2048

UCR 120B Custom Lab Project

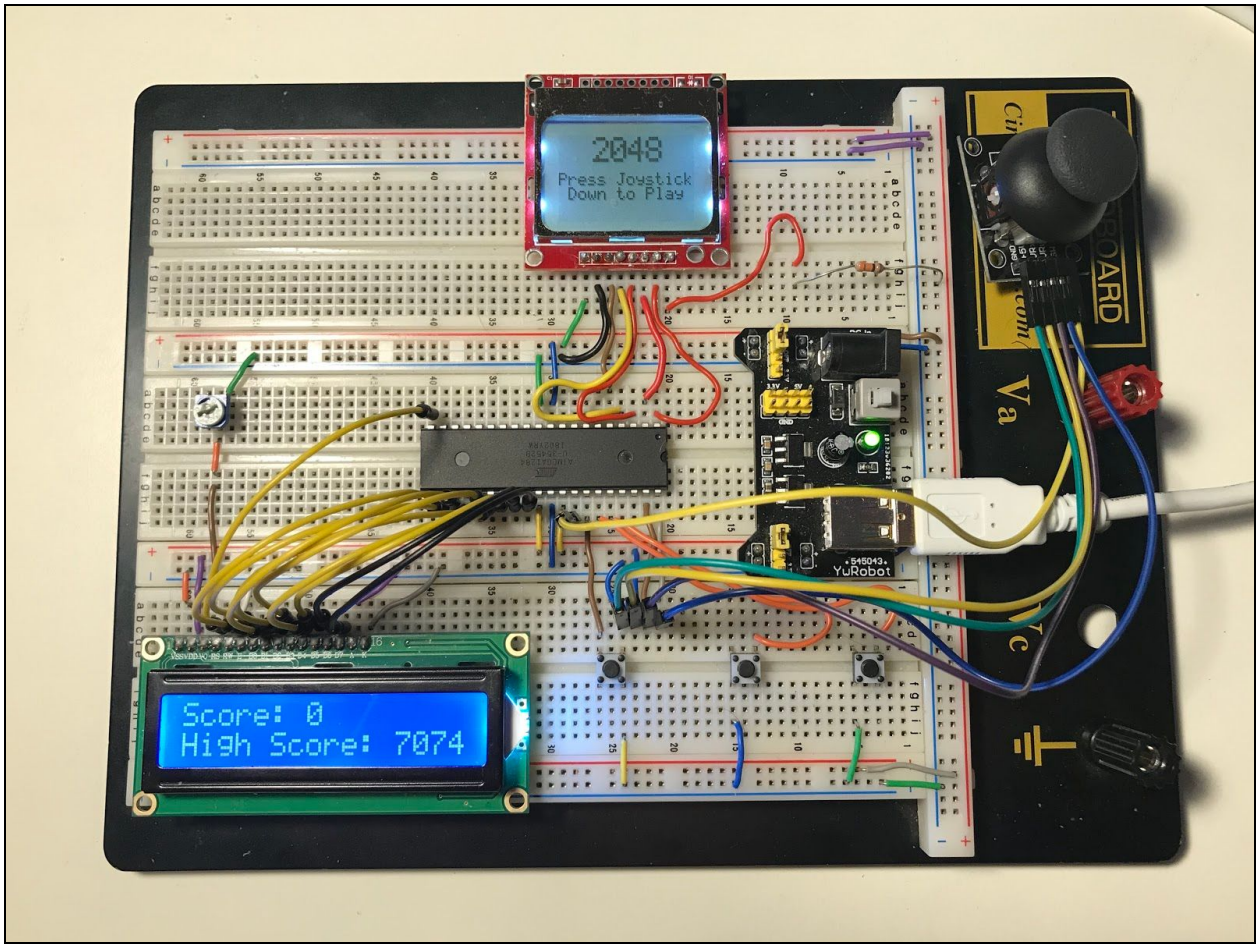
Fall 2019

Jacob Halvorson

Table of Contents

Introduction

For my project I decided to recreate an embedded systems version of the popular mobile game "2048." For this project I used a Nokia 5110 LCD display, 2x16 LCD display, joystick, and buttons. The joystick is used to emulate 'swiping' up, down, left, and right, which will cause the numbers on the grid to move accordingly on the Nokia display. The user's score and high score is displayed on the 2x16 LCD display. The user can also soft reset the game, reset the high score, and enter cheat mode using 3 different buttons.



