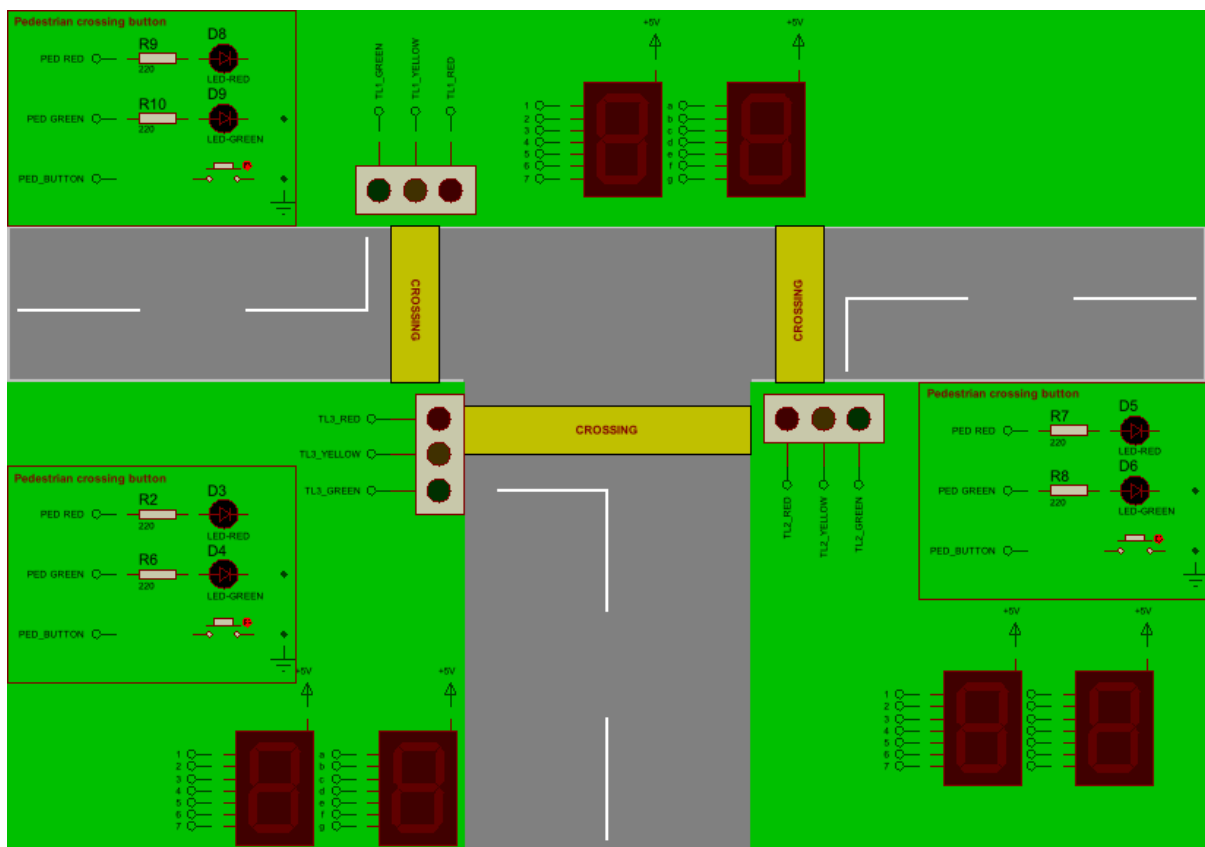


Electronics 1 – ACDB5002

Designing an Automatic Traffic Light Controller for a T-Junction Road



Jayden Fox – 2316329

Table of Contents

1. Introduction	4
2. Background Research.....	5
2.1 Foundations of Traffic Signal Control	5
2.2 Traditional and Time-Based Traffic Light Control Systems.....	5
2.3 Actuated and Adaptive Traffic Control Methods.....	5
2.4 Findings from Existing Literature	6
3. System Requirements	7
4. Circuit Design	9
4.1 Circuit Component List:.....	10
5. System Operation	11
6. Testing and Simulation Results	14
6.1 Prototype	14
6.2 Phase 1	15
6.3 Phase 2	17
6.4 Phase 3	19
6.5 Phase 4	21
6.6 Pedestrian Crossing Acknowledgment	23
7. Conclusion/Recommendations for future work	25
8. Bibliography	26
9. Appendix	28
9.1 Appendix A	28
9.2 Appendix B	28
9.3 Appendix C	33

Table of Figures

Figure 1 - PIC16F877A Pin Diagram [18]	8
Figure 2 - Whole Circuit Schematic Design	9
Figure 3 - Close-up of traffic control system	9
Figure 4 - Close-up of PIC and circuit inputs/outputs.....	10
Figure 5 - 1st half of flowchart.....	12
Figure 6 - 2nd half of flowchart	13
Figure 7 - Breadboard Prototype	14
Figure 8 - Phase 1 Starting (Prototype).....	15
Figure 9 - Phase 1 Starting (Simulation).....	16
Figure 10 - Phase 2 Starting (Prototype).....	17
Figure 11 - Phase 2 Starting (Simulation)	18
Figure 12 - Phase 3 Starting (Prototype).....	19
Figure 13 - Phase 3 Starting (Simulation)	20
Figure 14 - Phase 4 Starting (Prototype).....	21
Figure 15 - Phase 4 Starting (Simulation)	22
Figure 16 - Pedestrian crossing button press indication (Prototype).....	23
Figure 17 - Pedestrian crossing button press indication (Simulation).....	24
Figure 18 - Mapping microcontroller pins	28
Figure 19 - Declaring phase functions and hardware setup function.	28
Figure 20 - Bit pattern array for 7-segment 'countdown'.....	29
Figure 21 - Interrupt routine and hardware setup function.....	29
Figure 22 - Phase 1 code	30
Figure 23 - Phase 2 code	31
Figure 24 - Phase 3 code	31
Figure 25 - Phase 4 code	32
Figure 26 - 'Main' function initialisation code	33
Figure 27 - Main function 'while' loop code	33

1. Introduction

In urban traffic management and operations, a well-designed traffic control system is very important for the safety of all road users. By controlling the movement of vehicles and pedestrians on roads, it can significantly reduce the number of accidents that occur. This project focuses on the creation of a traffic light control system for a T-junction, as it is a common site where accidents are likely without effective traffic regulation.

The primary objective of this design is to implement a time-based signal control system that ensures predictable and smooth traffic flow, while also considering an appropriate time slot for pedestrian safety. The system will utilise a PIC microcontroller that will be programmed in C using the CCS compiler and simulated in Proteus. The setup includes three traffic lights per road, countdown timers on 7-segment displays, and a dedicated pedestrian crossing phase to ensure the safety of all users of the T-junction.

By using precise timing intervals and clear visual indications to create phases for all types of road users, this design aims to provide a reliable and efficient solution to automatic traffic management at T-junctions. The following report documents the research, design considerations, circuit development, and simulation results of the project, along with suggestions for future improvements.

2. Background Research

Several scholars have published documentation and reviews on traffic light control architecture, and this literature review focuses on traffic signal control with an emphasis on T-junctions, highlighting how traffic control strategies enhance the safety and efficiency of all road users.

2.1 Foundations of Traffic Signal Control

Traffic signal control addresses traffic congestion coming from a growing increase in urban vehicle demand, where expanding existing infrastructure is often slow or impractical. Therefore, signal timing and traffic control strategies are central to improving the efficiency and safety of traffic [1]. Systematic reviews describe fixed-time, actuated and adaptive applications where fixed-time methods are considered a baseline approach that rely on historical demand and cycle structures [2]. On the other hand, actuated and adaptive traffic control systems respond to real-time conditions to control the imminent traffic [3, 4].

2.2 Traditional and Time-Based Traffic Light Control Systems

Traditional time-based signal designs are a popular method of traffic control as they are easier to implement and offer predictable traffic light cycle changes, particularly at simpler applications like T-junctions [5]. These systems operate on a fixed cycle of signal phases, sometimes adjusted manually during off-peak hours when traffic volumes are lower as there will be less road users [6]. Case studies show that systems with properly configured cycles with defined phase sequences can deliver smooth flows and reduce conflicts between vehicle and pedestrian movements. At T-junctions, fixed-time systems remain common due to simplicity, and it allows all road users to have a predictable and specified time slot to safely use the T-junction [7]. However, fixed-time systems struggle under fluctuating demand, often leading to inefficiencies compared to actuated and adaptive control methods.

2.3 Actuated and Adaptive Traffic Control Methods

Adaptive traffic signal control (ATSC) is a dynamic system that uses real-time data from sensors to automatically adjust traffic light timings to optimise traffic flow [8]. Whereas actuated control is a system where sensors are used to detect vehicles and pedestrians and then automatically adjust signal timing to match real-time demand [3].

These methods have advanced significantly by having the ability to adapt to traffic demand, leading to delay and emissions reductions, through the use of reinforcement learning and other AI techniques [9]. When compared to fixed-time approaches, actuated and adaptive traffic control methods, such as SCOOT (Split Cycle Offset Optimisation Technique), can demonstrate improvements in efficiency. Although, actuated and adaptive systems are typically more complex and costly to install and maintain, making them less practical for smaller applications such as T-junctions [10].

2.4 Findings from Existing Literature

From the background research and literature review, a great understanding of the existing traffic control methods has been gained. Although actuated and adaptive systems (such as SCOOT in London) give significant reductions in delay and emissions, their installation and maintenance costs remain impractical for smaller applications like T-junctions [11]. Existing literature rarely mentions cost-effective embedded solutions for traffic control on T-junctions, and this gap highlights the need for the T-junction tailored solution this project aims to address.

3. System Requirements

The system must be able to control traffic effectively and ensure the safety of all road users on the T-junction. A fixed-time traffic control method will be implemented as it was best suited to the T-junction application. Each of the three sections of the T-junction must have a set of traffic lights and a pair of 7-segment displays, as well as pedestrian crossings with buttons and an indicating LED to ensure all road users know when they are safe to proceed.

The system will need to have different phases so each road can use the T-junction safely, and these phases will need to be timed to allow for predictable and smooth traffic flow. Additionally, the system needs to be cost-effective and must show reliable operation without failure across repeated cycles.

For this project, a variety of microcontrollers were considered, such as the Arduino Uno, STM32s and PIC16 microcontrollers. After reviewing the Arduino Uno, it offered flexibility and simplicity for easy creation of a variety of electronic projects, however, it had limited input/output pins and fewer built-in peripherals compared to the PIC16F877A [12, 13]. As the system's microcontroller needed to handle multiple traffic signals simultaneously, the Arduino Uno was shown to be less suitable for this project. In regard to the STM32 microcontroller, it was shown to provide high processing power and many advanced features, however, they are costly and quite complex to program [14].

To give an idea of the costs of each considered microcontroller:

- Arduino Uno: £22.31 [15]
- STM32F401RET7: £4.80 [16]
- PIC16F877A: £6.49 [17].

