CS/EE 120B Custom Laboratory Project Report
2048
Red Su
6/10/2025

Introduction

2048 is a strategic single-player game played on a 4x4 grid. The game's goal is to combine tiles of power of 2 until reaching the tile “2048.” And if the grid fills up before reaching 2048, game over. Each move, up, down, left, or right from the joystick, will randomly spawn a new tile, making the game more difficult as it progresses. The game is also presented with a 4D7S connected to a shift register that displays the score as the game progresses, and a button that can start or reset the game.

Build-upons

- Shift register used to connect to the 4D7S using SPI.
- HiLetgo Colorful SPI TFT LCD Display. Displays a start screen, the actual game, and a win/lose screen.
- Complex game logic
    - Win and lose conditions
    - Merging tile rules
    - Tile movement (up, down, left, right)
    - Spawning random tiles
    - Display any tile (2, 4, 8…, 2048) on any tile on the grid with the corresponding color.

User Guide

When starting the game, the user will be prompted with the starting screen that displays “2048” and a “START.”
Start the game by pressing the button to the left of the screen. Anytime throughout playing, the game can be reset by pressing the button again, returning to the start screen.

The game can be played using the joystick. When the game starts, 2 “two” tiles will spawn randomly. Every move made with the joystick (up, down, left, or right) will move pre-existing tiles in the joystick’s direction. i.e., moving the joystick right will move every tile in each row to the right. Moving up will move every tile in each column up. Each move will also add a random tile, either two or four, on the grid. A random tile spawns only when the move is valid. That means when a tile moves or when a tile gets merged. When two of the same tiles are on the same row or column, they can be merged.

To win the game, the player must merge the tiles until they reach the tile “2048.”

The game is lost when the 4x4 fills up completely without any possible moves left.

