

Electronics 1 Assignment - Ruben Bhatia

by Ruben Bhatia

Submission date: 23-Apr-2024 10:39AM (UTC+0100)

Submission ID: 230446042

File name: 33053_Ruben_Bhatia_Electronics_1_Assignment_-_Ruben_Bhatia_568563_1838969874.docx
(1.26M)

Word count: 2450

Character count: 11911

Electronics I Assignment



Ruben Bhatia

Electronic & Electrical Engineering / Electronics I

Submitted to:

UWTSD – Waterfront Campus

Introduction	2
Background Research	2
Circuit Design.....	4
Flowchart & Pseudocode	5
Software Listings.....	6
Simulation Screenshots	11
Test Outputs & Results	14
Conclusion & Recommendations On Future Work.....	15
References	15

Introduction

The aim of this exercise is to design a four traffic signal light system to allow for comprehensive traffic management. This design requires a set of 7-segment displays to show a timer counting to fifty seconds, where every ten seconds the traffic signal changes allowing for smooth traffic flow. The final ten seconds are to be used as a pedestrian crossing phase. The traffic signals must also turn off during this phase.

The simulation part of this experiment will be done entirely using the Proteus 8 software, as it allows for complete circuit design and simulation. The code will be written on MikroC Pro software as it allows for code editing and changes and allows for updates to the code used within the Proteus 8 simulation whilst it is active. This is an extremely useful function for testing.

The PIC used within the Proteus 8 simulation will be a PIC16F877A. The 7-segment displays will be a common anode display.

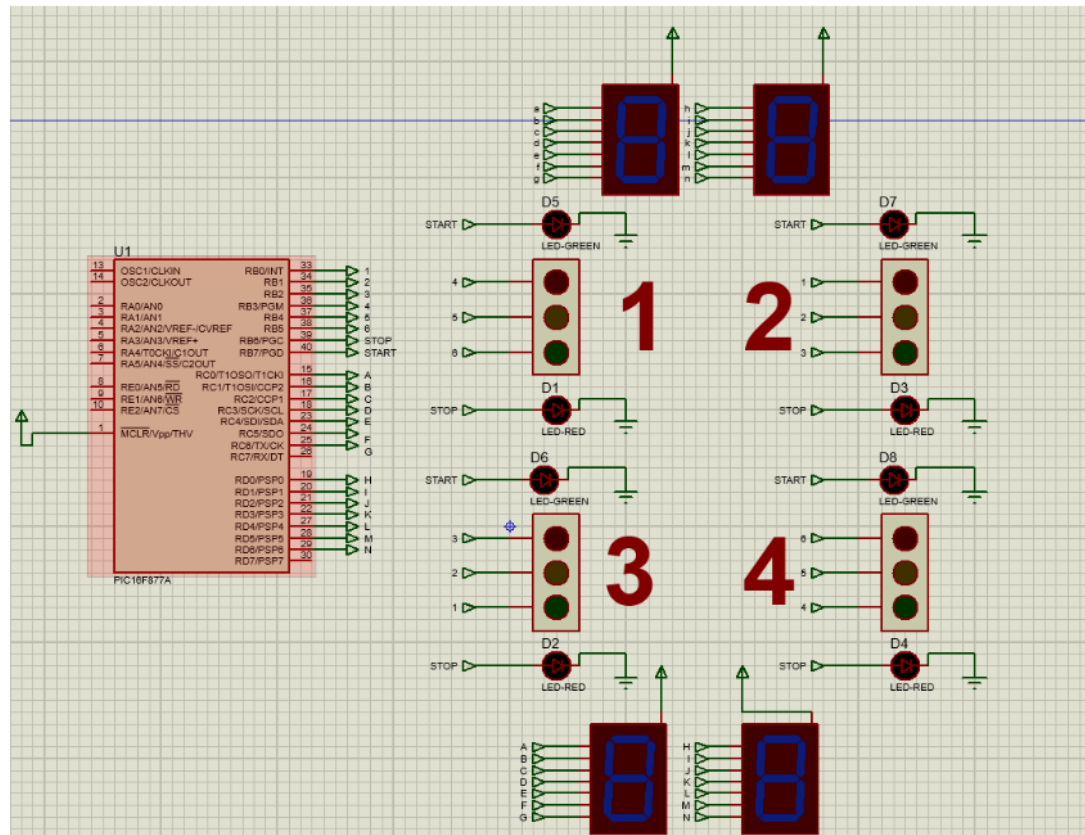
Background Research

The first part of background research to be conducted is to investigate how traffic lights function. This will be done as it may provide further insight into how to design the circuit and program the traffic signals themselves. Traffic lights generally either function “using a fixed period or vehicle triggered operation.” Fixed period operation is where the traffic signals change after a set period. This period remains constant unless a pedestrian crossing button is pressed but is otherwise uninterrupted when pedestrian utilities are not set in place. A vehicle triggered operation detects whether vehicles and/or pedestrians are nearby and alters these signals accordingly [1].” From this information it can be deduced that a fixed period simulation will be designed as the Proteus 8 software does not allow for the simulation of vehicles and pedestrians. In addition to this, a fixed period operation does not have as much randomness and volatility as a vehicle triggered operation would have and thus abides by the parameters set

by the exercise. Traffic lights also generally display 3 colors. "When red is displayed, it means stop. When red and amber are displayed it means stop and do not start until green is displayed. When green is displayed, the vehicle may continue moving forward. When yellow is displayed it means stop at the stop line. However, the vehicle may continue if it has already passed the stop line by the time the traffic light turns yellow [4]."

