

# MOBILE ROBOTS: OBSTACLE AVOIDANCE AND MANUVERING

By

MOHD SHAHRIMAN B M SHARIF

FINAL PROJECT REPORT

Submitted to the Electrical & Electronics Engineering Department  
In Partial Fulfillment of the Requirements  
For the Degree  
Bachelor of Engineering

(Electrical & Electronics Engineering)

Universiti Teknologi Petronas  
Bandar Seri Iskandar  
31750 Tronoh  
Perak Darul Ridzuan

©Copyright 2004

By

Mohd Shahrman B M Sharif,2004

t

TJ

211-415

.m697

2004

1) mobile robots

2) robotics

# CERTIFICATION OF APPROVAL

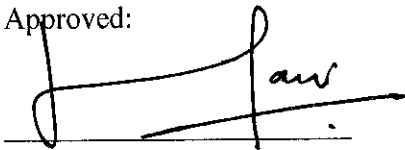
## MOBILE ROBOTS: OBSTACLE AVOIDANCE AND MANUEVERING

By

Mohd Shahrman B M Sharif

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

Approved:



Mr Mohd Haris B Md Khir  
Project Supervisor

UNIVERSITI TEKNOLOGI PETRONAS  
BANDAR SERI ISKANDAR  
31750 TRONOH PERAK  
DECEMBER 2004

## CERTIFICATION OF ORIGINALITY

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



---

Mohd Shahriman B M Sharif

## ACKNOWLEDGEMENTS

*In The name of Allah the Beneficent, the Merciful*

My utmost gratitude to Mr Mohd Haris Mohd Khir for allowing me to do a Final Year Project under his supervisions. Even with a tight time schedule, he has managed to support and give advices which are the key factor of finishing the project in time. Without his relentless effort to guide and supervise me, the project will not have been successful.

I would like to record my thank to Ms Nasreen and Mr Zuki , the coodirnator of Final Year Project. Without their management and coordination, the flow of this project might not be as expected.

Special thanks to Ms Siti Hawa who has given me 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 Azizan B Hashim, who has taught and give advice to me in many areas which was not known to me before. His help in giving ideas and programming skills is also one of the key factors of the success of this project.

Special thanks to Sarah Liyana Bt Roseley for being supportive in many ways through out this project. Without her warm support this project might not reach this level.

Last but not least, my utmost gratitude to my father M Sharif B M Lani and mother Nor Asmara Bt Ishak which has brought me up and help in many ways unimaginable. Only god may repay them. Also to my friends who 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

CERTIFICATION OF APPROVAL.....	i
CERTIFICATION OF ORIGINALITY .....	ii
ACKNOWLEDGEMENTS .....	iii
TABLE OF CONTENTS .....	4
LIST OF FIGURES.....	6
LIST OF TABLES.....	7
ABSTRACT .....	8
<b>CHAPTER 1: INTRODUCTION .....</b>	<b>9</b>
1.1 BACKGROUND OF STUDY .....	9
1.2 PROBLEM STATEMENT.....	10
1.3 OBJECTIVE AND SCOPE OF STUDY .....	11
1.4 THE RELEVANCY OF THE PROJECT .....	12
1.5 FEASIBILITY OF THE PROJECT WITHIN SCOPE AND TIME FRAME	12
<b>CHAPTER 2: LITERATURE REVIEW AND/OR THEORY .....</b>	<b>13</b>
2.1 STRUCTURE.....	13
2.2 MOBILITY AND MOVEMENTS.....	13
2.3 POWER DISTRIBUTION AND FAIL SAFE.....	14
2.4 SENSORS.....	15
2.5 MICROCONTROLLER.....	18
2.6 DIFFERENTIAL DRIVE CONTROL ALGORITHM.....	18
<b>CHAPTER 3: METHODOLOGY OF PROJECT WORK.....</b>	<b>21</b>
<b>CHAPTER 4: RESULTS AND DISCUSSION .....</b>	<b>24</b>
4.1 STRUCTURE AND MOBILITY.....	24
4.2 POWER DISTRIBUTION AND FAIL SAFE.....	26
4.3 ROTARY ENCODER.....	28
4.4 INFRARED SENSOR.....	31
4.5 ULTRASONIC SENSOR.....	33
4.6 LINE FOLLOWER .....	34
4.7 MICROPROCESSORS .....	35
4.8 ALGORITHM .....	38
4.8.1 <i>Coordinate to path conversion</i> .....	38
4.8.2 <i>Dead reckoning</i> .....	39
4.8.3 <i>Error correction codes</i> .....	40
4.8.4 <i>Obstacle Avoidance</i> .....	41
4.8.5 <i>Line Follower</i> .....	44
4.9 DRIVE CIRCUIT .....	45
4.10 RS232 COMMUNICATION .....	47
4.11 SERVO CONTROLLER.....	48
4.12 AS A WHOLE.....	50

<b>CHAPTER 5: RECOMMENDATION</b> .....	<b>52</b>
5.1 STRUCTURE DESIGN .....	52
5.2 DATA COMMUNICATION .....	52
5.3 ACCURACY .....	52
5.4 ACTUATORS .....	53
<b>CHAPTER 6: CONCLUSION</b> .....	<b>54</b>
<b>REFERENCES</b> .....	<b>56</b>
<b>APPENDIX A: MAIN SOURCE CODES</b> .....	<b>57</b>
<b>APPENDIX B: SERVO CONTROLLER SOURCE CODES</b> .....	<b>86</b>
<b>APPENDIX C: MOBILE ROBOT PARTS</b> .....	<b>88</b>
<b>APPENDIX D: L298 DATASHEET</b> .....	<b>91</b>
<b>APPENDIX E: PIC16F877 DATASHEET</b> .....	<b>92</b>
<b>APPENDIX F: PIC16F84A DATASHEET</b> .....	<b>93</b>
<b>APPENDIX G: OPERATION MANUAL</b> .....	<b>94</b>
<b>APPENDIX H: COMPONENTS LIST</b> .....	<b>95</b>
<b>APPENDIX I: ALGORITHM FLOWCHART</b> .....	<b>99</b>

## LIST OF FIGURES

FIGURE 2-1 A SIMPLE BLACK & WHITE ENCODER.....	15
FIGURE 2-2 IS1U60 INFRARED DETECTOR .....	16
FIGURE 2-3 PIC 16F877 FROM MICROCHIP .....	18
FIGURE 3-1 FLOW OF DESIGN .....	21
FIGURE 3-2 WORKFLOW FOR PART OR STAGES .....	22
FIGURE 4-1 LOCATION OF MOTOR AND TIRES IN RADIUS.....	24
FIGURE 4-2 ACTUAL DESIGN IMPLEMENTATIONS.....	25
FIGURE 4-3 ORTHOGRAPHIC DRAWING.....	25
FIGURE 4-4 5V VOLTAGE REGULATOR .....	27
FIGURE 4-5 A SIMPLE BLACK & WHITE ENCODER.....	28
FIGURE 4-6 OUTPUT SIGNAL FROM ENCODERS.....	29
FIGURE 4-7 INFRARED TRANSMITTER AND DETECTOR.....	30
FIGURE 4-8 INFRARED TX OSCILLATOR .....	31
FIGURE 4-9 MODULATED INFRARED SIGNALS .....	32
FIGURE 4-10 LINE FOLLOWER SENSOR .....	34
FIGURE 4-11 LINE FOLLOWER ALGORITHM.....	34
FIGURE 4-12 MICROCONTROLLER CONNECTIONS.....	36
FIGURE 4-13 AXIS ALGORITHM.....	38
FIGURE 4-14 DEAD RECKONING ALGORITHM.....	39
FIGURE 4-15 PWM NEEDED FOR A STRAIGHT LINE .....	40
FIGURE 4-16 ERROR CORRECTION CODES ALGORITHM .....	41
FIGURE 4-17 CASE 1: SIMPLE OBSTACLE (SAME ALGORITHM ON LEFT OR RIGHT MOVEMENT).....	41
FIGURE 4-18 CASE 2: OBSTACLE ON THE LEFT OR RIGHT (SAME ALGORITHM ON LEFT OR RIGHT MOVEMENT).....	42
FIGURE 4-19 CASE 3: WALL OBSTACLE (SAME ALGORITHM ON LEFT OR RIGHT MOVEMENT).....	43
FIGURE 4-20 OBSTACLE AVOIDANCE ALGORITHM.....	44
FIGURE 4-21 LINE FOLLOWER ALGORITHM.....	45
FIGURE 4-22 DOUBLE 4 A H-BRIDGE .....	46
FIGURE 4-23 RS232 COMMUNICATION CIRCUIT MODULE.....	47
FIGURE 4-24 WIDTH OF PWM FOR SERVO @ CASTER.....	48
FIGURE 4-25 SERVO CONTROLLER.....	49
FIGURE 4-26 PATH OF ROBOT .....	50
FIGURE 0-1 FRONT SETUP MOUNT OF EACH TIRES .....	88
FIGURE 0-2 INFRARED / ROTARY ENCODER / STRIPES .....	88
FIGURE 0-3 ULTRASONIC TRANSMITTER AND RECEIVER.....	89
FIGURE 0-4 INFRARED TRANSMITTER / DETECTOR .....	89
FIGURE 0-5 REAR SERVO CONTROL CASTER .....	89
FIGURE 0-6 THE WHOLE ROBOT LAYOUT .....	90

## LIST OF TABLES

TABLE 3-1 SEQUENCE OF PARTS AND TOOLS .....	23
TABLE 4-1 SPECIFICATION OF THE STRUCTURE .....	26
TABLE 4-2 SPECIFICATION OF VOLTAGE REGULATOR BOARD .....	27
TABLE 4-3 BINARY STATES FOR MOVEMENTS .....	37
TABLE 4-4 MICROCONTROLLER BOARD SPECIFICATION .....	37
TABLE 4-5 WIDTH OF PWM FOR SERVO @ CASTER .....	48



## ABSTRACT

The objective of the project is to design and implement a mobile robot. The robot will be able to avoid obstacles and have its own decision making capability. It will be a part of the preparation for the department of Electrical Electronics Engineering University Teknologi Petronas for various activities which require robotics design participation. As one of the final year electrical electronics student I have been given the chance to be part of the path finding.

The scope of the study will be mainly on the design and implementation of the robot from scratch or little knowledge. The study will be handled part by part for components needed for the robot. First will be the structure of the robot. Next the mobility and drive circuit will be design to enable the robot to be mobile. Sensors will be put in place so that the robot is able to “feel” and “see” it’s surrounding. When this is done a “brain” or microcontroller will be put in place so that it is able to control itself and make simple decision.

The methodology or approach can be divided into software and hardware. Basically the same methodology will be use again and again in the module design process. Finally the parts will be integrated as one mobile robot. This will become the final robot.

In the discussion part, the findings are being discussed in detail. The problem and the solution for the problem is being discussed base on the student point of view. The reader might get the idea on the limitations of a mobile robot as the size and weight increases. These are the factors that are becoming the bottle neck in this design project.

Before ending the chapter, some recommendation has been suggested for further improvement by future robotics builder. The suggestions are made base on the current available technology and also the experience gain by the student through out this design project.

# CHAPTER 1: INTRODUCTION

## 1.1 BACKGROUND OF STUDY

Autonomous robot has become common in this new world of high technology. Autonomous means that the robot is able to stand alone in doing some decision making such as avoiding obstacle and also to follow certain path without any human intervention. This is normally related to artificial intelligent which is still being study in leading university through out the world. The intelligence of the algorithm is implemented in small and even large scale robots.

The robot diverse from the type of the movement to the type of environment they were created for. There are bipedal robots which move on two sets of “foot” and also there are robots which use tires to move. There even robots which can fly with the help of some wings or blades which commonly originates form model planes or helicopters. What ever the size or the environment it is built for, the function of the robot is crucial depending on the task that has been specified for it. NASA astonishes the world when 2 of their autonomous robot have been safely landed on the surface of mars. These robots were sent to Mars due to the environment which is hostile to the human at the moment. These robots were able to conduct some experiments to gather information for the scientist millions of miles away. The crucial thing here is that both of these robots are autonomous. They are able to make decision on what path to choose and able to avoid any accident such as ending up in a crater. This is call obstacle avoidance intelligence.

## 1.2 PROBLEM STATEMENT

There is a need to design robot that is capable of carrying very high load which is roughly around 10kg. The robot must be mobile, which means the robot must be able to move in the direction required. Sensors must be use as to give information of the surrounding to the robots. These will be the eyes and ears for the robots. Finally the robot must be intelligent enough to think or decide on its own to do certain task such as avoiding obstacles on the route that has been predetermine.

Through out the period of study all these parts of the robot has been design and develop and finally integrated into one working robot. The robot has been developed part by part as to ensure the parts are working before all the components are integrated. The project is significant as it will develop the fundamentals of robotics needed for the university to further the study on this exciting area of electrical electronics engineering. Fields such as circuit design drive design, micro  $\mu$ rocessing, programming and mechanical design is being used to achieve this.

### **1.3 OBJECTIVE AND SCOPE OF STUDY**

The objective of the study is to design and implement the mobile robot with the obstacle avoidance capability. Besides that the robot must also be able to handle heavy load. This has led to design of sub parts of the robots

#### **1.3.1. Structure design**

The robot will need a structure or base body for the whole part can reside. The challenge is to design a sturdy body without compensating on the weight. Heavy weight for the body will lead to more power consumption and heavier on the design for the motor and gearing.

#### **1.3.2. Mobility and movement**

In order to be mobile, the robot needs to have “legs” to move. The design will only concentrate on robots which move using dc motors and tires. The specifications of the motor depend largely on the type of load that the robot will carry. These will be the major constraint besides the availability of the component locally. Drive circuit might also be needed to control the current and voltage supplied to the motors.

#### **1.3.3. Power distribution and fail safe**

The power that is being supplied to all the components needs to be regulated at a certain value. This will lead to some design on the power system for the robot. Some circuit will be needed to check the voltage status of the robot and also the charging of the batteries on the robot. As the size of the robot is relative large a fail safe mechanism will be implemented to avoid any hazard to its surrounding. Example of this fail safe is fuses and also current sensors.

#### **1.3.4. Sensors**

Sensors are transducers that can be used to measure some parameter of the surrounding. This information will then be process by the microprocessors to check for some event that will trigger another event. This is a way for the robot to see the world around it.

### **1.3.5. Microprocessors or Microcontrollers**

Microprocessors or microcontrollers are the brain for the robot. This will be the component that will decide what to do in conjunction of some event detected by the sensors. The microcontroller will be able to make some minor decision which in the end will control the movement of the robot base on the environments.

## **1.4 THE RELEVANCY OF THE PROJECT**

There are a lot of other universities that is doing research on robotics. It is really relevance to develop a robot in house to be established as one of major university. Besides that the robot will serve as basis for further studies in the future. Having an in house design robot will lead to detail understanding of the robot mechanism itself. This study will lead to further detail development and also applications on the in house robots. The university can use the research material to compete in robotics competition around the world.

## **1.5 FEASIBILITY OF THE PROJECT WITHIN SCOPE AND TIME FRAME**

The project is considered feasible based on the time given and also the abundance of information on robots from the internet and also the library.

The first half of the project period will concentrate more on the basic structure of the robot itself such as the structure, drive circuit and the power distribution and fail safe.

The second half of the project has been concentrated on the sensors and also the implementation of algorithm for the robot to be autonomous. The robot design process might be better if it was a team effort.

## **CHAPTER 2: LITERATURE REVIEW AND/OR THEORY**

Literature review is the research that has been done by collecting information from various sources such as from the internet and books. The research for the robot has been done part by part of the sub component for the robot such as the structure, mobility and movement, power distribution and fails safe, sensors and microprocessors.

### **2.1 STRUCTURE**

There are many shape and sizes for the structure. The design will depend on the application or the task the robot will do. For example a robot that is needed to be put into Martian land need to be able to withstand the corrosive environment, the high level of temperature difference and also light. The shape can be round which is good for tight spaces and also box like shape spaces. The shape and sizes does not matter much as long as the drive motor is sufficient to carry the load. Many robot designers would prefer to use light weigh material which is easy to be shape into the desired shape. Many of the design uses rounded shape as it will give better movement in tight space. Method of connecting these parts will be either using welding method or nut and screw method. For high specification robots the parts are specially molded and then the parts are created using some other techniques which gives high degree of precision.

### **2.2 MOBILITY AND MOVEMENTS**

The mobility of the robot will depend on some factors. First will be the estimated load that the robot needs to carry. This is the weight including the weight of the robot that will be moved by motors or other mechanism. The motor must have sufficient torque to move the load. This has been a great challenge. Number of tires will also determine the required torque for the robot. Rule of thumb is that as the number of tires increase the value for the start up torque will be less. This has been studied using some simple experiments in the lab. As to ease of giving mobility 3 tires

will be used, 2 of them will be coupled with dc motors and another one will be let free as a support. These combinations have been tested and prove to be workable.

### **2.3 POWER DISTRIBUTION AND FAIL SAFE**

In many large scale robots the power robots will be supplied by rechargeable batteries. There are many type of batteries which is suitable for robotics. The difference is in the material use for the batteries and also the number of Ampere that the batteries can supply in a specific number of periods. The higher the value of the number of ampere per hour for the batteries the longer it can supply power to the robot. The drawback is that the batteries will need special care such as a specific way of charging and discharging and also the price are relatively high.

Fail safe is important in robots as it uses a lot of expensive and sensitive components that might get destroy due to surges or short circuit. Fail safe circuit such as circuit breakers needs to be put in place to overcome this. Normally the fail safe feature will be implemented along with the voltage and regulator circuit. The circuit will detect any abnormality and decide to break the connection or do some other preventive measure.

Due to the size of the robot is relatively big another failsafe will be put in place such as remote off. This can avoid any unwanted accident due to the robot malfunction or breakdown. The remote can automatically turn off the robot from a distance.

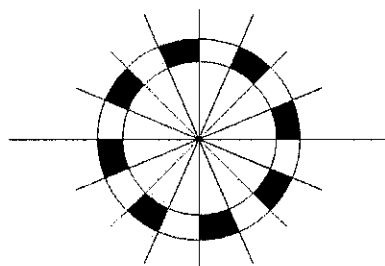
## 2.4 SENSORS

Sensors are transducers which are capable of changing one state of energy to another. These characteristics can be used to measure the analog world. This transducer will give output in current or voltage variation. We can then use this information to manipulate the movement of the robot. There are a lot of transducers than can be use. The sensor use is as follow;

### 2.4.1. Rotary Encoder

One of the ways to control the rotation or to get data from the wheel rotation is to apply rotary encoder setup to the robot. There are various ways to do this depending on the selection of sensor which is available for use. The basic idea of this technique is two detect to different state for example ON and OFF. By detecting the transition of ON and OFF the rate of change can be measured. This change of state implies to several information such as speed, position and synchronizaauon of one wheels. Having the rotary encoder can be helpful by feeding data to the microcontroller.

Normally the simplest setup is to have black and white strips across the wheels or couple to the wheels. These black and white strips can be detected by having a simple infrared transmitter and detector to detect its state. This signal will then be converted to a stream of 1 and 0 pulses which then fed to the microcontroller to be process to get information such as the speed and the distance travel. With further manipulation the data can be use for other more intelligent control of the robot.



**Figure 2-1 A simple black & white encoder**



### 2.4.2. Infrared Sensors

Infrared sensor is one of the most basic and cheapest sensors around. It uses modulated infrared light to detect obstacle. The transmitter will transmit a burst of signal which will then be detected by its detector. In the market there are numerous types of infrared diodes that can be use. They differ in size and also the radius in which the infrared can be used. One common problem with infrared sensor is false triggering. This happens when the detector accidentally detects ambient infrared from the surrounding and interprets it as an obstacle. This problem can be solved by few methods; one is by software and another one by hardware. In software the “polling” method can be use to identify which receiver is actually detecting the correct signal. This can led to reliable detection. By hardware, there are detectors in market which only detects infrared signal in a specific band of frequency. This is a more reliable solution to the problem. The only problem with this method is the transmitted infrared signal needs to be modulated at the frequency detectable by the detector. Having this the infrared can be use as one of the reliable sensors at a good distance.

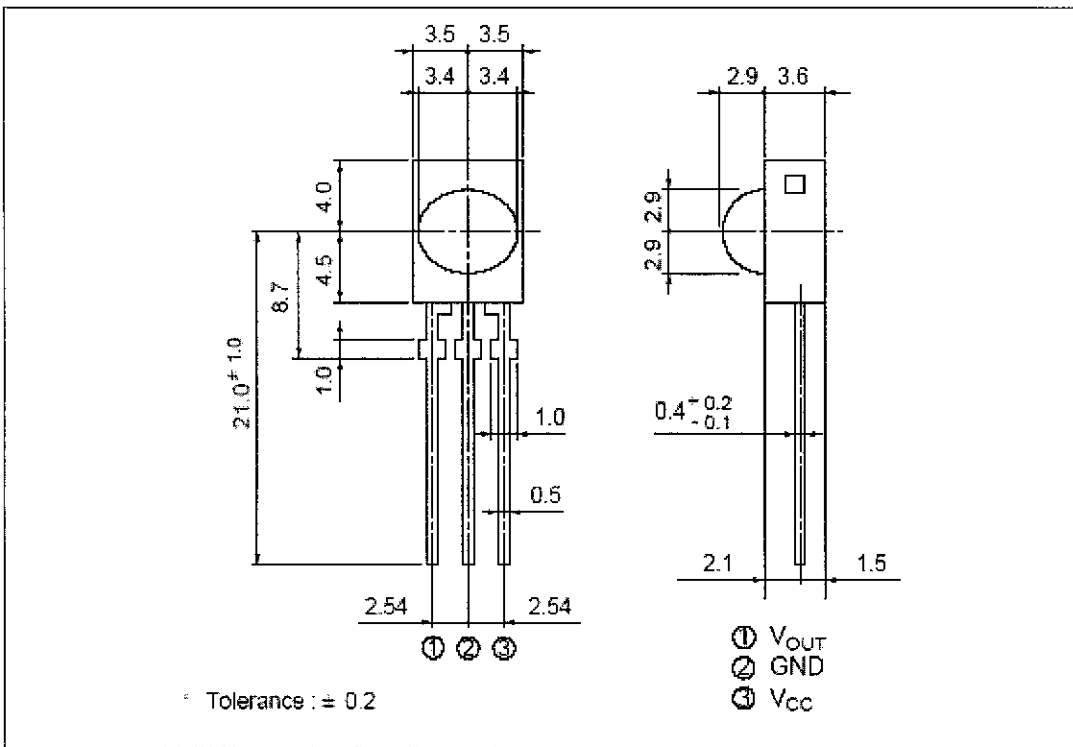


Figure 2-2 IS1U60 Infrared Detector

### **2.4.3. Ultrasonic Sensor**

Ultrasonic sensor is another common sensor use in robotics. This type of sensor uses high frequency sound burst which is generated by certain means at certain frequency. The bounced echo will be then picked up by the ultrasonic receiver which will gives the signals to a set of amplifiers and finally converted to digital signal by means of comparator or other method. The output from the ultrasonic can be use as simple as to detect obstacle or to be use as ranging module. In order to use the ultrasonic as a ranging module the output of the receiver needs to be connected to an analog to digital converted which is fed to the microprocessor. The microprocessor can then determine the range base on the voltage value on its input.

Ultrasonic sensor signal will attenuate at a distance. So in order to get a better range a higher power is needed to create the ultrasonic chirp. This is a problem if the robot carries a limited power supply. Another drawback is the accuracy of the sensor will fluctuates depending on the temperature and the air density of its surrounding. This is common to some of the “low grade” transducer that is used. With higher grades of transducers, they have design intelligent ways to compensate it surrounding effects.

### **2.4.4. Line Follower**

A line follower sensor is actually a set of sensor use to detect a “landmark” or route which can be black lines or maybe metal strips on the floor or path in which the robot moves on. Normally in some industry metal strips is placed under the path in which autonomous robots is used to carry some item from one point to another point accurately. This method can also be applied using other sensor such as infrared sensor which can be use to detect lines or variation of markings which then can be use to guide the robot to the destination. In this project the line follower circuit is used to reduce the error that is encounter by the robot.

## 2.5 MICROCONTROLLER

There are several microprocessors that can be used to control a robot. Normally the microprocessor used will also have embedded memory. When a memory is embedded the microprocessor is known as a microcontroller. There are many types of microcontroller available in the market such as PIC from Microchip, Atmel, HC family and many more. The difference is in the architecture and the language used. The basic and common microcontroller is the PIC. A version of PIC 16F877 is a potential candidate for the microcontroller as it has a lot of digital analog converter and PWM output that can be used for sensors and also driver circuit switching.

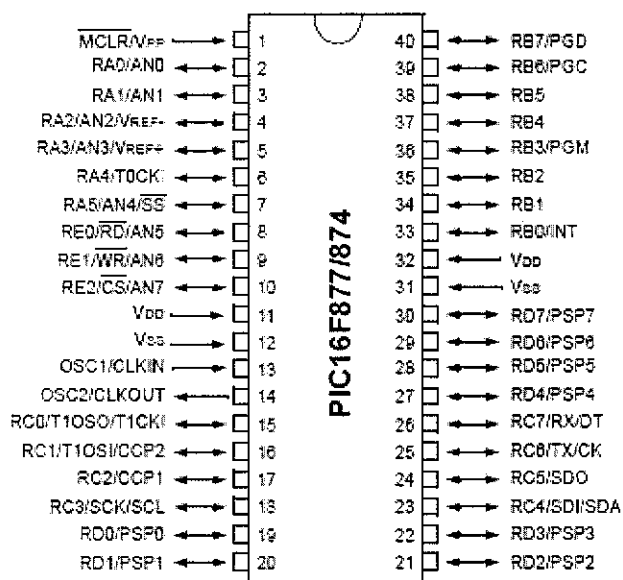


Figure 2-3 PIC 16F877 from Microchip

## 2.6 DIFFERENTIAL DRIVE CONTROL ALGORITHM

Dead reckoning (derived from deduce reckoning) is a simple mathematical procedure for determining the present location of a vessel by advancing some previous position through known course and velocity information over a given length of time (Dunlap & Shufeldt, 1972). The concept was adapted to automobile control in early 1910 and now it is used in robotics to answer some questions such as “where am I now”, “how far”, “how fast” and “where I need to go” for the robot.

