

Charlie Rainey - Electronics 1

by Charlie Rainey

Submission date: 28-Nov-2025 03:11PM (UTC+0000)

Submission ID: 269002274

File name: 57736_Charlie_Rainey_Charlie_Rainey_-_Electronics_1_634321_1794822930.docx (1.58M)

Word count: 3297

Character count: 16959

ACDB5002 Electronics I Assignment

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

Charlie Rainey

2414414

05/12/2025

Total Word Count: 2,493

Abstract

Traffic Congestion can be a major issue in largely populated areas as well as pedestrian safety whether it is a person trying to cross the road or someone commuting by bicycle. Traffic lights and traffic patterns are a suitable way to help reduce traffic jams whilst keeping everyone on the road safe at the same time. So, the aim of this assignment is to design and code a circuit using proteus an automatic traffic light controller for a T-junction Road to allow a smooth flow of traffic whilst taking pedestrian safety into account. The circuit was designed with a PIC16F877A microcontroller to allow smooth operation of three traffic lights all working at once with the 7-segment display timer counting down from 40-0. The first 30 seconds is for the operation of the traffic lights, whilst the last 10 seconds is for the operation of the pedestrian crossing lights. After the circuit was designed on proteus the program was to be written in 'c' language using ccs c compiler. This was to allow the microcontroller to interact with all the inputs and outputs and function the correct working of all the LEDs in the circuit.

Contents Page

Contents

Abstract.....	2
Contents Page.....	3
Introduction.....	4
Background Research.....	5
Design.....	6
Methodology.....	10
Analysis of Results.....	16
Conclusion and Recommendations	21
References	22
Figure 1 Three-way traffic light system schematic capture.....	6
Figure 2 - Close up of traffic light 1.....	7
Figure 3 - Close up of traffic light 2.....	7
Figure 4 - Close up of traffic light 3.....	8
Figure 5 - Close up of pedestrian LEDs.....	8
Figure 6 - Close up of 7-segment displays.....	9
Figure 7 - Close up of PIC microcontroller and crystal oscillator.....	9
Figure 8 - Header and configuration	10
Figure 9 - Pin definitions	10
Figure 10 - 7-segment display.....	10
Figure 11 - Hardware set up.....	11
Figure 12 - Case 1.....	11
Figure 13 - Case 2.....	12
Figure 14 - Case 3.....	12
Figure 15 - Case 4.....	13
Figure 16 - Case 5.....	14
Figure 17 - Reset cycle	14
Figure 18 - Operational flowchart.....	16
Figure 19 - Case 1 and 2 displayed.....	17
Figure 20 - Case 3 displayed.....	17
Figure 21 - Case 4 displayed.....	18
Figure 22 - Case 5 displayed.....	19
Figure 23 - Traffic light prototype	20

Introduction

A major issue in big cities with dense populations is traffic congestion. The number of cars waiting in line for service has dramatically increased as a result of the city's road network being used beyond its intended capacity. Large lines form, waiting times sharply rise, and overall output suffers since workers might not arrive at their destination on time [1]. Not only this but, the right to the highest level of safety and comfort on the street belongs to pedestrians. Sometimes drivers of cars and motorcycles put pedestrians in danger by breaking traffic laws, including speeding [2]. Therefore, a safety device is required to shield pedestrians from drivers that break traffic laws [2].

To prevent any of this from happening the design of an automatic traffic light controller for a T-Junction Road was designed using proteus. Electronic design engineers and technicians are the primary users of Proteus software, which they use to produce schematics and electronic prints for printed circuit boards (PCBs) and as a rapid prototyping tool for research and development [3]. Additionally, it is used at universities all over the world to educate students in PCB layout, embedded design, and electronics.

Once the circuit was designed and created on proteus the circuit had to be programmed and this was done using 'C' language. 'C' is a versatile programming language with efficient expression, advanced control flow and data structure features, and a wide range of operators and data types. It is more practical and efficient for many jobs than supposedly more powerful languages because of its universality and lack of limitations. Numerous programs, such as the UNIX operating system and the C compiler itself, operate in the C language.

Background Research

Right-of-way clarification is the primary purpose of traffic lights. By clearly dividing the travel times for cars and pedestrians in different directions, the red, yellow, and green lights essentially prevent disruption brought on by unnecessary rushing by letting everyone know when to stop and when to continue [4]. Misjudgements between cars and pedestrians at crossroads without traffic lights increase the likelihood of collisions. By providing proper advice, traffic lights can greatly lessen these confrontations and improve intersection safety [4]. Pedestrian safety is greatly aided by traffic signal lights. Pedestrians can confidently navigate busy junctions thanks to the safe crossing sites provided by pedestrian crossing signals and pedestrian lights [5]. Traffic management systems can guarantee that pedestrians have designated time to cross the street safely by combining pedestrian signals with vehicle traffic lights [5].

Based on preprogramed logic, microcontrollers are employed in traffic lights to regulate the timing and sequencing of the red, yellow, and green lights. In order to control traffic flow and lessen congestion, the system can dynamically modify light durations based on information from sensors (such as infrared or cameras) that identify the presence of vehicles [6]. Microcontrollers are used in modern systems for their adaptability, affordability, and capacity to incorporate sophisticated features like communication interfaces or emergency modes [6]. One affordable option is to create an intelligent traffic light controller using a microcontroller. When combined with contemporary wireless communication technology, the hardware system design can offer more powerful and adaptable services [7].

