

Electronics I

by Ayoob Mohammed Azhar

Submission date: 04-Dec-2025 06:11PM (UTC+0000)

Submission ID: 269630384

File name: 57126_Ayoob_Mohammed_Azhar_Electronics_I_634321_584083934.docx (8.82M)

Word count: 3015

Character count: 15613



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

Electronics I
(ACDB5002)

Electrical and Electronic Engineering

Ayoob Mohammed Azhar (2400571)

CONTENTS

Introduction	page 2
Background research	page 3
System design	
Circuit design (on Proteus)	page 4 - 5
Breadboard prototype	page 5
System logic	page 6 - 8
Simulation	
Proteus	page 9 - 15
Breadboard	page 15 - 18
Results	
Proteus	page 19
Breadboard	page 19
Conclusion	page 20
References	page 21
Appendices	page 22 - 24

Introduction

The purpose of this report is to demonstrate the development and design of a fixed-time traffic light controller system for a T junction. It aims to improve traffic flow and pedestrian safety by adjusting signal timing to precise intervals. The circuit was first designed and tested using Proteus (design suite software). The microcontroller used is a PIC 16F877A, which was compiled in the CCS C compiler and programmed in C with Proteus. The project was then attempted on a breadboard prototype to simulate real-world conditions.

Background research

Various types of traffic light controllers have been developed and implemented in the UK to date. The 'traditional' type of traffic light controller that is still used today is the fixed-time traffic controller [1]. It operates on a pre-programmed sequence to cycle through the red, amber, and green lights at fixed times on a consistent schedule [1]. Signal phase timings (for the red, amber, and green lights) at the T junction are calculated by gathering traffic flow data and then applying Webster's formula to allocate optimal phase timings [2]. They are suitable for T junctions with low-traffic and predictable, consistent traffic flow. The fixed-time traffic light controller is also simple to implement and requires less maintenance. However, they are not responsive to real-time traffic, and can cause unnecessary waiting time when there is less traffic. [1]

Moreover, the actuated traffic light controller has become more preferred at T junctions that have varying and unpredictable traffic. Incoming vehicles are detected by utilising sensors and/or cameras, which are then used to adjust signal timing based on actual traffic conditions. They are responsive to real-time traffic, which increases efficiency by improving traffic flow at off-peak times and minimising unnecessary waiting times. However, they are more expensive than the fixed-time traffic light controller, and need continuing maintenance for the detection systems. [1]

The more advanced type that can be seen at T junctions nowadays is the adaptive traffic light controller, which is used in urban areas with high traffic volume. An example of this in the UK is the MOVA (Microprocessor Optimised Vehicle Actuation) traffic light controller [3]. They use advanced algorithms to analyse traffic patterns in real-time, forecast traffic congestion, and adjust signal timing in real-time. This operates using a network of sensors/detectors, cameras, and a processor to collect and process data across several intersections. After that, the data is examined either locally or centrally to modify signal timing and phases based on the actual and expected traffic flow. They are very efficient as they optimise traffic flow along a series of interconnected intersections. They are very responsive to any time of day patterns and real-time traffic conditions. They provide the best balance of efficiency and safety. However, they do have a high set-up cost, a complex installation, and require expert operators. [1]

Aside from enhancing traffic flow, pedestrian safety is an essential consideration at T junctions with traffic light controllers. According to the Department for Transport UK, there must be dedicated pedestrian crossing phases with clear indications. A 'green man' light indicates that pedestrians can safely cross, and a 'red man' light indicates that pedestrians must wait and not cross, as the road is unsafe to cross at that time. Pedestrian countdown displays are also utilised to indicate the duration of the 'blackout' period (when no pedestrian light is shown), and therefore show pedestrians how much time they have to finish crossing the road. [4]

System design

Circuit design (on Proteus)

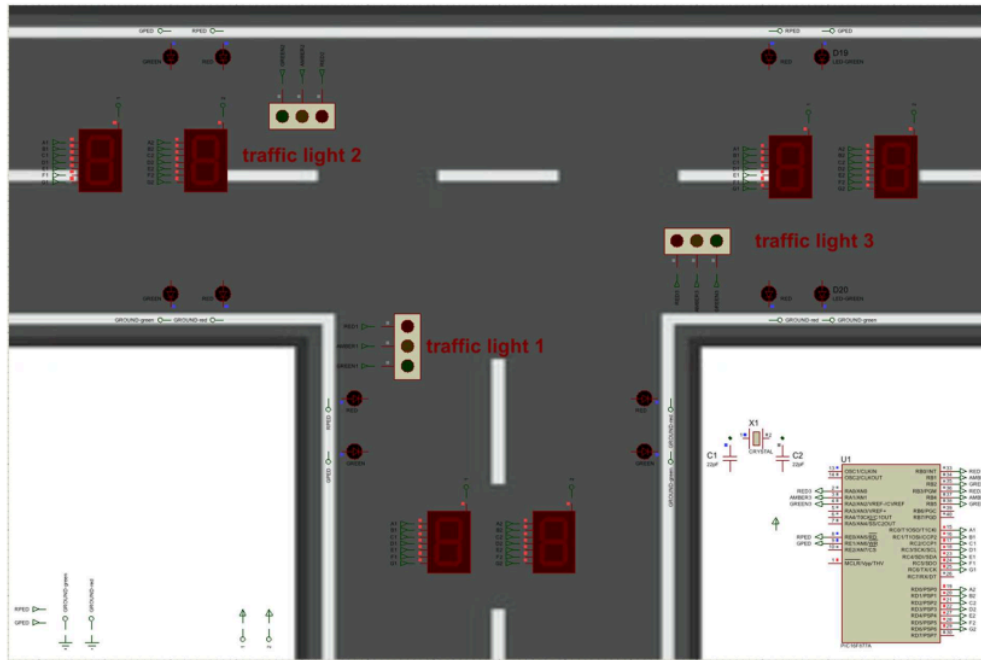


Figure 1: Circuit design in Proteus

For this project, a fixed-time traffic controller type was the most suitable choice because the task's requirements specified predetermined timing intervals, and the traffic light control system here doesn't need to rely on real-world traffic conditions. This includes changing the traffic signals every 10 seconds, having a dedicated pedestrian crossing phase at the final 10 seconds, and keeping the total cycle time at 40 seconds. Because this type requires fewer components, such as no sensors/detectors, it is simple to implement and cost-effective.

When considering the microcontroller for this system, a PIC microcontroller is chosen because the university has a license for the CCS C compiler, which is used to program the microcontroller. A 16F877A (40-pin package) was selected due to its familiarity with operating, and it effectively meets the project requirements. It has a clock speed of 20 MHz, offers five ports with 33 programmable I/O pins across them (out of a total of 40 pins), up to 8K x 14 words of Flash Program Memory, and up to 368 x 8 bytes of Data Memory (RAM) [5]. Because a fixed-time traffic controller type is used here, only simple programming of updates and time delays for the traffic lights, pedestrian lights and 7-segment displays is executed, requiring very little processing power; therefore, the 20 MHz clock speed is more than enough. The I/O pins of this PIC are enough to drive three sets of traffic light modules (using 9 pins in total), two 7-segment displays (using 14 pins) and the pedestrian lights (using two pins), all across the five ports. The program memory here is also more than sufficient for storing the program code required for this project.

In contrast, when deciding whether to choose a PIC in the PIC 18 series, it provides a higher clock speed (40MHz), more program memory (up to 2 Mbyte) and more programmable I/O pins, [6] although with a higher cost. However, these features are not necessary for the fixed-time traffic

controller type utilised in this project. Whereas the PIC 16F877A chosen here can reliably handle the requirements for this project at a lower cost, making it cost-effective.

In Figure 1, the crystal oscillator provides the clock frequency and timing reference for the PIC microcontroller. 22pF capacitors are used along with it to filter out noise and ensure a stable clock signal is generated. [7]

For the traffic lights and pedestrian lights, LEDs were chosen because they are cheap, they consume less power, and last a long time.

In Figure 1, traffic light modules (comprising red, amber, and green LEDs) were selected to make the schematic clearer and closer to a real single traffic light unit, whilst reducing additional wiring.

As shown in Figure 1, a set of red and green pedestrian lights at each side of the road was chosen to represent real-life pedestrian safety facilities at a T junction. One pin powers each set of green lights, whereas the other pin powers each set of red lights.

Here, 7-segment displays with a common anode configuration were used as the PIC is better at sinking(grounding) current from the display than sourcing(supplying) it to the display [8]. In Figure 1, the 7-segment displays for the tens digit are connected to port C, and the 7-segment displays for the ones digit are connected to port D.

