CECS 346 Fall 2020 Project 1

Traffic Light Controller

By

Len Quach

10/30/2020

Design a traffic light controller for the intersection of two equally busy one-way streets. The goal is to maximize traffic flow, minimize waiting time at a red light, and avoid accidents.

Introduction

The project including four major objectives: the understanding and implementing of indexed data structures; learning how to create a segmented software system; the study of real time synchronization by designing a finite state machine controller; learn how to use edge-trigger interrupt. We will define data structure for FSM, create fixed-time delays using the SysTick timer, and debugging real-time systems.

We design a traffic controller for the intersection of two equally busy one-way streets which are labeled as South and West. There are three inputs to LaunchPad, two are car sensors, and one is a pedestrian sensor.

We will interface 6 LEDs that represent the two Red-Yellow-Green traffic lights, and we will use the PF3 green LED for the "walk" light and the PF1 red LED for the "don't walk" light. The walk sequence should be showing three separate conditions: "walk", "hurry up" (LED flashing every 0.25s) and "don't walk".

Operation

https://youtu.be/iNQFdyHOl-w

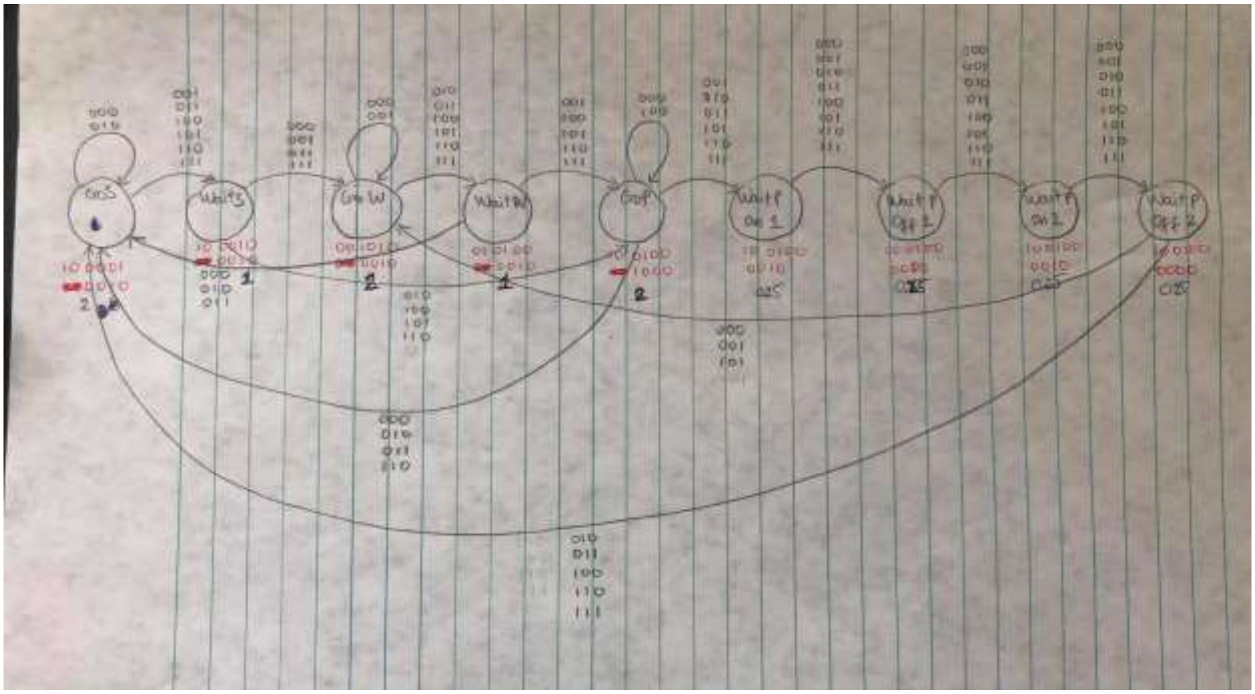
Theory

State table for the Moore FSM

State	**000**	**001**	**010**	**011**	**100**	**101**	**110**	**111**
GoS	goS	waitS	goS	waitS	waitS	waitS	waitS	waitS
WaitS	goW	goW	goP	goW	goP	goP	goW	goW
GoW	goW	goW	waitW	waitW	waitW	waitW	waitW	waitW
WaitW	goS	goP	goS	goS	goP	goP	goP	goP
GoP	goP	waitPOn1	waitPOn1	waitPOn1	goP	waitPOn1	waitPOn1	waitPOn1
WaitPOn1	waitPOff1	waitPOff1	waitPOff1	waitPOff1	waitPOff1	waitPOff1	waitPOff1	waitPOff1
WaitPOff1	waitPOn2	waitPOn2	waitPOn2	waitPOn2	waitPOn2	waitPOn2	waitPOn2	waitPOn2
WaitPOn2	waitPOff2	waitPOff2	waitPOff2	waitPOff2	waitPOff2	waitPOff2	waitPOff2	waitPOff2
WaitPOff2	goW	goW	goS	goS	goS	goW	goS	goS

