Faculty of Engineering and Architectural Science
Department of Computer and Electrical Engineering

Course Number	**COE718**
Course Title	**Embedded Systems Design**
Semester/Year	**Fall 2025**
Instructor	**Gul N. khan**
Teaching Assistant	

Report Title	**Final Project: Media Center Final Project**

Section No.	05
Due Date of Lab Performanc	
Report Submission Date	

Name	Student ID	Signature*
Hamza Malik	501112545	H.M

**By signing above you attest that you have contributed to this submission and confirm that all work you have contributed to this submission is your own work. Any suspicion of copying or plagiarism in this work will result in an investigation of Academic Misconduct and may result in a "0" on the work, an "F" in the course, or possibly more severe penalties, as well as a Disciplinary Notice on your academic record under the Student Code of Academic Conduct, which can be found online at: http://www.ryerson.ca/senate/policies/pol60.pdf. .*

ABSTRACT

This project presents the design, implementation, and verification of a real-time Embedded Media Center developed on the MCB1700 evaluation board. Leveraging the ARM Cortex-M3 microcontroller architecture and the Keil µVision Integrated Development Environment (IDE), the system integrates four distinct multimedia subsystems: a digital Photo Gallery, a USB-based Audio Player, an interactive Tic-Tac-Toe game, and a reaction-based Reflex Tester. The application demonstrates the successful management of constrained hardware resources, effectively employing Nested Vector Interrupt Controller (NVIC) routines, Analog-to-Digital Converter (ADC) peripherals, and Universal Serial Bus (USB) communication protocols to create a responsive and user-friendly experience. Validation testing confirms that all subsystems operate without latency or resource conflict, satisfying the course requirements for a complex, multi-threaded embedded system design.

INTRODUCTION

This project engineers a comprehensive, real-time Embedded Media Center utilizing the NXP LPC1768 microcontroller on the MCB1700 evaluation board. The objective was to synthesize multiple distinct multimedia functions into a single, cohesive system, simulating the architectural complexity of modern consumer electronics. The resulting application integrates four core subsystems: a digital Photo Gallery for bitmap rendering, a USB-based Audio Player for isochronous data streaming, and two interactive logic modules—Tic-Tac-Toe and a precision Reflex Tester. The technical development was conducted within the Keil µVision Integrated Development Environment (IDE), leveraging the specific capabilities of the ARM Cortex-M3 architecture. Key engineering focus was placed on the direct manipulation of hardware peripherals, including the Liquid Crystal Display (LCD) for graphical output, the Analog-to-Digital Converter (ADC) for real-time signal processing, and the Nested Vector Interrupt Controller (NVIC) for low-latency user input handling. Beyond individual module functionality, this project addresses the critical challenge of resource optimization in constrained embedded environments. By implementing a modular software architecture, the system demonstrates how to manage concurrent tasks processing user I/O, rendering graphics, and handling USB protocols without performance degradation. This report details the design methodology, driver implementation, and integration strategies required to bridge the gap between abstract software logic and physical hardware constraints.

HARDWARE ARCHITECTURE AND SYSTEM DESIGN IMPLEMENTATION

The hardware foundation of the media center is the MCB1700 development board, which utilizes the NXP LPC1768 microcontroller. This specific System-on-Chip (SoC) was selected for its versatility and extensive peripheral support, which are essential for handling the diverse requirements of multimedia applications. The LPC1768 operates at 100 MHz, providing sufficient clock cycles to handle the refresh rates required for the LCD while simultaneously polling for user input. The system architecture is designed around a central processing loop that interfaces directly with several onboard hardware components to facilitate user interaction and data output.

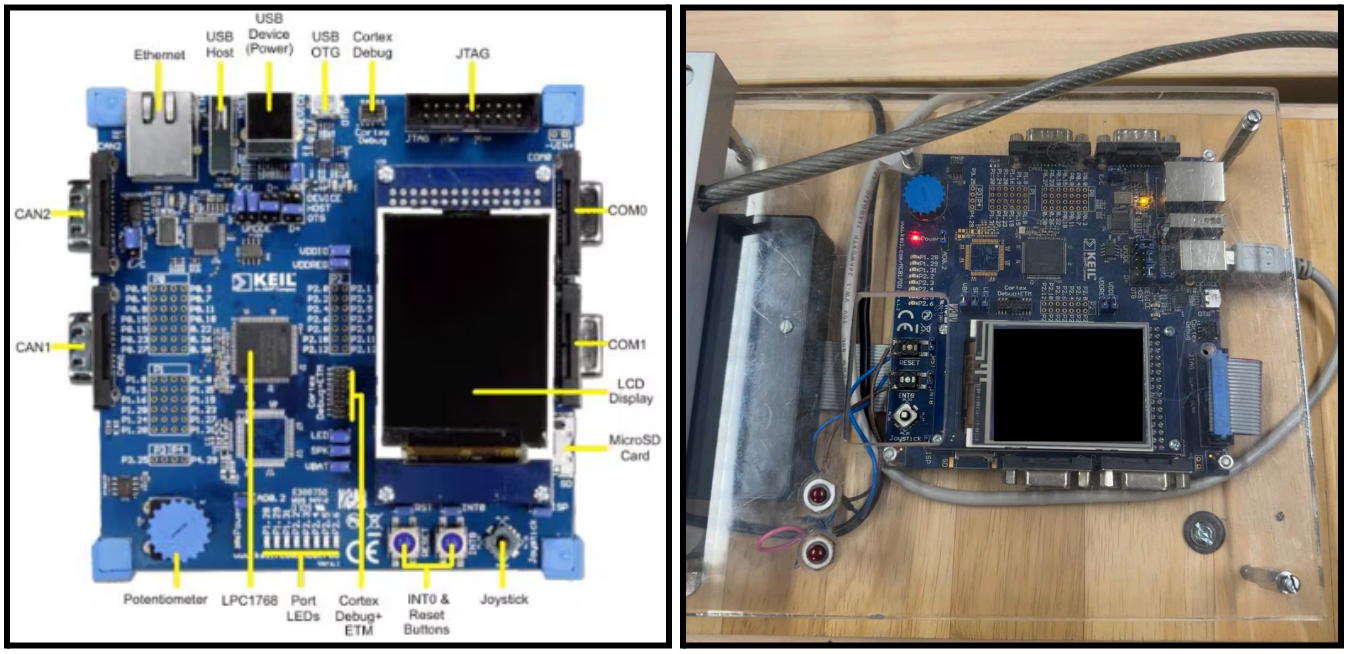

Figure 1: *The MCB1700 Hardware Architecture, illustrating the layout of the NXP LPC1768 MCU and its interface with the LCD, Joystick, and USB subsystems.*

The primary output device is the 320x240 pixel colour LCD (GLCD), which is responsible for rendering the Graphical User Interface (GUI). This screen is driven by a specialized interface that requires precise timing to prevent flickering or tearing artifacts during image transitions. The screen acts as the primary feedback mechanism for all modules: displaying the menu options, rendering high-resolution photographs, and drawing the dynamic game grids. The ability to control individual pixels on this display was paramount to the project's implementation.

