INTELLIGENT MONITORING AND CONTROLLING FOR A PARKING SYSTEM (CAR PARK)

By

NIK OTHMAN BIN MOHAMED

2251

A project dissertation submitted to the Electrical & Electronics Engineering Programme in Partial Fulfillment of the requirements for the

Bachelor of Engineering (Hons)

(Electrical & Electronics Engineering)

Universiti Teknologi Petronas
Bandar Seri Iskandar
31750 Tronoh
Perak Darul Ridzuan

CERTIFICATION OF APPROVAL

INTELLIGENT MONITORING AND CONTROLLING FOR A PARKING SYSTEM (CAR PARK)

by

Nik Othman Bin Mohamed

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

Approved:

[illegible]

Dr. Taj Mohammad Baloch
Senior Lecturer
Electrical and Electronics Engineering
New Academic Block No 22
Universiti Teknologi PETRONAS
Bandar Seri Iskandar
31750 Tronoh, Perak Darul Ridzuan, MALAYSIA

Dr. Taj Mohammad Baloch
Senior Lecturer,
Electrical & Electronics Engineering Department,
Universiti Teknologi Petronas.

UNIVERSITI TEKNOLOGI PETRONAS
TRONOH, PERAK

June 2005

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.

[illegible]

Nik Othman Bin Mohamed
2251,
Electrical & Electronics Engineering,
Universiti Teknologi Petronas.

ABSTRACT

The purpose of ***Final Year Project*** is as a continuation to develop a framework, which will enhance student's skills in the process of applying knowledge, expanding thoughts by solving problem independently and presenting findings at the final stage of execution. In understanding the rhythm of *Final Year Project*, each student is given one project to be done to fulfill the requirements stated.

In this project, he author has selected a project titled '***Intelligent Monitoring and Controlling for a Parking System (Car Park)***'. This project is basically to design and construct a simulation of parking system which uses the PIC Microcontroller with the input from sensory circuit. This report will discuss in detail about the project whereby it is divided into five main parts as summarized below.

The first part is the ***Introduction*** part, which comprises of a brief project background, problem statement and identification as well as the objective and scope of study. This part will describe more specifically about the background of the project and identify the circumstances that bring out the project for real. While the objective and scope of study will clarify the boundary of the project.

The second part covers the ***Literature Review*** part, which consists of some researches done regarding the project. Next part is ***Methodology*** that will highlight the stages of implementing this project. There are two sections in this part which are *procedure identification* and *tools required.* The fourth part is ***Result and Discussion*** which consists of all the solutions and designing parts of parking system which already been done until this week.

Lastly, ***Conclusion and Recommendation*** part is included to wrap up the entire report followed by the list of references.

ACKNOWLEDGEMENTS

First, the author would like to take this opportunity to express his sincere gratitude to all parties involved in making my Electrical and Electronic Engineering Final Year Design Project success. The author's deepest appreciation goes to project supervisor, Dr. Taj Mohammad Baloch for his support, guidance, patient and also, for his brilliant ideas in solving problems that the author faced throughout the design project.

The author also would like to record the special thank to Ms Nasreen, the coordinator of Final Year Project. Without their management and coordination, the flow of this project might not be as expected.

Special thanks also to Ms Siti Hawa who has given the author the numerous technical supports as a lab technician. She has been helpful to prepare the electronics components and parts for the project. Very special thanks to Mr. Ahmad Nizar B Muhaidin, Mr Razif B Saidun and Mr Azizan B Hashim, who have taught and gave advices to the author in many areas which was not known to the author before. Their help in giving ideas and programming skills is also one of the key factors of the success of this project.

Last but not least, the author would like to thanks the author's family and friends which have been supportive during the period of this project were held. Not to forget to all others who have help in the project directly and indirectly and might not be mentioned here.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

1.1 Background of Study

Car park system is one of the most important public facilities in each country in the world. In Malaysia, the development of this system is quite impressive because of the increasing demands and needs from the users especially in the big towns and the cities. In order to fulfil the demands, the development is not focused on the amount of the system only, but also in the design of the system itself.

Currently, most of the existing car parks do not have a systematic system. Most of them have been managed manually and are not efficient. There are no exact indicators which show the condition of the system and sometime, the car parks are overloaded. These ineffective conditions happened because of the lack of implementation in technologies which are available in the market today.

The project requires the author to design an "***Intelligent Monitoring and Controlling for a Parking System (Car Park)***". In order to do so, the author is required to do some study and research on the existing parking system first. The author also need to observe the technologies used and then, develop a new concept of implementation and controlling for the system.

1.2 Problem Statement and Identification

Nowadays, lack of implementation in technologies available, in designing the existing car park has given a big impact to the system which is currently been used. Looking to the inefficient and ineffective condition, it has lead the engineers to come out with a new controller and system in order to give better output, especially in the aspect of managing and maintaining the parking lot designed.

In this project, the author will create a new system which can be used to overcome the problems as mentioned before. The system is managed systematically by using C code that stored in the main controller, which is PIC Microprocessor. The controller itself is very important in the design, because it acts as a brain or *Central Processing Unit* (CPU) in a computer.

The system also will allow a known number of vehicles to enter the car park. Meaning that, the system can avoid from the overload situation that might happen in any time of operation. The author also puts an indicator in each of the parking slot to indicate either that particular slot is empty or not. This system helps the users to find the empty space quickly without need to waste their time looking for that slot.

The system also will have an indicator at the entrance of the car park to tell the users either the car park is already full or not. The indicator is very important because the users can know either the car park still have the empty space or not by only looking at the sign given and it is very useful during the peak hour.

The ability of the system to do up-counter and down-counter, which is to count the number of vehicles coming in or moving out, is very important in the design and need more attention on it. A part from that, the indicators, display board, auto-barrier system and sensors used are also significant to the project in order to perform all tasks as mentioned above.

1.3 Objective and Scope of Study

1.3.1 Objectives

1. To design an intelligent system that can be implemented in a car park.
2. To propose or provide new controller for parking lot control.
3. To design a complete circuit for the system.
4. To increase the safety factors in the design and planning.

1.3.2 Scope of Study / Work

There are several topics and issues that must be considered before proceeding any further in the design of the device. The scope of study depends mainly on these few areas:

1. Designing the sensor circuit.
2. Studying the behaviour of *Peripheral Interface Controller* (PIC) Microcontroller.
3. Programming the PIC Microcontroller with appropriate programming language.
4. Connection of the PIC Microcontroller and the devices involve.
5. Designing the circuit using the digital electronics fundamental knowledge.

The applications and devices that are considered must be reliable, cost-effective and practical for a feasible implementation. This will ensure that the design produced is economical and marketable.

The author is responsible to manage the project so that it is within the scope and time frame. Time management is important to provide equal attention on each task. On the other hand, it is essentially to ensure that the project is feasible and could be completed within the allocated time frame.

Please refer to **APPENDIX A** for the Gantt Chart.

CHAPTER 2

LITERATURE REVIEW

Up to this time, the author has figured out important information resources that will be helpful for this project as some of the parts in the project were changed. This information will act as the guide and reference throughout the project implementation. The author has conducted some researches from book sources as well as from internet. The researches done were on related particular subjects such as type sensors, motors, car park architecture, controller and its programming,

2.1 Weight In Motion Sensor

Figure 1: Sign of Weight in Motion Sensor

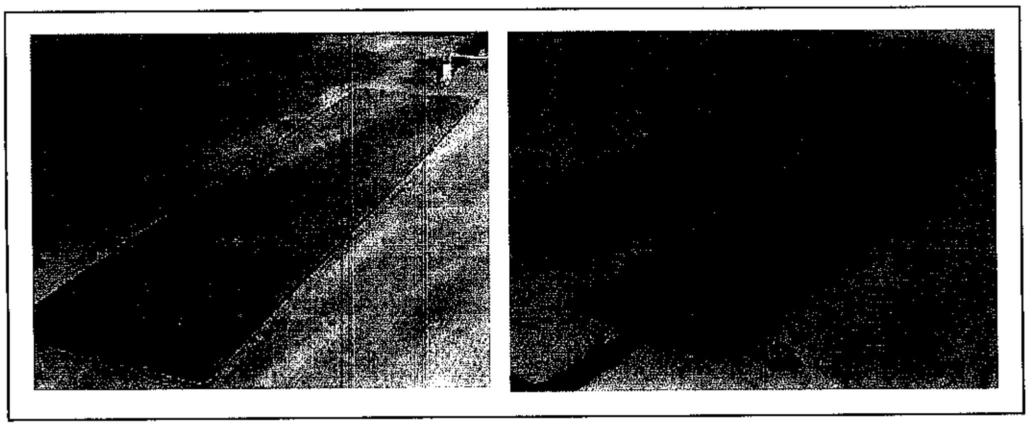


Figure 2: Before and After Installation of Weight in Motion Sensor

Weight in motion sensor is a device used to detect the presence of a vehicle which is moving in a certain place or location. This type of sensor is a relatively simple. A dynamic weighing system involves 3 parts. The first is an accurate weighing device such as weigh-pads or axle-pads. The output signal from these devices are then fed through the second component, a very high speed A/D converter to change the voltage signal into something that the computer can use. The third part is a software package on a computer or main controller that determines whether the weight has crossed over the weighing sensors or not. There will be a range of weight of load specified in the program software, to ensure that only vehicle's weight is detected. If the sensor detects a human weight, it will not send the input to the controller to be processed.

Figure 3: Real Application of Weight in Motion Sensor

All the vehicle drivers have to slow down at a speed between 5 and 10 km/h and drive over the weight sensors. At this time, the weight sensor will recognize the vehicles and send the input signal to the main controller. Some of the weight in motion sensors not only be able to sense existence vehicles but also can give total weight, individual axle weight and speed of the vehicle itself and these types of sensors usually are very expensive in the current market.

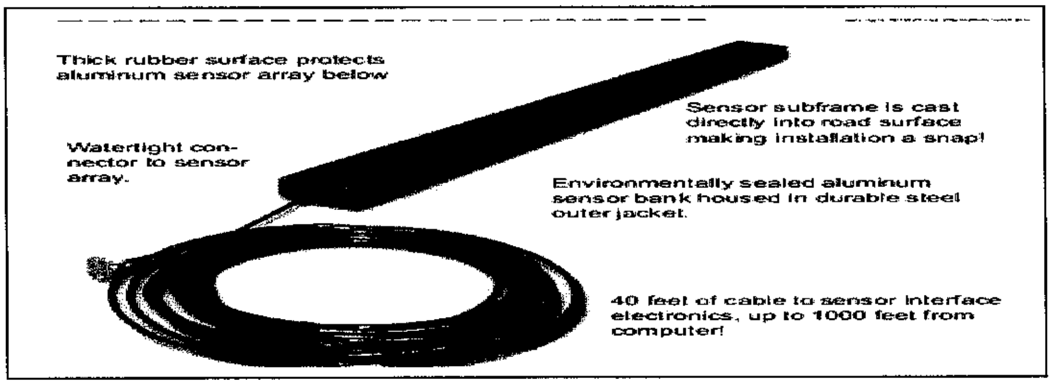

Thick rubber surface protects aluminum sensor array below
Sensor subframe is cast directly into road surface making installation a snap!
Watertight connector to sensor array.
Environmentally sealed aluminum sensor bank housed in durable steel outer jacket.
40 feet of cable to sensor interface electronics, up to 1000 feet from computer!

Figure 4: Main Component of Weight in Motion Sensor

2.2 Auto Barrier System (Servo Motor)

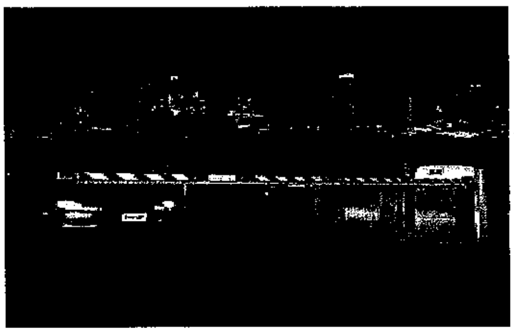


Figure 5: Auto Barrier System

Auto barrier system or auto gate system is one of the important parts in the design proposed. The main component of the system is a motor, which is a servo motor. A servo is a small device that has an output shaft. This shaft can be positioned to specific angular positions by sending the servo a coded signal. As long as the coded signal exists on the input line, the servo will maintain the angular position of the shaft. As the coded signal changes, the angular position of the shaft changes. In practice, servos are used in radio controlled airplanes to position control surfaces like the elevators and rudders. They are also used in radio controlled cars, puppets, and of course, robots.

Figure 6: A Futaba S-148 Servo

Servos are extremely useful in robotics and other applications which need a specific angle of task. The motors are small, as you can see by the picture above, have built in control circuitry, and are extremely powerful for their size. A standard servo such as the Futaba S-148 has 42 oz/inches of torque, which is pretty strong for its size. It also draws power proportional to the mechanical load. A lightly loaded servo, therefore, does not consume much energy. The guts of a servo motor are shown in the picture below. They are the control circuitry, the motor, a set of gears, the case and also the 3 wires that connect to the outside world. One is for power which is +5 volts, ground, and the white wire is the control wire.

Figure 7: A Servo Disassembled

The servo motor has some control circuits and a potentiometer (a variable resistor, know as pot) that is connected to the output shaft. In the picture above, the pot can be seen on the right side of the circuit board. This pot allows the control circuitry to monitor the current angle of the servo motor. If the shaft is at the correct angle, then the motor shuts off. If the circuit finds that the angle is not correct, it will turn the motor the correct direction until the angle is correct. The output shaft of the servo is capable of travelling somewhere around 180 degrees. Usually, it is somewhere in the 210 degree range, but it varies by manufacturer. A normal servo is used to control an angular motion of between 0 and 180 degrees. A normal servo is mechanically not capable of turning any farther due to a mechanical stop built on to the main output gear.

