

2218001 - Rebecca Coles - Electronics 1 T-Junction

by Rebecca Coles

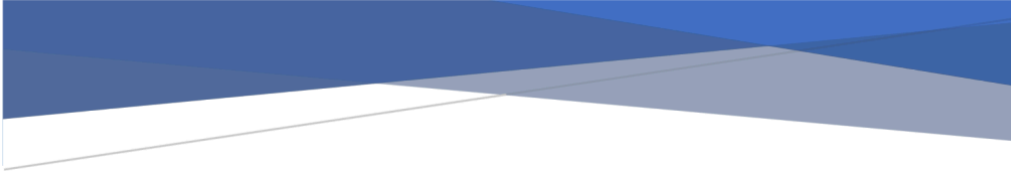
Submission date: 08-Jan-2025 03:05PM (UTC+0000)

Submission ID: 247652684

File name: 41397_Rebecca_Coles_2218001_-_Rebecca_Coles_-_Electronics_1_T-Junction_634321_1498447094.docx (776.3K)

Word count: 3144

Character count: 17141



DESIGNING AN AUTOMATIC TRAFFIC LIGHT CONTROLLER FOR A T-JUNCTION ROAD

Electronics 1
ACDB5002

Rebecca Coles
2218001

Electrical & Electronic Engineering

Table of Contents

Abstract..... 2

Introduction..... 2

Background Research..... 3

System Design 3

Circuit Design 4

Flow Chart 6

Pseudocode 7

Software Listings 11

Simulation – Screen grabs 12

Testing, Simulated & Results 14

Conclusion 14

References..... 16

Abstract

In this assignment we were required to design and implement a traffic light control system using a PIC16F877A microcontroller in Proteus software. The system will control the traffic lights and countdown timers to enhance traffic flow and pedestrian safety at a T-junction. A complete simulation of the system will demonstrate its functionality. The rest of this assignment will show in detail each task briefly described above.

Introduction

This project focuses on managing traffic at T-junctions and its importance to ensure smooth vehicle flow and safety for pedestrians. The project also focuses on designing a simple traffic light control system using basic electronic components like LEDs, 7-segment displays, and a microcontroller.

The system uses three sets of traffic lights to control vehicles and includes countdown timers to show the time left for each signal. It runs on a 40 second cycle, with 10 seconds for each traffic phase and a dedicated 10 seconds for pedestrians to cross safely. A microcontroller is programmed to control the lights and the countdown displays.

This project shows how electronics can be used to solve real-world problems. The result are reliable and ensure an efficient system that improves traffic flow and keeps pedestrians safe, making it useful for managing traffic at T-junctions.

In this design, LED lights (Red, Yellow and Green) will be used to represent traffic signals, whilst the 7-segment display will show a countdown timer, this will then allow both pedestrians and drivers know how much time is left before the lights change. A microcontroller will be used for both timing's and countdown display's.



Background Research

Traffic light systems are important for managing the flow of cars and pedestrians safely and efficiently at junctions. These systems are especially helpful in organising traffic to prevent accidents and reduce waiting times for both pedestrians and road users.

Traffic lights have evolved significantly over time, from manually controlled gas lamps in the 19th century to automated electric systems in the 20th century. Modern  2 c control systems incorporate advanced technologies such as microcontrollers, timers, and sensors to control traffic flow[1][2].

LEDs are widely used in these systems due to their high brightness, low power consumption, and durability.

Another key component would be the 7-segment displays, providing countdown timers that improve reliability for drivers and pedestrians. Advantages of these include predictability for drivers reducing last-minute decisions, enhancing pedestrian safety by clearly indicating crossing times, reducing stress and potential conflicts at junctions[3].

Microcontrollers acts as the brain of the system, using programmed code to control signal timings and sequence.

Pedestrian safety is a critical aspect of traffic management, and dedicated crossing phases have been proven to reduce accidents. These technologies and principles form the foundation for this project.

Importance of Traffic Control systems –

- Smooth vehicle flow by prioritising each road direction at one time.
- Reduced risk of accidents by providing clear visual cues.
- Improved pedestrian safety with dedicated crossing phases.



System Design

The proposed system in this project is designed for a T-junction, where three roads meet and are controlled by a traffic light system. It includes three sets of traffic lights, each with red, yellow, and green LEDs, positioned at the junction of each main road. There are also pedestrian crossing signals, with red and green LEDs which are also integrated into the system. 2 no. of 7-segment countdown displays are synchronised with the traffic lights to show the remaining time for each lighting sequence. The system operates on a 40 second cycle, allocating 10 seconds each for the main roads, followed by a dedicated 10 second pedestrian crossing phase, it also includes a 3 second transition period for each traffic light to change colour as shown in the code and 7-segment display. A microcontroller manages the timing and synchronises all of the components, ensuring seamless operation of the system.