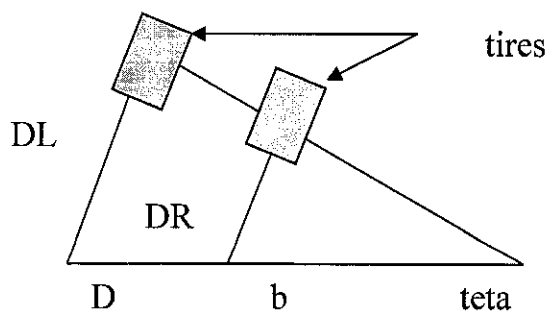
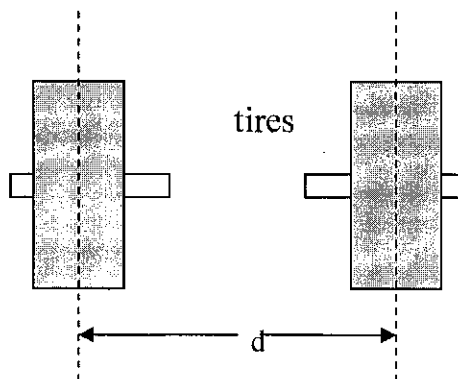
Using the rotary encoder the data needed can be fed to the microprocessor which will then monitor the rotation of the motors by implementing some mathematical analysis as below.

$$D = \frac{D_{left} + D_{right}}{2}$$

**Equation 1 Displacement**

$$V = \frac{V_{left} + V_{right}}{2}$$

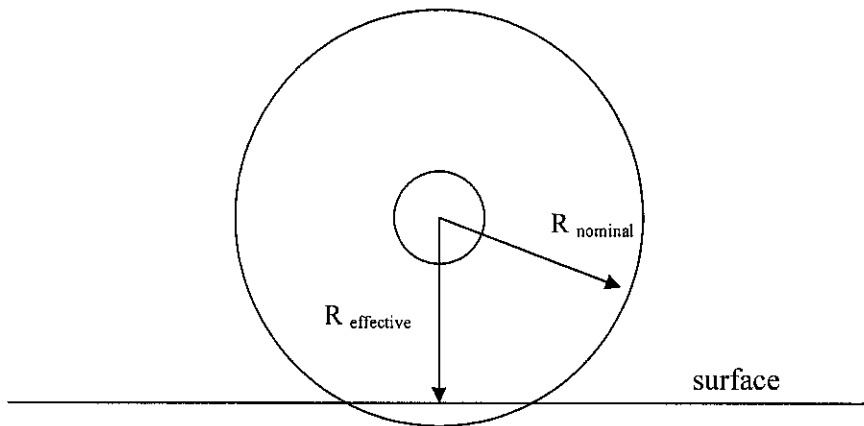
**Equation 2 Velocity**



$$\theta = \frac{D_{left} - D_{right}}{d}$$

**Equation 3 Theta**

This method of calculating theta is susceptible to error due to bump in the surfaces. Therefore the effective wheel radius needs to be considered to reduce the errors due to uneven surfaces.



Expressing in terms of encoder counts

$$D_l = \frac{2\pi N_l}{C_l} R_{el}$$

**Equation 4 Encoder counts left**

$$D_r = \frac{2\pi N_r}{C_r} R_{er}$$

**Equation 5 Encoder counts right**

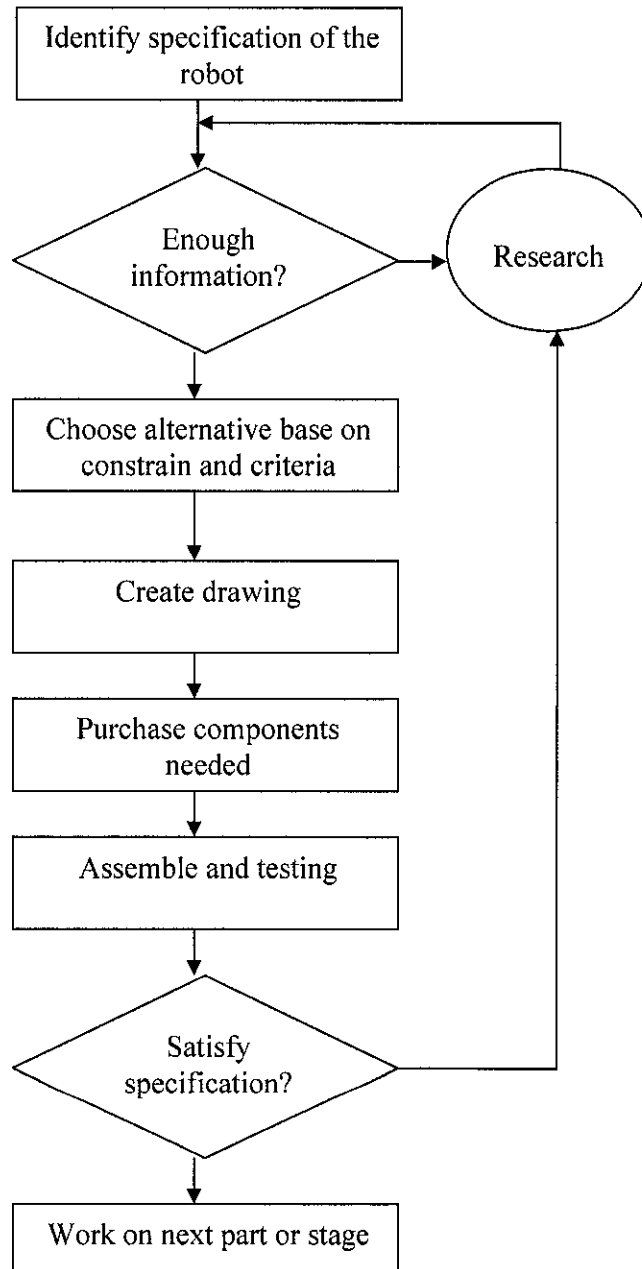
$N_x$  = number of count on left/right encoders

$C_x$  = encoder counts per wheel revolution

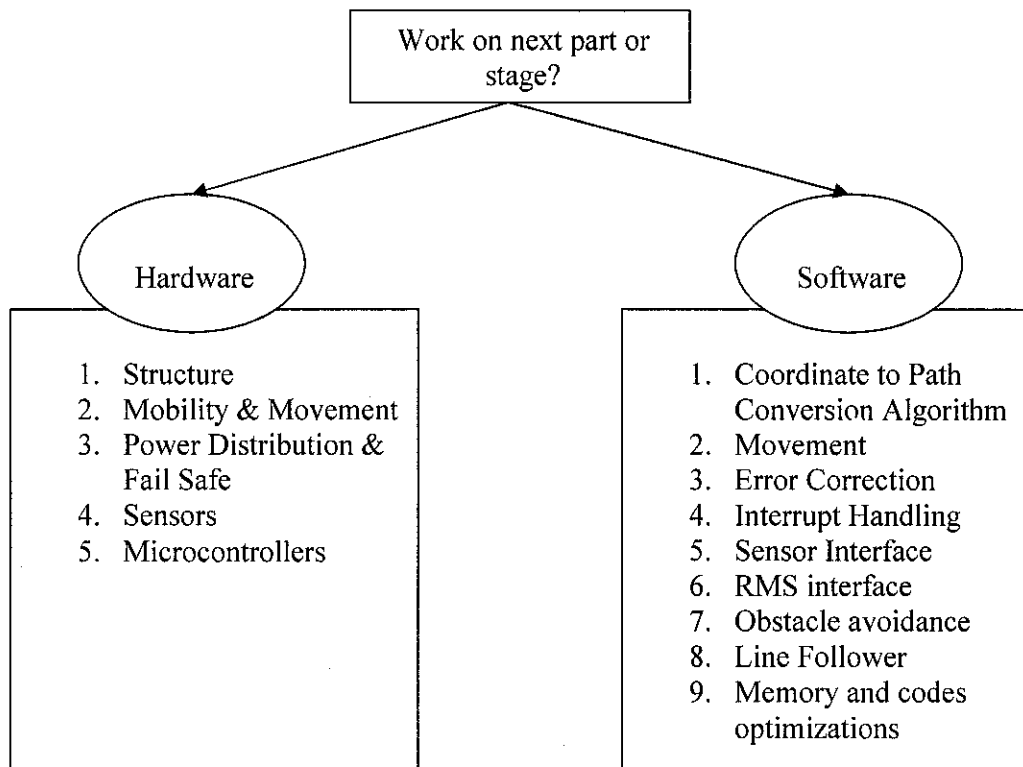
This method does not give a 100% accurate rotation as error might be introduced due to bumps on the surface and the friction factor of the surface, but this dead reckoning technique has been proven to be able to reduce the error rate if the technique is not being implemented.

## CHAPTER 3: METHODOLOGY OF PROJECT WORK

The flow of the project has been done according to a simple flow which is then applied through out the project. The same flow is used on each sub parts of the robot.



**Figure 3-1 Flow of design**



**Figure 3-2 Workflow for part or stages**

The first step is to identify the task or the general specification of the robot such as what is the intended purpose of the robot. When the information is available, the next step is to do research on the parts that need to be design and develop. The alternatives will be then layout and then the best alternatives will be chosen base on the constraints and the criteria. Constrains are mainly on the availability of the components locally at a reasonable price. Next step will be to transfer the specification into drawing. From the drawing the practicality will be identified. Next the design needs to be approved by the supervisors. When this is done the components can be purchase. When all the component are available the part will be assemble and then it will be tested to see whether it meets the intended specification. If the result is satisfactory, the work on the next part or module will continue. Below are stages of the parts in sequence. Some of the stages overlap each other as it can be conducted in parallel. (Refer Gantt chart).

**Table 3-1 Sequence of parts and tools**

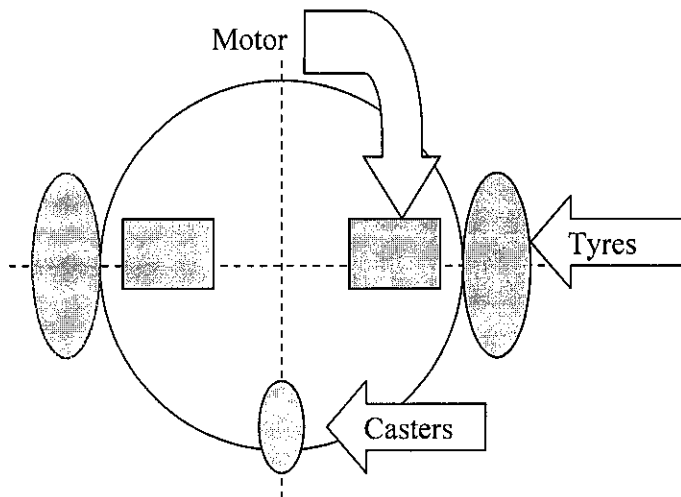
No	Parts or stage	Material or components used	Tools Used
1.	Structure	1. Metal Sheet	1. Drill 2. Arc welding
2.	Mobility and movement	1. Motor with gearbox 2. Tires & Caster 3. Nuts and screw	1. Screw driver 2. Pliers
3.	Power distribution and fail safe	1. Batteries ( seal lead acid) 2. Connectors & Wires	1. Solders 2. Multi meter 3. Screw driver
4.	Sensors	1. Ultrasonic, infrared transducers 2. Electronics components ( transistor, resistors etc )	1. Simulation software ( Multisim or Pspice ) 2. Breadboard 3. Solder 4. Multi meter 5. Oscilloscope
5.	Microprocessors	1. PIC16F877 (Microchip)	1. Breadboard 2. Electronics components 3. Oscilloscope 4. Mplab ( PIC programming software) 5. PIC C compiler 6. Connectors



## CHAPTER 4: RESULTS AND DISCUSSION

This part will consist of results and discussion on the project itself base on observation, findings and evaluation on the robot behavior and characteristics

### 4.1 STRUCTURE AND MOBILITY



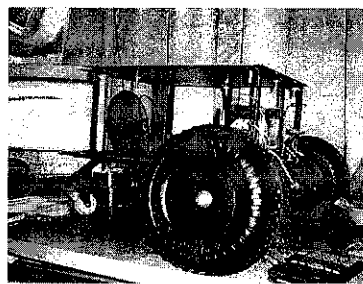
**Figure 4-1 Location of motor and tires in radius**

The structure design that is being implemented is as above. The robot will be driven by two independent dc motor which is mounted to the structure of the robot by the shaft that is connecting the tires and the gearbox. This connection has been tested as not reliable and further more reliable method of mounting is still being tested. This unreliable method of mounting has been the reason for the misalignment of the movement of the robot. The third wheel is a caster wheel which is only use as a supporting wheel. The third wheel is fixed using a servo in order to further reduce the error rate. A feedback navigation system needs to be developed to reduce the error.

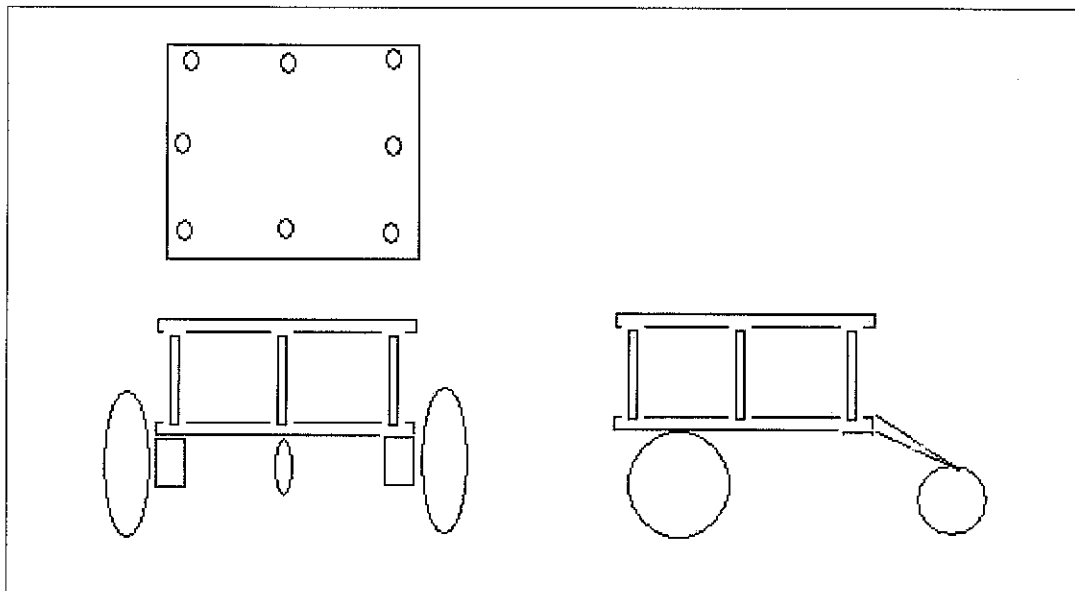
The position of the wheels is in radius of each other due to a reason. When the wheels are in radius of each other, theoretically the movement of the robot will be much easier and enable the robot to make a round turn and save the turning radius.

But in the implementation the caster radius has been increase slightly out of radius but it still work the same.

The structure is being built from mild steel which makes the structure strong but heavy. Two plates of thin metal is being use as the base and the top cover for the robot. The structure is then painted with silver paint to avoid corrosion from take place. The structure implements screw and detach system which enables the robot to be stored in a small area for transporting. One problem is that the nuts and screw needs to have secondary nuts as to ensure that screw and nuts will not move place after a while.



**Figure 4-2 Actual design implementations**



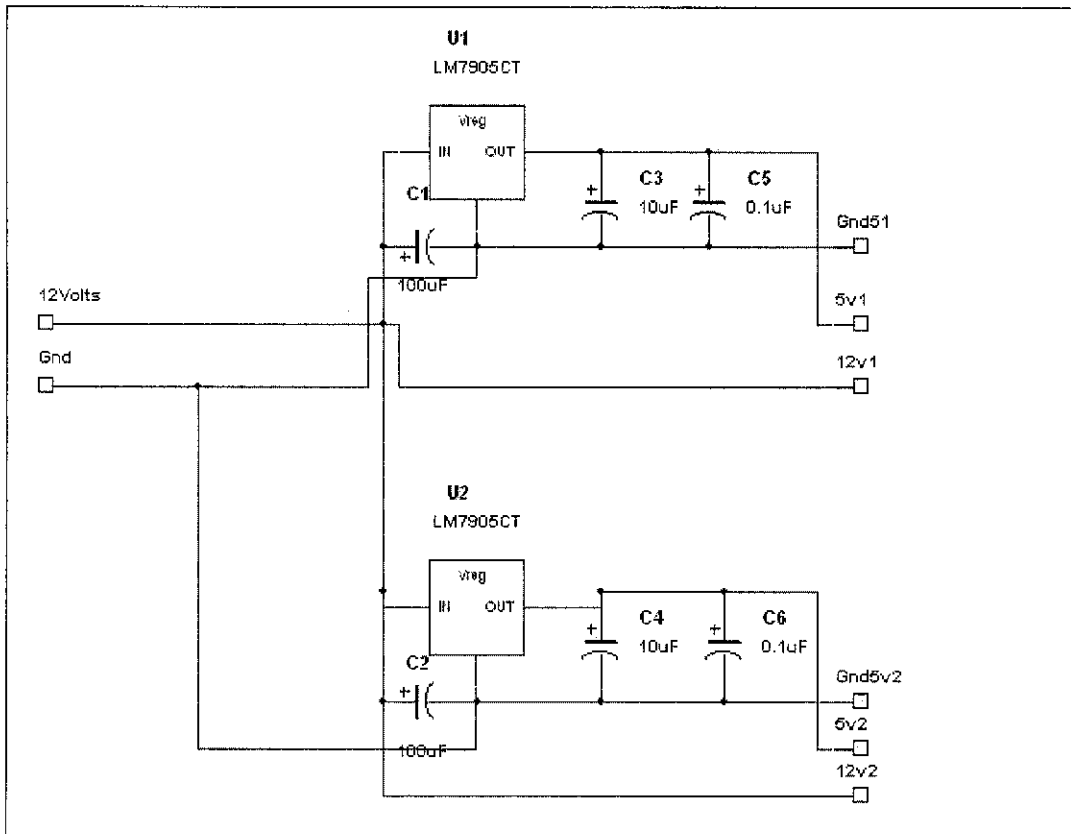
**Figure 4-3 Orthographic Drawing**

**Table 4-1 Specification of the structure**

Width	30 cm
Length	40 cm
Height	28 cm
Net Weight	2 kg
Material	Mild Steel

#### **4.2 POWER DISTRIBUTION AND FAIL SAFE**

In order to power up the robot, a sealed lead acid batteries will be the source of the power of the motor and the H-bridge board. In order to supply a high voltage due to the characteristics of the H bridge circuit, the 6.8V batteries are connected in parallel. The positive line of the battery line is connected to a fuse as to ensure safety to the circuit. The 12 to 13 volt that being supplied will be connected to the H bridge board (deliver power to the motor) and also to the regulator board which will supply a regulated 5v to the H bridge board. The microprocessor power circuit will be on separate board as to ensure electrical isolation from high current and voltage that might destroy the microcontroller.



**Figure 4-4 5v Voltage regulator**

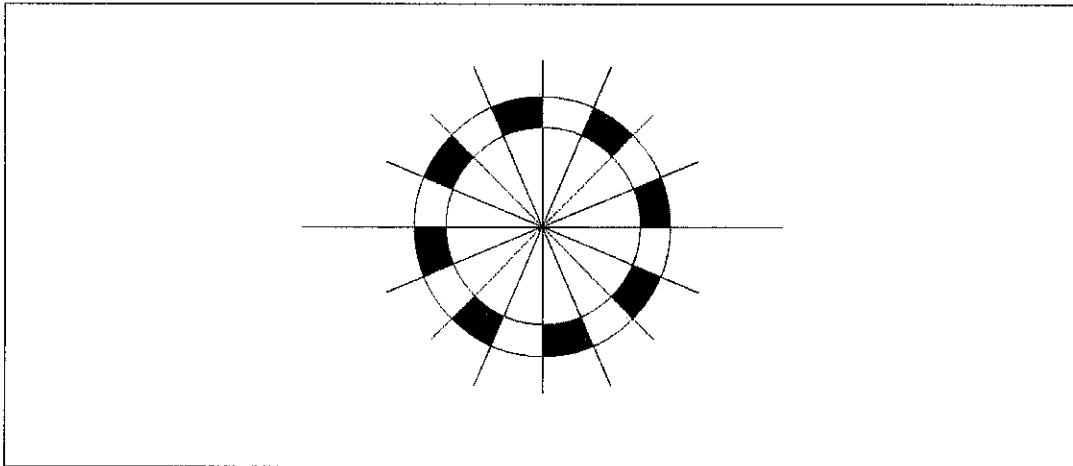
**Table 4-2 Specification of voltage regulator board**

Input Voltage	12 to 13
Output Voltage	4.89 to 5 Volt
Current	1A

In order to regulate the voltage to a level which can be used for digital circuit, a voltage regulator is used. The voltage regulator is a built in device which is capable of stepping down DC voltage to the desired voltage. Internally the circuit consist of transistor which open and close in the process to regulate the output voltage. If the voltage at the output is less the transistor will open more thus giving more voltage at the output. In terms of current the voltage regulator supplies more than we need which is around 1 A. In order to ensure that the voltage regulator work at its operating point, a heat sink is attached to the power regulator package.

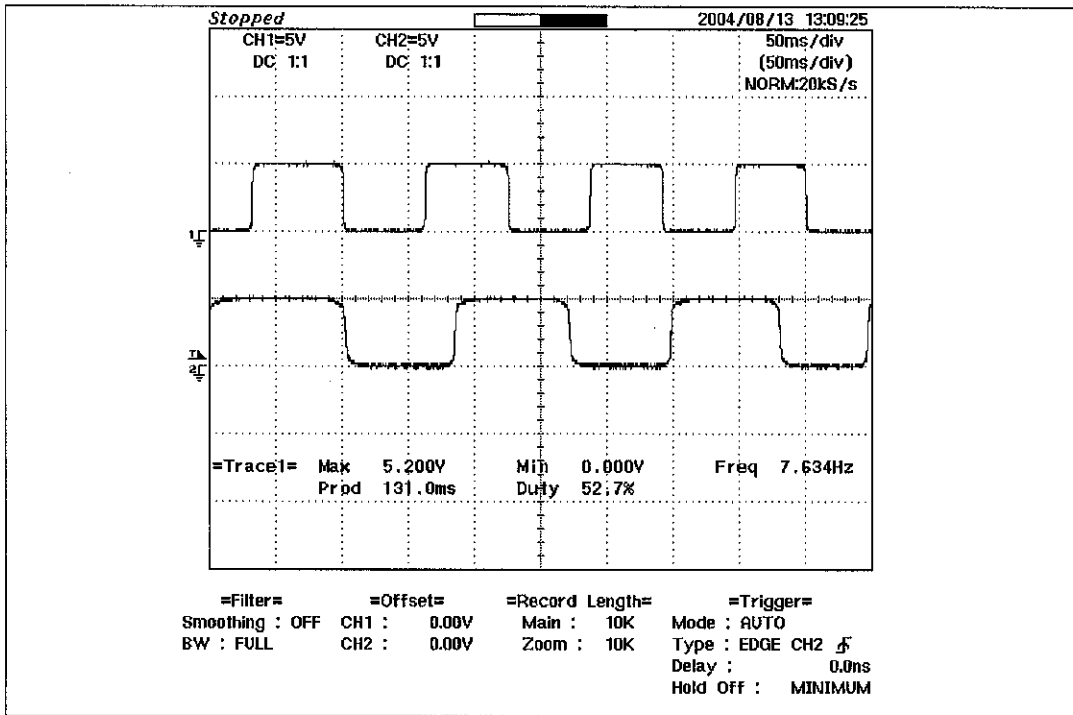
### 4.3 ROTARY ENCODER

The rotary encoder has been fixed on both front tires. On the tires a strip of black and white has been placed. In order to change this analog form into a digital form an infrared transmitter and detector is used. The infrared transmitter and detector circuit is capable of differentiating the black and white color. This signal is then fed into a NOT gate so that the digital value is much better.



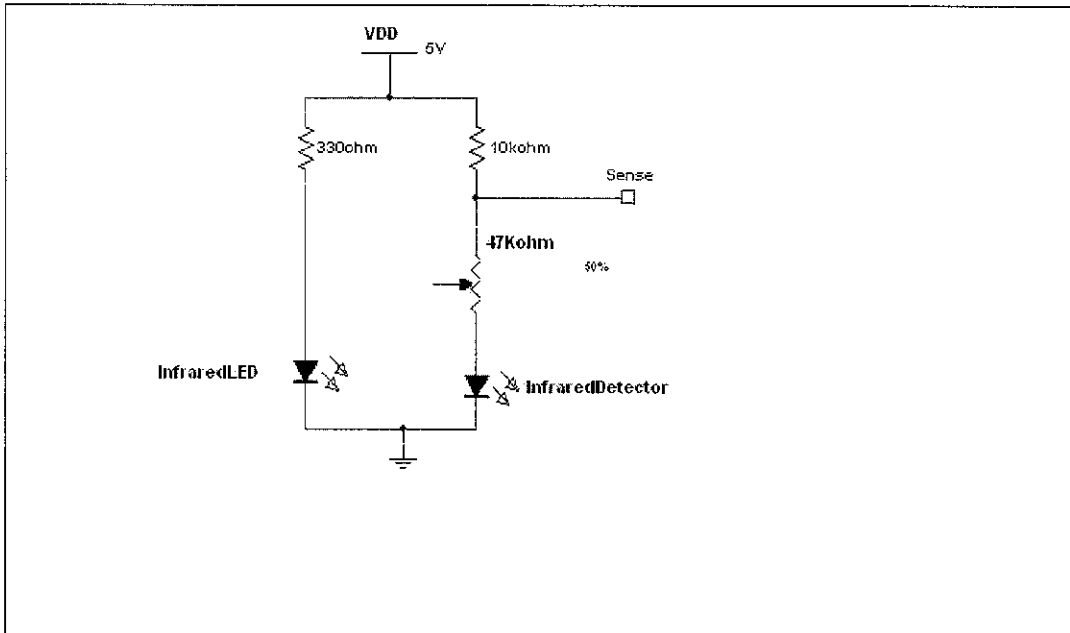
**Figure 4-5 A simple black & white encoder**

The output of the encoder can be seen as depicted below. From the signal itself we can see that with the same PWM value the speed on each motor is actually different.



**Figure 4-6 Output signal from encoders**

This digital signal is then fed into the microprocessors for manipulation in order to get the speed, distance travel and also in implementation of a simple error correction codes. Above is the signal that has been captured from the rotary encoder. The top signal refers to the left wheel encoder and the bottom signal refers to the right wheel encoders. The rotation of the wheel can now be represented in terms of pulse. This pulse is used to measure the distance travel by the robot. The distance value is also use in error correction codes to improve the accuracy of the robot.



**Figure 4-7 Infrared Transmitter and Detector**

The encoder use does not have a good resolution. For every pulse it represent 1.3cm in distance that the tires has turn. Therefore we can aspect at least an error rate of 1 pulse or 1.3cm per meter. From the test run done, the robot is capable of moving to the target located at certain distance at a good accuracy. This shows that the encoder is reliable for the analysis but in order to get higher resolution thus lesser error rate a more fine system needs to be implemented.

