PROJECT REPORT
TRAFFIC CONTROL UNIT

Student Name	***Student ID***
1. Abdullah Tariq	301297687
2. Omar Arshad	301292865
3. Syed Imad Azeem Rizvi	301297227

Group # 4

Report due date: 8th Apr 19

MSE – 450 REAL TIME AND EMBEDDED SYSTEMS
SCHOOL OF MECHATRONIC SYSTEMS ENGINEERING
SIMON FRASER UNIVERSITY

ABSTRACT

Growing number of road users and the limited resources provided by current infrastructures lead to ever increasing traveling times. Traffic in a city is very much affected by traffic light controllers. When waiting for a traffic light, the driver loses time, manpower, increases air pollution and the car waste fuel.

This project report is a solution to the problem caused by the existing system in the state. Some of these problems will be highlighted in this report and analysis of the intelligent traffic signal light control system will be made. This project report highlights the procedure to design and implement a robust traffic light control system capable of functioning with human inputs with a two way traffic flow as they reduce costs and prevent major accidents. We have used TM4C123GH6PM Tiva C series microcontroller as a hardware device/peripheral to control traffic flow. Code Composer Studio software is used to carry out our programming requirements. A complex circuitry was built on breadboard using digital logic to represent a simulation model for traffic control.

Table of Contents

ABSTRACT 1
INTRODUCTION 3
DESIGN GOALS 3
THEORY 4
METHODS 5
  Stage 1 5
  Stage 2 5
  Stage 3 5
  Stage 4 5
  Stage 5 5
IMPLEMENTATION & RESULTS 6
  Traffic Junction 6
  Pedestrian Crossing 6
CONCLUSION 7
APPENDIX 8

Figure 1 Final Circuit Design 4

INTRODUCTION

Purpose of this project is to get hand on experience with microcontroller and to implement it in broader variety. Project we chose is traffic light control, this project can help us develop a fundamental understanding of how a microprocessor based controller can be used to control a continuous process in a system.

Problem related to this traffic control system is that there are different systems within one whole system. So you can’t start with building everything from the beginning you need to ease your way in. Another problem which we had encounter is to observe how signals were interrupted between their sequence once a car approach a traffic light and how their period of traffic light alters with the frequency of traffic flow at an intersection.

In order to design fully working traffic light control system we first investigated different aspects of a control environment and combined them together to form a system. However to be more efficient with our design we have taken an initial start with simple model of lights. Our next stage is to make our system two way traffic control and then enhance it further by making it a four way traffic control. Once this point was reached successfully we then added complexity like increase and decrease in flow of traffic and pedestrian crossing.

DESIGN GOALS

The design process of the traffic light system will follow the procedure similar to that of a modern traffic control system. Due to the shortage of traffic light unit, we have constructed a traffic control unit model by sourcing and using the equipment list mentioned below,

1. Tiva C Series ARM Cortex M4 Microcontroller TM4C123GH6PM
2. Circuit Breadboard
3. LEDs. (Color: - Red, Yellow, Green)
4. 7 Segment
5. Push buttons
6. Resistors
7. 7447 Decoder IC
8. Electrical wires

These were the equipment required to implement our final design. The final structure of our traffic light unit is shown in figure 1.

