

Prifysgol Cymru
Y Drindod Dewi Sant
University of Wales
Trinity Saint David

200
1822 - 2022

L5 – Electronics 1 (ACDB5002)

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

David Mathew Lewis

2219439

Electrical and Electronic Engineering

Table of Contents

Introduction	3
Background Research.....	3
Components.....	3
PIC16F877A Microcontroller:.....	3
7-Segment Display	4
LEDs (Light Emitting Diodes) and Traffic Lights.....	5
Circuit Design	5
Flowchart	5
Proteus Overview.....	6
1. Header files.....	11
Microcontroller Configuration	12
2. Definitions	12
3. Pin Assignment	12
1. Declaring the Main Function	13
2. Declaring a Variable.....	13
3. Configuring Ports	13
1. While Loop.....	14
2. For Loop.....	14
3. Traffic Light Behavior.....	15
Conclusion.....	16
References	16

Introduction

The goal of this project is to design a traffic light control system for a T-junction, focusing on traffic management efficiency and pedestrian safety. The system is required to manage traffic light changes based on a time-based cycle, ensuring smooth traffic flow while incorporating a dedicated pedestrian phase. The design will be implemented using the PIC16F877A microcontroller and Proteus simulation software. The system will include traffic lights, 7-segment displays for time countdown, and a pedestrian crossing phase with synchronized signal changes.

Background Research

Traffic light control systems are a crucial part of urban infrastructure, helping regulate the flow of vehicles and pedestrians at intersections. There are different ways to manage traffic signal timings, with time-based systems being one of the most widely used. A well-designed control system should prioritise safety, minimise congestion, and allow for smooth transitions between signal phases.

Traffic Signal Phases: Typically, traffic lights operate in phases such as red, green, and amber. For a T-junction, there are three roads to manage, and the signals need to change at regular intervals to maintain smooth traffic flow.

Pedestrian Safety: The inclusion of a dedicated pedestrian crossing phase ensures that pedestrians can cross the road safely. This phase must be synchronized with the traffic signal system, where the traffic lights turn red for all directions during the crossing phase.

7-Segment Display: This is used to show the countdown for each signal phase. The system needs to provide a visual indication of the remaining time for the current signal phase.

Components

PIC16F877A Microcontroller:

The PIC16F877A microcontroller is a popular choice for designing embedded systems like traffic light control systems due to its rich feature set and flexibility. Here are the reasons why it is suitable for this project, along with references to its specifications and usage:

Adequate I/O Pins

The PIC16F877A has 40 pins with 33 I/O pins, sufficient to connect multiple LEDs, 7-segment displays, and other components for a traffic light control system.

This makes it ideal for managing the three traffic lights, 7-segment displays, and pedestrian signals simultaneously.

Built-in Timers

Equipped with three 8-bit and one 16-bit timer modules, the microcontroller can handle the precise timing intervals required for traffic light transitions and pedestrian phases.

Flash Memory for Program Storage

The microcontroller has 14 KB of flash memory, enough for the control algorithms and any additional functionality required in the project.

Compatibility with Proteus and Development Tools

The PIC16F877A is widely supported in simulation tools like Proteus and coding software such as CCS Compiler. This facilitates testing and debugging during the development process.

Low Power Consumption

It has a low power requirement, making it suitable for real-world traffic systems where energy efficiency is important.

Cost-Effectiveness

Compared to other microcontrollers with similar features, the PIC16F877A is cost-effective and widely available, making it a practical choice for educational and commercial projects.

The pin layout for this microcontroller is as below, taken from the data sheet[1]:

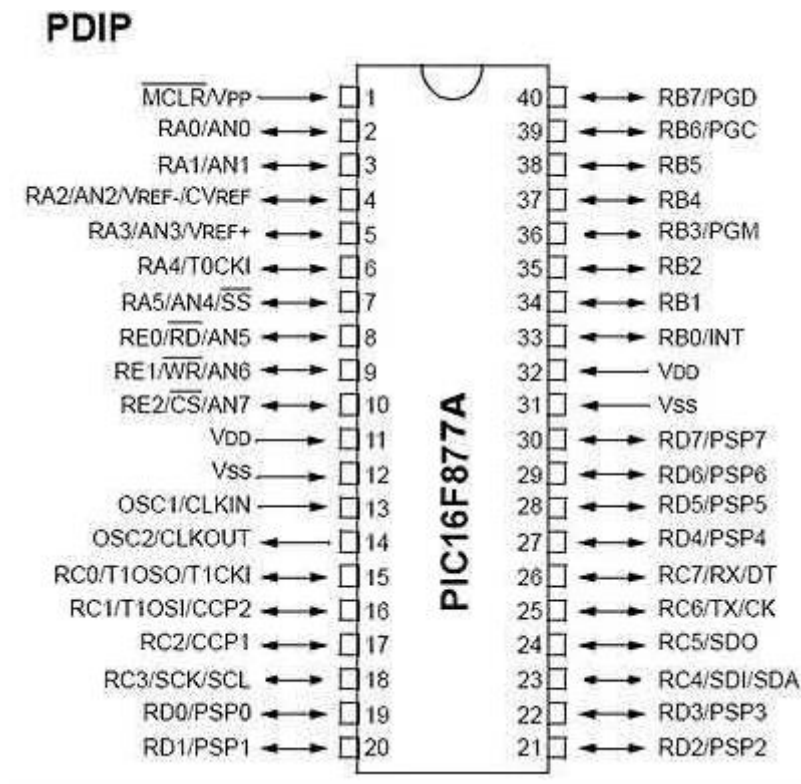


Figure 1

7-Segment Display

A 7-segment display is a popular electronic display device used to display decimal numbers and certain alphabetic characters. It is widely used in digital clocks, calculators, and electronic counters, including traffic light countdown systems.

