

ACDB5002 - 2315904

by Max Ladner

Submission date: 05-Dec-2025 09:15AM (UTC+0000)

Submission ID: 269695578

File name: 50083_Max_Ladner_ACDB5002_-_2315904_634321_902976718.odt (1.78M)

Word count: 3192

Character count: 18433

ACDB5002 Electronics 1

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

Max Ladner

2315904

05-12-25

Abstract

The following project involves the design, coding, and construction of a prototype for a timed traffic light system powered by a microcontroller. The system aims to replicate a real-world traffic light system scenario that manages vehicle and pedestrian signals with accurate time sequences. The development process included writing a pseudocode and producing a flow chart, to define system logic and sequencing. Following this a circuit design could be produced and correctly coded ready for simulation and physical assembly on a breadboard. The prototype was successful in demonstrating correct traffic light and pedestrian signal sequences; however, the seven-segment countdown display was excluded from the final prototype as challenges occurred in implementing in the final circuit. The project highlights the practical application of implementing embedded systems in traffic light management and provides a basis for further improvements, such as full integration of displays.

Contents

Abstract	2
Contents	3
Table of Figures	3
1.Introduction	4
1.1 Aims and Objectives	4
2.Background research	5
3.Design and Implementation	6
3.1 Pseudocode	6
3.2 Flow Chart	7
3.3 Circuit design	7
3.4 Component overview	8
3.5 Complete Circuit Workings.....	8
3.6 Source code	8
4.7 Prototyping.....	9
4. Results.....	11
5. Recommendations	12
6. Conclusion.....	12
References.....	13
APPENDIX A: FLOW CHART.....	15
APPENDIX B: SOURCE CODE.....	16

Table of Figures

Figure 1; Pseudocode	6
Figure 2; Circuit design	7
Figure 3; Breadboard circuit prototype	9
Figure 4; PickIt3 connection	10
Figure 5; Circuit in simulation	10
Figure 6; Breadboard prototype simulation	11

1. Introduction

The purpose of this project is to design and implement a timed traffic light system that simulates a real-world traffic scenario. Traffic light systems play a critical role in managing vehicle flow and ensuring pedestrian road safety. By implementing a microcontroller-based solution, this project demonstrates how embedded systems can be applied to automate traffic management processes effectively.

This project follows a structured development approach, beginning with the creation of pseudocode and flowcharts to define the system logic, followed by the design of the circuit in proteus9, and the development of a source code in CCS to control the sequencing of the traffic and pedestrian lights. After successful simulation testing, the circuit could then be prototyped on breadboard. The system is designed to operate with precise timing intervals, including a countdown displayed on seven-segment displays; to monitor progress in the sequence, this also helps to enhance usability and realism.

1.1 Aims and Objectives

The aim of the project is to successfully design and develop a working microcontroller-powered circuit to provide traffic management for a three-direction junction, where traffic movement is determined by a ten-second rotational time sequence, including a final ten-second sequence to allow for pedestrian traffic.

The objectives of this project are to:

- Develop a functional traffic light system with accurate timing sequences.
- Integrate pedestrian signals with traffic lights for improved safety.
- Prototype the design using a microcontroller and breadboard assembly.
- Show the circuit working through simulation and physical testing.

The project gives a practical demonstration of embedded system design principles and highlights the challenges and considerations involved in implementing real-time control systems.

2. Background research

Traffic lights are installed at road intersections to control the movement of vehicles from different directions. Their operation can vary, particularly when handling turns, which may involve dedicated signals or specific rules. These lights generally feature three indicators; red, amber, and green, arranged vertically. Each light uses a coloured lens, with red positioned at the top, amber in the middle, and green at the bottom. In use, the green light signals vehicles to proceed, the amber light warns that the signal will soon change, and the red light directs traffic to stop [1].

Microcontrollers play a critical role in traffic management controlled by embedded systems due to their ability to execute real-time control operations. They enable precise timing sequences, integration of sensors, and control of peripheral devices such as LEDs and displays. Traditional traffic light systems often work on fixed-time rotation cycles that have been configured based upon traffic history data or assumptions made based on this, systems such as these work well in predictable or low-traffic areas. [2]

The PIC16F877A microcontroller is widely used in embedded systems due to its versatility, affordability, and ease of programming, and is particularly suitable for traffic signal control applications because of its robust feature set and ability to handle real-time operations efficiently. The PIC offers a range of functionalities that make it ideal for managing timing sequences and peripheral devices in traffic light systems [3].

The PIC16F877A includes 33 input/output pins on multiple I/O ports, as well as integrated timers, which are essential for controlling multiple traffic lights and pedestrian signals simultaneously. Its EEPROM memory allows storage of configuration data, while the USART module supports serial communication for potential integration with remote monitoring systems. The device operates at clock speeds up to 20 MHz, allowing for accurate timing for the light phase changes and countdown displays [4].