This system will utilise a PIC microcontroller, more specifically the PIC16F877A as it is best suited to the traffic light control specification compared to other available microcontrollers. The PIC16F877A stood out as it will provide enough I/O pins for the system's operation (as seen in Figure 1) and it also supports external interrupts on pin RB0/INT, which will allow for a pedestrian crossing button to be used for the pedestrians to cross safely as soon as possible [18]. It was not the cheapest of the microcontrollers that were compared, however, it was still cheap and was good value for the application. The PIC16F877A is widely available and affordable, along with being widely compatible with development tools that will be used like Proteus and the MPLAB IDE [19].

40-Pin PDIP

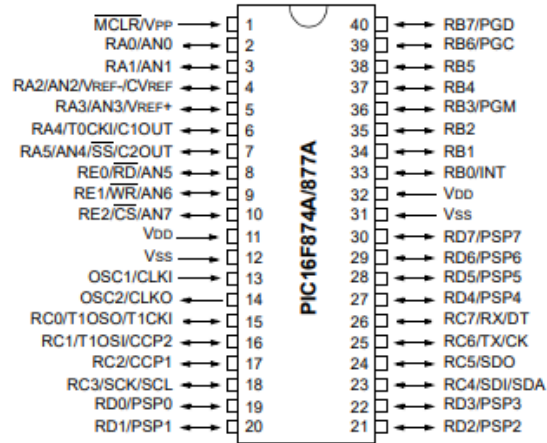


Figure 1 - PIC16F877A Pin Diagram [18]

4. Circuit Design

Based on the requirements of the system, the following circuit schematic (Figure 2) was created to allow for effective traffic control on the T-junction. As shown in Figure 3, each position on the T-junction has a traffic light facing the traffic and a pair of common anode 7-segment displays to show the countdown. Each road also has a pedestrian crossing system, which contains LEDs (green and red) and a button to indicate a pedestrian wants to cross. A separate orange LED, seen connected to pin A1 of the microcontroller in Figure 4, has also been included to inform the pedestrian that their button press has been acknowledged once they have pressed it. 'Default' terminal tags were used throughout the circuit for cleanliness and simplicity of connections to inputs and outputs.

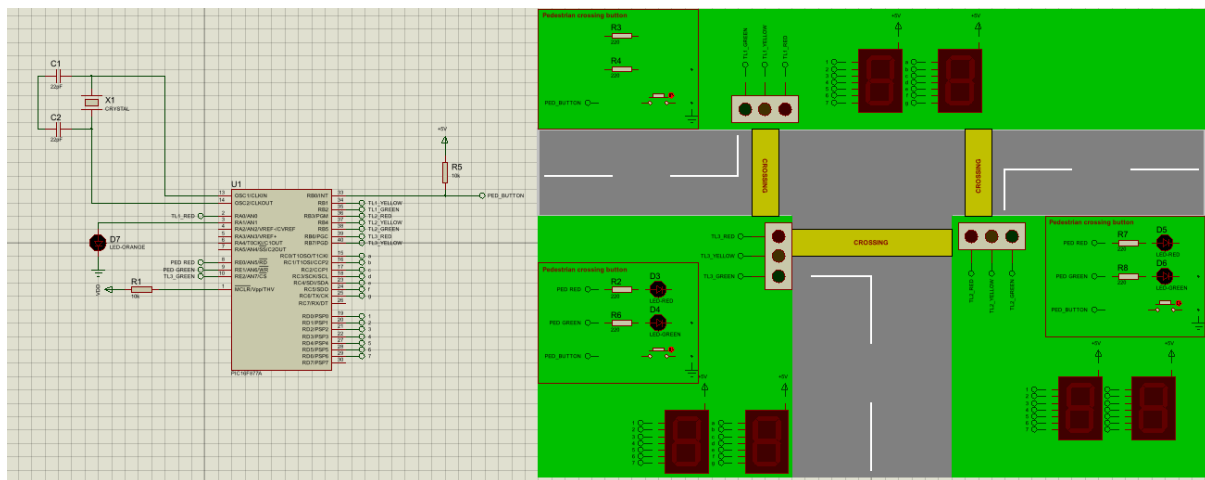


Figure 2 - Whole Circuit Schematic Design

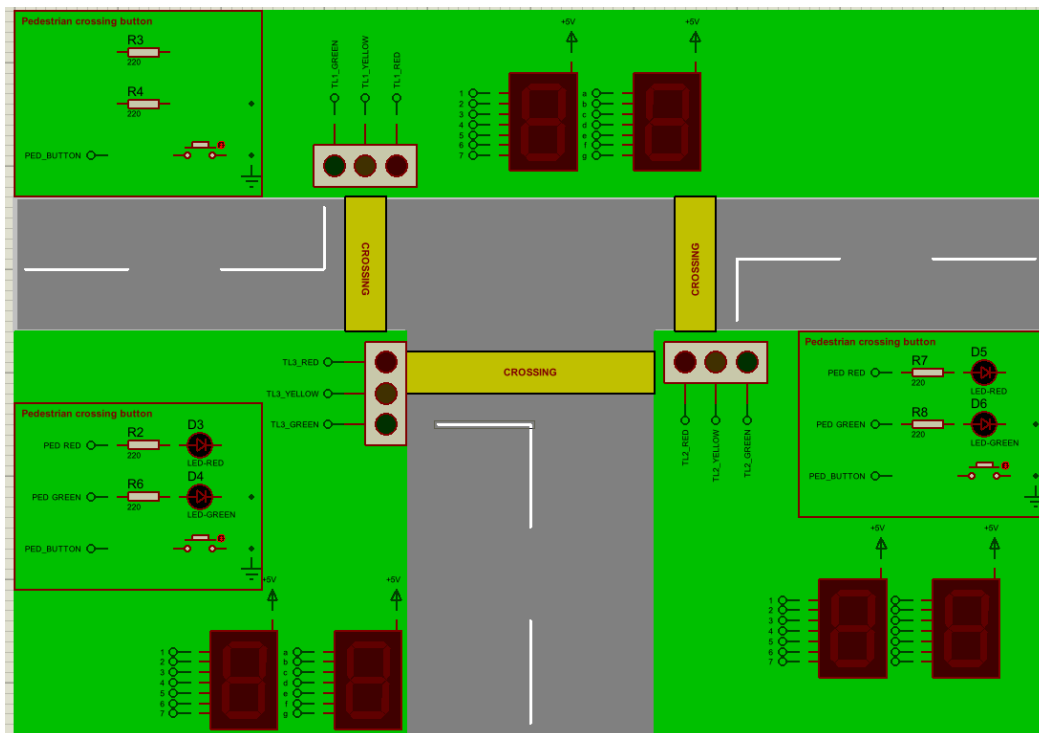


Figure 3 - Close-up of traffic control system

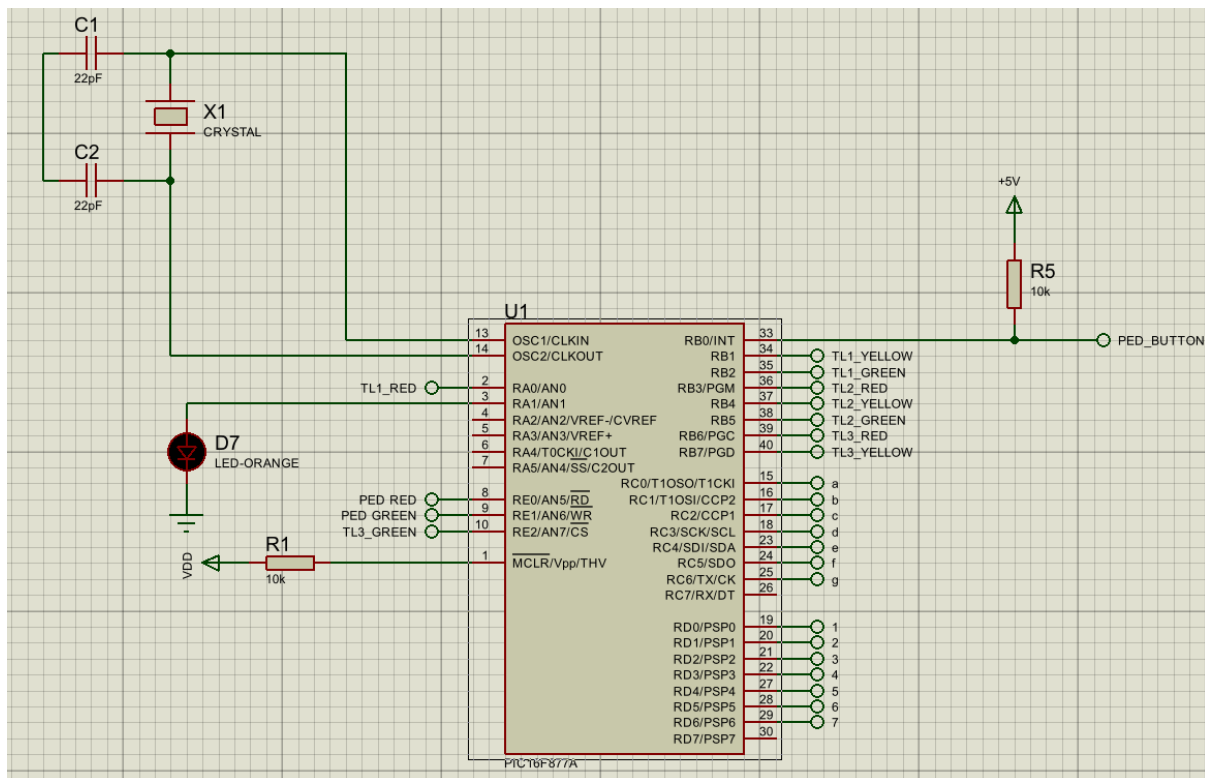


Figure 4 - Close-up of PIC and circuit inputs/outputs

4.1 Circuit Component List:

- 1 x PIC16F877A microcontroller
- 6 x 7-segment displays
- 3 x Traffic Lights (9 pins in total)
- 3 x Green LED (Pedestrian)
- 3 x Red LED (Pedestrian Crossing)
- 3 x Button (Pedestrian Crossing)
- 1 x Orange LED (Pedestrian Crossing Button Indication)
- 1 x Crystal Oscillator (20MHz)
- 2 x 22pF capacitors (value from crystal oscillator datasheet [20])
- 6 x 220 Ω resistor (for pedestrian crossing LEDs)
- 2 x 10k Ω resistor

The 7-segment displays used will be 3 pairs of common anode displays. In a common anode display, the anodes of all the LED segments are connected to the logic '1' [21]. Common anode 7-segment displays are best suited to this system as each segment is controlled by grounding its cathode through the microcontroller, meaning the microcontroller only needs to sink current. The choice of common anode displays simplifies the circuit design and aligns well with the PIC16F877A's current sinking abilities.

