UNIVERSITY OF CALIFORNIA, RIVERSIDE

BOURNS COLLEGE OF ENGINEERING

DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING

CS/EE 120B Custom Laboratory Project Report

The Dinosaur Game on AVR

JUSTIN FABABIER

June 13, 2024

Contents

1 Introduction

The project undertaken is a recreation of the popular "Chrome Dino" game, also known as the Dinosaur Game. This game is a simple, endless runner that appears in the Google Chrome web browser when there is no internet connection. The player controls a T-Rex dinosaur that runs continuously across a desert landscape. The primary objective is to avoid obstacles and survive as long as possible to achieve a high score.

The main purpose of this project is to independently design and implement a playable video game, specifically targeting an embedded device environment. "The Dinosaur Game on AVR" aims to recreate the beloved browser game using the Atmega328p microcontroller. This approach provides a hands-on experience in working with embedded systems, allowing a deeper understanding of microcontroller capabilities and limitations.

2 Additional Notes

The following build-upons were implemented:

1. HiLetgo 1.44" SPI TFT LCD 128-by-128 pixel display
2. Support for a second-player in real-time
3. Passive buzzer for dynamic sound control

All of the listed build-upons were successfully implemented.

Few shortcomings were realized. For one, a score counter was originally going to be implemented but was left out due to time constraints. For two, there was intended to be two variation of cacti sprites; only one is included due to time constraints. For three, dual player mode experiences sluggish performance.

Despite minor issues, the game performs to specification. Improvements to the project will be made after the Spring '24 quarter for personal fulfillment.

3 User Guide

The Dinosaur Game on AVR is an engaging project that involves the following user interactions and system behaviors:

3.1 Controls

- **Joystick**: The primary control device for the game is a joystick. The player uses the joystick to interact with the game as follows:
- **Press to Start**: Pressing the joystick initiates the game from the idle state, causing the dinosaur to start running.
- **Return to Idle**: In the event of a game over, player 1 can click their joystick to reset the game to the idle state. Player 2 can also interact with their joystick to either join the game or exit, switching the game between single and two-player modes.

3.2 Visual Outputs

- **ST7735 Graphic LCD**: The game features a black-and-white display that visually represents the game environment. Key elements displayed include:
  - **Dinosaur**: Controlled by the player(s), the dinosaur is the main character that runs and jumps over obstacles.
  - **Obstacles**: Cacti and birds appear on the screen, and the player must navigate the dinosaur to avoid these obstacles to continue playing.

3.3 Sound Mechanism

- **Passive Buzzer**: The game includes a passive buzzer that provides an audio experience. It plays background music while the game is in progress, adding an auditory element to the gameplay. The buzzer is inactive when the game is in the idle state or has ended.

3.4 Gameplay States and Indicators

- **LED Indicators**: The system uses three LEDs to represent different game states:
  - **White LED**: Indicates the game is in the idle state, waiting for the player to start.
  - **Green LED**: Indicates the game is in progress, with the dinosaur running and avoiding obstacles.
  - **Red LED**: Indicates the game is over, due to a collision with an obstacle.

3.5 Gameplay Mechanics

- **Single and Two-Player Modes**:
  - In single-player mode, the player controls one dinosaur and navigates through the obstacles.
  - In two-player mode, each player controls their own dinosaur. If either player collides with an obstacle, the game ends for both. Player 1 can reset the game to idle, and player 2 can join or leave by interacting by clicking their joystick button.

3.6 Additional Features

- **Obstacle Avoidance**: The core gameplay involves the player(s) making the dinosaur jump over obstacles to avoid collisions. If a collision occurs, the game immediately ends.
- **Mode Switching**: The game seamlessly switches between single and two-player modes based on player 2's joystick interactions.

4 Hardware Components Used

1. HiLetgo 1.44" SPI TFT LCD 128-by-128 pixel display
2. Passive buzzer (x1)
3. Joystick (x2)
4. LED (x3)
    - White LED (x1)
    - Green LED (x1)
    - Red LED (x1)
5. Elegoo Uno R3 board (x1)
    - Atmega 328p microcontroller
6. Breadboard (x3)
7. Jumper wires
8. 220 ohm resistors (x3)