The amount of power applied to the motor is proportional to the distance it needs to travel. So, if the shaft needs to turn a large distance, the motor will run at full speed. If it needs to turn only a small amount, the motor will run at a slower speed. This is called proportional control

The control wire is used to communicate the angle. The angle is determined by the duration of a pulse that is applied to the control wire. This is called Pulse Coded Modulation. The servo expects to see a pulse every 20 milliseconds (0.02 seconds). The length of the pulse will determine how far the motor turns. A 1.5 millisecond pulse, for example, will make the motor turn to the 90 degree position (often called the neutral position). If the pulse is shorter than 1.5 ms, then the motor will turn the shaft to closer to 0 degrees. If the pulse is longer than 1.5ms, the shaft turns closer to 180 degrees.

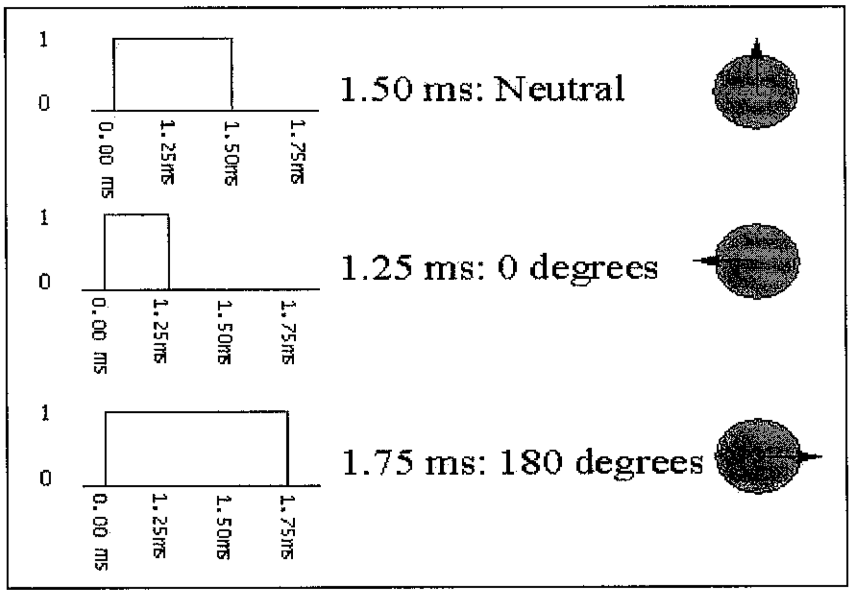


Figure 8: Duration of Pulse with Angle

As we can see in the *Figure 8*, the duration of the pulse dictates the angle of the output shaft (shown as the green circle with the arrow). Note that the times here are illustrative and the actual timings depend on the motor manufacturer. The principle, however, is the same.

2.3 Peripheral Interface Controller (PIC) Microcontroller

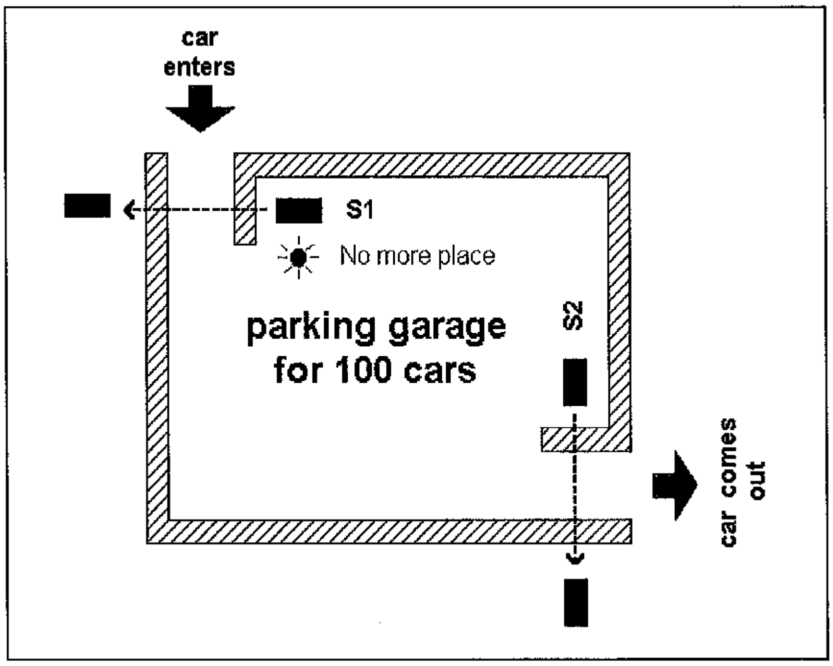


Figure 9: Overall Car Park System

For the project, the author has selected to use two types of PIC Microcontroller which are PIC16F877 and PIC16F84 as stated below:

2.3.1 PIC16F877

The chosen of microcontroller PIC16F877 belongs to the high-performance CPU, 40 pins which only 35 single word instructions to learn. This type of PIC also has full static design, low power, high speed CMOS Flash/EEPROM technology with interrupt capability up to 14 sources.

The interesting features in the PIC16F877 are:

1. The timer0: 8 bit timer/counter with 8-bit prescaler
2. Timer 1: 16 bit timer/counter with prescaler which can be incremented during SLEEP via external crystal/clock.

3. Timer 2: 8 bit timer/counter with 8 bit period.
4. 10 bit multi channel Analogue to Digital converter.

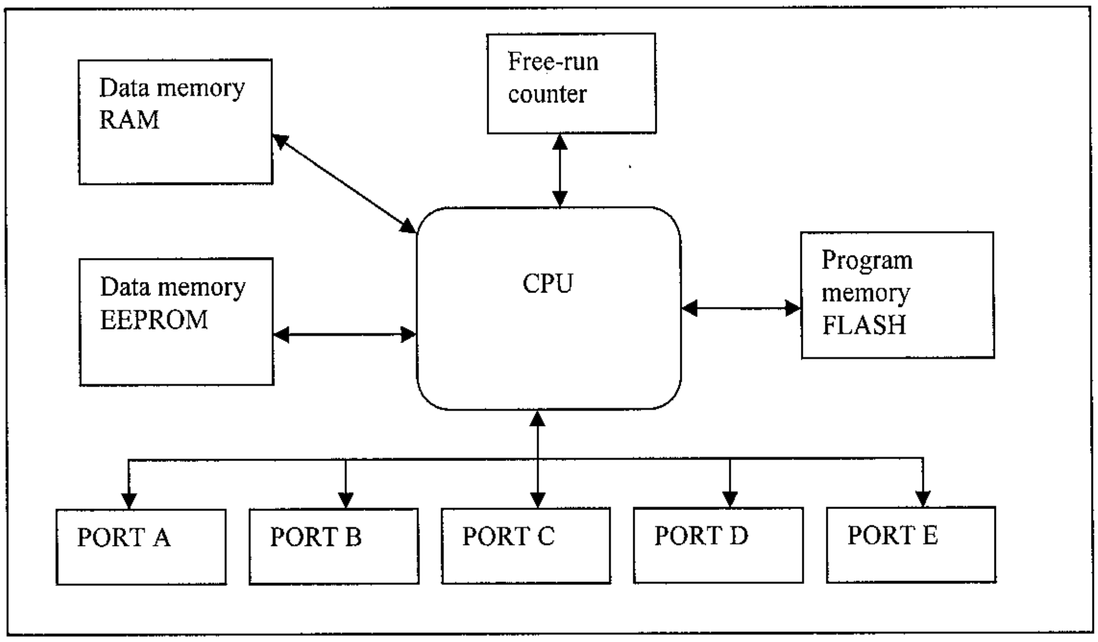


Figure 10: PIC16F877 Microcontroller Outline

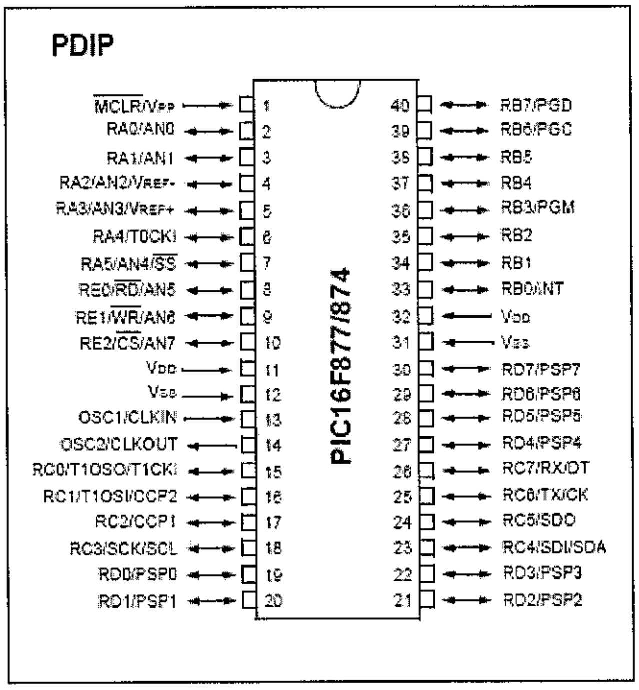


Figure 11: PIC16F877 Pin Out

Pin	Description
1	Reset input
2 – 7	PORT A pins
8 – 10	PORT E pins
11, 32	Positive power supply pole.
12, 31	Ground of power supply.
13 - 14	Pin assigned for connecting with an oscillator
15 – 18, 23 – 26	PORT C pins
19 – 22, 27 – 30	PORT D pins
33 – 40	PORT B pins

Table 1: Pin description of PIC16F877

Please refer to **APPENDIX B** for PIC16F877 datasheet.

2.3.2 PIC16F84

The chosen of microcontroller PIC16F84 belongs to the family of low-cost, high-performance, CMOS, fully-static, 8-bit microcontrollers. PIC16F84 has enhanced core features, eight-level deep stack, and multiple internal and external interrupt sources. The separate instruction and data buses of the Harvard architecture allow a 14-bit wide instruction word with a separate 8-bit wide data bus. Additionally, a large register set is used to achieve a very high performance level.

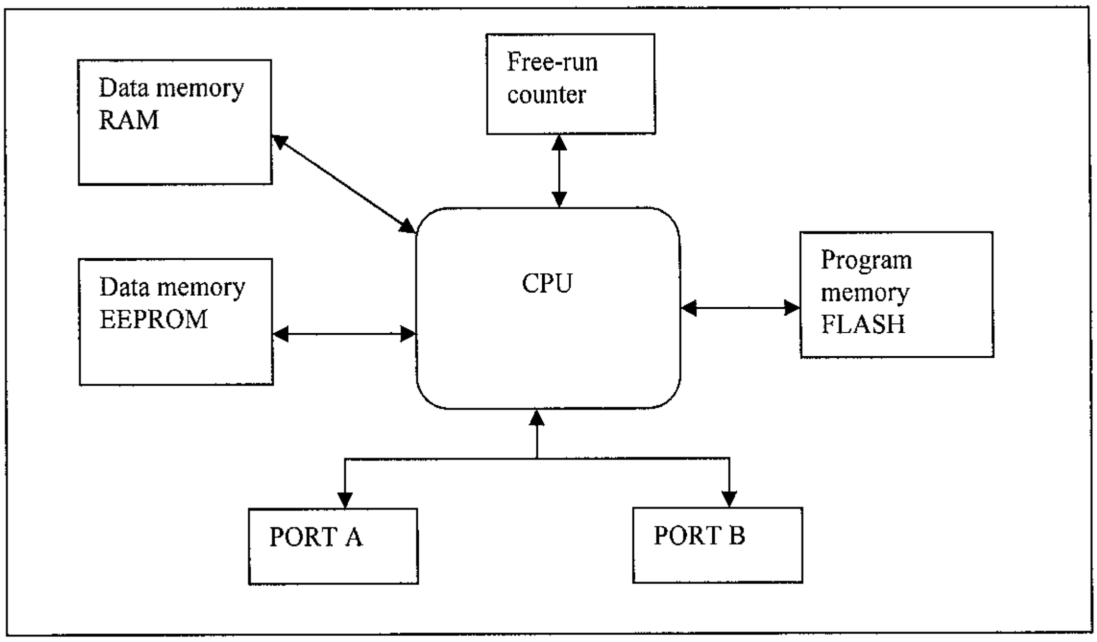


Figure 12: PIC16F84 Microcontroller Outline

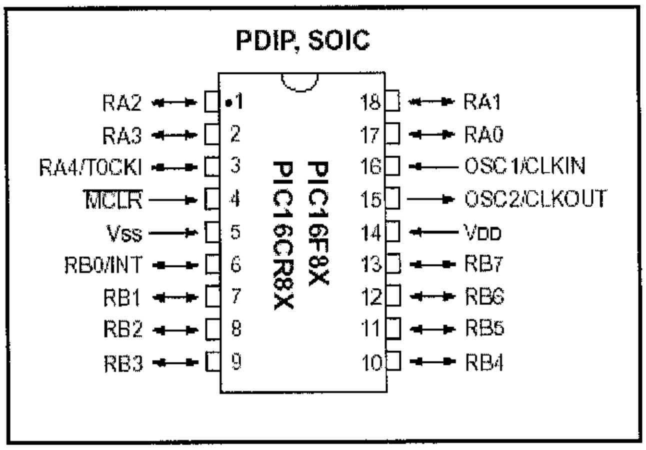


Figure 13: PIC16F84 pin out

Pin	Description
1 - 3, 17 - 18	PORT A pins
4	Reset input
5	Ground of power supply.
6 - 13	PORT B pins
14	Positive power supply pole.
15 - 16	Pin assigned for connecting with an oscillator

Table 2: Pin description of PIC16F84

PIC16F84 microcontrollers typically achieve a 2:1 code compression and up to a 4:1 speed improvement (at 20 MHz) over other 8-bit microcontrollers in their class. The PIC16F84 has up to 68 bytes of RAM, 64 bytes of Data EEPROM memory, and 13 I/O pins. A timer/counter is also available.

