Interdisciplinary Connections in a Mobile Computing and Robotics Course

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

ABSTRACT

Using robots and mobile devices in education can serve as a foundation for hands-on studies encompassing many different disciplines including computer and electrical engineering, computer science, mathematics, and physics. Educational robots and mobile technology have been shown to be successful in promoting student interest in these and other STEM (Science, Technology, Engineering, and Mathematics) disciplines. This paper describes an experience of using robotics and mobile computing in an upper-level Computer Science course. Our discussion focuses on leveraging the inherent symbiosis of these two topics to explore the connections between computing and other disciplines through experiential learning.

Categories and Subject Descriptors

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

Keywords

Android; mobile computing; mobile application development; Sphero; robotics

1. INTRODUCTION

Computing can no longer be considered as a standalone area of study. Instead, the definition of Computer Science (CS) is expanding to emphasize solving practical problems that are directly related to other disciplines. Robotics can make it easier to illustrate the synergistic application of concepts from computing, engineering, physics, and mathematics in different educational settings ranging from elementary school to the university level. Due to their tangible nature, robots and mobile devices have inherent intellectual and emotional qualities that appeal to a diverse array of learners of varying age, gender, or academic interest. Currently, robotics and mobile application development are widely used as a motivational tool to promote STEM disciplines in introductory courses, in summer camps, middle and high schools, as well as entry-level college courses [9,14].

Despite the success of using robotics and mobile computing to attract students to STEM disciplines, these two areas remain under-utilized in CS education. There are very few university-level programs in computing that offer courses in robotics or include robotics-related material beyond the entry-level 'teaser' courses [12]. Furthermore, adding robotics-related material in upper-level courses could provide an excellent environment to illustrate the interdisciplinary nature of modern practical problems solved by computing professionals.

This paper describes the author's experience to fill this void by using a simple robot in combination with mobile devices in a Systems Programming course. This experience explored the symbiotic relationship of robotics and mobile computing that offered ample opportunities to present students with simple, yet inherently interdisciplinary problems that helped students witness many connections between computing and other disciplines.

2. COMPUTING AND INTERDISCIPLINARY PROBLEMS

The need to prepare computer science graduates to work in interdisciplinary projects has long been recognized. In 1996, writing about the strategic directions for CS education, Tucker [18] suggested: "new curricula should be developed that stress the interactions between computer science and other disciplines." Today, more than ever, it is vital for CS programs to produce graduates that are able to meet the challenges of solving practical interdisciplinary problem. In fact, a report from the National Science Foundation [17] indicates: "Future generations of the U.S. science and engineering workforce will need to collaborate across national boundaries and cultural backgrounds, as well as across disciplines."

Both computing industry and research institutions are confronted with an increasing number of problems that are interdisciplinary by their nature. Yet, as Amoussou et al [1] point out, "we have created disciplinary silos of expertise needed to solve problems that are deep and highly focused." The next generation of computing professionals employed in the industry and computer scientists working on research projects must be capable to resolve the challenges stemming from the interdisciplinary nature of emerging problems. To achieve this, we must provide our current students with a rich variety of skills in interdisciplinary computing. Current research literature offers many examples of involving students into interdisciplinary research projects and interdisciplinary tracks, or programs. Examples of cross-connected disciplines include bioinformatics [4,10,20], geographic information technology [3,20], and physics [11]. Furthermore, interdisciplinary research projects have been successfully used to recruit and retain underrepresented students. For example, Beck et al [4] report a great success in using bioinformatics research projects to attract female students to computing and to encourage them to continue their studies at the graduate level.

Although student research projects provide an excellent environment for exploring interdisciplinary connections, it is important to provide students with an opportunity to experience the connections between computing and other disciplines in regular (not research-oriented) and lower-level courses. By doing so, students will gain cross-cutting computing skills and will be better prepared to tackle full-scale interdisciplinary projects later in their studies or after graduation.

