

sean upton E1

by Sean Upton

Submission date: 28-Nov-2025 02:59PM (UTC+0000)

Submission ID: 269000651

File name: 57737_Sean_Upton_sean_upton_E1_634321_735017052.docx (2.05M)

Word count: 2176

Character count: 10938

ACDB5002 ELECTRONICS 1

Assignment

Sean Upton

2410530

Abstract	3
Introduction	3
Background research	4
Fixed time control	4
Actuated control.....	4
circuit design	5
flowchart/pseudocode.....	6
Simulation.....	7
Set up	7
Case 1&2	8
Case 3	11
Case 4	13
Case 5	14
Results.....	15
Conclusion.....	16
Reference table	17

Abstract

The purpose of this project is to design and implement a traffic light system for a three-way T junction, intended to be used by pedestrians as well as vehicles. The project is based on a fixed time system that operates over a 40 second period. The software development was done using CCS compiler and simulated and tested using Proteus. The specifications require a 7-segment display that must count in reverse from 40.

Introduction

Across the world, there are countless standards for various types of equipment, which can create challenges during product development. However, no matter where you are, you will find the familiar red, yellow, and green traffic light system. The traffic light system as we know it today was first introduced in the 1920s [1] with vehicles becoming more accessible to the public, and there has been little innovation since then. This is a testament to the simple yet effective design that continues to allow road users to travel safely around the world.

The aim of this assignment is to develop a working three-way traffic light system that ensures safe driving for vehicle users and safe crossing for pedestrians. The design features four different scenarios, allowing each direction of the T-junction to proceed in turn, as well as a scenario where all vehicles must stop so pedestrians can cross safely. The full cycle operates over a forty-second period, with each scenario lasting ten seconds. This predictable sequence helps maintain a steady flow of traffic. A forty-second timer continuously counts down so that users can anticipate when the lights will change. Once the cycle is complete, the system automatically repeats.

The software used in this project includes the CCS Compiler, which is used to develop and process the code, and Proteus simulation software, which is used for visualisation and testing. The code will be run using a PIC16 microcontroller and is used to interface with the traffic lights and the 7-segment timer. The assignment will cover the research, design, and development of the project, followed by testing and a conclusion discussing the overall process.

Background research

Everyday people rely on traffic light systems to ensure safe and efficient travel, allowing them to commute to work on time with minimal delays, so it is important that the systems that are in place are effective at reducing the queue of traffic.

Fixed time control

The way in which traffic is controlled can vary depending on the situation at hand. In the past the most common way of controlling traffic lights was by having preset timer which would change the lights orientation. These timings were set by using historical observations [2], for example at 5pm it may allow for a longer green lights at a junction next to an industrial estate, this is done to ease the car queue line and have a better flow of traffic. This had its advantages but was not the most effective solution as it could only work to the pre-determined settings. One of the more popular methods of calculating optimal cycle time is the websters equation [3]

$$C_0 = \frac{1.5L + 5}{1 - \Sigma y}$$

Actuated control

The most common method used today is actuated control. This method of control uses live input to detect if vehicles are present, one example of this is using an induction loop which is installed below the roads surface. The presence of a vehicle alters the magnetic field, and this is processed by the control system, and it will adjust the traffic lights accordingly. This allows for a smoother flow of traffic and less delays. [4].