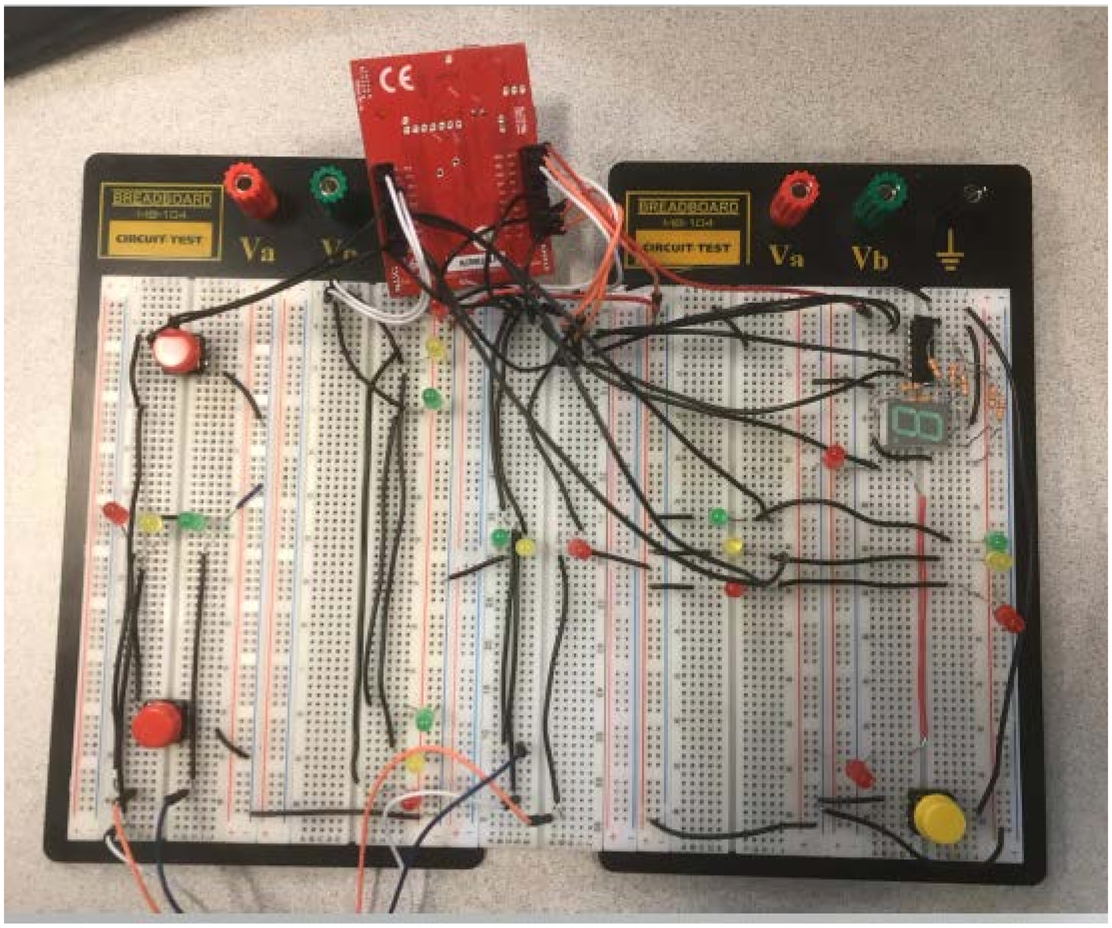

Figure 1 Final Circuit Design

The design objectives are listed below.

1. To understand the structure and operation of a microcontroller
2. To study the assembly language design and their programming technique
3. To understand how to make the interfacing to the microcontroller
4. To design a program that works together with a model of three- junction traffic light and sensors.

To build the model of three-junctions of intelligent traffic light that can overcome some of major problem of current traffic light.

THEORY

Traffic signal optimization can be framed as a control problem. However, application of control theory can vary based on the input parameters and the traffic parameters being controlled. Classical control theory involves a closed-loop system with sensors and feedback. The feedback is used to control states or outputs; input parameters (e.g., the number of cars waiting at a red light) have a direct effect on control parameters (e.g., adjusting traffic light cycle time). The input parameters are measured with sensors, and are processed by the controller. The result (the control signal) is used as input to the process, thereby closing the loop. In an open-loop system, there is no sensing or feedback from output, which makes it less robust than a closed-loop system. An example of an open-loop traffic light system is one in which historical traffic patterns are used to set the timing parameters; as long as the traffic flow follows the historical trend, the system performs well. However, variations from the norm, which can be expected, can lead to significant performance deterioration. A closed-loop control system will typically use both historical traffic data and real-time traffic conditions to make decisions.

METHODS

We followed the methods and procedures mentioned in the lab manual of the traffic control unit. There were five stages of our project.

