CECS 346 Fall 2021 Project 2

Drag Race

By

Abhishek Jasti

December 10,2021

Brief description: This Project helped me understand the concepts of interfacing to switches and LEDs, while using a Moore finite state machine and using interrupts. Performing explicit measurements on the circuits to verify they are operational.

CECS 346 Project 2 Report

INTRODUCTION: Given a Launchpad, 3 buttons in total: 2 buttons on the launchpad for left lane sensor and right lane sensor and 1 reset button on breadboard, using the internal resistors for the 2 buttons on launchpad and 1 10kohm resistor for the reset button on the breadboard, eight color LEDs: 2 red, 2 green, 4 yellow and 8 resistors for the LEDs.

OPERATION: In the beginning the traffic lights should start with no LEDs on. The sensors used are listed below with a brief detailed description.
SENSORS:

1) RESET SENSOR (Button) – PE0 being the reset sensor is used to reset the Christmas lights.
2) LEFT LANE SENSOR (Button) – PF4 being the left lane sensor determining what state the left lane lights (PB7 - yellow 1 left, PB6 – yellow 2 left, PB5 – green left, PB4 – red left) are in.
3) RIGHT LANE SENSOR (Button) – PF0 being the right lane sensor determining what state the right lane lights (PB3 - yellow 1 right, PB2 – yellow 2 right, PB1 – green right, PB0 – red right) are in.

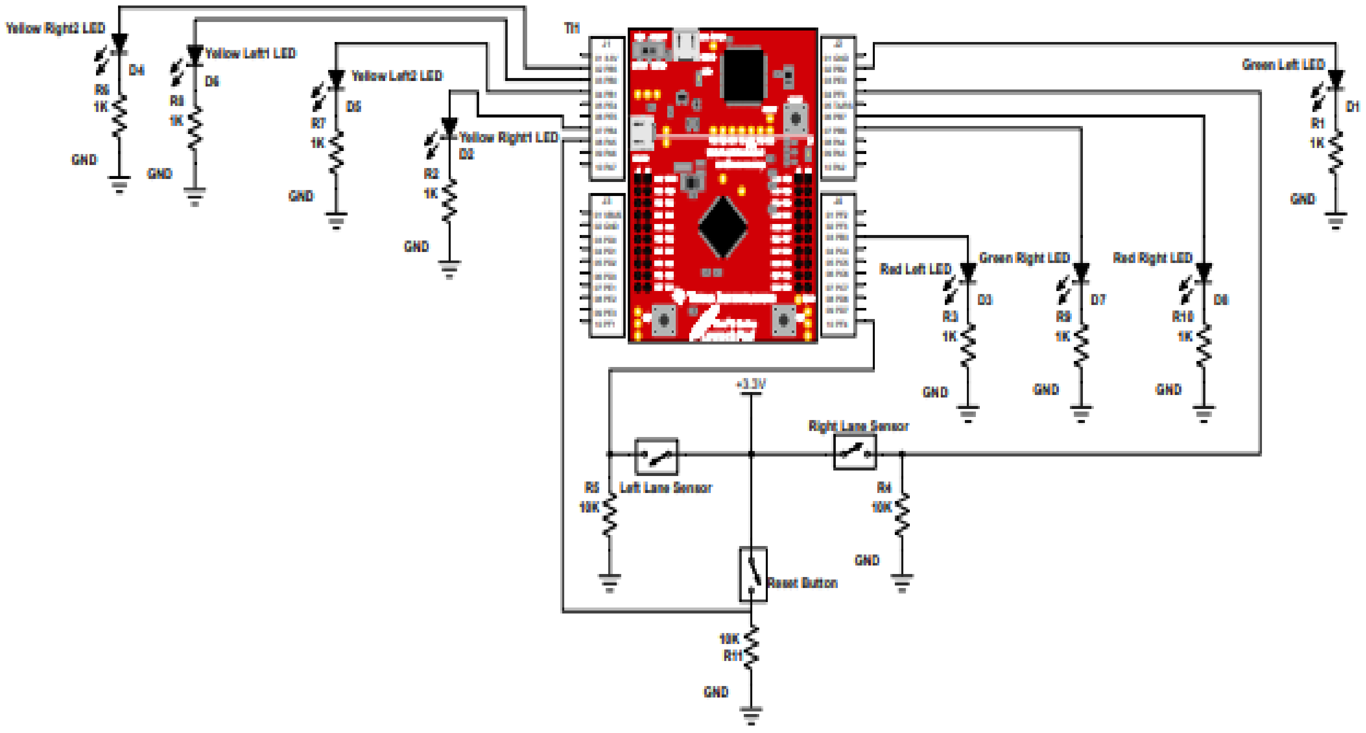
For example: When the Christmas lights are turned on with both lane sensors pressed the Christmas lights will cycle through the states, when the left lane sensor is released before the lights reach green, the state will go to false start to the left red light. This is vice versa for the right lane sensor.

In the Video below are the test cases on what the 3 sensors (LEFT Lane sensor, RIGHT Lane sensor, and RESET sensor) should do when different sensors are turned on and released.

Demonstration Video Link:

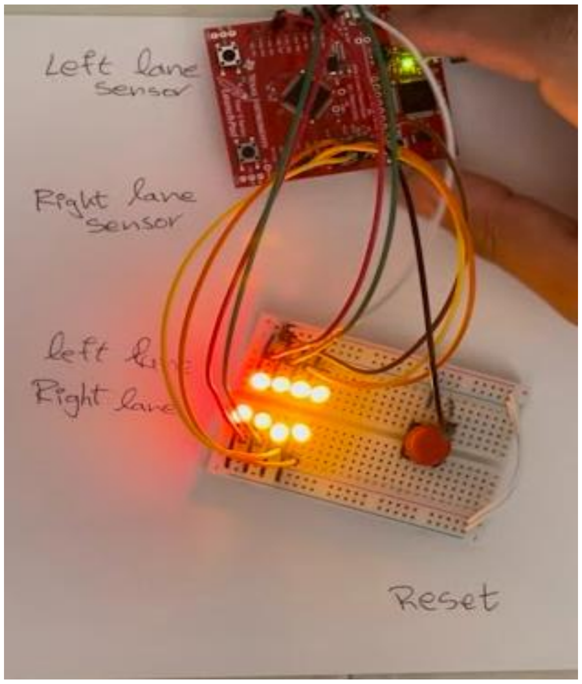
https://drive.google.com/file/d/1qeJRdPHLe6n-zNGKAdF82Exv-C-vbMB-/view?usp=sharing