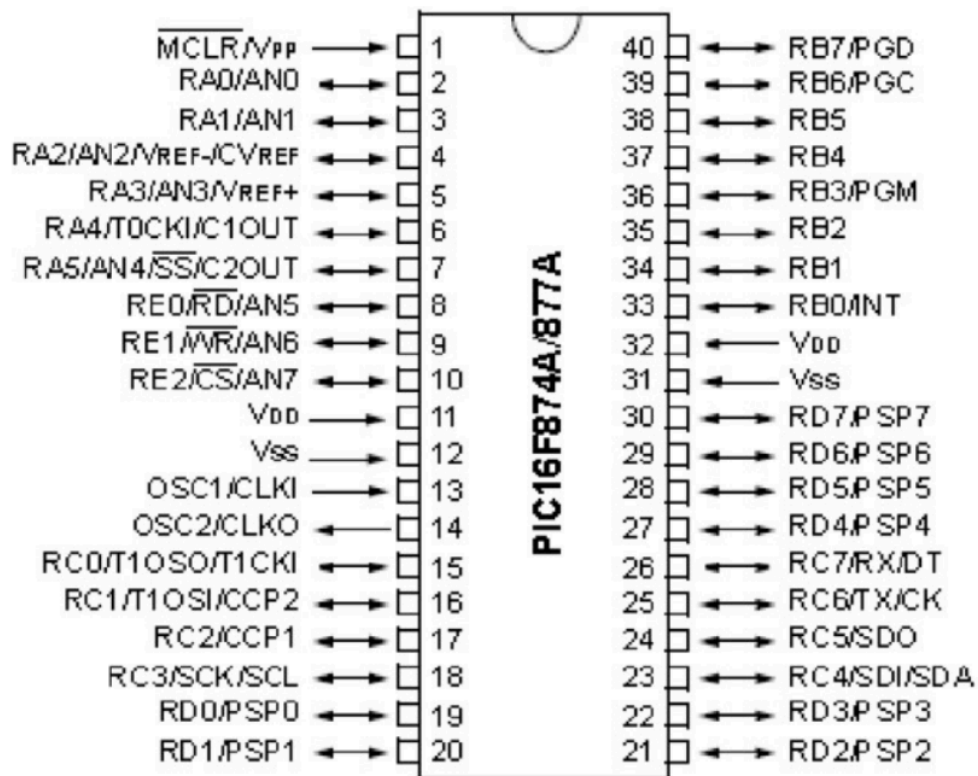
The most common adaptive system is the SCOOT (split cycle optimisation technique) it uses various sensors such as inductive loops and cameras and can adjust cycle time depending on the live inputs.

To ensure the traffic light system meets the UK regulations the “Traffic Signs Manual 2018” should be followed, paying close attention to chapter 6 “traffic control”. This highlights the set requirements for a T- junction [5].

circuit design

The design is a fixed time control system which operates over a 40 second cycle. 30 seconds for vehicles and 10 seconds for pedestrians to cross safely. The 7-segment display, displays where in the cycle the program is allowing for a predictable series of flow. The use of fixed control requires less time to develop and requires a simpler program. To have an actuated system would rely on extensive research and development as there is more room for errors with the code which could result in 2 lights displaying green, this would result in a serious safety concern.

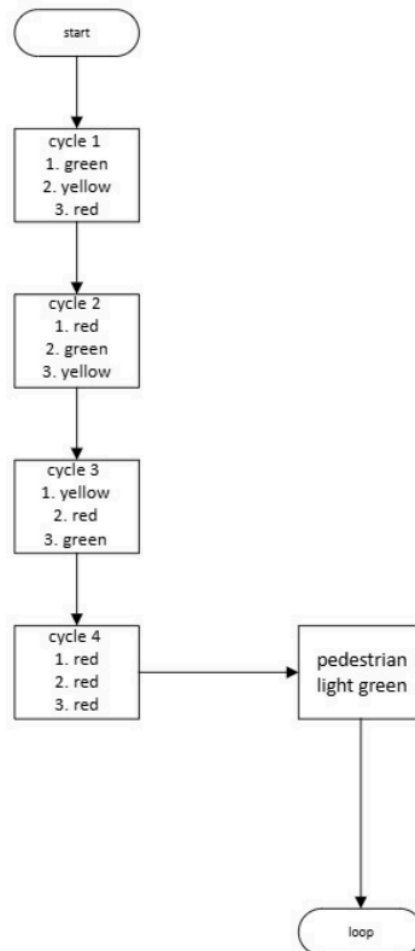
The microcontroller to be used is the PIC16F877A. This allows enough output pins for the 7 segment displays and the LEDs. It also leaves enough pins for future development such as user inputs. The PIC16F877A is low cost being around £5, due to this it is appropriate to use this to leave extra pins free. If a smaller microcontroller was to be used a PIC16F887 could still work. The 877A has 40 pins whereas the 887 has 28 pins [6]. As only PORTB, PORTC and PORTD pins are being used on the 877A the 887 would be acceptable. There is not much different on cost, costing £2.07. As only one unit is being used it seems more appropriate to use the 877A and leave room for expansion.



