Project Report

Project Title: 3x3x3 LED Cube with Bare Metal C

Team Members: Chris Immanuel I. Arcasa

Course: Microprocessor, Microcontroller Systems and Design (EEE184)

Instructor: Stephen Haim

Submission Date: ___________________________

1. Abstract (150–200 words)

This project presents the design and implementation of a 3×3×3 LED Cube using an AVR ATmega328P microcontroller. The objective was to develop a three-dimensional LED display capable of visually representing simple animations and lighting patterns through precise control of 27 individual LEDs. The system utilizes multiplexing and layered control to efficiently manage LED states while minimizing the number of required microcontroller output pins. The hardware design consists of a matrix of 3 layers and 9 columns, with each LED addressed by selectively enabling a layer and a corresponding column pin. The software implementation was developed in C, using direct register manipulation for optimal performance and minimal latency. Test routines successfully demonstrated stable, flicker-free animations such as layer scanning and column walking patterns. Key performance metrics include an update frequency of approximately 60 Hz per frame, ensuring visual smoothness. This project showcases the feasibility and efficiency of AVR-based LED cubes in creating interactive and educational embedded systems, with potential applications in visualization, artistic displays, and embedded systems education.

2. Introduction (300–400 words)

Background & Motivation

LED cubes offer an engaging and dynamic way to visualize three-dimensional patterns and animations, serving both as interactive artistic displays and as educational platforms to understand embedded system concepts. Traditionally, LED matrix projects are constrained to two dimensions; extending these designs into three dimensions introduces new challenges in circuit design, multiplexing, and real-time control. Our motivation for developing a 3×3×3 LED Cube stems from the desire to explore these concepts practically using AVR microcontroller technology. Specifically, this project leverages the ATmega328P microcontroller to demonstrate efficient hardware-software co-design in controlling 27 LEDs with minimal resources. This project also aims to provide an accessible demonstration of concepts such as GPIO control, time-division multiplexing, and power management, which are fundamental in embedded systems engineering.

Objectives

The project's primary objectives are:

- To design and construct a functional 3×3×3 LED Cube hardware prototype interfaced with an AVR ATmega328P microcontroller.
- To develop software capable of controlling individual LEDs and executing predefined animation patterns with minimal flicker and stable performance.
- To demonstrate effective multiplexing and layer-column addressing to reduce I/O pin usage while maintaining control over all 27 LEDs.

- To evaluate the performance and responsiveness of the LED Cube by testing animation smoothness, reliability, and visual quality.

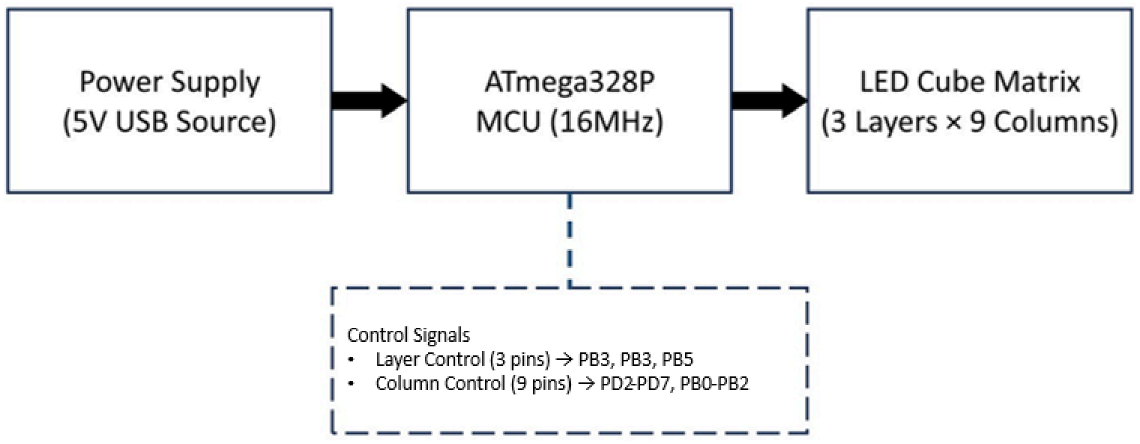
Scope & Limitations

The scope of this project focuses on the development of a basic but functional 3×3×3 LED Cube that demonstrates key embedded system control principles. The hardware is limited to 27 monochromatic LEDs (single color) arranged in a cubic structure without additional features such as color control (RGB LEDs), external sensors, or user interaction (buttons). The control logic is confined to simple predefined animations without real-time user programmability or wireless communication interfaces. Power is supplied via USB or regulated 5V supply, and the software is implemented in bare-metal C using AVR-GCC. Constraints include hardware resource limitations (number of available GPIO pins), timing accuracy for multiplexing without external timers, and manual soldering tolerances that may affect circuit integrity. Despite these constraints, the project successfully demonstrates the feasibility of a microcontroller-driven LED cube and provides a foundation for future enhancements.