The interesting features in the PIC16F84 are:

1. The SLEEP (power-down) mode offers power saving. The user can wake the chip from sleep through several external and internal interrupts and resets.
2. A highly reliable Watchdog Timer with its own on-chip RC oscillator provides protection against software lockup.

The reason of choosing PIC Microprocessor as one of the controller was because of the ability to control the display board as required. The display is configured as an up-counter, down-counter, and can produce a number of messages using letters of the alphabet that can be readily displayed. All the timing signals for the display are produced by the program which is the advantage of a microcontroller driving the display is flexibility.

Please refer to **APPENDIX C** for PIC16F84 datasheet.

2.4 PIC Software

PIC is a Windows-based PC application that provides a platform for developing C language code for Microchip's PICmicro microcontroller (MCU) families. Generically, PIC will be programmed for microcontroller routine by using C language instead of assembly. Then, the coding will be compiled and converted into hex files that are used to download all the routines into the microcontroller. This process requires software to download the data into microcontroller, which is WARP13.

From this point onwards, this software is used by the author to construct the programming instead of using MPLAB software which uses assembly language. The main reason is that LCD display requires a big size of routines and it is very complicated and time consuming to be designed by using assembly language. Another reason is the delay time routine. To create a very accurate time delay in assembly language, it requires more time than by using C. Time delay is important in communication between the microcontroller and the display processor, and also the motor stepping. This is because the PIC will calculate the sequences needed in order to give more accurate of delay value.

To work with the software, it is preferable that the programmer has the basic knowledge working in C or C++ language. The datasheet of the selected microcontroller series is also needed as several important parameters can be referred to it. Knowledge in assembly language also helps the programmer to understand further of some programming practices done in processing the data.

2.5 WARP 13 board

The WARP-13 is a device programmer designed to program the microchip PIC family of microcontrollers. The WARP-13 is a combination of very refined firmware and software designed to provide high levels of utility function, programming speed, reliability and ease of use. This programmer requires an external power supply of 12VAC or 16VDC. It has a diode bridge on the input, so it accepts both AC and DC.

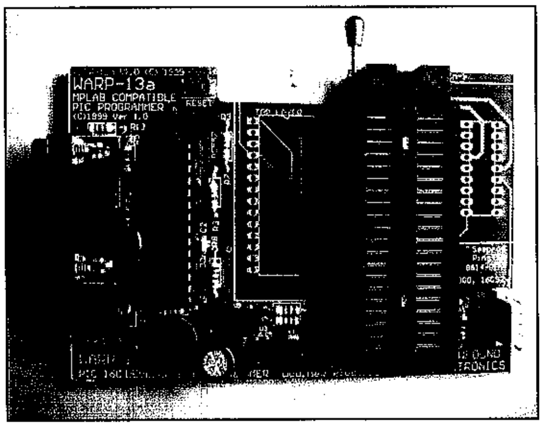


Figure 14: Warp13 Board

2.6 Liquid Crystal Display Module

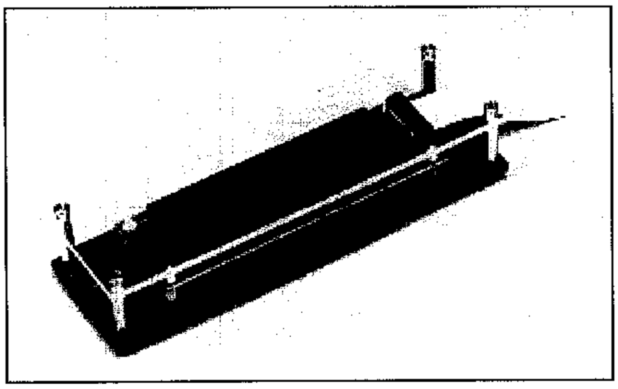


Figure 15: Liquid Crystal Display

Liquid Crystal Display Modules (LCD Module) are found in everything from digital watches to laptop. It is an optional accessory for the applications board. This LCD is invaluable for demonstrating the display of messages, warnings or instructions, as used on small instruments or on large public information boards such as those found at stations, airports and other places.

The interesting features about LCD are:

1. Wide variety of operating instructions such as display clear, cursor home, display On/Off, display cursor blink, cursor shift, display shift.
2. Internal automatic reset circuit upon power up.
3. Internal oscillator circuit.
4. CMOS circuitry.
5. Logic power sources which is single 5 voltages for normal temperature and dual voltage for extended temperature.
6. Operating temperature range which is 0 to +50 ^{o}c (standard type) and -20 to +70 ^{o}c ("H" type).

2.6.1 Dot Matrix LCD Architecture

Most LCD modules conform to a standard interface specification. A 14-pin access is provided having eight data lines, three control lines and three power lines. The connections are laid out in one of two common configurations, either two rows of seven pins, or a single row of 14 pins.

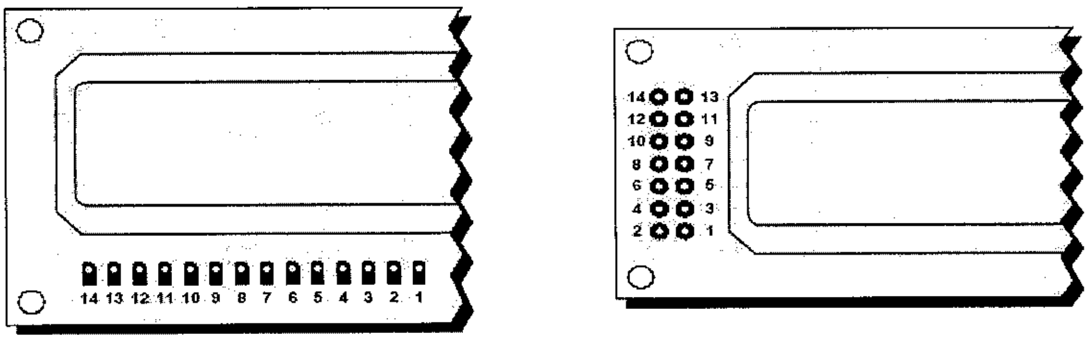


Figure 16: LCD Pinouts

Pin No.	Name	Function
1	Vss	Ground
2	Vdd	Positive supply
3	Vee	Contrast
4	RS	Register Select
5	R/W	Read / Write
6	E	Enable
7	D0	Data bit 0
8	D1	Data bit 1
9	D2	Data bit 2
10	D3	Data bit 3
11	D4	Data bit 4
12	D5	Data bit 5
13	D6	Data bit 6
14	D7	Data bit 7

Table 3: Pin Description of LCD

The function of each of the connections is shown as in *Table 3*. Although the LCD modules data sheets specify a 5V D.C supply, supplies around 4.5V to 5.5v works fine. Pin 4 is the *Register Select* (RS) line, the first of the three command control inputs. When this line is low, data bytes transferred to the display are treated as commands, and data bytes read from the display indicate its status. By setting the RS line high, character data can be transferred to and from the module.

Pin 5 is the *Read/Write* (R/W) line. This line is pulled low in order to write commands or character data to the module, or pulled high to read character data or status information from its registers.

Pin 6 is the *Enable* (E) line. This input is used to initiate the actual transfer of commands or character data between the module and the data lines. When writing to the display, data is transferred only on the high to low transition of this signal. However, when reading from the display, data will become available shortly after the low to high transition and remain available until the signal falls low again.

Pin 7 to 14 are the eight data bus lines (D0 to D7). Data can be transferred to and from the display, either as a single 8-bit byte or as two 4-bit "nibbles". In the latter case, only the upper four data lines (D4 to D7) are used.

Please refer to **APPENDIX D** for Standard LCD Character Table

CHAPTER 3

METHODOLOGY / PROJECT WORK

3.1 Procedure Identification

Basically, there are 4 steps of systematic approach for the project which is represented by general block diagram in *Figure 17* below. The description of each step is discussed as follow:

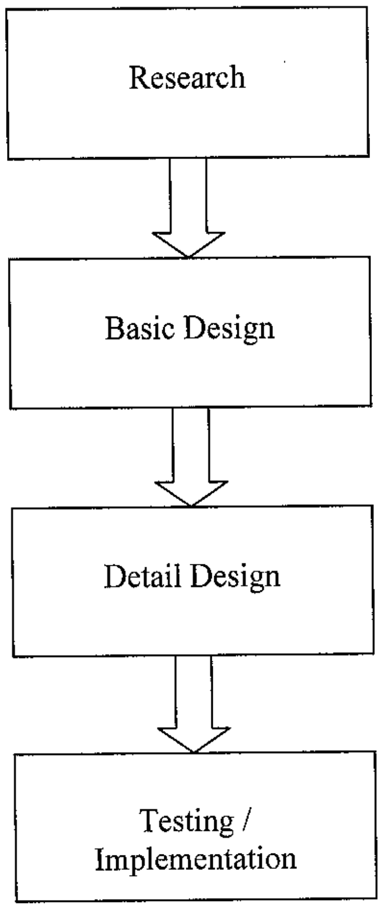


Figure 17: Overall Block Diagram

3.1.1 Research

Research is very useful especially in conducting a project. Research is done in order to get data and information as a reference and guidance. This research is conducted via book source as well as internet.

In ***Final Year Project I***, the author has conducted the research based on the previous assessment in this field. Some information also has been taken from the internet to guide the author on the project given. After narrowing the specification of the project, the author has come out to use only PIC Microcontroller as the main brain for the project, instead of combining two controllers, which are PIC Microcontroller and *Programmable Logic Controller* (PLC).

3.1.2 Basic Design

The materials and equipments are considered as the best alternative because during the basic design, a few alternatives of the project structure have been proposed. The structure of the project is the most crucial part in achieving the needs and goals set.

Basic design involved steps such as:

1. Review on the PIC Microcontroller and modelling techniques.

   In this part, the author had done several researches in order to gather all the relevant data regarding the scope defined. Basic understanding of the devices is the main priority in order to proceed with the next steps.

2. Specify the scope, design and technologies in the design propose.
   As mention in the problem statement, the author has listed down the criteria, scope and devices which are going to be used in the project proposed. (Please refer to *Chapter 1* for the scope and data)

3. Identify the concept of intelligent in the system.

   The ***intelligent*** concept, as stated in the project title means such below:

   - The system will be functioning 100% automatically, without need single person to manage it.
   - The system will tell the customer whether the parking is already full or not. If not, the system which has the indicators inside will show the location of the empty space in the parking lot.
   - The up-count and down-count function will ensure that overload condition will never happen in the Car Park.
   - Sensor is used to detect the incoming vehicles, which will cause the Auto Barrier System to be functioning.

4. Identify all types of circuits that will be utilized by using the Multism 2001 as to test the functionality of the circuits.

   There are several circuits which need to be understood and design. The author has collected several circuits as a reference based on the research done. Basically, there are 4 circuits all together. They are main controller circuit, servo motor circuit, 7 segments display circuit and liquid crystal display & light detecting resistor circuit.

3.1.3 Detail Design

The detail design is base on the best alternative selected during the basic design. The design will be more detail with the exact dimension of the project structure. The equipments and tools need to be used for the robot also are finalized during the detail design. As to get the good structure, the author will come out with the project simulation, programming development as well as circuit connection.

3.1.4 Testing / Implementation

The most crucial part in the methodology is testing and troubleshooting the project. All the complete connection and circuits need to be tested to identify any problem that might occur before the competition begin. All the problems need to troubleshoot during this period.

3.2 Tools Required

Hardware requirements:

- Sensor (which will be replaced with limit switch because of the cost factor).
- Servo Motor (Futaba S3003).
- Electronic circuit / Active components; resistors, capacitors and others.
- Light indicator (LED).
- Light Detecting Resistor (LDR).
- PIC16F877 and PIC16F84.
- Oscillator.
- 7 Segments Display.
- Liquid Crystal Display.
- Voltage Regulator 7805 and 7806.
- Materials use for prototype.
- Parking design (including spaces available and others).

Software requirements:

- PIC Microcontroller will be used as the programming to run the project.
- Multisim 2001 to simulate the circuits planned.

CHAPTER 4

RESULTS AND DISCUSSION

4.1 Circuits Connection

In this project, there are three specific circuits which are connected into the main controller circuit. They are servo motor circuit, 7 segments display circuit and liquid crystal display & light detecting resistor circuit. These circuits have their own function, which allow to accept or to send the input and output signal to the main controller. For 7 segments display circuit and servo motor circuit, they use different type of controller compare to the main controller circuit, which is PIC16F84. This PIC16F84 has 18 pins all together and is very suitable to be used as the second brain to the main controller, which is PIC16F877. This PIC16F877 has 40 pins and is use to control the overall operation of the system.

