

Final

by Gomer Morgan-Gray

Submission date: 05-Dec-2025 10:04AM (UTC+0000)

Submission ID: 269702412

File name: 50068_Gomer_Morgan-Gray_Final_634321_34309776.docx (2.46M)

Word count: 3541

Character count: 19559

L5 - Electronics I (EEE version)
(ACDB5002)

Gomer Morgan-Gray
2316794

Contents

Introduction	3
Background Research	4
Design.....	5
Component selection	5
Proteus Simulations	7
Breadboard Testing	8
Operational Logic	9
Sequencing.....	9
Pseudo code	9
Variable Table	11
C Program.....	11
Function (RTCC_isr)	11
Function (DispNumber)	12
Function (DispTrafficLights)	12
Main.....	13
Convenience features	14
System Validation & Results	15
Conclusion & Recommended Future Work	16
References	18
Table Of Figures	20
Appendix.....	21
Appendix A Flow Chart.....	21

Introduction

This report details the development, implementation and testing of a traffic light control system for a T-Junction. Effective management of a junction is essential for road safety and efficient traffic flow. The system will be built to control the flow of both traffic and pedestrians. The goal of the project was to create a traffic light control system that can manage multiple traffic lights whilst accommodating pedestrian requests.

The control system is designed to control three traffic lights, each containing red, amber and green signals to regulate traffic from all directions of the T-Junction. Correct sequencing of these signals is required to minimize congestion, prevent accidents and maintain a smooth traffic flow. In addition to the car control the system includes provisions for pedestrian crossings. This includes red and green pedestrian lights a push button for making crossing requests and a blue light for indicating that a crossing request has been received.

An additional feature to the system is the inclusion of a pair of seven segment displays, which provide a visual count down timer starting from 40 seconds counting down to zero before the cycle restarts. The timer gives clear feedback to pedestrians indicating where in the cycle the system is.

The controller will initially be tested making use of proteus a circuit design and simulation software. Making use of the software the circuit was able to be prototyped, the software to be used on the controller could also be developed to verify the effectiveness of the design including correct sequencing of the lights, response to the pedestrian crossing request button and accuracy of the countdown timer.

Background Research

The requirement of signalized junctions cannot be understated; they play a crucial role in improving traffic flow and safety at junction. By introducing traffic lights, junctions can more effectively manage traffic, by reducing the amount of “stop-and-go” driving. This reduction in repeated braking and accelerations leading to an improvement in fuel efficiency and overall vehicle emissions. Studies have shown that the addition of signals to a junction allow vehicles to maintain a more consistent speed [1].

In addition to the environmental benefits, it has been shown that signalized junctions can enhance safety. This is primarily due to the removal of human error as the decision of when to drive on is removed from the driver of the vehicle [2].

However, the introduction of signalized junctions necessitates the use of an automated control system to manage the traffic signals. There are 3 main approaches to automating traffic signals, The first and most simplistic method being fixed time control, this is where the sequencing and duration of the traffic light phases are fixed and pre-defined based on the program running within the controller [3].

The second approach to control is Actuated, this is where the controller gathers information from a range of different sensors to control the signals. The controller can dynamically alter which phase to change to next and adapt phase durations to best suit traffic conditions. Several sensor types are utilised for this purpose such as inductive ground loops, infra-red sensors and Computer vision cameras, all these however serve the same purpose of indicating where traffic is waiting [4, 5].

The Final type of traffic light controller is Adaptive; this type of controller makes use of real time traffic data to adjust signal timings appropriate. This is commonly used in areas with high levels of congestion. In situations like these the controller can be setup to adjust timing and sequencing to help reduce traffic and waiting times [6].

The timing and sequencing of the lights is crucial both for optimizing traffic flow and ensuring the safe operation of the junction. The time between the end of Green on one approach and the start on another is known as the “intergreen” Period. In the UK the amber period should be 3 seconds and the starting amber period (Red with amber) at 2 seconds [7].

As the controller proposed in this report is required to implement a pedestrian crossing phase also, additional safety considerations need to be made. The minimum allowed time for a pedestrian crossing phase in the UK is 4 seconds and will need to be considered whilst setting the phase timings in the controller [8].

A wide variety of commercially available controllers exist on the market, each providing different features. A commonly used controller is the ST950 Manufacturer by siemens.

This controller makes use of a 32-bit arm microcontroller and support a maximum of 32 phases and 240 digital inputs or outputs [9].

Design

Component selection

Microcontroller

The first and central decision during the design phase was which microcontroller to utilize. Reviewing possible options, it was decided that a Microchip PIC16F877A would be used for this task. This is a popular 8-bit microcontroller contains 368 bytes of memory, 8KB of flash data memory and operates at 20MHz. This MCU is commonly used within control and timing applications such as the traffic light controller in this case. The decision has been made for several reasons:

Sufficient I/O For the application:

The PIC16F877A provide 33 digital I/O pins, this allows for control of all 9 traffic signals, pedestrian crossings and countdown displays, without the requirement of I/O expanders [10].

Built-in timers:

For this application within a traffic light controller accurate timing is crucial for operation, The chosen MCU contains 3 hardware timers allowing for the use interrupts within the code [10].

The timers will also be taken advantage of for multiplexing the 7 segment displays as the software will be called repeatedly at a fixed interval.

Adequate performance:

The PIC16F877A Provides the necessary computational performance. The use of a faster 32-bit microcontroller is not required for this application as computationally heavy processes are not necessary for this application [11].

