SMART TRAFFIC LIGHT

By

MUSLIM BIN MUSTAPA

FINAL PROJECT REPORT

Submitted to the Electrical & Electronics Engineering Programme
in Partial Fulfillment of the Requirements
for the Degree
Bachelor of Engineering (Hons)
(Electrical & Electronics Engineering)

Universiti Teknologi Petronas
Bandar Seri Iskandar
31750 Tronoh
Perak Darul Ridzuan

CERTIFICATION OF APPROVAL

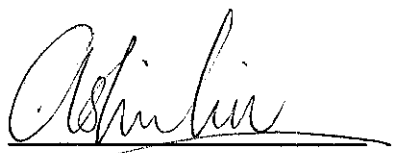
Smart Traffic Light

By

Muslim bin Mustapa

A project dissertation submitted to the
Electrical & Electronics Engineering Programme
Universiti Teknologi PETRONAS
in partial fulfillment of the requirement for the
Bachelor of Engineering (Hons)
(Electrical & Electronics Engineering)

Approved by,



(Mrs Norashikin bt Yahya)

UNIVERSITI TEKNOLOGI PETRONAS
TRONOH, PERAK
June 2007

CERTIFICATION OF ORIGINALITY

This is to certify that I am responsible for the work submitted in this project, that the original work is my own except as specified in the references and acknowledgements, and that the original work contained herein have not been undertaken or done by unspecified sources or persons.

MUSLIM BIN MUSTAPA

ABSTRACT

In this report a fuzzy logic model of traffic light controller at an intersection of two streets has been designed with an aim to improve the performance and offer flexibility of the traffic flow through the intersection. Fuzzy logic model of a traffic light controller have a lot of advantages against the conventional traffic light that is using fixed cycle time. One of the advantages is the ability to adjust the cycle time depending on the number of cars waiting behind the traffic light instead of fixed cycle time that used by the conventional traffic light.

This project is divided into two main parts that are the fuzzy logic toolbox and the traffic light circuit. The fuzzy logic toolbox is used to develop fuzzy logic system to generate the most accurate extension time for the traffic light to response to the number of cars waiting behind the traffic light. The traffic light circuit is used to control the light sequence based on the inputs received from sensors.

In the traffic light circuit, there are eight incremental sensor used to count numbers of car arrived and leaving the intersection. Each junction will have two incremental sensors. The incremental sensor was developed by using PIC16F84. From the incremental sensor it will send the data counted to another circuit to compute numbers car left at the junction by subtracting the data from second incremental sensor with data from the first incremental sensor. Then the data will be sent to another circuit that will compute the average number of cars at both intersections. From this circuit, it will send the data to the traffic light circuit that will decide the time extent based result obtained from MATLAB. This method had increased the traffic light efficiency compared to the conventional traffic light.

ACKNOWLEDGEMENT

First of all, I would like to express my gratitude to Allah s.w.t to make my project run smoothly as planned for the entire final year. I also would like to express my appreciation to the supervisor, Mrs Azrina for her guidance and encouragement to complete the project. Great thanks to other lecturers of Electrical & Electronics Engineering. Big thanks also go to all laboratory technicians, especially to Mrs Siti Hawa and Mrs Siti Fatimah for their great support and helpful assistance.

I would like also like to express a special thank to my family for their priceless supports, encouragements, constant love, valuable advices and their understanding. And also bunch of thanks to all my colleagues for the supports to this project.

Finally, million thanks to all parties who had contributed directly or indirectly to the success of this project.

Thank You

MUSLIM BIN MUSTAPA
Electrical and Electronics engineering
Universiti Teknologi PETRONAS

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF EQUATIONS

LIST OF APPENDICES

CHAPTER 1

INTRODUCTION

1.1 Background of study

As the number of cars has increased rapidly the traffic in big city seems to be busier. Everyone will rush to get in time. Efficient traffic light play important role to make sure the traffic is smooth so that user can plan their time well. Inefficient traffic light will raise a lot of problems such as traffic jam, time wasting and mental stressed for the user and increase the air pollution.

Conventional traffic light is not compatible anymore because the sequence is controlled by using fixed time value [1]. This is where smart traffic light is introduced to solve the traffic problem. A lot of method has been developed by the traffic light vendors to increase the efficiency of the traffic light such as timer based with fuzzy logic control using sensor, automatic control by using camera based detection and manual control by using remote control based on camera monitoring. [2][3]

One of the smart traffic light systems is using fuzzy logic [1]. In this project, traffic light using fuzzy logic will be developed to increase the efficiency of the traffic light system. Fuzzy logic has rapidly become one of the most successful of today's technologies for developing sophisticated control systems. The reason for which is very simple. Fuzzy logic addresses such applications perfectly as it resembles human decision making with an ability to generate precise solutions from certain or approximate information. It fills an important gap in engineering design methods left vacant by purely mathematical approaches (e.g. linear control design), and purely logic-based approaches (e.g. expert systems) in system design.

While other approaches require accurate equations to model real-world behaviors, fuzzy design can accommodate the ambiguities of real-world human language and logic. It provides both an intuitive method for describing systems in human terms and automates the conversion of those system specifications into effective models.

The use of fuzzy logic in this smart traffic light is to determine the most efficient time extension for each traffic light cycle based on the information of the number of cars obtained from the incremental sensor located at each junction.

1.2 Problem statement

Recently conventional traffic light has several disadvantages that lead to the reducing in efficiency and reliability of traffic light system. One of the disadvantages is during traffic jam or peak traffic hour, the traffic light could not response accurately in controlling the time extension for the traffic light cycle because the conventional traffic light is using fix time cycle. The other disadvantage of the conventional traffic light is that it has no synchronization with other nearest traffic light that will lead in to the traffic jam. For an example the distance between two traffic lights is 20 meters and the first traffic light is having green while the second traffic light is having red on the same direction. This could lead to the traffic jam during peak hour because the cars queue behind the second traffic light will reach up to the first traffic light. Cars behind the first traffic light need to wait until the second traffic light to have the green light to move forward.

A lot of factors must be considered in designing the traffic light system such as the location of the traffic light junction, distance of the nearest traffic light junction and the capacity of car using the junction during day and night. The designer should consider all of the factors before designing the traffic light system. However, conventional traffic light does not meet all of this consideration anymore.

Currently smart traffic light such as fuzzy logic has been used to overcome the disadvantages of the conventional traffic light. Beside of having fix timing for the conventional traffic light, it is better to reduce or extent the time based on the number of cars on each junctions. Sometimes it would be better to pass more cars at the green interval if there are not so many cars waiting behind the red light. On the other hand it would be better to change the green light to red light sooner if there less car at the green interval and many cars waiting behind the red light. This will increase the efficiency of the traffic light. There are no exact mathematical models to make the decision whether to extent or reduce the time. By using fuzzy logic it is relatively easy to determine the near optimal changing of lights for each situation. Fuzzy logic system was found to have good performance compared to human and conventional method [1].

This project will focuses on designing a traffic light using fuzzy logic to determine the time extension for the traffic light junction. The traffic light will be modeled for an intersection of two streets. The design has been implemented using eight incremental sensors that will increment the counts when car passed over it. It will count the number of cars on the street in the range of the allocated sensors. Then it wills decide the time extension based on number of cars in each street.

1.3 Objective and scope of study

1.3.1 Objectives

The objectives of this project are:

- To develop fuzzy logic system by using fuzzy logic tool box in MATLAB.
- To develop circuit that can collect data from the traffic light junction and send it to MATLAB software.
- To integrate fuzzy logic system with circuit developed to form a real time smart traffic light system.

1.3.2 Scope of study

The scope of this project includes the implementation of the fuzzy logic using MATLAB to develop the decision making system for the smart traffic light. Apart from that, Simulink is used to link the toolbox with external circuit and test the developed traffic light system.

Microchip's microcontroller PIC16F877 and PIC16F84 were used to develop the external circuit. The external circuit was developed to count the number of cars arrived and leaved the traffic light junction, get the total number of cars at each junction and send the data to MATLAB. When the data had been processed, MATLAB will send the data back to the external circuit. Then the external circuit will response to the data received.

The microcontroller used were developed using the C programming codes. PIC C Compiler is used to compile the C programming code to hex file.

CHAPTER 2

LITERATURE REVIEW AND THEORY

2.1 Traffic light specification

Traffic light implemented in this project is shown in Figure 1. It consists of four junctions such as north, west, south and east. In designing the traffic light, there are some assumption needs to be made:

i) The junction is an isolated four-way junction with traffic coming from the north, west, south and east directions;
ii) When traffic from the north and south moves, traffic from the west and east stops, and vice versa;
iii) No right and left turns are considered;
iv) The fuzzy logic controller will observe the density of the north and south traffic as one side and the west and east traffic as another side;

There are eight incremental sensors labeled as Sx (x=1, 2, 3, 4, 5, 6, 7, 8) as shown in Figure 1. The first sensor will count number of cars that reach the intersection and the latter one will count number of cars that pass the traffic lights. Every time car passes the sensor, it will increment its value by one. The amount of cars behind the traffic light will be the value different between two sensors.

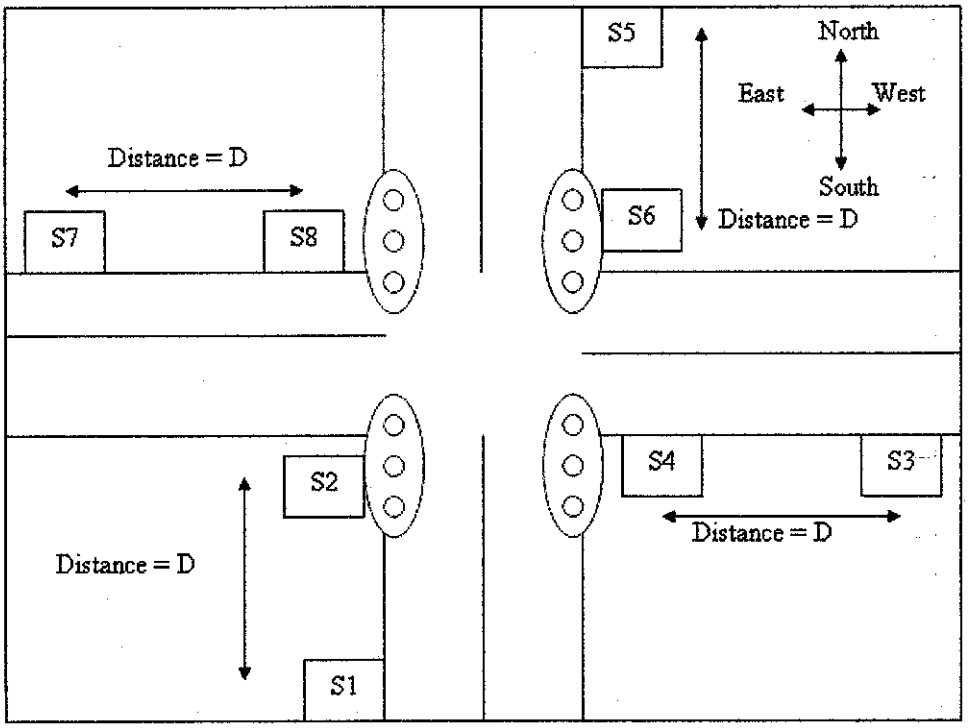


Figure 1: Traffic light junction illustration

The distance D in Figure 1 will determine the amount of car that will be waiting behind the traffic light. In this project the distance used is 40 meters. One car is estimated to be 5 meter long hence maximum number of cars can be between the sensors is 7 cars. Number of cars behind the traffic light is determined by using the equations shown below:

North: Number of cars = S5 – S6	1.1a
South: Number of cars = S1 – S2	1.1b
East: Number of cars = S7 – S8	1.1c
West: Number of cars = S3 – S4	1.1d

All the odd sensors (S1, S3, S5, S7) will count the number of cars arrived at the junction while the even sensors (S2, S4, S6, S8) will count number of car leaving the junction. Number of cars left behind the traffic light will be obtained by subtracting the number of cars arrived at the junction with the number of cars leaving the junction.

Equations shown below are used to get the total number of cars on the street.

$$\text{Number of cars on East-West} = (S7 - S8) + (S3 - S4) \quad 1.2a$$

$$\text{Number of cars on North-South} = (S5 - S6) + (S1 - S2) \quad 1.2b$$

The sum from the equation above will be the input parameters to the fuzzy logic that will be categorized as:

v) Cars behind red light

vi) Cars behind green light

The overall picture of how the system work is illustrated in Figure 2. Number of cars from north, south, east and west junction will be the input to the fuzzy decision making system. Based on the input the fuzzy system will decide the time extension for the current traffic light cycle.

