Tetris++

University of Basel - Computer Architecture

January 21, 2025

Group Members:
Diego Gonçalves Simao
Illia Solohub
Patrik Buetler

Abstract

This project focuses on the design and implementation of a Tetris game using an Arduino microcontroller, LED matrices, and additional hardware components. The primary goal was to recreate the classic game of Tetris in an embedded system, emphasizing resource management, hardware interfacing, and user interaction. The project employed modular programming techniques, dividing the logic into key components for block generation, game state management, and display rendering. Initial challenges included optimizing the LED matrix communication for smooth gameplay and managing the limited memory of the Arduino. These were addressed through iterative refinement of code and integration strategies. The final implementation successfully delivers a functional and visually engaging Tetris game, achieving the project's initial objectives. The game runs smoothly, with responsive controls and accurate representation of Tetris mechanics. This report details the methodology, challenges encountered, and solutions implemented, providing a replicable framework for similar embedded system projects.

1 Introduction

Tetris is a classic puzzle game that has captivated players worldwide since its creation in 1984. This project aims to recreate the game in an embedded systems context, using an Arduino microcontroller and LED matrices as the primary hardware components. The goal is not only to implement the core mechanics of Tetris but also to explore challenges unique to embedded programming, such as memory limitations, real-time processing, and hardware interfacing.

The motivation for this project stems from its multidisciplinary nature, combining software development, hardware design, and problem-solving skills. By implementing Tetris on a resource-constrained system, this project serves as an educational exercise to deepen understanding of embedded systems and their practical applications.

This report outlines the development process, starting from the project's objectives and moving through the design, implementation, and testing phases. It also highlights the challenges faced during development and the solutions devised to overcome them. The resulting system demonstrates how complex game mechanics can be implemented effectively on minimal hardware, offering valuable insights for future projects in the field of embedded systems.

2 Methodology

2.1 First Attempts and Redefinitions

The first Attempt to implement the main logic ended up to be fail for us, because it was a straight brute force with hundreds of for loops which was completely unreadable, this approach was discarded completely cause when we needed to reverse the direction of the tetraminos it required digging deep and changing a lot. So we decided that we need something more flexible, especially if we want to add special blocks and different control options. So we went with the solution described lower.

2.2 Approach and Solutions

As we are using a matrix made of 4 8x8 led displays which are supposed to be accessed individually and updated on the binary level, we figured out that it is a pretty uncomfortable, and a way to access the leds on the screen as pixel coordinates was made. Having that, and a tetramino(tetris block) as a separate struct sped up the development significantly and added a layer of flexibility, by resembling OOP. Also it made the entire thing less static

2.3 Software Enhancements

Improving the basic vanilla experience of Tetris is a corner stone of this project. A normal play through consists mainly of a player's ability to stack the blocks as tightly as possible while the generated blocks allow for such placements. Placement errors might be recovered from over time, line by line. The expansion to new block types came naturally as we discussed potential avenues for improvements. The idea behind new block types is the increased variability from run to run. A well placed explosion might clean out a misplaced block and gets the player back on track without stressing over some created gaps that might not be recoverable any time soon. We introduced two block types, one of which explodes several lines on impact, while the other breaks apart to fill in gaps below them. These block types have a 10 and 20 percent chance to be generated respectively. Normal blocks therefor appear 70% of the time. These probabilities might allow for long or infinite sessions. Optimized values could be determined with more play testing. Since we do not have visual distinctions between the block types due to the limitations of the screen, what you get is somewhat of a surprise.

2.3.1 Explosion Blocks

There were two main options we considered. Explosions could be local with a certain radius from the center or, as is implemented in different games of this type, explode the entire row horizontally. To avoid this block type being a detriment in your run we decided to clear entire rows where this block lands, as local explosions could create circular holes which are hard to fill using most of the available block forms.

2.3.2 Brittle Blocks

The goal of brittle blocks was clear from the start, instead of creating holes within a structure they should instead fill open gaps below. This idea seems to be of the kind that one can imagine looking awesome. In the end, due to lack of a visual distinction and having no animation for it, it ends up being some pixels that moved downwards for some reason. Positive implications on a players run hold true, we imagined it to look better than it is implement in practice though.

2.3.3 Challenges with Software Enhancements

We first build a functional base structure of Tetris, before working on these additions. This approach introduced a challenge of how we can best utilize the existing structures to introduce this new behavior. Any addition should be made by an informed programmer, so the first step included figuring out our hardware limitations and how we implemented all basic game functions as well as translating the game state to our screens. The screen can be manipulated by either rendering a predefined Tetris block anywhere within our screen's matrix representation or by setting single pixels to zero or one. Keeping this fact in mind, the calculations for an explosion around the impacting block could be realized by setting all surrounding pixels individually to zero. Instead we reused the logic of an existing function in the base game. Filling up an entire row triggers the game logic to clear the row and drop everything above downwards, increasing the player's score in the process. Now an explosion is simply filling the appropriate rows, after which the game triggers its subroutine to clear rows, simulating our explosion. Brittle blocks have mainly been a challenge in calculating gap sizes below a brittle block and make the right calculations. One early iteration on this problem included blocks falling upwards or several pixels landing on the same coordinate to name a few.