Circuit Design

The circuit design aims to control the traffic lights, countdown timers, and pedestrian crossing lights at a T-junction. The system is built around a PIC16F877A microcontroller, which manages the signals and updates the countdown timers.

Components used in Proteus –

- **Microcontroller – PIC16F877A**
 - This controls the whole system.
- **LED's – Traffic signals for both drivers and pedestrians**
 - Red, Yellow, and Green LEDs for each road (9 in total for three directions).
 - Pedestrian crossing LEDs: Red and Green (3 lots in total).
- **7-segment Displays**
 - Used for the count down timers (2 displays per road, 6 in total).
- **Buzzers (Optional)**
 - Adding a buzzer to the traffic light control system enhances pedestrian safety by providing an audible alert during the pedestrian crossing phase (1 per pedestrian crossing/road).
- **Power Supply**
 - Power supply for the microcontroller and LED's.

All components are connected and simulated in Proteus. This ensures traffic lights switch correctly every 10 seconds, countdown timers operate accurately, and the buzzer beeps during the pedestrian crossing phase only.

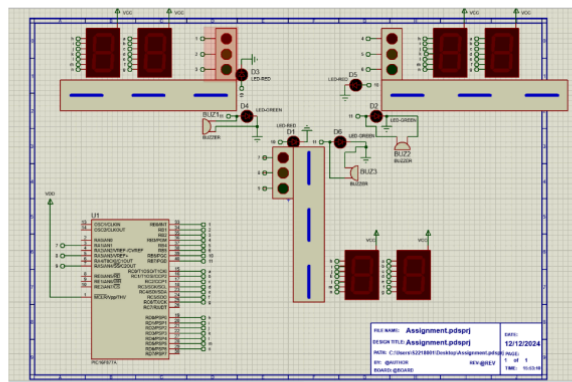


Fig.1

As you can see from Fig.1 a layout on Proteus. This gives a clear understanding of where each LED/display is connected to PIC16F877A microcontroller.

RB0 – RB5 – These control traffic light's 1 & 2 (left & right hand side).
 RA1, RA3 & RA5 – These control traffic light 3 (Middle).
 RC0 – RC7 & RD0 – RD7 – These control all 6 no. of 7-segment displays for the count down.
 RB6 & RB7 – These control all 3 pedestrians red/green LEDs.

I have connected all three buzzers to the green pedestrian crossing LEDs, ensuring that each buzzer is activated only when the corresponding green LED is illuminated. This setup guarantees that the buzzers will only sound during the green signal, providing an audible warning to pedestrians that it is safe to cross. This approach enhances safety by synchronising the auditory signal with the visual "go/cross" signal from the green pedestrian light. This can be shown in Fig.2.

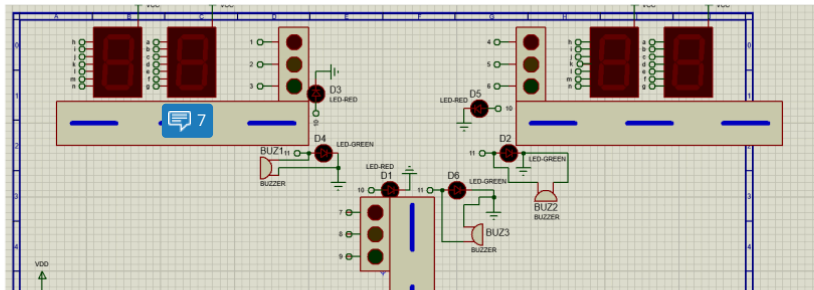


Fig.2

This design enhances pedestrian safety by offering both visual and auditory at the same time. This ensures that individuals, particularly those with visual impairments, can easily identify when it is safe to cross. The buzzer provides a clear, synchronised cue with the green pedestrian light, reinforcing the message of the "walk" signal and reducing the likelihood of accidents. Additionally, this approach improves accessibility, as it helps pedestrians navigate the crossing phase with confidence, even in noisy environments. The integration of these auditory signals also contributes to more efficient traffic management, reducing hesitation at cross roads and allows a smoother flow of both pedestrian and vehicle traffic. Overall, this design increases safety, accessibility, and traffic efficiency at the junction.

Flow Chart

The flowchart begins with the initialisation of the system. This is where all traffic and pedestrian signals are set to red, ensuring a safe starting condition. The system then enters a loop, continuously checking a timer to determine the sequence of the traffic lights. The cycle starts with traffic light 1 (LHS) green light being activated for 10 seconds, allowing vehicles to move while the pedestrian light remains red and the buzzer is off. After 10 seconds traffic light 1 (LHS) will go from green to yellow for 3 seconds, signalling that the light will soon turn red. The system then switches traffic light 2 (RHS) green light for 10 seconds while the pedestrian light remains red, followed by a yellow light for 3 seconds, then red. Next, the traffic light 3 (MIDD) green light for 10 seconds, while pedestrian light still remains red, followed again by a yellow for 3 seconds, then a red. Finally the pedestrian light will turn from red to green, accompanied by the buzzer sounding to indicate that crossing is safe to cross, during this time, all traffic lights will remain red. As the cycle concludes, all traffic and pedestrian lights turn red for 1 second before the cycle resets, repeating the entire process to manage traffic flow and pedestrian safety. Refer to flow chart below, Fig.3.