Low Cost & Wide availability:

The PIC is significantly more favourable in cost as compared to alternatives 32-bit microcontrollers.

The microcontroller is also widely available from typical electronic component suppliers such as Farnell, Mouser and DigiKey and is still in production.

Other microcontrollers were also considered such as the STM32F103 however was not chosen due to operating at 3.3V, this could be incompatible with the other components to be used. The benefits higher performance is also not required for this application [12].

Also considered was the intel 8051, this microcontroller has a long history of use within industrial control applications, however, significantly lacks more modern peripherals, with very limited timers and digital I/O, this would have necessitated the use of an I/O expander increasing complexity and cost [13].

Countdown Display

The next decision to be made was the type of display to be used for the countdown timer. As the only requirement for the display was numbers 7-segment displays were chosen, this is due to their simplicity and ease of implementation not needing a communication interface.

A common anode 2 display pair was chosen as the display shall count down from 40, Due to the display being a pair the digits are connected in parallel shown in Figure 1, This is more desirable over a pair of standard 7 segment displays as it makes use of less I/O Pins allowing for future expandability of the system. However, this necessitates the use of Multiplexing to display separate characters on each digit [14].

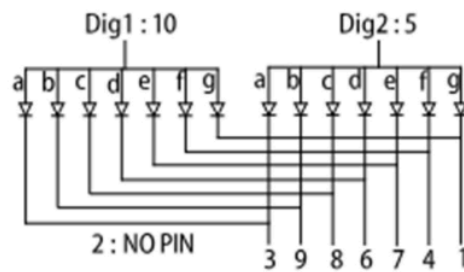


Figure 1 - Common Anode 7 Segment display pair Wiring

Proteus Simulations

The first stage of the creating the circuit within proteus was to create a block diagram of the circuit as shown in Figure 2. This allowed for easier implementation for the actual circuit diagram.

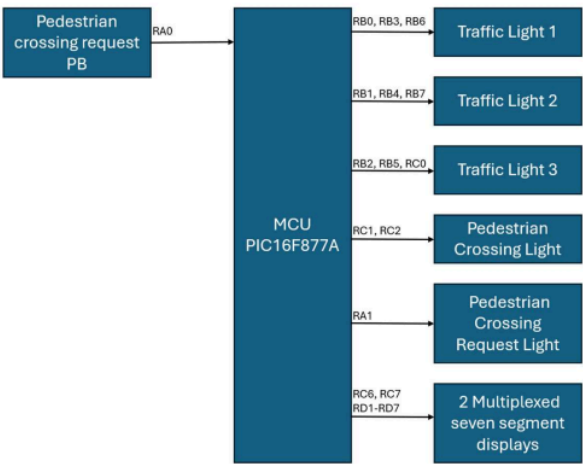


Figure 2 - System Block Diagram

Shown in Figure 3 The pic 16F877A was placed making use of net tags the connections could be made to the necessary components whilst keeping the circuit neat by minimizing crossing wires. A 20Mhz crystal resonator and load capacitors were also added as the clock source for the MCU.

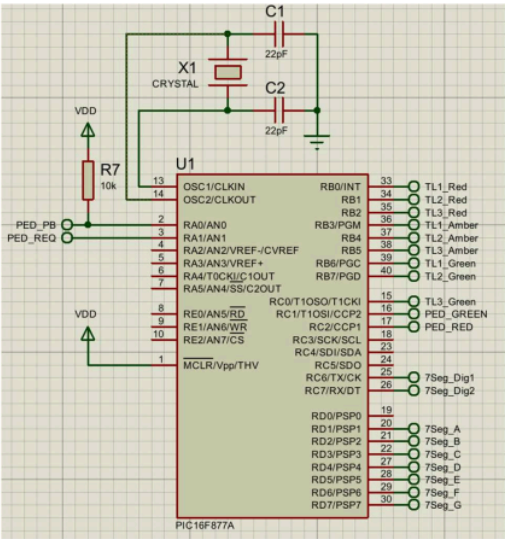


Figure 3 - Proteus PIC16F877A Connections

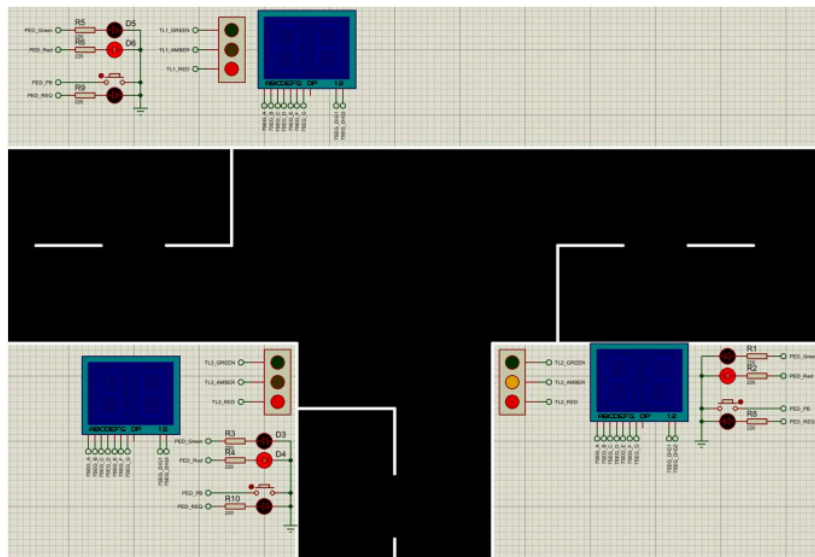


Figure 4 - Proteus Traffic Lights, Timers and Pedestrian crossings

Breadboard Testing

With the circuit completed within the simulations. Testing on physical hardware was now desired to analyze the feasibility of the controller. This was achieved using breadboards, these allow for rapid prototyping as soldering is not necessary.