Stage 1: - This was the basic initiative we took to setup the timing sequence of our traffic lights. We monitored the sequence of the traffic light for N/S direction and minimize our time for the ease of simulation however in real world traffic lights run with a long time cycle.

Stage 2: - In this part we added the E/W direction to the previous part and then adjusted there led sequence with respect to each other. For example, if green led is ON for N/S direction then red led is ON for E/W direction. We then added a delay when there is a switch in the direction to add safety.

Stage 3: - Here we considered the fact of traffic flow in order to optimize our code and make our system more efficient in terms of controlling real-time situation. We add 2 counters for each direction to Inc. /Dec. the traffic flow.

Stage 4: - The counters we added in the previous part were used to alter the timing of led according to the traffic flow. For example a certain amount of seconds were added to the green light when a certain amount of vehicle have passed the signal, however this was dependent to the number of vehicles waiting on the red signal direction. We put 2 push buttons in our circuit to call the interrupt function that was initialized in our code to deal with traffic flow.

Stage 5: - This stage is the final stage of our project. We added the pedestrian crossover to our circuit and implemented it in our code by using an interrupt function. We observe that this sequence also includes two 'Safety Red' sequence steps where both sets of red lights are on at the same time and also observe while the pedestrian crossing is not in use, the pedestrian traffic lights must be on green allowing the traffic to flow and the pedestrian light must be on red preventing the pedestrians from crossing.

The above mentioned stages were implemented in their respective order.

IMPLEMENTATION & RESULTS

Traffic Junction

To implement a closed loop system that will consider the real time condition and the previous state to adjust the cycle time for the traffic lights, we used the following strategy in our program. We first create to function for the traffic lights in the direction of North/South and East/West. These function were used to a fixed initial stage of the traffic lights. We then uses an interrupt service routine, initialize on one of the GPIO Ports of a microcontroller, to consider the situation of increase and decrease in traffic flow at the intersection. For this we use a counter variable to consider the flow in each of the two direction and by this we were able to adjust time period for the respective led to improve our system. As an example, if there is a high flow of traffic in N/S direction compare to E/W direction, the green led will stay on for a longer period of time. By adding these counter and using the concept of interrupt we were able to improvise an open loop traffic system to a closed loop system. Figure 2 shows the final junction traffic control logic.

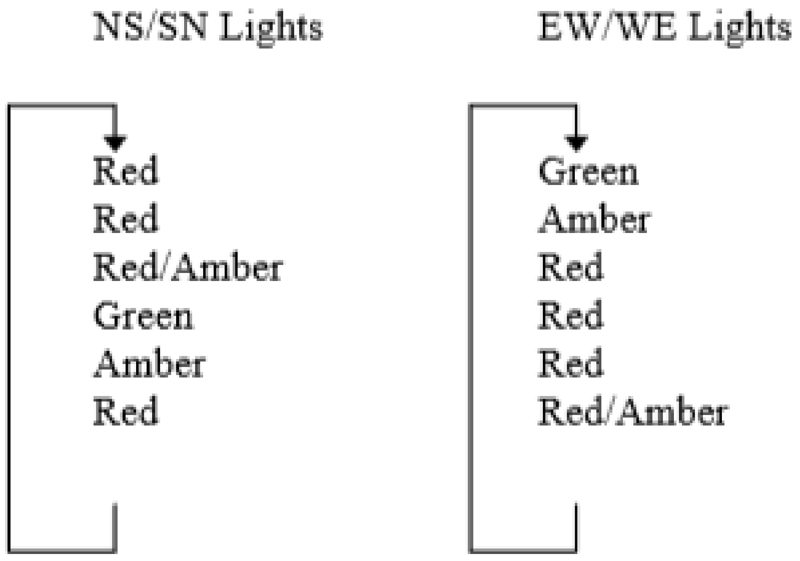

Figure 2: final junction traffic control logic

Pedestrian Crossing

This part of the project was the trickiest to solve. For a pedestrian to cross, there signal should be green with a countdown of 10 seconds. The problem we faced is to simultaneously run the countdown for the pedestrian while not disturbing the sequence of the traffic light signal at intersection. For this we use SYSTICK INTERRUPT HANDLER. This allows us, in our code to run it parallel with the main function while not disturbing the timing sequence of the traffic lights. Figure 3 shows final pedestrian traffic control logic.