3. INTERDISCIPLINARY NATURE OF ROBOTICS AND MOBILE DEVICES

Current publications describing the use of robotics and mobile devices in the computing curriculum emphasize their role to improve student motivation, the use of these devices as a learning context, as well as an outreach mechanism to attract elementary, middle, and high school students, including non-CS college-level students to computing and other STEM disciplines [7,15,16].

As an inherently interdisciplinary area, robotics integrates a wide spectrum of computing sub-disciplines (computer architecture, artificial intelligence, software engineering, embedded systems, etc.), along with mechanical and electrical engineering, physics, and mathematics. Consequently, studying robotics may help students witness and understand a strong symbiotic relationship among several STEM disciplines. Furthermore, using programmable devices such as robots or smartphones has been shown to stimulate student creativity and problem-solving skills [2,19]. 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 also provide an excellent platform for a holistic combination of practice and theory [8].

3.1 Robots and Mobiles in a Single Course

This work explores the combination of robotics material with mobile computing, which has been already advocated for [6,13]. In particular, this paper focuses on the experience of teaching robotics within the frameworks of a Systems Programming course, which includes topics on machine and operating system organization, hardware/software interfaces, hardware-specific constraints on software applications, and using application programming interfaces and system libraries for the design and development of systems applications. Implementing a range of capabilities in a complex hardware/software system requires students to utilize their skills and knowledge from such core competencies as software and system design, software engineering, and mathematical foundations of computing.

Both robots and mobile devices such as smartphones or tablets are perfect examples of systems with extreme integration: they combine a processor, at least one type of communication interface, and a diverse range of sensors within a single functional unit. Although the two types of devices have different design objectives (robots: electromechanical capabilities; mobiles: communication and computation), they share many common features. As a result of extreme integration, robots and mobiles represent a good example of a cross-disciplinary learning context, which extends far beyond computing alone. On the one hand, studying mobile computing and robotics requires students to have a good working knowledge of computer architecture, operating systems, and computer networking, and also reinforces the idea that none of the above areas of computing can exist in isolation from the others. On the other hand, both mobile computing and robotics help illustrate strong connections between computing and other STEM disciplines, such as physics, mechanical and electrical engineering, and mathematics.

Android and Sphero were used as the device platforms. Android was chosen as the mobile application development platform due to a number of factors: a) most students already have a good working knowledge of Java; b) application development environment, such as Eclipse, is supported on many operating systems; c) there are little to no associated costs; and d) Android devices suitable for development are readily available to many, if not all, students.

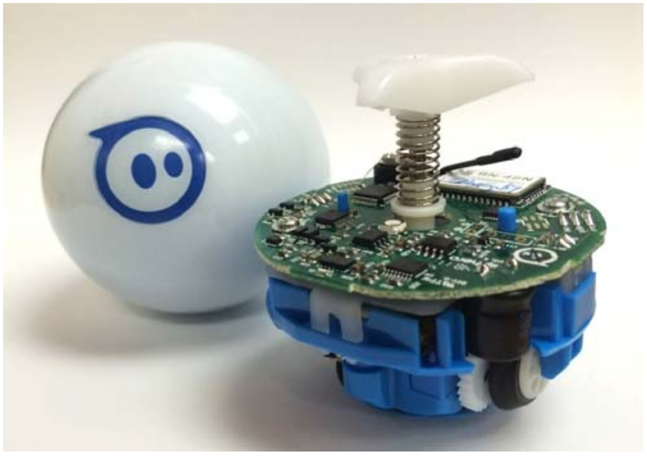

Figure 1. Sphero and its internals.