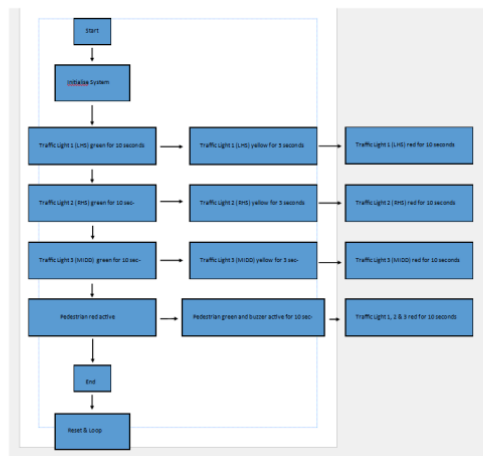


Fig.3

Pseudocode

START

//Initialise system

```
Set traffic light 1_green = FALSE
Set traffic light 1_yellow = FALSE
Set traffic light 1_red = TRUE
Set traffic light 2_green = FALSE
Set traffic light 2_yellow = FALSE
Set traffic light 2_red = TRUE
Set traffic light 3_green = FALSE
Set traffic light 3_yellow = FALSE
Set traffic light 3_red = TRUE
Set pedestrian_red = TRUE
Set pedestrian_green = FALSE
Buzzer = FALSE
```

//Start cycle

//TL1

```
Set traffic light 1_green = TRUE
Set traffic light 1_yellow = FALSE
Set traffic light 1_red = FALSE
Set traffic light 2_green = FALSE
Set traffic light 2_yellow = FALSE
Set traffic light 2_red = TRUE
Set traffic light 3_green = FALSE
Set traffic light 3_yellow = FALSE
Set traffic light 3_red = TRUE
Set pedestrian_red = TRUE
Set pedestrian_green = FALSE
Buzzer = FALSE
```

//Change over

```
Set traffic light 1_green = FALSE
Set traffic light 1_yellow = TRUE
Set traffic light 1_red = FALSE
Set traffic light 2_green = FALSE
Set traffic light 2_yellow = TRUE
Set traffic light 2_red = FALSE
Set traffic light 3_green = FALSE
Set traffic light 3_yellow = FALSE
Set traffic light 3_red = TRUE
Set pedestrian_red = TRUE
Set pedestrian_green = FALSE
Buzzer = FALSE
```

//TL2

```
Set traffic light 1_green = FALSE
Set traffic light 1_yellow = FALSE
Set traffic light 1_red = TRUE
Set traffic light 2_green = TRUE
Set traffic light 2_yellow = FALSE
Set traffic light 2_red = FALSE
Set traffic light 3_green = FALSE
Set traffic light 3_yellow = FALSE
Set traffic light 3_red = TRUE
Set pedestrian_red = TRUE
Set pedestrian_green = FALSE
Buzzer = FALSE
```

//Change over

```
Set traffic light 1_green = FALSE
Set traffic light 1_yellow = FALSE
Set traffic light 1_red = TRUE
Set traffic light 2_green = FALSE
Set traffic light 2_yellow = TRUE
Set traffic light 2_red = FALSE
Set traffic light 3_green = FALSE
Set traffic light 3_yellow = TRUE
Set traffic light 3_red = FALSE
Set pedestrian_red = TRUE
Set pedestrian_green = FALSE
Buzzer = FALSE
```

//TL3

```
Set traffic light 1_green = FALSE
Set traffic light 1_yellow = FALSE
Set traffic light 1_red = TRUE
Set traffic light 2_green = FALSE
Set traffic light 2_yellow = FALSE
Set traffic light 2_red = TRUE
Set traffic light 3_green = TRUE
Set traffic light 3_yellow = FALSE
Set traffic light 3_red = FALSE
Set pedestrian_red = TRUE
Set pedestrian_green = FALSE
Buzzer = FALSE
```

```
//Change over
Set traffic light 1_green = FALSE
Set traffic light 1_yellow = FALSE
Set traffic light 1_red = TRUE
Set traffic light 2_green = FALSE
Set traffic light 2_yellow = FALSE
Set traffic light 2_red = TRUE
Set traffic light 3_green = FALSE
Set traffic light 3_yellow = TRUE
Set traffic light 3_red = FALSE
Set pedestrian_red = TRUE
Set pedestrian_green = FALSE
Buzzer = FALSE
```

```
//Pedestrians
Set traffic light 1_green = FALSE
Set traffic light 1_yellow = FALSE
Set traffic light 1_red = TRUE
Set traffic light 2_green = FALSE
Set traffic light 2_yellow = FALSE
Set traffic light 2_red = TRUE
Set traffic light 3_green = FALSE
Set traffic light 3_yellow = FALSE
Set traffic light 3_red = TRUE
Set pedestrian_red = FALSE
Set pedestrian_green = TRUE
Buzzer = TRUE
```

```
//End of cycle
Set traffic light 1_green = FALSE
Set traffic light 1_yellow = FALSE
Set traffic light 1_red = TRUE
Set traffic light 2_green = FALSE
Set traffic light 2_yellow = FALSE
Set traffic light 2_red = TRUE
Set traffic light 3_green = FALSE
Set traffic light 3_yellow = FALSE
Set traffic light 3_red = TRUE
Set pedestrian_red = TRUE
Set pedestrian_green = FALSE
Buzzer = FALSE
Delay
```