The next section of background research would be which PIC microcontroller to use. The microcontroller selected must require enough pins for the simulation to function fully and be relatively simple in its design. It would require a minimum of 2 pins for the 7-segment display timers, as 7 of these would be used for the 'tens of seconds' display and the other 7 would be used for the 'seconds' display. Another 6 of these pins would be used for the traffic signals themselves and 2 more pins would be used for the 'start' and 'stop' pedestrian crossing signals. This would mean that a bare minimum of 22 pins would be required for a PIC microcontroller to be used within this simulation. This number of pins excludes pins designated for power to the circuit as well as any pins required for any additional features on the circuit which may be implemented. Therefore, the PIC16F877A has been chosen to be used within this simulation as it adheres to these parameters and has 40 pins to use which is 4 times more than enough pins for the simulation.

Circuit Design



[Figure 1]

Figure 1 shows the circuit design within Proteus 8. As stated previously, it uses a PIC16F877A microcontroller to control the circuit. In addition to this there are four 7-segment displays used as the timers to display each phase of the traffic signals and show pedestrians how long the signals will take to move to their next phase. There are numerous traffic signals to show traffic if it is okay to move or not. There are also numerous green and red LEDs to show pedestrians whether they can 'cross the road' or not.

As seen in Figure 1, multiple 7-segment displays are used. These displays are common anode 7-segment displays. This is used instead of a common cathode 7-segment display because the PIC used has pins which source current whereas a common cathode 7-segment display requires pins which sink current [2]. Two 7-segment displays are used in conjunction with one another as to display different units of time. The leftmost displays show time in the 'tens of seconds' whereas the rightmost displays show time in just 'seconds'. This is done to allow for the timers to count to 50 seconds as specified earlier in the report. Each value from 0-9 has a "designated truth table for which segments should be on and off" [3]. These truth tables also

show us what hexadecimal value can be used to display that specific number on the display. These will be used in an array when implementing this into the simulation's code.

Traffic light signals have been used instead of LEDs as they more accurately depict the function of real traffic lights within the simulation.

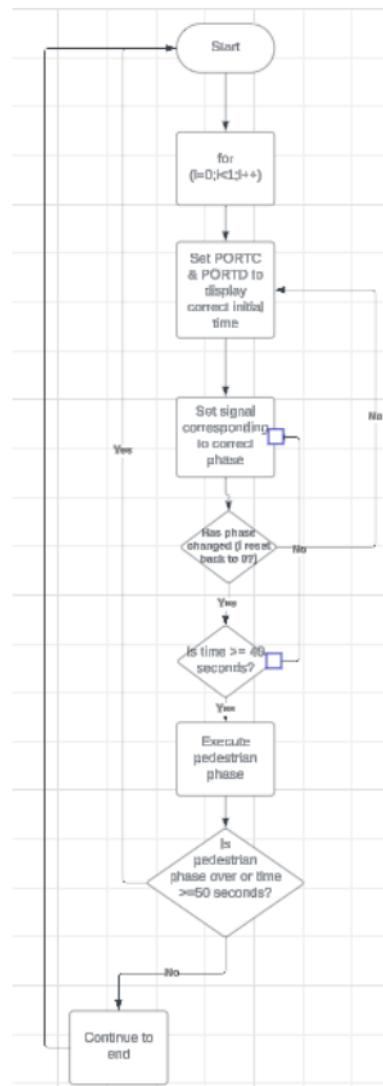
Both red and green LEDs have been used to show whether pedestrians can 'cross' or not. In the case of this simulation red means stop and green means go. Ideally, the red LEDs will remain on whilst the traffic signals are active and will turn off during the 10 second pedestrian phase. The green lights will remain off for all non-pedestrian phases and turn on during the pedestrian phase.

Most connections within the circuit are made using output and input terminals instead of wires. This has been done to clearly show where connections are made throughout the circuit. Additionally, the lack of wires within the circuit allow for a clearer display of all signals and timers, making it easier to gather information when making and testing the circuit.

