IRELE - MA1 Electrical Engineering

ELEC-Y418

Sensors and Microsystem Electronics Project: Snake Game

Professor :

KUIJK Maarten

Assistant :

BRUNEEL Tuur

LAPAUW Thomas

Authors :

BOLLENGIER Alexis

Academic year :

2024 - 2025

Contents

1 Introduction **1**

2 Gameplay Overview **2**

3 Implementation **4**

4 Conclusion **6**

1 Introduction

This project implements a snake game on an ATmega328p microcontroller using assembly language. The game has a dynamic display that is refreshed using a timer interrupt and a screen buffer method. Keyboard input is managed using the 2-step row-column scanning technique from the lab course, allowing the user to control the snake's movement with four directional keys, as well as restart and pause the game. Obstacles are initialized on the screen and food is randomly generated. As the snake eats food, the score is updated, and the snake's speed increases as the score increases.

The start screen is as below to invite the player to play by pressing the bottom left button as indicated on 1.1

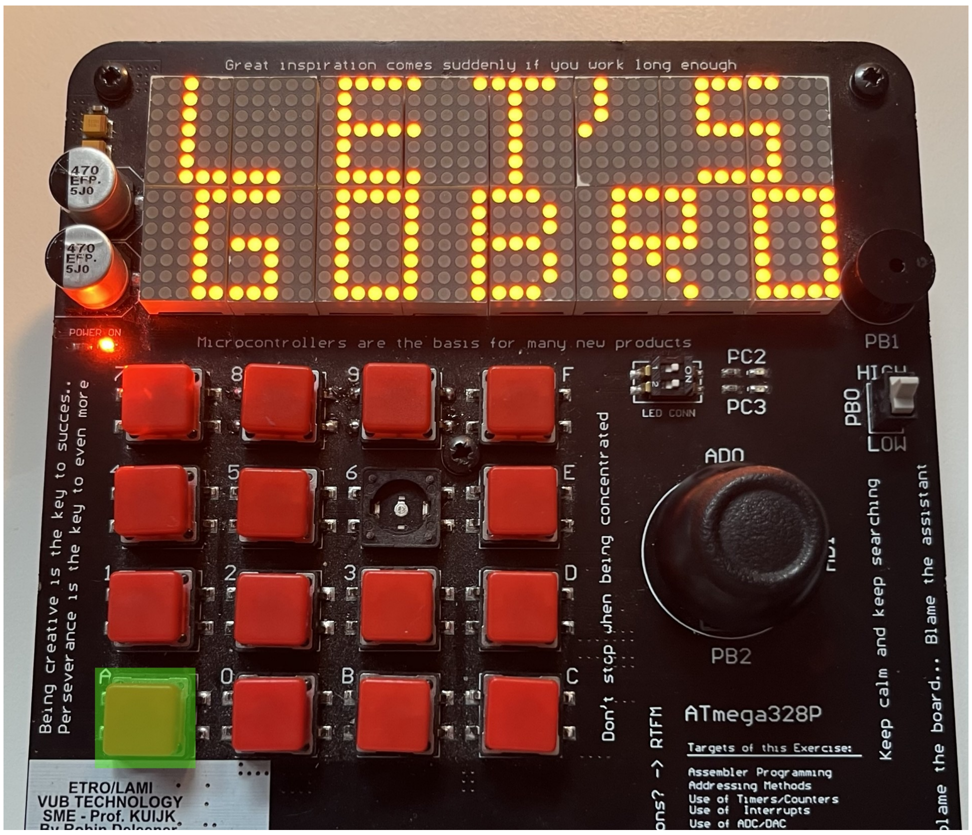
LET'S
GO BRO
Great inspiration comes suddenly if you work long enough
Microcontrollers are the basis for many new products
ATmega328P
ETRO/LAMI
VUB TECHNOLOGY
SME - Prof. KUIJK

Figure 1.1: Start Screen of Snake Game

2

Gameplay Overview

The Snake game is displayed on the LED matrix, as shown in Figure 2.1. The player controls the snake with four directional keys: *Up*, *Down*, *Left*, and *Right*. The snake will move continuously in the selected direction, and the player can change its direction at any time by pressing the corresponding buttons.