[Image 1]

flowchart/pseudocode

The flowchart 1 shows how the program is designed to work, have various cycles which allows each vehicle 10 seconds on a green signal. This will take 30 seconds. After this all-traffic flow lights display a will a red signal to allow the pedestrians to cross safely. Once this has been done the program will loop back to cycle 1.



[Flowchart 1]

Simulation

Set up

Here is where the set up for the code is complete, firstly setting the clock for the microcontroller, this is set at 20MHz which is the upper limit for the microcontroller being used, this is allowing for 20 million cycles per second, maximising the microcontrollers responsiveness. To make sure that this syncs with the simulation, the microcontroller on Proteus must also be set to 20mhz, this would result in timing issues if it was not complete.

The define pins are setting the names for each output, this makes the code look more user friendly and easier to follow. The microcontroller uses hexadecimal which requires the decimal number to be converted so they can be understood by the microcontroller.

```
3 #use delay(clock=20000000)
4
5 #define LED_1 PIN_B0
6 #define LED_2 PIN_B1
7 #define LED_3 PIN_B2
8 #define LED_Red1 PIN_B3
9 #define LED_Green1 PIN_B4
10 #define LED_wait PIN_B5
11 #define BUTTON PIN_A0
12
13 unsigned int display [10] = {0x90, 0x80, 0xF8, 0x82, 0x92, 0x99, 0xB0, 0xA4, 0xF9, 0xC0};
14
15 void setup_hw(void) {
16     set_tris_a(0x01);
17     output_a(0x00);
18
19     set_tris_b(0x00);
20     output_b(0x00);
21
22     set_tris_c(0x00);
23     output_c(0x00);
24
25     set_tris_d(0x00);
26     output_d(0x00);
27 }
```

[image 2]

Case 1&2

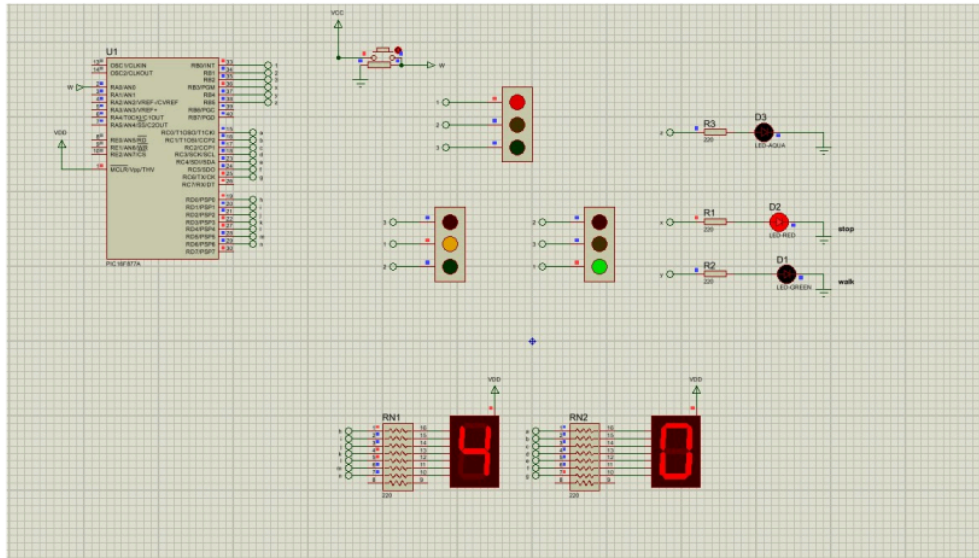
In the code there are 5 cases which allows for all the different scenarios required to play through. case 1 and 2 is responsible for the first scenario where the first set of lights are green and the rest are red. This needed to be split into 2 cases to allow for 7-segment display to work as intended. As shown in the code below it is shown that $i=9$, this is to allow the timer to start on 40 seconds, output D is constantly displaying the digit 4 and output C is counting down from 0-9. However as $i=9$ it will only display for 1 second. This will run for 1 second before proceeding to case 2. this case is showing the same outputs so results in no visual change for the traffic lights. Output D is shown as 0xB0 this is causing the 7-segment display to show the number 3.

```
30 void case1(void) {  
31     int i;  
32     for (i = 9; i < 10; i++) {  
33         output_d(0x99);  
34         output_c(display[i]);  
35     }
```