User input is captured through two primary mechanisms: the directional joystick and the analog potentiometer. The joystick serves as the main Human Interface Device (HID), allowing the user to traverse menu hierarchies and control dynamic elements within the games. It is mapped to the standard

directional inputs (North, South, East, West) and a central selection button. The software interprets these digital inputs to update the system state, such as moving a cursor or checking for a button press during the Reflex Test. Complementing this, the onboard potentiometer functions as a dedicated control interface for the audio subsystem. The continuous analog voltage signal from the potentiometer is processed by the microcontroller's 12-bit Analog-to-Digital Converter (ADC). This peripheral translates the physical rotation of the knob into a digital integer value between 0 and 4095, which is then used by the software to adjust the audio volume levels dynamically.

SOFTWARE DESIGN AND IMPLEMENTATION

The software architecture is constructed around a Finite State Machine (FSM) embedded within a continuous polling loop. This design pattern guarantees that the system occupies exactly one valid state at any given time—Main Menu, Photo Gallery, Audio Player, Tic-Tac-Toe, or Reflex Tester—preventing undefined behavior or invalid transitions. Upon startup, the system initializes to the Main Menu. The firmware continuously polls the joystick status; when a "Right" input is detected on a highlighted selection, the FSM triggers a transition handler that clears the display buffer and initializes the specific assets required for that module.

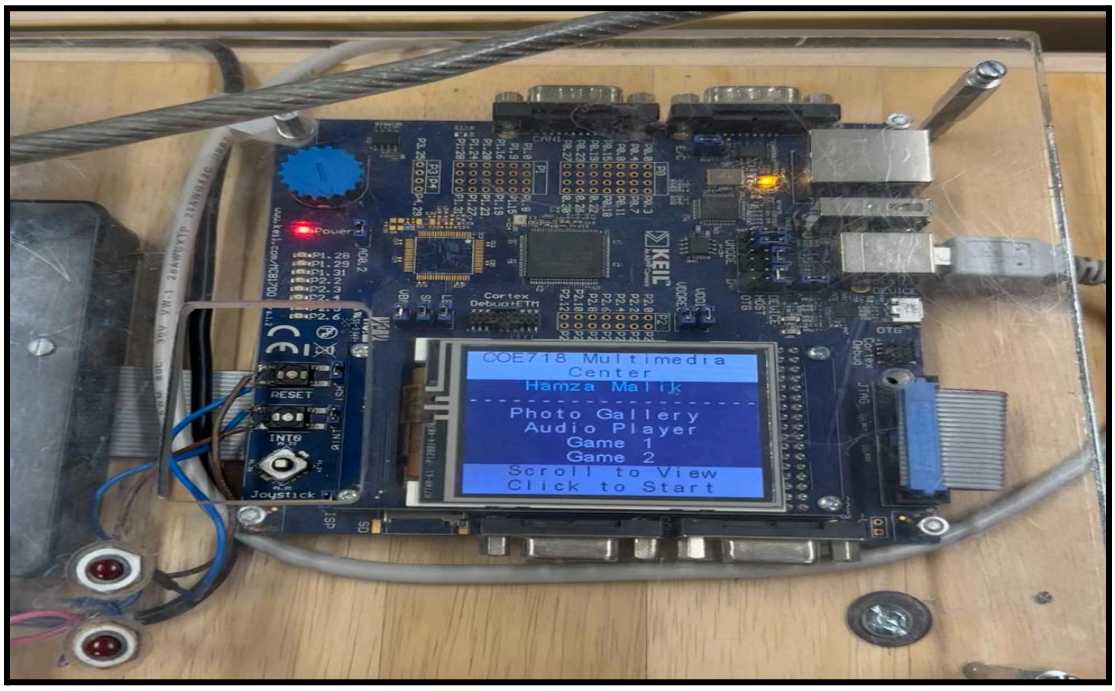

Figure 2: *The Main Interface Menu for* MCB1700 *NXP LPC1768 Microcontroller Multimedia Center*

A. Photo Gallery Module The Photo Gallery subsystem is responsible for the retrieval, management, and rendering of high-resolution bitmap assets. This module serves as a critical demonstration of the system's ability to manipulate the Liquid Crystal Display (LCD) frame buffer while managing the finite storage constraints of the on-chip Flash memory.

1. Interface Design and State Management Upon selecting the module from the main hub, the system transitions into a dedicated "Visual Mode." The user interface (UI) architecture shifts from a grid-based layout to a linear hierarchical list. As illustrated in the system screenshot, the menu presents three distinct asset slots (*View Image 1, View Image 2, View Image 3*) alongside navigation controls to *Return Home*. The firmware continuously polls the joystick's vertical axis to update the highlighted selection, providing immediate visual feedback to the user before a state transition is confirmed.