Input: P,S,W

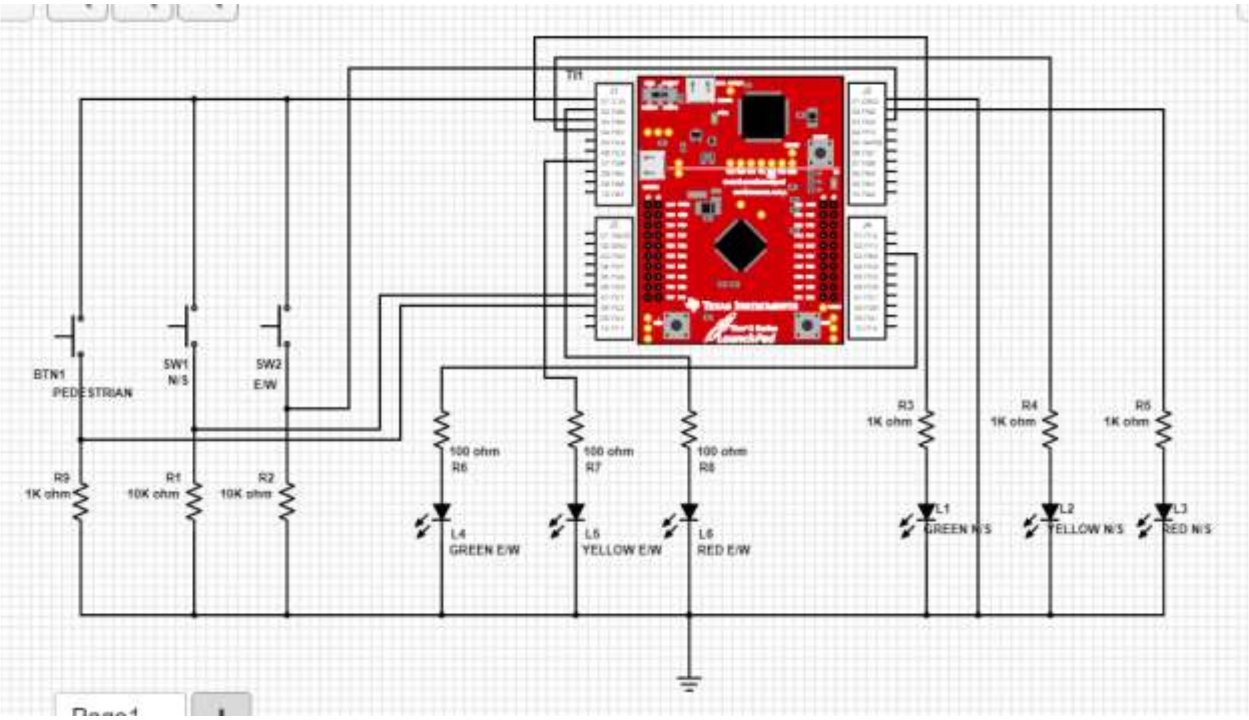
State diagram for the Moore FSM

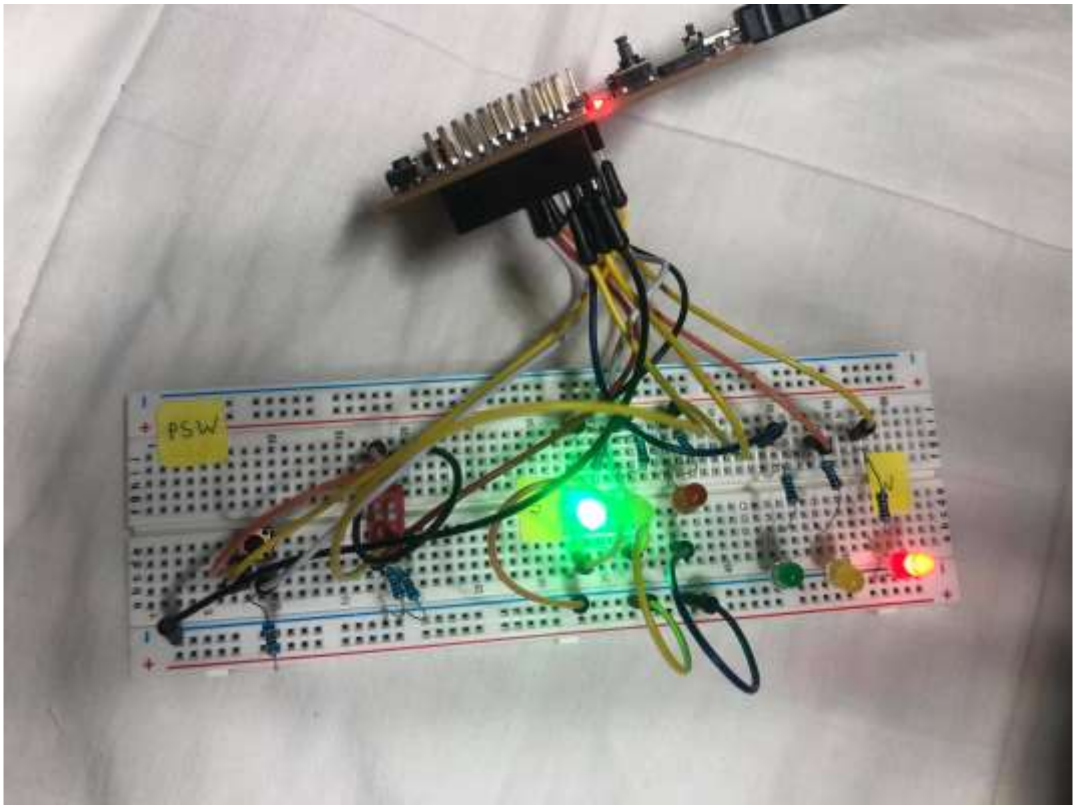


When none of the three sensors is true, stay in current state or finish transition to green. If one sensor is true, turn on the green for traffic light of that direction or “walk” for pedestrian and stay on as long as that sensor is true and no other sensor is true. If there are more than one sensor are true: cycle through the requests servicing them in a round robin fashion.