[Image 3]

```
35
36     output_high(LED_1);
37     output_low(LED_2);
38     output_low(LED_3);
39     output_high(LED_Red1);
40     output_low(LED_Green1);
41     delay_ms(1000);
42
43 }
44
45 }
46
47 void case2(void) {
48     int i;
49     for (i = 0; i < 10; i++) {
50         output_d(0xB0);
51         output_c(display[i]);
52
53         output_high(LED_1);
54         output_low(LED_2);
55         output_low(LED_3);
56         output_high(LED_Red1);
57         output_low(LED_Green1);
58         delay_ms(1000);
```

[image 4]



[image 5]

Case 3

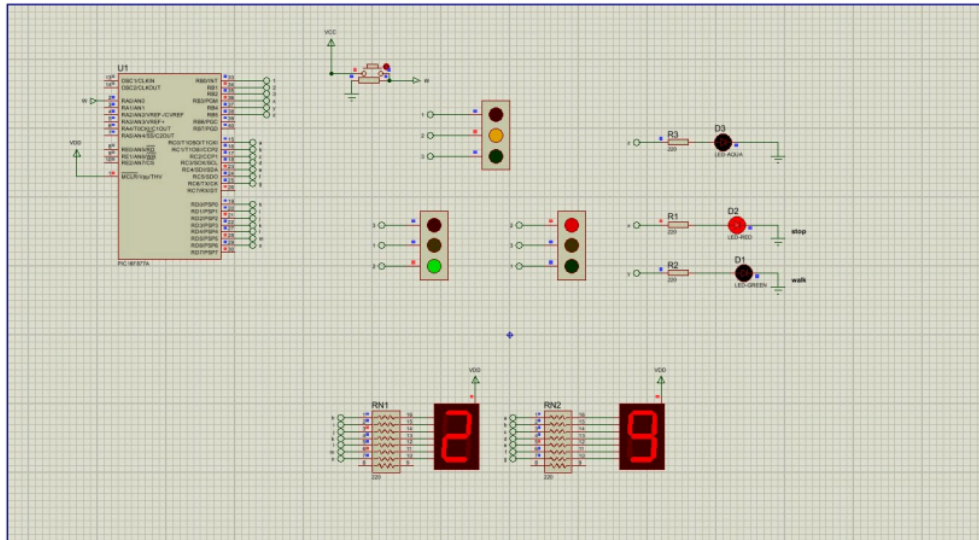
Case three is the first visible change in the traffic lights. the green has now been turned on to the second set of traffic lights, and the rest don't allow for passage. By looking at the output, you can see that LED2 is told to be high. It could be assumed that this would cause the yellow lights to be activated. However, when looking at the schematic you can see that each set of lights has the output LED1, LED2 and LED3 at different points, this allows for a simpler code and stops more than 1 light being green at a given point. Output D is displaying the number 2 and output C is counting from 9-0. Once this is complete the program moves onto case 4.

```
63 void case3(void) {  
64     int i;  
65     for (i = 0; i <10; i++) {  
66         output_d(0xA4);  
67         output_c(display[i]);  
68     }
```

[image 6]

```
69     output_low(LED_1);  
70     output_high(LED_2);  
71     output_low(LED_3);  
72     output_high(LED_Red1);  
73     output_low(LED_Green1);  
74     delay_ms(1000);  
75 }
```

[image 7]



[Image 8]