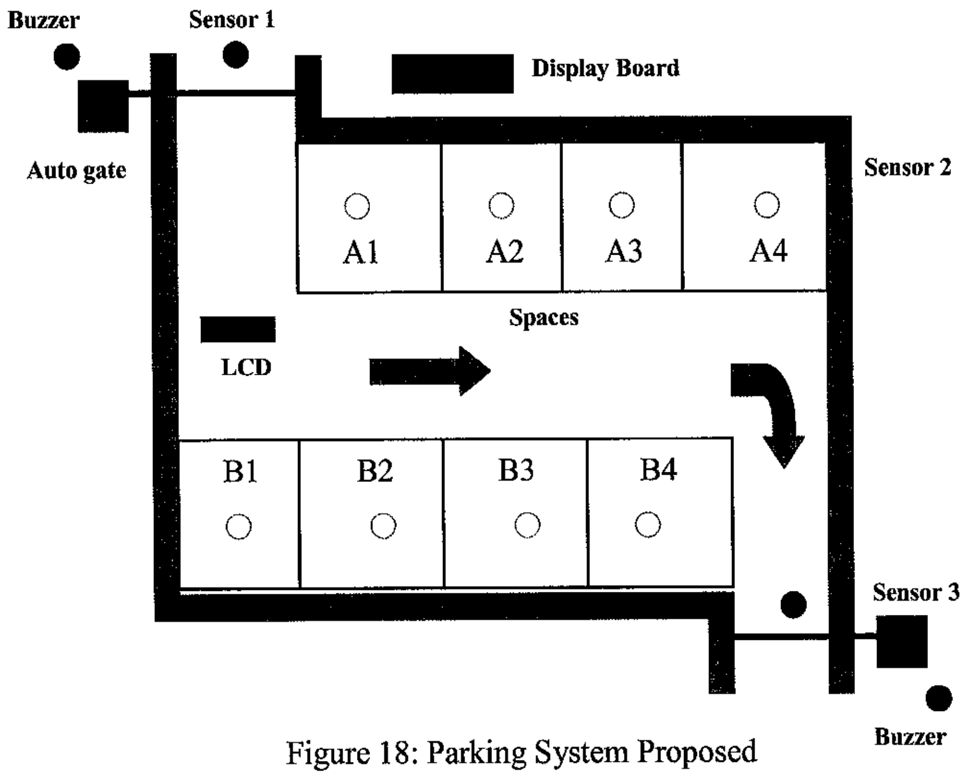


Figure 18: Parking System Proposed

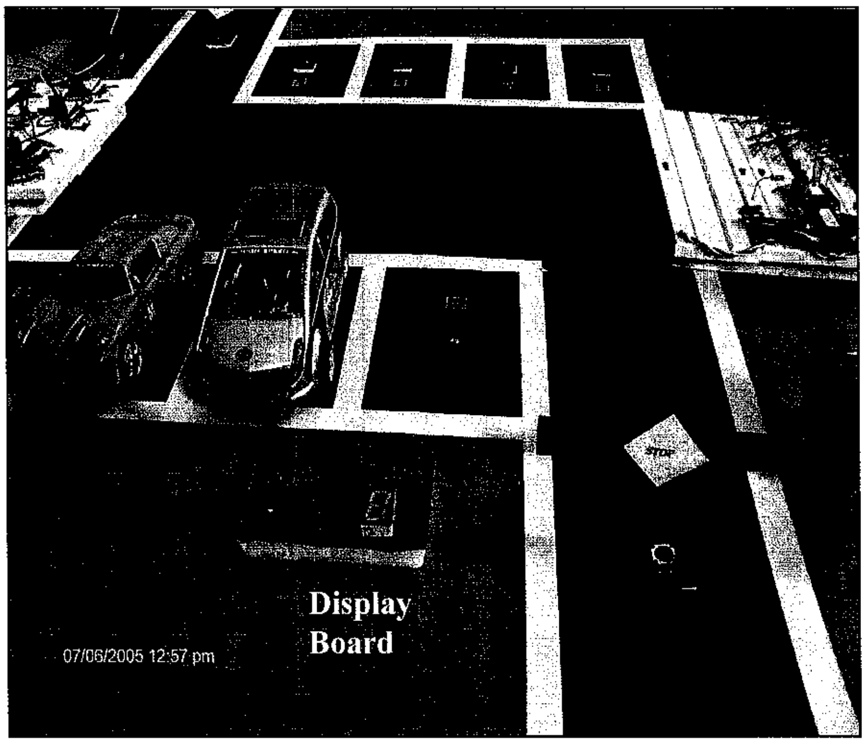


Figure 19: Overall Prototype

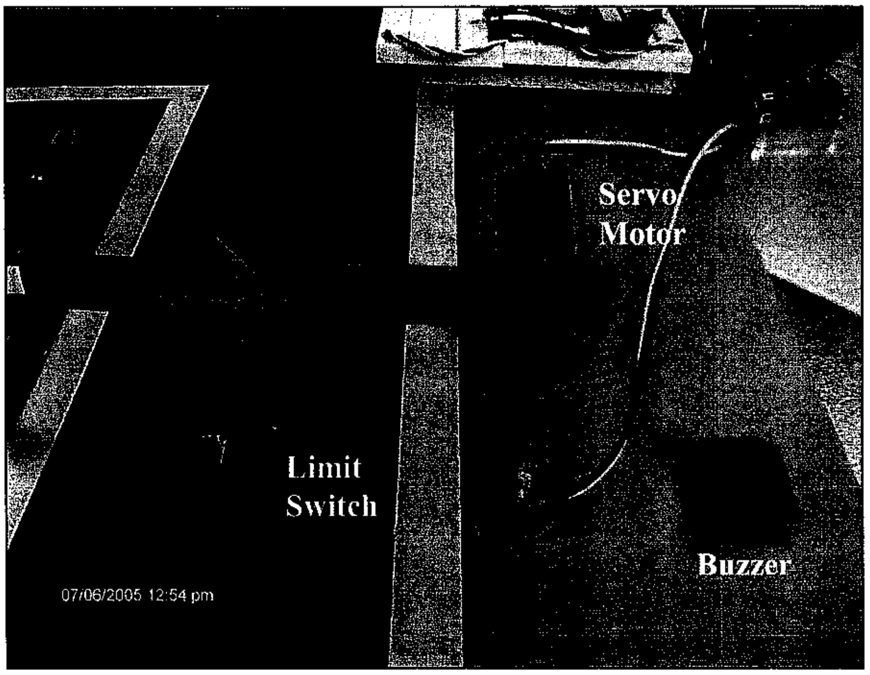


Figure 20: Arrangement of Limit Switch, Servo Motor and Buzzer

4.1.1 Main Controller Circuit

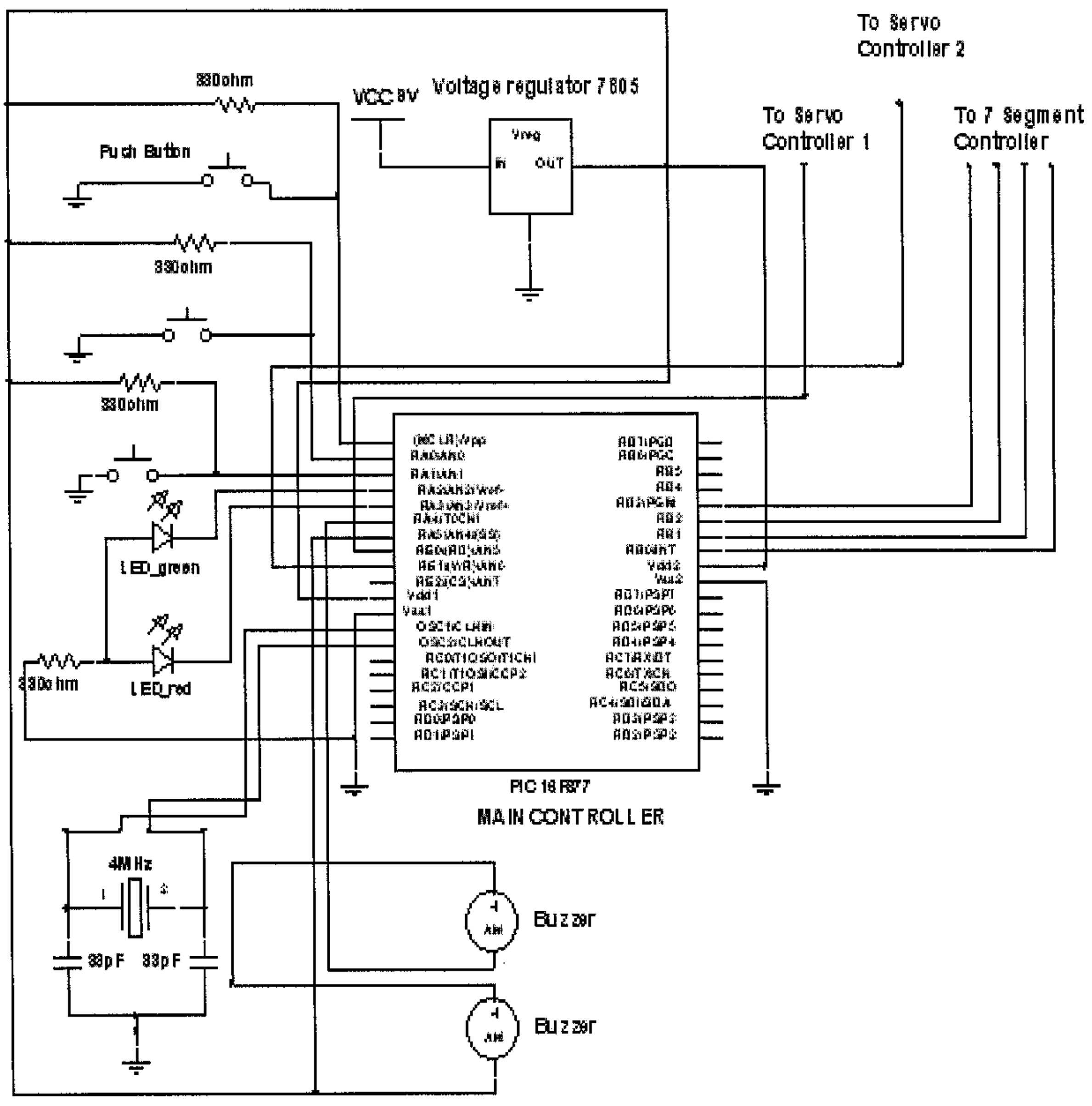


Figure 21: Main Controller Circuit

The main controller circuit which is using PIC16F877 as the main controller is the most important circuit for the project designed. It acts such as CPU or brain for the entire system proposed. Most of the data and signal will be sent and also will be received by this circuit to be analyzed first, before it gives an instruction for the appropriate action.

All the components involved in the circuit are shown as in the table below:

No	Electronic Components	Quantity
1	PIC16F877	1
2	Voltage Regulator 7805	1
3	Buzzer	2
4	Limit Switch	3
5	LED	2
6	Resistor 330 Ohm	4
7	Oscillator 4MHz	1
8	Capacitor 33pF	2

Table 4: Components of Main Controller Circuit

4.1.1.1 PIC16F877

It acts as the primary controller for the entire system. The PIC will be connected to the other circuits to enable the controller to communicate with the other part of the system. It will give a signal to the secondary controller when certain condition occurs. For example, if the primary controller detect by it sensor (limit switch) that there is a vehicle coming, it will send the signal to the servo motor controller to let the gate open.

4.1.1.2 Voltage Regulator 7805

It steps down the voltage Vcc from 9 V to 5 V. This is important to ensure that the components which are connected to the power supply will get the allowable voltage needed.

4.1.1.3 Buzzer

It is used to alert the users that the gate will be closed after the weight in motion sensor senses the back tyres of the vehicle.

4.1.1.4 Limit Switch

Limit switch is uses as a symbolic of the weight in motion sensor. This is due to the cost of weight in motion sensor which is higher and not suitable to be used in the small prototype created.

4.1.1.5 LED

Used to represent light indicator in the real system. There are two types of LED colour, the first one is GREEN which indicates "SPACE AVAILABLE" and the second one is RED which indicates "SPACE FULL / NOT AVAILABLE".

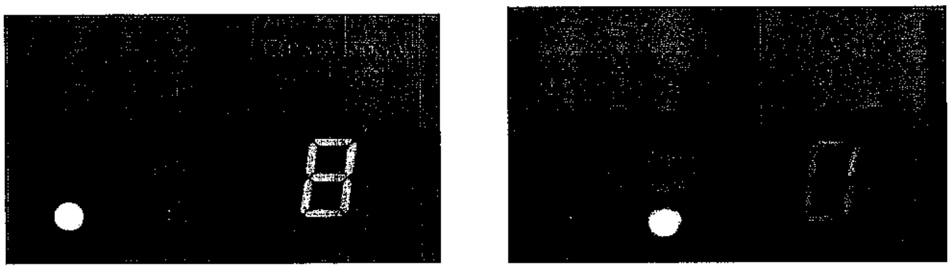


Figure 22: Space Indicator Based on the Remaining Spaces Available

4.1.1.6 Resistor

Used to ensure that the current flowing is not too high.

4.1.1.7 Oscillator

Used to generate pulse width modulation

4.1.1.8 Capacitor

Used to stabilize input signal and reduce noise.

Please refer to **APPENDIX E** for C coding of the Main Controller.