HARDWARE DESIGN:

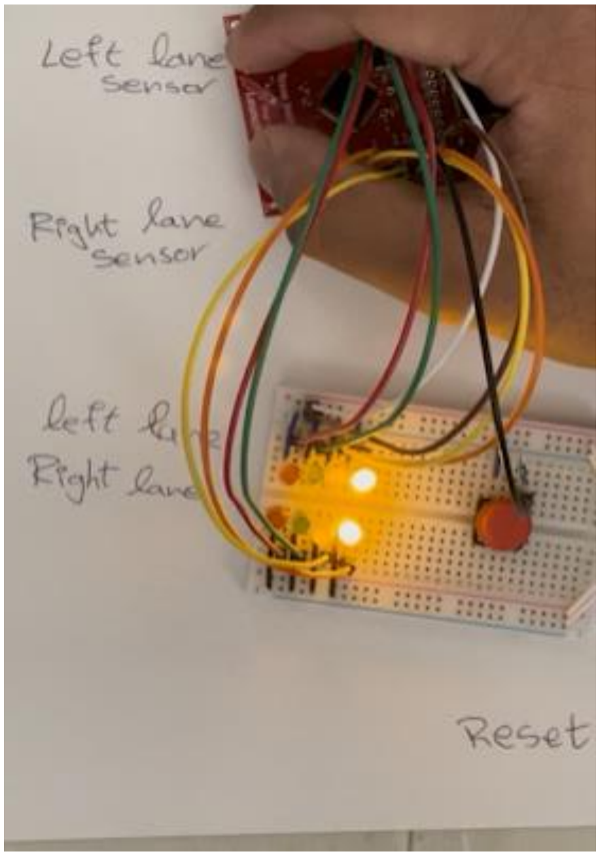


YELLOW 1 LEFT LED	PB7
YELLOW 2 LEFT LED	PB6
GREEN LEFT LED	PB5
RED LEFT LED	PB4
YELLOW 1 RIGHT LED	PB3
YELLOW 2 RIGHT LED	PB2
GREEN RIGHT LED	PB1
RED RIGHT LED	PB0
LEFT LANE SENSOR	PF4
RIGHT LANE SENSOR	PF0
RESET SENSOR	PE0