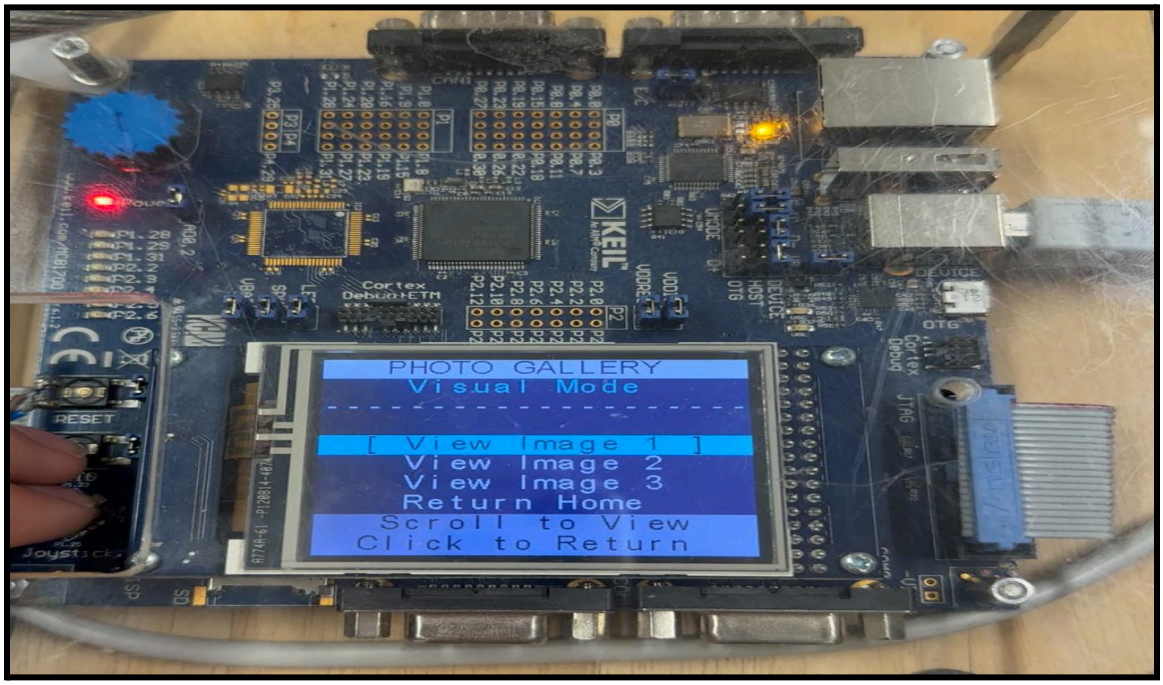

Figure 3a: *The 'Visual Mode' selection menu, demonstrating the hierarchical list structure and navigation prompts.*

2. Bitmap Rendering and Asset Optimization The system architecture eliminates the need for an external file system by embedding graphical assets directly into the microcontroller's firmware. All bitmap images are serialized into hexadecimal C-arrays and encapsulated within the on-chip Flash memory (ROM). The rendering pipeline employs a coordinate calculation routine that dynamically centers the image on the display, ensuring consistent alignment regardless of the source asset's dimensions.

The efficacy of this C-code generation process was validated using three distinct image categories:

- **Photographic Data ("Hamzas Fit"):** Evaluates the rendering of complex pixel variations and shading.
- **Vector Graphics ("Jerry"):** Verifies the accurate reproduction of solid colours and sharp definitions.
- **Iconography ("Music Note"):** Tests the contrast and clarity of simple geometric shapes.

These test cases confirm that the array conversion and rendering logic perform correctly across a spectrum of graphical formats.

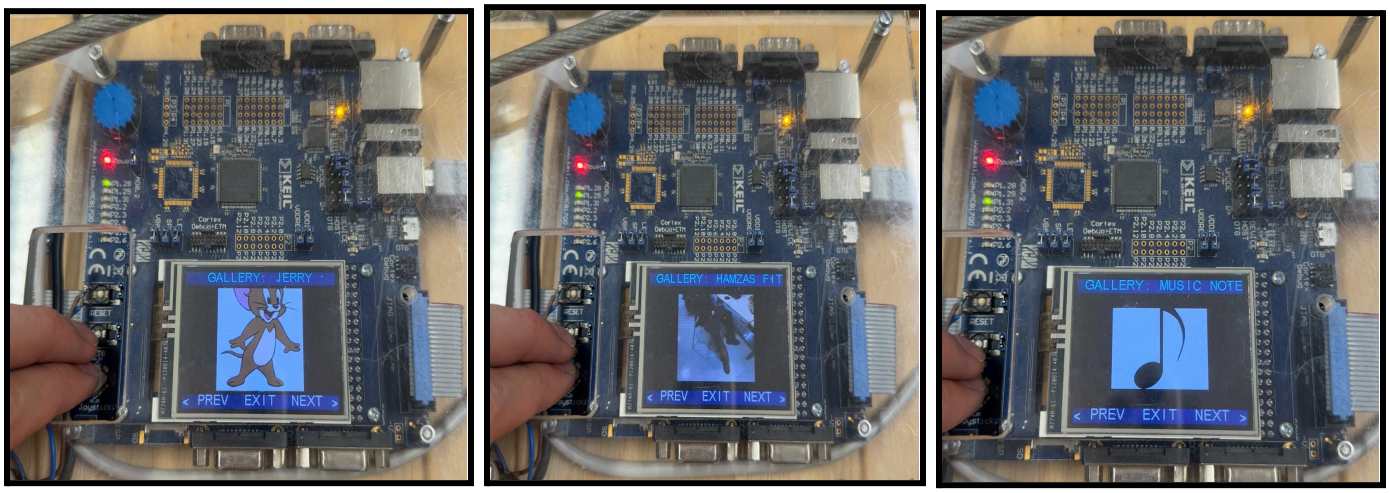

Figure 3b: *Implementation of C-array rendering across photographic, vector, and icon assets.*

B. Audio Player Module The Audio Player module configures the MCB1700 to function as a standard **USB Audio Class** device. This implementation transforms the evaluation board into a universal audio interface, enabling plug-and-play compatibility with host operating systems without the requirement for proprietary drivers.

1. Universal Media Streaming Adhering to standard isochronous transfer protocols, the system allows for real-time audio streaming from any standard media source. During verification, the device successfully decoded PCM audio streams from web-based platforms (such as *YouTube*) and desktop applications (such as *Spotify*), effectively routing the host's system audio through the microcontroller's output peripherals.