Flowchart & Pseudocode

The pseudocode for the simulation is as follows:

1. For (i=0;i<10;i++) // Repeat process until i=10
2. PORTC = XXXX // Display X on leftmost display (tens of seconds)
3. PORTD = [i] // Display i (seconds) on rightmost display
4. Signal 1 = Red, Signal 2 = Yellow, Signal 3 = Yellow, Signal 4 = Green, Pedestrian Stop = On
5. Delay_ms(1000) // Wait 1 second before repeating i increment loop
6. Repeat, changing PORTC value as 'tens of seconds increments' and signal inputs alternating between red, yellow and green every 10 seconds
7. When leftmost display = 4, set all signals to off and set Pedestrian Start to 'On' for 10 seconds
8. Repeat entirely



[Figure 2]

Figure 2 (see above) displays a flow chart for how the simulation should function.

Software Listings

```

StartPage Assignment.c
* sbit LED_Red at RB0_bit;
* sbit LED_Yellow at RB1_bit;
* sbit LED_Green at RB2_bit;
* sbit LED_Red1 at RB3_bit;
* sbit LED_Yellow1 at RB4_bit;
* sbit LED_Green1 at RB5_bit;
* sbit LED_Stop at RB6_bit;
* sbit LED_Start at RB7_bit;
*
10 int i; //count
* unsigned int display [10] = {0XC0, 0XF9, 0XA4, 0XB0, 0X99, 0X92, 0X82, 0XF8, 0X80, 0X90}; //values for 7-seg display (0-9)
*
* void main() //Main program
* {
*   TRISB = 0X00; //Set tristate B to 0
*   PORTB = 0X00; //Set PORTB to 0
*   TRISC = 0X00; //Set tristate C to 0
*   PORTC = 0X00; //Set PORTC to 0
*   TRISD = 0X00; //Set tristate D to 0
*   PORTD = 0X00; //Set PORTD to 0
20   do {
*     //do-while loop
*     //for 00-09 display
*     for (i=0;i<10;i++) //i count goes up by 1/second
*     {
*       PORTC = 0XC0; //sets tens of seconds timer to 0
*       PORTD = display [i]; //sets seconds timer to i
*
*       //Case 1
*       //Traffic Light 1
30       LED_Red = 0; //set 0
*       LED_Yellow = 1; //set signals 2 & 3 to yellow
*       LED_Green = 0; //set 0
*
*       //Traffic Light 2
*       LED_Red1 = 1; //set signal 1 to red and 4 to green
*       LED_Yellow1 = 0; //set 0

```

[Figure 3]

```

StartPage Assignment.c
* //Traffic Light 2
* LED_Red1 = 1; //set signal 1 to red and 4 to green
* LED_Yellow1 = 0; //set 0
* LED_Green1 = 0; //set 0
* LED_Stop = 1; //set pedestrian stop to on
* Delay_ms(1000); //delays 1 second
40 }
*
* //for 10-19 display
* for (i=0;i<10;i++) //i count goes up by 1/second
* {
*   PORTC = 0XF9; //sets tens of seconds timer to 1
*   PORTD = display [i]; //sets seconds timer to i
*   //Case 2
*   //Traffic Light 1
*   LED_Red = 0; //set 0
60   LED_Yellow = 0; //set 0
*   LED_Green = 1; //set signal 2 to green and 3 to red
*
*   //Traffic Light 2
*   LED_Red1 = 0; //set 0
*   LED_Yellow1 = 1; //set signals 1 & 4 to yellow
*   LED_Green1 = 0; //set 0
*   LED_Stop=1; //set pedestrian stop to on
*   Delay_ms(1000); //delays 1 second
* }
60
* //for 20-29 display
* for (i=0;i<10;i++) //i count goes up by 1/second
* {
*   PORTC = 0XA4; //sets tens of seconds timer to 2
*   PORTD = display [i]; //sets seconds timer to i
*   //Case 3
*   //Traffic Light 1
*   LED_Red = 0; //set 0
*   LED_Yellow = 1; //set signals 2 & 3 to yellow

```

[Figure 4]

```

Start Page Assignment.c
* PORTC = 0xA4; //sets tens of seconds timer to 2
* PORTD = display [1]; //sets seconds timer to i
* //Case 3
* //Traffic Light 1
* LED_Red = 0; //set 0
* LED_Yellow = 1; //set signals 2 & 3 to yellow
70 LED_Green = 0; //set 0
* //Traffic Light 2
* LED_Red1 = 0; //set 0
* LED_Yellow1 = 0; //set 0
* LED_Green1 = 1; //set signal 1 to green and 4 to red
* LED_Stop = 1; //set pedestrian stop to on
* Delay_ms(1000); //delays 1 second
* }
*
* //for 30-39 display
80 for (i=0;i<10;i++) //i count goes up by 1/second
{
* PORTC = 0xB0; //sets tens of seconds timer to 3
* PORTD = display [1]; //sets seconds timer to i
* //Case 4
* //Traffic Light 1
* LED_Red = 1; //set signal 2 to red and 3 to green
* LED_Yellow = 0; //set 0
* LED_Green = 0; //set 0
*
90 //Traffic Light 2
* LED_Red1 = 0; //set 0
* LED_Yellow1 = 1; //set signals 1 & 4 to yellow
* LED_Green1 = 0; //set 0
* LED_Stop = 1; //set pedestrian stop to on
* delay_ms(1000); //delays 1 second
* }
* //All Traffic Lights Off
* for (i=0;i<10;i++) //i count goes up by 1/second