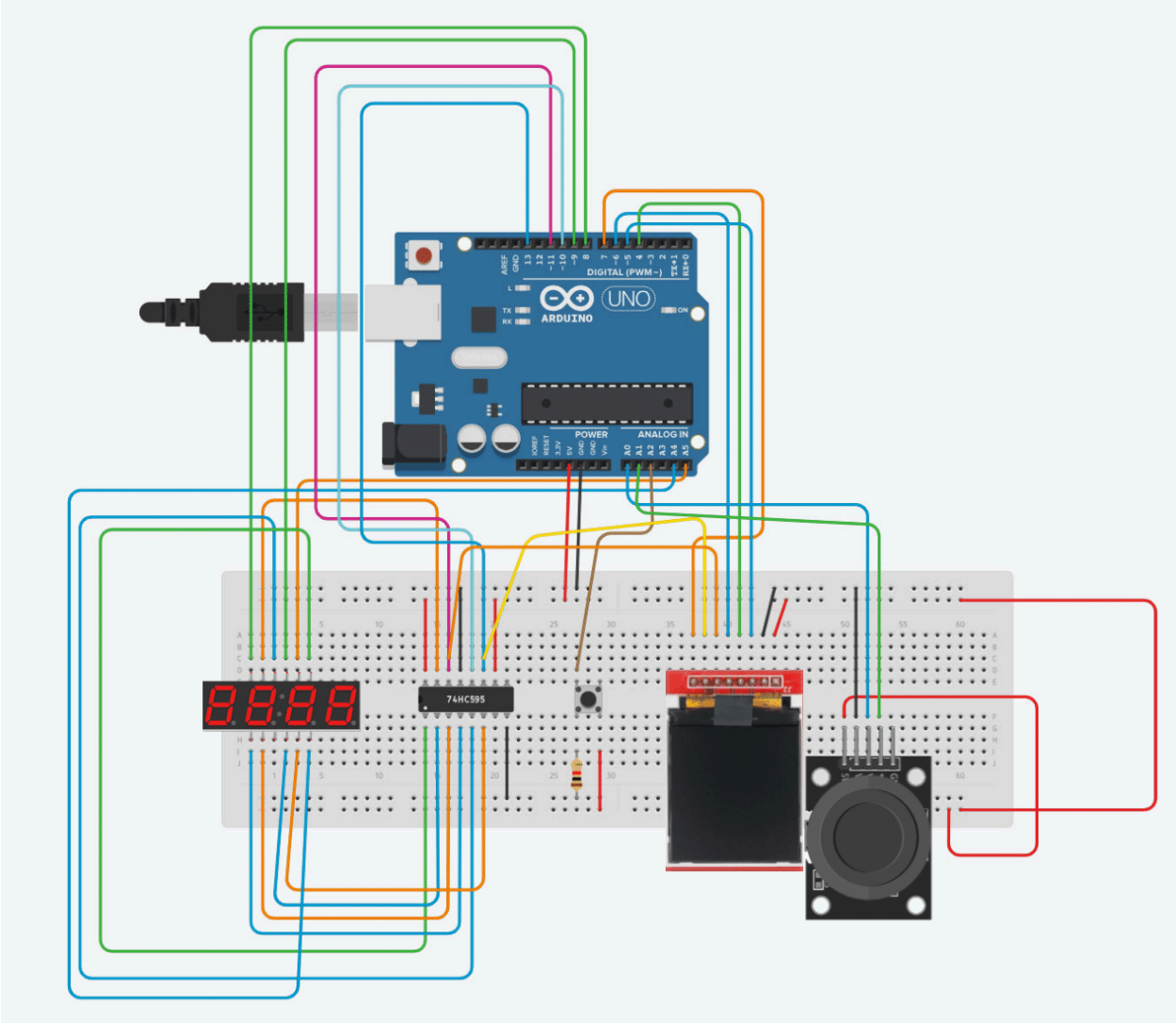
Hardware Components

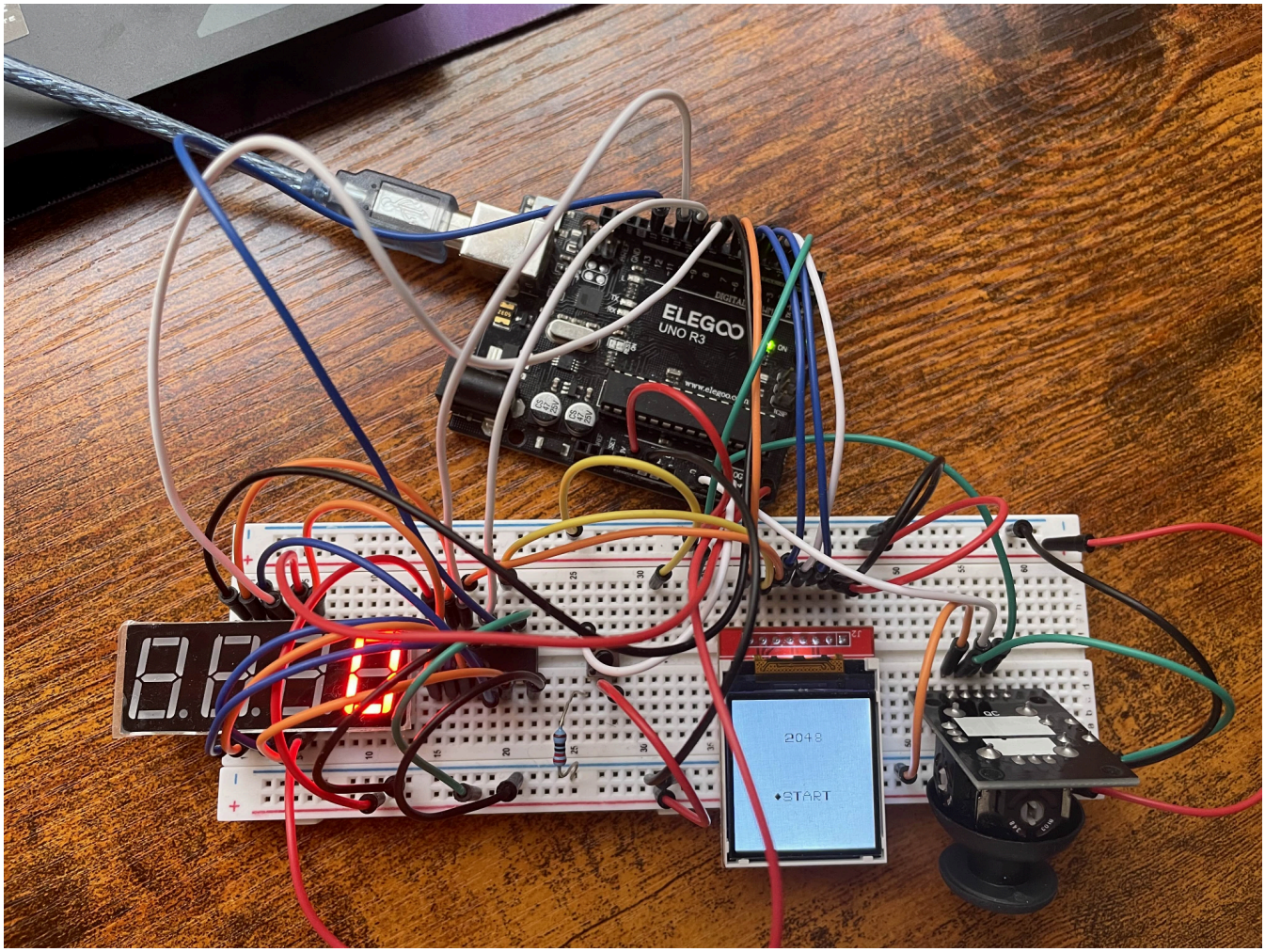
- 4D7S
- HiLetgo Colorful SPI TFT LCD Display
- Button
- Shift Register
- Joystick

