Enhancing an introductory programming course with physical computing modules

Miguel Angel Rubio and Rocio Romero-Zaliz
Departamento de Ciencias de la Computación e IA
University of Granada
Granada, Spain
marubio@ugr.es

Carolina Mañoso and Angel P. de Madrid
Departamento de Sistemas de Comunicación y Control
UNED - Universidad Nacional de Educación a Distancia
Madrid, Spain

***Abstract*— Learning to program can be very difficult for the students involved. Students must master language syntax, programming theory and problem solving techniques in a short period of time. A non-traditional approach might help students to overcome these difficulties. Several studies have proposed the use of the physical computing paradigm. This paradigm takes the computational concepts "out of the screen" and into the real world so that the student can interact with them.**

Following this paradigm we designed different learning modules -to be used in lectures and laboratory sessions- to teach C/C++ and MATLAB. Lecturers explain a computational concept and, afterwards, reinforce it using the physical computing modules. For example, conditional structures are illustrated using a photocell and several LEDs, arrays are explained using musical melodies, etc.

The effectiveness of the physical computing modules was assessed by means of learning outcomes and students perceptions. Surveys conducted at the beginning and end of the course were analyzed using the Technological Acceptance Model (TAM). Results indicate that the students were highly motivated and found the modules very enjoyable. As a consequence we observed a significant increase in the retention rate of this course.

***Keywords*— Arduino; Physical Computing; CS1; Introductory Programming; Novice Programmer**

I. Introduction

Lecturers in charge of an introductory programming course face a complex challenge [1]. Students must master language syntax, programming theory and problem solving techniques in a short period of time, something they usually struggle to do. In addition students often consider the subject to be unrelated to their core interests and feel uncomfortable when learning to program [2].

As a consequence introductory programming courses show poor results and high dropout rates [3], [4]. Quoting Carter and Jenkins [5]:

"*Few teachers of programming in higher education would claim that all their students reach a reasonable standard of competence by graduation. Indeed, most would confess that an alarmingly large proportion of graduates are unable to 'program' in any meaningful sense.*"

Several studies suggest that new teaching methodologies might help the student to overcome their initial difficulties and improve their results [6],[7]. Among these approaches the use of contextualized computing education is one of the most promising.

Contextualized computing education is defined as the use of a consistent application or domain area, which effectively covers the core areas of a computer science course [8]. Examples of contexts for introductory computer science include Media Computation [9], traditional manipulatives [10], and robotics [11]. These methods are not completely new: mathematics teachers have used similar methods for decades; physical objects are used in the teaching of mathematics since the beginning of the last century [12].

Students find contextualized approaches to learning introductory CS very attractive. Instead of writing an abstract program, students can learn about basic programs by programming a robot to exit a maze, animating a story, or creating light symphonies.

Among the paradigms that belong to the contextualized approach the physical computing paradigm has attracted increased attention in the last years. This paradigm takes the computational concepts "out of the screen" and into the real world so that the student can interact with them [13]. Resnick proposed a similar concept: "Digital manipulatives" [14], tangible objects with some computational capabilities.

Several studies have analyzed the feasibility of using physical computing principles in the teaching of computer programming, see for example [15]. However most of these proposals are based on using robots to teach programming and require students to possess design skills not always available.

One alternative approach –used in this study- is to develop small and simple systems capable of display interesting behaviors using electronic boards. This approach presents several advantages: the systems are simpler and easier to understand, they are more reliable, show more reproducible behaviors and the overall cost is lower.

The present study has two aims: (1) to design and implement several learning modules for an introductory programming course using the physical computing paradigm and (2) to evaluate these modules when taught to university students. We develop these learning modules on an open hardware platform -Arduino [16]- which is widely available.

II. Material and Methods

A. *Learning modules design*

In this study we have developed several learning modules for an introductory programming course at the university level. These modules can be used to teach C/C++, or MATLAB covering both compiled languages and interpreted ones. Different course approaches and teaching methodologies might benefit from their use.

We designed specific modules for lecture demonstrations and for laboratory sessions. In the laboratory sessions students had some hands-on time with the physical computing system. Several studies show that if, due to cost or time constraints, students are forced to rely only on teacher demonstrations, the benefits of using these systems are reduced [17].