Above is the circuit that is used for the rotary encoder. It consists of an infrared transmitter and also a infrared detector. The circuit will detect the black an white strips that has been place on the wheels of the robot. As the nature of the infrared in which only white or reflective surface will be detected by the detector. This is how the wheel rotation is converted into a stream of pulse by the circuit.

#### 4.4 INFRARED SENSOR

Infrared detector use in this circuit is placed in front tires of the robot. This sensor will be able to tell which side of the robot is having the obstacle. From the output of the sensors, certain maneuvers can be taken so that the robot is able to avoid it.

The infrared use is modulated at 38 kHz band frequency. This is done as the detector only detects a certain band of frequency and filter out the rest. This is done as to ensure the reliability of the sensor from detecting ambient infrared. In order to get the desired frequency modulation a 555 timer is used to generate the square wave needed. The circuit is shown as below.

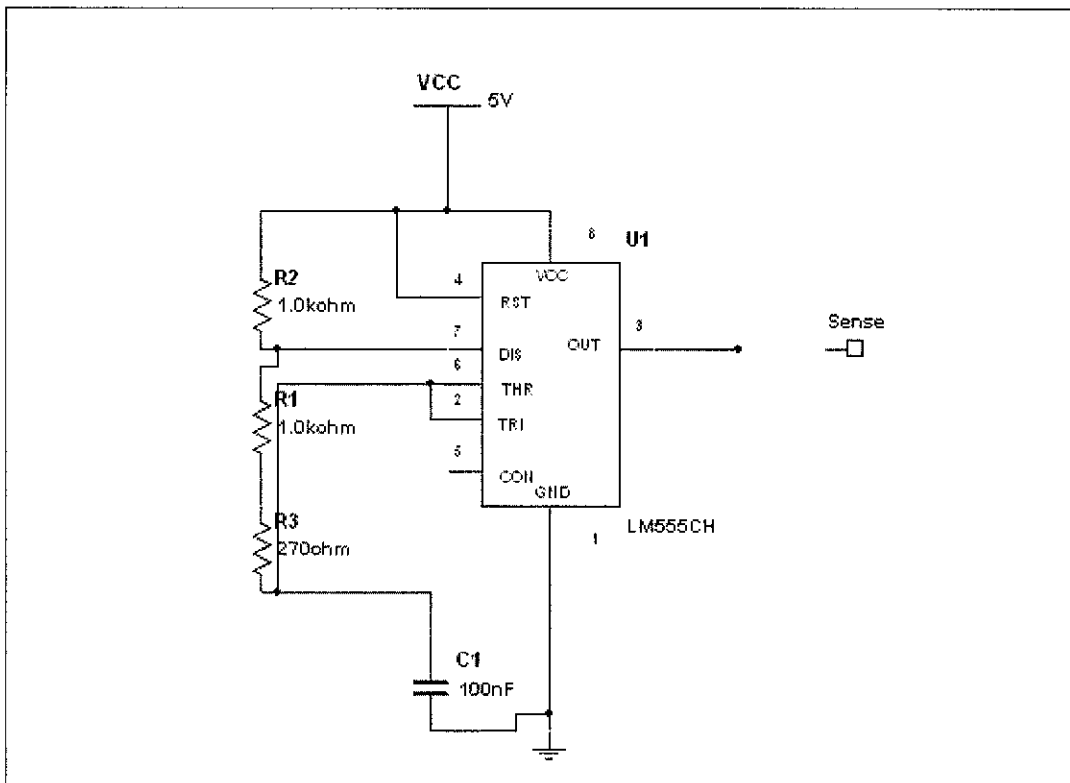


Figure 4-8 Infrared TX oscillator

In order to light up multiple infrared transmitter diodes, the output of the oscillator is then connected to a set of NPN transistor so that multiple infrared diodes can be light up at higher voltage than the one supplied by the output of the oscillator, by doing these 4 infrared diodes is used with 2 on each side.

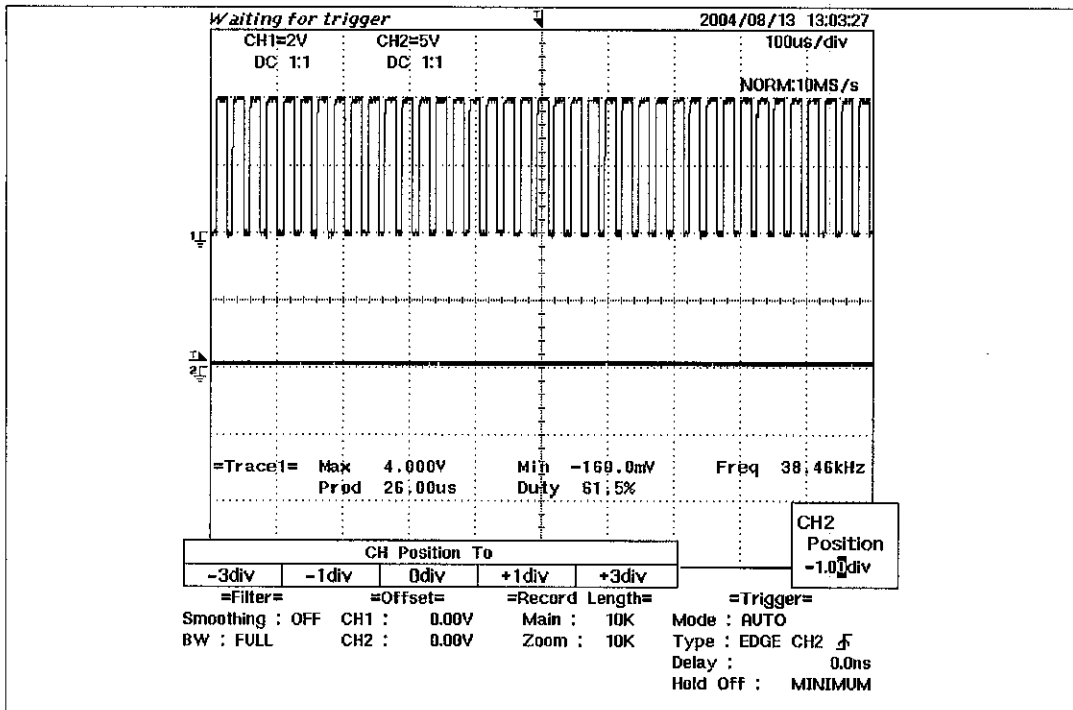


Some calculation can be done in order to get the desired oscillator frequency. Referring to the above circuit, the value of the capacitor and resistor can be estimate using the formula below.

$$f_c = \frac{1.44}{(R1 + R3)C}$$

**Equation 6 Oscillation frequency**

The value of capacitor is more limited that the value for the resistor. So in order to ease on the calculation and practicality, the C value is chosen to be 100 nf. By doing this the value of R1 and R2 has been chosen to be 1 k Ohm and R3 to be 270 ohm. Even though by calculation this value will give the needed modulation frequency, when the circuit was built on to the Vero board the frequency has change to 40 kHz. Fortunately the detector has a certain band pass value which it can detect.



**Figure 4-9 Modulated Infrared Signals**

The figure above shows the modulated signal that is being supplied to the infrared transmitter. The above signal is the signal that is being modulated at 38 KHz, while the bottom signal is the output from the detector. When the detector detects the bounce infrared signal the circuit will produce a low logic signal.

The drawback of this sensor is that it is not able to sense the obstacle if the obstacle is black in color. Most of the time the sensor is only reliable for a very short distance range. This range has been measure at 10 to 17 cm from the object. Another drawback of the sensor is that it is not as sensitive as ultrasonic; this sensor only detects some moving obstacles which bounce the infrared signal.

#### **4.5 ULTRASONIC SENSOR**

Ultrasonic sensor used comes from a DIY kit. The KIT was used as it is more reliable and due to the time limit which has come short. The sensor is reliable to detect an object at a minimum distance of half a meter a way from the robot. Once triggered, the circuit will latch the relay to create a high to low transition which is then detected by the microcontroller. The original circuit needs the circuit to be powered from a 12 volt dc supply. This is not suitable for the robot as the maximum power carried by the robot is already 12 volt. After some test, the circuit is still reliable if powered at 9v dc supply voltage. The sensor is sensitive enough that it will be triggered most of the time. In order to ensure that only the “real” obstacle is detected the microprocessor will enable or disable the output from the ultrasonic. This has prove to be practical than turning on and off of the ultrasonic circuit.

## 4.6 LINE FOLLOWER

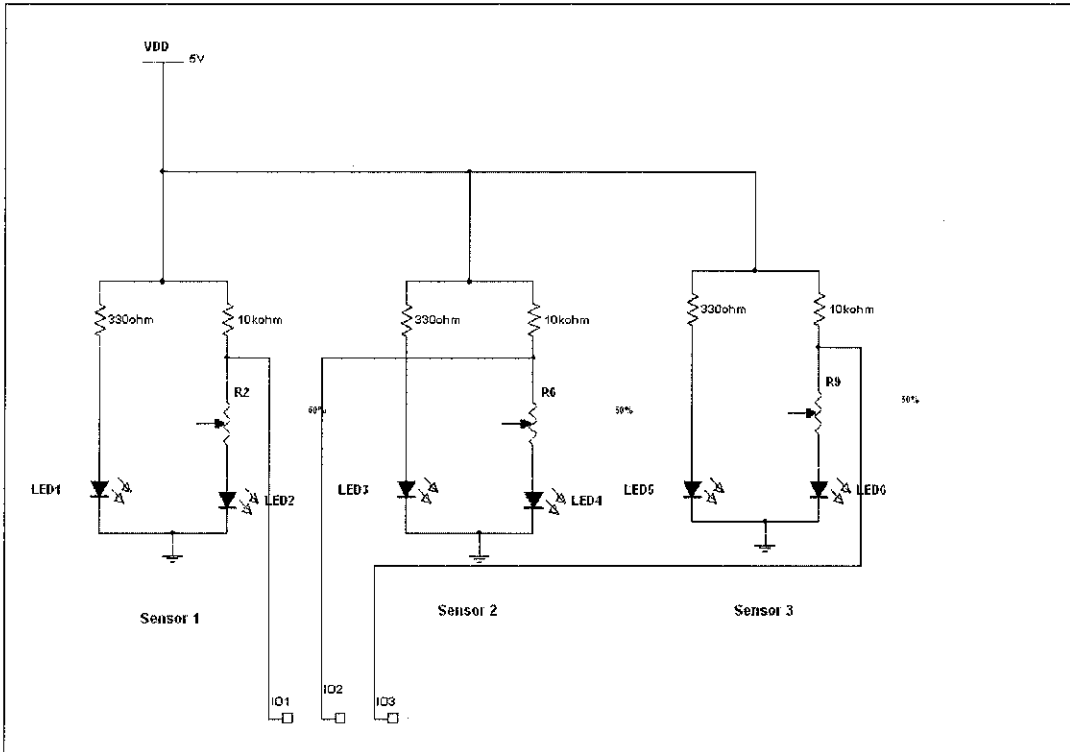


Figure 4-10 Line follower sensor

The line follower sensor consists of 3 set of infrared transmitter and detector connected in parallel to each other. The outputs of the sensor are connected to the microcontroller for processing.

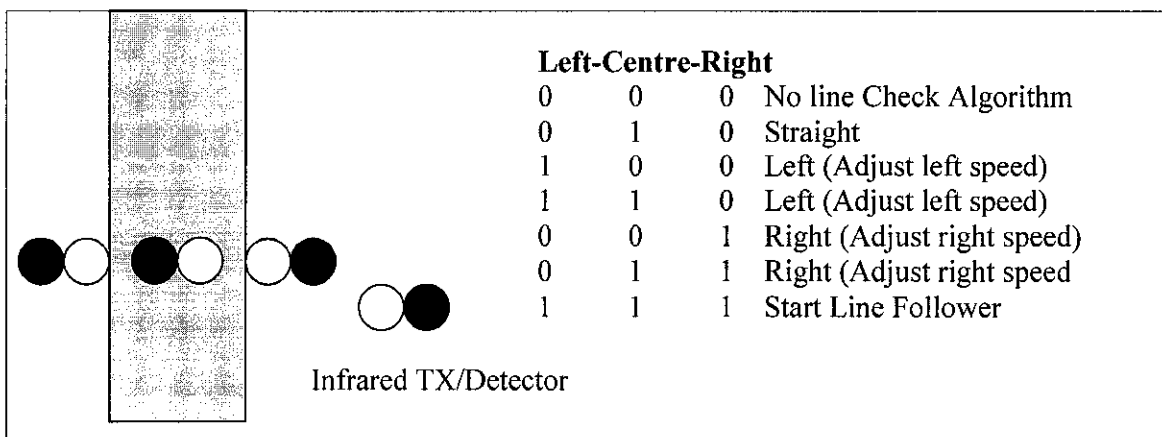


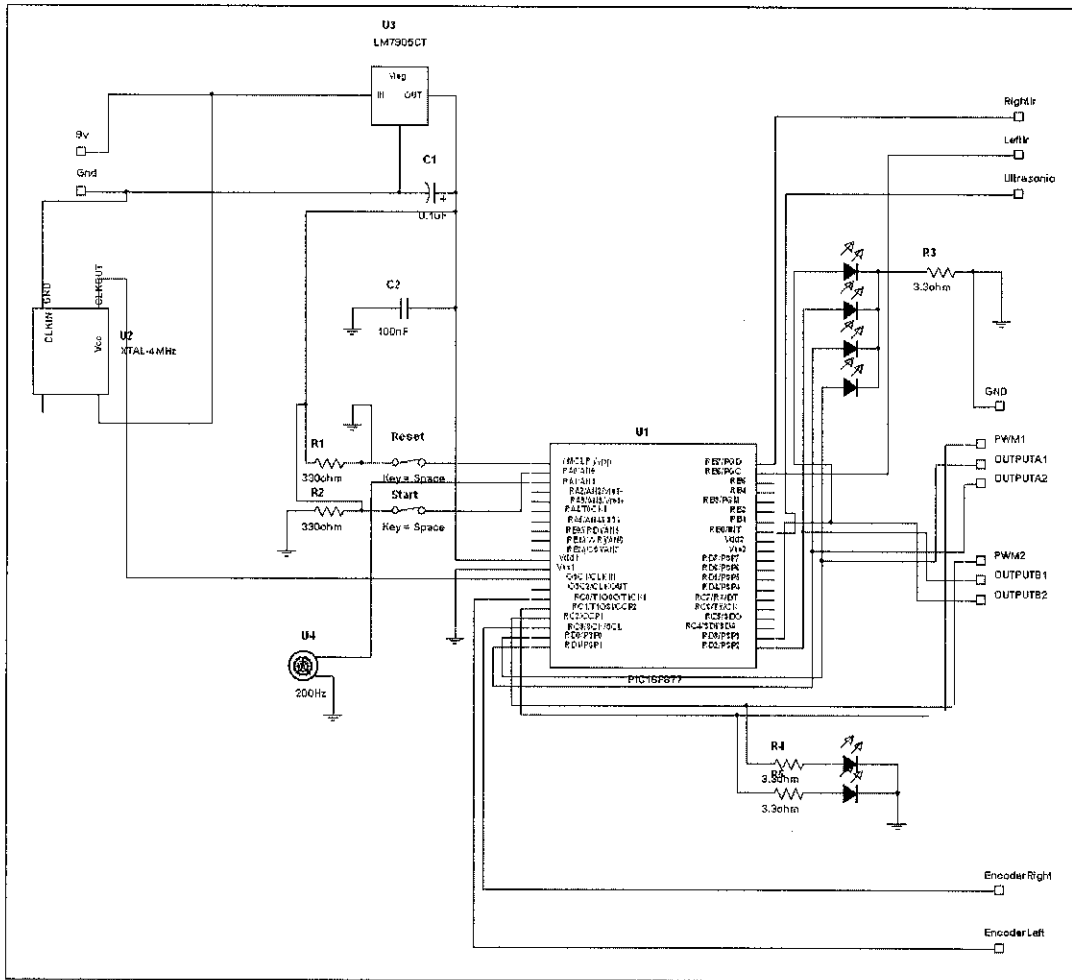
Figure 4-11 Line Follower Algorithm

The line that is placed on the floor needs to be on different color than the floor. In this case the line is black in color while the floor will have a lighter color. The sensor is used to guide the robot so that the error rate can be further reduces. This is not the solution for the error problem that is being faced but a supplementary sensor to help the robot to move in better line.

#### **4.7 MICROPROCESSORS**

There are several version of PIC. It is a programmable microcontroller that is produce by Microchip that can be use for many applications. The famous version is the P16F84 and P16F84a. For the project another high end PIC will be use. PIC 16F877 will be use as the main microcontroller due to some function that it is capable of. One is the Pulse Width Modulation (PWM) output pin which can be use for controller the motor drive circuit. Using the PWM signal the speed of the motor can be control. This will also helps to reduce the power consumption due to rapid switching. Another reason why this version is being chosen is due to the large number of digital to analog converter. These pins are suitable to be used with the ultrasonic transducers as the analog converter can be use to define distance and not only "1" or "0". This will increase the range of the output that the microcontroller can read from. Another important aspect of using PIC16F877 is due to the large area of built in ROM and RAM which is needed in order to program the algorithm to the robot.

The microcontroller can be program either by using assembly language or a high level compiler in C language. Programming in assembly language will make the code more optimize in term of memory managements but as the codes gets into complex loop and subroutine, keeping track of the codes will be difficult. High level of programming skills and experience will be needed if the codes are in assembly language. Another simpler method and more manageable is to use the C language compiler. What it does is to convert from normal C language codes into the assembly language before exporting the codes into the microcontroller memory. Most of the codes to set up the bits and mode of the microcontroller will be taken care by the built in function in the C compiler. The algorithm that is being implemented is much easier as the coding is more readable.



**Figure 4-12 Microcontroller Connections**

In hardware implementation the microcontroller needs a regulated 5V voltage supply which is being regulated using LM7805 voltage regulator. The input voltage is from a separate 9V battery. The microcontroller is capable to run at 20 MHz clock but a 4 MHz crystal clock is sufficient to the processing of the codes. Two normally open switches are used to control the operation of the microcontroller. The first one is used to give a low signal to Microcontroller clear pin which will restart the whole process of the controller if pressed. The next button will be used to signal a start sequence which will start the operation of the robot.

The output pins from the microcontroller will be fed to the H Bridge circuit, some LEDs and also a buzzer. The 6 pins that will be fed to the H bridge consist of 4 pins which will give the combination for the direction and 2 pins which will gives the PWM signal to the H bridge. The PWM signal is used to control the speed by varying

the ON time and OFF time of the H bridge circuit. This will result in average power output to the motor depending on the duty ratio selected or generated by the PWM pins.

**Table 4-3 Binary states for movements**

<b>Direction</b>	<b>Bit3</b>	<b>Bit2</b>	<b>Bit1</b>	<b>Bit0</b>
<b>Forward</b>	<b>1</b>	<b>0</b>	<b>1</b>	<b>0</b>
<b>Reverse</b>	<b>0</b>	<b>1</b>	<b>0</b>	<b>1</b>
<b>Right Turn</b>	<b>0</b>	<b>1</b>	<b>1</b>	<b>0</b>
<b>Left Turn</b>	<b>1</b>	<b>0</b>	<b>0</b>	<b>1</b>
<b>Stall</b>	<b>0</b>	<b>0</b>	<b>0</b>	<b>0</b>

**Table 4-4 Microcontroller Board Specification**

<b>Clock</b>	4 MHz
<b>Memory Size</b>	256 x 8bit of memory
<b>Pin0</b>	Master Reset
<b>Pin1</b>	Start or Run
<b>Pin17</b>	PWM1
<b>Pin16</b>	PWM2
<b>Pin22</b>	Bit4
<b>Pin21</b>	Bit3
<b>Pin20</b>	Bit1
<b>Pin19</b>	Bit0
<b>Pin 18</b>	Encoder left
<b>Pin 15</b>	Encoder right
<b>Pin 33</b>	Ultrasonic
<b>Pin 39</b>	Infrared Left
<b>Pin 40</b>	Infrared Right

## 4.8 ALGORITHM

### 4.8.1 Coordinate to path conversion

The way that the robot moves depends on the algorithm that is being used inside the microcontroller. In order to differentiate between the negative and positive axis of the robot, a flag is used inside the microcontroller to define the positive or negative value of the axis. A certain sequence of movement is set inside the "if loop" statement in order the robot to move the desired location. The path of the robot is predefined using a Cartesian coordinate system with the front of the robot as referenced.

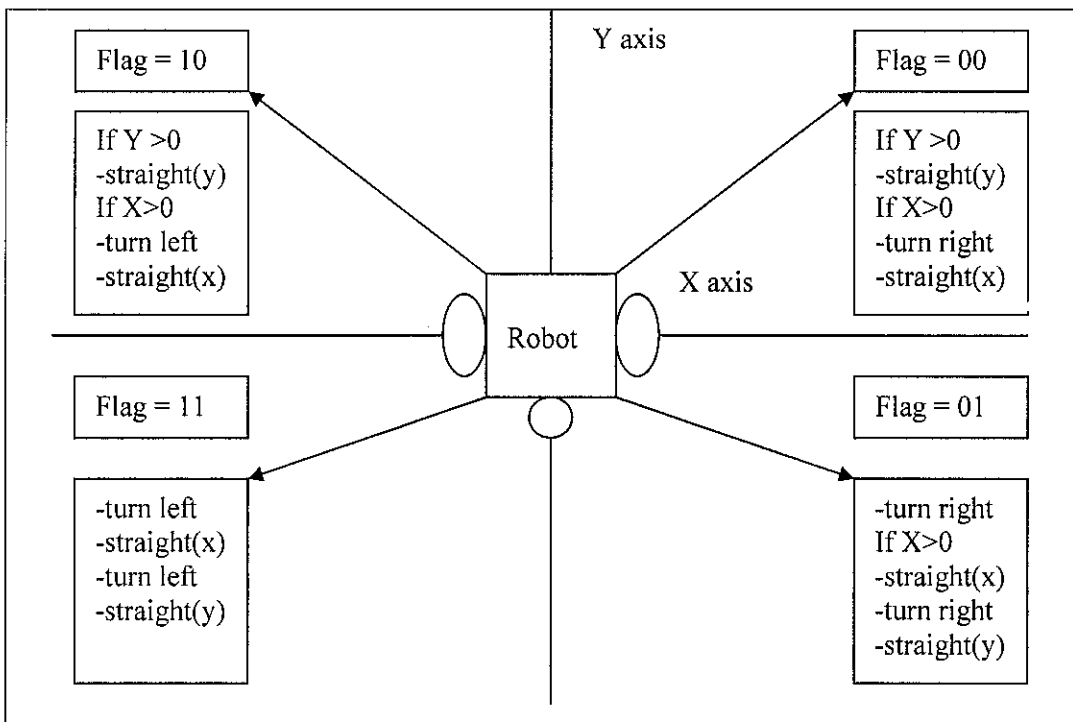


Figure 4-13 Axis algorithm

#### 4.8.2 Dead reckoning

In the robot a dead reckoning technique is used. The rotary encoder will be the input to the system. By having this accurate measurement system can be implemented thus giving the robot the capability to move point to point at higher accuracy.

The encoders consist of 16 black and white strips in other word for a full revolution 16 pulse will be detected.

$$pulse = \frac{1}{16}(2\pi r)$$

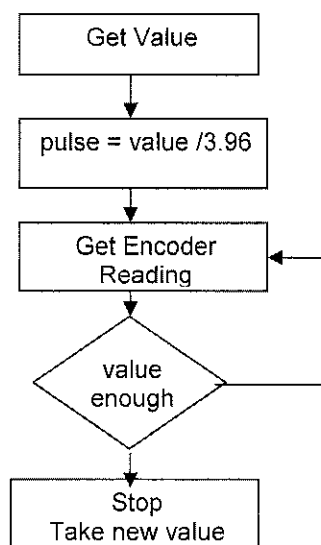
**Equation 7 No of pulse conversion**

*r= radius of the wheels*

By simple mathematics we can say that for every 1 meter there will be roughly 50 pulses. Thus in real time the coordinate that is being program into the robot in centimeters can be converted into number of pulse. Now the robot will move as long as the count of pulse has not reach the needed number of pulse.

In terms of accuracy, the accuracy is about 1 or 2 pulse or from 2 to 4 cm each time.

This is due to the floating point operation done by the microcontroller which will tend to converge the value to a certain fix value thus giving some error.



**Figure 4-14 Dead reckoning algorithm**



### 4.8.3 Error correction codes

In order to reduce the error as the robot moves forward an error correction codes has been implemented. At first a PID control was the intended choice but due to the time and knowledge limit, the error correction algorithm was derived from test conducted on the robot. The robot tends to move toward the left if the same PWM signal was given on both motor. Further analysis shows that one of the motor is actually driving reverse when it is moving forward. Thus due the gearing was intended to move in one direction the speed on both motor was different. By trial and error the best PWM value to get the robot to move straight was obtained.