5 Software Libraries and Header Files Used

- **helper.h**:
  - Includes essential AVR libraries for input/output operations (avr/io.h), interrupt handling (avr/interrupt.h), and delays (util/delay.h). This library provides utility functions to support various aspects of the game, simplifying the implementation of repetitive tasks and common operations.
- **objects.h**:
  - Defines the game's objects, such as the dinosaur and obstacles, and their properties. This header file streamlined the creation and management of game entities, which facilitate structures and bitmaps.
- **periph.h**:
  - Manages peripheral devices connected to the AVR microcontroller. This includes setting up performing analog-to-digital conversion.
- **serialATmega.h**:
  - Enables serial communication with the ATmega microcontroller, useful for debugging and monitoring the game's state during development.
- **spiAVR.h**:
  - Facilitates SPI communication between the AVR microcontroller and the ST7735 graphic LCD, ensuring efficient data transfer for displaying game graphics.
- **ST7735.h**:
  - Provides functions to interface with the ST7735 graphic LCD.
- **timerISR.h**:
  - Handles timer interrupts, crucial for timing-based operations such as game updates, controlling the buzzer, and managing the LEDs.

6 Wiring Diagram

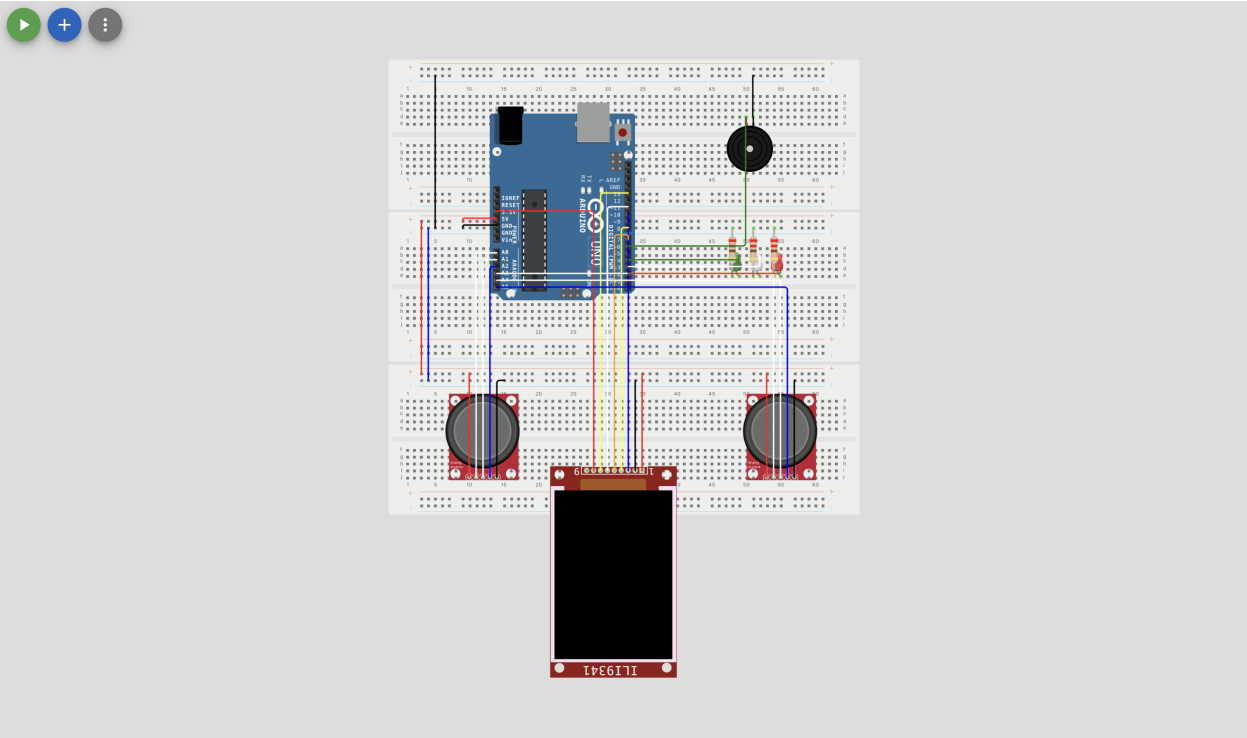


Figure 1: Wiring diagram for project *(Disclaimer: the LCD shown is different than the one used of the physical setup.)*

