Color Jump Project

April 11, 2025
Virginia Tech
ECE 2564: Embedded Systems
Myriam Bajani

I. Report Summary

This report explains how I built my Color Jump game on the MSP432. It includes what the game does, how I set up the system, how my code works, and extra bonus features I added like the Mixed mode with changing speed and points. I also included pictures of the game's different stages.

II. Project Description

The goal of my game is to keep the player alive by making sure their color matches the floor they're running on. The game starts with a Title Screen (Figure 1) that shows the game name and my name.

Figure 1: Title Screen

After three seconds, the user is taken to the Main Menu Screen (Figure 2), where they can navigate between options using the joystick.

Figure 2: Main Menu Screen

In the Menu Screen, players can choose to play the game, read the instructions (Figure 3), go to the options (Figure 4), or see the high scores (Figure 5). In the Options Screen, the player can switch between Easy, Hard, or Mixed difficulty modes using BB2.

Figure 3: Instructions Screen

Figure 4: Options Screen

Figure 5: High Scores Screen

In the actual Game Screen (Figure 6), the player runs on a floor made of random colors, with their score shown at the top. Orange square-shaped barriers also appear as obstacles that the player must avoid. Moreover, there's a color wheel where the middle color is the player's current color, and the joystick is used to switch to the surrounding colors.

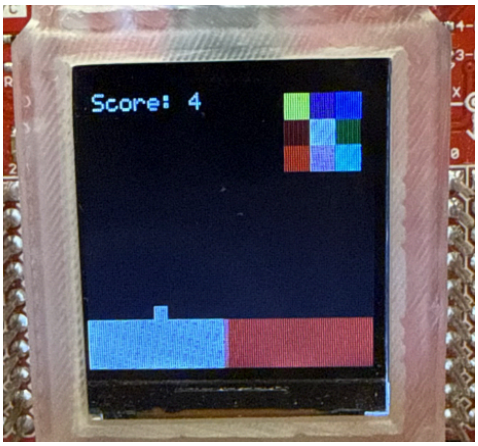

Figure 6: Game Screen

If the player lands on a block with the wrong color or hits one of the barriers, the game ends. The Game Over Screen (Figure 7) then shows the final score and gives the player the option to restart or return to the main menu.

Figure 7: Game Over Screen

My implementation satisfies all required features outlined in the project specification.

III. Microcontroller-Based Embedded System Architecture

The system architecture for the color game includes the MSP432, LaunchPad, and LCD BoosterPack, with communication handled through SPI:

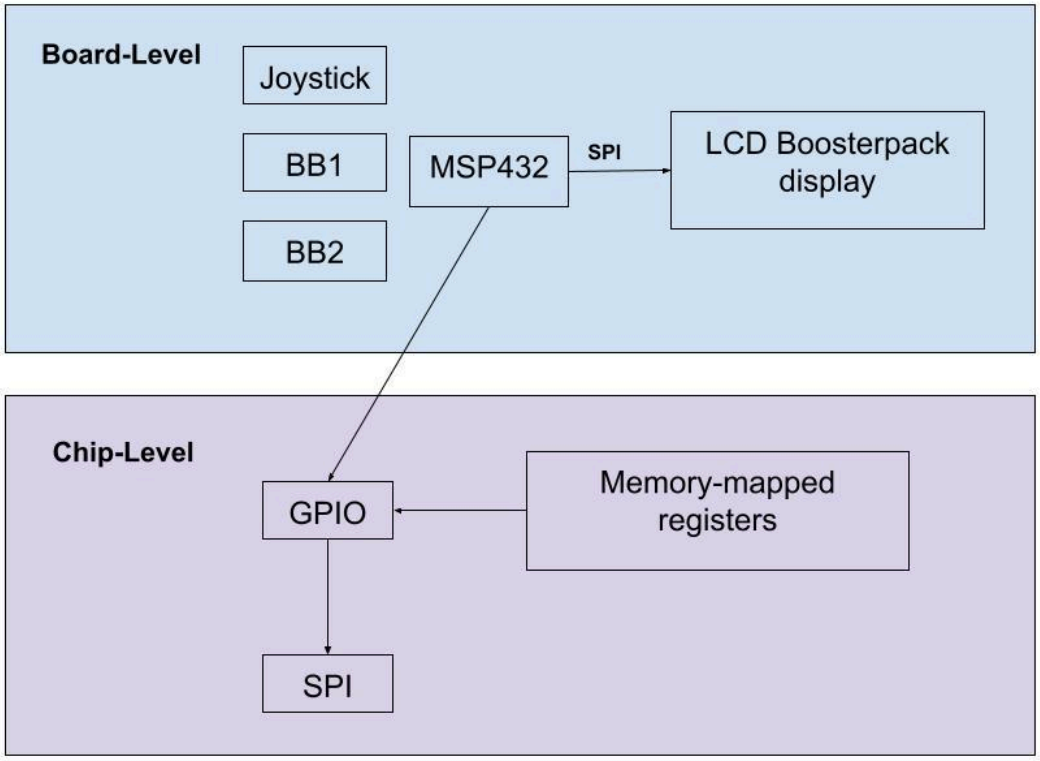

Figure 8: MSP432 Board-Level vs. Chip-Level Views

At the board-level, the system includes the MSP432 microcontroller, the LCD BoosterPack display, two boosterpack buttons (BB1 and BB2), and a joystick for user input. The LCD screen connects to the MSP432 using the SPI interface, which is used to draw all the visuals in the game. The joystick and boosterpack buttons are connected through the GPIO pins and are used for game controls, like moving the cursor, switching colors, and jumping. These inputs help the player interact with the game in real time.

At the chip-level, the MSP432 relies on memory-mapped registers to control the connected parts. GPIO handles the joystick and button inputs, while SPI is used to send data to the screen. The processor accesses these devices by reading from or writing to specific memory locations. This setup allows the game to update quickly and respond smoothly to the player's actions.

IV. Code Quality

For this project, I focused on making my code clean, efficient, and easy to understand.

First, I made sure to comment my code regularly as I worked through it. Since this was a long project, I wanted to keep track of my thought process.

I also avoided using hard-coded numbers in my code. Instead, I defined constants like #define

FLOOR_BLOCK_WIDTH 105, which helps make the code more flexible and easier to update if needed.

To keep my functions manageable, I ensured that no function exceeded 60 lines. I broke down my code into smaller functions for specific tasks, which were all clearly defined in proj2_app.h and used in the application loop to keep things organized.

Additionally, I made sure to avoid using global variables, even though I briefly used them for testing purposes. As I worked through my code, I continuously edited it to remove any temporary global variables. Eventually, I stored variables like the speed, score, and player coordinates within the application construct, and accessed them using app_p-> instead of relying on global variables.

Finally, I made sure my code was non-blocking by avoiding while loops. I tested this by ensuring that pressing LB1 would always light up LL1, no matter what stage of the program was running.

V. Bonus Features Achieved

I added several bonus features to my game to make it more fun and challenging.

First, instead of just 4 colors on the color wheel, I added 8 unique colors. The player can now swap colors using both the joystick's regular directions as well as diagonal movements.
Next, I added random orange square-shaped barriers on the floor that the player has to jump over. If the player hits a barrier, the game ends.

Additionally, I added a new difficulty mode that I called Mixed. In this mode, the floor starts scrolling at speed = 1.0, and every 5 seconds, it gets faster by 0.5 until it reaches a max of 6.0. I also made the game give more points per second as the speed increases, and barriers only start appearing once speed hits 2. The table below shows how the difficulty and points change over time:

Table 1: Mixed Mode Difficulty Table

Time Elapsed (s)	Speed	Points per second	Description
0-5	1.0	1	Starting speed
5-10	1.5	1	
10-15	2.0	1	Easy Mode speed

15-20	2.5	2	
20-25	3.0	2	
25-30	3.5	3	Hard Mode speed
30-35	4.0	3	
35-40	4.5	4	Extremely fast speed
40-45	5.0	4	
45-50	5.5	4	
50+	6.0	4	Max speed

Lastly, I designed and added my own custom images for the Title Screen and Game Over Screen, as shown in Figures 1 and 7.