The physical computing modules are designed to enhance the traditional teaching methodology, not replacing it. Lecturers will explain a computational concept using the traditional methodology and afterwards will reinforce it using the Arduino modules.

When designing these modules one of the first decisions was whether to use an electronic board or a robotic platform. Both present several advantages and disadvantages. Electronic board projects are inexpensive but it is time consuming to acquire and organize the electronic components. Robotic platforms, on the other hand, are easier to start with but more expensive.

We decided to work with an electronic board because robotic platforms are more complex and, therefore, their behavior is less reproducible in an educational laboratory [11]. Alvarez and Larranaga [18] found out that factors related to surface friction, battery load or light conditions affected significantly the behavior of the robot and hindered the students' work. We heeded McGill's warning that "*potential technical problems should be seriously considered since they can easily negate any potential positive motivational effect*" [19].

We have selected the Arduino microcontroller board [16] as the development platform. Arduino is an open hardware board that is becoming increasingly common within the teaching community [20]. A wide variety of developers have selected it as a development platform for all kinds of computational systems.

Arduino presents several advantages for our project. The creators of Arduino designed a very easy to use board: their main targets were artists and designers. Also, thanks to its open-source nature, it is supported by a vast user community who share their ideas, projects and solutions ranging from small science fair projects to full- scale robotics [21].

The contents of the lecture demonstrations and the laboratory sessions are directly related. It is our experience that lecture demonstrations create a desire to learn more about the inner workings of the system shown. We can take advantage of this interest if students find similar activities during the laboratory sessions.

Lecture demonstrations show physical examples of computational concepts. To engage the student we used LEDs of various colors, loudspeakers to generate melodies and servo motors to link movements with different programming elements. Lecture demonstrations use different perceptive elements –light, sound and movement– to reach a broader audience. It's been shown that the use of diverse perceptive paths enhances the student understanding [22].

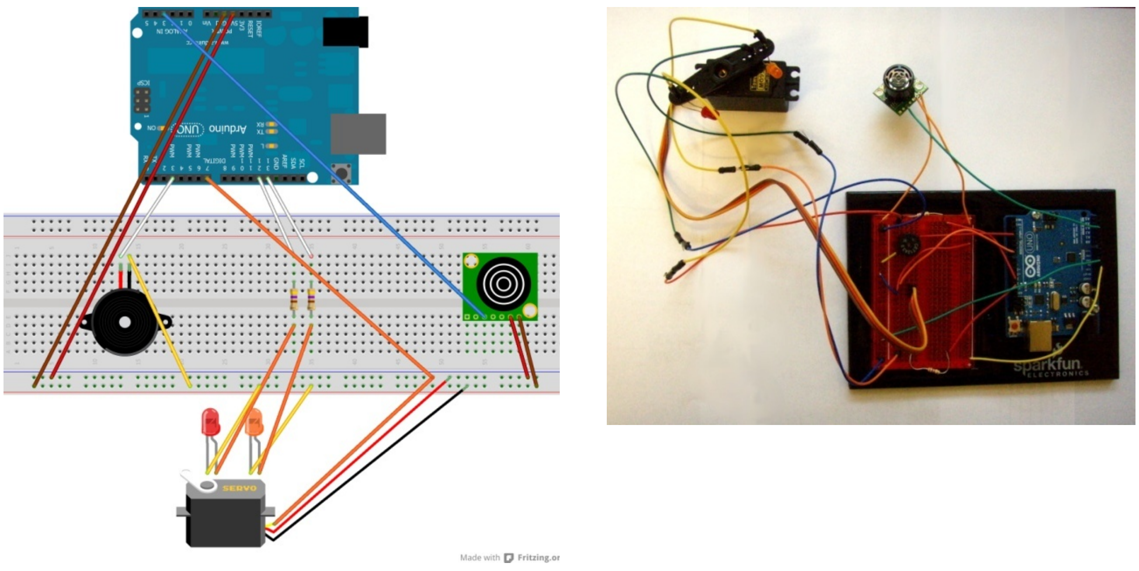


Fig. 1 Electronic circuit used in lecture demonstrations: design (left) and implementation (right). The design shows a piezoelectric loudspeaker (left), a servo motor with LEDs (center) and an ultrasonic sensor (right).

The lecture demonstrations can be performed using only two different electronic circuits, one of them can be seen in Fig. 1. That way we reduce the burden on the lecturers as the only have to mount two different protoboards at the beginning of the course. We also developed all the software code needed to perform the demonstrations.

