ASEE AMERICAN SOCIETY FOR ENGINEERING EDUCATION

Paper ID #40871

# Fostering Computational Thinking in Freshman Electrical Engineering Students through a Micro:Bit Code Project

**Dr. Sylmarie Davila-Montero, The Citadel**

Sylmarie Dávila-Montero serves as an assistant professor in the Electrical and Computer Engineering Department at The Citadel. Her teaching experience spans across various undergraduate levels, including freshmen, sophomores, juniors, and seniors. She received a B.S. degree in electrical engineering and a minor in applied mathematics (with high honors) from the University of Puerto Rico at Mayagüez, Puerto Rico, in 2015 and a Ph.D. degree in electrical engineering from Michigan State University (MSU), East Lansing, Michigan, in 2022. Sylmarie worked as a Systems Engineer for the MITRE Corporation in 2015 and as a Data Analyst/Contractor for the Environmental Protection Agency in 2018 and 2019. Her research interests include real-time processing of biomedical and social signals, efficient implementation of machine learning algorithms, design of wearable social behavior and health monitoring systems, and engineering education.

# Fostering Computational Thinking in Freshman Electrical Engineering Students through a Micro:Bit Code Project


## Abstract

Computational Thinking (CT) is an essential skill for budding engineers, particularly in the realm of Electrical and Computer Engineering. CT can be broadly defined as a problem-solving technique that involves dissecting complex problems into manageable steps, suitable for computer execution. Typically, students begin honing their CT skills during their initial programming-related courses. However, these courses often entail a steep learning curve as students grapple with programming rules, language syntax, and the array of tools at their disposal. For freshman electrical engineering students, who may perceive programming as an extraneous domain, this presents a unique challenge.

This paper will describe the approach that was taken in an Introduction to Computer Applications for Electrical Engineers class to nurture and evaluate CT among freshman electrical engineering students. The primary objective was to cultivate proficiency in programming concepts, engineering design, and computational thinking while employing Python and the Micro:Bit platform to construct a functional Morse code machine as their final class project. This hands-on project bridged the gap between theory and application, rendering abstract concepts tangible and fostering a deeper understanding of programming principles. Students were also required to apply programming knowledge to interface with sensors and actuators present in the Micro:Bit platform.

Overall, students engaged in a multifaceted learning experience, combining both theoretical and practical elements. They tackled key programming topics, including variables, arrays, logic and branching, loops, functions, and flowcharts, all while working towards the creation of their Morse code machine. By contextualizing these programming concepts within a real-world application, students were motivated to explore beyond their comfort zones and leverage CT skills to develop creative solutions.


## Introduction

As the world becomes more digitalized, new generations of engineers will be required to have a deep understanding of computer programming and its design process. Computational Thinking is the cognitive process that underlies programming, emphasizing problem-solving methodologies essential for effective algorithmic design and implementation [1]. It involves breaking down complex problems into manageable components, recognizing patterns, developing algorithms, and creating solutions that can be executed through programming languages. In the rapidly evolving landscape of technology, the cultivation of Computational Thinking skills is crucial for engineering students to thrive in their future professions [1].

Within the engineering design process, Computational Thinking acts as a foundational framework that supports effective problem-solving. Programming skills, as a practical application of computational thinking, enable engineers to implement algorithms and translate

abstract concepts into functional solutions. The iterative nature of the design process aligns seamlessly with the problem-solving strategies ingrained in Computational Thinking, creating a synergy that enhances the efficiency and creativity of engineering solutions [2][3].

However, traditionally, electrical engineering majors struggle to see the relevance of programming skills in their professional development. With the goal of motivating students to overcome this perception and emphasizing the practical applications of programming in their field, there is a need for innovative pedagogical approaches in their early engineering courses. Engaging students in hands-on projects, particularly those involving tangible devices like the Micro:Bit microcontroller, provides a concrete context for applying programming skills to solve real-world engineering problems [4].

At The Citadel, the initial programming course for electrical engineering majors is Computer Applications for Electrical Engineers, which is structured into two parts, each corresponding to different programming languages. The initial segment introduces computational thinking, programming concepts, and best practices using MATLAB. Subsequently, the second part builds upon the acquired knowledge, applying previously learned programming concepts and practices but using Python and its syntax. Additionally, during the latter part of the course, fundamental microcontroller concepts are introduced alongside the utilization of MicroPython for programming. This comprehensive class curriculum ensures students acquire proficiency in various programming languages and concepts, such as variables, arrays, data types, lists, dictionaries, logic and branches, loops, functions, and data visualization.