5. System Operation

The system operates continuously in a 40-second cycle across four phases, with each phase lasting 10 seconds and updating the traffic lights and pedestrian signals to ensure safe use of the T-junction. When a traffic light is green in one phase, it will turn red in the next. When an amber light is illuminated, it indicates the traffic light is due to turn green in the next phase. The proposed status of each light on the T-junction during the different phases can be seen in Table 1.

Table 1 - A table of how each phase will affect the different lights on the T-junction.

Phase	Traffic Lights 1	Traffic Lights 2	Traffic Lights 3	Pedestrian
1	RED	AMBER	GREEN	RED
2	AMBER	GREEN	RED	RED
3	GREEN	RED	AMBER	RED
4	RED	RED	RED	GREEN

If the pedestrian button is pressed, the system completes the current phase and then runs the pedestrian crossing phase (phase 4). Otherwise, it cycles sequentially through phases 1 to 4. A flowchart (Figures 5-6 and Appendix C) has been created to illustrate this logic, showing the transitions between phases through traffic light changes and includes constant pedestrian crossing button checks between phases for pedestrian safety.

Each phase follows a similar system where it illuminates the correct traffic lights on each road and checks to see if the cycle of the phase has ended based on the countdown. Each phase checks to see if the pedestrian button has been pressed and if it has, it will jump to the pedestrian crossing phase upon the conclusion of the current phase. If the pedestrian crossing button is never pressed, the system flows sequentially from phase 1 to phase 4. The system runs indefinitely to ensure safe use of the T-junction for vehicles from all three roads and pedestrians.

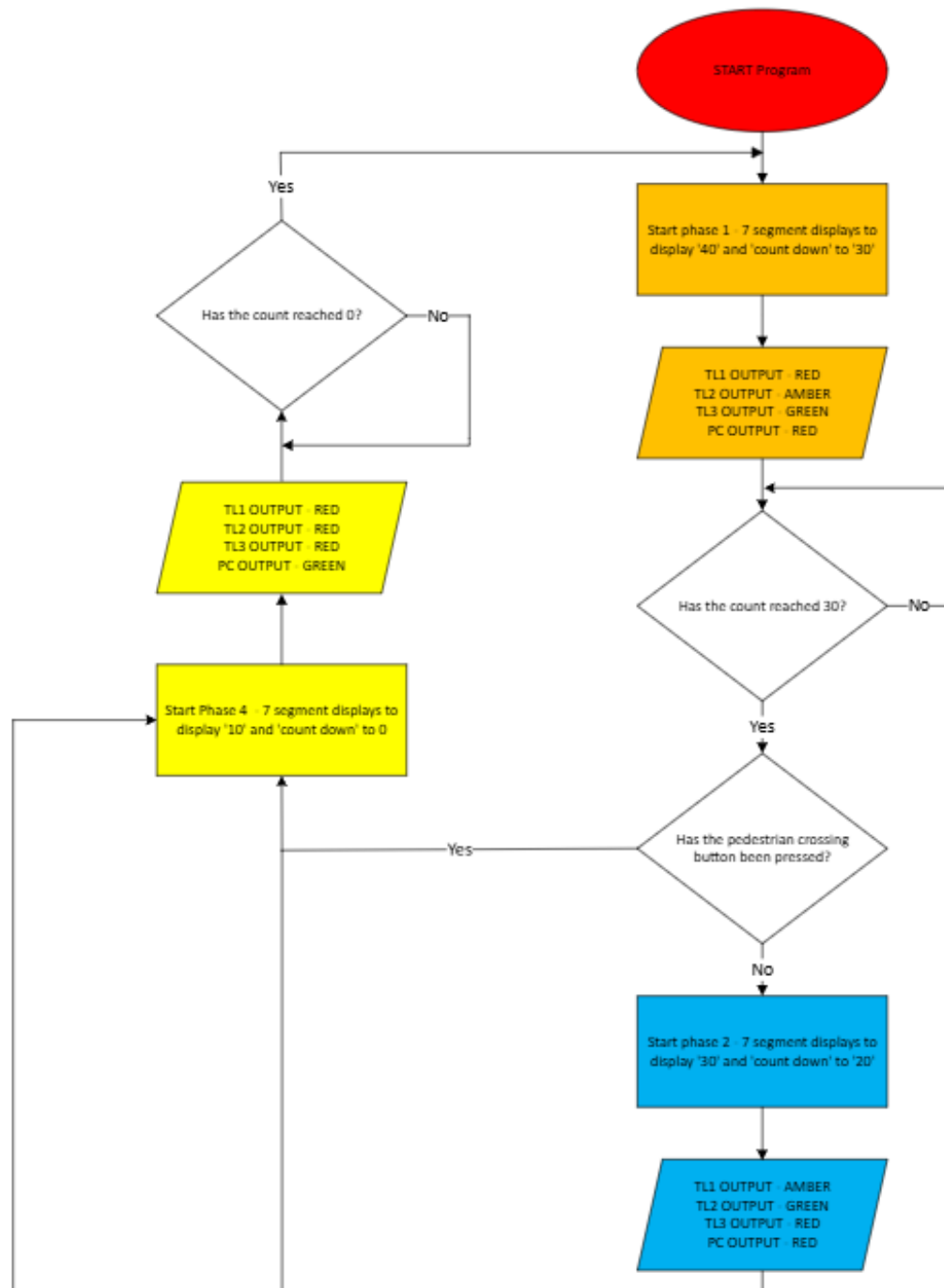


Figure 5 - 1st half of flowchart

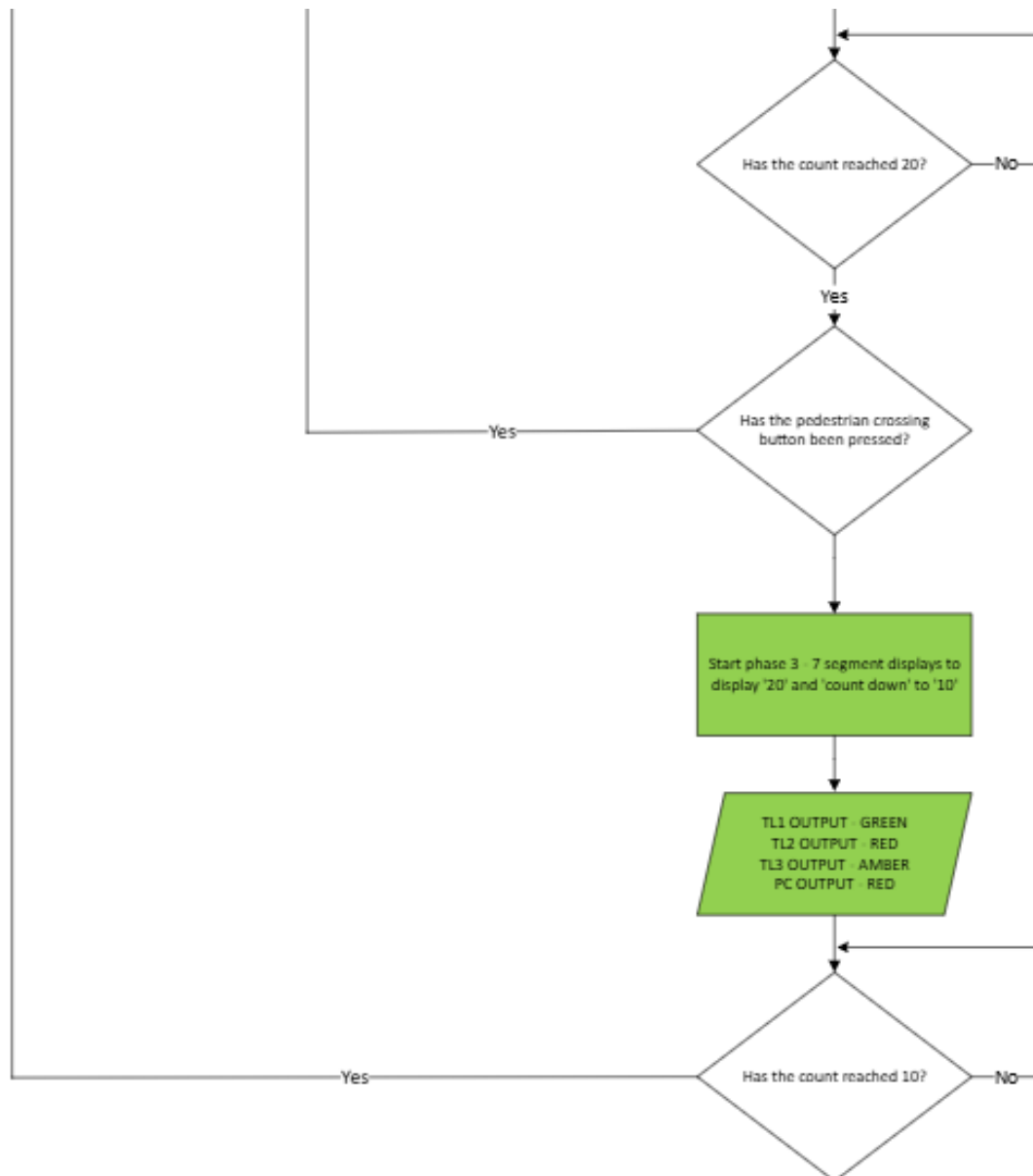


Figure 6 - 2nd half of flowchart

The program's code was programmed onto the PIC16F877A microcontroller using the CCS compiler and Proteus, for the system to operate. The program was coded in C and has been fully commented to explain each line's function. The program defines the system's input and output pin mappings, all four phase functions, and an interrupt routine to deal with any pedestrian crossing requests. In the C program, each phase function represents a 10-second timed state in the traffic light control system, by using a 'for' loop with a 1000ms delay per iteration. The full C code of the program and in-depth explanation of it can be found in Appendix B.

6. Testing and Simulation Results

This section of the report focuses on the testing of the system and program, along with a detailed insight into the results of the Proteus simulation and breadboard circuit. A video demonstration of the circuit has been included in the supporting files of this project (also linked in Appendix A).

6.1 Prototype

A breadboard prototype (Figure 7) was created to give a real-world example of the system's function, and test the program's compatibility and functionality to the T-junction application. The breadboard circuit and program functioned as expected, implementing the alternating traffic light control across the 3 sets of traffic lights in the correct time intervals.

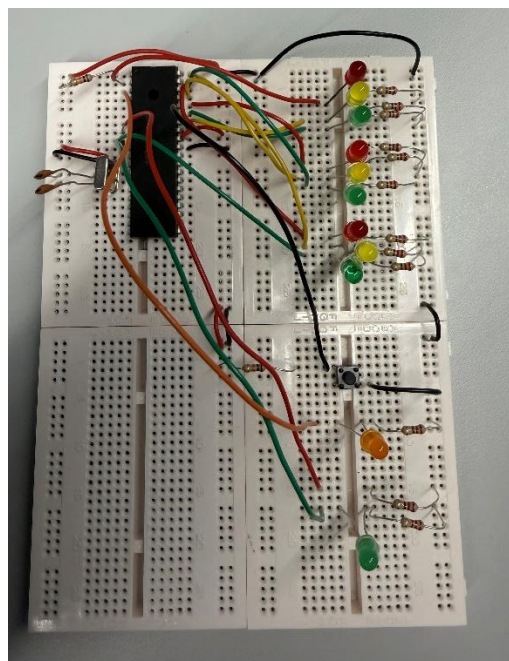


Figure 7 - Breadboard Prototype

Following the system simulations in Proteus, the status of the T-junction traffic lights at each phase is presented in Table 2. The system operates correctly, progressing sequentially through phases 1 to 4, with each transition occurring at the specified time interval. A further explanation of the results at each phase is provided below, with screengrabs to demonstrate the circuit simulation.

