CECS 346 Fall 2021 Project # 2

Drag Race

By

Anand Jasti

December 10, 2021

Design a drag race-based system with lights that drivers use to determine when to start the race, and determine the winner based on who starts the race first after the green lights are lit, with a reset button.

CECS 346 Project 2 Report

Introduction

Consider a drag race, there are two drivers who use the 'Christmas Tree' light signal to determine when to accelerate. The 'Christmas Tree' lights are simulated by our LEDs. A track has multiple sensors to determine if a care is in the staged correctly, which is simulated by pressing our lane sensor button. We have one lane sensor per lane (left lane and right lane). In this project the winner is determined by whoever releases their lane sensor button fastest after the green LEDs are lit. I implemented this using a Moore machine, edge interrupts for our buttons (lane sensors), and a SysTick timer interrupt to generate a delay. A reset button is also used to set the state machine to the initial state. The reset button will use level sensitive interrupt.

Operation

I used eight LEDs to represent the 'Christmas Tree' lights. Two buttons to represent the right and the left lane. One more button to represent the reset button. At the start all LEDs will turn on for one second, after one second all LEDs will turn off. After which if both buttons for the left and right lane are pressed the first set of yellow LEDs will light up for 0.5 seconds. Then if the both buttons are still pressed then the second set of yellow LEDs will light up of another 0.5 second. After which if both buttons are still pressed then the green LEDs will light up till one of the two buttons or both buttons are released. When the buttons are released the green LED on the side of the winner will stay lit for 1 second. The winner is determined by whoever releases their button fastest after the green LEDs are lit. After which the system will go back to the 'waiting for the buttons to be pressed' stage, the stage right after all the LEDs turn on and off. The red LEDs will light up for 1 second respectively if one or both buttons are released before the green LEDs are lit. When the reset button is pressed the system will go back to the initialize stage, the stage where all the LEDs turn on and off.

Link to Demonstration video:
https://drive.google.com/file/d/1McMNn8gbZcLFU151Ld7eRVO60FDPDq_x/view?usp=sharing

