Ben Pham

Dec 9, 2024

Among US Runner

Introduction

In my project, I am doing Among US Runners. It is a game where you avoid getting hit by spikes while moving your Among US character. It is a single-player game, and the goal is to avoid the spikes by moving the character across the screen using the remote as a controller. The player aims to get the highest score before crashing into the obstacles.

Some things that were not implemented were the increasing difficulty of obstacles the reasoning for this is because of the limit of using LCD as the period of the LCD has to be set to a limit to even register the obstacles. The IR Remote Sensor also sometimes needed to be spammed rather than pressing the button once on the remote to move up. This is due to the remote sensor sometimes won't pick up on time.

Build-upons

- (½) Custom Character Sprite: Shows an Among Us character running and spikes for the obstacles.
- (½) EERPROM: Stores the high score for the game.

- (1) Passive Buzzer for Music: Plays music when pressed the play button.
- (1) IR Remote: Controls the AMONG US character jumping, the music button, and the power button to play the game.

User Guide

The user would interact with the game by pressing the red power button on the remote. The game is on when connected to the computer system and should show the character sprite on the bottom with the top screen saying press power to play the game. The user would press the power button, and the game would start. Spikes from the bottom would come toward the user, and the goal is to get the highest score in the game. The score is outputted to the very right of the LCD screen. With the high score at the top right and the current score when playing at the bottom right. There is also a music feature where the player can choose the pause or play button, which is the very middle button with a circle around it. They will be able to play music while playing their game; if they don't like the music, they can just turn it off. The goal of the game is to try to get the highest score before dying to the obstacles and if you die, you can keep playing again.

Hardware Components Used

- Wires
- Passive Buzzers
- LCD
- IR Remote

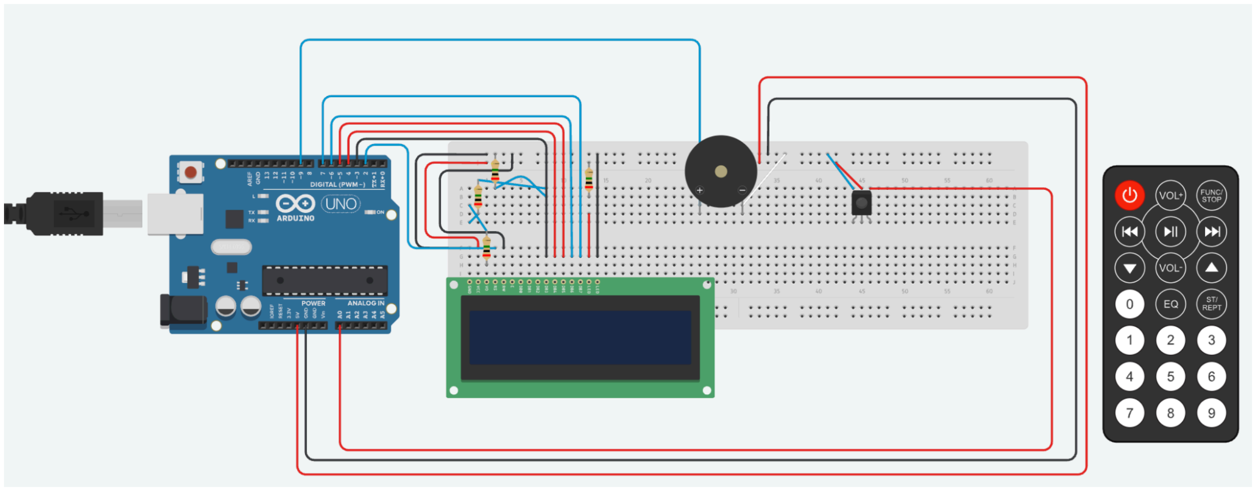
- IR Sensor
- 2K Resistors (4)

