ISBN: 978-989-8533-71-5 © 2017

USING ARDUINO TO TEACH PROGRAMMING TO FIRST-YEAR COMPUTER SCIENCE STUDENTS

Wee Lum Tan, Sven Venema and Ruben Gonzalez
School of ICT, Griffith University
Gold Coast, Queensland, Australia

ABSTRACT

Transitioning to university is recognised as a challenging endeavour for commencing students. For commencing Computer Science students specifically, evidence suggests a link between poor performance in introductory technical courses, such as programming, and high attrition rates. Building resilience in students, particularly at the start of their academic journey, can potentially reduce the likelihood of student attrition. One way to develop resilience is through strong disciplinary engagement. We aim to increase student engagement by using popular low cost embedded platforms that provide a physical environment for novice programmers. This physical programming environment allows novice programmers to better appreciate the broad application of programming to everyday objects. Several experiments with the physical computing devices were designed and implemented for a first-year Computer Science programming course. These experiments were evaluated through a survey. Results indicate that whilst students found the environment more challenging than the normal computer-based environment, they felt more engaged in the programming process and enjoyed seeing the practical applications of hardware programming.

KEYWORDS

Programming Concepts, Computer Science, Physical Computing, Arduino

1. INTRODUCTION

Transitioning to university is a challenging endeavour [Baik, Naylor and Arkoudis, 2015]. Lizzio's Five Senses of Success model [Lizzio 2006; Lizzio and Wilson 2010] outlines a number of aspects that can contribute to a successful transition. One of these indicators is the level of disciplinary engagement. Improving disciplinary engagement can build resilience in students and potentially reduce the likelihood of student attrition. Building resilience is particularly important at the start of one's academic studies.

There is ongoing evidence that commencing Computer Science students struggle with technical courses such as programming, and that poor performance in such courses is linked with high attrition rates [Watson and Li, 2014]. We posit that part of the challenge for students in these technical introductory programming courses is low disciplinary engagement. Some factors involved in this could be due to the abstract and inscrutable nature of the programming environment. A student cannot see inside the computer to understand how their program is executing and there are limited ways to link the behaviour of the program with its output [Milne and Rowe, 2002]. This becomes somewhat less problematic as the student's experience grows, but it can be a significant hurdle and source of frustration for the novice programmer and can lead to diminished engagement.

There is growing evidence [Richard 2010, Rubio et al. 2014, Maia et al. 2009] that programming in a physical environment using some form of physical computing modules is beneficial to the novice programmer. There is a direct connection between the program as written and the visible behaviour of the physical devices. The behaviour of the device (e.g. very high pitch tone or the motor moved too far to the left) can provide better, visual and auditory feedback to the novice programmer on the correctness of their program. This can alleviate some of the frustrations experienced by novice programmers.

The approach taken in this paper is to focus on improving disciplinary engagement, in this case programming in Computer Science. We aim to increase student engagement by using popular low cost embedded platforms that are the backbone of the "Internet of Things". The use of embedded platforms helps

student appreciate the broad application of programming to everyday objects and gives them a concrete focus for practical work that they can take home and integrate into their own study environments. Students will solve real world problems of a physical nature using the same technology used in everyday "smart" appliances. Students will be able to physically observe the outcomes of their solutions to problems, rather than as a text printout on a screen. They will also be able to reflect on the effectiveness of their solutions and will be able to experiment to obtain alternate solutions and evaluate the effectiveness of these in comparison to other potential solutions.

We have designed several experiments with the Arduino physical computing devices [Arduino] for a first-year Computer Science programming course. Despite students finding the environment more challenging than the normal computer-based environment, student feedback (via a survey) strongly indicates that students felt more engaged and appreciated the hands-on practical experience of hardware programming.

The remainder of this paper is organised as follows. In Section 2, we describe the programming experiments that we designed on the Arduino platform. We present our survey methodology and discuss the student feedback on the experiments in Section 3, and offer our conclusions in Section 4.