4.1.2 Servo Motor Circuit

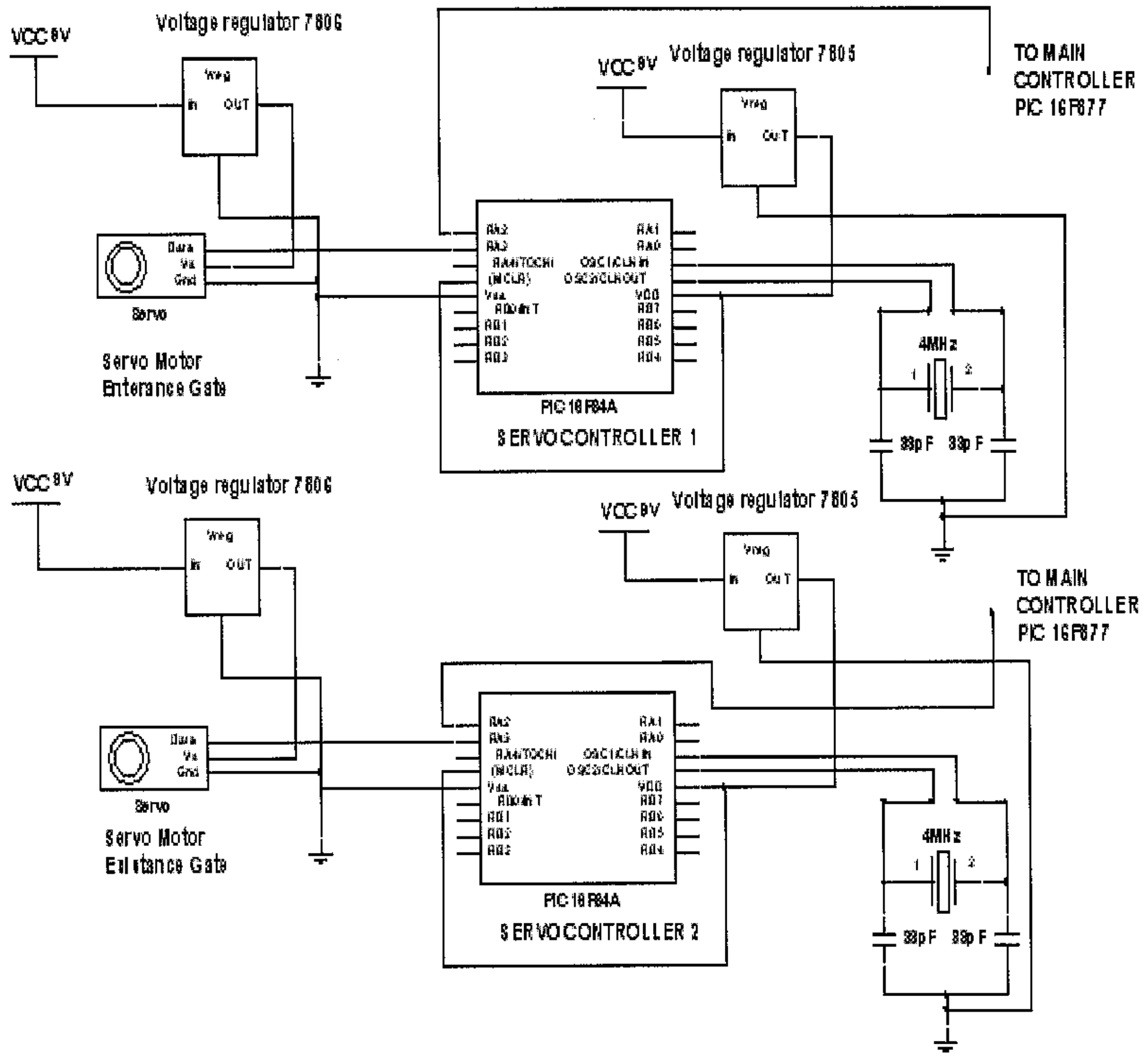


Figure 23: Servo Motor Circuit

The function of the circuit is to control the movement of the servo motor for auto barrier system. Each servo motor will be controlled by individual PIC16F84 in order to get better performance. There are two servo motors used which control the entry/exit of vehicles at the entrance gate and existence gate of the parking lot system. Each servo motor function when it receives a signal from the main controller. The signal is transmitted in the condition such as follow:

Signal from main controller	Movement of servo motor
0	Open
1	Close

Table 5: Movement of Servo Motor

All the components involved in the circuit are shown as in the table below:

No	Electronic Components	Quantity
1	PIC16F84	2
2	Voltage Regulator 7805	2
3	Voltage Regulator 7806	2
4	Servo Motor Futaba 3003	2
5	Oscillator 4MHz	2
6	Capacitor 33pF	4

Table 6: Components of Servo Motor Circuit

4.1.2.1 PIC16F84

The purpose of having PICs is to control the movement of the servo motors in the circuit. The two PICs will be connected to the main controller in order to get the input signal.

4.1.2.2 Voltage Regulator 7806

Step down the voltage Vcc from 9 V to 6 V. This is important to ensure that the components which are connected to the power supply will get the allowable voltage needed.

4.1.2.3 Servo Motor Futaba 3003

This motor is use to handle the movement of the gates of the parking lot. The signal will be received from PIC16F84 in the sense of Pulse Width Modulation. This pulse need to be supplied continuously in order get the fixed angular position.

4.1.2.4 Voltage Regulator 7805, Oscillator and Capacitors

**Please refer to Main Controller Circuit description.*

Please refer to **APPENDIX F** for C coding of the Servo Motor.

4.1.3 7 Segments Display Circuit

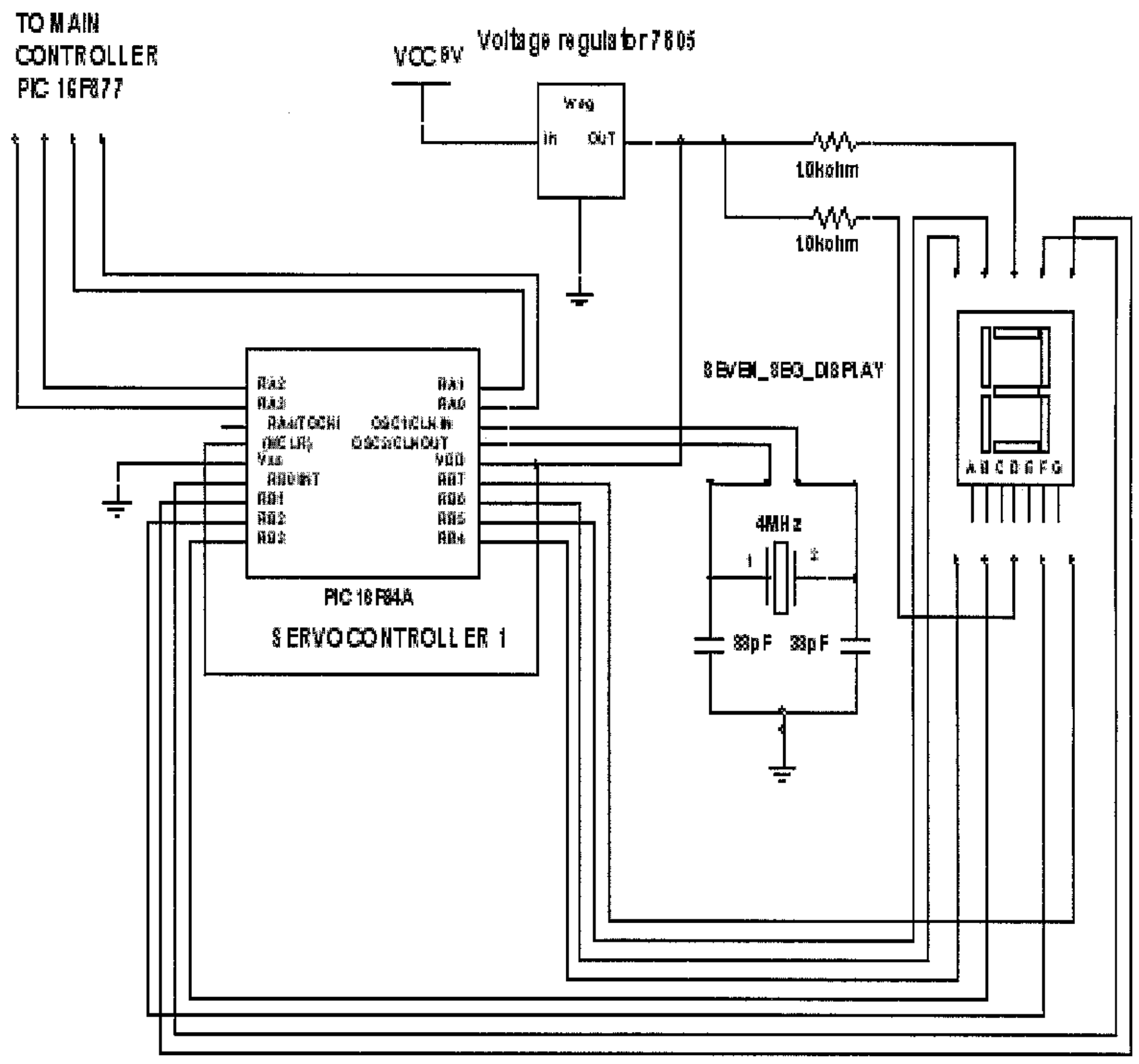


Figure 24: 7 Segments Display Circuit

The purpose of having this circuit is to provide the system with 7 segments display. As mentioned earlier, one of the features of the system is to have the up counter and down counter. This is to calculate the current number of the vehicles in the parking lot and to indicate whether the parking lot is already full or not. The 7 segments display shows the remaining number of spaces available based on the counting done.

When the spaces are full, the 7 segments display will show “0” number and RED colour LED will be ON. The input signal is transmitted in the condition as follow:

Signal from main controller	**Number Display By 7 Segment**
0000	8
0001	7
0010	6
0011	5
0100	4
0101	3
0110	2
0111	1
1000	0

Table 7: Numbers Display by 7 Segments

All the components involved in the circuit are shown as in the table below:

No	**Electronic Components**	**Quantity**
1	PIC16F84	1
2	Voltage Regulator 7805	1
3	7 Segments Display	1
4	Oscillator 4MHz	1
5	Capacitor 33pF	2
6	Resistor 1 kOhm	2

Table 8: Components of 7 Segments Display

4.1.3.1 PIC16F84

The purpose of having these PICs is to control the 7 segments display in the circuit. After receiving the input signal from the main controller, the PIC will activate its particular pin to come out with the number set in the programming. For example, to show number 4 at the 7 segments display, the main controller sends an input of "0100" (as show in *Table 7*) and the PIC will notice that the signal is for number 4. Then, the PIC will activate the appropriate pins which are connected to the particular blocks in the 7 segments ("f, b, g and d") to come out with number 4.

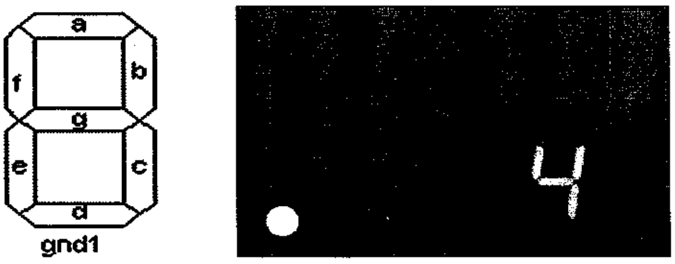


Figure 25: Blocks of 7 Segments Display Shows Number 4

4.1.3.2 7 Segments Display

Show the number based on the input supplied by the main controller.

4.1.3.3 Voltage Regulator 7805, Oscillator, Capacitor and Resistor

**Please refer to Main Controller Circuit description.*

Please refer to **APPENDIX G** for C coding of the7 Segments Display.

4.1.4 Liquid Crystal Display & Light Detecting Resistor Circuit.

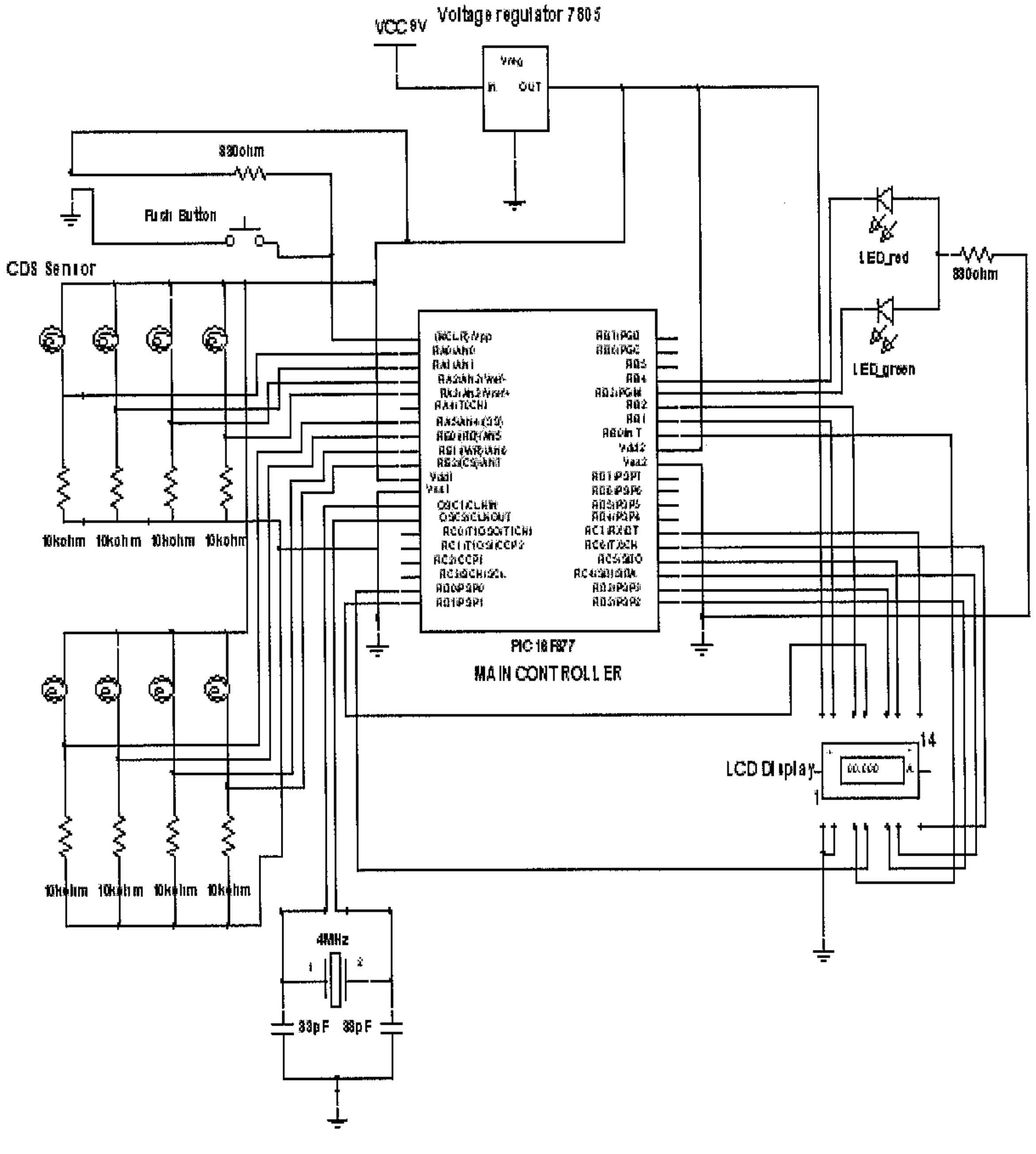


Figure 26: Liquid Crystal Display & Light Detecting Resistor Circuit.