Table 2 - A table to show the simulation results of each set of lights during each phase

Phase	Traffic Lights 1	Traffic Lights 2	Traffic Lights 3	Pedestrian
1	RED	AMBER	GREEN	RED
2	AMBER	GREEN	RED	RED
3	GREEN	RED	AMBER	RED
4	RED	RED	RED	GREEN

6.2 Phase 1

The cycle starts at phase 1, which allows vehicles on the central road position to safely use the T-junction. Table 3 displays the status of each traffic light set during phase 1, with the matching prototype and simulation outputs shown in Figure 8 and Figure 9.

Table 3 – Traffic Light Statuses during Phase 1.

Road/Junction Entry Point	Traffic Light Number	Current Illuminated Light
Leftmost Road	Traffic Lights Set 1 (TL1)	RED
Right Road	Traffic Lights Set 2 (TL2)	AMBER
Central Road	Traffic Lights Set 3 (TL3)	GREEN
Pedestrian Crossings	Pedestrian Crossing	RED

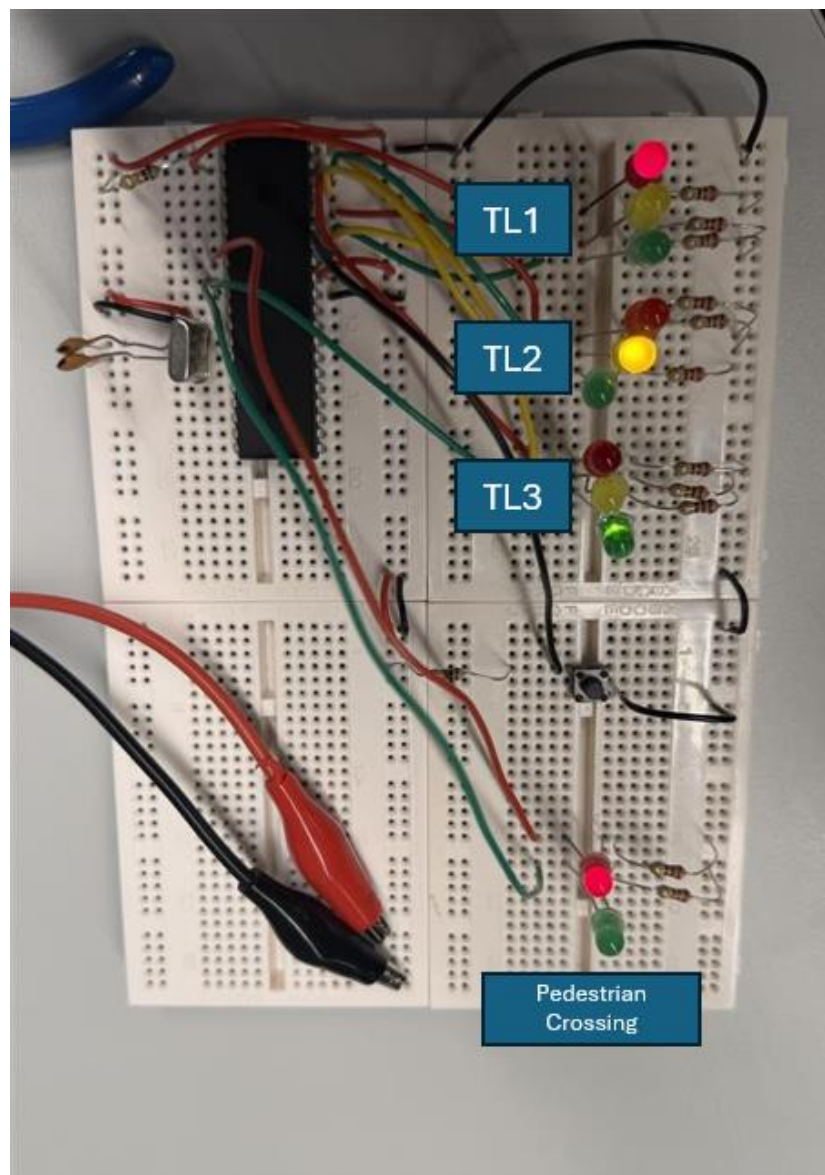
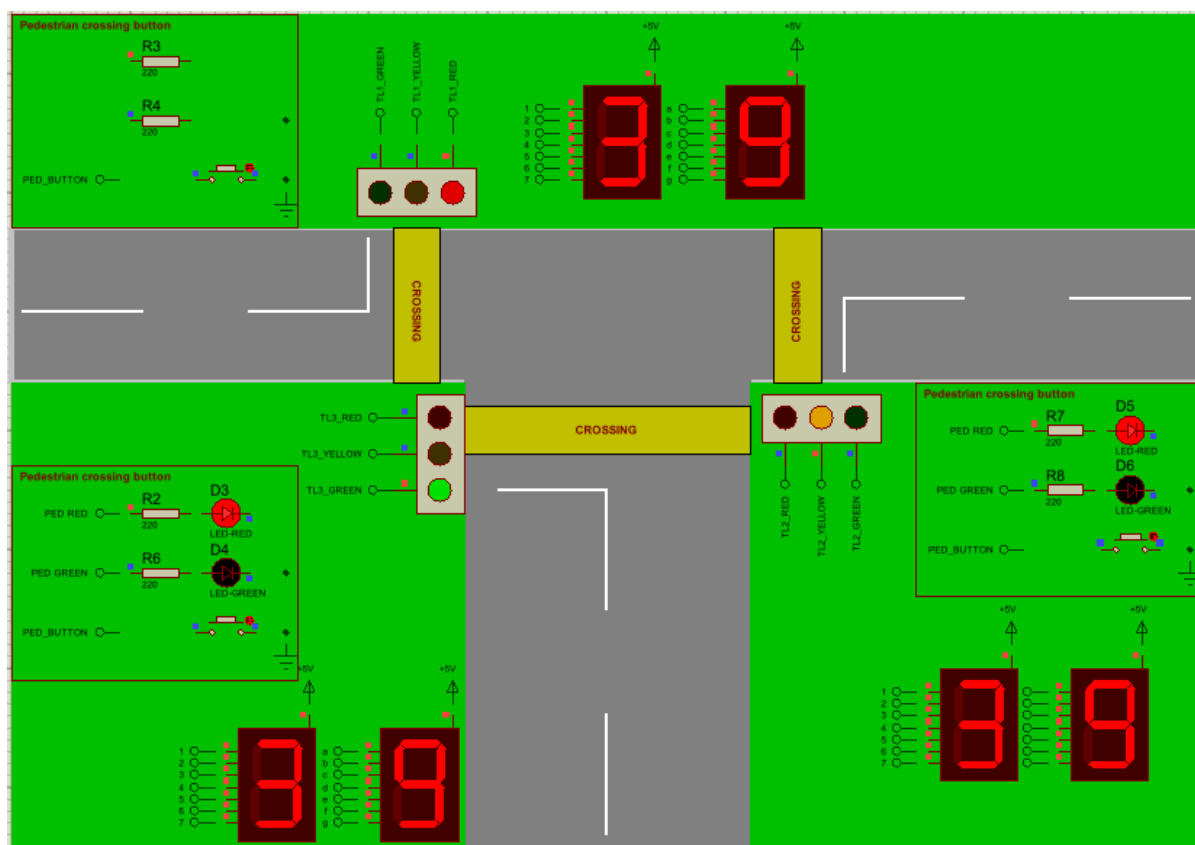


Figure 8 - Phase 1 Starting (Prototype)



6.3 Phase 2

Phase 2 begins when the countdown timer reaches 30 seconds and lasts 10 seconds. Table 4 displays the status of each traffic light set during phase 2, with the matching prototype and simulation outputs shown in Figure 10 and Figure 11.

Table 4 - Traffic Light Statuses during Phase 2

Road/Junction Entry Point	Traffic Light Number	Current Illuminated Light
Leftmost Road	Traffic Lights Set 1 (TL1)	AMBER
Right Road	Traffic Lights Set 2 (TL2)	GREEN
Central Road	Traffic Lights Set 3 (TL3)	RED
Pedestrian Crossings	Pedestrian Crossing	RED

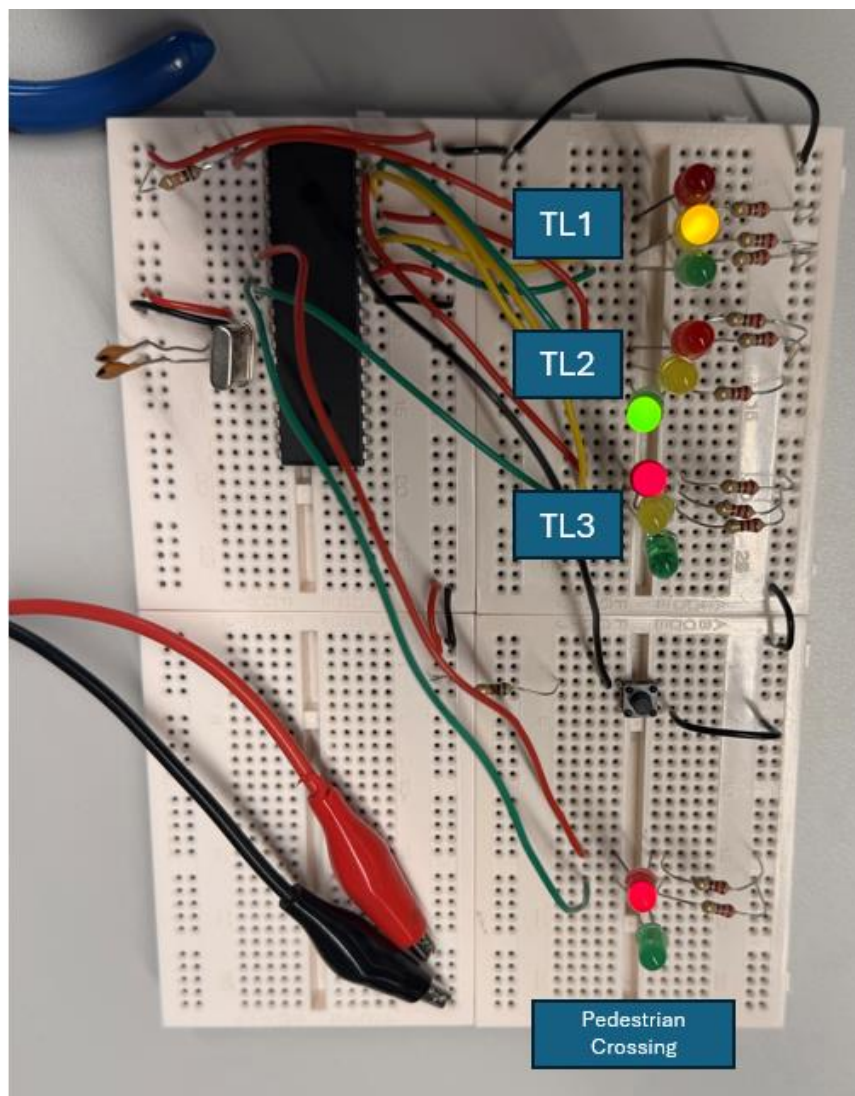


Figure 10 - Phase 2 Starting (Prototype)

