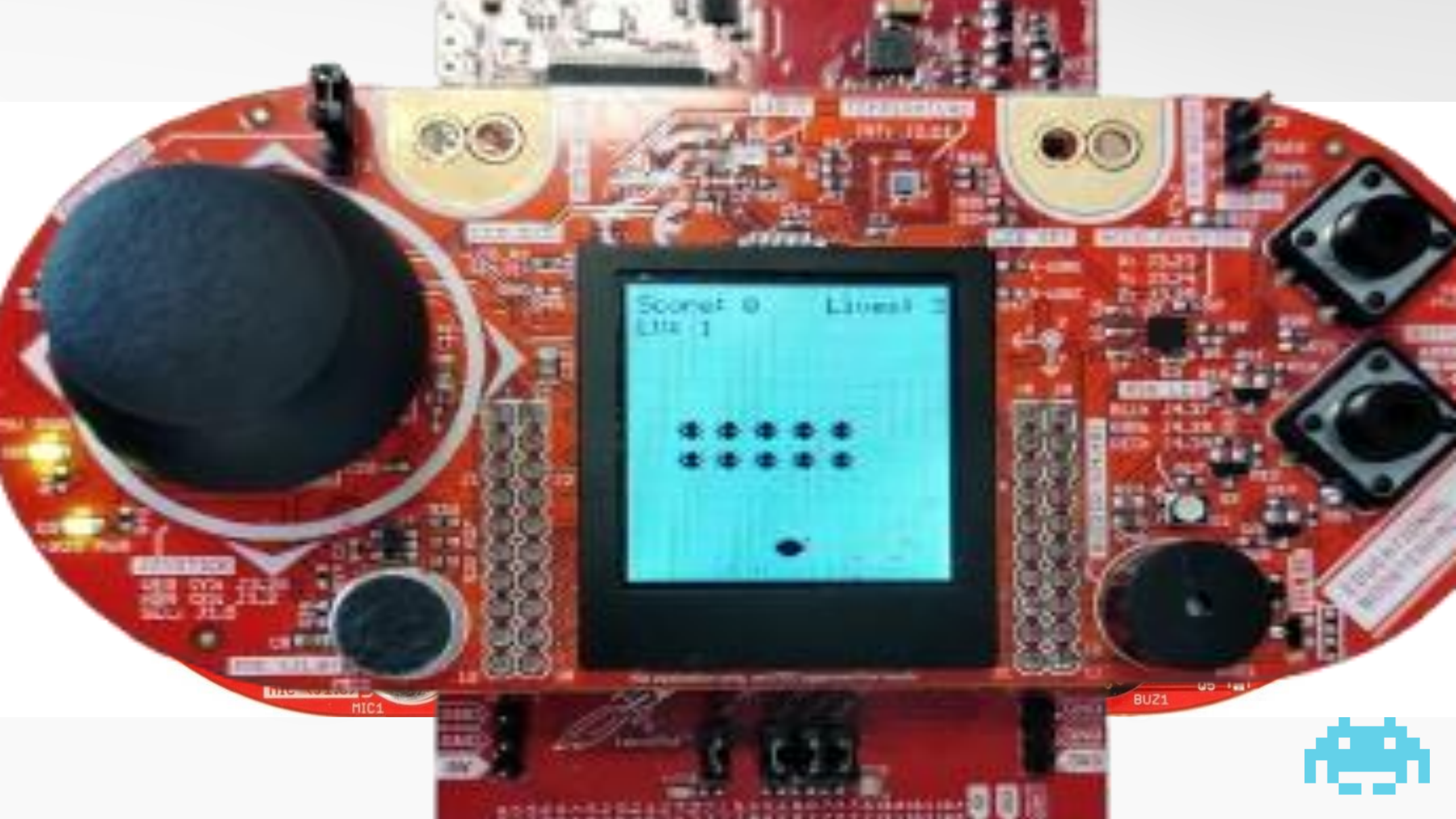

Score: 0
Level: 3
Life: 1
MIC1
BUZ1

SPACE INVADER


SPACE
INVADER

WORKING FLOW

DATA STRUCTURES

IMPORTANT CODE

TESTING

MEMBER ROLES

FUTURE WORK

WORKING FLOW

This project is a Retro arcade-style Space Invader

- Input handling (ADC,GPIO interrupts)
  - Joystick ADC14, Port1 button
- Game logic
  - Player, enemies, projectiles, collision detection, score system, 3 levels
- Graphics
  - LCD display

