ECE3810 Microprocessor System Design Laboratory

Laboratory Report 5

Name: Xu Zigeng

Student ID: 123090704

Date: 2025/12/15

The Chinese University of Hong Kong, Shenzhen

1 Objectives

In this project, we aimed to design and build a 2-player bouncing ball game based on the knowledge learned through ECE3810 Labs 1-5. Specifically, the objectives were:

- To implement a complete game system integrating GPIO, USART, LCD, External Interrupt, and Timer.
- To realize the basic bouncing ball game mechanics as specified in the project handout.
- To propose and implement improvements to make the game more challenging and engaging.

2 Basics and Setup

The system is built on the STM32F103 microprocessor. The hardware setup involves:

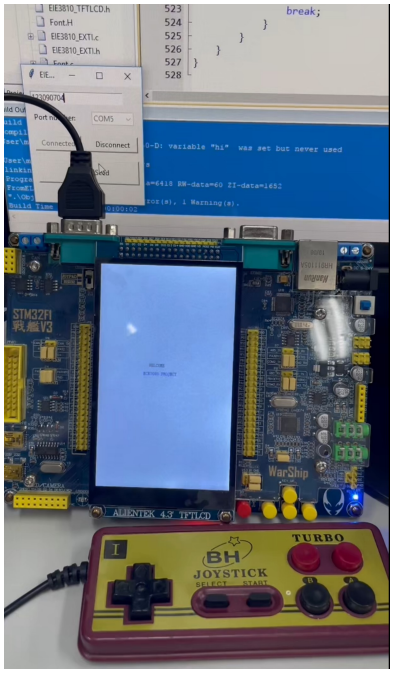
- **STM32 Board**: The core controller managing game logic and peripherals.
- **TFT LCD**: Displays the game interface, including the ball, paddles, obstacles, and status text.
- **Joypad**: Connected via COM3, used by Player B to control the top paddle.
- **On-board Keys**: Key0, Key1, Key2 used by Player A for control and system settings.
- **USART**: Used for communication with the PC to receive a random seed for the ball's initial direction.

Figure 1: Hardware setup showing the STM32 board, LCD, and Joypad connection.

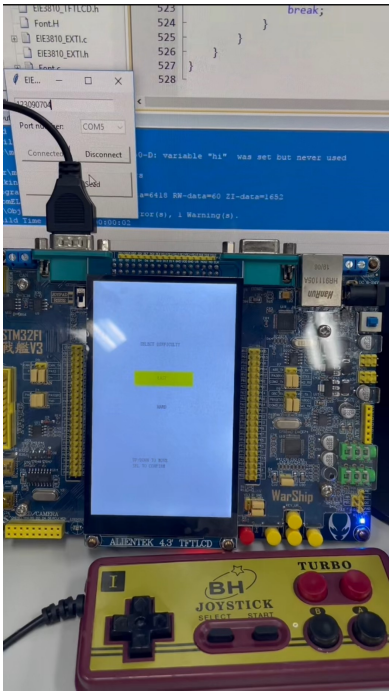
3 Experiment 1: Realization of Bouncing Ball Game

3.1 Initialization and Difficulty Selection

Upon startup, the system initializes all peripherals. The LCD displays the welcome screen. Player A uses Key_Up/Key1 or Player B uses the Joypad to select the difficulty level (Easy/Hard).



(a) Welcome screen with startup melody.