```

[Figure 5]

```

Start Page Assignment.c
//All Traffic Lights Off
for (i=0;i<10;i++) //i count goes up by 1/second
{
100 PORTC = 0X99; //sets tens of seconds timer to 4
    PORTD = display[i]; //sets seconds timer to i
    //Case 5
    //Traffic Light 1
    LED_Red = 0; //set 0
    LED_Yellow = 0; //set 0
    LED_Green = 0; //set 0
    //Traffic Light 2
    LED_Red1 = 0; //set 0
110 LED_Yellow1 = 0; //set 0
    LED_Green1 = 0; //set 0
    LED_Start = 1; //set pedestrian start to on
    Delay_ms(1000); //delays 1 second
}

for (i=0;i<1;i++) //i count goes up by 1/second
{
    PORTC = 0X92; //sets tens of seconds timer to 2
    PORTD = display[i]; //sets seconds timer to i
120 //Case 6
    //Traffic Light 1
    LED_Red = 0; //set 0
    LED_Yellow = 0; //set 0
    LED_Green = 0; //set 0
    //Traffic Light 2
    LED_Red1 = 0; //set 0
    LED_Yellow1 = 0; //set 0
    LED_Green1 = 0; //set 0
130 LED_Stop = 1; //set pedestrian stop to on
    Delay_ms(1000); //delays 1 second
}

```

[Figure 6]

```

//Traffic Light 2
LED_Red1 = 0; //set 0
LED_Yellow1 = 0; //set 0
LED_Green1 = 0; //set 0
130 LED_Stop = 1; //set pedestrian stop to on
    Delay_ms(1000); //delays 1 second
}
}
while(1); //do-while loop
}

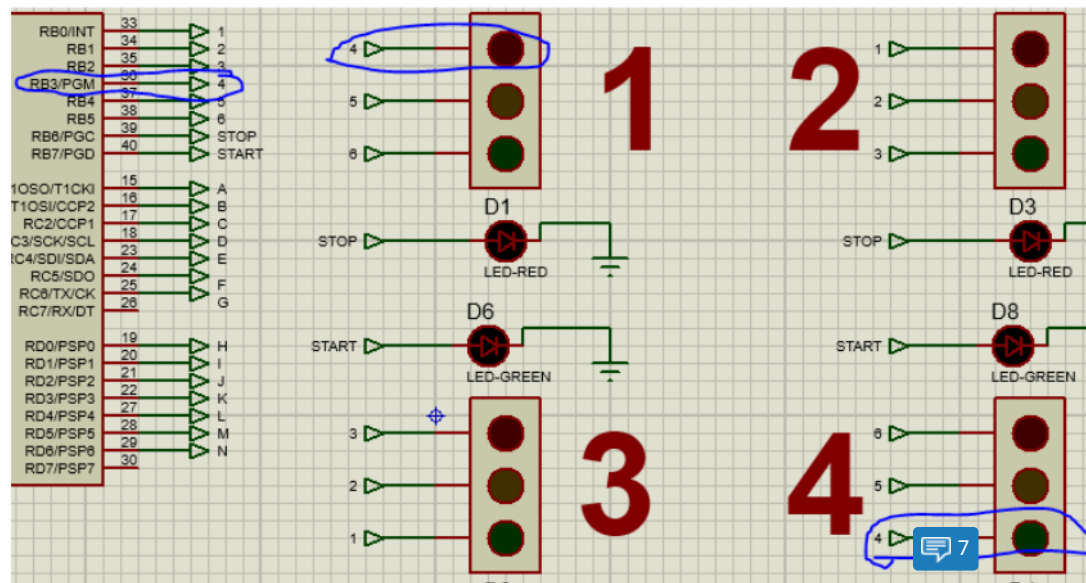
```

[Figure 7]

Figures 3-7 show the code written to execute the simulation. It was written within the MikroC program.