3. System Architecture & Design

3.1 Block Diagram

The system architecture of the 3×3×3 LED Cube consists of three primary modules: the microcontroller (ATmega328P), the LED cube matrix (columns and layers), and the power supply. The microcontroller acts as the central control unit, managing LED activation through GPIO outputs using multiplexing techniques. The diagram below illustrates the data flow and interaction among the modules.



3.2 Hardware Components

Component	Details
Microcontroller	ATmega328P – Clock speed: 16 MHz – Flash Memory: 32 KB – SRAM: 2 KB – GPIO: 23 I/O pins – Key Features: 3 timers, SPI/I2C/UART, ADC
LED Cube Matrix	27 x 5mm LEDs (monochrome) arranged in a 3×3×3 cubic grid – Layers (3): Common cathodes – Columns (9): Common anodes
Power Supply	5V regulated USB power supply – Max current: ~300 mA (estimated max LED current with multiplexing)
Current Limiting Resistors	220 Ω resistors for each column line
Wiring & Structural Materials	– Solid core wire (for columns & layers) – Perfboard (for base circuit connections) – Soldering for permanent joints

Communication Interfaces

- No external communication (UART/SPI/I2C) is used in this project, as LED control is strictly local.
- AVR-ISP (In-System Programming) is used during software upload.

3.3 Software Architecture

Modules & Libraries

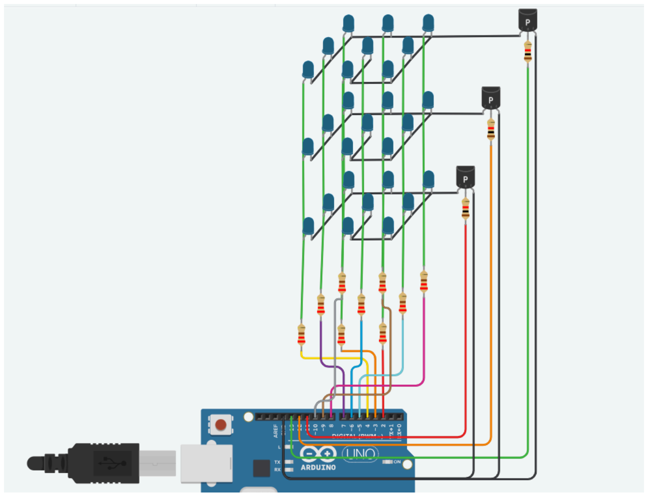
- GPIO Control Module: Responsible for setting PORTB and PORTD outputs to drive columns and layers.
- Animation Module: Contains pattern functions (e.g., layer scanning, cube blinking, column cycling).
- Timing Module: Uses _delay_ms() from <util/delay.h> for basic timing control (no external libraries used).
- No third-party libraries are integrated; only AVR-GCC built-in libraries.

