Alarik Damrow

Professor Gevik Sardarbegians

CECS 346 Sec1 (7147)

1 November 2022

Project 2 Report: Drag Strip

This lab was done to present the demonstration of interrupts and how I can use them for the situation that was given for this lab. This lab was not difficult to develop the FSM and other functions to make the transitions happen but become difficult when the interrupts were needed for certain functions. Lots of reading was done from the manual and class book to have an understanding of how to trigger an interrupt and how to set a scenario for a certain interrupt.

GPIO Ports and Pins Table

|  |  |
| --- | --- |
| Description | Pin |
| Right Player Control | PA3 |
| Left Player Control | PA4 |
| Reset | PE0 |
| Right Player Bottom Yellow Light | PB0 |
| Right Player Upper Yellow Light | PB1 |
| Right Player Green Light | PB2 |
| Right Player Red Light | PB3 |
| Left Player Bottom Yellow Light | PB4 |
| Left Player Upper Yellow Light | PB5 |
| Left Player Green Light | PB6 |
| Left Player Red Light | PB7 |

Input Description of Ports and Pins

|  |  |
| --- | --- |
| Pin | Purpose |
| PA3 | Pin is responsible for capturing the response of the right player for this drag strip. Response is captured via a button that is set. Sensor variable is responsible in capturing the response from this pin as the program shows. |
| PA4 | Pin is responsible for capturing the response of the left player for this drag strip. Response is captured via a button that is set. Sensor variable is responsible for capturing the response in the program |
| PE0 | Pin is responsible in triggering the reset interrupt that is programed. No variable is specified for this pin but a interrupt and handler is set for this pin. |

Output Description of Ports and Pins

|  |  |
| --- | --- |
| Pin | Purpose |
| PB0 | This is the far bottom left yellow light and is turned on when output contains 0x11 or any variations that contains 0x11. Light variable is responsible for triggering this pin based on FSM[S].Out1 specification |
| PB4 | This is the far bottom right yellow light and is turned on when output contains 0x11 or any variations that contains 0x11. Light variable is responsible for triggering this pin based on FSM[S].Out1 specification |
| PB1 | This is the upper left yellow light and is turned on when output contains 0x22 or any variations that contains 0x22. Light variable is responsible for triggering this pin based on FSM[S].Out1 specification |
| PB5 | This is the upper right yellow light and is turned on when output contains 0x22 or any variations that contains 0x22. Light variable is responsible for triggering this pin based on FSM[S].Out1 specification |
| PB2 | This is the left green light and is turned on when output contains 0x44 or any variations that contains 0x44. Light variable is responsible for triggering this pin based on FSM[S].Out1 specification |
| PB6 | This is the right green light and is turned on when output contains 0x44 or any variations that contains 0x44. Light variable is responsible for triggering this pin based on FSM[S].Out1 specification |
| PB3 | This is the left red light and is turned on when output contains 0x88 or any variations that contains 0x88. Light variable is responsible for triggering this pin based on FSM[S].Out1 specification |
| PB7 | This is the right red light and is turned on when output contains 0x88 or any variations that contains 0x88. Light variable is responsible for triggering this pin based on FSM[S].Out1 specification |