Interrupts Used:Timer Interrupt (TA0_0_IRQHandler): Updates game logic at regular intervals.
GPIO Interrupt (PORT1_IRQHandler): Handles button presses for shooting and menu navigation.
ADC Interrupt (ADC14_IRQHandler): Reads joystick position to move the player.

CORE SOFTWARE

Core Software Blocks of the System

- **Main Program (main.c)**
  - Initializes hardware and game elements.
  - Manages game states (menu, playing, game over).
  - Calls update and draw functions for player, enemies, and projectiles.
- **Player Module (player.c / player.h)**
  - Controls player movement and positioning.
- **Enemy Module (enemy.c / enemy.h)**
  - Manages enemy formation, movement, and attacks.
- **Projectile Module (projectile.c / projectile.h)**
  - Handles player and enemy projectiles.
  - Detects collisions with enemies and the player.
- **Graphics Module (graphics.c / graphics.h)**
  - Draws game elements (player, enemies, projectiles, screen etc).
  - Updates the screen with score, lives, and explosions.

REPRESENTATIVE CODE

Main

```
int main(void) {
    hardware_init();
    srand(time(NULL));
    menu_init();
    while (1) {
        MAP_PCM_gotoLPM0();
        if (game) {
         enemy_manager_update();
         enemy_manager_draw();
         projectile_manager_update();
         projectile_manager_draw();
         if (enemy_manager_has_reached_player()) {
              lives--;
              Graphics_clearDisplay(&g_sContext);
              game_init();
         }
         if (lives <= 0) {
              graphics_lose(&player, player.x, player.y);
              game = 0;
              menu = 1;
         }
         projectile_manager_check_collisions(&player);
         graphics_update_screen(score, lives, level);
         __delay_cycles(5000000);
        }
    }
```

Draw Enemy

```
void enemy_manager_draw(void) {
    uint8_t row;
    uint8_t col;
    for (row = 0; row < num_rows; row++) {
        for (col = 0; col < invaders_per_row; col++) {
         if (enemies[row][col].active) {
              // Erase the old position
              Graphics_setForegroundColor(&g_sContext,
GRAPHICS_COLOR_WHITE);

graphics_draw_single_invader(enemies[row][col].prev_x,
enemies[row][col].prev_y, level);
              // Draw the enemy at the new position
              Graphics_setForegroundColor(&g_sContext,
GRAPHICS_COLOR_BLACK);
              graphics_draw_single_invader(enemies[row][col].x,
enemies[row][col].y, level);
         } else {
              // Clear the enemy from its old position if inactive
              Graphics_setForegroundColor(&g_sContext,
GRAPHICS_COLOR_WHITE);

graphics_draw_single_invader(enemies[row][col].prev_x,
enemies[row][col].prev_y, level);
         }
        }
    }
}
```

TESTING

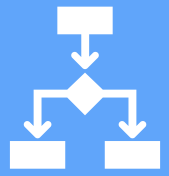
How did we test

- Played a lot of times
- Tested with different enemies, probability of shooting, lives, speed
- Added print statements to check correctness.

Problems Encountered

- Collisions
- Projectiles bug
- Graphic glitches
- Variables overflow

MEMBER ROLES



Alessio De Col

- Collisions, Inputs (Buttons, Joystick), Game Logics, Enemy Logic

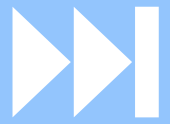
Antonio Amabile

- Draft C Code, Game Logic, Player Logic, Enemy Logic

Marko Markovic

- Collisions, Graphics, Game Logic, Projectile Logics, Enemy Logic

FUTURE WORK



Conclusions and Future Work

- New enemy types with different behaviors
- Adding sound effects
- Fixing additional instability