Space Invaders - 8 Bit Video Game

EE 396 : Design Lab Report

Tarun Kurethiya

220102100

under the guidance of

Dr. Ribhu Chopra

to the

DEPARTMENT OF ELECTRONICS AND ELECTRICAL ENGINEERING
INDIAN INSTITUTE OF TECHNOLOGY GUWAHATI
GUWAHATI - 781039, ASSAM

Abstract

This report details the development of a classic arcade-style game, *Space Invaders*, implemented on an Arduino Uno with a TFT display. The project explores real-time game logic, efficient rendering techniques, user input handling, and system-level optimization for resource-constrained environments. The implementation addresses challenges such as memory management, refresh rate optimization, and precise collision detection within the constraints of embedded hardware. The result is a fully functional interactive game that emulates the nostalgic gameplay experience of the original 1978 release while demonstrating practical applications of embedded systems concepts, including microcontroller programming, interrupt handling, and hardware interfacing.

Contents

1 Introduction **3**
1.1 Project Objectives . . . 3

2 Background and Game Overview **3**
2.1 About the Game . . . 3
2.2 Core Game Elements . . . 4

3 Materials Required and Detail Overview **5**
3.1 Hardware Components . . . 5
3.2 Microcontroller Specifications . . . 6
3.3 Display Selection . . . 7

4 Theoretical Background **7**
4.1 Game Mechanics . . . 7
4.2 SPI Communication . . . 8
4.3 Joystick Interface . . . 8

5 Design and Game Architecture **8**
5.1 System Architecture . . . 8
5.2 Hardware Connection Diagram . . . 9
5.3 Game States . . . 9
5.4 Game Object Classes . . . 10

6 Implementation Details **10**
6.1 Software Libraries . . . 10
6.2 Core Game Loop . . . 11
6.3 Optimized Sprite System . . . 11
6.4 Collision Detection Mechanism . . . 11

7 Physical Assembly **12**
7.1 Enclosure Design . . . 12
7.2 Final Assembly . . . 12

8 Challenges Faced **12**
8.1 Hardware Limitations 12
8.2 Gameplay Testing 12

9 Conclusion **13**

10 Future Work **13**

11 References **13**

1 Introduction

This report presents the design and development of a playable version of the classic arcade game *Space Invaders* using an Arduino Uno and a TFT display. The project encompasses the complete development workflow, including hardware configuration, software implementation using state machines, graphical rendering on the TFT screen, and user interaction through a joystick module.

1.1 Project Objectives

The primary objectives of this project are:

- Design and implement a simplified yet authentic version of Space Invaders on Arduino
- Create responsive user controls using analog joystick input
- Optimize graphics rendering for performance on limited hardware
- Demonstrate understanding of game development principles including collision detection, state management, and user interaction
- Apply embedded systems concepts in a practical, engaging application

2 Background and Game Overview

2.1 About the Game

Space Invaders, released in 1978 by Taito and designed by Tomohiro Nishikado, was one of the first fixed shooter arcade games and became a cultural phenomenon that revolutionized the video game industry. The gameplay involves a player-controlled laser cannon defending against descending waves of aliens. The objective is to eliminate the invaders before they reach the bottom of the screen.