Sphero shown in Figure 1 is a small ball-shaped robot manufactured by Orbotix and marketed as a gaming and entertainment device. Two internal motors enable Sphero to roll on a flat surface in any direction. Sphero is equipped with an accelerometer, gyroscope, and Bluetooth connectivity, which allows it to communicate with other devices via an open low-level API. High-level SDKs for iOS and Android provide a way to control the robot via an external application running on a mobile device. Our previous work [13] provides a detailed justification for the choice of Sphero as a hardware platform and explains why it is indeed well suited to teach robotics. Our previous work also offers an in-depth discussion of advantages and disadvantages of combining robotics and other topics such as mobile computing in a single course along with a detailed survey of the existing work on integrating robotics material within other topical courses. While the previous work aimed to make the case for integrating robotics-related topics into other courses, this paper focuses on exploring the benefits of pairing robotics with mobile computing and application development. Most importantly, this work emphasizes how hands-on experiences can help students gain better understanding of the existing connections between computing and other related disciplines, such as physics, mathematics, and engineering.

3.2 Hands-on Experiences

Experiential learning focuses on providing students with hands-on experiences in a practical project with tangible outcomes; it provides a rich and rewarding environment that motivates students to strive for success. Project-based learning [5] emphasizes creating functional artifacts that embody the knowledge acquired by students. By creating tangible results through design, development and experimentation, students are able to improve their competence in applying the acquired knowledge and skills, develop and reinforce their problem solving abilities, and practice creativity.

Table 1. Hands-on activities in the Android+Sphero course.

Lab/project	Objectives, topics and related areas
Android GUI	Objective: Build a simple Android app with a few different controls (layouts). Information technology: Software package dependencies; installing and using Eclipse. Software engineering: Model-View-Controller (MVC) pattern and how it is used in Android.
Mini web browser	Objective: Use Android intents to pass and process queries between different user activities and Android services. Operating systems: Inter-process communication. Computer networking: HTTP protocol and request handling. Human-computer interaction: Limitations of mobile user interfaces and interaction modalities.
Context-sensitive help system	Objective: Create a query-based system for retrieving text articles based on keywords. Database systems: persistent data storage and manipulation, SQL. Artificial intelligence: natural language processing.
Android sensors	Objective: Implement a shake counter. Physics: Laws of motion, acceleration, and gravity. Electrical engineering: Implementation of an accelerometer. Human-computer interaction: How to account for physical differences in motion produced by different users.
Intro to Sphero	Objective: Understand the role of each of Sphero's internal components. Electrical engineering: Functionality and integration of electronic and mechanical components inside the robot. Computer networking: Bluetooth protocol and its implementation in Android and Sphero.
Robotic motion	Objective: Learn how to programmatically communicate with Sphero and control its motion. Mechanical engineering: Two-wheeled support for lateral and circular motion. Physics: The force of friction, the role and impact of floor traction at low speeds.
Etch-a-sketch	Objective: Use Sphero as a 3D joystick and stream its sensor data to an Android device to implement an Etch-a-Sketch app. Linear algebra: 3D rotation matrix (roll, pitch, yaw). Embedded systems: Processing hardware sensor data in real time. Data structures: Maintaining a constantly changing non-graphical memory representation of the drawing.
Collision detection	Objective: Detect and study the collisions of Sphero with stationary and moving objects. Physics: Inertia and mass. Digital signal processing: Detecting meaningful changes in real-time accelerometer data. Game design: Design an Android app that would take advantage of Sphero collision detection for entertainment purposes.
Robot race (course project)	Objective: Enable Sphero to behave autonomously by following a complex script. 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. Operating systems: Design and implement a real-time multithreaded controller. 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 to control the robot. Computer networking: Use an HTTP-based connection to retrieve text-based scripts from a URL; implement asynchronous data transfer between the robotic sensors and a mobile device over Bluetooth. Linear algebra: Transform the robot's coordinate system and an absolute heading into a coordinate system with a relative heading; transform an arbitrary circular arc into a sequence of straight lines; convert the robot's odometry into real-world speed metrics. Physics: Understand and account for the effects of floor traction at low speeds.

