The Use of Physical Artefacts in Undergraduate Computer Science Teaching

Edward Currie(✉) and Carl James-Reynolds

Department of Computer Science, Middlesex University, London, England, UK
{e.currie,c.james-reynolds}@mdx.ac.uk

Abstract. This paper describes the introduction of the use of physical artefacts in the teaching of the undergraduate curriculum in the Department of Computer Science at Middlesex University. The rationale for the change is discussed, together with a description of the various technologies and the areas in which they were deployed. We conclude with a discussion of the outcomes of the work and the conclusions reached, prime amongst which are that the policy has been successful in motivating and engaging students, with a resultant improvement in student progression.

Keywords: Physical computing · Microcontroller · Robotics

1 Introduction

The study of Computer Science has always involved a balance of abstract concepts and practical work. Students sometimes find the former difficult to grasp in isolation and in the curriculum, they are not always closely integrated with the latter. Practical computer programming courses have often involved exercises that are not related to real-world problems and are often considered by students to be rather dull. This, in turn, has tended to make programming and other problem solving tasks seem to students like a necessary evil, rather than something exciting and engaging. This can affect the amount of effort that students are willing to put into their study, which together with the incremental nature of programming, tends to result in the 'falling behind and staying behind' reported by [1, 8]. This effect has been noted in programming courses at all levels and in all countries and cultures worldwide. Those who succeed in programming and problem solving tend to be those who immerse themselves in the subject. There is evidence that the use of physical artefacts can promote the necessary level of engagement and motivation for successful study in computer science. In the rest of this paper, we describe some examples of the deployment of physical technologies within the Information Technology and Computer Science undergraduate programmes at Middlesex University and discuss some of the outcomes.

G. Vincenti et al. (Eds.): eLeot 2016, LNICST 180, pp. 119–124, 2017.
DOI: 10.1007/978-3-319-49625-2_15

2 Related Work

Much of the existing literature focuses on encouraging the uptake of STEM-based subjects and use of physical computing in schools [3]. In the UK there has been a dramatic shift in the school curriculum and a new focus on computer programming; however this is not yet reflected in the experiences of those who are entering undergraduate study. Blikstein [2] has noted that some platforms such as Arduino expose children to too much detail, at too low a level of abstraction. Kato [6] has attempted to address this by the development of visual interfaces. This does not seem to be the case at undergraduate level, where Okita's [7] work suggests that students who learn via low transparency text based programming languages not only did equally well in assessment as those students who learned in high transparency visual coding environments, but additionally were better placed to solve new problems with unfamiliar materials.

Herger and Bodarky [5] identified, in workshops with schools, that it is important to manage time and resources effectively in order to complete planned exercises and this is equally important at undergraduate level.

Cambron [4] explored the use of Arduino in a first-year robot-based project as a first experience of electronics and processors and found it "invaluable for retention purposes". Rubio et al. [9] also found value in increased retention and engagement and the number of students who learned effectively. They also identified that the mean grades and number of high performance students did not change significantly.

3 Context

In 2013, the undergraduate programmes in computing at Middlesex University underwent a revalidation. For many years, students had stuggled with the computer programming and problem-solving strand of the BSc Computer Science (CS) programme, while the BSc Information Technology (IT) programme was very much management-oriented, with minimal computer programming content. Many CS students, after struggling with the programming content in the first year of their course, would transfer to the IT programme to avoid further study of programming. We were unhappy with this role for the IT programme, and with the poor progression rate of students on the CS programme, and it was decided that the revalidated programmes should both have a strong programming and problem-solving core thread. Furthermore, they should embrace the use of physical artefacts to motivate and engage students in project-based learning. Through this, students could be exposed to concrete implementations of theoretical concepts to reinforce their understanding. We now describe some of the technologies used and the areas in which they were deployed.

4 Arduino Microcontrollers

The use of Physical Computing in the IT curriculum was introduced 5 years ago as a trial with final year students and fully integrated since the revalidation of the programme. First year undergraduates take a module that introduces concepts such as