The real-world test as shown in Figure 5 made use of 2 standard breadboards, several complementary components that were omitted during the simulation had to be added such as resistors and capacitors due to real world limitations.

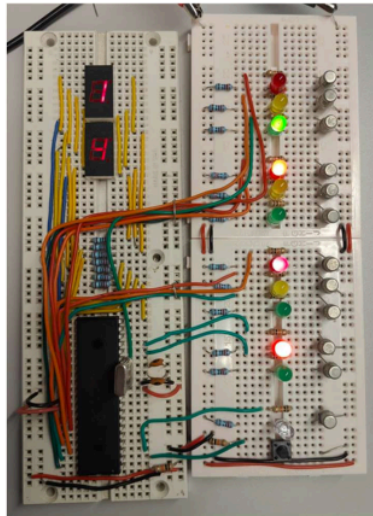


Figure 5 - Controller Breadboard Prototype

Operational Logic

Sequencing

Table 1 - Traffic Light sequencing Pattern

Phase	Duration	Traffic Light 1			Traffic Light 2			Traffic Light 3			Pedestrian Crossing	
		Red	Amber	Green	Red	Amber	Green	Red	Amber	Green	Red	Green
1	2	ON	ON	OFF	ON	OFF	OFF	ON	OFF	OFF	ON	OFF
2	6	OFF	OFF	ON	ON	OFF	OFF	ON	OFF	OFF	ON	OFF
3	2	OFF	OFF	OFF	ON	OFF	OFF	ON	OFF	OFF	ON	OFF
4	2	ON	OFF	OFF	ON	ON	OFF	ON	OFF	OFF	ON	OFF
5	6	OFF	OFF	OFF	OFF	OFF	ON	ON	OFF	OFF	ON	OFF
6	2	OFF	OFF	OFF	OFF	ON	OFF	ON	OFF	OFF	ON	OFF
7	2	OFF	OFF	OFF	ON	OFF	OFF	ON	ON	OFF	ON	OFF
8	6	OFF	OFF	OFF	ON	OFF	OFF	OFF	OFF	ON	ON	OFF
9	2	OFF	OFF	OFF	ON	OFF	OFF	OFF	ON	OFF	ON	OFF
10	10	OFF	OFF	OFF	ON	OFF	OFF	ON	OFF	OFF	OFF	ON

The sequencing of the lights is crucial to ensure safe operation of the intersection. The sequence as shown in Table 1 - Traffic Light sequencing Pattern was decided upon before integrating in code to understand key feature necessary for recreation.

Pseudo code

The software was first written in pseudo code. This was done to be able to quickly implement the software in the final C language to be used on the microcontroller.

MAIN Function:

Configure Timer0 to call RTCC Interrupt every 409uS

While TRUE:

 IF TimerCount > 0:

 Subtract 1 from TimerCount

 Else:

 Set TimerCount to 40

 IF CurrentPhaseDuration >= TrafficLightPattern[CurrentTrafficPhase][1]:

 IF PedestrianRequest TRUE AND Current Phase Can be changed to pedestrian:

 Set TimerCount to 10

 Set PedestrianRequest to False

 Set CurrentTrafficPhase to 9

 Set CurrentPhaseDuration to 0

 Else:

```
        Add 1 to currentTrafficPhase
        Set CurrentPhaseDuration to 0

    IF CurrentTrafficPhase > 9:
        Set CurrentTrafficPhase to 0
        Set CurrentPhaseDuration to 0
        Call DispTrafficLights(TrafficLightPattern[CurrentTrafficPhase][0])
        Add 1 to CurrentPhaseDuration
```

END MAIN

FUNCTION DispNumber(seg1, seg2):

```
IF Digit True:
    Set Digit to False
    Disable Segment 2
    Enable segment 1
    Output Bit pattern for seg1 to Port D
Else:
    Set Digit to True
    Disable Segment 1
    Enable segment 2
    Output Bit pattern for seg2 to Port D
```

END Function

FUNCTION DispTrafficLights(Lights):

```
Output TL1_Red = Lights AND bitmask for TL1 Red
Output TL2_Red = Lights AND bitmask for TL2 Red
Output TL3_Red = Lights AND bitmask for TL3 Red
Output TL1_Amber = Lights AND bitmask for TL1 Amber
Output TL2_Amber = Lights AND bitmask for TL2 Amber
Output TL3_Amber = Lights AND bitmask for TL3 Amber
Output TL1_Green = Lights AND bitmask for TL1 Green
Output TL2_Green = Lights AND bitmask for TL2 Green
```