To actively engage students in hands-on exercises during this introductory programming class, the Micro:Bit is integrated into the second half of the class curriculum. The Micro:Bit, shown in Figure 1, is a user-friendly and cost-effective tool that doesn't require extensive technical expertise, making it accessible to first-year undergraduates. This compact device can be powered either via USB from a personal computer or by battery. With functionalities like LED displays, buttons, sensors, and wireless communication, the Micro:Bit's hardware is complemented by a user-friendly integrated development environment (IDE), as illustrated in Figure 2 [5]. The diverse programming examples available empower students to experiment with fundamental programming and electronic projects, extending their exploration beyond traditional classroom or computer lab settings.

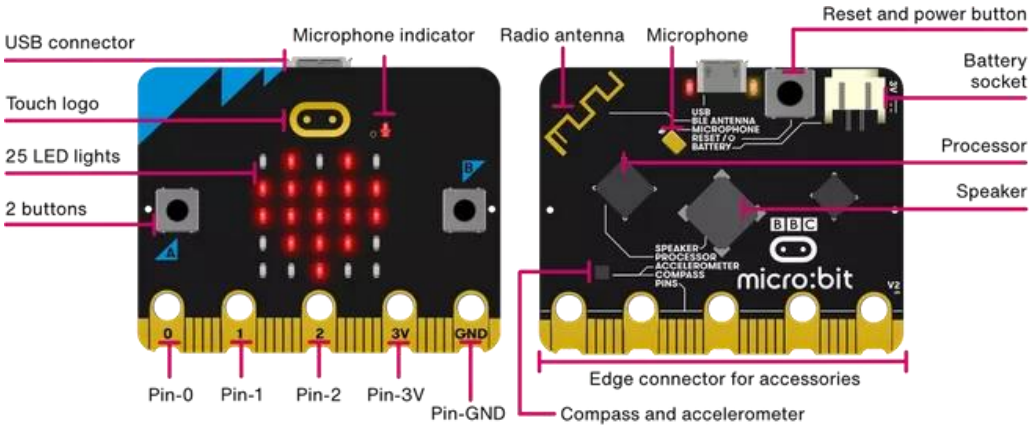


**Figure 1.** Front and back features overview of the BBC Micro:Bit V2. [5]

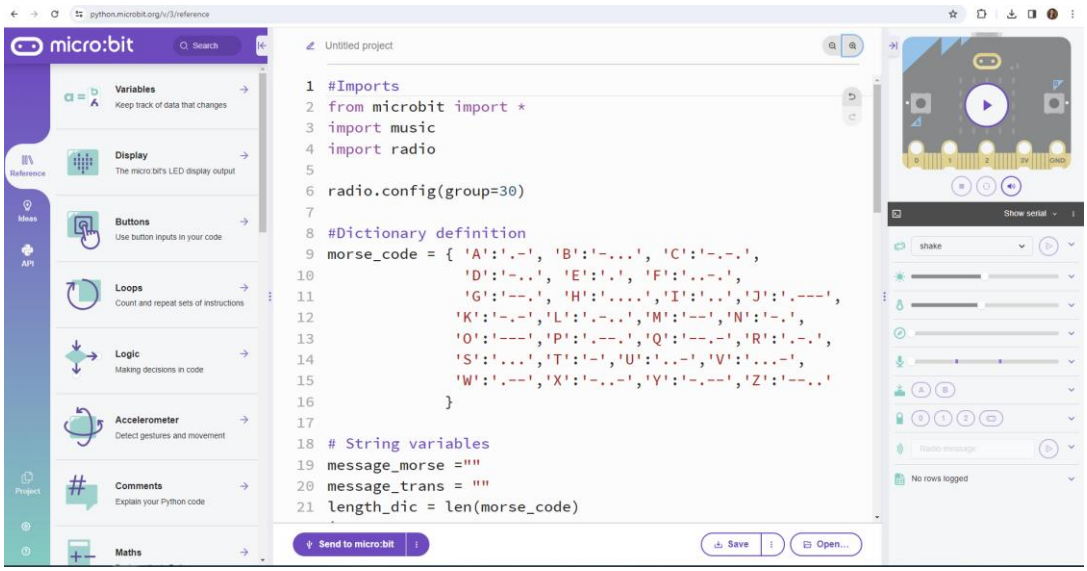


**Figure 2**. Python editor and simulator for the BBC Micro:Bit.

This paper delves into the implementation of a Micro:Bit code project tailored for freshman electrical engineering students. The project aims to bridge the gap between theoretical knowledge and practical application while nurturing essential Computational Thinking skills. The ensuing sections will include a synopsis of classroom activities leading to the code project, a detailed description of the project (covering objectives, required tools, and instructions), grading rubrics, and students' scores. Reflective insights from students highlight not only the successful creation of functional Morse code machines but also a profound appreciation for the practical applications of Computational Thinking and programming in the realm of Electrical Engineering.

## Classroom Activities Leading to Project

In the second part of the class, students were introduced to Python and various development environments, including Spyder from Anaconda 3, as well as the Python editor and simulator for the BBC Micro:Bit. Following the completion of each class module or topic, students were assigned brief programming exercises. These exercises required the use of Spyder to reinforce general Python syntax or the Micro:Bit Python editor to apply learned syntax in conjunction with libraries and physical inputs and outputs from the Micro:Bit V2. Additionally, the Micro:Bit classroom feature was employed during class sessions to facilitate the sharing coding between instructor and students and to monitor their individual progress [6]. An exemplar of one such classroom activity that served as a precursor to the coding project is illustrated in Figure 3.

## Code Project Description

The primary objective of the project was for students to demonstrate proficiency in designing computational solutions and employing programming concepts and tools using Python/MicroPython and the Micro:Bit. The essential tools for the project comprised a Micro:Bit and the online Micro:Bit Python editor. The project was subdivided into three components: (1) a coding segment, (2) a written report, and (3) a project demonstration, aiming to offer design experience akin to a Capstone class, where comprehensive design documentation is crucial.