Advantages of Using 7-Segment Displays

Simplicity: Easy to control and understand.

Cost-Effective: Inexpensive compared to LCDs or OLEDs.

Readable: Highly visible even in low-light conditions.

Limitations

Limited Characters: Can only display numbers and a few letters.

Power Consumption: Higher than LCDs due to multiple LEDs.

Applications in the Project

Use 7-segment displays to show the countdown timer for vehicles and pedestrians.

Mount one display on each side of the junction for clear visibility. Configure displays to count from 40 to 0, with the final 10 seconds dedicated to pedestrian crossing.[2]

LEDs (Light Emitting Diodes) and Traffic Lights

Each LED will be used to represent traffic signals (red, yellow, green) for vehicles and red/green for pedestrian crossing. Each LED is connected to a microcontroller pin, the microcontroller controls the state of each LED (ON or OFF) based on the timing logic for traffic light phases.

Circuit Design

Flowchart

Prior to beginning to code, creating a flowchart helps improve the clarity, efficiency, and quality of the program development process. It provides a structured way to think about the problem and its solution, ultimately leading to better code.

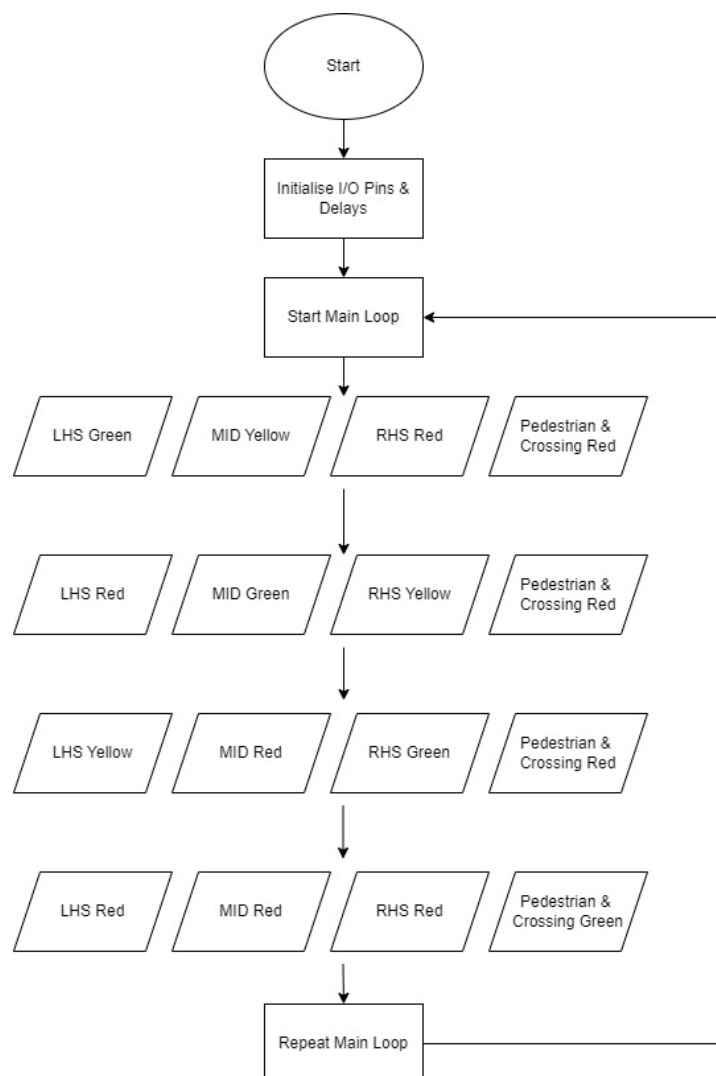


Figure 2

The flowchart in figure 2 shows the operation of an automatic traffic light system designed for a T-junction with a pedestrian crossing feature.

Initialisation: The system starts by checking the status of a push button meant for pedestrians.

Normal Operation (3-phase cycle):

The traffic lights operate in a three-phase sequence:

Phase 1: Road A has a green light, others are red.

Phase 2: Road B has a green light, others are red.

Phase 3: Road C has a green light, others are red.

Each phase runs for a set duration of 10 seconds.

Pedestrian Phase:

All traffic lights turn red.

Pedestrian crossing is activated with a green light for pedestrians and a flashing yellow light and buzzer for warning.