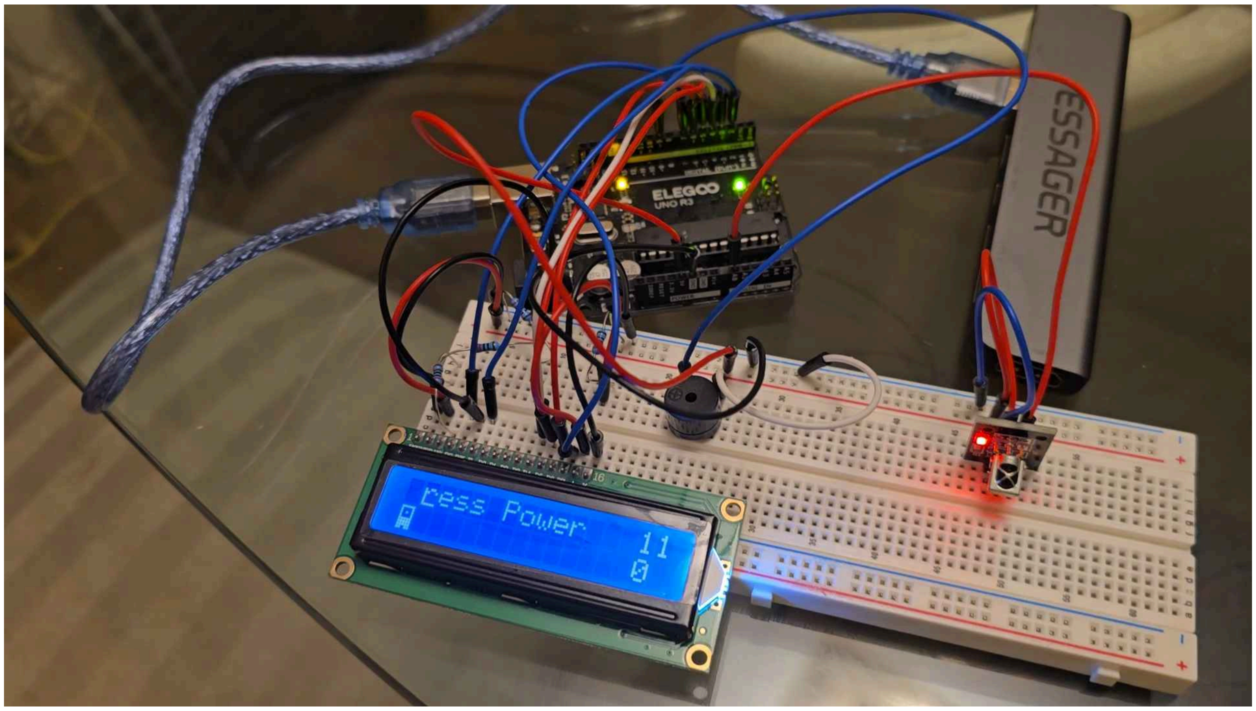
Software Libraries Used

- LCD.h
  - This helped with displaying the characters and obstacles for the game. It also helped display the score and high score.
- irAVR.h
  - Helped with the implementation of the remote control. This library is used to start the game when pressing the red power button. Play music in the middle button with the play and pause. THere is also jumping when pressing the up button.
- serialATmega.h
  - This helped with debugging the game to check for any errors in the states.
- Stdio.h
  - Helped with the conversion of string to int for the score of my game and the high score also.
- Stdlib.h
  - Helped with the inclusion of the boolean and global variables.
- timerISR.h
  - Helped with managing the tasks for my game.
- avr/eerprom.h