One of the main benefits to using the PIC16F877A is its cost-effectiveness, being inexpensive compared to more advanced microcontrollers while still providing sufficient processing power for traffic signal applications. It is compatible with development tools such as MPLAB IDE and CCS compiler, which helps to simplify programming and debugging. Additionally, the microcontroller's low power consumption and compact size make it suitable for installations where energy efficiency is required and space is constricted [3], [4].

In a typical traffic light system application, the PIC16F877A can manage multiple outputs for red, amber, and green LEDs for multiple directions, as well as the pedestrian signals. By using its internal timers, the microcontroller can execute precise countdown intervals which can be displayed on seven-segment displays connected to I/O ports. Conditional logic implemented in the source code allows the system to cycle through predefined sequences, ensuring safe and reliable traffic flow. Also, the microcontroller can be programmed to incorporate adaptive features, such as sensor-based input for dynamic timing adjustments [1], [2].

3. Design and Implementation

To achieve the project outcome a structured process for designing and developing a working prototype was followed. The first part of the process was to create a pseudocode and flowchart to define how the source code would function and flow. A circuit design process was then undertaken using Proteus9 software, components and values were identified and connected in an appropriate layout that could effectively function. With a final source code developed, the circuit could then be simulated to test the code function and to identify any problems and their solutions could be implemented, once the code was satisfactory to the brief, a prototype circuit could be developed using physical components and a breadboard.

3.1 Pseudocode

In the design process, creating a pseudocode provides a structured and logical representation of the system's intended functionality before the circuit diagram is developed. This approach helps identify requirements and potential issues early, ensuring the design aligns with the project objectives. By outlining the sequence of actions in a simplified format, the pseudocode serves as a blueprint for translating the brief into a working concept; which improves clarity, reduces errors, and ultimately leads to a more efficient and accurate circuit design [5].

```
1 START
2 WHILE (TRUE)
3   FOR count = 40 DOWNTO 0
4     tens = count/10
5     ones = count%10
6     DISPLAY tens on 7_seg_1
7     DISPLAY ones on 7_seg_2
8
9     IF count>32
10      Traffic Light 1: GREEN
11      Traffic Light 2: RED
12      Traffic Light 3: RED
13      Pedestrian: RED
14    ELSE IF count > 30
15      Traffic Light 1: YELLOW
16      Traffic Light 2: YELLOW
17      Traffic Light 3: RED
18      Pedestrian: RED
19    ELSE IF count > 21
20      Traffic Light 1: RED
21      Traffic Light 2: GREEN
22      Traffic Light 3: RED
23      Pedestrian: RED
24    ELSE IF count > 19
25      Traffic Light 1: RED
26      Traffic Light 2: YELLOW
27      Traffic Light 3: YELLOW
28      Pedestrian: RED
29    ELSE IF count > 12
30      Traffic Light 1: RED
31      Traffic Light 2: RED
32      Traffic Light 3: GREEN
33      Pedestrian: RED
34    ELSE IF count > 10
35      Traffic Light 1: RED
36      Traffic Light 2: RED
37      Traffic Light 3: YELLOW
38      Pedestrian: RED
39    ELSE IF count > 2
40      Traffic Light 1: RED
41      Traffic Light 2: RED
42      Traffic Light 3: RED
43      Pedestrian: GREEN
44    ELSE
45      Traffic Light 1: YELLOW
46      Traffic Light 2: RED
47      Traffic Light 3: RED
48      Pedestrian: GREEN
49    END IF
50  END FOR
51 END WHILE
```

Figure 1; Pseudocode

The pseudocode shown in figure 1 was created to identify the flow of code required to satisfy the brief. As shown, the main body of code consists of a simple two segment counter followed by an else if statement ladder, which states are dependent on the count. The ELSE IF ladder works by first using an IF statement to check an initial condition; if this condition is true the associated action is executed. If the condition is not met in the IF statement, the compiler will skip to the next condition, defined by an ELSE IF clause, if this condition is met, the associated action is then executed. Multiple ELSE IF clauses can be used in a ladder, and the ladder is completed by an ELSE statement [6]. In this brief, the ELSE IF ladder is defined by different time stamps of the count, changing the executed logic at specified times.

In the early stage of design, developing a flow chart offers a clear visual of the system's processes and decision points before creating the circuit diagram. By mapping inputs, outputs, decisions, and operational sequences in standardised symbols, a flow chart enables designers to detect logical gaps and inefficiencies upfront. This visual representation provides better alignment with project objectives, enhances communication among team members, and minimises misinterpretation. By providing both a conceptual overview and detailed workflow in one diagram, flow charts streamline the translation from system brief to circuit implementation, reducing design errors and accelerating development time [7]. See appendix A for a detailed flow chart designed to satisfy the brief of this project.

The next step in the design process is to create the final circuit diagram, which represents the physical implementation of the system. This step translates the logical structure into actual components such as resistors, capacitors, integrated circuits, and connections, ensuring that the design meets electrical and functional requirements [8].



