

BEng Embedded
Electronic Systems
Electronics I
ACDA5002
Joshua Ofosu Asare
2111739

Contents

Introduction	3
Background research	4
Similar product analysis	4
Portable Traffic Light – LZA 500 Compact Max.....	4
Portable Pedestrian Light – LZA 500 Compact Max.....	4
Circuit designs	5
Microcontroller choice.....	5
Analog-to-Digital Converter	5
Potentiometer replacement for an ultrasonic sensor	5
Use of NOR gates to reduce μ Controller pin usage	6
Flowchart	7
Parts list and cost estimate	8
Parts list.....	8
Cost	8
Testing.....	9
Test cases and results	9
Multiple buttons pressed during an active crossing.....	9
Equal volumes of traffic from different directions	9
Simulated outputs.....	Error! Bookmark not defined.
Results.....	Error! Bookmark not defined.
Conclusions	10
Recommendations	Error! Bookmark not defined.
Bibliography	11
Software listings.....	11

Introduction

This assignment looks at existing solutions to traffic and pedestrian management across 4-way intersections. These traffic management systems are essential in cluttered environments that can be a bottleneck due to the volume of traffic flowing through them. The circuit simulation made in Proteus prioritises traffic coming from a direction with greater a greater queue than currently passing traffic in order to try and reduce wait times. This also helps to avoid a case seen during testing where a small number of cars continuously entering the intersection from the same direction would block traffic flow from other directions until there were no cars passing by. The circuit explored in this assignment lacks some of the advanced features seen in competition such as remove management to reconfigure the devices while they have been deployed. Some edge cases have been handled that would prevent the system from operating as it would have been expected to otherwise. These edge cases were identified and fixed through testing and recording outputs.

Background research and similar product analysis

Portable Traffic Light – LZA 500 Compact Max



Figure 1: A portable traffic light [1]

This product is intended as a temporary replacement that can be deployed in an area and connected with multiple other units. This unit can be set up as a master or a slave device. It is advertised as being able to be set up.

An advanced feature this system has that falls outside the scope of this assignment is remote control from monitoring software, using 4G, that displays the live statuses of deployed units, informs users when batteries on units are low and contains GPS tracking.

Portable Pedestrian Light – LZA 500 Compact Max



Figure 2: A portable pedestrian light [1]

This product has fewer capabilities than the previous device. It has no option to be both a master or slave device. It is a lot simpler in design.