2. PROGRAMMING EXPERIMENTS ON THE ARDUINO PLATFORM

Arduino is an open-source physical computing platform that has been designed to make learning electronics and programming easier for students and beginners. It consists of a software integrated development environment (IDE) that allows users to write programs in the C/C++ programming languages, which are compiled, uploaded and executed on a microcontroller hardware platform (Figure 1).

In our first-year Computer Science introductory programming course, we have designed several laboratory modules with the Arduino platform to enhance the students' understanding of the basic programming concepts, and to enable them to see the practical real-world applications of the C/C++ programming languages.

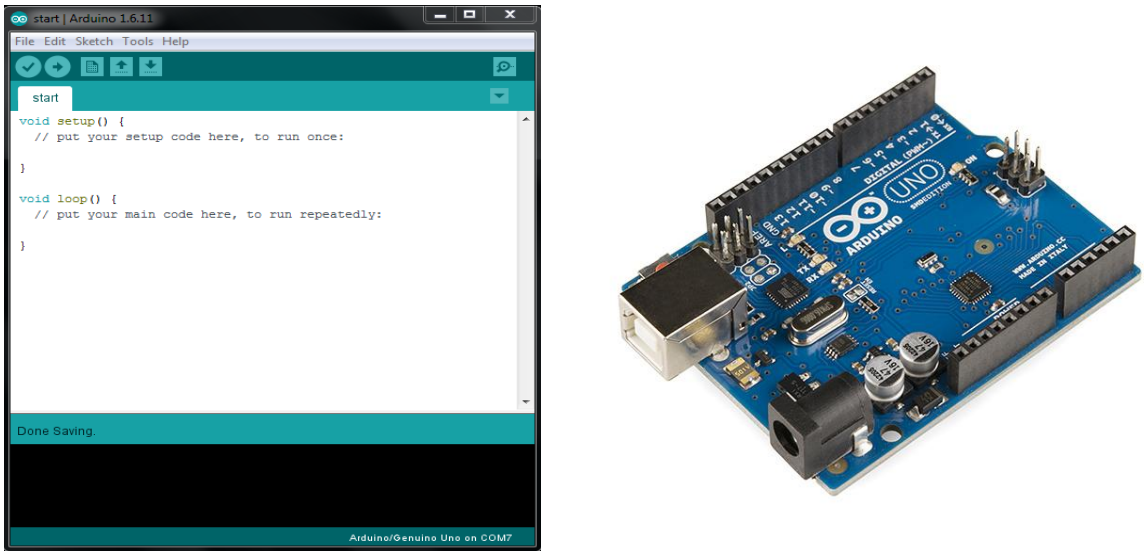

Figure 1. The Arduino Integrated Development Environment and the Arduino Hardware Platform

2.1 Utilising a Digital Compass to Control the Servo Motor's Rotation

This laboratory module consists of two sub-experiments (Figure 2):

In the first sub-experiment, students are directed to connect the servo motor to the Arduino board, and provided with a simple program that demonstrates how the servo motor can be set to point to a specified position (or angle). Students are then asked to extend the simple program to make the servo motor point to a series of different angles, using an array. This experiment teaches students the programming applications of arrays of numbers as well as some simple arithmetic operations.

In the second sub-experiment, students are asked to connect the digital compass to the Arduino board and provided with a simple program that shows how the Arduino board reads the data of the heading direction computed by the digital compass. Students are asked to convert the heading direction reading from radians to degrees, and then feed the reading into the servo motor via the Arduino board. Therefore, the servo motor will point to a specific angle corresponding to the digital compass' reading. Students are also asked to group the program code corresponding to the servo motor and digital compass into separate functions, thereby giving them the opportunity to practise calling and returning values from functions.

2.2 Generating Melodies with Sound Buzzer

In this laboratory module, students are given a sound buzzer connected to the Arduino board (Figure 3), and a program that demonstrates how the sound buzzer can be used to produce tones at particular frequencies. Students are then asked to extend the given program to play a melody (tune), with the students also given a list of notes and their corresponding frequencies. In writing the program to play a melody, students get to practise and apply their knowledge of the basic programming constructs such as selections, loops, and functions.

