Rhythm Game  
Custom Project Report  
Spring 2019  
By Jeremy Taraba

Table of Contents

[**Introduction**](https://docs.google.com/document/d/1-xy6VmtY8k6tF-gSGBFLczn0Hhce3uhBp-F_ejkmoHQ/edit#heading=h.n9o44neiquk5)[**2**](https://docs.google.com/document/d/1-xy6VmtY8k6tF-gSGBFLczn0Hhce3uhBp-F_ejkmoHQ/edit#heading=h.n9o44neiquk5)

[**Hardware**](https://docs.google.com/document/d/1-xy6VmtY8k6tF-gSGBFLczn0Hhce3uhBp-F_ejkmoHQ/edit#heading=h.vqmbsuo5p330)[**2**](https://docs.google.com/document/d/1-xy6VmtY8k6tF-gSGBFLczn0Hhce3uhBp-F_ejkmoHQ/edit#heading=h.vqmbsuo5p330)

[Parts List](https://docs.google.com/document/d/1-xy6VmtY8k6tF-gSGBFLczn0Hhce3uhBp-F_ejkmoHQ/edit#heading=h.xtqh2tzibtrr) [2](https://docs.google.com/document/d/1-xy6VmtY8k6tF-gSGBFLczn0Hhce3uhBp-F_ejkmoHQ/edit#heading=h.xtqh2tzibtrr)

[Pinout](https://docs.google.com/document/d/1-xy6VmtY8k6tF-gSGBFLczn0Hhce3uhBp-F_ejkmoHQ/edit#heading=h.8yptvhjzojq8) [3](https://docs.google.com/document/d/1-xy6VmtY8k6tF-gSGBFLczn0Hhce3uhBp-F_ejkmoHQ/edit#heading=h.8yptvhjzojq8)

[**Software**](https://docs.google.com/document/d/1-xy6VmtY8k6tF-gSGBFLczn0Hhce3uhBp-F_ejkmoHQ/edit#heading=h.w9543qe124on) **4**

[**Complexities**](https://docs.google.com/document/d/1-xy6VmtY8k6tF-gSGBFLczn0Hhce3uhBp-F_ejkmoHQ/edit#heading=h.sn48u4uktu3c) **6**

[Completed Complexities:](https://docs.google.com/document/d/1-xy6VmtY8k6tF-gSGBFLczn0Hhce3uhBp-F_ejkmoHQ/edit#heading=h.tp9dnsse4u9p) 6

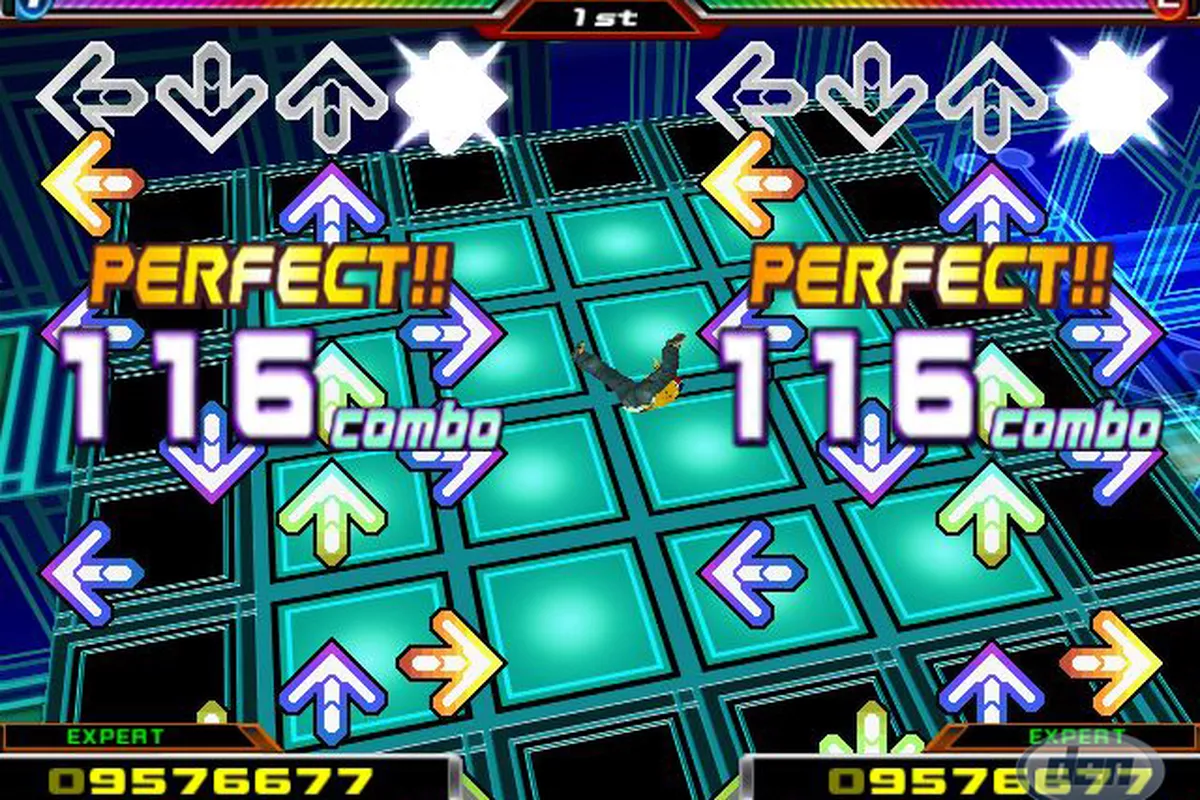
[**Youtube Link**](https://docs.google.com/document/d/1-xy6VmtY8k6tF-gSGBFLczn0Hhce3uhBp-F_ejkmoHQ/edit#heading=h.u9cxbem9510r) **7**