Circuit designs

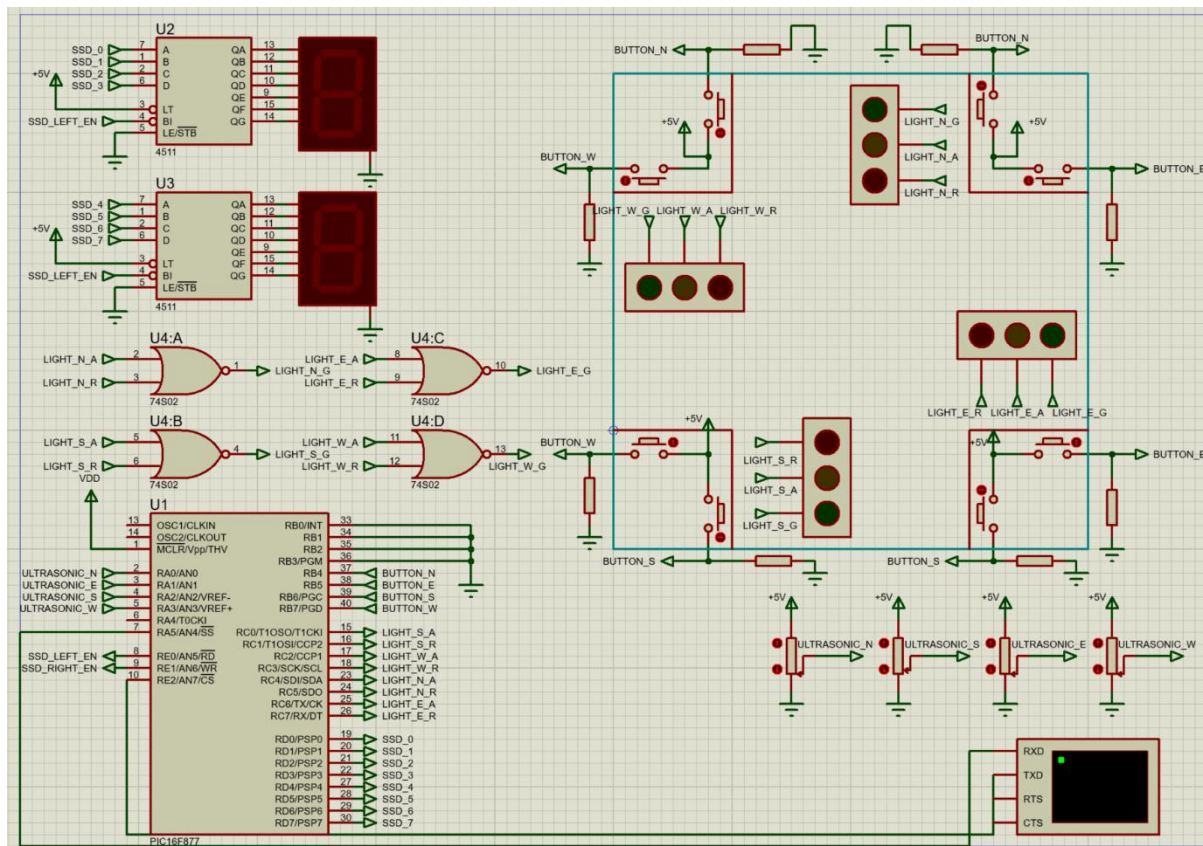


Figure 3: Proteus circuit schematic for a traffic light system

Microcontroller choice

A PIC16F877 was chosen as it has an appropriate number of GPIO pins for this task and support for interrupts. Interrupts have been used specifically to allow the ADC values from the potentiometers to be used without having to explicitly read them just before use. Two different interrupts were used: TIMER_2 and INT_RB. TIMER_2 is a timer triggered by the internal clock of the PIC when a counter overflow, while INT_RB is an interrupt triggered when a change on port b is detected. Specifically pins R0-R3. This is where the pedestrian buttons have been connected to minimise the amount of code that has to be written to handle pedestrian crossing.

Analog-to-Digital Converter

To allow for consistent timing between tasks and automatic ADC reads, some functionality has been wrapped in interrupts. Changing ADC channels, a delay for them to settle and reading the outputs to global variables are handled by the TIMER_2 interrupt which has been configured to run at its lowest frequency as a fast update is not required of this system. A settling time of 30µs was used to provide a margin of error over the minimum acquisition time specified by the manufacturer [1].

Seven Segment Display

Two seven segment displays are connected to a BCD which reduces the number of pins required from the PIC. Instead of using 7 pins for a single seven segment display, 10 pins are used across two displays. The SSD_LEFT_EN and SSD_RIGHT_EN pins are responsible for switching the displays on and off which provides a reduction in the idle power consumption of the system. The remaining 4 pins that are connected from the PIC to each BCD encode a 4-bit binary number which is decoded by the BCD and sent to its output pins, which are connected to the seven segment display. When the

function to signify pedestrians can cross is run, a countdown appears across the seven segment displays which starts at 10, and counts down to 0 before turning off.

Only a single seven segment display has been simulated as this signal is repeated across all 8 displays to pedestrians.

Traffic lights

The traffic lights selected for this circuit come from a proteus model that only require 3 pins, which helps to reduce the pin count in conjunction with the use of NOR gates. The traffic lights have a common cathode (not shown in the simulation), leaving only 3 pins for amber, red and green to be connected to the PIC.

Virtual Terminal

The virtual terminal connected serves no purpose for the final product other than debugging during testing.

Potentiometer replacement for an ultrasonic sensor

Analog potentiometers have been used in place of ultrasonic sensors for this simulation as the version of Proteus used (8.15) does not have support for an analogue ultrasonic sensor. The PIC16F877's internal ADC is used to convert the reading from the potentiometer, which ranges from +0VDC to +5VDC.

Use of NOR gates to reduce μ Controller pin usage

To reduce the total number of required pins from the PIC microcontroller, NOR gates have been connected to the output pins that control the amber and red lights in the traffic light units. When either amber or red lights are on, the green light remains off. When both are off, only then does the green light turn on. This is shown below in Table 1: Truth table for a NOR gate.

