Mobile Computing and Robotics in One Course: Why Not?

Stan Kurkovsky
Computer Science
Central Connecticut State University
1615 Stanley Street, New Britain, CT 06050, USA
kurkovsky@ccsu.edu

ABSTRACT

Robotic technology offers an excellent platform providing a hands-on learning environment for reinforcing theoretical topics in computer science, computer and electrical engineering, and mathematics. Robotics has been successfully used to promote student interest in computing and other STEM disciplines. However, students whose interest in computing may have been sparked or sustained by robots may be seeking more experience with robotics in the rest of the computing curriculum. This paper describes an effort to introduce robotics-related material into an existing upper-level course in mobile computing and discusses the rationale for such a pairing.

Categories and Subject Descriptors

K.3.2 [**Computers and Education**]: Computer and Information Science Education – *Computer science education, Curriculum.*

General Terms

Design, Experimentation, Human Factors.

Keywords

Android, mobile computing, mobile application development, Sphero, robotics.

1. INTRODUCTION

Using robots in the curriculum provides a convenient avenue to demonstrate and practice a synergistic application of concepts from computer science, engineering, physics, and mathematics. Robotics has an inherent appeal on both emotional and intellectual level that makes it attractive to a broad range of learners across multiple dimensions, such as age, gender, academic interest, or the chosen program of study. Robotics has become an increasingly popular topic to promote computing and use it as an engaging and motivational tool in introductory courses offered in middle and high schools, summer camps, as well as in CS 0 and CS I college-level courses [1,12]. This is very closely related to the increased availability of relatively cheap robotic platforms and kits, as well as inherent attractiveness of robotics and related technologies to the younger generation.

Hopefully, the effort to increase the appeal of computing by using robots in introductory courses will result in a noticeable influx of students and improved retention. However, since their interest in computing may have been sparked or sustained by robots, these students may be seeking more experience with robotics in the rest of the computing curriculum. As it stands right now, there are not too many undergraduate Computer Science programs that offer courses in robotics or include robotics-related material beyond CS 0 or CS I [20]. This paper describes the author's attempt to fill this void by exploring the possibility to combine robotics and mobile computing in a single course.

ITiCSE'13, July 1–3, 2013, Canterbury, England, UK.
Copyright 2013 ACM 978-1-4503-2078-8/13/07 ...$15.00.

2. ROBOTS IN CS CURRICULUM

The vast majority of literature on robotics in the computing curriculum focuses on using robots as a motivational tool for contextualized teaching and as an outreach tool to introduce K-12 and non-CS college students to computing [13]. McGill published an excellent survey of published curricular initiatives that use robotics to boost student motivation in introductory computing courses [14]; other surveys review available robotics education platforms for K-12 [3], undergraduate [19], and graduate programs [7]. Curricular activities involving educational robots typically address a variety of aspects in student learning ranging from communication skills to creativity and problem solving.

Outside of using robots as a motivational and engaging factor in introductory courses, there are very few undergraduate computer science programs offering courses in robotics. This can be partly explained by the fact that robotics has always been perceived as a more 'hardware-oriented' topic, which in the past may have driven CS students away from it. Equipment costs and lack of faculty expertise are also among the factors contributing to the students' lack of exposure to this important field. As an inherently interdisciplinary area, robotics brings together a broad range of computing sub-disciplines (such as computer architecture, artificial intelligence, software engineering, embedded systems, etc.), as well as mechanical and electrical engineering, physics, and mathematics. Therefore, introducing students to robotics can help them establish a better understanding of the relationship between computing and other science and engineering disciplines, and also provide students with an experience in designing and building complex systems that combine hardware and software.

But what happens to those students who make the choice to study computing and may be looking to learn about robotics beyond what they may have experienced in a course where robots were used to engage and motivate? Will they get any further exposure to robotics? Kay poses an interesting question whether this is a case of bait and switch when we use a contextualized approach (such as robotics or media computation) in an introductory course, but then leave students with nothing but dry theory and contrived examples in the courses that follow [11]. Although generally this question remains unanswered, a solution can be found by incorporating relevant contextual topics into the curriculum. When it comes to incorporating a particular emerging technology or a topic into the curriculum, institutions typically have three options: introduce a new course focusing on that topic, use an existing course, or incorporate the material as a unit(s) within other existing course(s) [4]. Offering a separate course may provide the benefits of a more focused and in-depth study of the topic at hand, but it is not always easy due to logistics and bureaucracy. Incorporating relevant topics into an existing course may often be the simplest way to address this problem.