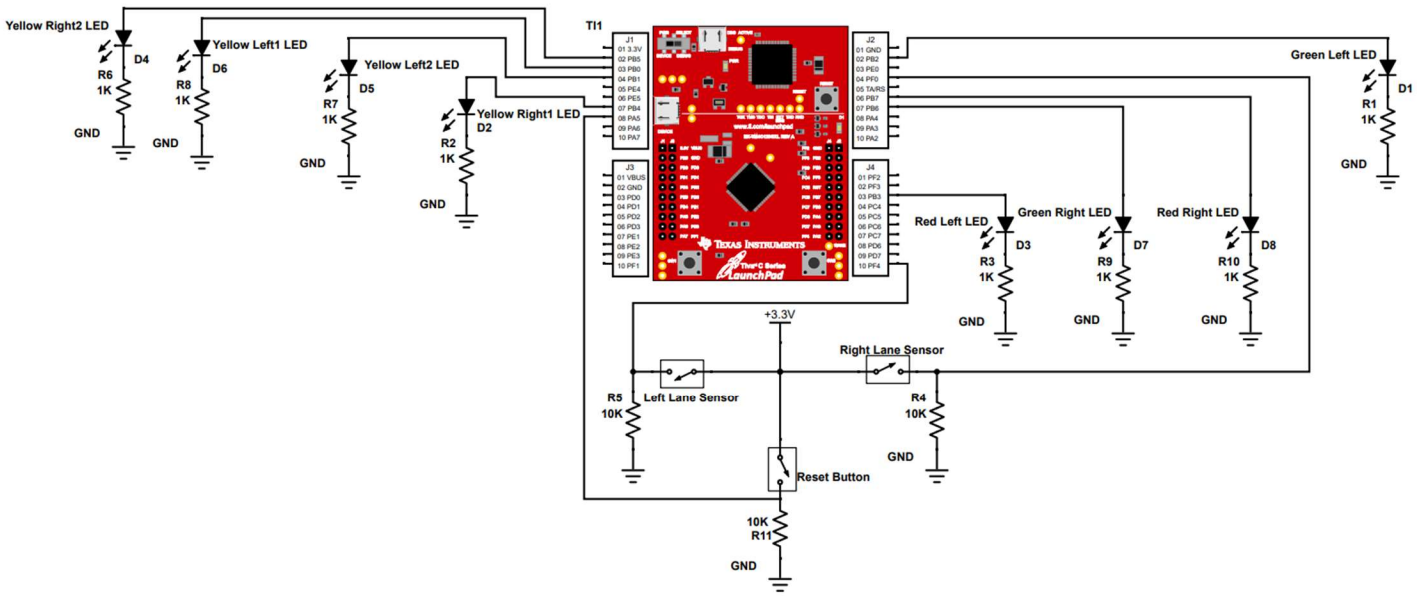
Theory

This project uses ARM Cortex TM4C123GH6PM Microcontroller, more specifically I used three of the six General-Purpose I/O ports (PB,PF,PA). In port B, I used all eight pins for the 'Christmas Tree' LEDs. In port A, I used pin 5 for the reset button. In port F, I

used the onboard buttons pin 0 and pin 4 for the left and right lane sensors (more specifically defined in Hardware design). In this project I used a Moore Finite State Machine to implement system requirements for this drag race based system. I also used SysTick Timer Interrupt provided by the microcontroller to generate delays for this drag race based system. For the left and right lane buttons I used edge triggered interrupts, and for the reset button I used level sensitive interrupts (more specifically defined in Software design).

Hardware design

Schematic:



Outputs:

Yellow Left1 LED	PB0
Yellow Left2 LED	PB1
Green Left LED	PB2
Red Left LED	PB3
Yellow Right1 LED	PB4
Yellow Right2 LED	PB5
Green Right LED	PB6
Red Right LED	PB7

Inputs:

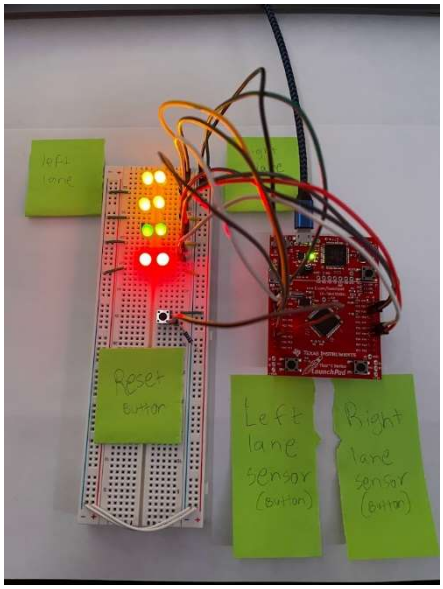
Left Sensor	**PF4**
Right Sensor	**PF0**
Reset button	**PA5**

The ‘Christmas Tree’ Lights are simulated by these eight LED outputs on port B pins7-0. All LEDs are using positive logic.

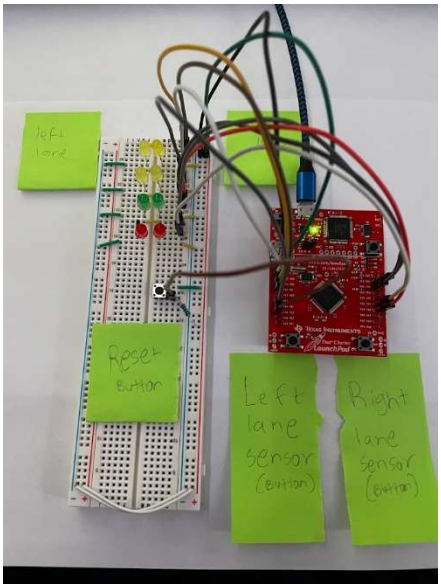
The left and right lane sensors are simulated by the left and right button inputs using negative logic on port F pin 4 and pin 0, using on board buttons. The reset button is an input using positive logic on port A pin 5.

