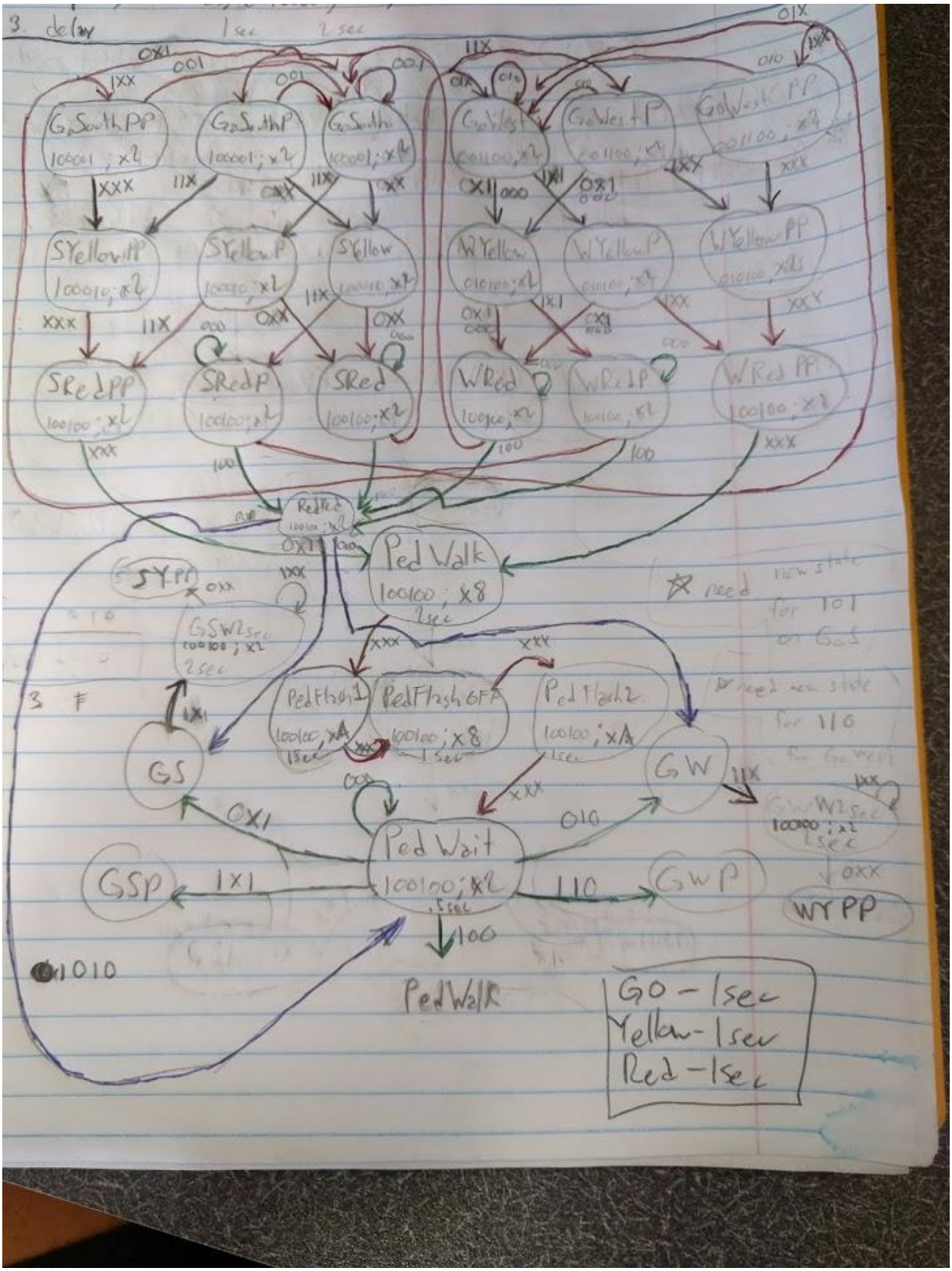
3. delay
1 sec
2 sec
GoSouthPP
GoSouthP
GoSouth
GoWestPP
GoWestP
GoWest
SYellowPP
SYellowP
SYellow
WYellowPP
WYellowP
WYellow
SRedPP
SRedP
SRed
WRedPP
WRedP
WRed
RedRed
PedWalk
100100 ; x8
1sec
SYPP
GSW2sec
2sec
PedFlash1
PedFlash GFA
PedFlash2
GS
GW
GSW2sec
1sec
WYPP
PedWait
100100; x2
.5sec
GSP
GWP
PedWalk
need new state for 101 or GoS
need new state for 110
GO - 1sec
Yellow - 1sec
Red - 1sec

```
// TableTrafficLight.c
// Zach Sisti, Elgin Allen
// 3/19/17


// east/west red light connected to PB5
// east/west yellow light connected to PB4
// east/west green light connected to PB3
// north/south facing red light connected to PB2
// north/south facing yellow light connected to PB1
// north/south facing green light connected to PB0
// pedestrian detector connected to PE2 (1=pedestrian present)
// north/south car detector connected to PE1 (1=car present)
// east/west car detector connected to PE0 (1=car present)
// "walk" light connected to PF3 (built-in green LED)
// "don't walk" light connected to PF1 (built-in red LED)
#include <stdint.h>
#include "tm4c123gh6pm.h"
#include "SysTick.h"
#include "TExaS.h"

// Declare your FSM linked structure here
struct State {
  uint32_t TrafficOut;
                uint32_t PedOut;
  uint32_t Time;
  uint32_t Next[8];};
typedef const struct State STyp;
//write states here
#define goSouth 0
#define goSouthP 1
#define goSouthPP 2
#define sYellow 3
#define sYellowP 4
#define sYellowPP 5
#define sRed 6
#define sRedP 7
#define sRedPP 8
#define goWest 9
#define goWestP 10
#define goWestPP 11
#define wYellow 12
#define wYellowP 13
#define wYellowPP 14
#define wRed 15
#define wRedP 16
#define wRedPP 17
#define pedWalk 18
#define pedFlash1 19
#define pedFlashOff 20
#define pedFlash2 21
#define pedWait 22
#define sWait2 23
#define wWait2 24
#define redPed 25
STyp FSM [26]={
{0x21,0x2,250,{sYellow,goSouth,sYellow,sYellow,sYellowP,sWait2,sYellowP,sWait2}}, //0
{0x21,0x2,250,{sYellow,goSouth,sYellow,sYellow,sYellowPP,sWait2,sYellowPP,sWait2}}, //1
{0x21,0x2,250,{sYellowPP,sYellowPP,sYellowPP,sYellowPP,sYellowPP,sWait2,sYellowPP,sWait2}}, //2