Case 4

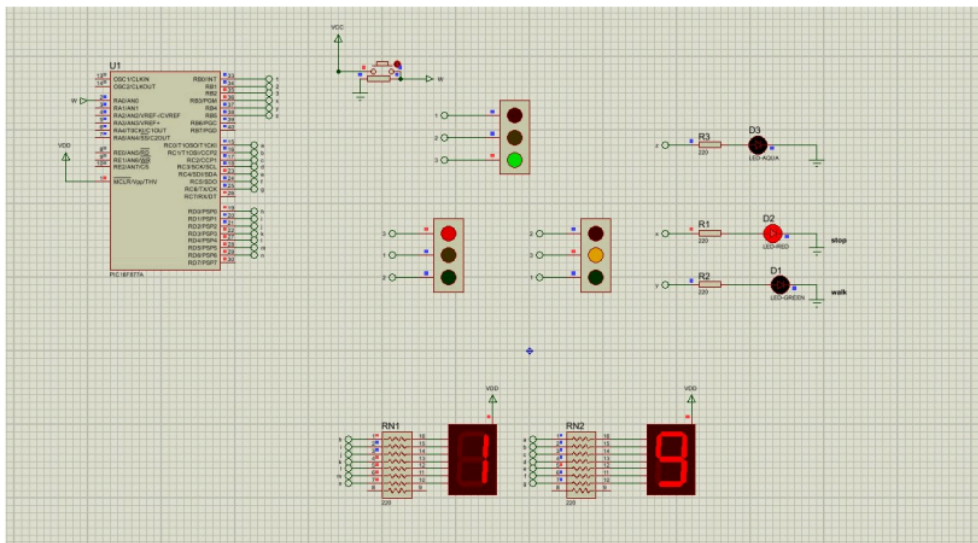
Next, we have case 4 which displays green on the final set of lights. Working the same as the previous cases, displaying the number 1 from output D and output C counting down from 9-0.

```

78 void case4(void) {
79     int i;
80     for (i = 0; i < 10; i++) {
81         output_d(0xF9);
82         output_c(display[i]);
83
84         output_low(LED_1);
85         output_low(LED_2);
86         output_high(LED_3);
87         output_high(LED_Red1);
88         output_low(LED_Green1);
89         delay_ms(1000);
90     }

```

[Image 9]



[image 10]

Case 5

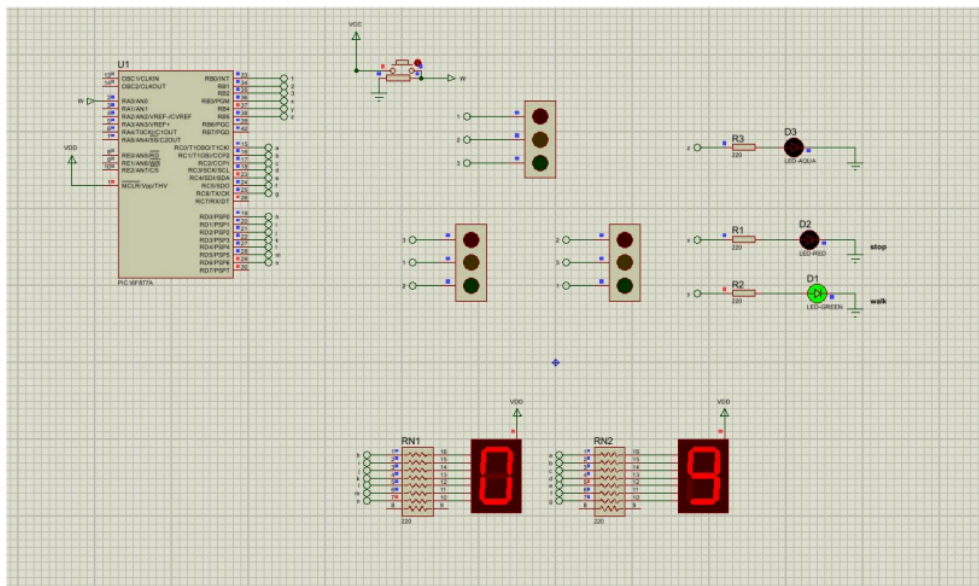
Finally, case 5 is to allow for the safe crossing of the pedestrians. Output D is now displaying 0 and output C is counting down from 9-0. All sets of traffic lights are displaying the red light; the green pedestrian light is now displaying and is giving them 10 seconds to cross. Once this is complete the code will then loop back to case one and the whole cycle will be repeated indefinitely.

```

93 void case5(void) {
94     int i;
95     for (i = 0; i < 10; i++) {
96         output_d(0xC0);
97         output_c(display[i]);
98
99         output_low(LED_1);
100        output_low(LED_2);
101        output_low(LED_3);
102        output_low(LED_Red1);
103        output_high(LED_Green1);