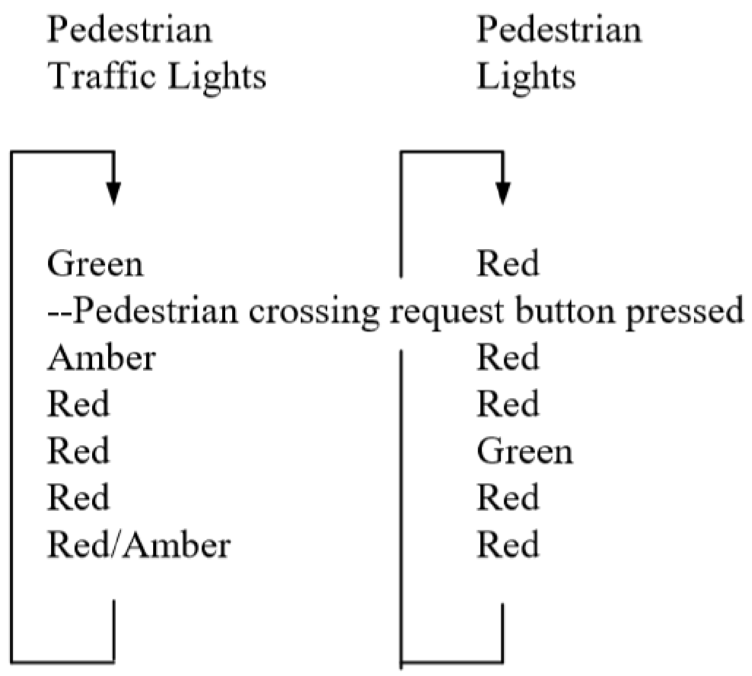
Pedestrian Traffic Lights
Pedestrian Lights
Green
Red
--Pedestrian crossing request button pressed
Amber
Red
Red
Red
Red
Green
Red
Red
Red/Amber
Red

Figure 3: final pedestrian traffic control logic

CONCLUSION

Lectures and lab in this course have helped a lot in this project. The project improved our understanding of a basic closed loop control system and embedded system programming. Programming of microcontroller can only be done by referring to its datasheet. This course project enhance our ability to analyze it's datasheet of how to read specific values, addresses to give command for unlocking ports, registers of the microcontroller. We managed to build and program a fully functioning traffic light control system.

