

Electronics One

Assignment One

Jack Spillett - Eastman Chemical Company

Introduction

This task involved designing a circuit, using either a PIC16F or a PIC18F, for a traffic light control system for a four-way junction, featuring: four traffic signal lights on each side of the road for comprehensive traffic management, a set of 7-segment displays on each side to count from 0 to 50 seconds, with the system to change traffic signals every 10 seconds for predictable and smooth traffic flow and the final 10 seconds of the signal cycle being reserved for a dedicated pedestrian crossing phase.

Proteus was initially used to design and test the circuit with MicroC being used to produce the program. After the design and testing on Proteus, a small amount of testing was performed using componentry built up on breadboards. To fulfil the requirements of the task, several stages needed to be completed, initially background research needed to be conducted for an understanding of traffic light systems and the fundamentals of Microcontroller programming. Afterwards the 'logic' of the system needed to be organised (using a table and flowchart), making programming the system simpler. Next, the Traffic Light System was designed and built on proteus, with programming then being started. With the final program completed and installed onto the Microcontroller within Proteus, simulations were then used to test the program. The final stage was to write the program to a Microcontroller and build the system using a breadboard, LEDs, etc. for comprehensive testing.

The purpose of assigning this task to students was to offer an opportunity to learn the fundamentals of electronics and programming, with a focus on microcontrollers, which are becoming ever more widely used in society.

Background Research

Traffic lights are used to maintain a consistent and safe flow of traffic and pedestrians. Traffic Lights use a standardised colour pattern to facilitate orderly movement and reduce the likelihood of collisions, primarily at junctions. In most cases, they provide a designated crossing time for pedestrians on busy roads for safe passage. However, while traffic lights enhance overall traffic management, they can also contribute to congestion and rear-end collisions. The standardised pattern used in the United Kingdom can be found in the Highway Code:

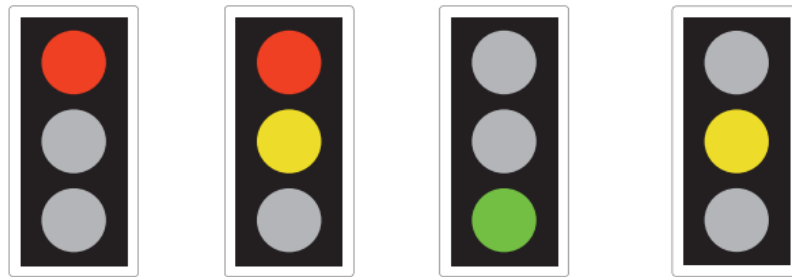


Figure 1: Highway Code Traffic Light Sequence, Source: GOV.UK [1]

A single red light means stop at the stop line, a red and amber light means prepare to start, a green light means to go if the way is clear and an amber light means stop and you may only go if you're past the stop line. The nature of the standardised signal means that all road users can understand the traffic light sequence no matter where they are in the United Kingdom.

A study conducted by Short et al. [2] compared the severity and number of accidents which occurred at junctions before and after traffic light systems were installed. The results showed a significant decrease of 34% in the number of "right-angle" collisions however there was a significant increase of 37% in "rear-end" collisions and a significant increase of 41% in "other" (head-on, vehicle/bicycle, fixed-object, etc.) collisions. The severity of the incidents decreased, with the rate of property damage only collisions increasing by 42%.

Though the Traffic Light sequence is standardised, the timing and pattern at which the Traffic Lights at junctions change depends on the needs of the situation i.e. traffic volume, flow consistency and location. To achieve the ideal pattern, different systems have been developed, these being: Fixed-Time Traffic Light Signals, Actuated Traffic Signals and Pre-Timed Traffic Signals.

Fixed Time Traffic Light Signals switch based upon predetermined signal durations for each phase of the signal cycle, regardless of the real-time traffic conditions. The signals normally follow a fixed pattern with each phase lasting for a set amount of time. The advantage of fixed-time signals is their simplicity, low cost of implementation and predictability of traffic flow, especially in areas with a consistent traffic pattern. However, a significant disadvantage of fixed-time signals is their inability to adapt to changing traffic conditions. For example, the ideal switching pattern would alter between periods of light traffic and heavy traffic, leading to traffic being forced to wait unnecessarily at red lights, leading to delays, frustration for drivers, an increase in fuel consumption and thus an increase in emissions. [6]

Actuated Traffic Light Signals operate based on real-time traffic conditions i.e. adjusting the signal timings in response to vehicle presence or motion. Actuated Traffic Light Signal Systems use various sensors, such as pressure sensors embedded in the road or overhead sensors like infrared or microwave detectors, to detect the presence of vehicles. When a sensor detects a vehicle approaching or waiting at an intersection, a signal is sent to the traffic light controller to adjust the signal timing accordingly, allowing for optimal traffic flow. One of the primary advantages of Actuated Traffic Light Signals is their ability to dynamically adapt to changing traffic patterns by only changing signals when necessary, reducing delays. Actuated Traffic Light Signals can also help conserve energy compared to fixed-time signals. However, the implementation of Actuated Traffic Light Signal Systems can be more costly due to the installation and maintenance of sensors. Additionally, the reliance on sensors introduces potential points of failure, leading to malfunctions or inaccuracies in the system. Despite these drawbacks, actuated traffic signals are widely used in areas with fluctuating traffic volumes and complicated junction layouts to improve overall traffic efficiency and safety. [7]

Pre-Timed Traffic Signals operate around predetermined signal timings which cycle through various patterns regardless of real-time traffic conditions. These systems typically have multiple timing patterns pre-programmed, allowing for the patterns to be changed based on the time of day. For example, during peak traffic hours, the pattern could prioritise longer green lights. While during off-peak hours, the green light phase may be shortened to accommodate lighter traffic. Pre-Timed Traffic Signals are designed to maintain a consistent flow of traffic and minimise unnecessary stops, reducing congestion and promoting smoother traffic movement. One advantage of pre-timed signals is their cost-effectiveness compared to Actuated Traffic Light Signal Systems. Additionally, they provide a degree of predictability for drivers, allowing them to anticipate signal changes. However, a disadvantage arises when traffic conditions deviate from the expected patterns, as pre-timed signals may not adapt dynamically to sudden changes in traffic volume or flow, potentially leading to delays and inefficiencies during unexpected traffic fluctuations. [8]

Traffic Light Systems are usually controlled using either Programmable Logic Controllers (PLCs) or Microcontrollers (for example Arduinos or PICs). PLCs are known for their robustness and reliability in industrial applications, making them well-suited for traffic light control systems which require high reliability and durability. Though Microcontrollers are not considered as robust, they instead offer a high level of versatility and flexibility. PLCs and Microcontrollers are designed with user-friendly programming interfaces: with PLCs often using ladder logic diagrams, which are easier to understand for individuals with a background in electrical engineering or industrial automation. Though Microcontrollers can be complicated to program, they can be programmed to perform a wide range of tasks beyond traffic light control, making them suitable for various applications. Programming Microcontrollers typically requires knowledge of programming languages such as C++ or C. PLCs and Microcontrollers are highly scalable, allowing for the expansion of the traffic light control system without significant changes to the existing infrastructure. The major upside of Microcontrollers is the cost, with Microcontrollers often being more cost-effective to implement than PLCs, especially for simpler Traffic Light Control Systems. The choice between Microcontrollers and PLCs depends on the specific requirements of the project i.e. the level of customisation needed, budget constraints and the expertise of those installing the system. Microcontrollers offer versatility and cost-effectiveness, while PLCs provide reliability, ease of use, and scalability. [6],[7],[9]