Reset: After the pedestrian phase, the system returns to the normal three-phase cycle

Proteus Overview

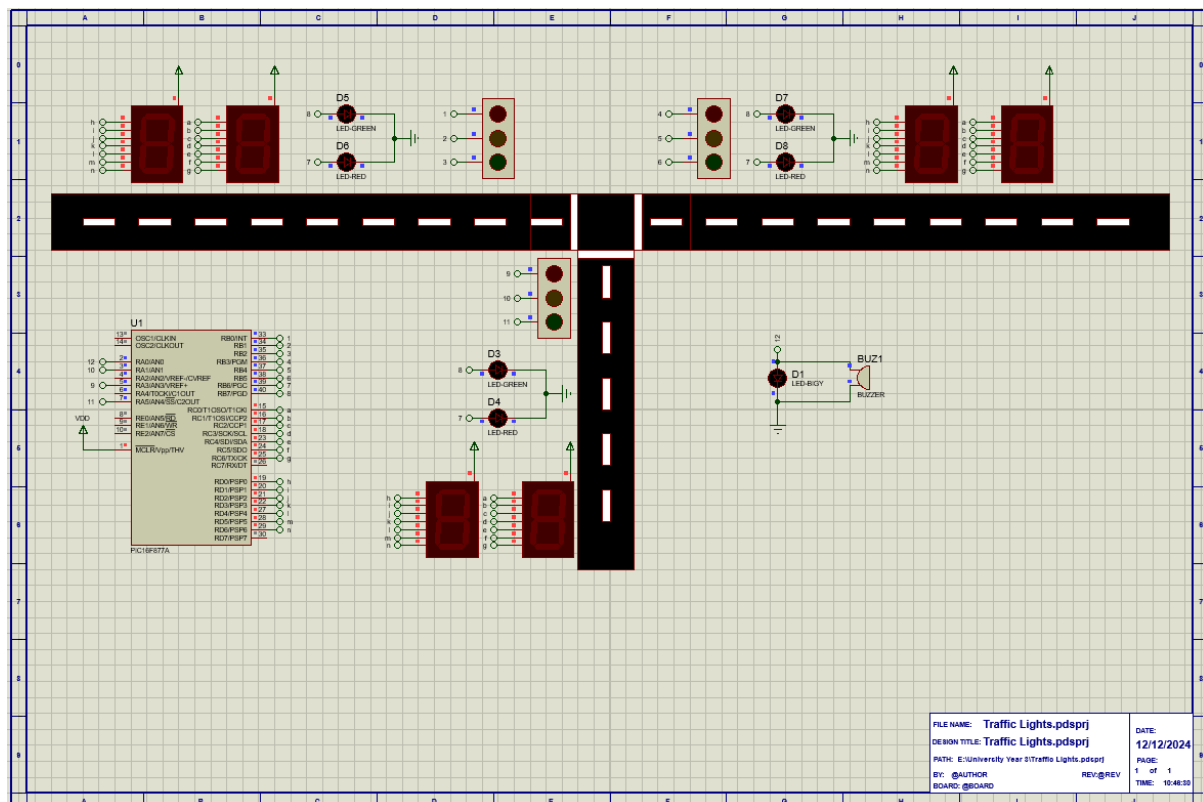


Figure 3

Figure 3 above shows the layout of the T-Junction. LHS is the left hand lane, MID is down the middle and RHS is the right hand lane. Each area has a timer, pedestrian crossing lights and traffic lights. It

also shows the pin-out of the PIC and the yellow light and buzzer on the right, which is used during pedestrian crossing.