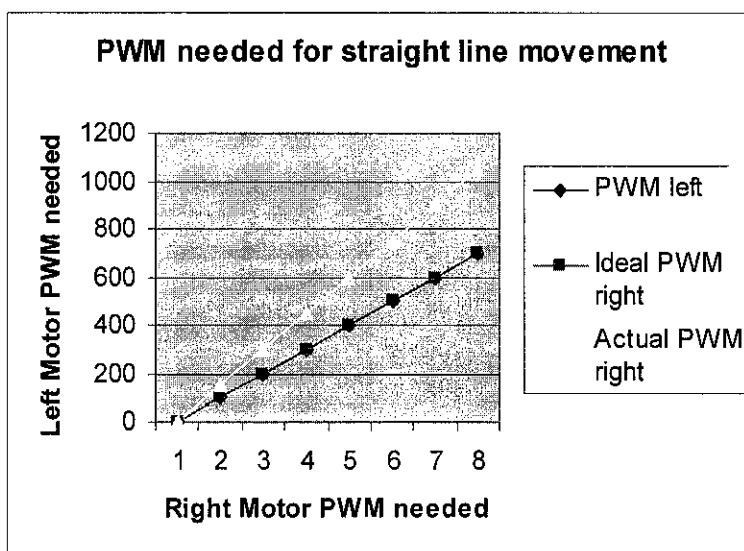
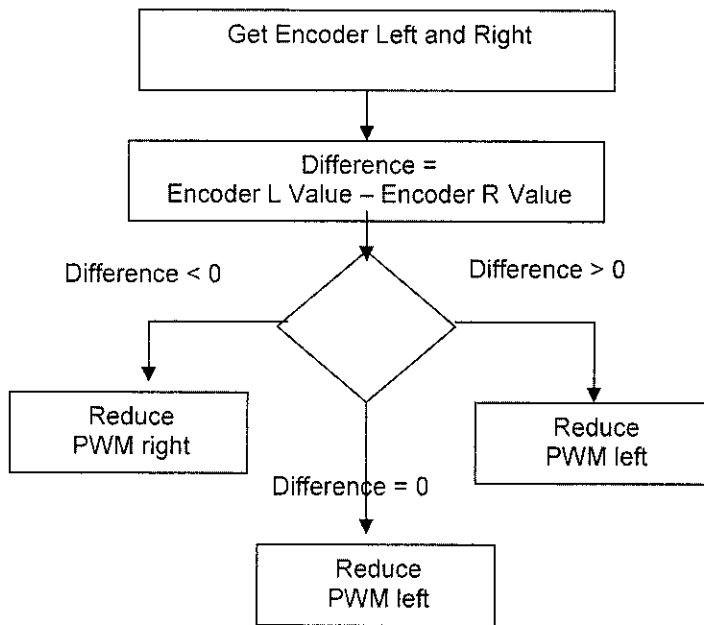


Figure 4-15 PWM needed for a straight line

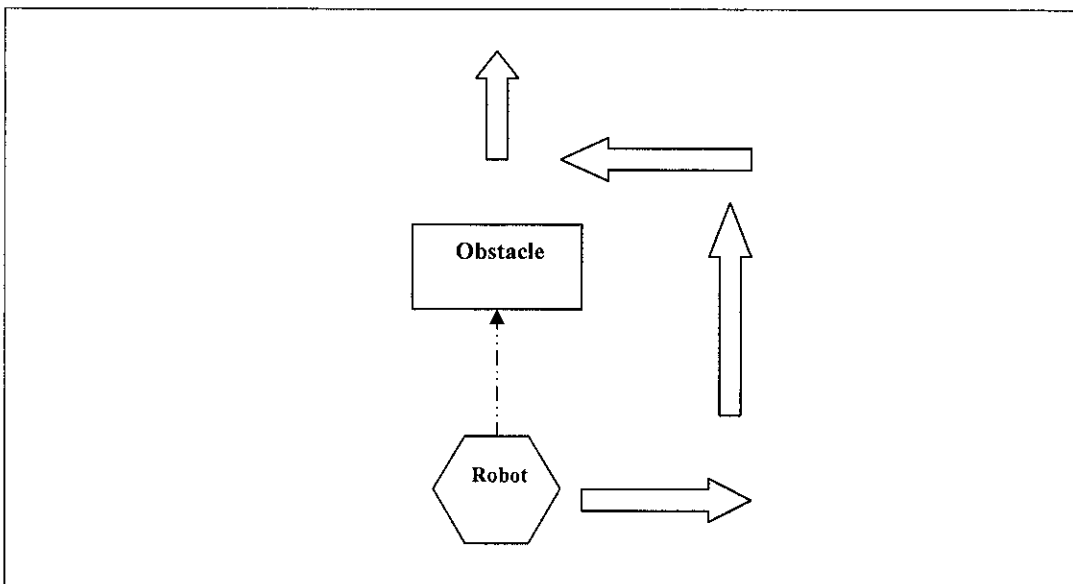
By analysis it can be seen that it is a not proportional relation between the PWM of the right motor and the left motor. From analysis the ratio of left PWM to right PWM is 1.5. Again by trial and error the algorithm is created to correct the robot movement. This is base on the actual pulse that has been counted by the encoder, if one side of the encoder counts more than the other the error correction will try to reduce the PWM on the side which has the bigger count. This has led to a better improvement but still some small error can be seen.



**Figure 4-16 Error correction codes algorithm**

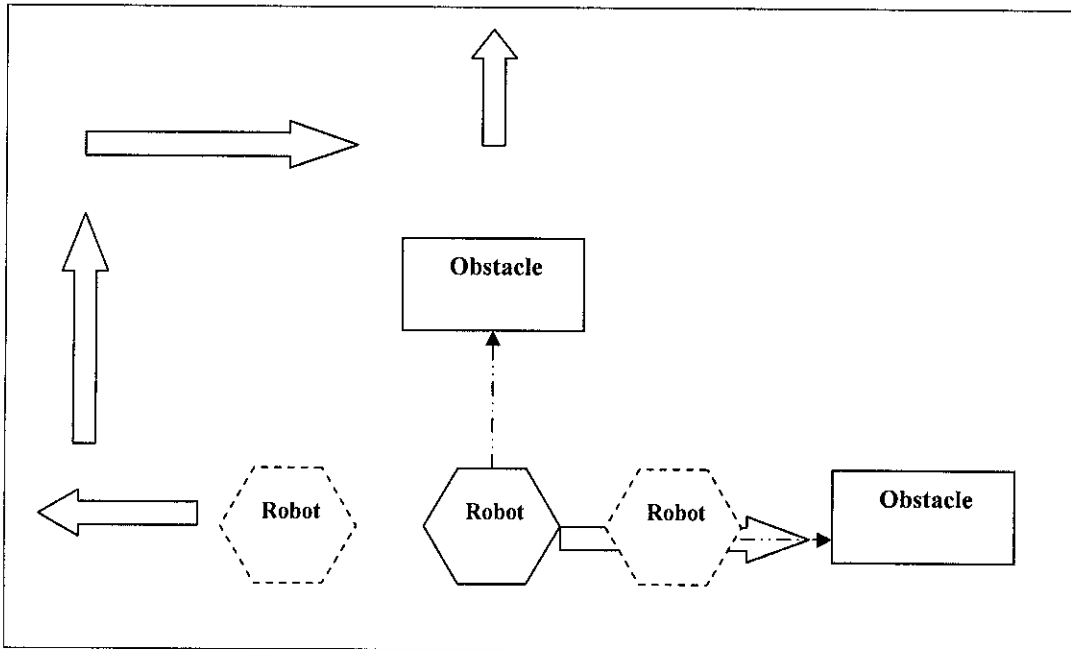
#### 4.8.4 Obstacle Avoidance

The mobile robot is equip with some intelligent to avoid the obstacle on its way through the predetermined path. In this project one drawback of the algorithm is that it does not subtract the distance travel for obstacle avoidance but it will safe the last distance travel and resume on the remainder as the obstacle avoidance routine ends.



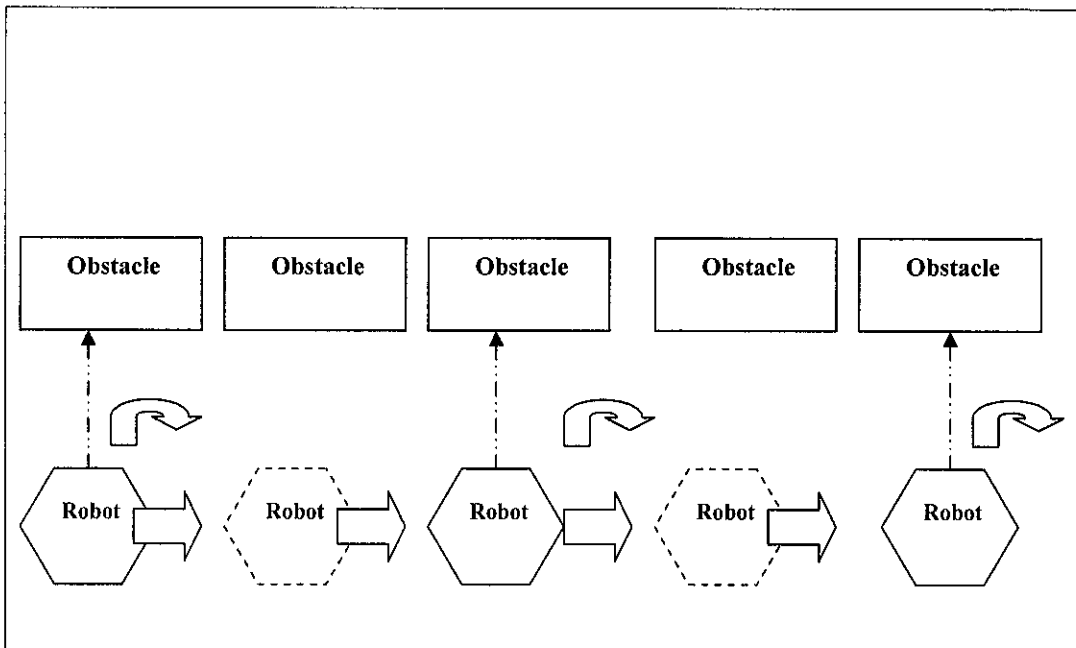
**Figure 4-17 Case 1: Simple Obstacle (Same algorithm on left or right movement)**

This is the simplest case the robot may find. What the robot will do is to turn right and then move forward about 2 meters then turning left and then moving forward again for a distance before turning left and again moving straight at a distance and finally make a right turn and ending the routine. The microcontroller will then resume from the last distance it traveled. This algorithm will work the same on the reverse side for example when the first step is to move to the left instead of right.



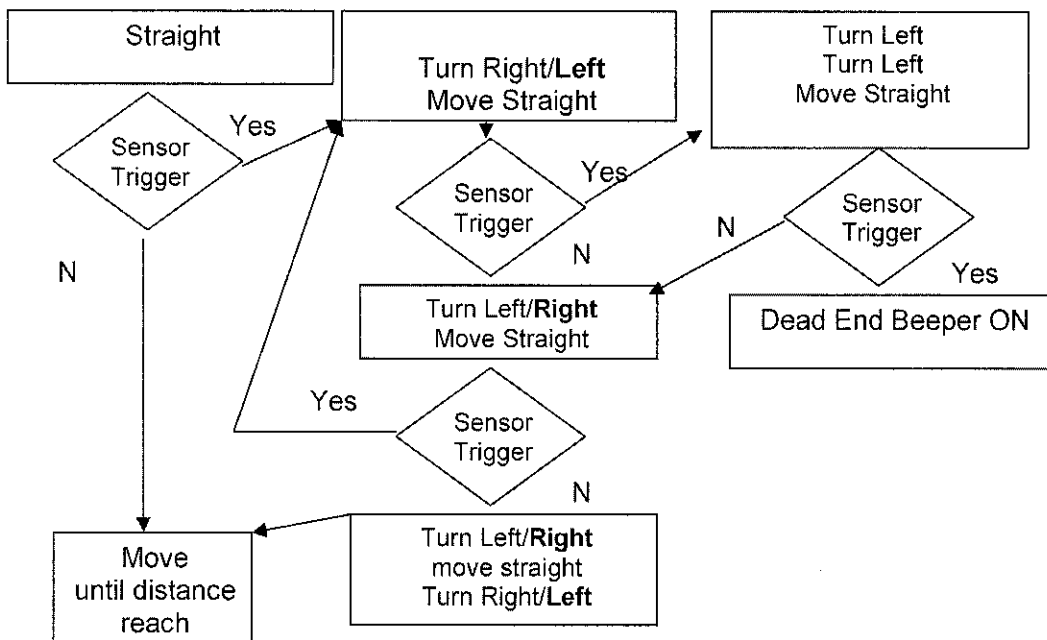
**Figure 4-18 Case 2: Obstacle on the left or right (Same algorithm on left or right movement)**

If the robot detects obstacle while on its way to correct the first case error, the robot will then turn back to the original route and try the left side of the obstacle instead. If the robot encounters the same obstacle then the robot will stop and sound a beeper sequence indicating a dead end.



**Figure 4-19 Case 3: Wall obstacle (Same algorithm on left or right movement)**

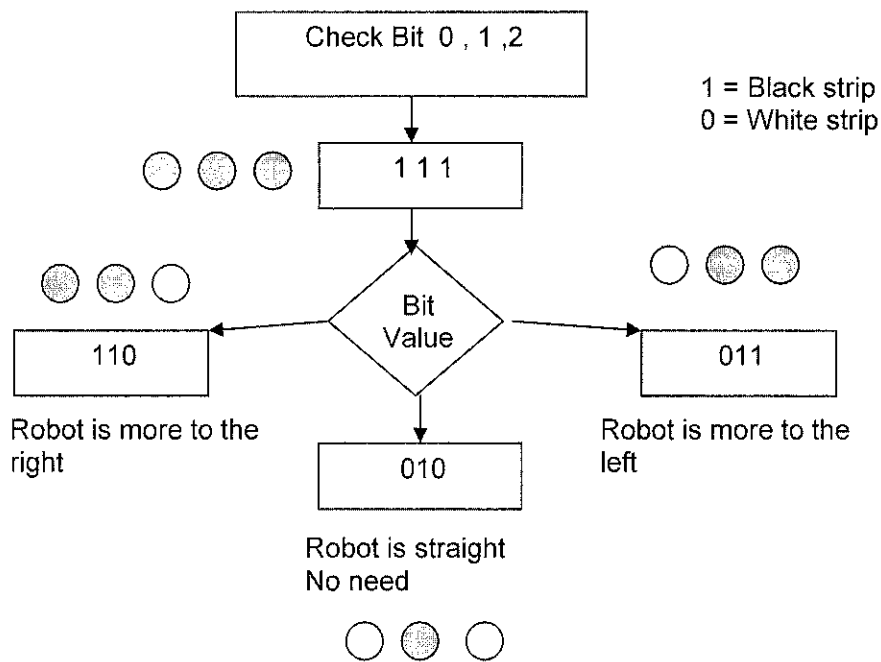
Due to the nature of the algorithm, the robot has wall following behavior. When the robot encounter a wall like obstacle, the robot will move to the right and continue moving before trying again to pass the wall. Again the wall is detected the robot will follow the same routine. If it encounter obstacle while moving to the right, the robot will try to move toward the left. Again using the same algorithm the robot will move and check until it found a way out. If again the robot encounter obstacle while moving to the left, the robot will sound the beeper indicating a dead end.



**Figure 4-20 Obstacle avoidance algorithm**

#### 4.8.5 Line Follower

The robot will enter line follower mode when it detects a straight black line. This will automatically put the robot into the line follower mode unless the overwrite switch is being activated. When the robot is in the line follower mode the algorithm will try to ensure that only the center sensor is being detected or triggered. This is how the robot will try to follow the black lines. If on side of the 3 sensor is activated with the center sensor, the algorithm has been written so that the side which is crossing the black line will be getting a slower value of PWM. This is how the robot maintains the movement on the black line.



**Figure 4-21 Line follower algorithm**

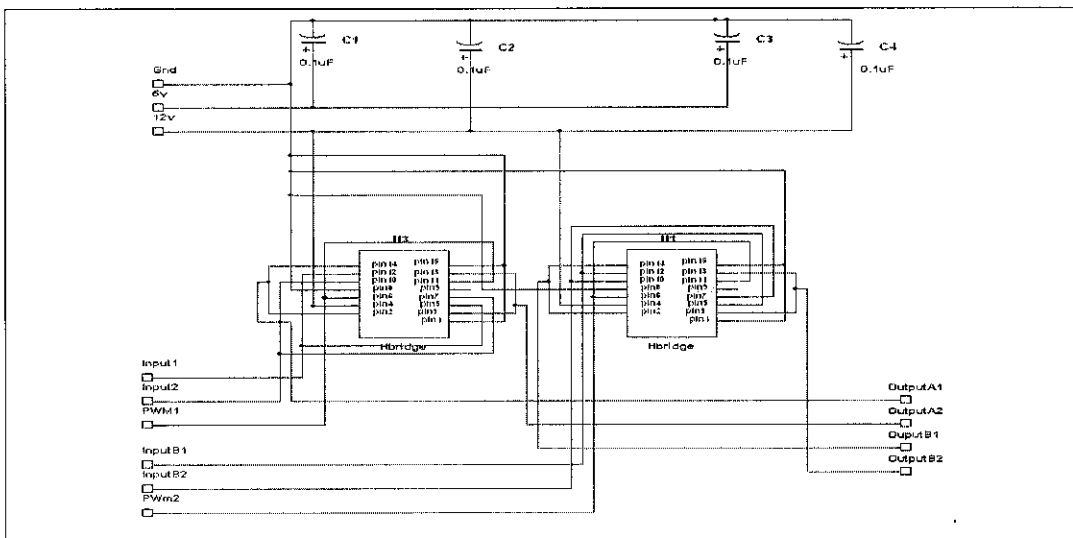
#### 4.9 DRIVE CIRCUIT

In order to supplied the needed power and be able to control the direction of the rotation of the motor thus the robot, a H bridge board has been design based on the application notes given by the manufacturer. The H Bridge was design to be able to allow maximum of 4A of current to be supplied to the motors. The amount of current varies depending on the load of the motor. For example the current needed to move the robot would be high at start up but decrease slightly at when running. The more the weight that the motor need to carry the higher the current will be needed to maintain the same speed. This has been a challenge as the maximum rating for the design will be around 4A. If the current that is being drawn is more than 4A the current will gives smoke signals.

The H bridge board consists of two L298 dual H bridge chip. In order to get the high current setup, some modification has been put in place. The digital voltage of the chip is supplied by the voltage regulation board which regulates 5v from the batteries. The H bridge board will have 2 levels of voltage, one at 5 volt and another one at 12 or 13 volt. The high voltage will be supplied to the motor through the circuit. In order to move the robot at start up 1.68A of current is needed at each motor.

The voltage level at this current would be around 6V which is half of the voltage supplied to the board. The board voltage supply of 13 volt is actually shared for both motor to ensure both motor gets the same amount of power assuming no losses and no defects to the components used. From the experiment, if the board uses a separate voltage source each at 6.8 volt the motor will move at a really low speed and torque, when measured the current is at 1.30A.

Further analysis shows that it is due to the internal design of L298 chip which will not allow higher current if the supplied voltage is less than a certain level. The solution was to connect the battery supplied in parallel and supply higher voltage thus allowing higher current to the motor. Now the problem arises is that the L298 chip heats up quickly and thus making its life shorter. Fortunately the H bridge board uses PWM signal to control the output power to the motor. Using a low duty cycle around 40% to 45% the motor speed is at a desirable range and the heat generated is less. Each drive forward or reverse the circuit will supply 4volt to the motor.



**Figure 4-22 Double 4 A H-Bridge**

#### 4.10 RS232 COMMUNICATION

The final part of the robot is a communication module and Robotic Management System (RMS) software has been developed with the help of Mr Azizan Hashim specifically for this robot. The RS232 chip is used to create a communication link between the PIC microcontroller and also the computer. The RMS provides the graphical user interface between the mobile robot and the computer. The RMS has been developed so that the user is able to plot the path of the robot to move. The software then converts the normal coordinate system to that the one the robot is able to understand.

The converted coordinate can be then uploaded to the RAM of the PIC. As long as the rest microcontroller is not in reset mode, the coordinate will be available in the microcontroller. If the user wants to change the coordinate, the robot needs to be reconnected back to the computer and then put into reset mode. Then the new coordinate can be reloaded to the RAM of the microcontroller.

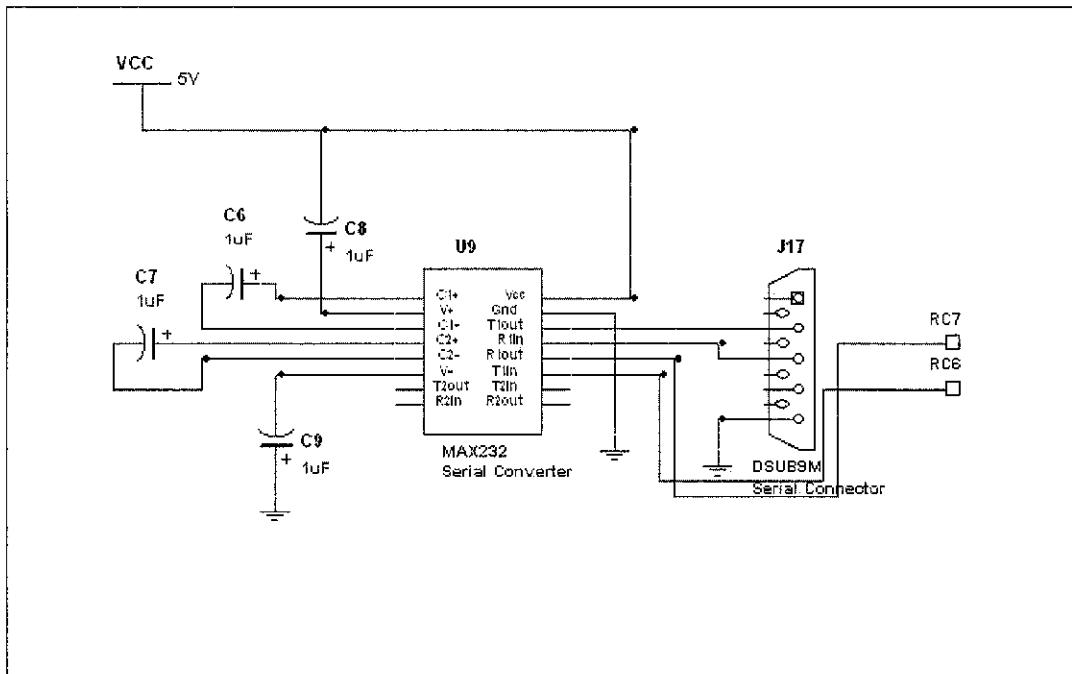


Figure 4-23 RS232 Communication Circuit Module



#### 4.11 SERVO CONTROLLER

The servo controller has been built using PIC16F84A. The sole purpose of this micron roller is to generate the required PWM pulse in order for the servo to work. The basic operation is simple to be implement in the microcontroller with caution in the programming sequence as the input to the servo is very sensitive to the pulse width generated.

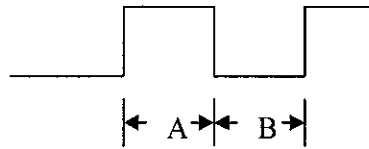


Figure 4-24 Width of PWM for serve @ Caster

Table 4-5 Width of PWM for serve @ Caster

A width (ms)	B Width (us)	Position of servo ( Caster wheels)	Input from PIC 16F877
17	225	180 degrees	010
17	700	45 degrees right	001
17	1125	Straight (0 degrees)	000
17	1300	45 degrees left	011
0	0	Free wheel	100

The servo controller will receive signal from the main microcontroller as referred above. The servo controller will then translate the movement and prepare the needed pulse so that the caster wheels attached to the servo is moved to the desired position.

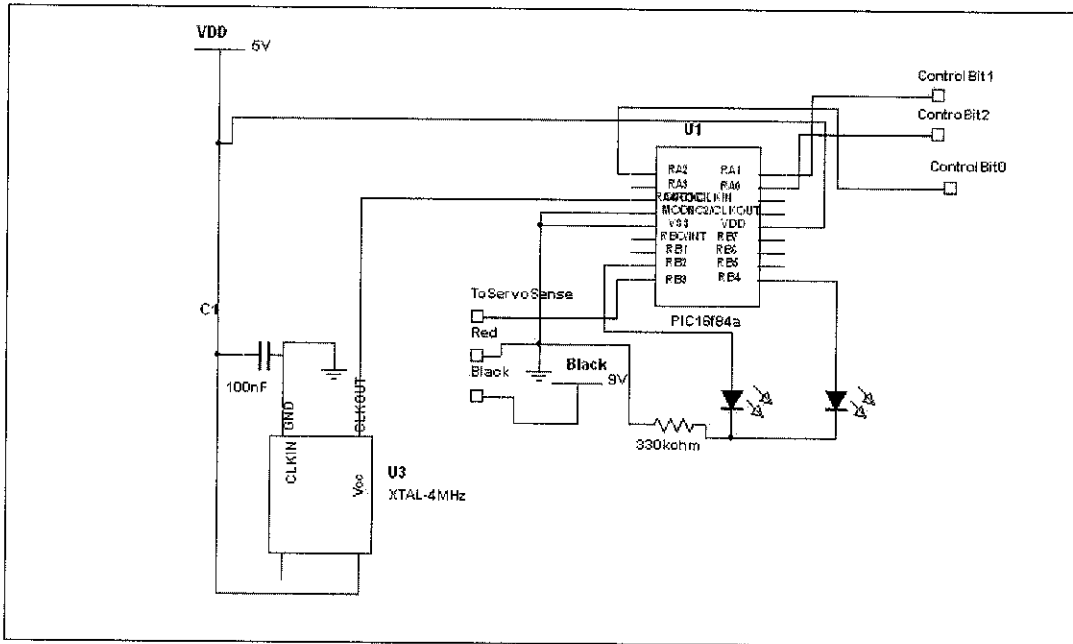


Figure 4-25 Servo Controller

4.12 AS A WHOLE

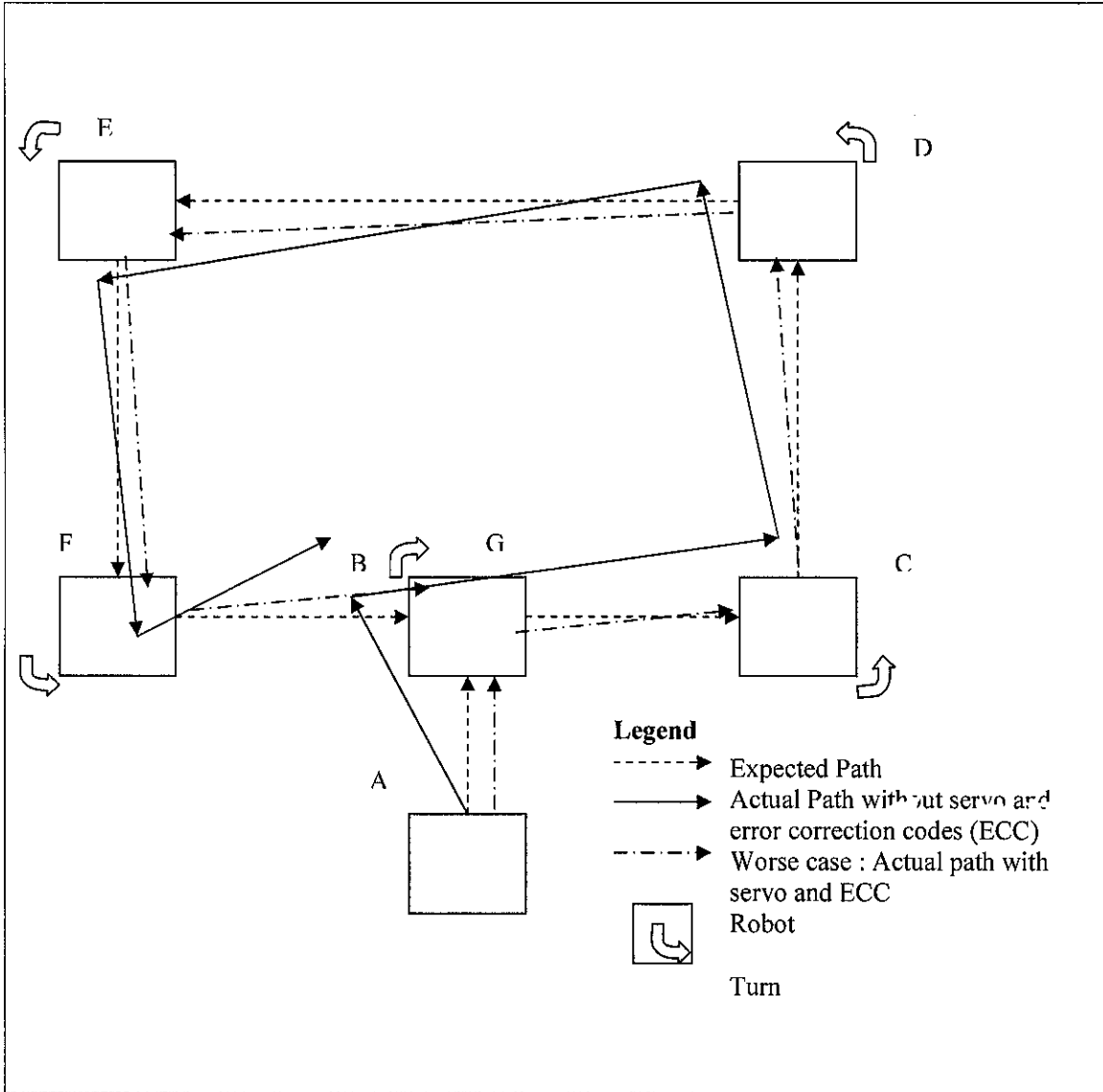


Figure 4-26 Path of robot

The robot should move from point A to point G as depicted by the path above. This is done by giving a coordinate to the microcontroller. The microcontroller will then convert this coordinate into a series of steps in order to achieve this movement. The robot is capable to measure the distance that it's has travel. When the robot reaches each point it will stop for a while and the beeper on the microcontroller board will beep to indicate the end of a step. The robot will then turn to the correct direction before moving on to the next point. The algorithm should move the robot in a path that will create a box.