2. Analog Control and Visual Feedback To provide a tactile user experience, the onboard **analog potentiometer** is utilized as the primary gain controller. The firmware continuously samples the voltage

across the wiper via the 12-bit Analog-to-Digital Converter (ADC). This digital value is then mapped to two concurrent feedback mechanisms:

- **LCD Interface:** A dynamic graphical bar renders on the display, expanding and contracting in real-time to provide precise visual confirmation of the volume level.
- **LED Array (Linear Scale):** The board's bank of GPIO-controlled LEDs functions as a physical volume indicator. The LEDs illuminate sequentially based on the potentiometer's rotation; complete illumination of the array indicates maximum volume (0dB), while a single active LED indicates minimum volume or near-silence.

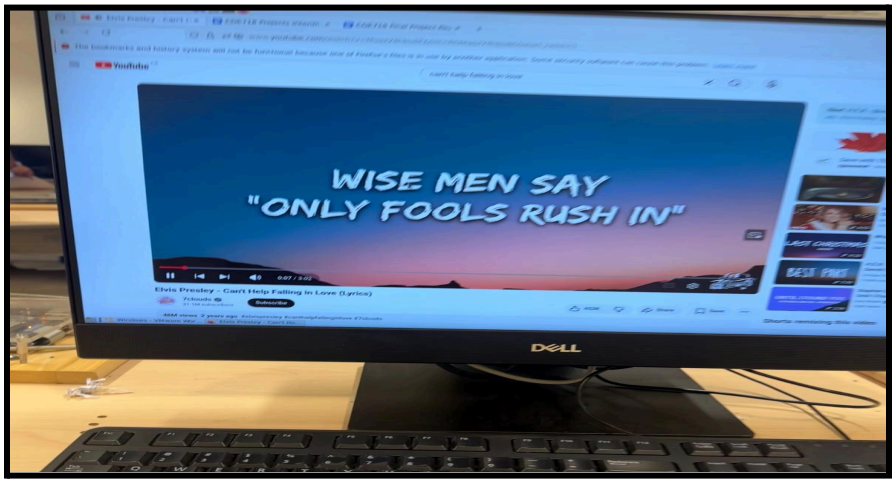

Figure 4a: *The Audio Subsystem in operation Streaming active content from YouTube.*

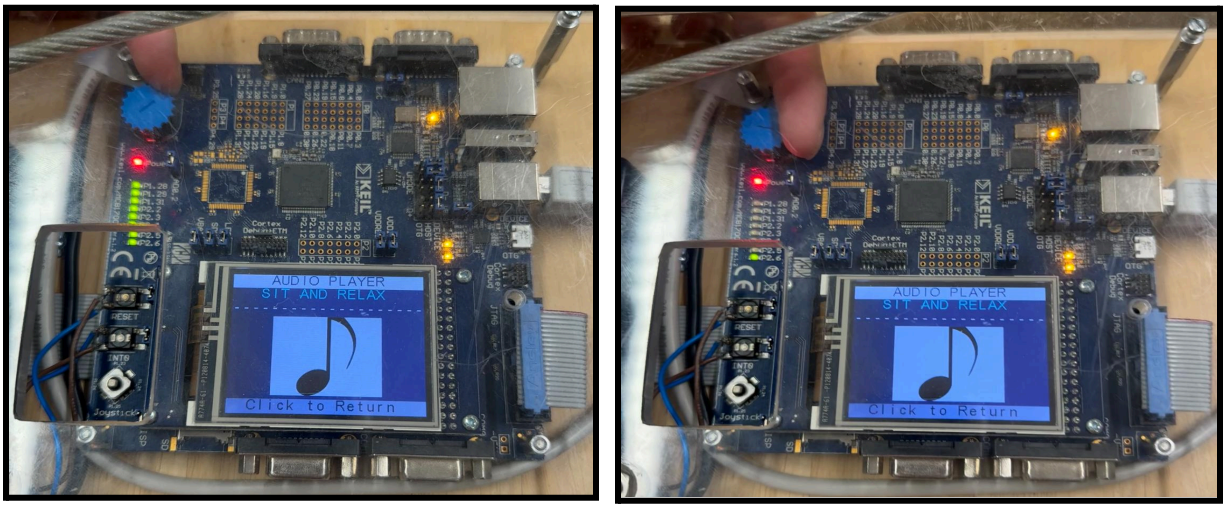

Figure 4b: *Manual volume adjustment utilizing the analog potentiometer, with the LED array fully illuminated to indicate minimum - maximum volume.*

C. Interactive Game 1 Module (Tic-Tac-Toe) The Tic-Tac-Toe module serves as the system's primary logic engine, implementing a complete two-player strategy game. Unlike the static rendering of the Photo Gallery, this subsystem requires dynamic memory management, real-time input validation, and continuous state evaluation.

1. Initialization and User Onboarding To ensure intuitive operation, the module does not immediately drop the user into the game grid. Instead, it follows a structured initialization sequence. First, a dedicated **Game Menu** allows the user to confirm their selection or return to the main hub. Upon confirmation, the system presents a **Rules & Controls** interface. This screen clearly articulates the input mapping ("Joystick to Move," "Click to Select") and the victory conditions ("Match 3 to Win"), ensuring the user understands the mechanics before play begins.