Table 1: Truth table for a NOR gate

Input A (Amber light)	Input B (Red light)	Output Q (Green Light)
0	0	1
0	1	0
1	0	0
1	1	0

Flowchart

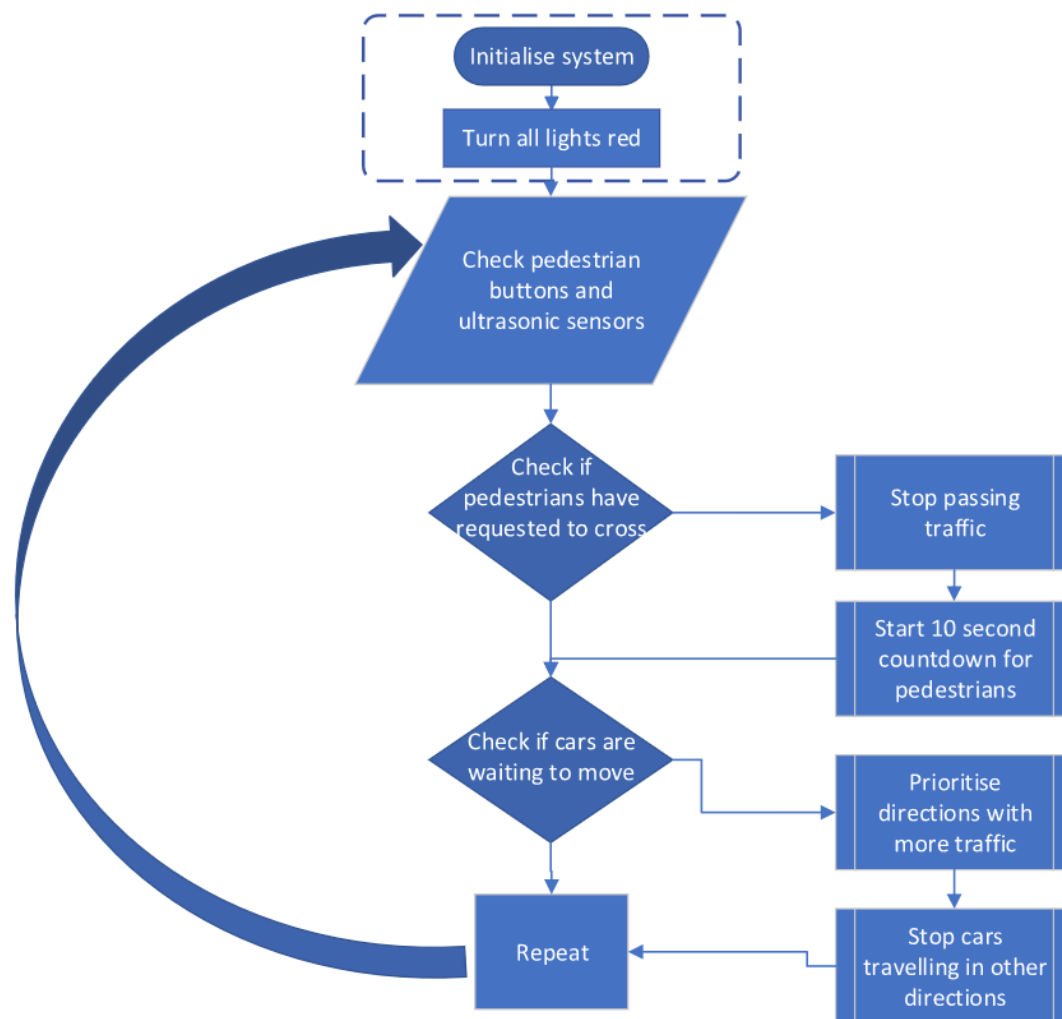


Figure 4: A flow chart of the program running on the PIC16F877

Figure 2 illustrates a high-level structure of the program running on the PIC16F877. The main loop doesn't contain much – a check to monitor if any of the pedestrian buttons have been pressed, if they have, then a sequence to stop passing traffic and then initiate a 10 second countdown to allow pedestrians to cross the road. Then, a check for any traffic is run to see if the current signals need to change. If there is a build up of traffic from a particular direction, that traffic will take priority over currently moving traffic if there are more vehicles waiting to pass.

