EGC331-01 Microcontroller System Design

Professor Otis

Traffic Control System

5/13/2024

Mary Seelmann

Abstract

This project serves as the capstone assignment for the Microcontroller System Design course. Using the STM32446RE board and Keil, the task was to create a physical setup and code a program for a multi-mode traffic controller that at the least included: a mode for a Rural Traffic controller implemented as a Finite State Machine, a mode for an Urban Traffic Controller with an interrupt-based pedestrian crosswalk feature, and a human-machine interface using a Liquid Crystal Display (LCD). The traffic controller described in this report implemented the Urban Traffic Controller as a Finite State Machine to make the comparison of the functions of the Urban and Rural Controllers easier. In addition, this traffic controller included two extra modes: a mode for a Four-Way Yield, in which the lights blink yellow, and a mode for Four-Way Stop, in which the lights blink red. These modes were added to simulate more of the functions of a real traffic controller.

Table of Contents

1 Introduction

The goal of this project was to create a universal traffic controller. The minimum requirements for the controller are that it can simulate the functionality of a Rural Traffic Controller implemented as a finite state machine and an Urban Traffic Controller with a pedestrian crosswalk feature. The system must also include a human-machine interface. This must be completed using the STM32446RE board in Keil-based C code.

2 Theory

The installation of traffic signals at intersections is essential for efficient traffic flow, as well as the safety of drivers and pedestrians. The type of traffic controller used for an intersection depends on the area. Rural areas tend to have a lower traffic flow and little to no pedestrian crossings. For these areas, vehicle detection sensors are implemented to change the lights based on traffic patterns so that drivers are not waiting at a red light for no reason. When traffic flow is very low, flashing red or yellow lights can be used to control the intersection. Flashing red lights tell drivers to come to a complete stop before crossing the intersection, while flashing yellow lights tell drivers to slow down and proceed with caution. Urban areas tend to have a high traffic flow, so there are almost always cars present at all four ways coming into the intersection. This is why urban traffic controllers tend to be on a time-based system rather than using sensors. Urban areas also have a high concentration of pedestrians crossing. It is important to have clear signs that pedestrians are crossing the street. Pedestrian crossing systems use lights and signs to alert drivers to stop for the crossing. A visible countdown timer gives pedestrians an indication of how long they have to cross before traffic flow resumes.

3 Design

3.1 Subsections

The traffic controller consists of four modes of operation and a human-machine interface using an LCD. All of the components are each detailed in the following sections.

3.1.1 Rural Traffic Controller

The Rural Traffic Controller was implemented as a Finite State Machine. This traffic controller consists of six lights: red, yellow, and green for both North-South (NS) and East-West (EW), as well as two sensors, NS and EW. The initial state, S0, of the system involves the NS light being green and the EW light being red. The system will remain in this state until a car is present at the E or W road, which will trigger the system to switch to the second state, S1, in which the NS light is yellow, and the EW light is red. After a short delay, the system will move to state S2, in which the NS light is red, and the EW light is green. The system will remain in this state until a car is present at the N or S road, which will trigger the system to switch to the fourth state, S3, in which the NS light is red, and the EW light is yellow. After a short delay, the system will move to state S0, and the cycle will continue. This idea can be seen in Figure 1 below.

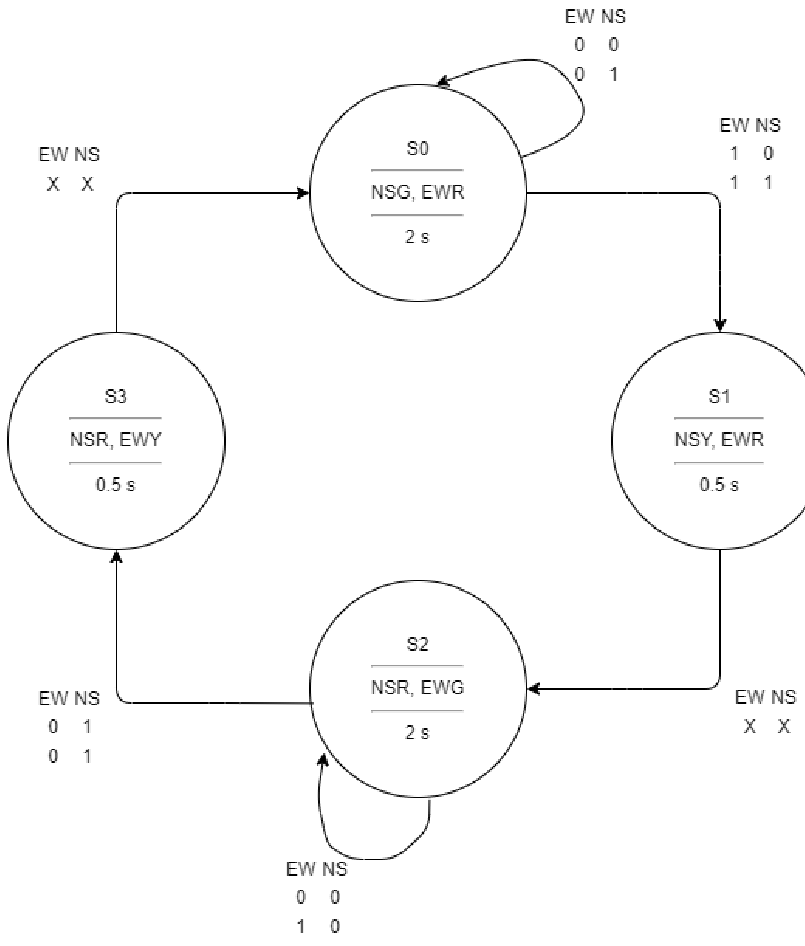

Figure 1 Rural Traffic Controller State Diagram

3.1.2 Urban Traffic Controller

Similar to the Rural Traffic Controller, the Urban Traffic Controller was implemented as a Finite State Machine to have consistency in the code. This traffic controller has the same four states as the Rural; however, the state changes are based solely on time delays. In addition, the Urban Traffic Controller also has a pedestrian crosswalk state. In this state, the NS and EW red lights are turned on and a countdown appears on the 7-segment display. The code for the numbers to appear on the display is stored in an array in descending order, from 9 to 0. The crosswalk function will display all ten numbers, onehalf second for each number. The program only enters this crosswalk state if the button is pressed. When the button is pressed, it triggers an interrupt that sets a variable to indicate that the button was pressed. After the states where one of the lights, either NS or EW, was yellow, the program will check to see whether or not that variable is set to 1. If it is true that the button was pressed, then the program will switch to the crosswalk state. After completing this state, the program will return to where it left off in the main four state cycle. This idea can be seen in Figure 2 below.