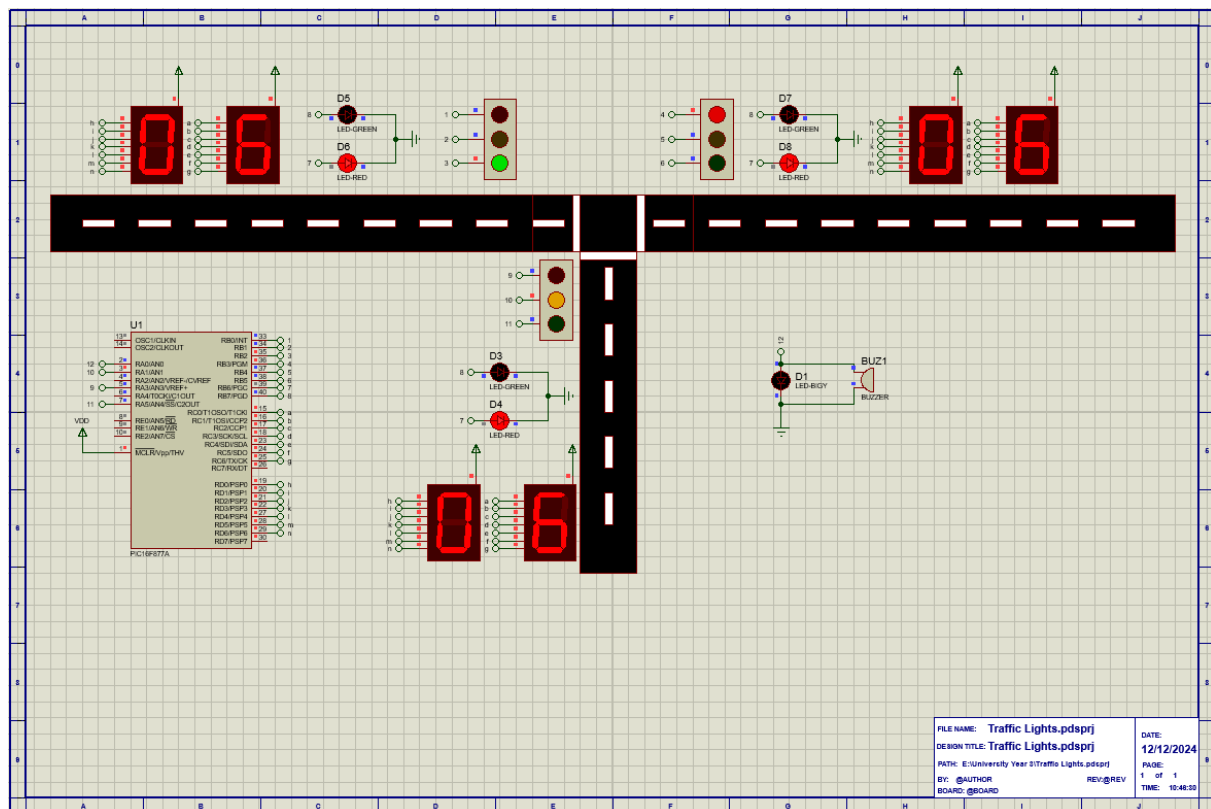


Figure 4

Figure 4 above the first 10 second phase of the cycle:

LHS traffic light green, MID yellow and RHS red. Pedestrian crossing red also.

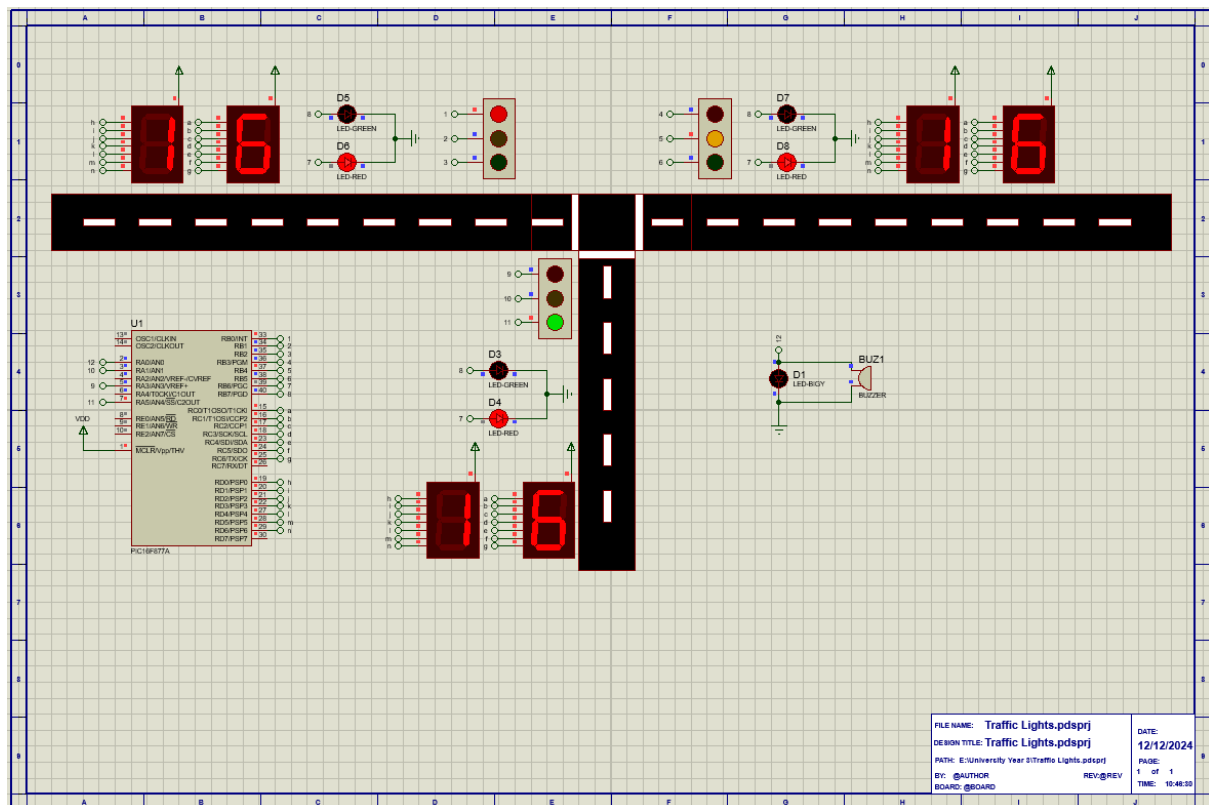


Figure 5

Figure 5 above shows the 10-20 second phase of the cycle:

LHS traffic light red, MID green and RHS yellow. Pedestrian crossing red also.