Lecture demonstrations were conducted following Mazur's suggestions [23]. A brief description of selected demonstrations follows[1]:

- We use the loudspeaker to teach arrays. We associate different arrays to different melodies. That way we can explore concepts as arrays concatenation or the difference between the position and the value of an array.
- Conditional structures are illustrated using a photocell and LEDs. We write in the classroom a small program that will light a variable number of LEDs depending on the light conditions.
- Loop concepts are reinforced using the ultrasonic sensor and the servo motor. During the lecture we implement a program that will continuously read from the proximity sensor. When the value drops below a certain threshold the servo motor and the associated LEDs are activated.

The laboratory session modules aim to link the Arduino demonstrations to the laboratory activities. Laboratory modules are based on the lecture demonstrations, that way we can take advantage of students' curiosity. For example, one activity asks the student to create a light pattern similar to those seen in science fiction television shows. Another involves the use of a temperature sensor, converting the temperature measurement from decimal to binary and finally showing it using LEDs. The laboratory modules use only one protoboard design to simplify the staff work.

After completing the design process we piloted the first version of the learning modules in an introductory programming course. We obtained extensive feedback from the students and modified the design accordingly.

B. Evaluation of learning modules

In our study we assessed whether the Arduino modules were effective in enhancing our students learning outcomes and their perception on programming.

To this end we compared the results obtained in two introductory programming courses in a biology degree. In this course students learn to write basic programs using MATLAB. The learning modules were randomly assigned to one group and the other was designated as the control.

We decided to restrict our analysis to data obtained from 18 years old freshmen students without any previous programming knowledge. Gomes et al [24] found -in a study on students' behaviors and attitudes towards learning to program- that freshmen and repeaters differed on personal perceptions, organization and learning approaches. Restricting our analysis to freshmen without previous programming knowledge we expect to obtain a clearer picture of the impact of the physical computing learning modules on our target population.

The students were assigned into one of the two groups – control and experimental- either by the university administrative staff or by online registration based on student schedule availability only. In the control group there were 39 students that fulfilled our criteria, in the experimental group the number of students included in the study was 38.

In the control group the teacher used traditional methods; PowerPoint slides and multimedia material were used to introduce theoretical concepts. The students would also discuss some generic examples using peer instruction techniques [25].

In the experimental group he enhanced these methods using the Arduino modules. The same teacher taught both courses in the same semester. Lesson plans and a course diary were used to guarantee both courses comparability.

When designing the learning outcomes analysis we looked for a standardized assessment tool. Unfortunately, there remains a notable lack of easily accessible and validated assessment tools in introductory programming [26]. Although some new assessment tools have been developed in the last years none was applicable to our study.

Tew and Guzdial [27] have developed and validated a standardized exam -the FCS1- that can be used with different programming languages and methodologies. In our study we could not use this test because it is not applicable to courses using contextualized computing methods. Another promising line of work is the development of concept inventories for introductory programming [28] but these inventories are not available yet.

We measured the students' learning achievements by means of an exam testing their programming knowledge and skills. The assessment was performed following the guidelines proposed by the BRACElet project [29] based on the SOLO taxonomy [30]. This taxonomy has been validated as a reliable tool to assess introductory programming exams [31]

We were also interested in the students' perception on programming. Several studies have found within STEM disciplines a relationship between student perceptions and learning outcomes [32].

There are different options to measure students' attitudes in introductory programming. The Computing Attitudes Survey (CAS) is a newly designed instrument developed by Tew et al. [33]. It focuses in the differences in perceptions between novices and experts programmers. We did not use it in our study because we are teaching programming to non-majors and we do not expect them to reach the expert level.

One available instrument that has been validated for non-majors is the survey developed by Hoegh et al [34]. This survey focuses on high level perceptions and constructs on the computer sciences field but we were interested only in attitudes towards programming.

[1] A more detailed description of the modules can be found at `http://wdb.ugr.es/~marubio/?page_id=532`

We have used the TAM model [35] to evaluate students' perception on programming. The TAM model is a powerful tool commonly used to predict the acceptance, adoption and real use of new technologies in production environments. In our case we were interested in assessing whether students had the intention to use programming in the future.