[**Known Bugs and Shortcomings**](https://docs.google.com/document/d/1-xy6VmtY8k6tF-gSGBFLczn0Hhce3uhBp-F_ejkmoHQ/edit#heading=h.grlzpb6vy2cq) **7**

[**Future work**](https://docs.google.com/document/d/1-xy6VmtY8k6tF-gSGBFLczn0Hhce3uhBp-F_ejkmoHQ/edit#heading=h.qbv6f31drpex) **7**

Introduction:

Rhythm games include music playing and some time of visual indicator of when to interact with the beat of the music. Usually there are multiple inputs (directional arrows, colored buttons) which are used to gain a higher score when pressed at the right time. The game ends either when a user misses multiple notes in a row or when the song is over. Also the volume of the music can be adjusted or muted all together.



# Hardware

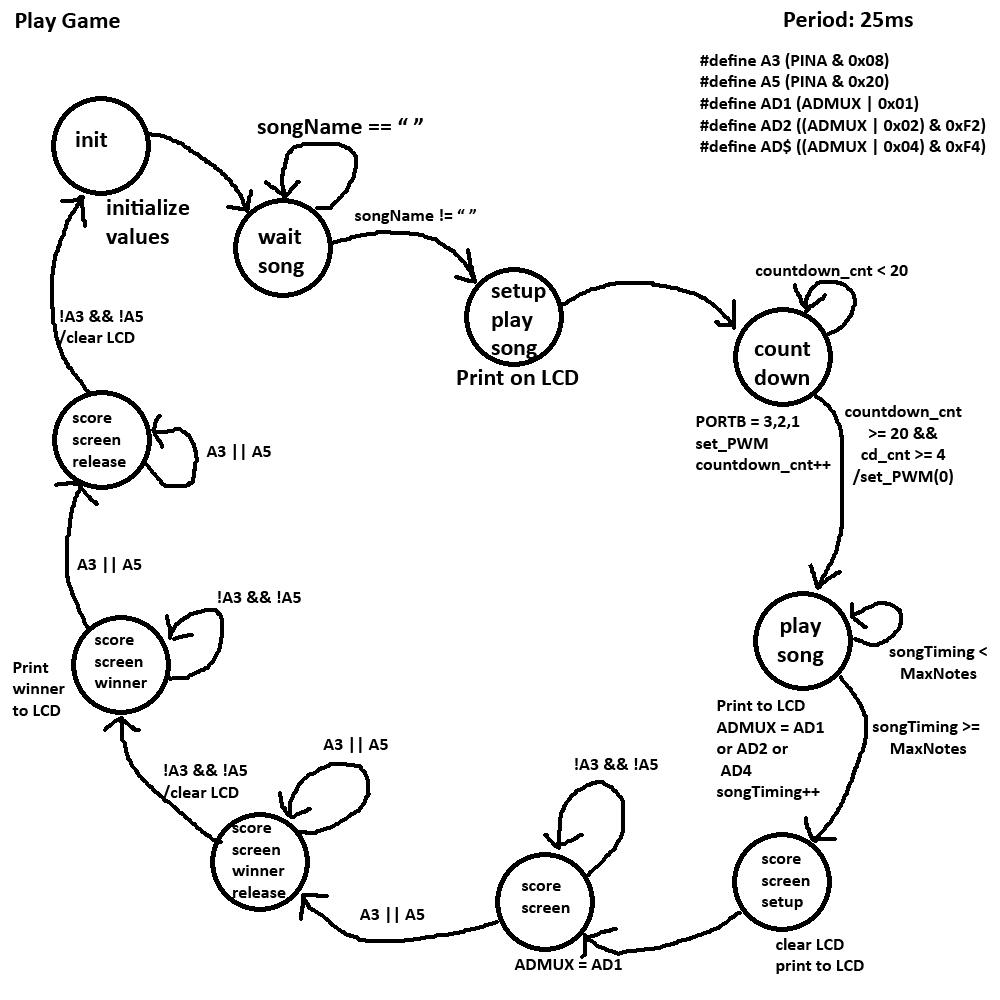
## Parts List

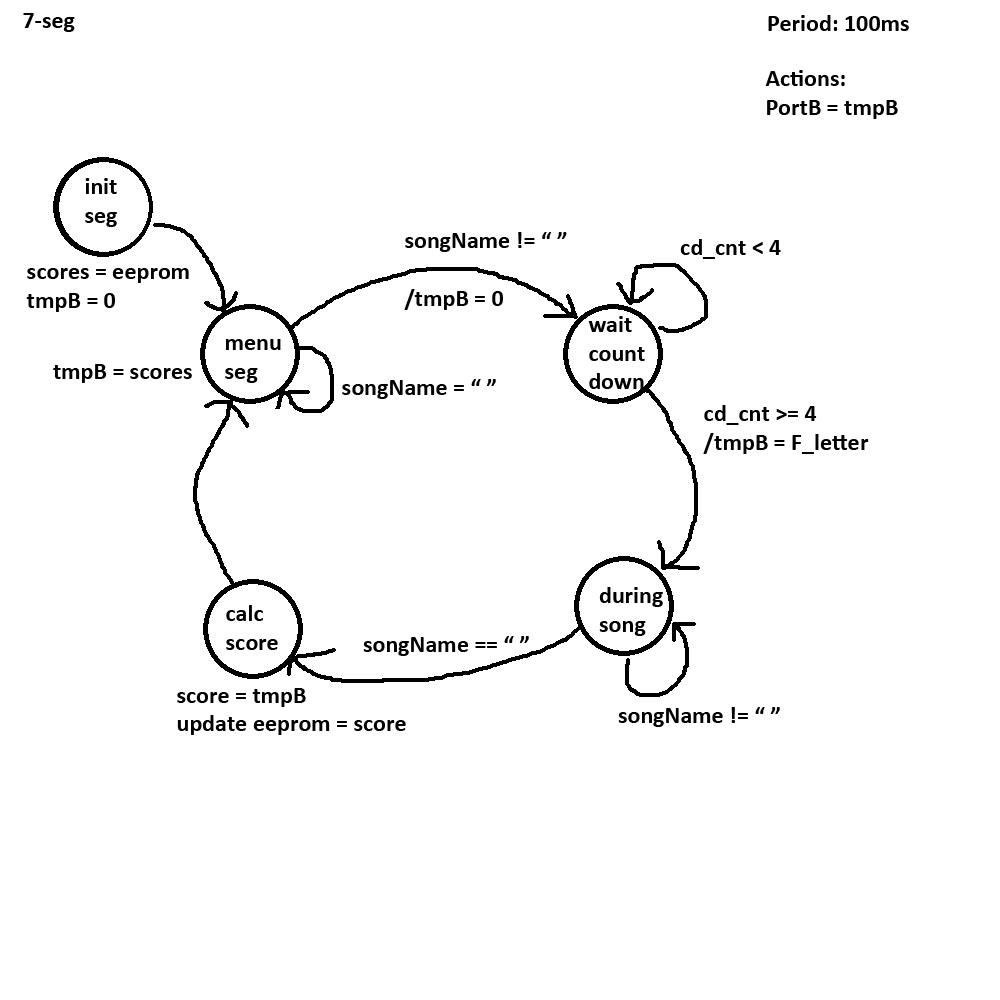
The hardware that was used in this design is listed below. The equipment that was not taught in this course has been bolded.