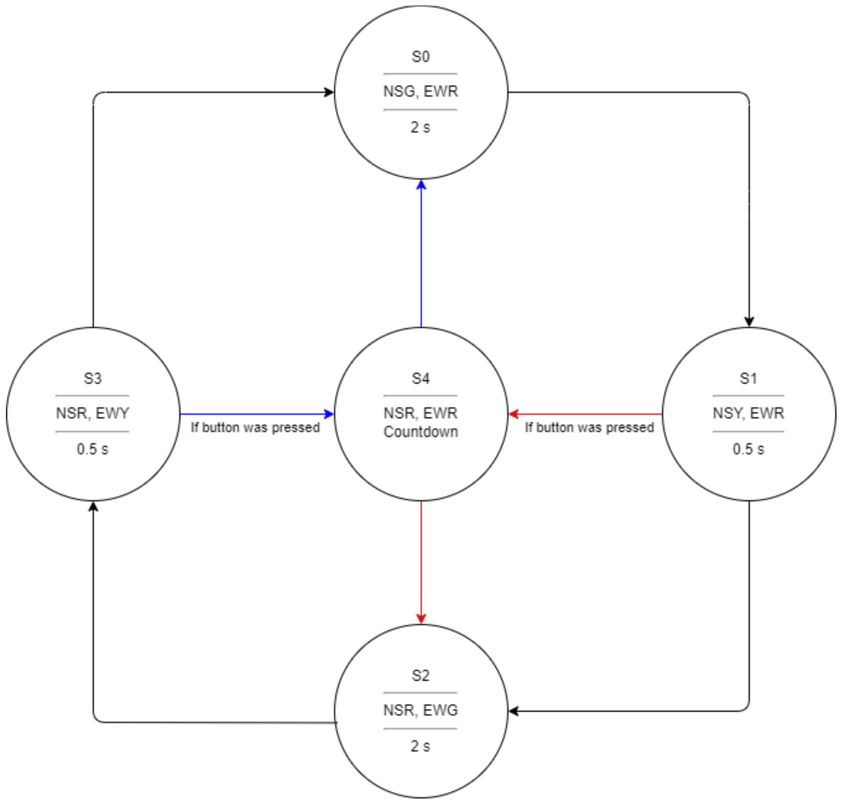

Figure 2 Urban Traffic Controller State Diagram

3.1.3 Four-Way Yield

The Four-Way Yield state is a simple mode where both the NS and EW lights blink yellow. Binary instructions to turn the yellow LEDs on are sent to the output data register. After a short delay, instructions to turn the LEDs off are sent. After another short delay, the cycle repeats. This idea can be seen in Figure 3 below.

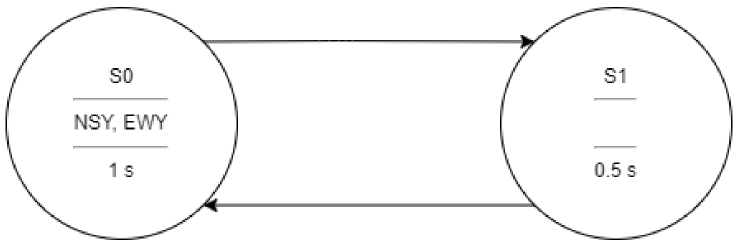

Figure 3 Yield State Diagram

3.1.4 Four-Way Stop

The Four-Way Stop state is similar to the Four-Way Yield state, except that both the NS and EW lights blink red. Binary instructions to turn the red LEDs on are sent to the output data register. After a short delay, instructions to turn the LEDs off are sent. After another short delay, the cycle repeats. This idea can be seen in Figure 4 below.

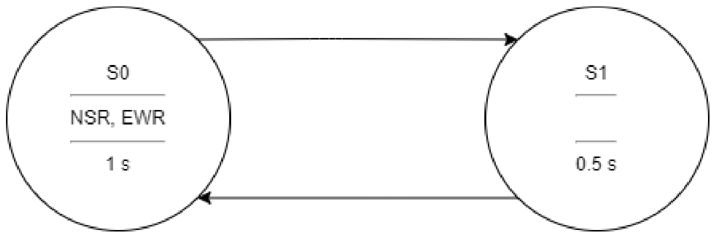

Figure 4 Stop State Diagram

3.1.5 Liquid Crystal Display (LCD)

The LCD consists of sixteen pins. Eight of the pins, DB0 to DB7, are data pins that accept input to be displayed. The pins VDD and VSS provide +5V power and ground, respectively. Pin V0 is used to control the contrast and is connected to a 10K potentiometer. The RS (register select) pin is used to alter the type of input received. The R/W (read/write) pin determines if data is being written to or read from the LCD. The E (enable) pin is used by the LCD to latch information presented to its data pins. The LED+ and LED- pins control the LED backlight. The LCD was used in 4-bit mode, in which the data is split into two parts and sent in two operations to pins DB4 to DB7, and the R/W pin is connected to ground.

3.2 Integration

All the aforementioned modes were combined along with the LCD to create a four-mode traffic controller with a human-machine interface. The Rural Traffic Controller is the default mode 0, the Urban Traffic Controller is mode 1, Four-Way Yield is mode 2, and Four-Way Stop is mode 3. Each mode number corresponds to a 2-bit value that is read in from a switch. The main function of the program consists of a continuous loop with four while loops inside. Each of the four loops corresponds to a mode of operation. The program will enter and remain in a specific mode's loop while the switch is set to that mode's number. When the switch is set to a different number, the program will break out of the current mode's loop after completing the last state it was in and enter the new mode. The two Finite State Machines, Rural and Urban, share a current state variable which allows for smooth transitions between the modes. The LCD provides the user with the current mode, as well as the current state that the traffic controller is in. These ideas can be seen in Figures 5, 6, and 7.