**First 40 minutes of class:**

- Read the "Choose your own adventure" section and choose one of the options.
- Create the chosen function and the program that will use it.
- Test your code!

**Last 10 minutes of class:**

- In one or two minutes, present your code to the class. Indicate which adventure you choose and explain your code.

**Choose your own adventure:**

Option 1: Lights
Create a function that will count steps using the accelerometer sensor. In the main body of the code, use your function and add instructions that will reset the step count to 0 when you press a button.

Option 2: Microphone
Create a function that will turn on the lights if the microphone detects a loud sound (like a clap) and then turns off the light if another loud sound is detected. In the main body of the code, use your function and add instructions that will count the number of times the light has been turned on, and display it before turning off the light.

Option 3: Temperature
Create a function that takes the temperature of the room when a button is pressed and compares it with the last reading. In the main body of the code, use your function to determine if the temperature is warmer or colder than before. Display the words "Warmer" or "Colder" depending on the results of the comparison between temperatures.

Option 4: Magnetometer
Create a function that displays if the Micro:Bit is pointing North, South, East, or West. In the main body of the code, use the function if a button is pressed. You can also make a distinct sound when it's pointing in a particular direction.

**Figure 3.** Example of a classroom activity that required the used of the Micro:Bit development environments.

Students were assigned the task of developing a Morse Code Machine. Despite Morse Code being commonly perceived as an outdated communication method, it still finds application in specific areas, notably as an alternative means of communication in emergency situations. Military personnel, for instance, are taught Morse Code as a backup communication method during emergencies.

For the coding component, students received instructions, as depicted in Figure 4, outlining design requirements while allowing them the flexibility to approach their coding assignment in any manner they preferred. Further instructions were provided for the written report, along with a report template, prompting students to cover aspects such as (1) an introduction to the project, (2) a list or description of general project components (including the programming language, hardware, and programming interface), (3) a description of the Micro:Bit components (including the sensors/inputs/outputs used in their project and their utilization within the code), (4) a breakdown of code components (variables, functions, loops, branches, etc.) with accompanying flowcharts, (5) an explanation of their testing procedures, detailing how they tested their code, and (6) a conclusion encapsulating lessons learned.