There are numerous ways to regulate traffic light time, but they can be broadly classified into two categories. Automatic controlling and manual controlling, which requires a trained individual to keep an eye on vehicles and regulate their flow. Automatic controlling is typically divided into two categories, sensor-based and image processing-based [8]. Sensor based approaches rely on sensor networks, such as magnetic loop detectors buried beneath the road or infrared radar sensors, to identify vehicles [8]. The image processing-based approach relies on photos taken from recordings captured by cameras installed on traffic lights [8]. The number of cars on each traffic light is calculated, and the traffic light is subsequently regulated based on this figure [8].

Design

The chosen circuit design is to implement a three-way traffic light system at a t-road junction including a pedestrian crossing. The circuit was to include timers to show the smooth operation and timing of the traffic light changes. The entire circuit was designed using proteus.

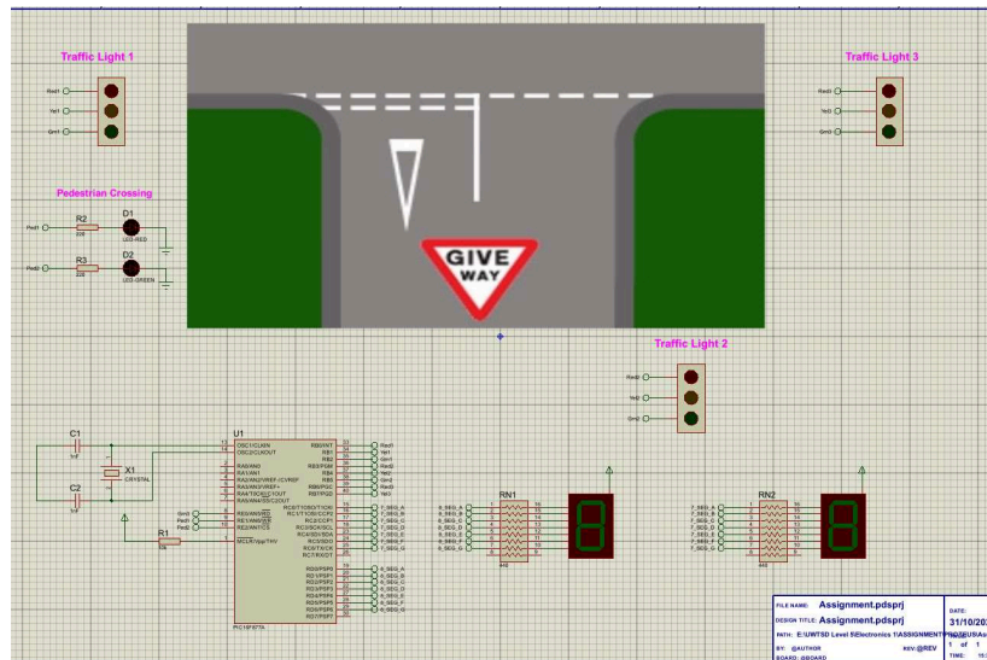


Figure 1 Three-way traffic light system schematic capture

Figure 1 shows the circuit outlay of the T-junction with the three sets of traffic lights around the road as well as the pedestrian crossing LEDs. In the bottom right is two 7-segment displays which are the timers, then finally on the bottom left is the PIC16F877A microcontroller connected to a crystal oscillator. The aim of this circuit is, that for the first 10 seconds 'traffic light 1' will operate, then the next 10 seconds 'traffic light 2' will operate, then the next 10 seconds 'traffic light 3' will operate and for the final 10 seconds all traffic lights will be red and then pedestrian LEDs will turn on. All whilst the two 7-segment displays count down from 40 seconds. The reason for using a PIC16F877A is because of its cost effectiveness versatility and robustness. This microcontroller has a wide range of applications and is commonly used in embedded systems.