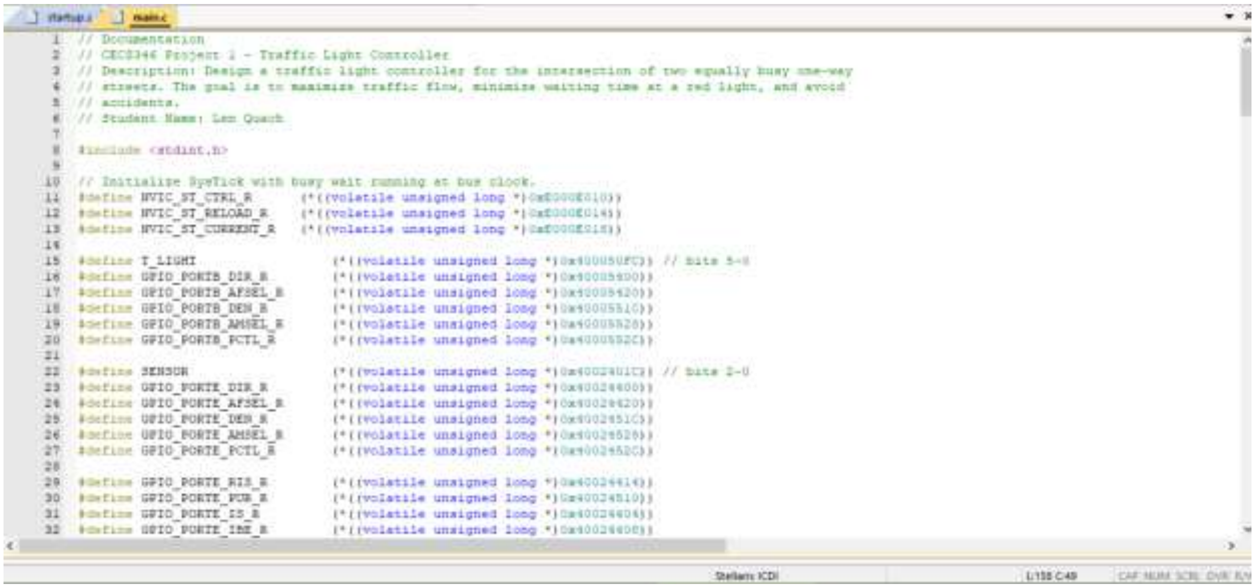
The time duration for green/walk 2 seconds, yellow/hurry 1 seconds. Red/don’t walk 3 seconds. We will implement a Moore machine and use SysTick timer to wait a prescribed amount of time. The state graph defines exactly what the system does in a clear and unambiguous fashion.

Hardware design

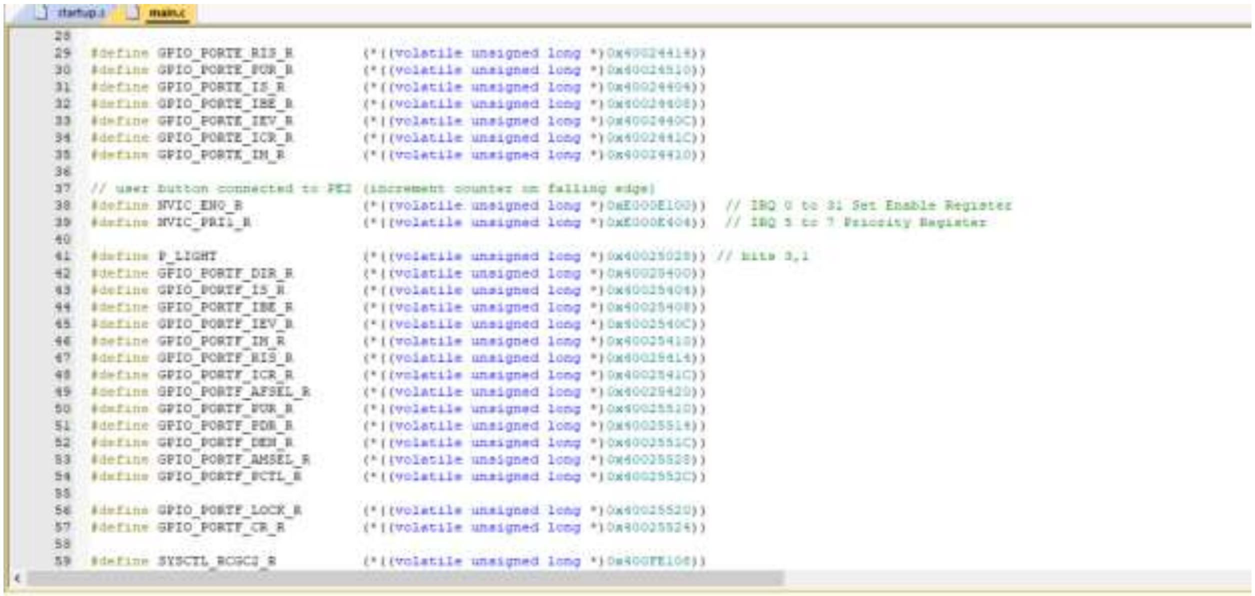

TI1
BTN1
PEDESTRIAN
SW1
N/S
SW2
E/W
R9
1K ohm
R1
10K ohm
R2
10K ohm
100 ohm
R6
100 ohm
R7
100 ohm
R8
L4
GREEN E/W
L5
YELLOW E/W
L6
RED E/W
R3
1K ohm
R4
1K ohm
R5
1K ohm
L1
GREEN N/S
L2
YELLOW N/S
L3
RED N/S
Page1