APPENDIX

```
#include <stdint.h>
#include <stdbool.h>
#include <stdlib.h>
#include "SysTickInts.h"
#include "PLL.h"
#include "tm4c123gh6pm.h"

//************************GPIO PORTS**************************//
void PortA_Init(void);
void PortB_Init(void);
void PortC_Init(void);
void PortE_Init(void);
void EdgecounterEW_Init(void);    // PortF
void EdgecounterEW_Init2(void);   // PortD

//************************Functions**************************//
void NS_ON_EW_OFF(void);
void NS_OFF_EW_ON(void);
void wait(int value);

//************************Interrupts**************************//
void disable_interrupts(void);
void enable_interrupts(void);
void wait_for_interrupts(void);
void IntDefaultHandler1(void);
void IntDefaultHandler2(void);
void ab(void);

//************************Variables**************************//
volatile int counterNS = 0;
volatile int counterEW = 0;
volatile int pass_count = 0;

//************************MAIN FUNCTION**************************//
int main(void){
   PLL_Init();          // bus clock at 80 MHz
   EdgecounterEW_Init(); // initialize GPIO Port F interrupt
   EdgecounterEW_Init2();// initialize GPIO Port D interrupt
   PortA_Init();
   PortB_Init();
   PortC_Init();
   PortE_Init();

   // Initial Conditions
   pass_count = 0; //set passenger counter
   GPIO_PORTE_DATA_R =0x08; // Traffic ON, Green light E/W
   GPIO_PORTB_DATA_R =0x02; // Traffic OFF, Red light N/S
   GPIO_PORTA_DATA_R =~(0x00); // Red light for pedestrain
   GPIO_PORTC_DATA_R =0x10; // Traffic OFF, Red light at pedestrain signal

   while(true){
      if(counterNS==0 && counterEW==0)
      {
         NS_ON_EW_OFF();
         wait(counterEW);
         NS_OFF_EW_ON();
         wait(counterNS);
      }
      else if((counterNS!=0 && counterEW==0) || (counterNS==0 && counterEW!=0) || (counterNS!=0 && counterEW!=0))
      {
         NS_ON_EW_OFF();
         wait(counterEW);
         NS_OFF_EW_ON();
```

```
            wait(counterNS);

            counterNS=0; //Traffic flow reset at N/S
            counterEW=0; //Traffic flow reset at E/W
        }
    }
}

//*************************GPIO PORTS INITIALIZATION FUNCTIONS***************************//

void PortA_Init(void){
    volatile unsigned long delay;
    SYSCTL_RCGC2_R |= 0x00000001;     // 1) A clock
    delay = SYSCTL_RCGC2_R;           // delay
    GPIO_PORTA_CR_R = 0xFF;           // allow changes to PA7-0
    GPIO_PORTA_AMSEL_R = 0x00;        // 3) disable analog function
    GPIO_PORTA_PCTL_R = 0x00000000;   // 4) GPIO clear bit PCTL
    //GPIO_PORTE_DIR_R |= 0x08;       // 5) PA3 input
    GPIO_PORTA_DIR_R = 0xFF;          // 5) PA7-0 output
    GPIO_PORTA_AFSEL_R = 0x00;        // 6) no alternate function
    // GPIO_PORTE_PUR_R = 0x08;       // enable pullup resistors on PA3
    GPIO_PORTA_DEN_R = 0xFF;          // 7) enable digital pins PA7-PA0
}

void PortB_Init(void){
    volatile unsigned long delay;
    SYSCTL_RCGC2_R |= 0x00000002;     // 1) B clock
    delay = SYSCTL_RCGC2_R;           // delay
    GPIO_PORTB_CR_R = 0x2F;           // allow changes to PB6,3-0
    GPIO_PORTB_AMSEL_R = 0x00;        // 3) disable analog function
    GPIO_PORTB_PCTL_R = 0x00000000;   // 4) GPIO clear bit PCTL
    //GPIO_PORTE_DIR_R |= 0x08;       // 5) PB3 input
    GPIO_PORTB_DIR_R = 0x0E;          // 5) PB3,PB2,PB1 output
