6-14-2024

# Let Them Build: Lessons From Introducing Hardware Programming and Implementing Service Learning In A Course Open To All Students

John Gallauger
*Boston College*, john.gallaugher@bc.edu

# "LET THEM BUILD!" LESSONS FROM INTRODUCING HARDWARE PROGRAMMING AND IMPLEMENTING SERVICE LEARNING IN A COURSE OPEN TO ALL STUDENTS

*TREO Paper*

John Gallaugher, Boston College, Chestnut Hill, MA, USA, john.gallaugher@bc.edu

## Abstract:

*Despite sustained demand for workers with programming skills and the importance of programming across disciplines, negative attitudes continue to deter many from studying programming. While python is the most popular programming language and the first-choice for introducing programming in many universities, it is often taught as a text-based introduction, abstracted from the types of user interfaces that students regularly encounter and find most engaging. The author sought to counteract this by creating a novel, repeatable, cross-disciplinary course accessible to all undergraduate students across all majors that engages students in programming experiences that are decidedly different from the conventional introduction many receive as their first university exposure to programming. The course teaches hardware programming in CircuitPython while engaging students in collaborative challenges, individual projects, as well as service learning through the creation and delivery of assistive technology projects.*

*Keywords: Education, Programming, Engineering, Assistive Technology.*

## 1 Introduction

Negative attitudes toward programming continue to deter many students from pursuing careers in technology, especially pressing, given the increased and sustained demand for technical workers (Gurer et all, 2019). The problem is made even more acute given the lack of women, black, indigenous, and latinx students pursing technical careers (Kulkarni et al, 2018).

While Python remains the leading programming language in both practice and education, female students have shown less enthusiasm for python programming than their male counterparts. However, programming tools that incorporate graphical environments are viewed more positively (Zdawczyk and Varma 2022). Courses that involve game play, design (Sharma et al., 2021) and project work (Hooshangi et al, 2022) are also seen as being more appealing to all students and in particular to female students. Noting this, the author used an action research approach to introduce and iteratively refine a novel course in an attempt to increase the appeal of, learning in, and performance during a programming-oriented course.

The course, "Physical Computing: Art, Robotics, and Tech for Good," teaches python programming, but enhances the learning of core concepts by using CircuitPython extensions to the language. Doing so allows students to learn programming fundamentals while also learning to manipulate computer hardware - gathering input from sensors, operating motors, playing sound, independently animating LED lights, working with Bluetooth and Wi-Fi to explore the Internet of Things, and more. The course further combines multiple pedagogical components: flipped class, project base learning, service learning, formative assessment, and summative assessment - working in concert to create an impactful learning experience that students find challenging, valuable, and enjoyable.

# 2 Course Approach

The approach outlined in the construction and refinement of this course uses several pedagogical techniques: flipped classroom, formative assessment, project-based learning, and community service.

## 2.1 Flipped Classroom

The flipped classroom technique removes much of the conventional lecture learning from the in-person classroom, and instead delivers lessons asynchronously outside of the classroom, during time normally allocated for homework. Student time spent in the classroom is focused largely on exercises normally targeted for homework. In this way, the lecture / homework experience is "flipped", with lecture happening asynchronously and homework typically occurring concurrently, in a classroom with faculty present (O'Flaherty and Phillips, 2015).

The flipped classroom has been shown to have many advantages for programming students (Giannakos, 2014). Since lectures are delivered via video, a student can stop and repeat portions of a lecture if they find a concept difficult to understand; especially useful in programming courses, where an easy-to-commit syntax error could cause a student to fall behind (Stone, 2012). It is also common for student skill to be widely distributed in introductory computing courses. A synchronous classroom may place bright students with less experience at a disadvantage relative to their peers. Inexperienced students may be self-conscious in asking for help in a synchronous classroom. As such, the use of a flipped class in programming courses can help improve student performance, reduce attrition, and improve retention that might otherwise result from the traditional lecture approach, with particular advantages for students who are new-to-programming (Latulipe et al, 2018).