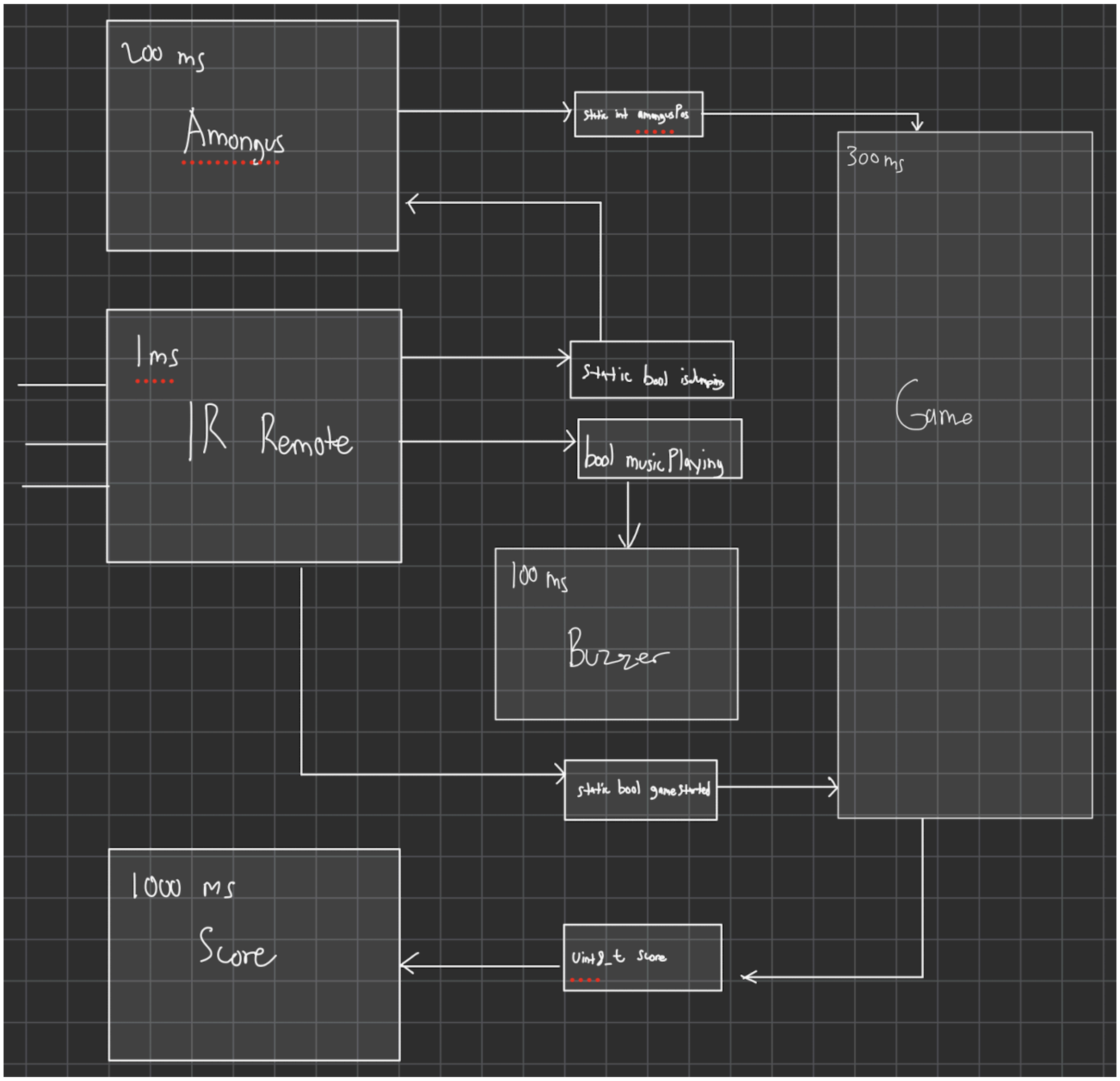
  - Helped keep track of the high score for my game and store it in the memory of my Arduino.