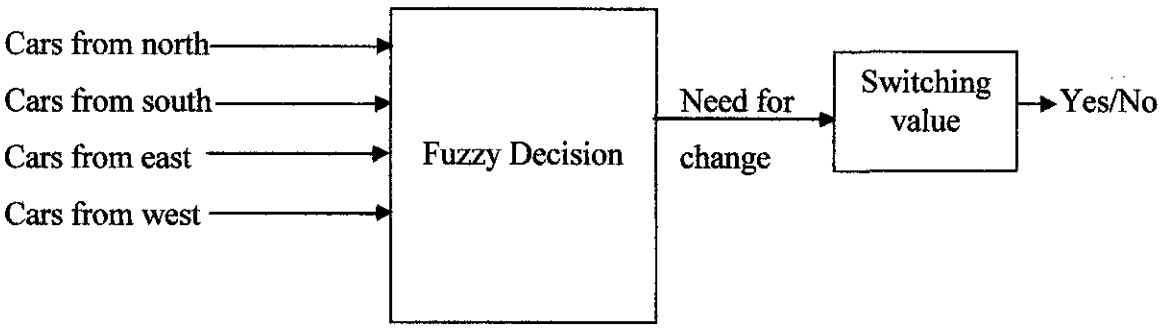


Figure 2: Fuzzy system

2.2 Project Block Diagram

The complete system of this project is shown in the project block diagram in Figure 3. There are 8 blocks that represent the incremental sensor, 4 blocks represent the subtraction circuit, 2 blocks represent addition circuit, 1 block represent Fuzzy Logic system in MATLAB and 1 block represent the traffic light circuit. All of the incremental sensors were developed by using PIC16F84 and the other circuits including the subtraction, addition and traffic light circuit were developed usingPIC16F877.

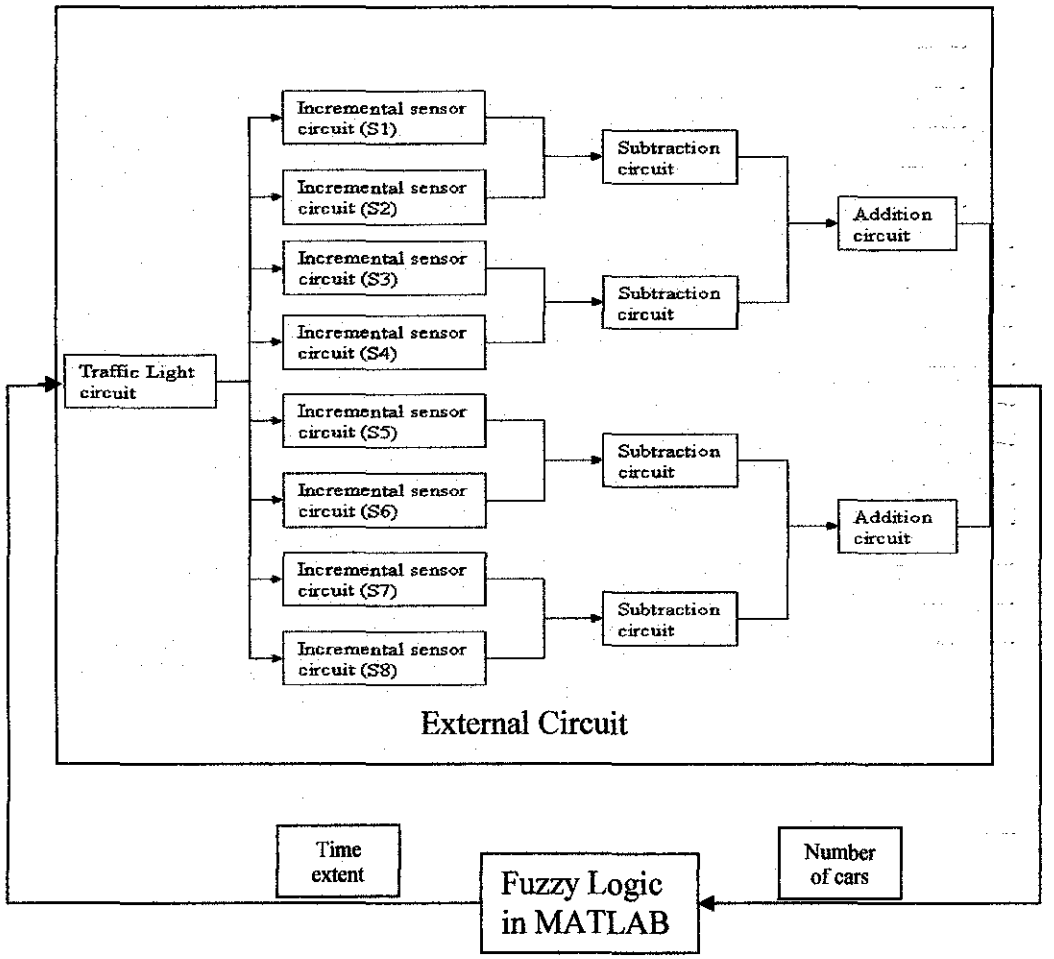


Figure 3: Project block diagram

The traffic light will start with giving green light to North-South street and giving red to East-West street. The incremental sensor will count the number of cars in 5 seconds. Then it will send the data to the subtraction circuit to get the number of cars at each junction. The data then will be sent to the addition circuit to get the total number of cars on North-South street and East-West street. After obtaining the total number of cars on each street, the data will be sent to the Fuzzy Logic system for the extension time decision making. When the decision has been made, the data will be sent to the traffic light circuit.

2.3 Fuzzy Logic

The project uses fuzzy logic toolbox to make the decision for the time extension. The reason of using fuzzy logic in the decision making is because it can control nonlinear system that would be impossible to model mathematically and it processes user defined rules that will be the standard to the target control system. It can be modified and adjusted easily to improve or alter the system performance.

Fuzzy logic theory was found by Professor Lofti Zadeh at thee University of California in 1965. He proposed a paradigm shift that first gained acceptance in the Far East and its successful application has ensured its adoption around the world. A paradigm is a set of rules and regulations which defines boundaries and tells us what to do to be successful in solving problems within these boundaries. The development of fuzzy set theory from conventional bivalent set theory is a paradigm shift. Bivalent set of theory is limiting a humanistic problem mathematically.[4]

Fuzzy decision processing is divided into three parts [1]:

i) Fuzzification
ii) Rule Evaluation
iii) Defuzzification

2.3.1 Fuzzification

This step consists of dividing each input parameter into a set of overlapping membership functions [5]. The range and shape of each membership function will be defined and labeled. In this project the car density behind the red and green light has been divided into the following membership function [1] shown in Table 1 and 2:

Table 1: Cars behind red

Label	Range
Low	[0,2]
Medium	[1,3]
High	[2,6]
Heavy	[5,7]

Table 2: Cars behind green

Label	Range
Low	[0,1]
Medium	[0,3]
High	[1,5]
Heavy	[4,7]

The output function which is the time extension has been divided into the following membership function shown in Table 3:

Table 3: Time extension (seconds)

Label	Range
Short	[0,3]
Medium	[1,6]
Long	[3,8]
Very long	[6,10]

2.3.2 Rule Evaluation

First rules are formulated using AND and OR operators. The format used is if then else. For an example if cars behind red are low AND cars behind green are low the time extension is short. Since there are 2 inputs and each have 4 membership functions, therefore the can be 4x4=16 rules. The rules set in this project are shown in Table 4 below:

Table 4: Rules set for this project

No.	Rules
1	If (cars behind red is low) AND (cars behind green is low) then (time extent is short)
2	If (cars behind red is medium) AND (cars behind green is low) then (time extent is short)
3	If (cars behind red is high) AND (cars behind green is low) then (time extent is short)
4	If (cars behind red is chaos) AND (cars behind green is low) then (time extent is short)
5	If (cars behind red is low) AND (cars behind green is medium) then (time extent is medium)
6	If (cars behind red is medium) AND (cars behind green is medium) then (time extent is medium)
7	If (cars behind red is high) AND (cars behind green is medium) then (time extent is short)
8	If (cars behind red is chaos) AND (cars behind green is medium) then (time extent is short)
9	If (cars behind red is low) AND (cars behind green is high) then (time extent is long)
10	If (cars behind red is medium) AND (cars behind green is high) then (time extent is long)
11	If (cars behind red is high) AND (cars behind green is high) then (time extent is long)
12	If (cars behind red is chaos) AND (cars behind green is high) then (time extent is medium)
13	If (cars behind red is low) AND (cars behind green is chaos) then (time extent is very long)
14	If (cars behind red is medium) AND (cars behind green is chaos) then (time extent is very long)
15	If (cars behind red is high) AND (cars behind green is chaos) then (time extent is long)
16	If (cars behind red is chaos) AND (cars behind green is chaos) then (time extent is long)

MIN_MAX algorithm has been used to calculate the rule strengths. This algorithm is widely used because it is relatively simple to implement on a computer. The MIN-MAX algorithm deals with one rule at the time and uses only a few memory bytes.

2.3.3 Defuzzification

This step is the process converting the fuzzified input into crisp output [5]. This process is done by the MATLAB software based on the rules evaluation.

2.4 MATLAB

MATLAB stands for matrix laboratory. It integrates computation, visualization, and programming in an easy-to-use environment where problems and solutions are expressed in familiar mathematical notation. Graphical user interface (GUI) is provided for different type of applications. The user friendly environment can help the user to decrease the

development time. There are a lot of support document that can help the user to understand the theory that lies behind the application. [5]

There is a Fuzzy Logic Toolbox in MATLAB software. Fuzzy Logic Toolbox is a collection of functions built on the MATLAB numeric computing environment. It provides tools to the user to create and edit fuzzy inference systems within the framework of MATLAB [5]. The fuzzy systems can be integrated into simulations with Simulink. This toolbox provides graphical user interface (GUI) tools to assist the user to design the system. The toolbox provides three categories of tools:

i) Command line functions
ii) Graphical interactive tools
iii) Simulink blocks and examples

2.5 Simulink

Simulink is a software package for modeling, simulating, and analyzing dynamic systems. It supports linear and nonlinear systems, modeled in continuous time, sampled time, or a hybrid of the two [5].

There are 2 purposes of using Simulink in this project that are:

i) Testing the system by inserting random input
ii) Connecting Fuzzy Logic toolbox with the traffic light circuit

2.6 Microcontroller

Microcontroller is a set of microprocessor system that consists of central processing unit, memory, input/output ports and timer [2]. It is a complete system in a single chip and able to perform simple various tasks to replace the high end microprocessor system.

The microcontroller that will be used in this project is PIC16F877 and PIC16F84. The manufacturer is the Microchip. The chips are small and consume very low power and have a high reliability. It can be programmed either by using its own assembly language or C programming code.

The microcontroller PIC16F84 is a high performance RISC (reduced set instruction computer) single-cycle microcontroller equipped with 13 input/output pins. It has 68 bytes of data RAM, 64 bytes of data EEPROM, 14 bit wide instruction word, and 8 bit wide data bytes that enable the chip to perform various command and instruction.[6]

The microcontroller PIC16F877 is also a high performance RISC (reduced set instruction computer) single-cycle microcontroller equipped with 33 input/output pins. It has 8K x 14 words of FLASH Program Memory, 368 x 8 bytes of Data Memory (RAM), 256 x 8 bytes of EEPROM Data Memory, 14 bit wide instruction word, and 8 bit wide data bytes.[7]

2.7 PIC C Compiler

PIC C Compiler is second party software that was developed to compile C programming code for the Microchip microcontroller. The Microchip Company only produces MPLAB software to program the PIC by using assembly language. The PIC C Compiler will compile the C programming code and produces hex file that can be read by the microcontroller.[2]

2.8 WARP 13 Programmer

WARP 13 programmer is used to load the hex file into the microcontroller. The hex code is the machine code for the microchip microcontroller. WARP 13 has the ability to erase all the machine codes that already loaded in the microcontroller, blank, verify, read and programmed the microcontroller. It can verify some setting that being done in the programming code such as the clock setting, and some special feature such as fuses. WARP 13 is really simple and easy to be used.