//Reset

End

The pseudocode begins by initialising the system, where all traffic and pedestrian signals are set to red, ensuring a safe starting condition. The cycle then starts with Traffic Light 1 (LHS) turning green for 10 seconds, allowing vehicles to move while the pedestrian signal remains red, and the buzzer is off. After 10 seconds, Traffic Light 1 (LHS) switches to yellow for 3 seconds, signalling vehicles to slow down as it prepares to turn red. Following this, Traffic Light 2 (RHS) turns green for 10 seconds, while Traffic Lights 1 (LHS) and Traffic Light 3 (MIDD) are both on red. Once the 10-second duration is over, Traffic Light 2 (RHS) turns yellow for 3 seconds, followed by red, while the pedestrian light remains red. Then, Traffic Light 3 (MIDD) turns green for 10 seconds. After 10 seconds, the middle road light switches to yellow for 3 seconds, signalling the end of the green light. Once the cycle completes, all three traffic lights turn red, which triggers the pedestrian light to turn green. This also activates the buzzer, signalling to pedestrians that it is safe to cross. At the end of the cycle, all signals turn red, and the system resets, preparing to start the cycle again. This structure ensures efficient traffic management and pedestrian safety, as the system continuously cycles through the phases to maintain smooth and safe traffic flow at the T-junction.

Software Listings

```

1 //include <avr/io.h>
2 #define XTALCLOCK 1000000
3 #define F_CPU XTALCLOCK
4 #define LED_RED_TL1 PDR_B0
5 #define LED_GREEN_TL1 PDR_B1
6 #define LED_YELLOW_TL1 PDR_B2
7 #define LED_RED_TL2 PDR_B3
8 #define LED_GREEN_TL2 PDR_B4
9 #define LED_YELLOW_TL2 PDR_B5
10 #define LED_RED_TL3 PDR_B6
11 #define LED_GREEN_TL3 PDR_B7
12 #define LED_YELLOW_TL3 PDR_B8
13 #define LED_RED_PED PDR_B9
14 #define LED_GREEN_PED PDR_B10
15 #define LED_YELLOW_PED PDR_B11
16 //segment display patterns for numbers 0-9
17 unsigned int display[10] = {0x00, 0x01, 0x02, 0x03, 0x04, 0x05, 0x06, 0x07, 0x08, 0x09};
18
19 void main() {
20     int i;
21     // Set PORTA as outputs
22     output_a(0x00);
23     // Set PORTB as outputs
24     output_b(0x00);
25     // Set PORTC as outputs
26     output_c(0x00);
27     // Set PORTD as outputs
28     output_d(0x00);
29     // Initialize PORTB to 0
30     output_b(0x00);
31     // Initialize PORTC to 0
32     output_c(0x00);
33     // Initialize PORTD to 0
34     output_d(0x00);
35     while (1) {
36         for (i = 0; i < 10; i++) {
37             // Update 7-segment display
38             output_c(display[i]);
39             // Turn on T1 red LED
40             output_high(LED_RED_TL1);
41             // Turn on T2 red LED
42             output_high(LED_RED_TL2);
43             // Turn on T3 red LED
44             output_high(LED_RED_TL3);
45             // Turn off T1 yellow LED
46             output_low(LED_YELLOW_TL1);
47             // Turn off T2 yellow LED
48             output_low(LED_YELLOW_TL2);
49             // Turn off T3 yellow LED
50             output_low(LED_YELLOW_TL3);
51             // Turn on T1 green LED
52             output_high(LED_GREEN_TL1);
53             // Turn on T2 green LED
54             output_high(LED_GREEN_TL2);
55             // Turn on T3 green LED
56             output_high(LED_GREEN_TL3);
57             // Turn on pedestrian red LED
58             output_high(LED_RED_PED);
59             // Turn on pedestrian green LED
60             output_low(LED_GREEN_PED);
61             delay_ms(1000);
62         }
63     }
64 }

