ASEE AMERICAN SOCIETY FOR ENGINEERING EDUCATION

Paper ID #40895

Teaching Computational Thinking Using Open-Source, High-Impact Practice
Project-based Approach

Dr. Salman Siddiqui, Georgia Southern University

Dr. Salman Siddiqui joined Georgia Southern in 2013 and is currently working as a Senior Lecturer in the Department of Electrical & Computer Engineering. He received his B.S.E.E., M.S.E.E., and Ph.D. in Electrical Engineering from Florida State University, Tallahassee, FL. His research interests include analysis, simulation, and control of human-robot systems; project-based education, STEM outreach, and application of new instructional technology in classroom instruction.

Dr. Rami Jubrail Haddad, Georgia Southern University

Rami J. Haddad is a Professor and Interim Chair in the Department of Electrical & Computer Engineering at Georgia Southern University. He received his B.Sc. degree in Electronics and Telecommunication Engineering from the Applied Sciences University, Amman, Jordan, in 2004, the M.S. degree in electrical and computer engineering from the University of Minnesota Duluth, Duluth, MN, USA, in 2006, and the Ph.D. degree from the University of Akron, Akron, OH, USA, in 2011. He is also the Founding Director of the Optical Networks and Smart Applications (ONSmart) Laboratory at GSU. His research interests include distributed power generation, smart grid applications, optical fiber communication/networks, machine learning/artificial intelligence, UAV ad-hoc networks, multimedia communications, multimedia bandwidth forecasting, and engineering education.

Teaching Computational Thinking Using Open-Source, High-Impact Practice Project-based Approach

Abstract

This paper explores a novel pedagogical approach incorporating Open Educational Resources (OER) and High Impact Practices (HIPs) for cultivating computational thinking in engineering education. It diverges from the conventional practice of introducing these concepts primarily through programming courses. Instead, this approach advocates for the integration of project-based hardware programming applications into the curriculum. The paper details a successful implementation of this methodology within a first-year computing course, utilizing Arduino and MATLAB as the primary tools. The core of this approach is to immerse students in hands-on hardware programming projects, aiming to foster a deeper engagement and enthusiasm for engineering applications and computational thinking. This method has demonstrated a significant enhancement in student performance within the course. Comprehensive statistical assessment methods were employed to validate the effectiveness of this teaching model. These included quantitative and qualitative analyses, offering a robust evaluation of the pedagogy's impact.

Furthermore, the paper presents a comparative analysis between this innovative teaching model and the traditional format of the same course. This comparison is critical in highlighting the advancements and improvements brought about by the project-based approach. The findings from this study offer valuable insights and evidence for the merit of integrating hands-on hardware programming in early engineering education, suggesting a transformative shift in teaching computational thinking to engineering students.

Introduction

The concept of computational thinking, fundamentally defined as the cognitive process of formulating problems and articulating their solutions in a manner executable by a computer [1-3], stands as a cornerstone in the education of engineering students, particularly during their first year. This skill, critical in the engineering discipline, involves translating the abstract dimensions of a physical problem into a computational framework, employing methodologies such as top-down or bottom-up approaches. Moreover, computational thinking is intrinsically linked with the engineering design process [3-5], underscoring its indispensability in the repertoire of engineering students.

Traditionally, the teaching of computational thinking in engineering education has predominantly relied on computer programming courses [6], often employing languages such as MATLAB, C, or JAVA. While effective to a degree, this conventional approach sometimes falls short of meeting freshman students' practical, hands-on engineering design expectations. The primary reason for this disconnect can be traced back to the inherently abstract and virtual nature of traditional programming instruction, which may not fully resonate with students' early experiences in engineering.

Recognizing the essential role of computational thinking in the engineering curriculum, this study proposes a more dynamic, engaging, and student-centric pedagogical strategy. It underscores the integration of Open Educational Resources (OER) and High Impact Practices (HIPs) to effectively bridge the gap between abstract computational concepts and tangible engineering applications. This innovative approach embraces a project-based learning paradigm [7-9], harnessing the potential of Arduino—a widely accessible, affordable, and open-source electronics prototyping platform. This strategy is designed to transform computational thinking into an interactive, hands-on learning experience, thereby aligning more closely with the practical aspects of engineering and enhancing student engagement.