CHAPTER 3

METHODOLOGY

3.1 Flow chart of the Smart Traffic Light Project

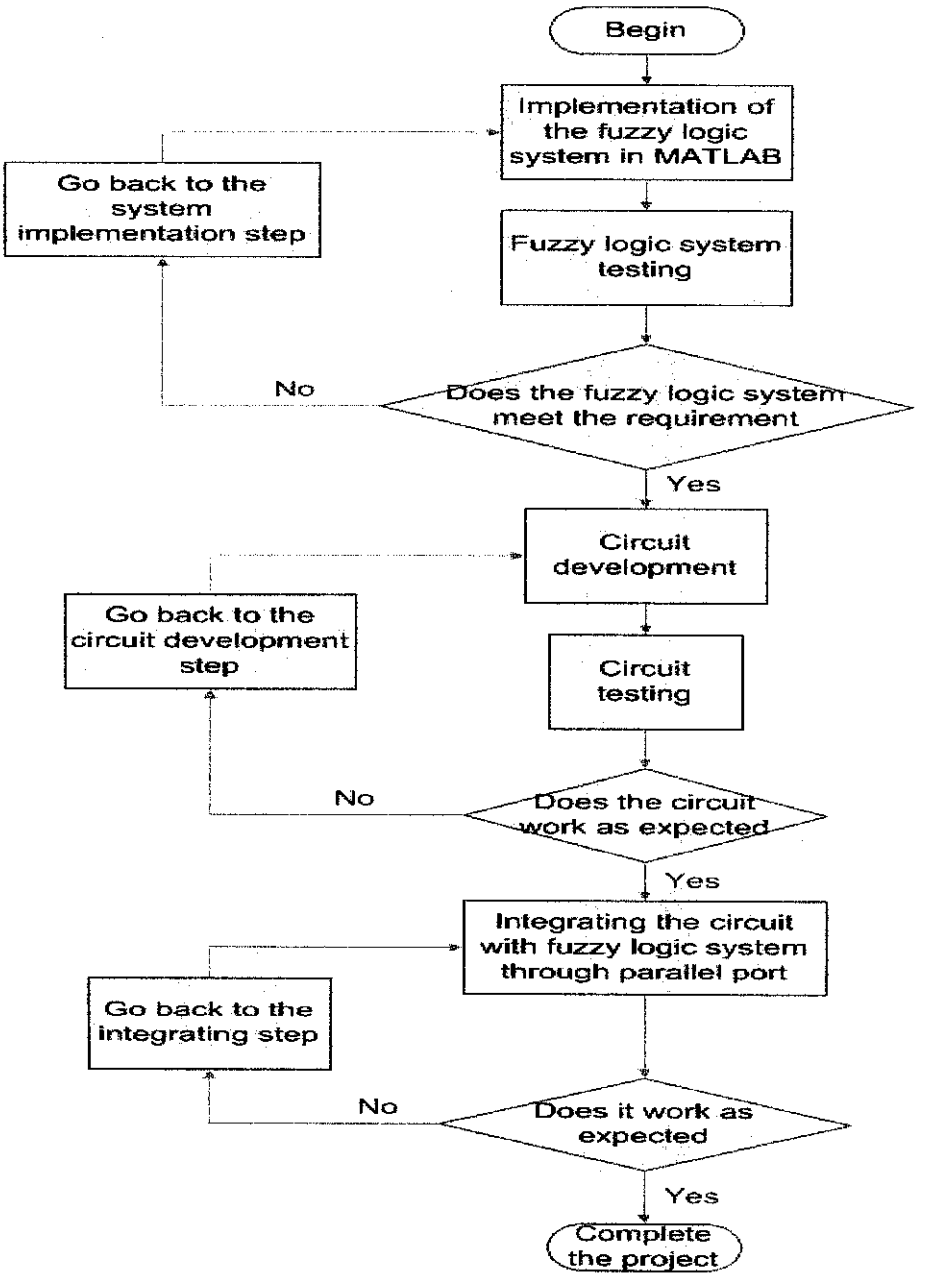


Figure 4: Flow chart of the Smart Traffic Light Project

As shown in Figure, 4 there are 5 steps taken to complete this project. The first step is to develop the fuzzy logic system using the configuration set in Chapter 2. When completed developing the fuzzy logic system, it will be tested using two methods that will be explain later. If the system met the requirement, the next step can be implemented that is to start developing the circuit. If the system does not meet the requirement, the first step must be implemented again.

The circuit development step is divided into two parts that are C code programming development and hardware development which will be explained later. If the circuit works as expected, next step can be implemented that is to integrate the fuzzy logic system with the circuit. If the circuit does not work as expected, the circuit development step needs to be implemented again.

The final step is to connect the developed circuit with the fuzzy logic system. If the final step work as expected hence the project is completed and if it is no, the final step need to be implemented again.

3.1.1 Implementation of the Fuzzy Logic System in MATLAB

There is some useful information provided by in MATLAB need to be read before starting the fuzzy logic system development. There two methods of developing the fuzzy logic systems that is:

i) By using the GUI

ii) By using the command line

It is better to develop the fuzzy logic system by using GUI because it can save times and easy to alter. There are three GUI windows need to be set in completing the fuzzy logic system development that is:

i) FIS (Fuzzy Inference System) editor

ii) Membership function editor

iii) Rule editor

3.1.1.1 FIS editor

The FIS editor will be the main window. Numbers of input and output can be set in FIS editor. The main setting for the system will be set in this window such as the configuration of the AND method, OR method, Aggregation, Implication and Defuzzification [5]. Labeling the input, output and the system name is done in this window.

3.1.1.2 Membership function editor

The membership function editor will be the window to input all the membership function being set early in the theory part. The right type of membership function need to be selected here. The input parameter can be set by changing the value in the space provided in the window. Final step was to label all the parameter inserted.

3.1.1.2 Rule editor

The rule editor window is used to set the rules based on the input and output parameter being set previously. These rules depend on human judgment. There two methods used in the rule editor that if AND, then and if OR, then [5]. The weight of each rule can be set based on the system designed. The system designed can be tested by using the rule viewer window just by inserting random input.

3.1.2 Fuzzy Logic system testing

There are two method can be used to test the fuzzy logic system. The first one is by using the rule editor window and the second one by using the Simulink.

3.1.2.1 Fuzzy Logic system testing using Simulink

Simulink consist of various blocks that represent various function. Some manuals on basic block function provided in MATLAB need to be read first. The blocks used in simulating the fuzzy logic system developed are:

i) Random number block: used to generate random number as the input
ii) Mux: to select input to the fuzzy logic controller
iii) Scope: to view the generated input and output
iv) Fuzzy logic controller: to integrate the fuzzy logic system being developed with other block

There some basic configuration setting needs to be change in the configuration parameter so that the simulation could be successful.

3.1.2.2 Fuzzy Logic system testing using Rule Editor Window

Rule Editor Window can be used to test the fuzzy logic system. There is a slot provided for the user to key in any random input. After entering the input, it will automatically give the output result.

3.1.3 Circuit development

The circuit development step is divided into two that are c programming code development and hardware development.

3.1.3.1 C code programming development for microcontroller

The C code programming development was break down into four parts:

i) Incremental sensor program: to count the number of car passing through it
ii) Subtraction program: to get the number of cars at each junction
iii) Addition program: to get the sum of cars on each street

iv) Traffic light program: to control the traffic flow based on the decision received from MATLAB.

The programming development break down will ease the process of developing and debugging the code.

3.1.3.2 Hardware development

In this step, all of the microcontroller used will be connected together to complete and test the circuit.

3.1.4 Circuit testing

All the codes developed need to be tested in the hardware. There are some basic settings in the microcontroller need to be discovered so that it will work well as expected. The circuit testing was done separately for each part to ease the debugging process.

3.1.5 Integrating the circuit with fuzzy logic system though parallel port

In this final step the tested circuit will be connected to the fuzzy logic system through the parallel port.

3.2 Tools required

This project required integration of software and hardware to produce the desired model. The following table shows the components used:

Table 5: Tools required

No	Hardware	No	Software
1	Microcontroller PIC16F84	1	MATLAB 7.0
2	Microcontroller PIC16F877	2	PIC C Compiler
3	Breadboard	3	WARP 13
4	4 MHz crystal oscillator		
5	LED		

CHAPTER 4
RESULTS AND DISCUSSION

4.1 MATLAB simulation

The MATLAB simulation for fuzzy logic system is the most important part of this project where it will be the point to measure the level of smartness of the traffic light system that will be designed. The more precise the output generated the more efficient the traffic light will be.

The accuracy of the output generated will depend on the configuration set in three main GUI windows stated in the Chapter 3. The main window for the FIS editor is shown in Figure 5 below.

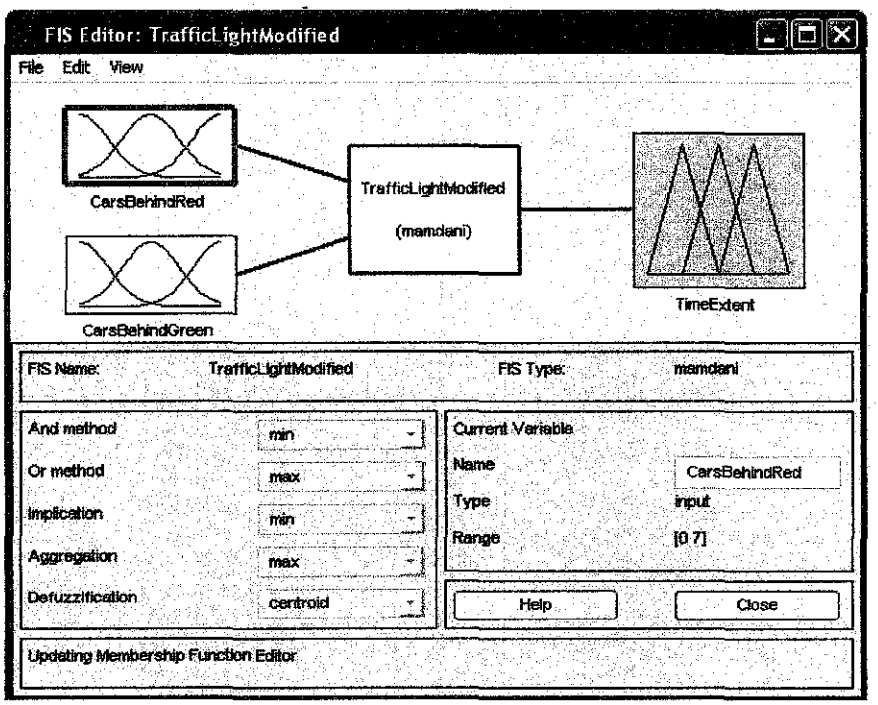


Figure 5: FIS Editor window

All of the main configurations are set in this window. The project was saved as TrafficLight as shown in Figure 5. The Mamdani's fuzzy inference method was used

in this project instead of Sugeno. The main difference between Mamdani and Sugeno is that the Sugeno output membership functions are either linear or constant. The reason of selecting Mamdani to be the fuzzy inference method is because it is intuitive, widespread acceptance and well suited to human input [5]. The configuration for the AND method, OR method, implication, aggregation, and defuzzification are set as shown in Figure 5 because the algorithm used in this project is MIN-MAX algorithm.

The input and output can be added in this window. There are two inputs and one output used for these projects that are CarsBehindRed, CarsBehindGreen and TimeExtent as shown in Figure 5. To configure the setting for the input and output, Membership Function Editor window as shown in Figure 6 need to opened.

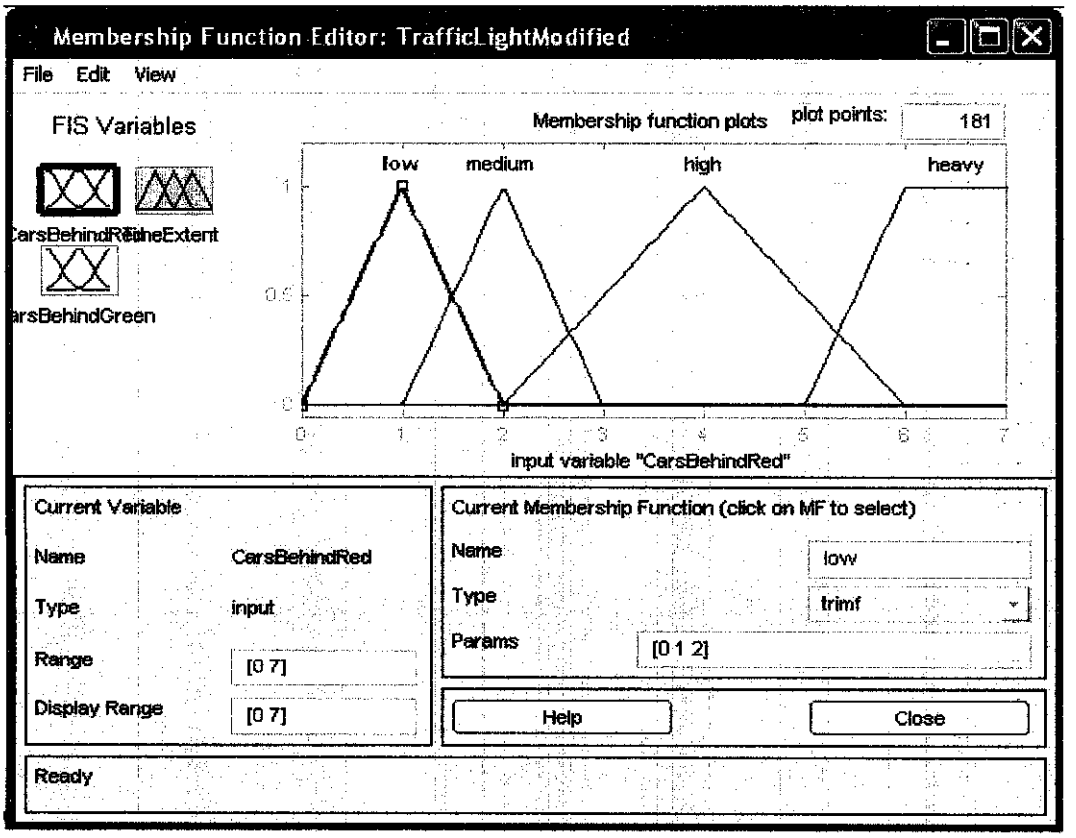


Figure 6: Membership Function Editor window (CarsBehindRed)

The membership function of the input (CarsBehindRed), input (CarsBehindGreen) and the output (TimeExtent) is set as shown in Figure 6, Figure 7 and Figure 8