To ensure that students complete asynchronous learning and understand the presented material, several techniques are used. First, each week students submit code-along work from their video lessons. Students also complete an online quiz after each video lecture. Lecture work is further reinforced by class time, a once-a-week, 2 hour 20 minute synchronous class where students arrive with laptops and hardware. The instructor often starts each class with a very short review of concepts covered in the prior week's video lessons, then presents students with a series of challenges which play out like games. Example: "You've learned to work with the accelerometer, animate LED lights, and play sounds. Now use these resources to build a magic wand." The instructor demonstrates a completed build or shows a video of a working build, then gives the students 15 to 30 minutes to complete the task. Students are offered an incentive to show their work to their instructor (e.g. come forward and wave the wand at a piece of candy, then select the snack if the build works). Students completing tasks ahead of classmates are then asked to help peers who may be struggling with concepts needed to solve the problem. This provides an additional opportunity for student-led learning designed to create a stronger classroom community.

The last half-hour of class is often spent with a hands-on quiz where each student works independently solving a challenge and submitting their work. The submission typically involves learning from prior video lessons and work reinforced through class challenges. In this way, even if a student has struggled with concepts, they have several opportunities for mastery before they are quizzed and graded on their ability to apply the concept independently in their own problem solving.

## 2.2 Assessment

New learners often make frequent mistakes while coding, and students improve as programmers as they recognize errors and develop mental models for data structures and logic execution. The course uses several types of low-stakes "formative" assessment mechanisms so that students who (as most will) struggle with new concepts are provided with a clear opportunity to see the correct answer, acquire concepts they may have missed or misunderstood, and prepare for more rigours, heavily graded summative assessments. These low-stakes assessments include weekly submissions from code-along lessons, re-takeable online quizzes, in-class challenges, and in-class quizzes involving real-time

independent project builds as well as concept questions. By the time students reach higher-stakes assessments (e.g. the final exams and projects) they have had an opportunity to try and correct nearly every concept introduced in the course, building confidence along with skill.

## 2.3 Project-based Learning

Students apply learning in three self-directed projects. A “Make Art” project defines “art” broadly. Students are shown examples from prior work, as well as examples that other artists and makers have created outside of class, to stimulate their own interest and ideas. Student creative application of their acquired skills have resulted in touch-reactive art; kinetic art; art that responds to environmental sensors (e.g. temperature or proximity); art using light animations, sound and audio; electronics-infused fashion; and games and toys that employ art concepts.

In the second, “Tech for Good” project, students work individually or in pairs to create an assistive technology project for real clients – members of an on-campus program for student aged 3 through 21 who have very severe developmental delays. Students work collaboratively to assess the needs of clients, they propose and evaluate ideas, they iterate through prototypes, and they arrive at a deployed and maintainable solution. As such, the course offers one of the few course-based opportunities for students to engage in actual product development, management, and deployment. Examples of past work include accessible games, a wheelchair accessible “cash-drawer” so a student could many a coffee shop, and a “dignity button” to provide bathroom privacy for students with physical limitations.

Final projects are even more open-ended, with examples of past student projects including a "robot-barista" that dispenses the selected coffee type with properly weighed and ground beans, and poured-over with water heated to a perfect temperature; and a marble maze platform remotely moved by an accelerometer’s shifts on x and y axes. All student work is presented in a final-day “celebration of student technology” where members of the university community and employers are invited to explore student work, science-fair-style. Many students have gained employer connections and joined startup teams based on connections made during these events.

## 2.4 Results and Opportunities

The course has proved tremendously successful, with positive feedback, increased student interest in programming careers, and many alumni have further pursued physical computing projects as hobbies or as careers. The various approaches combined in the course create a learning environment that welcomes and encourages a diversity of student backgrounds, enhances learning, increases retention, encourages learning improvement, promotes collaboration and peer learning, develops valuable soft skills, and results in real project deployment and rewards through community service. The author has made all flipped class learning videos (over 80), as well as lessons, slides, and past exams, available to other faculty members for free upon request so that others may leverage the approach and materials on their own.

References available upon request.