The objective is to introduce first-year engineering students to the fundamental principles of computational thinking and engineering design in a tangible, interactive manner. To this end, a first-year computing course was restructured to integrate Arduino hardware programming applications cohesively throughout the curriculum in tandem with MATLAB. This integration not only bridges the gap between computational thinking and the engineering design process but also demonstrates the tangible impact of programming in interfacing with real-world applications. This innovative approach aims to reignite student interest in engineering by providing a more holistic, hands-on educational experience.

Course Structure and Pedagogical Strategy

Original Course Model

The "Computing for Engineers" course, positioned at the freshman level within the electrical and computer engineering curriculum, serves as an introduction to computing, computational thinking, and engineering problem-solving, with a specific focus on MATLAB programming. This foundational course, requiring no prior programming experience but a co-requisite of Calculus I, progressively covers topics from basic programming principles to advanced concepts. Key areas include engineering essentials, ethics, communication skills, and the top-down problem-solving approach within the MATLAB Environment. The curriculum encompasses various programming control structures such as sequence, conditional, and repetition structures, followed by functions, numerical techniques, data modeling, cell arrays, structure arrays, and file operations. Traditionally, the course combined classroom lectures with lab exercises for the practical application of programming concepts. This 3-credit hour course included twice-weekly lectures of 50 minutes and a weekly lab session lasting an hour and forty minutes. The original grading structure is highlighted in Table I.

Table I- Course Assessment Components and Grade Allocation

Assessment Component	Weight
Exam#1	20%
Exam#2	20%
Final Exam	30%
Lab Projects	15%
Homework/Classwork	10%
Quizzes	5%

Restructured Course Model

In an effort to overcome the constraints of traditional teaching methods and inspired by research that highlights the advantages of project-based learning—such as increased student engagement, success, and retention—a dynamic and engaging teaching strategy centered around High Impact Practices (HIPs) was implemented. In the mid-semester, the course instructors introduced the Arduino microcontroller and the Sparkfun Inventors Kit. This strategic addition, in line with the principles of high-impact, project-based learning, followed lessons on repetition control structures and basic hardware programming. Students were then challenged to propose and execute projects in pairs, guided by the kit's manual. This innovative approach led to a significant boost in student engagement and performance.

Building on this success, the course underwent further restructuring in the following semester. A focus was placed on developing Open Educational Resources tailored to integrate project-based learning elements earlier in the course. The Sparkfun inventor's kits were introduced within the first month rather than midway through the semester. This adjustment allowed hardware applications to be woven into each weekly lab session, moving away from a previous sole focus on hardware. Such a change provided students with more time to engage with the sensors and electronic components of the kit, gradually building their skills to handle increasingly complex projects.

Furthermore, the course was enhanced to include both an oral presentation and a written report on the projects, adding depth to the learning experience. Reflecting these changes, the grading structure was revised to better align with this enriched, hands-on educational approach, as highlighted in Table II.

Table II- Course Assessment Components and Grade Allocation

Assessment Component	Weight
Exam#1	20%
Exam#2	20%
Final Exam	25%
Lab Projects	15%
Arduino Project	10%
Homework/Classwork	5%
Quizzes	5%

These modifications in the course structure and assessment demonstrate a strategic move towards a more hands-on, project-based learning approach, aligning with contemporary pedagogical trends in engineering education.

Hardware-based Programming Model

The innovative approach outlined in this paper involves the use of Arduino-UNO microcontroller, along with electronic components and sensors, to prototype electronic applications as a cornerstone for project-based learning. In this model, the Arduino-UNO microcontroller functions in a tethered configuration, synergizing with the MATLAB

development environment. This setup positions the Arduino-UNO as a server, responding to requests from MATLAB programs (acting as the client) via serial communication.