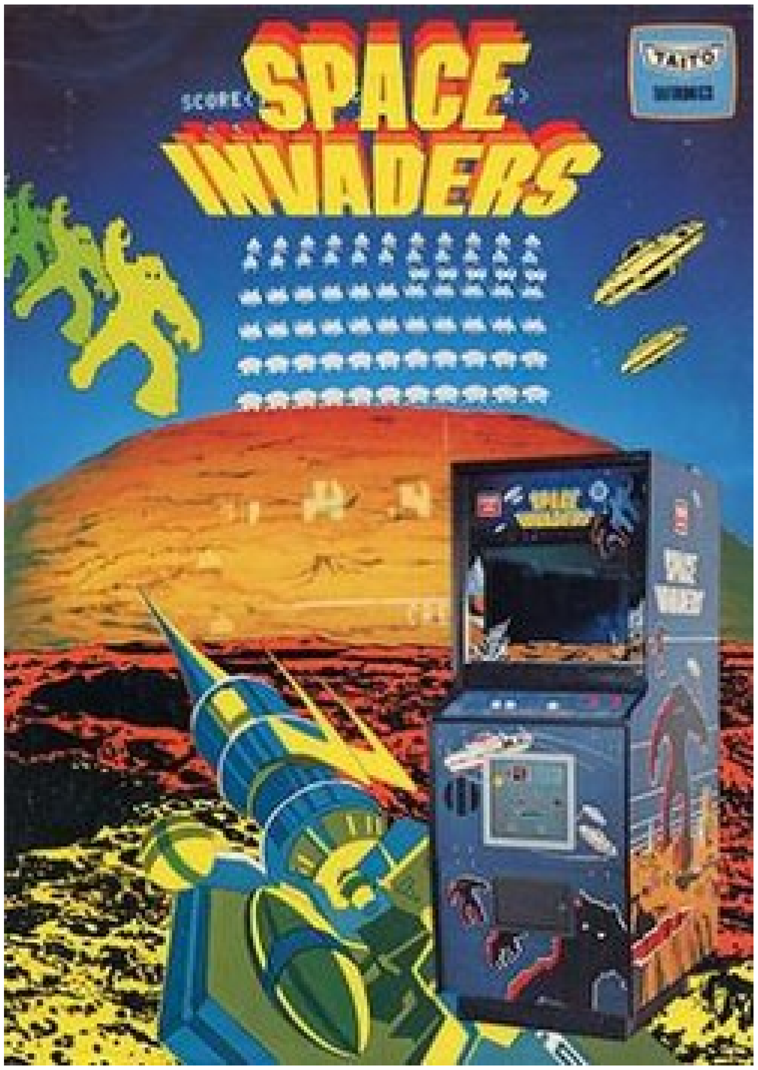

Figure 1: Original Space Invaders arcade cabinet from 1978

2.2 Core Game Elements

The original game featured several distinctive elements that this implementation aims to recreate:

- Player-controlled defensive cannon at the bottom of the screen
- Multiple rows of alien invaders moving horizontally and descending
- Protective barriers that gradually deteriorate from both player and alien fire
- Increasing difficulty as aliens move faster when fewer remain
- Distinctive sound effects that increase in tempo as gameplay progresses

3 Materials Required and Detail Overview

3.1 Hardware Components

- Arduino Uno microcontroller board
- 2.8T̈FT Display shield
- Analog joystick module with push button
- Power supply (5V 2A adapter)
- Jumper wires and breadboard
- MDF board, screws, and glue for housing
- Soldering tools and USB cable
- piezo buzzer for sound effects

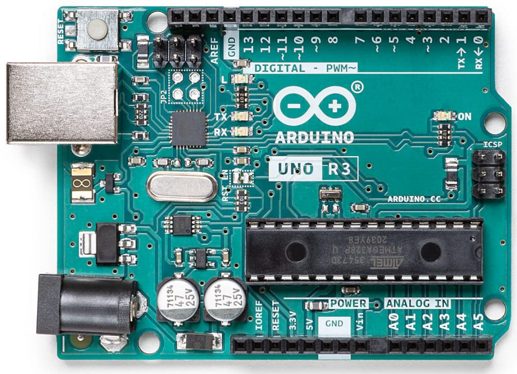

Figure 2: Arduino Uno microcontroller board

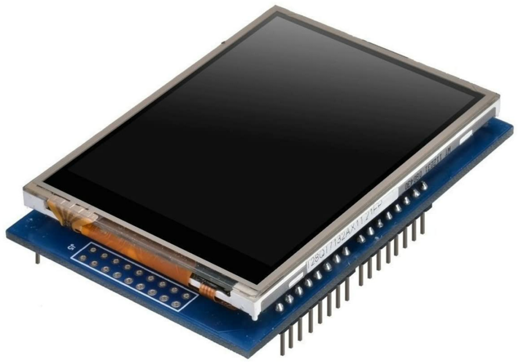

Figure 3: 2.8" TFT Display with ILI9341 controller

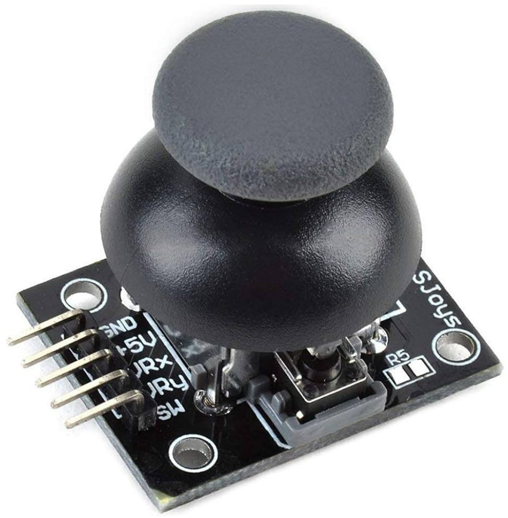

Figure 4: Analog joystick module with push button

3.2 Microcontroller Specifications

The Arduino Uno, based on the ATmega328P microcontroller, features:

- 32 KB flash memory (0.5 KB used by bootloader)
- 2 KB SRAM
- 1 KB EEPROM
- 16 MHz clock speed

- 14 digital I/O pins (6 with PWM capability)
- 6 analog input pins
- USB connection for programming and power

This microcontroller is chosen due to its simplicity, reliability, and extensive community support, making it ideal for educational projects and rapid prototyping. While not the most powerful option available, the resource constraints of the Arduino Uno create an interesting challenge for game development, requiring careful optimization of code and graphics rendering.