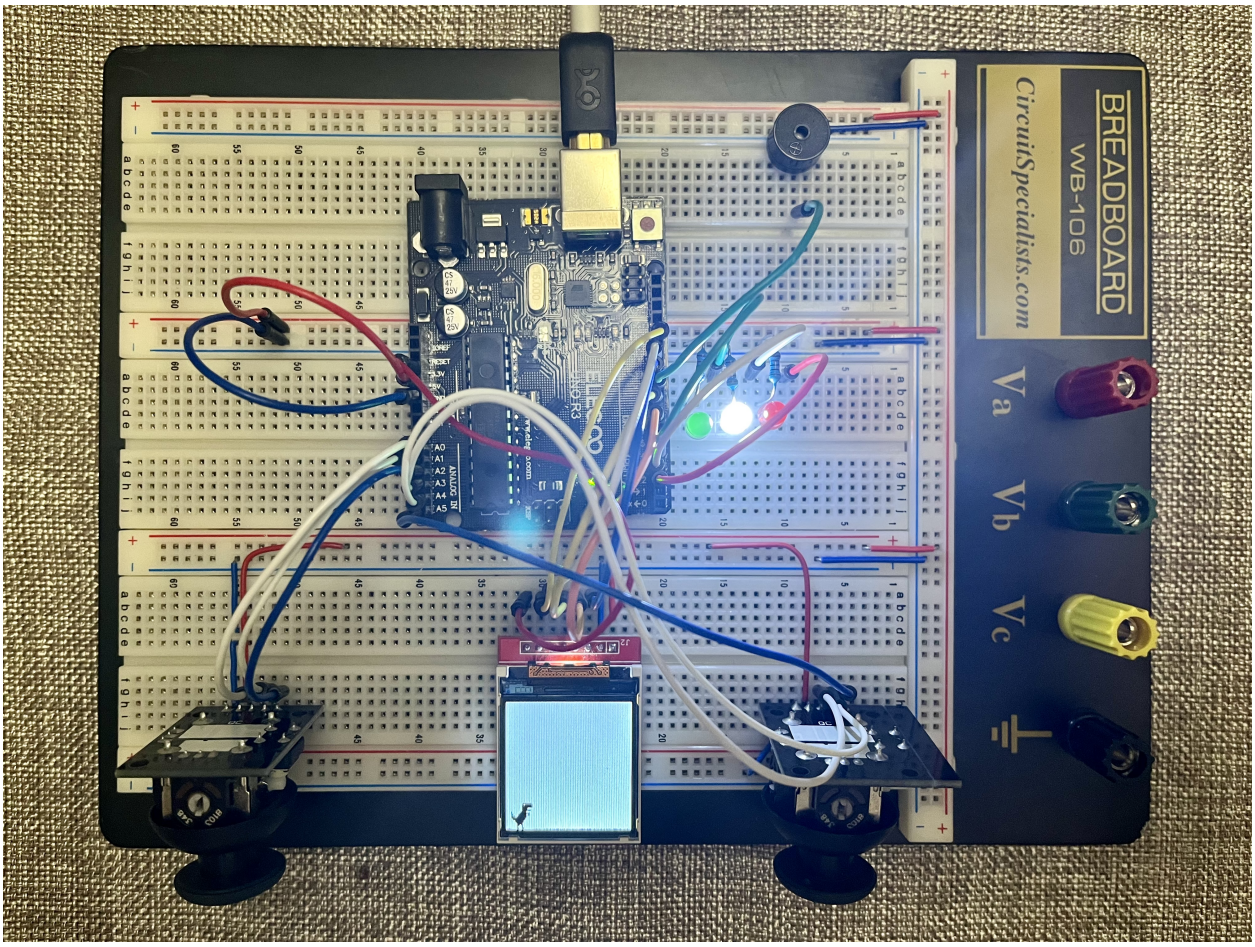


Figure 2: Physical wiring diagram.