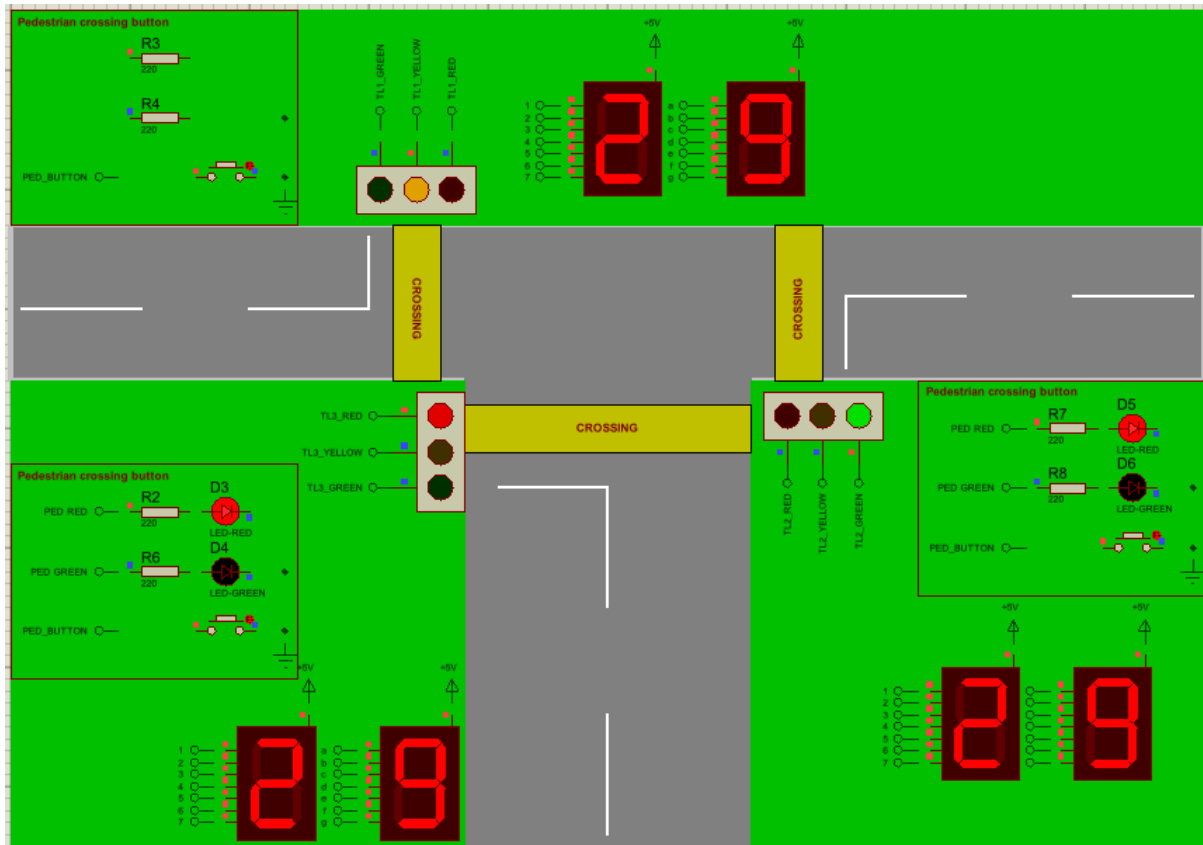


Figure 11 - Phase 2 Starting (Simulation)

6.4 Phase 3

Similarly to the previous phases, phase 3 begins when the countdown timer reaches 20 seconds and lasts 10 seconds. Table 5 displays the status of each traffic light set during phase 3, with the matching prototype and simulation outputs shown in Figure 12 and Figure 13. By the end of Phase 3, each of the three roads leading into the T-junction have been given an opportunity to safely use the T-junction.

Table 5 - Traffic Light Statuses during Phase 3

Road/Junction Entry Point	Traffic Light Number	Current Illuminated Light
Leftmost Road	Traffic Lights Set 1 (TL1)	GREEN
Right Road	Traffic Lights Set 2 (TL2)	RED
Central Road	Traffic Lights Set 3 (TL3)	AMBER
Pedestrian Crossings	Pedestrian Crossing	RED

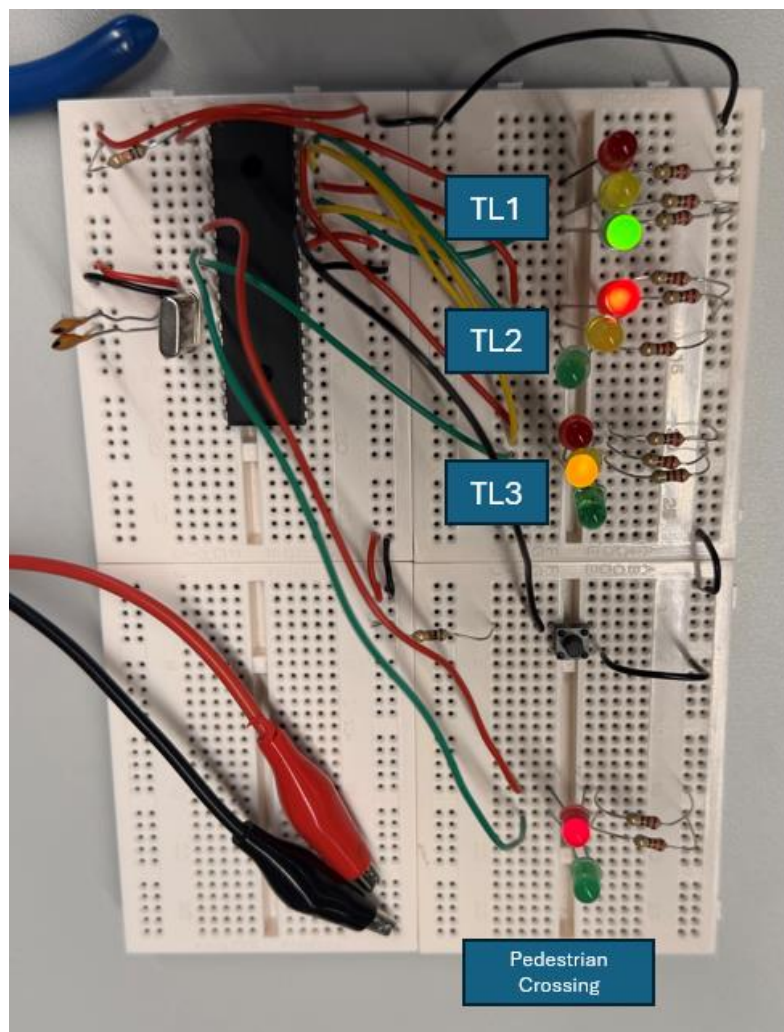


Figure 12 - Phase 3 Starting (Prototype)

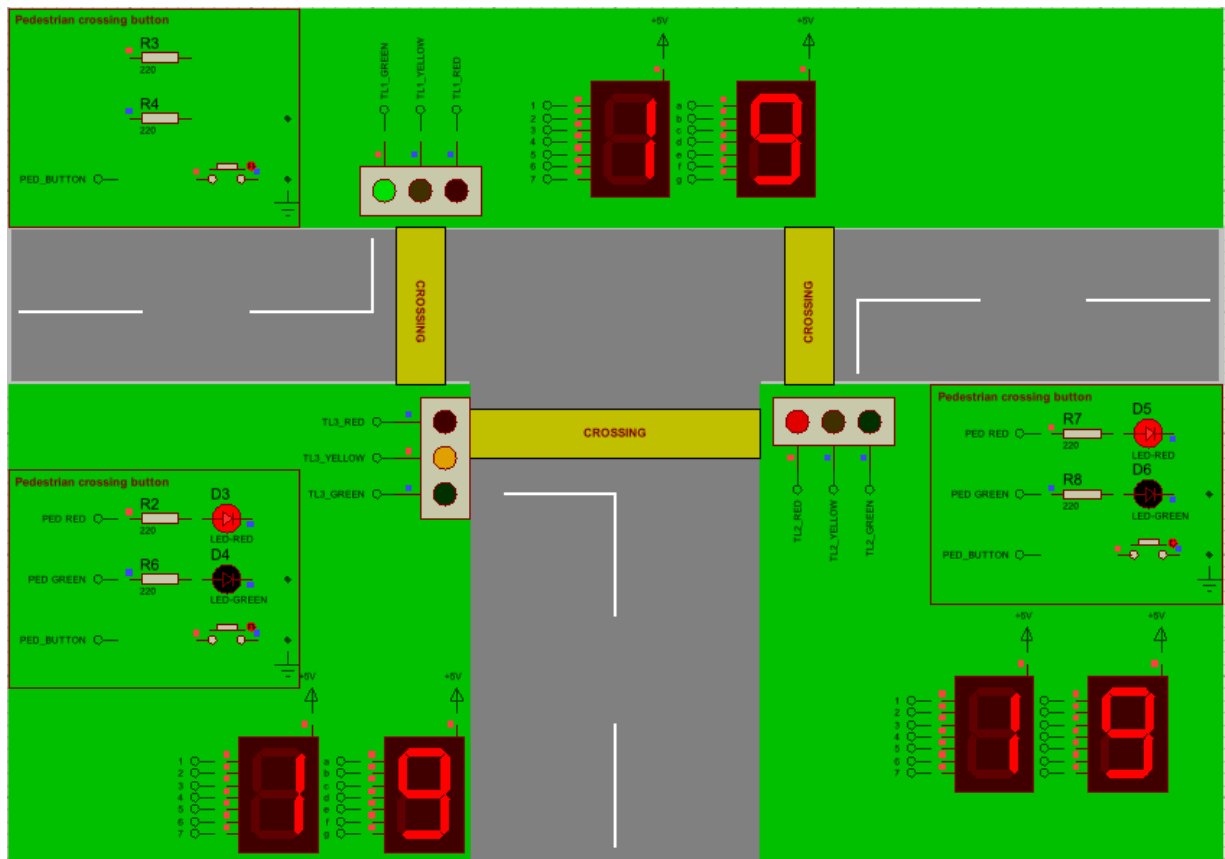


Figure 13 - Phase 3 Starting (Simulation)

6.5 Phase 4

Phase 4 begins after phases 1 to 3 have ended or immediately after any phase has ended if the pedestrian crossing button has been pressed. This phase is considered the dedicated pedestrian crossing phase and begins when the countdown on the 7-segment display enters the final 10 seconds of the 40-second cycle. Table 6 displays the status of each traffic light set during phase 4, with the matching prototype and simulation outputs shown in Figure 14 and Figure 15.