After the first run a problem is clear. The robot was having some problem in moving straight. At first the reason might lay in the structure or the way the gearbox and the tire is being mounted, but after some observation and evaluation the culprit was identified. The error was coming from the caster tires. As the surface of the test area is uneven the caster tire always get interrupted thus position at the wrong angle each time. The small difference in the caster angle has cause the robot to not be able to move straight. A test has been conducted in which the caster tire has been temporarily immobilized by masking tape it. The robot was found to be able to move straight.

The solution for this problem was to put a servo controlled caster. This design fix has shown a large improvement on the movement. This was not enough the robot still deviate if it moves further so error correction codes was implemented to fix the error. This also has reduced the error more. As the error was coming from the structure and also the behavior of the motor itself an optional method is implement which is to include a line follower to supplement the error correction codes.

For obstacle avoidance the robot is using 2 types of sensors. The ultrasonic will detect any obstacle that is directly in front of the robot while the infrared sensor will detect any obstacle in front of the tires. The two sensors are connected to one microcontroller so that certain algorithm will take place as the sensor detects any obstacles.

The robot will also follow black lines if the switch mode is turn on and if the line follower sensors detect a starting black strip on the floor. The robot is capable of following the line without much problem.

## **CHAPTER 5: RECOMMENDATION**

There are a lot of aspects on this design project that can be improved in order to get a better outcome. Here are some of the recommendations that can be improved in the future.

### **5.1 STRUCTURE DESIGN**

At the moment the structure was not designed properly with mechanical analysis such as static and dynamic. In the future a better design can be achieved if the project can be handled with the help from the mechanical department which can build a better light weight and stronger chassis than the one that is currently used for the project. Attachment with the mechanical department will ensure that the electrical student can concentrate on the electrical aspect of the robot and will not have problems with the structure-related problem.

### **5.2 DATA COMMUNICATION**

Due to time constraints the robot does not have any capability to communicate with any desktop in real time. Actually there are a lot of pre-fabricated modules that can interface with the robot microcontroller which can enable the robot to communicate or receive instructions in real-time basis. By having a real-time feedback from the desktop, the robot should be able to have advanced processing capability and maybe live video streaming from and on board camera.

### **5.3 ACCURACY**

In order to achieve a higher level of accuracy and control few aspects need to be improved in the future. First the components used for the project need to be the one designed for robotics, in other words all the electronics used should have a better quality than the normal components. Components such as motor and drive circuit should be bought or first designed properly from robotics shops which are available in the market. The use of normal components and modifying them to suit the robotics needs is not a good

idea if the accuracy level needed is high. Sensors and encoder should also be reliable and purchase or design with higher accuracy. This will help to reduce errors thus giving a good output in the end.

#### **5.4 ACTUATORS**

Grippers, hands are actuators that might be included in the further enhancement of the robot. This additional component will allow the robot to pick things up and put it at another location. A good actuator with the help of sensors should be use together or else the actuator will be useless to detect the item needed to be picked. With this addition the robot intelligent can be further enhanced.

## CHAPTER 6: CONCLUSION

The structure of the robot has been built from heavy but sturdy material. This has led to an unexpected problem in the weight of the robot thus the power needed to move a heavy robot. For this design project, the student assumed that the total weight of the robot which is 8 Kg to be the total load that the robot is capable of carrying over a distance. Even at this load the robot is capable of doing its objective.

The H-Bridge design using L298 is good enough for this project as long as the maximum weight does not exceed 8Kg in total. The problem of high current drawn will arise as the total current demand increases. The heating effect of the L298 chip is the main problem. At certain mode the circuit will kick into the safe mode thus making the power to the motor to be very low.

The ultrasonic is the main sensor used for this mobile robot. The range of the sensor is very big but the problem lies in the minimum distance the robot can sense. This is a problem as even at the lowest setting the distance is still far from the robot, making maneuvering in very small areas such as in the lab quite challenging.

The infrared sensor works nicely but not as expected. The sensors are able to detect obstacles as long as the surface reflects the infrared signal and also flat and sometimes need the obstacle to be moving. This has made the sensor less reliable than the ultrasonic sensor. In this design all the sensors have been put into OR gate orientation to increase the reliability and coverage of the sensors.

Intelligence of the robot lies in the complex and lengthy algorithm used in the microcontroller. It is not easy to implement such an algorithm if there is no good support from the programming side. With the help of an experienced programmer this problem can be solved in a short period of time. With a more intelligent robot, the longer the codes will be. In this project the bottleneck was at the microcontroller itself. Even using the best microcontroller available in the lab, the RAM and ROM were already at their fullest.

The software support of this mobile project is actually exceeding the objectives of the project. But in order to increase the functionality, the Robotic Management System has been developed with the help of Mr Azizan Hashim. This software operates not just to be used to upload the data into the microcontroller but also use in debugging the mobile robot. The software was able to give an “inside look” into the microcontroller.

Error in movement is the biggest problem of all. The main reason of this problem was from the structure make up. Unforeseen, the simplest part can be the part that will create the most problem towards the end. Even with the help of error correction and line follower sensor, the problem is still there. The worst part is that the reliability of the solution will depend on the condition of the structure during runtime. In the future, it is best that the structure is properly design and any loose end needs to be fixed properly.

Overall, the mobile robot has achieved the objective very well even giving valuable data for future projects on this area. The mobile robot manage to move from one point to the next point with minimal error and also capable to avoid obstacle on its way. The upload feature of this mobile robot is the first in this university and can be develop further in the future.

There are many obstacles that need to be overcome before a stable robotic platform is available in house. This mobile robot project and all other projects before this will be the stepping stone for future engineers to develop new and better robotics. Implementing knowledge gain from theory will be different from the knowledge gain from the experience itself. Having all the basics for developing a good robot in house will give a strong future on robotics later on. After solving the obstacles, we will have a good mobile robot. With the current design and available components, the objective of the project has been achieved.



## REFERENCES

- [1] Build a remote control robot David R .Shircliff, McGraw Hill
- [2] Obstacles Avoidance in Multi-Robots ,Gill Zomaya
- [3] Battle Bots the official guide, Clarkson, Mc Graw Hill
- [4] Robot androids and animatronics ( Second Edition ), John Iovine Graw Hill
- [5] Build Your Own Robot1, Karl Lunt
- [6] Animatronics A Guide to Animated Holiday Displays, Edwin Wise ,Prompt Publication
- [7] Sensors for Mobile Robot ,Theory and Application, H.R Everett
- [8] Applied Robotics ,Edwin Wise, Prompt Publications
- [9] L298 H-bridge Datasheet
- [10] LM78XX Voltage Regulator Datasheet
- [11] PIC 16F877 Datasheet and application
- [12] PIC C, Instruction Manual.
- [13] <http://www.mstracey.btinternet.co.uk/pictutorial/picmain.htm>
- [14] <http://www.galileo.org/robotics/design.html>
- [15] <http://www.seattlerobotics.org/encoder/index.html>
- [16] <http://www.picant.com/robot/robot.html>

## APPENDIX A: MAIN SOURCE CODES

/\*

INFORMATION:

1. Program=Mobile Robot
2. Programmers= Shahrman  
Azizan
3. Latest Update= 30.11.2004

PROGRAMMER'S NOTE:

1. Distance Calculation= Success
2. Tyre Error Correction= Success
3. Object Avoidance= Success (kerek14b.c)
4. Data Uploading= Success (kerek14b.c)
5. Line Follower= Success

\*/

```
#include <16F877.h>
```

```
#use delay(clock=4000000)
```

```
#fuses XT,NOPROTECT,NOWDT,NOLVP,PUT
```

```
#byte port_b=0x06
```

```
#byte port_c=0x07
```

```
#byte port_d=0x08
```

```
#use rs232(baud=9600, xmit=PIN_C6,rcv=PIN_C7)
```

```
#reserve 0x110:0x11F
```

```
#reserve 0x190:0x19F
```

```
//Warning: if x=0,signx must be 0
```

```
//Warning: if y=0,signy must be 0
```

```
//PORT # = PIN # FUNCTION
```

```
// A = PIN 0 [2] Start Switch
```

```
// PIN 1 [3] Buzzer
```

```
// PIN 2 Override Line Follower
```

```
// PIN 3
```

```
// Pin 4 [6] !!!! warning only pull low
```

```
// PIN 5
```

```
// B = PIN 0 Ultrasonic Sensor
```

```
// PIN 1
```

```
// PIN 2
```

```
// PIN 3
```

```
// PIN 4
```

```
// PIN 5
```

```
// PIN 6 Ir left
```

```
// PIN 7 Ir right
```

```

// C = PIN 0 [15] Input A (right) encoder
//   PIN 1 [16] CCP2 PWM motor /C1
//   PIN 2 [17] CCP1 PWM motor /C2
//   PIN 3 [18] Input B (left) encoder /C3
//   PIN 4 [23]
//   PIN 5 [24]
//   PIN 6 [25] Tx
//   PIN 7 [26] Rx

// D = PIN 0 [19] Right Motor LSB
//   PIN 1 [20] Right Motor MSB
//   PIN 2 [21] Left Motor LSB
//   PIN 3 [22] Left Motor MSB
//   PIN 4 [27] Servo A0
//   PIN 5 [28] Servo A1
//   PIN 6 [29] Servo A2
//   PIN 7 [30] Reserved

//           E           = PIN 0   Line Sensor (Left)
//                   PIN 1   Line Sensor (Center)
//                   PIN 2   Line Sensor (Right)

//Drive Train=DCBA, DC=right wheel, BA=left wheel
//Drive Data: 00=No movement, 10=Forward, 01=Reverse

//Line sensor: Black=HIGH, White=LOW

//data_status
//0=No transmission, 1=Start byte accepted,transmission in progress, 2=Data error, 3=finished transmission
byte data_status=0;
byte data_count=1;
byte data_counter=0;

//*****UTK interrupt flow*****
byte flow_flag=0;
byte is_mark=0;
byte is_main_interrupted=0;
//*****

void straight(int distance_incm); //distance in cm units
void turn_right(void);
void turn_left(void);
void stop(void);

void initialize(void);
void pwm_right(long dutyratio);
void pwm_left(long dutyratio);
void interdelay(void);

```

```

void beeper_off(int maxcount);

void allinone(int signx,int x,int signy, int y);

byte buffer_rb,buffer_ext;

int i;
#define INTF_BIT = 0x0B.1

//Interrupt for serial on-received
#define int_rda
void serial_interrupt()
{
    int data_rcv;
    int data_id;
    int data_type_id;

    /*
    Note:
    data_counter=0                start byte
    data_counter=1                no of points
    data_counter=2 to n+1 points data
    data_counter=n+2                stop byte
    */

    data_rcv=getch();

    //If there is no transmission
    if (data_status==0)
    {
        //Check for start byte
        if (data_rcv==0xFF)
        {
            //Accept for next=no of points
            data_counter=1;
            data_count=0;
            //Transmission in progress
            data_status=1;
        }
    }
    else
    //If transmission has been started
    if (data_status==1)
    {
        //Extract for data count
        if (data_counter==1)
        {
            if (data_rcv>0)
            {

```

```

//data count
                                data_count=data_rcv;

//Clear RAM
                                for (i=0;i<=15;i++)
{
                                write_bank(2,i,0x00);
                                write_bank(3,i,0x00);
}

data_counter=2;
}
else
{
                                data_status=2;
                                printf("DATA COUNT error! \n\n");
}
}
else
                                //Extract for coordinate data and stop byte
if (data_counter>1)
{

data_id=(data_counter-2)/4;

//Extract for coordinate data
if (data_id<data_count)
{
                                data_type_id=data_counter-((data_id*4)+2);

switch(data_type_id)
{
                                case 0:
                                {
                                                write_bank(2,data_id,data_rcv);
                                break;
                                }

                                case 1:
                                {
                                                write_bank(2,data_id+8,data_rcv);
                                break;
                                }

                                case 2:
                                {
                                                write_bank(3,data_id,data_rcv);
                                break;
                                }
}
}
}

```

```

        case 3:
        {
                write_bank(3,data_id+8,data_rcv);
                break;
        }
    }

    //Increase counter
    data_counter=data_counter+1;

}

else
//Extract for stop byte
if (data_id==data_count)
{
    if (data_rcv==0xFE)
    {
                data_status=3;
    }
    else
    {
        data_status=2;
        printf("STOP BYTE error! \n\n");
    }
}
}

//If there is an error
if (data_status==2)
{
    //ERROR ROUTINE!

                //Restart procedure
                data_status=0;
                data_count=1;
                data_counter=0;

        printf("Reload data or reset PIC \n\n");
        beeper_off(5);
}

//If data completed
if (data_status==3)
{
    //FINISH ROUTINE!
        printf("Successful! \n\n",flow_flag);
        printf("Data count= %u \n\n",data_count);

    //Read all memory

```

```

        for (i=0;i<=(data_count-1);i++)
        {
            printf("%u\t%u,%u,%u,%u\n",i+1,read_bank(2,i),read_bank(2,i+8),read_bank(3,i),read_bank(3,i+8));
        }

            //Restart procedure
            data_status=0;
            data_counter=0;

            printf("Press START switch to resume \r\n");
            beeper_off(4);
        }

    }
}

#int_ext
void ext_isr(void);

#int_rb
void ext_rb(void);

##### MAIN CODES STARTS #####
void main()
{
    int count,signx,x,signy,y;

    printf("\nWelcome to RMS v1.3 \r\n");
    beeper_off(1);

    INTF_BIT = 0;
    disable_interrupts(INT_RDA);
    disable_interrupts(INT_RB);
    disable_interrupts(INT_EXT);
    disable_interrupts(global);

    setup_ccp1(ccp_off);
    setup_ccp2(ccp_off);

    set_tris_a(0xFF); // use all as input at the moment .. maybe status led after this
    set_tris_b(0xF1); // not use at the moment, use for sensor interrupt
    set_tris_c(0x89); // use for PWM and encoder input 0000 1001
    set_tris_d(0x00); // all use for motor controller and servo

    port_b=0xF1;

    initialize();
}

```

```

port_b_pullups(TRUE);
delay_ms(1000);

//*****DATA UPLOADING*****
//Clear RAM
for (i=0;i<=15;i++)
{
    write_bank(2,i,0x00);
    write_bank(3,i,0x00);
}

//RDA and global ON
enable_interrupts(INT_RDA);
enable_interrupts(global);

printf("Upload data \r\n");
beeper_off(2);

while (true)
{
    if (input(PIN_A0))
    {
        //RDA and global OFF
        disable_interrupts(INT_RDA);
        disable_interrupts(global);

        break;
    }
}

//*****

ext_int_edge(H_TO_L);
INTF_BIT = 0;

//Global ON
enable_interrupts(global);

delay_ms(1000);

beeper_off(3);

while(true)
{
    printf("Press START switch to run \r\n");
    while (true)
    {
        if (input(PIN_A0))
        {
            break;
        }
    }
}

```



```

    for (count=0;count<=(data_count-1);count++)
    {
        //printf("Point=%u \r\n",count+1);

        signx=read_bank(2,count);
        x=read_bank(2,count+8);
        signy=read_bank(3,count);
        y=read_bank(3,count+8);

        //printf("signx[%u]=%u, x[%u]=%u, ",count,signx,count,x);
        //printf("signy[%u]=%u, y[%u]=%u \r\n",count,signy,count,y);

        allinone(signx,x,signy,y);

        beeper_off(2);
    }

    setup_ccp1(ccp_off);
    setup_ccp2(ccp_off);

    initialize();
    beeper_off(5);
}

while (1);
}

##### MAIN CODES ENDS #####

##### INITIALIZATION CODES STARTS #####
void initialize(void)
{
    port_c=0x00;
    port_d=0x00;
}
##### INITIALIZATION CODES ENDS #####

##### INTERDELAY -PAUSE- START #####
void interdelay(void)
{
    printf("Interdelay \r\n");
    port_d=0x00;
    delay_ms(1000);
}

```

```

}
##### INTERDELAY -PAUSE- ENDS #####

##### PATH ALGO STARTS #####
void allinone (int signx,int x,int signy, int y)
{
    if (signx==0 && signy==0)
    {
//If goto Axis 1

        if (y>0)
        {
            straight(y);
            interdelay();
        }
        if (x>0)
        {
            turn_right();
            interdelay();
            straight(x);
            interdelay();
        }
    }
    else if (signx==1 && signy==0)
    {
//If goto Axis 2

        if (y>0)
        {
            straight(y);
            interdelay();
        }

//Note:x must be > 0 because signx=1
        turn_left();
        interdelay();
        straight(x);
        interdelay();
    }
    else if (signx==1 && signy==1)
    {
//If goto Axis 3

//Note:x must be > 0 because signx=1
        turn_left();
        interdelay();
        straight(x);

```

```

interdelay();

//Note:y must be > 0 because signy=1

turn_left();
interdelay();
straight(y);
interdelay();
}
else if (signx==0 && signy==1)
{
//If goto Axis 4

turn_right();
interdelay();

if (x>0)
{
straight(x);
interdelay();
}

//Note:y must be > 0 because signy=1
turn_right();
interdelay();
straight(y);
interdelay();
}
}

// ##### PATH ALGO ENDS #####

// ##### BASIC MOVEMENTS STARTS #####
//added err codes
//Drive Train=DCBA, DC=right wheel, BA=left wheel
//Drive Data: 00=No movement, 10=Forward, 01=Reverse

void straight(int distance_incm)
{

int state_right,state_left;

//actual distance
float actual_total_avg;

float distance_pulse;

//tak berubah
long duty_right;
long duty_left;

```

```

//dir_path:
//0.0=same pulse, 1.0=right>left (overshoot left), 2.0=left>right (overshoot right)
    float pulse_diff;
float dir_path;

float pulse_total_right;
float pulse_total_left;
int distance;

float pulse_total_right_temp;
float pulse_total_left_temp;
int distance_temp;

//dir_line:
//0=straight inline
//1=small overshoot left previously, 2=small overshoot right previously
//3=large overshoot left previously, 4=large overshoot right previously
//5=lost!
//6=dummy value (at startup of detection ONLY)
int linefollow_mode;
int dir_line;

//Jump to marker3
    pulse_total_right=0.0;
pulse_total_left=0.0;
distance=distance_incm;

is_main_interrupted=0;

    goto marker3;

marker2:    //route on returning from all interrupt flow

    //reserved: pulse_total_right
//reserved: pulse_total_left
//reserved: distance
beeper_off(4);

    pulse_total_right=pulse_total_right_temp;
pulse_total_left=pulse_total_left_temp;
distance=distance_temp;

is_main_interrupted=0;

    goto marker3;

marker1: //route for interrupt flow

    pulse_total_right=0.0;

```

```

pulse_total_left=0.0;

    if (flow_flag==2 || flow_flag==5)
    {
        distance=200;
    }
    else
    {
        distance=100;
    }

//Jump to marker1
    goto marker3;

marker3:    //common route for all

    pulse_diff=0.0;
    dir_path=0.0;

    actual_total_avg=0.0;

    duty_right=330;
    duty_left=500;

//Reset to search for line follow mode
    linefollow_mode=0;

    //printf("Straight \r\n");

    //use pin C0 as right wheel encoder input
    //use pin C3 as left wheel encoder input
    //read initial state of encoder
    state_right = input(PIN_C0);
    state_left= input(PIN_C3);

//send pwm
    pwm_right(duty_right);
    pwm_left(duty_left);

    //Send signal to servo
    port_d=0x00;
    delay_ms(500);

//release
    port_d=0x40;

    output_bit(PIN_C4,0); //nitestaje
    output_bit(PIN_C5,0); //nitestaje

    //ON EXT, ON RB before moving

```

```

//*****
is_mark=0;
INTF_BIT = 0;
enable_interrupts(INT_EXT);
enable_interrupts(INT_RB);
//*****

//moving the robot
// servo DC(right wheel) BA(left wheel)
//0100 1010
port_d=0x4A;

// 1 pulse =7.853cm/rotation.
distance_pulse=(float)distance/3.9269;

////////////////////////////////////
//hitam = 0 putih =1
while (actual_total_avg < distance_pulse)
{

//-----CHECK FOR LINE FOLLOW MODE-----
if (linefollow_mode==0)
{
//if senses all 3 black stripes
if (input(PIN_E0)==1 && input(PIN_E1)==1 && input(PIN_E2)==1)
{
linefollow_mode=1;
beeper_off(1);

output_bit(pin_C4,0); //nitestaje
output_bit(pin_C5,0); //nitestaje

//Reset duty cycle to be normal again
duty_right=330;
duty_left=500;
//send pwm
pwm_right(duty_right);
pwm_left(duty_left);

//Set dummy value
dir_line=6;

printf("Line \n");
}
}

//-----

//if interrupt occur, main not yet flagged
if (is_mark==1 && is_main_interrupted==0)

```

```

{
  //Reserve state
      pulse_total_right_temp=pulse_total_right;
      pulse_total_left_temp=pulse_total_left;
      distance_temp=distance;

      is_main_interrupted=1;
}

      //If YES (main is flagged), and small interrupt not yet flagged
//*****YES*****
if (is_main_interrupted==1 && is_mark==1)
{
      switch(flow_flag)
      {
          case 1:
          {
              turn_right();
              interdelay();

              goto marker1;

              break;
          }

          case 7:
          {
              turn_left();
              interdelay();
              turn_left();
              interdelay();

              goto marker1;

              break;
          }

          case 4:
          {
              turn_left();
              interdelay();

              goto marker1;

              break;
          }

          case 8:
          {
              //worst case scenario
              beeper_off(10);
              break;
          }
      }
}

```

```

    }
}
//*****YES*****

//If main is flagged, and small interrupt is flagged OR main is never flagged
//*****STRAIGHT CALCULATION*****

if ((is_main_interrupted==1 && is_mark==0) || (is_main_interrupted==0))
{

    //////////////////////////////////////measure distance pulse////////////////////////////////////
    //right
    if (input(PIN_C0) == 0 && state_right == 1)
    {
        pulse_total_right=pulse_total_right+1;
        state_right = 0;
    }
    else
    if (input(PIN_C0) == 1 && state_right ==0)
    {
        pulse_total_right=pulse_total_right+1;
        state_right = 1;
    }

    //left
    if (input(PIN_C3) == 0 && state_left == 1)
    {
        pulse_total_left=pulse_total_left+1;
        state_left = 0;
    }
    else
    if (input(PIN_C3) == 1 && state_left ==0)
    {
        pulse_total_left=pulse_total_left+1;
        state_left = 1;
    }

    //////////////////////////////////////measure distance pulse////////////////////////////////////

    ////////////////////////////////////// ECC //////////////////////////////////////
//If not in Line Follow Mode, do the Error Correction
if (linefollow_mode==0)
{
    if (pulse_total_right==pulse_total_left && dir_path!=0.0)
    {
        duty_right=330;
        duty_left=500;
        pwm_right(duty_right);
        pwm_left(duty_left);
    }
}
}

```



```

dir_path=0.0;

output_bit(pin_C4,0); //nitestaje
output_bit(pin_C5,0); //nitestaje
}
else
if (pulse_total_right>pulse_total_left)
{
pulse_diff=pulse_total_right-pulse_total_left;//right>left slow right faster left

if (pulse_diff>1.0 && dir_path!=1.0)
{
duty_right=450;
//duty_left=450;
pwm_right(duty_right);
//pwm_left(duty_left);

dir_path=1.0;

output_bit(pin_C4,1); //nitestaje
output_bit(pin_C5,0); //nitestaje
}
}
else
if (pulse_total_right<pulse_total_left)//right <left slow left faster right
{
pulse_diff=pulse_total_left-pulse_total_right;

if (pulse_diff>1.0 && dir_path!=2.0)
{
//duty_right=300;
duty_left=450;
//pwm_right(duty_right);
pwm_left(duty_left);

dir_path=2.0;

output_bit(pin_C4,1); //nitestaje
output_bit(pin_C5,1); //nitestaje
}
}
}
else
////////////////////////////////////Line Follow////////////////////////////////////
//If in Line Follow Mode, do the Line Follow Correction
if (linefollow_mode==1)
{
//E0=Left,E1=Center,E2=Right

```

```

//If small overshoot to the right
    if (input(PIN_E0)==1 && input(PIN_E1)==1 && input(PIN_E2)==0)
    {
        if (dir_line!=2)
        {
            output_bit(pin_C4,1);    //nitestaje
            output_bit(pin_C5,0);    //nitestaje

            //printf("OverR \r\n");

            //Decrease left PWM
            duty_right=400; //default=330
            duty_left=200; //sblm ni 300, default=500
            pwm_right(duty_right);
            pwm_left(duty_left);

            dir_line=2;
        }
    }
else
//If small overshoot to the left
    if (input(PIN_E0)==0 && input(PIN_E1)==1 && input(PIN_E2)==1)
    {
        if (dir_line!=1)
        {
            output_bit(pin_C4,0);    //nitestaje
            output_bit(pin_C5,1);    //nitestaje

            //Decrease right PWM
            duty_right=180; //sblm ni 200, default=330
            duty_left=500; //default=500
            pwm_right(duty_right);
            pwm_left(duty_left);

            //printf("OverL \r\n");

            dir_line=1;
        }
    }
else
//If large overshoot to the right
    if (input(PIN_E0)==1 && input(PIN_E1)==0 && input(PIN_E2)==0)
    {
        if (dir_line!=4)
        {
            output_bit(pin_C4,1);    //nitestaje
            output_bit(pin_C5,0);    //nitestaje

            //printf("OverR \r\n");