This circuit is an indicator and sensor type of circuit which is used inside of the parking lot. In the author's prototype, there will be 8 spaces available and each of the space is provided with a *Light Detecting Resistor* (LDR) to detect any existing vehicle at that particular space. This is done in order to determine whether that space is available or already occupied. The signal triggered will be sent to the controller and will be shown in the LCD provided.

All the components involved in the circuit are shown as in the table below:

No	Electronic Components	Quantity
1	PIC16F877	1
2	LDR Sensor	8
3	LCD	1
4	Voltage Regulator 7805	1
5	LED	2
7	Capacitor	2
4	Limit Switch	1
6	Resistor 330 Ohm	2
7	Oscillator 4MHz	1
8	Capacitor 33pF	2

Table 9: Components of Liquid Crystal Display & Light Detecting Resistor Circuit

4.1.4.1 PIC16F877

It acts as the controller for the circuit. The main function of this PIC is to control the operation of the LDR and LCD in the design proposed. Any input gave by the LDR will be updated in LCD through this PIC.

4.1.4.2 Light Detecting Resistor (LDR)

It used to detect the incoming vehicles at the spaces available. This sensor is put at the basement of each space in the parking lot. When a vehicle get into the space, the area under that vehicle will be dimmed, which will activate the LDR sensor to give a signal to the controller. To use the LDR, here will be a threshold voltage which is used to set how dimness the area before the sensor is activated.

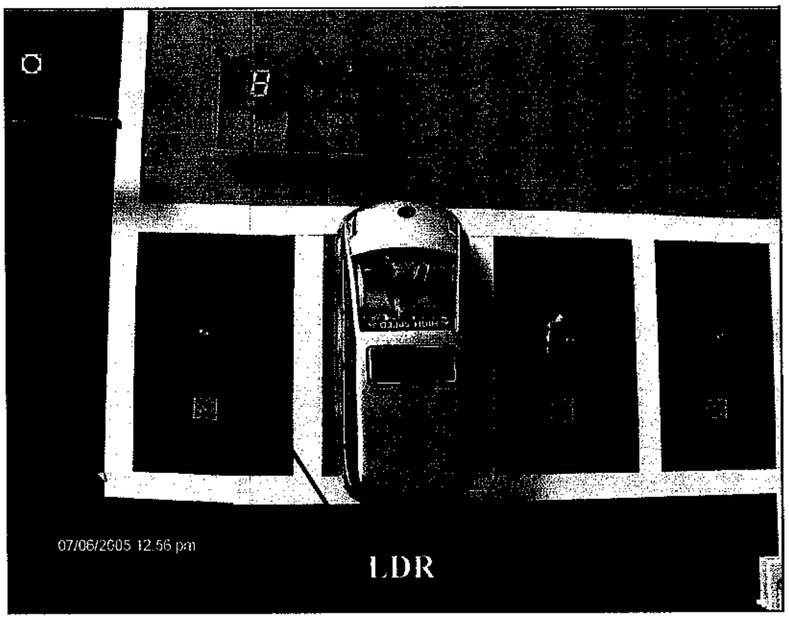


Figure 27: Location of LDR

4.1.4.3 Liquid Crystal Display (LCD)

It uses as the indicator for the system to show the empty spaces available in the parking lot. For example, if there is no vehicle in the parking system, the status of the spaces will be shown by the LCD as in *Figure 28* below:

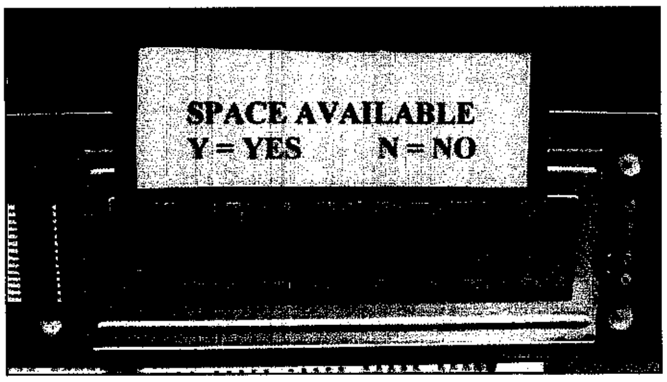


Figure 28: LCD Shows Current Space's Condition

If there is a vehicle, the status for that particular space occupied by the vehicle will be changed from " Y " to " N ".

4.1.4.4 Limit Switch

It used as a reset button for the PIC Controller.

4.1.4.5 LED, Voltage Regulator 7805, Oscillator, Capacitor and Resistor

**Please refer to Main Controller Circuit description.*

Please refer to **APPENDIX H** for C coding of LCD and LDR.

CHAPTER 5

CONCLUSION AND RECOMMENDATION

5.1 Conclusion

The first requirement of this project is to build an intelligent parking system which is 100 percent automatically handle by the controller. This has been achieved by using *Peripheral Interface Controller* (PIC) Microcontroller as the brain for the project designed. In addition, by having *Liquid Crystal Display* (LCD) Module, it helps the users to find the spaces available faster without wasting their time.

To make this project possible, the knowledge of programming in both C and assembly language is also required as to program the microcontroller. Enhancement in the programming can be done by reading materials and tutorials available on the microcontroller manufacturer websites. They also provide some tips and advices on how their product can be used in the applications.

Another requirement in this project is to make the system possible to count the total number of vehicles coming in and going out, for the parking lot design. The indicator helps the users to know the current number of vehicles in the car park and at the same time tells the users whether it is full or not.

It could be deemed that the project has been successful as the prototype designed is able to show all the features describes in the design objective.

5.2 Recommendation

Although the system is successfully workable, there are certain recommendations that need to be highlighted. They are discussed herein below:

1. To increase the safety factors in the parking lot, it is recommended to add a CCTV System. This system is used to detect and record is there is any crime happens in the area of the system.
2. The line/ wire connection of the system need to be checked frequently. So, as a recommendation, we can add a sensor such as power line filter to check the condition of the wiring and at the same time sends a report if there is any fault occurs.
3. The fabrications of the circuit also need to be considered since the system looks messy with the improper wiring on the circuit board. The circuits look messy with the wires and there was a short circuit problem when some of the circuits were not unconnected properly. The circuit can still be improved by using the PCB as to make the circuit design more systematic. The troubleshooting problem will be easier if the PCB were utilized for this project.

REFERENCES

Book:

1. J. Car, Joseph, ***Electronic Circuit Guidebook – Electro-optics***, Prompt Publication, 1997
2. Robert L. Boylestad, Louis Nashelsky, Electronic Devices and Circuit Theory, eight editions, Prentice Hall, 2002.
3. Mohan, Undeland, Robbins, Power Electronics (Converter, Applications and Design), John Willey & Sons, Inc, 2003.
4. Tocci Widmer, Digital Systems Principles and Application, eight editions, Prentice Hall, 2001

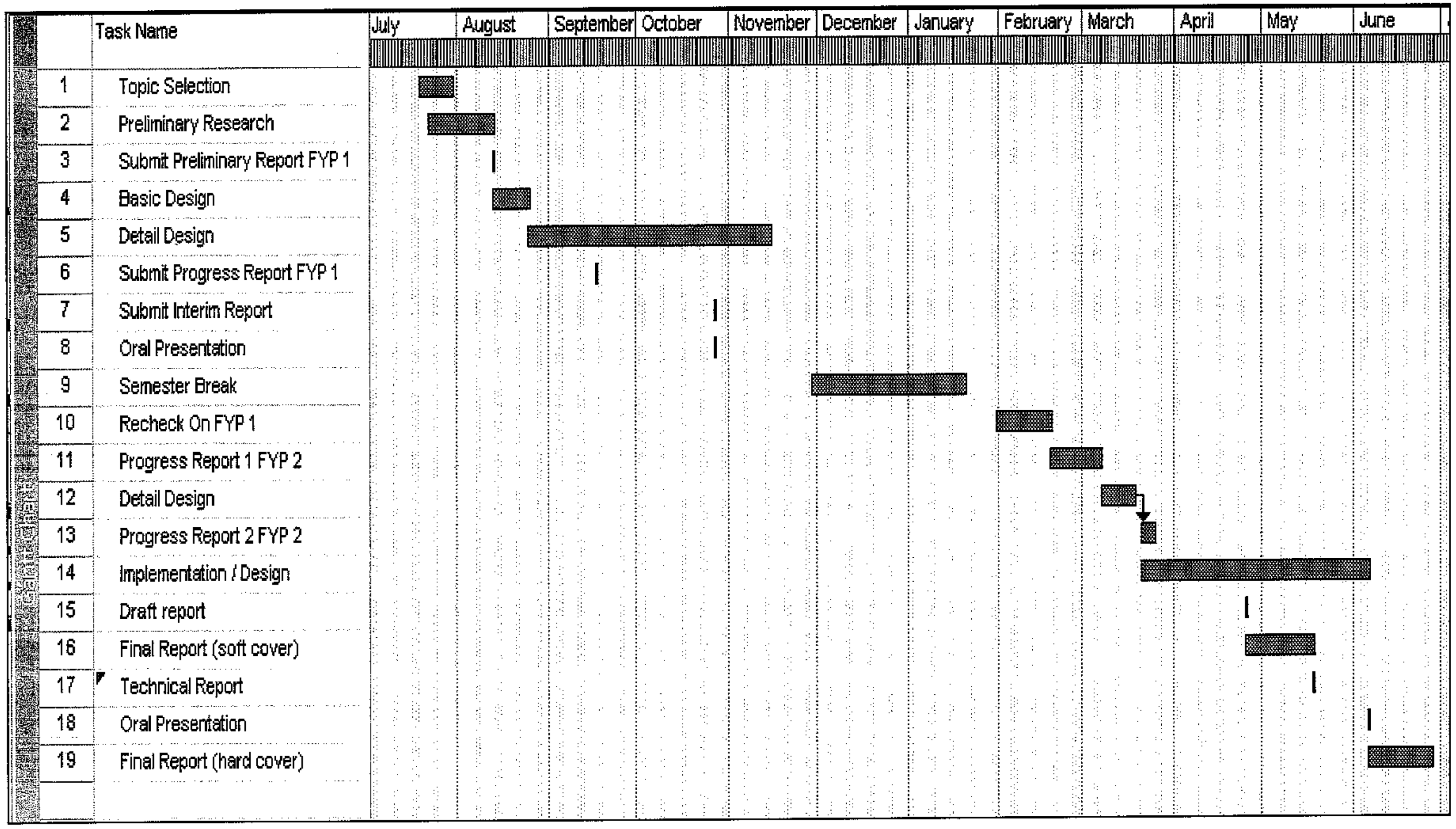
Website:

6. http://www.seattlerobotics.org/guide/servos.html
7 http://mechatronics.mech.northwestern.edu/design_ref/actuators/servo_motor intro. html
8. http://www.massload.com/weigh_in_motion_pg.html
9. http://www.ridders.com/news_88.html
10. http://www.fbk.com/electricity-electronics/24-202.asp
11. http://www.mikroelektronika.co.yu/english/product/books/PICbook/0_Uvod.htm

APPENDICES

APPENDIX A

Gantt Chart


Task Name
1 Topic Selection
2 Preliminary Research
3 Submit Preliminary Report FYP 1
4 Basic Design
5 Detail Design
6 Submit Progress Report FYP 1
7 Submit Interim Report
8 Oral Presentation
9 Semester Break
10 Recheck On FYP 1
11 Progress Report 1 FYP 2
12 Detail Design
13 Progress Report 2 FYP 2
14 Implementation / Design
15 Draft report
16 Final Report (soft cover)
17 Technical Report
18 Oral Presentation
19 Final Report (hard cover)
July
August
September
October
November
December
January
February
March
April
May
June