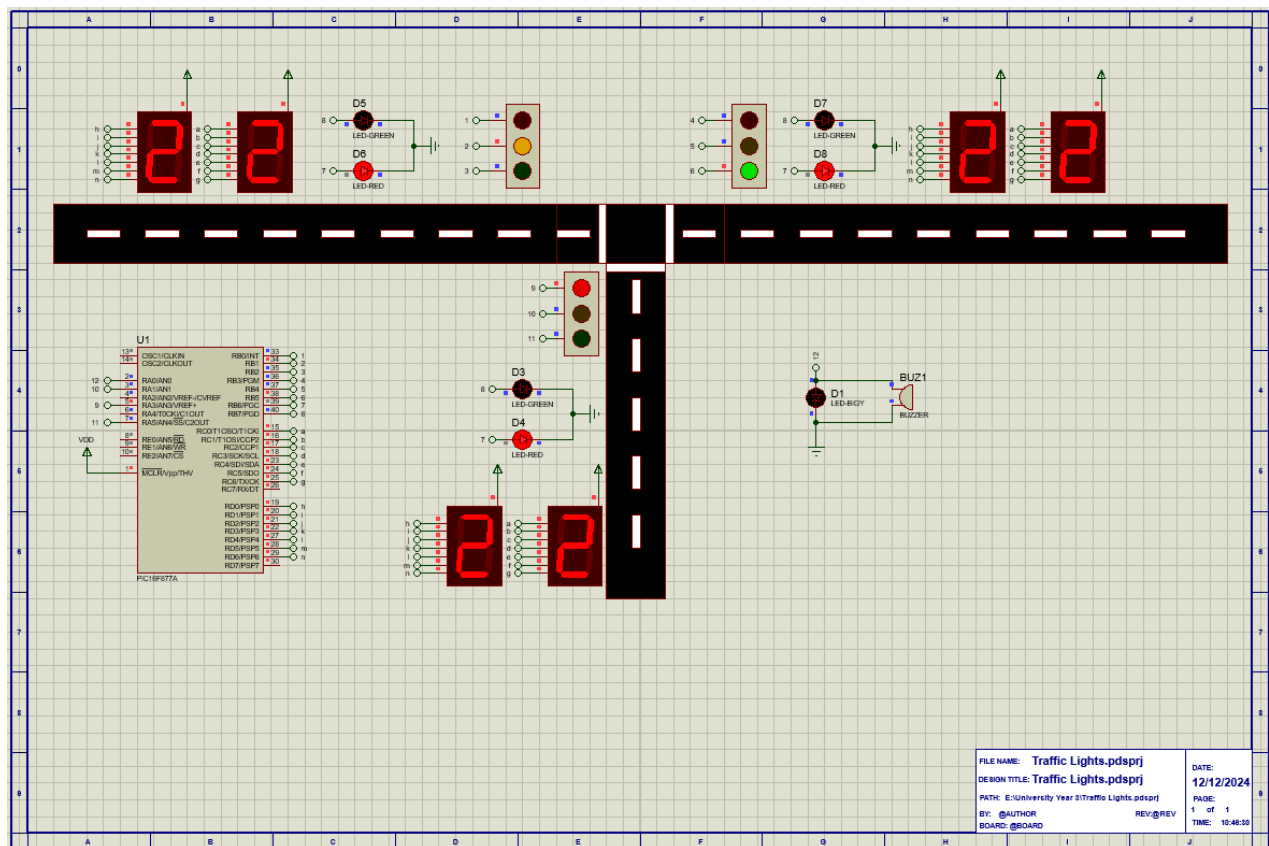


Figure 6

Figure 6 above shows the 20-30 second phase of the cycle, which is the last green traffic light cycle:

LHS traffic light yellow, MID red and RHS green. Pedestrian crossing also red.

