CERTIFICATION OF APPROVAL

Implementation of an autonomous mobile robot navigation algorithm using C language

by

Shafeq Marwan b Subra Mullisi

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

Approved:

Dr Irraivan Elamvazuthi
Project Supervisor

UNIVERSITI TEKNOLOGI PETRONAS
TRONOH, PERAK

December 2009

CERTIFICATION OF ORIGINALITY

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

Shafeq Marwan b Subra Mullisi

ABSTRACT

Navigation is a major challenge for autonomous, mobile robots. The problem can basically be divided into positioning and path planning. This report basically discusses the study and also work that has been done from previous of the chosen topic, which is **Implementation of an Autonomous Mobile Robot Navigation Algorithm using 'C' language.** The objective of the project is to develop navigation algorithm for the autonomous mobile robot. After that, implement the navigation algorithm on the mobile robot and without using the external sensor for navigation of the mobile robot to reach the specified point. From the encoder programming to the motor speed control, this project basically focusing on how to control the motor speed movement such as rotation speed, turning speed, turning angle of the robot by controlling the movement of the motor and distance travel or displacement of the mobile robot from initial point to end point through some path that required turning algorithm and forward movement in controlled speed.

ACKNOWLEDGEMENTS

I would like to take this opportunity to acknowledge and thank everyone that has given me all the supports and guidance throughout the whole period of completing the final year project. Firstly, many thanks to the university and the Final Year Project coordinators that have coordinated and made the necessary arrangements for this study.

I must also acknowledge the endless help and support received from my supervisor, Dr. Irraivan throughout the whole period of completing the final year project. His guidance and advices are very much appreciated and help me a lot in my task to finish this FYP report.

I would also like to thank the lab technicians in UTP, especially Mrs Siti Hawa for advice and also guidance for this final report. Her continuous support and help throughout the whole period are very much appreciated.

Finally, I would like to express my gratitude to my fellow colleagues for their help and ideas throughout the completion of this study. Thank you all.

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER 1
INTRODUCTION

1.1 Background of study

Autonomous mobile robot is a robot that can move from one point to other point without human guidance. Simple mobile robot usually consists of controller and also wheel. Nowadays, there are many type of autonomous mobile robot have been design, from the Mars Pathfinder mission until the cleaning robot. All those robotic design were made combining several mechanism of navigation algorithm. These algorithms make the autonomous mobile robot navigate well in the real world environment. The successful design of navigation for autonomous mobile robot depend on the robot control architecture including identify movement, path planning, sensing and others. Most of the today mobile robot use sensor to navigate itself autonomously from one point to the others. In this project, the navigation algorithm will be focusing on the C language programming, without external sensor. The studies that have been carried out were focused on how to control the movement of the mobile robot by controlling the motor using the information from encoder of the robot.

The main reasons using C language for the navigation algorithm on the autonomous robot without using the sensor can be summarize as below:

1) Easier for coding process depending on various condition
2) Encoder is easier to program using the C language
3) Without the external sensor, internal sensor much more depends on programming language.
4) To implement navigation algorithm of the autonomous robot on the C language platform.

1.2 Problem Statement

Various researches and project have been done regarding the navigation algorithm of the autonomous robot. The abilities of the mobile robot to navigate through the space depend on how it can identify the structure of area, identify the exact position of it before it can recalculate to the next move. This can only be achieve with present of sensing method, for example using the line tracking sensor to sense the right path of movement, visual analyze method to visual the environment and identify the area, using the grid base navigation method by focusing on calculation of distance and time for the robot to travel [1] and other navigation method. So this project is base on close-loop navigation algorithm using the C language. The C language programming will be implementing to control the motor using the encoder to identify speed, distance, path and the direction of the mobile robot. The difficulties that need to be overcome are how to make sure the mobile robot can navigate through the space without external sensing system

1.3 Objective and scope of study

The main objectives of this research are:

- To implement the close-loop position control and navigation algorithm using the C language.
- Also to prove the validation of those algorithm after implementation on autonomous robot.

The study will be focused on:

- Develop the autonomous robot which is consisting of several components.
- Study on C language on close-loop navigation algorithm for autonomous robot.
- Speed control and angle displacement of the servo motor (encoder).
- Analyzing movement of the autonomous robot using the virtual grid.
- Develop the real-time task to testing the performance of the algorithm method.