    GPIO_PORTB_AFSEL_R = 0x00;        // 6) no alternate function
    // GPIO_PORTE_PUR_R = 0x08;       // enable pullup resistors on PB3
    GPIO_PORTB_DEN_R = 0x0F;          // 7) enable digital pins PB3-PB0
}

void PortE_Init(void){
    volatile unsigned long delay;
    SYSCTL_RCGC2_R |= 0x00000010;     // 1) E clock
    delay = SYSCTL_RCGC2_R;           // delay
    GPIO_PORTE_CR_R = 0x2F;           // allow changes to PE5,3-0
    GPIO_PORTE_AMSEL_R = 0x00;        // 3) disable analog function
    //GPIO_PORTE_PCTL_R = 0x00000000; // 4) GPIO clear bit PCTL
    //GPIO_PORTE_DIR_R |= 0x08;       // 5) PE3 input
    GPIO_PORTE_DIR_R = 0xFF;          // 5) PE2,PE1,PE0 output
    GPIO_PORTE_AFSEL_R = 0x00;        // 6) no alternate function
    //GPIO_PORTE_PUR_R = 0x08;        // enable pullup resistors on PE3
    GPIO_PORTE_DEN_R = 0x0F;          // 7) enable digital pins PE3-PE0
}


void PortC_Init(void){
    volatile unsigned long delay;
    SYSCTL_RCGC2_R |= 0x00000004;     // 1) C clock
    while ((SYSCTL_PRGPIO_R & 0x00000004) == 0)
    {};
    delay = SYSCTL_RCGC2_R;           // delay
    GPIO_PORTC_CR_R = 0xF0;           // allow changes to PC7-4
    GPIO_PORTC_AMSEL_R = 0x00;        // 3) disable analog function
    // GPIO_PORTC_PCTL_R = 0x1;       // 4) GPIO clear bit PCTL
    //GPIO_PORTE_DIR_R |= 0x08;       // 5) PC3 input
    GPIO_PORTC_DIR_R = 0xF0;          // 5) PC3,PC2,PC1 output
    GPIO_PORTC_AFSEL_R = 0x00;        // 6) no alternate function
    //GPIO_PORTC_PUR_R = 0x0F;        // enable pullup resistors on PC3
```

```
  GPIO_PORTC_DEN_R = 0xF0;         // 7) enable digital pins PC7-PC4
}

void EdgecounterEW_Init(void) {
  SYSCTL_RCGC2_R |= 0x00000020;          // activate clock for PortF
  while ((SYSCTL_PRGPIO_R & 0x00000020) == 0)
  {};                        // wait until PortF is ready
  GPIO_PORTF_LOCK_R = 0x4C4F434B;        // unlock GPIO PortF
  GPIO_PORTF_CR_R = 0x1F;               // allow changes to PF4-0
  GPIO_PORTF_AMSEL_R = 0x00;            // disable analog on PortF
  GPIO_PORTF_PCTL_R = 0x00000000;        // use PF4-0 as GPIO
  GPIO_PORTF_DIR_R = 0x06;              // PF4,PF0 in, PF3-1 out
  GPIO_PORTF_AFSEL_R = 0x00;            // disable alt function on PF
  GPIO_PORTF_PUR_R = 0x11;              // enable pull-up on PF0,PF4
  GPIO_PORTF_DEN_R = 0x1F;              // enable digital I/O on PF4-0

  GPIO_PORTF_IS_R=0x00;
  GPIO_PORTF_IBE_R=0x00;
  GPIO_PORTF_IEV_R =0x00;
  //GPIO_PORTF_IM_R=0xFF;
  NVIC_EN0_R |=0x40000000;
  GPIO_PORTF_IM_R=0x11;
  //  GPIO_PORTF_IM_R=0x01;
}

void EdgecounterEW_Init2(void){

  SYSCTL_RCGC2_R |= 0x00000008;          // (a) activate clock for port F
  counterNS = 0;                         // (b) initialize pass_count and wait for clock
  GPIO_PORTD_DIR_R &= ~0x01;            // (c) make PF4 in (built-in button)
  GPIO_PORTD_AFSEL_R &= ~0x01;          //     disable alt funct on PF4
  GPIO_PORTD_DEN_R |= 0x01;             //     enable digital I/O on PF4
  // GPIO_PORTE_PCTL_R &= ~0x000F0000;   //  configure PF4 as GPIO
  GPIO_PORTD_AMSEL_R &= ~0x01;          //    disable analog functionality on PF4
  GPIO_PORTD_PUR_R |= 0x01;             //     enable weak pull-up on PF4
  GPIO_PORTD_IS_R &= ~0x01;             // (d) PF4 is edge-sensitive
