**Milestone Three**

**Enhancement 2 Narrative**

**Algorithms and Data Structure**

**Justin Byrd**

**justin.byrd@snhu.edu**

**CS-499-H7089 Computer Science Capstone**

**Professor Brooke Goggin**

**11/26/2023**

# Briefly describe the artifact. What is it? When was it created?

The artifact central to my enhancements is the "RGB LED Cube Control Software," a project initiated during my Electrical Engineering Technology Associate degree program's capstone in 2015. Initially, I constructed a 4x4x4 LED Cube with RGB LEDs, shown below, integrating it with an FPGA development board and crafting custom firmware to manipulate the 52 control lines governing the 64 LEDs.

A green box with wires

Description automatically generated

In a recent pursuit of rekindling my enthusiasm for the project, I designed additional circuitry to interface my RGB LED Cube Control Software with an Arduino Uno, facilitating control over the RGB LEDs. Prior to the start of this course, I had developed a simple functional user interface (UI), shown below, establishing communication with the Arduino Uno through a COM Port. This initial UI empowered me to illuminate one color of one LED based on user-defined color, X, Y, and Z values.

A screenshot of a computer screen

Description automatically generated

Having accomplished the hardware integration and circuitry enhancements, the focus of my Computer Science Capstone project pivots decisively to the evolution of the RGB LED Cube Control Software. With the foundational hardware components in place and the initial user interface demonstrating functional control over individual LEDs, my aim is to delve into advanced features and optimizations within the software realm. This shift marks the latest chapter in the ongoing development of a project that seamlessly marries my technical skills in both hardware and software domains.

# Justify the inclusion of the artifact in your ePortfolio. Why did you select this item? What specific components of the artifact showcase your skills and abilities in algorithms and data structure? How was the artifact improved?

The inclusion of the RGB LED Cube Control Software in my ePortfolio is justified by the substantial improvements introduced during the second Enhancement phase of development. This phase delves into the intricacies of software architecture and algorithmic design, showcasing a range of advanced software development skills.

## Cube Control Tab Functionality:

A screenshot of a computer

Description automatically generated

The augmentation of the Cube Control Tab functionality reveals my expertise in graphical user interface design. The introduction of 64 comboBoxes, each representing an RGB LED on the cube, underscores my ability to manage and implement dynamic user interfaces. The incorporation of color selection options and a pattern duration input demonstrates a keen focus on user interaction and functionality.

## New Classes to Support Cube Control Tab:

The creation of the led and cubeControl classes represents a significant architectural improvement. The led class encapsulates the properties and behaviors of individual RGB LEDs, emphasizing the principles of encapsulation and modularity. Meanwhile, the cubeControl class serves as a comprehensive representation of the entire 4x4x4 RGB LED Cube, showcasing my proficiency in designing and implementing complex data structures.

## Cube Timing Algorithm:

The Cube Timing Algorithm addresses critical challenges in balancing power consumption and communication speed. My ability to optimize the timing of control line manipulations demonstrates a deep understanding of hardware constraints. The solution involves a meticulous consideration of power draw limitations, communication speed challenges inherent in serial communication, and the implementation of advanced techniques such as Arduino port manipulation to achieve the desired visual effect.

## Timing Operation of the cube:

The Cube Timing Algorithm exhibits a strategic approach to managing power consumption and communication speed, showcasing a profound awareness of the intricacies involved in hardware manipulation. The algorithm's primary objective is to efficiently control the state of a 4x4x4 RGB LED cube over a specified duration. To achieve this, the algorithm engages with the constraints of the hardware by utilizing serial communication through an Arduino board.

The core mechanism involves cyclically lighting individual slices of the cube in a manner imperceptible to the human eye, creating the illusion of a fully lit cube. This approach not only considers the visual impact but also addresses the limitations of hardware speed and power consumption. The algorithm employs a time-driven execution loop to control the duration of the visual effect, ensuring optimal power usage and preventing unnecessary strain on the hardware.