Gantt Chart

APPENDIX B

PIC16F877 Datasheet

PIC16F87X

28/40-pin 8-Bit CMOS FLASH Microcontrollers

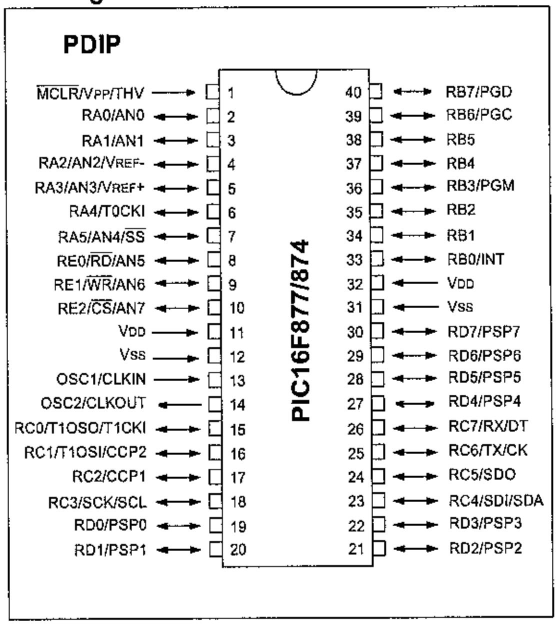
Devices Included in this Data Sheet:

- 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 and Industrial temperature ranges
- Low-power consumption:
  - < 2 mA typical @ 5V, 4 MHz
  - 20 μA typical @ 3V, 32 kHz
  - < 1 μA typical standby current

Pin Diagram



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 I^{2}C™ (Master/Slave)
- Universal Synchronous Asynchronous Receiver Transmitter (USART/SCI) with 9-bit address detection
- Parallel Slave Port (PSP) 8-bits wide, with external $\overline{RD}$, $\overline{WR}$ and $\overline{CS}$ controls (40/44-pin only)
- Brown-out detection circuitry for Brown-out Reset (BOR)

APPENDIX C

PIC16F84 Datasheet

PIC16F8X

18-pin Flash/EEPROM 8-Bit Microcontrollers

Devices Included in this Data Sheet:

- PIC16F83
- PIC16F84
- PIC16CR83
- PIC16CR84
- Extended voltage range devices available (PIC16**LF**8X, PIC16**LCR**8X)

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 - 10 MHz clock input
  DC - 400 ns instruction cycle

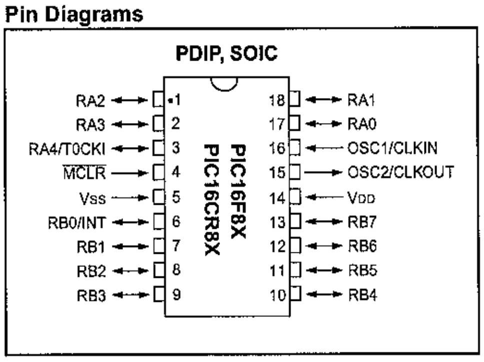
Device	Program Memory (words)	Data RAM (bytes)	Data EEPROM (bytes)	Max. Freq (MHz)
PIC16F83	512 Flash	36	64	10
PIC16F84	1 K Flash	68	64	10
PIC16CR83	512 ROM	36	64	10
PIC16CR84	1 K ROM	68	64	10

- 14-bit wide instructions
- 8-bit wide data path
- 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
- 1000 erase/write cycles Flash program memory
- 10,000,000 erase/write cycles EEPROM data memory
- EEPROM Data Retention > 40 years

Peripheral Features:

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

Pin Diagrams



Special Microcontroller Features:

- In-Circuit Serial Programming (ICSP™) - via two pins (ROM devices support only Data EEPROM programming)
- 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

CMOS Flash/EEPROM Technology:

- Low-power, high-speed technology
- Fully static design
- Wide operating voltage range:
  - Commercial: 2.0V to 6.0V
  - Industrial: 2.0V to 6.0V
- Low power consumption:
  - < 2 mA typical @ 5V, 4 MHz
  - 15 μA typical @ 2V, 32 kHz
  - < 1 μA typical standby current @ 2V

APPENDIX D

Standard LCD Character Table

High order bit / Low order bit	0000	0010	0011	0100	0101	0110	0111	1010	1011	1100	1101	1110	1111
X X X X0000	CG RAM (1)		0	@	P	`	p		ー	タ	ミ	α	p
X X X X0001	(2)	!	1	A	Q	a	q	。	ア	チ	ム	ä	q
X X X X0010	(3)	"	2	B	R	b	r	「	イ	ツ	メ	β	θ
X X X X0011	(4)	#	3	C	S	c	s	」	ウ	テ	モ	ε	∞
X X X X0100	(5)	$	4	D	T	d	t	、	エ	ト	ヤ	μ	Ω
X X X X0101	(6)	%	5	E	U	e	u	・	オ	ナ	ユ	σ	ü
X X X X0110	(7)	&	6	F	V	f	v	ヲ	カ	ニ	ヨ	ρ	Σ
X X X X0111	(8)	'	7	G	W	g	w	ァ	キ	ヌ	ラ	g	π
X X X X1000	(1)	(	8	H	X	h	x	ィ	ク	ネ	リ	√	x̄
X X X X1001	(2)	)	9	I	Y	i	y	ゥ	ケ	ノ	ル	⁻¹	y
X X X X1010	(3)	*	:	J	Z	j	z	ェ	コ	ハ	レ	j	千
X X X X1011	(4)	+	;	K	[	k	{	ォ	サ	ヒ	ロ	ˣ	万
X X X X1100	(5)	,	<	L	¥	l	|	ャ	シ	フ	ワ	¢	円
	(6)	-	=	M	]	m	}	ュ	ス	ヘ	ン	£	÷
	(7)	.	>	N	^	n	→	ョ	セ	ホ	゛	ñ	
	(8)	/	?	O	_	o	←	ッ	ソ	マ	゜	ö	█

Standard LCD Character Table

APPENDIX E

Main Controller C-Coding

```
/*
        Programmer: Azizan Hashim/Nik Othman Mohamed
  Program:              Intelligent Monitoring and Controlling for a Parking System
  Date:                 2 June 2005
*/

/*
        PIN A5= Buzzer Out                                                   (O)
        PIN A4= Buzzer In                                                    (O)
        PIN A3= LED Full Car Entered                           (O)
        PIN A2= LED Less Car Entered                           (O)
        PIN A1= Gate Sensor Out                                      (I)
        PIN A0= Gate Sensor In                                       (I)

        PIN B7=
        PIN B6=
        PIN B5=
        PIN B4=
        PIN B3= 7 Segment Bit 3                                      (O)
        PIN B2= 7 Segment Bit 2                                      (O)
        PIN B1= 7 Segment Bit 1                                      (O)
        PIN B0= 7 Segment Bit 0                                      (O)

        PIN C7=
        PIN C6=
        PIN C5=
        PIN C4=
        PIN C3=
        PIN C2=
        PIN C1=
        PIN C0=

        PIN D7=
        PIN D6=
        PIN D5=
        PIN D4=
        PIN D3=
        PIN D2=
        PIN D1=
        PIN D0=

        PIN E2=
        PIN E1= Servo Out (2) Data                             (O)
        PIN E0= Servo In (1) Data                              (O)
*/

#include <16F877.H>
#use    delay(clock=4000000)
#fuses  XT,NOPROTECT,NOWDT,NOLVP

#byte     port_b=0x06

void BUZZER_BEEP(int buzzer_num, int maxcount);

void main()
{
        int free_space;

        int previous_sensor_in;
        int previous_sensor_out;

        int current_sensor_in;
        int current_sensor_out;

        int counter_sensor_in;
        int counter_sensor_out;

set_tris_b(0x00);

//Buzzer Initialization
output_bit(PIN_A4,1);       //Buzzer in
output_bit(PIN_A5,1);       //Buzzer out

//Indicator Initialization
output_bit(PIN_A2,1);       //LED Less Car Entered
output_bit(PIN_A3,0);       //LED Full Car Entered

//Gate Initialization
output_bit(PIN_E0,1);       //Gate in closed
output_bit(PIN_E1,1);       //Gate out closed

//7-Segment Initialization
port_b=0x08;
free_space=8;

        previous_sensor_in=1;
        previous_sensor_out=1;
        counter_sensor_in=0;
        counter_sensor_out=0;


while(true)
{
                //Check in sensor
                current_sensor_in=input(PIN_A0);

  //If there is trigger
  if (current_sensor_in==0 && previous_sensor_in==1)
  {
                          previous_sensor_in=0;

    //Increase counter
    counter_sensor_in=counter_sensor_in+1;

    //If first trigger (car gets in)
    if (counter_sensor_in==1)
    {
                            output_bit(PIN_E0,0);       //Gate in opened
    }
    else
    //If second trigger (car pass by)
    if (counter_sensor_in==2)
    {

                                  BUZZER_BEEP(0,3);
                            output_bit(PIN_E0,1);       //Gate in closed

      //Minus space
      if (free_space>0)
      {
        free_space=free_space-1;
      }

      //Update 7-segment
```

```
        port_b=free_space;

                                        //Reset
                                        counter_sensor_in=0;
      }


    }
    else
                    if (current_sensor_in==1)
    {
                                previous_sensor_in=1;
    }

          }
}

void BUZZER_BEEP(int buzzer_num, int maxcount)
{
  int counter;

          if (buzzer_num==0)
  {

                    for (counter=1;counter<=maxcount;counter++)
          {
                              output_bit(PIN_A4,0);
              delay_ms(200);
                              output_bit(PIN_A4,1);
          delay_ms(200);
            }

  }
  else
          if (buzzer_num==1)
  {

                    for (counter=1;counter<=maxcount;counter++)
          {
                              output_bit(PIN_A5,0);
              delay_ms(200);
                              output_bit(PIN_A5,1);
          delay_ms(200);
            }

  }

}
```

APPENDIX F

Servo Motor C-Coding

```
/*
           Programmer: Azizan Hashim/Nik Othman Mohamed
  Date:                        2 June 2005
*/

#include <16F84A.H>
#use              delay(clock=4000000)
#fuses  XT,NOPROTECT,NOWDT

#define  delay_m902250      // - (-90)
#define  delay_m451800      // \ (-45)
#define  delay_0            1350 //  | (0)
#define  delay_p45 810      // / (+45)
#define  delay_p90 450  //  - (+90)

#define  delay_low  20

void main()
{
  while(true)
  {
   // 0
   if (input(PIN_A2)==0)
   {
                output_high(PIN_A3);
         delay_us(delay_0);

                output_low(PIN_A3);
         delay_ms(delay_low);
   }
                else
   // m90
   if (input(PIN_A2)==1)
   {
                output_high(PIN_A3);
         delay_us(delay_m90);

                output_low(PIN_A3);
         delay_ms(delay_low);
   }
  }
}
```

APPENDIX G

7 Segments C-Coding

```
#include <16F84A.h>
#use    delay(clock=4000000)
#fuses  XT,NOPROTECT,NOWDT

#byte port_a=0x05
#byte port_b=0x06

int display_array[10]={0xC0,0xF9,0xA4,0xB0,0x99,0x92,0x82,0xF8,0x80,0x90};

void main()
{
  int input_byte;

  set_tris_a(0xFF);
  set_tris_b(0x00);
          port_b=0xFF;

  while(true)
  {
          input_byte=port_a&0x0F;
    port_b=display_array[input_byte];
  }
}
```