7 Task Diagram

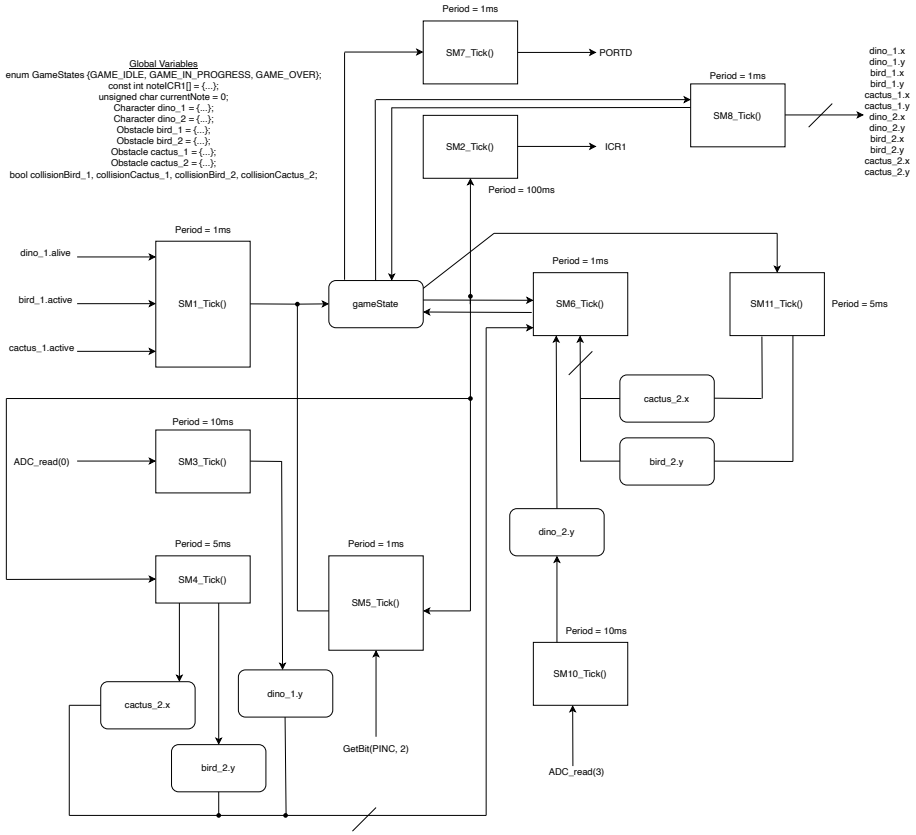


Figure 3: Task diagram

8 SynchSM Diagrams

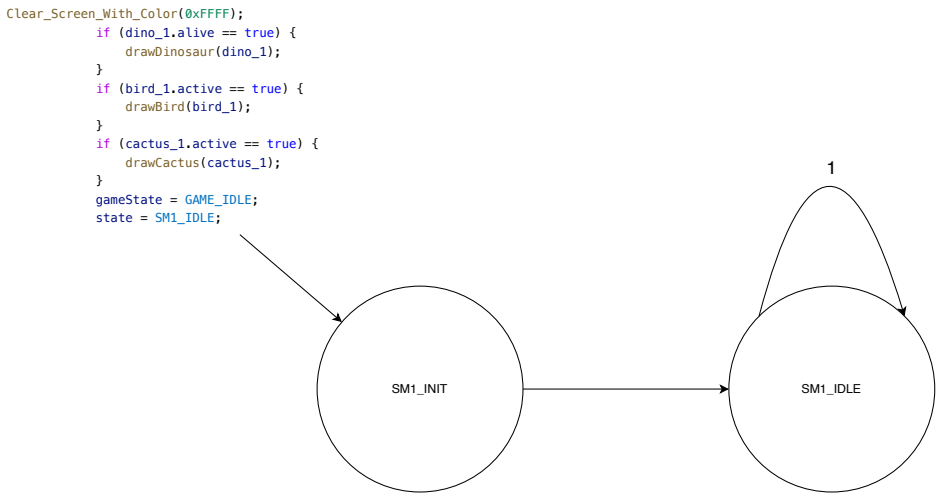


Figure 4: State Machine 1 - Initialize game screen
Task period - 1ms

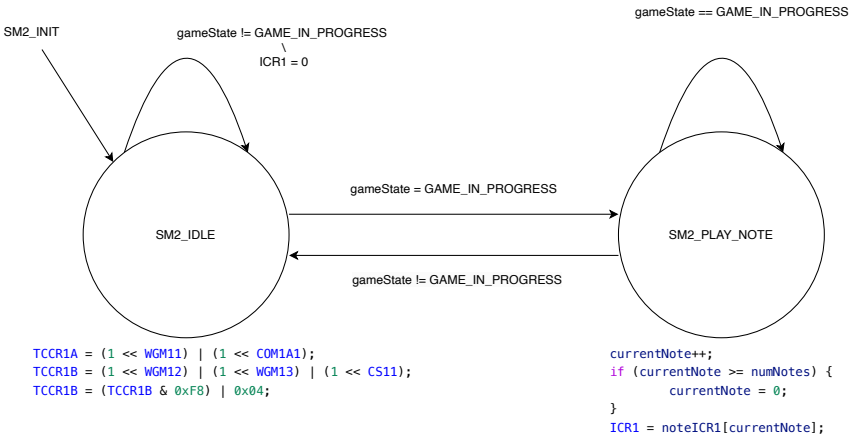


Figure 5: State Machine 2 - Passive buzzer for background music
Task period - 1ms