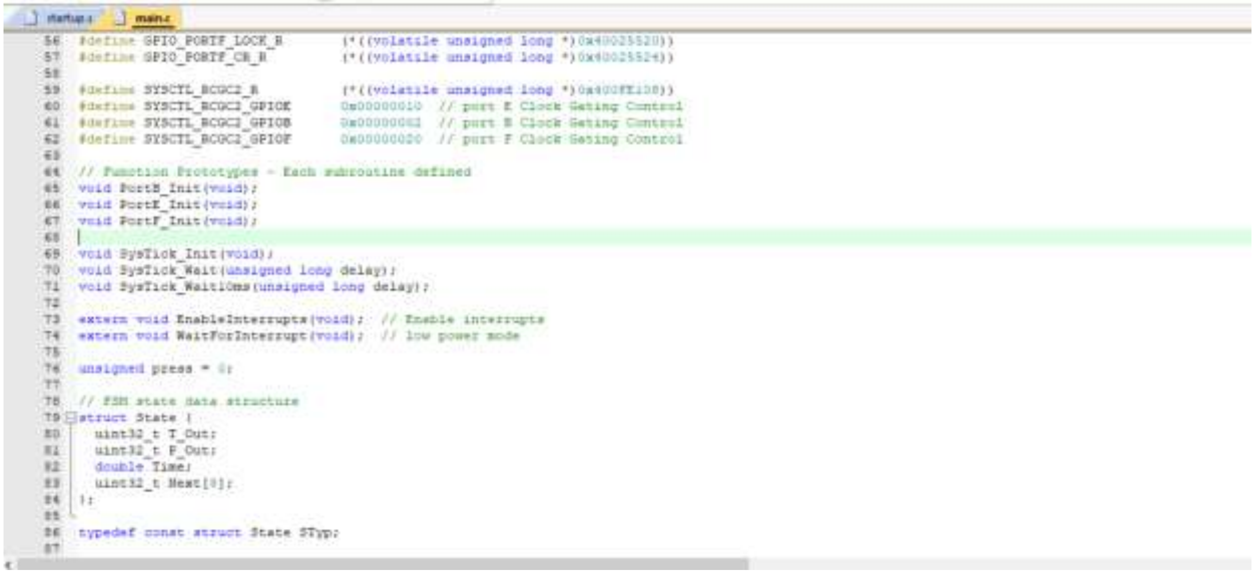
Software design



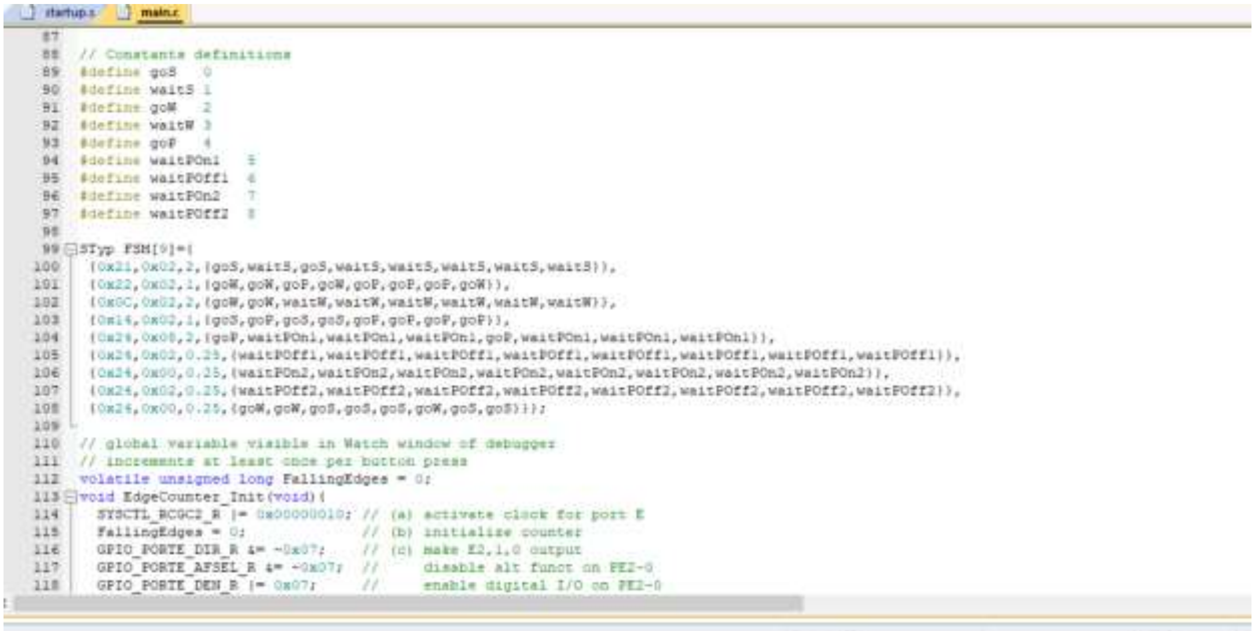
```c
// Documentation
// CECS346 Project 1 - Traffic Light Controller
// Description: Design a traffic light controller for the intersection of two equally busy one-way
// streets. The goal is to maximize traffic flow, minimize waiting time at a red light, and avoid
// accidents.
// Student Name: Len Quach

#include <stdint.h>

// Initialize SysTick with busy wait running at bus clock.
#define NVIC_ST_CTRL_R      (*((volatile unsigned long *)0xE000E010))
#define NVIC_ST_RELOAD_R    (*((volatile unsigned long *)0xE000E014))
#define NVIC_ST_CURRENT_R   (*((volatile unsigned long *)0xE000E018))

#define T_LIGHT                 (*((volatile unsigned long *)0x400050FC)) // bits 5-0
#define GPIO_PORTB_DIR_R        (*((volatile unsigned long *)0x40005400))
#define GPIO_PORTB_AFSEL_R      (*((volatile unsigned long *)0x40005420))
#define GPIO_PORTB_DEN_R        (*((volatile unsigned long *)0x4000551C))
#define GPIO_PORTB_AMSEL_R      (*((volatile unsigned long *)0x40005528))
#define GPIO_PORTB_PCTL_R       (*((volatile unsigned long *)0x4000552C))

#define SENSOR                  (*((volatile unsigned long *)0x4002401C)) // bits 2-0
#define GPIO_PORTE_DIR_R        (*((volatile unsigned long *)0x40024400))
#define GPIO_PORTE_AFSEL_R      (*((volatile unsigned long *)0x40024420))
#define GPIO_PORTE_DEN_R        (*((volatile unsigned long *)0x4002451C))
#define GPIO_PORTE_AMSEL_R      (*((volatile unsigned long *)0x40024528))
#define GPIO_PORTE_PCTL_R       (*((volatile unsigned long *)0x4002452C))

#define GPIO_PORTE_RIS_R        (*((volatile unsigned long *)0x40024414))
#define GPIO_PORTE_PUR_R        (*((volatile unsigned long *)0x40024510))
#define GPIO_PORTE_IS_R         (*((volatile unsigned long *)0x40024404))
#define GPIO_PORTE_IBE_R        (*((volatile unsigned long *)0x40024408))
```