Parts list and cost estimate

Parts list

Table 2: A parts and cost estimation for a traffic system

Part	Link	Cost per unit	Count	Total
PIC16F877 Microcontroller	RS-Online	£8.443	1	£8.44
SN74HC02N Quad 2-Input NOR Logic Gate	RS-Online	£0.539	1	£0.54
Traffic Light	RS-Online	£306.47	4	£1,225.88
N-Channel MOSFET	RS-Online	£1.471	12	£17.65
Push button	RS-Online	£4.01	8	£32.08
Resistors	RS-Online	£0.232	1	£0.23
BCD Counter	RS-Online	£0.631	16	£10.10
Seven Segment Display	RS-Online	£2.88	16	£46.08
Ultrasonic Sensor	RS-Online	£214.15	4	£856.60
Wire	RS-Online	£109.27	1	£109.27
Total				£2,306.87

The voltage of chosen traffic light is 20-30V, with a current draw of 360mA. Despite being omitted from the circuit diagram. The pins of the PIC μ Controller cannot provide enough current to power a single coloured light in the traffic light. A suitable N-Channel MOSFET has been selected that would be responsible for switching power to each of the coloured lights. A standard resistor array has been selected as it reduces the total parts count and has enough to cover all of the push buttons used in the design. The traffic light and ultrasonic sensor make up the vast majority of the cost for the product – the ultrasonic sensor has a high price as it needs to work over a large distance. Wire is an additional cost that will have to be factored in, since the various sensors, displays and lights will be distributed over a wide area and not be placed immediately next to one another.

Cost

The total cost of this system is estimated to be £2,306.87. This is a reasonable cost for a solution deployed to main roads where signalling that is not only based around time delays is required.

Testing, simulated outputs and results

Test cases and results

The following cases were tested against the simulated circuits and results recorded:

Multiple buttons pressed during an active crossing

One pedestrian button is pressed but one or more buttons are pressed during the countdown as pedestrians have already started to walk across the road.

Interrupts are disabled during this period specifically so that no other pedestrian button pressed can affect the operation of the system during crossing. During testing, pressing buttons during the crossing period had no effect on the counter.

Equal volumes of traffic from different directions

In the case that there is an equal volume of traffic coming from two directions, there is the potential for the program to send no signals due to comparisons returning false.

This turned out to be a fatal issue with the system, it would get stuck and the lights would not change, however pedestrian button functionality worked as before. Only when one of the potentiometers values was increased would the system return to normal behaviour.

This was a problem initially so a comparison was changed from:

```
if (north_south > east_west)
    to:
if ((north_south != 0) & (north_south + 1 > east_west))
```

In the case that north_south and east_west are equal to each other, the additional +1 added to north_south means that it carries a heavier weight and will be preferred over east_west flowing traffic. The check that north_south != 0 is to prevent this if check from passing when north_south is 0 and always taking priority when there is no traffic.

Throughout testing, a mix of the virtual terminal and "PIC CPU\Variables" in the debugging view were used. The virtual terminal is able to provide live data without having to pause the program. For example, when an interrupt was fired, it would be printed to the virtual terminal. The PIC CPU\Variables window in the debug view was useful as it allowed a view of the raw value variables in memory held.

Virtual Terminal

INT IN RB

Figure 5: A screenshot of the virtual terminal from Proteus

Simulation Log	Watch Window	PIC CPU/Variables - U1		
PIC CPU/Variables - U1				
Name	Address	Type	Value	Previous Value
<u>_RETURN_</u>	0x0078	byte	0x01	
> <u>countdown</u>	0x003A	byte[11]	byte[11]	
<u>traffic_stat</u>	0x0039	byte	'\0'	
<u>buttons</u>	0x0038	byte	0x01	
<u>east_west</u>	0x0036	word	0	
<u>north_south</u>	0x0034	word	0	
<u>us_west</u>	0x0032	word	0	
<u>us_south</u>	0x0030	word	25	
<u>us_east</u>	0x002E	word	0	
<u>us_north</u>	0x002C	word	0	
<u>PSP_DATA</u>	0x0008	byte	0x07	
<u>CCP_2_HIGH</u>	0x001C	byte	'm'	
<u>CCP_2_LOW</u>	0x001B	byte	'm'	
<u>CCP_2</u>	0x001B	word	28013	
<u>CCP_1_HIGH</u>	0x0016	byte	'm'	
<u>CCP_1_LOW</u>	0x0015	byte	'm'	
<u>CCP_1</u>	0x0015	word	28013	
<u>_invar2</u>	0x0053	byte	0xA4	

Figure 6: A screenshot of the variables from the debug view in Proteus

Conclusions and recommendations

Conclusion

It is possible to implement a pedestrian and traffic light system that will respond to various signals input in order to prioritise traffic and pedestrians while avoiding having pedestrians walking as cars are crossing. The circuit designed and simulated serves as a good foundation that can be built on top of to include further functionality to improve the reliability and consistency of the system.