Breadboard prototype

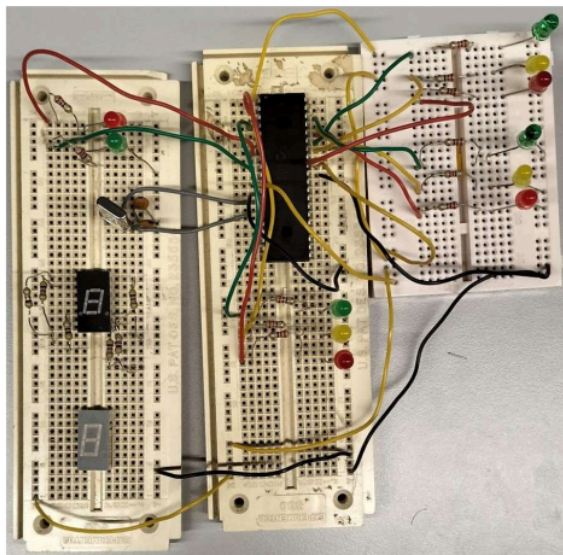


Figure 2: Circuit layout on breadboard

Figure 2 shows the attempt to build the traffic light control system circuit on a breadboard using jumper wires and the following components: an Op Amp, LEDs, resistors, a crystal oscillator, capacitors and seven-segment displays (not included during circuit operation). Unlike in Proteus, which has internal resistances for ideal simulation, the resistors are used here to limit current to the LEDs.

System logic

The following pseudocode describes the operation of the fixed-time traffic light controller in this project.

//Initialisation and setup of ports

Clear all ports and set them as outputs (for traffic lights, pedestrian lights and 7-segment displays).

//Main loop

Repeat forever

//Phase 1

Turn ON the red lights for traffic lights 1, 2 and 3.

Turn OFF the pedestrian green light.

Turn ON the pedestrian red light.

Set the displays to start from 40, setting 4 on the tens digit display and 0 on the ones digit display.

Delay for 1 second.

Turn ON the traffic light 1 amber light.

Set the displays to go down to 39 by updating 3 on the tens digit display and 9 on the ones digit display.

Delay for 1 second.

Turn OFF the red light and amber light for traffic light 1.

Turn ON the traffic light 1 green light.

Set the displays to go down to 33 by updating the ones digit display, setting the digits 8, 7, 6, 5, 4, and then 3.

Delay for 1 second between each digit update.

//(A total of 6 seconds)

Turn ON the traffic light 1 amber light.

Turn OFF the traffic light 1 green light.

Set the displays to go down to 31 by updating the ones digit display, setting the digits 2 and then 1.

Delay for 1 second between each digit update.

//(A total of 2 seconds)

//Phase 2

Turn OFF the traffic light 1 amber light.

Turn ON the traffic light 1 red light.

Turn ON the traffic light 2 amber light.

Set the displays to go down to 30 by updating the ones digit display to 0.

Delay for 1 second.

Turn OFF the red light and amber light for traffic light 2.

Turn ON the traffic light 2 green light.

Set the displays to go down to 24 by updating 2 on the tens digit display and 9 on the ones digit display.

Delay for 1 second.

Then set the ones digits to 8, 7, 6, 5, and then 4.

Delay for 1 second between each digit update.

//(A total of 6 seconds)

Turn ON the traffic light 2 amber light.

Turn OFF the traffic light 2 green light.

Set the displays to go down to 22 by updating the ones digit display, setting the digits 3 and then 2.

Delay for 1 second between each digit update.

//(A total of 2 seconds)

Turn ON the traffic light 2 red light.

Turn OFF the traffic light 2 amber light.

Set the displays to go down to 21 by updating the ones digit display to 1.

Delay for 1 second.

//Phase 3

Turn ON the traffic light 3 amber light.

Set the displays to go down to 20 by updating the ones digit display to 0.

Delay for 1 second.

Turn OFF the red light and amber light for traffic light 3.

Turn ON the traffic light 3 green light.

Set the displays to go down to 14 by updating 1 on the tens digit display and 9 on the ones digit display.

Delay for 1 second.

Then set the ones digits to 8, 7, 6, 5, and then 4.

Delay for 1 second between each digit update.

//(A total of 6 seconds)

Turn ON the traffic light 3 amber light.

Turn OFF the traffic light 3 green light.

Set the displays to go down to 12 by updating the ones digit display, setting the digits 3 and then 2.

Delay for 1 second between each digit update.

//(A total of 2 seconds)

Turn ON the traffic light 3 red light.

Turn OFF the traffic light 3 amber light.

Set the displays to go down to 11 by updating the ones digit display to 1.

Delay for 1 second.

//Pedestrian crossing phase

//All traffic lights will have their red light ON from their last updated code.

Turn OFF the pedestrian red light.

Turn ON the pedestrian green light.

Set the displays to go down to 10 by updating the ones digit display to 0.

Delay for 1 second.

Set the displays to go down to 01 by updating 0 on the tens digit display and 9 on the ones digit display.

Delay for 1 second.

Then set the ones digits to 8, 7, 6, 5, 4, 3, 2, and 1.

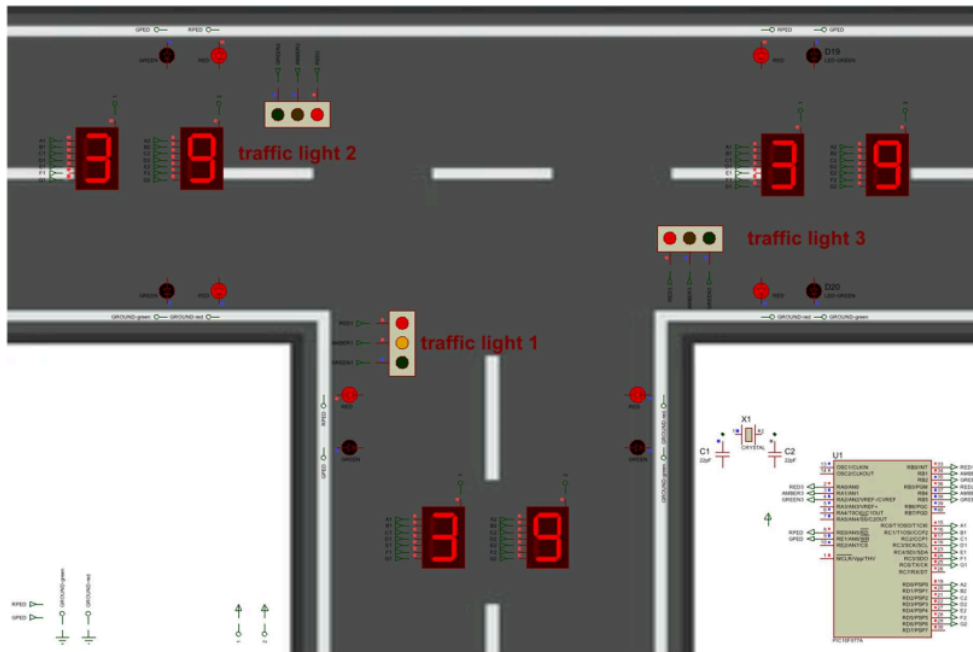
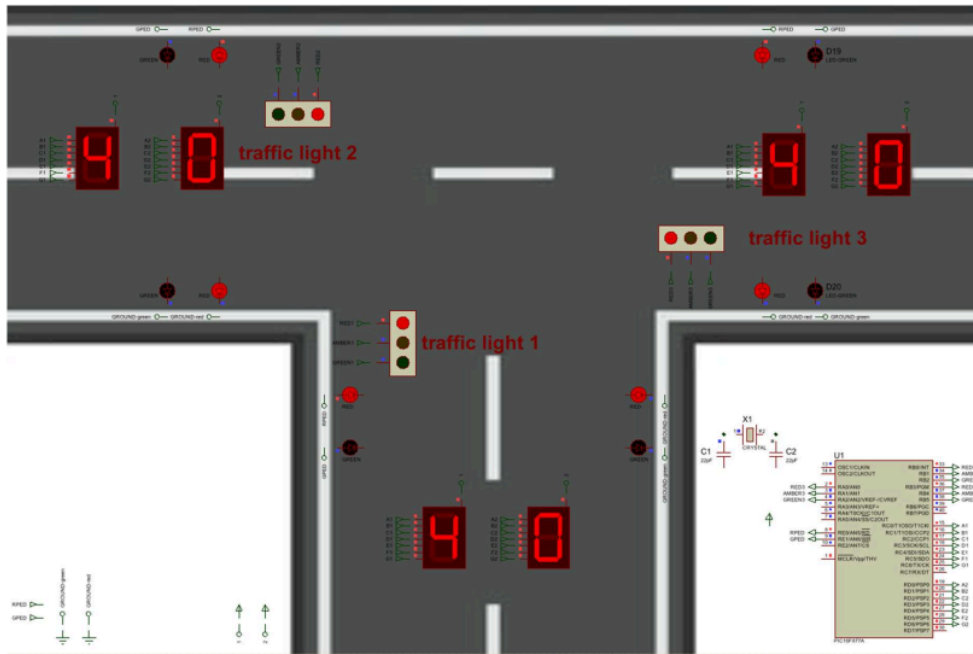
Delay for 1 second between each digit update.