State Table of FSM

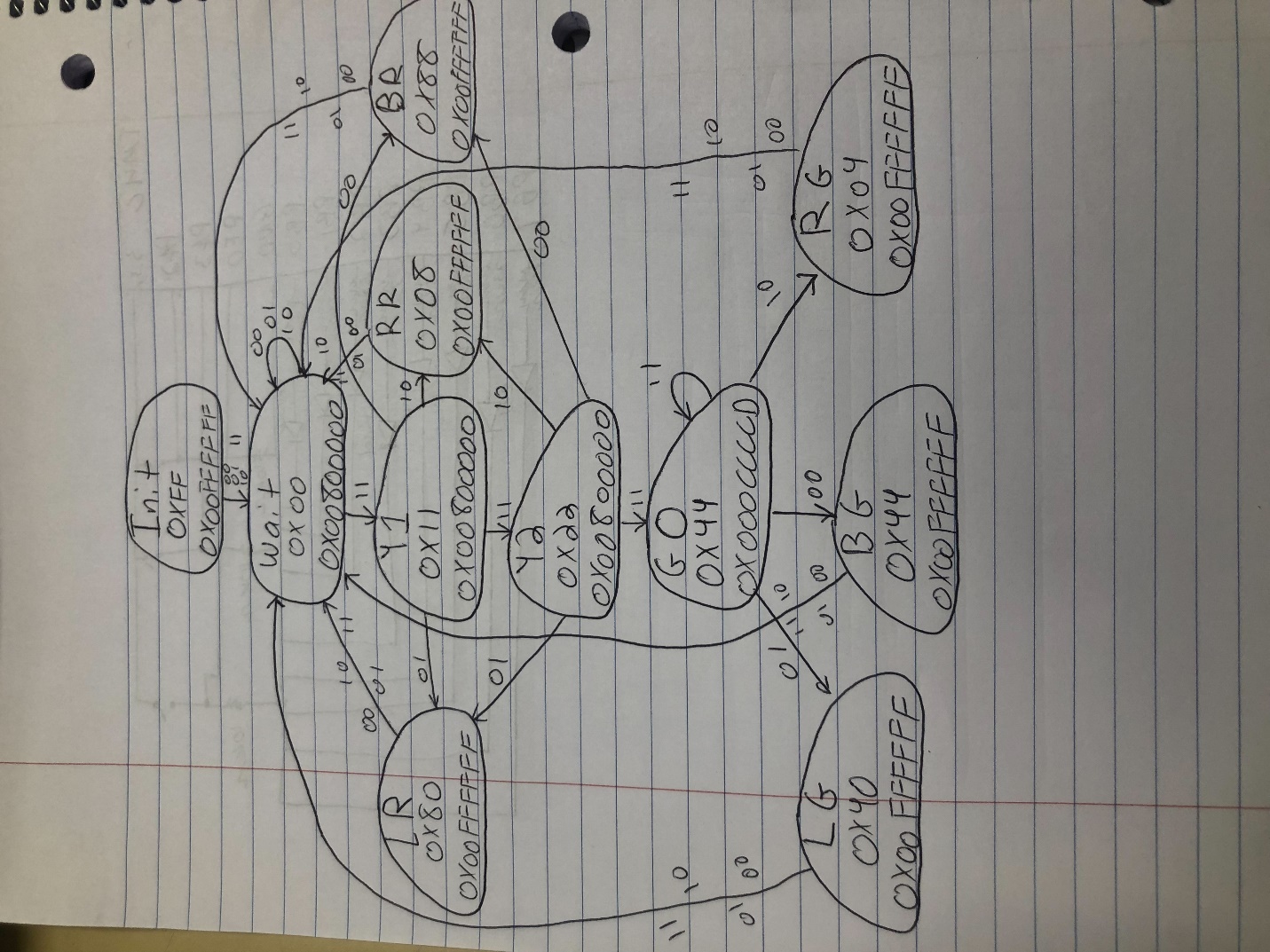
States and what is done

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Number | Name | Output | PB3-0 | PB7-4 |
| 0 | Init | 0xFF | Red, Yellow, Yellow, Green | Red, Yellow, Yellow, Green |
| 1 | Wait | 0x00 | None | None |
| 2 | Y1 | 0x11 | Yellow | Yellow |
| 3 | Y2 | 0x22 | Yellow | Yellow |
| 4 | Go | 0x44 | Green | Green |
| 5 | LR | 0x80 | Red | None |
| 6 | RR | 0x08 | None | Red |
| 7 | BR | 0x88 | Red | Red |
| 8 | LG | 0x40 | Green | None |
| 9 | RG | 0x04 | None | Green |
| 10 | BG | 0x44 | Green | Green |