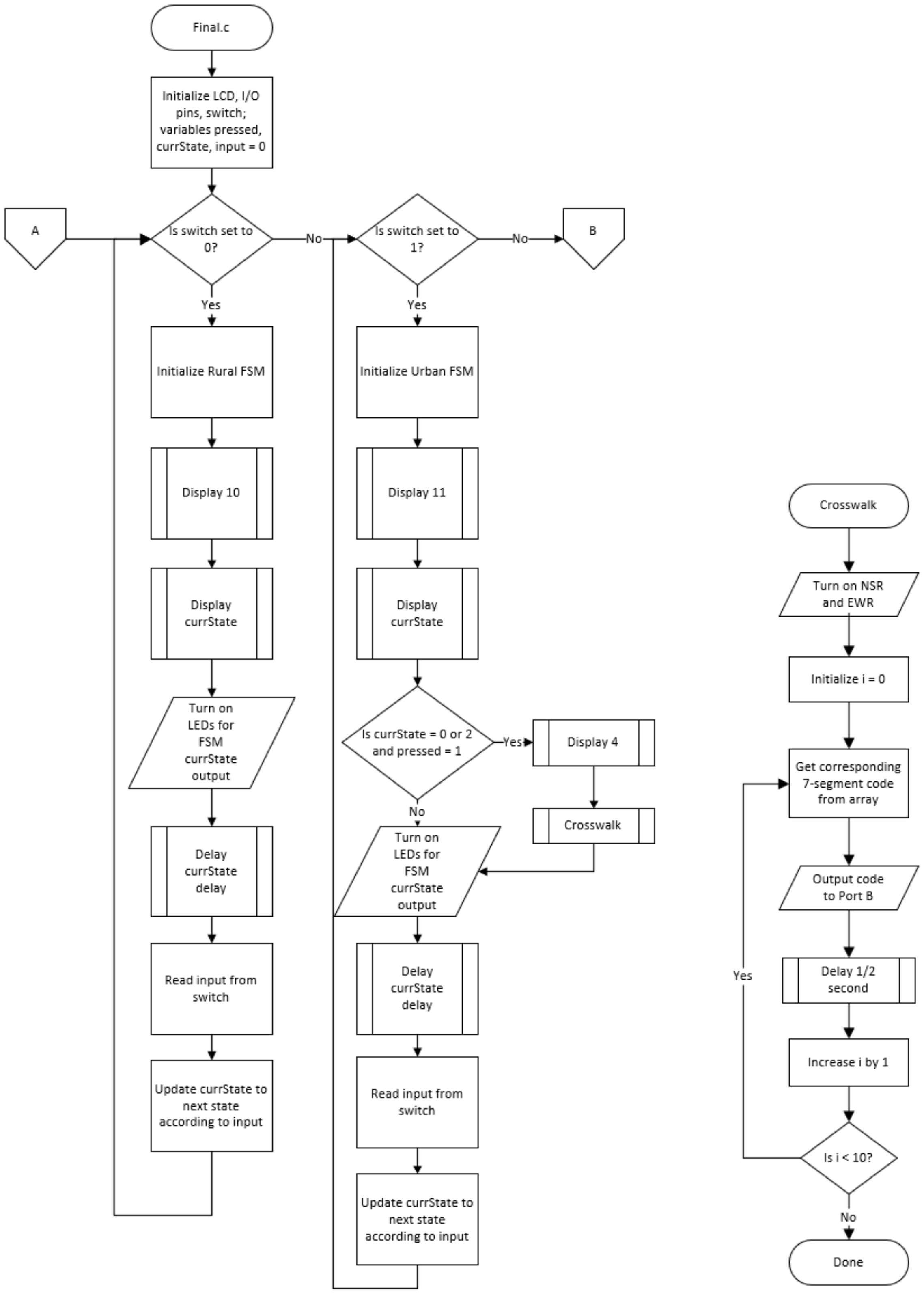

Figure 5 Flowchart for Modes 0 and 1

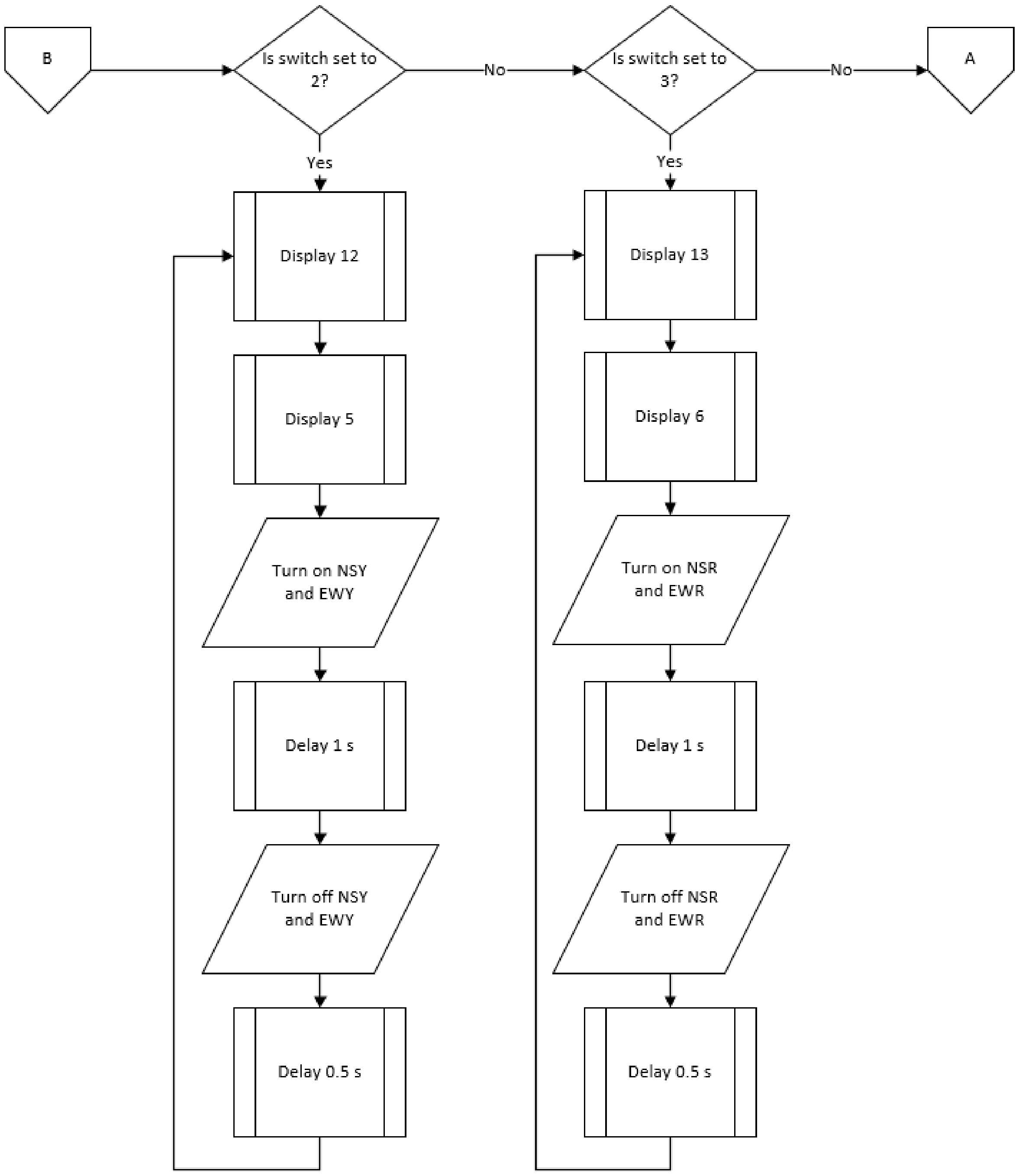

Figure 6 Flowchart for Modes 2 and 3

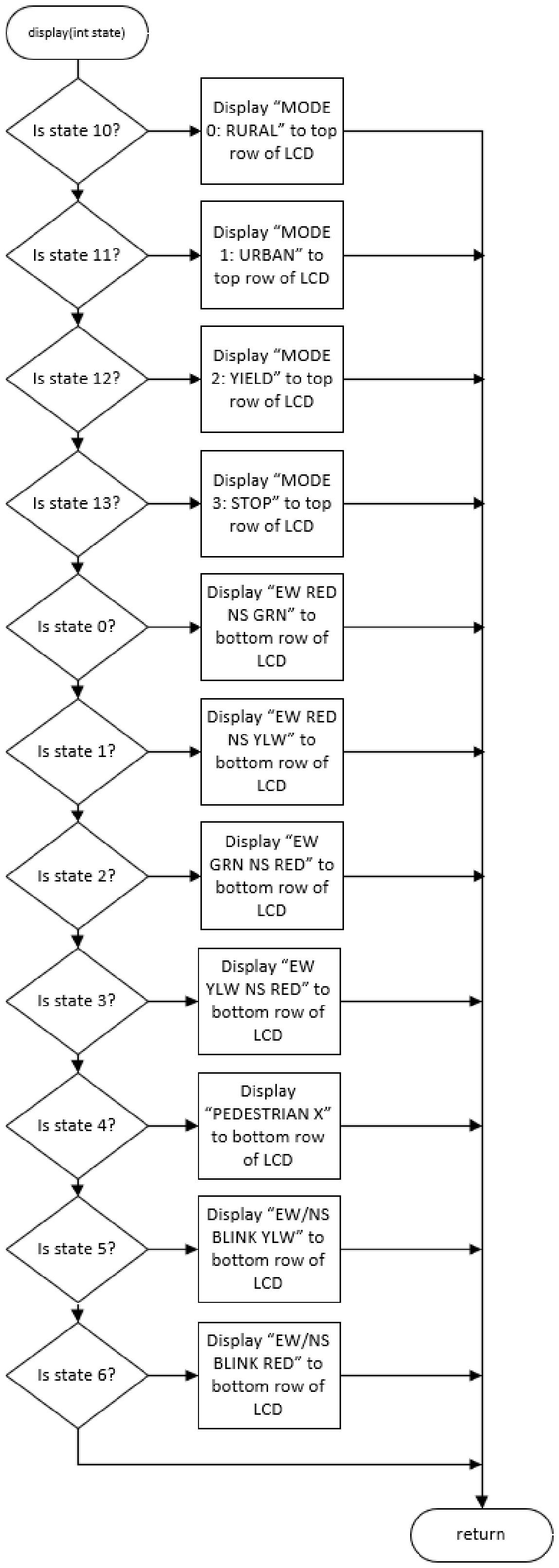
display(int state)
Is state 10?
Display "MODE 0: RURAL" to top row of LCD
Is state 11?
Display "MODE 1: URBAN" to top row of LCD
Is state 12?
Display "MODE 2: YIELD" to top row of LCD
Is state 13?
Display "MODE 3: STOP" to top row of LCD
Is state 0?
Display "EW RED NS GRN" to bottom row of LCD
Is state 1?
Display "EW RED NS YLW" to bottom row of LCD
Is state 2?
Display "EW GRN NS RED" to bottom row of LCD
Is state 3?
Display "EW YLW NS RED" to bottom row of LCD
Is state 4?
Display "PEDESTRIAN X" to bottom row of LCD
Is state 5?
Display "EW/NS BLINK YLW" to bottom row of LCD
Is state 6?
Display "EW/NS BLINK RED" to bottom row of LCD
return

Figure 7 Flowchart for LCD display

The physical setup consists of the STM32446RE board, a breadboard, 6 LEDs (two sets of red, yellow, and green lights for NS and EW), a 4 position DIP switch, a 7-segment display, an LCD, a potentiometer, resistors, and wires. The right two positions (bits 0 and 1) on the switch are used to change between the 4 modes, while the left two positions (bits 2 and 3) are used to represent the NS and EW sensors, respectively, for the Rural Traffic Controller. The physical setup for the traffic controller can be seen in Figure 8 below. The pin mappings can be seen in Figures 9 and 10.