Logic

	Route 1	Route 2	Route 3	Route 4
0 - 10s	Traffic: Red Amber Pedestrian: Red	Traffic: Red Pedestrian: Red	Traffic: Red Amber Pedestrian: Red	Traffic: Red Pedestrian: Red
10 - 20s	Traffic: Green Pedestrian: Red	Traffic: Red Amber Pedestrian: Red	Traffic: Green Pedestrian: Red	Traffic: Red Amber Pedestrian: Red
20 - 30s	Traffic: Amber Pedestrian: Red	Traffic: Green Pedestrian: Red	Traffic: Amber Pedestrian: Red	Traffic: Green Pedestrian: Red
30 - 40s	Traffic: Red Pedestrian: Red	Traffic: Amber Pedestrian: Red	Traffic: Red Pedestrian: Red	Traffic: Amber Pedestrian: Red
40 - 50s	Traffic: Red Pedestrian: Green	Traffic: Red Pedestrian: Green	Traffic: Red Pedestrian: Green	Traffic: Red Pedestrian: Green

Table 1: Traffic Light Sequence Logic Table

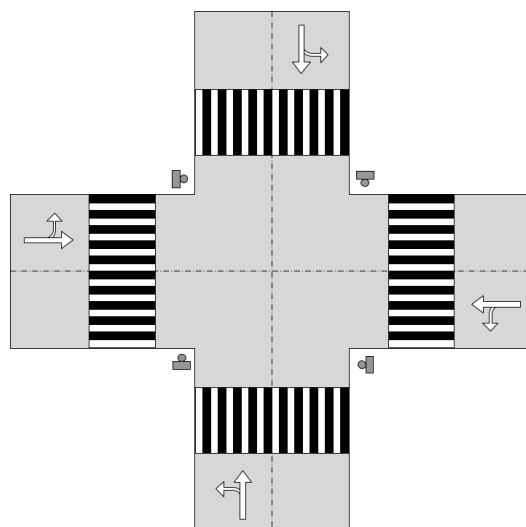


Figure 2: Traffic Light Sequence Road Arrangement

Flowchart

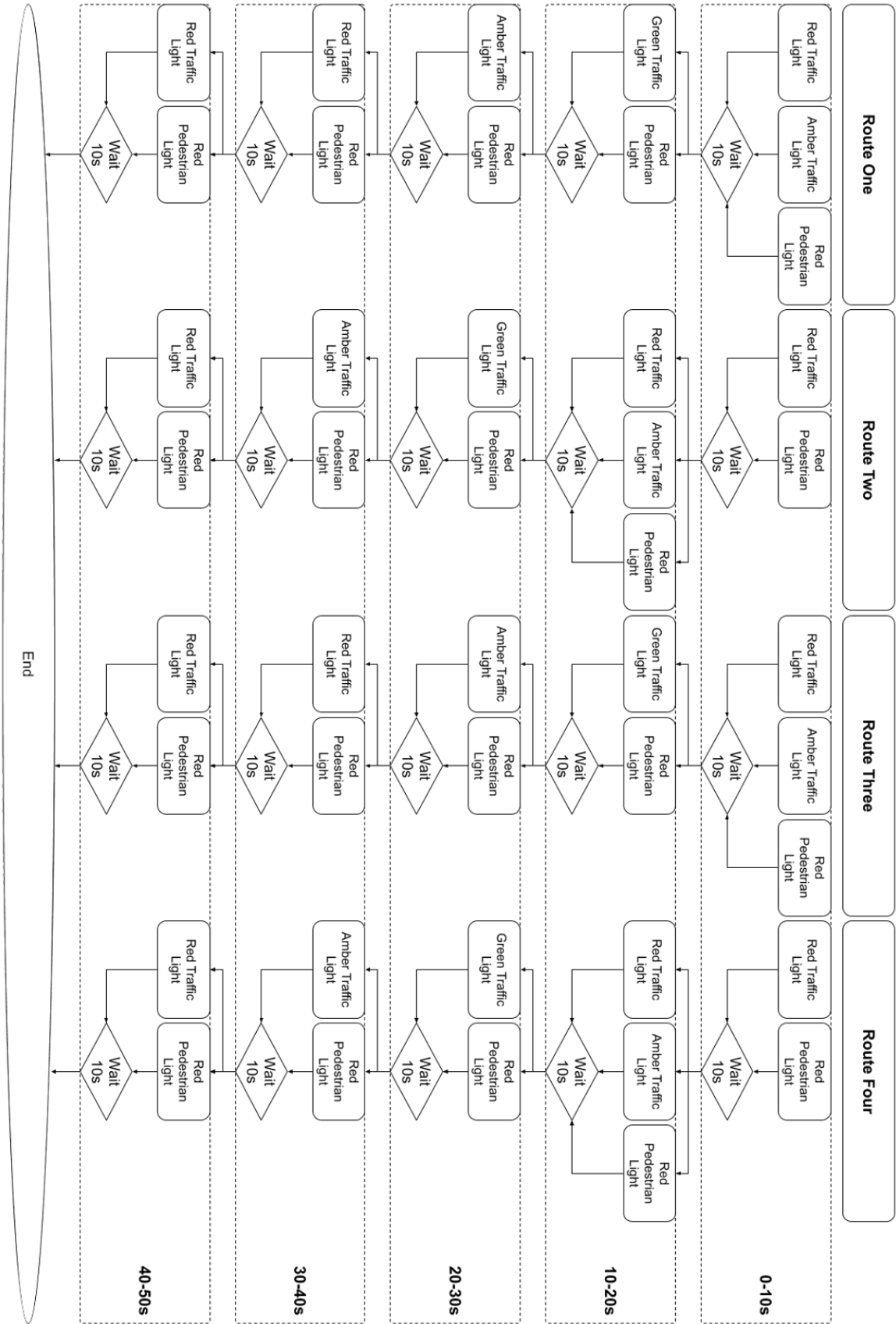


Figure 3: Traffic Light Sequence Flow Chart

Final Circuit Design

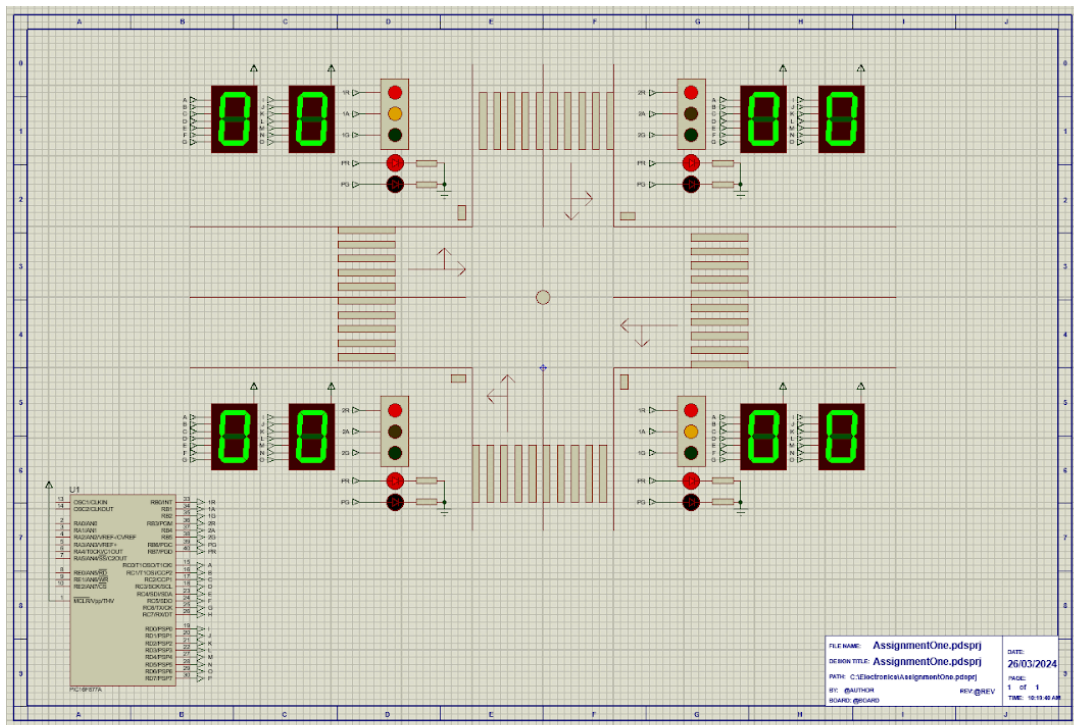


Figure 4: Final Circuit Design on Proteus

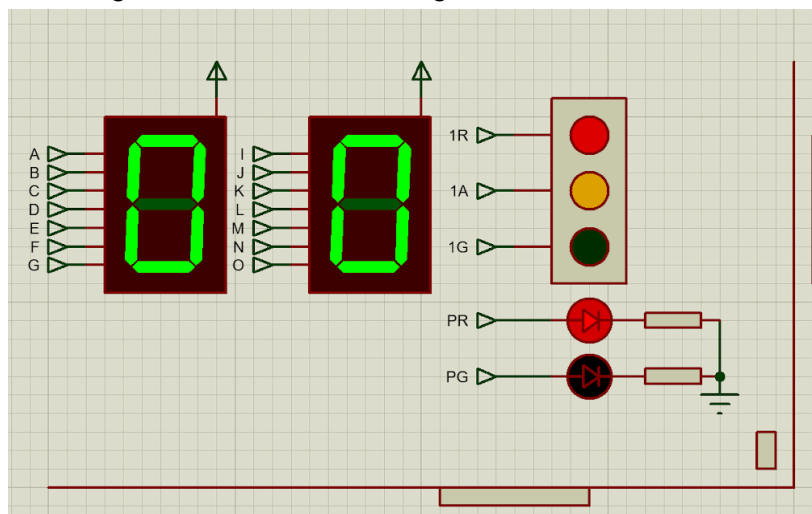


Figure 5: Traffic Lights Arrangement on Proteus

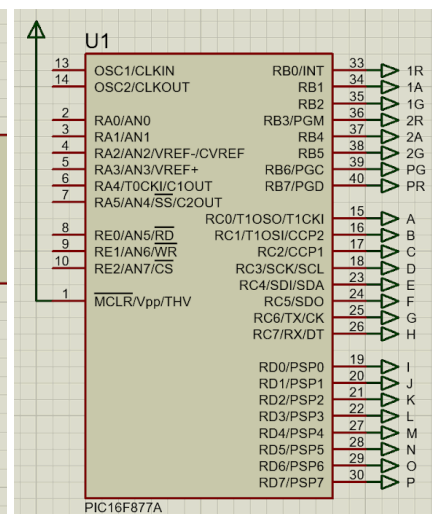


Figure 6: PIC on Proteus

To justify the requirements of the assignment a PIC16 (16F877A) was used within the software Proteus 8. Given the simplicity of the junction and design requirements, a Microcontroller seems more suitable in this example than a PLC. Overall, a PIC16F877A was used alongside, 4 resistors, 4 Red LEDs, 4 Green LEDs, 4 Traffic Light Units and 8 Seven Segment Displays. Rather than connecting the terminals of the PIC output to the components directly, Inputs and Output Terminals were used to reduce confusion. 