In terms of time complexity, the nested loops iterating through the 16 [x][\*][z] slices contribute to a time complexity of O(n^2), where 'n' represents the cube size. This design choice aligns with the fixed size of the RGB LED cube, demonstrating a conscious effort to balance computational efficiency with the specific requirements of the hardware.

Furthermore, the integration of Arduino port manipulation and careful synchronization with the COMPORT communication adds a layer of sophistication. The algorithm's consideration of hardware speed disparities, as evidenced by the strategically placed wait statements, ensures that the RGBLEDCCS operates harmoniously with the COMPORT, preventing data buffering issues.

The Cube Timing Algorithm exemplifies a thoughtful and skillful approach to addressing challenges inherent in hardware control. The strategic balance between time complexity, power consumption, and communication speed underscores a meticulous consideration of the project's unique constraints, showcasing an advanced understanding of both algorithmic and hardware principles.

## Managing Pattern Duration:

To control how long a pattern remained illuminated on the cube, a sophisticated timestamp mechanism was introduced. Before entering the execution loop, a timestamp was captured, and a second timestamp was recorded at the conclusion of each cube iteration. By calculating the time difference between these two timestamps, the algorithm determined whether to continue or exit the execution loop based on the user-specified duration. This meticulous time management further refined the Cube Timing Algorithm, providing precise control over the display duration for each pattern on the RGB LED Cube.

## Data Parsing Algorithm:

The Data Parsing Algorithm serves as a pivotal component in the RGB LED Cube Control Software, laying the groundwork for seamless integration with a forthcoming database module in Enhancement 3. The implementation of this algorithm not only showcases a forward-thinking approach to software design but also underscores a commitment to crafting extensible and adaptable solutions.

### Implementation Details:

#### Function Design:

To facilitate the future loading of cube patterns from a database, a specialized function was meticulously crafted. This function accepts a 64-character string, with each character representing integer values ranging from 0 to 7. These integer values correspond to the predefined colors (black, red, green, blue, yellow, purple, cyan, white) for the RGB LEDs.

#### String Parsing:

Upon receiving the 64-character string, the algorithm systematically parses each character, converting it into an integer value. This parsing operation is crucial, as it establishes a direct mapping between the database representation of cube patterns and the internal color representation within the software.

#### LED Object Update:

With the parsed integer values, the algorithm proceeds to update each of the 64 LED objects within the cubeControl class. This step ensures that the internal state of the RGB LED Cube accurately reflects the pattern specified by the database. Each LED is assigned the color corresponding to its designated integer value, aligning with the predefined color scheme.

#### UI Synchronization:

The Data Parsing Algorithm goes beyond internal updates; it extends its influence to the graphical user interface (UI). The parsed integer values not only dictate the colors of the LED objects but also dynamically update the choices and colors of the corresponding combo boxes in the UI. This synchronization ensures that the user interface accurately mirrors the loaded cube patterns, providing a cohesive and intuitive user experience.

#### Time Complexity:

In terms of time complexity, the Data Parsing Algorithm demonstrates efficiency with a linear time complexity of O(n), where 'n' represents the size of the input string (64 characters). The algorithm's design ensures that each character is processed individually, contributing to a linear relationship between the input size and the execution time. This linear scaling is particularly advantageous for managing larger datasets, showcasing a pragmatic consideration of potential future expansions in database patterns. The algorithm's ability to parse and update the cube's internal state with a linear time complexity aligns with best practices in algorithmic efficiency, reinforcing the software's responsiveness and adaptability.

In summary, the enhancements made in the second phase of the RGB LED Cube Control Software project underscore my advanced skills in software development. From intricate UI design to the implementation of complex algorithms addressing power consumption and communication speed challenges, this artifact serves as a testament to my ability to conceive, design, and implement sophisticated software solutions. The ongoing evolution of this project aligns with my dedication to continuous improvement and innovation in the realm of software development.