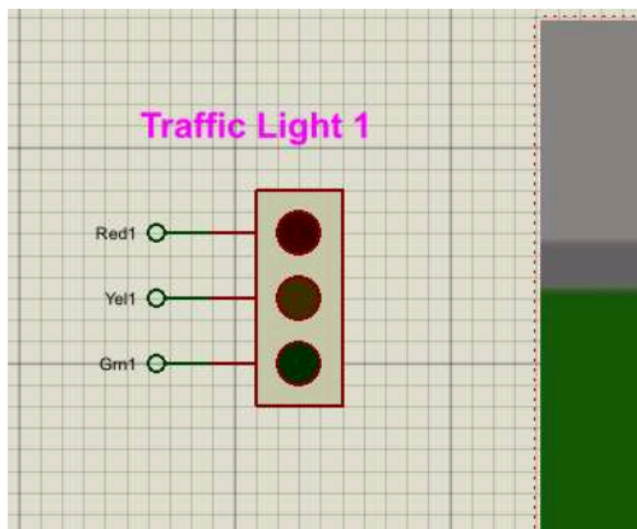


Figure 2 - Close up of traffic light 1

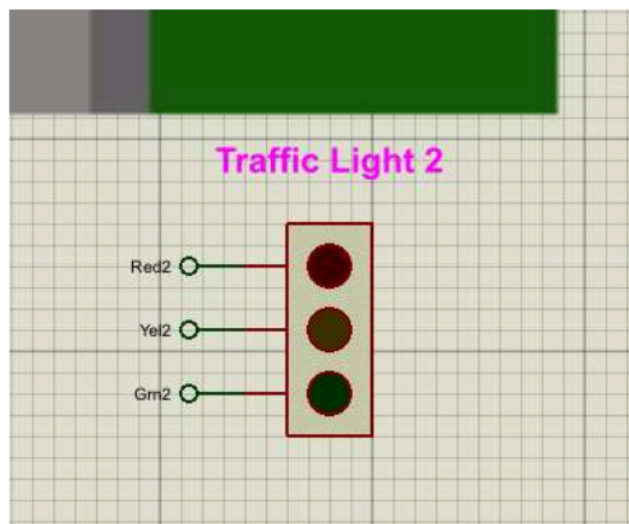


Figure 3 - Close up of traffic light 2

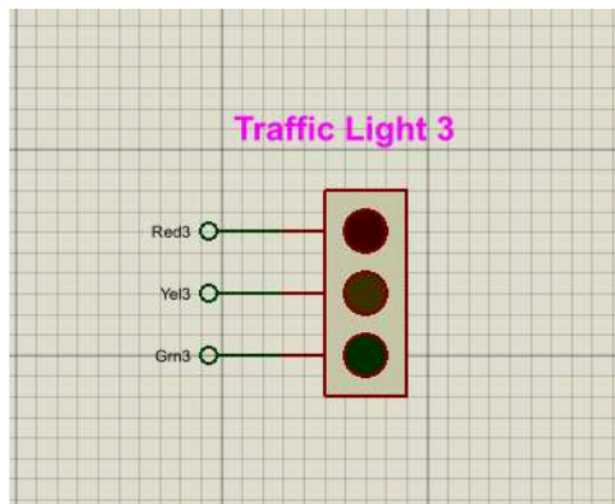


Figure 4 - Close up of traffic light 3

Figures 2, 3 and 4 show a close up of each traffic lights to show how they are connected properly and what each output is labelled as. Each output is connected to the PIC in port B and one output 'Grn3' is connected into port E as there was no room left in port B.

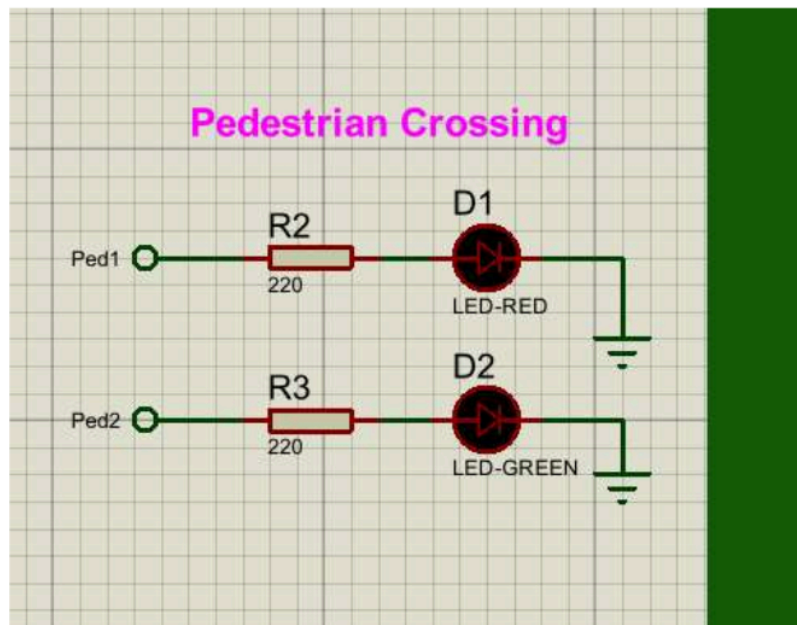


Figure 5 - Close up of pedestrian LEDs

Figure 5 shows the two different LEDs for the pedestrian crossings connected to two 220-ohm resistors. They have two separate outputs and both outputs are connected to Port E of the pic. When the traffic lights are in operation the red LED is illuminated as traffic is flowing but when it is safe for the pedestrian to cross the green LED will illuminate.

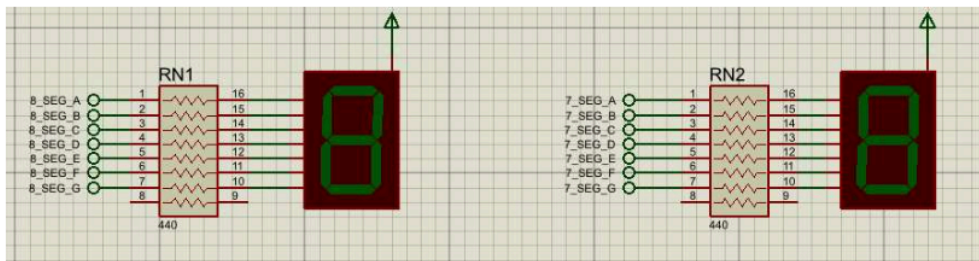


Figure 6 - Close up of 7-segment displays

Figure 6 displays the 7-segment display timers. RN1 counts the tens so in this case from 4 down to 0 and RN2 counts the single seconds so from 9 down to 0. Each display is connected to a bank of 440-ohm resistors and has 7 outputs. Each one to light up the correct section of the display. The 7_SEG outputs are connected to port C and the 8_SEG outputs are connected to port D.