Output TL3_Green = Lights AND bitmask for TL3 Green

Output PED_Red = Lights AND bitmask for Pedestrian Red

Output PED_Green = Lights AND bitmask for Pedestrian Green

END FUNCTION

Variable Table

Table 2 - Variable Table

Variable name	Data Type	Use
digit	Boolean	Stores which 7 segment display to display to
PedestrianRequest	Boolean	Indicates that there is currently a pedestrian crossing request
TimerCount	Integer	Stores the current count of the timer
CurrentPhaseDuration	Integer	Stores the elapsed duration of the current phase
CurrentTrafficPhase	Integer	Stores the current Traffic phase to be show
segCharacters	Integer Array	Stores bit patterns for numbers on the seven-segment display
TrafficLightPattern	Long Array 2D	Stores the traffic sequence pattern and duration for each phase

C Program

The program was broken up into several different functions for handling key functions. This was done to ensure modularity of the code and allow for simplifying repetitive sections of code. These functions and their use are as follows:

Function (RTCC_isr)

Shown in Figure 6 is the RTCC interrupt function, this is a void function returning no data. This function is called by the hardware timer within the PIC16F877A at a fixed 409uS interval this value was chosen as it allowed for no visible flickering in the display.

The use of this function is to continuously check the state of the pedestrian push button and set the “PedestrianRequest” Flag high if pressed. It is also used to repeatedly call the “DispNumber” function whilst also passing in the 10’s place digit and 1’s place of the countdown timer to be displayed.

```
#INT_RTCC
void RTCC_isr(void)
{
    DispNumber((TimerCount / 10), (TimerCount % 10));
    if(!input(PedestrianRequestPB)){PedestrianRequest = TRUE;}
}
```

Figure 6 - RTCC Interrupt Function Code

Function (DispNumber)

Shown in Figure 7 is the DispNumber Function this is again a void function, however this variable takes 2 parameters “seg1” and “seg2” these are both integers and have the purpose of storing the number to be displayed on the 7 segment display.

The purpose of this function is to display the given numbers on the seven-segment display. Due to the nature of the seven-segment display being a display pair multiplexing is required. Multiplexing is where only 1 display is active at a time however switches between both at a rapidly taking advantage of persistence of vision for both to appear solid [15].

```
void DispNumber(int seg1, int seg2 ){
    if(digit){
        digit = FALSE;
        output_low(Seg_Dig2);
        output_high(Seg_Dig1);
        output_d((segCharacters[seg1] << 1));
    }else{
        digit = TRUE;
        output_low(Seg_Dig1);
        output_high(Seg_Dig2);
        output_d((segCharacters[seg2] << 1));
    }
}
```

Figure 7 - DispNumber Function Code

Function (DispTrafficLights)

```
void DispTrafficLights(long Lights){
    output_bit(TL1_Red, Lights & 0B0000000000000001);
    output_bit(TL2_Red, Lights & 0B0000000000000010);
    output_bit(TL3_Red, Lights & 0B0000000000000100);
    output_bit(TL1_Amber, Lights & 0B0000000000001000);
    output_bit(TL2_Amber, Lights & 0B0000000000010000);
    output_bit(TL3_Amber, Lights & 0B0000000000100000);
    output_bit(TL1_Green, Lights & 0B0000000010000000);
    output_bit(TL2_Green, Lights & 0B0000000100000000);
    output_bit(TL3_Green, Lights & 0B0000001000000000);
    output_bit(PED_Red, Lights & 0B0000010000000000);
    output_bit(PED_Green, Lights & 0B0000100000000000);
}
```

Figure 8 - DispTrafficLights Function

The DispTrafficLights shown in Figure 8 is a void function returning no data and takes 1 parameter “Lights” this is a long data type, the purpose of which is to store the current traffic sequences light bit pattern.

The purpose of this function is to set the output pins on the microcontroller when required. This is done by making use of bitmasks which separate the individual bits of the “Lights” variable reducing the number of separate variables by storing the state of all the lights within one variable as shown in Figure 9.

```
long TrafficLightPattern[10][2] = {
  { 0b000001000001111, 2}, // Traffic Light 1 prepare to go
  { 0b000001001000110, 6}, // Traffic Light 1 GO
  { 0b000101000001110, 2}, // Traffic Light 1 Prepare to stop
  { 0b000001000010111, 2}, // Traffic Light 2 prepare to go
  { 0b000001010000101, 6}, // Traffic Light 2 GO
  { 0b000101000010101, 2}, // Traffic Light 2 Prepare to stop
  { 0b000001000100111, 2}, // Traffic Light 3 prepare to go
  { 0b000001100000011, 6}, // Traffic Light 3 GO
  { 0b000101000100011, 2}, // Traffic Light 3 Prepare to stop
  { 0b000010000000111, 11}, // Pedestrian Crossing
}; // ^^^^^^^^^^^^^^^^^ ^
//   XXX| | | | | | | | | |
//   | | | | | | | | | | Phase duration (n * Main Loop Delay)
//   | | | | | | | | | | Traffic Light 1 Red
//   | | | | | | | | | | Traffic Light 2 Red
//   | | | | | | | | | | Traffic Light 3 Red
//   | | | | | | | | | | Traffic Light 1 Amber
//   | | | | | | | | | | Traffic Light 2 Amber
//   | | | | | | | | | | Traffic Light 3 Amber
//   | | | | | | | | | | Traffic Light 1 Green
//   | | | | | | | | | | Traffic Light 2 Green
//   | | | | | | | | | | Traffic Light 3 Green
//   | | | | | | | | | | Pedestrian Red Light
//   | | | | | | | | | | Pedestrian Green Light
//   | | | | | | | | | | Phase able to be interrupted for pedestrian crossing request
```

Figure 9 - Traffic Light Sequence Code Representation

Main

the main function is where the code that is to be run repeatedly is placed, it handles all the logic of switching between phases, handling pedestrian requests and the countdown timer.