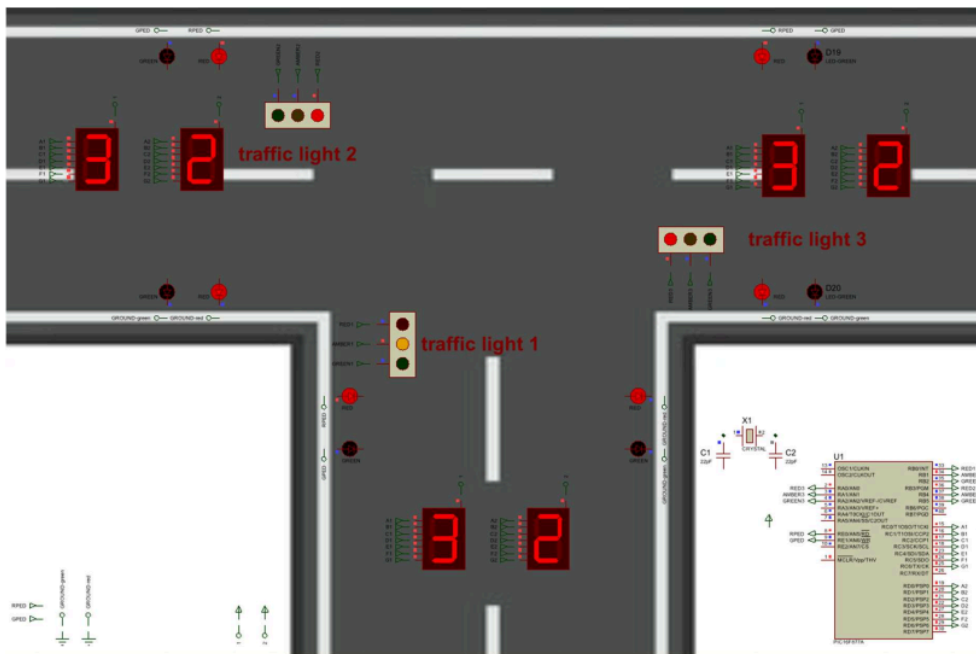
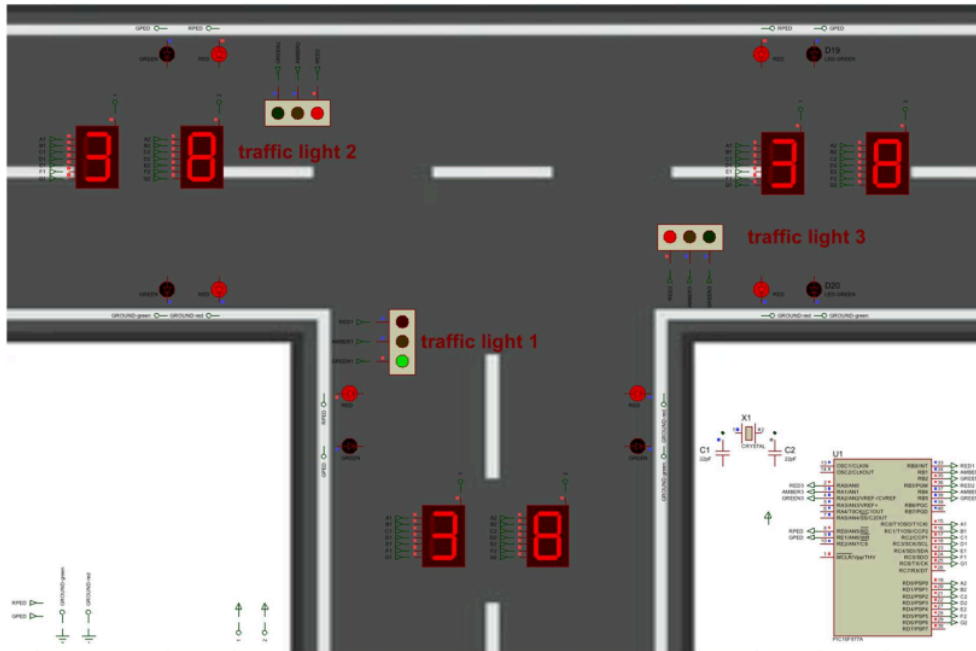
Return to the start of the loop.

The full program code listing can be viewed in Appendices A to E.