Your code should be able to:

- Take Morse code messages using the push buttons (push button 'a' and push button 'b'), where one is used to input dashes and the other one is used to input dots.
- Provide feedback in the form of sound for each input dash and dot.
- Once a message has been completely inputted, display the decoded message using the alphabet and send that message via radio.
- Provide feedback to the users if they have inputted an invalid Morse code that can't be associated with a letter or a number.
- Translate Morse code into words and transmit translated messages over radio to another Micro:Bit.

You are allowed to use any of the other sensors (the capacitive touch sensor, the accelerometer, etc.) to make your code more robust and flexible.

Your code should have the following code components:

- Contain at least 1 While and 1 For loop.
- Contain if-else statements.
- Contain at least one function declaration and a function call.

Your code should be completely documented.

You should test your code using the Micro:Bit simulator.

For testing the radio transmission, ask for a classmate's help. This is the only part of the project where you are allowed to "collaborate" with another student.

**Figure 4**. Coding component instructions.

Upon project completion, students were mandated to showcase their code functionality, including the flowcharts, to their professor. This demonstration included to show the ability to decode and transmit a message to another Micro:Bit. Students were granted a two-week timeframe for project completion.

## Grading Rubrics

The grading rubrics that were used to evaluate students' projects can be seen in Figures 5, 6, and 7. These rubrics were distributed to students for their reference when the project description was provided. The project held a total value of 100 points, with 50 points allocated for the code, 35 points for the written report, and 15 points for the project demonstration.

## Results

In this class, a total of 17 students were enrolled when the project was conducted. The average scores were as follows: 43.65 ± 6.75 for the Code, 30.51 ± 5.19 for the written report, and 12.74 ± 3.62 for the project demonstration. Figure 8 provides a visual representation of the score distribution for each component (code, written, demonstration) as well as the overall score distribution. The average overall project score was 86.89 ± 13.83. Specifically, one student achieved a score in the range of 40-60 out of 100, three students scored between 60-80 out of 100, and 13 students earned scores in the range of 80 – 100 out of 100.

| Criterion | Max. Points | Score | Comments |
|---|---|---|---|
| **Design of output (20%)** | | | |
| Program displays correct output (no bugs) | **2** | | |
| Provide feedback in the form of sound to users for each dash or dot that is inputted | **2** | | |
| Display the decoded message using the alphabet | **2** | | |
| Send that message via radio | **2** | | |
| Provide feedback to the users if they have inputted an invalid Morse code | **2** | | |
| **Require components (20%)** | | | |
| At least 1 While loop | **2** | | |
| At least 2 For loop | **2** | | |
| if-else statements | **2** | | |
| At least one function declaration | **2** | | |
| A function call | **2** | | |
| **Design of logic (20%)** | | | |
| Program is logically well designed | **10** | | |
| **Standards (20%)** | | | |
| Program is stylistically well designed | **5** | | |
| Variables, functions, imports, and main code are well positioned throughout the code. | **5** | | |
| **Documentation (20%)** | | | |
| Program is well documented | **10** | | |
| | | | |
| **Total Score** | **50** | | |

**Figure 5**. Code evaluation rubric.

A sample of the students' reflections expressed in their written report as part of their conclusion are shown below:

- Reflection sample 1: "This project was definitely a challenge for me. I struggled in the beginning with the coding because I felt lost and wasn't sure what to do. Over time, I began to think it through and found a way to successfully make the desired code. This project was a lot of fun, as I thoroughly enjoyed showing it to my friends, and impressing them with my device. I also enjoyed the problem-solving aspect of it, and how it was left up to me to use my knowledge from the semester to create a code of my own. I was able to successfully accomplish the goal of the project, and created a working morse code transmitter/translator, and I learned several things that initially I was unsure of. I learned more about for loops and while loops especially, and seeing them at work in my microcontroller helped me gain a better understanding of their practical usage. Overall, this project was a huge success, and I thoroughly enjoyed it."

- Reflection sample 2: "I learned a lot about syntax, logic and optimization throughout the time working on this project… I learned about the difference between Python and MicroPython as well since many of the resources that I found online for Python were not compatible with Micro:Bit."