As seen in Figures 1 & 3, bits are set according to where the signal connections occur within Proteus 8. The count is also set as 'i'. The unsigned int display '[10]' is the array of hexadecimal values for a common anode 7-segment display to show numbers 0 to 9. The order of which these hexadecimal values are placed within the array ascends from displaying 0 (0XC0) to 9 (0X90) which means that when the code is executed the 7-segment displays will count upwards from 0 to 9. The void main() is the main void and is where the commands for the entire simulation will be. Tristates and ports B,C and D are all set to 0 so all pins are off initially. This will prevent errors from occurring within the program allow for all these pins to be recognized as off before they are turned on. A do-while loop is then set so the program may run indefinitely. The 'i' count then begins incrementing at the beginning of the first phase (00-09 seconds). It states that i is set to 0 and increments by one until it is no longer less than 10 in which case it is then set back to 0. PORTC is then set to 0XC0 to display a '0' on the leftmost displays. PORTD is set to display [i] to show not only the count but also to set the rightmost displays to show the time in seconds.

Figures 1, 3 & 4 show the next step of the first phase of the simulation. For the first set of traffic lights, signals 2 & 3 are set to yellow. This is done by setting LED_Yellow to 1. By setting LED_Red and LED_Green to 0 only yellow will be displayed on traffic lights 2 & 3. For the second set of traffic lights, signal 1 is set to red and signal 4 is set to green. This is done by setting LED_Red1 to 1 as well as LED_Yellow1 and LED_Green1 to 0 to display only red and green for signals 1 and 4 respectively (see Figure 1). Alternating which side of the traffic lights LED_Red1 is connected to, a red light and green light can appear simultaneously on two separate signals, even though the source for both signals originate at the same pin on the PIC. Figure 8 shows a closer look at this.



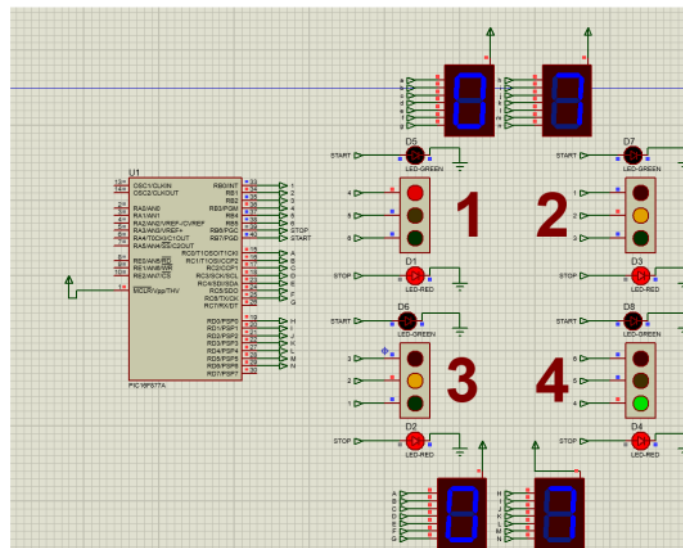
[Figure 8]

Then, the red pedestrian LED (LED_Stop) is set to 1, turning it on. After all signals have been set, a 1 second delay occurs and then *i* increments by 1. This process repeats itself until *i* is no longer less than 10 and is set back to 0. Once this occurs, the second phase begins. This process repeats itself five times in total and can be seen in Figures 3 to 7. No signal displays red or green more than once (other than the pedestrian signals) thanks to the alternating inputs as mentioned previously. Yellow is displayed only when the traffic lights are transitioning between red and green. This means that all traffic lights have a set phase as to when they are set red and green and no two lights are set to the same color other than yellow simultaneously. This pattern of alternating lights continues up until the 7-segment displays show 39 seconds, as when they show 40 seconds (see Figure 6) all traffic lights are set to 0 and turned off, as well as the LED_Stop light. The LED_Start light is set to 1, turning it on, allowing for a 10 second period where pedestrians can 'cross safely'.

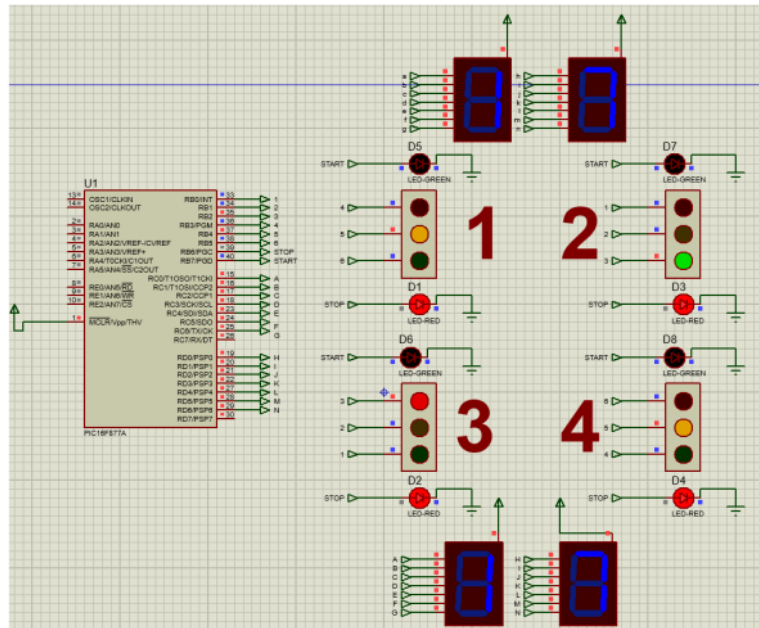
Finally, a buffer period of one second is set (see Figures 6 & 7) to allow for the cycle of phases to reset without error.