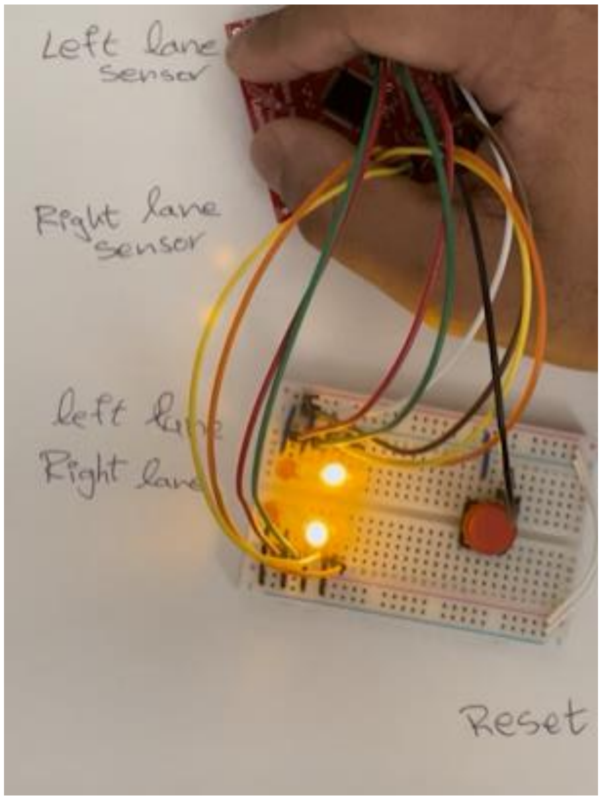
Initialize state



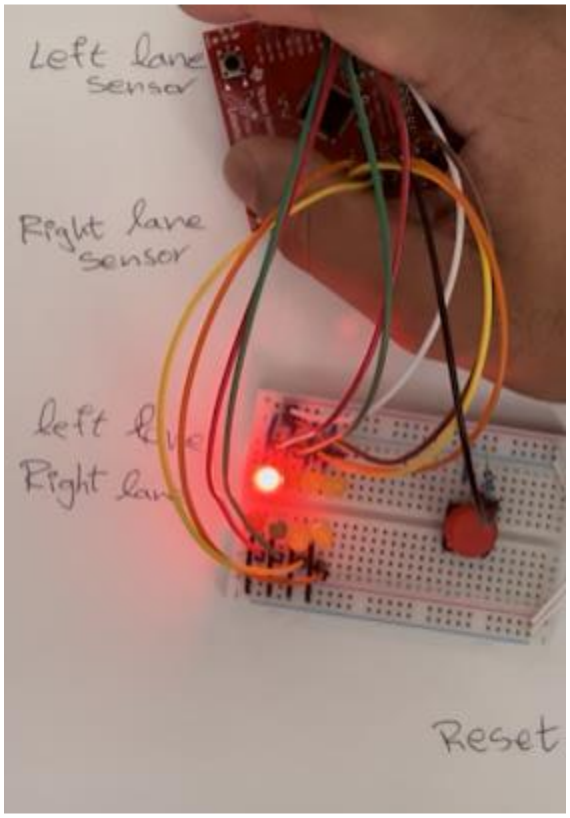
Countdown y1 state



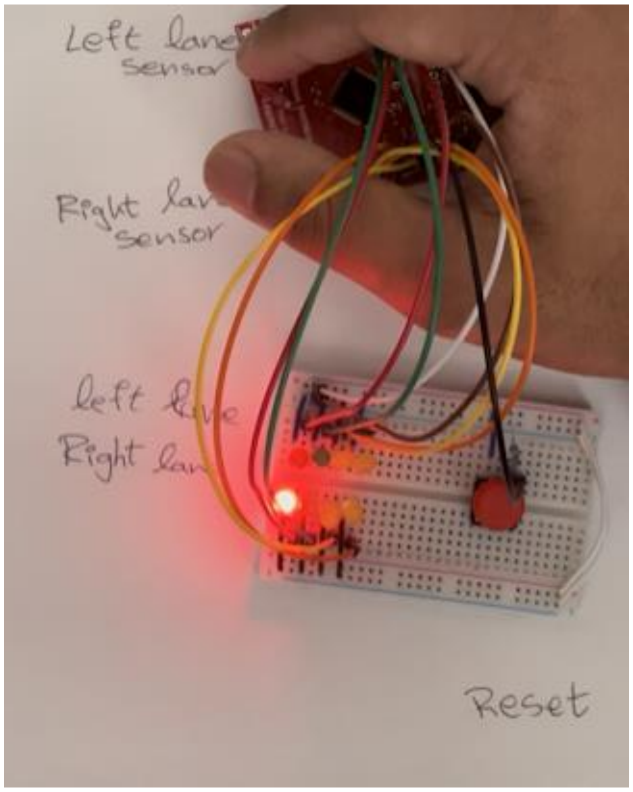
Countdown y2



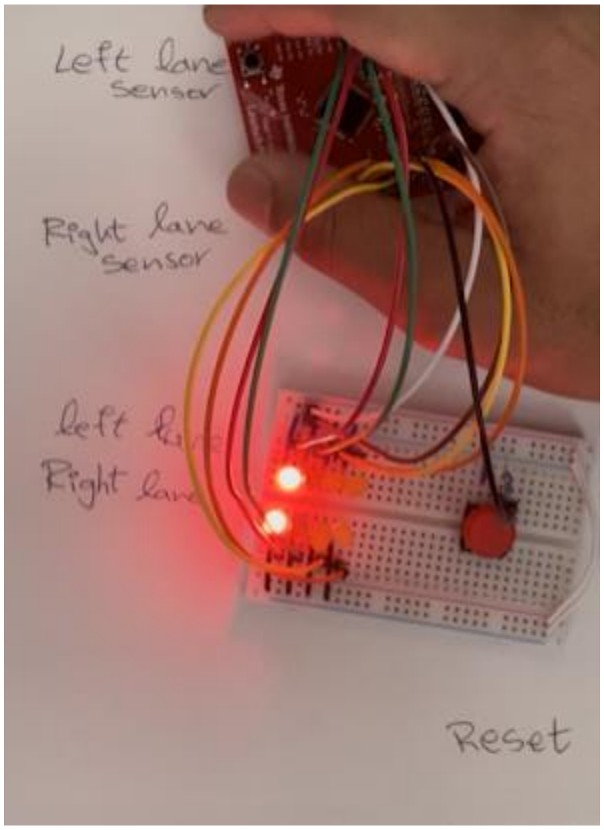
False start left



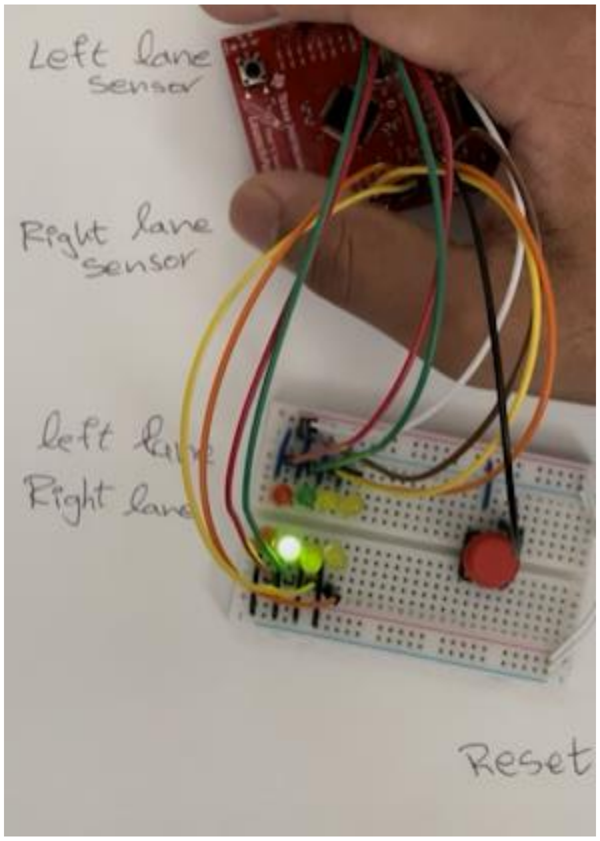
False start right



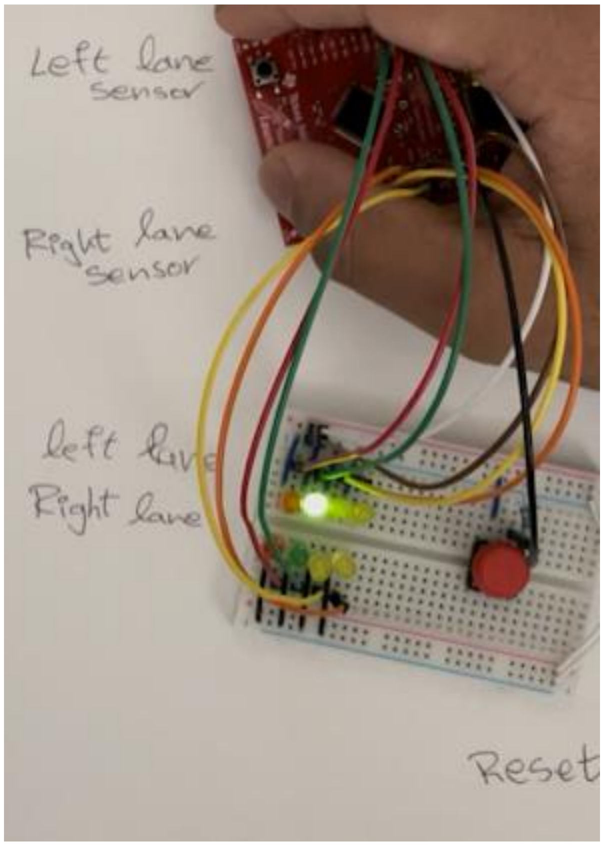
False start both



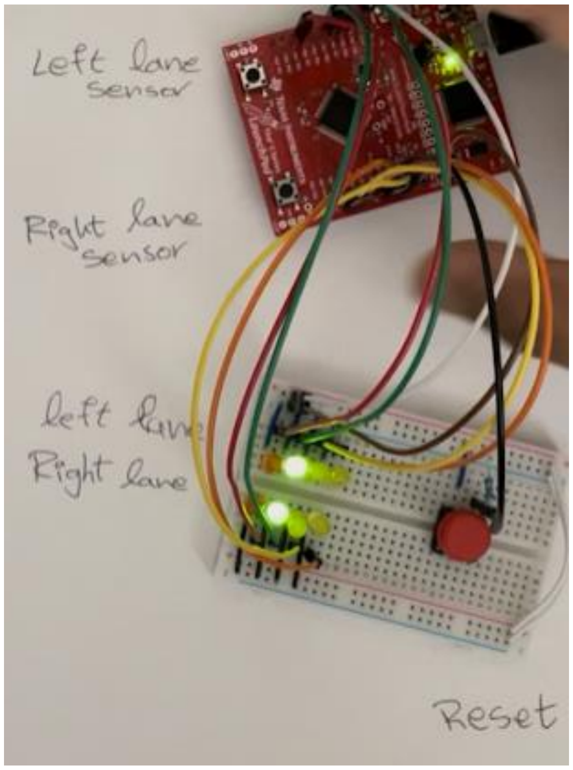
Win right



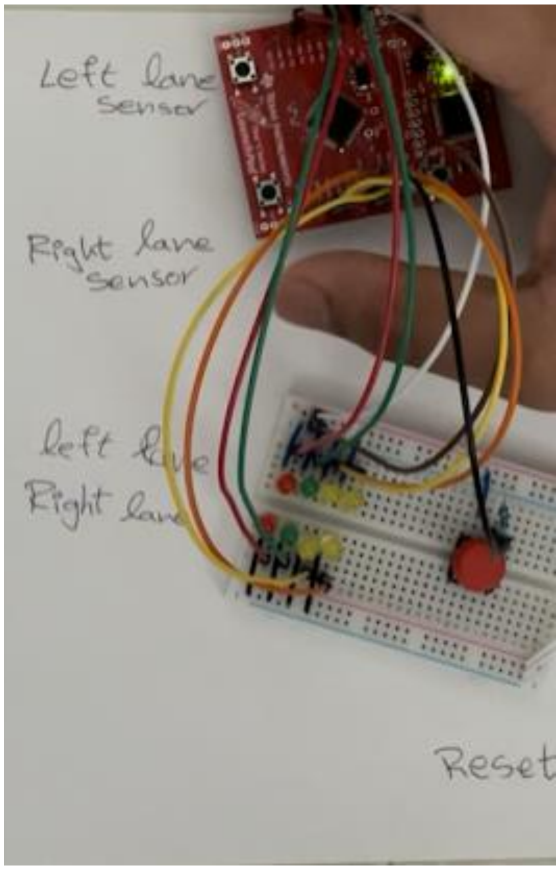
Win left



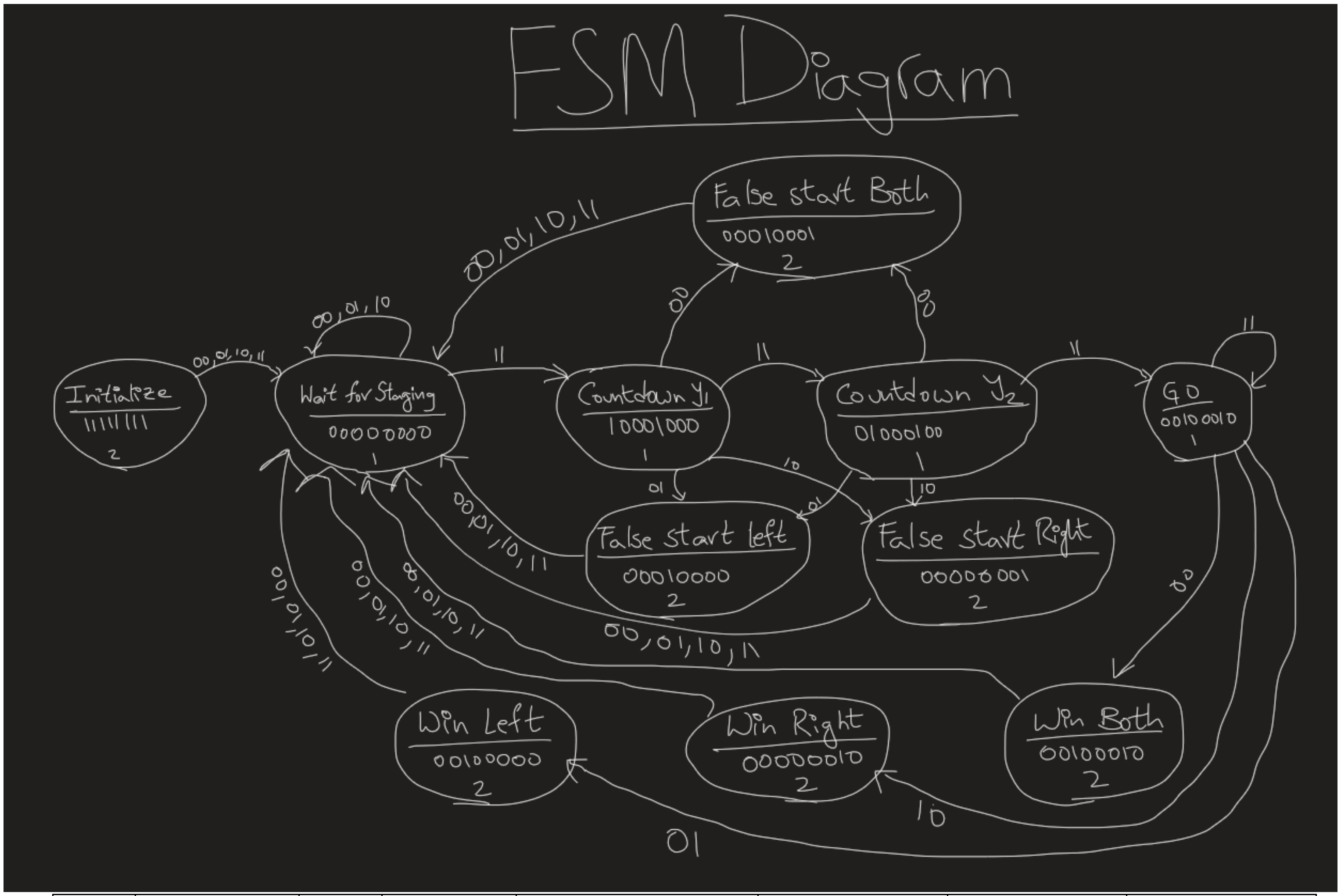
Win both



Wait for staging

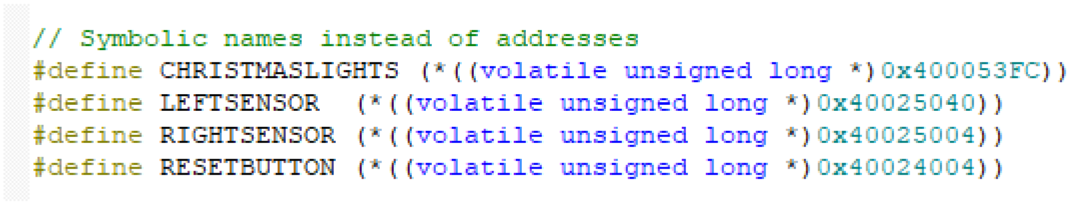


SOFTWARE DESIGN:



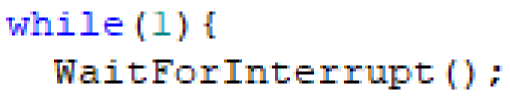
State Index	Current State	Time (0.5s)	OUTPUTS (PORTB)	INPUT 00 (left & right Pressed)	INPUT 01 (Right Released)	INPUT 10 (Left Released)	INPUT 11 (Both Released)
0	Initialize	2	0xFF	waitforstaging	waitforstaging	waitforstaging	waitforstaging