CHAPTER 2
LITERATURE REVIEW

2.1 Introduction

Autonomous robot in other word means a robot that can perform specific task without any guidance (or more accurately without continuous guidance) from the human [2]. For the mobile robot, the basic construction of autonomous robot normally consists of a servo motor, wheel, PIC (programmable interface chip) and also power supply. To navigate the autonomous mobile robot, coding instruction must be program in the PIC of the autonomous robot. In this project, coding process will be done in C language. The main function of the coding is for controlling the movement and also the speed of the servo motor. By controlling the spend and movement of the servo motor, the angle displacement for the mobile robot and also the speed along with movement distance for the mobile robot can be set and control.

2.2 Sensor for dead-reckoning

Dead reckoning is one of the techniques for determining the location of vehicle using mathematical procedure by advancing some previous position on known course and also from the velocity or speed information over the length of time (Dunlap & Shufeldt, 1972)]. Most of the land-base mobile robots rely on the dead-reckoning system as the main navigation method, but there still problem pop out from this dead-reckoning system that need side control system and also sensor to reduce the problem and error [3].

There are several types of sensors in use today for dead-reckoning of the mobile robot navigation. Since most of the mobile robot rely on locomotion of wheel, the understanding of sensor for angular position and also velocity of wheel movement should be very important to design the navigation system for the mobile robot. The

sensors that most use for the dead-reckoning system are:

- Optical encoder
- Induce encoder
- Capacitive encoder
- Brush encoder
- Synchros
- Resolver
- Magnetic encoder

But each of the sensors is only suitable for some particular function, in order to use the sensor, some issue must be considered and suitable sensor should be chosen. For this project, optical encoder sensor will be selected. There are two types of optical encoder which is:

- Incremental optical encoder
- Absolute optical encoder

2.2.1 Incremental Encoder

This encoder consist of two channels which to be use to determine the angular motion of motor's shaft, most of today incremental optical encoder use 2 channel to provide ability for the encoder to determine the direction of rotation and can be used to calculate the position. Each of the channels is 90 degree out of phase. This allows the decoding to determine which phase is leading the other then the rotation direction can be determined.

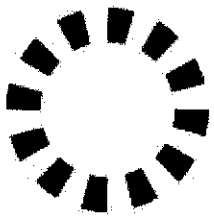

Figure 1: Incremental optical encoder disk [4]

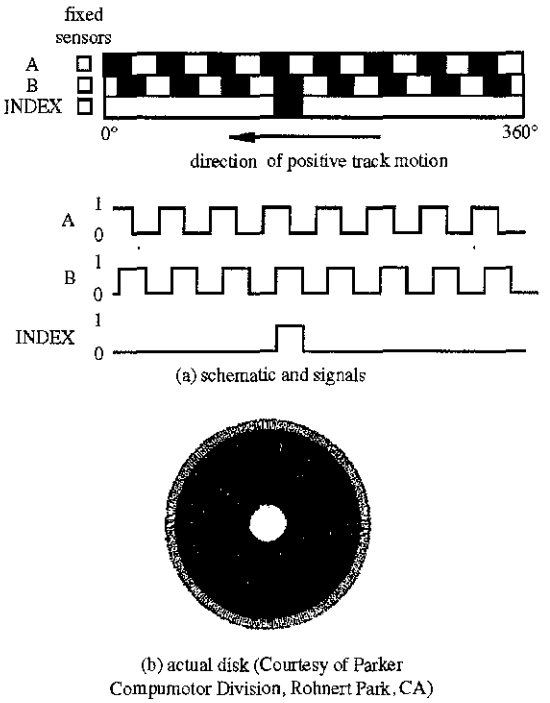

Figure 2: Incremental optical encoder signal schematic [5]

2.3 Drive system

For the drive system of mobile robot, several design issue need to be examine. This will provide some information for the selection of drive system for the mobile robot. The design issues for drive configuration are [3]:

- Maneuverability
- Controllability
- Traction
- Stability
- Efficiency
- Environmental effect
- Navigational consideration

2.3.1 Differential drive

The differential drive system has two motor mount in fixed position on the left and also right side of the mobile robot. This design need additional passive wheel which

can freely move without being controlled. With this configuration, mobile robot can move and turn freely by adjusting the speed of the motor drive.