Software Libraries

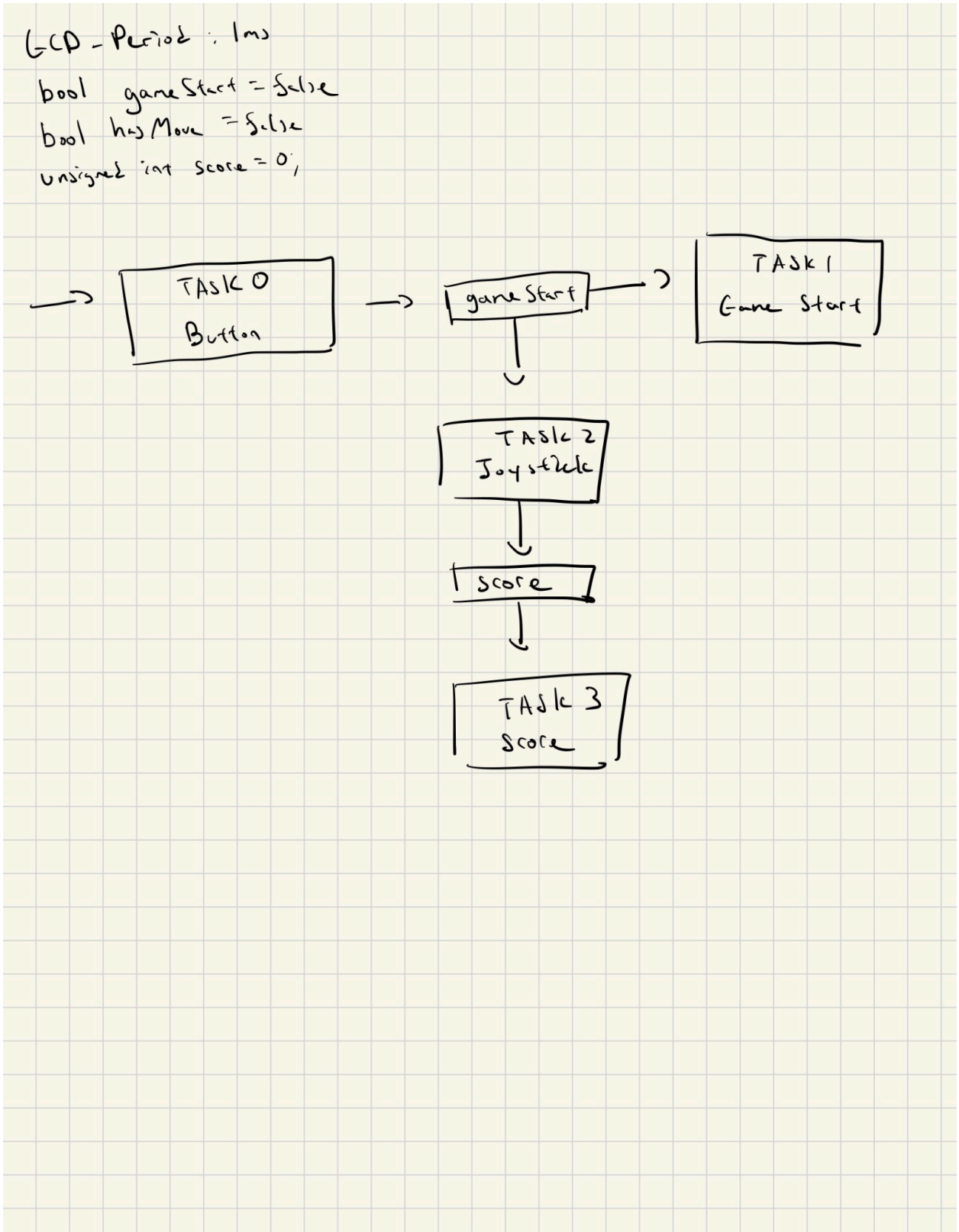
- spiAVR
    - This was used to make both the shift register and LCD display work. Using SPI_SEND() to send data.
- periph
    - This was provided from Lab 7. This was used to read the value of the joystick. Mainly using ADC_read() to read the position of the joystick.
- timerISR
    - Also provided from Lab 7. This was used to work with time in my system.