Pejcinovic et al [36] have shown that a significant percentage of students that learn to program in their first university year do not use this knowledge during their studies. Using the TAM model we can estimate the future use of the technology –computer programming in our case- from the user perceptions.

In the TAM model (Fig. 2) the parameters measured are the perception of usefulness, perception of ease of use and the behavioral intention to use. These are defined by Davis [35] as:

a) Perceived usefulness: the degree to which an individual believes that using a particular system would enhance his or her job performance.

b) Perceived ease of use: the degree to which an individual believes that using a particular system would be free of physical and mental effort.

c) Behavioral intention to use: the degree to which an individual has formulated plans to use a certain system in the future.

The TAM model has been used to estimate the future use of a wide variety of new innovation in information technology. Several studies have applied this model in educational environments [37], [38]. The TAM model has been also validated in meta-analysis that involved dozens of studies [39], [40].

Using the TAM model we conducted several surveys at the beginning, midterm and end of the semester. These surveys were anonymous to reduce any bias in students' answers that may occur if they believed their answers would affect their course grade.

These surveys contained questions about the students' attitude towards programming. In the experimental group we also collected the opinion towards Arduino after the students used it in the lab. Both surveys used a Likert scale to collect the students' opinion.

III. Results

We assessed the learning modules evaluating the learning outcomes and the perceptions of the students involved in the study. Learning outcomes were estimated from the results of the final exam and several surveys were conducted to obtain the students perceptions throughout the course.

A. Learning outcomes

We used the final exam to assess students learning in the control and the experimental group. Both groups completed the same exam at the same time. This allowed us to make a straightforward comparison.

As we restricted our analysis to students without any previous programming knowledge both the experimental and control group were equivalent at the beginning of the course.

The main difference found is related to the percentage of the students that did not pass the programming exam, the failure rate. In Fig. 3 we can observe that the control group had a significantly higher failure rate, 25.6%, than the experimental group, 7.9%.

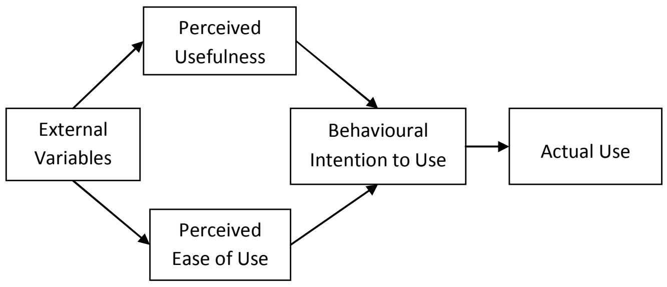


Fig. 2 Original TAM model proposed by Davis [35]

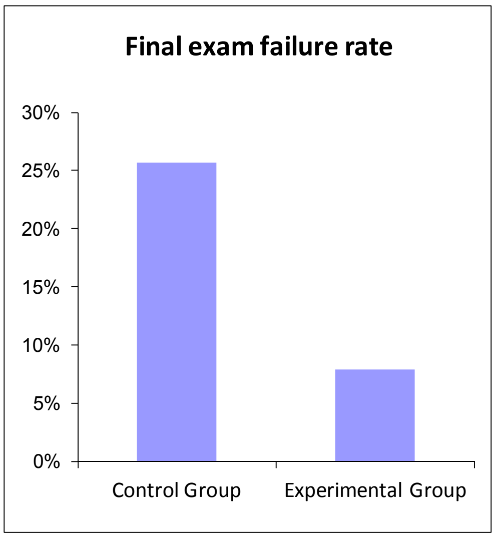


Fig. 3 Percentage of students that failed the programming final exam. The control group has a significantly higher failure rate than the experimental group. The difference is significant at the 10% level.

TABLE I. FINAL EXAM RESULTS

	Control Group	*Experimental Group*	*p-value*
Mean Grade	5.7	5.9	0.7114
Students that failed	10	3	0.0654*
High achieving students	10	7	0.6249

* result significant at the 10% level

TABLE II. STUDENTS PERCEPTION ON ARDUINO

	Mean score. 1(min)-7(max) scale
Perceived Usefulness	5.58
Perceived Ease of Use	4.80
Perceived Enjoyment	6.31

Related measurements can be found in Table 1. We collect there the number of students that failed, the number of high achieving students (those obtaining a mark over 90%) and the mean grade for both groups. P-values were obtained running the Welch Two Sample t-test for the mean grade, the Fisher's Exact Test for Count Data for the number of students that failed and 2-sample test for equality of proportions for the number of high achieving students.