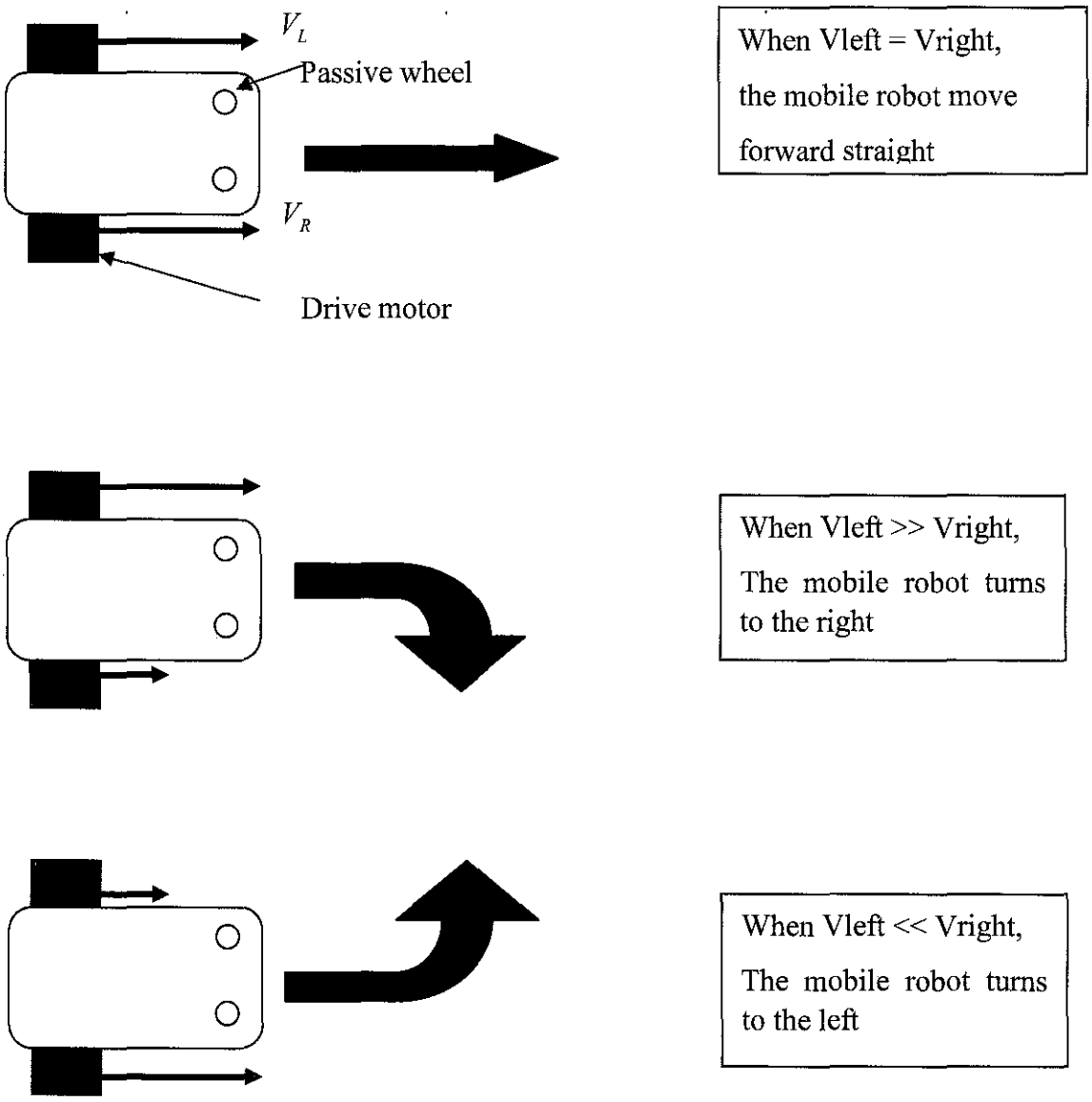

Figure 3: Mobile robot movement

Drive motor velocity is controlled by the optical encoder, the displacement of wheel and velocity of differential drive is available to be calculate, but the calculation is influenced by several parameter that need to be know.