The primary goal of this hardware-based programming model is to foster authentic learning experiences in first-year computing courses, enhancing students' grasp of computational thinking. This model is meticulously crafted to heighten engineering students' engagement by immersing them in real-world system development. It effectively bridges the gap between the theoretical aspects of programming and the practical, applied nature of engineering. Under instructor guidance, this model creates a collaborative, project-based learning environment that offers numerous educational benefits:

1. **Authentic Learning Environment**: Students engage in the creation of tangible, real-world products, which enhances the applicability and relevance of their learning experience.
2. **Bridging Theoretical and Practical Divides**: This approach narrows the gap between abstract programming concepts and the tangible, hands-on nature inherent in engineering disciplines.
3. **Foundation in Engineering Principles**: By focusing on hardware-based programming, the model addresses fundamental engineering principles and hands-on design at the freshman level.
4. **Collaborative Learning and Teamwork**: The environment fosters collaboration and teamwork, enhancing students' sense of community and mutual support.
5. **Capstone-Like Projects**: Students are given the opportunity to apply their learning in comprehensive projects, which solidifies their understanding of the core concepts.
6. **Early Development of Communication Skills**: The model encourages the development of communication skills through presentations and report writing, essential competencies in the engineering field.
7. **Enhanced Course Performance and Success**: By integrating these elements, the model aims to significantly improve students' overall performance and success rates in the course.

In summary, this hardware-based programming model represents a progressive shift in engineering education, emphasizing practical application, collaboration, and real-world relevance to enrich the learning experience of engineering students.

Implementation and Evaluation

The innovative Open-Source Hardware-based Programming model was integrated into a freshman-level 'Computing for Engineers' course, marking a significant shift in the pedagogical approach. Within this framework, students were tasked with undertaking a diverse array of projects. These projects were meticulously designed to not only challenge the students' understanding of computational thinking but also to stimulate their creativity in engineering design solutions. The scope of the projects undertaken was broad and inventive, encompassing a variety of applications. Project examples included the development of a music jukebox, an alarm clock, a secure lockbox, a ‘Simon Says’ memory-enhancing game, a motion-sensor-based security system, and the design and construction of an autonomous robot. These projects underscore the model’s effectiveness in fostering both technical proficiency and innovative thinking among students.

A notable aspect of this implementation was the requirement for students to code in MATLAB, diverging from the readily available C-programming resources for Arduino on the web. This requirement ensured that students engaged in original coding efforts, enhancing their problem-solving skills and deepening their understanding of MATLAB, a prominent tool in engineering.

To prepare the students for these projects, they were introduced to the Sparkfun Arduino Kit through a demonstration highlighting the use of various components. This included activities such as controlling blinking LEDs, obtaining inputs from push buttons, reading values from a potentiometer, displaying data collected from a temperature sensor, and generating frequencies on a piezo buzzer. However, the ultimate goal of these projects was to integrate multiple kit components into functional circuits aimed at solving real-world problems.

Throughout these projects, common skills were developed by the students. These skills encompassed the ability to control and read data from sensors, actuators, and other hardware components, write code to effectively interface with these hardware elements to meet user requirements, design circuits using the Fritzing software, troubleshoot both hardware and software issues within their projects, collaborating within a team, and enhancing verbal and written communication skills by presenting the project to peers and composing a project report. To illustrate the tangible outcomes of this educational model, Figures 1 to 5 in the paper showcase a selection of the student projects. These examples serve not only as a testament to the students' ingenuity and skill but also as an endorsement of the model's effectiveness in enhancing the educational experience in engineering courses.

The Alarm Clock Project (highlighted in Figure 1) served as a significant motivator for learning various hardware and software aspects while producing a product used in daily life. Students were exposed to hardware components such as photoresistors, temperature sensors, LEDs, and an LCD screen with adjustable brightness controlled by a potentiometer. Their programming skills were put to the test as they created a functional system that involved extracting time information from the system, converting military time to standard time format, efficiently representing data due to LCD screen constraints, and effectively troubleshooting both hardware and software issues.