The Android+Sphero course described here was structured around a number of hands-on lab projects, which, as a rule, were offered once a week. Each lab typically required students not only to practice the theoretical topics and skills emphasized by the most recent course material, but also required students to reach further into other topics of computing, or required students to apply the knowledge and skills from other STEM disciplines. These connections are summarized in Table 1. As illustrated by the topics listed in the table, especially towards the end of the course, the hands-on activities required students to branch out into the related STEM disciplines in order to fully understand the requirements of each activity and successfully reach its objectives.

The final project played an integral role in this course. As shown in Table 1, the objective of the project is to design and develop a complex software system that meets a set of specified requirements with the features ranging from providing a high-level user interface, meeting mid-level data storage considerations, to ensuring a correct low-level interaction with a hardware controller. Furthermore, students are required to make design choices to balance the complexity of the system by distributing its features across various levels of architecture granularity and abstraction paradigms.

The functionality of the project had many layers, but its core was formed by the student implementation of four basic commands to control the robot:

- *Roll* to move the robot along a straight line to the specified distance at the given speed;
- *Turn* to make the robot rotate in place to the specified angle;
- *Arc* to roll the robot along a segment of the circle with the given radius, angle, and speed; and
- *Color* to change the robot's internal LED color to the specified RGB value.

This course project emphasized the importance of evaluating the trade-offs resulting from different design choices at many levels of the system's architectural design. The robot's Arc command could serve as a good example. One way to make the robot follow the path shaped as a segment of the circle is to approximate it by executing a repeating sequence of two other commands, Roll and

Turn. This approach may be easier to implement because reusing two much simpler commands results in breaking down the complexity of the Arc command. At the same time, some computations will be required to derive the length of the Roll and the angle of the Turn to correctly approximate the Arc with the given radius and the overall angle. On the other hand, the Arc command can be implemented by programming the robot at the lower level by indirectly accessing its two motors. Once the robot starts moving straight, a periodic adjustment of its heading would accomplish the objectives of the Arc command. This approach will require converting the linear speed of the robot to tangential speed and sending a sequence of heading adjustment commands in real time. Although the two approaches to implementing the Arc command would typically result in achieving the same overall goal, the original design choices have a significant impact on the outcome. The first approach is a discrete approximation of the curved path and is much simpler to implement. The second approach results in a continuous motion of the robot, but requires a substantial amount of experimentation and testing to get it right.

4. DID STUDENTS SEE THE CONNECTIONS?

The Systems Programming course combining mobile computing and robotics was offered in the Fall 2012 and again in the Fall 2013 semester. In order to better understand the effectiveness of combining these two topics in a single course and leveraging their symbiotic relationship to illustrate the connections between many STEM disciplines, students were asked to complete pre- and post-course surveys. Some of the results are presented here.

Table 2. Student demographics.

Characteristic	2012	2013
Course enrollment	22	23
Male	19	21
Female	3	2
Freshmen	0	0
Sophomores	1	0
Juniors	6	8
Seniors	15	15
Completed surveys	20	20

The objective of this study was to determine whether this course made a positive impact on the students' perception of computing as an area of study and practice, which is concerned with solving interdisciplinary problems. Both at the beginning and at the end of the semester during both course offerings, students were presented with a list of topics and subjects both within and outside of computing, as shown in Table 3. Using a five-point Likert scale, students were asked to rate whether the knowledge and skills in the corresponding areas is useful in a) projects involving robotics, and b) projects involving mobile application development.

Table 3. Pool of topics and subjects used in the study.

Algorithms;	Database systems;
Artificial intelligence;	Electrical engineering;
Assembler language;	Embedded systems;
Compiler construction;	Human-computer interaction;
Computer architecture;	Mathematics;
Computer graphics	Mechanical engineering;
Computer networking;	Object-oriented programming;
Computer security;	Operating systems;
Computer vision;	Physics;
Data structures;	Software engineering.