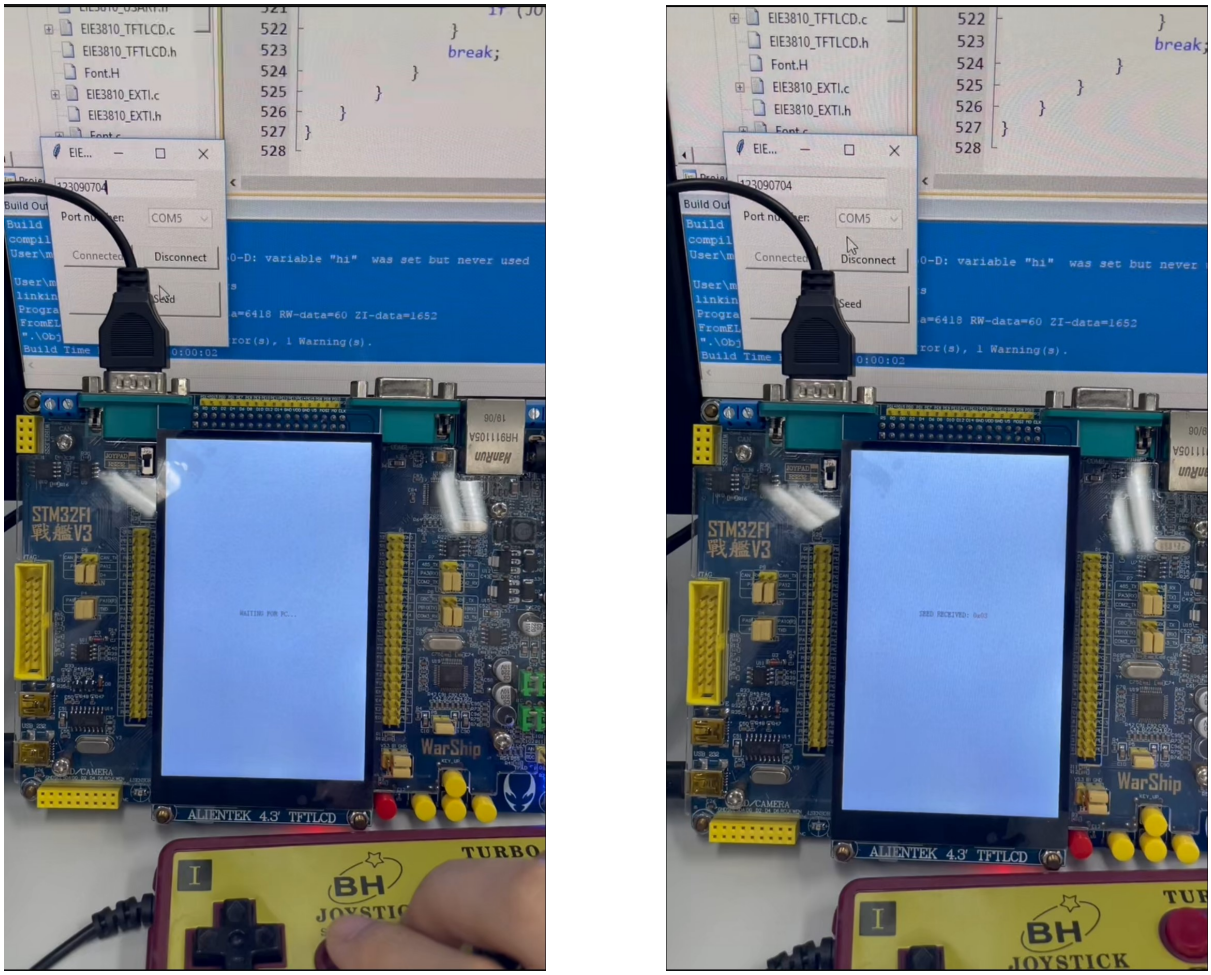


(b) Difficulty selection interface (Easy/Hard).

Figure 2: Initial startup screen and difficulty selection.

3.2 Random Direction via USART

After difficulty selection, the game waits for a random seed. The Python script `ECE3080_PC.exe` sends a random number (0-7) via USART. The system receives this seed to determine the ball's initial launch direction.



(a) LCD showing waiting status.

(b) Showing the random seed.

Figure 3: USART communication process for random seed generation.

3.3 Countdown and Game Start

Once the seed is received, a 3-second countdown is displayed. The ball then launches in the direction corresponding to the received random number.

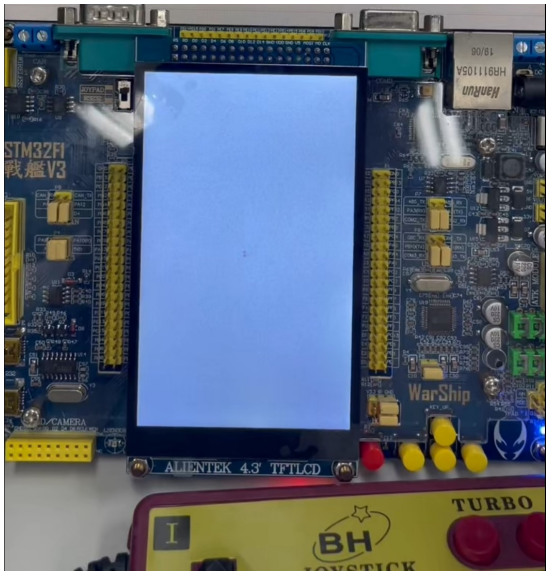


Figure 4: 3-second countdown before the game starts.

3.4 Gameplay and Physics

During the game:

- **Player A** controls the bottom paddle using Key2 (Left) and Key0 (Right).
- **Player B** controls the top paddle using the Joypad (Left/Right).
- The ball bounces off the side walls and paddles.
- A buzzer sounds upon collision.
- The elapsed time and bounce count are updated in real-time on the LCD.