## Video:

A visual walkthrough encapsulated in the video linked below provides a dynamic exploration of these updates. The video not only discusses the intricacies of the software enhancements but also showcases the live functionality of the RGB LED Cube Control Software.

Link to video hosted on YouTube - [Enhancement 2](https://www.youtube.com/watch?v=M9_ifvqkotE)

# Did you meet the course objectives you planned to meet with this enhancement in Module One? Do you have any updates to your outcome-coverage plans?

## Outcome One:

The first outcome I met with this enhancement was, "Design and evaluate computing solutions that solve a given problem using algorithmic principles and computer science practices and standards appropriate to its solution, while managing the trade-offs involved in design choices".

### Managing Trade-offs:

In meeting Outcome one, I successfully navigated the delicate balance between communication speed and power consumption limitations inherent in my hardware. The Cube Timing Algorithm, a pivotal component, addressed critical challenges in optimizing the timing of control line manipulations. This required a nuanced understanding of hardware constraints, leading to the implementation of advanced techniques like Arduino port manipulation.

### Effective Timing Algorithms:

The design of effective timing algorithms was crucial to achieving the desired visual effects. Specifically, the Timing Operation of the cube involved strategic illumination of only four RGB LEDs at any given moment. This approach optimized power consumption and visual impact. However, to create an illusion of the entire cube being illuminated simultaneously, the control lines had to cycle rapidly through sixteen sets of four LEDs. This careful orchestration was central to conveying a unified and dynamic visual experience.

Moreover, to enhance user customization, I implemented a feature allowing patterns to stay illuminated for a user-defined duration. This meticulous consideration in timing algorithms provided users with control over the display duration for each pattern on the RGB LED Cube, aligning with Outcome one’s emphasis on managing trade-offs and design choices.

In evaluating the performance characteristics, the timing operation algorithm exhibits a time complexity of O(n^2), where ‘n’ represents the size of the cube, ensuring efficiency for the fixed cube size.

### Parsing Algorithm for Future Loading:

Furthermore, the Data Parsing Algorithm, a key element introduced, was designed to facilitate the future loading of cube patterns from a database. This forward-thinking solution required crafting a specialized function that systematically parsed a 64-character string, establishing a direct mapping between database representation and internal color representation within the software. This demonstrated a proactive approach to adaptability and extensibility in software design. In evaluating its performance characteristics, the algorithm exhibits a linear time complexity of O(n), ensuring efficient processing of input data.

## Outcome Two:

The second outcome I met with this enhancement was, "Demonstrate and ability to use well-founded and innovative techniques, skills, and tools in computing practices for the purpose of implementing computer solutions that deliver value and accomplish industry-specific goals"

### UI Design and Implementation:

In addressing Outcome two, the augmentation of Cube Control Tab functionality underscored my expertise in graphical user interface design. The introduction of 64 comboBoxes, each representing an RGB LED, showcased my ability to manage and implement dynamic user interfaces effectively. This innovative UI design enhances user interaction and functionality.

### Encapsulation and Modularity:

Additionally, the creation of the led and cubeControl classes represented a significant architectural improvement. Emphasizing principles of encapsulation and modularity, these classes provided a structured and organized approach to manage the properties and behaviors of individual RGB LEDs and the entire 4x4x4 RGB LED Cube, respectively. This showcased proficiency in designing and implementing complex data structures, aligning with Outcome 2's emphasis on well-founded and innovative techniques.

In summary, the second enhancement phase not only addressed the outlined outcomes but also showcased a comprehensive approach, integrating advanced algorithmic design, hardware optimization, and innovative UI implementation. The ongoing evolution of the project aligns seamlessly with my commitment to continuous improvement and innovation in the realm of software development.

# Reflect on the process of enhancing and/or modifying the artifact. What did you learn as you were creating it and improving it? What challenges did you face?