Wiring Diagram


DIGITAL (PWM~)
UNO
ARDUINO
POWER
ANALOG IN
74HC595



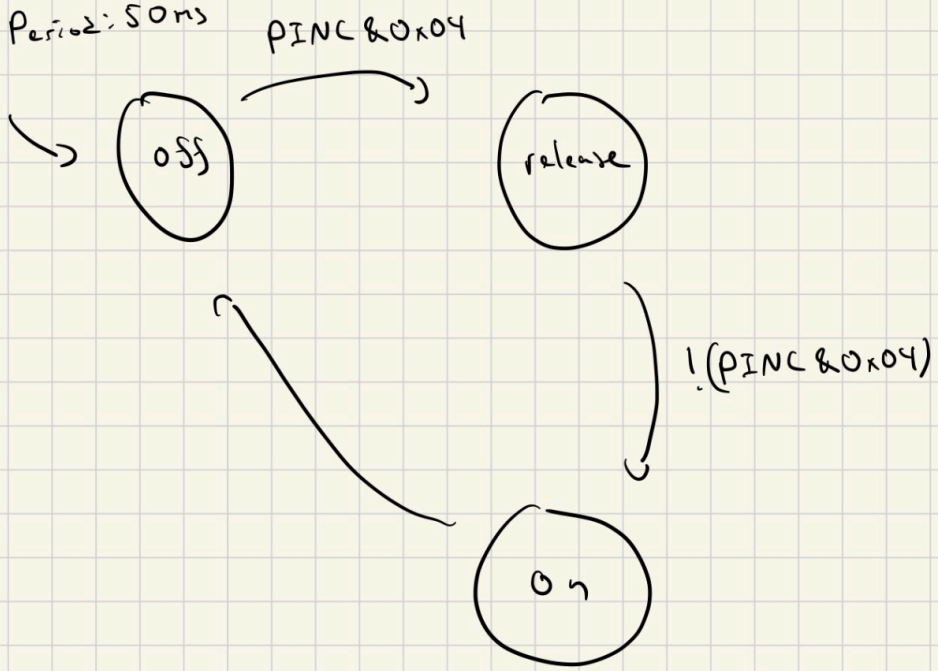
Task Diagram



SynchSM Diagrams