```

```

//Decrease left PWM
duty_right=500; //default=330
duty_left=110; //sbim ni 300, default=500
pwm_right(duty_right);
pwm_left(duty_left);

dir_line=4;
}
}
else
//If large overshoot to the left
if (input(PIN_E0)==0 && input(PIN_E1)==0 && input(PIN_E2)==1)
{
if (dir_line!=3)
{
output_bit(pin_C4,0); //nitestaje
output_bit(pin_C5,1); //nitestaje

//printf("OverL \r\n");

//Decrease right PWM (ni yg ada problem)
duty_right=120; //default=330
duty_left=500; //default=500
pwm_right(duty_right);
pwm_left(duty_left);

dir_line=3;
}
}
else
//If it is in-line straight
if ((input(PIN_E0)==0 && input(PIN_E1)==1 && input(PIN_E2)==0) ||
(input(PIN_E0)==1 && input(PIN_E1)==1 && input(PIN_E2)==1))
{
if (dir_line!=0)
{
output_bit(pin_C4,0); //nitestaje
output_bit(pin_C5,0); //nitestaje

duty_right=330;
duty_left=500;
pwm_right(duty_right);
pwm_left(duty_left);

dir_line=0;
}
}
else
//If it is lost from the line
if (input(PIN_E0)==0 && input(PIN_E1)==0 && input(PIN_E2)==0)

```

```

    {
    if (dir_line!=5)
    {
        output_bit(pin_C4,1);    //nitestaje
        output_bit(pin_C5,1);    //nitestaje

        //printf("Lost \r\n");

        //If previously overshoot to the left
        if (dir_line==1 || dir_line==3)
        {
            // servo DC(right wheel) BA(left wheel)
            //0100 0010
            port_d=0x42;

            delay_ms(100);

            //0100 1010
            port_d=0x4A;

            //Decrease right PWM (ni yg ada problem)
            duty_right=120; //default=330
            duty_left=500; //default=500
            pwm_right(duty_right);
            pwm_left(duty_left);

        }
        else
        //If previously overshoot to the right
        if (dir_line==2 || dir_line==4)
        {
            // servo DC(right wheel) BA(left wheel)
            //0100 1000
            port_d=0x48;

            delay_ms(100);

            //0100 1010
            port_d=0x4A;

            //Decrease left PWM
            duty_right=500; //default=330
            duty_left=110; //sblm ni 300, default=500
            pwm_right(duty_right);
            pwm_left(duty_left);

        }
        else
        //If previously is in-line straight
        if (dir_line==0)

```

```

    {

        duty_right=330; //default=330
        duty_left=500; //default=500
        pwm_right(duty_right);
        pwm_left(duty_left);

        //Out of line follow mode (and return to ECC mode)
        linefollow_mode=0;
        beeper_off(1);

        //Reset for ECC
        dir_path=0.0;
    }

    dir_line=5;
}
}

}

        ///////////////////////////////////////////////////Calculate average total pulse//////////////////////////////////////
        actual_total_avg=((float)pulse_total_left+(float)pulse_total_right)/2.0;
        //printf(">>L=%3.2f, R=%3.2f, A=%3.2f\n",pulse_total_left,pulse_total_right,actual_total_avg);
        ///////////////////////////////////////////////////Calculate average total pulse//////////////////////////////////////

    }

//*****STRAIGHT CALCULATION*****

}

//*****WHILE LOOP*****

//OFF after moving
//*****
disable_interrups(INT_RB);
disable_interrups(INT_EXT);
//*****

//If NO, change state flag
//*****NO*****
if (is_main_interrupted==1 && is_mark==0)
{

switch(flow_flag)
{

```

```

case 1:
{
    flow_flag=2;
    //printf("F=%u \r\n",flow_flag);

    interdelay();
    turn_left();
    interdelay();

    goto marker1;
    break;
}

case 2:
{
    flow_flag=3;
    //printf("F=%u \r\n",flow_flag);

    interdelay();
    turn_left();
    interdelay();

    goto marker1;
    break;
}

case 3:
{
    flow_flag=0;
    //printf("F=%u \r\n",flow_flag);

    interdelay();
    turn_right();
    interdelay();

    //Return to original main
    goto marker2;
    break;
}

case 4:
case 7:
{
    flow_flag=5;
    //printf("F=%u \r\n",flow_flag);

    interdelay();
    turn_right();
    interdelay();

```

```

        goto marker1;
    break;
}

case 5:
{
    flow_flag=6;
    //printf("F=%u \r\n",flow_flag);

    interdelay();
    turn_right();
    interdelay();

    goto marker1;
    break;
}

case 6:
{
    flow_flag=0;
    //printf("F=%u \r\n",flow_flag);

    interdelay();
    turn_left();
    interdelay();

    //Return to original main
    goto marker2;
    break;
}

}

}
//*****NO*****
}

```

```

void turn_right(void)
{
    int state_right,state_left;
    //int req_rotate_pulse=15;
    float req_rotate_pulse=10.0;
    int pulse_total_right=0,pulse_total_left=0;
    float actual_total_avg=0.0;

    long duty_right=330;
    long duty_left=500;
    pwm_right(duty_right);

```

```

pwm_left(duty_left);

//printf("Right \r\n");

//Send signal to servo
port_d=0x20;
delay_ms(200);

//0010 0110
port_d=0x26;

//read initial state of encoder
state_right = input(pin_C0);
state_left= input(pin_C3);

pulse_total_right=0;
pulse_total_left=0;

//hitam = 0 putih =1
while (actual_total_avg < req_rotate_pulse)
{
    //right
    if (input(pin_C0) == 0 && state_right == 1)
    {
        pulse_total_right=pulse_total_right+1;
        state_right = 0;
    }
    else
        if (input(pin_C0) == 1 && state_right ==0)
        {
            pulse_total_right=pulse_total_right+1;
            state_right = 1;
        }

    //left
    if (input(pin_C3) == 0 && state_left == 1)
    {
        pulse_total_left=pulse_total_left+1;
        state_left = 0;
    }
    else
        if (input(pin_C3) == 1 && state_left ==0)
        {
            pulse_total_left=pulse_total_left+1;
            state_left = 1;
        }

    actual_total_avg=((float)pulse_total_left+(float)pulse_total_right)/2.0;
}

```



```

        //Send signal to servo
        port_d=0x20;
        delay_ms(200);
    }

void turn_left(void)
{
    int state_right,state_left;
    //int req_rotate_pulse=15;
    float req_rotate_pulse=10.0;
    int pulse_total_right=0,pulse_total_left=0;
    float actual_total_avg=0.0;

    long duty_right=330;
    long duty_left=500;
    pwm_right(duty_right);
    pwm_left(duty_left);

    //printf("Left \r\n");

    //Send signal to servo
    port_d=0x20;
    delay_ms(200);

    //0010 1001
    port_d=0x29;

    //read initial state of encoder
    state_right = input(pin_C0);
    state_left= input(pin_C3);

    pulse_total_right=0;
    pulse_total_left=0;

    //hitam = 0 putih =1
    while (actual_total_avg < req_rotate_pulse)
    {
        //right
        if (input(pin_C0) == 0 && state_right == 1)
        {
            pulse_total_right=pulse_total_right+1;
            state_right = 0;
        }
        else
            if (input(pin_C0) == 1 && state_right ==0)
            {
                pulse_total_right=pulse_total_right+1;
                state_right = 1;
            }
    }
}

```

```

        //left
        if (input(pin_C3) == 0 && state_left == 1)
        {
            pulse_total_left=pulse_total_left+1;
            state_left = 0;
        }

        else
        if (input(pin_C3) == 1 && state_left ==0)
        {
            pulse_total_left=pulse_total_left+1;
            state_left = 1;
        }

actual_total_avg=((float)pulse_total_left+(float)pulse_total_right)/2.0;
    }

    //Send signal to servo
    port_d=0x20;
    delay_ms(200);
}

void stop()
{
    printf("Stop \r\n");

    port_d=0x00;
    delay_ms(500);
}

##### BASIC MOVEMENTS ENDS #####

##### MOTOR CONTROL PWM STARTS #####
void pwm_right(long dutyratio)
{
    setup_timer_2(T2_DIV_BY_16,249,1);

    setup_ccp1(ccp_pwm);
    set_pwm1_duty(dutyratio);
}

void pwm_left(long dutyratio)
{
    setup_timer_2(T2_DIV_BY_16,249,1);

    setup_ccp2(ccp_pwm);
    set_pwm2_duty(dutyratio);
}

```

```

}
##### MOTOR CONTROL PWM ENDS #####

##### BEEPER CODES STARTS #####
void beeper_off(int maxcount)
{
    int count;

    for (count=1;count<=maxcount;count++)
    {
        output_high(PIN_A1);
        delay_ms(100);
        output_low(PIN_A1);
        delay_ms(100);
    }
}

##### BEEPER CODES ENDS #####

#int_ext
void ext_isr()
{
    if (is_mark==0)
    {
        buffer_ext=port_b;
        disable_interrupts(INT_EXT);
        disable_interrupts(INT_RB);
        stop();
        beeper_off(4);

        printf("(F) \r\n");

        //tukar flag
        switch(flow_flag)
        {
            case 0:
            case 2:
            case 3:
            {
                flow_flag=1;
                //printf("F=%u \r\n",flow_flag);
                break;
            }

            case 1:
            {
                flow_flag=7;
                //printf("F=%u \r\n",flow_flag);
                break;
            }
        }
    }
}

```

```

        case 4:
        case 7:
    {
        flow_flag=8;
        //printf("F=%u \n\n",flow_flag);
        break;
    }

        case 5:
        case 6:
    {
        flow_flag=4;
        //printf("F=%u \n\n",flow_flag);
        break;
    }
}

    is_mark=1;
}

//Delay wajib
delay_ms(5000);

}

#int_rb
void ext_rb()
{
    buffer_rb=port_b&0xF0;

    if (buffer_rb==0xB0 || buffer_rb==0x70)
    {
        if (is_mark==0)
        {
            disable_interrupts(INT_RB);
            disable_interrupts(INT_EXT);
            stop();
            beeper_off(4);

            if (buffer_rb==0xB0)
            {
                printf("(L) \n\n");

                //tukar flag
                switch(flow_flag)
                {
                    case 0:
                    case 2:

```

```

        case 3:
        {
            flow_flag=1;
            //printf("F=%u \r\n",flow_flag);
            break;
        }

        case 1:
        {
            flow_flag=7;
            //printf("F=%u \r\n",flow_flag);
            break;
        }

        case 4:
        case 7:
        {
            flow_flag=8;
            //printf("F=%u \r\n",flow_flag);
            break;
        }

        case 5:
        case 6:
        {
            flow_flag=4;
            //printf("F=%u \r\n",flow_flag);
            break;
        }
    }
    else
    if (buffer_rb==0x70)
    {
        printf("(R) \r\n");

        //tukar flag
        switch(flow_flag)
        {
            case 0:
            case 2:
            case 3:
            {
                flow_flag=1;
                //printf("F=%u \r\n",flow_flag);
                break;
            }

            case 1:

```

```

        {
            flow_flag=7;
            //printf("F=%u \r\n",flow_flag);
            break;
        }

        case 4:
        case 7:
        {
            flow_flag=8;
            //printf("F=%u \r\n",flow_flag);
            break;
        }

        case 5:
        case 6:
        {
            flow_flag=4;
            //printf("F=%u \r\n",flow_flag);
            break;
        }
    }

    is_mark=1;
}

//Delay wajib
    delay_ms(1000);

}
else
{
    //printf("(X) \r\n",buffer_rb);

    INTF_BIT = 0;
    enable_interrupts(INT_RB);
}
}

```

## APPENDIX B: SERVO CONTROLLER SOURCE CODES

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

#define delay_1 225 //us
#define delay_2 700 //us
#define delay_3 1125 //us
#define delay_4 1300 //us

#define delay_pulse 17 //ms

void move_1(void);
void move_2(void);
void move_3(void);
void move_4(void);

void main()
{
    while (true)
    {
        //1
        if (input(PIN_A1)==1 && input(PIN_A0)==0)
        {
            output_low(PIN_B2);
            output_high(PIN_B4);

            move_1();
        }
        //2

    else if (input(PIN_A1)==1 && input(PIN_A0)==1)
    {
        output_high(PIN_B2);
        output_high(PIN_B4);

        move_4();
    }
    //3

    else if (input(PIN_A1)==0 && input(PIN_A0)==0)
    {
        output_low(PIN_B2);
        output_low(PIN_B4);

        move_3();
    }
    //4
    else if (input(PIN_A1)==0 && input(PIN_A0)==1)
    {
        output_high(PIN_B2);
        output_low(PIN_B4);

        move_2();
    }
    }
}

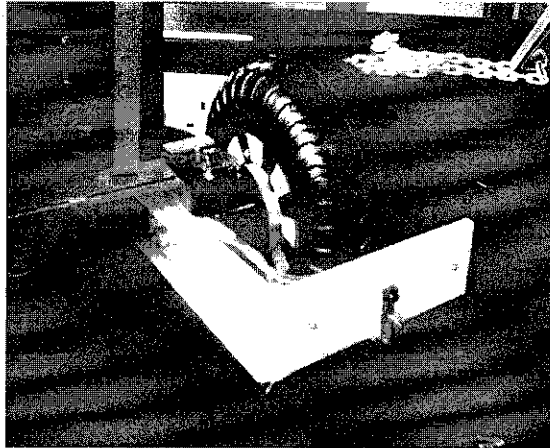
void move_1()
{
    output_high(PIN_B3);
    delay_us(delay_1);

    output_low(PIN_B3);
    delay_ms(delay_pulse);
}
```

```
}  
  
void move_2()  
{  
    output_high(PIN_B3);  
    delay_us(delay_2);  
  
    output_low(PIN_B3);  
    delay_ms(delay_pulse);  
}  
  
void move_3()  
{  
    output_high(PIN_B3);  
    delay_us(delay_3);  
  
    output_low(PIN_B3);  
    delay_ms(delay_pulse);  
}  
  
void move_4()  
{  
    output_high(PIN_B3);  
    delay_us(delay_4);  
  