```

Fig.4

```

65 //include <avr/io.h>
66 #define XTALCLOCK 1000000
67 #define F_CPU XTALCLOCK
68 #define LED_RED_TL1 PDR_B0
69 #define LED_GREEN_TL1 PDR_B1
70 #define LED_YELLOW_TL1 PDR_B2
71 #define LED_RED_TL2 PDR_B3
72 #define LED_GREEN_TL2 PDR_B4
73 #define LED_YELLOW_TL2 PDR_B5
74 #define LED_RED_TL3 PDR_B6
75 #define LED_GREEN_TL3 PDR_B7
76 #define LED_YELLOW_TL3 PDR_B8
77 #define LED_RED_PED PDR_B9
78 #define LED_GREEN_PED PDR_B10
79 #define LED_YELLOW_PED PDR_B11
80 //segment display patterns for numbers 0-9
81 unsigned int display[10] = {0x00, 0x01, 0x02, 0x03, 0x04, 0x05, 0x06, 0x07, 0x08, 0x09};
82
83 void main() {
84     int i;
85     // Set PORTA as outputs
86     output_a(0x00);
87     // Set PORTB as outputs
88     output_b(0x00);
89     // Set PORTC as outputs
90     output_c(0x00);
91     // Set PORTD as outputs
92     output_d(0x00);
93     // Initialize PORTB to 0
94     output_b(0x00);
95     // Initialize PORTC to 0
96     output_c(0x00);
97     // Initialize PORTD to 0
98     output_d(0x00);
99     while (1) {
100         for (i = 0; i < 10; i++) {
101             // Update 7-segment display
102             output_c(display[i]);
103             // Turn on T1 red LED
104             output_high(LED_RED_TL1);
105             // Turn on T2 red LED
106             output_high(LED_RED_TL2);
107             // Turn on T3 red LED
108             output_high(LED_RED_TL3);
109             // Turn off T1 yellow LED
110             output_low(LED_YELLOW_TL1);
111             // Turn off T2 yellow LED
112             output_low(LED_YELLOW_TL2);
113             // Turn off T3 yellow LED
114             output_low(LED_YELLOW_TL3);
115             // Turn on T1 green LED
116             output_high(LED_GREEN_TL1);
117             // Turn on T2 green LED
118             output_high(LED_GREEN_TL2);
119             // Turn on T3 green LED
120             output_high(LED_GREEN_TL3);
121             // Turn on pedestrian red LED
122             output_high(LED_RED_PED);
123             // Turn on pedestrian green LED
124             output_low(LED_GREEN_PED);
125             delay_ms(1000);
126         }
127     }
128 }

```

Fig.6

```

101 //include <avr/io.h>
102 #define XTALCLOCK 1000000
103 #define F_CPU XTALCLOCK
104 #define LED_RED_TL1 PDR_B0
105 #define LED_GREEN_TL1 PDR_B1
106 #define LED_YELLOW_TL1 PDR_B2
107 #define LED_RED_TL2 PDR_B3
108 #define LED_GREEN_TL2 PDR_B4
109 #define LED_YELLOW_TL2 PDR_B5
110 #define LED_RED_TL3 PDR_B6
111 #define LED_GREEN_TL3 PDR_B7
112 #define LED_YELLOW_TL3 PDR_B8
113 #define LED_RED_PED PDR_B9
114 #define LED_GREEN_PED PDR_B10
115 #define LED_YELLOW_PED PDR_B11
116 //segment display patterns for numbers 0-9
117 unsigned int display[10] = {0x00, 0x01, 0x02, 0x03, 0x04, 0x05, 0x06, 0x07, 0x08, 0x09};
118
119 void main() {
120     int i;
121     // Set PORTA as outputs
122     output_a(0x00);
123     // Set PORTB as outputs
124     output_b(0x00);
125     // Set PORTC as outputs
126     output_c(0x00);
127     // Set PORTD as outputs
128     output_d(0x00);
129     // Initialize PORTB to 0
130     output_b(0x00);
131     // Initialize PORTC to 0
132     output_c(0x00);
133     // Initialize PORTD to 0
134     output_d(0x00);
135     while (1) {
136         for (i = 0; i < 10; i++) {
137             // Update 7-segment display
138             output_c(display[i]);
139             // Turn on T1 red LED
140             output_high(LED_RED_TL1);
141             // Turn on T2 red LED
142             output_high(LED_RED_TL2);
143             // Turn on T3 red LED
144             output_high(LED_RED_TL3);
145             // Turn off T1 yellow LED
146             output_low(LED_YELLOW_TL1);
147             // Turn off T2 yellow LED
148             output_low(LED_YELLOW_TL2);
149             // Turn off T3 yellow LED
150             output_low(LED_YELLOW_TL3);
151             // Turn on T1 green LED
152             output_high(LED_GREEN_TL1);
153             // Turn on T2 green LED
154             output_high(LED_GREEN_TL2);
155             // Turn on T3 green LED
156             output_high(LED_GREEN_TL3);
157             // Turn on pedestrian red LED
158             output_high(LED_RED_PED);
159             // Turn on pedestrian green LED
160             output_low(LED_GREEN_PED);
161             delay_ms(1000);
162         }
163     }
164 }