Shown in Figure 10 is the logic for handling the counter, the value to be displayed is stored in “timerCount” and is reduced by 1 every loop until zero, where its value is reset back to 40.

```
if(TimerCount > 0){
  TimerCount--;
}else{
  TimerCount = 40;
}
```

Figure 10 - Main Code Section 1

The next section of code shown in Figure 11 is responsible for controlling changing phases. The first if statement checks if the elapsed time of the current phase has reached the set duration in traffic light pattern and therefore the current phase needs to be changed.

The following if statement then checks if a pedestrian request has been made and that the current phase can be changed to a pedestrian crossing, this ensures that the light is able to run the amber phase allowing traffic time to stop before allowing pedestrians to cross.

If a pedestrian request is to be changed to the code clears the request flag and sets the current phase to a pedestrian crossing. If it is not to be a pedestrian crossing the code increments the current phase.

```
if(CurrentPhaseDuration >= TrafficLightPattern[CurrentTrafficPhase][1]){
    if(PedestrianRequest & (TrafficLightPattern[CurrentTrafficPhase][0] >> 11)){
        TimerCount = 10;
        PedestrianRequest = FALSE;
        CurrentTrafficPhase = 9;
        CurrentPhaseDuration = 0;
    }else{
        CurrentTrafficPhase ++;
        CurrentPhaseDuration = 0;
    }
}
```

Figure 11 - Main Code Section 2

Convenience features

Several features were added to the program to allow for future modifications and expandability, the first Pin definitions shown in Figure 12. If an alteration was to be made to the hardware altering where devices were connected to there is only one place where this would need to be altered in the program.

```
#define TL1_Red PIN_B0
#define TL2_Red PIN_B1
#define TL3_Red PIN_B2

#define TL1_Amber PIN_B3
#define TL2_Amber PIN_B4
#define TL3_Amber PIN_B5

#define TL1_Green PIN_B6
#define TL2_Green PIN_B7
#define TL3_Green PIN_C0

#define PED_Green PIN_C1
#define PED_Red PIN_C2

#define PedestrianRequestPB PIN_A0
#define PedestrianRequestIndicator PIN_A1

#define Seg_Dig1 PIN_C6
#define Seg_Dig2 PIN_C7
```

Figure 12 - Code Pin Definitions

In addition to this shown in Figure 9 - Traffic Light Sequence Code Representation The sequence can be fully adjusted including timings and patterns in 1 location, this would be beneficial for commissioning where adjustments may be required.

System Validation & Results

With the both the circuit and software complete validation of the controller could now begin, this involved running the simulation and comparing the output of the circuit to the expected results based of the specification.

The output of the controller in both simulations and breadboard (Figure 13) prototype matched the expected results exactly, correctly following the pre-defined sequence. This ensured that safety was maintained as no two approaches where green at any one time.

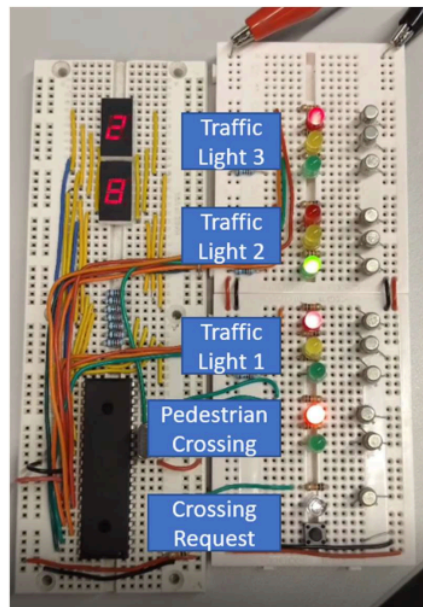


Figure 13 - Testing setup used for breadboard

The response of the pedestrian crossing button was also as expected, illuminating the request LED when the button was pressed, and the system then correctly waited until an appropriate phase of the sequence to change to the pedestrian crossing phase, ensuring time for oncoming traffic to stop before allowing pedestrians to cross.

In addition, the behaviour of the countdown timer was correct, counting down from 40 to zero and correctly resetting. The counter remained in sync with the traffic light sequence even through multiple full sequences and uses of the pedestrian crossing request button where it also correctly skipped to 10 seconds as expected for the pedestrian crossing phase shown in Figure 14.

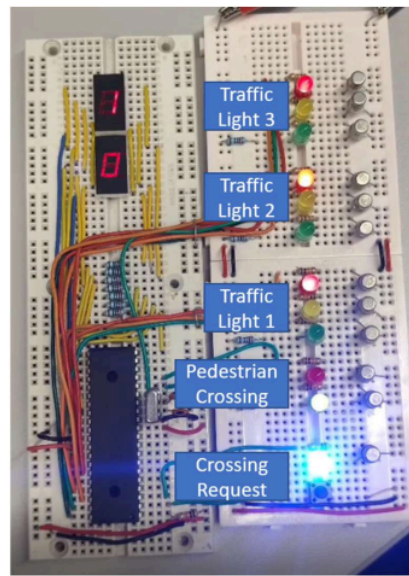


Figure 14 - Breadboard test correctly skipping to crossing phase

Conclusion & Recommended Future Work

The development of the traffic light control system documented in this report demonstrated the feasibility of using the PIC16F877A microcontroller in a T-junction controller. Using simulation in proteus and physical testing with a breadboard prototype, the system consistently performed to the predefined operational requirements. The sequencing operated reliably, ensuring safe transitions and always preventing conflicts in green phases. The pedestrian request feature behaved as intended, transitioning to the pedestrian phase only when the junction was in a safe state. In addition, the multiplexed seven-segment countdown timer proved stable and synchronized with the traffic light sequence, maintaining correct timing even with interruption from the pedestrian phase.