Wiring Diagram



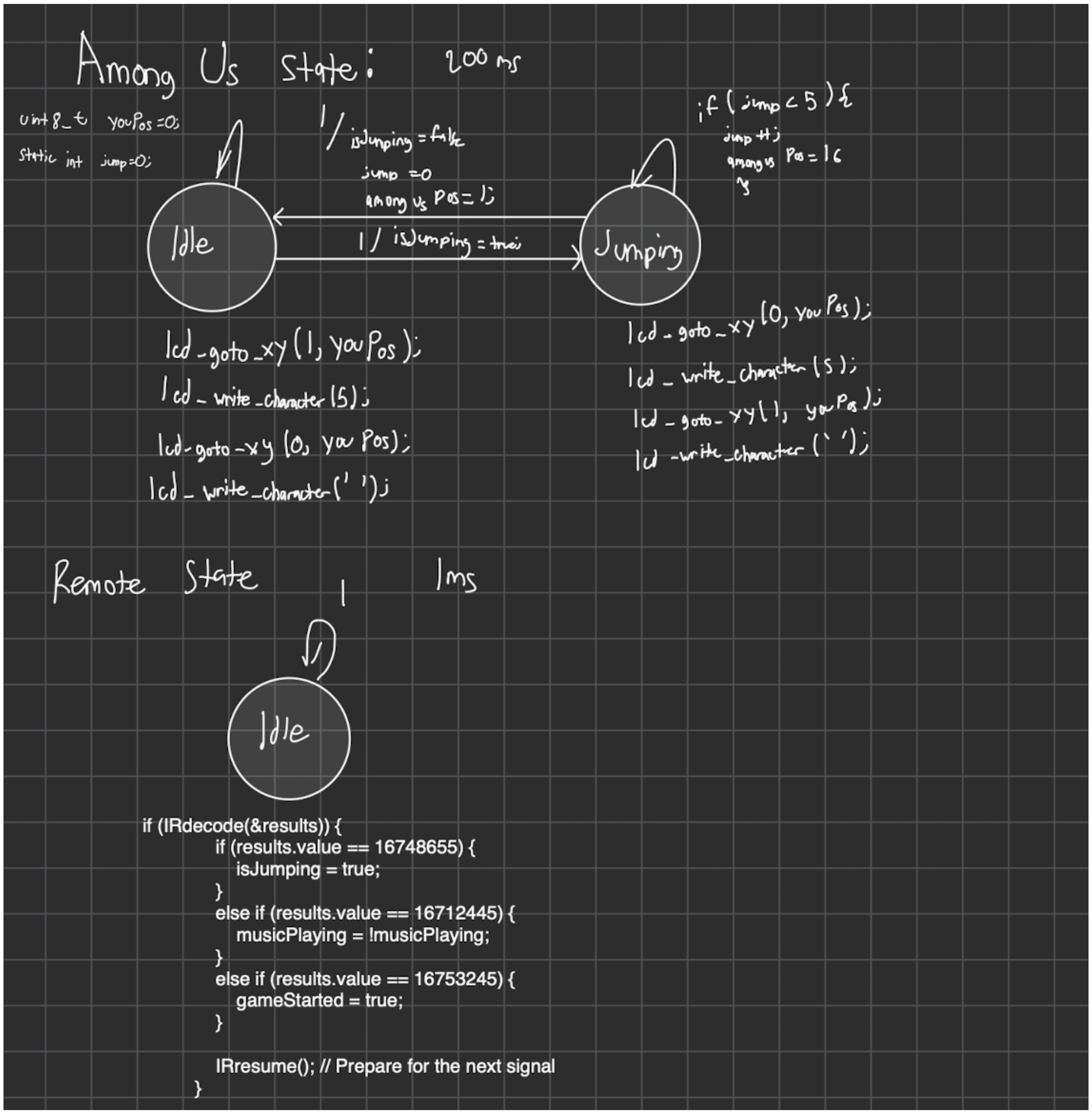


Task Diagram



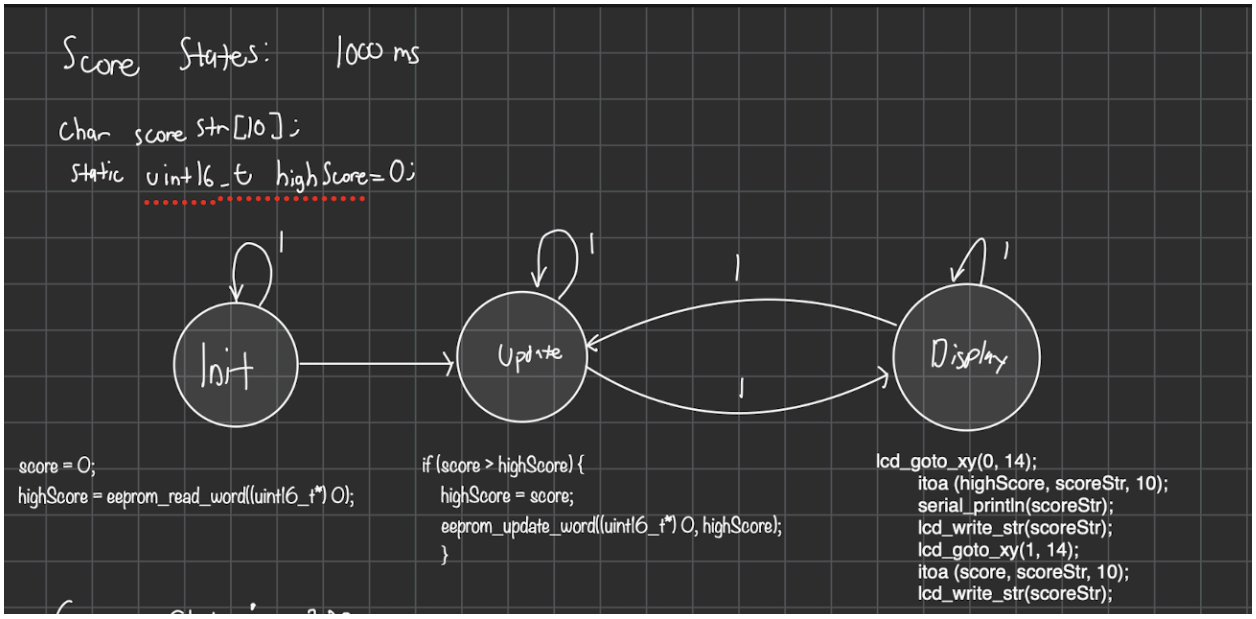
SynchSM Diagrams

Draw a SynchSM diagrams for **all tasks** in the system.



```
if (IRdecode(&results)) {
        if (results.value == 16748655) {
            isJumping = true;
        }
        else if (results.value == 16712445) {
            musicPlaying = !musicPlaying;
        }
        else if (results.value == 16753245) {
            gameStarted = true;
        }