smart homes, embedded systems, sensors and automation, personal online presence, and simple machine learning. Six weeks are dedicated to Physical Computing workshops using Arduino. Although most of the coursework is completed over a period of three weeks, it was considered to be important to give the students time to try out ideas and to encourage a sense of "playing" with the technology. The two-year trial had highlighted some of the difficulties students had with mapping schematics to breadboards and identification of components such as resistors. Students were also frustrated with the practical difficulties of rewiring breadboards at the start of each taught session. A kit was therefore developed as part of an undergraduate project [14], that allows the use of a wide range of sensors; however, these were prewired and accessed by simply patching them in using 3.5 mm jack to jack patch cables. Students were originally given a specific group challenge, but are now presented with a set of criteria to which their projects must conform; the actual project is negotiated with the tutor. As the students concurrently study a Java programming module, the challenge is not just about writing code from scratch, but the process of reuse and modification of existing code, for incorporation into the system. Through this, students gain insights into the techniques of real-world development of software projects. In practice over the last two years, approximately 40% of students have chosen to move beyond the kit to using breadboards directly and have had the technical expertise to feel comfortable with this. There have been a range of different reasons for this, but greater flexibility in developing their products seems to have been paramount among these. Students post their work on Social Media, which is an important part of maintaining a portfolio and personal profile, but also serves to allow others to critique their work and for employers, family and friends to view their work in an easily accessible form.

Arduino microcontrollers have also been deployed in the first year of the CS programme. Here, they are used for group-based projects over a period of weeks, running in parallel with the students' other studies. These projects are used to give the students practical experience of otherwise rather abstract concepts such as finite state machines, set theory, functional programming and propositional logic. The students learn Racket, a multi-paradigm dialect of Lisp, and they are able to control the Arduino directly with this language by running the Arduino Service Interface Protocol (ASIP), which was developed for this purpose and is available at [11]. Typical projects have included a three-way traffic light system and games such as noughts and crosses and battleships. CS students also learn about assembly programming with the Arduino, through use of the Atmel Studio simulator.

In a third year Multimedia Engineering module, students use the Arduino to develop a multimedia experience. This is criteria-based and must include the use of sensors and the control of media. The assessment criteria include "meaningful interaction" and a "fun" component as well as an overview of the processes involved. As in the first year, plenty of time is given to explore ideas. Those students who engage with the work produce a wide range of interesting artefacts. It is permitted to use existing code, but this has to be modified or adapted to give a new type of interaction, with any code written by others being clearly referenced. Examples of projects include 'light chimes' – sensing the position of a swinging torch to generate music, interactive T-shirts that respond to proximity of others and smart home automation.

5 Robots

A bespoke robotic platform (MIRTO; Middlesex Robotic plaTfOrm) was developed for use with the first year CS students [12]. This comprises a set of HUB-ee wheels [13], an Arduino Teensy and Raspberry Pi computer running Linux. The robot is equipped with quadrature encoding, wireless card, bump sensors and infrared detectors for use in line-following algorithms and similar. The Teensy interfaces to these components and the Raspberry Pi is connected to the Teensy via its serial port. The ASIP protocol is used to enable the robot to be controlled by Racket programs loaded onto the Raspberry Pi.

The robots are used in projects that reinforce a number of CS concepts. For example, the functional programming concept of higher order functions is used to map Arduino pin-setting functions across a number of pins and the concepts of functions as first class objects and side effects are demonstrated through lists of functions employed in causing the robot to explore an unknown area. Another example is the use of Racket Contracts to specify required robot behavior [10]. Through the use of Linux, students also gain familiarity with use of a command line interface, which most are unfamiliar with, having grown up using only graphical user interfaces.

Student projects using this technology have included PID line-following algorithms using the IR sensors and controlling robots remotely through web pages, the Twitter API and email servers. A number of students go well beyond the taught material and one team competed successfully in the Eurobot national championships, coming 4th out of 17 teams.

6 Logic Circuits

The CS students also build simple combinatorial and sequential logic circuits using components such as logic gates, adders, clock sources and breadboards. This reinforces their knowledge of a number of concepts, including propositional logic. The latter is an example of the holistic approach to the curriculum, as students see the same concept in their study of fundamental underpinnings and in their programming workshops.

7 Discussion and Conclusions

Some of the benefits of engaging students with a physical computing approach are

- Motivation through hands on experience
- A chance to experience a whole lifecycle from concept to prototype
- Opportunities to engage with family, friends, potential employers and contribute to online communities
- Understanding testing strategies and designing tests
- Opportunities to be creative with open-ended assignments
- Engaging with current debate about sensors/data/internet of things

Motivating students and encouraging an exploratory outlook in their learning is important in terms of retention and also for their induction into university culture. Working with physical manifestations challenges students perceptions of computing and require them to work in new ways, with success in the set tasks giving them confidence to tackle new problems. One of the key areas that is explored through physical computing is the need to develop test strategies to clearly identify the nature of any problems. Errors may lie in hardware, software or the communication between devices and students need to develop analytical skills to identify where the problems lie. Physical computing also lends itself to group work and in the first year, this allows students to develop a stronger cohort identity.