    output_low(PIN_B3);  
    delay_ms(delay_pulse);  
}
```



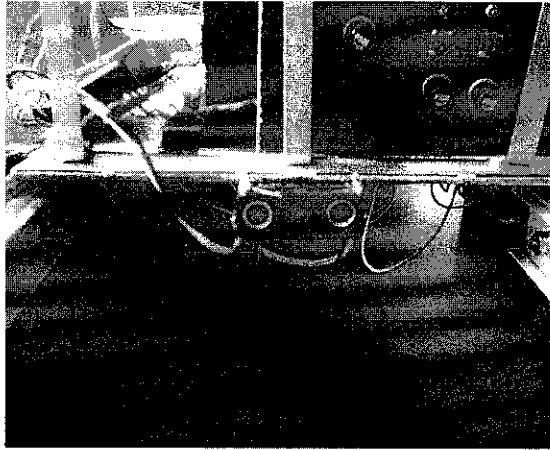
## APPENDIX C: MOBILE ROBOT PARTS



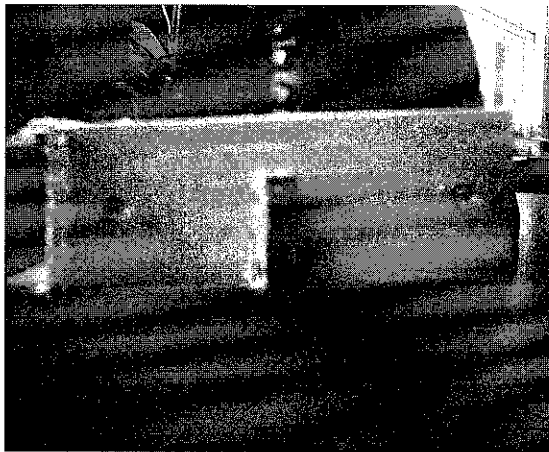
**Figure 0-1 Front setup mount of each tires**



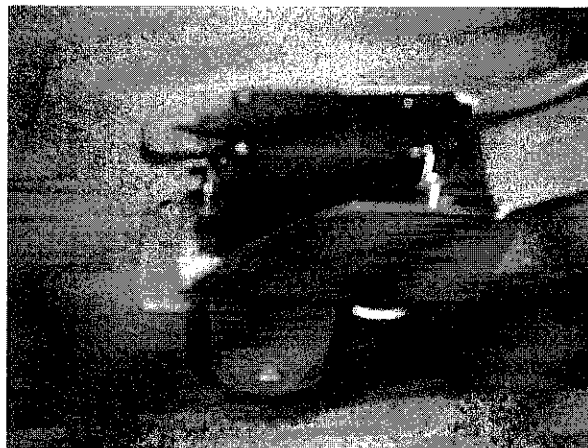
**Figure 0-2 Infrared / Rotary Encoder / Stripes**



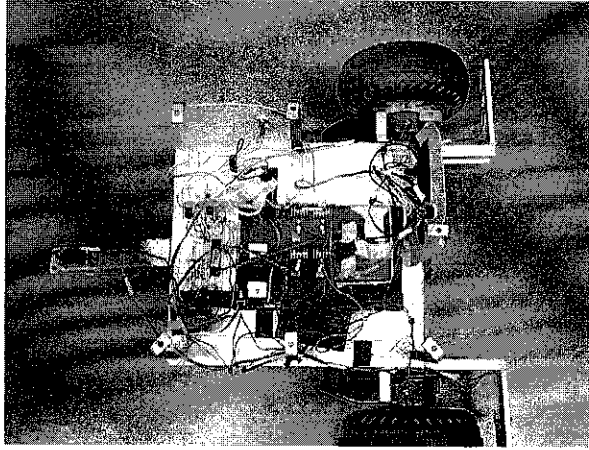
**Figure 0-3 Ultrasonic Transmitter and Receiver**



**Figure 0-4 Infrared Transmitter / Detector**



**Figure 0-5 Rear Servo Control Caster**

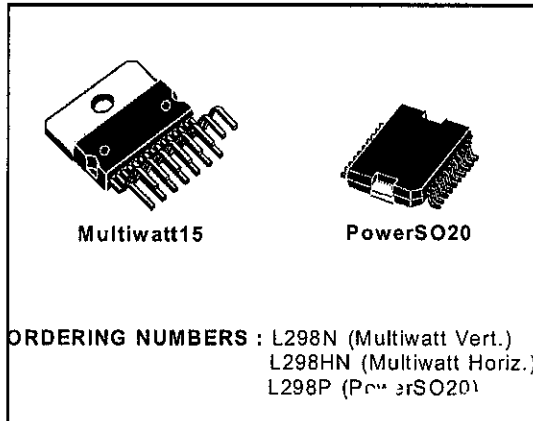


**Figure 0-6 The whole robot layout**

## **APPENDIX D: L298 Datasheet**

## DUAL FULL-BRIDGE DRIVER

- OPERATING SUPPLY VOLTAGE UP TO 46 V
- TOTAL DC CURRENT UP TO 4 A
- LOW SATURATION VOLTAGE
- OVERTEMPERATURE PROTECTION
- LOGICAL "0" INPUT VOLTAGE UP TO 1.5 V (HIGH NOISE IMMUNITY)

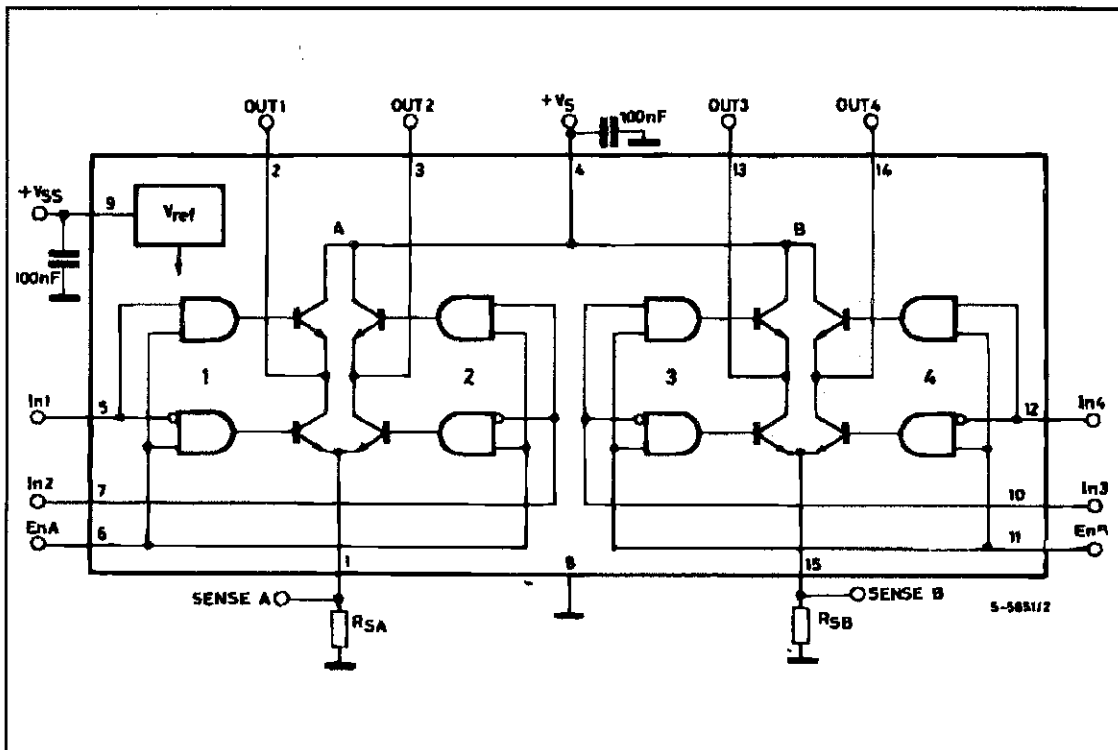


### DESCRIPTION

The L298 is an integrated monolithic circuit in a 15-lead Multiwatt and PowerSO20 packages. It is a high voltage, high current dual full-bridge driver designed to accept standard TTL logic levels and drive inductive loads such as relays, solenoids, DC and stepping motors. Two enable inputs are provided to enable or disable the device independently of the input signals. The emitters of the lower transistors of each bridge are connected together and the corresponding external terminal can be used for the connection of an external sensing resistor. An additional supply input is provided so that the logic works at a lower voltage.

nection of an external sensing resistor. An additional supply input is provided so that the logic works at a lower voltage.

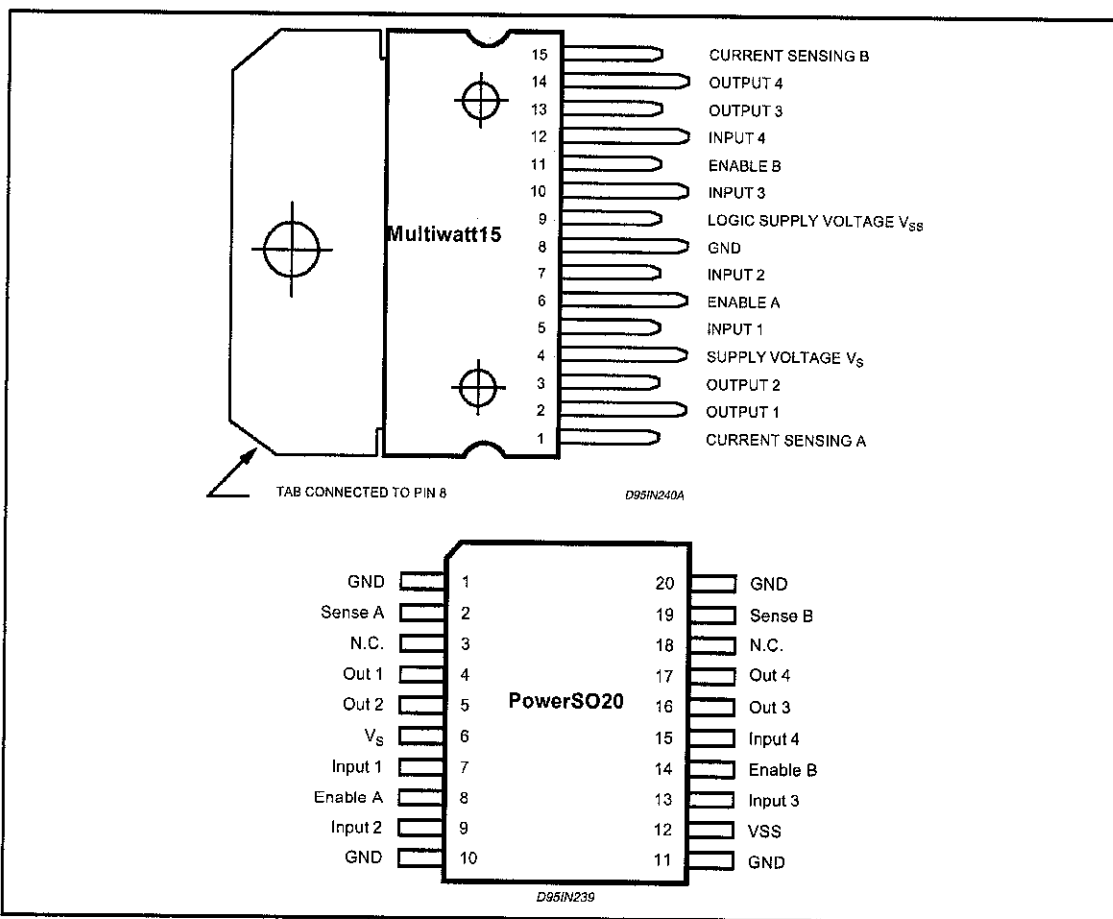
### BLOCK DIAGRAM



**ABSOLUTE MAXIMUM RATINGS**

Symbol	Parameter	Value	Unit
$V_s$	Power Supply	50	V
$V_{SS}$	Logic Supply Voltage	7	V
$V_i, V_{en}$	Input and Enable Voltage	-0.3 to 7	V
$I_o$	Peak Output Current (each Channel)		
	- Non Repetitive ( $t = 100\mu s$ )	3	A
	- Repetitive (80% on -20% off; $t_{on} = 10ms$ )	2.5	A
	-DC Operation	2	A
$V_{sens}$	Sensing Voltage	-1 to 2.3	V
$P_{tot}$	Total Power Dissipation ( $T_{case} = 75^\circ C$ )	25	W
$T_{op}$	Junction Operating Temperature	-25 to 130	$^\circ C$
$T_{stg}, T_j$	Storage and Junction Temperature	-40 to 150	$^\circ C$

**PIN CONNECTIONS (top view)**



**THERMAL DATA**

Symbol	Parameter		PowerSO20	Multiwatt15	Unit
$R_{th\ j-case}$	Thermal Resistance Junction-case	Max.	-	3	$^\circ C/W$
$R_{th\ j-amb}$	Thermal Resistance Junction-ambient	Max.	13 (*)	35	$^\circ C/W$

(\*) Mounted on aluminum substrate



## PIN FUNCTIONS (refer to the block diagram)

MW.15	PowerSO	Name	Function
1;15	2;19	Sense A; Sense B	Between this pin and ground is connected the sense resistor to control the current of the load.
2;3	4;5	Out 1; Out 2	Outputs of the Bridge A; the current that flows through the load connected between these two pins is monitored at pin 1.
4	6	V <sub>S</sub>	Supply Voltage for the Power Output Stages. A non-inductive 100nF capacitor must be connected between this pin and ground.
5;7	7;9	Input 1; Input 2	TTL Compatible Inputs of the Bridge A.
6;11	8;14	Enable A; Enable B	TTL Compatible Enable Input: the L state disables the bridge A (enable A) and/or the bridge B (enable B).
8	1,10,11,20	GND	Ground.
9	12	V <sub>SS</sub>	Supply Voltage for the Logic Blocks. A 100nF capacitor must be connected between this pin and ground.
10; 12	13;15	Input 3; Input 4	TTL Compatible Inputs of the Bridge B.
13; 14	16;17	Out 3; Out 4	Outputs of the Bridge B. The current that flows through the load connected between these two pins is monitored at pin 15.
–	3;18	N.C.	Not Connected

ELECTRICAL CHARACTERISTICS (V<sub>S</sub> = 42V; V<sub>SS</sub> = 5V, T<sub>J</sub> = 25°C; unless otherwise specified)

Symbol	Parameter	Test Conditions	Min.	Typ.	Max.	Unit
V <sub>S</sub>	Supply Voltage (pin 4)	Operative Condition	V <sub>IH</sub> +2.5		46	V
V <sub>SS</sub>	Logic Supply Voltage (pin 9)		4.5	5	7	V
I <sub>S</sub>	Quiescent Supply Current (pin 4)	V <sub>en</sub> = H; I <sub>L</sub> = 0		13	22	mA
		V <sub>i</sub> = L		50	70	mA
		V <sub>i</sub> = H				
		V <sub>en</sub> = L			4	mA
I <sub>SS</sub>	Quiescent Current from V <sub>SS</sub> (pin 9)	V <sub>en</sub> = H; I <sub>L</sub> = 0		24	36	mA
		V <sub>i</sub> = L		7	12	mA
		V <sub>i</sub> = H				
		V <sub>en</sub> = L			6	mA
V <sub>IL</sub>	Input Low Voltage (pins 5, 7, 10, 12)		-0.3		1.5	V
V <sub>IH</sub>	Input High Voltage (pins 5, 7, 10, 12)		2.3		V <sub>SS</sub>	V
I <sub>IL</sub>	Low Voltage Input Current (pins 5, 7, 10, 12)	V <sub>i</sub> = L			-10	μA
I <sub>IH</sub>	High Voltage Input Current (pins 5, 7, 10, 12)	V <sub>i</sub> = H ≤ V <sub>SS</sub> -0.6V		30	100	μA
V <sub>en</sub> = L	Enable Low Voltage (pins 6, 11)		-0.3		1.5	V
V <sub>en</sub> = H	Enable High Voltage (pins 6, 11)		2.3		V <sub>SS</sub>	V
I <sub>en</sub> = L	Low Voltage Enable Current (pins 6, 11)	V <sub>en</sub> = L			-10	μA
I <sub>en</sub> = H	High Voltage Enable Current (pins 6, 11)	V <sub>en</sub> = H ≤ V <sub>SS</sub> -0.6V		30	100	μA
V <sub>CEsat(H)</sub>	Source Saturation Voltage	I <sub>L</sub> = 1A	0.95	1.35	1.7	V
		I <sub>L</sub> = 2A		2	2.7	V
V <sub>CEsat(L)</sub>	Sink Saturation Voltage	I <sub>L</sub> = 1A (5)	0.85	1.2	1.6	V
		I <sub>L</sub> = 2A (5)		1.7	2.3	V
V <sub>CEsat</sub>	Total Drop	I <sub>L</sub> = 1A (5)	1.80		3.2	V
		I <sub>L</sub> = 2A (5)			4.9	V
V <sub>sens</sub>	Sensing Voltage (pins 1, 15)		-1 (1)		2	V

ELECTRICAL CHARACTERISTICS (continued)

Symbol	Parameter	Test Conditions	Min.	Typ.	Max.	Unit
T <sub>1</sub> (V <sub>i</sub> )	Source Current Turn-off Delay	0.5 V <sub>i</sub> to 0.9 I <sub>L</sub> (2); (4)		1.5		μs
T <sub>2</sub> (V <sub>i</sub> )	Source Current Fall Time	0.9 I <sub>L</sub> to 0.1 I <sub>L</sub> (2); (4)		0.2		μs
T <sub>3</sub> (V <sub>i</sub> )	Source Current Turn-on Delay	0.5 V <sub>i</sub> to 0.1 I <sub>L</sub> (2); (4)		2		μs
T <sub>4</sub> (V <sub>i</sub> )	Source Current Rise Time	0.1 I <sub>L</sub> to 0.9 I <sub>L</sub> (2); (4)		0.7		μs
T <sub>5</sub> (V <sub>i</sub> )	Sink Current Turn-off Delay	0.5 V <sub>i</sub> to 0.9 I <sub>L</sub> (3); (4)		0.7		μs
T <sub>6</sub> (V <sub>i</sub> )	Sink Current Fall Time	0.9 I <sub>L</sub> to 0.1 I <sub>L</sub> (3); (4)		0.25		μs
T <sub>7</sub> (V <sub>i</sub> )	Sink Current Turn-on Delay	0.5 V <sub>i</sub> to 0.9 I <sub>L</sub> (3); (4)		1.6		μs
T <sub>8</sub> (V <sub>i</sub> )	Sink Current Rise Time	0.1 I <sub>L</sub> to 0.9 I <sub>L</sub> (3); (4)		0.2		μs
f <sub>c</sub> (V <sub>i</sub> )	Commutation Frequency	I <sub>L</sub> = 2A		25	40	KHz
T <sub>1</sub> (V <sub>en</sub> )	Source Current Turn-off Delay	0.5 V <sub>en</sub> to 0.9 I <sub>L</sub> (2); (4)		3		μs
T <sub>2</sub> (V <sub>en</sub> )	Source Current Fall Time	0.9 I <sub>L</sub> to 0.1 I <sub>L</sub> (2); (4)		1		μs
T <sub>3</sub> (V <sub>en</sub> )	Source Current Turn-on Delay	0.5 V <sub>en</sub> to 0.1 I <sub>L</sub> (2); (4)		0.3		μs
T <sub>4</sub> (V <sub>en</sub> )	Source Current Rise Time	0.1 I <sub>L</sub> to 0.9 I <sub>L</sub> (2); (4)		0.4		μs
T <sub>5</sub> (V <sub>en</sub> )	Sink Current Turn-off Delay	0.5 V <sub>en</sub> to 0.9 I <sub>L</sub> (3); (4)		2.2		μs
T <sub>6</sub> (V <sub>en</sub> )	Sink Current Fall Time	0.9 I <sub>L</sub> to 0.1 I <sub>L</sub> (3); (4)		0.35		μs
T <sub>7</sub> (V <sub>en</sub> )	Sink Current Turn-on Delay	0.5 V <sub>en</sub> to 0.9 I <sub>L</sub> (3); (4)		0.25		μs
T <sub>8</sub> (V <sub>en</sub> )	Sink Current Rise Time	0.1 I <sub>L</sub> to 0.9 I <sub>L</sub> (3); (4)		0.1		μs

- 1) Sensing voltage can be -1 V for t ≤ 50 μsec; in steady state V<sub>sens</sub> min ≥ -0.5 V.
- 2) See fig. 2.
- 3) See fig. 4.
- 4) The load must be a pure resistor.

Figure 1 : Typical Saturation Voltage vs. Output Current.

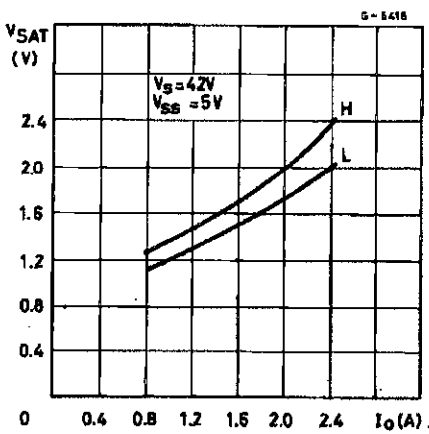
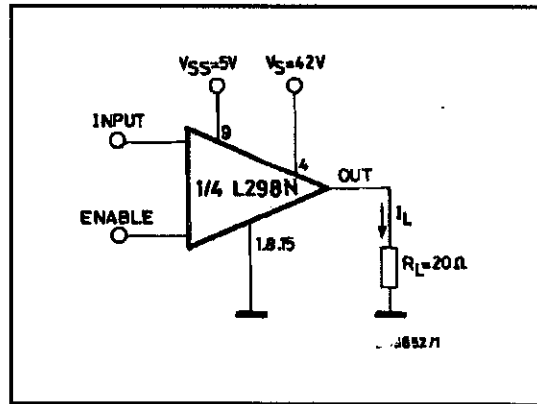


Figure 2 : Switching Times Test Circuits.



Note : For INPUT Switching, set EN = H  
 For ENABLE Switching, set IN = H





## **APPENDIX E: PIC16F877 Datasheet**



# PIC16F87X

## 28/40-Pin 8-Bit CMOS FLASH Microcontrollers

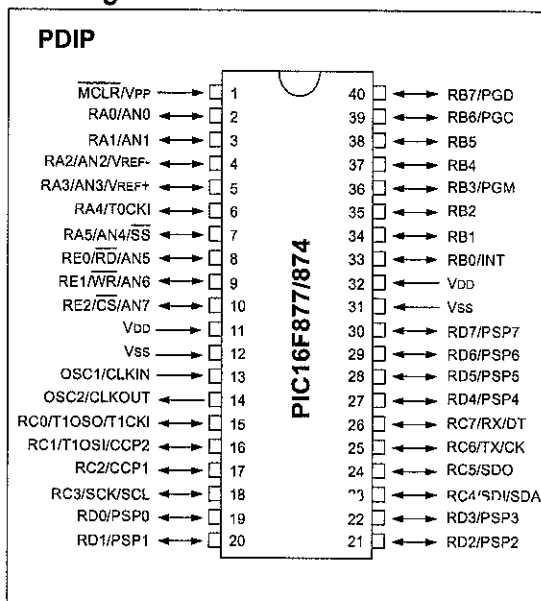
### Devices Included in this Data Sheet:

- PIC16F873                      • PIC16F876
- PIC16F874                      • 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



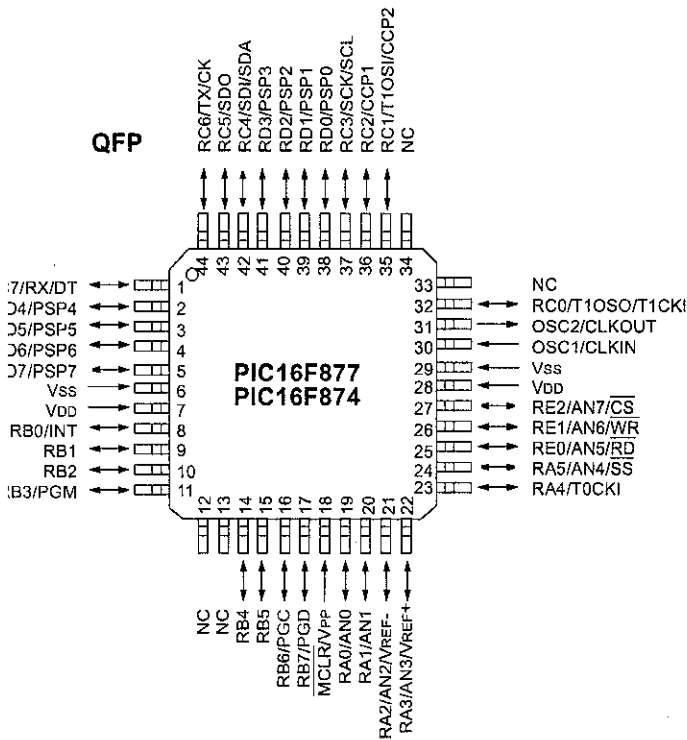
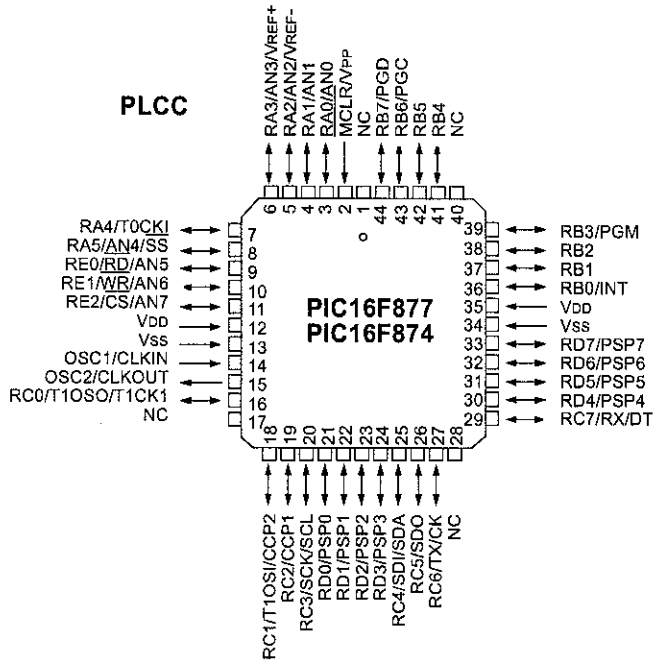
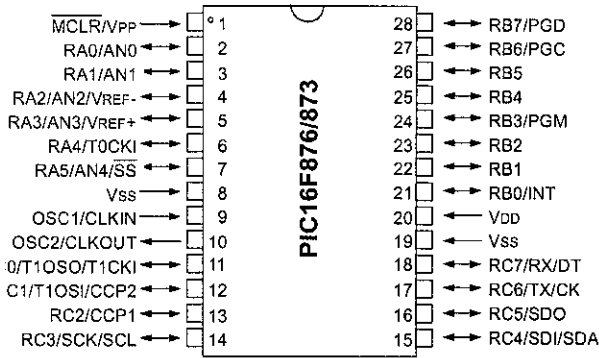
### Peripheral Features:

- 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<sup>2</sup>C™ (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 control's (40/44-pin only)
- Brown-out detection circuitry for  
Brown-out Reset (BOR)

# IC16F87X

## Diagrams

### PDIP, SOIC



# PIC16F87X

<b>Key Features PICmicro™ Mid-Range Reference Manual (DS33023)</b>	<b>PIC16F873</b>	<b>PIC16F874</b>	<b>PIC16F876</b>	<b>PIC16F877</b>
Operating Frequency	DC - 20 MHz	DC - 20 MHz	DC - 20 MHz	DC - 20 MHz
RESETS (and Delays)	POR, BOR (PWRT, OST)	POR, BOR (PWRT, OST)	POR, BOR (PWRT, OST)	POR, BOR (PWRT, OST)
FLASH Program Memory (14-bit words)	4K	4K	8K	8K
Data Memory (bytes)	192	192	368	368
EEPROM Data Memory	128	128	256	256
Interrupts	13	14	13	14
I/O Ports	Ports A,B,C	Ports A,B,C,D,E	Ports A,B,C	Ports A,B,C,D,E
Timers	3	3	3	3
Capture/Compare/PWM Modules	2	2	2	2
Serial Communications	MSSP, USART	MSSP, USART	MSSP, USART	MSSP, USART
Parallel Communications	—	PSP	—	PSP
10-bit Analog-to-Digital Module	5 input channels	8 input channels	5 input channels	8 input channels
Instruction Set	35 instructions	35 instructions	35 instructions	35 instructions

# C16F87X

## Table of Contents

Device Overview .....	5
Memory Organization.....	11
I/O Ports .....	29
Data EEPROM and FLASH Program Memory.....	41
Timer0 Module .....	47
Timer1 Module .....	51
Timer2 Module .....	55
Capture/Compare/PWM Modules .....	57
Master Synchronous Serial Port (MSSP) Module.....	65
Addressable Universal Synchronous Asynchronous Receiver Transmitter (USART) .....	95
Analog-to-Digital Converter (A/D) Module.....	111
Special Features of the CPU.....	119
Instruction Set Summary.....	135
Development Support .....	143
Electrical Characteristics.....	149
DC and AC Characteristics Graphs and Tables.....	177
Packaging Information .....	189
Appendix A: Revision History .....	197
Appendix B: Device Differences .....	197
Appendix C: Conversion Considerations .....	198
Customer Support.....	199
Customer Response.....	207
Customer Response.....	208
C16F87X Product Identification System .....	209

## TO OUR VALUED CUSTOMERS

Our intention is to provide our valued customers with the best documentation possible to ensure successful use of your Microchip products. To this end, we will continue to improve our publications to better suit your needs. Our publications will be refined and enhanced as new volumes and updates are introduced.

If you have any questions or comments regarding this publication, please contact the Marketing Communications Department via email at [docerrors@mail.microchip.com](mailto:docerrors@mail.microchip.com) or fax the **Reader Response Form** in the back of this data sheet to (480) 792-4150. We welcome your feedback.

### Obtain Current Data Sheet

To obtain the most up-to-date version of this data sheet, please register at our Worldwide Web site at:

<http://www.microchip.com>

You can determine the version of a data sheet by examining its literature number found on the bottom outside corner of any page. The last character of the literature number is the version number, (e.g., DS30000A is version A of document DS30000).

### Errata

An errata sheet, describing minor operational differences from the data sheet and recommended workarounds, may exist for current devices. As device/documentation issues become known to us, we will publish an errata sheet. The errata will specify the revision of silicon and revision of document to which it applies.

To determine if an errata sheet exists for a particular device, please check with one of the following:

Microchip's Worldwide Web site; <http://www.microchip.com>

Our local Microchip sales office (see last page)

The Microchip Corporate Literature Center; U.S. FAX: (480) 792-7277

When contacting a sales office or the literature center, please specify which device, revision of silicon and data sheet (include literature number) you are using.

### Customer Notification System

Register on our web site at [www.microchip.com/cn](http://www.microchip.com/cn) to receive the most current information on all of our products.

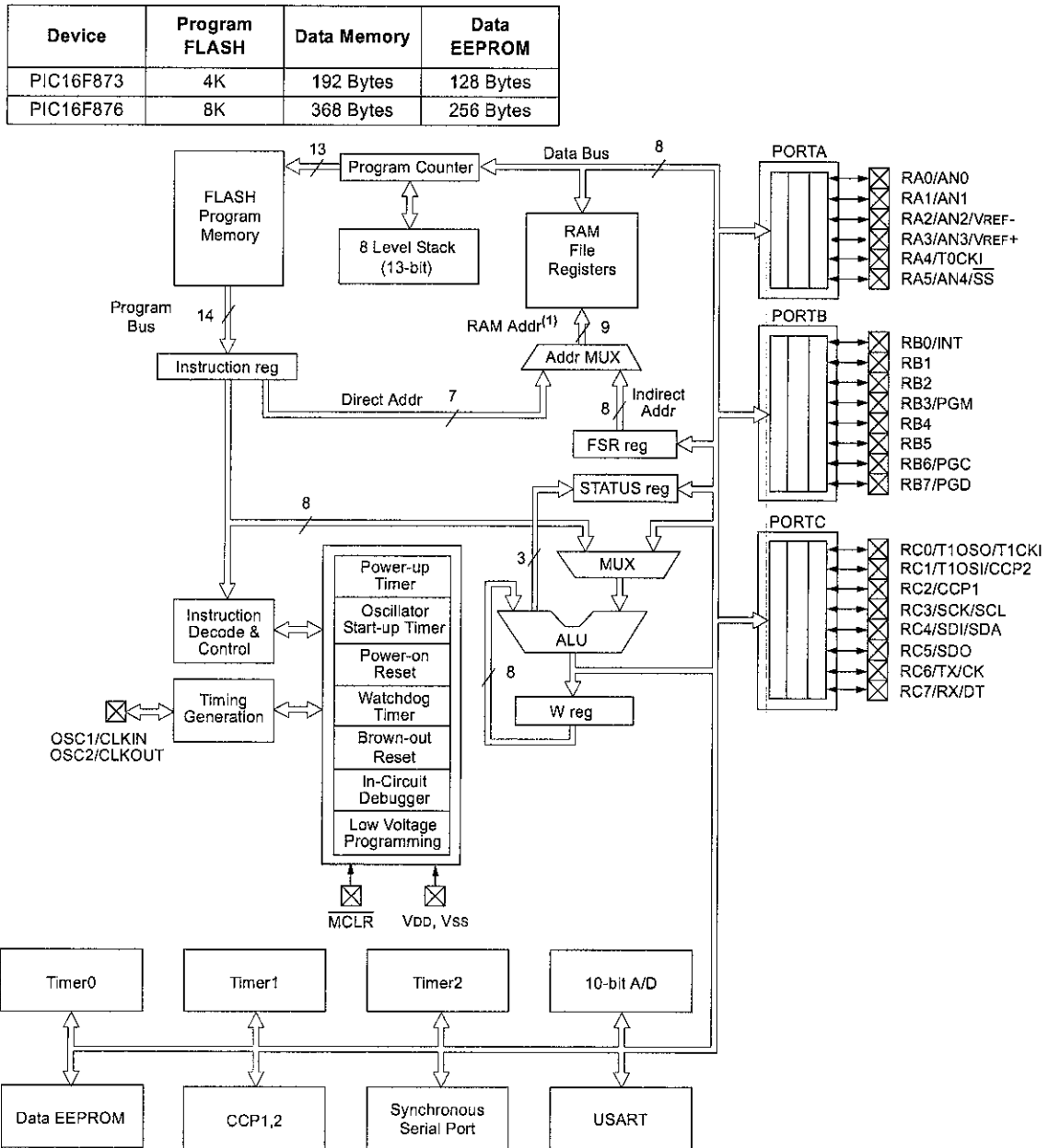
## 1.0 DEVICE OVERVIEW

This document contains device specific information. Additional information may be found in the PICmicro™ Mid-Range Reference Manual (DS33023), which may be obtained from your local Microchip Sales Representative or downloaded from the Microchip website. The Reference Manual should be considered a complementary document to this data sheet, and is highly recommended reading for a better understanding of the device architecture and operation of the peripheral modules.

There are four devices (PIC16F873, PIC16F874, PIC16F876 and PIC16F877) covered by this data sheet. The PIC16F876/873 devices come in 28-pin packages and the PIC16F877/874 devices come in 40-pin packages. The Parallel Slave Port is not implemented on the 28-pin devices.

The following device block diagrams are sorted by pin number; 28-pin for Figure 1-1 and 40-pin for Figure 1-2. The 28-pin and 40-pin pinouts are listed in Table 1-1 and Table 1-2, respectively.

**FIGURE 1-1: PIC16F873 AND PIC16F876 BLOCK DIAGRAM**

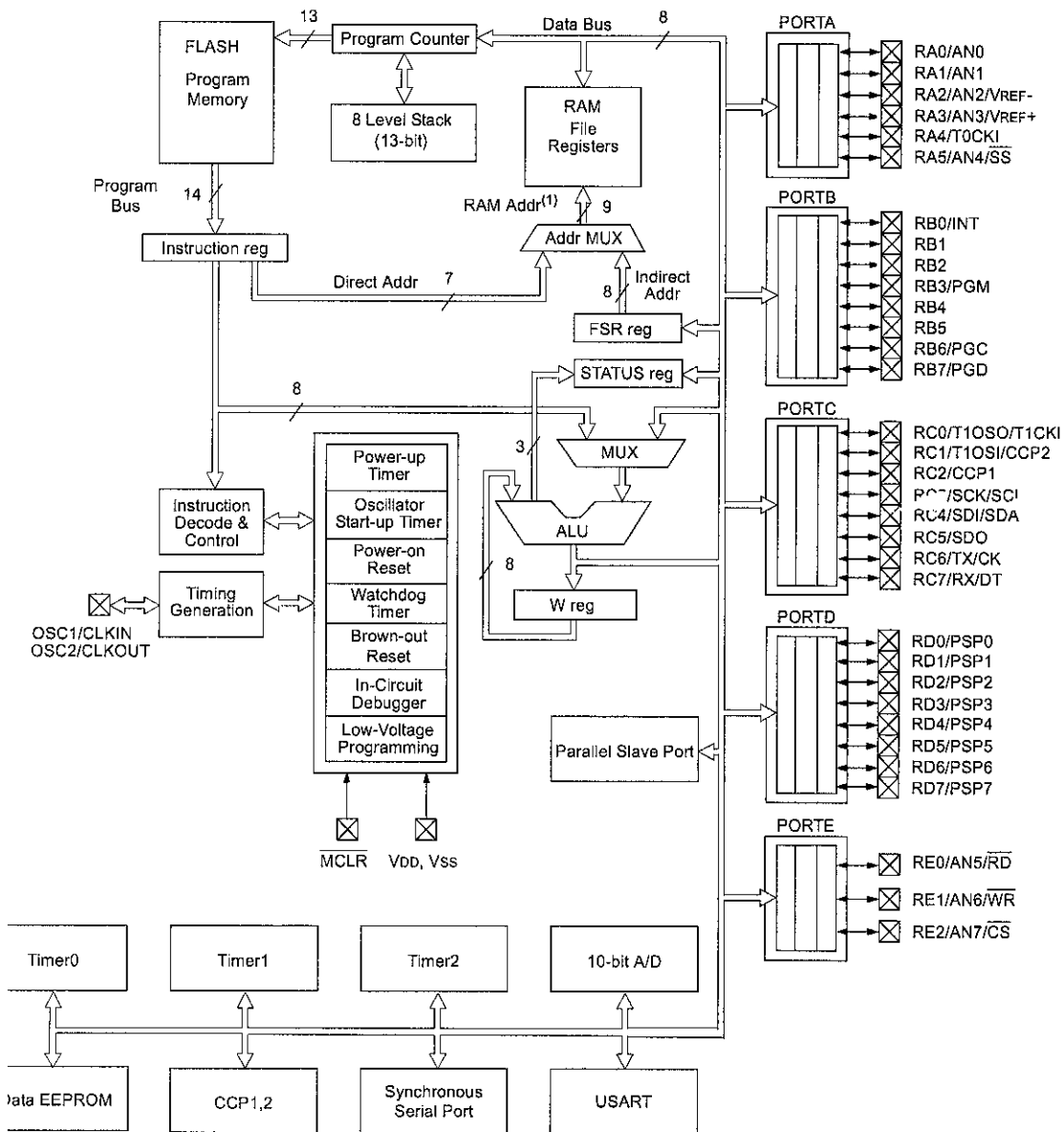


Note 1: Higher order bits are from the STATUS register.

# C16F87X

JRE 1-2: PIC16F874 AND PIC16F877 BLOCK DIAGRAM

Device	Program FLASH	Data Memory	Data EEPROM
PIC16F874	4K	192 Bytes	128 Bytes
PIC16F877	8K	368 Bytes	256 Bytes



Note 1: Higher order bits are from the STATUS register.

**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 <sup>(3)</sup>	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.
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.