2D Graphics were added to represent the junction, including Traffic Direction Arrows, rectangles representing the location of the Traffic Lights, Roads and Crossings. A 16F877A was used specifically due to familiarity with the Microcontroller, other Microcontrollers could have been used i.e. an 18F45K20.

The PIC required a power supply on terminal one, the Seven Segment Displays required a power supply on the anode. The PIC terminals B0 to B7 are used to supply power to the Traffic Lights: Terminals C0 to C6 are used for the 'tens' Seven Segment Display and terminals D0 to D6 are used for the 'ones' Seven Segment Display. Following Figure 7, the wiring between the output terminals of the PIC and the Seven Segment Display can be understood.

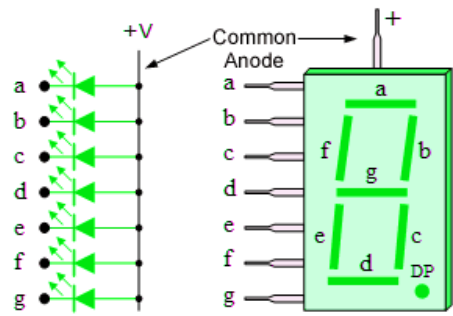


Figure 7: Seven Segment Display Pinout, Source: www.electronicsforu.com [3]

Wiring Arrangement					
PIC Terminal	Connection	Endpoint	PIC Terminal	Connection	Endpoint
B0	1R	Traffic Light One Red Light	C4	E	Seven Segment - Tens Terminal E
B1	1A	Traffic Light One Amber Light	C5	F	Seven Segment - Tens Terminal F
B2	1G	Traffic Light One Green Light	C6	G	Seven Segment - Tens Terminal G
B3	2R	Traffic Light Two Red Light	C7	H	Not Used
B4	2A	Traffic Light Two Amber Light	D0	I	Seven Segment - Ones Terminal A
B5	2G	Traffic Light Two Green Light	D1	J	Seven Segment - Ones Terminal B
B6	PG	Pedestrian Green Light	D2	K	Seven Segment - Ones Terminal C
B7	PR	Pedestrian Red Light	D3	L	Seven Segment - Ones Terminal D
C0	A	Seven Segment - Tens Terminal A	D4	M	Seven Segment - Ones Terminal E
C1	B	Seven Segment - Tens Terminal B	D5	N	Seven Segment - Ones Terminal F
C2	C	Seven Segment - Tens Terminal C	D6	O	Seven Segment - Ones Terminal G
C3	D	Seven Segment - Tens Terminal D	D7	P	Not Used