7

3.4 Component overview

3.4.1 Microcontroller

The PIC16F877A is implemented to execute the programmed logic that controls the traffic lights and countdown. It receives clock signals from the crystal oscillator and uses internal timers to manage state changes. The microcontroller outputs signals to the LEDs and seven-segment displays, ensuring synchronisation between the countdown and the light transitions [9]. This

3.4.2 Crystal Oscillator and Capacitors

The crystal oscillator, along with capacitors C2 and C3, provides a stable clock source for the microcontroller. This ensures precise timing for the countdown and light changes; this is critical in traffic systems to maintain predictable intervals and avoid confusion for road users [10].

3.4.3 Resistor Networks

The resistor networks connected to the seven-segment displays limit current to each segment, preventing damage and ensuring consistent brightness. These networks simplify the design by grouping resistors into a single resistor bank [11].

3.4.4 Seven-Segment Displays

The two seven-segment displays show the countdown from 40 seconds, giving drivers and pedestrians clear feedback on the remaining time before the lights change. This would improve safety and traffic flow by reducing sudden stops and hesitation [12].

3.4.5 LEDs and Resistors

The LEDs represent the traffic light signals (red, amber, green) for three directions, and additional LEDs (D1 and D2) indicate pedestrian states (STOP and CROSS). Each LED is paired with a resistor to limit current, ensuring longevity and preventing overheating. The microcontroller activates these LEDs based on the programmed timing sequence [11].

3.5 Complete Circuit Workings

The crystal oscillator sets the timing base, which the microcontroller uses to count down from 40 seconds. During the countdown, the seven-segment displays update in real time, and the microcontroller switches the traffic lights and pedestrian indicators at the predefined intervals, as per the brief. This integrated approach ensures synchronisation between visual signals and numeric countdown, improving clarity and reducing errors in traffic control.

3.6 Source code

To satisfy the brief and be able to simulate the circuit, a source code had to be prepared in C language coding in the CCS compiler, a copy of this code can be seen in appendix B.

The program begins by configuring the microcontroller ports as outputs to control the LEDs representing traffic and pedestrian lights, as well as the seven-segment displays. A clock speed of 8 MHz is set to ensure accurate timing operations. The seven-segment display patterns for digits 0–9 are stored in an array, allowing the countdown to be displayed in real time.

A central countdown timer runs from 40 seconds to zero, representing the full cycle of the traffic light sequence. For each second, the program calculates the tens and one's digits and updates the corresponding seven-segment displays. This provides a clear indication of the remaining time for each phase.

The traffic light control is implemented through a series of conditional statements that activate specific lights based on the current countdown value. The sequence is divided into distinct phases:

- Phase 1 (40–32 seconds): Traffic Light 1 is green, while Lights 2 and 3 remain red. The pedestrian signal shows red.
- Phase 2 (32–30 seconds): Lights 1 and 2 switch to amber, preparing for a change in traffic flow.
- Phase 3 (30–21 seconds): Traffic Light 2 turns green, Light 1 turns red, and Light 3 remains red.
- Phase 4 (21–19 seconds): Lights 2 and 3 display amber, indicating an upcoming transition.
- Phase 5 (19–12 seconds): Traffic Light 3 turns green, while Lights 1 and 2 remain red.
- Phase 6 (12–10 seconds): Light 3 switches to amber, preparing for pedestrian priority.
- Phase 7 (10–2 seconds): All traffic lights show red, and the pedestrian signal turns green, allowing safe crossing.
- Final Phase (last 2 seconds): Light 1 switches to amber, resetting the cycle for the next sequence.

As discussed in the above section, *pseudocode*, this is achieved using an ELSE IF ladder sequence [6].

4.7 Prototyping

To verify the circuit design and source code were compatible with each other, a prototype stage was undertaken using a simple breadboard construction. Breadboard constructions are commonly used in prototyping as it allows circuits to be constructed and tested quickly without soldering, which allows for changes and alterations to be made easily, breadboarding also allows for configuration changes and amendments to be made before a permanent PCB layout is decided on [13].

The initial breadboard prototype can be seen in figure 3, The Microcontroller was encoded using a Pikkit3 encoder, shown in figure 4, this is achieved by connecting the microcontroller to the encoder, which is connected by USB to a computer, and using the MPLAB®X IDE software to download the HEX file for the code and encrypting to the PIC. The microcontroller is then able to be implemented in the circuit. [14]

The rest of the circuit was wired in according to the circuit design and the pin-out reference for the PIC16F877A chip from the datasheet [15]. Once correctly connected to the circuit testing could be undertaken to evaluate the projects results.