Overall, the project demonstrated effective integration of both hardware design and software development, making use of a modular code design ensuring future expandability and ease of maintenance. The PIC16F877A offered sufficient I/O capability and timing accuracy for use within this system. Successful validation in both simulated and hardware tests indicates the validity of this design for a small-scale traffic control application.

Despite several successes of this controller, enhancements could be made to improve future iterations of the controller. A key area of improvement would be the use of sensor-based actuation, making use of infrared or inductive sensors would allow for dynamic response to real-time traffic conditions. This would improve efficiency by reducing unnecessary waiting times during low-traffic periods. The system could also be extended

supporting adaptive timing algorithms, enabling more intelligent decisions based on real world data.

The hardware could also be improved, by making use of a more modern 32-bit microcontroller several additional features could be implemented, such as data logging, communication interfaces such as RS485 or ethernet, wireless connectivity, or integration with a centralized traffic management system. A important feature to add to ensure safety would be failsafe design, this would ensure the system defaults to a safe state in case of hardware or software failure. Finally, implementing a robust PCB design would allow for correct electrical protection and output devices allowing for deployment in real world environments and increase long-term longevity.

References

- [B. G. 1. Y. P. 1. J. N. 3. Z. W. 1. Y. C. 1. G. X. 1. K. W. 1. ao Li 1, "Analysis and
1 Comparative Study of Signalized and Unsignalized," *energies*, pp. 21-22, 2023.
]
- [S. H. H. 1. Justin P. Schorr *, "Safety propensity index for signalized and unsignalized
2 intersections," *Elsevier*, p. 104, 2013.
]
- ["The Function and Importance of a Traffic Light Controller," CEF, [Online]. Available:
3 [https://www.cef.co.uk/library/traffic-light-](https://www.cef.co.uk/library/traffic-light-controller?msockid=0572c9892bfc68010d1ddf2d2a046985)
] controller?msockid=0572c9892bfc68010d1ddf2d2a046985. [Accessed 28 11 2025].
- [Y. S. a. H. S. Malik Tubaishat, "Adaptive Traffic Light Control with Wireless Sensor
4 Networks," in *IEEE*, Las Vegas, NV, USA, 2007.
]
- [J. L. Y. Z. Y. Chen, "A Self-Adaptive Traffic Light Control System Based on Speed of
5 Vehicles," in *IEEE*, Vienna, Austria, 2016.
]
- [X. Y. ., H. L. ., a. Y. L. Yizhe Wang, "A Review of the Self-Adaptive Traffic Signal
6 Control System Based on Future Traffic Environment," *Journal of Advanced*
] *Transportation*, 2018.
- [J. Kennedy and B. Sexton, "Literature review of road safety at traffic signals and
7 signalised crossings," *TRL*, 2009.
]
- [Department of Transport, "THE DESIGN OF PEDESTRIAN CROSSINGS," Department
8 of Transport, 1995.
]
- [SIEMENS, "Siemens ST950 Datasheet," 2015. [Online]. Available:
9 [https://assets.new.siemens.com/siemens/assets/api/uuid:af072c82-5edd-4a6e-](https://assets.new.siemens.com/siemens/assets/api/uuid:af072c82-5edd-4a6e-b446-c5d5ffddf954/st950.pdf)
] b446-c5d5ffddf954/st950.pdf. [Accessed 1 12 2025].
- [Microchip, "PIC16F87XA Datasheet," 29 11 2012. [Online]. Available:
1 [https://ww1.microchip.com/downloads/aemDocuments/documents/MCU08/Produ](https://ww1.microchip.com/downloads/aemDocuments/documents/MCU08/ProductDocuments/DataSheets/39582C.pdf)
] ctDocuments/DataSheets/39582C.pdf. [Accessed 30 11 2025].

0

]

[M. Palmer, "A Comparison of 8-Bit Microcontrollers," 01 03 2002. [Online]. Available:

1 <http://www.t-es-t.hu/download/microchip/an520d.pdf>. [Accessed 30 11 2025].

1

]

[STMicroelectronics, "Datasheet - STM32F103x8 STM32F103xB," 28 7 2025. [Online].

1 Available:

2 [https://www.google.com/url?sa=t&source=web&rct=j&opi=89978449&url=https://w](https://www.google.com/url?sa=t&source=web&rct=j&opi=89978449&url=https://www.st.com/resource/en/datasheet/stm32f103c8.pdf&ved=2ahUKEwjn7-68l5qRAxWfXUEAHWCNL-EQFnoECAwQAQ&usg=AOvVaw0rd6l_7fuhTLdZOoycvGV5)

3 [ww.st.com/resource/en/datasheet/stm32f103c8.pdf&ved=2ahUKEwjn7-](https://www.st.com/resource/en/datasheet/stm32f103c8.pdf&ved=2ahUKEwjn7-68l5qRAxWfXUEAHWCNL-EQFnoECAwQAQ&usg=AOvVaw0rd6l_7fuhTLdZOoycvGV5)

68l5qRAxWfXUEAHWCNL-

EQFnoECAwQAQ&usg=AOvVaw0rd6l_7fuhTLdZOoycvGV5. [Accessed 30 11 2025].