Table 2: Wiring Arrangement

Final Program

```

1  int i;
2  unsigned int display[10] = {0XC0, 0XF9, 0XA4, 0XB0, 0X99, 0X92, 0X82, 0XF8, 0X80, 0X90}; //Declairs Variable
3  //Hexadecimal Values for Seven Segment Display
4
5  sbit LED_Flow_One_Red at RB0_bit; //Variable LED_Flow_One_Red for Output B0
6  sbit LED_Flow_One_Yellow at RB1_bit; //Variable LED_Flow_One_Yellow for Output B1
7  sbit LED_Flow_One_Green at RB2_bit; //Variable LED_Flow_One_Green for Output B2
8  sbit LED_Flow_Two_Red at RB3_bit; //Variable LED_Flow_Two_Red for Output B3
9  sbit LED_Flow_Two_Yellow at RB4_bit; //Variable LED_Flow_Two_Yellow for Output B4
10 sbit LED_Flow_Two_Green at RB5_bit; //Variable LED_Flow_Two_Green for Output B5
11 sbit LED_Pedestrian_Green at RB6_bit; //Variable for Pedestrian Green for Output B6
12 sbit LED_Pedestrian_Red at RB7_bit; //Variable for Pedestrian Green for Output B7
13
14 void main() //Start of Program Logic
15 {
16     TRISB = 0X00; // Port B as Output
17     PORTB = 0X00; // Port B Logic Low
18     TRISC = 0X00; // Port C as Output
19     PORTC = 0X00; // Port C Logic Low
20     TRISD = 0X00; // Port D as Output
21     PORTD = 0X00; // Port D Logic Low
22     do // Start of Do-While Loop
23     {
24         for (i=0; i<10; i++) // After 0 Seconds
25         {
26             PORTC = 0XC0; // Port C Output Set Point
27             PORTD = display[i]; // Port D Output Set Point
28
29             // Route One - Get Ready
30             LED_Flow_One_Red = 1; // Traffic Light One - Red Light On
31             LED_Flow_One_Yellow = 1; // Traffic Light One - Amber Light On
32             LED_Flow_One_Green = 0; // Traffic Light One
33             // Route Two - Stop
34             LED_Flow_Two_Red = 1; // Traffic Light Two - Green Light On
35             LED_Flow_Two_Yellow = 0; // Traffic Light Two
36             LED_Flow_Two_Green = 0; // Traffic Light Two
37             // Pedestrian - Stop
38             LED_Pedestrian_Green = 0; // Pedestrian Light
39             LED_Pedestrian_Red = 1; // Pedestrian Light - Red Light On
40             Delay_ms(1000); // 10 Second Delay
41         }
42         for (i=0; i<10; i++) // After 9 Seconds
43         {
44             PORTC = 0XF9; // Port C Output Set Point
45             PORTD = display[i]; // Port D Output Set Point
46
47             // Route One - Go
48             LED_Flow_One_Red = 0; // Traffic Light One
49             LED_Flow_One_Yellow = 0; // Traffic Light One
50             LED_Flow_One_Green = 1; // Traffic Light One - Green Light On
51             // Route Two - Get Ready
52             LED_Flow_Two_Red = 1; // Traffic Light Two - Red Light On
53             LED_Flow_Two_Yellow = 1; // Traffic Light Two - Amber Light On
54             LED_Flow_Two_Green = 0; // Traffic Light Two
55             // Pedestrian - Stop
56             LED_Pedestrian_Green = 0; // Pedestrian Light
57             LED_Pedestrian_Red = 1; // Pedestrian Light - Red Light On
58             Delay_ms(1000); // 10 Second Delay
59         }
60         for (i=0; i<10; i++) // After 19 Seconds
61         {
62             PORTC = 0XA4; // Port C Output Set Point
63             PORTD = display[i]; // Port D Output Set Point
64
65             // Route One - Prepare to Stop
66             LED_Flow_One_Red = 0; // Traffic Light One
67             LED_Flow_One_Yellow = 1; // Traffic Light One - Amber Light On
68             LED_Flow_One_Green = 0; // Traffic Light One
69             // Route Two - Go
70             LED_Flow_Two_Red = 0; // Traffic Light Two
71             LED_Flow_Two_Yellow = 0; // Traffic Light Two
72             LED_Flow_Two_Green = 1; // Traffic Light Two - Green Light On
73             // Pedestrian - Stop
74             LED_Pedestrian_Green = 0; // Pedestrian Light
75             LED_Pedestrian_Red = 1; // Pedestrian Light - Red Light On
76             Delay_ms(1000); // 10 Second Delay
77         }
78         for (i=0; i<10; i++) // After 29 Seconds
79         {
80             PORTC = 0XB0; // Port C Output Set Point
81             PORTD = display[i]; // Port D Output Set Point
82
83             // Route One - Stop
84             LED_Flow_One_Red = 1; // Traffic Light One - Red Light On
85             LED_Flow_One_Yellow = 0; // Traffic Light One
86             LED_Flow_One_Green = 0; // Traffic Light One
87             // Route Two - Prepare to Stop
88             LED_Flow_Two_Red = 0; // Traffic Light Two
89             LED_Flow_Two_Yellow = 1; // Traffic Light Two - Amber Light On
90

```

```

90 // Route Two - Repair to Stop
LED_Flow_Two_Red = 0;
LED_Flow_Two_Yellow = 1;
LED_Flow_Two_Green = 0;
// Pedestrian - Stop
LED_Pedestrian_Green = 0;
LED_Pedestrian_Red = 1;

Delay_ms(1000);
}
for (i=0; i<10; i++)
{
PORTC = 0X99;
PORTD = display [i];

// Route One - Stop
LED_Flow_One_Red = 1;
LED_Flow_One_Yellow = 0;
LED_Flow_One_Green = 0;
// Route Two - Stop
LED_Flow_Two_Red = 1;
LED_Flow_Two_Yellow = 0;
LED_Flow_Two_Green = 0;
// Pedestrian - Go
LED_Pedestrian_Green = 1;
LED_Pedestrian_Red = 0;

Delay_ms(1000);
}
for (i=0; i<1; i++)
{
// After 39 Seconds

// Port C Output Set Point
// Port D Output Set Point

// Traffic Light Two - Red Light On
// Traffic Light One
// Traffic Light One
// Traffic Light Two - Red Light On
// Traffic Light Two
// Traffic Light Two
// Pedestrian Light - Green Light On
// Pedestrian Light

// 10 Second Delay

// After 49 Seconds

for (i=0; i<1; i++)
{
// After 49 Seconds

// Port C Output Set Point
// Port D Output Set Point

// Traffic Light One - Red Light On
// Traffic Light One
// Traffic Light One
// Traffic Light Two - Red Light On
// Traffic Light Two
// Traffic Light Two
// Pedestrian Light - Green Light On
// Pedestrian Light

// Delay

// End of Do-While Loop
}
while(1);
}

```

Figure 8: Finale Program on MicroC

The program was written and compiled using Micro C Software; the program is written into the software as C and the software then compiles the program into several different languages which can be used with a variety of different controllers. The PIC16 in this example uses the subsequent HEX file, shown below. The program has been annotated to provide a better understanding of the working principles so any changes can be made quite easily.