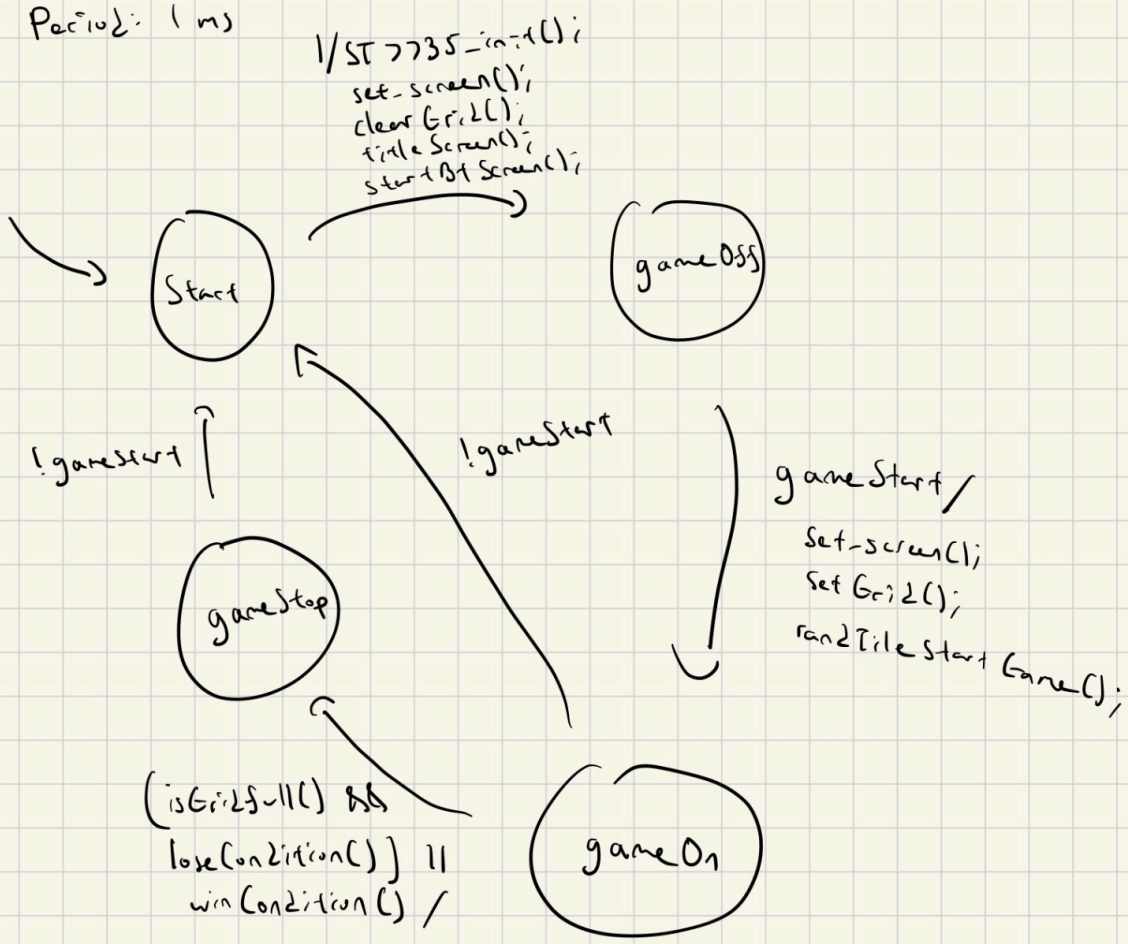
TASK 0

Period: 50ms



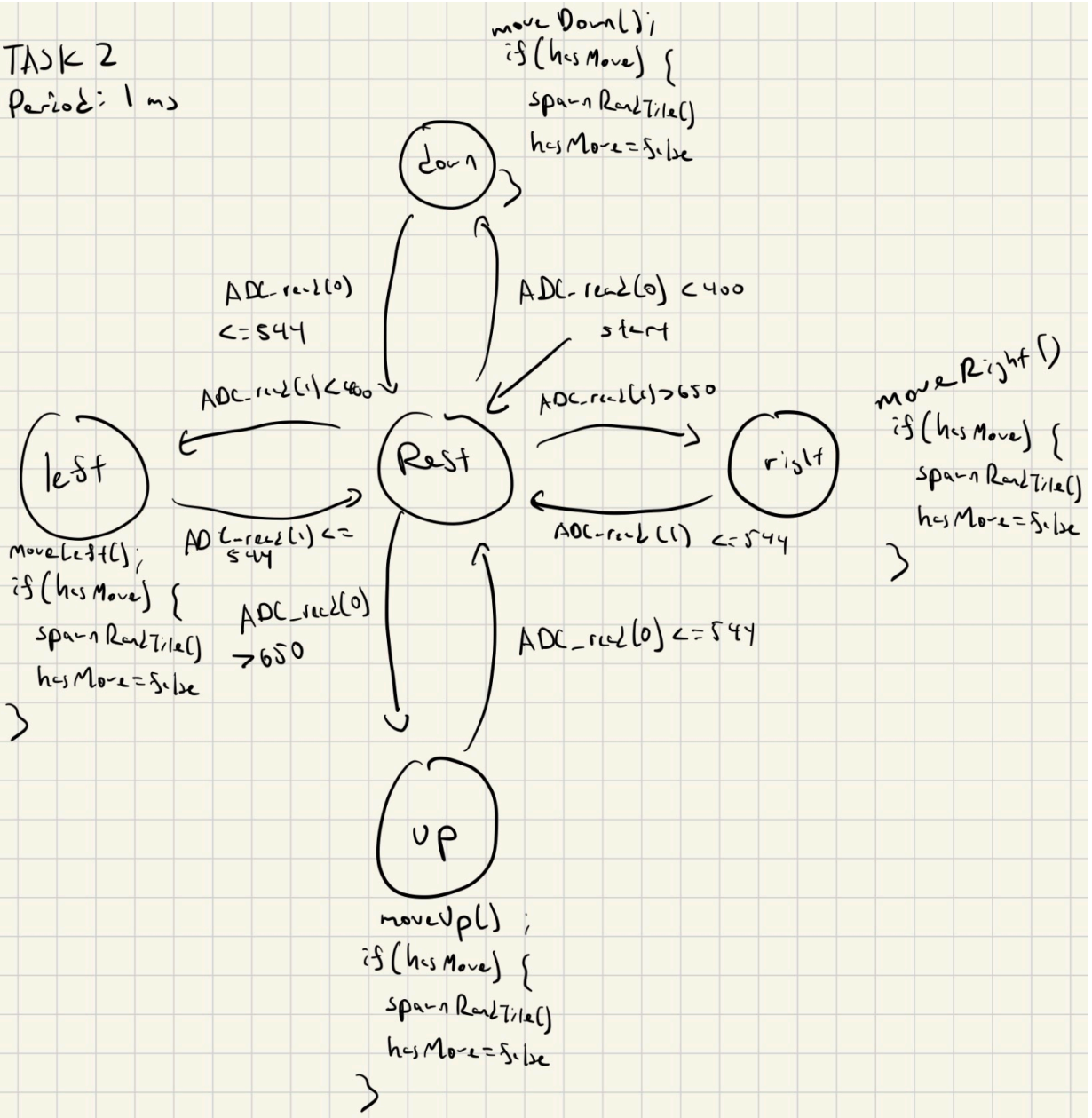

```
if(!gameStart)
    gameStart = true;
else
    gameStart = false;
```