```

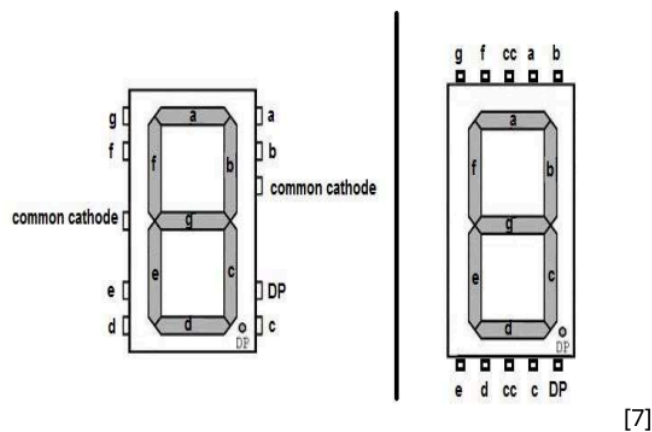
[image 11]



[image 12]

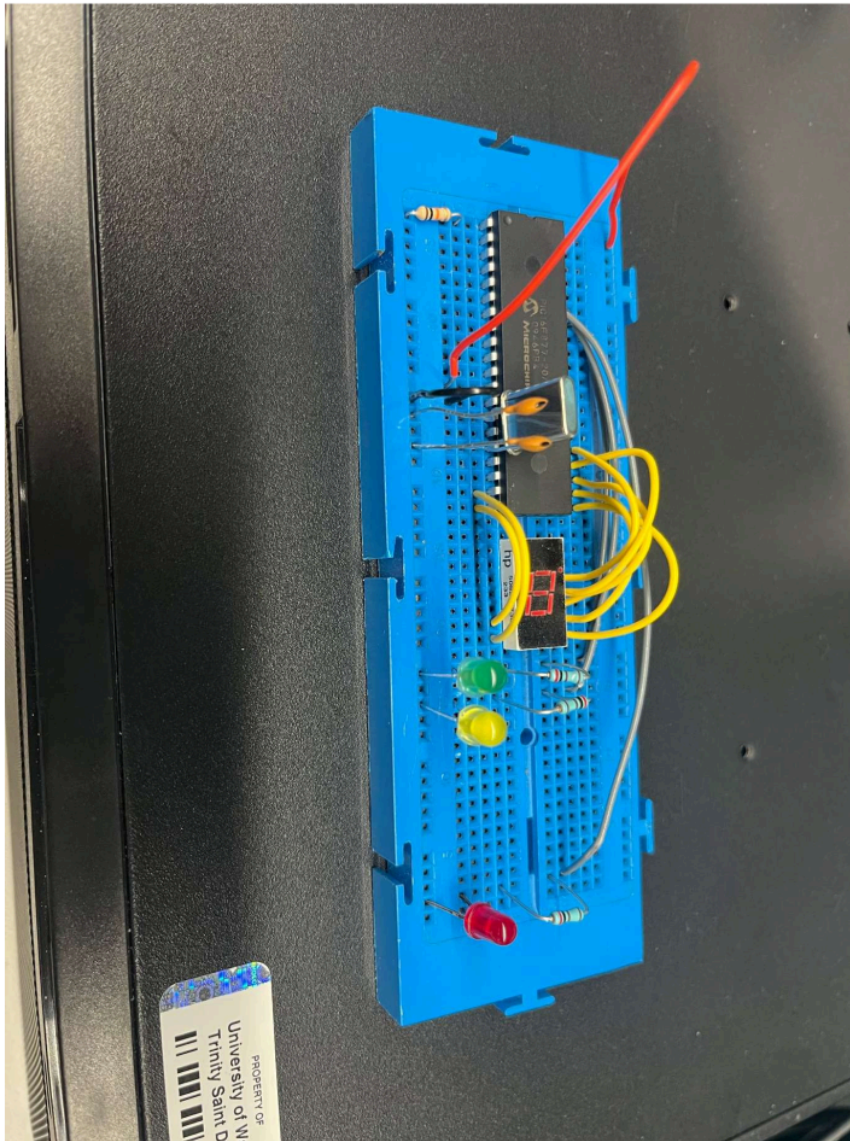
Results

The image below shows the simplified circuit which is used as a physical prototype. It includes a single 7-segment display and the first set of LEDs, the PIC16F877A is connected to a 20MHz crystal oscillator which is necessary for the microcontrollers timing. The transfer the program to the microcontroller the “PICKIT3” is used alongside MPLAB IPE using the .hex file from PCW.