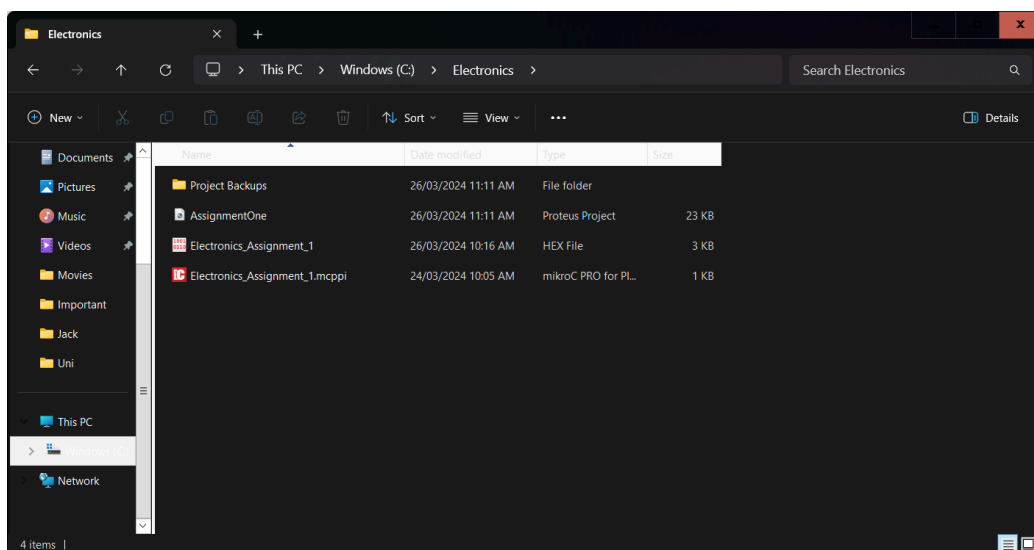


Figure 9: Hex File from MicroC

Simulation Screen Shots

0 - 9 Seconds

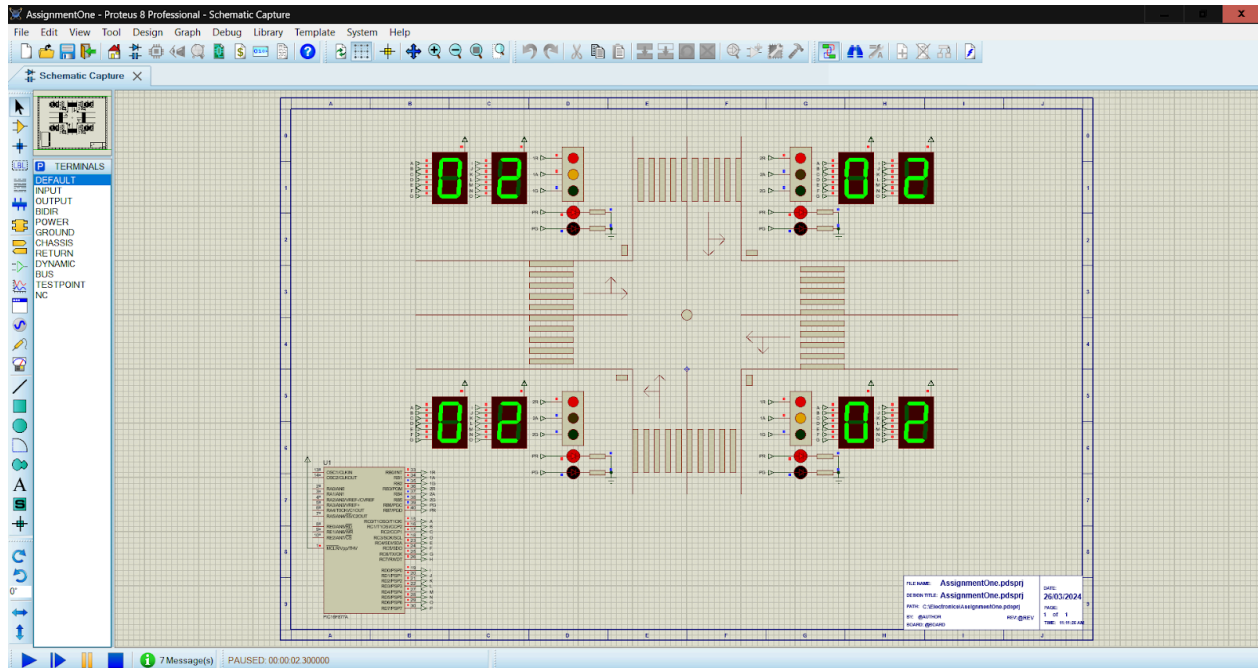


Figure 10: 0 - 9 Seconds on Proteus

10 - 19 Seconds

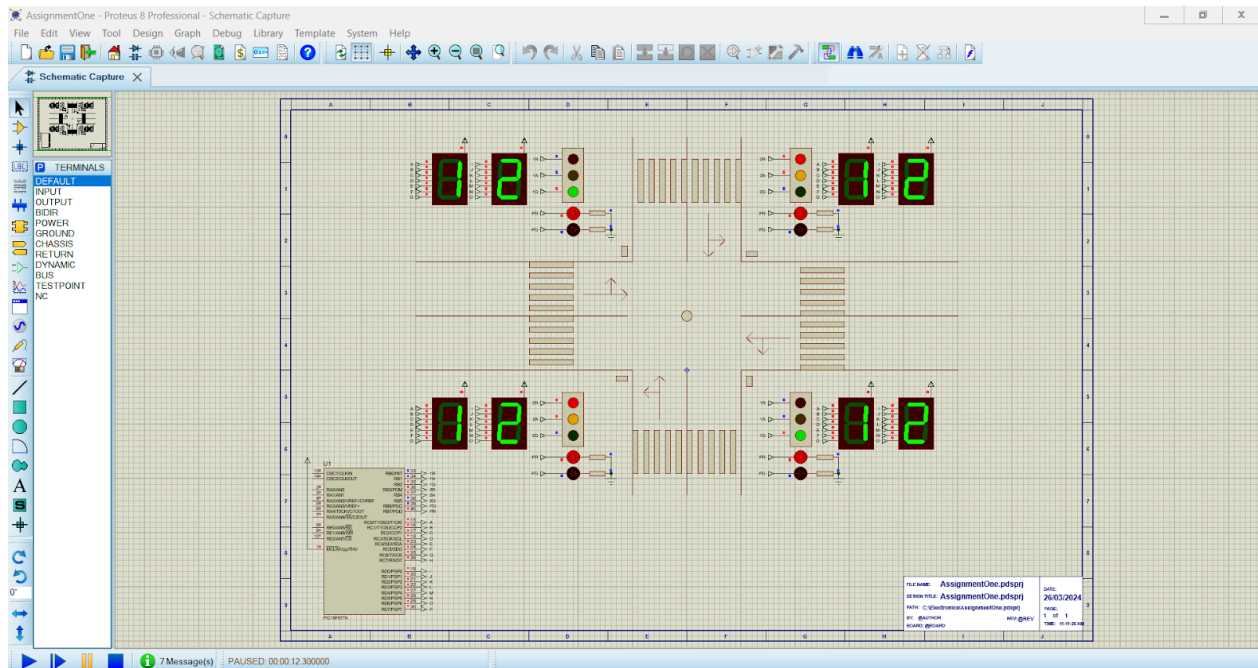


Figure 11: 10 - 19 Seconds on Proteus

20 - 29 Seconds

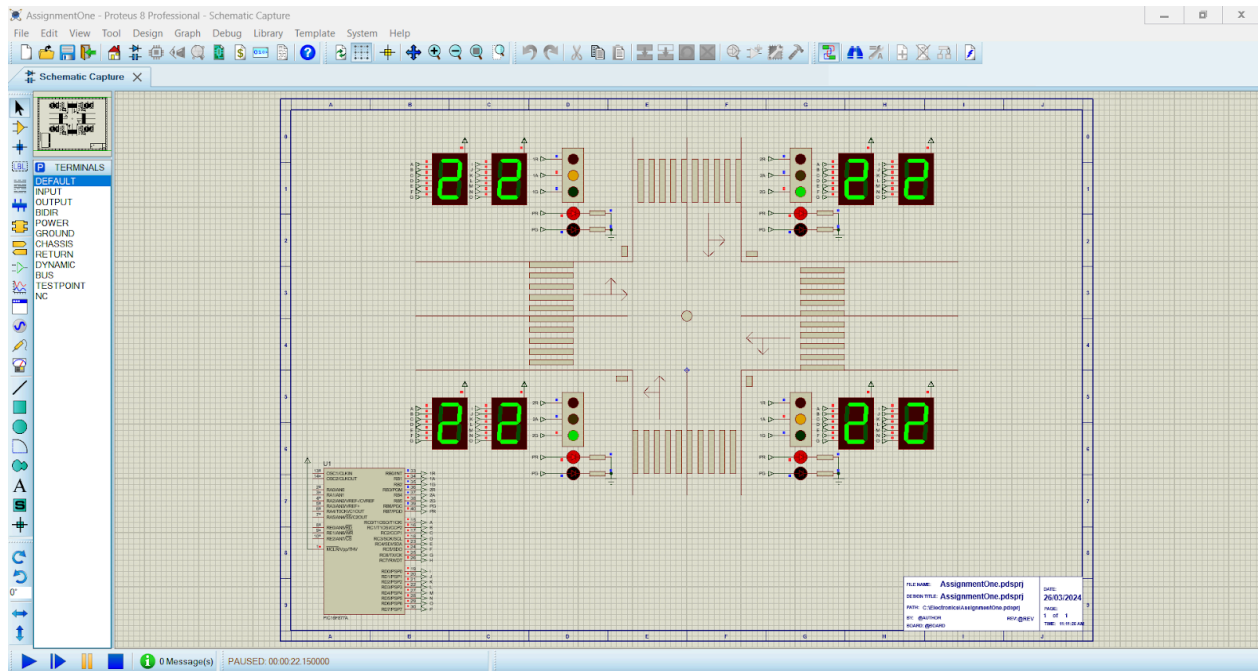


Figure 12: 20 - 29 Seconds on Proteus

30 - 39 Seconds

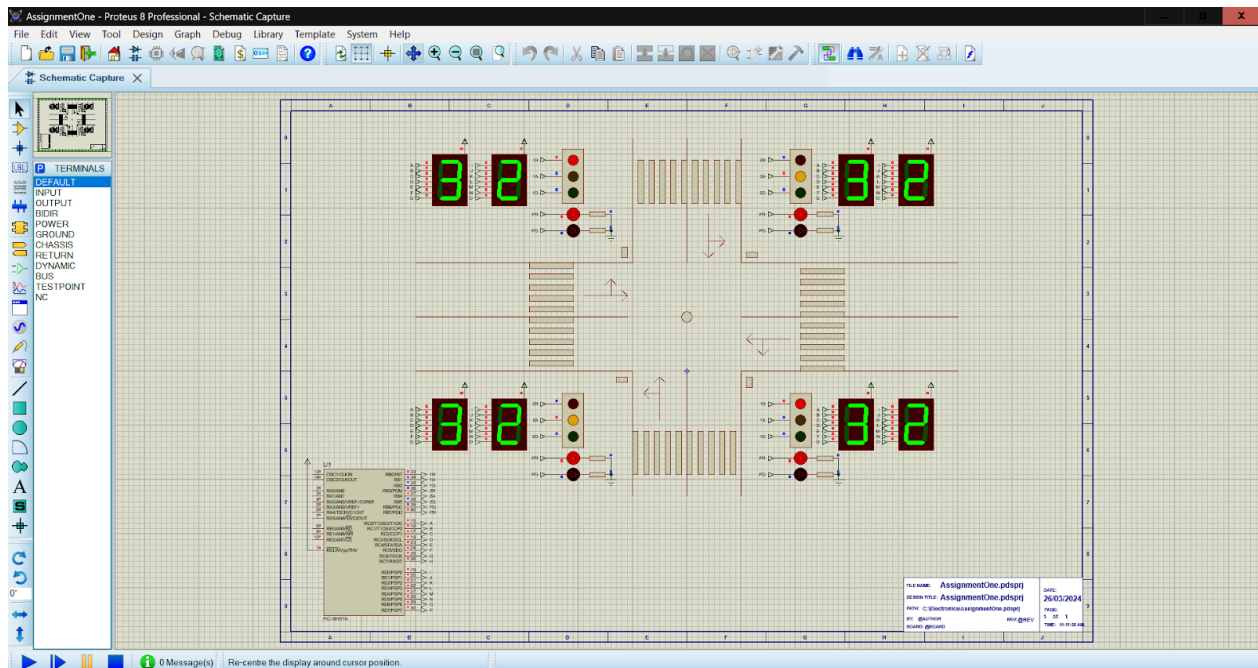


Figure 13: 30 - 39 Seconds on Proteus

40 - 49 Seconds

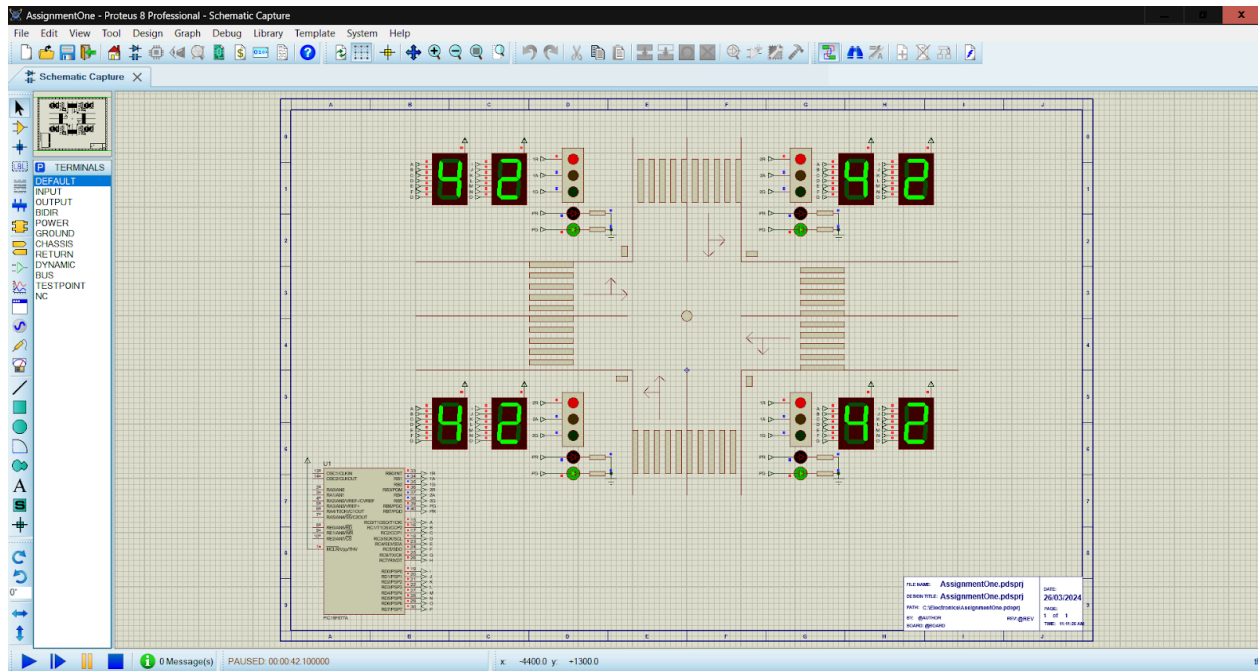


Figure 14: 40 - 49 Seconds on Proteus

50 Seconds

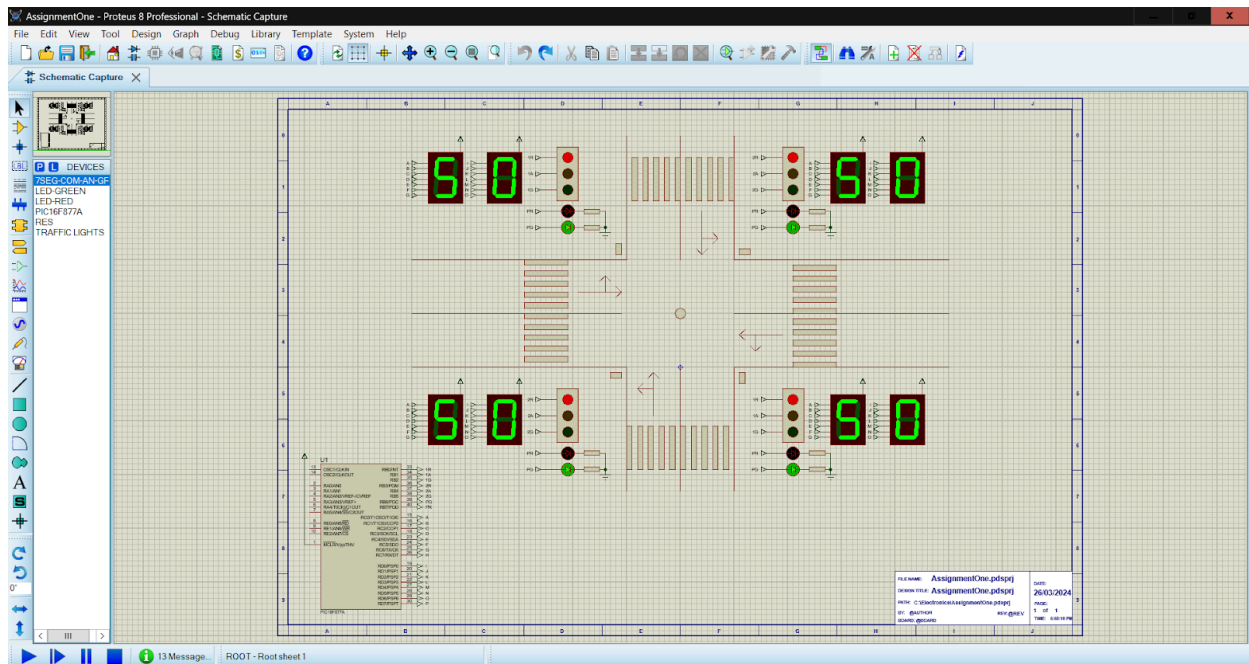


Figure 15: 50 Seconds on Proteus

Simulation Recording

https://drive.google.com/file/d/1yNvRLzmbMb675r4gCVDFs-5q9MTHZ7_O/view?usp=sharing

Practical Testing Results

Taking a practical approach to testing, as opposed to solely relying on simulations, enables a more comprehensive examination of the system. A practical approach unveils faults or design flaws that may not be apparent during simulation in software like Proteus. However, in the practical testing phase of this circuit, with the use of a breadboard and various components, no major problems or faults emerged, contrary to expectations. A PicStart Plus Development Tool was used to program the PIC within MicroC, a plethora of LEDs were used to simulate Traffic and Pedestrian Lights, a Variable Power Supply was used to supply power to the circuit, jumper wires were also used as well as a Breadboard. The biggest difficulties encountered were wiring up the PIC16F877A and the 7 Segment Displays using the relevant Pinouts and then finally loading the program to the PIC itself. If more time was spent developing the system practically, a Prototyping Board would be used instead of a Breadboard so the system could be understood simply rather than following jumper wires. There would also be a lower risk of components losing connection to the board and the 7 Segment displays could be next to each other rather than horizontally (a shortcoming of using the breadboard). [3], [4]