The game starts when a direction button is pressed. Food appears at random positions on the screen, and when the snake eats the food the score increases, and a new piece of food is placed on the screen. Obstacles are placed all over the screen, and the snake has to avoid them to avoid crashing and ending the game. When the snake reaches the edge of the screen, it wraps around and reappears on the other side. As the score increases, the snake's speed increases, making the game more challenging.

The game can be paused at any time by pressing the "Pause" button, and to resume, the player must press a directional button.

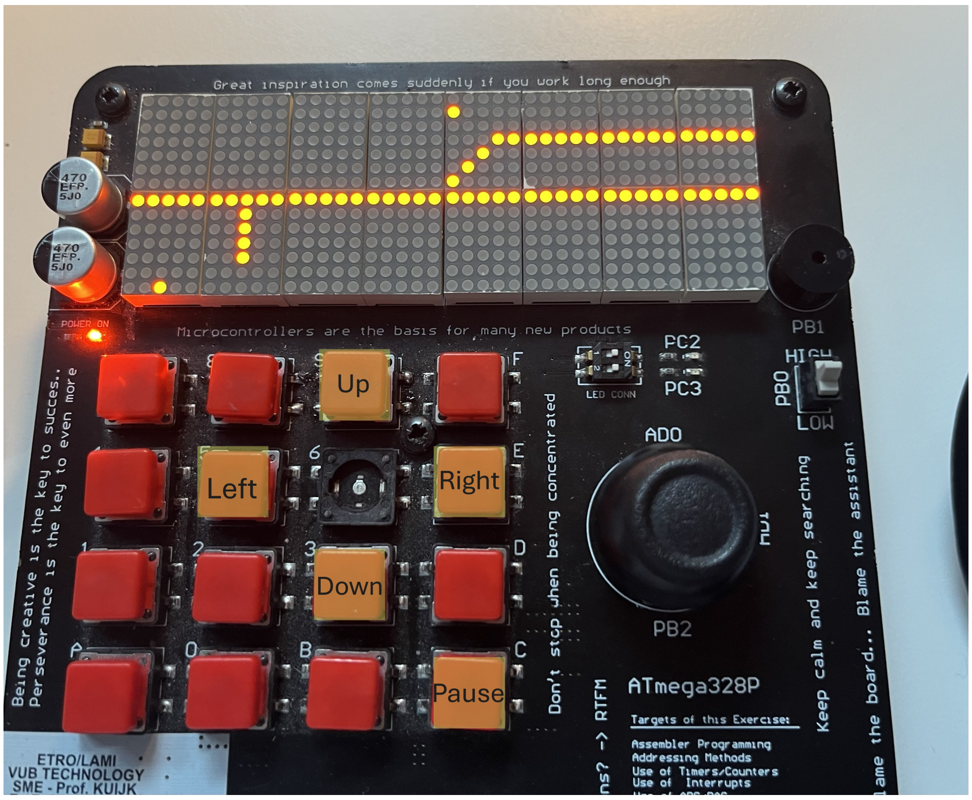
Great inspiration comes suddenly if you work long enough
Microcontrollers are the basis for many new products
Up
Left
Right
Down
Pause
PB1
PC2
PC3
HIGH
LOW
PB0
AD0
PB2
ATmega328P
Being creative is the key to succes..
Perseverance is the key to even more
Don't stop when being concentrated
Keep calm and keep searching
Blame the board... Blame the assistant
ETRO/LAMI
VUB TECHNOLOGY
SME - Prof. KUIJK

Figure 2.1: Gameplay Screen of Snake Game

Once the game is over, the "GAME OVER" message is displayed on the screen, as shown in Figure 2.2, along with the player's score, which is displayed from 0 to 9. The player can directly redo a game or go to the start screen using the corresponding button.