3. ROBOTICS AND MOBILE COMPUTING

If robotics-related material were to become a substantial part of another course, which computing topic would offer a good pairing? Good candidates may include computer architecture [10], artificial intelligence [9], computer vision [8], embedded systems [15], etc. The objective of this paper, however, is not to offer a comparative analysis of possible choices and argue for the best combination. Instead, based on the author's extensive experience with mobile computing, this paper explores a combination of robotics with mobile application development for Android.

There are very few reports in the literature that describe any experience of combining the use of robotics and mobile computing throughout an entire course, or even in a single hands-on activity. Uludag et al [21] describe a simple lab activity that uses App Inventor [23] to create an application that connects to a Lego Mindstorms robotic device and has the capability of starting and stopping its motor. A number of reports indicate that Android is a suitable platform for teaching embedded systems at both undergraduate [15] and graduate level [22].

Using programmable devices, such as robots or smartphones, has been shown to stimulate student creativity and problem-solving skills [2]. Hands-on experimentation with tangible real-world objects supports the principles of constructionist teaching and learning, which helps students organize and transfer theoretical knowledge to practice through experience. Programmable devices are not only fun to work with; they provide an excellent platform for a holistic combination of practice and theory [8].

Robots and mobile devices such as smartphones exemplify extreme integration: both combine a powerful processor, communication capabilities, and a diverse range of sensors. Despite the obvious differences in the design emphasis (electromechanical capabilities in robotics and computational and communication richness in mobile devices), these two kinds of devices have a lot in common. Extreme integration allows robotic devices and smartphones leverage their ability to sense and interact with the real-world objects. Using robotics technology and other hands-on educational contexts promotes developing better critical thinking and problem solving skills, which are essential for student success in the STEM fields. Indeed, mobile devices have been used throughout the CS curriculum: as a learning context, as a tool to improve student engagement and motivation, and as a focus of study in mobile computing and mobile application development courses [5]. Both mobile computing and robotics are crosscutting areas of CS in that they require students to have a working knowledge of computer architecture, operating systems, and computer networking, while reinforcing the notion that neither of these areas exists in isolation from the others.

4. THE CHOICE OF PLATFORMS

Having made the decision to combine robotics with mobile application development, it is important to make a good choice of hardware platforms that are well suited for the educational environment while providing enough flexibility to illustrate the richness of each discipline, as well as their close relationship with each other.

4.1 Android

The choice of Android as a mobile application development platform is easy to justify:

Low learning curve. The vast majority of students already know Java, Android's primary development language.

Flexibility of development platform. Application development environment (e.g. Eclipse) is supported by multiple operating systems.

Low to non-existent costs. Unless students plan to distribute their mobile apps via Google Play store that requires a one-time $25 registration fee, there are no costs associated with Android application development.

Availability of devices. Android is the most popular mobile platform, and students do not need a top of the line or the most up to date device for development purposes.

4.2 Sphero

Sphero is a small robotic ball manufactured by Orbotix (www.gosphero.com) that is equipped with internal motors allowing it to roll on a flat surface in any direction. It is equipped with a number of sensors and has an open low-level API to communicate with other devices via Bluetooth. With high-level SDKs for iOS and Android platforms, Sphero can be controlled by an external application running on a mobile device, or by macros and orbBasic code executed by the robot itself. The following factors influenced the choice of Sphero as a suitable robotic platform:

It is a robot. Despite its limited functionality, first and foremost, Sphero is a robot. Section 4.3 gives a grounded justification why Sphero is a suitable platform to expose students to the most important elements of robotics.

Emphasis on communication. As a platform, Sphero makes a strong emphasis on communication between the robot and iOS or Android devices, which not only makes it especially suitable for pairing with the Android material, but also offers a rich context to explore many important issues of complex hardware/software systems, which are not always covered well in the curriculum.

Open SDKs. Orbotix provides iOS and Android SDKs along with a set of well-documented sample applications.

Variety of uses. In addition to its most obvious application as a robot, Sphero can also be used in a number of other ways, which include a 3D controller and a moving fiducial for augmented reality applications, which are supported by the open SDK.