- Reflection sample 3: “The project demonstrated the multifaceted process required to create a functional product. Each stage of creating the product required a methodical approach to ensure quality and reliability. After completing each stage, the final product met all the requirements described in the instruction except for requiring a For loop in the code. My method of organizing the interaction between each component did not need a For loop to be implemented. From this I learned that each program can be made in a variety of ways. Also, the planning of your overall structure and functions helped a lot

| Criterion | Max. Points | Score | Comments |
|---|---|---|---|
| **Technical content (77%)** | | | |
| All requested deliverables included | **4** | | |
| Introduction and conclusion written as described in the report template | **5** | | |
| General project components described as requested in the report template | **3** | | |
| Micro:Bit components described as requested in the report template | **4** | | |
| Code components and description described as requested in the report template | **5** | | |
| Testing procedure described as requested in the report template | **3** | | |
| Appropriate level of detail and thoroughness of documentation | **3** | | |
| **Organization (15%)** | | | |
| Information is presented in effective order. Excellent structure of paragraphs and transitions enhances readability and comprehension. | **5** | | |
| **Presentation (8%)** | | | |
| Good grammar and style | **3** | | |
| | | | |
| **Total Score** | **35** | | |

**Figure 6**. Written report evaluation rubric.

| Criterion | Max. Points | Score | Comments |
|---|---|---|---|
| **Functionality of Code Demonstration (5%)** | | | |
| Code performs as it should (Decodes and transmit a message) | **5** | | |
| **Presentation (10%)** | | | |
| Explanation of how project objectives were achieved was included | **4** | | |
| Explanation of flowcharts was included | **3** | | |
| The student demonstrates full knowledge of code functionality | **3** | | |
| | | | |
| **Total Score** | **15** | | |