        IRresume(); // Prepare for the next signal
    }
```

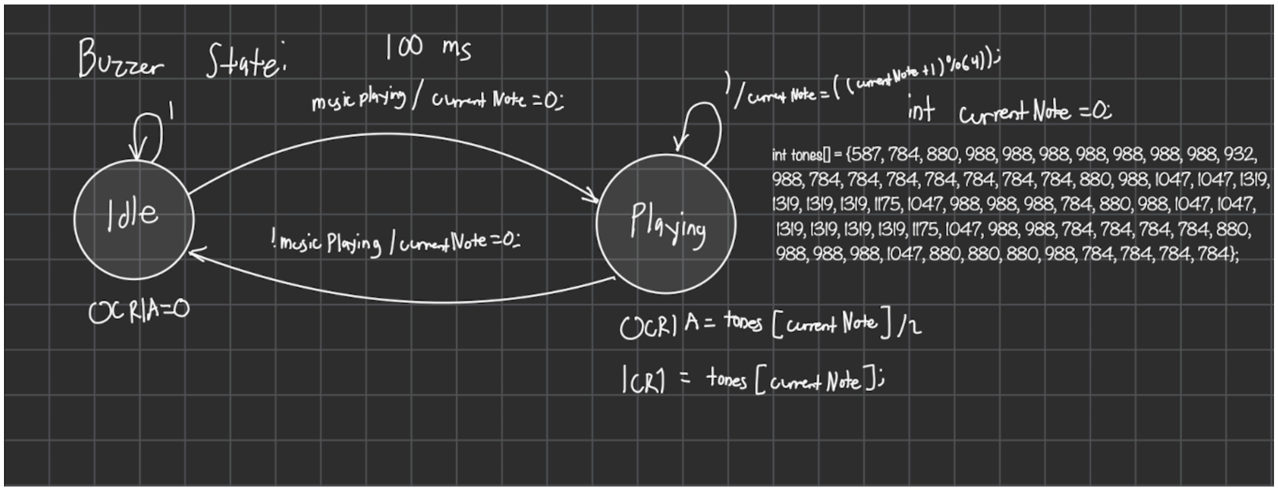


```
char scoreStr[10];
static uint16_t highScore=0;
```

```
score = 0;
highScore = eeprom_read_word((uint16_t*) 0);
```

```
if (score > highScore) {
    highScore = score;
    eeprom_update_word((uint16_t*) 0, highScore);
}
```

```
lcd_goto_xy(0, 14);
    itoa (highScore, scoreStr, 10);
    serial_println(scoreStr);
    lcd_write_str(scoreStr);
    lcd_goto_xy(1, 14);
    itoa (score, scoreStr, 10);
    lcd_write_str(scoreStr);
```