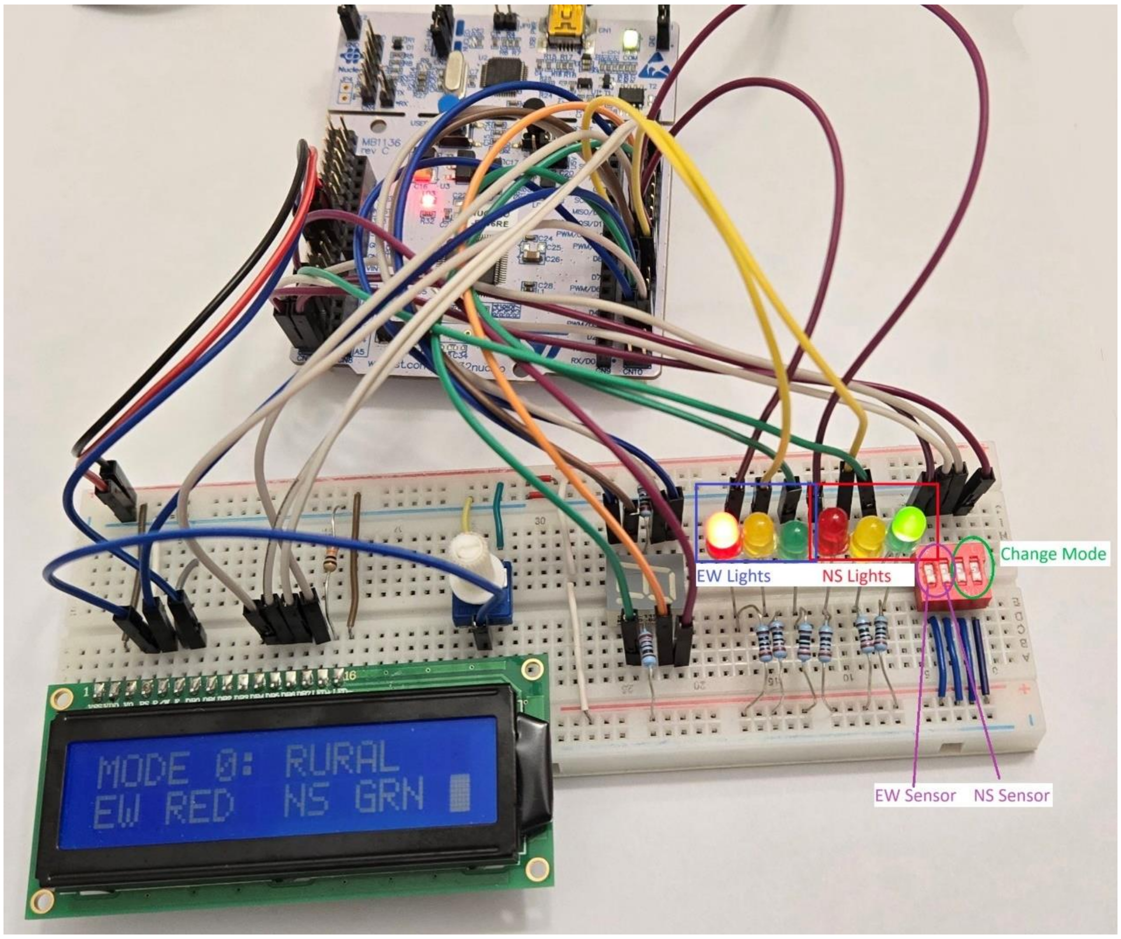

Figure 8 Traffic Controller Physical Setup

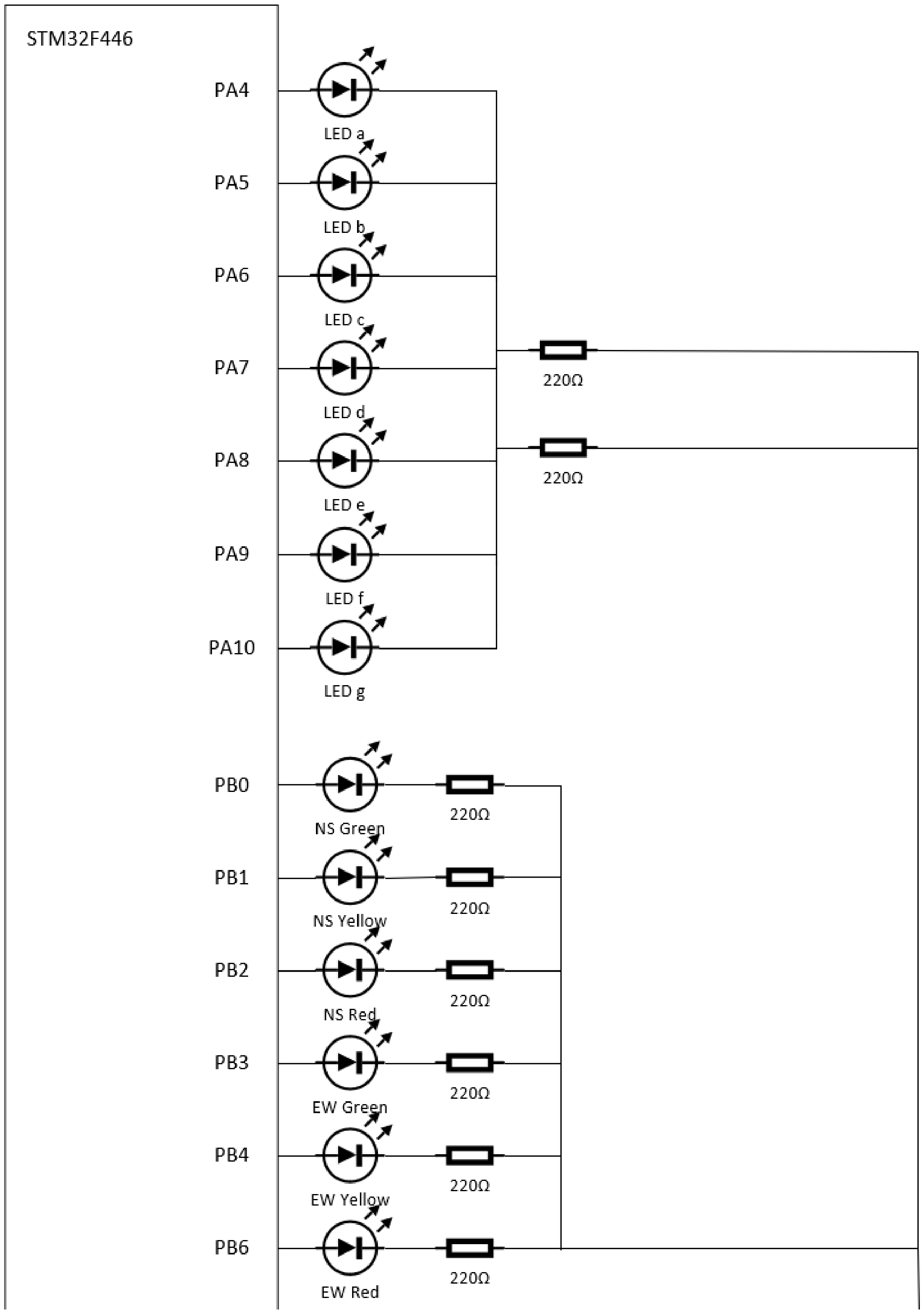

Figure 9 Traffic Controller Pin Mapping part 1

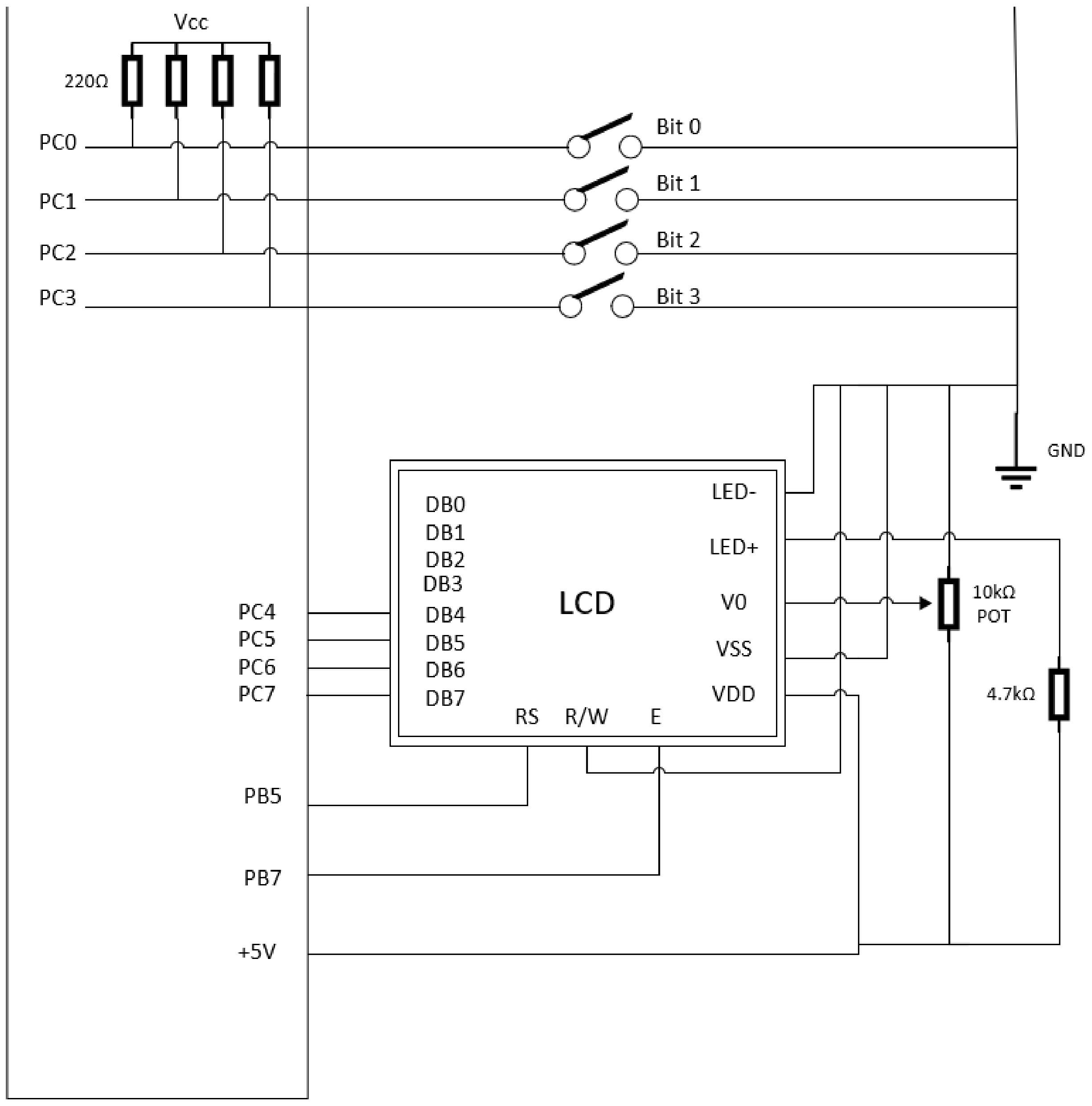

Figure 10 Traffic Controller Pin Mapping part 2

4 Verification

The default mode of the four-mode traffic controller is the Rural Traffic Controller. The system remains in this mode while ‘0’ is read in from the switch. The program is responsive to the NS and EW sensors, as the system will remain in one of the “green states” (NSG/EWR or NSR/EWG) until the sensor that corresponds with the red light of the green state is triggered. The system switches to the Urban Traffic Controller when ‘1’ is read in from the switch. The program cycles through the four states with a time delay between each transition. If the blue button is pressed, the system will enter the pedestrian crossing state after one of the “yellow states” (NSY/EWR or NSR/EWY). The system switches to the Four-Way Yield mode when ‘2’ is read in from the switch. In this mode, the NS and EW yellow LEDs blink together. The system switches to the Four-Way Stop mode when ‘3’ is read in from the switch. In this mode, the NS and EW red LEDs blink together. The number and name for the selected mode is displayed on the top line of the LCD. The second line of the LCD displays the current state of the LEDs for all modes or that the system is in the pedestrian crossing state for the Urban Traffic Controller. Functional verification of this traffic controller can be viewed at the link listed in Appendix A.

5 Results and Conclusion

The system works as expected and achieves all of the goals set. This project allowed students to bring together what was learned in the lab assignments for this course. Most of the challenges faced in this project were during the process of combining each subsection together onto one board. It took detailed planning to determine how to configure all the parts of this project.

Bibliography

Jovanis, Paul P. and Hobbs, F.D.. "traffic control". Encyclopedia Britannica, 4 Nov. 2023, https://www.britannica.com/technology/traffic-control. Accessed 11 May 2024.

“Significance of Traffic Technology on Rural Highways.” *Traffic Technology on Rural Highways | ELTEC Traffic Technology*, Electrotechnics Corporation, elteccorp.com/news/other/significance-of-traffic-technology-on-rural-highways/. Accessed 11 May 2024.