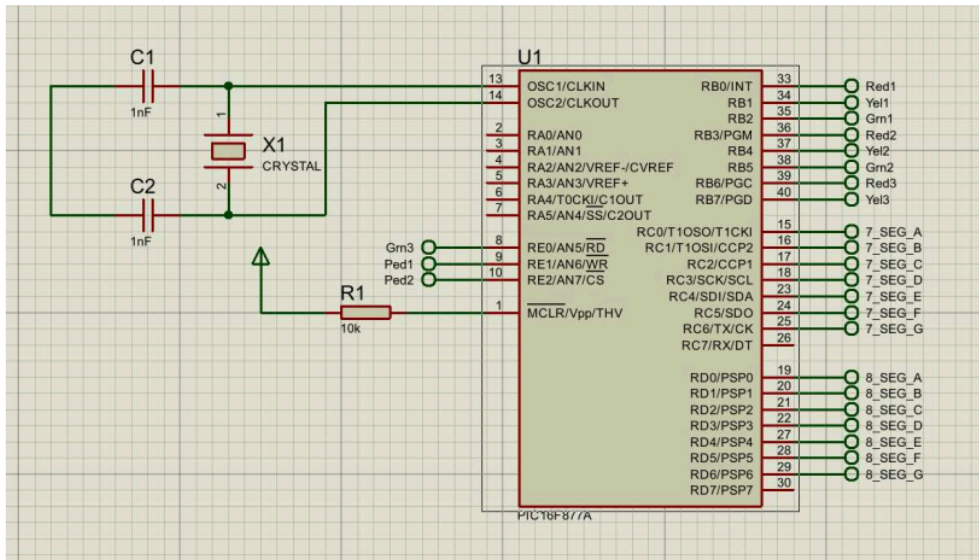


Figure 7 - Close up of PIC microcontroller and crystal oscillator

Figure 7 shows the PIC16F877A microcontroller that was chosen. It also shows clearly what each port all the outputs are connected to and how the microcontroller can operate. The reason for using this specific microcontroller is because The PIC16F877A is a popular microcontroller for a variety of embedded systems because it is inexpensive, adaptable, and simple to program [9]. The PIC also offers a suitable range of ports to connect outputs and if something was to be added then there is a spare port that has not been used. Not only this but figure 7 also shows the crystal oscillator that is connected to the microcontroller. The core of almost all frequency control systems is a vibrating quartz crystal. Quartz crystal oscillators, which are electronic circuits that exploit the mechanical resonances of vibrating crystals of piezoelectric materials to create periodically varying electrical signals, are sources of exact frequency and accurate time [10].

Methodology

To allow full operation of the traffic lights and the 7-segment display, the circuit had to be coded. To do this the CCS C compiler app was used and the code was written in 'C' language. This was then able to be compiled from CCS to Proteus where the circuit was built and the circuit was then fully operational.

```

1
2
3
4
5
6
7
8 #include <16F877A.h>
9 #use delay(clock=8000000)
10

```

Figure 8 - Header and configuration

Figure 8 is the start of the code, where the correct device header is included and tells the compiler that the PIC is running from an 8MHz clock. This ensures that there is correct timing for delay functions.

```

10
11 // Define traffic and pedestrian light pins
12 #define RED_1 PIN_B0
13 #define YEL_1 PIN_B1
14 #define GRN_1 PIN_B2
15 #define RED_2 PIN_B3
16 #define YEL_2 PIN_B4
17 #define GRN_2 PIN_B5
18 #define RED_3 PIN_B6
19 #define YEL_3 PIN_B7
20 #define GRN_3 PIN_E0
21 #define PED_1 PIN_E1
22 #define PED_2 PIN_E2
23

```

Figure 9 - Pin definitions

Figure 9 shows each traffic light and pedestrian light as an output with their own readable names. Each output has been defined to its own pins in port B and port E. This improves clarity and makes hardware changes easier.

```

23
24 // 7-segment display digit patterns (assuming common cathode)
25 unsigned int display[10] = {0x90, 0x80, 0xF8, 0x82, 0x92, 0x99, 0xB0, 0xA4, 0xF9, 0xC0};
26 void setup_hw(void) {
27

```

Figure 10 - 7-segment display

Figure 10 shows each digit from 0-9 being given to the 7-segment display and it gets stored for later in the code. Each value corresponds to which segments are on or off.

```

27
28  set_tris_b(0x00); // Set PORTB as output for LEDs
29  output_b(0x00);  // Initialize PORTB to 0
30  set_tris_c(0x00); // Set PORTC as output for 7-segment display
31  output_c(0x00);  // Initialize PORTC to 0
32  set_tris_d(0x00); // Set PORTD as output for 7-segment display
33  output_d(0x00);  // Initialize PORTD to 0
34  set_tris_e(0x00); // Set PORTE as output for LEDs
35  output_e(0x00);  // Initialize PORTE to 0
36 }

```

Figure 11 - Hardware set up

Figure 11 displays that port B, C, D and E have been configured as outputs. All traffic lights, pedestrian lights and 7-segment displays are directly driven from these ports. Set_tris_b(0x00) means that all pins on that port are outputs, same for C, D and E. output_b(0x00) clears port B after the start up, same again for C, D and E.