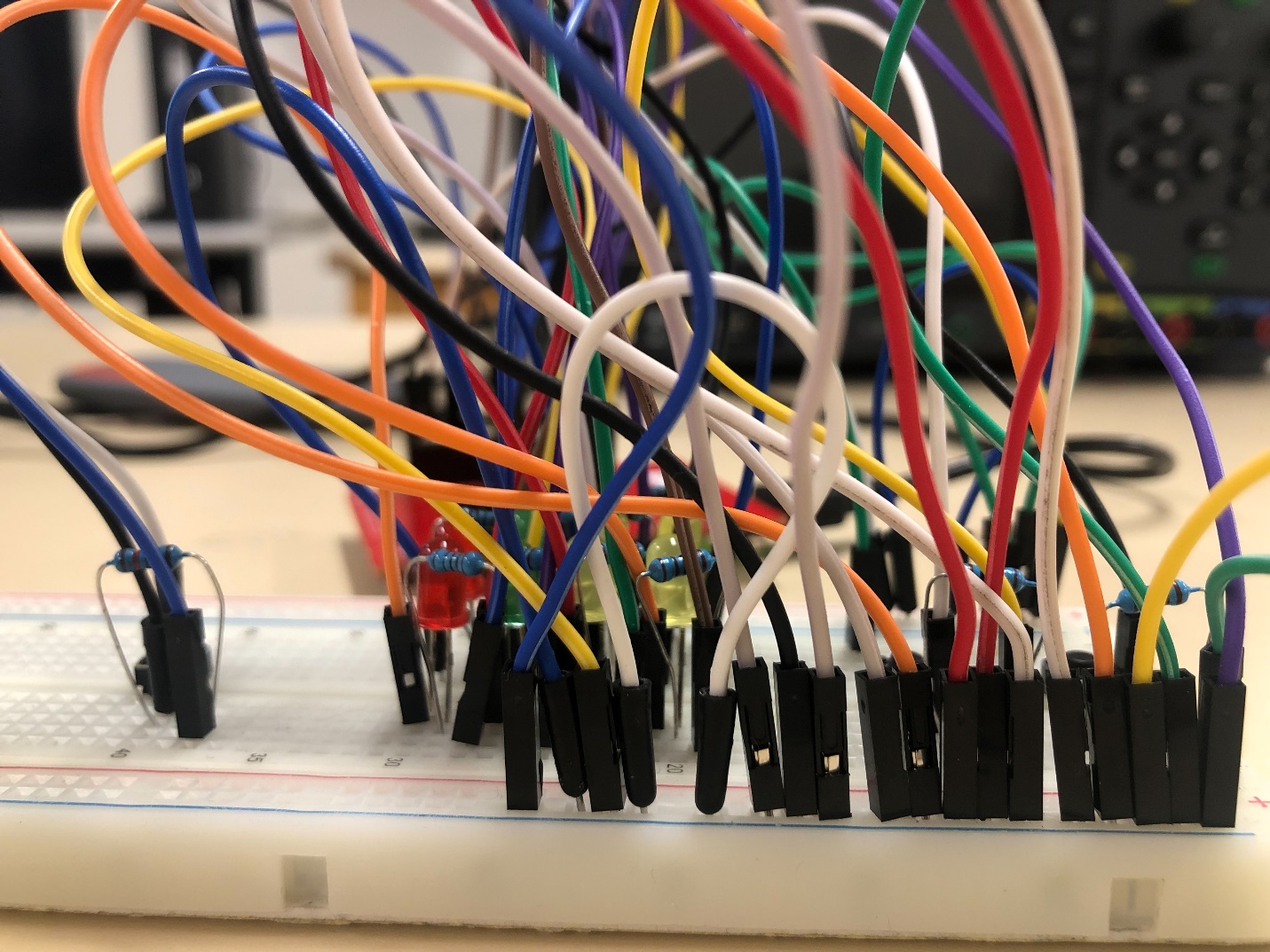
States and the inputs

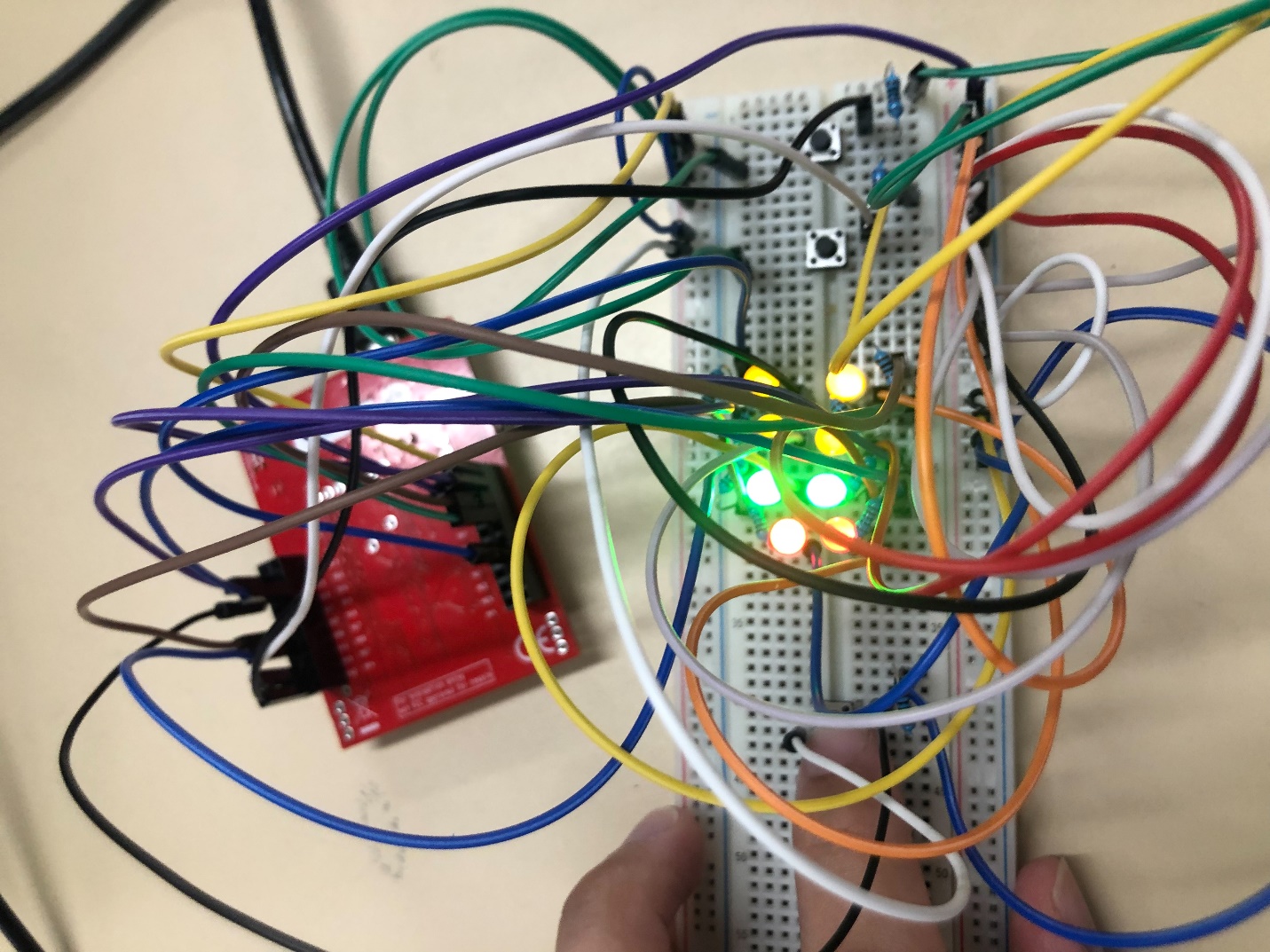
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Number | Name | Input = 0 | Input = 1 | Input = 2 | Input = 3 |
| 0 | Init | Wait | Wait | Wait | Wait |
| 1 | Wait | Wait | Wait | Wait | Y1 |
| 2 | Y1 | BR | LR | RR | Y2 |
| 3 | Y2 | BR | LR | RR | Go |
| 4 | Go | BG | LG | RG | Go |
| 5 | LR | Wait | Wait | Wait | Wait |
| 6 | RR | Wait | Wait | Wait | Wait |
| 7 | BR | Wait | Wait | Wait | Wait |
| 8 | LG | Wait | Wait | Wait | Wait |
| 9 | RG | Wait | Wait | Wait | Wait |
| 10 | BG | Wait | Wait | Wait | Wait |