[I. Corporation, "Intel 803xAH/805xAH Datasheet," 01 02 2008. [Online]. Available:

1 [https://www.google.com/url?sa=t&source=web&rct=j&opi=89978449&url=https://w](https://www.google.com/url?sa=t&source=web&rct=j&opi=89978449&url=https://www.keil.com/dd/docs/datashts/intel/80xxah_ds.pdf&ved=2ahUKEwiH5PHAmJqRAxXwQ0EAHSMPMEwQFnoECAwQAQ&usg=AOvVaw3tkH3_98GXiAq08VQOIAGo)

3 [ww.keil.com/dd/docs/datashts/intel/80xxah_ds.pdf&ved=2ahUKEwiH5PHAmJqRAx](https://www.keil.com/dd/docs/datashts/intel/80xxah_ds.pdf&ved=2ahUKEwiH5PHAmJqRAxXwQ0EAHSMPMEwQFnoECAwQAQ&usg=AOvVaw3tkH3_98GXiAq08VQOIAGo)

] [XwQ0EAHSMPMEwQFnoECAwQAQ&usg=AOvVaw3tkH3_98GXiAq08VQOIAGo](https://www.keil.com/dd/docs/datashts/intel/80xxah_ds.pdf&ved=2ahUKEwiH5PHAmJqRAxXwQ0EAHSMPMEwQFnoECAwQAQ&usg=AOvVaw3tkH3_98GXiAq08VQOIAGo).

[Accessed 30 11 2025].

[H. Wan-Fu, "The design of a six-digit digital clock with a four-digit seven-segment

1 display module," in *IEEE*, Yichang, China , 2011.

4

]

[G. Leyva-Hernandez, "Application Note LED-based Multiplexed Display

1 Implementation Using TI Programmable Logic Devices," Texas Instruments, 2025.

5

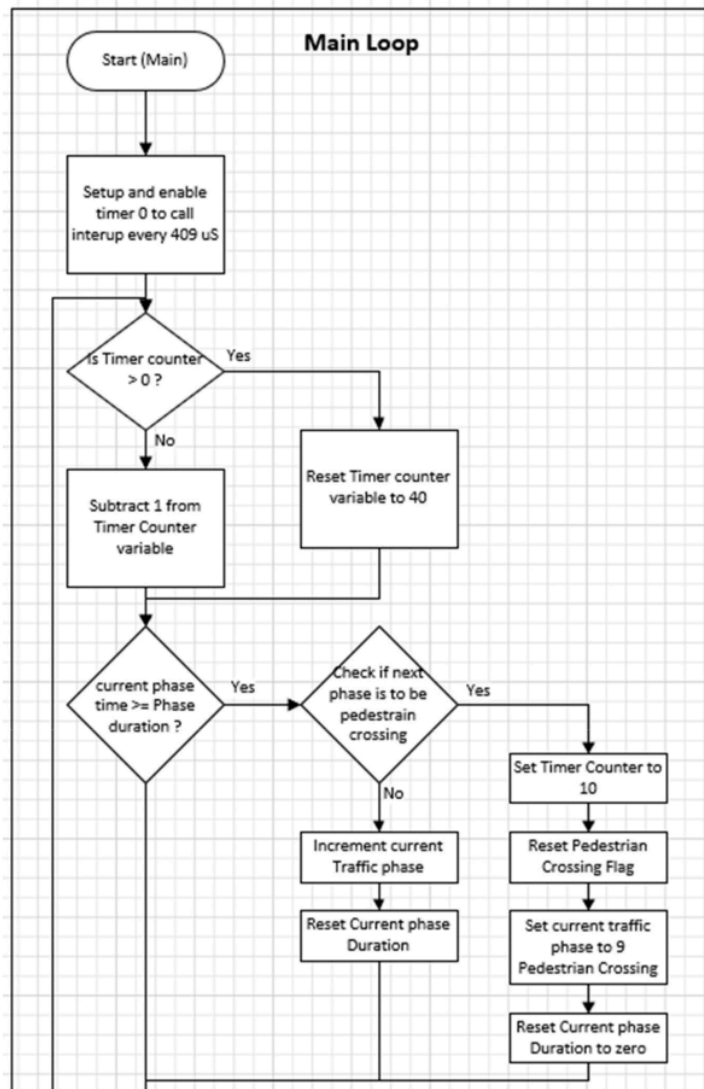
]