Recommendations

- Work to further develop on top of this assignment could include:
Similar to competition, adding in a capability to remotely control [2] and configure deployments from software responsible for managing all deployed configurations. This adds the benefit of not needing a person on site to adjust the system to the type of traffic that is expected to flow through an area. For example, during a period of time where foot traffic is a lot higher than normal, lower priority could be placed on the traffic and greater on pedestrians by giving them more time to cross in each window.
- Allowing the system to run off of batteries. This advantage to this is resilience to unforeseen downtime and intermittency in the national grid. Having lots of traffic lights and pedestrian crossing signals off due to a fault elsewhere could create a hazardous crossing that can be avoided by adding redundancy to the system.
- Adaptation and fault detection. On a give day if the system notices particular trends, such as increased traffic or foot traffic. It could increase or decrease the period that pedestrians have to cross the road, up to a configured minimum or maximum.
- A further reduction in the number of pins used is possible. All buttons could be connected to the same line as only a single signal is needed due to traffic having to stop flowing across all of the 4-way intersection.

Bibliography

- [1] Microchip, "PIC16F87X - 28/40-Pin 8-Bit CMOS FLASH Microcontrollers," 29 11 2012. [Online]. Available: <https://ww1.microchip.com/downloads/aemDocuments/documents/MCU08/ProductDocuments/DataSheets/30292D.pdf>. [Accessed 21 04 2023].
- [2] DVSA (Driver and Vehicle Standards Agency), "Traffic Lights in the UK - Meanings, Sequence & Rules for Learner Drivers," [Online]. Available: <https://theorytest.org.uk/traffic-lights/>. [Accessed 21 04 2023].
- [3] Traffic Group Signals, "Portable Traffic Signals - Traffic Group Signals," [Online]. Available: <https://www.trafficgroupsignals.com/products/portable-traffic-signals/>. [Accessed 21 04 2023].
- [4] Mervyn Lambert Plant LTD, "Products | Traffic Lights For Sale," [Online]. Available: <https://www.trafficlightsales.co.uk/products>. [Accessed 21 04 2023].

Software listings

/** Main.c file generated by CCS New Project wizard

*

* Created: Wed Mar 22 2023

* Processor: PIC16F877

* Compiler: CCS for PIC

* Author: 2111739

*/

#include <16F877.h>

#include "main.h"

#use delay(crystal = 8MHz)

// Global variables

unsigned long int us_north = 0;

unsigned long int us_east = 0;

unsigned long int us_south = 0;

unsigned long int us_west = 0;

unsigned long int north_south = 0;

unsigned long int east_west = 0;

/** Global state variables */

unsigned int buttons = 0;

unsigned int traffic_state = 0;

/** List holding Seven Segment display sequences */

unsigned int countdown[] = {0x10, 0x9, 0x8, 0x7, 0x6, 0x5, 0x4, 0x3, 0x2, 0x1, 0x0};

/* When a pedestrian button is pressed, this ISR is triggered. */

```

#INT_RB
void RB_isr(void)
{
    printf("INT IN RB \r\n");
    buttons = 1;
    delay_ms(200); // debouncing to prevent multiple triggers
    clear_interrupt(INT_RB);
}

/** Timer 2 cycles through each ADC channel and saves the current value to a global variable */
#INT_TIMER2
void TIMER2_isr(void)
{
    set_adc_channel(0); // Change ADC channel
    delay_us(30);      // Delay to allow the ADC to settle. Required after changing channel and before
reads.
    us_north = read_adc(); // Storing the ADC value in a global variable

    set_adc_channel(1);
    delay_us(30);
    us_east = read_adc();

    set_adc_channel(2);
    delay_us(30);
    us_south = read_adc();

    set_adc_channel(3);
    delay_us(30);
    us_west = read_adc();

    clear_interrupt(INT_TIMER2); // Clearing the interrupt so it can trigger again.
}

int main(void)
{
    setup_adc_ports(AN0_AN1_AN2_AN3_AN4);
    setup_adc(ADC_CLOCK_INTERNAL);

    setup_timer_2(T2_DIV_BY_1, 125, 16); // Interrupt for reading ADC values

    stop_traffic();          // Stop all traffic before interrupts are enabled
    output_d(0xff);          // Invalid BCD combination to turn the Seven Segment Displays off

    enable_interrupts(INT_RB); // Turn on Interrupts for changes on pins B4-7 (Pedestrian buttons)
    enable_interrupts(INT_TIMER2); // Turn on ADC read interrupt
    enable_interrupts(GLOBAL);

    unsigned char count = 0x10;