The rest of the code is split up into five different cases. Each case responds to a different stage of the traffic light cycle. Every case uses a loop of 10 seconds and sets the pedestrian and traffic light states.

```

37 void case1(void) {
38   int i;
39   for (i=9; i<10; i++) {
40     output_d(0x99); // 7-segment display '4'
41     output_c(display[i]); // Update 7-segment counter
42     output_high(RED_1); // RED 1 Light On
43     output_low(RED_2);
44     output_low(RED_3);
45     output_low(YEL_1);
46     output_high(YEL_2); // Yellow 2 light on
47     output_low(YEL_3);
48     output_low(GRN_1);
49     output_low(GRN_2);
50     output_high(GRN_3); // Green 3 light on
51     output_high(PED_1); // Red pedestrian light to stay on
52     output_low(PED_2); // Green pedestrian light to stay off
53     delay_ms(1000);
54   }

```

Figure 12 - Case 1

Figure 12 displays the first out of five cases. The first case sets the 7-segment displays to count down from 40-30 seconds. Here the output is set high on RED_1, YEL_2, GRN_3 and PED_1 which is to turn all of them traffic lights on and the output is set low on the rest to keep them off.

```

56 void case2(void) {
57     int i;
58     for (i=0; i<10; i++) {
59         output_d(0xB0); // 7-segment display '3'
60         output_c(display[i]); // Update 7-segment counter
61         output_high(RED_1); // RED 1 Light On
62         output_low(RED_2);
63         output_low(RED_3);
64         output_low(YEL_1);
65         output_high(YEL_2); // Yellow 2 Light on
66         output_low(YEL_3);
67         output_low(GRN_1);
68         output_low(GRN_2);
69         output_high(GRN_3); // Green 3 light on
70         output_high(PED_1); // Red pedestrian light to stay on
71         output_low(PED_2); // Green pedestrian light to stay off
72         delay_ms(1000);
73     }

```

Figure 13 - Case 2

Figure 13 displays the second case out of five cases. This case sets the 7-segment displays to count down from 39-30 seconds. The outputs are the same as the first case as these two cases are the same. The only difference is case 1 gets the timer to start counting from 40 rather than 39.

```

75 void case3(void) {
76     int i;
77     for (i=0; i<10; i++) {
78         output_d(0xA4); // 7-segment display '2'
79         output_c(display[i]); // Update 7-segment counter
80         output_low(RED_1);
81         output_low(RED_2);
82         output_high(RED_3); // Red 3 light on
83         output_high(YEL_1); // Yellow 1 light on
84         output_low(YEL_2);
85         output_low(YEL_3);
86         output_low(GRN_1);
87         output_high(GRN_2); // Green 2 light on
88         output_low(GRN_3);
89         output_high(PED_1); // Red pedestrian light stays on
90         output_low(PED_2); // Green pedestrian light stays off
91         delay_ms(1000);
92     }
93 }

```

Figure 14 - Case 3

Figure 14 displays the third case out of five cases. This case sets the 7-segment display to countdown from 30-20 seconds. However, the outputs are set different here as RED_3, YEL_1, GRN_2 and PED_1 are set high, so they are turned on and the rest are set low to stay off.

```

93 }
94 void case4(void) {
95     int i;
96     for (i=0; i<10; i++) {
97         output_d(0xF9); // 7-segment display '1'
98         output_c(display[i]); // Update 7-segment counter
99         output_low(RED_1);
100        output_high(RED_2); // Red 2 light on
101        output_low(RED_3);
102        output_low(YEL_1);
103        output_low(YEL_2);
104        output_high(YEL_3); // Yellow 3 light on
105        output_high(GRN_1); // Green 1 light on
106        output_low(GRN_2);
107        output_low(GRN_3);
108        output_high(PED_1); // Red pedestrian on
109        output_low(PED_2); // Green pedestrian off
110        delay_ms(1000);
111    }
112 }

```

Figure 15 - Case 4

Figure 15 displays the fourth case out of five cases. This case sets the 7-segment display to countdown from 20-10 seconds. This time the outputs that are set high are RED_2, YEL_3, GRN_1 and PED_1. The rest of the lights are off as they are all set low.

```

113 void case5(void) {
114     int i;
115
116     // ---19...10---
117     for (i=0; i<10; i++) {
118         output_d(0xC0); // 7-segment display '1'
119         output_c(display[i]); // Update 7-segment counter
120
121         // LED states
122         output_high(RED_1);
123         output_high(RED_2);
124         output_high(RED_3);
125         output_low(YEL_1);
126         output_low(YEL_2);
127         output_low(YEL_3);
128         output_low(GRN_1);
129         output_low(GRN_2);
130         output_low(GRN_3);
131         output_low(PED_1);
132         output_high(PED_2);
133         delay_ms(1000);
134     }

```

Figure 16 - Case 5

Figure 16 is the last case where the 7-segment displays countdown from 10-1 seconds. All red traffic light outputs are set high, so all red traffic lights are on to prevent any cars from driving. PED_2 is also set high, so the green pedestrian light comes on telling pedestrians it is safe to cross the road.

```

135
136     // --- then 00 once ---
137     output_d(0xC0); // 7-segment display '0'
138     output_c(display[0]); // Right digit 0
139
140     output_low(RED_1);
141     output_low(RED_2);
142     output_low(RED_3);
143     output_low(YEL_1);
144     output_low(YEL_2);
145     output_low(YEL_3);
146     output_low(GRN_1);
147     output_low(GRN_2);
148     output_low(GRN_3);
149     output_low(PED_1);
150     output_low(PED_2);
151     delay_ms(1000);
152 }
153 void main() {

```

Figure 17 - Reset cycle

Figure 17 shows all outputs low so every led and traffic lights are off. This only happens when both 7-segment displays are displaying 0. This acts as a reset state before the cycle restarts.