## What I Have Learned:

### BAUD Rates and Serial Communication:

Exploring ways to increase communication speeds led me to delve into the intricacies of BAUD Rates in serial communication. Understanding how data is transmitted between the software and Arduino Uno, and the impact of different BAUD Rates on speed, was crucial for optimizing the Cube Timing Algorithm. This experience provided valuable insights into the nuances of efficient serial communication, enhancing my proficiency in this fundamental aspect of hardware-software interaction.

### Ctime Library and Timing Algorithm:

The implementation of the Timing Operation of the cube involved incorporating the ctime library into the algorithm. This library, dealing with time-related functions, played a key role in achieving precise control over the display duration for each pattern. Learning about and leveraging the ctime library not only refined the Cube Timing Algorithm but also expanded my knowledge of using specialized libraries to enhance software functionality.

### QSerialPort Library:

My focus on enhancing communication speed also prompted a deeper understanding of the QSerialPort Library. Exploring its features and functionalities allowed me to make informed decisions about optimizing data transmission between the software and the Arduino Uno. Gaining proficiency in utilizing this library is a testament to the importance of leveraging existing tools to overcome challenges in software development.

### Commenting Standards:

Emphasizing good commenting standards and adopting clear comment headers during this enhancement phase underscored the importance of code documentation. Ensuring that my code is well-documented not only aids in my understanding of the logic during subsequent phases but also promotes collaboration and ease of maintenance. This focus on documentation aligns with industry best practices and reflects a commitment to producing high-quality, understandable code.

## Challenges faced:

### Power Draw Challenge:

In the initial stages of designing the Cube Timing Algorithm, a critical challenge surfaced — managing power draw while illuminating the RGB LEDs. Given the hardware specifications where only one RGB LED should be powered at any given time, exceeding this limit would lead to inadequate current for each LED, resulting in color fading or undesired hue shifts. This constraint necessitated a meticulous approach to ensure optimal power distribution among the LEDs.

### Communication Speed Challenge:

A fundamental requirement for the RGB LED Cube was the need to manipulate the control lines rapidly, creating the illusion that the entire cube is simultaneously illuminated. This presented a unique challenge due to the constraints of serial communication. Inherent delays in sending data to the Arduino resulted in a visual effect reminiscent of an older monitor with a slow refresh rate, leading to perceptible blinking rather than a seamless display.

### Power Draw vs. Communication Speed Solution:

To address the power draw challenge, a systematic testing process was conducted to determine the maximum number of LEDs that could be powered concurrently without compromising color integrity. The optimal solution emerged when it was found that lighting four RGB LEDs simultaneously was within the power constraints, reducing the number of control line manipulations from 64 (one per LED) to 16 (one per 4 LEDs). This not only conserved power but also streamlined the control process.

In tandem with optimizing power draw, a two-fold strategy was implemented to overcome communication speed challenges. First, the Baud Rate of the COMPORT was significantly increased, enhancing the rate at which data could be transmitted. Secondly, a critical adjustment was made to the Arduino Uno's software, employing port manipulation instead of the slower digital write functions. This dual-pronged approach proved effective in ensuring that control lines cycled swiftly enough to create a seamless visual experience for the observer, with the entire cube appearing constantly lit, devoid of any blinking or color issues.

### Addressing Serial Communication Buffering:

Even with these optimizations, a final challenge emerged — buffering issues in the QSerialPort. The software was operating at a pace faster than the COMPORT could handle, leading to a breakdown in the timing algorithm. The solution involved incorporating a waitForBytesWritten statement within the execution loop. This strategic addition ensured synchronization between the software and COMPORT, preventing data from being excessively buffered and enabling a harmonious operation of the Cube Timing Algorithm.

## Conclusion:

In summary, the second enhancement phase was instrumental in not only providing effective solutions to specific challenges but also in significantly expanding my knowledge in critical areas of serial communication, timing algorithms, library utilization, and code documentation standards. The challenges encountered, such as managing power draw and optimizing communication speed to achieve the desired visual effects, served as practical testing grounds for the theoretical concepts explored. These hands-on experiences deepened my understanding of the underlying principles and intricacies involved in real-world software development.