Appendix A

Functional Verification Video: https://youtu.be/b-EOmDm32F4?si=9TU4kv-BiZwOtiVd

Github: https://github.com/mseelmann/Universal-Traffic-Controller

Appendix B

```
/* Final Project
 *
 * The Traffic Light setup is connected to the Nucleo-FF446RE board as follows:
 * PB0-PB4 for NSG, NSY, NSR, EWG, EWY, and PB6 for EWR
 *
 * The 7-segment display is connected to the board as follows:
 * PA4 to PA10
 *
 * The LCD controller is connected to the board as follows:
 * PC4-PC7 for LCD D0-D7, respectively.
 * PB5 for LCD R/S
 * LCD R/W is tied to ground
 * PB7 for LCD EN
 *
 * PC13 for push button interrupt
 * PC0-PC3 for switch input
 */

#include "stm32f4xx.h"

/* LCD */
#define RS 0x20     /* PB5 mask for reg select */
#define EN 0x80     /* PB7 mask for enable */
void LCD_nibble_write(char data, unsigned char control);
void LCD_command(unsigned char command);
void LCD_data(char data);
void LCD_init(void);
void PORTS_init(void);
void display(int state);

/* Urban with Crosswalk Interrupt */
void urban_init(void);
void pedestrian(void);
int pressed = 0; /*interrupt variable */

/* Rural FSM */
int currState = 0;
int input = 0;

void delayMs(int n);

int main(void) {
             RCC->AHB1ENR |= 7; /* enable GPIOA, GPIOB, and GPIOC clocks */

             /* initialize LCD controller */
    LCD_init();

             urban_init();
```

```
		/* Switch */
		GPIOC->PUPDR = 0x00000055; /* enable internal pull-up resistors for
pins C0-C3 */

    while(1) {
		/* RURAL */
			while((GPIOC->IDR & 0x03)==0x00) {
					display(10);

					/* FSM Setup */
					struct State {
						int output;
						int delay;
						int nextState[4];
					};

					typedef const struct State rural;
						#define S0 0
						#define S1 1
						#define S2 2
						#define S3 3

					rural FSM[4] = {
						{0x41,3000,{S0,S0,S1,S1}},	/* state 0:
NSG/EWR, 3s */
						{0x42,1500,{S2,S2,S2,S2}},    /* state 1:
NSY/EWR, 1.5s */
						{0x0C,3000,{S2,S3,S2,S3}},	/* state 2:
NSR/EWG, 3s */
						{0x14,1500,{S0,S0,S0,S0}},    /* state 3:
NSR,EWY, 1.5s */
					};

					display(currState);
					GPIOB->ODR = (FSM[currState].output);
					delayMs(FSM[currState].delay);
					input = (GPIOC->IDR & 0x0000000C)>>2;
					currState = FSM[currState].nextState[input];
				}
		/* URBAN */
				while((GPIOC->IDR & 0x03)==0x01) {
					display(11);

					/* FSM Setup */
					struct State {
						int output;
						int delay;
						int nextState[4];
					};

					typedef const struct State urban;
						#define S0 0
						#define S1 1
						#define S2 2
						#define S3 3

					urban FSM[4] = {
						{0x41,3000,{S1,S1,S1,S1}},	/* state 0:
NSG/EWR, 3s */
						{0x42,1500,{S2,S2,S2,S2}},    /* state 1:
NSY/EWR, 1.5s */
						{0x0C,3000,{S3,S3,S3,S3}},	/* state 2:
NSR/EWG, 3s */
```

```
                                        {0x14,1500,{S0,S0,S0,S0}},    /* state 3:
NSR,EWY, 1.5s */
                                 };

                                 display(currState);
                                 if((currState==0 || currState==2) && pressed==1) {
                                        display(4);
                                        /* if so, then call pedestrian */
                                        pedestrian();
                                 }
                                 GPIOB->ODR = (FSM[currState].output);
                                 delayMs(FSM[currState].delay);
                                 input = (GPIOC->IDR & 0x0000000C)>>2;
                                 currState = FSM[currState].nextState[input];
                          }
                          /* 4-WAY YIELD */
                          while((GPIOC->IDR & 0x03)==0x02) {
                                 display(12);
                                 display(5);

                                 GPIOB->ODR = 0x12;
                                 delayMs(1000);
                                 GPIOB->ODR = 0x00;
                                 delayMs(500);
                          }
                          /* 4-WAY STOP */
                          while((GPIOC->IDR & 0x03)==0x03) {
                                 display(13);
                                 display(6);

                                 GPIOB->ODR = 0x44;
                                 delayMs(1000);
                                 GPIOB->ODR = 0x00;
                                 delayMs(500);
                          }
    }
}

void display(int state) {
                    if(state==10) {
                                 LCD_command(0x80); /* set cursor to first line */
                                 LCD_data('M');
                                 LCD_data('O');
                                 LCD_data('D');
                                 LCD_data('E');
                                 LCD_data(' ');
                                 LCD_data('0');
                                 LCD_data(':');
                                 LCD_data(' ');
                                 LCD_data('R');
                                 LCD_data('U');
                                 LCD_data('R');
                                 LCD_data('A');
                                 LCD_data('L');
                    } else if(state==11) {
                                 LCD_command(0x80);
                                 LCD_data('M');
                                 LCD_data('O');
                                 LCD_data('D');
                                 LCD_data('E');
                                 LCD_data(' ');
                                 LCD_data('1');
                                 LCD_data(':');
```

```
                                  LCD_data(' ');
                                  LCD_data('U');
                                  LCD_data('R');
                                  LCD_data('B');
                                  LCD_data('A');
                                  LCD_data('N');
                    } else if(state==12) {
                                  LCD_command(0x80);
                                  LCD_data('M');
                                  LCD_data('O');
                                  LCD_data('D');
                                  LCD_data('E');
                                  LCD_data(' ');
                                  LCD_data('2');
                                  LCD_data(':');
                                  LCD_data(' ');
                                  LCD_data('Y');
                                  LCD_data('I');
                                  LCD_data('E');
                                  LCD_data('L');
                                  LCD_data('D');
                    } else if(state==13) {
                                  LCD_command(0x80);
                                  LCD_data('M');
                                  LCD_data('O');
                                  LCD_data('D');
                                  LCD_data('E');
                                  LCD_data(' ');
                                  LCD_data('3');
                                  LCD_data(':');
                                  LCD_data(' ');
                                  LCD_data('S');
                                  LCD_data('T');
                                  LCD_data('O');
                                  LCD_data('P');
                                  LCD_data(' ');
                    } else if(state==0) {
                                  LCD_command(0xC0); /* set cursor to second line */
                                  LCD_data('E');
                                  LCD_data('W');
                                  LCD_data(' ');
                                  LCD_data('R');
                                  LCD_data('E');
                                  LCD_data('D');
                                  LCD_data(' ');
                                  LCD_data(' ');
                                  LCD_data('N');
                                  LCD_data('S');
                                  LCD_data(' ');
                                  LCD_data('G');
                                  LCD_data('R');
                                  LCD_data('N');
                             LCD_data(' ');
                    } else if(state==1) {
                                  LCD_command(0xC0);
                                  LCD_data('E');