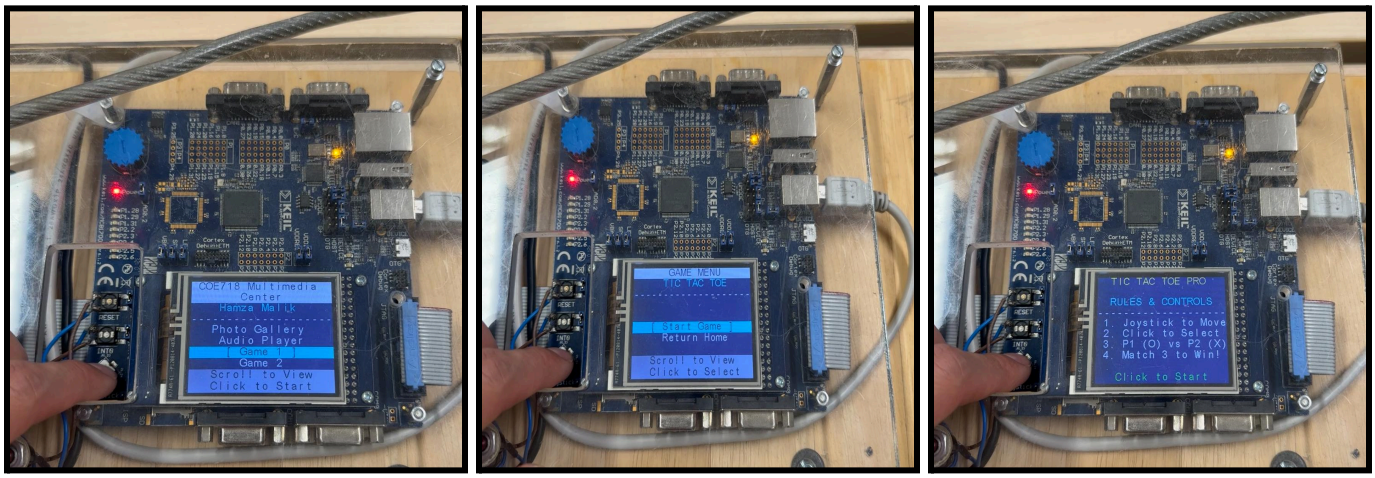

Figure 5a: *Game initialization sequence showing the sub-menu navigation and the instruction set overlay.*

2. Gameplay Mechanics and State Management The game board is represented in the system memory as a 3 by 3 integer matrix. The graphical engine renders this grid using vector primitives (lines) rather than bitmaps to optimize refresh rates.

- **Score Tracking:** The top corners of the display (labeled P1 and P2) update in real-time to track the session score, stored in static variables that persist between rounds.
- **Input Handling:** The joystick allows players to navigate the grid. Player 1 is assigned the "O" token, and Player 2 is assigned the "X" token. The system enforces turn-based logic, preventing a player from overwriting an occupied cell or moving out of turn.

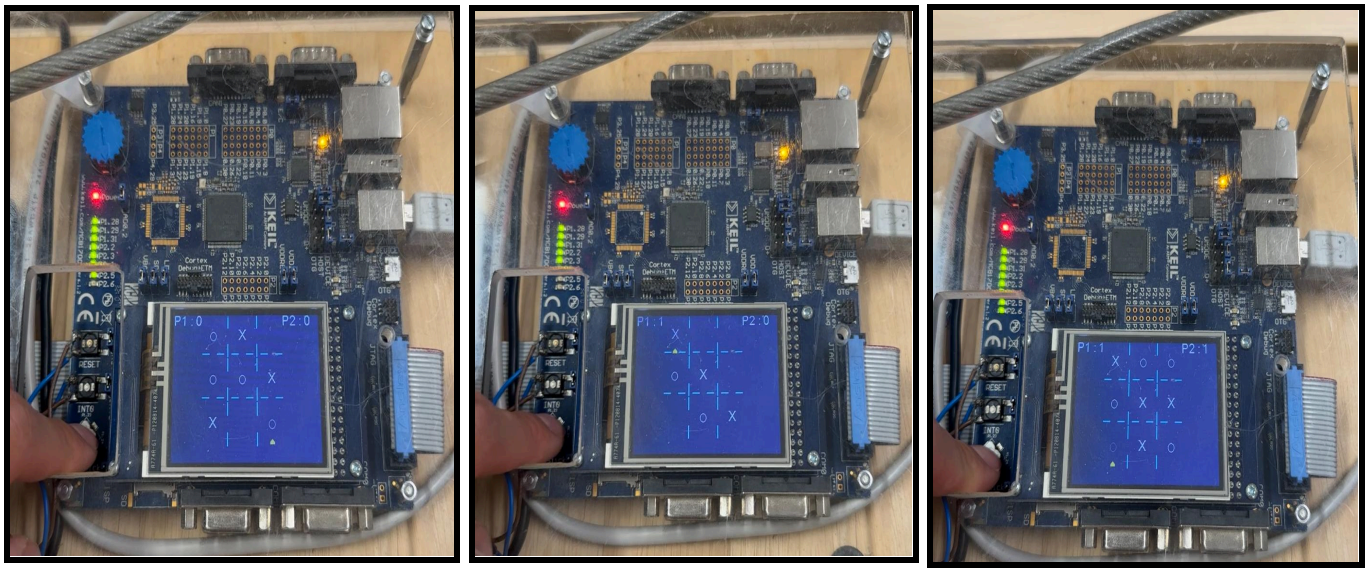

Figure 5b: *Active gameplay state showing the 3x3 grid, player tokens (X and O), and the real-time scoreboard at the top of the display.*

3. Logic Verification and End-States A critical engineering requirement was the deterministic detection of all possible game outcomes to ensure the integrity of the match. The firmware implements a validation routine that triggers immediately following every input event. This algorithm executes a comprehensive scan of the 3 x 3 matrix, evaluating eight distinct geometric vectors: three horizontal rows, three vertical columns, and two diagonal axes.

The system was rigorously stress-tested to verify that it correctly identifies and handles the three distinct terminal states without latency or logic errors:

- **Player 1 Victory Condition:** The algorithm detects a contiguous vector of 'O' tokens (Value: 1). Upon detection, the game state is immediately halted to prevent further input, and the global score variable for Player 1 is incremented.
- **Player** 2 Victory **Condition:** The algorithm detects a contiguous vector of 'X' tokens (Value: 2). Similar to the P1 condition, the input interrupts are disabled, and the Player 2 score is updated.
- **Tie Game (Stalemate) Condition:** This state represents a specific edge case where the total move counter reaches 9 (indicating a full grid), yet the vector scan returns no winning combinations. The system correctly identifies this as a draw rather than a standard game-over error.

Post-Game State Transitions: Upon reaching any of these terminal states, the system triggers a **blocking modal overlay**. This interface interrupts the main game loop to announce the result ("Player 1 Wins", "Player 2 Wins", or "Tie Game") and waits for a specific user acknowledgement (Click to Continue). This design ensures the user has time to review the final board state before the firmware clears the memory matrix and resets the grid for the subsequent match.