The displacement of the mobile robot, D is given by equation 1 [3]:

$$D = D_L + D_R / 2 \qquad (1)$$

D = displacement of mobile robot (platform)

D_L= displacement of left wheel

D_R = displacement of right wheel

Equation for the velocity of the mobile robot is also same as displacement equation.

Suppose that at sampling interval I the left and right wheel encoders show a pulse increment of N_L and N_R, respectively. Suppose further that:

$$C_m = \pi D_n / n C_e \quad (2)$$

Where:

C_m = Conversion factor that translates encoder pulses into linear wheel displacement

D_n = Nominal wheel diameter (in mm)

C_e = Encoder resolution (in pulses per revolution)

n = Gear ratio of the reduction gear between the motor (where the encoder is attached) and the drive wheel

Using the equation (3), the incremental travel distance for left and right wheel can be calculated:

$$\Delta U_{L/R} = C_m \times N_{L/R} \quad (3)$$

From the equation above, the incremental travel distance for the center point of mobile robot, ΔU_i can be calculated using below equation:

$$\Delta U_i = (\Delta U_L + \Delta U_R)/2 \quad (4)$$

From the information of the travel distance, the mobile robot incremental orientation, $\Delta\theta$ will be deriving from equation (5):

$$\Delta\theta = (\Delta U_R - \Delta U_L)/b \quad (5)$$

Where b is wheelbase of mobile robot. The value is measured from the two contact point between floor and wheels. For the new relative orientation, θ_i value can be calculated by below equation:

$$\theta_i = \theta_{i-1} - \Delta\theta i$$

And relative position of center point is:

$x_i = x_{i-1} + \Delta U_i \cos\theta_i$

$y_i = y_{i-1} + \Delta U_i \sin\theta$

Where

x_i, y_i - relative position of the robot's center point c at instant I.

2.3.2 Tri-cycle drive

Tri-cycle drives different form the differential drive in term of its configuration and also drives system. The drive wheel is at front wheel and two passive wheels at the rear of mobile robot. Linear velocity and angular velocity are completely decoupled. For the straight driving, the drive wheel just positioning in straight direction and driven forward [3]. The problem associated with the tricycle-drive is the center gravity of the mobile robot tends to move away from the driven wheel when traversing up an incline and can cause traction loss.

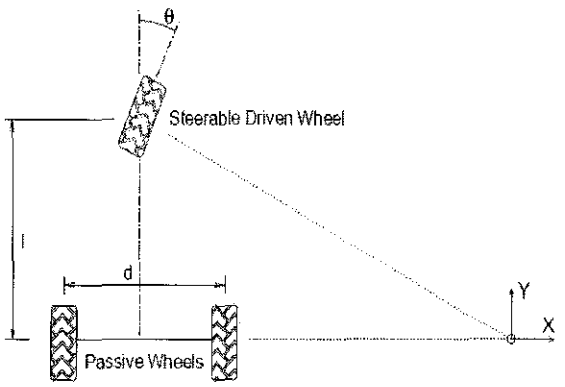

Figure 4: Tricycle-drive configuration [3]

2.3.3 Ackermann steering

These drive configurations have different approach, Ackermann Steering drive consist of separate drive and steering system. Rear wheels are drive system and front wheels are steering system, this allows the driven and steering can be control in separate way.

2.4 Control System

A good control system is both accurate and precise toward achieving its desired result. Note that accuracy and precision mean different things. Accuracy is the ability to achieve results very close to the desired result. Precision is the ability to get similar results every time.

For the mobile robot control system, there are two control theories that have been used in today control system depending on the application, which are closed loop control and open loop control.

Open loop control is easier to construct and mostly used in application that doesn't need precision and also high accuracy.

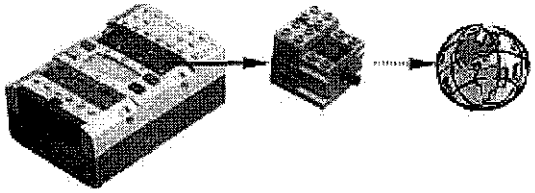

Figure 5: Open loop control [7]

For the closed loop control system, it is more complicated system to construct but it gives more accurate and also more precise data for control of the mobile robot.