Analysis of Results



Figure 18 - Operational flowchart

Figure 18 is a flowchart that shows the operation of the three-way traffic light control system that was built on proteus. It shows what lights should be on at each 10 second segment of the operation and how the system turns all traffic lights off towards the end for the pedestrian crossing.

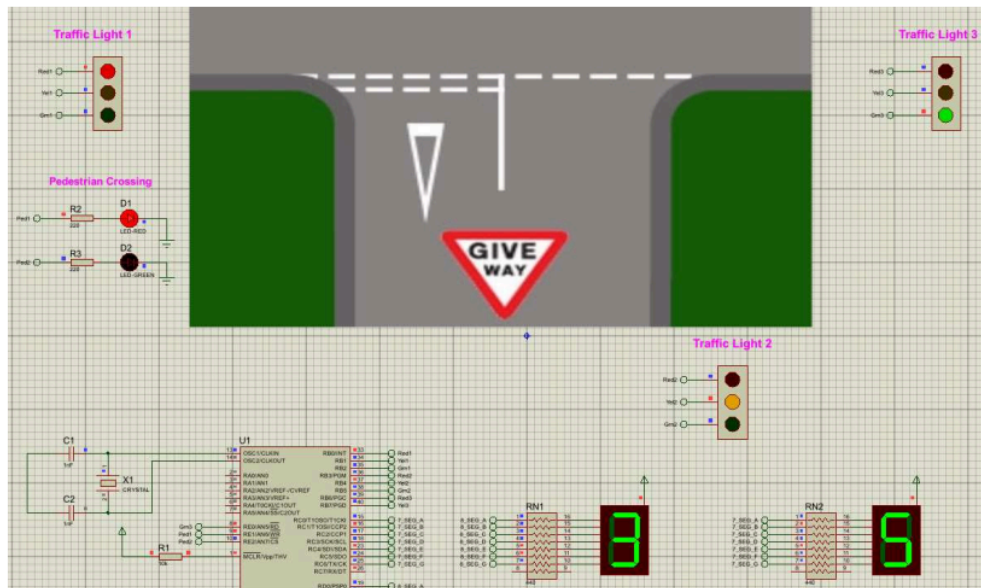


Figure 19 - Case 1 and 2 displayed

Figure 19 shows case 1 and 2 in proteus with the circuit operating. The time counts down from 40-30seconds with traffic light 1 on red, traffic light 2 on amber, traffic light 3 on green and the pedestrian crossing on red. This clearly matches with the first section of the flowchart, so it allows awareness that the circuit operation is working properly.

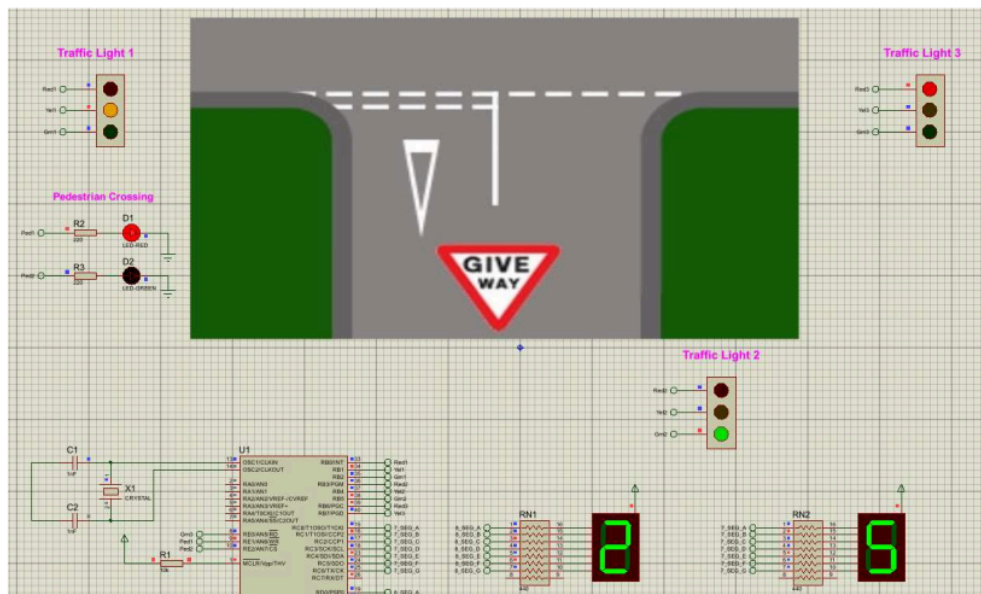


Figure 20 - Case 3 displayed

Figure 20 displays case 3 operating in proteus. This follows the second part of the flowchart as the timer can be seen counting down from 30-20 seconds with traffic light 1 on amber, traffic light 2 on green, traffic light 3 on red and the pedestrian crossing on red.