```

```

while (True)
{
    if (buttons != 0)
    {
        if (traffic_state == 1) {
            red_traffic(LIGHT_N_R, LIGHT_N_A, LIGHT_S_R, LIGHT_S_A);
        } else if (traffic_state == 2) {
            red_traffic(LIGHT_E_R, LIGHT_E_A, LIGHT_W_R, LIGHT_W_A);
        }
        disable_interrupts(GLOBAL);
        delay_ms(3000);
        pedestrians();
        enable_interrupts(INT_RB); // Re-enable interrupts
        enable_interrupts(INT_TIMER2);
        enable_interrupts(GLOBAL);
    }
    else
    {
        check_for_traffic(); // Use the values read from the ADC to determine if there is any traffic
    }
}
}

```

/** Checks for traffic across North-South and East-West traffic lights */

```

void check_for_traffic()
{
    north_south = us_north + us_south;
    east_west = us_east + us_west;

    if (north_south + 1 > east_west) // Prioritise the side with more traffic
    {
        if (traffic_state == 2 | traffic_state == 3) // If traffic is moving in another direction
        {
            red_traffic(LIGHT_E_R, LIGHT_E_A, LIGHT_W_R, LIGHT_W_A);
            amber_traffic(LIGHT_N_R, LIGHT_N_A, LIGHT_S_R, LIGHT_S_A);
        }
        else if (traffic_state == 1) // If already green
            break;
        traffic_state = 1;
        green_traffic(LIGHT_N_R, LIGHT_N_A, LIGHT_S_R, LIGHT_S_A);
    }
    else if (east_west > north_south) // Prioritise the side with more traffic
    {
        if (traffic_state == 1 | traffic_state == 3)
        {
            red_traffic(LIGHT_N_R, LIGHT_N_A, LIGHT_S_R, LIGHT_S_A);

```

```

    amber_traffic(LIGHT_E_R, LIGHT_E_A, LIGHT_W_R, LIGHT_W_A);
}
else if (traffic_state == 2)
    break;
traffic_state = 2;

green_traffic(LIGHT_E_R, LIGHT_E_A, LIGHT_W_R, LIGHT_W_A);
}
else
{
    traffic_state = 3;
}
}

void amber_traffic(int signal_1_r, int signal_1_a, int signal_2_r, int signal_2_a)
{
    delay_ms(1000);
    output_high(signal_1_a); // Get ready
    output_high(signal_2_a);
    delay_ms(1000);
}

void green_traffic(int signal_1_r, int signal_1_a, int signal_2_r, int signal_2_a)
{
    output_low(signal_1_r);
    output_low(signal_2_r); // Go
    output_low(signal_1_a);
    output_low(signal_2_a);
}

void red_traffic(int signal_1_r, int signal_1_a, int signal_2_r, int signal_2_a)
{
    output_high(signal_1_a); // Stop if safe
    output_high(signal_2_a);
    delay_ms(1000);
    output_high(signal_1_r); // Stop
    output_high(signal_2_r);
    output_low(signal_1_a);
    output_low(signal_2_a);
    delay_ms(1000);
}

void stop_traffic()
{
    output_high(LIGHT_N_R); // Turn all lights red
    output_high(LIGHT_E_R);
    output_high(LIGHT_S_R);
    output_high(LIGHT_W_R);
}

```

```
output_low(LIGHT_N_A);  
output_low(LIGHT_E_A);  
output_low(LIGHT_S_A);  
output_low(LIGHT_W_A);  
}
```

```
void pedestrians()  
{  
    stop_traffic();  
    output_high(SSD_LEFT_EN); // Enable Seven Segment Displays  
    output_high(SSD_RIGHT_EN);  
    for (int i = 0; i < 11; i++)  
    {  
        output_d(countdown[i]);  
        delay_ms(1000);  
    }  
    output_low(SSD_LEFT_EN); // Disable Seven Segment Displays  
    output_low(SSD_RIGHT_EN);  
  
    buttons = 0; // Turn off pedestrian waiting flag  
}
```