Simulation

Proteus





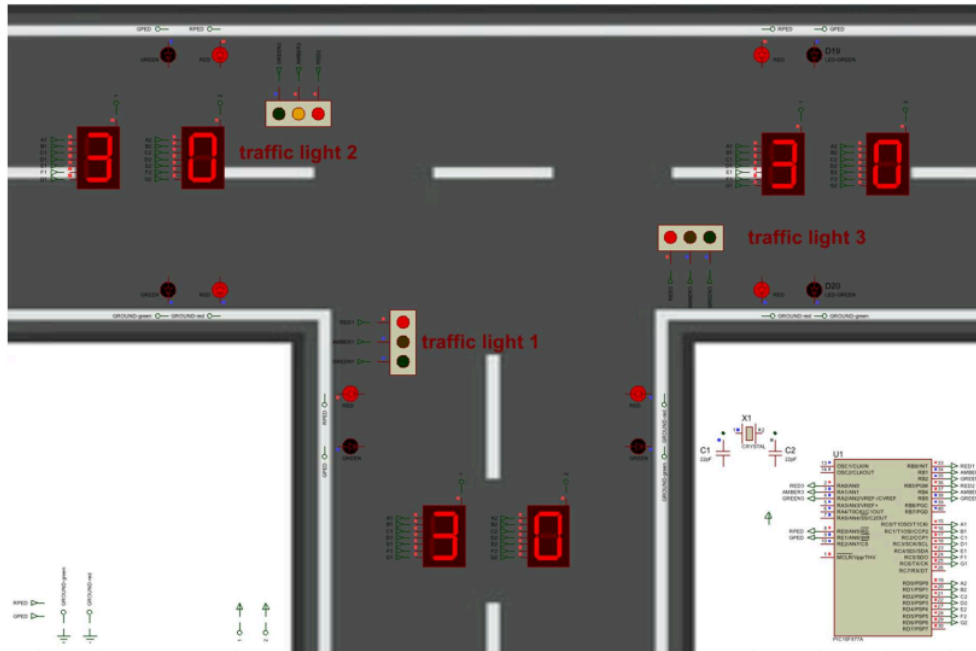


Figure 7: Phase 2

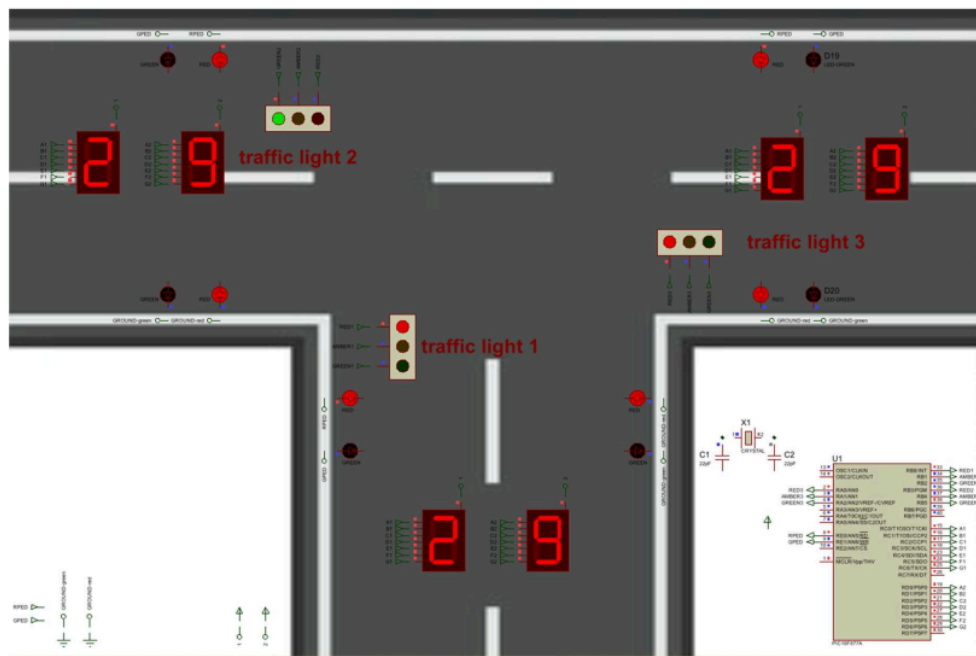


Figure 8: Phase 2

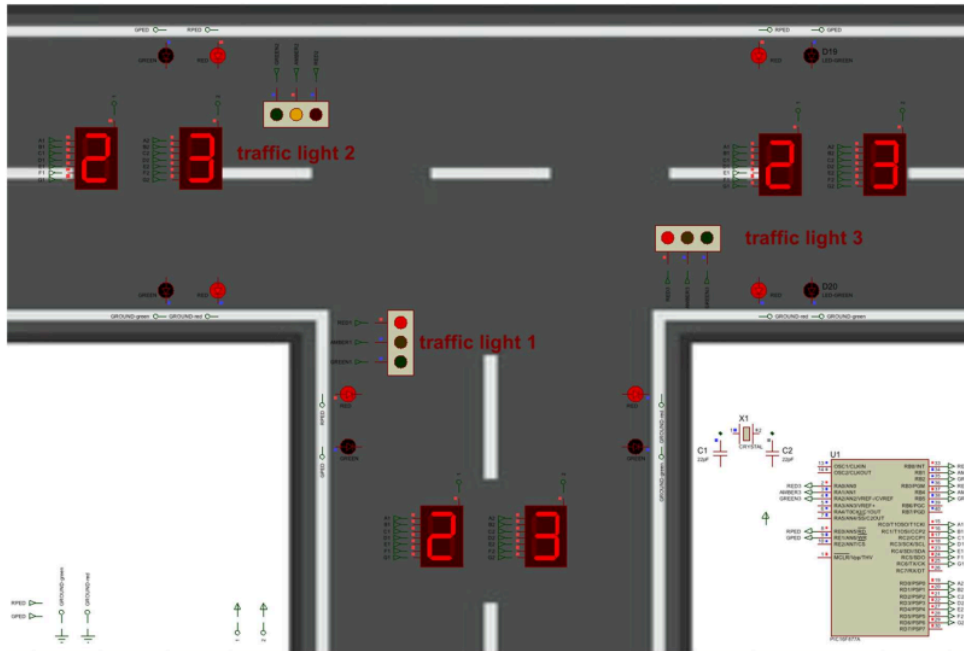


Figure 9: Phase 2

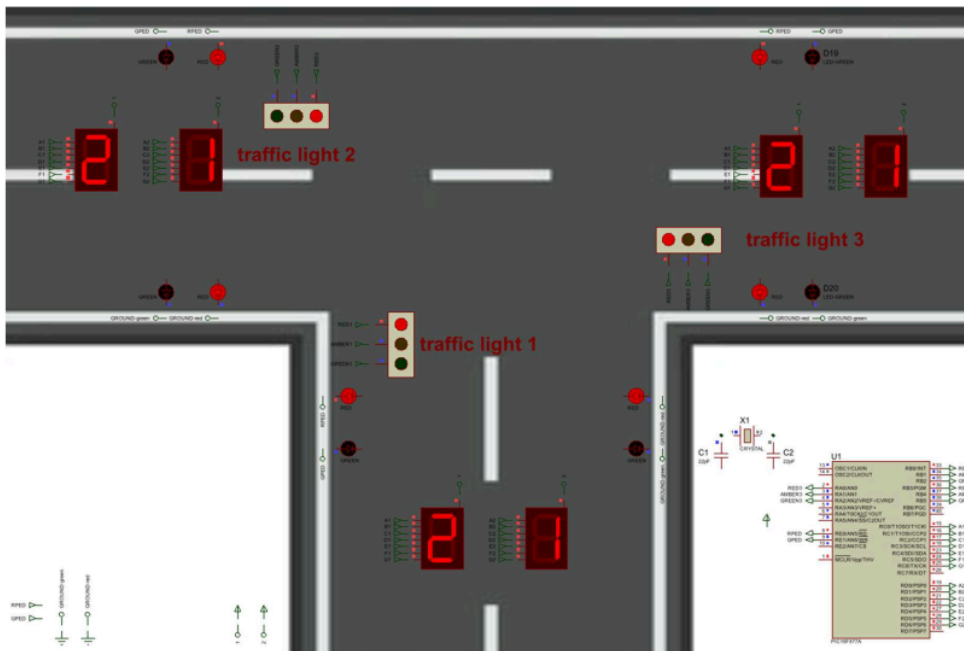


Figure 10: Phase 2

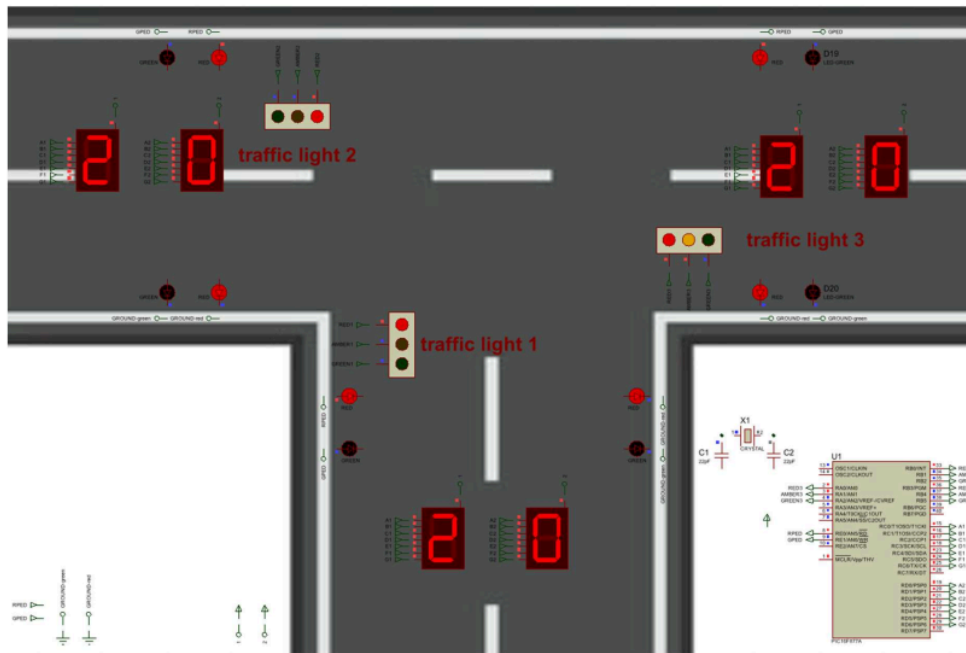


Figure 11: Phase 3

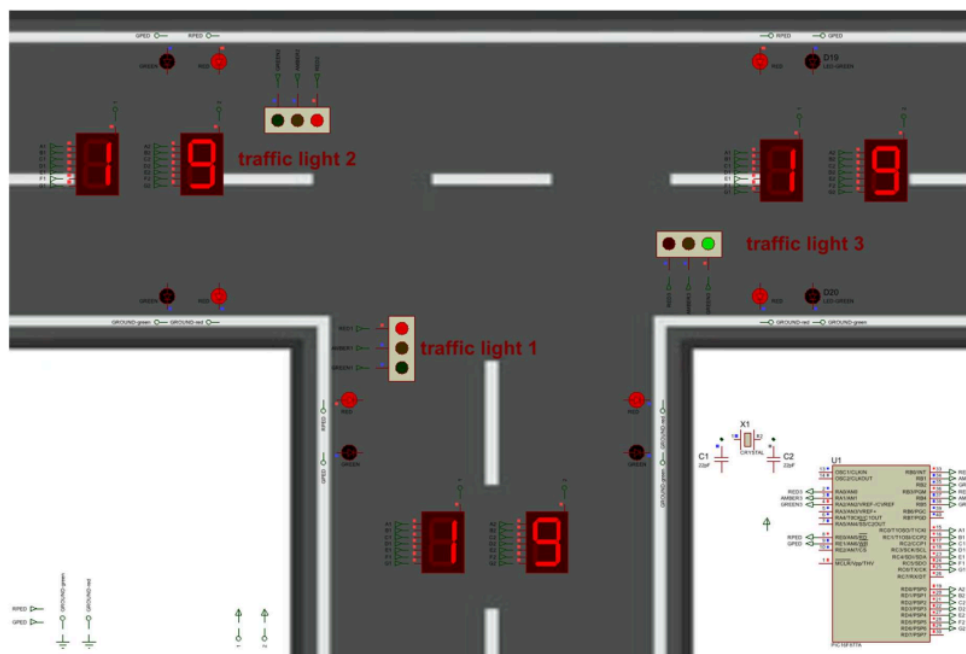


Figure 12: Phase 3