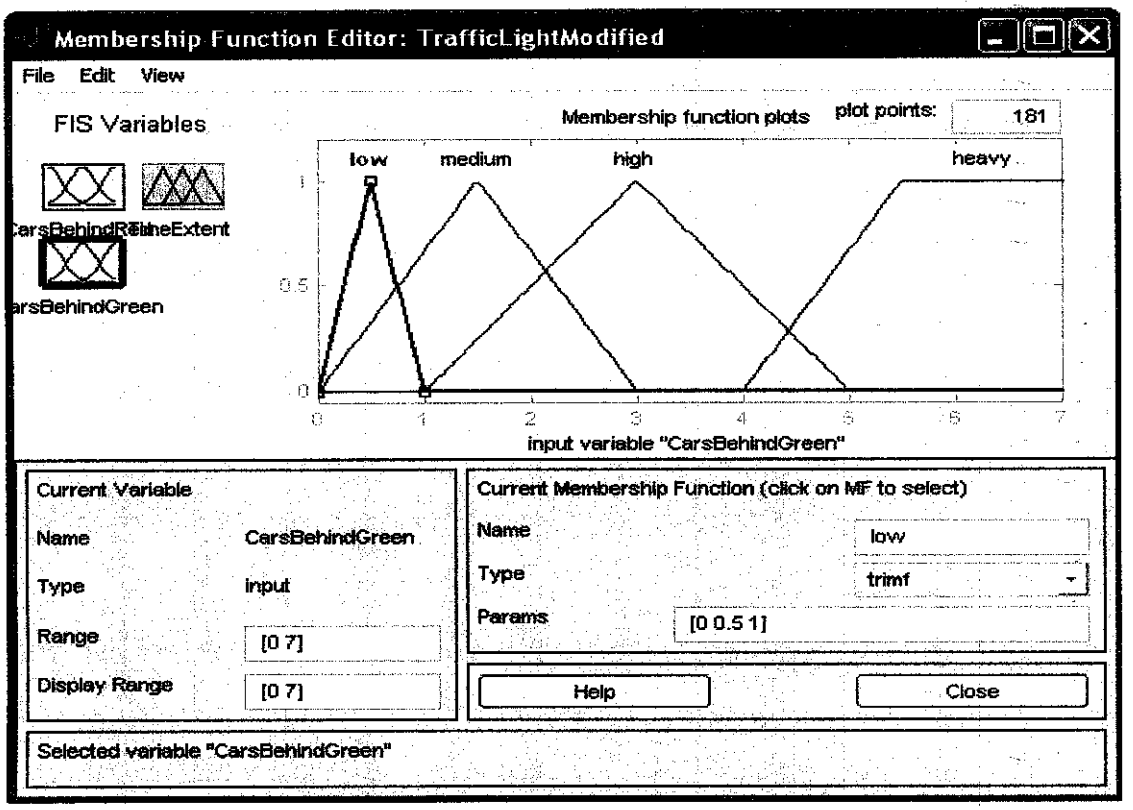


Figure 7: Membership Function Editor window (CarsBehindGreen)

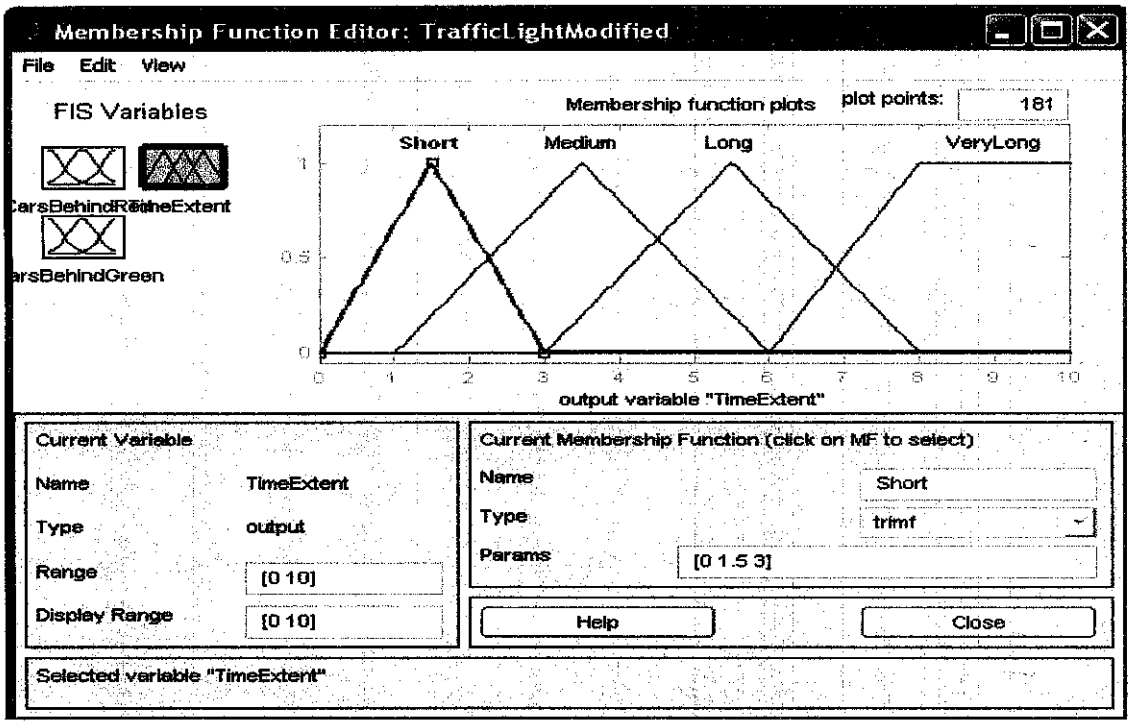


Figure 8: Membership Function Editor window (TimeExtent)

All of the setting shown in Figure 6, 7 and 8 are based on the parameter set in Chapter 2. All the type of membership function used is using triangular type named as trimf [5]. Except for the heavy membership function in both inputs and VeryLong membership function in the output are using trapezoidal type named as trapmf [5]. The parameters of each membership function can be set by inserting the value at "Params" box provided in the GUI. The range is set referring to the minimum and maximum value of the inputs and output.

The next step is to set the rules for the fuzzy logic system based on the membership function set before. The Rule Editor window is shown in Figure 9.

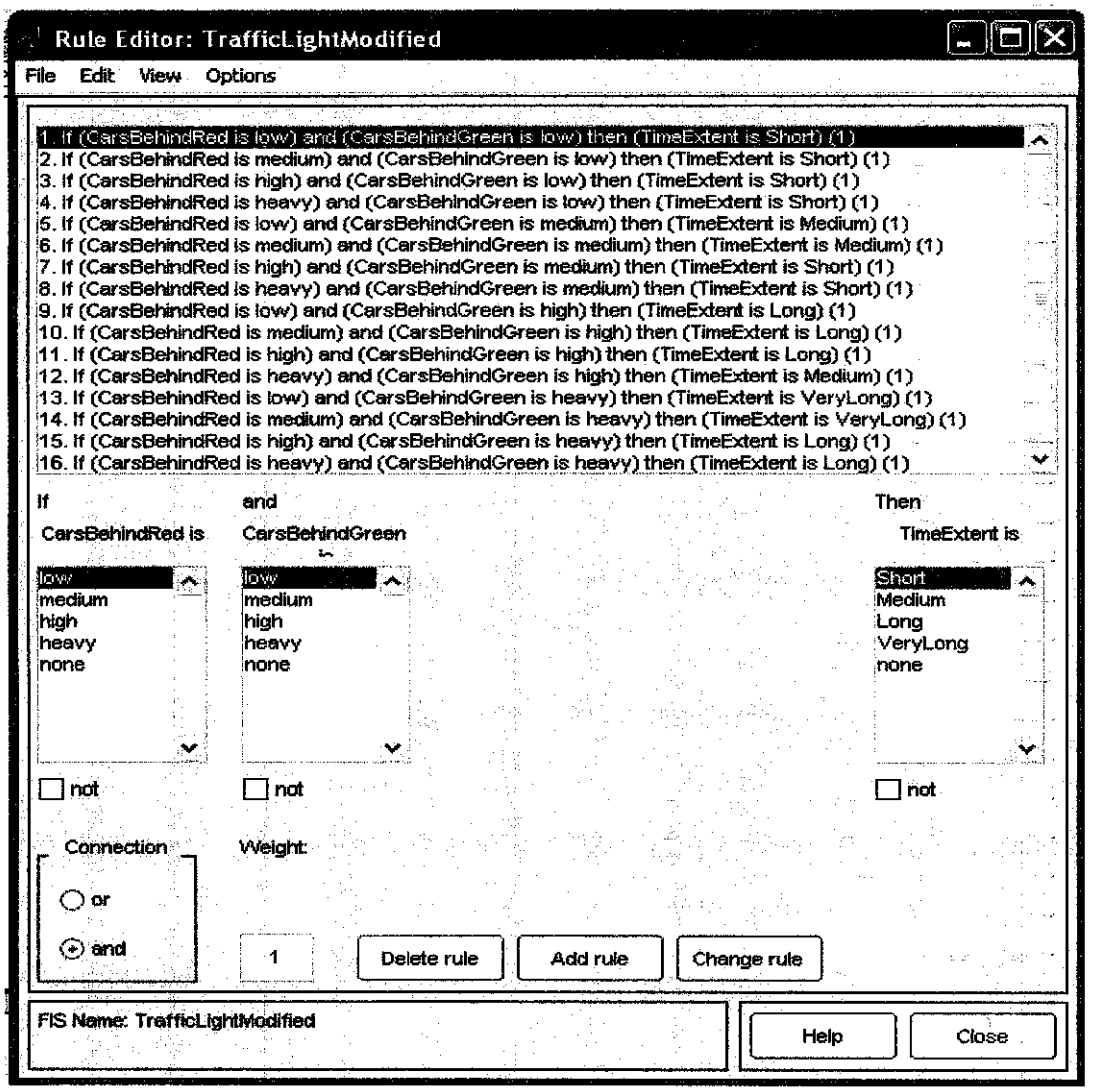


Figure 9: Rule Editor Window

There are overall of 16 rules being set for this project. In this window, rules can be added by clicking the "Add rule" button and deleted by clicking the "Delete rule" button provided. To add the rule, one of the membership function from each input and output is selected and the connection between the input must be set either OR or AND. Then the "Add rule" button is pressed and the rule added will appear at the bottom of the window.

When all of the rules have been set, the next step was to test the fuzzy logic system. Rule Viewer window shown in Figure 10 was used to test the fuzzy logic system. There is a box provided to set the input as shown in figure 10. The input that going to be tested is inserted in that box and the enter button was pressed to generate the output. The output is shown at the bottom right of the window labeled as TimeExtent.

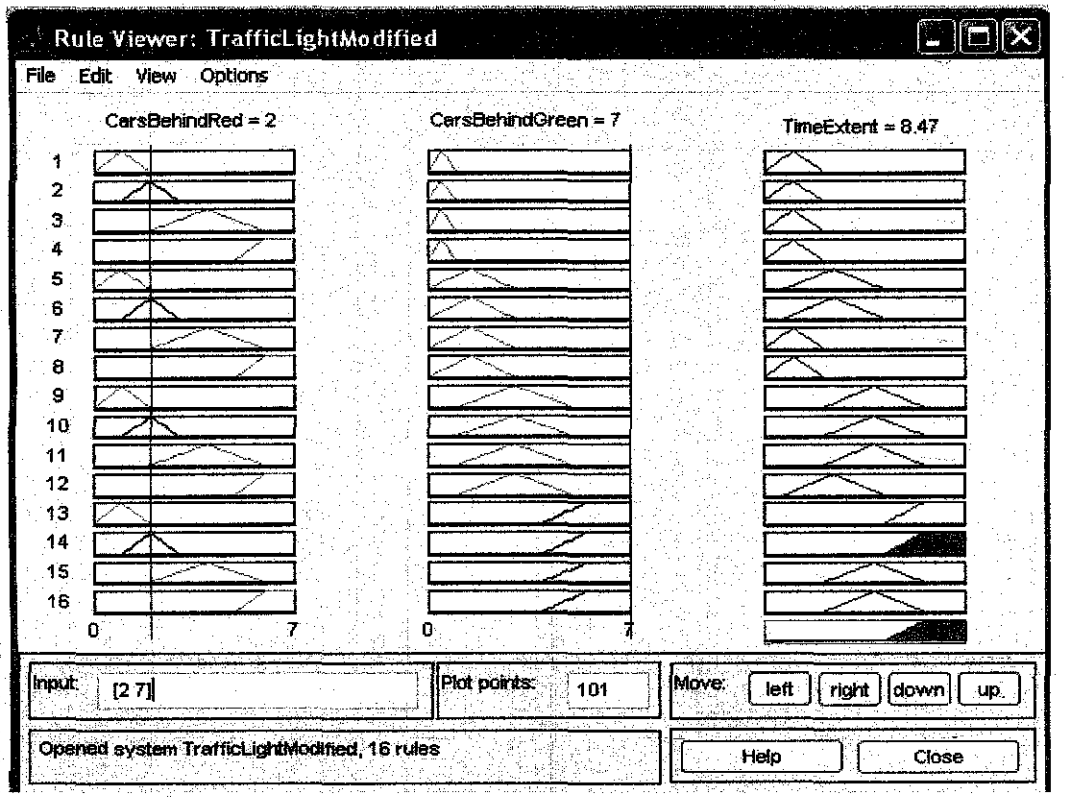


Figure 10: Rule Viewer Window

The value tested need to be entered one by one to check the output generated by the fuzzy logic system. All the outputs generated are shown in Table 6.