The exam's mean grade has similar values in both groups with no significant differences. If we look at the number of high achieving students the control group has a larger number of high achieving students than the experimental group, although the difference is not significant. From these results we can conclude that the control group has a more extreme distribution –more students that fail or are high achievers- than the experimental group.

B. *Students' perception on Arduino*

We conducted a survey on the experimental group asking students about their opinion on the Arduino platform. Results –shown in Table 2- were very positive. The perceived enjoyment of using the platform stands out with a score of 90%.

Students' comments -both formal and informal- were very encouraging. One student wrote: "*incredibly useful lesson, now the whole course makes sense*". A more graphic comment is shown in Fig. 4.

C. *Students' perception on programming*

In our study we conducted three surveys: at the beginning of the course, at the midterm exam and at the final exam. We analyzed the survey reliability using Cronbach's alpha obtaining in all cases values over 0.7 a quality threshold widely accepted [34].

In these surveys we measured the values of the TAM model parameters in the control and the experimental group at the beginning of the course, at midterm and at the final exams. Results are shown in Fig. 5.

At the beginning of the course the experimental and control groups were equivalent: there were no significant differences in perceived usefulness, perceived ease of use or intention to use.

If we analyze the change of these parameters during the course we see that there is a statistically significant increase on the perceived ease of use of programming. In the experimental group it goes from 2.14 to 2.87 and in the control group it goes from 2.34 to 2.80. That indicates that, thanks to the course, students perceive programming as an easier endeavor.

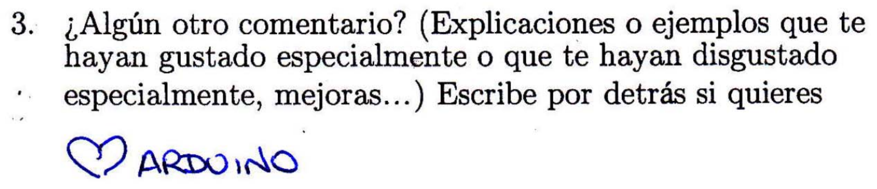
An interesting fact is that there is no significant variation of the perceived usefulness or the intention to use during the course for both groups. It seems that lectures and lab sessions do not affect the student perceptions in these areas.

There is no significant difference in the behavior of the experimental and control group parameters. Perceived usefulness and intention to use behave in a similar manner in both groups. The perceived ease of the experimental group shows a slightly higher growth rate than the control group but the difference in not statistically significant.

IV. DISCUSSION

In this study we have designed and implemented several modules to teach introductory programming. These modules comprise lecture demonstrations and laboratory sessions. They aim to enhance the traditional teaching methodology without replacing it. In the design process we have used the principles of the physical computing paradigm.

We evaluated the modules in an introductory programming course and found that they were highly effective. When



3. ¿Algún otro comentario? (Explicaciones o ejemplos que te hayan gustado especialmente o que te hayan disgustado especialmente, mejoras...) Escribe por detrás si quieres

♡ ARDUINO

Fig. 4 A comment by a student. The question reads "Any other comment? (Explanations or examples that got your attention, improvements…) Write on the backside if needed."

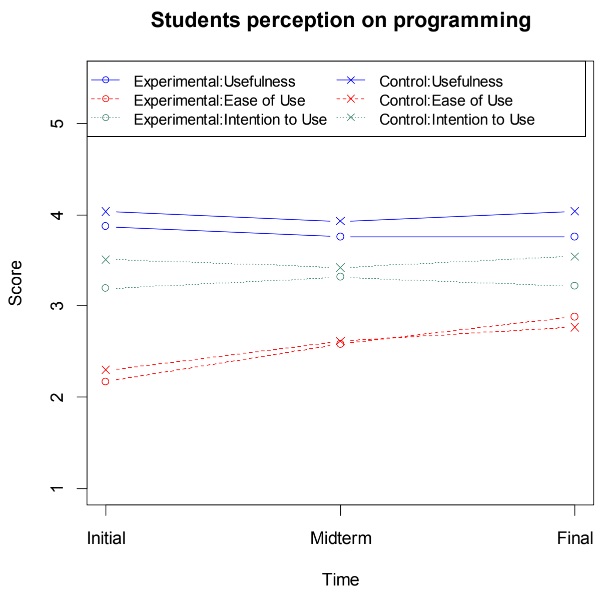


Fig. 5 Students' perception on programming. There is no significant difference between the experimental and control group. Ease of use shows a significant difference between the beginning and the end of the course in both groups.

comparing failure rates the experimental group showed a significant lower value: 8% versus 26% in the control group. No significant difference was observed in the mean grades or in the number of high achieving students.