{0x22,0x2,250,{sRed,sRed,sRed,sRed,sRedP,sRedP,sRedP,sRedP}}, //3
{0x22,0x2,250,{sRed,sRed,sRed,sRed,sRedPP,sRedPP,sRedPP,sRedPP}}, //4
{0x22,0x2,250,{sRedPP,sRedPP,sRedPP,sRedPP,sRedPP,sRedPP,sRedPP,sRedPP}}, //5
{0x24,0x2,250,{sRed,goSouth,goWest,goWest,redPed,goSouthP,goWestP,goWestP}}, //6
{0x24,0x2,250,{sRed,goSouth,goWest,goWest,redPed,goWestPP,goWestPP,goWestPP}}, //7
{0x24,0x2,250,{pedWalk,pedWalk,pedWalk,pedWalk,pedWalk,pedWalk,pedWalk,pedWalk}},//8
{0x0C,0x2,250,{wYellow,wYellow,goWest,wYellow,wYellowP,wYellowP,wWait2,wWait2}},//9
```

```
{0x0C,0x2,250,{wYellow,wYellow,goWest,wYellow,wYellowPP,wYellowPP,wWait2,wWait2}},//10
{0x0C,0x2,250,{wYellowPP,wYellowPP,wYellowPP,wYellowPP,wYellowPP,wYellowPP,wWait2,wWait2}},//11
{0x14,0x2,250,{wRed,wRed,wRed,wRed,wRedP,wRedP,wRedP,wRedP}},//12
{0x14,0x2,250,{wRed,wRed,wRed,wRed,wRedPP,wRedPP,wRedPP,wRedPP}},//13
{0x14,0x2,250,{wRedPP,wRedPP,wRedPP,wRedPP,wRedPP,wRedPP,wRedPP,wRedPP,}},//14
{0x24,0x2,250,{wRed,goSouth,goWest,goSouth,redPed,goSouthP,goSouthP,goSouthP}},//15
{0x24,0x2,250,{wRed,goSouth,goWest,goSouth,redPed,goSouthPP,goWestPP,goSouthPP}},//16
{0x24,0x2,250,{pedWalk,pedWalk,pedWalk,pedWalk,pedWalk,pedWalk,pedWalk,pedWalk}},//17
{0x24,0x8,500,{pedFlash1,pedFlash1,pedFlash1,pedFlash1,pedFlash1,pedFlash1,pedFlash1,pedFlash1}},//18
{0x24,0xA,100,{pedFlashOff,pedFlashOff,pedFlashOff,pedFlashOff,pedFlashOff,pedFlashOff,pedFlashOff,pedFlashOff}},//19
{0x24,0x8,100,{pedFlash2,pedFlash2,pedFlash2,pedFlash2,pedFlash2,pedFlash2,pedFlash2,pedFlash2}},//20
{0x24,0xA,100,{pedWait,pedWait,pedWait,pedWait,pedWait,pedWait,pedWait,pedWait}},//21
{0x24,0x2,250,{pedWait,goSouth,goWest,goSouth,redPed,goSouthP,goWestP,goWestP}},//22
{0x21,0x2,500,{sYellowPP,sYellowPP,sYellowPP,sYellowPP,sWait2,sWait2,sWait2,sWait2}},//23
{0x0C,0x2,500,{wYellowPP,wYellowPP,wYellowPP,wYellowPP,wWait2,wWait2,wWait2,wWait2}},//24
{0x24,0x2,100,{pedWait,goSouth,goWest,goSouth,pedWalk,pedWalk,pedWalk,pedWalk}}};//25



void EnableInterrupts(void);
int main(void){ volatile unsigned long delay;
        int32_t S;
        int32_t Input;