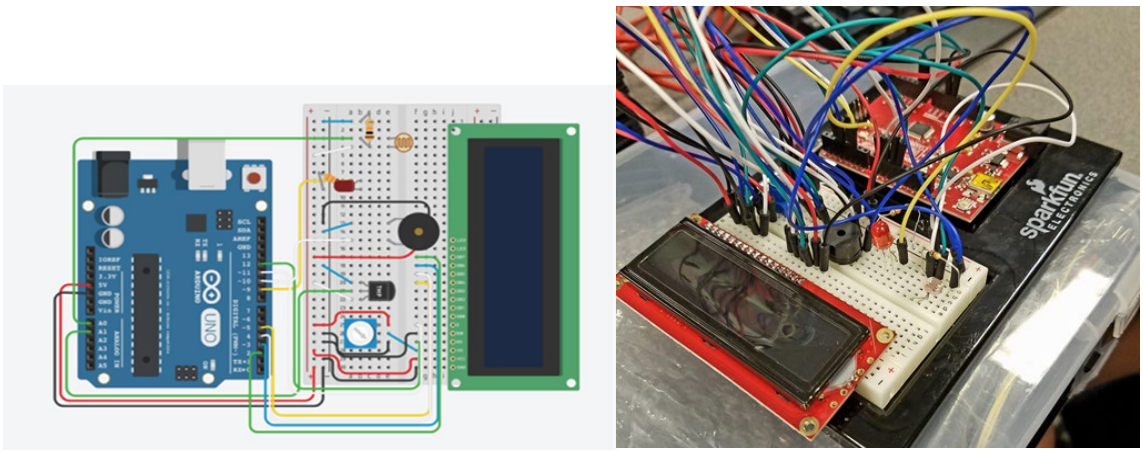

Figure 1. Alarm Clock Project

The SparkFire Prevention System (highlighted in Figure 2) focused on designing a fire safety system for scenarios like grease fires, employing components such as a temperature sensor, piezo buzzer, RGB LED, diode, transistor, and DC motor. Beyond constructing the hardware and programming it to detect temperature fluctuations and control the motor, students developed a prototype to illustrate its operation.

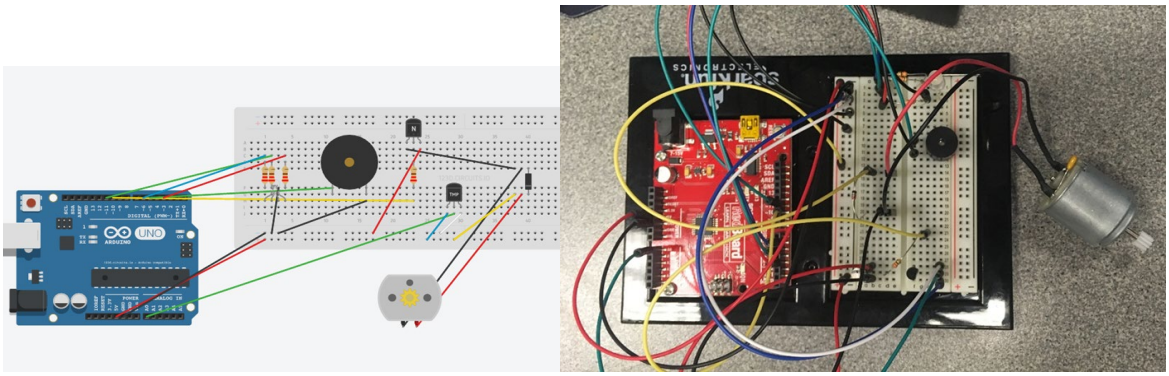

Figure 2. Sparkfire Prevention System Project

The Jukebox Project (highlighted in Figure 3) aimed to simulate the creation of personal or business entertainment products. Although the hardware components mainly consisted of pushbuttons, resistors, and buttons, the software programming aspect was particularly intensive. Students delved into random number generation to select music from four different genres associated with each pushbutton. Additionally, they programmed three different songs within each genre, culminating in a functional system.