RA0/AN0	2	2	I/O	TTL	<p>PORTA is a bi-directional I/O port.</p> <p>RA0 can also be analog input0.</p> <p>RA1 can also be analog input1.</p> <p>RA2 can also be analog input2 or negative analog reference voltage.</p> <p>RA3 can also be analog input3 or positive analog reference voltage.</p> <p>RA4 can also be the clock input to the Timer0 module. Output is open drain type.</p> <p>RA5 can also be analog input4 or the slave select for the synchronous serial port.</p>
RA1/AN1	3	3	I/O	TTL	
RA2/AN2/VREF-	4	4	I/O	TTL	
RA3/AN3/VREF+	5	5	I/O	TTL	
RA4/T0CKI	6	6	I/O	ST	
RA5/SS/AN4	7	7	I/O	TTL	
RB0/INT	21	21	I/O	TTL/ST <sup>(1)</sup>	<p>PORTB is a bi-directional I/O port. PORTB can be software programmed for internal weak pull-up on all inputs.</p> <p>RB0 can also be the external interrupt pin.</p> <p>RB3 can also be the low voltage programming input.</p> <p>Interrupt-on-change pin.</p> <p>Interrupt-on-change pin.</p> <p>Interrupt-on-change pin or In-Circuit Debugger pin. Serial programming clock.</p> <p>Interrupt-on-change pin or In-Circuit Debugger pin. Serial programming data.</p>
RB1	22	22	I/O	TTL	
RB2	23	23	I/O	TTL	
RB3/PGM	24	24	I/O	TTL	
RB4	25	25	I/O	TTL	
RB5	26	26	I/O	TTL	
RB6/PGC	27	27	I/O	TTL/ST <sup>(2)</sup>	
RB7/PGD	28	28	I/O	TTL/ST <sup>(2)</sup>	
RC0/T1OSO/T1CKI	11	11	I/O	ST	<p>PORTC is a bi-directional I/O port.</p> <p>RC0 can also be the Timer1 oscillator output or Timer1 clock input.</p> <p>RC1 can also be the Timer1 oscillator input or Capture2 input/Compare2 output/PWM2 output.</p> <p>RC2 can also be the Capture1 input/Compare1 output/PWM1 output.</p> <p>RC3 can also be the synchronous serial clock input/output for both SPI and I<sup>2</sup>C modes.</p> <p>RC4 can also be the SPI Data In (SPI mode) or data I/O (I<sup>2</sup>C mode).</p> <p>RC5 can also be the SPI Data Out (SPI mode).</p> <p>RC6 can also be the USART Asynchronous Transmit or Synchronous Clock.</p> <p>RC7 can also be the USART Asynchronous Receive or Synchronous Data.</p>
RC1/T1OSI/CCP2	12	12	I/O	ST	
RC2/CCP1	13	13	I/O	ST	
RC3/SCK/SCL	14	14	I/O	ST	
RC4/SDI/SDA	15	15	I/O	ST	
RC5/SDO	16	16	I/O	ST	
RC6/TX/CK	17	17	I/O	ST	
RC7/RX/DT	18	18	I/O	ST	
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.  
**Note 2:** This buffer is a Schmitt Trigger input when used in Serial Programming mode.  
**Note 3:** This buffer is a Schmitt Trigger input when configured in RC oscillator mode and a CMOS input otherwise.



# C16F87X

## LE 1-2: PIC16F874 AND PIC16F877 PINOUT DESCRIPTION

Pin Name	DIP Pin#	PLCC Pin#	QFP Pin#	I/O/P Type	Buffer Type	Description
1/CLKIN	13	14	30	I	ST/CMOS <sup>(4)</sup>	Oscillator crystal input/external clock source input.
2/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.
$\overline{R}/V_{PP}$	1	2	18	I/P	ST	Master Clear (Reset) input or programming voltage input. This pin is an active low RESET to the device.
AN0	2	3	19	I/O	TTL	PORTA is a bi-directional I/O port. RA0 can also be analog input0.
AN1	3	4	20	I/O	TTL	RA1 can also be analog input1.
AN2/VREF-	4	5	21	I/O	TTL	RA2 can also be analog input2 or negative analog reference voltage.
AN3/VREF+	5	6	22	I/O	TTL	RA3 can also be analog input3 or positive analog reference voltage.
T0CKI	6	7	23	I/O	ST	RA4 can also be the clock input to the Timer0 timer/counter. Output is open drain type.
$\overline{SS}/AN4$	7	8	24	I/O	TTL	RA5 can also be analog input4 or the slave select for the synchronous serial port.
INT	33	36	8	I/O	TTL/ST <sup>(1)</sup>	PORTB is a bi-directional I/O port. PORTB can be software programmed for internal weak pull-up on all inputs. RB0 can also be the external interrupt pin.
	34	37	9	I/O	TTL	
	35	38	10	I/O	TTL	
PGM	36	39	11	I/O	TTL	RB3 can also be the low voltage programming input.
	37	41	14	I/O	TTL	Interrupt-on-change pin.
	38	42	15	I/O	TTL	Interrupt-on-change pin.
PGC	39	43	16	I/O	TTL/ST <sup>(2)</sup>	Interrupt-on-change pin or In-Circuit Debugger pin. Serial programming clock.
PGD	40	44	17	I/O	TTL/ST <sup>(2)</sup>	Interrupt-on-change pin or In-Circuit Debugger pin. Serial programming data.

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

- 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
RC0/T1OSO/T1CKI	15	16	32	I/O	ST	PORTC is a bi-directional I/O port. 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 <sup>2</sup> C 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 <sup>2</sup> C 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.
RD0/PSP0	19	21	38	I/O	ST/TTL <sup>(3)</sup>	PORTD is a bi-directional I/O port or parallel slave port when interfacing to a microprocessor bus.
RD1/PSP1	20	22	39	I/O	ST/TTL <sup>(3)</sup>	
RD2/PSP2	21	23	40	I/O	ST/TTL <sup>(3)</sup>	
RD3/PSP3	22	24	41	I/O	ST/TTL <sup>(3)</sup>	
RD4/PSP4	27	30	2	I/O	ST/TTL <sup>(3)</sup>	
RD5/PSP5	28	31	3	I/O	ST/TTL <sup>(3)</sup>	
RD6/PSP6	29	32	4	I/O	ST/TTL <sup>(3)</sup>	
RD7/PSP7	30	33	5	I/O	ST/TTL <sup>(3)</sup>	
RE0/ $\overline{RD}$ /AN5	8	9	25	I/O	ST/TTL <sup>(3)</sup>	PORTE is a bi-directional I/O port. 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 <sup>(3)</sup>	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 <sup>(3)</sup>	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.  
**Note 2:** This buffer is a Schmitt Trigger input when used in Serial Programming mode.  
**Note 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).  
**Note 4:** This buffer is a Schmitt Trigger input when configured in RC oscillator mode and a CMOS input otherwise.

## APPENDIX F: PIC16F84A Datasheet

## 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

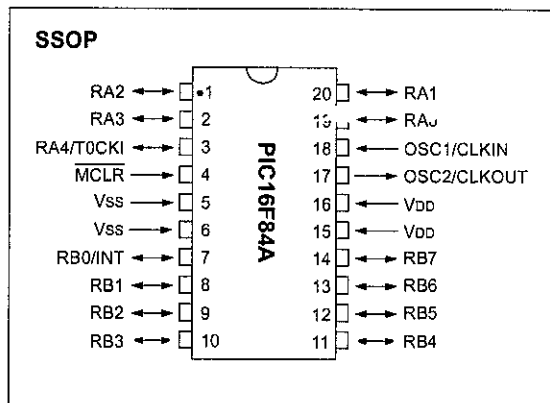
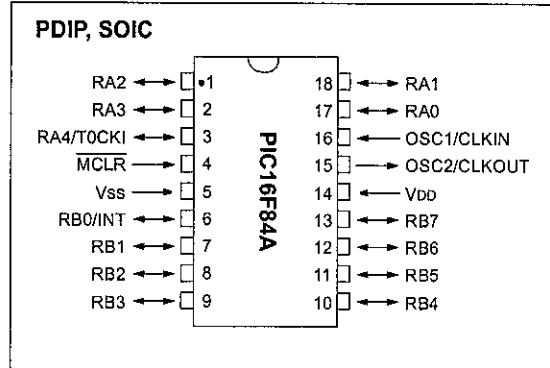
### Peripheral Features:

- 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



### 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

# IC16F84A

## Table of Contents

Device Overview .....	3
Memory Organization .....	5
Data EEPROM Memory .....	13
I/O Ports .....	15
Timer0 Module .....	19
Special Features of the CPU .....	21
Instruction Set Summary .....	35
Development Support.....	43
Electrical Characteristics .....	49
DC/AC Characteristic Graphs .....	61
Packaging Information.....	71
Appendix A: Revision History .....	75
Appendix B: Conversion Considerations.....	76
Appendix C: Migration from Baseline to Mid-Range Devices .....	78
Index .....	79
Online Support.....	83
Reader Response .....	84
IC16F84A Product Identification System .....	85

## TO OUR VALUED CUSTOMERS

It is our intention to provide our valued customers with the best documentation possible to ensure successful use of your Microchip products. To this end, we will continue to improve our publications to better suit your needs. Our publications will be refined and enhanced as new volumes and updates are introduced.

If you have any questions or comments regarding this publication, please contact the Marketing Communications Department via e-mail at [docerrors@mail.microchip.com](mailto:docerrors@mail.microchip.com) or fax the **Reader Response Form** in the back of this data sheet to (480) 792-4150. We welcome your feedback.

### Obtain Current Data Sheet

To obtain the most up-to-date version of this data sheet, please register at our Worldwide Web site at:

<http://www.microchip.com>

You can determine the version of a data sheet by examining its literature number found on the bottom outside corner of any page. The last character of the literature number is the version number, (e.g., DS30000A is version A of document DS30000).

### Errata

An errata sheet, describing minor operational differences from the data sheet and recommended workarounds, may exist for current devices. As device/documentation issues become known to us, we will publish an errata sheet. The errata will specify the revision of silicon and revision of document to which it applies.

To determine if an errata sheet exists for a particular device, please check with one of the following:

Microchip's Worldwide Web site; <http://www.microchip.com>

Your local Microchip sales office (see last page)

The Microchip Corporate Literature Center; U.S. FAX: (480) 792-7277

When contacting a sales office or the literature center, please specify which device, revision of silicon and data sheet (include literature number) you are using.

### Customer Notification System

Register on our web site at [www.microchip.com/cn](http://www.microchip.com/cn) to receive the most current information on all of our products.

## 1.0 DEVICE OVERVIEW

This document contains device specific information for the operation of the PIC16F84A device. Additional information may be found in the PICmicro™ Mid-Range Reference Manual, (DS33023), which may be downloaded from the Microchip website. The Reference Manual should be considered a complementary document to this data sheet, and is highly recommended reading for a better understanding of the device architecture and operation of the peripheral modules.

The PIC16F84A belongs to the mid-range family of the PICmicro® microcontroller devices. A block diagram of the device is shown in Figure 1-1.

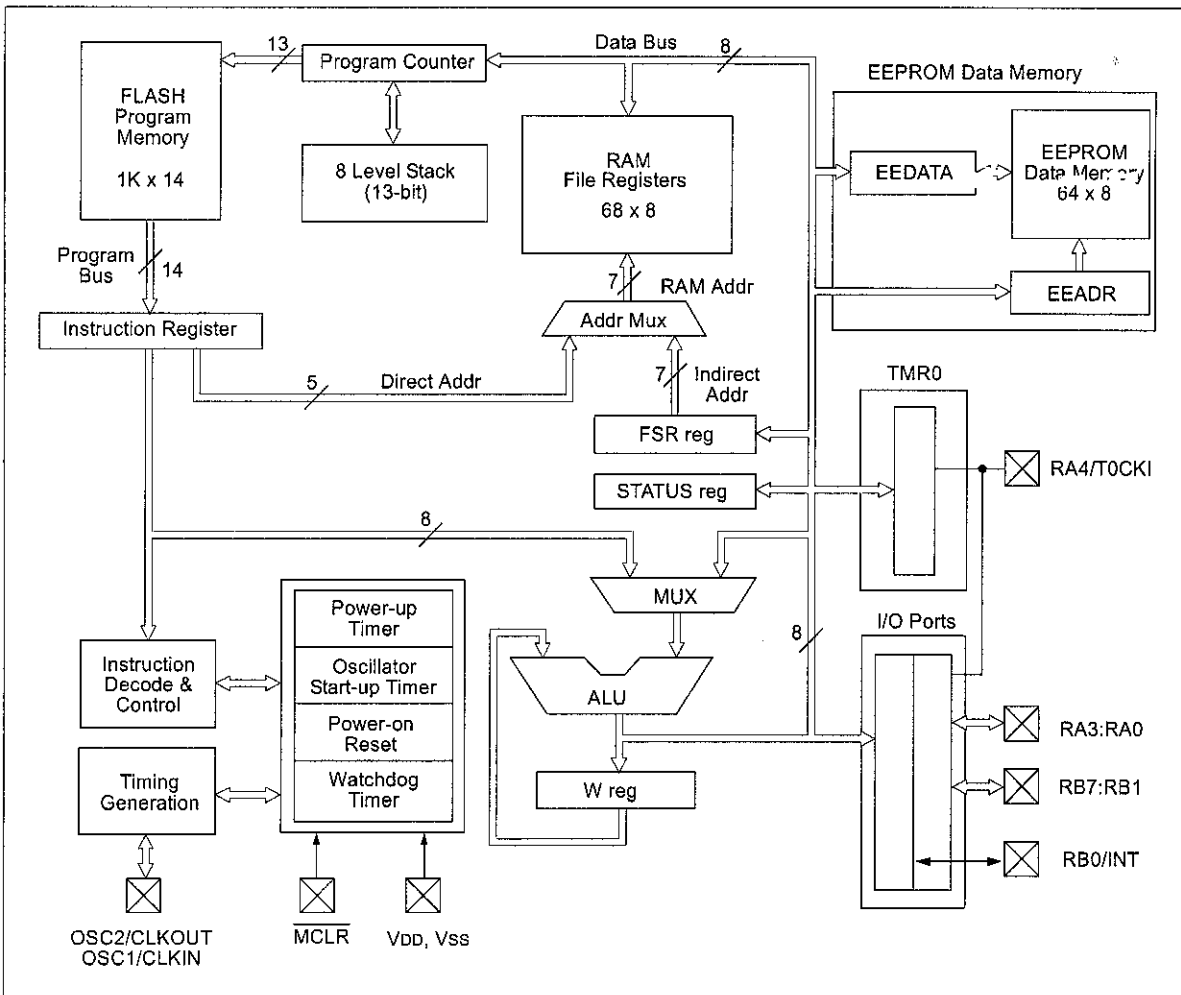
The program memory contains 1K words, which translates to 1024 instructions, since each 14-bit program memory word is the same width as each device instruction. The data memory (RAM) contains 68 bytes. Data EEPROM is 64 bytes.

There are also 13 I/O pins that are user-configured on a pin-to-pin basis. Some pins are multiplexed with other device functions. These functions include:

- External interrupt
- Change on PORTB interrupt
- Timer0 clock input

Table 1-1 details the pinout of the device with descriptions and details for each pin.

**FIGURE 1-1: PIC16F84A BLOCK DIAGRAM**



# IC16F84A

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 <sup>(3)</sup>	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 is 1/4 the frequency of OSC1 and denotes the instruction cycle rate.
$\overline{MR}$	4	4	4	I/P	ST	Master Clear (Reset) input/programming voltage input. This pin is an active low RESET to the device.
RA0	17	17	19	I/O	TTL	PORTA is a bi-directional I/O port.  Can also be selected to be the clock input to the TMR0 timer/counter. Output is open drain type.
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	
RB0/INT	6	6	7	I/O	TTL/ST <sup>(1)</sup>	PORTB is a bi-directional I/O port. PORTB can be software programmed for internal weak pull-up on all inputs.  RB0/INT can also be selected as an external interrupt pin.  Interrupt-on-change pin. Interrupt-on-change pin. Interrupt-on-change pin. Serial programming clock. Interrupt-on-change pin. Serial programming data.
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	
RB5	11	11	12	I/O	TTL	
RB6	12	12	13	I/O	TTL/ST <sup>(2)</sup>	
RB7	13	13	14	I/O	TTL/ST <sup>(2)</sup>	
$\overline{V_{SS}}$	5	5	5,6	P	—	Ground reference for logic and I/O pins.
$V_{DD}$	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

- 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 G: Operation Manual**

1. Load RMS software
  - a. Can be found in the CD
2. Connect the connection to mobile robot RS232
  - a. Connect between the robot RS232 module at the back of the robot with the computer COM port.
3. Turn on Main Power
  - a. The switch is located at the front of the robot
4. Turn on microcontroller power
  - a. Located at the left back side of the robot
  - b. Turn on line follower mode if needed.
5. Set the path on the RMS
  - a. Click on the box on the RMS interface
6. Load the coordinate
  - a. Click UPLOAD TO ROBOT
7. Press start button to confirm
  - a. Located at the left back side of the robot
8. Disconnect cable from PC to RS232
  - a. Disconnect the cable from the robot to the pc
9. Press start to run
  - a. The button is located at the left back side of the robot



## APPENDIX H: Components List

### *Structure and Mobility*

No	Item	Quantity	Where to get
1	Geared DC motor	2	Bicycle Shop near to Ipoh Mosque
2	Steel Frame		
3	Screws and nuts		

### *Power Distribution and Fail Safe*

No	Item	Quantity	Where to get
1	LM7805		Lab Store
2	LM7809		Lab Store
3	Fuses 4A	3	Meyer Electronics (Ipoh)
4	Vero Board	2	Lab Store
5	100uF Capacitor	1 for each regulator	Lab Store
6	0.1uf Capacitor	1 for each regulator	Lab Store
7	10uF Capacitor	1 for each regulator	Lab Store
8	Connectors		State Electronics (Ipoh)

***Rotary Encoder (For 2 sets )***

No	Item	Quantity	Where to get
1	Infrared TX and RX set	2	State Electronics (Ipoh)
2	330 Ohm	2	Lab Store
3	10K Ohm	2	Lab Store
4	47k Ohm Trimmer	2	Lab Store
5	LM 7805	1	Lab Store
6	NOT GATE	1	Lab Store

***Infrared Sensor***

No	Item	Quantity	Where to get
1	NE555 Timer	1	Lab Store
2	8 Pin Socket	1	Lab Store
3	Infrared TX Diodes	4	Lab Store
4	ISU160	2	Lab Store
5	47uF Capacitor	2	Lab Store
6	47 Ohm	2	Lab Store
7	1k Ohm	2	Lab Store
8	270k Ohm	1	Lab Store
9	100nF Capacitor	1	Lab Store

***Ultrasonic Sensor (DIY KIT)***

No	Item	Quantity	Where to get
1	ESCOL ES-15	1	Meyer Ipoh

### ***Line Follower***

No	Item	Quantity	Where to get
1	Infrared TX and RX set	3 Set	State Electronics Ipoh
2	330k Ohm	3	Lab Store
3	10k Ohm	3	Lab Store
4	40k Ohm Trimmers	3	Lab Store
5	Connector	1	State Electronics (Ipoh)

### ***Microprocessor @ Microcontroller***

No	Item	Quantity	Where to get
1	LM7805	1	Lab Store
2	5V Buzzer	1	Lab Store
3	4Mhz Xtal	1	Lab Store
4	Push Button (NO)	2	Lab Store
5	PIC 16F877	1	Lab Store @ <a href="http://www.microchip.com">www.microchip.com</a>
6	0.1uF Capacitor	1	Lab Store
7	100nF Capacitor	1	Lab Store
8	LED	6	Lab Store
9	3.3k Ohm	3	Lab Store
10	100uF	2	Lab Store
11	Connectors		State Electronics
12	Vero Board		

### ***Drive Circuit***

No	Item	Quantity	Where to get
1	L298D	2	State Electronics
2	0.1uF Capacitor	4	Lab Store
3	IN4004@ IN4001	8	Lab Store
4	Connectors		State Electronics
5	Vero Board		Lab Store

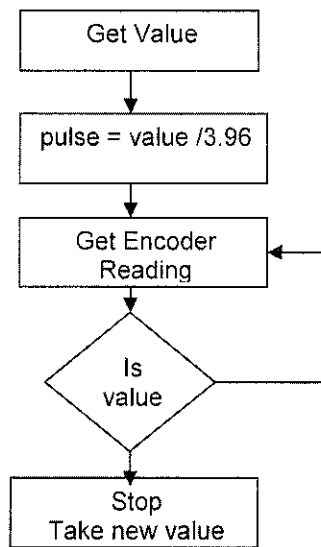
### ***RS232 Communication***

No	Item	Quantity	Where to get
1	MAX 232	1	State Electronics
2	1uF Capacitor	4	State Electronics @ Meyer Electronics
3	IC Socket	1	Lab Store
4	Male or Female D Connectors	2	Lab Store

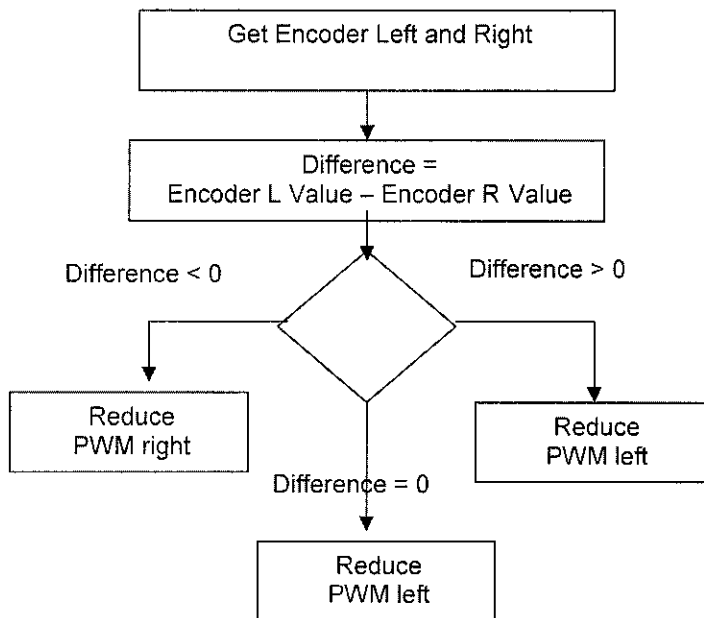
### ***Servo Controller***

No	Item	Quantity	Where to get
1	PIC 16F84 @ 16F84A	1	Lab Store
2	4Mhz Xtal	1	Lab Store
3	100nF Capacitor	1	Lab Store
4	330k Ohm	1	Lab Store
5	Led	2	Lab Store
6	Vero Board		Lab Store
7	Connectors	1	State Electronics

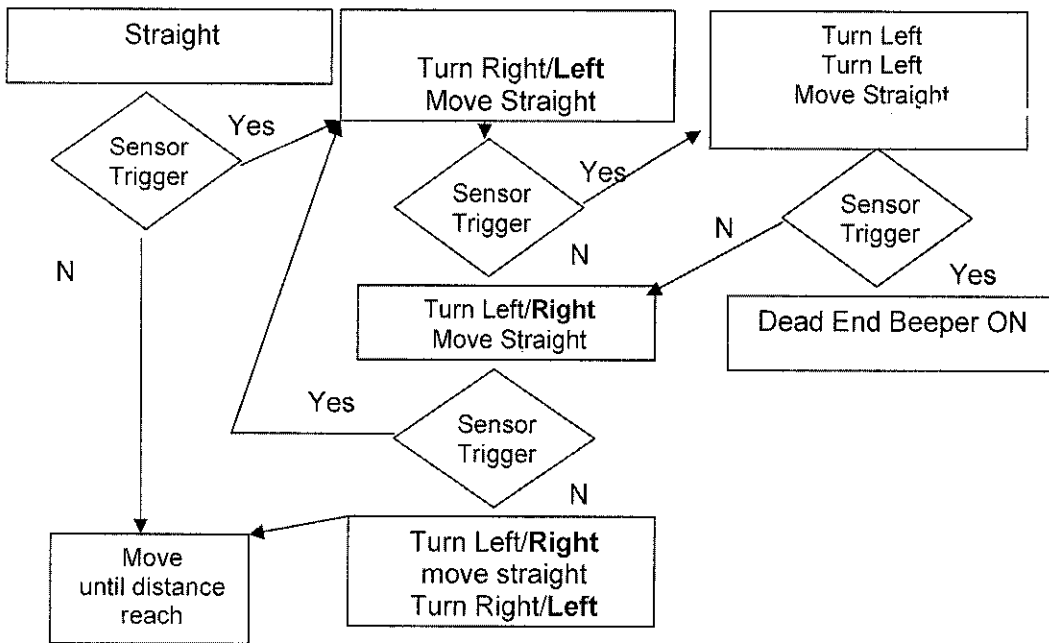
## APPENDIX I: Algorithm Flowchart



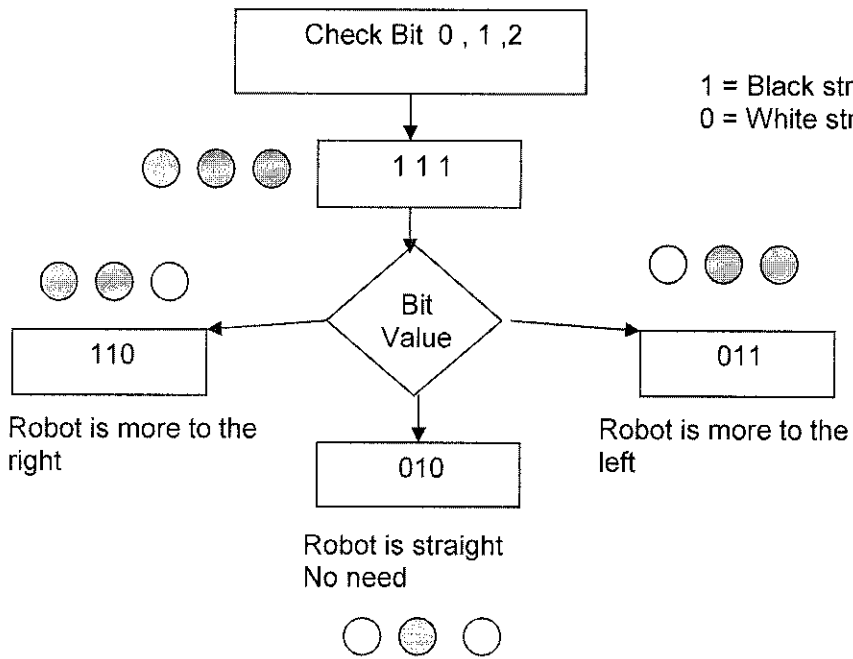
Dead reckoning algorithm



Error correction codes (ECC) algorithm



Obstacles avoidance algorithm



Line follower algorithm