 TExaS_Init(SW_PIN_PE210, LED_PIN_PB543210); // activate traffic simulation and set system clock to 80 MHz
 SysTick_Init();
 EnableInterrupts();
        SYSCTL_RCGC2_R |= 0x32;
        delay = SYSCTL_RCGC2_R;
        GPIO_PORTE_AMSEL_R &= ~0x3F;            //DISABLE ANALOG FUNC PE0-5
        GPIO_PORTE_PCTL_R &= ~0x00FFFFFF; // ENABLES REG GPIO FUNC PE0-5
 GPIO_PORTE_DIR_R |= 0x3F;   // SET PE0-5 OUTPUTS
 GPIO_PORTE_AFSEL_R &= ~0x3F; // REGULAR FUNC PE0-5
 GPIO_PORTE_DEN_R |= 0x3F;    //  DIGITAL ENABLE PE0-5

        GPIO_PORTB_AMSEL_R &= ~0x07; // DISABLE ANALOG FUNC PB0-2
 GPIO_PORTB_PCTL_R &= ~0x00000FFF; // ENABLES REG GPIO FUNC PB0-2
 GPIO_PORTB_DIR_R &= ~0x07;    // SET PB0-2 INPUT
 GPIO_PORTB_AFSEL_R &= ~0x07; // REGULAR FUNC PB0-2
        GPIO_PORTB_DEN_R |= 0x07;    // DIGITAL ENABLE PB0-2

        GPIO_PORTF_AMSEL_R &= ~0x0A;            // DISABLE ANALOG FUNC PF1,3
        GPIO_PORTF_PCTL_R &= ~0x0000F0F0; // ENABLE REG GPIO FUNC PF1,3
 GPIO_PORTF_DIR_R |= 0x0A;   // SET PF1,3 OUTPUT
 GPIO_PORTF_AFSEL_R &= ~0x0A; // REGULAR FUNC PF1,3
 GPIO_PORTF_DEN_R |= 0x0A;    // DIGITAL ENABLE PF1,3
 //FSM Engine
 while(1){
   GPIO_PORTE_DATA_R = FSM[S].TrafficOut;  // set lights
   GPIO_PORTF_DATA_R = FSM[S].PedOut;
                SysTick_Wait10ms(FSM[S].Time);
   Input = GPIO_PORTB_DATA_R;     // read sensors
   S = FSM[S].Next[Input];
 }
}
```

```
// SysTick.c
// Implements two busy-wait based delay routines
#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))
void SysTick_Init(void){
        NVIC_ST_CTRL_R  = 0;
        NVIC_ST_RELOAD_R = 0x00FFFFFF;
        NVIC_ST_CURRENT_R = 0;
        NVIC_ST_CTRL_R  = 0x00000005;
}
// The delay parameter is in units of the 80 MHz core clock. (12.5 ns)
void SysTick_Wait(uint32_t delay){
        NVIC_ST_RELOAD_R = delay-1;
        NVIC_ST_CURRENT_R = 0;
        while((NVIC_ST_CTRL_R&0x00010000)==0){
        //Wait for count flag
        }

}

// Time delay using busy wait.
// waits for count*10ms
// 10000us equals 10ms
void SysTick_Wait10ms(uint32_t delay){
        uint32_t i;
        for( i=0; i<delay; i++){
                SysTick_Wait(800000);
                //wait 10ms
        }
}
```