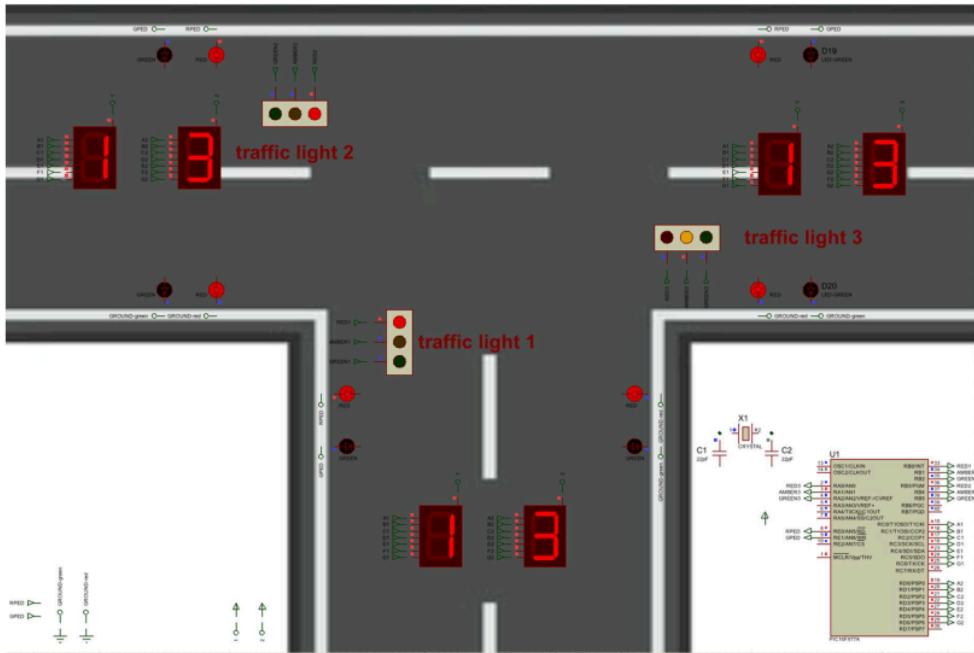


Figure 13: Phase 3

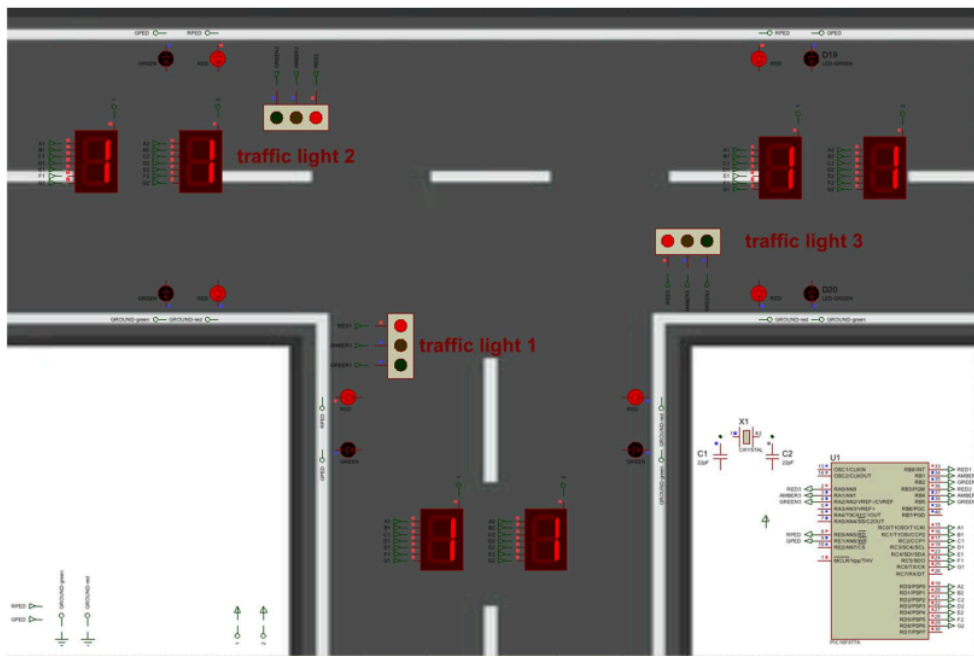


Figure 14: Phase 3

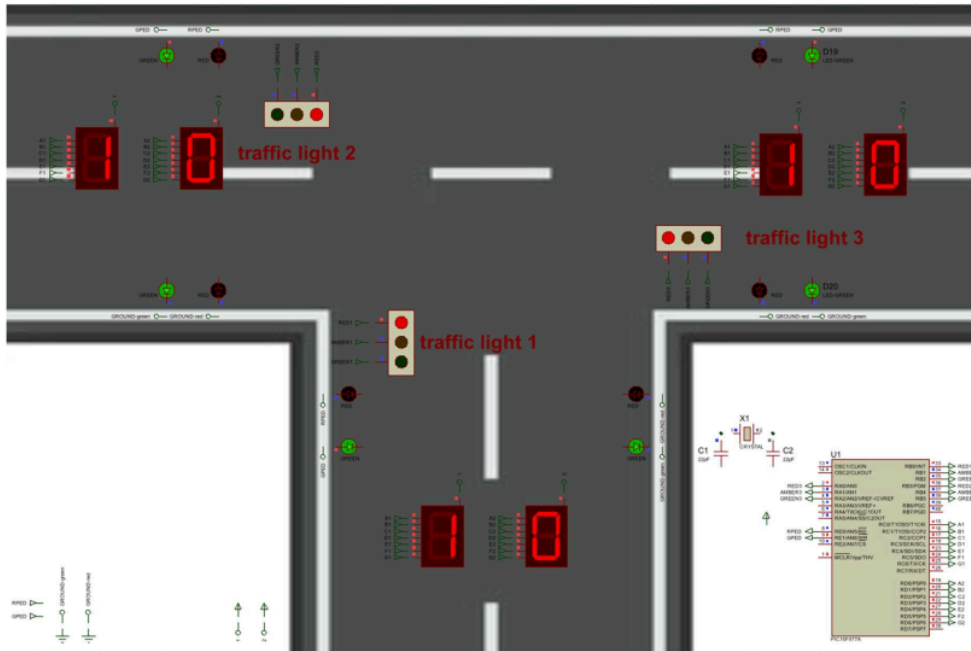


Figure 15: Pedestrian crossing phase

Breadboard

On the breadboard, the top right set of LEDs is traffic light 1, the bottom right set of LEDs is traffic light 2, and the middle set of LEDs is traffic light 3.