This is consistent with results obtained in other studies [41]. Guzdial [42] applied the contextualized approach in his media computing class and also found that the retention rate increased significantly but the best students did not learn more. The contextualized approach was able to increase the number of students able to complete the course but students learned the same as in the traditional setting.

In that sense it seems that this approach helps the students to get involved in the subject, but does not constitute a cognitive help. We agree with Guzdial that these results probably show that the new approach is effective because, thanks to it, more students find the subject relevant.

Students found the learning modules useful and highly enjoyable and found reasonable the effort necessary to complete the lab sessions. They perceived them as a valuable learning experience. Students stated that more laboratory sessions should be devoted to this kind of experiences.

We did not find any significant effect on the students' perception on programming: both the experimental and control group showed similar responses. We were surprised to find out that a semester course had a negligible effect on the students' perception on the usefulness of programming although Gomes et al [24] also obtained in their study that students' attitudes show very little change during a semester course.

We did find a significant signal in the evolution of students' perceptions on programming difficulty. Both the experimental and control group students thought that programming was easier at the end of the semester. This result deserves a more thorough study to clarify what is the actual effect of an introductory programming course on students' attitudes.

Our study has several limitations. We have only used these modules for one year in one degree, biology. We believe that the results will be similar in the empirical sciences and engineering disciplines. In more formal fields, like mathematics and statistics, these modules might be less effective. We plan to extend this study to these disciplines and compare the results obtained with those obtained till now.

Another limitation is the fact that the teacher in charge of both courses was involved in the developing of the learning modules. These modules might be less effective when used by teachers that are less familiar with the material.

One future line of work is to adapt these learning materials to other programming languages. We are interested in including an open-source interactive language. MATLAB is a very powerful platform but the fact that is proprietary hampers its development in academic environments. Using open interactive environments like iPython [43] would increase these modules usefulness.

V. CONCLUSIONS

We have developed an introductory programming teaching resource that enhances students learning. These modules follow the principles of the physical computing paradigm using the Arduino board as the physical platform. These modules can be used to teach C/C++ and MATLAB.

We have used them in an introductory programming course and compared the results with those obtained in a traditional course. We found that when using these modules the students' failure rate decreased significantly. Surprisingly the mean grade and the number of high achieving students did not change significantly. This suggests that although a higher proportion of students got involved in the subject they did not learn more than with the traditional approach.

Students found the learning modules useful and fun. They expressed visibly their satisfaction and asked for more sessions using the physical computing approach. Nonetheless, the students' perceptions on programming were only slightly affected by these modules. Both groups showed the same behavior in terms of the programming usefulness, ease of use and intention to program in the future.

Teaching introductory programming to university students is a challenge: students perceive the subject to be difficult and boring and feel uncomfortable during the course. The application of the physical computing paradigm engages students more effectively and increases the proportion of students that learn effectively.

Acknowledgment

This work is supported by the University of Granada (PID/13-54). The authors wish to thank the students of the Biology degree at the UGR that participated in the experience.

References

[1] A. Robins, J. Rountree, and N. Rountree, "Learning and teaching programming: A review and discussion," *Computer Science Education*, vol. 13, no. 2, pp. 137–172, 2003.

[2] H. Qin, "Teaching computational thinking through bioinformatics to biology students," in *ACM SIGCSE Bulletin*, 2009, vol. 41, no. 1, pp. 188–191.

[3] M. McCracken, V. Almstrum, D. Diaz, M. Guzdial, D. Hagan, Y. B.-D. Kolikant, C. Laxer, L. Thomas, I. Utting, and T. Wilusz, "A multi-national, multi-institutional study of assessment of programming skills of first-year CS students," *ACM SIGCSE Bulletin*, vol. 33, no. 4, pp. 125–180, 2001.