3 Results

The "Tetris++" project successfully created a working version of the classic Tetris game on Arduino. By adding a gyroscope, players can control and rotate the tetrominoes by tilting the device, making the game more interactive.

We moved away from the initial complex brute-force code, which was hard to read and difficult to modify, to a simpler structure using pixel coordinates and tetromino struct. This change made the code easier to understand and maintain. Additionally, switching to a pixel-based LED matrix system allowed for smooth and clear visual updates across the 4x8x8 LED displays.

These improvements met our main objectives and set the stage for future upgrades, such as adding special blocks and exploring other control methods.

4 Division of Labor

The planned initial division of labor has changed over time. The following table includes our first draft.

Person	**Responsible for**
Everyone	Shopping/Planning of Hardware
Everyone	Assembly of Hardware
Illia	Gyroscope controls
Diego	Base game
Diego	Accessories
Patrik	Software Enhancements
Patrik	Dark Mode

Table 1: Initial work assignment

Over time the responsibilities changed and we were interested in other parts of our project. Subsequent meetings caused a restructuring as this second table shows.

Person	Responsible for
Everyone	Shopping/Planning of Hardware
Everyone	Testing and Debuggind
Diego	Assembly of Hardware
Diego	Accessories
Patrik	Gyroscope controls
Illia	Base game
Patrik	Software Enhancements
Patrik	Dark Mode (dropped)
Diego	Demo

Table 2: Final work assignment

5 Critical Self-Assessment

In-person meetings have been very helpful, productive and improved the flow for our project. Attempts were made to split the labor in a way that reduced interactions between responsibilities. This philosophy increased autonomy, but introduced larger gaps in gaining a wide range of experiences that usually come from and are a goal of projects like this.
We finished the code separately from the hardware assembly, assuming there to be enough time for the hardware assembly to go smoothly and the code to run without complications. Fortunately for us there have not been any major complications.

6 Conclusions

The **"Tetris++"** project successfully developed a functional version of the classic Tetris game using Arduino hardware. By integrating a gyroscope, we introduced an attention-taking control mechanism that allows players to maneuver and rotate tetrominoes through device tilting, enhancing the overall gameplay experience. Transitioning from a complex brute-force coding approach to a more organized structure using pixel coordinates and tetromino structs significantly improved code readability and maintainability. Additionally, adopting a pixel-based LED matrix system ensured smooth and clear visual updates across the 4x8x8 LED displays.

Throughout the project, valuable lessons were learned in data representation, hardware-software integration, and optimization within hardware constraints. The challenges faced, such as managing multiple LED displays and refining the control logic, provided deeper insights into embedded systems programming and reverse engineering processes.

Moving forward, the foundation laid by **"Tetris++"** opens up opportunities for further enhancements, including the addition of special blocks, exploring alternative control methods, and expanding the game's features to increase its complexity and replay value. This project not only met our primary objectives but also equipped us with the skills and knowledge to tackle more advanced projects in embedded programming and interactive hardware design.

7 References

7.1 Code base

The entire code is available on Github at `https://github.com/PatrikBuetler/tetris-`

7.2 Hardware

We used various hardware and devices here is a list:

- Arduino Mega MEGA
- 4 LED matrices MAX7219
- 5 Push buttons for user input
- Power supply 9V
- Jumper wires and connectors
- Resistors and other circuit components 10k OHM
- 2x Breadboard for prototyping
- Gyroscope MP6050

7.3 Costs

We bought the ELEGOO 'The Most Complete Starter Kit' for the arduino MEGA, it had most of the stuff that we needed. The only hardware that we needed to buy was the gyroscope and the four matrices. In total we spent:

- ELEGOO Starter Kit 67 CHF
- Gyroscope MP6050 13 CHF
- 4x Matrix MAX7219 19 CHF
- Total: 100 CHF

8 References

Inspiration: `https://www.youtube.com/watch?v=XuL_Dmm5V4Y&ab_channel=Devopedia`
`https://projecthub.arduino.cc/RomanSixty/lx-arduino-tetris-708606`
Starter kit:
`https://www.elegoo.com/blogs/arduino-projects/elegoo-mega-2560-the-most-complete-starter-kit-tutorial?srsltid=AfmBOopupDCUx4k3GlIwDG9ThsCZb3olse-rFXBHS9r1umHbmG4WhPqe`
MP6050:
`https://www.reichelt.com/ch/de/shop/produkt/entwicklerboards_-_beschleunigung_gyroskop_mit_header_mpu-60-266105?country=ch&CCTYPE=private&LANGUAGE=de`
MAX7219:
`https://www.3dsvet.eu/izdelek/modul-max7219-32x8-led-rdeca/`
Libraries: The libraries we use are listed on the github repository: `https://github.com/PatrikBuetler/tetris-/blob/main/utils/libraries.txt`

Disclaimer

The source code in this project was developed by the project team as part of a university course. While parts of the logic and structure were inspired by the classic Tetris game, all programming and implementation were independently written by the authors. Any external libraries or components used are credited in the references section and the utils section. ChatGPT was used solely for improving the wording and structure of documentation and not for generating code.