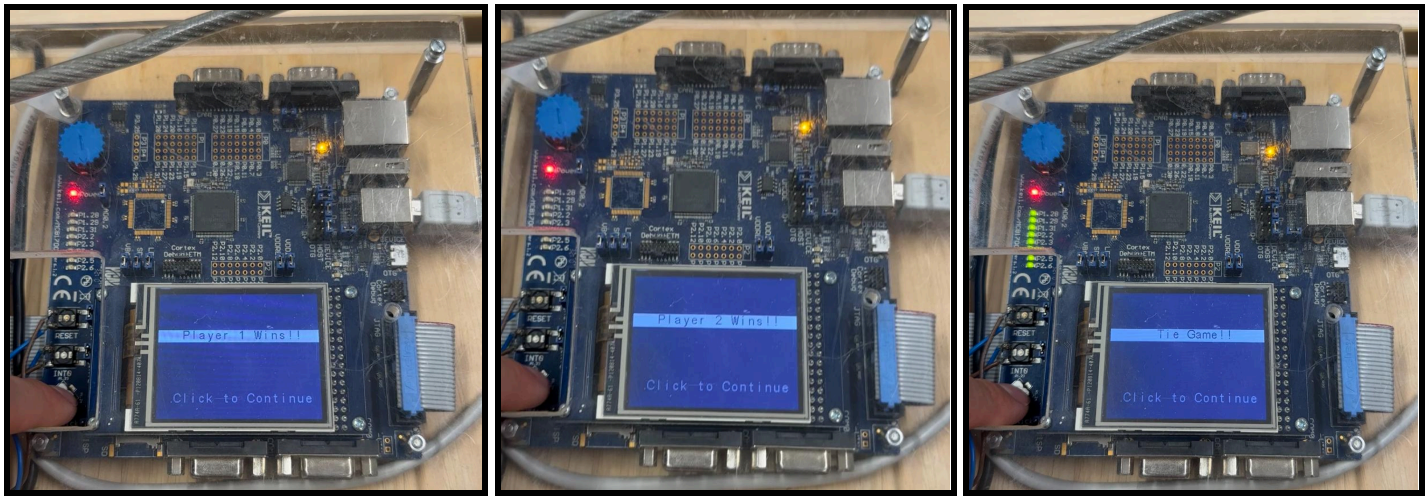

Figure 5c: Verification of the game logic engine, demonstrating the correct identification of Player 1 Victory, Player 2 Victory, and Tie Game states.

D. Reflex Tester Module ("Reflex Hunter") The Reflex Tester module serves as the system's primary demonstration of hard real-time constraints and interrupt latency management. Titled **"Reflex Hunter,"** this subsystem is engineered not merely as a diagnostic tool but as a fully gamified utility that tests the user's cognitive reaction speeds against a stochastic stimulus.

1. User Onboarding and Gamification Strategy To enhance user engagement and provide clear performance metrics, the module utilizes a structured onboarding sequence. Upon selection from the Main Menu, the user is presented with two informational screens that define the competitive parameters of the session:

- **Scoring Logic:** Unlike a binary pass/fail system, the firmware implements a weighted scoring algorithm based on specific reaction-time windows.
    - **Perfect (< 200ms):** Awarded **+5 Points** for elite reaction speeds.
    - **Fast (200ms – 300ms):** Awarded **+3 Points** for standard human reaction speeds.
    - **Slow (> 300ms):** Awarded **+1 Point** for delayed responses.

    - **Mistake:** A penalty of **-1 Point** is applied for anticipating the signal (False Start) or pressing the incorrect directional input.
- **Ranking Hierarchy:** To provide long-term feedback, the system defines a grading scale ranging from **"S (GOD)"** for scores exceeding 20 points, down to **"F (FAIL)"** for scores below 5. This persistent ranking system validates the application's ability to maintain state variables across multiple iteration loops.