4.3 Can Sphero Be Used to Teach Robotics?

Touretzky describes "seven big ideas" for teaching robotics [20], which collectively could shape the students' understanding of the fundamental ideas in the field of robotics. Each of the seven ideas poses a question, which uses the robotics context to expose students to a deeper and often rather complex computing concept, but at the same time offers a tangible answer that is easy to demonstrate and understand. Although Sphero cannot be considered a truly autonomous robot, we believe that it can be used to expose students to each of the seven big ideas, which are discussed below.

4.3.1 How Do Robots Know What to Do?
Specially designed algorithms and a clearly articulated set of constraints and goals control the behavior of a truly autonomous robot. A simple robot such as Sphero may have a simplified way of 'knowing' how to seek its goals. Sphero's behavior can be programmed by macros or orbBasic code executed on the device, which can make decisions based on its sensor readouts. Sphero can also be controlled by and communicate with an application running on an iOS or Android device.

4.3.2 How Do Robots See the World?
Robots equipped with light sensors or cameras can implement computer vision algorithms to identify real-world objects surrounding them. Sphero's sensors include an accelerometer, a magnetometer, and a gyroscope. In the absence of a camera or a light sensor, Sphero's ability to 'see' the world is limited to detecting collisions with other objects, which can be registered by changes in the accelerometer readouts. Applying Fourier transform can help correctly detect a spike in the accelerometer data stream indicating a collision with an obstacle. The shape and magnitude of the spike may help determine the angle of collision; comparing collision data from multiple Spheros can also help detecting a robot-to-robot collision.

4.3.3 How Do Robots Know Where They Are?
Generally, robots use odometry (estimation of the direction and the distance traveled) or landmark identification (which can include identification of natural artifacts using computer vision, or using fiducials, such as visual tags or RFID tags). Sphero relies on odometry to maintain its own coordinate system and its current heading.

4.3.4 How Do Robots Know Where to Go?
A typical robot navigates the world and avoids obstacles by using a path planner to search for suitable routes; an execution monitor ensures that the robot stays on the chosen route. Sphero does not have a built-in path planner or plan execution monitor, but the features of both can be implemented either by an external mobile application communicating with the robot, or using orbBasic code executed by the robot itself.

4.3.5 How Do Robots Control Their Bodies?
Kinematic solvers and kinematic trees are used to translate between the robot body coordinates and joint angles. The absence of any external moving parts requires shifting the focus of discussion from kinematics to the algorithms controlling the two motors inside the robotic ball.

4.3.6 What Can We Do When a Robot Becomes Too Complex for One Person to Fully Understand It?
Any robot is a very complex system comprising both hardware and software parts designed for close synergistic cooperation. Abstraction of functionality and modular design must be used to manage complexity of any robotic system. Sphero provides developers with different ways to program the robot at three different levels of abstraction: low-level orbBasic code executed by the robot itself, macros, and SDK-based apps running on an external mobile device. Furthermore, Sphero Android and iOS SDKs provide access to the robot's features at different levels of abstraction ranging from telling the robot to roll at a given speed or catching collision events to controlling the settings of a single motor or reading individual serialized packets containing raw accelerometer data.

4.3.7 How Do We Calculate the Quantities Needed To Make a Robot Function?
Robotics as a discipline is grounded in mathematics. Geometry, linear algebra, and trigonometry are essential in planning and executing the movement of a robot. Programming complex paths for Sphero often requires transforming one coordinate system to another. Enabling Sphero to move specific distances requires mapping and calibrating its time-based motor control and speeds measured in percentages of the maximum velocity into a different measurement system. Digital signal processing, such as applying Fourier transform, is needed to extract meaning out of a stream of sensor readouts, for example for detecting Sphero collisions with obstacles. Robots equipped with cameras use computer vision algorithms, which rely on feature extraction algorithms, such as Hough transform, to detect basic shapes.

5. ANDROID+SPHERO IN PRACTICE
Any mobile application development course needs to emphasize the features that are unique to the mobile platform, e.g. taking advantage of the data collected by sensors, integrating live feed from the camera, using enhanced connectivity options to communicate with other services and devices, integration of telephony and messaging, etc. A course that combines mobile computing with the elements of robotics needs to leverage the features common to both types of devices. Such a course should also have a strong focus on hands-on activities that emphasize these similarities, as well as a symbiotic relationship that exists between mobile and robotics devices in a single complex system. Keeping these considerations in mind, course topics could include the following:

Introduction. Why is application development for a mobile platform different compared to other platforms? What are the differences and similarities between Android and other mobile operating systems?