Table 6: Outputs generated from rule Viewer Window

Cars behind red	Cars behind green	Time extent (seconds)	Rounded (seconds)
1	1	2.72	3
1	2	3.5	4
1	3	4.17	4
1	4	4.5	5
1	5	4.69	5
1	6	5.5	6
1	7	5.5	6
2	1	2.53	3
2	2	3.5	4
2	3	4.03	4
2	4	4.39	4
2	5	4.82	5
2	6	5.5	6
2	7	5.5	6
3	1	2.72	3
3	2	3.5	4
3	3	4.17	4
3	4	4.5	5
3	5	4.69	5
3	6	5.5	6
3	7	5.5	6
4	1	2.53	3
4	2	3.5	4
4	3	4.03	4
4	4	4.39	4
4	5	4.82	5
4	6	5.5	6
4	7	5.5	6
5	1	2.72	3
5	2	3.14	3
5	3	3.81	4
5	4	4.17	4
5	5	4.3	4
5	6	5.5	6
5	7	5.5	6
6	1	1.5	2
6	2	1.5	2
6	3	3.47	3
6	4	4	4
6	5	4.27	4
6	6	5.5	6
6	7	5.5	6

4.2 Simulink Simulation

The next step was to test the fuzzy logic system using the Simulink. Simulink is a software package for modeling, simulating, and analyzing dynamic systems. It supports linear and nonlinear systems, modeled in continuous time, sampled time, or a hybrid of the two. Instant access to all the analysis tools in MATLAB is provided, so that results can be analyzed and visualized.[5]

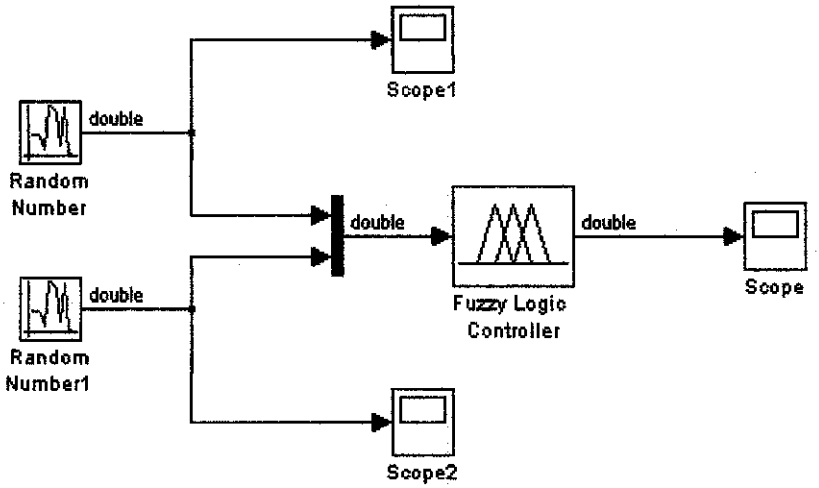


Figure 11: Simulink blocks to test the fuzzy logic system

The Simulink blocks developed are shown in Figure 11. The random number block is used to generate random number ranging from 0 to 7 to be the input. Scope 1 and Scope 2 are used to view the generated input. Both block of the random input are connected to the multiplexer before entering the fuzzy logic controller block. This is because the fuzzy logic controller box cannot receive 2 inputs at one time. Multiplexer need to be added so that the fuzzy logic controller will be able to receive more than two inputs. The Scope is used to view the generated output that is the extended time.

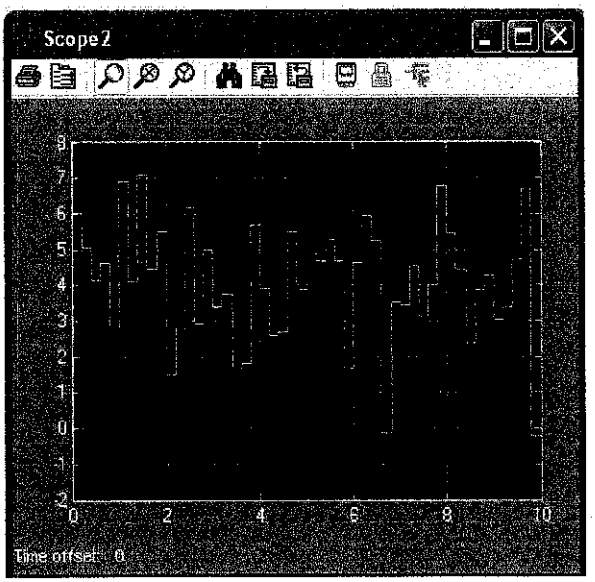


Figure 12: Input 1- Cars behind green

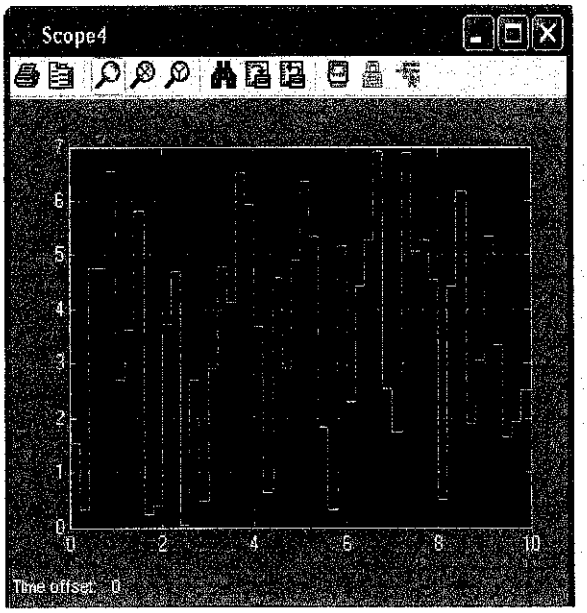


Figure 13: Input 1- Cars behind red

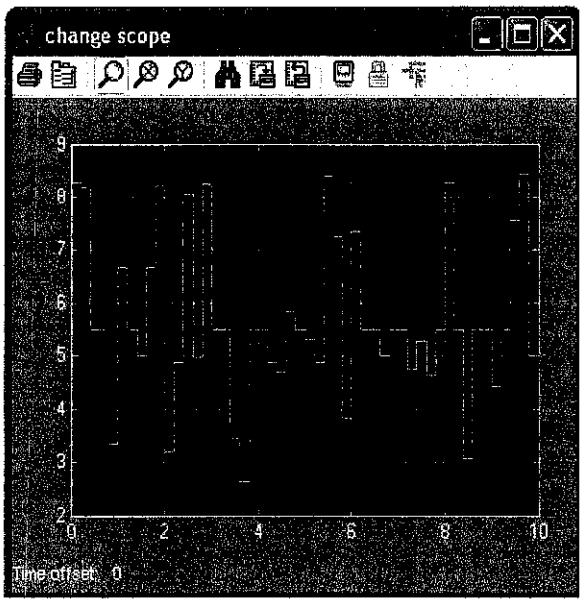


Figure 14: Output- Time extension

Figure 12 and 13 are the generated input from the random input generator and Figure 14 is the output from the fuzzy logic system. The output obtained from the Simulink simulation is different from the output obtained from the Rule Editor window. This is because the inputs in the Rule Editor Window are all round number where the input from Simulink simulation are mixed of round number and fraction number. The Simulink simulation was done to test the system capability and to observe the output generated.

4.3 C code programming and circuit development

The development of the C code programming must be parallel with the development of the circuit because the connection of the circuit will depend on the C programming code developed for the circuit. The schematics of the circuit developed are shown in Figure 15, 16 and 17. The circuit is break down into small part to simplify the explanation. The one shown in Figure 16 is part of the complete circuit that will do the counting job. The counting job will start from the incremental sensors that will count the number of cars pass trough it up until getting the total number of cars on each street.

Then the total number of cars on each street will be sent to the MATLAB. But due to the time constraint, the integrating part of the circuit with MATLAB software could not be completed. As an alternative, all the generated output values obtained in Table 6 will be programmed inside the PIC16F877 of the traffic light circuit schematic. The result obtained is still the same either the circuit obtained the output directly from MATLAB or the output have been programmed inside the PIC16F877. The only disadvantages is that the parameter set for the membership function could not be alter instantly if the output have been programmed inside the chip. The PIC16F877 need to be reprogrammed to alter the membership function because the changes of the membership function will generate different output.

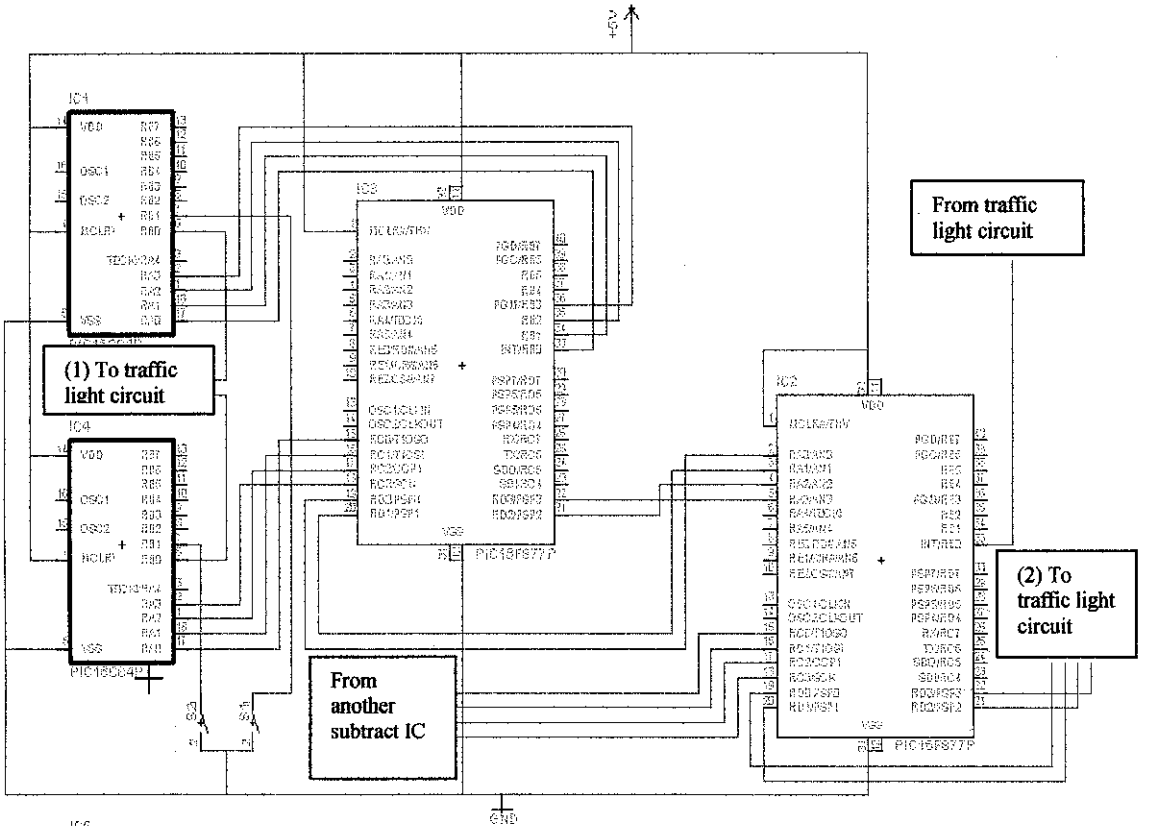


Figure 15: Counter circuit schematic

The counter circuit schematic is shown in Figure 15. The sensors are represented by the two switches S1 and S2 shown in Figure 15. The sensors are connected to ground because the input pins are set as active low. IC1 and IC4 are using PIC16F84 which are

connected to the sensor. It will count the number of cars that pass through the sensor when the enable pin RB0 that is active low is set to low. When the enable pin RB0 is high it will send the data to IC3 that is PIC16F877. The RB0 pin is controlled by the traffic light circuit pin A1. The IC3 will subtract the number of cars detected by S1 from S2 to get the total number of cars behind the traffic light. When the enable pin at IC2 that is active low is set low, the data will be sent to IC2 that is PIC 16F877 to calculate average number of cars on each street. The unconnected port on RC0 to RC3 shown in Figure 13 should be connected to the same circuit like the one connecting the IC1, IC3 and IC4 as shown in Figure 17. This mean that there will 8 PIC16F84 are used to connect to the sensor, 4 PIC16F877 used to get the number of cars behind each traffic light and 2 PIC 16F877 used to get the total number of cars on each street.

The unconnected port RD0 to RD3 will be connected to the traffic light circuit to send the data as shown in Figure 17.

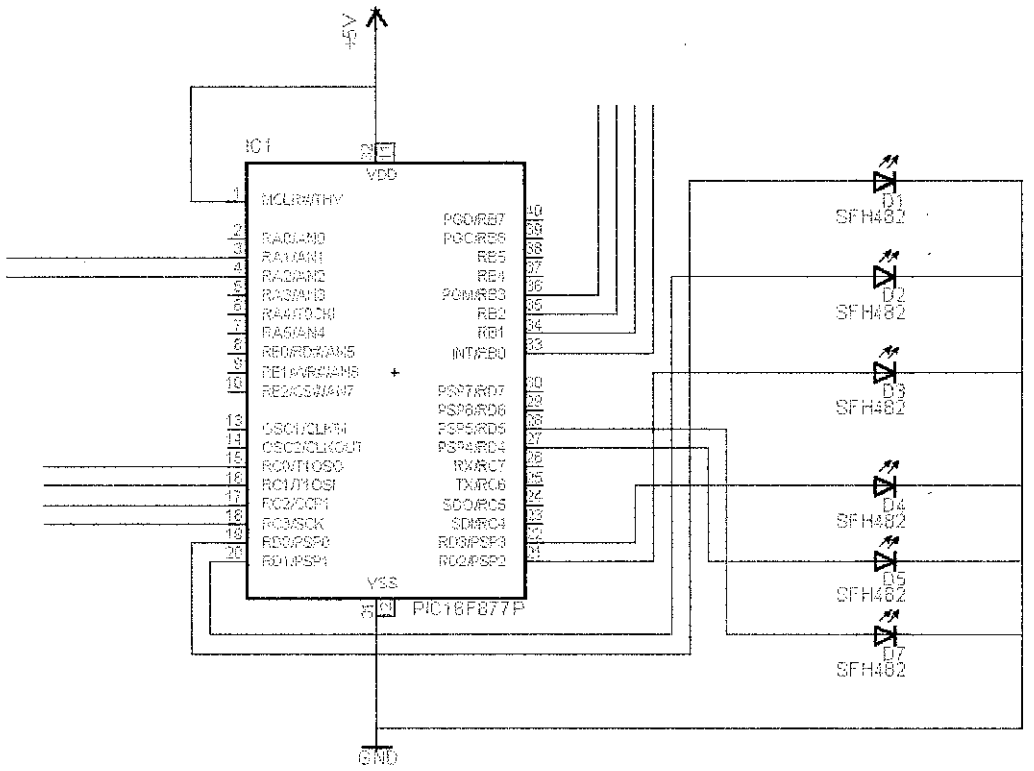


Figure 16: Traffic Light circuit schematic

The traffic light circuit will receive the data though port RB0 to RB3 and RC0 to RC3 from the counter circuit as shown in Figure 16. Then it will change time extension based on the output generated by MATLAB that have been programmed inside the PIC16F877. There are 6 LEDs that consist of 2 green, 2 red, and 2 yellow color to represent the traffic light on the East-West and the North–West street.