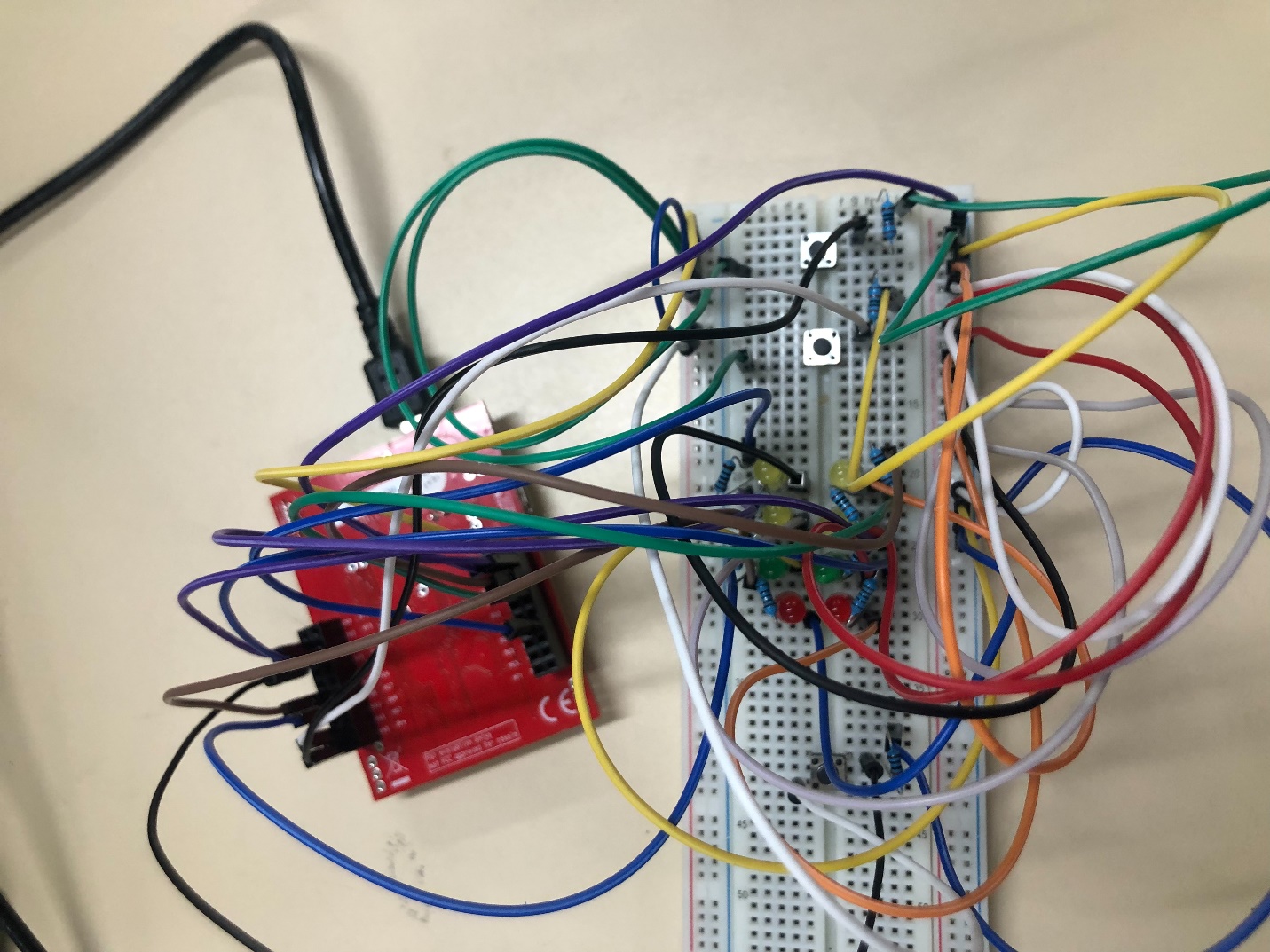
FSM Diagram



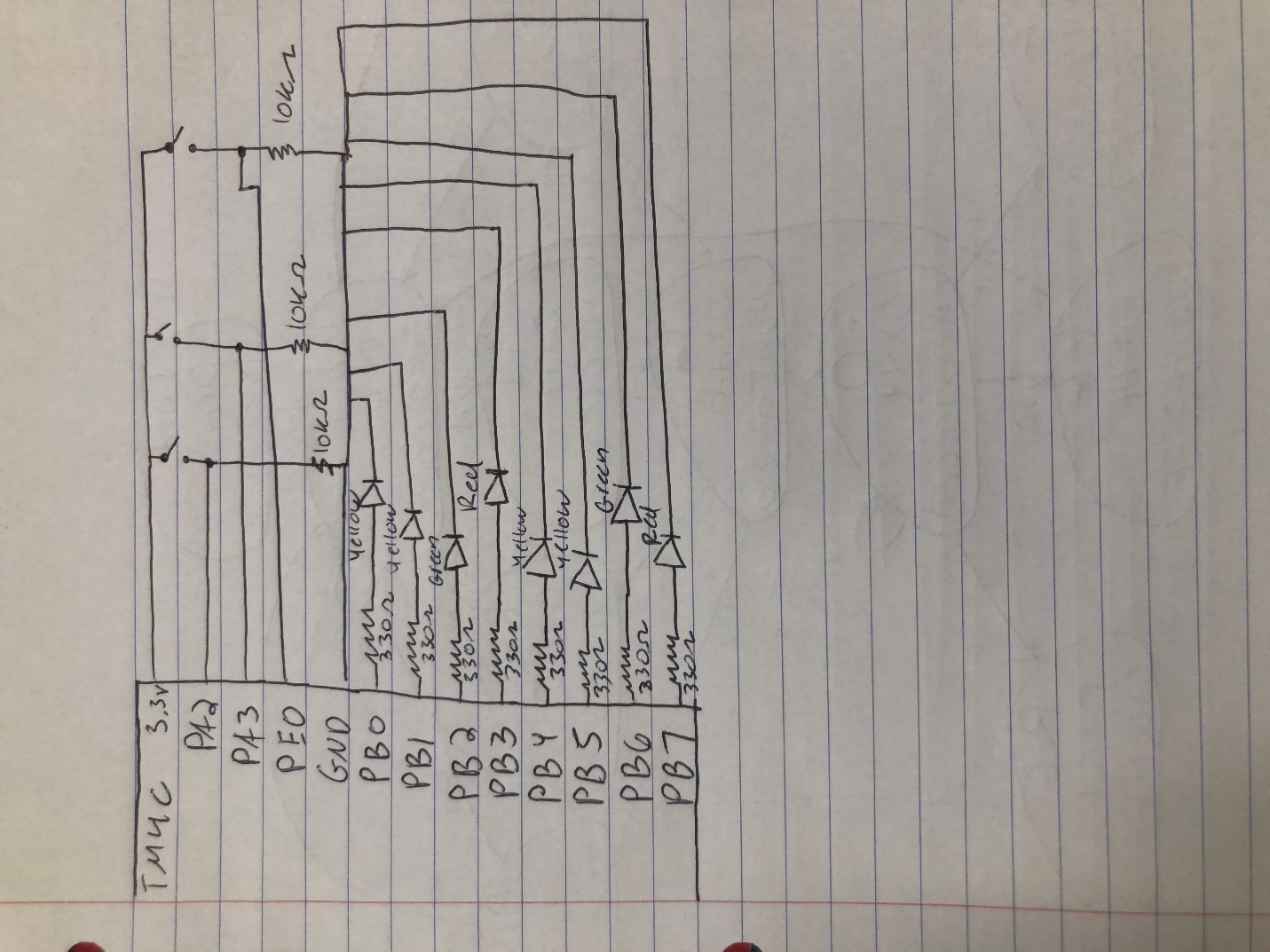
Photographs of Hardware





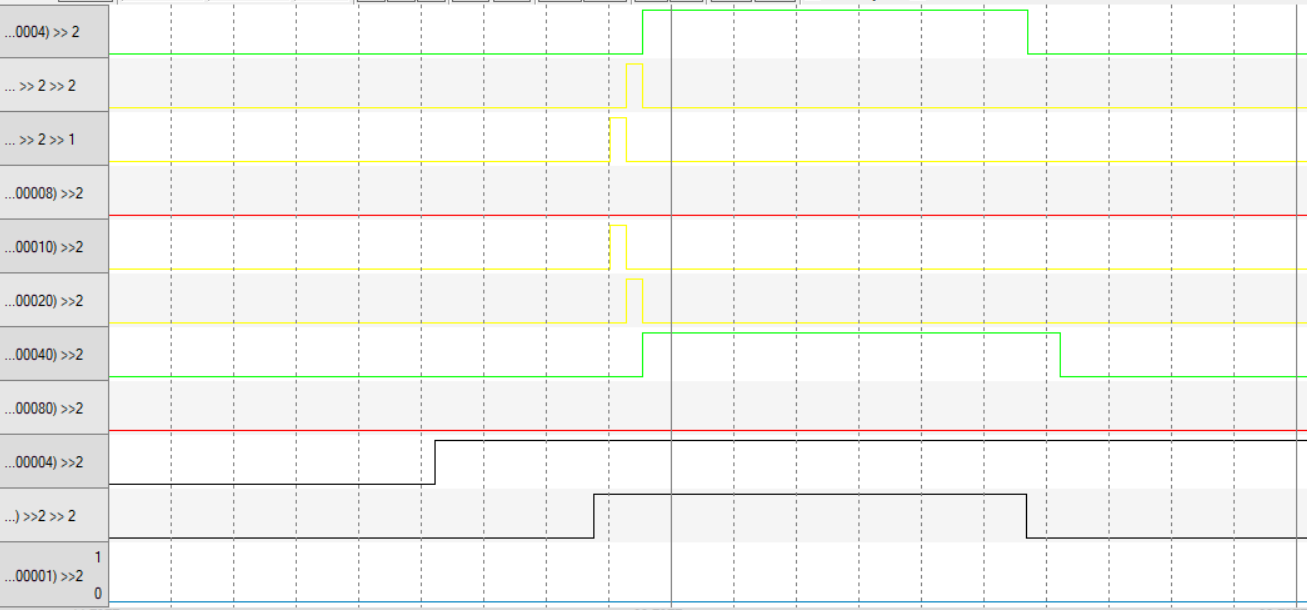


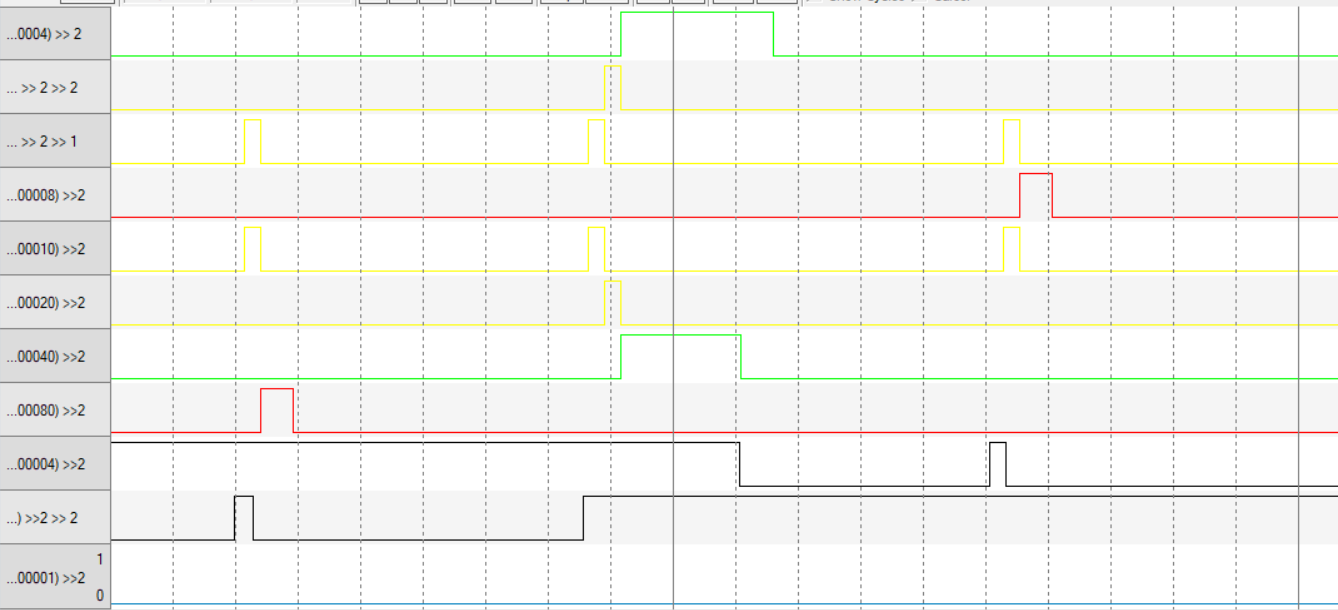
Hardware Diagram



Analyzer Photos







Source Code

//Port F

#define NVIC\_EN0\_R (\*((volatile unsigned long \*)0xE000E100)) // IRQ 0 to 31 Set Enable Register

#define NVIC\_PRI7\_R (\*((volatile unsigned long \*)0xE000E41C)) // IRQ 28 to 31 Priority Register

#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 SYSCTL\_RCGC2\_R (\*((volatile unsigned long \*)0x400FE108))

#define SYSCTL\_RCGC2\_GPIOF 0x00000020 // port F Clock Gating Control

#include "tm4c123gh6pm.h"

#include <stdint.h>

//Port F