1	waitforstaging	1	0x00	waitforstaging	waitforstaging	waitforstaging	countdowny1
2	countdowny1	1	0x88	falsestartboth	falsestartleft	falsestartright	countdowny2
3	countdowny2	1	0x44	falsestartboth	falsestartleft	falsestartright	go
4	go	1	0x22	winboth	winleft	winright	go
5	falsestartleft	2	0x10	waitforstaging	waitforstaging	waitforstaging	waitforstaging
6	falsestartright	2	0x01	waitforstaging	waitforstaging	waitforstaging	waitforstaging
7	falsestartboth	2	0x11	waitforstaging	waitforstaging	waitforstaging	waitforstaging
8	winleft	2	0x20	waitforstaging	waitforstaging	waitforstaging	waitforstaging
9	winright	2	0x02	waitforstaging	waitforstaging	waitforstaging	waitforstaging
10	winboth	2	0x22	waitforstaging	waitforstaging	waitforstaging	waitforstaging

Defining CHRISTMASLIGHTS with PORTB with bits 0, 1, 2, 3, 4, 5, 6, and 7
Defining RIGHT AND LEFT SENSORS with PORTF with bits 0, and 4
Defining REST SENSOR with PORTE with bit 0.



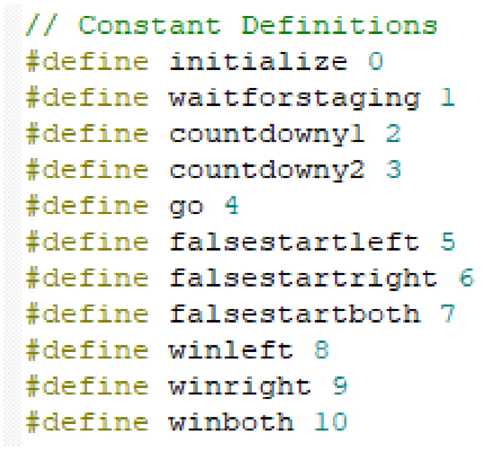
```
// Symbolic names instead of addresses
#define CHRISTMASLIGHTS (*((volatile unsigned long *)0x400053FC))
#define LEFTSENSOR   (*((volatile unsigned long *)0x40025040))
#define RIGHTSENSOR  (*((volatile unsigned long *)0x40025004))
#define RESETBUTTON  (*((volatile unsigned long *)0x40024004))
```

In the while loop:
Waiting for interrupt



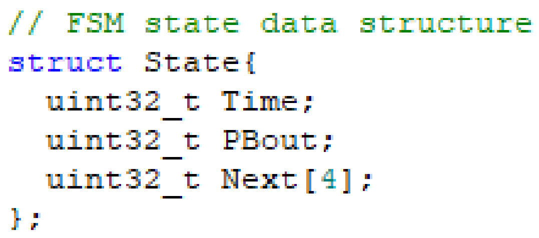
```
while(1){
  WaitForInterrupt();
```

Defining constants for FSM

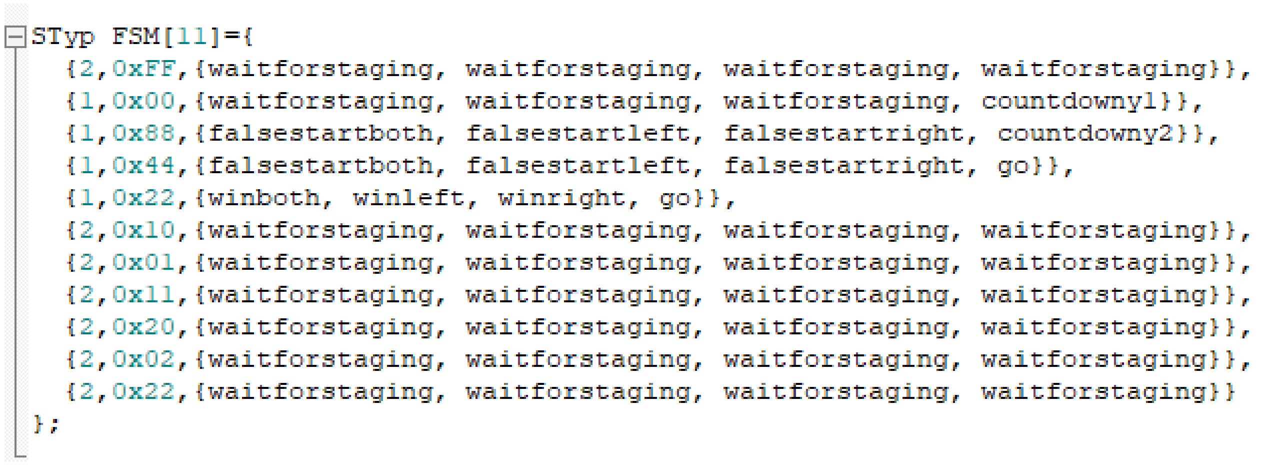


```
// Constant Definitions
#define initialize 0
#define waitforstaging 1
#define countdowny1 2
#define countdowny2 3
#define go 4
#define falsestartleft 5
#define falsestartright 6
#define falsestartboth 7
#define winleft 8
#define winright 9
#define winboth 10
```