[4] R. Lister, E. S. Adams, S. Fitzgerald, W. Fone, J. Hamer, M. Lindholm, R. McCartney, J. E. Mostr□m, K. Sanders, O. Seppälä, and others, "A multi-national study of reading and tracing skills in novice programmers," *ACM SIGCSE Bulletin*, vol. 36, no. 4, pp. 119–150, 2004.

[5] J. Carter and T. Jenkins, "Gender and programming: What's going on?," in *ACM SIGCSE Bulletin*, 1999, vol. 31, no. 3, pp. 1–4.

[6] J. Wells, R. M. Barry, and A. Spence, "Using Video Tutorials as a Carrot-and-Stick Approach to Learning," *IEEE Transactions on Education*, vol. 55, pp. 453–458, 2012.

[7] K. E. Merrick, "An Empirical Evaluation of Puzzle-Based Learning as an Interest Approach for Teaching Introductory Computer Science," *IEEE Transactions on Education*, vol. 53, pp. 677–680, 2010.

[8] M. Guzdial, "Does contextualized computing education help?," *ACM Inroads*, vol. 1, no. 4, pp. 4–6, 2010.

[9] M. Guzdial, "A media computation course for non-majors," in *ACM SIGCSE Bulletin*, 2003, vol. 35, no. 3, pp. 104–108.

[10] "Computer Science Unplugged," 2014. [Online]. Available: http://csunplugged.org. [Accessed: Apr-2014].

[11] M. Cuéllar and M. Pegalajar, "Design and implementation of intelligent systems with LEGO Mindstorms for undergraduate computer engineers," *Computer Applications in Engineering Education*, 2011.

[12] P. S. Moyer, "Are we having fun yet? How teachers use manipulatives to teach mathematics," *Educational Studies in Mathematics*, vol. 47, no. 2, pp. 175–197, 2001.

[13] G. T. Richard, "Employing Physical Computing in Education: How Teachers and Students Utilized Physical Computing to Develop Embodied and Tangible Learning Objects," *The International Journal of Technology, Knowledge and Society*, 2010.

[14] M. Resnick, F. Martin, R. Berg, R. Borovoy, V. Colella, K. Kramer, and B. Silverman, "Digital manipulatives: new toys to think with," in *Proceedings of the SIGCHI conference on Human factors in computing systems*, 1998, pp. 281–287.

[15] A. Ruthmann, J. M. Heines, G. R. Greher, P. Laidler, and C. Saulters II, "Teaching computational thinking through musical live coding in scratch," in *Proceedings of the 41st ACM technical symposium on Computer science education*, 2010, pp. 351–355.

[16] M. Banzi, *Getting Started with arduino*. Make, 2009.

[17] J. S. Kay, "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*, 2011, pp. 177–182.

[18] A. Alvarez and M. Larranaga, "Using LEGO mindstorms to engage students on algorithm design," in *Frontiers in Education Conference, 2013 IEEE*, 2013, pp. 1346–1351.

[19] M. M. McGill, "Learning to program with personal robots: Influences on student motivation," *ACM Transactions on Computing Education (TOCE)*, vol. 12, no. 1, p. 4, 2012.

[20] J. Grasel, W. Vonnegut, and Z. Dodds, "Bitwise Biology: Crossdisciplinary Physical Computing Atop the Arduino," in *2010 AAAI Spring Symposium Series*, 2010.

[21] L. Hill and S. Ciccarelli, "Using a low-cost open source hardware development platform in teaching young students programming skills," in *Proceedings of the 13th annual ACM SIGITE conference on Information technology education*, 2013, pp. 63–68.

[22] S. Ainsworth, "The functions of multiple representations," *Computers & Education*, vol. 33, no. 2, pp. 131–152, 1999.

[23] C. Crouch, A. P. Fagen, J. P. Callan, and E. Mazur, "Classroom demonstrations: Learning tools or entertainment?," *American Journal of Physics*, vol. 72, p. 835, 2004.

[24] A. J. Gomes, A. N. Santos, and A. J. Mendes, "A study on students' behaviours and attitudes towards learning to program," in *Proceedings of the 17th ACM annual conference on Innovation and technology in computer science education*, 2012, pp. 132–137.

[25] C. H. Crouch and E. Mazur, "Peer instruction: Ten years of experience and results," *American Journal of Physics*, vol. 69, p. 970, 2001.