Android platform. What is common between Android and Linux? How does Android software stack work and what does it consist of? How are typical Android apps started, executed, and terminated?

Android app UI. How do Android apps interact with the user? What is the role of the Model-View-Controller design pattern in Android application development?

Intents. How do Android apps communicate with each other and with the operating system? What can be done to leverage the functionality of other Android apps?

Persistent data. What options are there for Android apps to store and retrieve persistent data? What makes working with persistent data on a mobile device different from doing the same on a desktop?

Sensors. What types of sensors are typically available to Android apps and how to work with them? Why do we need virtual sensors and what are their advantages? How to create an event-driven app that works with sensor readouts?

Introduction to robotics with Sphero. What are the features of a typical robotic device? What can Sphero do and is it really an autonomous robot? What's inside of it and how do its components work together?

Motion. How do you program Sphero to move? How does it know where to go? What sensors make Sphero keep its course and remember its heading after collisions?

Sensor data. What can apps do with the sensor data streaming from Sphero? What sensor data can be streamed and how can it be transformed and interpreted to become more useful? How to use Sphero as a 3D controller for other applications?

Collision detection. How does Sphero detect collisions with other objects? What is the role of Fourier transform in detecting collisions? Why is there a data reliability threshold in collision detection?

Autonomous behavior. What is the fundamental difference between three models of controlling Sphero: sending individual commands, executing macros, and running orbBasic programs? What are the characteristics of an autonomous robot and can Sphero become one?

The prerequisites of our course combining mobile application development and robotics is positioned in the program ensure that all students have a substantial programming experience and a working knowledge of trigonometry and linear algebra. The course is structured so that there are two weekly meetings lasting 1 hour and 15 minutes each. Typically, every week is dedicated to a single topic with a lecture and demonstrations during the first meeting and a hands-on lab during the second. Students work in teams of two on every lab and the course project, which promotes good teamwork skills. Such an arrangement also helps alleviate any possible equipment-related problems: students can share one Android device and one Sphero per team.

5.1 Hands-on Labs

No mobile application development or robotics course would be complete without an extensive set of labs aimed to complement the theoretical concepts with practical hands-on experiences. A sequence of hands-on labs of increasing levels of complexity is also essential to keeping students engaged with the course material. Additionally, staging the material so that every new experience reuses and builds upon the results of the previous lab helps maximize the depth of the covered material. The goal of each lab is to produce a fully functional Android app. Labs include the following:

- Creating Android app layouts and different Android views using XML;
- Retrieving data from a remote host with the help of intents and HTTP connections;
- Manipulating shared preferences and local resource data;
- Implementing an Android shake counter with accelerometer readouts;
- Driving Sphero with a simple GUI-based controller;
- Creating an etch-a-sketch app (or a snake game) and using Sphero as a 3D controller; and
- Experimenting with detecting collisions with walls and other robots.

Students are encouraged to complete and demo each lab during the class time, but this may not always be possible, in which case they have at least two full days to complete their work. Each lab is accompanied with a set of scaffolding code, which is ready to be imported into Eclipse. Android code included in the scaffolding projects spares students from setting up the Eclipse project, creating the user interface, and performing other routine work. This helps students focus on the task at hand concentrating on the topic of the current lab exercise. Scaffolding code provided with each lab ensures that regardless of their previous work, students always have a level starting point, which maximizes their chances for success.

5.2 Course Project

Project-based learning [6,17] creates an environment where students acquire such soft skills as time management, project planning, and effective teamwork, that are not always an explicit part of the CS curriculum, but are demanded by the employers. A team-based course project allowed students to practice many concepts presented in this course by creating an application for scripting the behavior of a Sphero robot. This project required students to apply the following skills that underscore the interdisciplinary nature of the course:

- Mobile computing: implementing a complex Android application utilizing a broad range of features specific to the mobile platform;
- Operating systems: design and implement a real-time multithreaded controller for a wirelessly connected robot;
- Human-computer interaction: design a user-friendly mobile interface for script editing, storage, and retrieval;
- Programming languages: design and implement a simple parser for custom scripts controlling the robot's behavior;
- Computer communications and networking: implement an HTTP-based connection to retrieve text-based scripts from a URL; implement asynchronous data transfer between the robot's sensors and a mobile device over Bluetooth;
- Robotics: use a set of well-defined commands to control the motors of the robot enabling it to move at given speeds: roll along a straight line of a given length, roll along an arc with a given radius and angle; turn in place; and change the LED color;
- Linear algebra: transform the robot's coordinate system and an absolute heading into a coordinate system with a relative heading; convert the robot's odometry into real-world speed metrics;
- Physics: understand and account for the effects of floor traction at low speeds.

The course project culminated in a robot race, in which Spheros competed for accuracy of navigation. At the time of competition, student teams were given a script that described a course shown in Figure 1 and Figure 2.

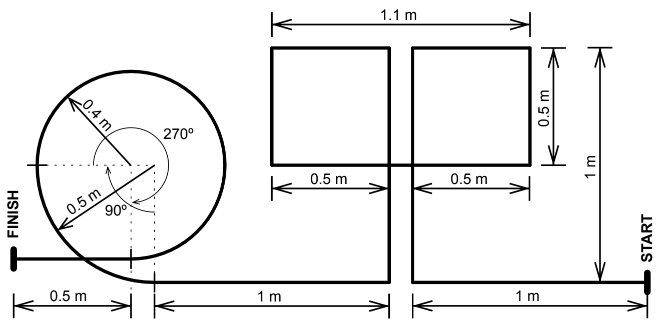

Figure 1. Sphero race track (drawn to scale).

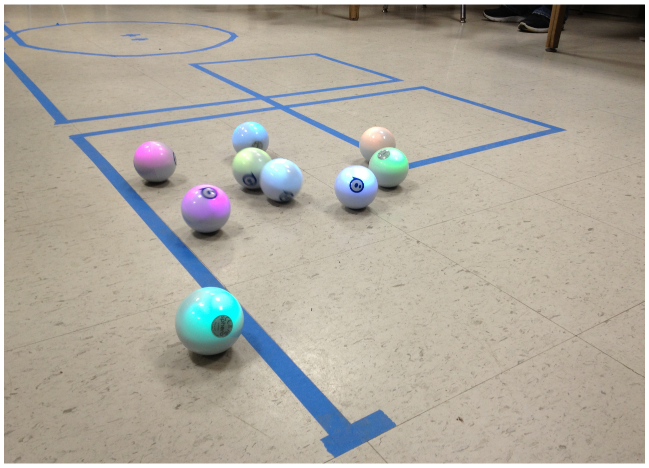

Figure 2. Sphero robots on the race track.

Student competitions such as the Sphero race offer an opportunity to support the intellectual growth of students by helping them find their own solution to a problem and understand that there could be many different, but correct ways to reach the goal, that there could be multiple right answers to the same research question, and that there could be more than one working solution to the stated practical problem. Such an approach promotes critical thinking by encouraging students to identify problems, find and evaluate different solutions, work in teams toward achieving a common goal, and apply theoretical concepts in practice [6]. Furthermore, student project competitions in general, as well as those involving robotics, can help get more students involved in research, offer students additional motivational factors, and make the learning process more experiential [16,19].

6. STUDENT FEEDBACK

At the conclusion of the course, student provided feedback representing their perception of the course quality, learning outcomes, and possible improvements.

The vast majority of students appreciated the combination of mobile computing and robotics and their symbiotic relationship:

> *Sphero+Android is a unique medium and they work very well together. It holds interest as a subject matter and Android seems like a relevant system to learn.*
>
> *It's totally different from any other CS course, and in a good way! You get to use programming knowledge in a fun way and it's a cool feeling to see something move/happen with your code.*
>
> *Mobile programming is growing as an almost essential component of CS education. Robotics is awesome. You get to flex your creative muscles. We play with toys!*

When asked about the most interesting hands-on components of the course, a clear consensus emerged: all students preferred either the final course project or the etch-a-sketch lab, the two activities that made the biggest emphasis on combining robotics and mobile application development.