User Guide

Rules

- "Non-greedy" movement. The tiles that were created by combining other tiles do not combine again during the same move. For example moving the tile row of [2] [2] [2] [2] to the right results in the rile row ……..[4] [4] as opposed to ……..[8].
- "Move direction priority". If more than one variant of combining is possible, move direction shows one that will take effect. For example, moving the tile row of ….[2] [2] [2] to the right results in the tile row of ……..[2] [4] as opposed to ……..[4] [2].
- When the user makes a move, a new tile is randomly generated in an empty tile. The new number will be 4 10% of the time and the rest are 2.
- If no tiles move or combine during a move, no new tile is generated. Only valid moves will generate a new tile.
- You win the game by getting a '2048' tile. After you win the first time you can keep playing to get a higher score.
- You lose the game when the grid is completely full and there are no more possible moves.

Controls

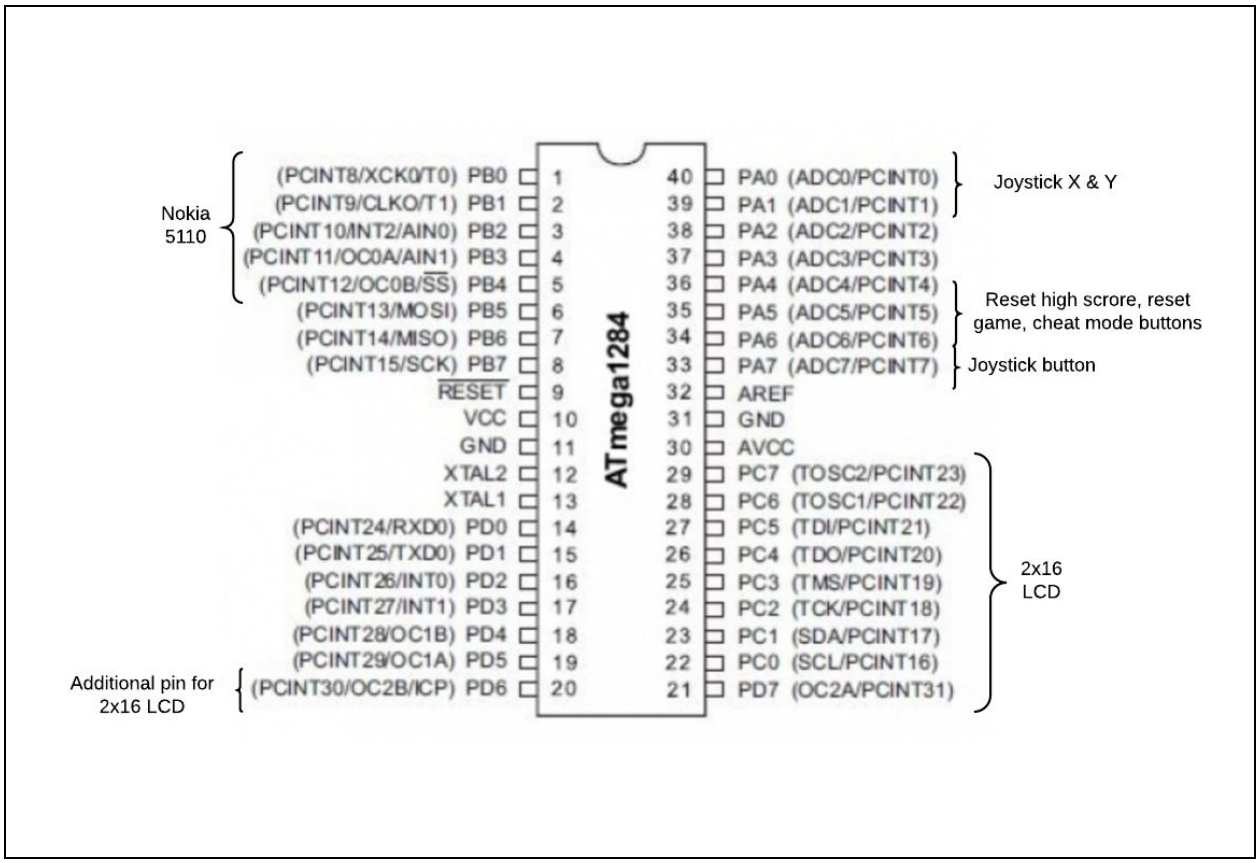
- To start the game, turn the power on and click down on the joystick.
- To move a tile on the grid, move the joystick in the direction you want the tiles to move. Orient the joystick so that the pins coming out of it are pointing down.
- To reset the game, click on the button connected to PA5.
- To reset the high score, click on the button connected to PA4.
- To enter cheat mode, hold down the button connected to PA6 while you click the joystick down on the menu screen. This will start you off with four 1024 tiles.