Table 6 - Traffic Lights Statuses during Phase 4

Road/Junction Entry Point	Traffic Light Number	Current Illuminated Light
Leftmost Road	Traffic Lights Set 1 (TL1)	RED
Right Road	Traffic Lights Set 2 (TL2)	RED
Central Road	Traffic Lights Set 3 (TL3)	RED
Pedestrian Crossings	Pedestrian Crossing	GREEN

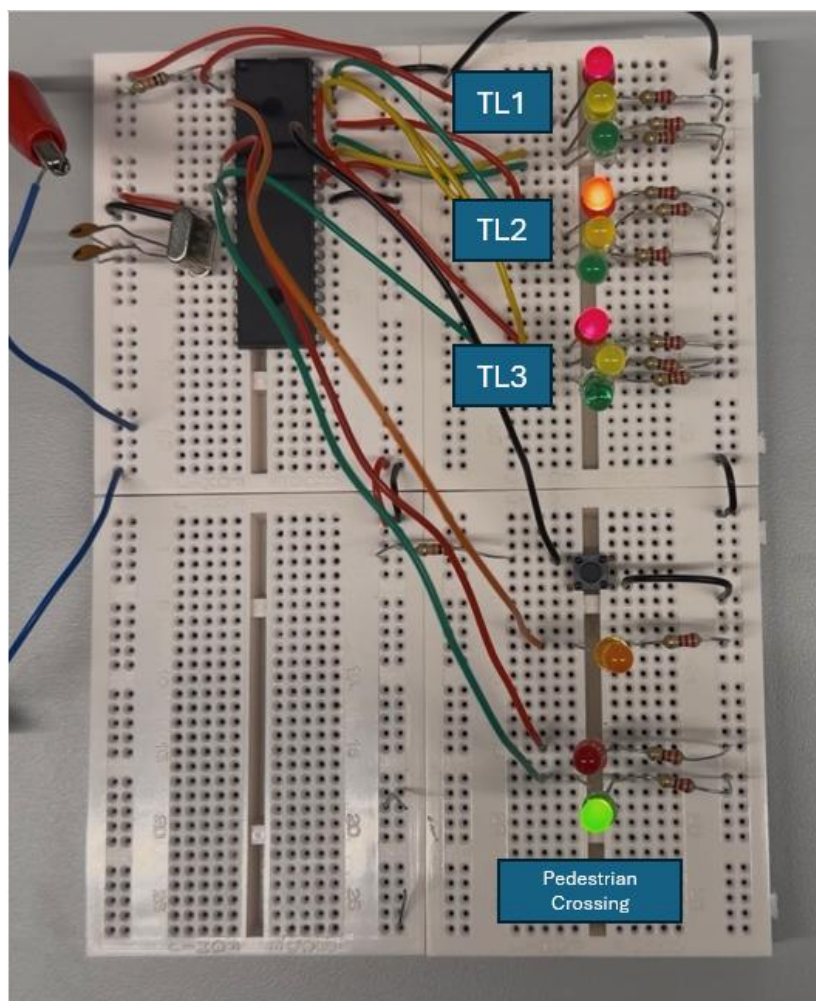


Figure 14 - Phase 4 Starting (Prototype)

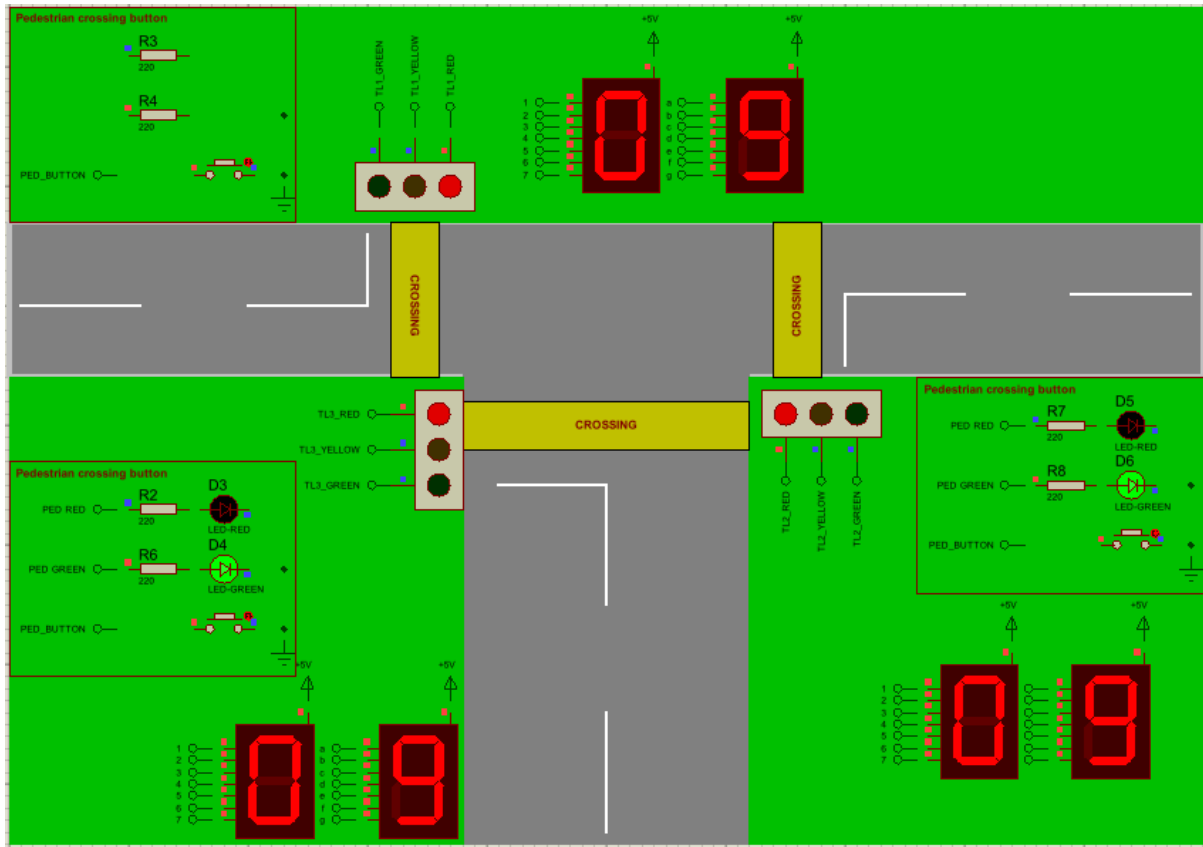


Figure 15 - Phase 4 Starting (Simulation)

6.6 Pedestrian Crossing Acknowledgment

In Figure 16 and Figure 17, it can be seen that the pedestrian button has been pressed as its corresponding LED has been lit. This LED confirms that the button has been pressed, however, the system is waiting for the current phase of traffic control to finish before it can introduce the pedestrian crossing phase (phase 4). This LED will be turned off when the pedestrian crossing phase is introduced.

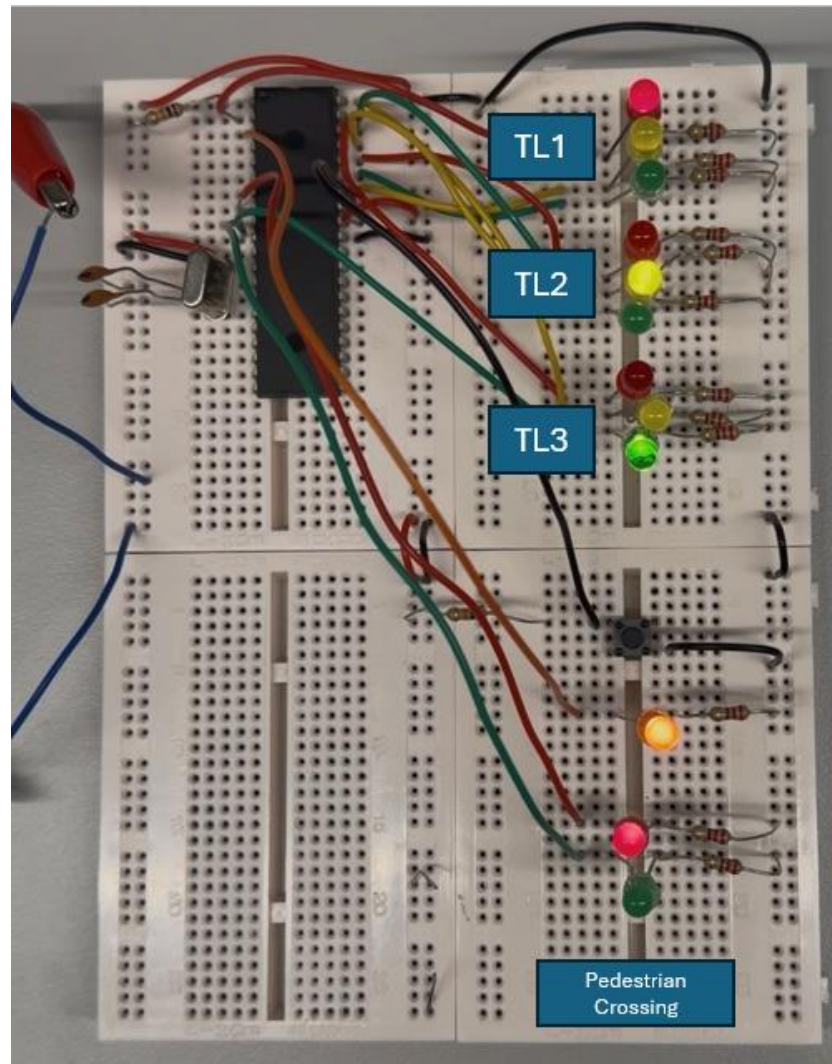


Figure 16 - Pedestrian crossing button press indication (Prototype)