```

Fig.5

```

165 //include <avr/io.h>
166 #define XTALCLOCK 1000000
167 #define F_CPU XTALCLOCK
168 #define LED_RED_TL1 PDR_B0
169 #define LED_GREEN_TL1 PDR_B1
170 #define LED_YELLOW_TL1 PDR_B2
171 #define LED_RED_TL2 PDR_B3
172 #define LED_GREEN_TL2 PDR_B4
173 #define LED_YELLOW_TL2 PDR_B5
174 #define LED_RED_TL3 PDR_B6
175 #define LED_GREEN_TL3 PDR_B7
176 #define LED_YELLOW_TL3 PDR_B8
177 #define LED_RED_PED PDR_B9
178 #define LED_GREEN_PED PDR_B10
179 #define LED_YELLOW_PED PDR_B11
180 //segment display patterns for numbers 0-9
181 unsigned int display[10] = {0x00, 0x01, 0x02, 0x03, 0x04, 0x05, 0x06, 0x07, 0x08, 0x09};
182
183 void main() {
184     int i;
185     // Set PORTA as outputs
186     output_a(0x00);
187     // Set PORTB as outputs
188     output_b(0x00);
189     // Set PORTC as outputs
190     output_c(0x00);
191     // Set PORTD as outputs
192     output_d(0x00);
193     // Initialize PORTB to 0
194     output_b(0x00);
195     // Initialize PORTC to 0
196     output_c(0x00);
197     // Initialize PORTD to 0
198     output_d(0x00);
199     while (1) {
200         for (i = 0; i < 10; i++) {
201             // Update 7-segment display
202             output_c(display[i]);
203             // Turn on T1 red LED
204             output_high(LED_RED_TL1);
205             // Turn on T2 red LED
206             output_high(LED_RED_TL2);
207             // Turn on T3 red LED
208             output_high(LED_RED_TL3);
209             // Turn off T1 yellow LED
210             output_low(LED_YELLOW_TL1);
211             // Turn off T2 yellow LED
212             output_low(LED_YELLOW_TL2);
213             // Turn off T3 yellow LED
214             output_low(LED_YELLOW_TL3);
215             // Turn on T1 green LED
216             output_high(LED_GREEN_TL1);
217             // Turn on T2 green LED
218             output_high(LED_GREEN_TL2);
219             // Turn on T3 green LED
220             output_high(LED_GREEN_TL3);
221             // Turn on pedestrian red LED
222             output_high(LED_RED_PED);
223             // Turn on pedestrian green LED
224             output_low(LED_GREEN_PED);
225             delay_ms(1000);
226         }
227     }
228 }

```

Fig.7

The software listings for the T-junction traffic light control system consist of the complete code written above Fig.4,5,6 & 7. This code manages the operation of the traffic lights, pedestrian signals, and buzzer. The code is designed to control three traffic lights (left-hand side, right-hand side, and middle road) using a microcontroller PIC16F877A. It ensures that each traffic light cycles through green, yellow, and red with appropriate timing intervals set (e.g., 10 seconds for green, 3 seconds for yellow). The software also manages the pedestrian signals, turning on both pedestrian green light and buzzer for safe crossing. After completing a cycle, the system resets and continues to control the lights in the loop. These software listings include functions for initialising the system, controlling the timings of all signal changes and ensuring pedestrian safety by activating the buzzer during the pedestrian crossing phase. The code is written in C and compiled using PCW before being uploaded to the microcontroller, which governs the entire simulation and operation of the traffic light system.

Simulation – Screen grabs

The overall screen grab of the simulation provides a comprehensive view of the T-junction traffic light control system. It displays the complete circuit schematic, showcasing the interconnected components, including the PIC16F877A microcontroller, the traffic light LEDs for the left-hand side (LHS), right-hand side (RHS), and middle road (MIDD), as well as the pedestrian signals and buzzer.