  GPIO_PORTD_IBE_R &= ~0x01;            //     PF4 is not both edges
  GPIO_PORTD_IEV_R &= ~0x01;            //     PF4 falling edge event
  GPIO_PORTD_ICR_R = 0x01;              // (e) clear flag4
  GPIO_PORTD_IM_R |= 0x01;              // (f) arm interrupt on PF4
  NVIC_PRI0_R = (NVIC_PRI0_R&0xFF00FFFF)|0xA0000000; // (g) priority 5
  NVIC_EN0_R |= 0x8;                   // (h) enable interrupt 30 in NVIC
  enable_interrupts();                 // (i) Enable global Interrupt flag (I)
}

//*************************INTERRUPT FUNCTIONS***************************//

/* Disable interrupts by setting the I bit in the PRIMASK system register */
void disable_interrupts(void) {
  __asm("    CPSID  I\n"
        "    BX     LR");
}

/* Enable interrupts by clearing the I bit in the PRIMASK system register */
void enable_interrupts(void) {
  __asm("    CPSIE  I\n"
        "    BX     LR");
}

/* Enter low-power mode while waiting for interrupts */
void wait_for_interrupts(void) {
  __asm("    WFI\n"
        "    BX     LR");
}

/* Interrupt service routine for SysTick Interrupt */
```

```
// Executed every 12.5ns*(period)
void SysTick_Handler(void){
   pass_count++;
   if (pass_count >= 7) {
      GPIO_PORTC_DATA_R = 0x20;
   }
   if (pass_count >= 14) {
      GPIO_PORTC_DATA_R = 0x40;
   }
   if (pass_count >= 21) {
      GPIO_PORTA_DATA_R = 0x90;
   }
   if (pass_count >= 28) {
      GPIO_PORTA_DATA_R = 0x80;
   }
   if (pass_count >= 35) {
      GPIO_PORTA_DATA_R = 0x70;
   }
   if (pass_count >= 42) {
      GPIO_PORTA_DATA_R = 0x60;
   }
   if (pass_count >= 49) {
      GPIO_PORTA_DATA_R = 0x50;
   }
   if (pass_count >= 56) {
      GPIO_PORTA_DATA_R = 0x40;
   }
   if (pass_count >= 63) {
      GPIO_PORTA_DATA_R = 0x30;
   }
   if (pass_count >= 70) {
      GPIO_PORTA_DATA_R = 0x20;
   }
   if (pass_count >= 77) {
      GPIO_PORTA_DATA_R = 0x10;
   }
   if (pass_count >= 84) {
      GPIO_PORTA_DATA_R = 0x00;
   }
   if (pass_count >= 91){
      NVIC_ST_CTRL_R = 0;
      pass_count=0;
      GPIO_PORTA_DATA_R = ~(0x00);
      GPIO_PORTC_DATA_R = 0x10;
   };
}

void IntDefaultHandler1(void){
   GPIO_PORTF_ICR_R = 0x11;

   if (GPIO_PORTF_DATA_R == 0x01){
      wait(200000);
      counterEW = counterEW + 1; // Increase Traffic Flow at E/W
      if(counterEW > 5){
         counterEW = 0; // Reset Traffic flow
      }
   }
   if (GPIO_PORTF_DATA_R == 0x10){
      wait(200000);
      counterNS = counterNS + 1; // Increase Traffic Flow at N/S
      if(counterNS > 5){
         counterNS = 0; // Reset Traffic flow
      }
   }
}
```

```
void IntDefaultHandler2(void){
   GPIO_PORTD_ICR_R = 0x01;
   ab();
}

void ab(void){
   SysTick_Init(80000000);       // initialize SysTick timer
}

//************************OTHER FUNCTIONS***************************//

void NS_ON_EW_OFF(void){

   GPIO_PORTE_DATA_R =0x04;
   wait(10000000);
   GPIO_PORTE_DATA_R =0x02;
   wait(5000000);
   GPIO_PORTB_DATA_R =0x08;
   wait(9000000);
}
void NS_OFF_EW_ON(void){

   GPIO_PORTB_DATA_R =0x04;
   wait(10000000);
   GPIO_PORTB_DATA_R =0x02;
   wait(5000000);
   GPIO_PORTE_DATA_R =0x08;
   wait(9000000);
}
void wait(int value ){
   int i=0;
   int temp = value+1;
   int times = 10000000*temp;

   if (value<=5)
   {
      while(i<times)
      {i++;};
   }
   else{
      while(i<value)
      {i++;};
   }
}
```