Simulation Screenshots

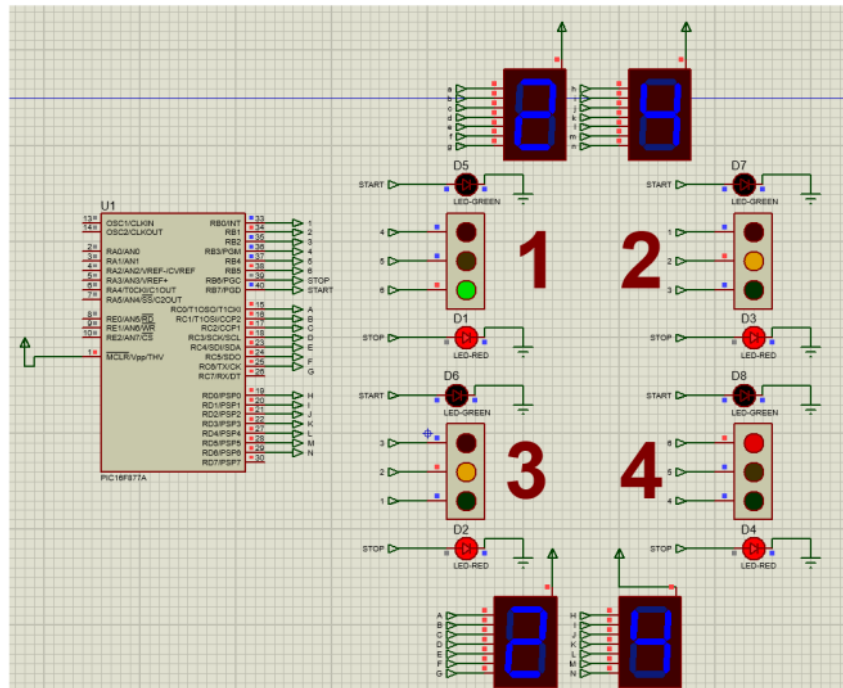
[Figure 9]



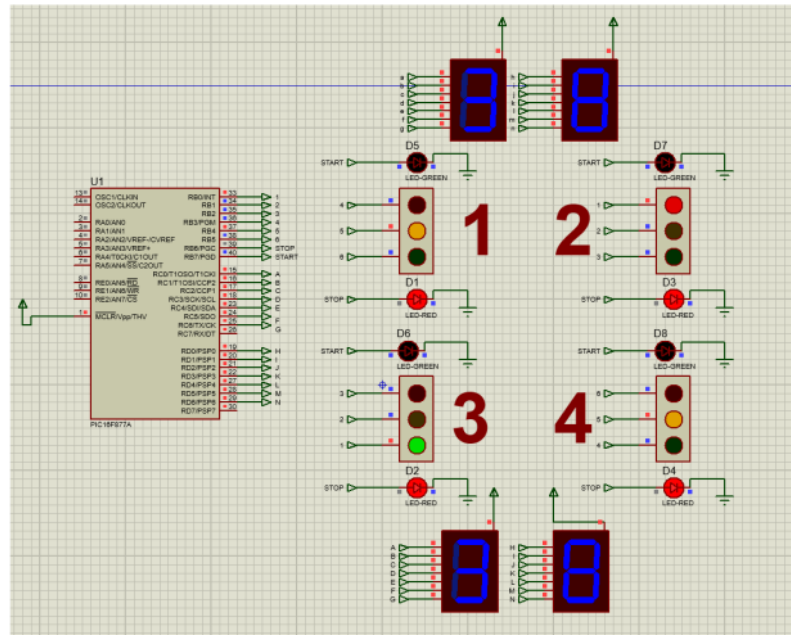
[Figure 10]