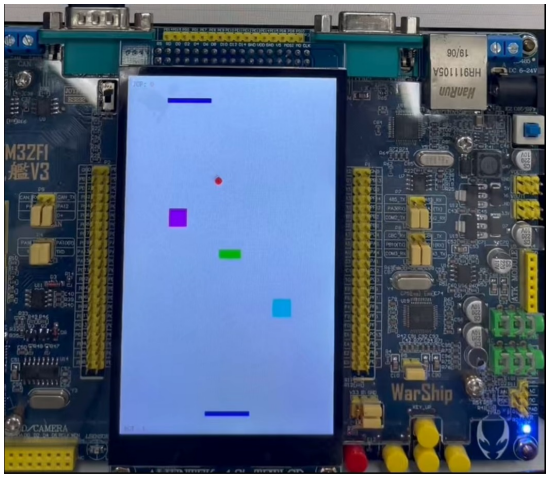


Figure 5: Active gameplay showing the ball, paddles, and status information.

3.5 Game Over and Reset

If a player fails to catch the ball, the game ends. The winner is announced, and the final stats are shown. After a few seconds, the game returns to the start screen.

Figure 6: Game Over screen displaying the result and winner.

4 Experiment 2: Improve the Bouncing Ball Game

4.1 Proposed Improvements

To enhance the gameplay depth and user experience, I introduced several major improvements beyond the basic requirements:

1. **Static Obstacles**: Added obstacles in the arena to create unpredictable bounce trajectories.
2. **Dynamic Acceleration**: The ball's speed increases after every paddle hit, making the game progressively harder (rally intensity).
3. **Best-of-Three Match System**: Implemented a scoring system where the first player to win 2 rounds wins the match, adding a competitive tournament feel.
4. **Audio Feedback**: Added a welcome melody at startup and distinct sound effects for collisions (paddles vs. obstacles).

4.2 Realization

4.2.1 Static Obstacles

Three obstacles were defined to disrupt simple reflection angles:

- Center (Green): A rectangular block in the middle.
- Top Left (Magenta) & Bottom Right (Cyan): Square blocks near the player zones.

```
typedef struct {
    s16 x, y, w, h;
    u16 color;
} Obstacle;
// ... obstacles initialized in array ...
```

Listing 1: Obstacle Definition

4.2.2 Dynamic Acceleration

To prevent infinite rallies, the ball's vertical velocity (`vy`) is incremented by a small factor upon every paddle collision, capped at a maximum limit.

```
// Inside collision detection
if (abs(ball.vy) < MAX_SPEED) {
    if (ball.vy > 0) ball.vy += ACCEL_STEP;
    else ball.vy -= ACCEL_STEP;
}
```

Listing 2: Velocity Acceleration Logic

4.2.3 Best-of-Three Logic

Global counters track the number of rounds won by each player. The game loops until one player reaches 2 wins.

```
if (winner == PLAYER_A) wins_a++;
else wins_b++;

if (wins_a >= 2 || wins_b >= 2) {
    current_state = MATCH_OVER; // End of match
} else {
    Reset_Round(); // Start next round
}
```

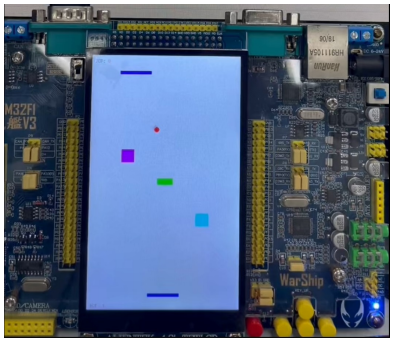
Listing 3: Match Winning Condition

4.2.4 Audio Effects

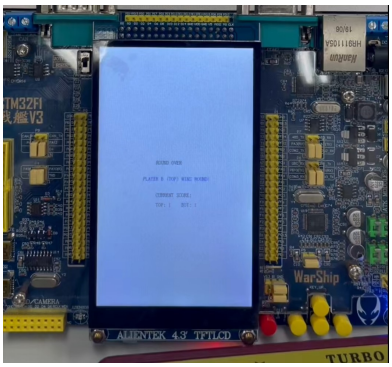
A buzzer module is used to generate sound. A sequence of tones plays at the `WELCOME` state, and short beeps are triggered during collisions.

4.3 Results

The improvements significantly increased the game's engagement. The **acceleration** mechanic ensures rounds conclude eventually, while the **obstacles** force players to stay alert. The **sound effects** provide immediate feedback, and the **Best-of-Three** rule makes victory feel more earned.



(a) Game with static obstacles (Green, Magenta, Cyan).



(b) Best-of-Three match scoring system.

Figure 7: Improved game features: obstacles and match scoring.

5 Conclusion

This project successfully integrated various STM32 peripherals to create a functional two-player game. Experiment 1 demonstrated mastery of basic embedded system concepts (GPIO, Interrupts, Timers, USART, LCD). Experiment 2 further explored game logic and collision physics by introducing obstacles, resulting in a more engaging and challenging experience.