The difference between the pre-course and post-course results was assessed using a one-tailed t-test to determine those areas in which the changes in student responses were statistically significant. Table 4 and Table 5 show the results of statistical analysis indicating the values for the mean and standard deviation in student responses in both pre- and post-test groups, the change in student response (Δ), as well as the t and p values obtained from the t-test. Student responses from the Fall 2012 and Fall 2013 were merged together into the corresponding pre- and post-test sets of results to increase the pool size for the resulting n=40 (Table 2). For the sake of brevity, these tables only contain data for the items with any statistical significance ($p<0.05$ and $p<0.1$).

Table 4. Connections and effects of robotics.

Item descriptor		Mean	SD	Δ	t	P
Computer vision	pre-test	3.40	1.19	0.45	1.812	0.085
	post-test	3.85	0.81			
Electrical engineering	pre-test	3.90	1.12	0.50	2.058	0.053
	post-test	4.40	0.75			
Embedded systems	pre-test	3.00	1.34	0.80	2.329	0.030
	post-test	3.80	1.28			
Mathematics	pre-test	3.85	0.99	0.55	2.377	0.028
	post-test	4.40	0.75			
Mechanical engineering	pre-test	3.65	1.09	0.50	1.805	0.086
	post-test	4.15	1.18			
Physics	pre-test	3.40	0.99	0.80	3.204	0.004
	post-test	4.20	0.83			

4.1 Connections and Effects of Robotics

As a result of completing this course, students were able to better see the importance of skills and knowledge in several of the areas listed in Table 3 when applied to the projects involving robotics. The analysis of pre- and post-test student responses (Table 4) indicates that there was a statistically significant change ($p<0.05$) in the following areas:

Embedded systems. A number of student activities focused on working with the Sphero's on-board sensors, understanding their functionality, as well as processing the real-time data received from these sensors and making the system react in response to changes in the sensor readouts. During lectures, students were introduced to the high level overview of Sphero architecture with a particular emphasis on the features supporting real-time communication between the robot and other devices. In the Etch-a-sketch lab (Table 1), students disabled the robot's stabilization mechanism to make the robot function as a 3D joystick. Real-time accelerometer data was streamed from Sphero and processed on an Android device to analyze the movements of the robot in space and create an on-screen drawing.

Mathematics. The final course project provided ample opportunities for students to apply some of their math skills. In order to enable the robot to follow the script and navigate the racetrack, the robot's coordinate system and its absolute heading needed to be transformed into a coordinate system with a relative heading. As described in Section 3.2, one of the most challenging parts of the project was implementing the Arc command, which could be done by transforming an arbitrary circular arc into a sequence of straight lines.

Physics. Sphero's shape and mechanics have a significant impact on how the robot moves and interacts with the environment. Due to the lack of a direct contact of Sphero's wheels with the floor surface, traction with the floor is not always perfect, especially on hard surfaces and at low speeds. Also, hard surfaces often do not provide enough traction when the robot starts moving at a high speed. These factors had a significant impact on implementing the robot's motion commands in the final project: students had to conduct many experiments to ensure that the robot behaves consistently at different speeds and on different floor surfaces. During the collision detection lab, students were able to gain hands-on experience with the effect of inertia and mass on the spikes of accelerometer readouts as the collision was registered. Consequently, students had a great deal of in-depth experimentation with Sphero's on-board accelerometer and gained a thorough understanding of the role of gravity on the process of detecting collisions.

As a result of their experiences in this course, students could also better see the importance of these three areas, although with a weaker statistical significance ($p<0.1$):

Electrical and **mechanical engineering.** In order to gain a full mastery of both high-level and low-level features of the robot, students had to get a good grasp on the functionality and integration of electronic and mechanical components inside the Sphero. In particular, to be able to correctly implement the Turn and Arc commands of the final project, students needed a good understanding how the two motors can support both lateral and circular motion of the robot.