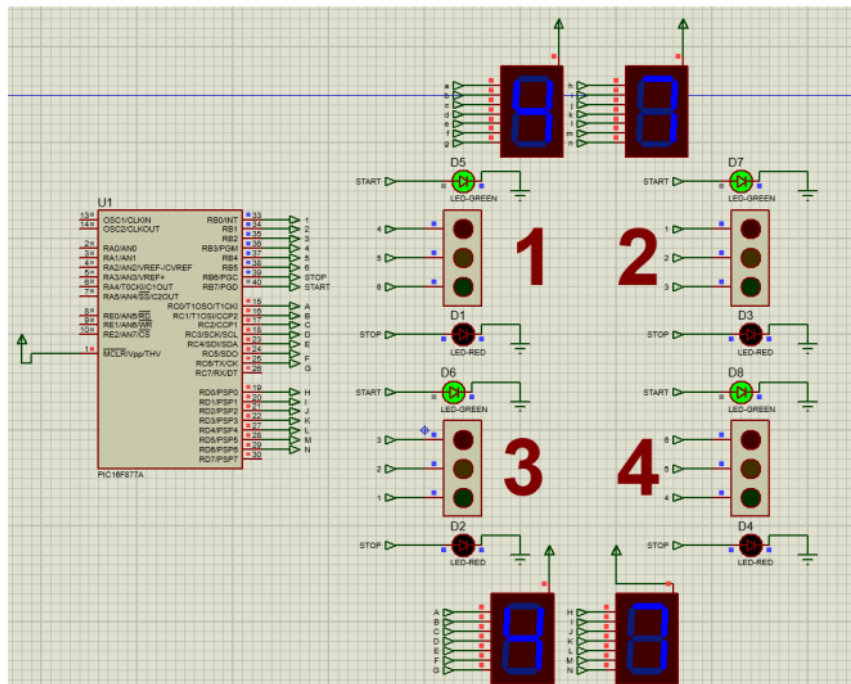
[Figure 11]



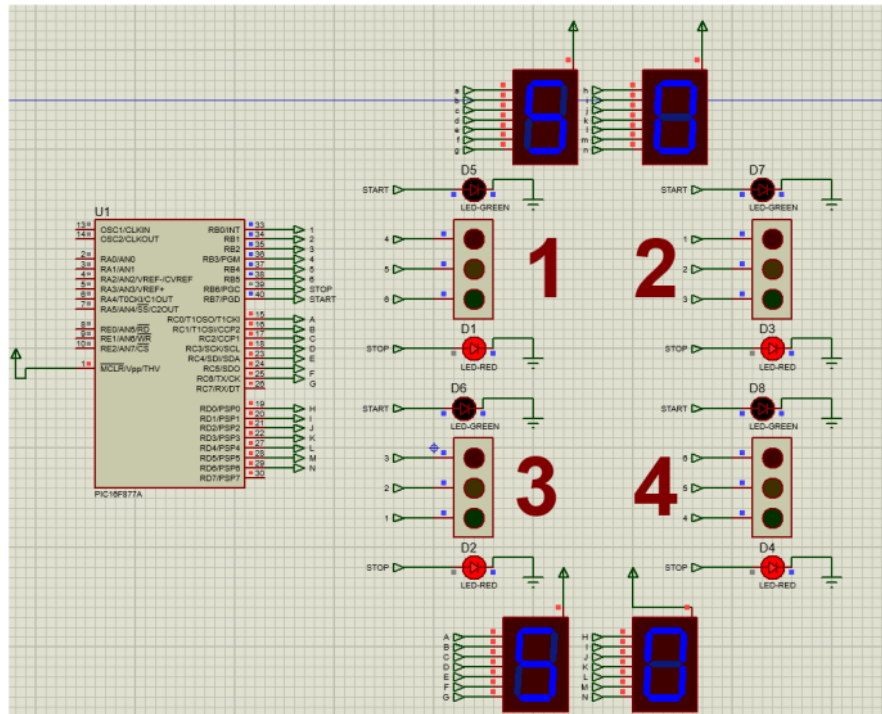
[Figure 12]



[Figure 13]



[Figure 14]



Test Outputs & Results

Figure 9 shows the first phase of the simulation. As described earlier, it shows the leftmost displays (the 'tens of seconds' timer) as 0 and the rightmost displays as 'i'. In addition to this, it shows two yellow lights, a red and a green light for traffic lights 1 to 4. Finally it shows the LED_Stop lights as on and the LED_Start lights as off.

Figures 10 – 12 show variations of the phase listed above, these being phases 2 to 4 respectively. They all show the traffic lights displaying different color lights and all have the LED_Stop lights as on. The 7-segment displays throughout these phases also show a variety of times all respective to their phases as coded (see Figures 3-5). They all show that the first part of the simulation works correctly and abides by the set parameters.

Figure 13 shows the fifth and final phase of the simulation. All traffic light signals are off as well as the LED_Stop lights. The LED_Start lights have been turned on during the final 40-49 seconds of the simulation. This also shows that this phase of the simulation is functioning as intended.

Figure 14 shows second 50 of the simulation. As mentioned, this final second acts as a buffer and bridges the end of a full cycle of phases within the simulation to a reset of the simulation, beginning the cycle again. All traffic light signals are set to off, as well as the LED_Start signals. The LED_Stop signal is set to on. Once the counter hits one at this buffer stage, the count

resets back to zero, the 7-segment displays show a time of 00 seconds and the traffic lights display their initial colors thus resetting the cycle. From this point the simulation continues from the beginning and starts counting back up to 50.