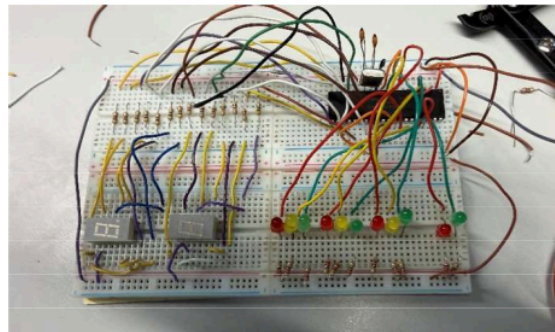


Figure 3; Breadboard circuit prototype

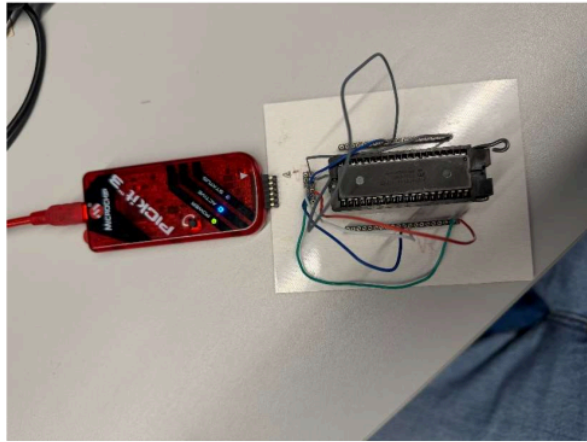


Figure 4; PickIt3 connection.

4. Results

Figure 5 illustrates the working simulation of the circuit. Each traffic light illuminates in the correct sequence, and the pedestrian light operates appropriately. The seven-segment displays are also visible, counting down in one-second intervals from forty, as required.

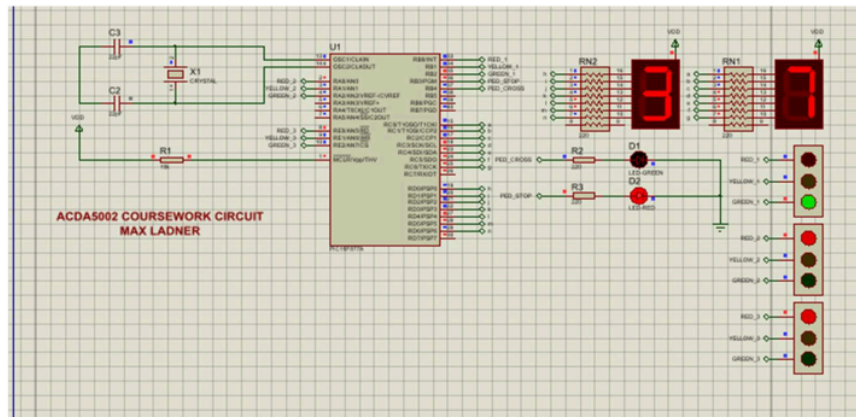


Figure 5; Circuit in simulation

Furthering these results, Figure 6 shows the breadboard prototype of the circuit. Like the simulated version, the prototype successfully follows the timed sequence, displaying the appropriate traffic and pedestrian light outputs at each stage. However, the seven-segment display was not included in the physical prototype due to integration issues during testing. To ensure a functional demonstration within the allotted time, the decision was made to omit the displays temporarily and focus on completing a simplified, working prototype.

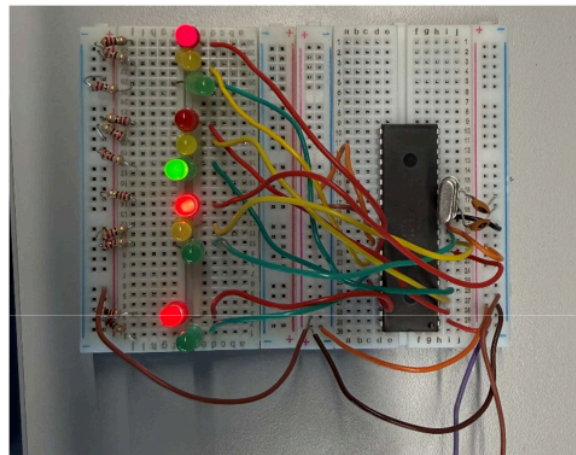


Figure 6; Breadboard prototype simulation

The design and implementation of the timed traffic light system was successfully completed through a structured development process. A detailed pseudocode and flowchart were produced to outline the system logic, followed by the circuit design and source code development for the microcontroller. The final circuit was assembled on a breadboard and tested, demonstrating correct timing sequences and reliable operation of the traffic light phases as intended.

5. Recommendations

Based on the design and testing outcomes, several key improvements have been identified to enhance the circuit's functionality and overall project performance. One significant enhancement would be the integration of a pedestrian button into the circuit. This feature would allow the operational sequence to pause for a predetermined interval, enabling pedestrians to cross safely at any point before the system resumes its normal cycle. Additionally, during the physical prototyping stage, developing and implementing a more refined wiring diagram would be highly beneficial. A clear and organised diagram would not only simplify the assembly process but also ensure a cleaner, more repeatable design that can be easily replicated or scaled. Lastly, additional time and attention should be dedicated to ensuring the correct operation of the seven-segment displays during prototyping. This step is crucial for validating the design and guaranteeing reliable functionality in the final implementation.