Structuring the Finite states

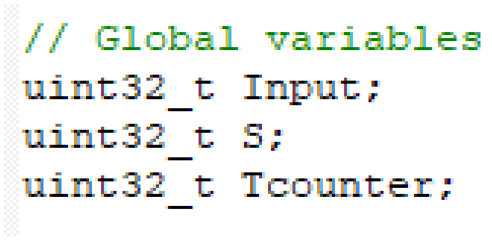


```
// FSM state data structure
struct State{
  uint32_t Time;
  uint32_t PBout;
  uint32_t Next[4];
};
```



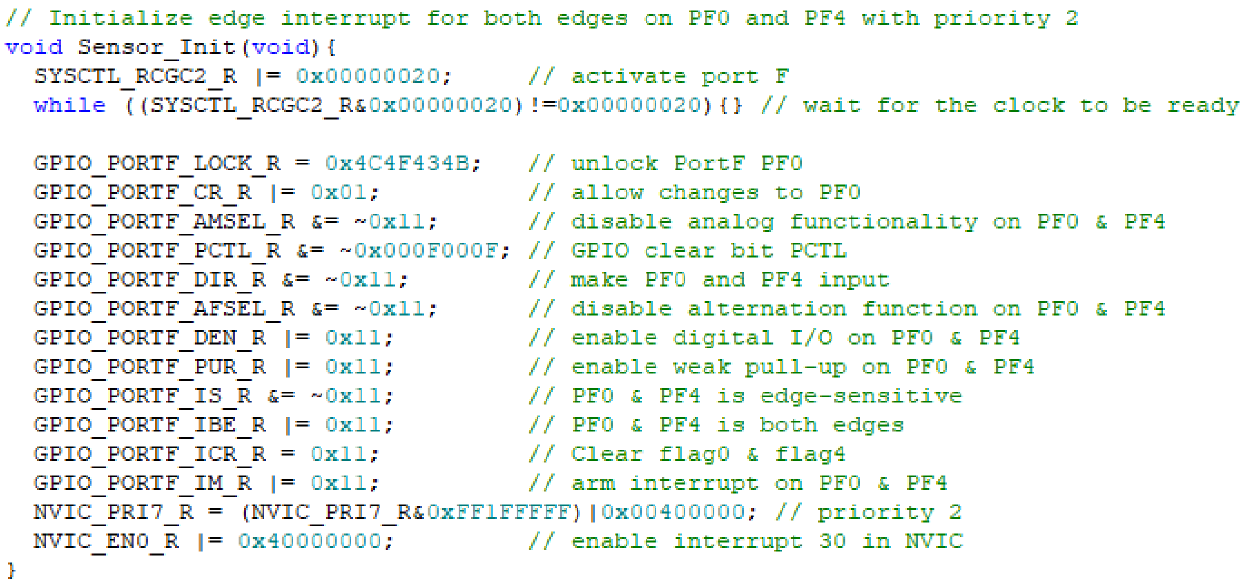
```
STyp FSM[11]={
  {2,0xFF,{waitforstaging, waitforstaging, waitforstaging, waitforstaging}},
  {1,0x00,{waitforstaging, waitforstaging, waitforstaging, countdowny1}},
  {1,0x88,{falsestartboth, falsestartleft, falsestartright, countdowny2}},
  {1,0x44,{falsestartboth, falsestartleft, falsestartright, go}},
  {1,0x22,{winboth, winleft, winright, go}},
  {2,0x10,{waitforstaging, waitforstaging, waitforstaging, waitforstaging}},
  {2,0x01,{waitforstaging, waitforstaging, waitforstaging, waitforstaging}},
  {2,0x11,{waitforstaging, waitforstaging, waitforstaging, waitforstaging}},
  {2,0x20,{waitforstaging, waitforstaging, waitforstaging, waitforstaging}},
  {2,0x02,{waitforstaging, waitforstaging, waitforstaging, waitforstaging}},
  {2,0x22,{waitforstaging, waitforstaging, waitforstaging, waitforstaging}}
};
```

Initializing Global Variables



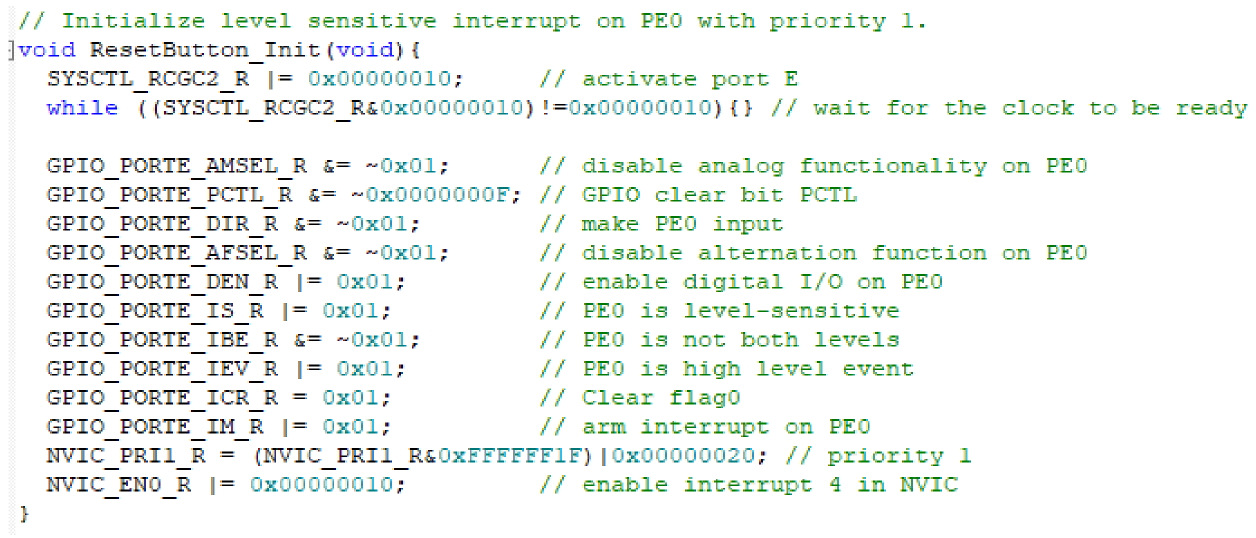
```
// Global variables
uint32_t Input;
uint32_t S;
uint32_t Tcounter;
```