Conclusion & Recommendations On Future Work

To conclude, the background research and testing conducting has led to a successful execution of the task. All aspects of the simulation and code function as intended and are simplified in their design. The simulation is clear and each phase within it shows the desired changes.

A recommendation on future work would be that further testing may have been conducted to turn the signals from all off to all showing red lights in phase 5 of the simulation. This is because real traffic lights display a red light when a pedestrian light shows green and thus the simulation should have been made as so.

Another recommendation on future work would be that if more complex code was written and used, yellow lights shown by the traffic signals could transition from yellow to another color. Instead of this being done at the beginning of a new phase of the simulation, it could potentially be done during a phase and fade from yellow to red. Although only one of the four signals can show green at any given time, fading the lights from yellow to red would provide a more realistic look to the traffic lights as this would replicate them in a real world scenario.

References

- [1] Auto Mate Systems Ltd., "How Modern Traffic Lights Signals Work?," Auto Mate Systems, Unknown Unknown Unknown. [Online]. Available: <https://www.automatesystems.co.uk/traffic-lights-signals-work>. [Accessed 19 4 2024].
- [2] EG Projects, "Difference between Common Anode and Cathode seven segment display," Engineers Garage, Unknown Unknown Unknown. [Online]. Available: <https://www.engineersgarage.com/common-anode-and-cathode-7-segment-display/>. [Accessed 20 4 2024].
- [3] CD Team, "7 Segment Display Pinout, Codes, Working, and Interfacing," electronicsforu.com, 8 8 2023. [Online]. Available: <https://www.electronicsforu.com/resources/7-segment-display-pinout-understanding>. [Accessed 20 4 2024].
- [4] GOV.UK, "Light signals controlling traffic," GOV.UK  8 Unknown Unknown Unknown. [Online]. Available: <https://www.gov.uk/guidance/the-highway-code/light-signals-controlling-traffic>. [Accessed 21 4 2024].



Electronics 1 Assignment - Ruben Bhatia

GRADEMARK REPORT

FINAL GRADE

GENERAL COMMENTS

65 / 100

PAGE 1

PAGE 2



Comment 1 |  **Knowledge & Understanding (20)**

Good, few of the types of traffic light systems are identified. Still there are may other traffic light systems available in the market. Writing about those controllers will enhance your report

PAGE 3



Comment 2

very true



Comment 3

More than 4 pins.



Comment 4 |  **Cognitive & Intellectual (20)**

Good, selection on a microcontroller not only depend upon the number pins, but also certain other parameters such as cost, performance, utilization of memory, its reliability, power consumption should also be considered.

PAGE 4



Comment 5 |  **Knowledge & Understanding (20)**

Good, numbering the working / flow sequence of traffic light is appreciable.

PAGE 5

PAGE 6

PAGE 7

PAGE 8

PAGE 9

PAGE 10



Comment 6 |  Practical Application Sk(30)

Good explanations are provided on the working of the simulation.



Comment 7 |  Practical Application Sk(30)

The Design could be more pleasing if some road layouts are designed to mimic the actual working of the traffic light systems.

PAGE 11

PAGE 12

PAGE 13

PAGE 14

PAGE 15



Comment 8 |  Engagement with Literature(10)

More citing's are needed to support the literature review

PAGE 16

GRADING FORM: EC1

RUBEN BHATIA

64.78

ENGAGEMENT WITH LITERATURE(10)

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



Limited identification of relevant concepts Limited explanation of significance, with several key points missing. 2.5

Limited background research with few sources. 1.68

Limited variety of sources. Several references are incorrectly formatted or cited. 1

5.18

KNOWLEDGE & UNDERSTANDING (20)

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



Demonstrates limited knowledge of traffic light controller products and their market requirements. 5.9

Reflects some understanding of the underlying principles in analysing different traffic light controllers. 6

11.9

COGNITIVE & INTELLECTUAL (20)

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

13.2



Provides an few review of the microcontroller selection, with basic analysis of supporting modules and ICs. 4.2

Shows adequate innovative thinking in problem-solving and feature additions. 4.83

Provides adequate pseudocode that represents the system operation, though with some errors. Pseudocode is generally understandable but may have some gaps or unclear sections. 4.14

PRACTICAL APPLICATION SK(30)

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



The circuit design layout is adequate, demonstrating a basic understanding of circuit design principles. 6.5

Provides clear and accurate simulated output results that validate the functionality of the traffic light controller. 14

20.5

TRANSFERABLE SKILLS (20)

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

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

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



The report is well-organized, with a logical flow of ideas that is easy to follow. 5

Most necessary definitions, acronyms, and

14

abbreviations are provided and correctly used. 4
The report is adequately professional and
presented, with noticeable areas for improvement.
There are some spelling or grammatical errors.
The report is mostly complete but may lack some
necessary aspects. The quality of the report is
adequate, with generally clear formatting but
lacking in some areas. 5