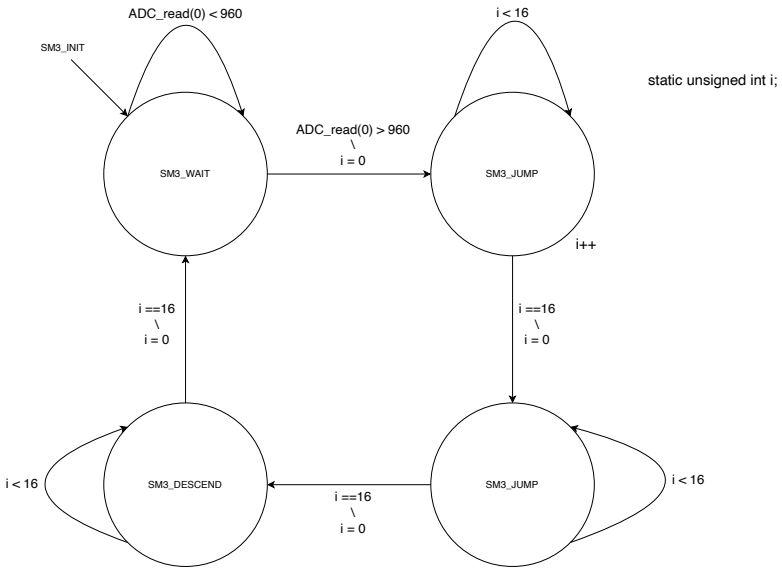


Figure 6: State Machine 3 - dino_1 jumping mechanic
Task period - 100ms

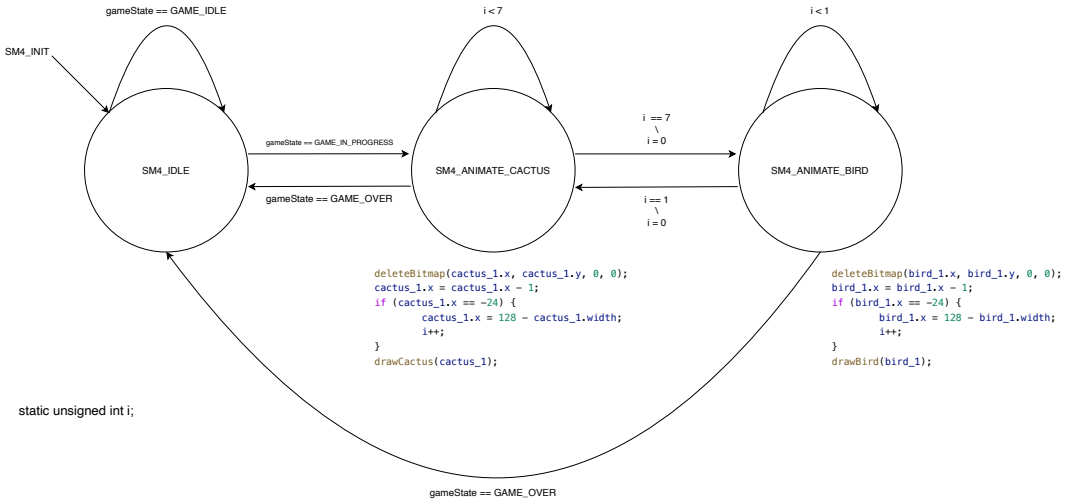


Figure 7: State machine 4 - Obstacle animation & generation for player 1
Task period - 10ms