Hardware

Parts/Technology

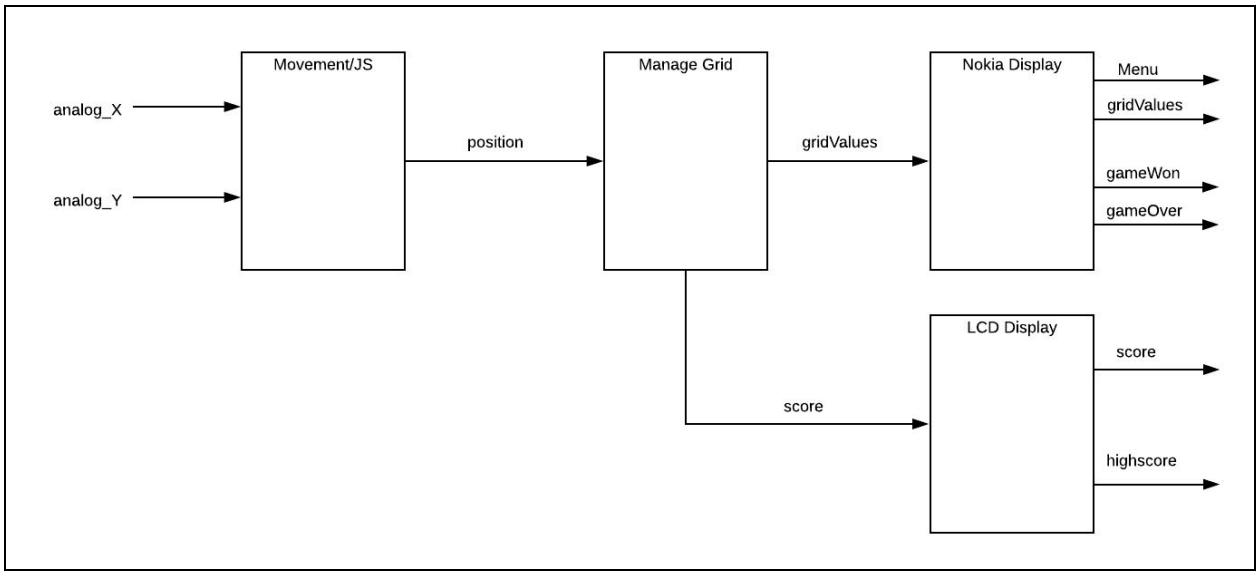
- ATMega1284 microcontroller
- Atmel Studio 7
- 2x16 LCD Screen
- Nokia 5110 LCD Display
- Mechanical Buttons
- Joystick