#define GPIO\_PORTF\_DATA\_R (\*((volatile unsigned long \*)0x400253FC))

#define GPIO\_PORTF\_DIR\_R (\*((volatile unsigned long \*)0x40025400))

#define GPIO\_PORTF\_AFSEL\_R (\*((volatile unsigned long \*)0x40025420))

#define GPIO\_PORTF\_PUR\_R (\*((volatile unsigned long \*)0x40025510))

#define GPIO\_PORTF\_DEN\_R (\*((volatile unsigned long \*)0x4002551C))

#define GPIO\_PORTF\_LOCK\_R (\*((volatile unsigned long \*)0x40025520))

#define GPIO\_PORTF\_CR\_R (\*((volatile unsigned long \*)0x40025524))

#define GPIO\_PORTF\_AMSEL\_R (\*((volatile unsigned long \*)0x40025528))

#define GPIO\_PORTF\_PCTL\_R (\*((volatile unsigned long \*)0x4002552C))

#define GPIO\_LOCK\_KEY 0x4C4F434B

//Port B

#define GPIO\_PORTB\_OUT (\*((volatile unsigned long \*)0x400053FC)) // bits 7-0

#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))

//Port A

#define GPIO\_PORTA\_IN (\*((volatile unsigned long \*)0x400043FC)) // bits 4-2

#define GPIO\_PORTA\_DATA\_R (\*((volatile unsigned long \*)0x400043FC))

#define GPIO\_PORTA\_DIR\_R (\*((volatile unsigned long \*)0x40004400))

#define GPIO\_PORTA\_AFSEL\_R (\*((volatile unsigned long \*)0x40004420))

#define GPIO\_PORTA\_PUR\_R (\*((volatile unsigned long \*)0x40004510))

#define GPIO\_PORTA\_DEN\_R (\*((volatile unsigned long \*)0x4000451C))

#define GPIO\_PORTA\_LOCK\_R (\*((volatile unsigned long \*)0x40004520))

#define GPIO\_PORTA\_CR\_R (\*((volatile unsigned long \*)0x40004524))

#define GPIO\_PORTA\_AMSEL\_R (\*((volatile unsigned long \*)0x40004528))