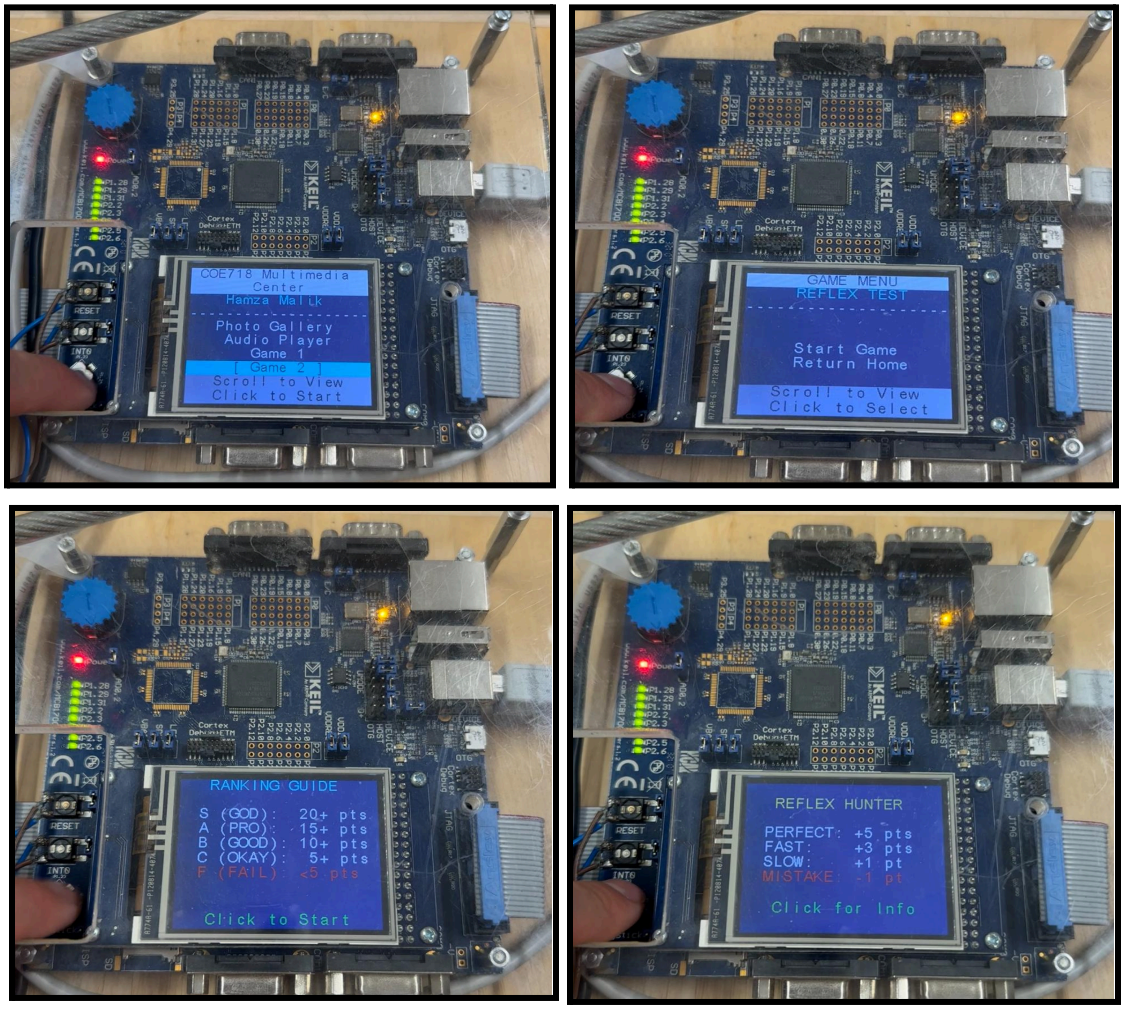

Figure 6a: Reflex Tester onboarding sequence, detailing navigation, competitive ranking tiers, and point-scoring logic.

2. Precision Timing Architecture and Stimulus Generation The core engineering challenge of this module was ensuring millisecond-level accuracy. The system avoids software-based delay loops (which are imprecise) in favor of hardware timers.

- **State 1: The "Ready" Phase:** When a round begins, the display renders "READY..." and the system initiates a hardware timer. To prevent the user from predicting the stimulus, the firmware generates a **pseudo-random wait interval**. This is achieved by reading the current value of the free-running system tick counter, ensuring that the delay between the "Ready" prompt and the stimulus is non-deterministic and varies every round.
- **State 2: The Stimulus:** Once the random interval expires, the screen creates an immediate visual interrupt (clearing the buffer or flashing text). Simultaneously, the high-frequency measurement timer is started

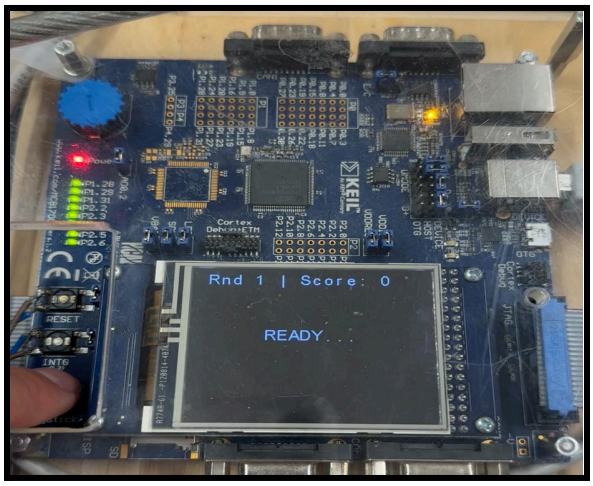

Figure 6b: *The pre-stimulus initialization phase. The system idles in the 'Ready' state while the random number generator calculates the wait interval.*

3. Interrupt-Driven Capture and Feedback To achieve high-precision measurement, the system leverages the Cortex-M3's Nested Vector Interrupt Controller (NVIC) to prioritize joystick inputs over background tasks. Upon user actuation, a GPIO interrupt is triggered, causing the processor to immediately vector to the Interrupt Service Routine (ISR). Inside this routine, the current value of the high-frequency hardware timer is captured to lock in the exact moment of response.

- **Latency Calculation:** The system computes the reaction time by determining the elapsed cycles between the stimulus generation timestamp and the input capture timestamp. This raw duration is then converted into milliseconds and compared against the scoring lookup table.
- **Real-Time Visualization:** The display buffer is updated instantaneously within the process loop to reflect the result.

- **Successful Capture:** If the calculated latency falls within a valid scoring tier, the screen renders the classification in green (e.g., "PERFECT! +5 PTS").
- **Input Fault / False Start:** If the interrupt is triggered prior to the visual stimulus (indicating a premature reaction) or originates from an incorrect directional pin, the error handler is invoked. The interface renders a fault message ("WRONG KEY" or "MISTAKE") in red, and the penalty logic is executed to deduct points from the session total

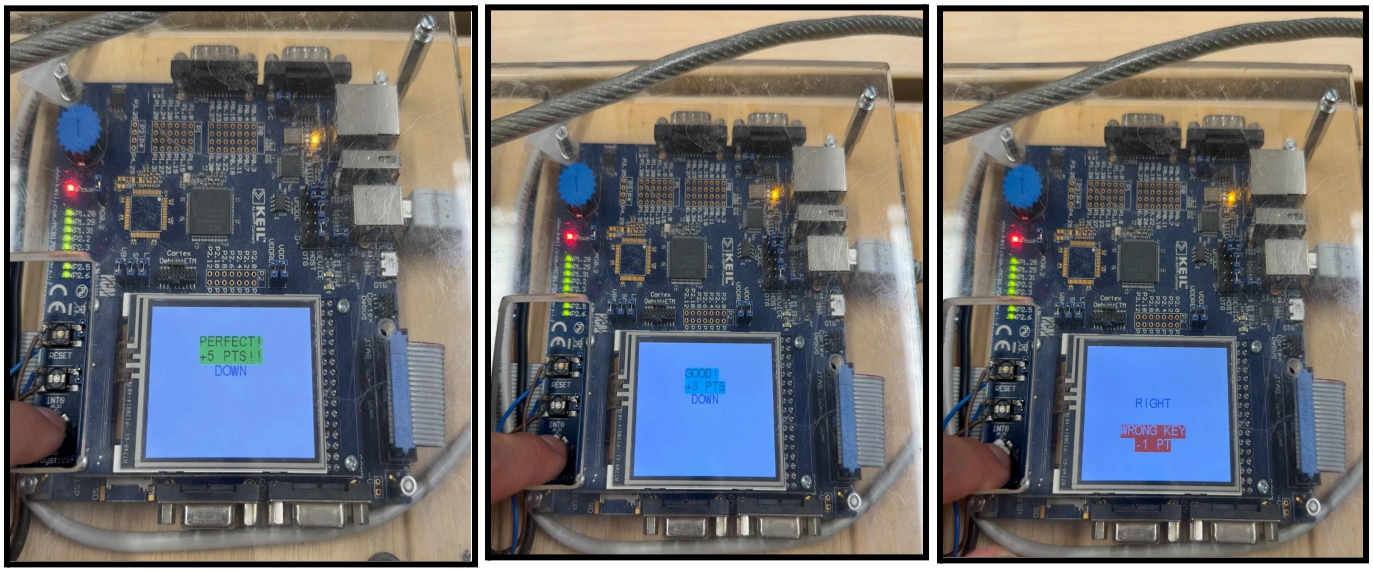

Figure 6c: *The real-time feedback loop. (Left) Elite reaction time triggering maximum points. (Center) Standard reaction time. (Right) Error handling routine triggering a score penalty.*