> *Etch-a-sketch: it opened up new doors for using a robot as an interface device.*
>
> *Final project: real application and very challenging to figure out how to get distance/arc to work.*
>
> *The course project: though large, it integrated many subject areas and it was satisfying to see the results.*

Problems related to technical issues, such as setting up Eclipse and debugging the Sphero code were the most frustrating to students. Aside from the first hands-on lab activity being too easy, there were very few other complaints about the course content.

7. CONCLUSION AND FUTURE WORK

It is generally accepted that the broad goal of using educational robots is not to create experts in robotics, but rather to help students develop essential real-world competencies [8,18]. In the context of intermediate and advanced computing courses, experiences with robotics could offer students an opportunity to combine the theoretical material from various sub-disciplines of computing and apply them in an engaging practical setting. Experiences described in this paper also give students competencies in the areas that are normally not covered by other topical courses in computing. These include working with complex hardware/software systems at a lower level, getting hands-on systems programming experience, integrating hardware sensors with control software, understanding practical aspects of real-time and embedded systems, working with uncertainty of the robot's perception of the real world through sensors, and applying best software engineering practices to solve problems.

Unsurprisingly, student feedback revealed that there is a lot of room for improvement. The objective of the first course offering described here was to balance the material equally between mobile computing and robotics, placing the introduction of Sphero in the middle of the semester. However, in order to better prepare students for the final project, the course schedule may need to be more front-loaded, thus freeing the end of the semester of any new theoretical material and giving students more time to experiment.

8. ACKNOWLEDGMENTS

The author would like to thank his students who enthusiastically completed all labs and projects in the Android+Sphero course and provided very valuable feedback. The author is also thankful to the entire Orbotix team for providing Sphero robots, guidance, and feedback throughout the development of the course described here.

9. REFERENCES

[1] Anderson, M., McKenzie, A., Wellman, B., Brown, M., Vrbsky, S. 2011. Affecting attitudes in first-year computer science using syntax-free robotics programming. *ACM Inroads*, 2, 3 (Aug. 2011), 51-57. http://doi.acm.org/10.1145/2003616.2003635.

[2] Apiola, M., Lattu, M., Pasanen, T.A. 2010. Creativity and intrinsic motivation in computer science education: experimenting with robots. In *Proceedings of the 15th annual conference on Innovation and technology in computer science education* (ITiCSE '10). ACM, New York, NY, USA, 199-203. http://doi.acm.org/10.1145/1822090.1822147.

[3] Ben Brahim, T., Marghitu, D. & Weaver, J. 2012. A survey on robotic educational platforms for K-12. In *Proceedings of World Conference on E-Learning in Corporate, Government, Healthcare, and Higher Education 2012,* 41-48. Chesapeake, VA: AACE.

[4] Bogle, S.A., Potter, W.D. 2011. Using robot based learning to enhance CS curriculum delivery, In *Proceedings of the 11th IEEE International Conference on Advanced Learning*

Technologies (ICALT) (Jul. 2011), 576-578, http://dx.doi.org/10.1109/ICALT.2011.174.

[5] Burd, B., Barros, J.P., Johnson, C., Kurkovsky, S., Rosenbloom, A. Tillman, N. 2012. Educating for mobile computing: addressing the new challenges. In *Proceedings of the final reports on Innovation and technology in computer science education 2012 working groups* (ITiCSE-WGR '12), ACM, New York, NY, USA. 51-63. http://doi.acm.org/10.1145/2426636.2426641.

[6] Cappelleri, D. J., Vitoroulis, N. 2013. The robotic decathlon: project-based learning labs and curriculum design for an introductory robotics course. *IEEE Transactions on Education*, [EARLY ACCESS], Feb. 2013. http://dx.doi.org/10.1109/TE.2012.2215329.

[7] Carter, J., Matthews, S., Coupland, S. 2011. Teaching robotics at the postgraduate level: Assessment and feedback for on site and distance learning. In *Proceedings of International Conference on Robotics in Education*. 171-176.

[8] Cielniak, G., Bellotto, N., Duckett, T. 2012. Integrating mobile robotics and vision with undergraduate computer science. *IEEE Transactions on Education.* http://dx.doi.org/10.1109/TE.2012.2213822.

[9] Dodds, Z., Greenwald, L., Howard, A., Tejada, S., Weinberg, J. 2006. Components, curriculum, and community: Robots and robotics in undergraduate AI education. *AI Magazine*, 27, 1 (Spring 2006), 11-22.