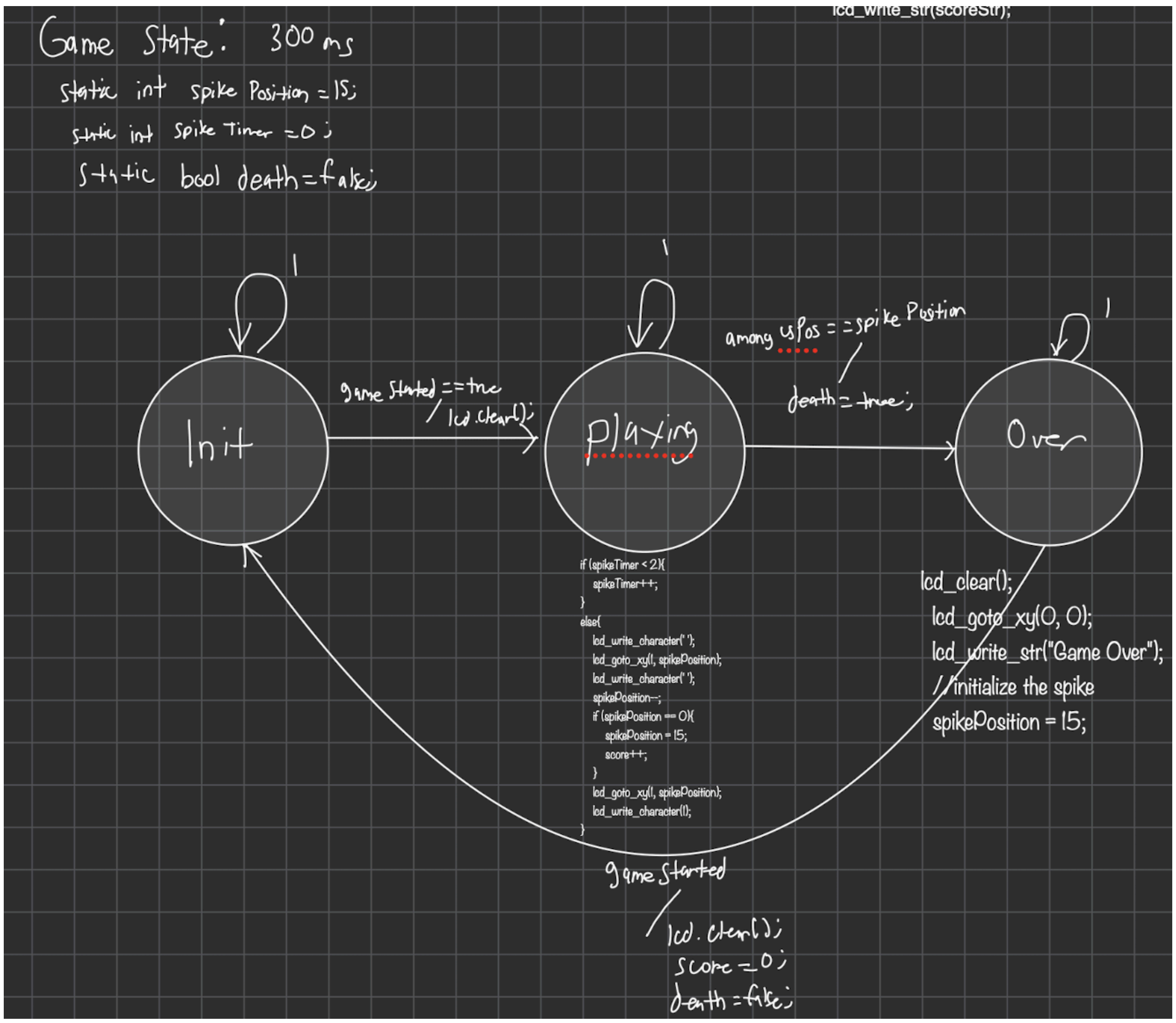

Buzzer State: 100 ms
Idle
Playing
1
music playing / current Note = 0;
!music Playing / currentNote = 0;
) / current Note = ((currentNote +1)%64));

```
int currentNote =0;
```

```
int tones[] = {587, 784, 880, 988, 988, 988, 988, 988, 988, 988, 932,
988, 784, 784, 784, 784, 784, 784, 784, 880, 988, 1047, 1047, 1319,
1319, 1319, 1319, 1175, 1047, 988, 988, 988, 784, 880, 988, 1047, 1047,
1319, 1319, 1319, 1319, 1175, 1047, 988, 988, 784, 784, 784, 784, 880,
988, 988, 988, 1047, 880, 880, 880, 988, 784, 784, 784, 784};
```

```
OCR1A=0
```

```
OCR1A = tones [current Note] /2
ICR1 = tones [current Note];
```



```
static int spike Position = 15;
static int spike Timer = 0;
static bool death = false;
```

```
if (spikeTimer < 2){
    spikeTimer++;
}
else{
    lcd_write_character(' ');
    lcd_goto_xy(1, spikePosition);
    lcd_write_character(' ');
    spikePosition--;
    if (spikePosition == 0){
        spikePosition = 15;
        score++;
    }
    lcd_goto_xy(1, spikePosition);
    lcd_write_character(1);
}
```

```
lcd_clear();
 lcd_goto_xy(0, 0);
 lcd_write_str("Game Over");
 //initialize the spike
 spikePosition = 15;
```