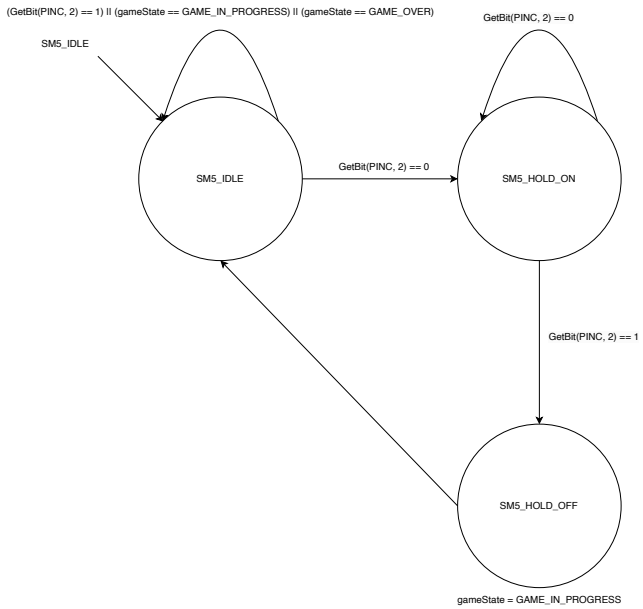


Figure 8: State machine 5 - Game start handler
Task period - 5ms

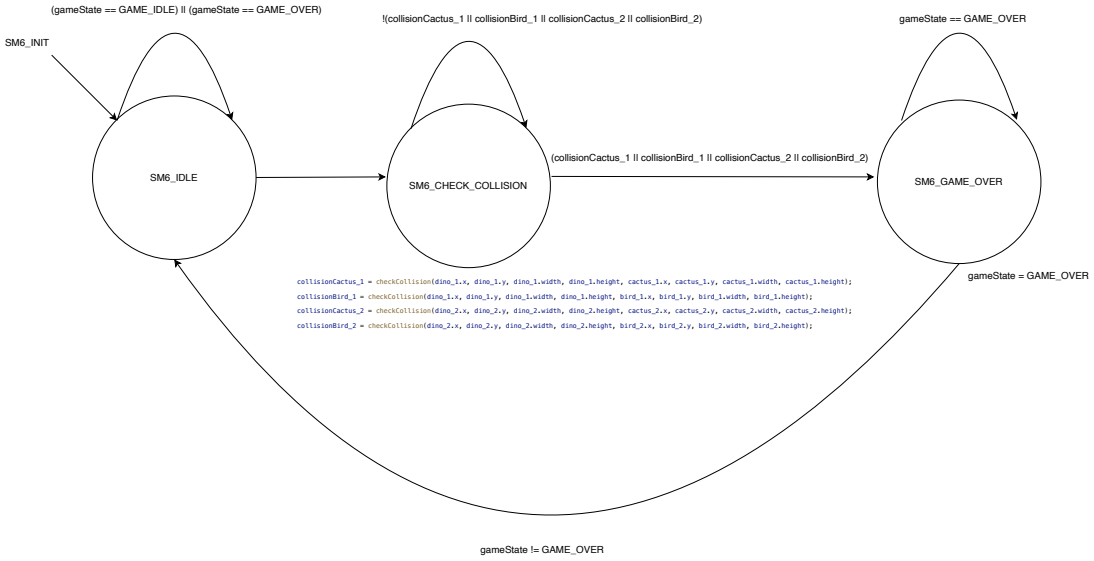
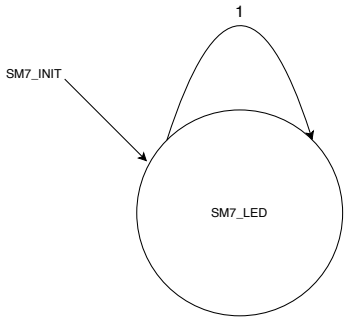


Figure 9: State machine 6 - Game end handler
Task period - 1ms




```
if (gameState == GAME_IDLE) {
        PORTD = SetBit(PORTD, 2, 0);
        PORTD = SetBit(PORTD, 3, 1);
        PORTD = SetBit(PORTD, 4, 0);
}
if (gameState == GAME_IN_PROGRESS) {
        PORTD = SetBit(PORTD, 2, 0);
        PORTD = SetBit(PORTD, 3, 0);
        PORTD = SetBit(PORTD, 4, 1);
}
if (gameState == GAME_OVER) {
        PORTD = SetBit(PORTD, 2, 1);
        PORTD = SetBit(PORTD, 3, 0);
        PORTD = SetBit(PORTD, 4, 0);
}
```