Pinout

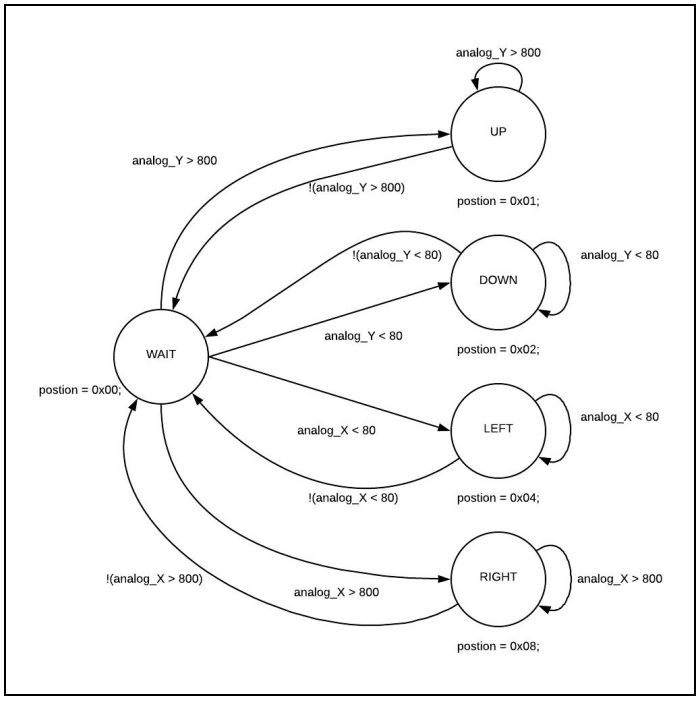


Software

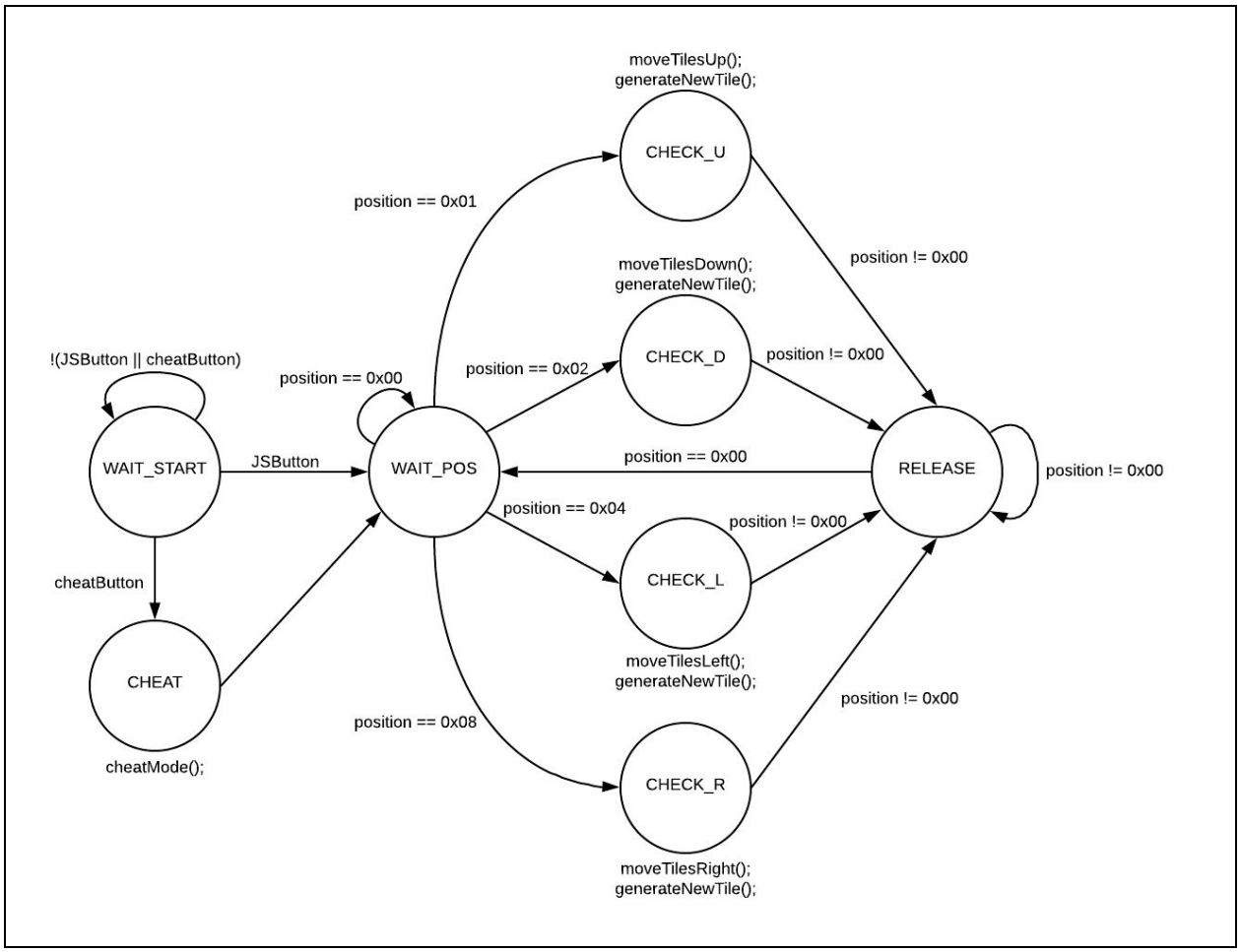
Task Diagram



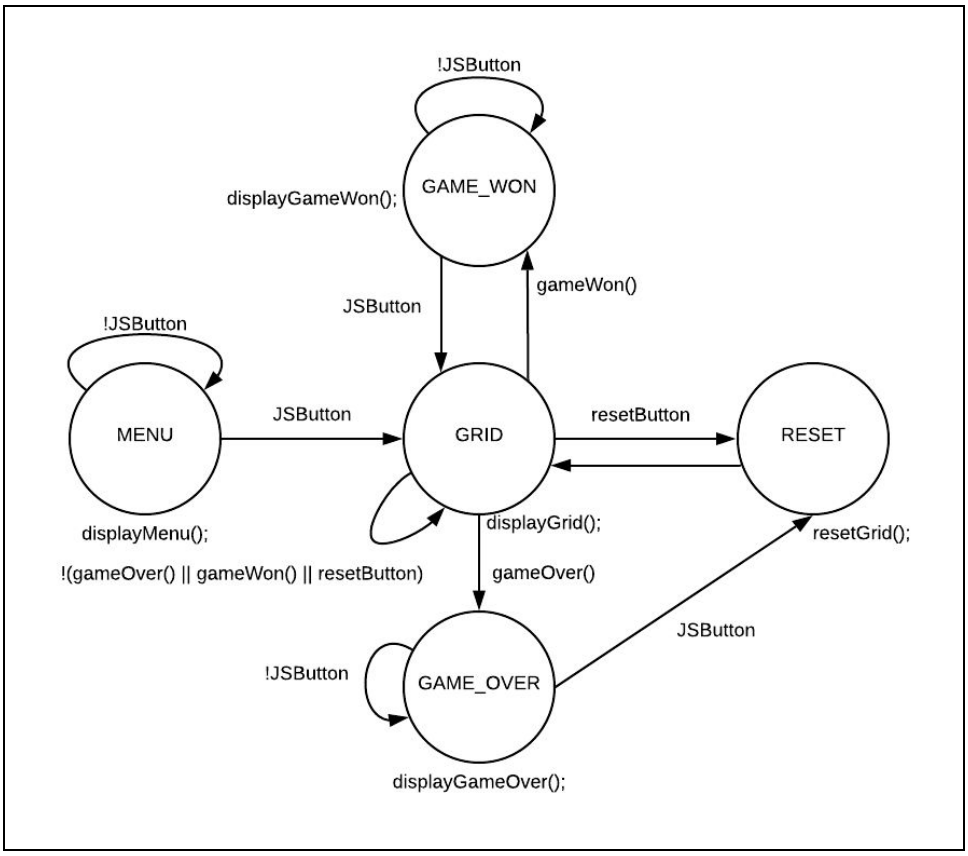
Joystick State Machine



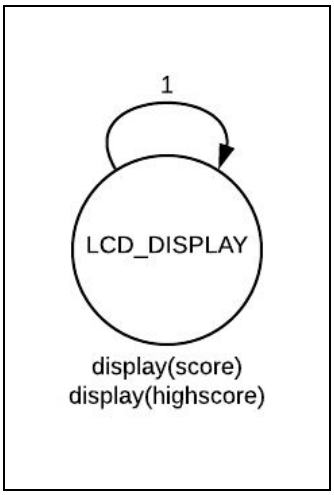
Manage Grid State Machine



Nokia Display State Machine



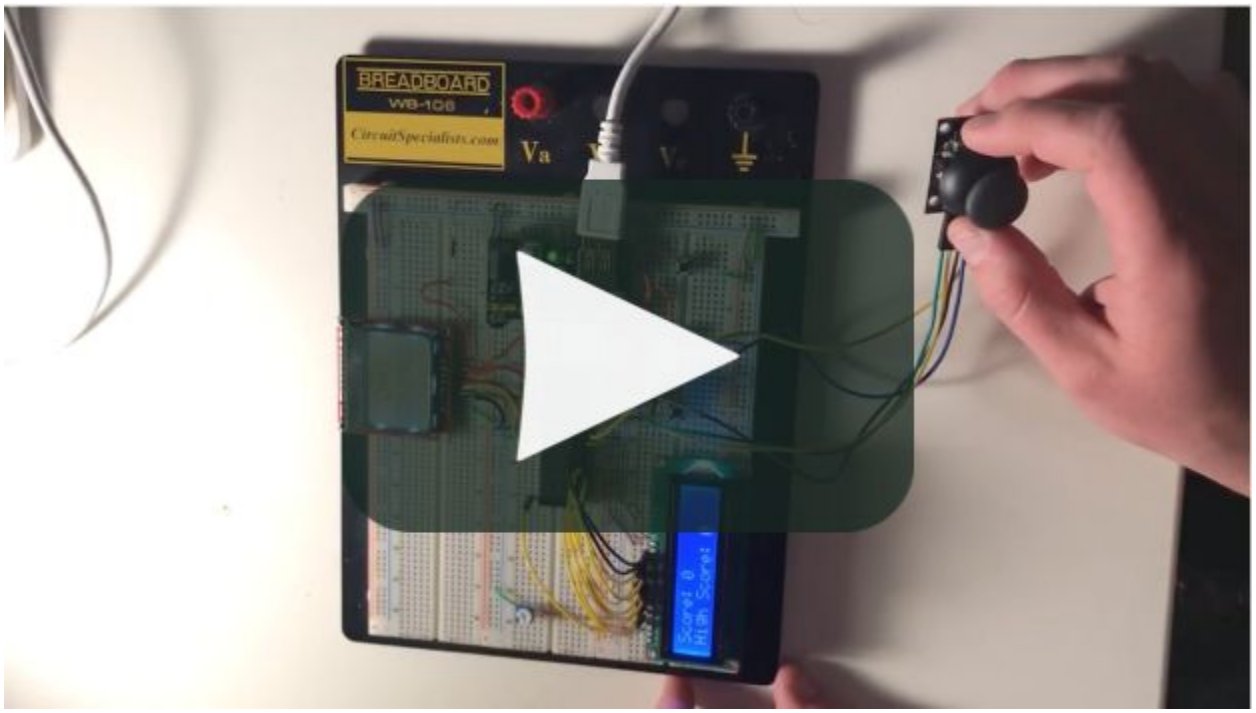
LCD Display State Machine



Complexities

- Integrating Joystick analog input with other tasks
- Programming Nokia 5110 screen to display the game grid and other informational messages
- Game logic required for moving and generating tiles
- Use of EEPROM to save highest score achieved
- Total: 3.5 complexities

Demo Video



https://www.youtube.com/watch?v=dQdFcRQEJZM

Source Files

Click on subheadings to view files

SM_ticks.h