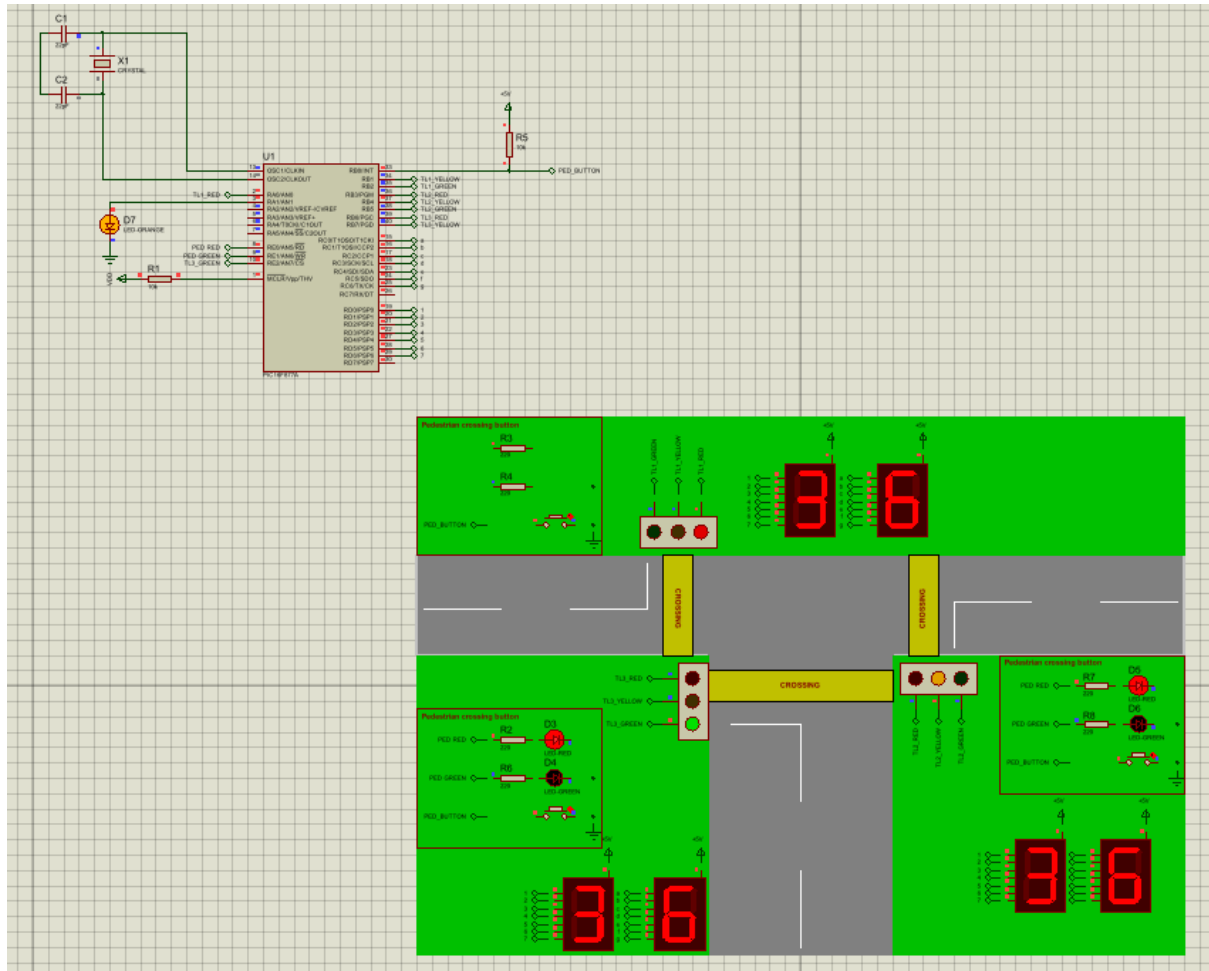


Figure 17 - Pedestrian crossing button press indication (Simulation)

7. Conclusion/Recommendations for future work

The main aim of this project was to design and develop a traffic light control system, that could operate with precise timing intervals to regulate traffic flow effectively. The system was successfully designed around a PIC16F877A microcontroller, and made good use of its features and abilities to give fixed-time traffic light control at a T-junction. The results from the prototype and simulation confirm that the system runs reliably in consistent 40 second cycles, with the four phases lasting 10 seconds each to ensure all road users get an equal opportunity to safely use the T-junction. Additionally, the inclusion of the pedestrian crossing button gives the pedestrians the ability to cross safely as soon as possible.

To improve on the current simple fixed-time system, an actuated or adaptive control method, similar to SCOOT, could be explored to introduce their benefits and improve the efficiency of the traffic control system. The capabilities and functionality of the system could also be improved by introducing the use of Internet of Things (IOT) smart sensors to detect upcoming heavier traffic on a road and then adapt the timings of the traffic light changes. Renewable power sources could also be implemented to make the system more sustainable.

In conclusion, the work completed on this system demonstrates a good balance between cost, reliability, safety and practicality and offers a competent solution to traffic control at T-junctions.

8. Bibliography

- [1] N. Agatz, M. Hewitt and B. W. Thomas, ““Make no little plans”: Impactful research to solve the next generation of transportation problems,” *Networks*, vol. 77, no. 2, pp. 269-286, 2020.
- [2] A. Hamilton, B. Waterson, T. Cherrett, A. Robinson and I. Snell, “The evolution of urban traffic control: changing policy and technology,” *Universities' Transport Study Group UK Annual Conference 2012*, vol. 36, no. 1, pp. 24-43, 2013.
- [3] A. Kalašová, A. Hájník, S. Kubařák, J. Beňuš and V. Harantová, “THE IMPACT OF ACTUATED CONTROL ON THE ENVIRONMENT AND THE TRAFFIC FLOW,” *Journal of Applied Engineering Science*, vol. 20, no. 2, pp. 305-314, 2021.
- [4] K. L. Hong and H. F. Chow, “Adaptive traffic control system: Control strategy, prediction, resolution, and accuracy,” *Journal of Advanced Transportation*, vol. 36, no. 3, pp. 323-347, 2010.
- [5] S. S. S. M. Qadr, M. A. Gökçe and E. Öner, “State-of-art review of traffic signal control methods: challenges and opportunities,” *European Transport Research Review*, no. 12, 2020.
- [6] O. Younis and N. Moayeri, “Employing Cyber-Physical Systems: Dynamic Traffic Light Control at Road Intersections,” *IEEE Internet of Things Journal*, vol. 4, no. 6, pp. 2286 - 2296, 2017.
- [7] P. Mirchandani and L. Head, “A real-time traffic signal control system: architecture, algorithms, and analysis,” *Transportation Research Part C: Emerging Technologies*, vol. 9, no. 6, pp. 415-432, 2001.
- [8] Y. Wang, X. Yang, Y. Liu and H. Liang, “A Review of the Self-Adaptive Traffic Signal Control System Based on Future Traffic Environment,” *Journal of Advanced Transportation*, vol. 2018, no. 1, 2018.
- [9] P. Mannion, J. Duggan and E. Howley, “An Experimental Review of Reinforcement Learning Algorithms for Adaptive Traffic Signal Control,” *Autonomic Road Transport Support Systems*, p. 47–66, 2016.
- [10] A. Stevanovic, I. Dakic and M. Zlatkovic, “Comparison of adaptive traffic control benefits for recurring and non-recurring traffic conditions,” *IET Intelligent Transport Systems*, vol. 11, no. 3, pp. 142-151, 2015.

- [11] A. A. Şerban and M. Frunzete, "Analysis and comparison between urban traffic control systems," in *2022 IEEE 28th International Symposium for Design and Technology in Electronic Packaging (SIITME)*, Bucharest, Romania, 2022.
- [12] R. H. SUDHAN, M. G. KUMAR, A. U. PRAKASH, S. A. ROOPA DEVI and P. SATHIYA, "ARDUINO ATMEGA-328," *INTERNATIONAL JOURNAL OF INNOVATIVE RESEARCH IN ELECTRICAL, ELECTRONICS, INSTRUMENTATION AND CONTROL ENGINEERING*, vol. 3, no. 4, pp. 27-29, 2015.
- [13] VAYUYAAN, "ARDUINO UNO," VAYUYAAN, 2020.
- [14] G. Brown, "Discovering the STM32 microcontroller," 2013.
- [15] R. Components, "Arduino Uno Rev 3," RS Components, [Online]. Available: <https://uk.rs-online.com/web/p/arduino/7154081>. [Accessed 3rd December 2025].
- [16] M. Electronics, "STM32F401xD & STM32F401xE High-Performance MCUs," Mouser Electronics, [Online]. Available: <https://www.mouser.co.uk/c/?marcom=181358829>. [Accessed 3rd December 2025].
- [17] R. Components, "Microchip, 8bit PIC, PIC16F Microcontroller, 20MHz, 14.3 kB, 256 B Flash, 40-Pin PDIP," RS Components, [Online]. Available: https://uk.rs-online.com/web/p/microcontrollers/4671690?cm_mmc=UK-PLA-DS3A-_-google-_-CSS_UK_EN_PMAX_Catch+All-_-_-4671690&matchtype=&&gclid=EA1aIQobChMI5c2Mh5. [Accessed 3rd December 2025].
- [18] Microchip, "PIC16F87XA Data Sheet," [Online]. Available: <https://ww1.microchip.com/downloads/en/devicedoc/39582b.pdf>. [Accessed 28th November 2025].
- [19] Y. N. Soe and K. K. K. Oo, "Traffic Light Security Alarm Control System Using," *International Journal of Scientific and Research Publications*, vol. 9, no. 7, pp. 956-960, 2019.
- [20] IQD, "HC49 Quartz Crystal Specification," October 2015. [Online]. Available: <https://www.farnell.com/datasheets/2048105.pdf>. [Accessed 2nd December 2025].
- [21] F. Asadi, "Display Information on Seven Segments," in *Digital Circuits Laboratory Manual*, Springer, Cham, 2023, pp. 125-135.

9. Appendix

9.1 Appendix A

Breadboard Circuit Video Demonstration link:

[CWPrototype.MOV](#)

9.2 Appendix B

Full C Program from Proteus

The code begins with assigning user-friendly names to the microcontroller pins based on what will be connected to each pin (Figure 18). The code uses the directive '#define' to make the code easier to read and maintain, as instead of remembering which pin number corresponds to which LED or button, descriptive names can be used.

```
11 //Define LED pins
12 #define TL1_RED PIN_A0           //Traffic Light 1 - Red assigned to PIN A0
13 #define TL1_YELLOW PIN_B1       //Traffic Light 1 - Yellow assigned to PIN B1
14 #define TL1_GREEN PIN_B2        //Traffic Light 1 - Green assigned to PIN B2
15
16 #define TL2_RED PIN_B3           //Traffic Light 2 - Red assigned to PIN B3
17 #define TL2_YELLOW PIN_B4       //Traffic Light 2 - Yellow assigned to PIN B4
18 #define TL2_GREEN PIN_B5        //Traffic Light 2 - Green assigned to PIN B5
19
20 #define TL3_RED PIN_B6           //Traffic Light 3 - Red assigned to PIN B6
21 #define TL3_YELLOW PIN_B7       //Traffic Light 3 - Yellow assigned to PIN B7
22 #define TL3_GREEN PIN_E2        //Traffic Light 3 - Green assigned to PIN E2
23
24 #define PED_RED PIN_E0           //Pedestrian red light assigned to Pin E0
25 #define PED_GREEN PIN_E1        //Pedestrian green light assigned to Pin E1
26
27 #define PED_BUTTON PIN_B0       //Pedestrian button assigned to Pin B0
28
```

Figure 18 - Mapping microcontroller pins

Figure 19 shows there are four functions (phase 1, phase 2, phase 3, phase 4 and the hardware setup) which are declared, and they are defined later in the program.

```
29 void phase1(void);               //TL1 = Red, TL2 = Yellow, TL3 = Green
30 void phase2(void);               //TL1 = Yellow, TL2 = Green, TL3 = Red
31 void phase3(void);               //TL1 = Green, TL2 = Red, TL3 = Yellow
32 void phase4(void);               // All TLs red + pedestrian green
33 void setup_hw(void);
```

Figure 19 - Declaring phase functions and hardware setup function.