#define GPIO\_PORTA\_PCTL\_R (\*((volatile unsigned long \*)0x4000452C))

#define GPIO\_PORTA\_ODR\_R (\*((volatile unsigned long \*)0x4000450C))

#define GPIO\_PORTA\_IS\_R (\*((volatile unsigned long \*)0x40004404))

#define GPIO\_PORTA\_IBE\_R (\*((volatile unsigned long \*)0x40004408))

#define GPIO\_PORTA\_ICR\_R (\*((volatile unsigned long \*)0x4000441C))

#define GPIO\_PORTA\_IM\_R (\*((volatile unsigned long \*)0x40004410))

#define NVIC\_PRI0\_R (\*((volatile unsigned long \*)0xE000E400))

//Port E

#define GPIO\_PORTE\_DATA\_R (\*((volatile unsigned long \*)0x400243FC))

#define GPIO\_PORTE\_DIR\_R (\*((volatile unsigned long \*)0x40024400))

#define GPIO\_PORTE\_AFSEL\_R (\*((volatile unsigned long \*)0x40024420))

#define GPIO\_PORTE\_PUR\_R (\*((volatile unsigned long \*)0x40024510))

#define GPIO\_PORTE\_DEN\_R (\*((volatile unsigned long \*)0x4002451C))

#define GPIO\_PORTE\_LOCK\_R (\*((volatile unsigned long \*)0x40024520))

#define GPIO\_PORTE\_CR\_R (\*((volatile unsigned long \*)0x40024524))

#define GPIO\_PORTE\_AMSEL\_R (\*((volatile unsigned long \*)0x40024528))

#define GPIO\_PORTE\_PCTL\_R (\*((volatile unsigned long \*)0x4002452C))

#define GPIO\_PORTE\_ODR\_R (\*((volatile unsigned long \*)0x4002450C))

#define GPIO\_PORTE\_IS\_R (\*((volatile unsigned long \*)0x40024404))

#define GPIO\_PORTE\_IBE\_R (\*((volatile unsigned long \*)0x40024408))

#define GPIO\_PORTE\_ICR\_R (\*((volatile unsigned long \*)0x4002441C))

#define GPIO\_PORTE\_IM\_R (\*((volatile unsigned long \*)0x40024410))

#define NVIC\_PRI1\_R (\*((volatile unsigned long \*)0xE000E404))

//SysTick Interrupts

#define NVIC\_SYS\_PRI3\_R (\*((volatile unsigned long \*)0xE000ED20))

#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 NVIC\_EN0\_R (\*((volatile unsigned long \*)0xE000E100))

#define SYSCTL\_RCGC2\_R (\*((volatile unsigned long \*)0x400FE108))

#define SENSOR (\*((volatile unsigned long \*)0x40004030)) // Port A for buttons

#define RSENSOR (\*((volatile unsigned long \*)0x40024004))

#define RESET (\*((volatile unsigned long \*)0x40025004)) // Port F for buttons

#define LIGHT (\*((volatile unsigned long \*)0x400053FC)) // Port B for LEDs

#define SYSCTL\_RCGC2\_GPIOB 0x00000002 // port B Clock Gating Control

#define SYSCTL\_RCGC2\_GPIOA 0x00000001 // port A Clock Gating Control

#define SYSCTL\_RCGC2\_GPIOF 0x00000020 // port F Clock Gating Control

#define SYSCTL\_RCGC2\_GPIOE 0x00000010 // port F Clock Gating Control

void DisableInterrupts(void); // Disable interrupts

void EnableInterrupts(void); // Enable interrupts

long StartCritical (void); // previous I bit, disable interrupts

void EndCritical(long sr); // restore I bit to previous value

void WaitForInterrupt(void); // low power mode

void Ports\_Init(void);

void SysTick\_Init(unsigned long period);

void SysTick\_Handler (void);

void GPIOPortA\_Handler(void);

void GPIOPortE\_Handler(void);

// global variable visible in Watch window of debugger

// increments at least once per button press

struct State {

unsigned long Out1;

unsigned long Time;

unsigned long Next[4];};

typedef const struct State STyp;

#define Init 0

#define Wait 1

#define Y1 2

#define Y2 3

#define Go 4

#define LR 5

#define RR 6

#define BR 7

#define LG 8

#define RG 9

#define BG 10

STyp FSM[11]={

{0xFF,0x00FFFFFF,{Wait,Wait,Wait,Wait}},

{0x00,0x00800000,{Wait,Wait,Wait,Y1}},

{0x11,0x00800000,{BR,LR,RR,Y2}},

{0x22,0x00800000,{BR,LR,RR,Go}},

{0x44,0x000CCCCD,{BG,LG,RG,Go}},

{0x80,0x00FFFFFF,{Wait,Wait,Wait,Wait}},

{0x08,0x00FFFFFF,{Wait,Wait,Wait,Wait}},

{0x88,0x00FFFFFF,{Wait,Wait,Wait,Wait}},

{0x40,0x00FFFFFF,{Wait,Wait,Wait,Wait}},

{0x04,0x00FFFFFF,{Wait,Wait,Wait,Wait}},

{0x44,0x00FFFFFF,{Wait,Wait,Wait,Wait}}

};

unsigned long S; // index to the current state

unsigned long Input;

unsigned long In1;