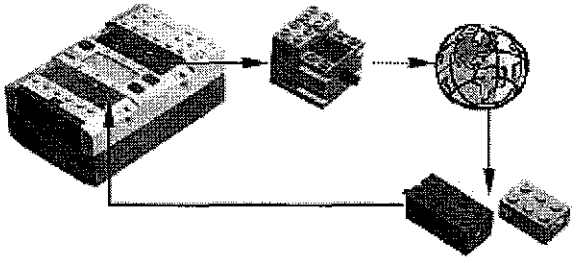

Figure 6: Closed loop control [7]

The closed loop control obtain feedback data from the controller output to be fed in the controller back for the correction if there is any undesired state of data which is didn't achieve the requirement of the system. Because of this feedback data, the closed loop control system is more precise and more accurate control method.

Two main aspects that will be control in the mobile robot are velocity and also steering. These two aspects are the most important control parameter in dead-reckoning method of mobile robot navigation system. To achieve the accuracy of the speed and also steering control, the closed loop control system were used to give

more accurate and precise control. The data of mobile robot speed movement will be fed back to the controller and calculation will be made to determine the accuracy for the mobile robot speed movement and also steering movement, from the data that have been calculate, the controller will execute proper instruction to control the motor of the mobile robot for better result. This process repeatedly going and will create more accurate data.

2.5 Dead-reckoning navigation

Dead-reckoning is the most widely used navigation method for mobile robot positioning. It is well known that dead-reckoning provides good short-term accuracy, is inexpensive, and allows very high sampling rates. However, the fundamental idea of dead-reckoning is the integration of incremental motion information over time, which leads inevitably to the accumulation of errors. Particularly, the accumulation of orientation errors will cause large position errors which increase proportionally with the distance traveled by the robot. Despite these limitations, most researchers agree that dead-reckoning is an important part of a robot navigation system, and that navigation tasks will be simplified if dead-reckoning accuracy can be improved [6].

Dead-reckoning is used in almost all mobile robots, for various reasons:

1. Dead-reckoning data can be fused with absolute position measurements to provide better and more reliable position estimation

2. Dead-reckoning can be used in-between absolute position updates with landmarks. Given a required positioning accuracy, increased accuracy in dead-reckoning allows for less frequent absolute position updates. As a result, fewer landmarks are needed for a given travel distance.

3. Many mapping and landmark matching algorithms assume that the robot can maintain its position well enough to allow the robot to look for landmarks in a limited area, and to match features in that limited area to achieve short processing time and to improve matching correctness

2.6 Virtual Grid

This virtual grid actually is creating to make the testing for the movement of autonomous mobile robot easier. This plane actually consist of drawing of area with specifies length and width. The purpose of this drawing is for easier to calculate the distance of the robot has travel from the original or starting point. When the distances have been determine, the movement of the mobile robot can be control by controlling the encoder of the motor. This enable user to specify at which distance the mobile robot should turn; left or right, by specify the value of the encoder. Below is the example of the virtual grid navigation map:

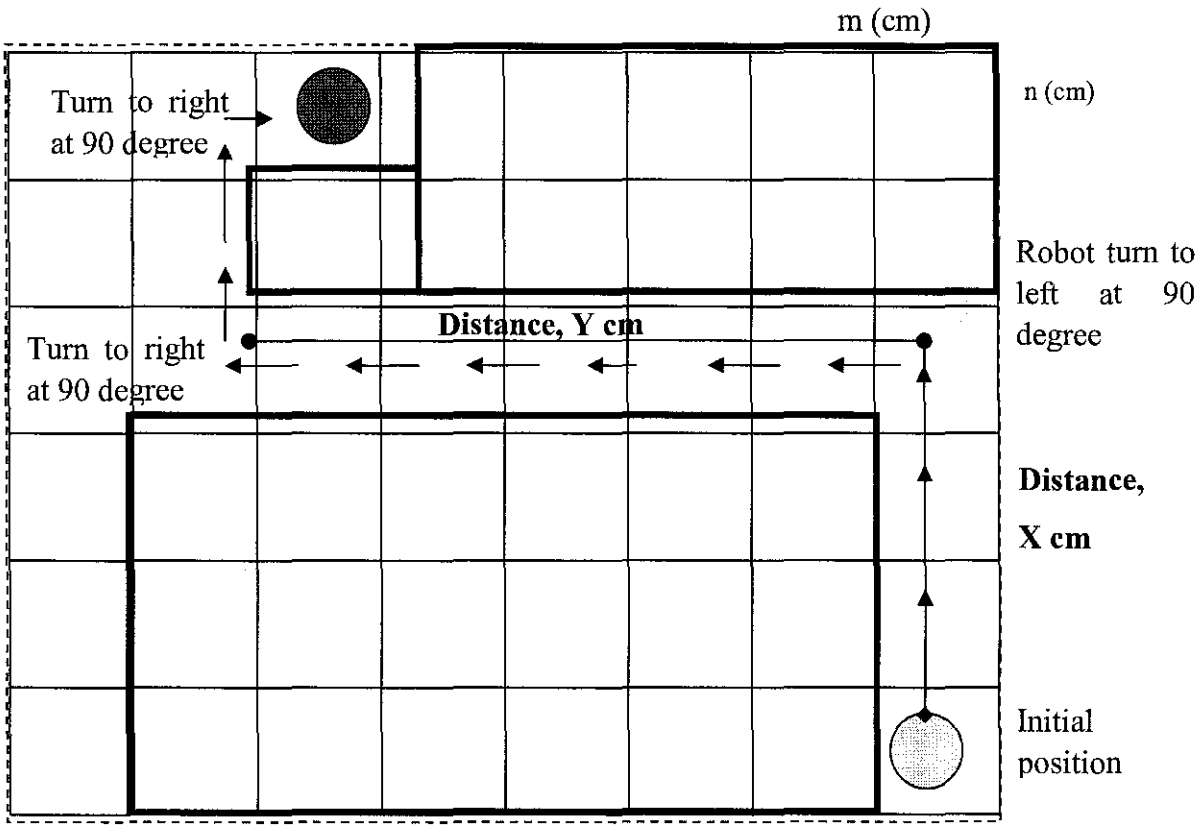

Figure 7: Virtual Grid layout

2.7 Microcontroller

A microcontroller is a small computer and it can only perform simple task.

Microcontroller consists of:

- Processor that executes programs. Processor execute program digitally. All instruction given to the processor should be in digital form.

- Program Memory to store the program that has been compiled successfully by the programmer.
- RAM (random-access memory) to store "variables."
- IO Port to connect sensor, keypad, LED, Relay and so on.
- Timer to count the time to execute some process.

For this project, PIC that will be use is microcontroller PIC 16f877 which is having many function and easy to setup. Using this PIC also enable the user to coding using the C programming using variant of gcc. This chip also was designed to operate with signal processing capabilities and also have several advantages toward this project.

To program the Microcontroller, 3 thing need to be done:

1. Write the program using an integrated development environment like MIGCC.exe or MPLab
2. Compile the text using a compiler program
3. Download the compiled program into a chip

The sample program to control the hardware is shown below:

```
main( )
{
outputHigh (PORTB,0);          //turn on the LED (outputHigh)
delayMs (500);                 //wait 500 mini-second
outputLow (PORTB,0);           //turn off the LED (outputLow)
delayMs (1000);                //wait 1000 mini-second
```

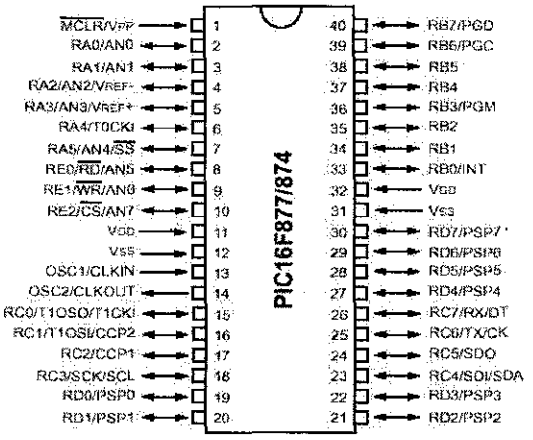

Figure 8: PIC16f877/874 layout [9]