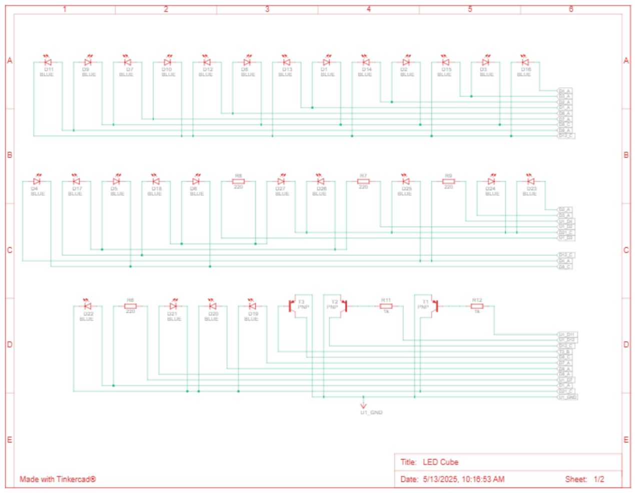
Control Flow

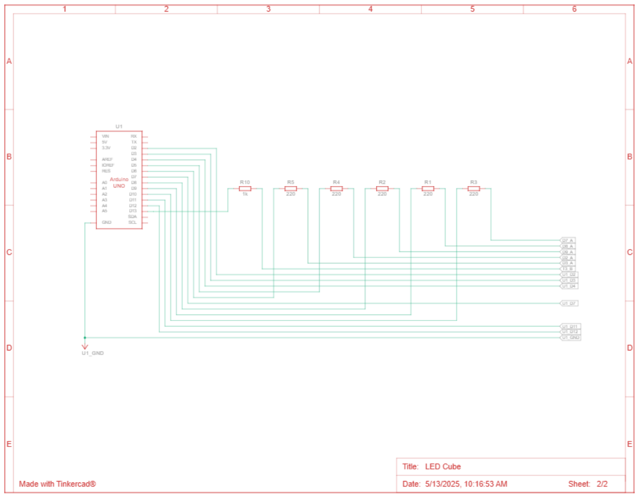
The software operates on a simple main program loop controlling animation sequences. Time-division multiplexing is achieved by sequentially activating layers and refreshing column states rapidly enough (~60 Hz) to leverage persistence of vision.

4. Implementation Details

Circuit Design & PCB Layout







Code Snippets

The LED cube logic was written in a hard coded manner bare-metal C using AVR-GCC, without external libraries. Key patterns and lighting effects are achieved by manipulating `PORTx` and `DDRx` registers directly.

Timing & Resource Management

The system relies on software multiplexing, cycling through layers and columns rapidly to exploit persistence of vision (POV) and simulate simultaneous illumination:

A delay of 2–5 ms per frame for each layer was used to balance brightness and flicker-free animation.

Performance Strategy: No interrupts or ISRs were used; timing is fully managed using _delay_ms() for simplicity. Delays were kept minimal (1–5 ms per LED group) to ensure a complete cube refresh occurs at ~60 Hz. The total frame refresh time across all layers is ~16–20 ms, well within real-time display requirements.

Memory Usage: The code avoids large data arrays to conserve SRAM. Repetition and patterns are hardcoded for clarity and deterministic timing, ideal for small-scale AVR projects.

Constraints Managed: No third-party libraries to avoid memory overhead. Minimal stack use since the project doesn't involve recursion or complex data structures.

5. Testing & Results

Test Plan

The testing phase was divided into two key areas: functional testing and performance evaluation.

- Functional Testing

  The objective of functional testing was to verify:

  - Correct mapping of microcontroller pins to the physical cube (i.e., each pin controls the intended LED column or layer).
  - Proper operation of multiplexing logic (i.e., only the intended LED lights up when column and layer are activated).
  - Accurate pattern execution (e.g., wave, rain, diagonal).

  Methodology:

  - A custom test routine was written to individually activate each LED in the cube by scanning all column–layer combinations.
  - Manual observation was used to verify that only one LED was on at a time and that there was no ghosting or unintended illumination.
  - A multimeter was used during early tests to confirm voltage behavior and check for floating voltages across layers or columns.

- Performance Testing

    The aim was to validate smooth animation rendering and persistence of vision under software-based multiplexing.

    Methodology:

    - Predefined animations (rain, wave, triangular pulse) were run continuously.
    - Delays between LED transitions were gradually decreased until flickering became noticeable, then increased to optimal values (typically ~2–3 ms per layer).
    - Stability was observed over extended runtime (>5 minutes) to detect overheating, flicker, or timing inconsistencies.

Test Results

- All 27 LEDs were successfully controlled through 9 column and 3 layer pins using direct register manipulation.
- Animations ran smoothly with no visible flickering when layer update delay was kept between 2–5 milliseconds.
- The column and layer control logic worked reliably, and persistence of vision effectively created the illusion of multiple LEDs being lit simultaneously.
- Functional objectives such as cube construction, basic and complex animations, and direct pin control using AVR bare-metal C were fully achieved.