This is the header file for SM_ticks.h. It contains all of the declarations for shared variables, enums, tick functions, and helper functions.

SM_ticks.c

This file contains the implementation of all the tick functions and helper functions.

main.c

This file initializes the I/O, displays, and shared variables. It also declares an array of task structs and initializes the data members. It then executes the program continuously with a simple RTOS that uses a for loop to iterate through the tasks.

Additional Files

Some additional files that were used include nokia5110.h, nokia5110.c, and nokia5110_chars.h. These files can be found at https://github.com/LittleBuster/avr-nokia5110. Other files included scheduler.h, timer.h, io.h, and io.c provided by the CS department.

Future Work

In the future I would like to add more functionality to the game. One example being to actually show the tiles sliding into each other rather than instantly updating the new values. Another addition to the game I'd like to include is storing the grid in EEPROM so the game can continue where it left off when the power is turned back on. While these are a few additions I want to add to 2048, I ultimately want to turn this into a mini GameBoy that can be used to play 3 or so different games on, that stores user accounts and highscores for each user.

What I Learned

- Concurrent Synchronous State Machine design
- Interfacing new hardware and using datasheets to learn how to use them
- Techniques for debugging embedded systems
- Using a simple RTOS task scheduler
- Initializing ADC and using multiple analog input values
- Using EEPROM to store values in nonvolatile memory