                                  LCD_data('W');
                                  LCD_data(' ');
                                  LCD_data('R');
                                  LCD_data('E');
                                  LCD_data('D');
                                  LCD_data(' ');
                                  LCD_data(' ');
```

```
                        LCD_data('N');
                        LCD_data('S');
                        LCD_data(' ');
                        LCD_data('Y');
                        LCD_data('L');
                        LCD_data('W');
                   LCD_data(' ');
          } else if(state==2) {
                        LCD_command(0xC0);
                        LCD_data('E');
                        LCD_data('W');
                        LCD_data(' ');
                        LCD_data('G');
                        LCD_data('R');
                        LCD_data('N');
                        LCD_data(' ');
                        LCD_data(' ');
                        LCD_data('N');
                        LCD_data('S');
                        LCD_data(' ');
                        LCD_data('R');
                        LCD_data('E');
                        LCD_data('D');
                   LCD_data(' ');
          } else if(state==3) {
                        LCD_command(0xC0);
                        LCD_data('E');
                        LCD_data('W');
                        LCD_data(' ');
                        LCD_data('Y');
                        LCD_data('L');
                        LCD_data('W');
                        LCD_data(' ');
                        LCD_data(' ');
                        LCD_data('N');
                        LCD_data('S');
                        LCD_data(' ');
                        LCD_data('R');
                        LCD_data('E');
                        LCD_data('D');
                   LCD_data(' ');
          } else if(state==4) {
                        LCD_command(0xC0);
                        LCD_data('P');
                        LCD_data('E');
                        LCD_data('D');
                        LCD_data('E');
                        LCD_data('S');
                        LCD_data('T');
                        LCD_data('R');
                        LCD_data('I');
                        LCD_data('A');
                        LCD_data('N');
                        LCD_data(' ');
                        LCD_data('X');
                        LCD_data(' ');
                        LCD_data(' ');
                   LCD_data(' ');
          } else if(state==5) {
                        LCD_command(0xC0);
                        LCD_data('E');
                        LCD_data('W');
                        LCD_data('/');
```

```
                                 LCD_data('N');
                                 LCD_data('S');
                                 LCD_data(' ');
                                 LCD_data('B');
                                 LCD_data('L');
                                 LCD_data('I');
                                 LCD_data('N');
                                 LCD_data('K');
           LCD_data(' ');
                                 LCD_data('Y');
                                 LCD_data('L');
                                 LCD_data('W');
                     } else if(state==6) {
                                 LCD_command(0xC0);
                                 LCD_data('E');
                                 LCD_data('W');
                                 LCD_data('/');
                                 LCD_data('N');
                                 LCD_data('S');
                                 LCD_data(' ');
                                 LCD_data('B');
                                 LCD_data('L');
                                 LCD_data('I');
                                 LCD_data('N');
                                 LCD_data('K');
           LCD_data(' ');
                                 LCD_data('R');
                                 LCD_data('E');
                                 LCD_data('D');
                     }
}

void urban_init(void) {
             __disable_irq(); /* global disable IRQs */
             RCC->APB2ENR |= 0x4000; /* enable SYSCFG clock */
             GPIOB->MODER &= ~0x0000FFFF; /* clear pin mode */
             GPIOB->MODER = 0x00005555; /* set B pins to output mode */
             GPIOA->MODER &= ~0x00FFFF00; /*clear pin mode */
             GPIOA->MODER |= 0x00555500; /* set A pins to output mode */

             /* configure PC13 for push button interrupt */
             GPIOC->MODER &= ~0x0C000000;        /* clear pin mode to input mode */
             SYSCFG->EXTICR[3] &= ~0x00F0;       /* clear port selection for EXTI13
*/
             SYSCFG->EXTICR[3] |= 0x0020;        /* select port C for EXTI13 */
             EXTI->IMR |= 0x2000;                /* unmask EXTI13 */
             EXTI->FTSR |= 0x2000;               /* select falling edge trigger */
             NVIC_EnableIRQ(EXTI15_10_IRQn);
    __enable_irq();                      /* global enable IRQs */
}

/* Interrupt Handler */
void EXTI15_10_IRQHandler(void) {
        pressed = 1; /* it's true that button was pressed*/
        EXTI->PR = 0x2000;  /* clear interrupt pending flag */
}

/* Countdown */
void pedestrian(void) {
       GPIOB->ODR = 0x000000044; /* NSR, EWR ON */
       int countdown[10] = {0x6f, 0x7f, 0x07, 0x7d, 0x6d, 0x66, 0x4f, 0x5b, 0x06,
0x3f};
       for(int i=0; i<10; i++) {
            GPIOA->ODR = (countdown[i])<<4; /* display number */
            delayMs(750);
        }
        /* Blink 0 three times */
        for(int j=0; j<3; j++) {
            GPIOA->ODR = (0x00)<<4;
            delayMs(750);
            GPIOA->ODR = (0x3f)<<4;
            delayMs(750);
        }
        GPIOA->ODR = 0x00; /* clear display */
        pressed = 0; /* reset button pressed to false */
}

/* initialize GPIOB/C then initialize LCD controller */
void LCD_init(void) {
    PORTS_init();

    delayMs(20);                /* LCD controller reset sequence */
    LCD_nibble_write(0x30, 0);
    delayMs(5);
    LCD_nibble_write(0x30, 0);
    delayMs(1);
    LCD_nibble_write(0x30, 0);
    delayMs(1);

    LCD_nibble_write(0x20, 0);  /* use 4-bit data mode */
    delayMs(1);
    LCD_command(0x28);          /* set 4-bit data, 2-line, 5x7 font */
    LCD_command(0x06);          /* move cursor right */
    LCD_command(0x01);          /* clear screen, move cursor to home */
    LCD_command(0x0F);          /* turn on display, cursor blinking */
}

void PORTS_init(void) {
    /* PORTB 5 for LCD R/S */
    /* PORTB 7 for LCD EN */
    GPIOB->MODER &= ~0x0000CC00;    /* clear pin mode */
    GPIOB->MODER |=  0x00004400;    /* set pin output mode */
    GPIOB->BSRR = 0x00800000;       /* turn off EN */

    /* PC4-PC7 for LCD D4-D7, respectively. */
    GPIOC->MODER &= ~0x0000FF00;    /* clear pin mode */
    GPIOC->MODER |=  0x00005500;    /* set pin output mode */
}

void LCD_nibble_write(char data, unsigned char control) {
    /* populate data bits */
    GPIOC->BSRR = 0x00F00000;       /* clear data bits */
    GPIOC->BSRR = data & 0xF0;      /* set data bits */

    /* set R/S bit */
    if (control & RS)
        GPIOB->BSRR = RS;
    else
        GPIOB->BSRR = RS << 16;

    /* pulse E */
    GPIOB->BSRR = EN;
    delayMs(0);
    GPIOB->BSRR = EN << 16;
}
```

```c
void LCD_command(unsigned char command) {
    LCD_nibble_write(command & 0xF0, 0);    /* upper nibble first */
    LCD_nibble_write(command << 4, 0);      /* then lower nibble */

    if (command < 4)
        delayMs(2);         /* command 1 and 2 needs up to 1.64ms */
    else
        delayMs(1);         /* all others 40 us */
}

void LCD_data(char data) {
    LCD_nibble_write(data & 0xF0, RS);      /* upper nibble first */
    LCD_nibble_write(data << 4, RS);        /* then lower nibble */

    delayMs(1);
}

/* 16 MHz SYSCLK */
void delayMs(int n) {
    int i;

    /* Configure SysTick */
    SysTick->LOAD = 16000;  /* reload with number of clocks per millisecond */
    SysTick->VAL = 0;       /* clear current value register */
    SysTick->CTRL = 0x5;    /* Enable the timer */

    for(i = 0; i < n; i++) {
        while((SysTick->CTRL & 0x10000) == 0) /* wait until the COUNTFLAG is set */
            { }
    }
    SysTick->CTRL = 0;      /* Stop the timer (Enable = 0) */
}
```