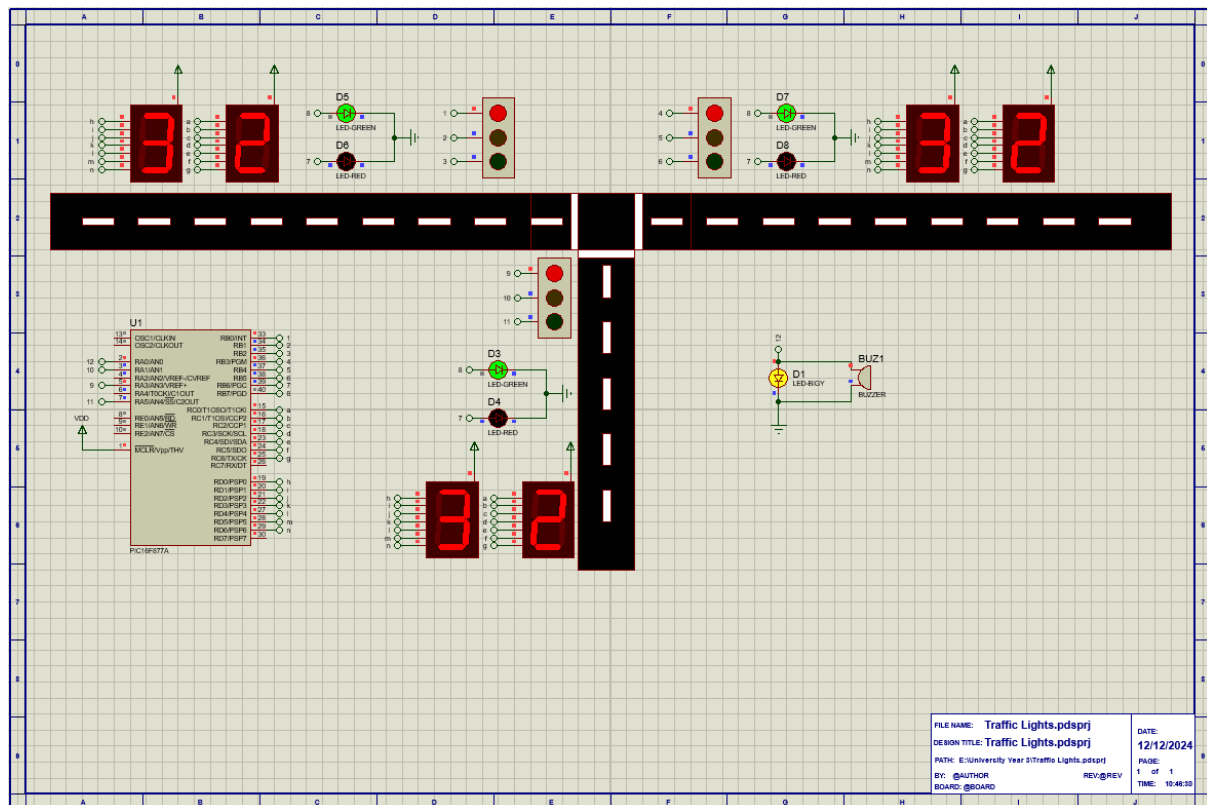


Figure 7

Figure 7 above is the final part of the cycle – pedestrian crossing phase. All traffic lights are red, the pedestrian crossing lights are green, with a flashing yellow light and buzzer as a warning signal. Although the buzzer can't be heard, using an oscilloscope on Proteus confirms that the buzzer does work.

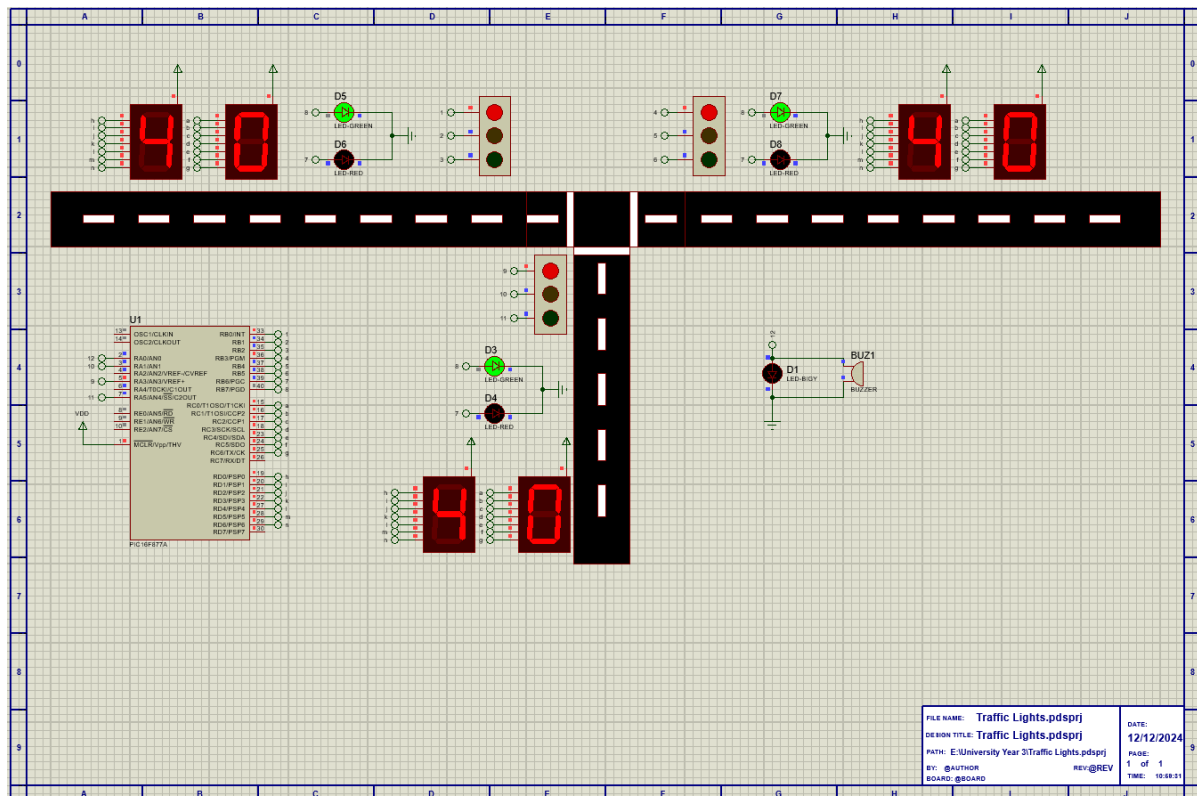


Figure 8

Figure 8 above shows the complete cycle at 40 seconds.

```

1  #include <main.h>
2  #include <16F877A.h>
3  #fuses XT,NowDT, NOPROTECT, NOLVP
4  #use delay(clock=2000000)
5
6  // Define Pins
7  #define LED_RED_LHS PIN_B0
8  #define LED_YELLOW_LHS PIN_B1
9  #define LED_GREEN_LHS PIN_B2
10 #define LED_RED_RHS PIN_B3
11 #define LED_YELLOW_RHS PIN_B4
12 #define LED_GREEN_RHS PIN_B5
13 #define LED_RED_MID PIN_A3
14 #define LED_YELLOW_MID PIN_A1
15 #define LED_GREEN_MID PIN_A5
16 #define PED_RED PIN_B6
17 #define PED_GREEN PIN_B7
18 #define CROSS PIN_A0

```

Figure 9

Figure 9 above includes the required header files and microcontroller settings and definitions:

1. Header files

#include <main.h>: This declares the existence of functions that are named in the .c files so that these functions can be called in other files.