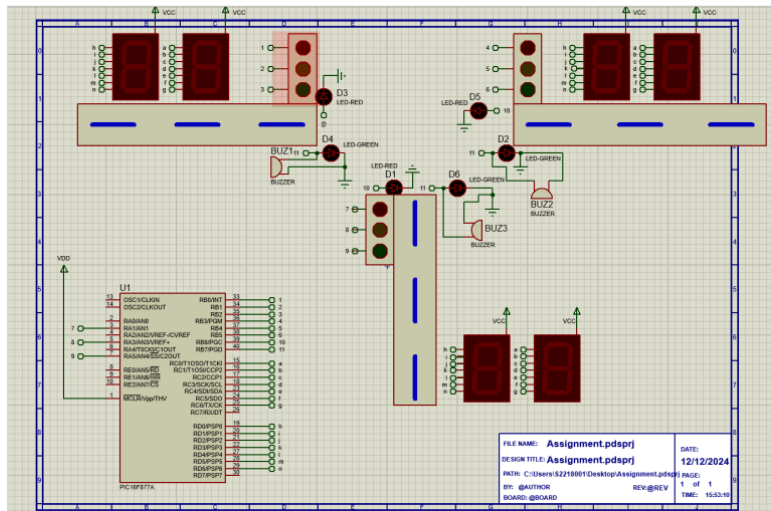


Fig.8

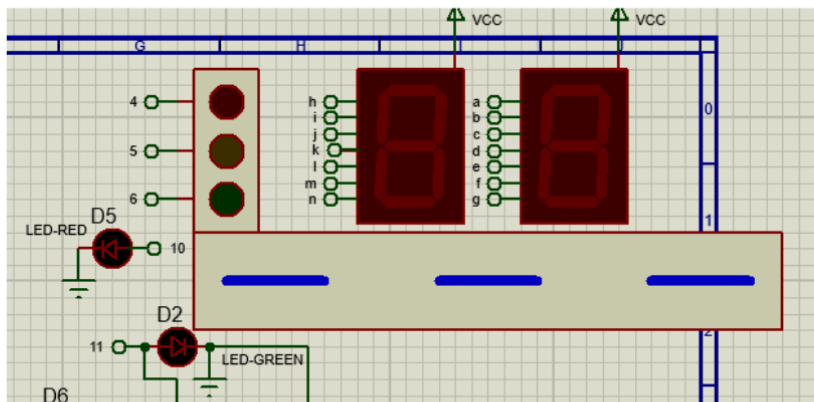


Fig.9

Fig.9 shows the complete setup of a single traffic light configuration. This includes the 7-segment display, which is programmed to show a countdown timer indicating the remaining time for the current phase. The setup also features traffic light LEDs (green, yellow, and red) that control vehicle movement, transitioning through their sequences to regulate traffic efficiently. Additionally, the configuration incorporates pedestrian LEDs (red and green), which manage pedestrian crossing activities by signalling when it is safe or unsafe to cross the road. A buzzer is also included as an auditory alert, which activates during the pedestrian green phase to enhance safety by notifying pedestrians that it is safe to cross. This setup represents one part of the overall system and demonstrates how the components are integrated to manage a single traffic lane while maintaining coordination with the rest of the T-junction.

I have included the Proteus simulation file along with all relevant documents to view the traffic light control system's operation over its complete 40 second cycle. The simulation will show the full system, highlighting the changes in the traffic light signals, pedestrian signals, and buzzer activation throughout each sequence. This includes the green, yellow, and red transitions for the three traffic lights (LHS, RHS, and middle road), as well as the dedicated pedestrian crossing phase where the green pedestrian light and buzzer are activated for safety. This gives a better understanding of how the system runs and ensures everything works as intended.

[Assignment - Traffic Lights.zip](#)

Testing, Simulated & Results

Testing

The traffic light control system was tested in Proteus to ensure all parts worked correctly and efficiently. The testing checked the timing of the traffic lights, the pedestrian signals, the buzzer, and the 7-segment display. The 40 second cycle was observed to ensure smooth transitions, no conflicting lights, and proper activation of the pedestrian crossing phase only when all traffic lights were red.

Simulated

Traffic Lights – All 3 traffic light (LHS, RHS, and middle) changed through green, yellow, and red as programmed. Both red and green lasting 10 seconds and yellow only 3 seconds.

Pedestrian Lights/sounder - The pedestrian green light and buzzer were only activated on the dedicated crossing sequence. During this time period, all traffic lights were red.

7-segment Display - The display counted up for 40 seconds, showing the remaining time for each phase and resetting at the start of a new cycle. It also shows the 3 second changeover period when going from green to yellow or vice versa.

Results

The results of the simulation confirmed the system worked as designed, with accurate timings and no errors in light transitions. The traffic lights transitioned smoothly, ensuring safe and efficient traffic and pedestrian management. It also showed the 7-segment display clearly counting up, and the buzzer activated correctly during the pedestrian phase.

These results validate the system's design and functionality for real-world application.

Conclusion

The T-junction traffic light control system worked successfully, meeting all the goals set out. It managed traffic smoothly with accurate timings for each traffic light and pedestrian lights and ensured pedestrian safety by activating the green pedestrian light and buzzer during the crossing phase only. The 7-segment display worked well, showing clear countdowns for each light sequence. Overall, the system is reliable and effective.