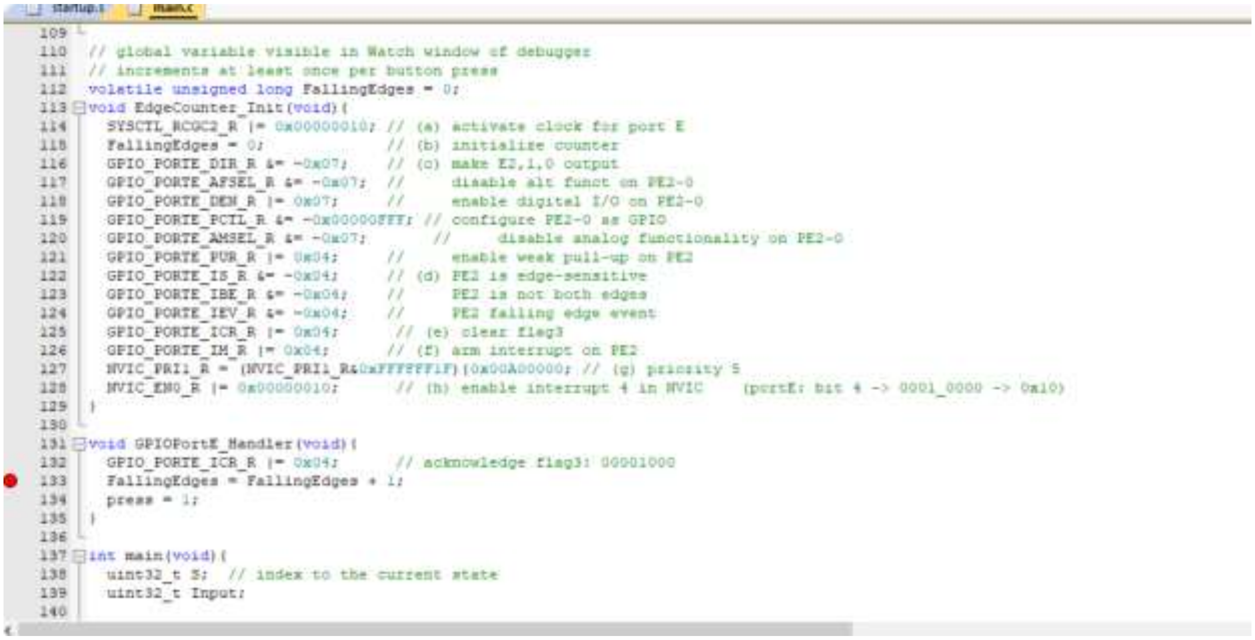
```c

#define GPIO_PORTE_RIS_R        (*((volatile unsigned long *)0x40024414))
#define GPIO_PORTE_PUR_R        (*((volatile unsigned long *)0x40024510))
#define GPIO_PORTE_IS_R         (*((volatile unsigned long *)0x40024404))
#define GPIO_PORTE_IBE_R        (*((volatile unsigned long *)0x40024408))
#define GPIO_PORTE_IEV_R        (*((volatile unsigned long *)0x4002440C))
#define GPIO_PORTE_ICR_R        (*((volatile unsigned long *)0x4002441C))
#define GPIO_PORTE_IM_R         (*((volatile unsigned long *)0x40024410))

// user button connected to PE2 (increment counter on falling edge)
#define NVIC_EN0_R              (*((volatile unsigned long *)0xE000E100))  // IRQ 0 to 31 Set Enable Register
#define NVIC_PRI1_R             (*((volatile unsigned long *)0xE000E404))  // IRQ 5 to 7 Priority Register

#define P_LIGHT                 (*((volatile unsigned long *)0x40025028)) // bits 3,1
#define GPIO_PORTF_DIR_R        (*((volatile unsigned long *)0x40025400))
#define GPIO_PORTF_IS_R         (*((volatile unsigned long *)0x40025404))
#define GPIO_PORTF_IBE_R        (*((volatile unsigned long *)0x40025408))
#define GPIO_PORTF_IEV_R        (*((volatile unsigned long *)0x4002540C))
#define GPIO_PORTF_IM_R         (*((volatile unsigned long *)0x40025410))
#define GPIO_PORTF_RIS_R        (*((volatile unsigned long *)0x40025414))
#define GPIO_PORTF_ICR_R        (*((volatile unsigned long *)0x4002541C))
#define GPIO_PORTF_AFSEL_R      (*((volatile unsigned long *)0x40025420))
#define GPIO_PORTF_PUR_R        (*((volatile unsigned long *)0x40025510))
#define GPIO_PORTF_PDR_R        (*((volatile unsigned long *)0x40025514))
#define GPIO_PORTF_DEN_R        (*((volatile unsigned long *)0x4002551C))
#define GPIO_PORTF_AMSEL_R      (*((volatile unsigned long *)0x40025528))
#define GPIO_PORTF_PCTL_R       (*((volatile unsigned long *)0x4002552C))

#define GPIO_PORTF_LOCK_R       (*((volatile unsigned long *)0x40025520))
#define GPIO_PORTF_CR_R         (*((volatile unsigned long *)0x40025524))

#define SYSCTL_RCGC2_R          (*((volatile unsigned long *)0x400FE108))
```