Pictures of Hardware System:

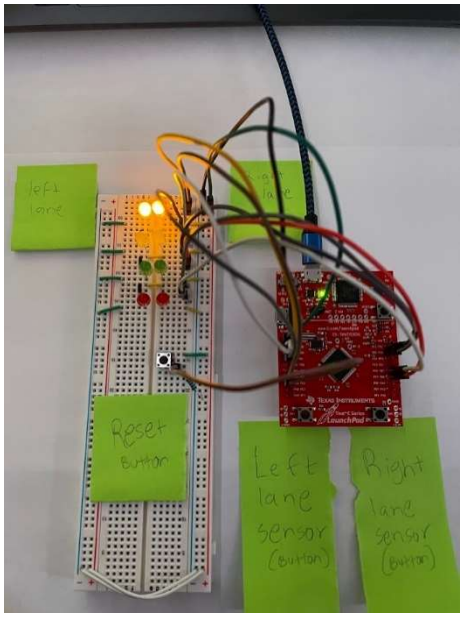
Initialize State:



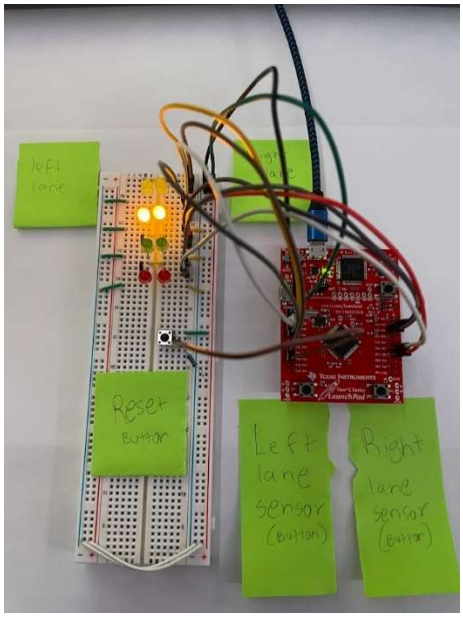
WaitForStaging State:



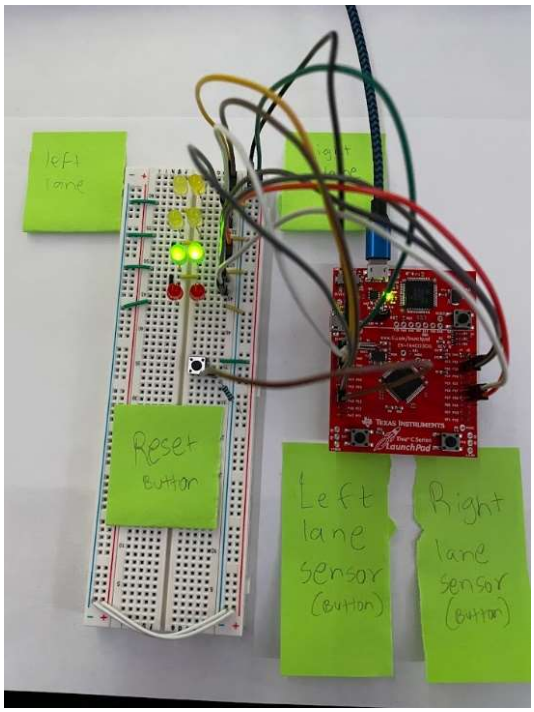
CountdownY1 State:



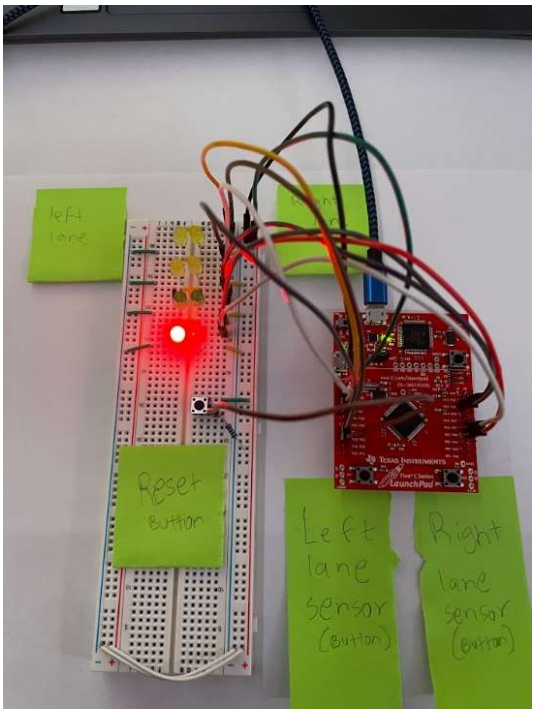
CountdownY2 State:



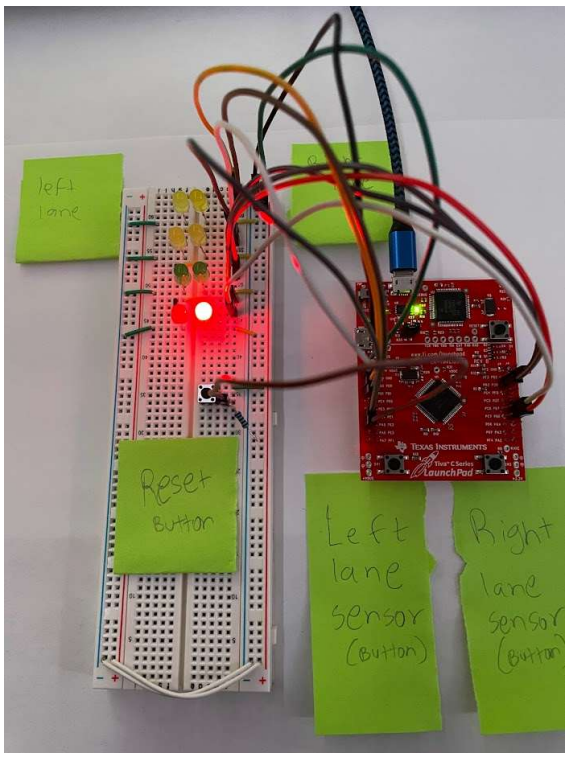
Go State:



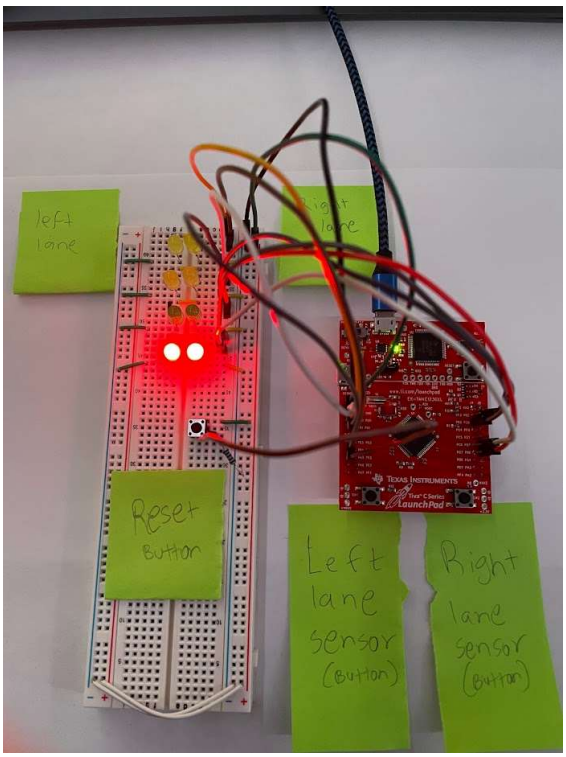
FalseStartLeft State:



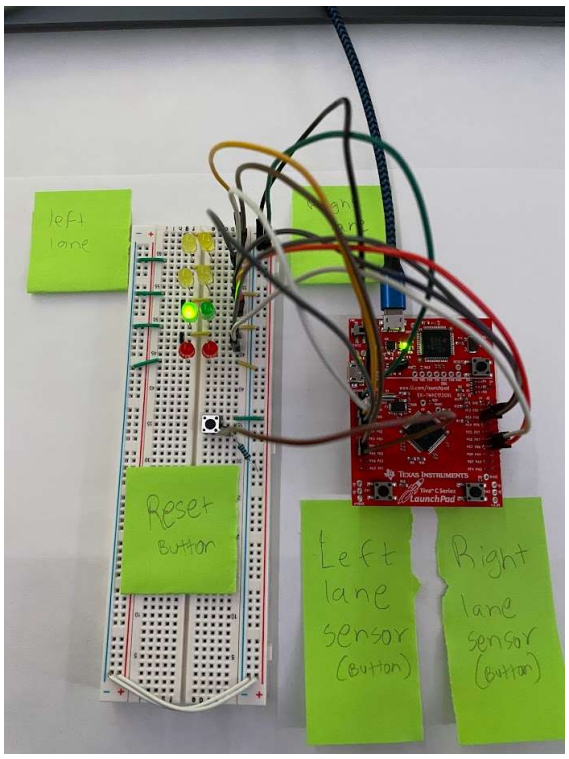
FalseStartRight State:



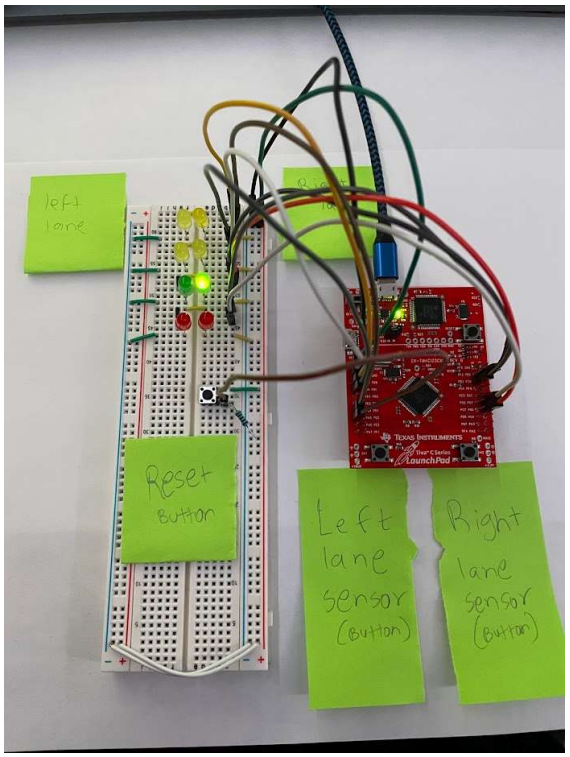
FalseStartBoth State:



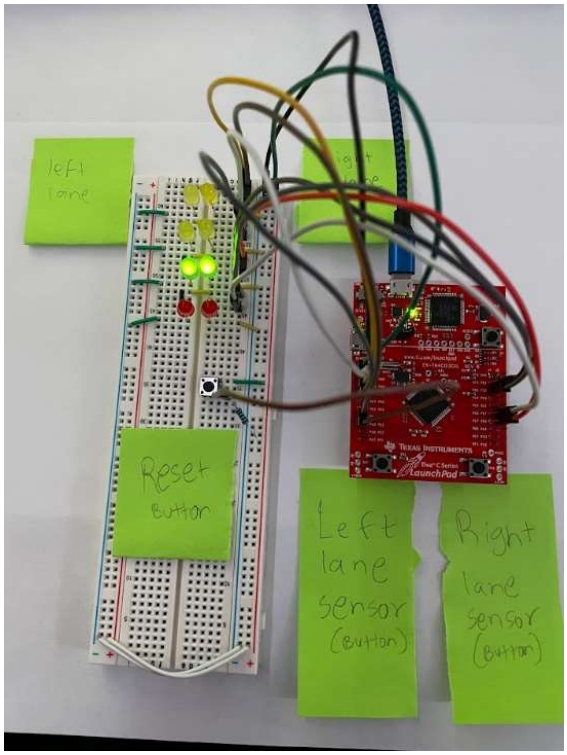
WinLeft State:



WinRight State:



WinBothState:



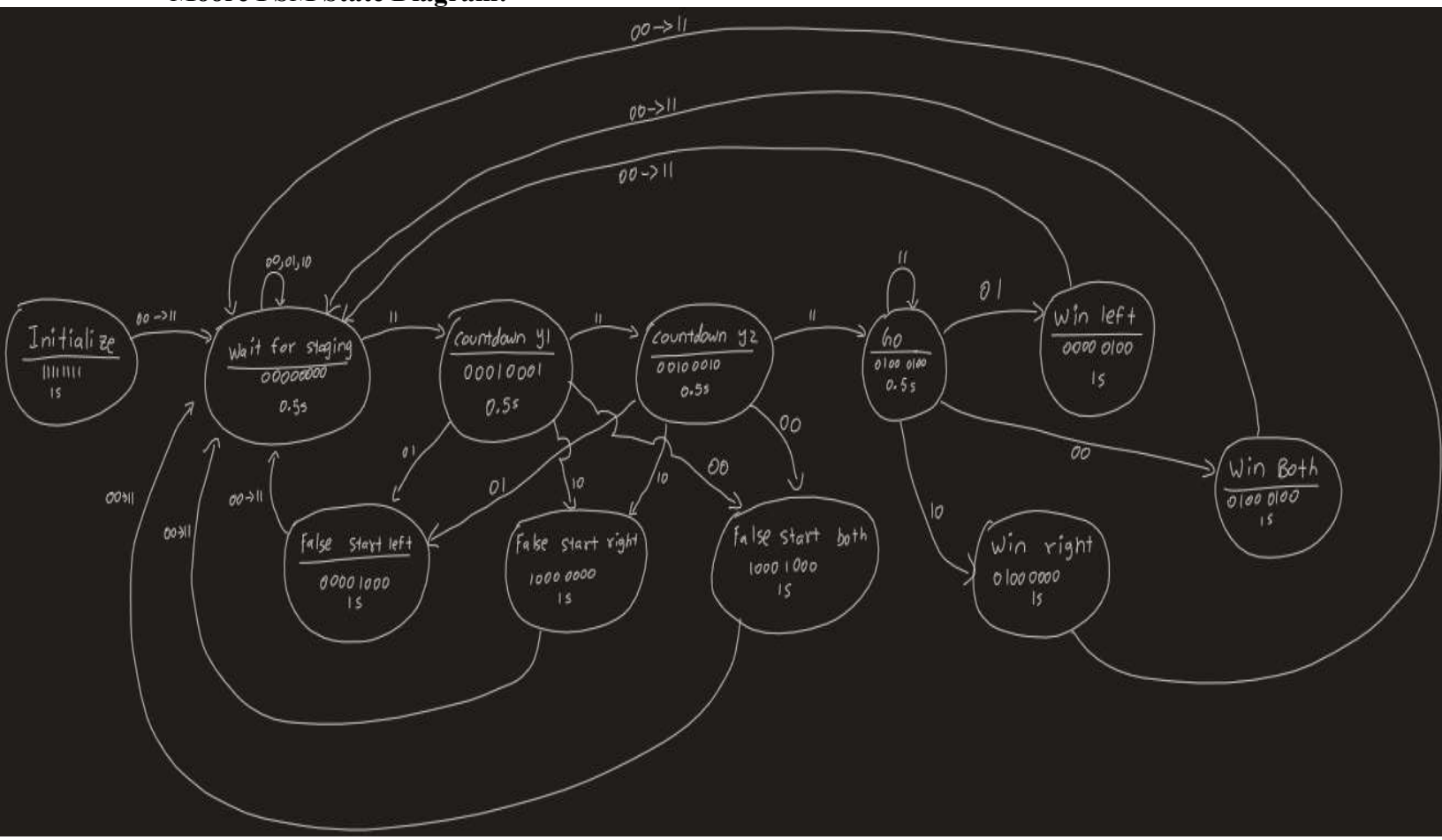
Software design

Using a Moore Finite State Machine to implement this drag race based system, the system starts at state Initialize which turns on all LEDs. After which the state machine unconditionally transitions into WaitForStaging state where no LEDs are lit. In this state, the state machine checks every 0.5 seconds if both left and right lanes are staged, in other words the state machine checks if both left and right buttons are pressed every 0.5 seconds. If both lanes are staged the system transition to CountdownY1, otherwise the state machine will stay in the WaitForStaging state. In the CountdownY1 state only the first set of yellow LEDs are lit for 0.5 seconds, if the left lane or the right lane buttons or both buttons are released the state machine transitions to the FalseStart (Left/Right/Both) state respectively based on which button was released first. If both lane buttons are still pressed the state machine tranisitions to the next state CountdownY2. In CountdownY2 only the second set of yellow LEDs are lit for 0.5 seconds, if the left lane or the right lane buttons or both buttons are released the state machine tranisitions to the FalseStart (Left/Right/Both) state respectively based on which button was released first. If both lane buttons are still press the state machine tranisitions to the next state Go. In this Go state both the green LEDs are lit, then the state checks every 0.5s if both the lane buttons are still pressed. If both lane buttons are still pressed the state machine stays in the Go state. If the left lane or the right lane buttons or both buttons are released the state machine transitions to Win (Left/Right/Both) state respectively based on which button was released first. In the FalseStart (Left/Right/Both) state only the left or right or both red LED(s) are lit respectively based on what state the state machine is in. After 1 second the state machine unconditionally transitions into the WaitForStaging state. In the Win (Left/Right/Both) state only the left or right or both green LED(s) are lit respectively based on what state the state machine is in. After 1 second the state machine unconditionally transitions in the WaitForStaging state. When the reset button is pressed, the SysTick timer is stopped and the state machine transitions into the Initialize state.

Moore FSM State Table:

Current State	Time(0.5s)	Output(PB7-PB0)	Inputs(PF4,PF0):00	01	10	11
Initialize	2	11111111	WaitForStaging	WaitForStaging	WaitForStaging	WaitForStaging
WaitForStaging	1	00000000	WaitForStaging	WaitForStaging	WaitForStaging	CountdownY1
CountdownY1	1	00010001	FalseStartBoth	FalseStartLeft	FalseStartRight	CountdownY2
CountdownY2	1	00100010	FalseStartBoth	FalseStartLeft	FalseStartRight	Go
Go	1	01000100	WinBoth	WinLeft	WinRight	Go
FalseStartLeft	2	00001000	WaitForStaging	WaitForStaging	WaitForStaging	WaitForStaging
FalseStartRight	2	10000000	WaitForStaging	WaitForStaging	WaitForStaging	WaitForStaging
FalseStartBoth	2	10001000	WaitForStaging	WaitForStaging	WaitForStaging	WaitForStaging
WinLeft	2	00000100	WaitForStaging	WaitForStaging	WaitForStaging	WaitForStaging
WinRight	2	01000000	WaitForStaging	WaitForStaging	WaitForStaging	WaitForStaging
WinBoth	2	01000100	WaitForStaging	WaitForStaging	WaitForStaging	WaitForStaging

Moore FSM State Diagram:



Conclusion

The Drag Race project was very fun to implement. It took a while for me to figure out the how to implement the finite state machine into my software, but other than that everything else was straight forward. It was hard not being able to test the code in Keil simulator, so knowing that I couldn't test my code beforehand, I instead chose to test my hardware beforehand so if something wasn't working as intended I knew that the problem would be in the software. During this project I learned more about how to implement and use a Moore Finite State Machine into software, I also learned how to debug my software code on my hardware using Keil debugging tool. Overall this was a fun project to do even though it took a long time to finish.