For first year students, assessment sets a threshold for progression, but does not count towards their final classification. For the final year students, it is important that assessment enables grades assigned to accurately reflect each student's achievement. The Arduino community actively encourages code reuse, but this must be properly cited and documented so that a student's individual contribution can be evaluated.

Another issue with physical computing is the need to engage with the work over a period of time. Students often become adept at managing deadlines with a "just in time" approach. The physical computing tasks require more time than students might expect when they have less experience of this type of work. However, emphasising the exploratory nature of the projects and providing the necessary time in the labs helps to overcome this.

Developing kits, storing them and ensuring that damaged or missing items are replaced is also time consuming. Breadboards are not ideal, as often circuits are unreliable and temporary in nature. On the other hand, there is a loss of flexibility in using self-contained kits. Introducing soldering makes it difficult to reuse components and requires specialist lab provision, but is the better option for final year projects where students may wish to keep the artefact and be able to demonstrate it without the risk of failure.

Use of a Physical Computing approach has been beneficial in Computer Science and IT in motivating students and helping them engage with the area. It does not offer an easy option for the tutors, as they have to manage more equipment and ensure that students get through initial thresholds such as a working circuit and code, help students develop test strategies and negotiate the deliverables. For final year students, it is important that tutors are continually aware of student progress and that the "rules" for code and hardware reuse are clear.

Physical computing is not a panacea for teaching programming and it is important that tasks relate to the area being taught. However, we have found that it does help students with understanding some of the more abstract aspects of programming and other computing concepts. It also plays a major role in student motivation; we often find that students do not want to leave at the end of their lab sessions. Progression rates have improved since the introduction of physical computing and many students have engaged in external activities to show off their work. These include open days, National Science Week events and robotics competitions as described above.

References

1. Ahadi, A., Lister, R., Teague, D.: Falling behind early and staying behind when learning to program. In: Proceedings of the 25th Psychology of Programming Conference, PPIG (2014)
2. Blikstein, P.: Gears of our childhood: constructionist toolkits, robotics, and physical computing, past and future. In: Proceedings of the 12th International Conference on Interaction Design and Children, pp. 173–182. ACM (2013)
3. Buechley, L., Eisenberg, M., Catchen, J., Crockett, A.: The LilyPad Arduino: using computational textiles to investigate engagement, aesthetics, and diversity in computer science education. In: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, pp. 423–432. ACM (2008)
4. Cambron, M.E.: Using the Ardunio in freshmen design. In: Sixth Annual (FYEE) First Year Engineering Experience Conference on Enhancing the First Year of Engineering Education College Station, TX, 7–8 August 2014
5. Herger, L.M., Bodarky, M.: Engaging students with open source technologies and Arduino. In: 2015 IEEE Integrated STEM Education Conference (ISEC), pp. 27–32. IEEE (2015)
6. Kato, Y.: Splish: a visual programming environment for Arduino to accelerate physical computing experiences, pp. 3–10. IEEE (2010). doi:10.1109/C5.2010.20
7. Okita, S.Y.: The relative merits of transparency: investigating situations that support the use of robotics in developing student learning adaptability across virtual and physical computing platforms: relative merits of transparency in learning adaptability. Br. J. Educ. Technol. **45**, 844–862 (2014). doi:10.1111/bjet.12101
8. Robins, A.: Learning edge momentum: a new account of outcomes in CS1. Comput. Sci. Educ. **20**, 37–71 (2010). doi:10.1080/08993401003612167
9. Rubio, M.A., Romero-Zaliz, R., Mañoso, C., de Madrid, A.P.: Enhancing an introductory programming course with physical computing modules. In: 2014 IEEE Frontiers in Education Conference (FIE), pp. 1–8. IEEE (2014)
10. Boender, J., Currie, E., Loomes, M., Primiero, G., Raimondi, F.: Teaching functional patterns through robotic applications. In: Proceedings TFPIE 2015: The Fourth International Workshop on Trends in Functional Programming in Education (2015)
11. Racket Asip Client Library. https://github.com/fraimondi/racket-asip
12. The MIddlesex Robotic plaTfOrm (MIRTO). https://github.com/fraimondi/myrtle
13. Creative Robotics: HUB-ee, About-HUBee-Wheels. http://www.creative-robotics.com/
14. Clarkson, R.: A Self Contained Arduino Toolbox, Middlesex University Undergraduate Project submitted, 24 April 2014