6. Conclusion

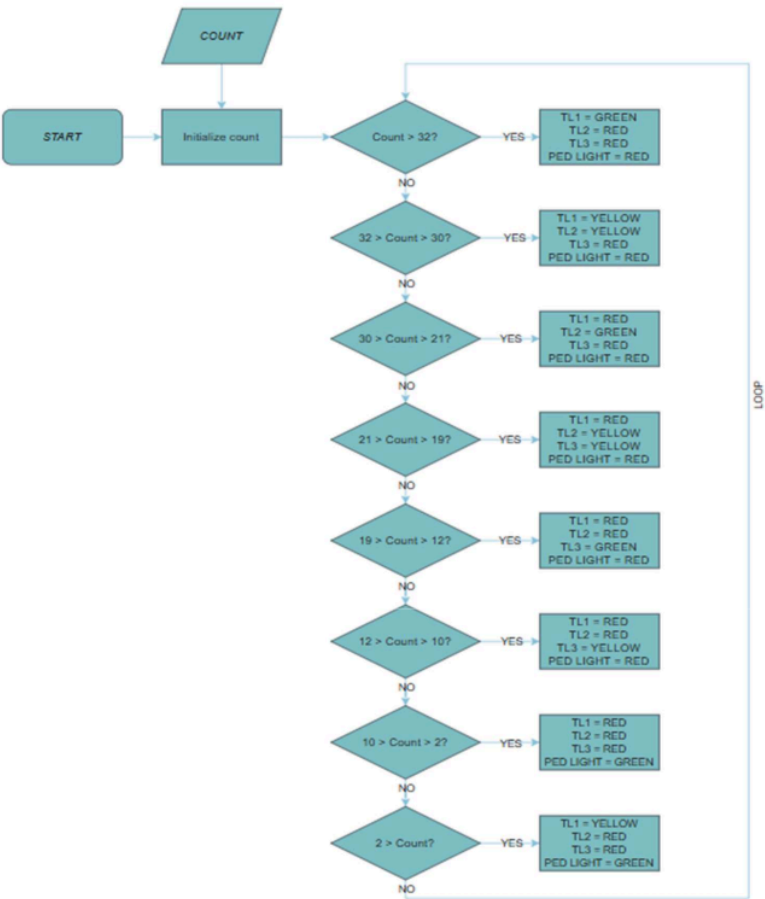
The project successfully achieved its primary aim of designing, coding, and prototyping a timed traffic light system using a microcontroller. Through a structured development process, the system logic was defined using pseudocode and flowcharts, followed by circuit design, simulation, and physical assembly. Testing confirmed that the prototype operated correctly, with traffic and pedestrian lights following the intended timed sequence.

Although the seven-segment countdown display was not fully implemented in the physical prototype due to integration challenges, the core functionality of the system was demonstrated effectively. This outcome highlights the practical application of embedded systems in traffic management and provides valuable insights into the design and implementation process.

References

- [1] A. A. Muhammed *et al.*, "Design and Implementation of a Microcontroller-Based Adaptive Four-Way Traffic Light Control System for Traffic Optimization," *J. Power Energy Control Res.*, Oct. 2024.
- [2] S. S. T. Shantheeveeran and S. R. L., "Traffic Light Control using Sensors and Microcontroller," *Int. J. Eng. Res. Technol.*, vol. 14, no. 8, Aug. 2021.
- [3] Microchip Technology Inc., "PIC16F877A Overview and Applications," Microchip Technical Documentation, 2023.
- [4] Microchip Technology Inc., "PIC16F877A Data Sheet: 8-Bit CMOS FLASH Microcontrollers," Microchip Technology, 2023.
- [5] O. A. Chernov and V. I. Mityakov, "Pseudocode programming of precedents in conceptual designing of automated systems," in *Proc. 2011 5th Int. Conf. Appl. Information and Commun. Technol. (AICT)*, Tashkent, Uzbekistan, Oct. 2011, pp. X–X. doi:10.1109/AICT.2011.6111032
- [6] Programiz, "#9: If Else Statements in C | [2025] C Programming for Beginners," *YouTube*, Dec. 08, 2021. <https://www.youtube.com/watch?v=K8mntKyBJGc&list=PL98qAXLA6aftD9ZInjpLhdQAOFI8xIB6e> (accessed Dec. 03, 2025).
- [7] M. Tereso, A. Pratas, L. Beirante, D. Martinho, F. Bento, and R. Vardasca, "A Web Tool for Representing Algorithmic Flowcharts and Its Support for Pseudocode Learning," in *Adv. Res. Technol. Inf. Innov. Sustain.*, CCIS, vol. 2345, International Conf. on Advanced Research in Technologies, Information, Innovation and Sustainability (ARTIIS 2024), 2025, pp. 381–391
- [8] A. S. Sedra and K. C. Smith, *Microelectronic Circuits*, 8th ed. New York, NY, USA: Oxford Univ. Press, 2020.
- [9] M. Predko, *Programming and Customizing PIC Microcontrollers*, 3rd ed. New York, NY, USA: McGraw-Hill, 2008.
- [10] J. W. Valvano, *Embedded Systems: Real-Time Interfacing to ARM Cortex-M Microcontrollers*, 2nd ed. Austin, TX, USA: CreateSpace, 2017.
- [11] A. S. Sedra and K. C. Smith, *Microelectronic Circuits*, 8th ed. New York, NY, USA: Oxford Univ. Press, 2020.
- [12] T. L. Floyd, *Digital Fundamentals*, 11th ed. Harlow, U.K.: Pearson, 2015.
- [13] R. Seo, "Importance of Breadboards in Electronics Prototyping," *Pcbrunner.com*, Mar. 26, 2025. <https://www.pcbrunner.com/why-are-breadboards-important-in-electronics-prototyping/>
- [14] "In-Circuit Debugger Using." Accessed: Dec. 03, 2025. [Online]. Available: <https://ww1.microchip.com/downloads/aemDocuments/documents/OTH/ProductDocuments/MISC/50002010B.pdf>
- [15] "PIC16F87XA Data Sheet 28/40/44-Pin Enhanced Flash Microcontrollers." Available: <https://ww1.microchip.com/downloads/en/devicedoc/39582b.pdf>