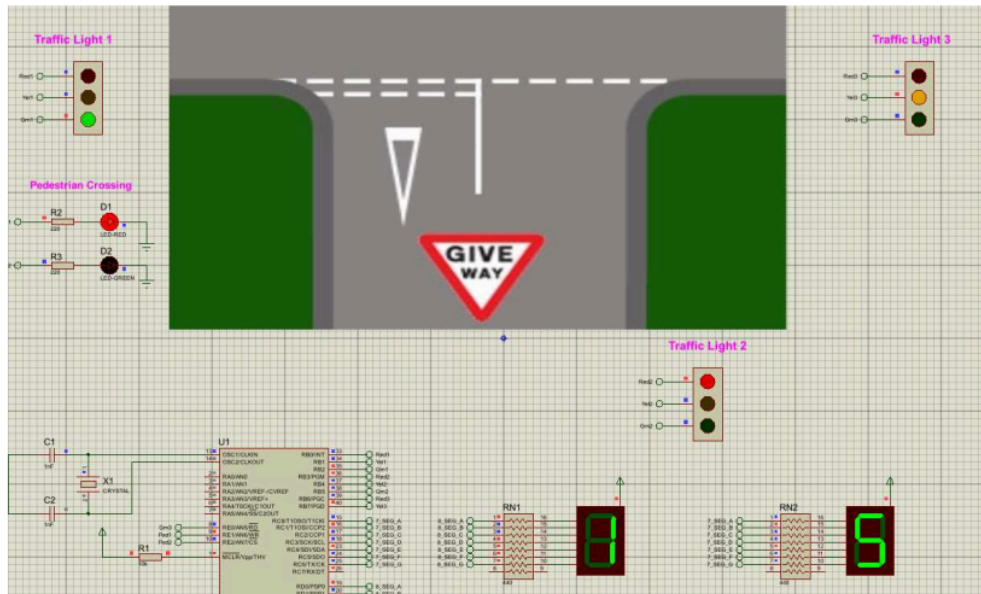


Figure 21 - Case 4 displayed

Figure 21 is showing the counter go from 20-10 seconds which is the fourth case of the sequence. Here, traffic light 1 is on green, traffic light 2 on red, traffic light 3 on amber and the pedestrian crossing still on red. This sequence is still following the operation of the created flowchart.

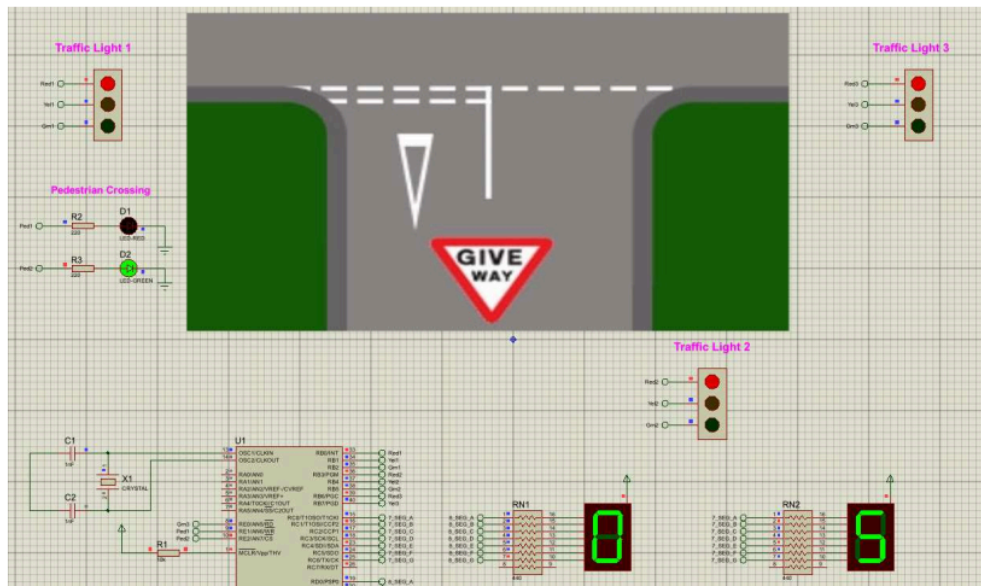


Figure 22 - Case 5 displayed

The final case is displayed above in figure 22, this is where all traffic lights have been turned to red and the pedestrian light is on green to display that it is safe to cross the road. This happens for the final 10 seconds of the circuit operation. This shows that each case of the circuit operation follows the flowchart for what is expected to happen, meaning the circuit is fully operational and works as required.