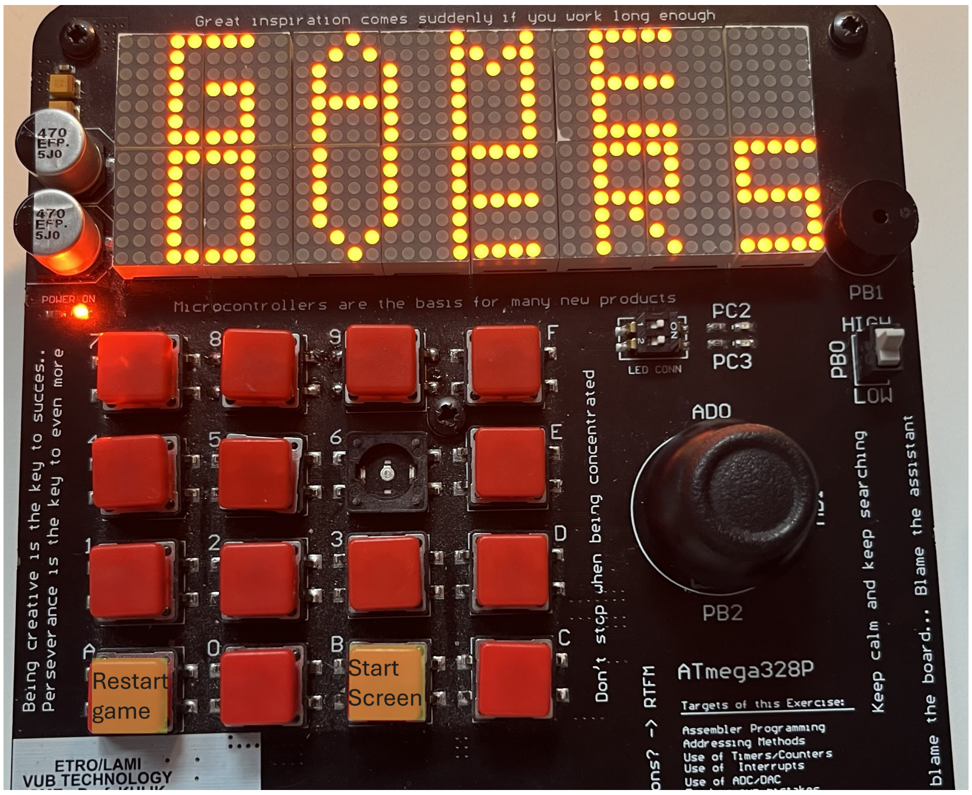

Figure 2.2: Game Over Screen with Score Displayed

3 Implementation

The Snake game is implemented on the ATmega328p microcontroller using assembly language, which integrates game logic, input handling, and display management.

Keyboard input is handled by the *Rowdetection* macro, which uses a 2-step scanning technique as described in the lab slides. The *ReadKeyboard* function calls this macro to process user input to control the snake's movement, pause, or restart the game.

The display is managed by a memory buffer starting at address 0x0100. Routines like *SetPosBuffer* update the snake's position on the LED matrix, while *ClearScreen* clears the buffer and then the screen. Timer 0 is used to periodically refresh the display, ensuring that the game state is updated without blocking other tasks, with the line-by-line display refresh designed to be imperceptible to the human eye.

The snake's movement is controlled by the *SnakeMain* routine, which updates the snake's position based on the direction stored in *SnakeDirection*. Obstacles are initialized by the *InitObstacles* routine and food is randomly placed by *GenerateFoodPos*. Collision detection is handled by the *CheckObstacles* and *CheckFoodCollision* functions, ensuring that the snake interacts correctly with obstacles and food.

The Start and Game Over screen is implemented using a CharTable as also explained in the lab slide. The CharTable is placed in flash memory immediately after the code and consists of 24 entries of 8 bytes each, where each byte encodes one row for the characters. When the *GameOver* or *LetsGO* routine runs, it loads the appropriate character indices from the CharTable.