APPENDIX A: FLOW CHART



APPENDIX B: SOURCE CODE

```

1 #include <16F877A.h>
2 #use delay(clock=8000000) // Clock speed 8 MHz
3
4 // Pin definitions; Including each state for each traffic light and the pedestrian lights
5 #define T_RED1 PIN_B0
6 #define T_YELLOW1 PIN_B1
7 #define T_GREEN1 PIN_B2
8 #define T_RED2 PIN_A0
9 #define T_YELLOW2 PIN_A1
10 #define T_GREEN2 PIN_A2
11 #define T_RED3 PIN_E0
12 #define T_YELLOW3 PIN_E1
13 #define T_GREEN3 PIN_E2
14 #define P_RED PIN_B3
15 #define P_GREEN PIN_B4
16
17 // 7-segment display patterns for numbers 0-9 (Common Anode)
18 unsigned int display[10] = {0xC0,0xF9,0xA4,0xB0,0x99,0x92,0x82,0xF8,0x80,0x90}; //count down from 9-0 in HEX numbers
19 unsigned int i = 0;
20 unsigned int tens = 0; //first seven seg display integer defined (tens value)
21 unsigned int ones = 0; //second seven seg display integer defined (ones value)
22 unsigned int count = 0; //count integer defined
23 //=====
24 void main()
25 {
26     set_tris_a(0x00);
27     output_a(0x00);
28     set_tris_b(0x00);
29     output_b(0x00);
30     set_tris_c(0x00); // Set PORTC as output FOR 7-SEG (ones)
31     output_c(0x00); // Initialise PORTC to 0
32     set_tris_d(0x00); // Set PORTD as output FOR 7-SEG (tens)
33     output_d(0x00); // Initialise PORTD to 0
34
35     while (1)
36     {
37         for (count = 40; count > 0; count--) // Countdown from 40 to 0
38         {
39             tens = count / 10; // Tens digit
40             ones = count % 10; // Ones digit
41
42             output_d(display[tens]); // Display tens on 7-seg 1
43             output_c(display[ones]); // Display ones on 7-seg 2
44
45             //=====
46             // TRAFFIC IN DIRECTION 1 TO FLOW FIRST, WHEN THE COUNT IS BETWEEN 40s AND 32s TRAFFIC LIGHT 1 WILL BE GREEN, TRAFFIC LIGHTS 2 AND 3 WILL BE RED, AND PEDESTRIAN LIGHTS SHOW RED.
47             //=====
48             if (count > 32) //WHEN COUNT IS GREATER THAN 32s
49             {
50                 output_high(T_GREEN1); //Traffic light 1; GREEN light ON
51                 output_low(T_YELLOW1); //Traffic light 1; YELLOW light OFF
52                 output_low(T_RED1); //Traffic light 1; RED light OFF
53
54                 output_high(T_RED2); //Traffic light 2; RED light ON
55                 output_low(T_YELLOW2); //Traffic light 2; YELLOW light OFF
56                 output_low(T_GREEN2); //Traffic light 2; GREEN light OFF
57
58                 output_high(T_RED3); //Traffic light 3; RED light ON
59                 output_low(T_YELLOW3); //Traffic light 3; YELLOW light OFF
60                 output_low(T_GREEN3); //Traffic light 3; GREEN light OFF
61
62                 output_high(P_RED); //RED PEDESTRIAN light ON
63                 output_low(P_GREEN); //GREEN PEDESTRIAL light OFF
64             }