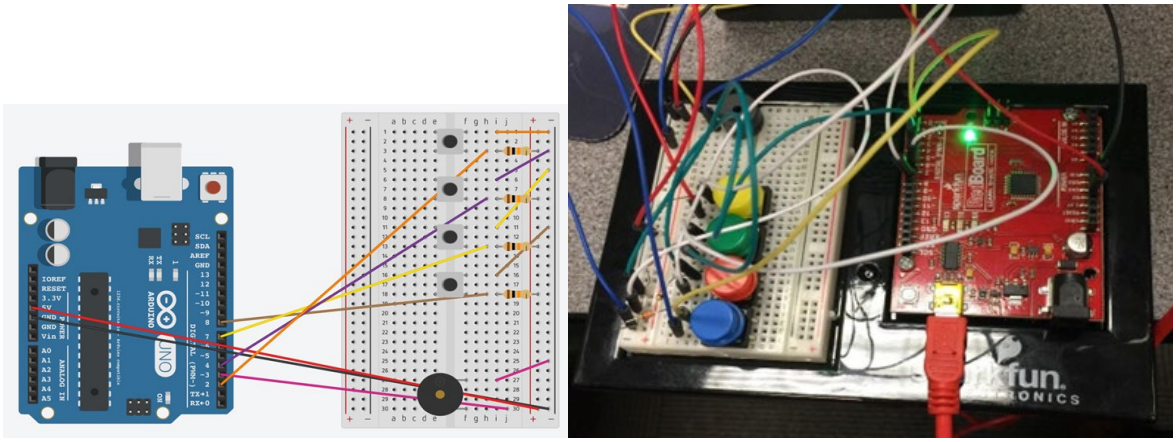

Figure 3. Jukebox Project

The Environment Control System Project (highlighted in Figure 4) underscored the design of systems tailored to consumer or industrial needs. Hardware components included photoresistors, temperature sensors, servo motors, DC motors, diodes, and transistors. System control was software-based, requiring specific programming commands to operate multiple hardware features.