APPENDIX H

LCD & LDR C-Coding

```
/*
        Programmer: Azizan Hashim/Nik Othman Mohamed
  Program:                Intelligent Monitoring and Controlling for a Parking System
  Date:                   2 June 2005
*/

/*
        PIN A5=
        PIN A4=
        PIN A3=
        PIN A2=
        PIN A1=
        PIN A0=

        PIN B7=
        PIN B6=
        PIN B5=
        PIN B4=
        PIN B3=
        PIN B2= LCD E Bit
        (O)
        PIN B1= LCD RS Bit                                              (O)
        PIN B0= LCD R/W Bit                                             (O)

        PIN C7=
        PIN C6=
        PIN C5=
        PIN C4=
        PIN C3=
        PIN C2=
        PIN C1=
        PIN C0=

        PIN D7= LCD DB7 Bit                                             (O)
        PIN D6= LCD DB6 Bit                                             (O)
        PIN D5= LCD DB5 Bit                                             (O)
        PIN D4= LCD DB4 Bit                                             (O)
        PIN D3= LCD DB3 Bit                                             (O)
        PIN D2= LCD DB2 Bit                                             (O)
        PIN D1= LCD DB1 Bit                                             (O)
        PIN D0= LCD DB0 Bit                                             (O)

        PIN E2=
        PIN E1=
        PIN E0=
*/

#include <16F877.H>
#device *=16 ADC=10 //PCWH
#fuses   XT,NOPROTECT,NOWDT,NOLVP
#use             delay(clock=4000000)

#byte    port_d=0x08

void LCD_write(int data_RS, int data_DB);     //Function to write into the LCD processor
int LCD_read(int data_RS);                                      //Function to read from the
LCD processor
void LCD_busycheck(void);                                       //Function to check the LCD
processor busy or not
void LCD_display(int linenum);                            //Function to display to LCD using data in
LCD_array[]
```

```
void display_check(void);                                                    //Just display all 'full square'
character (char=0xFF)
void display_intro(void);                                                    //Just display personalized intro

int LCD_array[20];                  //This is temporary array to write LCD line by line

void main()
{
        //For ADC
        long result_adc;

  int result_char_d;
  int result_char_c;
  int result_char_b;
  int result_char_a;

  //Port initialization
        set_tris_d(0x00);

  //***LCD INITIALIZATION***
  //------------------------
        port_d=0x00;
        output_bit(PIN_B2,0);
        output_bit(PIN_B1,0);
        output_bit(PIN_B0,0);

  //Wait for the busy flag to be HIGH
  delay_ms(20);

  //Re-initialization of function set
        LCD_busycheck();
        LCD_write(0,0x38);

  //Display OFF
        LCD_busycheck();
        LCD_write(0,0x08);

  //Clear display
        LCD_busycheck();
        LCD_write(0,0x01);

  //Entry mode set
        LCD_busycheck();
        LCD_write(0,0x06);
  //------------------------

  //***ADC INITIALIZATION***
  //------------------------
        //Setup the ADC
        setup_adc_ports(NO_ANALOGS);
        setup_adc(ADC_CLOCK_DIV_32);
  //------------------------

  display_check();
  display_intro();

  while(true)
  {
          //Display OFF
                LCD_busycheck();
                LCD_write(0,0x08);
```

```
    //*****ADC AQCUISITION*****
                    //------------------------
                    //Change channel 1
            set_adc_channel(0);

    //Delay for channel changes
    delay_us(20);

                    result_adc=read_adc();

                    result_char_d=result_adc/1000;
                    result_char_c=(result_adc-((long)result_char_d*1000))/100;
                    result_char_b=(result_adc-((long)result_char_d*1000)-((long)result_char_c*100))/10;
                    result_char_a=(result_adc-((long)result_char_d*1000)-((long)result_char_c*100)-
((long)result_char_b*10));

              LCD_array[0]=result_char_d+48;
            LCD_array[1]=result_char_c+48;
              LCD_array[2]=result_char_b+48;
              LCD_array[3]=result_char_a+48;
              LCD_array[4]=' ';

                    //Change channel 2
            set_adc_channel(1);

    //Delay for channel changes
    delay_us(20);

                    result_adc=read_adc();

                    result_char_d=result_adc/1000;
                    result_char_c=(result_adc-((long)result_char_d*1000))/100;
                    result_char_b=(result_adc-((long)result_char_d*1000)-((long)result_char_c*100))/10;
                    result_char_a=(result_adc-((long)result_char_d*1000)-((long)result_char_c*100)-
((long)result_char_b*10));

            LCD_array[5]=result_char_d+48;
              LCD_array[6]=result_char_c+48;
            LCD_array[7]=result_char_b+48;
              LCD_array[8]=result_char_a+48;
            LCD_array[9]=' ';

                    //Change channel 3
            set_adc_channel(2);

    //Delay for channel changes
    delay_us(20);

                    result_adc=read_adc();

                    result_char_d=result_adc/1000;
                    result_char_c=(result_adc-((long)result_char_d*1000))/100;
                    result_char_b=(result_adc-((long)result_char_d*1000)-((long)result_char_c*100))/10;
                    result_char_a=(result_adc-((long)result_char_d*1000)-((long)result_char_c*100)-
((long)result_char_b*10));

              LCD_array[10]=result_char_d+48;
            LCD_array[11]=result_char_c+48;
              LCD_array[12]=result_char_b+48;
            LCD_array[13]=result_char_a+48;
              LCD_array[14]=' ';
```

```
                    //Change channel 4
          set_adc_channel(3);

      //Delay for channel changes
      delay_us(20);

                    result_adc=read_adc();

                    result_char_d=result_adc/1000;
                    result_char_c=(result_adc-((long)result_char_d*1000))/100;
                    result_char_b=(result_adc-((long)result_char_d*1000)-((long)result_char_c*100))/10;
                    result_char_a=(result_adc-((long)result_char_d*1000)-((long)result_char_c*100)-
((long)result_char_b*10));

          LCD_array[15]=result_char_d+48;
            LCD_array[16]=result_char_c+48;
          LCD_array[17]=result_char_b+48;
            LCD_array[18]=result_char_a+48;
          LCD_array[19]=' ';

                    //-------------------------


                    //Write to line 1
          LCD_display(0);



      //*****ADC AQCUISITION*****
                    //-------------------------
                    //Change channel 5
          set_adc_channel(4);

      //Delay for channel changes
      delay_us(20);

                    result_adc=read_adc();

                    result_char_d=result_adc/1000;
                    result_char_c=(result_adc-((long)result_char_d*1000))/100;
                    result_char_b=(result_adc-((long)result_char_d*1000)-((long)result_char_c*100))/10;
                    result_char_a=(result_adc-((long)result_char_d*1000)-((long)result_char_c*100)-
((long)result_char_b*10));

            LCD_array[0]=result_char_d+48;
          LCD_array[1]=result_char_c+48;
            LCD_array[2]=result_char_b+48;
            LCD_array[3]=result_char_a+48;
            LCD_array[4]=' ';

                    //Change channel 6
          set_adc_channel(5);

      //Delay for channel changes
      delay_us(20);

                    result_adc=read_adc();

                    result_char_d=result_adc/1000;
                    result_char_c=(result_adc-((long)result_char_d*1000))/100;
                    result_char_b=(result_adc-((long)result_char_d*1000)-((long)result_char_c*100))/10;
                    result_char_a=(result_adc-((long)result_char_d*1000)-((long)result_char_c*100)-
((long)result_char_b*10));

          LCD_array[5]=result_char_d+48;
            LCD_array[6]=result_char_c+48;
          LCD_array[7]=result_char_b+48;
            LCD_array[8]=result_char_a+48;
          LCD_array[9]=' ';

                    //Change channel 7
          set_adc_channel(6);

      //Delay for channel changes
      delay_us(20);

                    result_adc=read_adc();

                    result_char_d=result_adc/1000;
                    result_char_c=(result_adc-((long)result_char_d*1000))/100;
                    result_char_b=(result_adc-((long)result_char_d*1000)-((long)result_char_c*100))/10;
                    result_char_a=(result_adc-((long)result_char_d*1000)-((long)result_char_c*100)-
((long)result_char_b*10));

            LCD_array[10]=result_char_d+48;
          LCD_array[11]=result_char_c+48;
            LCD_array[12]=result_char_b+48;
          LCD_array[13]=result_char_a+48;
            LCD_array[14]=' ';

                    //Change channel 8
          set_adc_channel(7);

      //Delay for channel changes
      delay_us(20);

                    result_adc=read_adc();

                    result_char_d=result_adc/1000;
                    result_char_c=(result_adc-((long)result_char_d*1000))/100;
                    result_char_b=(result_adc-((long)result_char_d*1000)-((long)result_char_c*100))/10;
                    result_char_a=(result_adc-((long)result_char_d*1000)-((long)result_char_c*100)-
((long)result_char_b*10));

          LCD_array[15]=result_char_d+48;
            LCD_array[16]=result_char_c+48;
          LCD_array[17]=result_char_b+48;
            LCD_array[18]=result_char_a+48;
          LCD_array[19]=' ';

                    //-------------------------

                    //Write to line 2
          LCD_display(1);

            //Display ON
                    LCD_busycheck();
                    LCD_write(0,0x0C);

      //Delay for sampling time
      delay_ms(50);

    }
```

```
}

void LCD_write(int data_RS, int data_DB)
{
          //Disable
          output_bit(PIN_B2,0);

  //Set DB as output
          set_tris_d(0x00);

          //Set register select
          output_bit(PIN_B1,data_RS);
  //Set write sequence
          output_bit(PIN_B0,0);
  //Write data
  port_d=data_DB;

  //Min delay=140 ns
  delay_us(1);

          //Enable
          output_bit(PIN_B2,1);

  //Min delay=450 ns
  delay_us(1);

          //Disable
          output_bit(PIN_B2,0);

  //Min delay=10 ns
  delay_us(1);

  //Clear data
  port_d=0x00;
          //Clear register select
          output_bit(PIN_B1,0);
  //Clear write sequence
          output_bit(PIN_B0,0);
}

int LCD_read(int data_RS)
{
          int data_DB;

          //Disable
          output_bit(PIN_B2,0);

  //Set DB as input
          set_tris_d(0xFF);

          //Set register select
          output_bit(PIN_B1,data_RS);
  //Set read sequence
          output_bit(PIN_B0,1);

  //Min delay=140 ns
  delay_us(1);

          //Enable
          output_bit(PIN_B2,1);
  //Min delay=1320 ns
  delay_us(2);

  //Read data
  data_DB=port_d;

          //Disable
          output_bit(PIN_B2,0);

  //Min delay=20 ns
  delay_us(1);

  //Set DB as output again (as default)
          set_tris_d(0x00);
  //Clear data
  port_d=0x00;
          //Clear register select
          output_bit(PIN_B1,0);
  //Clear write sequence
          output_bit(PIN_B0,0);

  return data_DB;
}

void LCD_busycheck()
{
          int busy_buffer;

  //Hold on if LCD is busy
  while(true)
  {
                  busy_buffer=LCD_read(0);
    busy_buffer=busy_buffer&0x80;

                  //if busy_buffer=0x80, the LCD is busy
    if (busy_buffer==0x00)
    {
                          break;
    }
  }

}

void LCD_display(int linenum)
{
  //linenum: 0=Line 1, 1=Line 2

  int count1;
  int outputchar;

          LCD_busycheck();

  if (linenum==0)
  {
    //Display 1st Line
    //----------------
    //Return home 1st line (DDR address=0x00)
          LCD_write(0,0x80);
  }
  else
  if (linenum==1)
  {
    //Display 2nd Line
    //----------------
    //Return home 2nd line (DDR address=0x40)
            LCD_write(0,0xC0);
  }

  for(count1=0;count1<=19;count1++)
  {
            outputchar=LCD_array[count1];

                        LCD_busycheck();
                        LCD_write(1,outputchar);
  }

}

void display_intro()
{
  //Display OFF
            LCD_busycheck();
            LCD_write(0,0x08);

  LCD_array[0]=' ';
  LCD_array[1]=' ';
  LCD_array[2]=' ';
  LCD_array[3]=' ';
  LCD_array[4]='I';
  LCD_array[5]='n';
  LCD_array[6]='t';
  LCD_array[7]='e';
  LCD_array[8]='l';
  LCD_array[9]='l';
  LCD_array[10]='i';
  LCD_array[11]='g';
  LCD_array[12]='e';
  LCD_array[13]='n';
  LCD_array[14]='t';
  LCD_array[15]=' ';
  LCD_array[16]=' ';
  LCD_array[17]=' ';
  LCD_array[18]=' ';
  LCD_array[19]=' ';

            //Write to line 1
  LCD_display(0);

  LCD_array[0]=' ';
  LCD_array[1]=' ';
  LCD_array[2]=' ';
  LCD_array[3]=' ';
  LCD_array[4]='P';
  LCD_array[5]='a';
  LCD_array[6]='r';
  LCD_array[7]='k';
  LCD_array[8]='i';
  LCD_array[9]='n';
  LCD_array[10]='g';
  LCD_array[11]=' ';
  LCD_array[12]='L';
  LCD_array[13]='o';
  LCD_array[14]='t';
  LCD_array[15]=' ';
```

```
LCD_array[16]=' ';
LCD_array[17]=' ';
LCD_array[18]=' ';
LCD_array[19]=' ';

        //Write to line 2
LCD_display(1);

//Display ON
        LCD_busycheck();
        LCD_write(0,0x0C);

delay_ms(1000);

//Display OFF
        LCD_busycheck();
        LCD_write(0,0x08);

LCD_array[0]=' ';
LCD_array[1]=' ';
LCD_array[2]=' ';
LCD_array[3]=' ';
LCD_array[4]='D';
LCD_array[5]='e';
LCD_array[6]='s';
LCD_array[7]='i';
LCD_array[8]='g';
LCD_array[9]='n';
LCD_array[10]='e';
LCD_array[11]='d';
LCD_array[12]=' ';
LCD_array[13]='b';
LCD_array[14]='y';
LCD_array[15]=':';
LCD_array[16]=' ';
LCD_array[17]=' ';
LCD_array[18]=' ';
LCD_array[19]=' ';

        //Write to line 1
LCD_display(0);

LCD_array[0]=' ';
LCD_array[1]='N';
LCD_array[2]='i';
LCD_array[3]='k';
LCD_array[4]=' ';
LCD_array[5]='O';
LCD_array[6]='t';
LCD_array[7]='h';
LCD_array[8]='m';
LCD_array[9]='a';
LCD_array[10]='n';
LCD_array[11]=' ';
LCD_array[12]='M';
LCD_array[13]='o';
LCD_array[14]='h';
LCD_array[15]='a';
LCD_array[16]='m';
LCD_array[17]='e';
LCD_array[18]='d';
LCD_array[19]=' ';
```

```
          //Write to line 2
  LCD_display(1);

  //Display ON
          LCD_busycheck();
          LCD_write(0,0x0C);

  delay_ms(1000);
}

void display_check()
{
  //Display OFF
          LCD_busycheck();
          LCD_write(0,0x08);

  LCD_array[0]=0xFF;
  LCD_array[1]=0xFF;
  LCD_array[2]=0xFF;
  LCD_array[3]=0xFF;
  LCD_array[4]=0xFF;
  LCD_array[5]=0xFF;
  LCD_array[6]=0xFF;
  LCD_array[7]=0xFF;
  LCD_array[8]=0xFF;
  LCD_array[9]=0xFF;
  LCD_array[10]=0xFF;
  LCD_array[11]=0xFF;
  LCD_array[12]=0xFF;
  LCD_array[13]=0xFF;
  LCD_array[14]=0xFF;
  LCD_array[15]=0xFF;
  LCD_array[16]=0xFF;
  LCD_array[17]=0xFF;
  LCD_array[18]=0xFF;
  LCD_array[19]=0xFF;

          //Write to line 1
  LCD_display(0);
          //Write to line 2
  LCD_display(1);

  //Display ON
          LCD_busycheck();
          LCD_write(0,0x0C);

  delay_ms(500);
}
```