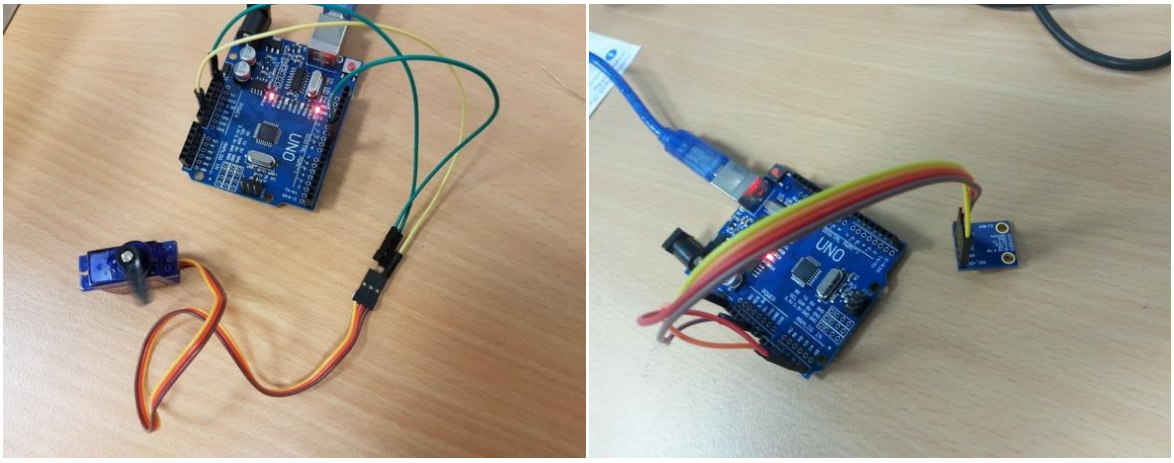

Figure 2. Servo Motor & Digital Compass

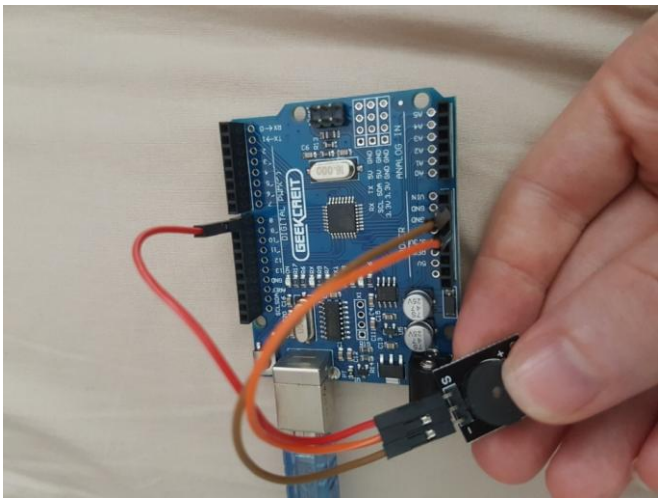

Figure 3. Sound Buzzer

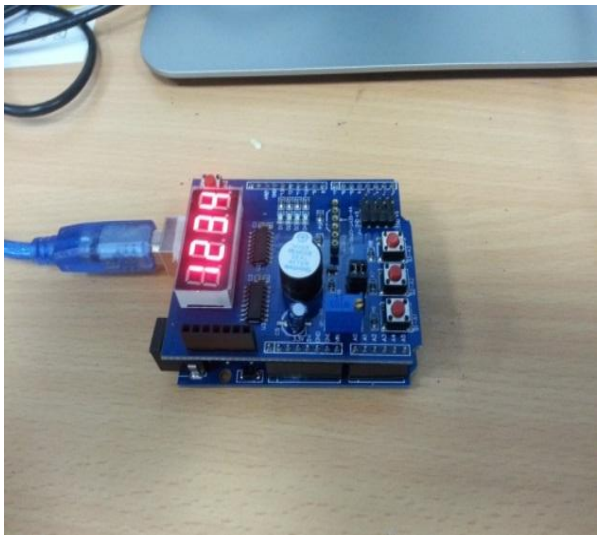

Figure 4. Digital Multi-Function Shield

2.3 Click Counter with Digital Multi-Function Shield

In this laboratory module, students are given a digital multi-function shield attached to the Arduino board (Figure 4). The multi-function shield has both a seven-segment display and several buttons. Two simple programs are given to the students, demonstrating how to detect button presses and how to drive the seven-segment display. Students are then asked to write a program that implements a counter, with the button presses as input. Essentially, the seven-segment display will show the value of the counter, with the value incrementing from 0 to 100 for each button press, and then looping back to 0 again. This exercise gives the students opportunities to write a program that utilises functions, the selection construct, arrays, and the modulo/division arithmetic operations.