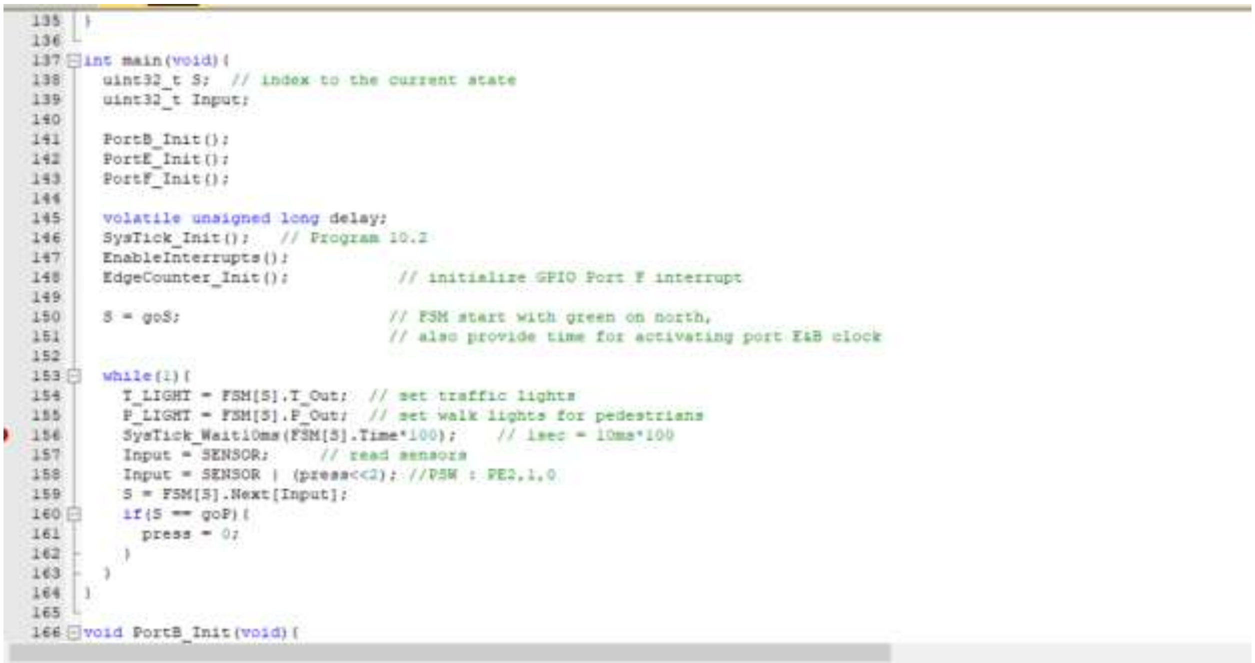
```c
#define GPIO_PORTF_LOCK_R       (*((volatile unsigned long *)0x40025520))
#define GPIO_PORTF_CR_R         (*((volatile unsigned long *)0x40025524))

#define SYSCTL_RCGC2_R          (*((volatile unsigned long *)0x400FE108))
#define SYSCTL_RCGC2_GPIOE      0x00000010  // port E Clock Gating Control
#define SYSCTL_RCGC2_GPIOB      0x00000002  // port B Clock Gating Control
#define SYSCTL_RCGC2_GPIOF      0x00000020  // port F Clock Gating Control

// Function Prototypes - Each subroutine defined
void PortB_Init(void);
void PortE_Init(void);
void PortF_Init(void);

void SysTick_Init(void);
void SysTick_Wait(unsigned long delay);
void SysTick_Wait10ms(unsigned long delay);

extern void EnableInterrupts(void);  // Enable interrupts
extern void WaitForInterrupt(void);  // low power mode

unsigned press = 0;

// FSM state data structure
struct State {
  uint32_t T_Out;
  uint32_t P_Out;
  double Time;
  uint32_t Next[8];
};

typedef const struct State STyp;

```



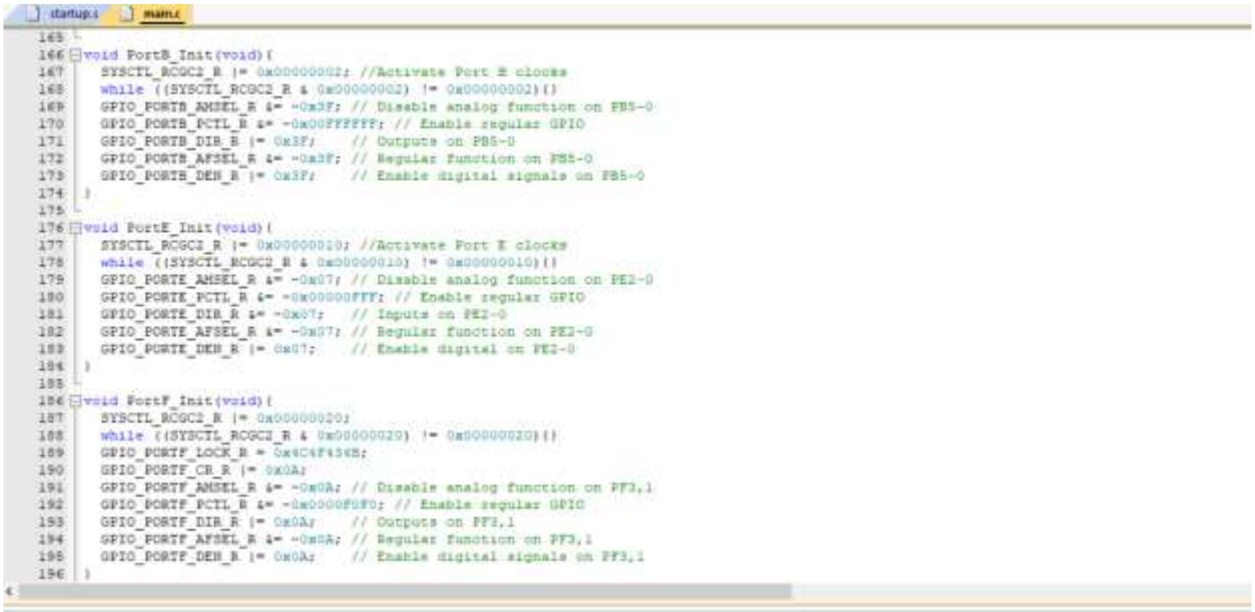
```c

// Constants definitions
#define goS   0
#define waitS 1
#define goW   2
#define waitW 3
#define goP   4
#define waitPOn1   5
#define waitPOff1  6
#define waitPOn2   7
#define waitPOff2  8

STyp FSM[9]={
 {0x21,0x02,2,{goS,waitS,goS,waitS,waitS,waitS,waitS,waitS}},
 {0x22,0x02,1,{goW,goW,goP,goW,goP,goP,goP,goW}},
 {0x0C,0x02,2,{goW,goW,waitW,waitW,waitW,waitW,waitW,waitW}},
 {0x14,0x02,1,{goS,goP,goS,goS,goP,goP,goP,goP}},
 {0x24,0x08,2,{goP,waitPOn1,waitPOn1,waitPOn1,goP,waitPOn1,waitPOn1,waitPOn1}},
 {0x24,0x02,0.25,{waitPOff1,waitPOff1,waitPOff1,waitPOff1,waitPOff1,waitPOff1,waitPOff1,waitPOff1}},
 {0x24,0x00,0.25,{waitPOn2,waitPOn2,waitPOn2,waitPOn2,waitPOn2,waitPOn2,waitPOn2,waitPOn2}},
 {0x24,0x02,0.25,{waitPOff2,waitPOff2,waitPOff2,waitPOff2,waitPOff2,waitPOff2,waitPOff2,waitPOff2}},
 {0x24,0x00,0.25,{goW,goW,goS,goS,goS,goW,goS,goS}}};

// global variable visible in Watch window of debugger
// increments at least once per button press
volatile unsigned long FallingEdges = 0;
void EdgeCounter_Init(void){
  SYSCTL_RCGC2_R |= 0x00000010; // (a) activate clock for port E
  FallingEdges = 0;             // (b) initialize counter
  GPIO_PORTE_DIR_R &= ~0x07;    // (c) make E2,1,0 output
  GPIO_PORTE_AFSEL_R &= ~0x07;  //     disable alt funct on PE2-0
  GPIO_PORTE_DEN_R |= 0x07;     //     enable digital I/O on PE2-0
```