Figure 10: State machine 7 - Game start handler
Task period - 1ms

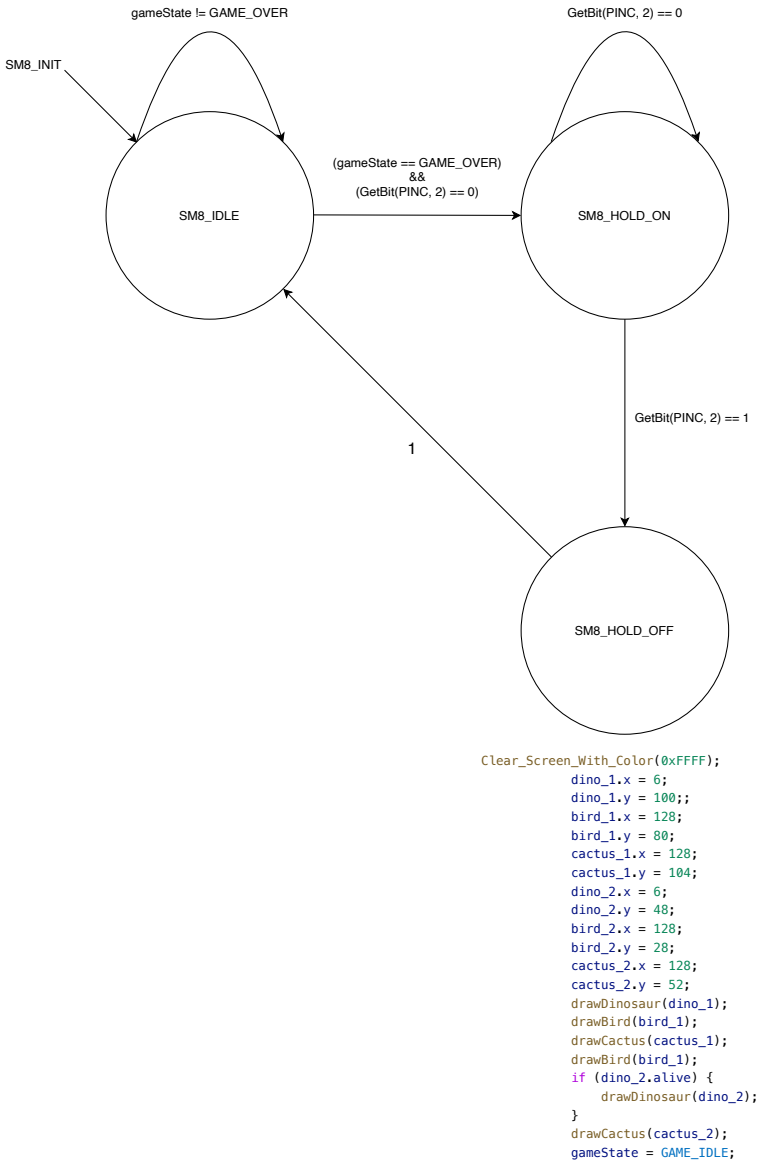
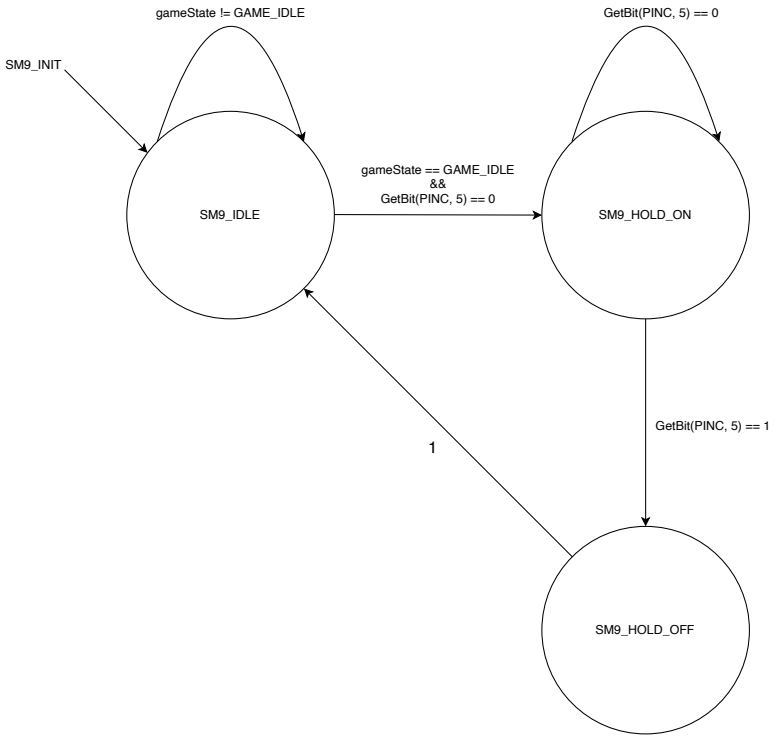


Figure 11: State machine 8 - Game end handler
Task period - 1ms




```
if (dino_2.alive == false) {
        dino_2.alive = true;
        drawDinosaur(dino_2);
}
else if (dino_2.alive == true) {
        dino_2.alive = false;
deleteBitmap(dino_2.x, dino_2.y, 36, 36);
}
```