Computer vision. Sphero is equipped with an accelerometer and a magnetometer, and these are the only sensors that it can use to perceive the surrounding environment. Although students in this course did not have any experience working with light sensors or with implementing computer vision algorithms, they successfully used the accelerometer for sensing collisions with surrounding objects. As a result, they gained a thorough appreciation for the importance of computer vision in the robotics projects.

Table 5. Connections and effects of mobile computing.

Item descriptor		Mean	SD	Δ	t	P
Computer architecture	pre-test	2.90	0.91	0.50	1.839	0.081
	post-test	3.40	1.27			
Human-computer interaction	pre-test	4.40	0.75	0.35	2.187	0.041
	post-test	4.75	0.44			
Operating systems	pre-test	3.40	1.14	0.45	1.751	0.095
	post-test	3.85	0.99			
Physics	pre-test	2.25	0.91	0.60	2.287	0.033
	post-test	2.85	1.09			

4.2 Connections and Effects of Mobile Computing

Similarly to robotics, mobile computing and application development were also a fertile ground for making connections between computing and other disciplines. The experiences of this course helped students better see the importance of the areas described below when applied to projects involving mobile application development. The analysis of pre- and post-test student responses (Table 5) indicates that there was a statistically significant change ($p<0.05$) in the following areas:

Physics. Prior to being introduced to Sphero, students had to complete a lab exploring the use of sensors that an Android device is equipped with. For example, students had to use the accelerometer to detect a situation when the device is in a free fall or to determine whether the device is being shaken. During such hands-on experiences, students applied such physics concepts as acceleration, motion, and gravity to solve a practical computing problem.

Human-computer interaction (HCI). As the data shown in Table 5 suggests, students came into the course with a preconception that the skills and knowledge in the area of HCI are important in mobile application development. A statistically significant change in the student feedback indicates that the experience of this course reinforced this concept even further. Students were able to consider many limitations and features of mobile devices from the software developer's perspective. In particular, the limitations of a small screen, a lack of or a very small size of the physical keyboard are the most obvious constraints. Voice and gesture interaction, tactile feedback, and physical device manipulation are among the interaction modalities that mobile application developers must be proficient with. In particular, in one of the labs students needed to learn how to account for physical differences in motion produced by different users when implementing a device shake counter. Although HCI is an interdisciplinary field by its very nature, it was not our objective to determine whether students were more interested in the computing or social, engineering, and design aspects of HCI.

As a result of their experiences in this course, students could also better see the importance of these three areas applied in a mobile application development project, although with a weaker statistical significance ($p<0.1$):

Operating systems and **computer architecture.** Mobile application development provides ample opportunities to learn about many aspects of the inner workings of a computing device. For example, communication between components and services receives a substantial emphasis in this course in order to fully explain the fundamental Android principle of using intents. The concept of activity lifecycle also plays a fundamental role in Android. This concept is reinforced within every single lab, thus giving students a substantial exposure to the principles of process management in an operating system. A continued emphasis on the constrained hardware resources available in a mobile device helps students gain a better understanding of computer architecture principles in order to become effective mobile application developers.

5. CONCLUSION

The need to produce computing professionals who are ready to tackle interdisciplinary problems has been recognized by the industry, academia, and the government. So far, most educational efforts concentrated on offering interdisciplinary research projects, as well as interdisciplinary tracks and programs. The objective of this work was to show that interdisciplinary connections could be successfully made in a standalone CS course. The success of this effort could be demonstrated not only through the statistical analysis presented above, but also by quoting the students who answered a number of open-ended questions included in the post-test survey instrument. The following two student answers exemplify the underlying opinion that students expressed when asked to describe the connections between CS and other disciplines:

> *CS embodies all the best parts of engineering and math, specifically creative problem solving, but is infinitely more rewarding because it is relevant to modern society.*
>
> *I believe that the knowledge of math and physics is necessary for anyone who studies CS.*

Incorporating experiential learning activities focused on programming robotic and mobile computing devices helped achieve many of the course goals. Most importantly, by participating in hands-on learning activities involving a complex tangible system, students were able to apply their knowledge and skills from a range of STEM disciplines: computing, mathematics, physics, as well as mechanical and electrical engineering. As a result, students could witness the application of these interdisciplinary connections in a practical setting. The post-test survey asked students whether this course changed their opinion about these connections; the following quote embodies the underlying theme in the student responses:

I saw the connections from the beginning, but the course helped demonstrate them in a tangible way.