TASK 1
Period: 1 ms

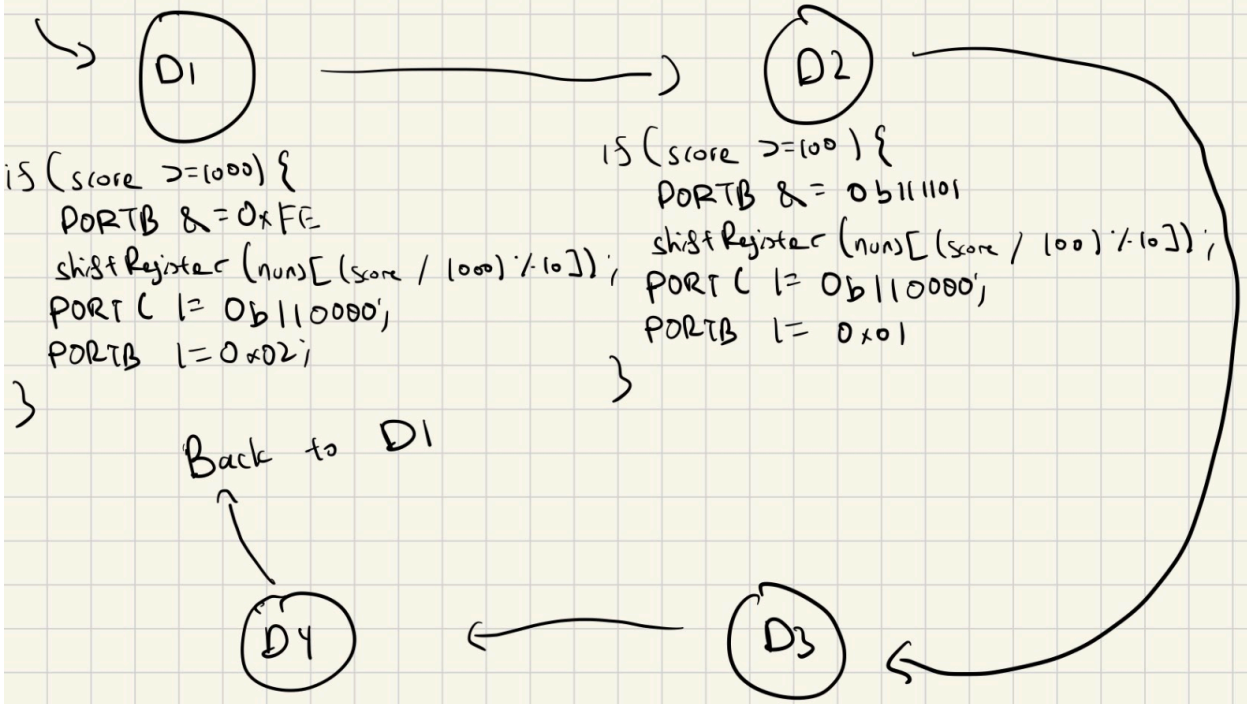



```
if (isGridFull && loseCondition()) {
    set_screen();
    loseScreen();
}
else if (winCondition()) {
    set_screen();
    winScreen();
}
```

TASK 2
Period: 1 ms



TASK 3
Period: 1ms




```
if (score >=1000) {
  PORTB &= 0xFE
  shiftRegister (nums[(score / 1000) %10]);
  PORTC |= 0b110000;
  PORTB |=0x02;
}
```

```
if (score >=100 ) {
  PORTB &= 0b1111101
  shiftRegister (nums[(score / 100) %10]);
  PORTC |= 0b110000;
  PORTB |= 0x01
}
```

```
if (score >=10 ) {
  PORTB &=0b0111111;
  shiftRegister (nums[(score / 10 ) %10]);
  PORTC |= 0b010000;
  PORTB |=0x03;
}
```

```
PORTB &=0b10111
shiftRegister (nums[(score / 10)]
PORTC |= 0b100000;
PORTB |=0x03;
```

Back to D1