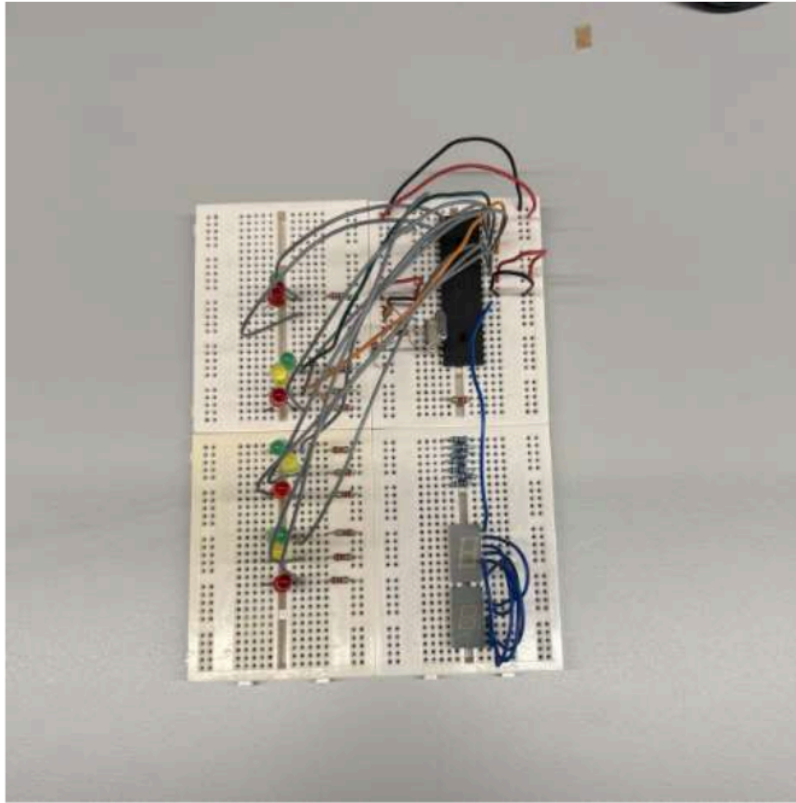


Figure 23 - Traffic light prototype

Figure 23 is an attempt at building the three-way traffic light system using a breadboard and components. This is only a prototype and is not fully operational yet. The same components were used that were displayed in the proteus circuit diagram such as the PIC16F877A. There was a struggle to get the traffic light LEDs to operate in their proper sequence as well as getting the 7-segment displays to turn on that why this is still a prototype.

Conclusion and Recommendations

The development of an automatic traffic light controller for a T-junction road shows how embedded systems, microcontroller programming, and digital circuit design can be integrated to address real world problems with traffic management and pedestrian safety. Throughout this project, the objective was to design and implement a fully functional system capable of coordinating three vehicle traffic lights and a pedestrian crossing using a PIC16F877A microcontroller. The successful operation of the circuit in proteus highlights the effectiveness of the chosen components.

The project's design phase proves a good structure for the system, including three traffic light units, two 7-segment displays, and pedestrian LEDs. The PIC16F877A microcontroller proved particularly suitable due to its versatility, robustness, and the availability of sufficient I/O ports for all required outputs. The inclusion of a crystal oscillator allows accurate timing, which is essential for both LED switching and the countdown sequence. By dividing the program into five operational cases, the timing of each traffic phase and the pedestrian phase was made systematic, reliable, and easy to validate.

The program phase, completed using the CCS C compiler, was crucial in translating the design into a functioning control system. Each element of the circuit, from LED states to the 7-segment displays, responded to the programmed logic, demonstrating that the methodology was well-structured and that the circuit was logically okay. The analysis of results further confirmed this, as each operational phase matched with what was expected as shown in the flowchart.

Although the breadboard prototype did not reach full functionality, the attempt provided practical insight into the challenges of transitioning from simulation to physical implementation. As a recommendation more time should be allowed for the prototype to achieve the desired functionality. A better understanding of how the 7-segment displays operate and where connections should be made on the breadboard to allow them to operate is required. Another improvement would be to achieve full operation of the traffic lights on the prototype as only certain LEDs were illuminating as required.

Overall, the project successfully met its aims. It delivered a clear demonstration of how microcontrollers can support efficient, automated traffic control solutions that enhance both vehicle flow and pedestrian safety. The work also provided valuable experience in embedded system design, digital logic, and practical circuit assembly.

References

- [1]
A. Ezzat, H. Farouk, K. El-Kilany, and A. Moneim, "Optimization Using Simulation of Traffic Light Signal Timings," 2014. Accessed: Nov. 28, 2025. [Online]. Available: <https://ieomsociety.org/ieom2014/pdfs/499.pdf>

- [2]
D. Hardiyanto, D. Anggun, and M. Rojali, "Pedestrian Crossing Safety System at Traffic Lights based on Decision Tree Algorithm," *International Journal of Advanced Computer Science and Applications*, vol. 10, no. 8, 2019, doi: <https://doi.org/10.14569/ijacsa.2019.0100850>.

- [3]
E. & H. C. Warangal NIT, "What is Proteus," *Medium*, Oct. 14, 2021. <https://ea-ham-nitw.medium.com/what-is-proteus-6d408f5d2d5c>

- [4]
"Traffic Lights: Working Principles & Importance in Road Safety," *Uken-industry.com*, 2025. <https://www.uken-industry.com/traffic-lights-how-they-work-and-their-importance-in-road-> (accessed Nov. 28, 2025).

- [5]
FAMA TRAFFIC, "What is the Meaning and Importance of Traffic Signal Lights?," *Ledtrafficlight.cn*, Jul. 15, 2024. <https://www.ledtrafficlight.cn/meaning-and-importance-of-traffic-signal-lights>

- [6]
A. A. Muhammed, O. Gregory, and E. O. Aigbodion, "Design and Implementation of a Microcontroller-Based Adaptive Four-Way Traffic Light Control System for Traffic Optimization," *Journal of Power, Energy, and Control*, vol. 1, no. 2, pp. 92–102, Oct. 2024, doi: <https://doi.org/10.62777/pec.v1i2.34>.

- [7]
J. Pang, "Review of microcontroller based intelligent traffic light control," vol. 7, pp. 1–5, Oct. 2015, doi: <https://doi.org/10.1109/cewit.2015.7338166>.

- [8]
"Smart controlling for traffic light time | IEEE Conference Publication | IEEE Xplore," *ieeexplore.ieee.org*. <https://ieeexplore.ieee.org/abstract/document/8257768>

- [9]
RayMing, "What is PIC microcontroller ?," *RAYPCB*, Aug. 30, 2021. <https://www.raypcb.com/pic-microcontrollers/>

[10]

H. Zhou, C. Nicholls, T. Kunz, and H. Schwartz, "Frequency Accuracy & Stability Dependencies of Crystal Oscillators," 2008. Available: <https://kunuz-pc.sce.carleton.ca/thesis/CrystalOscillators.pdf>