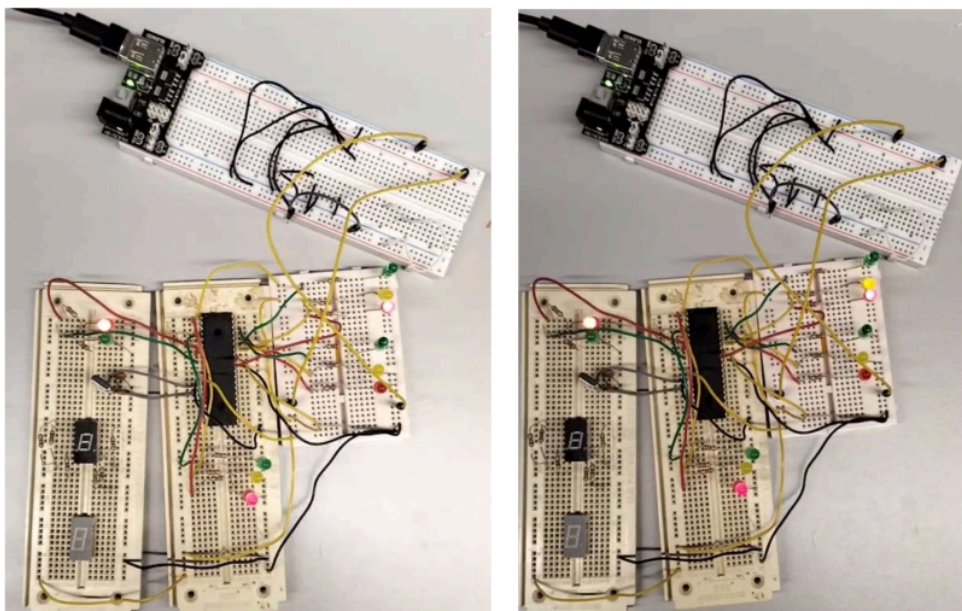


Figure 16: Phase 1 on the breadboard

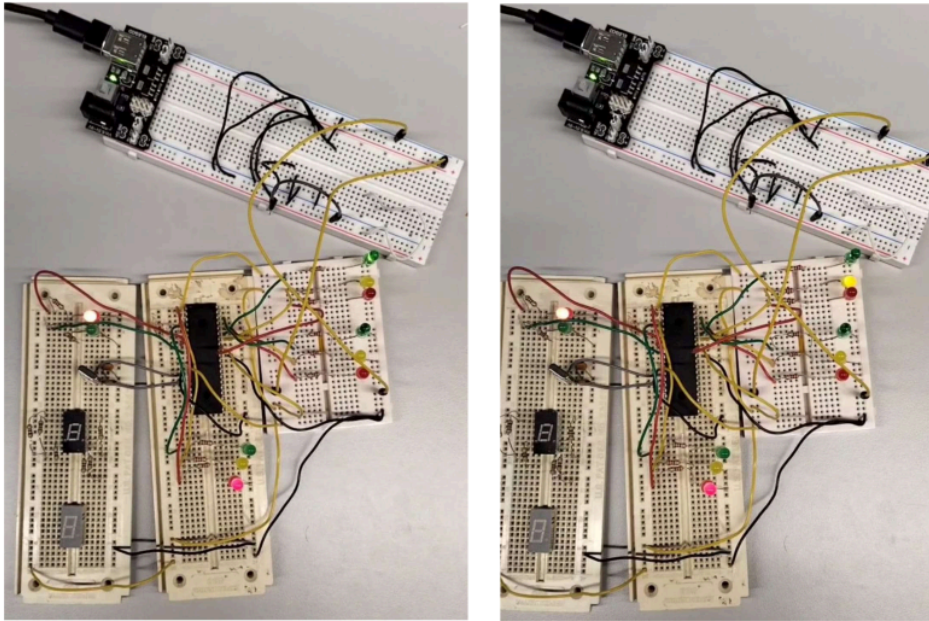


Figure 17: Phase 1 on the breadboard

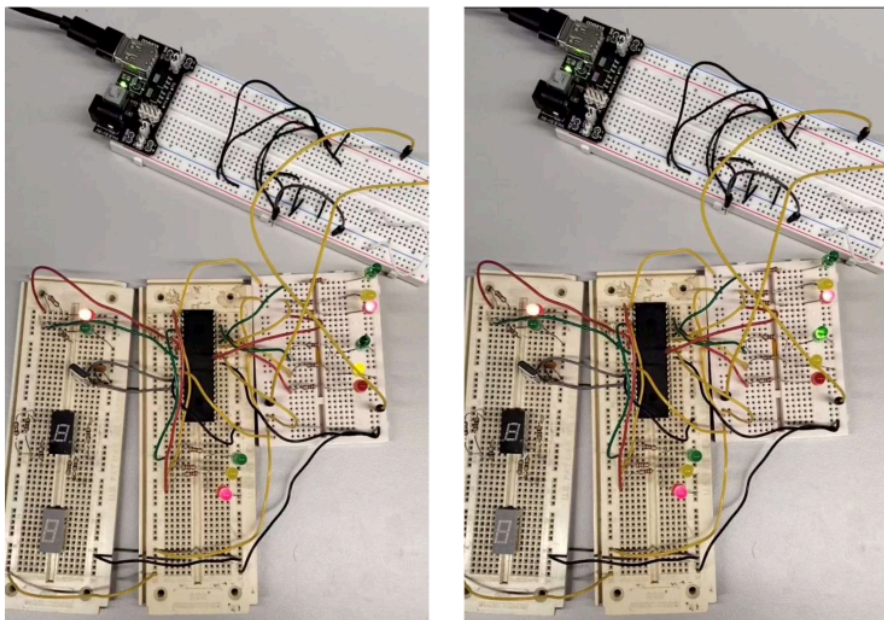


Figure 18: Phase 2 on the breadboard

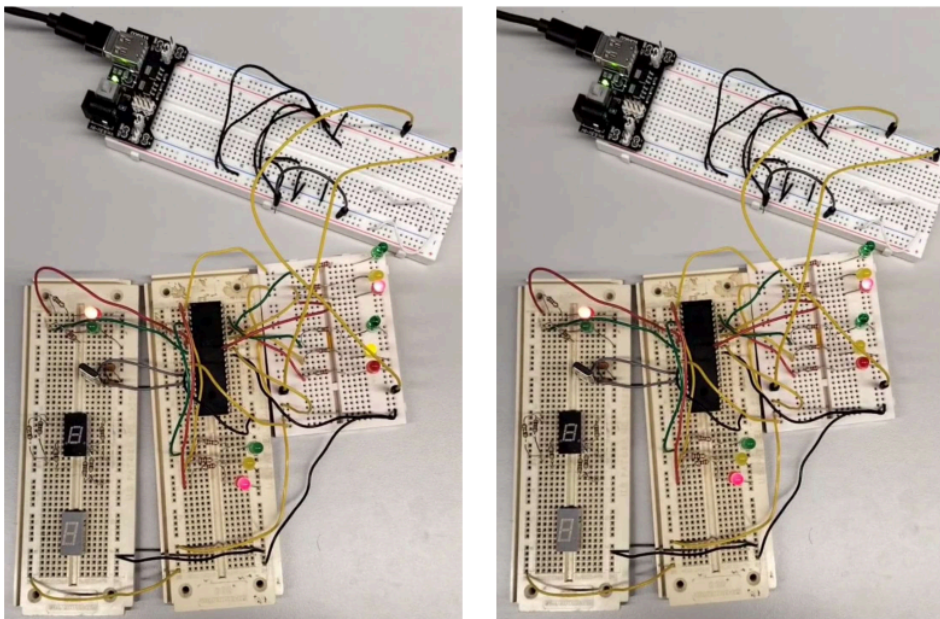


Figure 19: Phase 2 on the breadboard

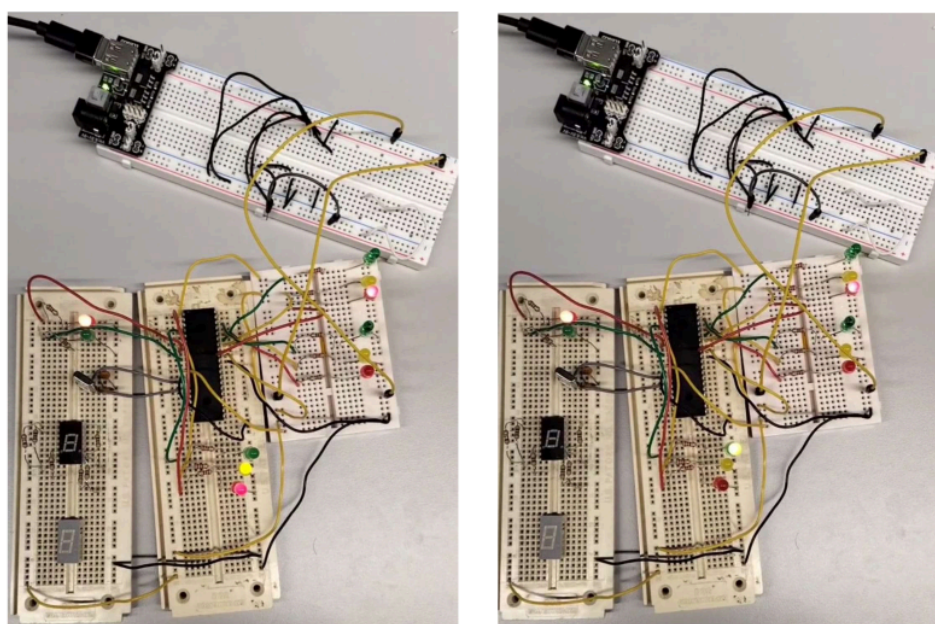


Figure 20: Phase 3 on the breadboard

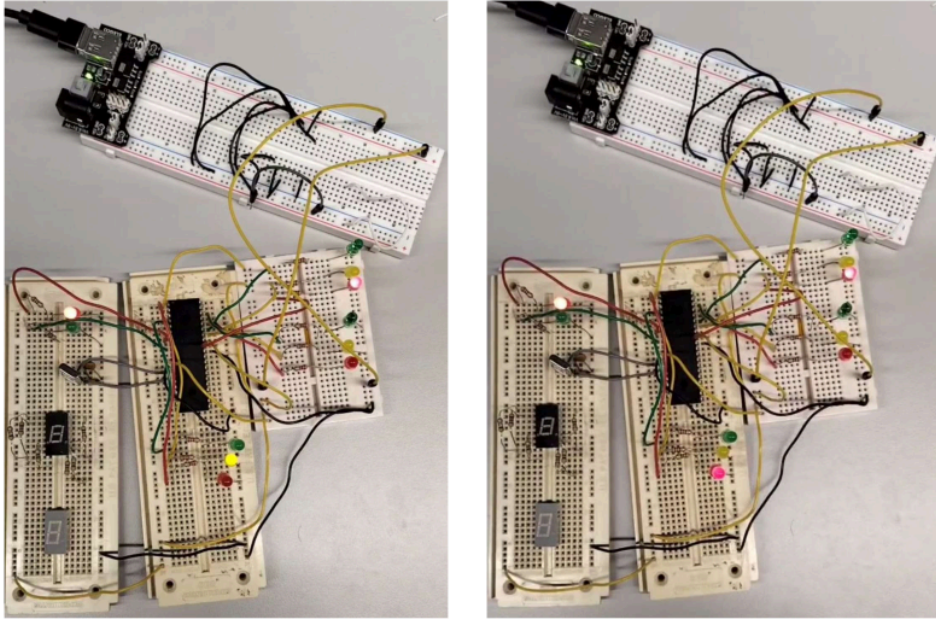


Figure 21: Phase 3 on the breadboard

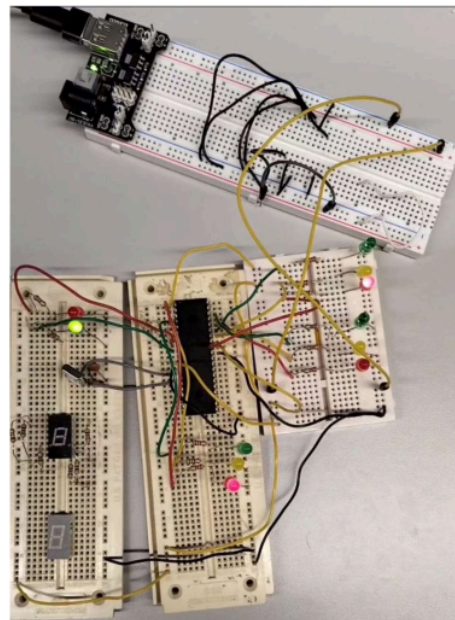


Figure 22: Pedestrian crossing phase

Results

Proteus

As shown in Figures 3-15, the simulated outputs were screenshotted to obtain the results.

In Figure 3, when the displays started the countdown at 40 seconds, the pedestrian red lights were signalled, and all the traffic lights signalled red for 1 second. In Figure 4, after the displays were updated to 39 seconds, traffic light 1 signalled red and amber together for 1 second. In Figure 5, after the displays updated to 38 seconds, traffic light 1 signalled green for 6 seconds. In Figure 6, after the displays updated to 32 seconds, traffic light 1 signalled amber for 2 seconds.

In Figure 7, after the displays updated to 30 seconds, traffic light 1 signalled red, and traffic light 2 signalled red and amber together for 1 second. In Figure 8, after the displays updated to 29 seconds, traffic light 2 signalled green for 6 seconds. In Figure 9, after the displays updated to 23 seconds, traffic light 2 signalled amber for 2 seconds. In Figure 10, after the displays updated to 21 seconds, traffic light 2 signalled red for 1 second.

In Figure 11, after the displays updated to 20 seconds, traffic light 3 signalled red and amber together for 1 second. In Figure 12, after the displays updated to 19 seconds, traffic light 3 signalled green for 6 seconds. In Figure 13, after the displays updated to 13 seconds, traffic light 3 signalled amber for 2 seconds. In Figure 14, after the displays updated to 11 seconds, traffic light 3 signalled red for 1 second.

In Figure 15, after the displays updated to 10 seconds, the pedestrian green lights were signalled for 10 seconds, and all the traffic lights stayed at red.

Breadboard

The results from the breadboard were pictured as shown in Figures 16-22. Note that the 7 segments were not included in the breadboard circuit; therefore, the results were only for the traffic lights' sequence. The timings for each phase can be confirmed with the attached video (sent along with the assignment files).

In Figure 16, for 1 second, the pedestrian red light was signalled, and two of the traffic lights signalled red, except for traffic light 2 (its red LED was not lighting up). Next, traffic light 1 signalled red and amber together for 1 second. In Figure 17, traffic light 1 signalled green for 6 seconds. Next, traffic light 1 signalled amber for 2 seconds.

In Figure 18, for 1 second, traffic light 1 signalled red, and traffic light 2 signalled amber. Next, traffic light 2 signalled green for 6 seconds. In Figure 19, traffic light 2 signalled amber for 2 seconds. There were no changes for the next second here.

In Figure 20, for 1 second, traffic light 3 signalled red and amber together. Next, traffic light 3 signalled green for 6 seconds. In Figure 21, traffic light 3 signalled amber for 2 seconds. Next, traffic light 3 signalled red for 1 second.

In Figure 22, the pedestrian green light was signalled for 10 seconds, and the working traffic lights' red lights stayed red.

The red LED of traffic light 2 was found to be faulty, which affected the traffic light sequencing. However, the overall program logic still operated correctly, as confirmed by the Proteus simulation.