#include <16F877A.h>: Includes device-specific definitions for the PIC16F877A microcontroller, like registers, pins, and internal peripherals.

Microcontroller Configuration [3]

#fuses: This is a generally used by CCS to name it's pre-processor directive.

XT: Means an external crystal oscillator is used for the clock.

NOWDT: This will disable the watchdog timer, so after a time out, the MCU will reset.

NOPROTECT: This allows program memory to be read externally.

NOLVP: This is used to prevent accidental reprogramming of the chip.

#use delay(clock=20000000): This tells the compiler that the microcontroller is running at a 20 MHz so functions such as delay_ms() can be calibrated correctly.

2. Definitions

#define: These assign readable names to the IO pins of the microcontroller, making it easier to manage.

3. Pin Assignment

Traffic Light LEDs:

Left-Hand Side (LHS): PIN_B0, PIN_B1, PIN_B2 are used for red, yellow, and green LEDs for the left-side traffic lights.

Right-Hand Side (RHS): PIN_B3, PIN_B4, PIN_B5 are used for red, yellow, and green LEDs for the right-side traffic lights.

Middle (MID): PIN_A3, PIN_A1, PIN_A5 are used for middle traffic lights.

Pedestrian Lights:

PIN_B6 and PIN_B7 control the pedestrian traffic lights (red and green).

Cross:

PIN_A0 is used to activate the crossing light and buzzer.

```
21 //7 segment displayt digit patterns (assuming common cathode)
22 unsigned int display[10] = {0xC0, 0xF9, 0xA4, 0xB0, 0x99, 0x92, 0x82, 0xF8, 0x80, 0x90};
```

Figure 10

Figure 10 above contains hexadecimal patterns for displaying digits 0-9 on a 7-segment display. Each code represents the segments to turn on/off for a specific digit.

Digit	Hex Pattern	Binary (Segments)	Segments ON
0	0xC0	11000000	a, b, c, d, e, f
1	0xF9	11111001	b, c
2	0xA4	10100100	a, b, d, e, g
3	0xB0	10110000	a, b, c, d, g
4	0x99	10011001	b, c, f, g
5	0x92	10010010	a, c, d, f, g
6	0x82	10000010	a, c, d, e, f, g
7	0xF8	11111000	a, b, c
8	0x80	10000000	All segments
9	0x90	10010000	a, b, c, f, g

[4]

```
24 void main() {  
25     int i;  
26     set_tris_a(0x00);    //Set Port A as outputs  
27     output_a(0x00);      //Initialise Port A as output  
28     delay_ms(1000);      //1 second delay  
29     set_tris_b(0x00);    // Set Port B as outputs  
30     output_b(0x00);      // Initialise Port B to 0  
31     delay_ms(1000);      // 1 second delay  
32     set_tris_c(0x00);    //Set Port C as outputs  
33     output_c(0x00);      //Initialise Port C to 0  
34     set_tris_d(0x00);    //Set Port D as outputs  
35     output_d(0x00);      //Initialise Port D to 0
```

Figure 11

1. Declaring the Main Function

The **main()** function is where the program begins.

2. Declaring a Variable

int i; declares an integer, which is used for counting in the program.

3. Configuring Ports

set_tris_x(0x00): Sets the ports as outputs by writing 0x00 (all bits = 0) to the TRIS registers. TRIS registers control the direction of the port pins:

0: Pin is an output.

1: Pin is an input.

output_a(0x00): Sets all the output pins of Port A to 0 (low voltage level or OFF).