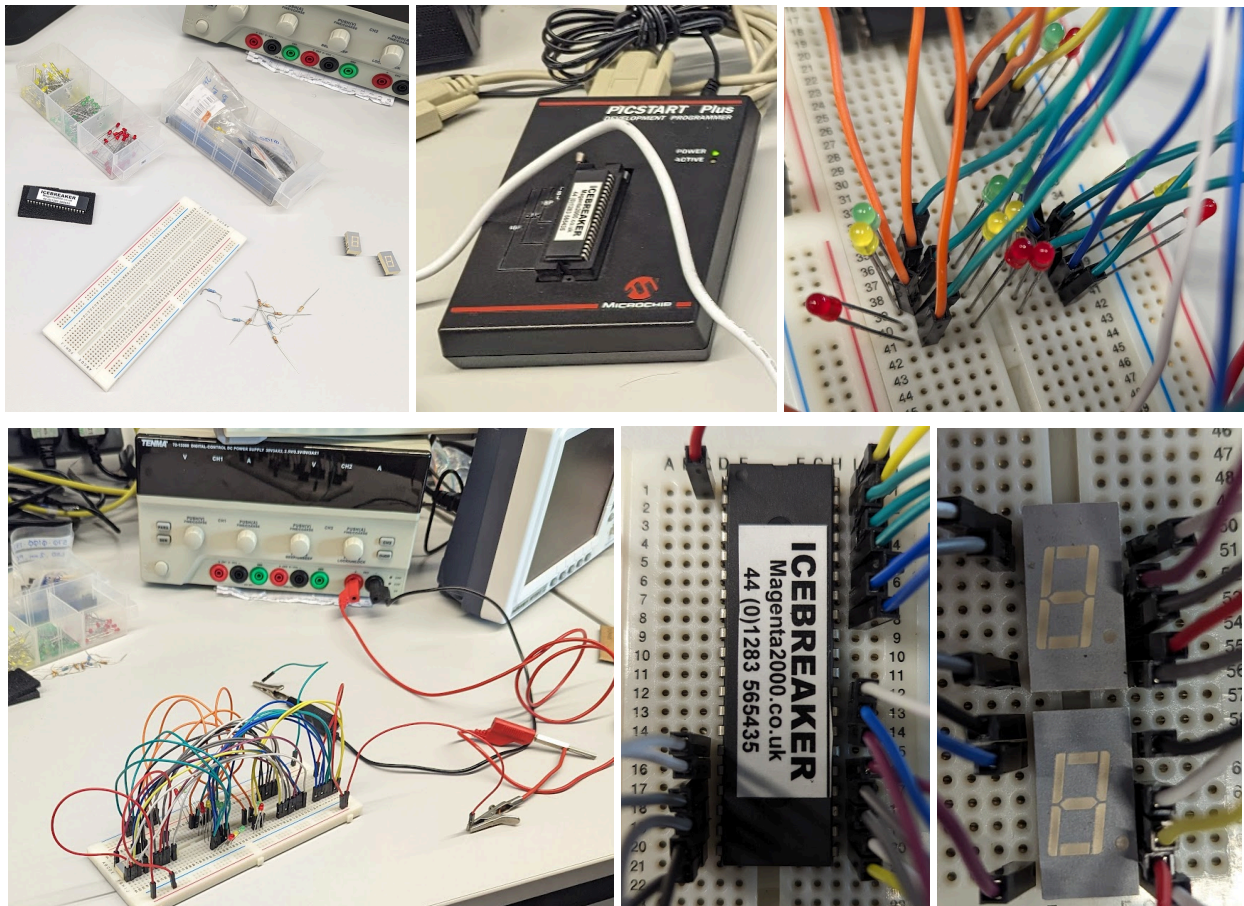


Figure 16: Collection of Photos from Practical Testing

Conclusion and Recommendations

The purpose of this task was to design a circuit, using either a PIC16F or a PIC18F, for a traffic light control system for a four-way junction, featuring: four traffic lights on each side of the road for comprehensive traffic management, a set of 7-segment displays on each side to count from 0 to 50 seconds, with the system to change traffic signals every 10 seconds for predictable and smooth traffic flow and the final 10 seconds of the signal cycle being reserved for a dedicated pedestrian crossing phase. This report has presented evidence of the completion of this task, with Proteus being used to design and model the system, MicroC being used to produce the program for the Microcontroller and practical testing being completed to show an added understanding of electronics and a greater overview of the requirements if the system was used in the real world.

The four most obvious recommendations for this task if completed in the future would be to alter the time delay between the lights changing so they were not all 10 seconds as this is not realistic. The reason for this is a result of the assignment brief.

The second recommendation would be that a PIC alone could not be used to switch actual traffic lights which usually operate on 240V A.C. as their maximum operating voltage is 6.0V as standard. As a result, contactors or relays would have to be used within the circuit for the switching of the 240V.

The next recommendation would be to alter the timings so that Route One isn't on Amber (stopping) while Route Two is on Green (go). If this was used in the real world this would likely cause a collision. The system was programmed this way to meet the standardised sequence laid out by the Highway Code while keeping to the 50 second time limit and 10 second switching time. The simplest solution to the issue would be to increase the time limit to 60 seconds however this could also be achieved by removing the 10 second switching time requirement.

A PIC16 was used as a result of the assignment brief however PICs are steadily being replaced by Arduinos, though they are slightly more expensive, they offer a greater range of capabilities and are often considered easier to program. As a result, using an Arduino could provide a more realistic example of if this task was given in the real world.

Overall the task has been completed to meet the criteria laid out in the assignment brief, a PIC16 has been used to provide control of a four-way traffic light system, with the incorporation of a set of 7-segment displays on each side to count from 0 to 50 seconds. As mentioned in the assignment brief, the system changes traffic signals every 10 seconds for predictable and smooth traffic flow and the final 10 seconds of the signal cycle has been reserved for a dedicated pedestrian crossing phase.

References

- [1]Gov.UK, "Light Signals Controlling Traffic - the Highway Code - Guidance - GOV.UK," www.gov.uk, Sep. 17, 2023. <https://www.gov.uk/guidance/the-highway-code/light-signals-controlling-traffic> (accessed Mar. 26, 2024).
- [2]M. S. Short, G. A. Woelfl, and C.-J. Chang, "Effects of Traffic Signal Installation on Accidents," *Accident Analysis & Prevention*, vol. 14, no. 2, pp. 135–145, Apr. 1982, doi: [https://doi.org/10.1016/0001-4575\(82\)90080-X](https://doi.org/10.1016/0001-4575(82)90080-X) (accessed Mar. 26, 2024).
- [3]Saurabh, "7 Segment Display Pinout and Working | Basics for Beginners," *Electronics For You*, Jun. 04, 2019. <https://www.electronicsforu.com/resources/7-segment-display-pinout-understanding> (accessed Mar. 26, 2024).
- [4]Components 101, "PIC16F877A: Introduction, Pin Diagram, Pin Description, Features & Datasheet," Components101, 2017. <https://components101.com/microcontrollers/pic16f877a-pin-diagram-description-features-datasheet> (accessed Apr. 09, 2024).
- [5]Microchip, "Dynamic Product Page | Microchip Technology," www.microchip.com, Aug. 01, 2022. <https://www.microchip.com/en-us/product/PIC16F877A> (accessed Apr. 09, 2024).
- [6]AutoMate Systems, "How Modern Traffic Lights Signals Work? - Auto Mate Systems Ltd.," www.automatesystems.co.uk, Oct. 05, 2015. <https://www.automatesystems.co.uk/traffic-lights-signals-work#:~:text=Fixed%20period%20traffic%20lights%20signals> (accessed Apr. 09, 2024).
- [7]Prof. T. V. Mathew, "Vehicle Actuated Signals," www.civil.iitb.ac.in, Sep. 28, 2023. https://www.civil.iitb.ac.in/tvm/nptel/576_VAS/web/web.html#:~:text=They%20repeat%20a%20preset%20constant (accessed Apr. 09, 2024).
- [8]Greenlight Traffic Engineering, "A Guide to Types of Traffic Signals - Greenlight Traffic Engineering, LLC," Greenlight Traffic Engineering, Jun. 21, 2023. <https://www.greenlighttrafficengineering.com/a-guide-to-types-of-traffic-signals/> (accessed Apr. 09, 2024).
- [9]SecoSouth, "How Do Traffic Light Control Systems Actually Work?," SecoSouth, Oct. 18, 2021. <https://secosouth.net/2021/10/how-do-traffic-light-control-systems-actually-work/> (accessed Apr. 09, 2024).
- [10]P. Horowitz and W. Hill, "The Art of Electronics, by Paul Horowitz and Winfield Hill," 1989. <https://eclass.uniwa.gr/modules/document/index.php?course=EEE265&download=/6218412aGLbn/5e6db8cd5KNP.pdf> (accessed Apr. 09, 2024).
- [11]D. Ibrahim, *PIC BASIC: Programming and Projects*. Elsevier, 2001. https://books.google.co.uk/books?hl=en&lr=&id=HlhuovdeStQC&oi=fnd&pg=PP1&dq=pic+programming&ots=8B9tx0a2v7&sig=wxIB7yBTvYhmScxr9VWnoLCstM0&redir_esc=y#v=onepage&q=pic%20programming&f=false (accessed Apr. 09, 2024).
- [12]J. Sanchez and M. P. Canton, *Microcontroller Programming*. CRC Press, 2018. (accessed Apr. 09, 2024).