Initializing edge interrupt for both edges on PF0 and PF4 with priority 2



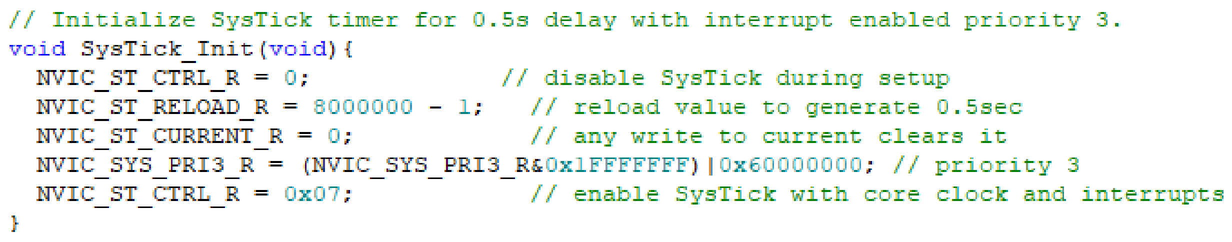
```
// Initialize edge interrupt for both edges on PF0 and PF4 with priority 2
void Sensor_Init(void){
  SYSCTL_RCGC2_R |= 0x00000020;      // activate port F
  while ((SYSCTL_RCGC2_R&0x00000020)!=0x00000020){} // wait for the clock to be ready

  GPIO_PORTF_LOCK_R = 0x4C4F434B;    // unlock PortF PF0
  GPIO_PORTF_CR_R |= 0x01;           // allow changes to PF0
  GPIO_PORTF_AMSEL_R &= ~0x11;       // disable analog functionality on PF0 & PF4
  GPIO_PORTF_PCTL_R &= ~0x000F000F; // GPIO clear bit PCTL
  GPIO_PORTF_DIR_R &= ~0x11;         // make PF0 and PF4 input
  GPIO_PORTF_AFSEL_R &= ~0x11;       // disable alternation function on PF0 & PF4
  GPIO_PORTF_DEN_R |= 0x11;          // enable digital I/O on PF0 & PF4
  GPIO_PORTF_PUR_R |= 0x11;          // enable weak pull-up on PF0 & PF4
  GPIO_PORTF_IS_R &= ~0x11;          // PF0 & PF4 is edge-sensitive
  GPIO_PORTF_IBE_R |= 0x11;          // PF0 & PF4 is both edges
  GPIO_PORTF_ICR_R = 0x11;           // Clear flag0 & flag4
  GPIO_PORTF_IM_R |= 0x11;           // arm interrupt on PF0 & PF4
  NVIC_PRI7_R = (NVIC_PRI7_R&0xFF1FFFFF)|0x00400000; // priority 2
  NVIC_EN0_R |= 0x40000000;          // enable interrupt 30 in NVIC
}
```

Initializing level sensitive interrupt on PE0 with priority 1



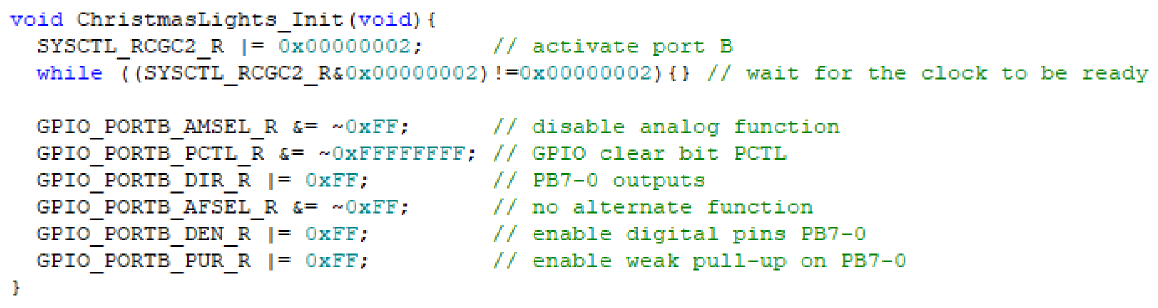
```
// Initialize level sensitive interrupt on PE0 with priority 1.
void ResetButton_Init(void){
  SYSCTL_RCGC2_R |= 0x00000010;      // activate port E
  while ((SYSCTL_RCGC2_R&0x00000010)!=0x00000010){} // wait for the clock to be ready

  GPIO_PORTE_AMSEL_R &= ~0x01;       // disable analog functionality on PE0
  GPIO_PORTE_PCTL_R &= ~0x0000000F; // GPIO clear bit PCTL
  GPIO_PORTE_DIR_R &= ~0x01;         // make PE0 input
  GPIO_PORTE_AFSEL_R &= ~0x01;       // disable alternation function on PE0
  GPIO_PORTE_DEN_R |= 0x01;          // enable digital I/O on PE0
  GPIO_PORTE_IS_R |= 0x01;           // PE0 is level-sensitive
  GPIO_PORTE_IBE_R &= ~0x01;         // PE0 is not both levels
  GPIO_PORTE_IEV_R |= 0x01;          // PE0 is high level event
  GPIO_PORTE_ICR_R = 0x01;           // Clear flag0
  GPIO_PORTE_IM_R |= 0x01;           // arm interrupt on PE0
  NVIC_PRI1_R = (NVIC_PRI1_R&0xFFFFFF1F)|0x00000020; // priority 1
  NVIC_EN0_R |= 0x00000010;          // enable interrupt 4 in NVIC
}
```