Figure 12: State machine 9 - Player 2 handler
Task period - 1ms

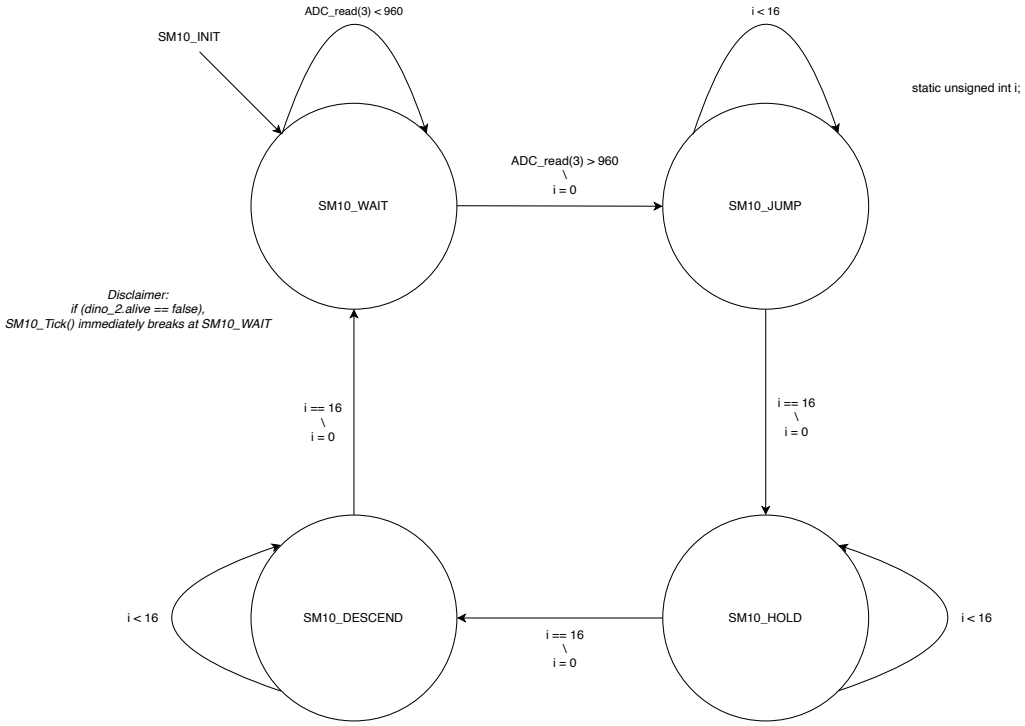


Figure 13: State machine 10 - dino_2 jumping mechanic
Task period - 10ms

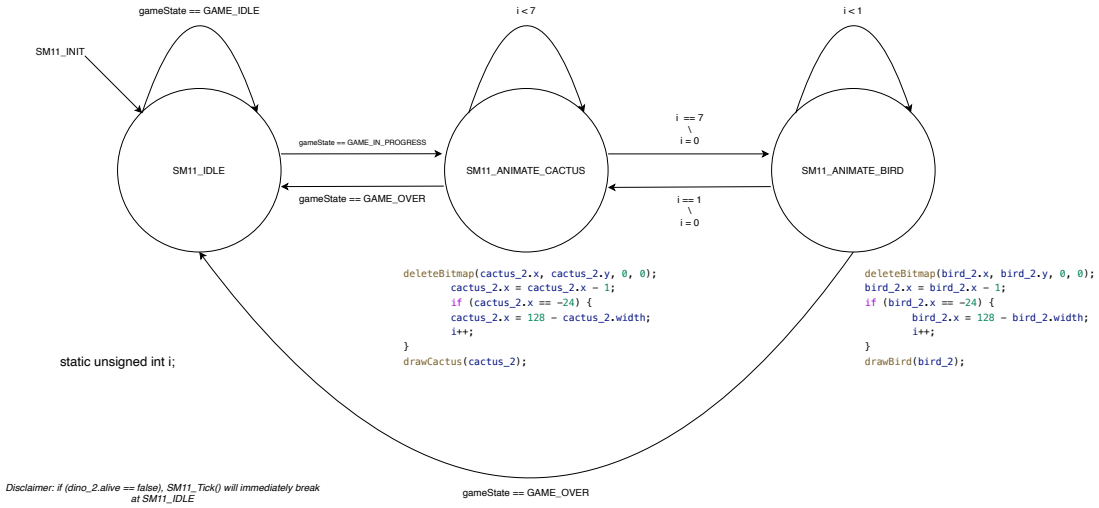


Figure 14: State machine 11 - Obstacle animation & generation for player 2
Task period - 5ms

9 Acknowledgements

Thank you to Professor Brisk, TA Ratnodeep "RB" Bandyopadhyay, and TA Marios Nicolaides your instruction and guidance throughout the CS/EE 120B Spring ’24 course.