Timer 1 was initially used to control game speed and handle user input. However, with the addition of features such as the start and game-over screens, the Timer 1 interrupts were removed. Nevertheless, the Timer 1 code remains in place for potential future improvements to the game, if should time permit.[1]

[1] The sheme bloc can also be uploaded via the xml code provided in the zip file in *draw.io*

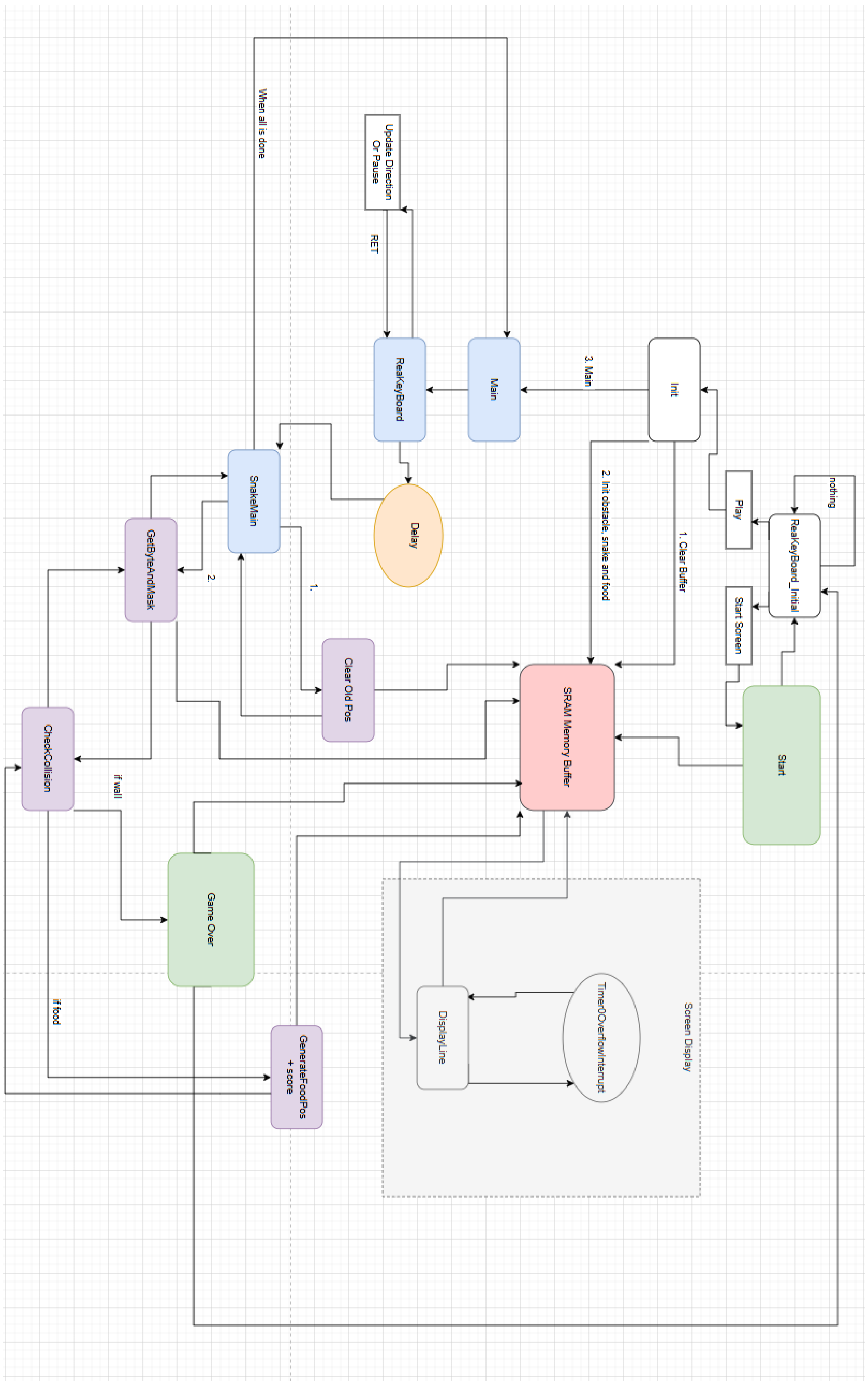

Figure 3.1: Block Shematic of Project Structure

4 Conclusion

In conclusion, the Snake game works as intended on the ATmega328p microcontroller, successfully implementing timers, interrupts, and the keyboard matrix. For future improvements, the snake's length could increase with each food item consumed, and different difficulty levels could be introduced by adjusting the number of obstacles and initial speed. These improvements would increase the challenge and replayability of the game.