3.3 Display Selection

The 2.8" TFT display with ILI9341 controller was selected for this project because it offers:

- Hardware SPI interface for faster data transfer
- Compatibility with the Adafruit GFX and ILI9341 libraries
- Shield form factor that directly connects to Arduino Uno

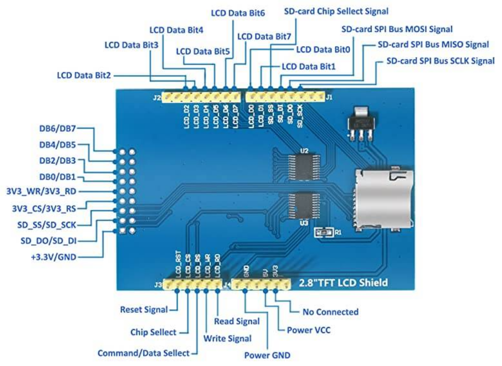

Figure 5: TFT shield pinout diagram showing connections to Arduino

4 Theoretical Background

4.1 Game Mechanics

The game mechanics of Space Invaders follow specific patterns:

- **Invader Movement Pattern**: The invaders move in a zigzag horizontal pattern, descending one step after hitting the screen edge. This creates an increasing sense of urgency as they get closer to the player.
- **Projectile Physics**: The player can fire projectiles to eliminate the enemies. Only one player projectile can be active at a time, requiring strategic timing.

- **Collision Detection**: Simple bounding box collision detection is implemented to determine hits between projectiles and game objects.
- **Game Over Conditions**: If an invader reaches the player level or all player lives are depleted, the game ends. Victory is achieved by eliminating all invaders.

4.2 SPI Communication

The TFT display communicates with the Arduino using Serial Peripheral Interface (SPI):

- A synchronous serial data protocol used for short-distance communication
- Master-slave architecture where Arduino acts as master and TFT as slave
- Four signal lines: MOSI (Master Out Slave In), MISO (Master In Slave Out), SCK (Serial Clock), and SS (Slave Select)
- Data transfer rates up to several MHz, depending on the hardware

4.3 Joystick Interface

A 2-axis analog joystick provides user input through:

- X and Y potentiometers that return analog values (0-1023) representing position
- A digital push button for firing actions
- Analog-to-digital conversion on Arduino maps these values to game control inputs
- Center position calibration to account for manufacturing variations

5 Design and Game Architecture

5.1 System Architecture

The complete system architecture consists of:

- **Input Layer**: Joystick module providing directional control and button actions
- **Processing Layer**: Arduino Uno running game logic, collision detection, and state management
- **Display Layer**: TFT screen showing game elements and user interface
- **Audio Layer**: Piezo buzzer for sound effects

5.2 Hardware Connection Diagram

The hardware components are connected as follows:

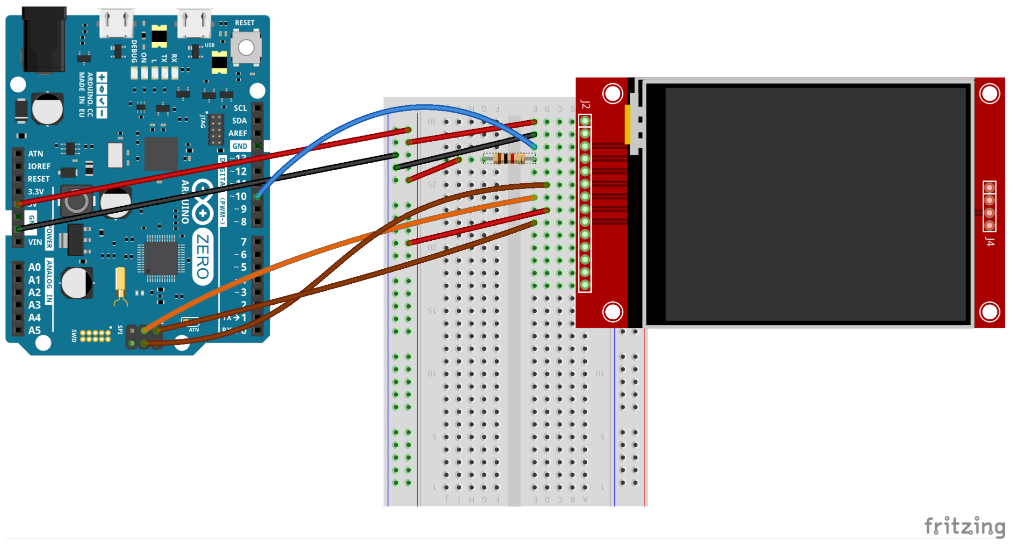

Figure 6: circuit diagram showing connection to TFT Display

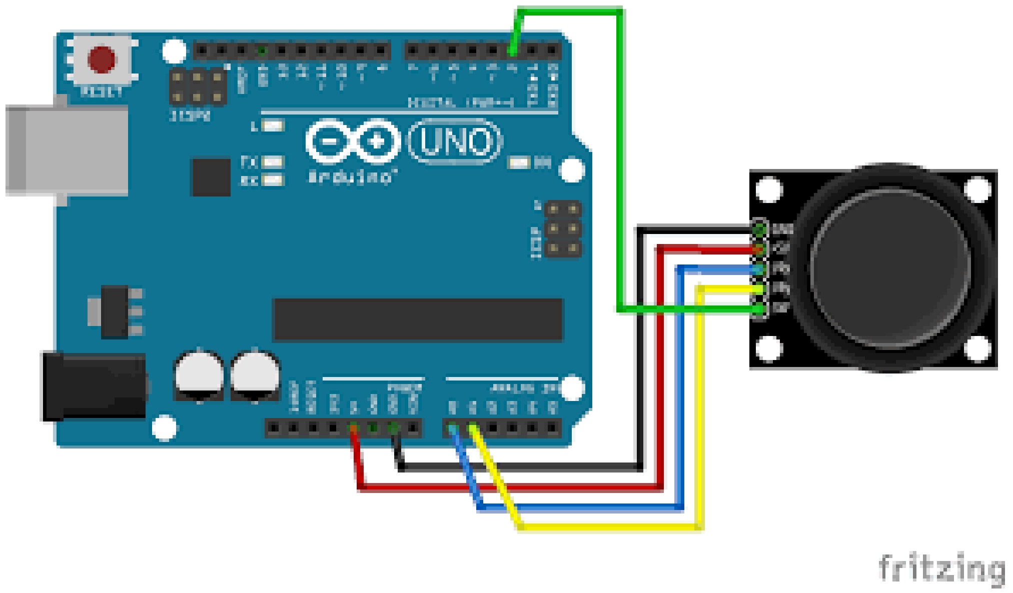

Figure 7: circuit diagram showing connection to joystick

5.3 Game States

The game is structured using a finite state machine approach with the following states:

- **Title Screen** – Displays welcome message, game title, and waits for user input to start.

- **Game Start** – Initializes game assets, enemy positions, player position, and resets score.
- **In-Game** – Main game loop handling logic, rendering, and input processing.
- **Pause** – Optional state that freezes game activity until resumed.
- **Game Over** – Displayed when all enemies are eliminated (victory) or player loses.
- **High Score** – Optional state for displaying and updating high scores.

5.4 Game Object Classes

The implementation uses object-oriented design with the following classes:

- **Player**: Represents the player's ship at the bottom of the screen
  - Properties: position, width, height, speed, lives
  - Methods: move(), fire(), render(), reset()
- **Enemy**: Represents an alien invader
  - Properties: position, type, width, height, alive status
  - Methods: move(), render(), isHit()
- **Bullet**: Represents projectiles fired by player or enemies
  - Properties: position, direction, active status
  - Methods: move(), render(), checkCollision()

6 Implementation Details

6.1 Software Libraries

The implementation relies on several key libraries:

- **Adafruit_GFX**: Provides core graphics primitives (shapes, text)
- **Adafruit_ILI9341**: Display driver specific to the ILI9341 controller
- **SPI**: Handles the SPI communication protocol
- **EEPROM**: Optional library for storing high scores permanently

6.2 Core Game Loop

The core game loop serves as the central framework driving the entire gameplay, ensuring smooth operation and user responsiveness. Within this loop, several tasks continuously occur:

- **Input Processing**: The joystick input is regularly checked to determine player actions, like moving the player's spaceship or firing bullets.
- **Game State Management**: The loop maintains and updates different stages or "states" of the game, such as the title screen, active gameplay, and the game-over screen. It employs a switch-case structure to clearly and efficiently transition between these states. During active gameplay, the logic updates (such as enemy movement and bullet positions) occur at precise intervals to maintain consistent game speed and ensure a fair playing experience.
- **Rendering Control**: The game's visuals are refreshed at regular intervals separate from game logic updates. This separation helps maintain stable and visually appealing animations without slowing down the game logic or causing visual lag.