Conclusion

One paper that was researched demonstrated how a PIC-based microcontroller was used at junctions by using IR sensors to analyse and measure traffic density. The data is then used to adjust signal duration across 'normal', 'jam', and 'soft' modes dynamically in real time. The design also implements emergency mode using an XBee wireless system supported by RF emitters to inform the detection of emergency vehicles such as ambulances, which then allows traffic lights to quickly change signals so that those vehicles can smoothly pass through congested junctions without delay. [9]

To conclude, this report demonstrated the successful completion of the fixed-time traffic light controller project for a T junction using the PIC 16F8477A, considering the primary method (using the Proteus design suite software). It showed the operation of a smooth traffic light sequencing for a T junction, assuming predictable traffic flow, and a safe pedestrian crossing phase, along with countdown displays of the total cycle.

However, when looking back at the challenges faced during the breadboard build, the old breadboard I had worked with had some poor or broken internal connections, so components had to be moved around, which took time. Additionally, one of the LEDs used was faulty.

Moreover, future recommendations would be to implement this project with functioning 7-segment displays on the breadboard, using a bigger breadboard to accommodate the remaining pedestrian lights and 7-segment displays, so that the breadboard prototype and simulation are similar to the design on Proteus.

References

- [1] "Types of Traffic Light Controllers: Fixed-Time, Actuated & Adaptive," traffic sign custom, 22nd May 2025. [Online]. Available: <https://trafficsigncustom.com/light-controllers/>. [Accessed 24th November 2025].
- [2] R. Surisetty and S. N. Sekhar, "Designing of a Traffic Signaling System at T-Intersection," *International Journal of Engineering Research and Application*, vol. 7, no. 4, pp. 82-86, 2017.
- [3] "MOVA," TRL Software, [Online]. Available: <https://trlsoftware.com/software/intelligent-signal-control/mova/>. [Accessed 24th November 2025].
- [4] "Traffic Signs Manual, Chapter 6, Traffic Control," GOV.UK, 2019. [Online]. Available: <https://assets.publishing.service.gov.uk/media/5df0e29fed915d15f42c4820/dft-traffic-signs-manual-chapter-6.pdf>. [Accessed 24th November 2025].
- [5] "PIC16F87XA Data Sheet," Microchip Technology, [Online]. Available: <https://ww1.microchip.com/downloads/en/devicedoc/39582b.pdf#page=3.08>. [Accessed 29th November 2025].
- [6] "PIC18 Microcontroller Family," Microchip Technology, [Online]. Available: <https://ww1.microchip.com/downloads/en/DeviceDoc/30327b.pdf#page=1.00>. [Accessed 29th November 2025].
- [7] "Crystal Oscillator: Circuit, Frequency & Working Principle," Electrical4U, [Online]. Available: <https://www.electrical4u.com/crystal-oscillator/>. [Accessed 1st December 2025].
- [8] "The Difference Between Common Anode and Common Cathode Displays," betlux, [Online]. Available: <https://betlux.com/the-difference-between-common-anode-and-common-cathode-displays/#:~:text=Common%20Anode%20displays%20are%20better,current%20directly%2C%20without%20additional%20drivers..> [Accessed 1st December 2025].
- [9] B. Ghazal, K. ElKhatib, K. Chahine and M. Kherfan, "Smart traffic light control system," *2016 Third International Conference on Electrical, Electronics, Computer Engineering and their Applications (EECEA)*, Beirut, Lebanon, pp. 140-145, 2016.