[10] Doerschuk, P., Liu, J., Mann, J. 2009. INSPIRED broadening participation: first year experience and lessons learned. In *Proceedings of the 14th annual ACM SIGCSE conference on Innovation and technology in computer science education* (ITiCSE '09). ACM, New York, NY, USA, 238-242. http://doi.acm.org/10.1145/1562877.1562952.

[11] Kay, J.S. 2011. Contextualized approaches to introductory computer science: the key to making computer science relevant or simply bait and switch? In *Proceedings of the 42nd ACM technical symposium on Computer science education* (SIGCSE '11). ACM, New York, NY, USA, 177-182. http://doi.acm.org/10.1145/1953163.1953219.

[12] Lee, I., Martin, F., Denner, J., Coulter, B., Allan, W., Erickson, J., Malyn-Smith, J., Werner, L. 2011. Computational thinking for youth in practice. *ACM Inroads*, 2, 1 (Feb. 2011), 32-37. http://doi.acm.org/10.1145/1929887.1929902.

[13] Major, L., Kyriacou, T., Brereton, O. P. 2011. Systematic literature review: Teaching novices programming using robots, In Proceedings of *15th Annual Conference on Evaluation & Assessment in Software Engineering (EASE 2011)*, 21-30, Apr. 2011. http://dx.doi.org/10.1049/ic.2011.0003.

[14] McGill, M.M. 2012. Learning to Program with Personal Robots: Influences on Student Motivation. *Transactions on Computing Education.* 12, 1 (Mar. 2012). http://doi.acm.org/10.1145/2133797.2133801.

[15] Muppala, J.K. 2011. Teaching embedded software concepts using Android. In *Proceedings of the 6th Workshop on Embedded Systems Education* (Oct. 2011). ACM, New York, NY, USA, 32-37. http://doi.acm.org/10.1145/2077370.2077375.

[16] Murphy, R.R. 2001. 'Competing' for a robotics education, *Robotics & Automation Magazine, IEEE*, 8, 2 (Jun. 2001), 44-55. http://dx.doi.org/10.1109/100.932757.

[17] Qidwai, U. 2011. Fun to learn: project-based learning in robotics for computer engineers. *ACM Inroads*. 2, 1 (Feb. 2011), 42-45. http://doi.acm.org/10.1145/1929887.1929904.

[18] Pitti, K., Curto, B., Garcia, J. Moreno, V. 2010. NXT workshops: Constructionist learning experiences in rural areas. In *Proceedings of International Workshop "Teaching robotics, teaching with robotics"*, Darmstadt, Germany, Nov. 2010.

[19] Ruzzenente, M., Koo, M., Nielsen, K., Grespan, L., Fiorini, P. 2012. A review of robotics kits for tertiary education. In *Proceedings of International Workshop Teaching Robotics Teaching with Robotics: Integrating Robotics in School Curriculum*, 153-162, Riva del Garda, Italy, Apr. 2012.

[20] Touretzky, D.S. 2012. Seven big ideas in robotics, and how to teach them. In *Proceedings of the 43rd ACM technical symposium on Computer Science Education* (SIGCSE '12). ACM, New York, NY, USA, 39-44. http://doi.acm.org/10.1145/2157136.2157152.

[21] Uludag, S., Karakus, M., Turner, S. W. 2011. Implementing IT0/CS0 with scratch, app inventor for Android, and Lego Mindstorms. In *Proceedings of the 2011 conference on Information technology education* (SIGITE '11) (Oct. 2011). ACM, New York, NY, USA, 183-190. http://doi.acm.org/10.1145/2047594.2047645.

[22] Wang, M.-T., Huang, P.-C., Lee, J.-K., Lai, S.-H., Jang, R., Chang, C.-F., Liu, C.-W., Kuo, T.-W., Liao, S. 2010. Support of Android lab modules for embedded system curriculum. In *Proceedings of the 2010 Workshop on Embedded Systems Education* (Oct 2010). ACM, New York, NY, USA. http://doi.acm.org/10.1145/1930277.1930281.

[23] Wolber, D. 2011. App inventor and real-world motivation. In *Proceedings of the 42nd ACM technical symposium on Computer science education* (Jun. 2011). ACM, New York, NY, USA, 601-606. http://doi.acm.org/10.1145/1953163.1953329.