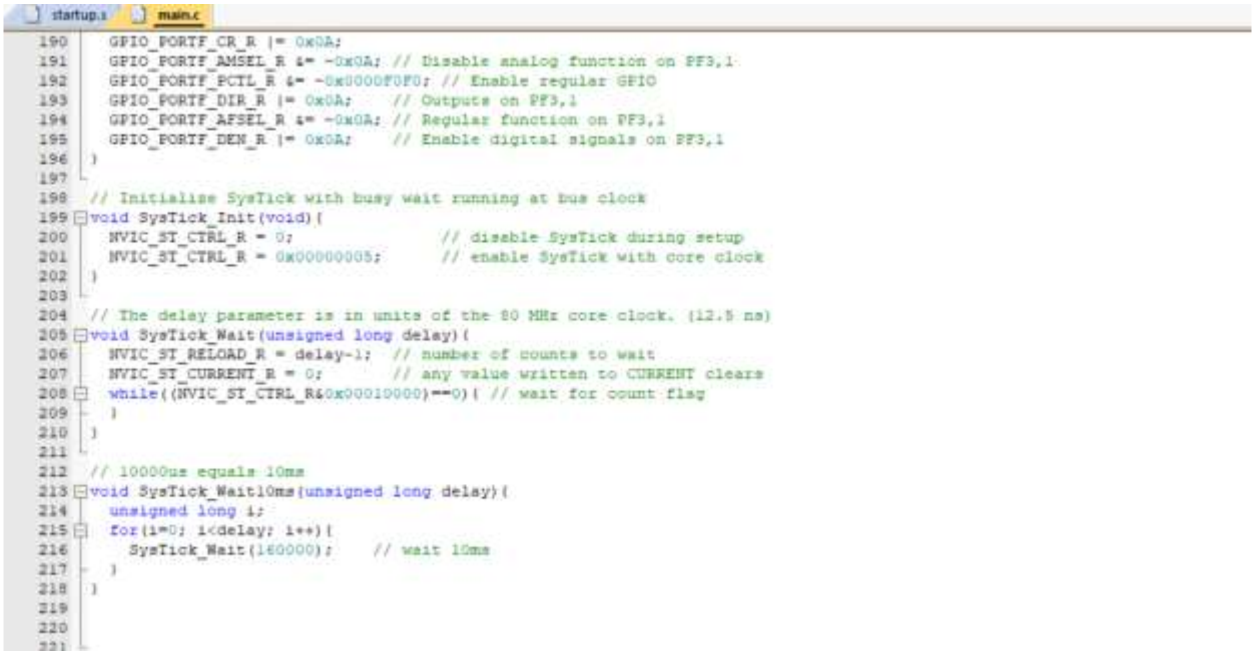
```c

// global variable visible in Watch window of debugger
// increments at least once per button press
volatile unsigned long FallingEdges = 0;
void EdgeCounter_Init(void){
  SYSCTL_RCGC2_R |= 0x00000010; // (a) activate clock for port E
  FallingEdges = 0;             // (b) initialize counter
  GPIO_PORTE_DIR_R &= ~0x07;    // (c) make E2,1,0 output
  GPIO_PORTE_AFSEL_R &= ~0x07;  //     disable alt funct on PE2-0
  GPIO_PORTE_DEN_R |= 0x07;     //     enable digital I/O on PE2-0
  GPIO_PORTE_PCTL_R &= ~0x00000FFF; // configure PE2-0 as GPIO
  GPIO_PORTE_AMSEL_R &= ~0x07;      //     disable analog functionality on PE2-0
  GPIO_PORTE_PUR_R |= 0x04;     //     enable weak pull-up on PE2
  GPIO_PORTE_IS_R &= ~0x04;     // (d) PE2 is edge-sensitive
  GPIO_PORTE_IBE_R &= ~0x04;    //     PE2 is not both edges
  GPIO_PORTE_IEV_R &= ~0x04;    //     PE2 falling edge event
  GPIO_PORTE_ICR_R |= 0x04;      // (e) clear flag3
  GPIO_PORTE_IM_R |= 0x04;      // (f) arm interrupt on PE2
  NVIC_PRI1_R = (NVIC_PRI1_R&0xFFFFFF1F)|0x00A00000; // (g) priority 5
  NVIC_EN0_R |= 0x00000010;      // (h) enable interrupt 4 in NVIC    (portE: bit 4 -> 0001_0000 -> 0x10)
}

void GPIOPortE_Handler(void){
  GPIO_PORTE_ICR_R |= 0x04;      // acknowledge flag3: 00001000
  FallingEdges = FallingEdges + 1;
  press = 1;
}

int main(void){
  uint32_t S;  // index to the current state
  uint32_t Input;