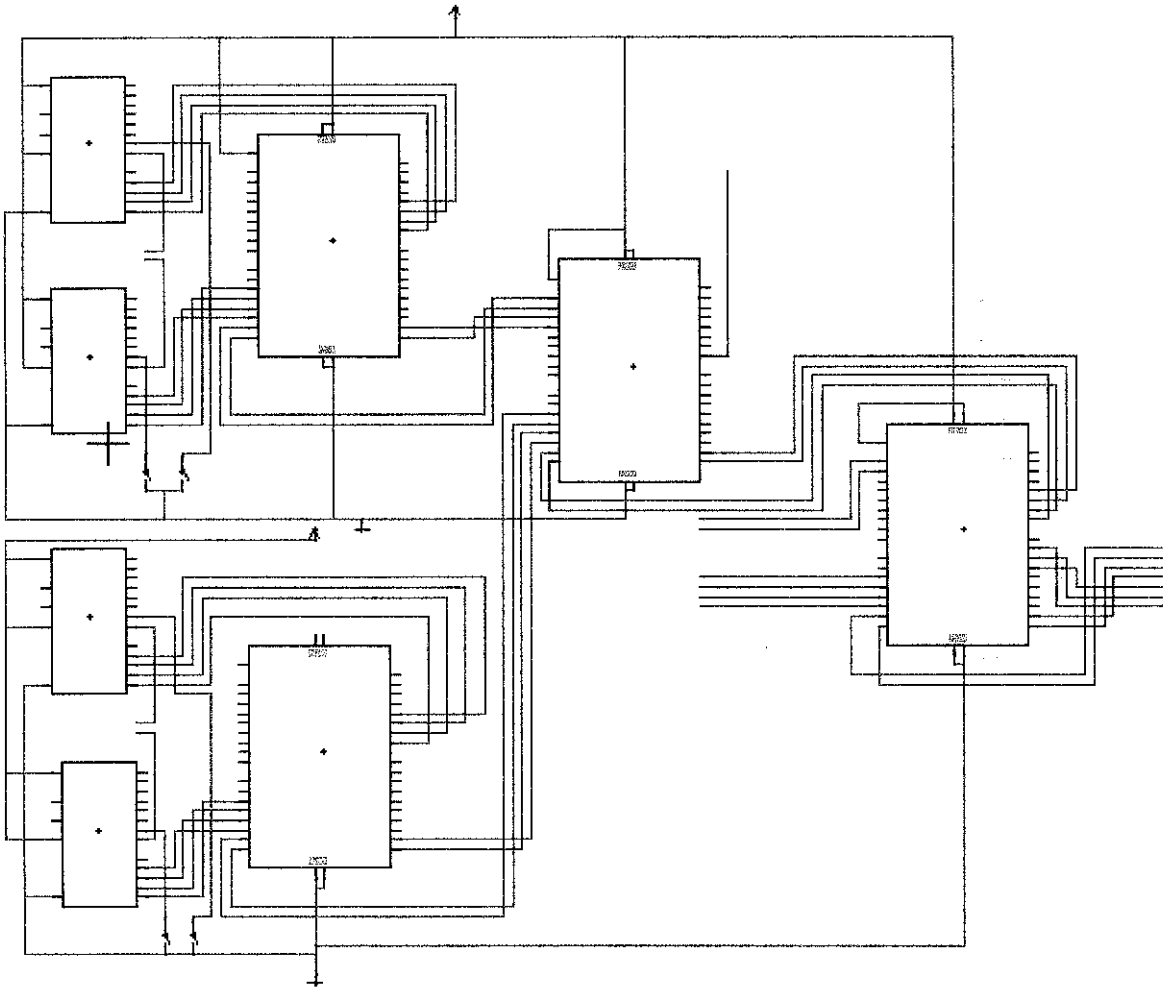


Figure 17: Part of the circuit schematic

The programming codes are divided into four parts. There first part is for the sensor using PIC16F84a, second part is for the subtraction using PIC16F877, third part is for the addition using PIC16F877 and the final part is the traffic light using PIC16F877.All the programming codes are shown in Appendix I.

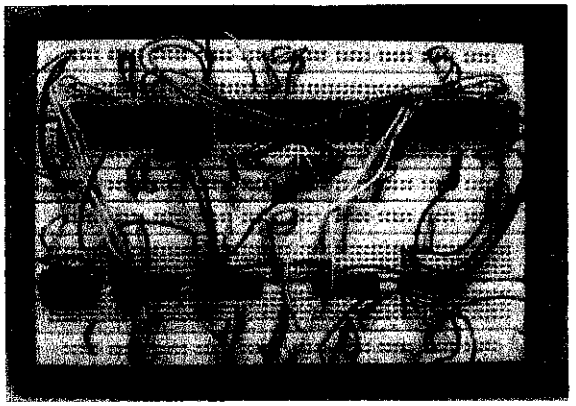


Figure 18: Counter circuit constructed

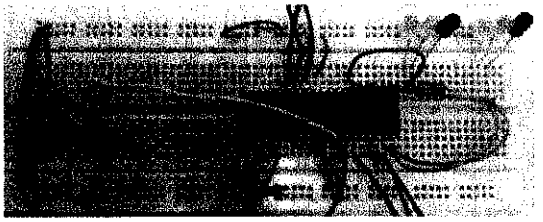


Figure 19: Traffic light circuit constructed

The constructed circuits are shown in Figure 18 and 19. For the time being the circuits were constructed on the breadboard to simplify the debugging step.

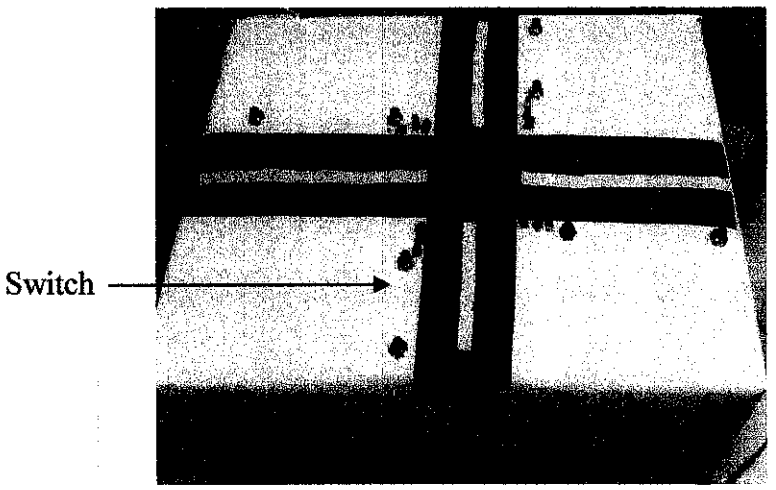


Figure 20: Final product

Figure 20 shows the final product of this project. The switches represent the sensor. There are 8 switches being used to represent 8 incremental sensors. The circuits were hidden in the box.

CHAPTER 5

CONCLUSIONS AND RECOMMENDATION

5.1 Conclusion

The main objective of this project is to build a prototype of smart traffic light using fuzzy logic. The C code programming and the circuit had been developed and tested to obtain the best result. The smart traffic light wills count the number of cars by using the sensor allocated at each junction. the total number of cars will be sent to the subtraction circuit to find the total number of cars behind each traffic light. The result will be sent to the addition circuit to find the total number of cars on the East-West and North-South streets. Then the total number of cars will be sent to the traffic light circuit to decide the time extent by comparing the total number of cars on each street.

The cycle time extent will depend on the number of cars waiting behind the traffic light. Basically when the number of cars behind the red light are higher compare to the number of cars behind the green light, the time extent will be smaller and vice versa. The best time extents were generated from MATLAB based on the rules being set. Based on the generated values, the programming codes and circuits were developed.

The smartness of this traffic light is that it cans response to the traffic condition based on the information received from the sensor. This had increased the traffic light efficiency compared to the conventional traffic light.

5.1 Recommendation

There are a lot of features can be added to this traffic light to increase its level of smartness. One of the features could be added is using different time cycle system depending on the traffic condition. For an example during peak hour, the time cycle system used will be different from the normal hour.

REFERENCES

[1] Dr. Devinder Kaur, Elisa Konga, Esa Konga "Fuzzy Traffic Light Controller" IEEE Journals

[2] Barnett, Cox & O'cull, Embedded C Programming and the Microchip PIC, Thomson Delmar Lerning, 2004

[2] http://www.ameinfo.com/85148.html

[3] http://www.manilatimes.net/others/special/2002/dec/17/20021217spe1.html

[4] Zader Lotfi "Knowledge Representation in Fuzzy Logic" IEEE Journals

[5] Fuzzy Logic Toolbox for use with MATLAB, Fuzzy Logic Toolbox User's Guide, Mathworks Inc., 1997

[6] PIC16F84A Data Sheet, Microchip, Microchip Technology Inc, 2001

[7] PIC16F877 Data Sheet, Microchip, Microchip Technology Inc, 2001

APPENDIX I

Programming code for the sensor circuit

```
#include<16f84a.h>
#use delay(clock=4000000)
#fuses XT, NOWDT
int inc, num;
void main(void)
{

                port_b_pullups(TRUE);
                set_tris_a(0x00);
                set_tris_b(0xff);


while(1)
 {


     inc = 0;
        output_a(0x00);

        while(!input(PIN_B0))
        {

                if(!input(PIN_B1))
                {
                delay_ms(500);
                while(!input(PIN_B1));

                     inc++;
                delay_ms(500);

                     num = inc;
                }

        }

    output_a(num);


 }
}
```

Programming code for the subtraction circuit

```
#include<16f877.h>
#use delay(clock=4000000)
#fuses XT, NOWDT, NOLVP
int a, b, sub;
void main(void)
{


                set_tris_b(0xff);
                set_tris_c(0xff);
                set_tris_d(0x00);



while(1)
 {
        output_d(0x00);
        a=input_b();
        b=input_c();
        sub = a-b;
        output_d(sub);
 }
}
```

Programming code for the addition circuit

```
#include<16f877a.h>
#use delay(clock=4000000)
#fuses XT, NOWDT, NOLVP
int a, b;
void main(void)
{

                set_tris_b(0xff);
                set_tris_c(0xff);
                set_tris_d(0x00);



while(1)
 {

        a=input_b();
        b=input_c();

  output_d(a + b);

 }
}
```