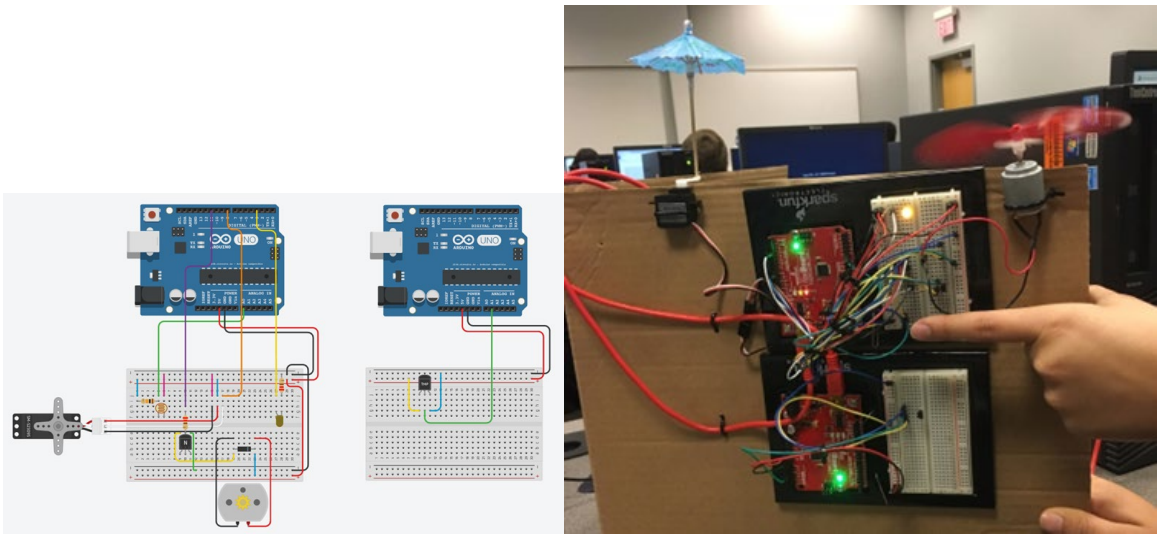

Figure 4. Environment Control System Project

Lastly, the Simon Says Game Project (highlighted in Figure 5) illustrated the use of games as a means to address real-world problems, such as improving memory skills. Hardware components included pushbuttons, LEDs, a piezo buzzer, and resistors. Software development necessitated the implementation of a random sequence generator to create increasingly complex patterns as the user progressed through the game, the use of arrays for pattern storage and validation, scorekeeping, and auditory feedback through tone generation during gameplay.

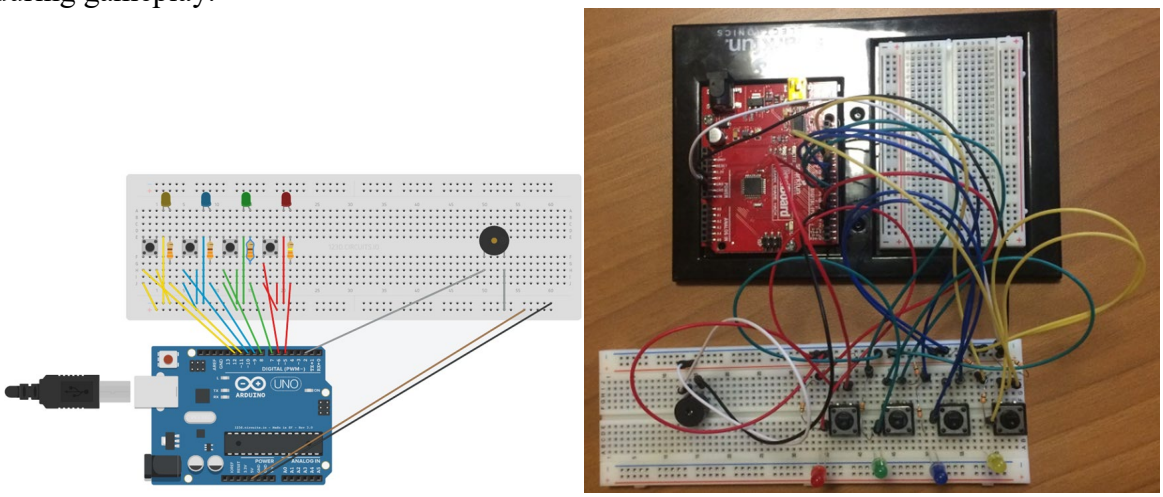

Figure 5. Simon Says Game Project

The application and evaluation of the project-based model within a first-year engineering course revealed its considerable potential in enhancing students' technical skills, promoting creativity, and equipping them for more complex challenges in their engineering education. The hypothesis posited that the implementation of a project-based model in such a course would lead to improved student performance.

To empirically test this hypothesis and measure the model's effectiveness, a comparative study was conducted using two separate iterations of the same course, designated as control

and test groups. The assessment of student performance in this study was based on a comparative statistical analysis. The control group consisted of students enrolled in the traditional format of the first-year computing course, focusing exclusively on Matlab without the inclusion of project-based elements. Conversely, the test group participated in the same course, incorporating the newly proposed open-resource, project-based model.

The study encompassed 48 first-year students, with 25 in the control group and 23 in the test group. Their academic performance was evaluated through their final course grades, calculated based on assessment criteria detailed in Tables I and II.