Initializing SysTick time for 0.5s delay with interrupt enable priority 3



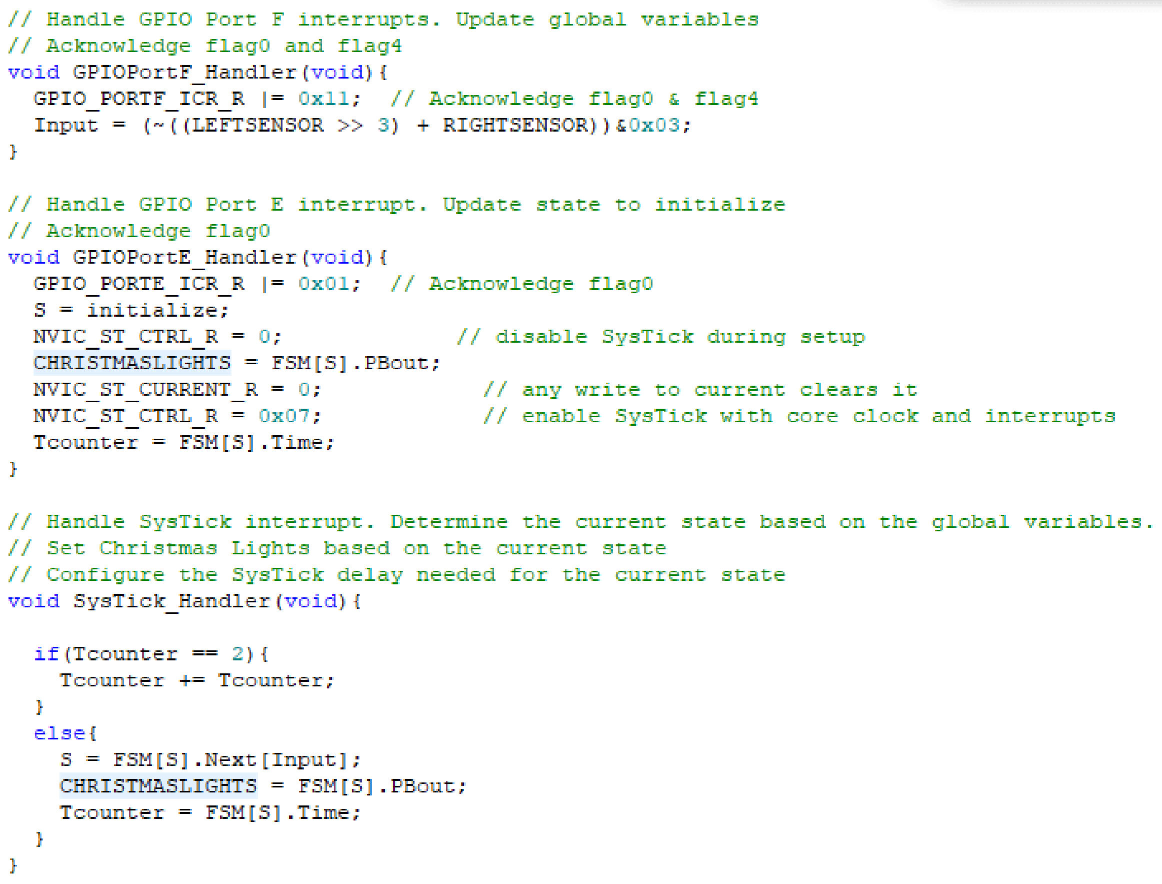
```
// Initialize SysTick timer for 0.5s delay with interrupt enabled priority 3.
void SysTick_Init(void){
  NVIC_ST_CTRL_R = 0;               // disable SysTick during setup
  NVIC_ST_RELOAD_R = 8000000 - 1;    // reload value to generate 0.5sec
  NVIC_ST_CURRENT_R = 0;             // any write to current clears it
  NVIC_SYS_PRI3_R = (NVIC_SYS_PRI3_R&0x1FFFFFFF)|0x60000000; // priority 3
  NVIC_ST_CTRL_R = 0x07;             // enable SysTick with core clock and interrupts
}
```

Initializing PB7-0 for LEDs



```
void ChristmasLights_Init(void){
  SYSCTL_RCGC2_R |= 0x00000002;      // activate port B
  while ((SYSCTL_RCGC2_R&0x00000002)!=0x00000002){} // wait for the clock to be ready

  GPIO_PORTB_AMSEL_R &= ~0xFF;       // disable analog function
  GPIO_PORTB_PCTL_R &= ~0xFFFFFFFF; // GPIO clear bit PCTL
  GPIO_PORTB_DIR_R |= 0xFF;          // PB7-0 outputs
  GPIO_PORTB_AFSEL_R &= ~0xFF;       // no alternate function
  GPIO_PORTB_DEN_R |= 0xFF;          // enable digital pins PB7-0
  GPIO_PORTB_PUR_R |= 0xFF;          // enable weak pull-up on PB7-0
}
```

PORTF and PORTE handler along with the SysTick Handler



```
// Handle GPIO Port F interrupts. Update global variables
// Acknowledge flag0 and flag4
void GPIOPortF_Handler(void){
  GPIO_PORTF_ICR_R |= 0x11;  // Acknowledge flag0 & flag4
  Input = (~((LEFTSENSOR >> 3) + RIGHTSENSOR))&0x03;
}

// Handle GPIO Port E interrupt. Update state to initialize
// Acknowledge flag0
void GPIOPortE_Handler(void){
  GPIO_PORTE_ICR_R |= 0x01;  // Acknowledge flag0
  S = initialize;
  NVIC_ST_CTRL_R = 0;             // disable SysTick during setup
  CHRISTMASLIGHTS = FSM[S].PBout;
  NVIC_ST_CURRENT_R = 0;            // any write to current clears it
  NVIC_ST_CTRL_R = 0x07;            // enable SysTick with core clock and interrupts
  Tcounter = FSM[S].Time;
}

// Handle SysTick interrupt. Determine the current state based on the global variables.
// Set Christmas Lights based on the current state
// Configure the SysTick delay needed for the current state
void SysTick_Handler(void){

  if(Tcounter == 2){
    Tcounter += Tcounter;
  }
  else{
    S = FSM[S].Next[Input];
    CHRISTMASLIGHTS = FSM[S].PBout;
    Tcounter = FSM[S].Time;
  }
}
```

CONCLUSION:

Implementing this project onto the board seemed much easier than developing the code to run the Christmas lights. Ran into issues with building the board, mainly with wiring the pins to board to make sure the LEDs are visible. The buttons were very sensitive, so I had issues with pressing the buttons to hard or releasing too early. This made my LEDs turn on when they are not supposed to be and vice versa. As you can see in the hardware design, my LEDs are very dim because of wires blocking it. This project was very helpful with understand how to interface to buttons and LEDs, while using a Moore FSM.