3. METHODOLOGY

In our first-year Computer Science introductory programming course, students attend weekly two-hour lectures and are given weekly homework programming exercises. The Arduino laboratory experiments described in Section 2 were offered to the students during the weekly two-hour laboratories (after having been introduced to the basic programming constructs), and are designed to enhance the students' learning experience and engagement with the course. These Arduino labs were optional and were not assessed. During the lab sessions, both the lecturer and tutor would provide one-on-one assistance to the students, giving hints and tips on how to solve the programming problems.

In this study, our objective is to determine the effectiveness of the Arduino experiments on the students' understanding and application of the basic programming concepts, and hence their engagement with the course. We developed a set of survey questions, which were distributed to the students during the laboratory sessions in the last week of the semester. The survey questions asked the students to grade their experience with the Arduino experiments on a 4-point scale from *Strongly Disagree* (1) to *Strongly Agree* (4). Student responses were completely anonymous. For the survey, the cohort size is 70 with n = 27 students participating (38.6% response rate). It should be noted that since the survey was conducted in the last week of semester in laboratory sessions with no summative assessment, student attendance was well below average for that week's sessions.

4. RESULTS AND DISCUSSION

Student feedback has been encouraging, as seen from their responses to the survey questions. The questions are listed in Table 1, while the student responses are tabulated and shown in Figure 5. Some student comments are also shown in Table 2. These student comments are specifically related to the Arduino experiments, and were obtained from a separate end-of-semester course evaluation survey that is run by the university.

Figure 5 shows that while the students found the programming exercises on the Arduino platform to be more challenging compared to programming on the normal computer-based environment (Q1 and Q2), they appreciated the practical aspects of the Arduino programming exercises and were more engaged in the process (Q3, Q4, and Q5). Students found the Arduino programming exercises to be more interesting and easier to troubleshoot as they were able to easily see (or hear) the "outputs" from their programs. For example, during the "Generating Melodies with Sound Buzzer" experiment, students enjoyed troubleshooting their code to generate an actual melody from the initial screeching tone that their programs generated. Student comments indicated that they had fun doing the Arduino experiments, and they were interested to further explore the capabilities of programming the Arduino hardware.

Table 1. Survey Questions on the Arduino Experiments

Questions	Description
Q1	The Arduino programming was easier than the normal programming
Q2	I was more confident in Arduino programming than in normal programming
Q3	The Arduino programming helped me to explore the practical uses of C programming in real world applications
Q4	The Arduino labs were interesting
Q5	I want to learn more about Arduino programming

Table 2. Student Comments on the Arduino Experiments

Student	Comments
1	I appreciate the whole idea to get a hands on approach to programming hardware
2	Engaging and interesting content, which I am excited to explore further
3	Arduino tutorials were fun
4	I enjoyed the Arduino