Programming code for the traffic light circuit

```
#include <16F877.h>
#use delay(clock=4000000)
#fuses XT,NOWDT,NOLVP
#use fixed_io(d_outputs=PIN_D7,PIN_D6,PIN_D5,PIN_D4,PIN_D3,PIN_D2,PIN_D1,PIN_D0)
#use fast_io(A)

#use fast_io(B)

#use fast_io(C)

int time_extent();

void main()
{

        while(1)
        {

                set_tris_a(0x00);
                set_tris_b(0xff);
                set_tris_c(0xff);
                set_tris_d(0x00);



                                        //initially start with east-west and west-east has the yellow
                //output_low(PIN_D0);   //RED EW & WE
                //output_high(PIN_D1);  //YELLOW EW & WE
                //output_low(PIN_D2);   //GREEN EW & WE

                //output_high(PIN_D3);  //RED SN & NS
                //output_low(PIN_D6);   //YELLOW SN & NS
                //output_low(PIN_D7);   //GREEN SN & NS

                output_d(0x0A);
                delay_ms(2000);


                                        //east-west, west-east has the red and south-north, north-south
has the green
                //output_high(PIN_D0);  //RED EW & WE
                //output_low(PIN_D1);   //YELLOW EW & WE
                //output_low(PIN_D2);   //GREEN EW & WE

                //output_low(PIN_D3);   //RED SN & NS
                //output_low(PIN_D6);   //YELLOW SN & NS
                //output_high(PIN_D7);  //GREEN SN & NS


                output_d(0x21);
                output_low(PIN_A1);
                delay_ms(5000);
```

```
            time_extent();
    output_high(PIN_A1);

                                                    //south-north and south-north has the yellow

            //output_high(PIN_D0);   //RED EW & WE
            //output_low(PIN_D1);    //YELLOW EW & WE
            //output_low(PIN_D2);    //GREEN EW & WE

            //output_low(PIN_D3);    //RED SN & NS
            //output_high(PIN_D6);   //YELLOW SN & NS
            //output_low(PIN_D7);    //GREEN SN & NS


        output_d(0x11);
            delay_ms(2000);



                                    // south-north, north-south has the red and west-east, east-west
has the green
            //output_low(PIN_D0);    //RED EW & WE
            //output_low(PIN_D1);    //YELLOW EW & WE
            //output_high(PIN_D2);   //GREEN EW & WE

            //output_high(PIN_D3);   //RED SN & NS
            //output_low(PIN_D6);    //YELLOW SN & NS
            //output_low(PIN_D7);    //GREEN SN & NS


            output_d(0x0C);

            output_low(PIN_A1);
            delay_ms(5000);
            time_extent();
    output_high(PIN_A1);


  }
}

        int time_extent(void)
        {


        if((input_b()==0) && (input_c()==0))
            return 0;

            else if((input_b()==0) && (input_c()==1))
            {       delay_ms(2000);
                    return 0;
            }

            else if((input_b()==0) && (input_c()==2))
            {       delay_ms(2000);
```

```
        return 0;
}

else if((input_b()==0) && (input_c()==3))
{       delay_ms(3000);
        return 0;
}

else if((input_b()==0) && (input_c()==4))
{       delay_ms(4000);
        return 0;
}

else if((input_b()==0) && (input_c()==5))
{       delay_ms(4000);
        return 0;
}

else if((input_b()==0) && (input_c()==6))
{       delay_ms(5000);
        return 0;
}

else if((input_b()==0) && (input_c()==7))
{       delay_ms(5000);
        return 0;
}



else if((input_b()==1) && (input_c()==0))
{
        return 0;
}

else if((input_b()==1) && (input_c()==1))
{       delay_ms(3000);
        return 0;
}

else if((input_b()==1) && (input_c()==2))
{       delay_ms(4000);
        return 0;
}

else if((input_b()==1) && (input_c()==3))
{       delay_ms(4000);
        return 0;
}

else if((input_b()==1) && (input_c()==4))
{       delay_ms(5000);
        return 0;
}

else if((input_b()==1) && (input_c()==5))
```

```
{       delay_ms(5000);
        return 0;
}

else if((input_b()==1) && (input_c()==6))
{       delay_ms(6000);
        return 0;
}

else if((input_b()==1) && (input_c()==7))
{       delay_ms(6000);
        return 0;
}



else if((input_b()==2) && (input_c()==0))
{
        return 0;
}

else if((input_b()==2) && (input_c()==1))
{       delay_ms(3000);
        return 0;
}

else if((input_b()==2) && (input_c()==2))
{       delay_ms(4000);
        return 0;
}

else if((input_b()==2) && (input_c()==3))
{       delay_ms(4000);
        return 0;
}

else if((input_b()==2) && (input_c()==4))
{       delay_ms(4000);
        return 0;
}

else if((input_b()==2) && (input_c()==5))
{       delay_ms(5000);
        return 0;
}

else if((input_b()==2) && (input_c()==6))
{       delay_ms(6000);
        return 0;
}

else if((input_b()==2) && (input_c()==7))
{       delay_ms(6000);
        return 0;
}
```

```
else if((input_b()==3) && (input_c()==0))
{
        return 0;
}

else if((input_b()==3) && (input_c()==1))
{       delay_ms(3000);
        return 0;
}

else if((input_b()==3) && (input_c()==2))
{       delay_ms(4000);
        return 0;
}

else if((input_b()==3) && (input_c()==3))
{       delay_ms(4000);
        return 0;
}

else if((input_b()==3) && (input_c()==4))
{       delay_ms(5000);
        return 0;
}

else if((input_b()==3) && (input_c()==5))
{       delay_ms(5000);
        return 0;
}

else if((input_b()==3) && (input_c()==6))
{       delay_ms(6000);
        return 0;
}

else if((input_b()==3) && (input_c()==7))
{       delay_ms(6000);
        return 0;
}


else if((input_b()==4) && (input_c()==0))
{
        return 0;
}

else if((input_b()==4) && (input_c()==1))
{       delay_ms(3000);
        return 0;
}

else if((input_b()==4) && (input_c()==2))
{       delay_ms(4000);
        return 0;
```

```
}

else if((input_b()==4) && (input_c()==3))
{       delay_ms(4000);
        return 0;
}

else if((input_b()==4) && (input_c()==4))
{       delay_ms(4000);
        return 0;
}

else if((input_b()==4) && (input_c()==5))
{       delay_ms(5000);
        return 0;
}

else if((input_b()==4) && (input_c()==6))
{       delay_ms(6000);
        return 0;
}

else if((input_b()==4) && (input_c()==7))
{       delay_ms(6000);
        return 0;
}



else if((input_b()==5) && (input_c()==0))
{
        return 0;
}

else if((input_b()==5) && (input_c()==1))
{       delay_ms(3000);
        return 0;
}

else if((input_b()==5) && (input_c()==2))
{       delay_ms(3000);
        return 0;
}

else if((input_b()==5) && (input_c()==3))
{       delay_ms(4000);
        return 0;
}

else if((input_b()==5) && (input_c()==4))
{       delay_ms(4000);
        return 0;
}

else if((input_b()==5) && (input_c()==5))
{       delay_ms(4000);
```

```
        return 0;
}

else if((input_b()==5) && (input_c()==6))
{       delay_ms(6000);
        return 0;
}

else if((input_b()==5) && (input_c()==7))
{       delay_ms(6000);
        return 0;
}



else if((input_b()==6) && (input_c()==0))
{
        return 0;
}

else if((input_b()==6) && (input_c()==1))
{       delay_ms(2000);
        return 0;
}

else if((input_b()==6) && (input_c()==2))
{       delay_ms(2000);
        return 0;
}

else if((input_b()==6) && (input_c()==3))
{       delay_ms(3000);
        return 0;
}

else if((input_b()==6) && (input_c()==4))
{       delay_ms(4000);
        return 0;
}

else if((input_b()==6) && (input_c()==5))
{       delay_ms(4000);
        return 0;
}

else if((input_b()==6) && (input_c()==6))
{       delay_ms(6000);
        return 0;
}

else if((input_b()==6) && (input_c()==7))
{       delay_ms(6000);
        return 0;
}
```

```
else if((input_b()==7) && (input_c()==0))
{
        return 0;
}

else if((input_b()==7) && (input_c()==1))
{       delay_ms(2000);
        return 0;
}

else if((input_b()==7) && (input_c()==2))
{       delay_ms(2000);
        return 0;
}

else if((input_b()==7) && (input_c()==3))
{       delay_ms(3000);
        return 0;
}

else if((input_b()==7) && (input_c()==4))
{       delay_ms(4000);
        return 0;
}

else if((input_b()==7) && (input_c()==5))
{       delay_ms(4000);
        return 0;
}

else if((input_b()==7) && (input_c()==6))
{       delay_ms(6000);
        return 0;
}

else if((input_b()==7) && (input_c()==7))
{       delay_ms(6000);
        return 0;
}

}
```

APPENDIX II

MICROCHIP

PIC16F84A

18-pin *Enhanced* FLASH/EEPROM 8-Bit Microcontroller

High Performance RISC CPU Features:

- Only 35 single word instructions to learn
- All instructions single-cycle except for program branches which are two-cycle
- Operating speed: DC - 20 MHz clock input
  DC - 200 ns instruction cycle
- 1024 words of program memory
- 68 bytes of Data RAM
- 64 bytes of Data EEPROM
- 14-bit wide instruction words
- 8-bit wide data bytes
- 15 Special Function Hardware registers
- Eight-level deep hardware stack
- Direct, indirect and relative addressing modes
- Four interrupt sources:
  - External RB0/INT pin
  - TMR0 timer overflow
  - PORTB<7:4> interrupt-on-change
  - Data EEPROM write complete

Peripheral Features:

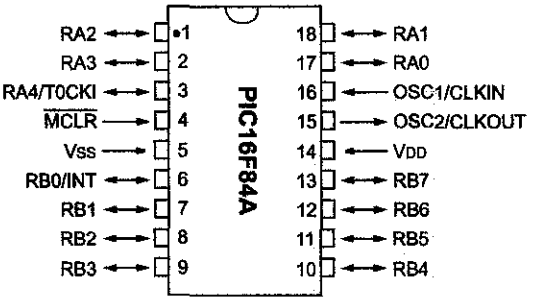
- 13 I/O pins with individual direction control
- High current sink/source for direct LED drive
  - 25 mA sink max. per pin
  - 25 mA source max. per pin
- TMR0: 8-bit timer/counter with 8-bit programmable prescaler

Special Microcontroller Features:

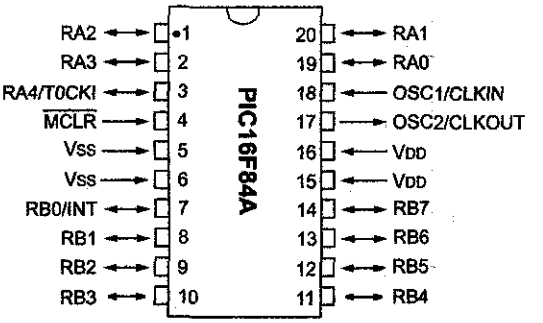
- 10,000 erase/write cycles *Enhanced* FLASH Program memory typical
- 10,000,000 typical erase/write cycles EEPROM Data memory typical
- EEPROM Data Retention > 40 years
- In-Circuit Serial Programming™ (ICSP™) - via two pins
- Power-on Reset (POR), Power-up Timer (PWRT), Oscillator Start-up Timer (OST)
- Watchdog Timer (WDT) with its own On-Chip RC Oscillator for reliable operation
- Code protection
- Power saving SLEEP mode
- Selectable oscillator options

Pin Diagrams

PDIP, SOIC



SSOP



CMOS Enhanced FLASH/EEPROM Technology:

- Low power, high speed technology
- Fully static design
- Wide operating voltage range:
  - Commercial: 2.0V to 5.5V
  - Industrial: 2.0V to 5.5V
- Low power consumption:
  - < 2 mA typical @ 5V, 4 MHz
  - 15 µA typical @ 2V, 32 kHz
  - < 0.5 µA typical standby current @ 2V

TABLE 1-1: PIC16F84A PINOUT DESCRIPTION

Pin Name	PDIP No.	SOIC No.	SSOP No.	I/O/P Type	Buffer Type	Description
OSC1/CLKIN	16	16	18	I	ST/CMOS(3)	Oscillator crystal input/external clock source input.
OSC2/CLKOUT	15	15	19	O	—	Oscillator crystal output. Connects to crystal or resonator in Crystal Oscillator mode. In RC mode, OSC2 pin outputs CLKOUT, which has 1/4 the frequency of OSC1 and denotes the instruction cycle rate.
MCLR	4	4	4	I/P	ST	Master Clear (Reset) input/programming voltage input. This pin is an active low RESET to the device.
						PORTA is a bi-directional I/O port.
RA0	17	17	19	I/O	TTL	
RA1	18	18	20	I/O	TTL	
RA2	1	1	1	I/O	TTL	
RA3	2	2	2	I/O	TTL	
RA4/T0CKI	3	3	3	I/O	ST	Can also be selected to be the clock input to the TMR0 timer/counter. Output is open drain type.
						PORTB is a bi-directional I/O port. PORTB can be software programmed for internal weak pull-up on all inputs.
RB0/INT	6	6	7	I/O	TTL/ST(1)	RB0/INT can also be selected as an external interrupt pin.
RB1	7	7	8	I/O	TTL	
RB2	8	8	9	I/O	TTL	
RB3	9	9	10	I/O	TTL	
RB4	10	10	11	I/O	TTL	Interrupt-on-change pin.
RB5	11	11	12	I/O	TTL	Interrupt-on-change pin.
RB6	12	12	13	I/O	TTL/ST(2)	Interrupt-on-change pin. Serial programming clock.
RB7	13	13	14	I/O	TTL/ST(2)	Interrupt-on-change pin. Serial programming data.
VSS	5	5	5,6	P	—	Ground reference for logic and I/O pins.
VDD	14	14	15,16	P	—	Positive supply for logic and I/O pins.

Legend: I= input O = Output I/O = Input/Output P = Power
— = Not used TTL = TTL input ST = Schmitt Trigger input

Note 1: This buffer is a Schmitt Trigger input when configured as the external interrupt.
2: This buffer is a Schmitt Trigger input when used in Serial Programming mode.
3: This buffer is a Schmitt Trigger input when configured in RC oscillator mode and a CMOS input otherwise.