To improve the system, a change that could be made, would be the addition of a push button at each pedestrian crossing. This feature would replicate real-world traffic light systems, allowing pedestrians to signal when they are waiting to cross. By including this, the system would become more responsive, ensuring that pedestrians do not have to wait unnecessarily long for the lights to change. This addition would also enhance the overall efficiency of the system, as it could adjust the timings of the pedestrian crossing phase based on actual demand, rather than following a fixed cycle.

Another important improvement would be the inclusion of a battery backup. In the event of a power cut, the system would continue to operate without interruption, ensuring the safety of both pedestrians and vehicles. Traffic lights are important at junctions, and having a reliable power source during emergencies would prevent accidents or confusion. A backup power system, such as a battery or solar-powered solution, would ensure that the traffic signals stay functional during power cuts, maintaining safety and reducing the likelihood of traffic disruptions[4][5].

References

- [1] <https://www.inclusivecitymaker.com/1868-2019-a-brief-history-of-traffic-lights/#:~:text=December%2010%2C%201868%3A%20the%20official,lit%20semaphore%20to%20ensure%20visibility.>
- [2] <https://www.theguardian.com/notesandqueries/query/0,5753,-1460,00.html>
- [3] <https://www.sciencedirect.com/science/article/abs/pii/S1369847824001566>
- [4] <https://malque.pub/ojs/index.php/msj/article/download/5224/2658/32768>
- [5] <https://www.bbmlcd.com/a-news-eco-friendly-illumination-the-efficiency-of-traffic-lights-with-solar-panels>

49/100

PAGE 1

PAGE 2

PAGE 3



Comment 1 | Engagement with Literature(10)

The concepts of traffic light controller is well identified.

PAGE 4



Comment 2 | Knowledge & Understanding (20)

Could be better if some types of the traffic light controllers are mentioned.



Comment 3 | Engagement with Literature(10)

Good! The significance of Traffic lights are mentioned.



Comment 4 | Cognitive & Intellectual (20)

It is essential to explain how the selection of microcontroller is made.

PAGE 5



Comment 5 | Knowledge & Understanding (20)

Good. The components are explained briefly.

PAGE 6



Comment 6 | Cognitive & Intellectual (20)

The ides of having 3 buzzers along with the green LED is amazing.



Comment 7 | Practical Application Sk(30)

Good Presentation of Circuit Layout

PAGE 7

**Comment 8** | Transferable Skills (20)

The codlings are already seen in the screen shot. There is no need to include it here.

**Comment 9** | Cognitive & Intellectual (20)

Good

GRADING FORM: EC1

REBECCA COLES

48.96

ENGAGEMENT WITH LITERATURE(10)

a. Identification on Concepts of Traffic Light Controller (4);its significance (2) b. Background Research (3) c. Diversity of sources and Referencing Skills(1)



a. Information related to the traffic Light Controller is well identified and its significance were captured. 4.2

b. Good! Engagement with an appropriate range of literature, including sources retrieved independently. 1.5

c. Referencing conventions evident but limited. 0.4

6.1

KNOWLEDGE & UNDERSTANDING (20)

a. Knowledge on various products of traffic light controller and Market Requirements and Specifications(10) b. Reflect clearly the underlying principles on analysing different traffic light controllers with adequate and appropriate Justifications(10)



- a.Limited knowledge on various products/parts of traffic light Controller and critical understanding of the well-established concepts and principles within the subject area. Market Requirements and Specifications are not Mentioned. 4
- b.Good! Principle of using Traffic Lights are explained with minor gaps in the selection of material. 5

9

COGNITIVE & INTELLECTUAL (20)

a. Critical review on the selection of Microcontroller and its supporting Modules, ICs, costs(7) b. Innovative thinking in problem solving and feature additions(7) c. Pseudocode showing the system operation(6)



- a.Little or no Critical evaluation on selection of Microcontroller and its concepts. 1.4
- b.Clearly distinguishes between fact, opinion, & acknowledges value judgments are seen. 5.6
- c.Pseudocode written has some errors but works and gives minimal results. 2.88

9.86

PRACTICAL APPLICATION SK(30)

a. Circuit Design Layout(10). b. Testing and Debugging and Simulated Output Results(20)



- a. A Traffic Light Controller circuit was designed with minimal specifications 4
- b.Simulation is running but the results are not upto the satisfaction. But still works with the traffic signal and pedestrain LED. 9

13

TRANSFERABLE SKILLS (20)

a. Logical order of ideas, Flow of ideas in the report from the beginning to end with appropriate chapters, sub chapters and continuation(7)"

b. Definitions, Acronyms and Abbreviations (6) c.

Professionalism, Spelling and Grammar, Complete Report, Proper Presentation, Quality of the report(7)



a. Mostly coherent, organised work, in a suitable structure and is for the most part clearly expressed. 4

b. Most of the Definitions, Acronyms and Abbreviations are captured. 4

c. Professionalism is maintained in the complete report. Proper Presentation and the Quality of the report is appreciable. 4

11