```

```

65 //=====
66 //FIRST SEQUENCE CHANGE, BETWEEN 32s AND 30s, TRAFFIC LIGHT 1 WILL CHANGE FROM GREEN TO AMBER, TRAFFIC LIGHT 2 WILL CHANGE FROM RED TO AMBER, TRAFFIC LIGHT 3 AND PEDESTRIAN LIGHTS CONTINUE TO SHOW RED
67 //=====
68     else if (count > 30)           //WHEN COUNT IS GREATER THAN 30s
69     {
70         output_high(T_YELLOW1);    //Traffic Light 1; YELLOW light ON
71         output_low(T_GREEN1);      //Traffic Light 1; GREEN light OFF
72         output_low(T_RED1);        //Traffic Light 1; RED light OFF
73
74         output_high(T_YELLOW2);    //Traffic Light 2; YELLOW light ON
75         output_low(T_RED2);        //Traffic Light 2; RED light OFF
76         output_low(T_GREEN2);      //Traffic Light 2; GREEN light OFF
77
78         output_high(T_RED3);       //Traffic Light 3; RED light ON
79         output_low(T_YELLOW3);     //Traffic Light 3; YELLOW light OFF
80         output_low(T_GREEN3);      //Traffic Light 3; GREEN light OFF
81
82         output_high(P_RED);        //RED PEDESTRIAN light ON
83         output_low(P_GREEN);       //GREEN PEDESTRIAN light OFF
84     }
85 //=====
86 //SECOND SEQUENCE CHANGE, BETWEEN 30s AND 21s, TRAFFIC LIGHT 1 WILL CHANGE FROM AMBER TO RED, TRAFFIC LIGHT 2 WILL CHANGE FROM AMBER TO GREEN, TRAFFIC LIGHT 3 AND PEDESTRIAN LIGHTS CONTINUE TO SHOW RED
87 //=====
88     else if (count > 22)           //WHEN COUNT IS GREATER THAN 21s
89     {
90         output_high(T_RED1);       //Traffic light 1; RED light ON
91         output_low(T_YELLOW1);     //Traffic light 1; YELLOW light OFF
92         output_low(T_GREEN1);      //Traffic light 1; GREEN light OFF
93
94         output_high(T_GREEN2);     //Traffic light 2; GREEN light ON
95         output_low(T_YELLOW2);     //Traffic light 2; YELLOW light OFF
96         output_low(T_RED2);        //Traffic light 2; RED light OFF
97
98         output_high(T_RED3);       //Traffic light 3; RED light ON
99         output_low(T_YELLOW3);     //Traffic light 3; YELLOW light OFF
100        output_low(T_GREEN3);      //Traffic light 3; GREEN light OFF
101
102        output_high(P_RED);        //RED PEDESTRIAN light ON
103        output_low(P_GREEN);       //GREEN PEDESTRIAN light OFF
104    }
105 //=====
106 //THIRD SEQUENCE CHANGE, BETWEEN 21s AND 19s, TRAFFIC LIGHT 2 WILL CHANGE FROM GREEN TO AMBER, TRAFFIC LIGHT 3 WILL CHANGE FROM RED TO AMBER, TRAFFIC LIGHT 1 AND PEDESTRIAN LIGHTS WILL CONTINUE TO SHOW RED
107 //=====
108     else if (count > 20)           //WHEN COUNT IS GREATER THAN 19s
109     {
110         output_high(T_RED1);       //Traffic light 1; RED light ON
111         output_low(T_GREEN1);      //Traffic light 1; GREEN light OFF
112         output_low(T_YELLOW1);     //Traffic light 1; YELLOW light OFF
113
114         output_high(T_YELLOW2);    //Traffic light 2; YELLOW light ON
115         output_low(T_RED2);        //Traffic light 2; RED light OFF
116         output_low(T_GREEN2);      //Traffic light 2; GREEN light OFF
117
118         output_high(T_YELLOW3);    //Traffic light 3; YELLOW light ON
119         output_low(T_RED3);        //Traffic light 3; RED light OFF
120         output_low(T_GREEN3);      //Traffic light 3; GREEN light OFF
121
122         output_high(P_RED);        //RED PEDESTRIAN light ON
123         output_low(P_GREEN);       //GREEN PEDESTRIAN light OFF
124     }
125 //=====
126 //FOURTH SEQUENCE CHANGE, BETWEEN 19s AND 12s, TRAFFIC LIGHT 2 WILL CHANGE FROM AMBER TO RED, TRAFFIC LIGHT 3 WILL CHANGE FROM AMBER TO GREEN, TRAFFIC LIGHT 1 AND PEDESTRIAN LIGHTS WILL CONTINUE TO SHOW RED
127 //=====
128     else if (count > 12)           //WHEN COUNT IS GREATER THAN 12s
129     {
130         output_high(T_RED1);       //Traffic light 1; RED light ON
131         output_low(T_YELLOW1);     //Traffic light 1; YELLOW light OFF
132         output_low(T_GREEN1);      //Traffic light 1; GREEN light OFF
133
134         output_high(T_RED2);       //Traffic light 2; RED light ON
135         output_low(T_YELLOW2);     //Traffic light 2; YELLOW light OFF
136         output_low(T_GREEN2);      //Traffic light 2; GREEN light OFF
137
138         output_high(T_GREEN3);     //Traffic light 3; GREEN light ON
139         output_low(T_YELLOW3);     //Traffic light 3; YELLOW light OFF
140         output_low(T_RED3);        //Traffic light 3; RED light OFF
141
142         output_high(P_RED);        //RED PEDESTRIAN light ON
143         output_low(P_GREEN);       //GREEN PEDESTRIAN light OFF
144     }