delay_ms(1000): Introduces a delay of 1000 milliseconds (1 second) before continuing.

```

37 while (1) {
38     // LHS Green
39     for (i = 0; i < 10; i++)
40     {
41         output_d(0xC0); // '0'
42         output_c(display[i]); // Update 7-segment display with count 0-9
43         output_low(LED_RED_LHS); // Turn on LHS green LED, middle yellow and RHS red
44         output_high(LED_RED_RHS);
45         output_low(LED_RED_MID);
46         output_low(LED_YELLOW_LHS);
47         output_low(LED_YELLOW_RHS);
48         output_high(LED_YELLOW_MID);
49         output_high(LED_GREEN_LHS);
50         output_low(LED_GREEN_RHS);
51         output_low(LED_GREEN_MID);
52         output_low(CROSS); // Turn off pedestrian LED
53         output_high(PED_RED); // Turn ON Red LED
54         // output_low(PED_GREEN);
55         delay_ms(1000); // Delay 1 second delay
56     }
57     // Mid Green
58     for (i = 0; i < 10; i++)
59     {
60         output_d(0xF9); // '1'
61         output_c(display[i]); // Update 7-segment display with count 0-9
62         output_high(LED_RED_LHS); // Turn on middle green LED, RHS yellow, LHS red
63         output_low(LED_RED_RHS);
64         output_low(LED_RED_MID);
65         output_low(LED_YELLOW_LHS);
66         output_high(LED_YELLOW_RHS);
67         output_low(LED_YELLOW_MID);
68         output_low(LED_GREEN_LHS);
69         output_low(LED_GREEN_RHS);
70         output_high(LED_GREEN_MID);
71         output_low(CROSS); // Turn off pedestrian LED
72         output_high(PED_RED); // Turn ON Red LED
73         // output_low(PED_GREEN);
74         delay_ms(1000); // Delay 1 second delay
75     }
76     // RHS Green
77     for (i = 0; i < 10; i++)
78     {
79         output_d(0xA4); // '2'
80         output_c(display[i]); // Update 7-segment display with count 0-9
81         output_low(LED_RED_LHS); // Turn RHS green LED, yellow LHS and red middle
82         output_low(LED_RED_RHS);
83         output_high(LED_RED_MID);
84         output_high(LED_YELLOW_LHS);
85         output_low(LED_YELLOW_RHS);
86         output_low(LED_YELLOW_MID);
87         output_low(LED_GREEN_LHS);
88         output_high(LED_GREEN_RHS);
89         output_low(LED_GREEN_MID);
90         output_low(CROSS); // Turn off pedestrian LED
91         output_high(PED_RED); // Turn ON Red LED
92         // output_low(PED_GREEN);
93         delay_ms(1000); // Delay 1 second delay
94     }

```

Figure 12

Here's an explanation of Figure 12:

1. While Loop

The **while (1)** creates an infinite loop, which ensures the program continuously runs the code inside.

2. For Loop

The **for** loop counts from $i = 0$ to $i = 9$ which displays the value of i on the 7-segment display.

3. Traffic Light Behavior

Left-Hand Side (LHS):

Green LED is ON (output_high(LED_GREEN_LHS)).

Red and yellow LEDs are OFF.

Right-Hand Side (RHS):

Red LED is ON (output_high(LED_RED_RHS)).

Green and yellow LEDs are OFF.

Middle Traffic Lights:

Yellow LED is ON (output_high(LED_YELLOW_MID)).

Red and green LEDs are OFF.

Pedestrian Crossing:

Red pedestrian LED is ON (output_high(PED_RED)), indicating "Don't Cross."

Green pedestrian LED and the crossing signal are OFF.

```

96 // FOR PEDESTRIAN CROSSING
97   for (i = 0; i < 10; i++)
98   {
99       output_d(0x80); //'3'
100      output_c(display[i]);           // Update 7-segment display with count 0-9
101      output_high(LED_RED_LHS);       // TURN ON RED LEDS
102      output_high(LED_RED_RHS);
103      output_high(LED_RED_MID);
104      output_low (LED_YELLOW_LHS);
105      output_low (LED_YELLOW_RHS);
106      output_low (LED_YELLOW_MID);
107      output_low (LED_GREEN_LHS);
108      output_low (LED_GREEN_RHS);
109      output_low (LED_GREEN_MID);
110      output_toggle(CROSS);           //Turn off pedestrian LED
111      output_low(PED_RED);            //Turn ON Red LED
112      output_high(PED_GREEN);         //Go
113      delay_ms(1000);                 //Delay 1 second delay
114   }
115   for (i = 0; i < 1; i++)
116   {
117       output_d(0x99);                //'4'
118       output_c(display[i]);           // Update 7-segment display with count
119       delay_ms(1000);                 //1 second delay
120   }
121 }
122 }
```

Figure 13

Figure 13 represents the Pedestrian crossing phase.

Conclusion

The design and implementation of the automatic traffic light control system for a T-junction was successfully completed as planned. The project achieved the objective, demonstrating a functional three-phase traffic light system with an additional pedestrian crossing phase.

During the process, 1 output for the MID red traffic light did not operate. Upon further research this was because a digital output couldn't be used on that particular pin, the pin was used for analogue signals only.

A few additional advancements that would be beneficial to the system could be:

- A pedestrian crossing push-button. The system would not need to enter the pedestrian cycle if there are no pedestrians waiting to cross. Vice versa, the push-button could be used as an override to enter the pedestrian cycle quicker, especially if there is less traffic.
- An upgraded yellow light for the traffic lights. Commonly used in the real world, the yellow 'get-ready' light appears seconds before the green light. With this system, the yellow light stays on at each lane for 10 seconds, realistically this is too long as the drivers would be getting ready to drive.
- A induction loop could be used to detect the traffic, therefore, instead of a timer, a count of the traffic for example may be used prior to switching lanes.

In conclusion, this project highlights the effectiveness of microcontroller-based systems in traffic management and pedestrian safety.

References

- [1] Microchip pinout - https://microcontrollerslab.com/pic16f877a-introduction-features/#google_vignette
- [2] M. V. K. & M. Rohit, Principles of Electrical Engineering and Electronics. S. Chand Publishing, 2006.
- [3] A. Pal, Microcontrollers: Principles and Applications. PHI Learning Pvt. Ltd., 2011.
- [4] R. C. Seacord, Effective C: An Introduction to Professional C Programming. No Starch Press, 2020.