchell, "The Raspberry Pi single-board computer will revolutionise computer science teaching [For & Against]," *Engineering & Technology,* vol. 7, pp. 26-26, 2012.

[35] MakeBlock, "connection," http://www.makeblock.cc/wp-content/uploads/2012/11/connection1.jpg, Ed., ed, 2013.

[36] ShenzhenHuluRoboticTechnology. (2012). *Makeblock - Aluminum Robot Kit*. Available: http://makeblock.cc/

[37] T. Tam, "OpenBeam 1515 x 1000mm, clear anodized," http://www.openbeamusa.com/images/products/229.jpg, Ed., ed, 2013.

[38] T. Tam. (2013). *OpenBeamUSA.com - Home*. Available: http://www.openbeamusa.com/

[39] T. Tam. (2012). *OpenBeam - An open source miniature construction system by Terence Tam — Kickstarter*. Available: http://www.kickstarter.com/projects/ttstam/openbeam-an-open-source-miniature-construction-sys

[40] J. Huggins, "bitPad - Tri-joint," http://www.flickr.com/photos/68386867@N05/6384780527/, Ed., ed, 2011.

[41] J. Huggin. (2013). *Bitbeam | The Open Source Building Toy*. Available: http://bitbeam.org/

[42] P. Jergenson. (2013). *Gridbeam - open source building system for a post carbon future*. Available: http://www.gridbeam.com/

[43] P. Jergenson, R. Jergenson, and W. Keppel, *How to Build With Grid Beam*: New Society Publishers, 2008.

[44] Scratch, "Scratch 2.0: Create your own block," http://1.bp.blogspot.com/-s0Z7HmigCkY/TY6VV5c1IkI/AAAAAAAAAB8/SqyulusFK1o/s1600/jump-pixels-height.png, Ed., ed, 2011.

[45] MIT. (2012, September 10). *Scratch | Home | imagine, program, share*. Available: http://scratch.mit.edu/

[46] M. Resnick, J. Maloney, A. Monroy-Hernández, N. Rusk, E. Eastmond, K. Brennan*, et al.*, "Scratch: programming for all," *Communications of the ACM,* vol. 52, pp. 60-67, 2009.

[47] C. Blackmore. (2012). *Enchanting*. Available: http://enchanting.robotclub.ab.ca/tiki-index.php

[48] PlayfulInvention. (2013). *PicoBoard - Sensor Board that works with MIT's Scratch*. Available: http://www.picocricket.com/picoboard.html

[49] N. Fraser, "Blockly Sample," http://blockly.googlecode.com/svn/wiki/sample.png, Ed., ed, 2013.

[50] N. Fraser. (2013). *blockly - A visual programming editor*. Available: http://code.google.com/p/blockly/

[51] D. Li, "Getting Started with Arduino in ArduBlock: Example 01," http://blog.ardublock.com/wp-content/uploads/2012/02/untitled1.jpg, Ed., ed, 2012.

[52] D. Li. (2012). *ArduBlock*. Available: ardublock.com

[53] A. Bowyer. (2012). *RepRap - RepRap Wiki*. Available: http://reprap.org/wiki/Main_Page

[54] P. Blikstein, "Digital Fabrication and 'Making'in Education: The Democratization of Invention."

[55] MakerbotIndustries. (2012). *Thingiverse - Digital Designs for Physical Objects*. Available: http://www.thingiverse.com/

[56] badBrick. (2013). *Brick Mount Base Plate for Arduino UNO R3*. Available: http://www.thingiverse.com/thing:62139

[57] J. Gonzalez-Gomez. (2011). *MiniSkybot Robot V1.0*. Available: http://www.thingiverse.com/thing:7989

[58] G. Langevin. (2013). *Head for Robot InMoov*. Available: http://www.thingiverse.com/thing:67676

[59] A. Gupta. (2013). *Toys from Trash*. Available: http://www.arvindguptatoys.com/toys.html

[60] A. Gupta, "Aha! activities," *Bhopal: Eklavya Publication,* 2007.