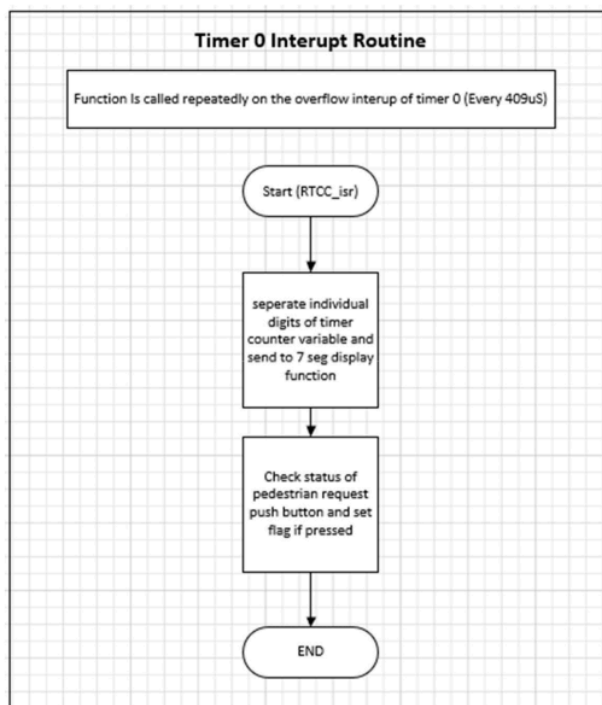
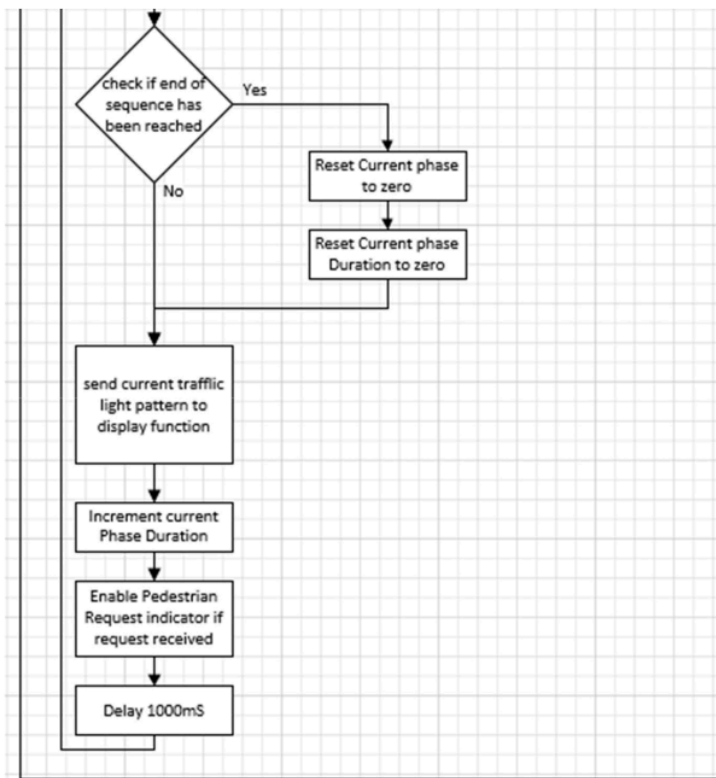
Table Of Figures

Figure 1 - Common Anode 7 Segment display pair Wiring.....	6
Figure 2 - System Block Diagram	7
Figure 3 - Proteus PIC16F877A Connections	7
Figure 4 - Proteus Traffic Lights, Timers and Pedestrian crossings	8
Figure 5 - Controller Breadboard Prototype	8
Figure 6 - RTCC Interrupt Function Code	11
Figure 7 - DispNumber Function Code	12
Figure 8 - DispTrafficLights Function.....	12
Figure 9 - Traffic Light Sequence Code Representation	13
Figure 10 - Main Code Section 1	13
Figure 11 - Main Code Section 2	14
Figure 12 - Code Pin Definitions	14
Figure 13 - Testing setup used for breadboard	15
Figure 14 - Breadboard test correctly skipping to crossing phase.....	16

Appendix

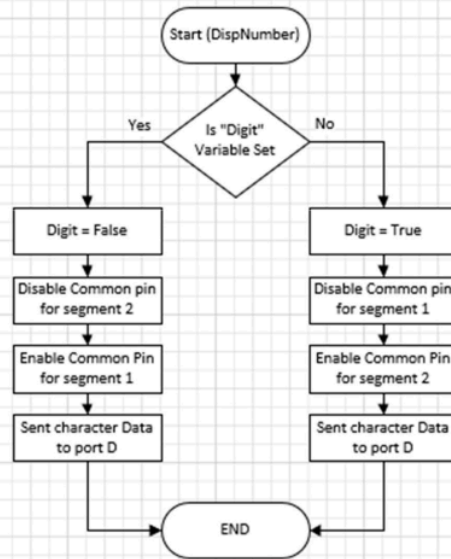
Appendix A Flow Chart





7 segment Display Function

Function is called repeatedly From Timer 0 Interup, function flips between segment 1 and 2, sending character data to each depending what has been passed to it, Multiplexing the displays



Traffic Light Display Function

Takes a variable "Lights" each bit of this variable indicates the status of the lights, function updates status of pins based on this variable

Start (DispTrafficLights)

Update Pin "TL1_Red" based on bit 1 of Lights Variable

Update Pin "TL2_Red" based on bit 2 of Lights Variable

Update Pin "TL3_Red" based on bit 3 of Lights Variable

Update Pin "TL1_Amber" based on bit 4 of Lights Variable

Update Pin "TL2_Amber" based on bit 5 of Lights Variable

Update Pin "TL3_Amber" based on bit 6 of Lights Variable

Update Pin "TL1_Green" based on bit 7 of Lights Variable

Update Pin "TL2_Green" based on bit 8 of Lights Variable

Update Pin "TL3_Green" based on bit 9 of Lights Variable

Update Pin "PED_Red" based on bit 10 of Lights Variable

Update Pin "PED_Green" based on bit 11 of Lights Variable

END