APPENDIX III

PIC16F87X

28/40-Pin 8-Bit CMOS FLASH Microcontrollers

Devices Included in this Data Sheet:

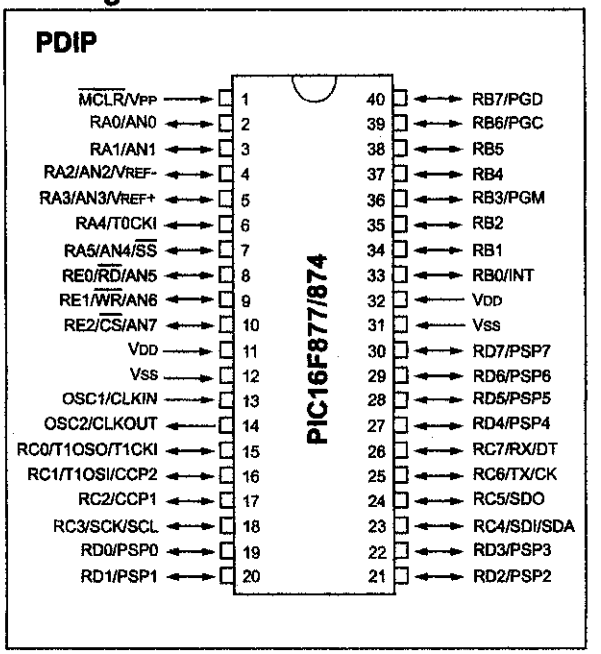
- PIC16F873
- PIC16F874
- PIC16F876
- PIC16F877

Microcontroller Core Features:

- High performance RISC CPU
- Only 35 single word instructions to learn
- All single cycle instructions except for program branches which are two cycle
- Operating speed: DC - 20 MHz clock input
  DC - 200 ns instruction cycle
- Up to 8K x 14 words of FLASH Program Memory,
  Up to 368 x 8 bytes of Data Memory (RAM)
  Up to 256 x 8 bytes of EEPROM Data Memory
- Pinout compatible to the PIC16C73B/74B/76/77
- Interrupt capability (up to 14 sources)
- Eight level deep hardware stack
- Direct, indirect and relative addressing modes
- Power-on Reset (POR)
- Power-up Timer (PWRT) and
  Oscillator Start-up Timer (OST)
- Watchdog Timer (WDT) with its own on-chip RC oscillator for reliable operation
- Programmable code protection
- Power saving SLEEP mode
- Selectable oscillator options
- Low power, high speed CMOS FLASH/EEPROM technology
- Fully static design
- In-Circuit Serial Programming™ (ICSP) via two pins
- Single 5V In-Circuit Serial Programming capability
- In-Circuit Debugging via two pins
- Processor read/write access to program memory
- Wide operating voltage range: 2.0V to 5.5V
- High Sink/Source Current: 25 mA
- Commercial, Industrial and Extended temperature ranges
- Low-power consumption:
  - < 0.6 mA typical @ 3V, 4 MHz
  - 20 µA typical @ 3V, 32 kHz
  - < 1 µA typical standby current

Pin Diagram

PDIP



Peripheral Features:

- Timer0: 8-bit timer/counter with 8-bit prescaler
- Timer1: 16-bit timer/counter with prescaler, can be incremented during SLEEP via external crystal/clock
- Timer2: 8-bit timer/counter with 8-bit period register, prescaler and postscaler
- Two Capture, Compare, PWM modules
  - Capture is 16-bit, max. resolution is 12.5 ns
  - Compare is 16-bit, max. resolution is 200 ns
  - PWM max. resolution is 10-bit
- 10-bit multi-channel Analog-to-Digital converter
- Synchronous Serial Port (SSP) with SPI™ (Master mode) and I²C™ (Master/Slave)
- Universal Synchronous Asynchronous Receiver Transmitter (USART/SCI) with 9-bit address detection
- Parallel Slave Port (PSP) 8-bits wide, with external RD, WR and CS controls (40/44-pin only)
- Brown-out detection circuitry for Brown-out Reset (BOR)

TABLE 1-1: PIC16F873 AND PIC16F876 PINOUT DESCRIPTION

Pin Name	DIP Pin#	SOIC Pin#	I/O/P Type	Buffer Type	Description
OSC1/CLKIN	9	9	I	ST/CMOS(3)	Oscillator crystal input/external clock source input.
OSC2/CLKOUT	10	10	O	—	Oscillator crystal output. Connects to crystal or resonator in crystal oscillator mode. In RC mode, the OSC2 pin outputs CLKOUT which has 1/4 the frequency of OSC1, and denotes the instruction cycle rate.
$\overline{MCLR}$/VPP	1	1	I/P	ST	Master Clear (Reset) input or programming voltage input. This pin is an active low RESET to the device.
					PORTA is a bi-directional I/O port.
RA0/AN0	2	2	I/O	TTL	RA0 can also be analog input0.
RA1/AN1	3	3	I/O	TTL	RA1 can also be analog input1.
RA2/AN2/VREF-	4	4	I/O	TTL	RA2 can also be analog input2 or negative analog reference voltage.
RA3/AN3/VREF+	5	5	I/O	TTL	RA3 can also be analog input3 or positive analog reference voltage.
RA4/T0CKI	6	6	I/O	ST	RA4 can also be the clock input to the Timer0 module. Output is open drain type.
RA5/$\overline{SS}$/AN4	7	7	I/O	TTL	RA5 can also be analog input4 or the slave select for the synchronous serial port.
					PORTB is a bi-directional I/O port. PORTB can be software programmed for internal weak pull-up on all inputs.
RB0/INT	21	21	I/O	TTL/ST(1)	RB0 can also be the external interrupt pin.
RB1	22	22	I/O	TTL	
RB2	23	23	I/O	TTL	
RB3/PGM	24	24	I/O	TTL	RB3 can also be the low voltage programming input.
RB4	25	25	I/O	TTL	Interrupt-on-change pin.
RB5	26	26	I/O	TTL	Interrupt-on-change pin.
RB6/PGC	27	27	I/O	TTL/ST(2)	Interrupt-on-change pin or In-Circuit Debugger pin. Serial programming clock.
RB7/PGD	28	28	I/O	TTL/ST(2)	Interrupt-on-change pin or In-Circuit Debugger pin. Serial programming data.
					PORTC is a bi-directional I/O port.
RC0/T1OSO/T1CKI	11	11	I/O	ST	RC0 can also be the Timer1 oscillator output or Timer1 clock input.
RC1/T1OSI/CCP2	12	12	I/O	ST	RC1 can also be the Timer1 oscillator input or Capture2 input/Compare2 output/PWM2 output.
RC2/CCP1	13	13	I/O	ST	RC2 can also be the Capture1 input/Compare1 output/ PWM1 output.
RC3/SCK/SCL	14	14	I/O	ST	RC3 can also be the synchronous serial clock input/output for both SPI and I^2C modes.
RC4/SDI/SDA	15	15	I/O	ST	RC4 can also be the SPI Data In (SPI mode) or data I/O (I^2C mode).
RC5/SDO	16	16	I/O	ST	RC5 can also be the SPI Data Out (SPI mode).
RC6/TX/CK	17	17	I/O	ST	RC6 can also be the USART Asynchronous Transmit or Synchronous Clock.
RC7/RX/DT	18	18	I/O	ST	RC7 can also be the USART Asynchronous Receive or Synchronous Data.
VSS	8, 19	8, 19	P	—	Ground reference for logic and I/O pins.
VDD	20	20	P	—	Positive supply for logic and I/O pins.

Legend: I = input O = output I/O = input/output P = power
— = Not used TTL = TTL input ST = Schmitt Trigger input

Note 1: This buffer is a Schmitt Trigger input when configured as the external interrupt.
2: This buffer is a Schmitt Trigger input when used in Serial Programming mode.
3: This buffer is a Schmitt Trigger input when configured in RC oscillator mode and a CMOS input otherwise.

ABLE 1-2: PIC16F874 AND PIC16F877 PINOUT DESCRIPTION

Pin Name	DIP Pin#	PLCC Pin#	QFP Pin#	I/O/P Type	Buffer Type	Description
OSC1/CLKIN	13	14	30	I	ST/CMOS(4)	Oscillator crystal input/external clock source input.
OSC2/CLKOUT	14	15	31	O	—	Oscillator crystal output. Connects to crystal or resonator in crystal oscillator mode. In RC mode, OSC2 pin outputs CLKOUT which has 1/4 the frequency of OSC1, and denotes the instruction cycle rate.
MCLR/VPP	1	2	18	I/P	ST	Master Clear (Reset) input or programming voltage input. This pin is an active low RESET to the device.
						PORTA is a bi-directional I/O port.
RA0/AN0	2	3	19	I/O	TTL	RA0 can also be analog input0.
RA1/AN1	3	4	20	I/O	TTL	RA1 can also be analog input1.
RA2/AN2/VREF-	4	5	21	I/O	TTL	RA2 can also be analog input2 or negative analog reference voltage.
RA3/AN3/VREF+	5	6	22	I/O	TTL	RA3 can also be analog input3 or positive analog reference voltage.
RA4/T0CKI	6	7	23	I/O	ST	RA4 can also be the clock input to the Timer0 timer/ counter. Output is open drain type.
RA5/SS/AN4	7	8	24	I/O	TTL	RA5 can also be analog input4 or the slave select for the synchronous serial port.
						PORTB is a bi-directional I/O port. PORTB can be soft-ware programmed for internal weak pull-up on all inputs.
RB0/INT	33	36	8	I/O	TTL/ST(1)	RB0 can also be the external interrupt pin.
RB1	34	37	9	I/O	TTL	
RB2	35	38	10	I/O	TTL	
RB3/PGM	36	39	11	I/O	TTL	RB3 can also be the low voltage programming input.
RB4	37	41	14	I/O	TTL	Interrupt-on-change pin.
RB5	38	42	15	I/O	TTL	Interrupt-on-change pin.
RB6/PGC	39	43	16	I/O	TTL/ST(2)	Interrupt-on-change pin or In-Circuit Debugger pin. Serial programming clock.
RB7/PGD	40	44	17	I/O	TTL/ST(2)	Interrupt-on-change pin or In-Circuit Debugger pin. Serial programming data.

Legend: I = input O = output I/O = input/output P = power
— = Not used TTL = TTL input ST = Schmitt Trigger input

Note 1: This buffer is a Schmitt Trigger input when configured as an external interrupt.
2: This buffer is a Schmitt Trigger input when used in Serial Programming mode.
3: This buffer is a Schmitt Trigger input when configured as general purpose I/O and a TTL input when used in the Parallel Slave Port mode (for interfacing to a microprocessor bus).
4: This buffer is a Schmitt Trigger input when configured in RC oscillator mode and a CMOS input otherwise.

TABLE 1-2: PIC16F874 AND PIC16F877 PINOUT DESCRIPTION (CONTINUED)

Pin Name	DIP Pin#	PLCC Pin#	QFP Pin#	I/O/P Type	Buffer Type	Description
						PORTC is a bi-directional I/O port.
RC0/T1OSO/T1CKI	15	16	32	I/O	ST	RC0 can also be the Timer1 oscillator output or a Timer1 clock input.
RC1/T1OSI/CCP2	16	18	35	I/O	ST	RC1 can also be the Timer1 oscillator input or Capture2 input/Compare2 output/PWM2 output.
RC2/CCP1	17	19	36	I/O	ST	RC2 can also be the Capture1 input/Compare1 output/PWM1 output.
RC3/SCK/SCL	18	20	37	I/O	ST	RC3 can also be the synchronous serial clock input/ output for both SPI and I^2C modes.
RC4/SDI/SDA	23	25	42	I/O	ST	RC4 can also be the SPI Data In (SPI mode) or data I/O (I^2C mode).
RC5/SDO	24	26	43	I/O	ST	RC5 can also be the SPI Data Out (SPI mode).
RC6/TX/CK	25	27	44	I/O	ST	RC6 can also be the USART Asynchronous Transmit or Synchronous Clock.
RC7/RX/DT	26	29	1	I/O	ST	RC7 can also be the USART Asynchronous Receive or Synchronous Data.
						PORTD is a bi-directional I/O port or parallel slave port when interfacing to a microprocessor bus.
RD0/PSP0	19	21	38	I/O	ST/TTL(3)	
RD1/PSP1	20	22	39	I/O	ST/TTL(3)	
RD2/PSP2	21	23	40	I/O	ST/TTL(3)	
RD3/PSP3	22	24	41	I/O	ST/TTL(3)	
RD4/PSP4	27	30	2	I/O	ST/TTL(3)	
RD5/PSP5	28	31	3	I/O	ST/TTL(3)	
RD6/PSP6	29	32	4	I/O	ST/TTL(3)	
RD7/PSP7	30	33	5	I/O	ST/TTL(3)	
						PORTE is a bi-directional I/O port.
RE0/$\overline{RD}$/AN5	8	9	25	I/O	ST/TTL(3)	RE0 can also be read control for the parallel slave port, or analog input5.
RE1/$\overline{WR}$/AN6	9	10	26	I/O	ST/TTL(3)	RE1 can also be write control for the parallel slave port, or analog input6.
RE2/$\overline{CS}$/AN7	10	11	27	I/O	ST/TTL(3)	RE2 can also be select control for the parallel slave port, or analog input7.
VSS	12,31	13,34	6,29	P	—	Ground reference for logic and I/O pins.
VDD	11,32	12,35	7,28	P	—	Positive supply for logic and I/O pins.
NC	—	1,17,28, 40	12,13, 33,34		—	These pins are not internally connected. These pins should be left unconnected.

Legend: I = input O = output I/O = input/output P = power
— = Not used TTL = TTL input ST = Schmitt Trigger input

Note 1: This buffer is a Schmitt Trigger input when configured as an external interrupt.
2: This buffer is a Schmitt Trigger input when used in Serial Programming mode.
3: This buffer is a Schmitt Trigger input when configured as general purpose I/O and a TTL input when used in the Parallel Slave Port mode (for interfacing to a microprocessor bus).
4: This buffer is a Schmitt Trigger input when configured in RC oscillator mode and a CMOS input otherwise.