```



```
}

int main(void){
  uint32_t S;  // index to the current state
  uint32_t Input;

  PortB_Init();
  PortE_Init();
  PortF_Init();

  volatile unsigned long delay;
  SysTick_Init();   // Program 10.2
  EnableInterrupts();
  EdgeCounter_Init();           // initialize GPIO Port F interrupt

  S = goS;                      // FSM start with green on north,
                                // also provide time for activating port E&B clock

  while(1){
    T_LIGHT = FSM[S].T_Out;  // set traffic lights
    P_LIGHT = FSM[S].P_Out;  // set walk lights for pedestrians
    SysTick_Wait10ms(FSM[S].Time*100);    // 1sec = 10ms*100
    Input = SENSOR;      // read sensors
    Input = SENSOR | (press<<2); //PSW : PE2,1,0
    S = FSM[S].Next[Input];
    if(S == goP){
      press = 0;
    }
  }
}

void PortB_Init(void){
```



```

void PortB_Init(void){
  SYSCTL_RCGC2_R |= 0x00000002; //Activate Port B clocks
  while ((SYSCTL_RCGC2_R & 0x00000002) != 0x00000002){}
  GPIO_PORTB_AMSEL_R &= ~0x3F; // Disable analog function on PB5-0
  GPIO_PORTB_PCTL_R &= ~0x00FFFFFF; // Enable regular GPIO
  GPIO_PORTB_DIR_R |= 0x3F;    // Outputs on PB5-0
  GPIO_PORTB_AFSEL_R &= ~0x3F; // Regular function on PB5-0
  GPIO_PORTB_DEN_R |= 0x3F;    // Enable digital signals on PB5-0
}

void PortE_Init(void){
  SYSCTL_RCGC2_R |= 0x00000010; //Activate Port E clocks
  while ((SYSCTL_RCGC2_R & 0x00000010) != 0x00000010){}
  GPIO_PORTE_AMSEL_R &= ~0x07; // Disable analog function on PE2-0
  GPIO_PORTE_PCTL_R &= ~0x00000FFF; // Enable regular GPIO
  GPIO_PORTE_DIR_R &= ~0x07;   // Inputs on PE2-0
  GPIO_PORTE_AFSEL_R &= ~0x07; // Regular function on PE2-0
  GPIO_PORTE_DEN_R |= 0x07;    // Enable digital on PE2-0
}

void PortF_Init(void){
  SYSCTL_RCGC2_R |= 0x00000020;
  while ((SYSCTL_RCGC2_R & 0x00000020) != 0x00000020){}
  GPIO_PORTF_LOCK_R = 0x4C4F434B;
  GPIO_PORTF_CR_R |= 0x0A;
  GPIO_PORTF_AMSEL_R &= ~0x0A; // Disable analog function on PF3,1
  GPIO_PORTF_PCTL_R &= ~0x0000F0F0; // Enable regular GPIO
  GPIO_PORTF_DIR_R |= 0x0A;    // Outputs on PF3,1
  GPIO_PORTF_AFSEL_R &= ~0x0A; // Regular function on PF3,1
  GPIO_PORTF_DEN_R |= 0x0A;    // Enable digital signals on PF3,1
}
```



```c
  GPIO_PORTF_CR_R |= 0x0A;
  GPIO_PORTF_AMSEL_R &= ~0x0A; // Disable analog function on PF3,1
  GPIO_PORTF_PCTL_R &= ~0x0000F0F0; // Enable regular GPIO
  GPIO_PORTF_DIR_R |= 0x0A;    // Outputs on PF3,1
  GPIO_PORTF_AFSEL_R &= ~0x0A; // Regular function on PF3,1
  GPIO_PORTF_DEN_R |= 0x0A;    // Enable digital signals on PF3,1
}

// Initialise SysTick with busy wait running at bus clock
void SysTick_Init(void){
  NVIC_ST_CTRL_R = 0;                // disable SysTick during setup
  NVIC_ST_CTRL_R = 0x00000005;       // enable SysTick with core clock
}

// The delay parameter is in units of the 80 MHz core clock. (12.5 ns)
void SysTick_Wait(unsigned long delay){
  NVIC_ST_RELOAD_R = delay-1;  // number of counts to wait
  NVIC_ST_CURRENT_R = 0;       // any value written to CURRENT clears
  while((NVIC_ST_CTRL_R&0x00010000)==0){ // wait for count flag
  }
}

// 10000us equals 10ms
void SysTick_Wait10ms(unsigned long delay){
  unsigned long i;
  for(i=0; i<delay; i++){
    SysTick_Wait(160000);    // wait 10ms
  }
}
```

Conclusion

My success in this project is better understand and implement of indexed data structures, learn how to create a segmented software system including linked data structures, creating fixed time delay using SysTick time , and debugging real time systems by designing a state machine.

I think the hardest part of this project is designing the correct finite state machine. I had a lot of errors on the fsm with the older lab description version. However, with the newer version, it is so much clearer to follow the rules and as to give the fair chances for each participant. The next part that I also had many troubles on was the interrupt to implement the pedestrian button.