6. ACKNOWLEDGMENTS

The author would like to thank all students enrolled in the course described in this paper, as well as the Orbotix team for providing Sphero robots, guidance, and valuable feedback.

7. REFERENCES

[1] Amoussou, G.-A. Boylan, M., Peckham, J. 2010. Interdisciplinary computing education for the challenges of the future. In *Proceedings of the 41st ACM technical symposium on Computer science education* (SIGCSE '10). ACM, New York, NY, USA, 556-557.

[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.

[3] Barr, J., Erkan, A. 2012. Educating the educator through computation: what GIS can do for computer science. In *Proceedings of the 43rd ACM technical symposium on Computer Science Education* (SIGCSE '12). ACM, New York, NY, USA, 355-360.

[4] Beck, J., Buckner, B., Nikolova, O. 2007. Using interdisciplinary bioinformatics undergraduate research to recruit and retain computer science students. In *Proceedings of the 38th SIGCSE technical symposium on Computer science education* (SIGCSE '07). ACM, New York, NY, USA, 358-361.

[5] Blumenfeld, P., Soloway, E., Marx, R., Krajcik, J., Guzdial, M., Palincsar, A. 1991. Motivating project-based learning: Sustaining the doing, supporting the learning. *Educational psychologist*, *26,* 3-4, 369 398. 1991.

[6] 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.

[7] Burd, B., Barros, J., 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.

[8] Cielniak, G., Bellotto, N., Duckett, T. 2013. Integrating mobile robotics and vision with undergraduate computer science. *IEEE Trans. on Education*, 56, 1 (Feb. 2013), 48-53.

[9] 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.

[10] Doom, T. Raymer, M. Krane, D. Garcia, O. 2003. Crossing the interdisciplinary barrier: a baccalaureate computer science option in bioinformatics, *IEEE Transactions on Education*, 46, 3, 387-393, Aug. 2003.

[11] Gentile, L., Caudill, L., Fetea, M., Hill, A., Hoke, K., Lawson, B., Lipan, O., Kerckhove, M., Parish, C., Stenger, K., Szajda, D. 2012. Challenging disciplinary boundaries in the first year: a new introductory integrated science course for STEM majors. *Journal of College Science Teaching*, 41, 5, 44-50, May 2012.

[12] 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.

[13] Kurkovsky, S. 2013. Mobile computing and robotics in one course: why not? In *Proceedings of the 18th ACM conference on Innovation and technology in computer science education* (ITiCSE '13). ACM, New York, NY, USA, 64-69.

[14] 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.

[15] 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.

[16] McGill, M.M. 2012. Learning to Program with Personal Robots: Influences on Student Motivation. *Trans. on Computing Education.* 12, 1 (Mar. 2012).

[17] National Science Foundation. 2006. *Investing in America's Future: Strategic Plan 2006–2011.* Arlington, VA.

[18] Tucker, A.B. 1996. Strategic directions in computer science education. *ACM Comput. Surv.* 28, 4 (Dec. 1996), 836-845.

[19] 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.

[20] Zhang, M., Lundak, E., Lin, C.C., Gegg-Harrison, T., Francioni, F. 2007. Interdisciplinary application tracks in an undergraduate computer science curriculum. *In Proceedings of the 38th SIGCSE technical symposium on Computer science education* (SIGCSE '07). ACM, New York, NY, USA, 425-429.