```

```

145 //=====
146 //FIFTH SEQUENCE CHANGE, BETWEEN 12s AND 10s, TRAFFIC LIGHT 1 WILL CHANGE FROM GREEN TO AMBER, TRAFFICS LIGHTS 1, 2 AND PEDESTRIAN LIGHTS WILL CONTINUE TO SHOW RED
147 //=====
148     else if (count > 10)           //WHEN COUNT IS GREATER THAN 10s
149     {
150         output_high(T_RED1);        //Traffic light 1; RED light ON
151         output_low(T_GREEN1);       //Traffic light 1; GREEN light OFF
152         output_low(T_YELLOW1);      //Traffic light 1; YELLOW light OFF
153
154         output_high(T_RED2);        //Traffic light 2; RED light ON
155         output_low(T_YELLOW2);       //Traffic light 2; YELLOW light OFF
156         output_low(T_GREEN2);       //Traffic light 2; GREEN light OFF
157
158         output_high(T_YELLOW3);     //Traffic light 3; YELLOW light ON
159         output_low(T_RED3);         //Traffic light 3; RED light OFF
160         output_low(T_GREEN3);       //Traffic light 3; GREEN light OFF
161
162         output_high(P_RED);         //RED PEDESTRIAN light ON
163         output_low(P_GREEN);        //GREEN PEDESTRIAN light OFF
164     }
165 //=====
166 //SIXTH SEQUENCE CHANGE, BETWEEN 10s AND 2s, ALL TRAFFIC LIGHTS WILL SHOW RED, PEDESTRIAN LIGHT WILL SHOW GREEN
167 //=====
168     else if (count > 2)           //WHEN COUNT IS GREATER THAN 2s
169     {
170         output_high(T_RED1);        //Traffic light 1; RED light ON
171         output_low(T_YELLOW1);       //Traffic light 1; YELLOW light OFF
172         output_low(T_GREEN1);       //Traffic light 1; GREEN light OFF
173
174         output_high(T_RED2);        //Traffic light 2; RED light ON
175         output_low(T_YELLOW2);       //Traffic light 2; YELLOW light OFF
176         output_low(T_GREEN2);       //Traffic light 2; GREEN light OFF
177
178         output_high(T_RED3);        //Traffic light 3; RED light ON
179         output_low(T_YELLOW3);       //Traffic light 3; YELLOW light OFF
180         output_low(T_GREEN3);       //Traffic light 3; GREEN light OFF
181
182         output_low(P_RED);          //RED PEDESTRIAN light OFF
183         output_high(P_GREEN);       //GREEN PEDESTRIAN light ON
184     }
185 //=====
186 //FINAL SEQUENCE CHANGE, FOR THE FINAL 2 SECONDS OF THE COUNT, TRAFFIC LIGHT 1 WILL CHANGE TO AMBER, TRAFFIC LIGHTS 1 AND 2 REMAIN RED AND PEDESTRIAN LIGHT REMAINS GREEN, THIS IS TO CONTINUE THE LOOP AT THE RETURN TO 40s
187 //=====
188     else                           //FOR THE REMAINDER OF THE COUNT
189     {
190         output_high(T_YELLOW1);     //Traffic light 1; YELLOW light ON
191         output_low(T_GREEN1);       //Traffic light 1; GREEN light OFF
192         output_low(T_RED1);         //Traffic light 1; RED light OFF
193
194         output_high(T_RED2);        //Traffic light 2; RED light ON
195         output_low(T_YELLOW2);       //Traffic light 2; YELLOW light OFF
196         output_low(T_GREEN2);       //Traffic light 2; GREEN light OFF
197
198         output_high(T_RED3);        //Traffic light 3; RED light ON
199         output_low(T_YELLOW3);       //Traffic light 3; YELLOW light OFF
200         output_low(T_GREEN3);       //Traffic light 3; GREEN light OFF
201
202         output_low(P_RED);          //RED PEDESTRIAN light OFF
203         output_high(P_GREEN);       //GREEN PEDESTRIAN light ON
204     }
205
206     delay_ms(200);                //ALLOW A 200ms DELAY FOR THE COUNT TO REPLICATE SECONDS
207 }
208 }
209 }
210

```