4. Session Aggregation and Final Reporting Upon the conclusion of the predefined measurement iterations, the firmware exits the active gameplay loop and transitions to the **Summary State**. This phase is critical for synthesizing the raw session data into actionable user feedback.

- **Data** Aggregation and **Classification Routine:** Throughout the active session, a persistent accumulator variable tracks the net score, accounting for both performance bonuses (e.g., +5 for Perfect) and penalty deductions (e.g., -1 for Errors). When the session terminates, the system passes this final integer value to a **classification algorithm**. This routine compares the aggregate score against the constant ranking thresholds defined in the initialization header.
    - This ensures that the qualitative grade assigned is deterministically mapped to the user's numeric performance.

- **Result Visualization and Validation:** The **Final Results** interface renders a comprehensive summary to the display. It presents the **Raw Numeric Score** to provide precise feedback, alongside the **Semantic Letter Grade** (Rank) to provide immediate context.
    - **High-Performance Validation:** As shown in the left figure, a score of **13** correctly triggers a **"RANK: B (GOOD)"** classification.
    - **Low-Performance Validation:** Conversely, the right figure demonstrates that a score of **3** accurately triggers a **"RANK: F (FAIL)"** classification.
    - This contrast confirms that the scoring logic effectively handles the entire spectrum of possible user outcomes.
- **Session Termination and State Reset:** To prevent the user from accidentally bypassing their results, the screen implements a **blocking input loop**. The interface requires an explicit physical acknowledgement ("Click to Exit") via the joystick. Once confirmed, the firmware executes a cleanup routine—flushing the session variables, zeroing the score accumulator, and resetting the random number generator seed—before returning the Finite State Machine to the Main Menu.

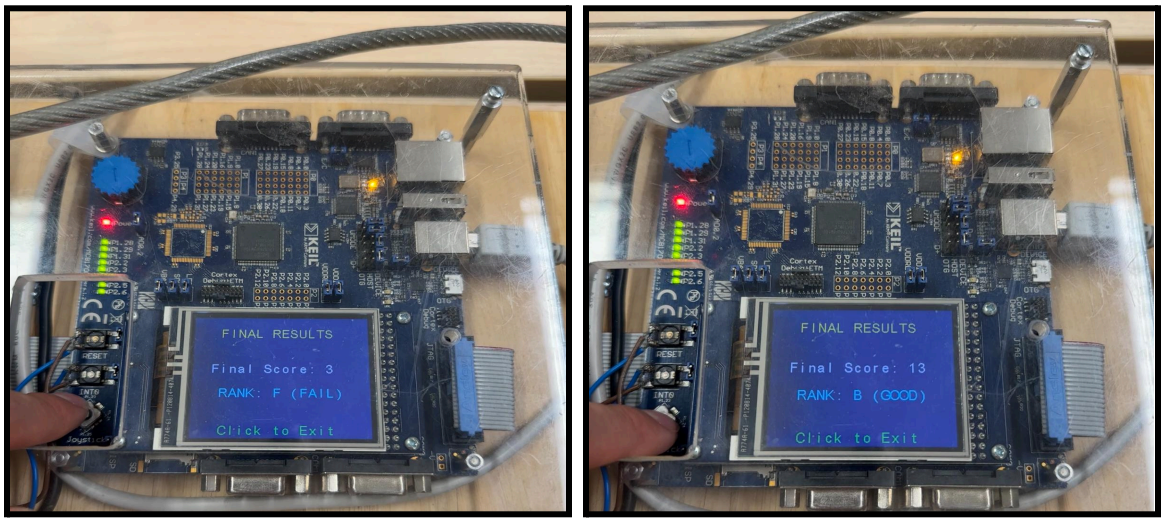

Figure 6d: Post-session analysis screens displaying the aggregated score and the final assigned rank based on performance

CONCLUSION

In conclusion, I successfully engineered a functional Embedded Media Center on the MCB1700 platform. I integrated four distinct subsystems—Audio Player, Photo Gallery, Tic-Tac-Toe game, and Reflex Tester game into a single multi-threaded application. The final system meets all design requirements, demonstrating that my software can effectively control hardware peripherals like the potentiometer and LCD in real-time. This project allowed me to apply theoretical concepts to a real-world implementation,

significantly strengthening my skills in hardware-software integration and concurrency. Ultimately, this experience simulated a professional engineering lifecycle, proving the importance of careful planning and architecture in embedded design.

References

[1] D. W. Lewis, *Fundamentals of Embedded Software with the ARM Cortex-M3*, 2nd ed. Upper Saddle River, NJ: Pearson, 2013.

[2] Element14 Community, "Embest EM-LPC1700-66 Evaluation Board." [Online]. Available: https://community.element14.com/products/devtools/technicallibrary/w/documents/10147/embest-em-lpc1700-66-evaluation-board

[3] G. N. Khan, "COE718 - Final Project: Media Center," Dept. Elect. and Comput. Eng., Toronto Metropolitan Univ., 2025. [Online]. Available: https://www.ecb.torontomu.ca/~courses/coe718/labs/Media-Center.pdf

[4] G. N. Khan, "Embedded System Case Studies," class notes for *COE718: Embedded Systems Design*, Dept. Elect. and Comput. Eng., Toronto Metropolitan Univ., Toronto, ON, 2025

[5] M. Wolf, *Computers as Components: Principles of Embedded Computing System Design*, 4th ed. Cambridge, MA: Morgan Kaufmann, 2016