[26] A. E. Tew, "Assessing Fundamental Introductory Computing Concept Knowledge in a Language Independent Manner," Georgia Institute of Technology, Atlanta, GA, USA, 2010.

[27] A. E. Tew and M. Guzdial, "The FCS1: A Language Independent Assessment of CS1 Knowledge," in *Proceedings of the 42Nd ACM Technical Symposium on Computer Science Education*, 2011, pp. 111–116.

[28] K. Goldman, P. Gross, C. Heeren, G. L. Herman, L. Kaczmarczyk, M. C. Loui, and C. Zilles, "Setting the Scope of Concept Inventories for Introductory Computing Subjects," *Trans. Comput. Educ.*, vol. 10, no. 2, pp. 5:1–5:29, 2010.

[29] R. Lister, T. Clear, D. J. Bouvier, P. Carter, A. Eckerdal, J. Jacková, M. Lopez, R. McCartney, P. Robbins, O. Seppälä, and others, "Naturally occurring data as research instrument: analyzing examination responses to study the novice programmer," *ACM SIGCSE Bulletin*, vol. 41, no. 4, pp. 156–173, 2010.

[30] R. Lister, B. Simon, E. Thompson, J. L. Whalley, and C. Prasad, "Not seeing the forest for the trees: novice programmers and the SOLO taxonomy," in *ACM SIGCSE Bulletin*, 2006, vol. 38, no. 3, pp. 118–122.

[31] T. Clear, J. Whalley, R. Lister, A. Carbone, M. Hu, J. Sheard, B. Simon, and E. Thompson, "Reliably classifying novice programmer exam response using the SOLO taxonomy," in *21st Annual NACCQ Conference, NACCQ, Auckland, New Zealand*, 2008, pp. 23–30.

[32] J. C. Valentine, D. L. DuBois, and H. Cooper, "The relation between self-beliefs and academic achievement: A meta-analytic review," *Educational Psychologist*, vol. 39, no. 2, pp. 111–133, 2004.

[33] A. E. Tew, B. Dorn, and O. Schneider, "Toward a Validated Computing Attitudes Survey," in *Proceedings of the Ninth Annual International Conference on International Computing Education Research*, 2012, pp. 135–142.

[34] A. Hoegh and B. M. Moskal, "Examining science and engineering students' attitudes toward computer science," in *Frontiers in Education Conference, 2009. FIE'09. 39th IEEE*, 2009, pp. 1–6.

[35] F. Davis, "User acceptance of information technology: system characteristics, user perceptions and behavioral impacts," *International journal of man-machine studies*, vol. 38, no. 3, pp. 475–487, 1993.

[36] B. Pejcinovic, M. Holtzman, M. Chrzanowska-Jeske, and P. K. Wong, "Just because we teach it does not mean they use it: Case of programming skills," in *Frontiers in Education Conference, 2013 IEEE*, 2013, pp. 1287–1289.

[37] H. M. Selim, "An empirical investigation of student acceptance of course websites," *Computers & Education*, vol. 40, no. 4, pp. 343–360, 2003.

[38] I.-F. Liu, M. C. Chen, Y. S. Sun, D. Wible, and C.-H. Kuo, "Extending the TAM model to explore the factors that affect Intention to Use an Online Learning Community," *Computers & Education*, vol. 54, no. 2, pp. 600–610, 2010.

[39] M. Turner, B. Kitchenham, P. Brereton, S. Charters, and D. Budgen, “Does the technology acceptance model predict actual use? A systematic literature review,” *Information and Software Technology*, vol. 52, no. 5, pp. 463–479, 2010.

[40] W. R. King and J. He, “A meta-analysis of the technology acceptance model,” *Information & Management*, vol. 43, no. 6, pp. 740–755, 2006.

[41] L. Porter, M. Guzdial, C. McDowell, and B. Simon, “Success in introductory programming: what works?,” *Communications of the ACM*, vol. 56, no. 8, pp. 34–36, 2013.

[42] M. Guzdial, “Exploring hypotheses about media computation,” in *Proceedings of the ninth annual international ACM conference on International computing education research*, 2013, pp. 19–26.

[43] F. Perez and B. E. Granger, “IPython: a system for interactive scientific computing,” *Computing in Science & Engineering*, vol. 9, no. 3, pp. 21–29, 2007.