[Image 13]

The 7-segment display has 10 pins used to control what number is displayed. When a pin gets a high input, it lights up the corresponding LED on the display. Having the pin out allows you to easily find any faults with the circuitry as if the cables are corrected incorrectly the number won't be displayed as intended.



Conclusion, recommendations on future work

The smooth and safe flow of traffic is dependent on continuous innovation the use of actuated control systems proves to be the leading method of doing so. The various sensors and processors allow for live control to ease queues of vehicles. The method of timed systems used in the project now appear to be outdated but provide a solid foundation for understanding and designing traffic light systems. The program operates

over a 40 second cycles giving drivers adequate time to pass and passengers to cross. The use of the PIC16f887A allows for room for expansion if there was ever a need for more inputs or outputs. However, to save costs the PIC16f887 could be used.

The software for the project was complete using CCS Compiler and simulated using Proteus, this showed that all 5 cases work as intended, as shown in the flowchart. The use of the 7-segment allows for a predictable interface which benefits all road users.

In the program used there were 5 cases where each light was always a different colour, this is not accurate to how real systems work. The yellow light was on for 10 seconds and this could be misleading to drivers as it usually only yellow for 1-2 seconds. The Traffic Signs Manual states that “The period during which an amber signal is displayed is fixed at 3 seconds (s) and the red/amber signal at 2 s, with a tolerance of 0.25 s either way.” Another issue from the project was developing the prototype. In future further research would need to be complete to achieve a working physical product in the allowed time.

Overall, the traffic light control system that has been developed highlights the fundamental principles of traffic light systems. Research has been conducted on the advantages and disadvantages of the timed system and actuated system. With the actuated control appearing to be more appropriate in areas which have larger flows of traffic such as large junctions and roundabouts whereas the timed control seems to be more appropriate in quieter areas due to less maintenance. Finally, to improve the project further input pins such as a button for pedestrians to use could be implemented.

Reference table

[1]

“When were the first traffic lights installed?,” *Nationalmotormuseum.org.uk*, 2024.
<https://nationalmotormuseum.org.uk/help-centre/motoring-firsts/when-were-the-first-traffic-lights-installed/>

[2]

L. Li, D. Wen, and D. Yao, “A Survey of Traffic Control With Vehicular Communications,” *IEEE Transactions on Intelligent Transportation Systems*, vol. 15, no. 1, pp. 425–432, Feb. 2014, doi: <https://doi.org/10.1109/tits.2013.2277737>.

[3]

K. B. Kesur, “Optimization of mixed cycle length traffic signals,” *Journal of Advanced Transportation*, vol. 48, no. 5, pp. 431–442, May 2012, doi: <https://doi.org/10.1002/atr.1190>.

[4]

A. Mocholi-Salcedo, J. H. Arroyo-Nunez, V. M. Milian-Sanchez, M. J. Palomo-Anaya, and A. Arroyo-Nunez, "Magnetic Field Generated by the Loops Used in Traffic Control Systems," *IEEE Transactions on Intelligent Transportation Systems*, vol. 18, no. 8, pp. 2126–2136, Dec. 2016, doi: <https://doi.org/10.1109/tits.2016.2632972>.

[5]

"Traffic Signs Manual," 2018. Available:

<https://assets.publishing.service.gov.uk/media/5df0e29fed915d15f42c4820/dft-traffic-signs-manual-chapter-6.pdf>

[6]

"Dynamic Product Page | Microchip Technology," www.microchip.com.

<https://www.microchip.com/en-us/product/PIC16F877A>

[7]

CD-Team, "7 Segment Display Pinout and Working | Basics for Beginners," *Electronics For You*, Jun. 04, 2019. <https://www.electronicsforu.com/resources/7-segment-display-pinout-understanding>