In essence, this loop structure carefully balances input handling, game logic updates, and rendering tasks to create a seamless and responsive gaming experience on resource-constrained hardware like the Arduino Uno.

6.3 Optimized Sprite System

Due to the Arduino Uno's limited memory and processing power, a specialized sprite system was developed. Instead of rendering detailed graphics or large images, compact sprite representations were used. These sprites are stored efficiently in the microcontroller's program memory (flash memory), significantly reducing RAM usage.

To render these sprites, the system selectively draws only necessary pixels on the TFT display, skipping transparent or unused pixels to optimize rendering speed. This method allows the game to run smoothly, even on resource-constrained hardware.

6.4 Collision Detection Mechanism

Collision detection between various game elements (such as player bullets and enemy targets) utilizes a simplified but effective bounding-box algorithm. Each game object has defined rectangular dimensions (width and height). By comparing these rectangles' positions, the game determines whether objects intersect, triggering events such as destroying enemies or incrementing the player's score.

For instance, when the player fires a bullet, the system continuously checks whether this bullet overlaps with any active enemy sprites. Upon detecting a collision, appropriate actions are taken—such as removing the enemy, updating the score, and deactivating the bullet. This method ensures accurate, real-time collision handling while remaining computationally efficient.

7 Physical Assembly

7.1 Enclosure Design

A custom enclosure was designed to house the game components in an ergonomic form factor using MDF Board.

7.2 Final Assembly

The final assembly process involved:

1. Mounting the TFT display to the Arduino using header pins
2. Connecting the joystick module via jumper wires
3. Securing all components to the MDF base plate
4. Adding cosmetic details to enhance the arcade aesthetic
5. Installing the optional speaker for sound effects

8 Challenges Faced

During development, several technical challenges were encountered and addressed:

8.1 Hardware Limitations

Display Upgrade from LCD Initially, we used a 16x2 character LCD, but its limited resolution and lack of graphics support made it unsuitable for a game like Space Invaders. **Improvement:** Switching to a 2.8" TFT display gave us full control over pixel-based graphics, enabling smoother gameplay and a more immersive experience.

8.2 Gameplay Testing

The game was tested with various scenarios:

- **Responsiveness**: The player control and firing action were responsive to joystick inputs with minimal latency.
- **Collision Detection**: Bullet-enemy interactions were consistently accurate across frames.
- **Stability**: The game loop maintained consistent performance even after extended play sessions.

9 Conclusion

The final game runs smoothly and is fully playable, delivering a fun and responsive experience with no noticeable lag. It successfully meets all the goals we set at the start. This project has been incredibly rewarding, teaching us practical skills in embedded systems, game design, and hardware integration.

10 Future Work

- Add score tracking, save high scores, and introduce multiple difficulty levels.
- Enhance gameplay by allowing enemies to shoot, adding shields, and including a bonus mystery ship.
- Improve visual and audio experience with animated sprites and better sound effects.
- Include a game menu and support multiplayer with an extra joystick.
- Make the setup portable with battery power and clearer sound using a speaker and amplifier.

11 References

1. Arduino Official Documentation, "Arduino Uno Rev3," `https://docs.arduino.cc/hardware/uno-rev3/`
2. Adafruit Industries, "Adafruit GFX Graphics Library," `https://learn.adafruit.com/adafruit-gfx-graphics-library`
3. Adafruit Industries, "Adafruit ILI9341 Library," `https://github.com/adafruit/Adafruit_ILI9341`
4. Space Invaders History, "The Definitive Space Invaders," `https://www.spaceinvaders.de/`
5. Cook, M., "The Design of Space Invaders," in Game Design Theory Practice, Wordware Publishing, 2002.
6. Kent, S., "The Ultimate History of Video Games," Three Rivers Press, 2001.
7. Zaks, R., "Programming the 6502," Sybex, 1983. (Reference for understanding low-level programming concepts similar to those used in original arcade machines)
8. Williams, A., "History of Digital Games: Developments in Art, Design and Interaction," CRC Press, 2017.

9. Bodnar, J., "Arduino TFT Display and Font Library Tutorial," `http://www.circuitbasics.com/arduino-tft-display-and-font-library/`

10. Margolis, M., "Arduino Cookbook," O'Reilly Media, 2020.