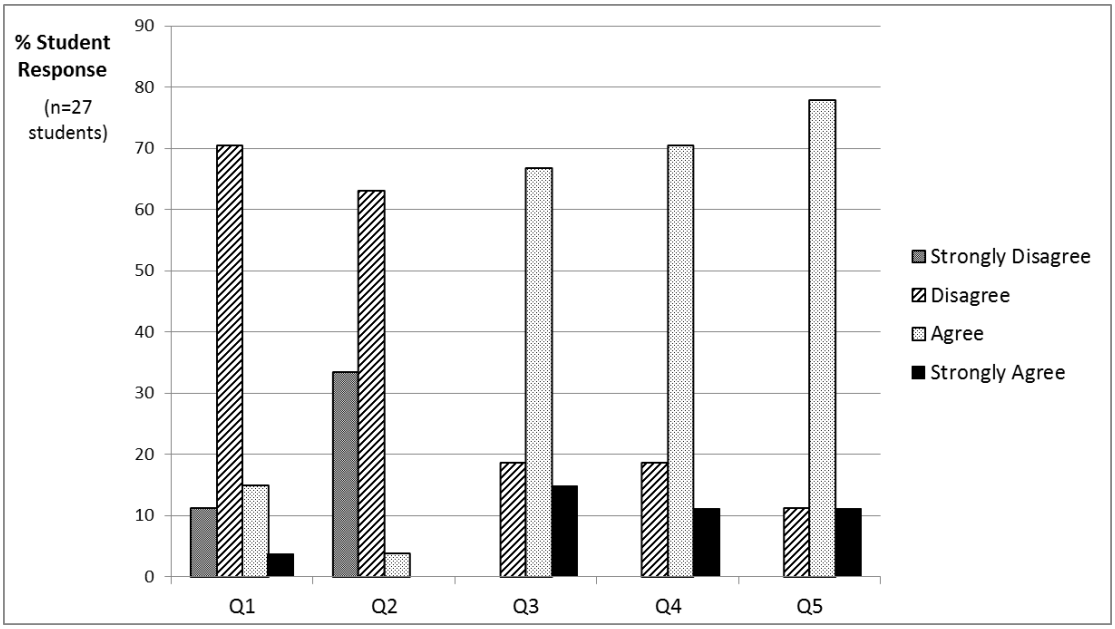

Figure 5. Student Responses to Arduino Experiments

5. CONCLUSIONS

We have designed several programming experiments with the Arduino physical computing devices and offered them to students in a first-year Computer Science introductory programming course. The aims are to enhance the students' understanding of the basic programming concepts, and to allow them to see the practical real-world applications of C/C++ programming languages. Student feedback indicates that while they found the programming exercises to be challenging, they were more engaged in the process and were appreciative of the practical experience of hardware programming. This bodes well for the disciplinary engagement process since programming is a key threshold concept area that is foundational to all of Computer Science.

For our future work, we plan to provide more preparatory lecture materials on the Arduino platform to scaffold the students' initiation to the Arduino programming experiments. We will also design more Arduino experiments to give the students opportunities to write programs using more advanced programming constructs like linked lists and pointers.

REFERENCES

Arduino. Retrieved from http://www.arduino.cc

Baik, C., Naylor, R. & Arkoudis, S. (2015). The First Year Experience in Australian Universities: Findings from Two Decades, 1994 – 2014. Melbourne Centre for the Study of Higher Education (CSHE), Melbourne: University of Melbourne.

Lizzio, A. (2006). Designing an Orientation and Transition Strategy for Commencing Students: A Conceptual Summary of Research and Practice. First Year Experience Project. Griffith University

Lizzio, A. & Wilson, K. (2010). Strengthening Commencing Students' Sense of Purpose: Integrating Theory and Practice. *Proceedings of the 13th Pacific Rim First Year in Higher Education Conference.*

Maia, L. et al. (2009). An Experience to use Robotics to Improve Computer Science Learning. *Proceedings of the 39th IEEE Frontiers in Education Conference*.

Milne, I. & Rowe, G. (2002). Difficulties in Learning and Teaching Programming—Views of Students and Tutors. *In Education and Information Technologies*, vol. 7, no. 1 pp. 55-66.

Richard, G.T. (2010). Employing Physical Computing in Education: How Teachers and Students Utilized Physical Computing to Develop Embodied and Tangible Learning Objects. *In The International Journal of Technology, Knowledge and Society*, vol. 4, pp. 93-102.

Rubio, M.A. et al. (2014). Enhancing an Introductory Programming Course with Physical Computing Modules. *Proceedings of the 2014 IEEE Frontiers in Education Conference (FIE)*.

Watson, C. & Li, F.W. (2014). Failure Rates in Introductory Programming Revisited. *Proceedings of the 2014 ACM Conference on Innovation & Technology in Computer Science Education*.