**Figure 7**. Project demonstration rubric.

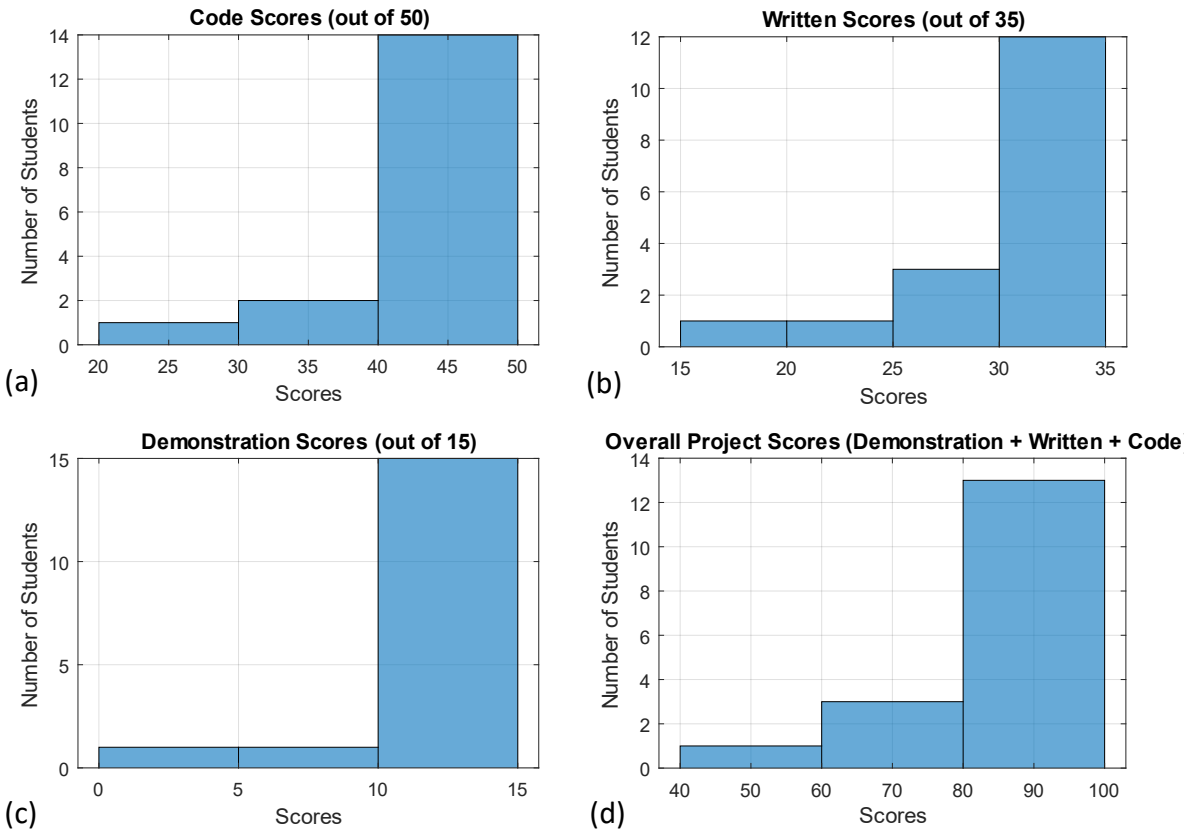


**Figure 8**. Project scores.

when coding and even more when debugging/testing. Parts that I learned the most from are habits that assist with keeping everything in order, such as spacing while coding makes it easier to read and minimize errors, and making sure to create back up saves of code during debugging because when you fix one thing you might break another."

- Reflection sample 4: "This project seems to be a great summary of all the topics we have covered this semester. We were able to use the code we have been learning to apply it to a usable functioning device. I was able to complete all the objectives of the project in my code. One of the things that I learned is that the more ways you see your code, for example flowchart or writing about your code or coding or commenting code, the better you understand your code. When I was making my flowchart, it made me relook at my program and try to better organize it and clear out some of the things that were more redundant.

- Reflection sample 5: "While completing this project, I learned a good amount of information, converting Morse code into text and then transmitting/ receiving the message over the micro-bit processor. I have always been interested in Morse code. It is a handy way of sending a message without any other means of technology. I have been using Python for a while now, but being able to translate it into a real-world scenario has made a massive difference in my interpretation of using code…"

- Reflection sample 6: "…This project taught how to implement python into a real-world application. This is especially important for Electrical/Computer Engineers because it takes the things that are learned in classes and puts them into practice with things, they make work on in the future."

**Conclusion**

In conclusion, the integration of the Micro:Bit code project into the freshman electrical engineering curriculum fosters computational thinking skills and emphases the practical applications of programming within the engineering design process. The project, centered around the creation of a Morse Code Machine, provided students with a tangible and engaging context to apply their programming knowledge and problem-solving skills acquired during the introductory programming class. The results indicate that the majority of students not only successfully completed the project but also achieved high scores, reflecting a solid grasp of computational concepts and programming proficiency.

The reflections from students further affirm the positive impact of the project on their learning experiences. Students expressed a newfound appreciation for the practical applications of Python and MicroPython, recognizing the relevance of coding skills in real-world scenarios. The project's structured components, including coding, written reports, and project demonstrations, facilitated a comprehensive understanding of the design process, mirroring the expectations of a capstone class.

Furthermore, the diverse reflections highlight the multifaceted learning outcomes, encompassing syntax understanding, logic, optimization, and the ability to apply Python concepts to a tangible device. The iterative nature of the project encouraged students to refine their coding practices and reinforced the importance of organization and planning in programming projects.

In essence, the Micro:Bit code project has proven to be an effective pedagogical tool in motivating and engaging freshman electrical engineering students, bridging the gap between theoretical knowledge and practical application. The project's success not only reinforces the significance of integrating tangible projects in programming education but also underscores the enduring relevance of computational thinking and programming skills in the evolving landscape of electrical engineering.

**References**

[1] J. Wing, "Computational Thinking," Communications of the ACM, vol. 49, no. 3, pp. 33-35, 2006.

[2] S. Bundy, "Computational Thinking is Pervasive," Journal of Scientific and Practical Computing, vol. 1, no. 2, pp. 67-72, 2007.

[3] M. J. Klopfer, "Using the Technology of Today, in the Classroom Today: The Instructional Power of Digital Games, Social Networking, Simulations and How Teachers Can Leverage

Them," in Handbook of Research on Effective Electronic Gaming in Education, IGI Global, 2008, pp. 1-16.

[4] M. Resnick et al., "Scratch: Programming for All," Communications of the ACM, vol. 52, no. 11, pp. 60-67, 2009.

[5] [BBC micro:bit webpage]. https://microbit.org/ (accessed Feb. 6, 2023).

[6] [BBC micro:bit classroom webpage]. https://classroom.microbit.org/ (accessed Feb. 6, 2023).