Observed Deviations & Fixes

- Floating voltages were observed initially during manual LED testing. This was resolved by explicitly driving all unused layer and column pins to known states (HIGH or LOW), avoiding tri-stated pins.
- In early code versions, incorrect assumptions were made about the LED orientation (common anode vs. cathode). Once identified, pin logic was inverted to align with the cube's actual wiring (layers source current, columns sink current).

6. Lessons Learned & Future Work

Challenges

Several challenges were encountered during the development and testing of the 3×3×3 LED Cube, most of which centered around hardware control and low-level programming.

- Ghosting and Floating Voltages
    - During manual LED testing, several unintended LEDs appeared to glow faintly. These were traced back to floating (uncontrolled) pin states on unselected layers or columns. The solution involved explicitly clearing or

driving all unused pins to known voltage levels (HIGH or LOW) in the code to eliminate unintended conduction paths.

- Timing and Multiplexing
    - Achieving stable and flicker-free animations using only software-based multiplexing required careful tuning of delay intervals. Without hardware timers or interrupts, timing had to be manually balanced to ensure smooth visual transitions while avoiding excessive CPU blocking or LED flicker.

Improvements

While the project was successful in meeting its objectives, several improvements and extensions could significantly enhance the system in future iterations.

- Interrupt-Driven Multiplexing
    - Using hardware timers and interrupts (e.g., Timer1 Compare Match Interrupt) could offload multiplexing from the main loop, allowing smoother animations and freeing up the CPU to manage higher-level logic or user interaction.
- User Input Integration
    - Adding buttons or a serial interface would enable real-time control over the cube's animation modes, brightness, or pattern speed. This would make the system interactive and more educational.
- RGB LED Cube
    - Upgrading to RGB LEDs would allow multicolor animations and more visually rich displays. This would require PWM control and more complex current regulation, as well as significant expansion of pin usage or external drivers.
- External Drivers / Shift Registers
    - To expand beyond 3×3×3 or to reduce the number of direct I/O pins used, incorporating components like 74HC595 shift registers or LED driver ICs (e.g., MAX7219) would allow for higher-density LED cubes with less strain on the microcontroller.

7. Conclusion (100–150 words)

This project successfully demonstrated the design, implementation, and testing of a 3×3×3 LED Cube driven by an AVR ATmega328P microcontroller. Through the use of multiplexing and precise control over general-purpose I/O pins, I was able to animate 27 LEDs in a three-dimensional space using only 12 microcontroller output lines. The system was developed entirely in bare-metal C, emphasizing efficient hardware-software interaction without reliance on third-party libraries or abstraction layers.

The LED cube achieved all of its functional objectives, including accurate LED addressing, visually smooth animations, and low-level timing control. Key challenges—such as LED orientation, ghosting due to floating pins, and real-time multiplexing—were successfully addressed through iterative debugging, hardware probing, and code refinement.

Beyond being a visually engaging system, the project served as a practical exercise in embedded systems fundamentals, including GPIO control, timing management, and hardware troubleshooting. It highlights the potential of AVR-based platforms for low-cost, educational 3D visual systems and lays a foundation for future development, including user interaction, RGB lighting, and advanced control schemes.

In conclusion, the 3×3×3 LED Cube not only fulfilled its immediate technical goals but also demonstrated the broader capabilities of microcontroller-based embedded design in creative, interactive applications.

8. References

ES Tech Knowledge. 2020, January 18. *3x3x3 LED cube.* https://www.youtube.com/watch?v=dsDrm--20T8

Tinkering With Terrius. 2017, December 1. *Simple to build 3x3x3 LED Cube! | TWT #31*. https://www.youtube.com/watch?v=6OmwGV1dGK4

PicoGirl. 2013, April 23. *DIY PicoMiniCube 3x3x3 LED Cube electronic soldering kit for $25 tutorial included*. https://www.youtube.com/watch?v=IXf4AuczJp0

OpenAI. (2024). *ChatGPT (Version GPT-4.5)* [Large language model]. https://chat.openai.com/

DeepSeek. (2024). *DeepSeek Chat* [Large language model]. https://chat.deepseek.com/

Microchip Technology. (n.d.). *ATmega328P datasheet*. Retrieved from https://ww1.microchip.com/downloads/en/DeviceDoc/Atmel-7810-Automotive-Microcontrollers-ATmega328P_Datasheet.pdf