void EdgeCounter\_Init(void){

DisableInterrupts();//EnableInterrupts(); // (i) Clears the I bit

//Port E

GPIO\_PORTE\_IS\_R &= ~0x01; // (d) PF4 is edge-sensitive

GPIO\_PORTE\_IBE\_R &= ~0x01; // PF4 is not both edges

GPIO\_PORTE\_ICR\_R = 0x01; // (e) clear flag

GPIO\_PORTE\_IM\_R |= 0x01; // (f) arm interrupt on PE1

NVIC\_PRI1\_R = (NVIC\_PRI7\_R&0xFF00FFFF)|0x00200000; // (g) priority 1

//Port A

GPIO\_PORTA\_IS\_R &= ~0x0C; // (d) PF4 is edge-sensitive

GPIO\_PORTA\_IBE\_R &= 0x0C; // PF4 is not both edges

GPIO\_PORTA\_ICR\_R = 0x0C; // (e) clear flag

GPIO\_PORTA\_IM\_R |= 0x0C; // (f) arm interrupt on PA2-3

NVIC\_PRI1\_R = (NVIC\_PRI7\_R&0xFF00FFFF)|0x00400000; // (g) priority 2

NVIC\_EN0\_R |= 0x00000011;

EnableInterrupts();

}

//debug code

int main(void){

Ports\_Init();

EdgeCounter\_Init();

S = Init;

LIGHT = FSM[S].Out1;

SysTick\_Init(FSM[S].Time);

while(1)

{

WaitForInterrupt();

}

}

void Ports\_Init(void){

volatile unsigned long delay;

SYSCTL\_RCGC2\_R = 0x33; // 1) A B E F

delay = SYSCTL\_RCGC2\_R; // 2) no need to unlock

//Port A

GPIO\_PORTA\_DATA\_R = 0x400043FC;

GPIO\_PORTA\_CR\_R = 0x1C;

GPIO\_PORTA\_AMSEL\_R &= 0x00; // 3) disable analog function on PA3-2

GPIO\_PORTA\_PCTL\_R &= 0x00000000; // 4) enable regular GPIO

GPIO\_PORTA\_DIR\_R &= 0x00; // 5) inputs on PA3-2

GPIO\_PORTA\_AFSEL\_R &= 0x00; // 6) regular function on PA3-2

GPIO\_PORTA\_DEN\_R |= 0x0C; // 7) enable digital on PA3-2

GPIO\_PORTA\_ODR\_R |= 0x38;

//Port F

GPIO\_PORTF\_LOCK\_R = 0x4C4F434B; // 2) unlock GPIO Port F

GPIO\_PORTF\_CR\_R = 0x1F; // allow changes to PF4-0

GPIO\_PORTF\_AMSEL\_R = 0x00; // 3) disable analog on PF

GPIO\_PORTF\_PCTL\_R = 0x00000000; // 4) PCTL GPIO on PF4-0

GPIO\_PORTF\_DIR\_R = 0x0E; // 5) PF4,PF0 in, PF3-1 out

GPIO\_PORTF\_AFSEL\_R = 0x00; // 6) disable alt funct on PF7-0

GPIO\_PORTF\_PUR\_R = 0x11; // enable pull-up on PF0 and PF4

GPIO\_PORTF\_DEN\_R = 0x1F; // 7) enable digital I/O on PF4-0

//Port B

GPIO\_PORTB\_AMSEL\_R &= ~0xFF;

GPIO\_PORTB\_PCTL\_R &= ~0x00FFFFFF; // 4) enable regular GPIO

GPIO\_PORTB\_DIR\_R |= 0xFF; // 5) outputs on PB5-0

GPIO\_PORTB\_AFSEL\_R &= ~0xFF; // 6) regular function on PB5-0

GPIO\_PORTB\_DEN\_R |= 0xFF; // 7) enable digital on PB5-0

//Port E

GPIO\_PORTE\_CR\_R |= 0x01;

GPIO\_PORTE\_AMSEL\_R &= ~0x01; // 3) disable analog function on PE1

GPIO\_PORTE\_PCTL\_R &= 0x00000000; // 4) enable regular GPIO

GPIO\_PORTE\_DIR\_R &= ~0x01; // 5) inputs on PE1

GPIO\_PORTE\_AFSEL\_R &= ~0x01; // 6) regular function on PE1

GPIO\_PORTE\_DEN\_R |= 0x01; // 7) enable digital on PE1

}

void SysTick\_Init(unsigned long period) {

NVIC\_ST\_CTRL\_R = 0; //disable SysTick during setup

NVIC\_ST\_RELOAD\_R = period-1; //reload value

NVIC\_ST\_CURRENT\_R = 0; //any write to current clears it

NVIC\_SYS\_PRI3\_R = (NVIC\_SYS\_PRI3\_R&0x00FFFFFF) | 0x60000000; // priority 3

NVIC\_ST\_CTRL\_R = 0x07;

EnableInterrupts();

}

void SysTick\_Handler() {

In1 = SENSOR >> 2; // read sensors

Input = In1;

S = FSM[S].Next[Input];

LIGHT = FSM[S].Out1; // set lights

SysTick\_Init(FSM[S].Time);

}

void GPIOPortA\_Handler()

{

GPIO\_PORTA\_ICR\_R = 0x0C;

}

void GPIOPortE\_Handler()

{

GPIO\_PORTE\_ICR\_R = 0x01;

S = Init;

LIGHT = FSM[S].Out1;

SysTick\_Init(FSM[S].Time);

}