Appendices

```
#include <main.h>

void main()
{
    set_tris_b(0x00);    // set port B as output for traffic light 1 and 2
    set_tris_a(0x00);    // set port A as output for traffic light 3
    set_tris_e(0x00);    // set port E as output for pedestrian lights
    set_tris_c(0x00);    // set port C as output for 7-segment display (tens digit)
    set_tris_d(0x00);    // set port D as output for 7-segment display (ones digit)

    output_b(0x00);      // initialise port B to 0
    output_a(0x00);      // initialise port A to 0
    output_e(0x00);      // initialise port E to 0
    output_c(0x00);      // initialise port C to 0
    output_d(0x00);      // initialise port D to 0
}
```

Appendix A

```
while(TRUE)
{
    //phase1
    output_high(pin_b0);    // traffic light 1 red light ON
    output_high(pin_b3);    // traffic light 2 red light ON
    output_high(pin_a0);    // traffic light 3 red light ON
    output_low(pin_e1);     // pedestrian green light OFF
    output_high(pin_e0);    // pedestrian red light ON
    output_c(0x99);         // display '4' (tens digit) on the 7-segment display
    output_d(0xc0);         // display '0' (ones digit) on the 7-segment display
    delay_ms(1000);        // add a delay of 1 second

    output_high(pin_b1);    // traffic light 1 amber light ON
    output_c(0xb0);         // display '3' (tens digit) on the 7-segment display
    output_d(0x90);         // display '9' (ones digit) on the 7-segment display
    delay_ms(1000);        // add a delay of 1 second

    output_low(pin_b0);     // traffic light 1 red light OFF
    output_low(pin_b1);     // traffic light 1 amber light OFF
    output_high(pin_b2);    // traffic light 1 green light ON
    output_d(0x80);         // display '8' (ones digit) on the 7-segment display
    delay_ms(1000);        // add a delay of 1 second
    output_d(0xf8);         // display '7' (ones digit) on the 7-segment display
    delay_ms(1000);        // add a delay of 1 second
    output_d(0x82);         // display '6' (ones digit) on the 7-segment display
    delay_ms(1000);        // add a delay of 1 second
    output_d(0x92);         // display '5' (ones digit) on the 7-segment display
    delay_ms(1000);        // add a delay of 1 second
    output_d(0x99);         // display '4' (ones digit) on the 7-segment display
    delay_ms(1000);        // add a delay of 1 second
    output_d(0xb0);         // display '3' (ones digit) on the 7-segment display
    delay_ms(1000);        // add a delay of 1 second

    output_high(pin_b1);    // traffic light 1 amber light ON
    output_low(pin_b2);     // traffic light 1 green light OFF
    output_d(0xa4);         // display '2' (ones digit) on the 7-segment display
    delay_ms(1000);        // add a delay of 1 second
    output_d(0xf9);         // display '1' (ones digit) on the 7-segment display
    delay_ms(1000);        // add a delay of 1 second
}
```

Appendix B

```

//phase2
output_low(pin_b1);      // traffic light 1 amber light OFF
output_high(pin_b0);     // traffic light 1 red light ON
output_high(pin_b4);     // traffic light 2 amber light ON
output_d(0XC0);          // display '0' (ones digit) on the 7-segment display
delay_ms(1000);          // add a delay of 1 second

output_low(pin_b3);      // traffic light 2 red light OFF
output_low(pin_b4);      // traffic light 2 amber light OFF
output_high(pin_b5);     // traffic light 2 green light ON
output_c(0xA4);          // display '2' (tens digit) on the 7-segment display
output_d(0X90);          // display '9' (ones digit) on the 7-segment display
delay_ms(1000);          // add a delay of 1 second
output_d(0X80);          // display '8' (ones digit) on the 7-segment display
delay_ms(1000);          // add a delay of 1 second
output_d(0XF8);          // display '7' (ones digit) on the 7-segment display
delay_ms(1000);          // add a delay of 1 second
output_d(0X82);          // display '6' (ones digit) on the 7-segment display
delay_ms(1000);          // add a delay of 1 second
output_d(0X92);          // display '5' (ones digit) on the 7-segment display
delay_ms(1000);          // add a delay of 1 second
output_d(0X99);          // display '4' (ones digit) on the 7-segment display
delay_ms(1000);          // add a delay of 1 second

output_high(pin_b4);     // traffic light 2 amber light ON
output_low(pin_b5);      // traffic light 2 green light OFF
output_d(0XB0);          // display '3' (ones digit) on the 7-segment display
delay_ms(1000);          // add a delay of 1 second
output_d(0XA4);          // display '2' (ones digit) on the 7-segment display
delay_ms(1000);          // add a delay of 1 second

output_high(pin_b3);     // traffic light 2 red light ON
output_low(pin_b4);      // traffic light 2 amber light OFF
output_d(0XF9);          // display '1' (ones digit) on the 7-segment display
delay_ms(1000);          // add a delay of 1 second

```

Appendix C

```

//phase3
output_high(pin_a1); // traffic light 3 amber light ON
output_d(0XC0); // display '0' (ones digit) on the 7-segment display
delay_ms(1000); // add a delay of 1 second

output_low(pin_a0); // traffic light 3 red light OFF
output_low(pin_a1); // traffic light 3 amber light OFF
output_high(pin_a2); // traffic light 3 green light ON
output_c(0XF9); // display '1' (tens digit) on the 7-segment display
output_d(0X90); // display '9' (ones digit) on the 7-segment display
delay_ms(1000); // add a delay of 1 second
output_d(0X80); // display '8' (ones digit) on the 7-segment display
delay_ms(1000); // add a delay of 1 second
output_d(0XF8); // display '7' (ones digit) on the 7-segment display
delay_ms(1000); // add a delay of 1 second
output_d(0X82); // display '6' (ones digit) on the 7-segment display
delay_ms(1000); // add a delay of 1 second
output_d(0X92); // display '5' (ones digit) on the 7-segment display
delay_ms(1000); // add a delay of 1 second
output_d(0XA4); // display '4' (ones digit) on the 7-segment display
delay_ms(1000); // add a delay of 1 second

output_high(pin_a1); // traffic light 3 amber light ON
output_low(pin_a2); // traffic light 3 green light OFF
output_d(0XB0); // display '3' (ones digit) on the 7-segment display
delay_ms(1000); // add a delay of 1 second
output_d(0XA4); // display '2' (ones digit) on the 7-segment display
delay_ms(1000); // add a delay of 1 second

output_high(pin_a0); // traffic light 3 red light ON
output_low(pin_a1); // traffic light 3 amber light OFF
output_d(0XF9); // display '1' (ones digit) on the 7-segment display
delay_ms(1000); // add a delay of 1 second

```

Appendix D

```

//pedestrian crossing phase
output_low(pin_e0); // pedestrian red light OFF
output_high(pin_e1); // pedestrian green light ON

output_d(0XC0); // display '0' (ones digit) on the 7-segment display
delay_ms(1000); // add a delay of 1 second
output_c(0XC0); // display '0' (tens digit) on the 7-segment display
output_d(0X90); // display '9' (ones digit) on the 7-segment display
delay_ms(1000); // add a delay of 1 second
output_d(0X80); // display '8' (ones digit) on the 7-segment display
delay_ms(1000); // add a delay of 1 second
output_d(0XF8); // display '7' (ones digit) on the 7-segment display
delay_ms(1000); // add a delay of 1 second
output_d(0X82); // display '6' (ones digit) on the 7-segment display
delay_ms(1000); // add a delay of 1 second
output_d(0X92); // display '5' (ones digit) on the 7-segment display
delay_ms(1000); // add a delay of 1 second
output_d(0X99); // display '4' (ones digit) on the 7-segment display
delay_ms(1000); // add a delay of 1 second
output_d(0XB0); // display '3' (ones digit) on the 7-segment display
delay_ms(1000); // add a delay of 1 second
output_d(0XA4); // display '2' (ones digit) on the 7-segment display
delay_ms(1000); // add a delay of 1 second
output_d(0XF9); // display '1' (ones digit) on the 7-segment display
delay_ms(1000); // add a delay of 1 second

}

```

Appendix E