Figure 6 displays the normal distribution fit of the final course grades between the control and test groups. The results from this analysis highlighted a significant disparity in both the mean and standard deviation of grades between the two groups. Notably, the average score for the control group was 71.04, in contrast to the test group, which averaged 79.24. This finding represents an improvement of over 11.5% in overall academic performance as measured by course grades in the test group. However, it was observed that the test group exhibited a higher standard deviation in grades, suggesting greater variation from the mean. This variation is attributed to the necessity of further refining the Open Educational Resources (OER) material to enhance its user-friendliness and adaptability to diverse student requirements. These findings corroborate the initial hypothesis, indicating that the project-based model enhances student performance and fosters a more engaging and effective learning environment.

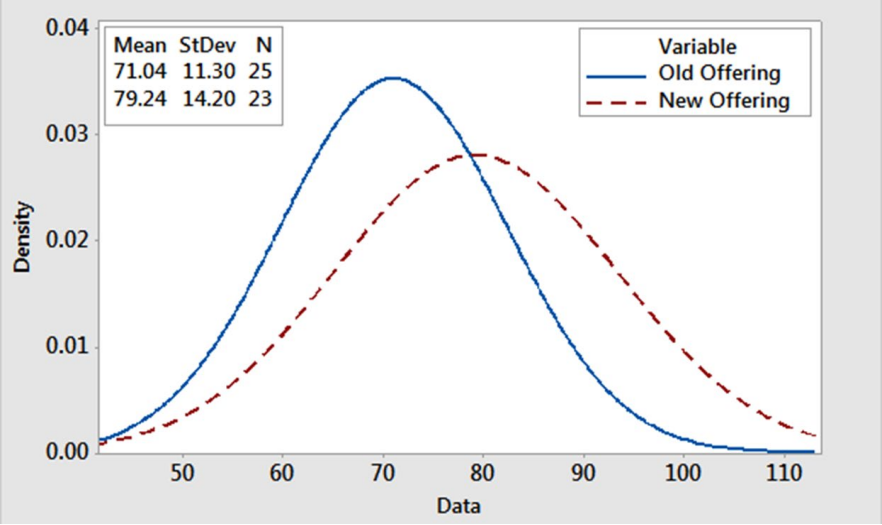

Figure 6. Normal fitting of students' final exam grades of two offerings

To rigorously assess and confirm the preliminary results, an extensive statistical analysis was undertaken using Minitab statistical software. The research posited a null hypothesis asserting the absence of statistical differences in the final course grades between the control and test groups due to the implementation of the model. The chosen method for hypothesis testing was the one-way analysis of variance (ANOVA), employing a 5% significance level ($p=0.05$) as the probability of error criterion. The dependent variable in this analysis was the students' final course grades at the conclusion of the course. The primary factor under examination was the treatment effect, which was modeled based on the final course grades

in both the control and test groups. The two-level treatment compared the traditional delivery of the computational thinking course (control group) against the innovative delivery using the OER project-based model (test group).

The results, as depicted in Figure 7, yielded a *p*-value of 0.031, falling below the established threshold of 0.05 for significance. Consequently, the null hypothesis, which suggested no significant difference between the groups, was rejected at a confidence level of 96.9%. This finding led to the conclusion that there is a statistically significant difference between the control and test groups, thereby validating the efficacy of the proposed model. Additionally, considering the relatively small sample size in this study, the Cohen's effect size was calculated. The obtained Cohen's d values, ranging from 0.577 to 0.726, indicate that the model had a medium to large effect on student performance despite the limitations imposed by the sample size.

General Linear Model: Final Exam Grade versus Offering

Method

Factor Information

Factor	Type	Levels	Values
Offering	Fixed	2	Proposed, Traditional

Analysis of Variance

Source	DF	Adj SS	Adj MS	F-Value	P-Value
Offering	1	805.3	805.3	4.94	0.031
Error	46	7499.3	163.0		
Total	47	8304.6			

Figure 7. The outcome of the one-way ANOVA analysis

In addition to the quantitative outcomes, student satisfaction with the hardware programming component was also reflected in their final course evaluations conducted toward the end of the semester. Selected responses from these evaluations illustrate the positive reception of the course material and its practical applications, as follows:

- "*I liked how useful the material is and how many helpful resources were available to learn the material.*"
- "*I liked learning coding and interfacing with hardware like Arduino. Allows me to get ahead and learn more things.*"
- "*The Arduino project was fun and the labs are good too.*"
- "*The course itself is easy to engage in because it is fun to learn MATLAB and its uses (applications)*"
- "*I liked being able to apply knowledge in the real world.*"
- "*The work was challenging but enjoyable.*"
- "*We got to explore practical applications of what we have learned so far with sensor and Arduino board.*"

These responses indicate a strong appreciation for the practical, hands-on approach of the course, highlighting the successful integration of theory and application in teaching hardware programming and its relevance in real-world contexts.

Conclusions

The implementation of a high-impact, project-based learning approach, utilizing the Arduino Sparkfun Inventor's Kit in a freshman engineering course, demonstrated significant improvements in student engagement, enthusiasm for engineering applications, and overall performance. This approach effectively cultivated computational thinking skills and fostered collaborative engineering design work among students. Notably, the integration of hardware programming into lectures and labs enabled all students, regardless of prior experience, to develop a comprehensive understanding of the subject matter. This leveling of the playing field allowed inexperienced students to achieve competencies comparable to their more experienced peers, particularly in hardware-related tasks. Additionally, students with prior programming experience were able to create more optimized code.

The course also served as an effective precursor for subsequent programming courses, such as C Programming. Students who completed this course reported ease in transitioning to more advanced programming, aided by their familiarity with simulation tools like Tinkercad, which closely mirrored the hardware components used in the Arduino-based course.

Overall, the study concluded that the introduction of hardware programming in the course led to a statistically significant difference in student performance, as validated by a confidence level exceeding 96.9% in comparative analyses. This outcome underscores the efficacy of the project-based approach in enhancing computational thinking and technical skills in freshman engineering students, thereby affirming its value as an educational strategy in engineering curricula.

Bibliography

[1] J. M. Wing, "Computational thinking's influence on research and education for all," *Italian Journal of Educational Technology*, vol. 25, no. 2, pp. 7-14, 2017.

[2] F. K. Cansu and S. K. Cansu, "An Overview of Computational Thinking," *International Journal of Computer Science Education in Schools*, vol. 3, no. 1, pp. 17–30, 2019.

[3] R. J. Haddad and Y. Kalaani, "Can computational thinking predict academic performance?" in *2015 IEEE Integrated STEM Education Conference*, Princeton, NJ, USA, 2015, pp. 225-229.

[4] H. Ehsan, A. P. Rehmat, and M. E. Cardella, "Computational thinking embedded in engineering design: capturing computational thinking of children in an informal engineering design activity," in *Int J Technol Des Educ*, vol. 31, pp. 441–464, 2021.

[5] R. J. Haddad and Y. Kalaani, “Cross-Disciplinary Perceptions of the Computational Thinking among Freshmen Engineering Students,” in *ASEE Southeastern Section Conference*, Gainesville, FL, 12-14 April 2015.

[6] F. B. Flórez, R. Casallas, M. Hernández, A. Reyes, S. Restrepo, and G. Danies, "Changing a Generation’s Way of Thinking: Teaching Computational Thinking Through Programming," *Review of Educational Research*, vol. 87, no. 4, pp. 834–860, 2017.

[7] S. Bell, "Project-based learning for the 21st century: Skills for the future," *The Clearing House*, vol. 83, no. 2, pp. 39-43, 2010.

[8] A. Sahin, "STEM project-based learning: Specialized form of inquiry-based learning," in *STEM Project-Based Learning: An Integrated Science, Technology, Engineering, and Mathematics (STEM) Approach*, pp. 59-64, 2013.

[9] S. Groß, M. Kim, J. Schlosser, D. Lluch, C. Mohtadi, and D. Schneider, “Fostering computational thinking in engineering education: Challenges, examples, and best practices,” in *IEEE Global Engineering Education Conference, EDUCON*, pp. 450-459, 2014.