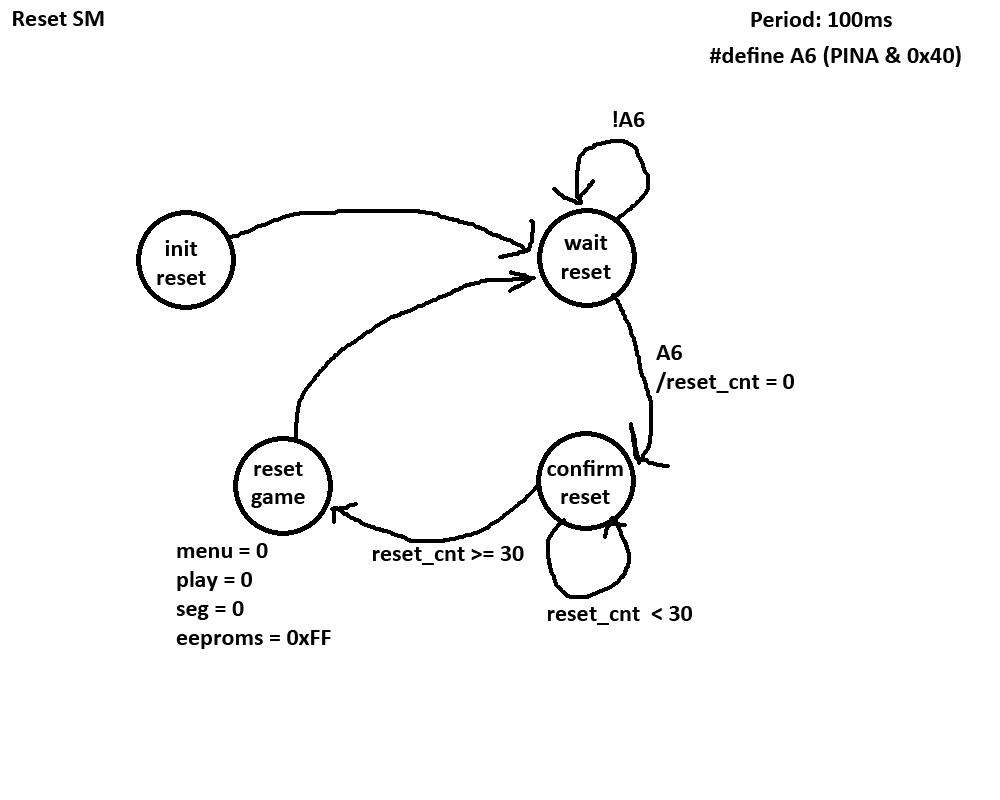
* ATMega1284
* 7-segment LED
* Green and red LEDs
* Buttons
* **2 Joysticks**
* LCD Screen
* Buzzer

## Pinout

# Software







# Complexities

## Completed Complexities:

* Integrating and calibrating a single joystick
* Integrating and calibrating a second joystick to work simultaneous to the first joystick
* Using EEPROM to save the high score (Letter value)
* Creating custom characters on the LCD screen
* Game logic for two players. syncing arrows sliding across the screen with music notes being played by PWM and correct or incorrect LED identifier, with updating score and simultaneous joystick inputs.

# Youtube Link

https://youtu.be/9snbK4j6Rz8

# Known Bugs and Shortcomings

* Joystick direction is saved during play so if you move the joystick to the right and then let it set back to center the game will show you are still holding it to the right and act accordingly. Reason is because when I set it to reset if no direction is pushed the game would flicker between right and no direction if you held it to the right causing some notes to count as incorrect when it flickered to no direction.
* There’s only one set of PWM frequencies and one set of arrows so even though you select a different song it would play the exact same song, however your score for each song is individualized. Example: if you played “Easy song” it would sound the same as “Hard song” but the scores are independent of each other. Reason for this is because I attempted to create a struct to hold all the data for each individual song (frequencies and arrows) but it froze the program for some reason. One solution would be to just add each song as independent variables then we can grab that specific frequency + arrow combo depending on the name of the song chosen.
* When sending arrows down the LCD only one can be sent at a time meaning the songs can’t be too fast, We could figure out a way to fix this using individual timing for the notes

# Future work

* Figure out how to make a struct that can hold all the data for the songs that doesn’t freeze the game every time it’s programmed.
* Add a pause button so you can pause the song if you need a break
* Add a mute button so mute the game if you need to
* Make joysticks show if no direction is displayed
* Maybe second buzzer for more complex songs
* Add a way to send multiple notes down the LCD not just one at a time
* Move the actions in the transition to the actions switch case