In Figure 20, an array containing hex values is shown and they correspond to bit patterns used to display the number '0' to '9' on the 7-segment displays. These patterns control which segments of the displays light up to form each number, and they are used in the countdown logic to show the time remaining of the current phase and cycle. Figure 20 also shows the pedestrian button flag that tracks whether the pedestrian crossing button has been pressed. This flag is set to '1' by the external interrupt when the button is pressed and is checked constantly in the main loop to decide whether the pedestrian crossing phase (phase 4) needs to be run.

```

35 //7-segment display patterns for numbers 0-9
36 unsigned int display[10] = {0x90, 0x80, 0xF8, 0x82, 0x92, 0x99, 0xB0, 0xA4, 0xF9, 0xC0};
37 int ped_button_pressed = 0;           //1 = button pressed, 0 = normal cycle

```

Figure 20 - Bit pattern array for 7-segment 'countdown'

Figure 21 shows the interrupt service routine function used for the pedestrian crossing button press. This function is used to set the indicating light on and the flag variable to 1, when the pedestrian crossing button has been pressed. When the current phase finishes, the system will recognise the flag has been set and will jump to the pedestrian crossing phase. The hardware function configures the microcontroller's I/O ports.

```

39 #INT_EXT
40 void EXT_isr(void)
41 {
42     output_high(PIN_A1);
43     ped_button_pressed = 1;
44 }
45
46 void setup_hw(void)
47 {
48     set_tris_a(0x00);
49     output_b(0x00);           //initialise portB to 0
50     set_tris_b(0x01);       //set portB as output for LEDs
51     set_tris_c(0x00);       //set portC as output for 1st 7-segment display (ones)
52     output_c(0x00);         //initialise portC to 0
53     set_tris_d(0x00);       //set portD as output for 2nd 7-segment display (tens)
54     output_d(0x00);         //initialise portD to 0
55     set_tris_e(0x00);       //set portE as output for Pedestrian Lights
56     output_e(0x00);         //initialise portE to 0
57 }

```

Figure 21 - Interrupt routine and hardware setup function

The programming of each phase's operation can be seen in Figure 22 through to Figure 25. Each phase updates the 7-segment displays by using 'output_d(0x**)' to show a static digit (0 to 4) on one display and then using 'output_c(display[i])' to show a countdown from 9 to 0 on the other display. Each traffic light in the different phases is coded according to Table 1, and this is done by using 'output_low(PIN_XX)' or 'output_high(PIN_XX)' on each of the nine traffic lights throughout each phase. Each phase ensures only one road can use the T-junction at a time, with clear transitions between each phase.

```

59 void phase1(void)
60 {
61     int i;
62     for (i = 0; i < 10; i++)
63     {
64         output_d(0xB0);           //7-segment display pattern for '3'
65         output_c(display[i]);      //Update 7-segment counter
66         output_high(TL1_RED);      //Traffic Light 1 Red LED ON
67         output_low(TL1_YELLOW);    //Traffic Light 1 Yellow LED OFF
68         output_low(TL1_GREEN);     //Traffic Light 1 Green LED OFF
69         output_low(TL2_RED);       //Traffic Light 2 Red LED OFF
70         output_high(TL2_YELLOW);   //Traffic Light 2 Yellow LED ON
71         output_low(TL2_GREEN);     //Traffic Light 2 Green LED OFF
72         output_low(TL3_RED);       //Traffic Light 3 Red LED OFF
73         output_low(TL3_YELLOW);    //Traffic Light 3 Yellow LED OFF
74         output_high(TL3_GREEN);    //Traffic Light 3 Green LED ON
75         output_high(PED_RED);      //Pedestrian red LED ON
76         output_low(PED_GREEN);     //Pedestrian green LED OFF
77         delay_ms(1000);            //delay 1s
78     }
79 }

```

Figure 22 - Phase 1 code

```

81 void phase2(void)
82 {
83     int i;
84     for (i = 0; i < 10; i++)
85     {
86         output_d(0xA4);           //7-segment display pattern for '2'
87         output_c(display[i]);     //Update 7-segment counter
88         output_low(TL1_RED);      //Traffic Light 1 Red LED OFF
89         output_high(TL1_YELLOW);  //Traffic Light 1 Yellow LED ON
90         output_low(TL1_GREEN);    //Traffic Light 1 Green LED OFF
91         output_low(TL2_RED);      //Traffic Light 2 Red LED OFF
92         output_low(TL2_YELLOW);   //Traffic Light 2 Yellow LED OFF
93         output_high(TL2_GREEN);   //Traffic Light 2 Green LED ON
94         output_high(TL3_RED);     //Traffic Light 3 Red LED ON
95         output_low(TL3_YELLOW);   //Traffic Light 3 Yellow LED OFF
96         output_low(TL3_GREEN);    //Traffic Light 3 Green LED OFF
97         output_high(PED_RED);     //Pedestrian red LED ON
98         output_low(PED_GREEN);    //Pedestrian green LED OFF
99         delay_ms(1000);          //delay 1s
100     }
101 }

```

Figure 23 - Phase 2 code

```

103 void phase3(void)
104 {
105     int i;
106     for (i = 0; i < 10; i++)
107     {
108         output_d(0xF9);           //7-segment display pattern for '1'
109         output_c(display[i]);     //Update 7-segment counter
110         output_low(TL1_RED);      //Traffic Light 1 Red LED OFF
111         output_low(TL1_YELLOW);   //Traffic Light 1 Yellow LED OFF
112         output_high(TL1_GREEN);   //Traffic Light 1 Green LED ON
113         output_high(TL2_RED);     //Traffic Light 2 Red LED ON
114         output_low(TL2_YELLOW);   //Traffic Light 2 Yellow LED OFF
115         output_low(TL2_GREEN);    //Traffic Light 2 Green LED OFF
116         output_low(TL3_RED);     //Traffic Light 3 Red LED OFF
117         output_high(TL3_YELLOW);  //Traffic Light 3 Yellow LED ON
118         output_low(TL3_GREEN);    //Traffic Light 3 Green LED OFF
119         output_high(PED_RED);     //Pedestrian red LED ON
120         output_low(PED_GREEN);    //Pedestrian green LED OFF
121         delay_ms(1000);          //delay 1s
122     }
123 }

```

Figure 24 - Phase 3 code

```

125 void phase4(void)
126 {
127     output_low(PIN_A1);
128     int i;
129     for (i = 0; i < 10; i++)
130     {
131         output_d(0xC0);           //7-segment display pattern for '0'
132         output_c(display[i]);     //Update 7-segment counter
133         output_high(TL1_RED);     //Traffic Light 1 Red LED ON
134         output_low(TL1_YELLOW);   //Traffic Light 1 Yellow LED OFF
135         output_low(TL1_GREEN);    //Traffic Light 1 Green LED OFF
136         output_high(TL2_RED);     //Traffic Light 2 Red LED ON
137         output_low(TL2_YELLOW);   //Traffic Light 2 Yellow LED OFF
138         output_low(TL2_GREEN);    //Traffic Light 2 Green LED OFF
139         output_high(TL3_RED);     //Traffic Light 3 Red LED ON
140         output_low(TL3_YELLOW);   //Traffic Light 3 Yellow LED OFF
141         output_low(TL3_GREEN);    //Traffic Light 3 Green LED OFF
142         output_low(PED_RED);      //Pedestrian red LED OFF
143         output_high(PED_GREEN);   //Pedestrian green LED ON
144         delay_ms(1000);           //delay 1s
145     }
146 }

```

Figure 25 - Phase 4 code

The 'output_low(PIN_A1)' in phase 4 (seen in Figure 25) has been included to reset the pedestrian button indicator.

Figure 26 shows the code under the main function of the system, which is the central control loop for the traffic light control. The program under this function starts by setting up the external interrupt to detect a pedestrian button press and ensures the system is ready to respond to interrupts. It then configures all I/O ports on the microcontroller by calling 'setup_hw()', as seen earlier.

Phase 4 resets the pedestrian indicator using 'output_low(PIN_A1)' and is triggered when 'ped_button_pressed' is equal to 1. The 'ped_button_pressed' flag is set by the interrupt service routine seen in Figure 21. The phases are run inside a 'while(TRUE)' loop (Figure 27) to ensure the traffic control never ends. After each phase, an if statement is used to check if the pedestrian button has been pressed. If it has, the pedestrian crossing phase (phase 4) runs as soon as the current phase has finished. At the end of the loop, phase 4 always runs to guarantee pedestrians safe use of the T-junction, then displays "40" on the 7-segment displays to restart the cycle and countdown.

```

148 void main()
149 {
150     ext_int_edge(H_T0_L);           //Configure external interrupt to trigger on a high-to-low transition (falling edge)
151     clear_interrupt(INT_EXT);       //Clear any pending external interrupt flag before enabling
152     enable_interrupts(INT_EXT);     //Enable external interrupt (RB0 pin for pedestrian button)
153     enable_interrupts(GLOBAL);     //Enable global interrupt handling
154     setup_hw();                    //Call hardware setup routine to configure ports and initialize outputs

```

Figure 26 - 'Main' function initialisation code

```

156     while(TRUE)
157     {
158         phase1();                  //runs phase 1
159         if(ped_button_pressed ==1) //checks if pedestrian crossing button has been pressed
160         {
161             phase4();              //if it has, run the pedestrian crossing phase after current phase has finished
162             ped_button_pressed = 0; //reset pedestrian button pressed flag
163         }
164
165         phase2();                  //runs phase 2
166         if(ped_button_pressed ==1) //checks if pedestrian crossing button has been pressed
167         {
168             phase4();              //if it has, run the pedestrian crossing phase after current phase has finished
169             ped_button_pressed = 0; //reset pedestrian button pressed flag
170         }
171
172         phase3();                  //run phase 3
173         if(ped_button_pressed==1) //checks if pedestrian crossing button has been pressed
174         {
175             phase4();              //if it has, run the pedestrian crossing phase after current phase has finished
176             ped_button_pressed = 0; //reset pedestrian button pressed flag
177         }
178
179         phase4();                  //runs phase 4 if all other phases have run without a pedestrian button press.
180
181
182         output_d(0x99);            //7-segments display '40'
183         output_c(display[9]);      //Update 7-segment counter
184         delay_ms(1000);            //delay 1s
185     }

```

Figure 27 - Main function 'while' loop code

9.3 Appendix C

Full flowchart can be found in the supporting documents folder.

Link: [Electronics 1 CW Flowchart.vsd](#)