2.8 Comparison between internal and external sensor

Mobile robot is not like manipulator robot which can travel through without the help of sensor and not involved with collision that the mobile robot not aware of. For mobile robot to detect its position and orientation directly with respect to its environment, we will call this sensor "external sensor". Sensors for sensing wheel rotation and steering angle will be called "internal sensors"[8]. The method of controlling the movement of the mobile robot is bit complicated and need high accuracy of calculation for the movement. For example, using the coding only for the movement control will create a fixed movement only and not very flexible to various condition of operation. Meanwhile using sensor for example, can create more freedom movement for the mobile robot which is more flexible. By using the internal sensor (motor steering and angle displacement), it is good when to control in open place, for experiment movement and easy to construct, this method however need a very detail derivation of calculation and accurate data for the movement which is very important. The external sensor such as IR sensor, vision sensor and other is bit expensive but it help a lot in create a linear movement of the mobile robot with minimal error compare to the internal sensor. But the best solution is combining these two sensor together can create a better mobile robot in term of movement linearity.

CHAPTER 3
DESIGN

3.1 Mobile robot design

Figure 9 shows the isometric view of the mobile robot that has been design for the project.

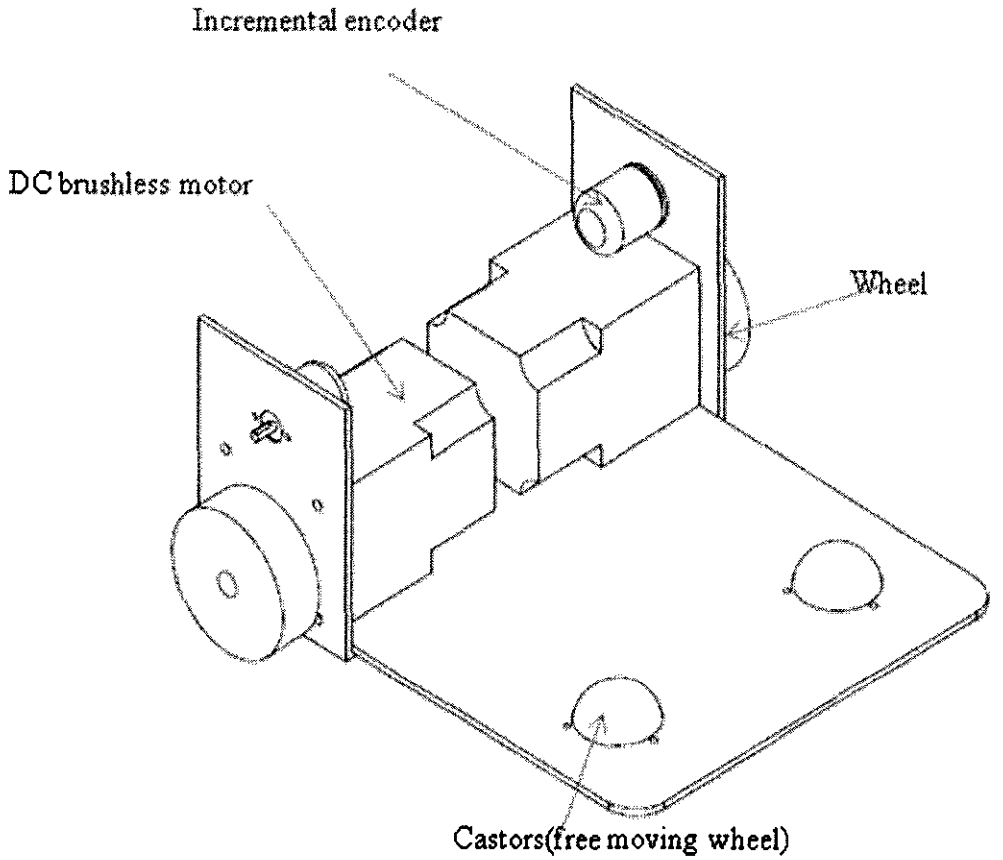

Figure 9: Isometric view of mobile robot

The construction of the mobile robot consists of a pair of wheels, 2 set of brushless DC motor, two castors and also two incremental encoders which both on the left wheel and also on the right wheel.

The Figure 10 shows the upper view of the mobile robot

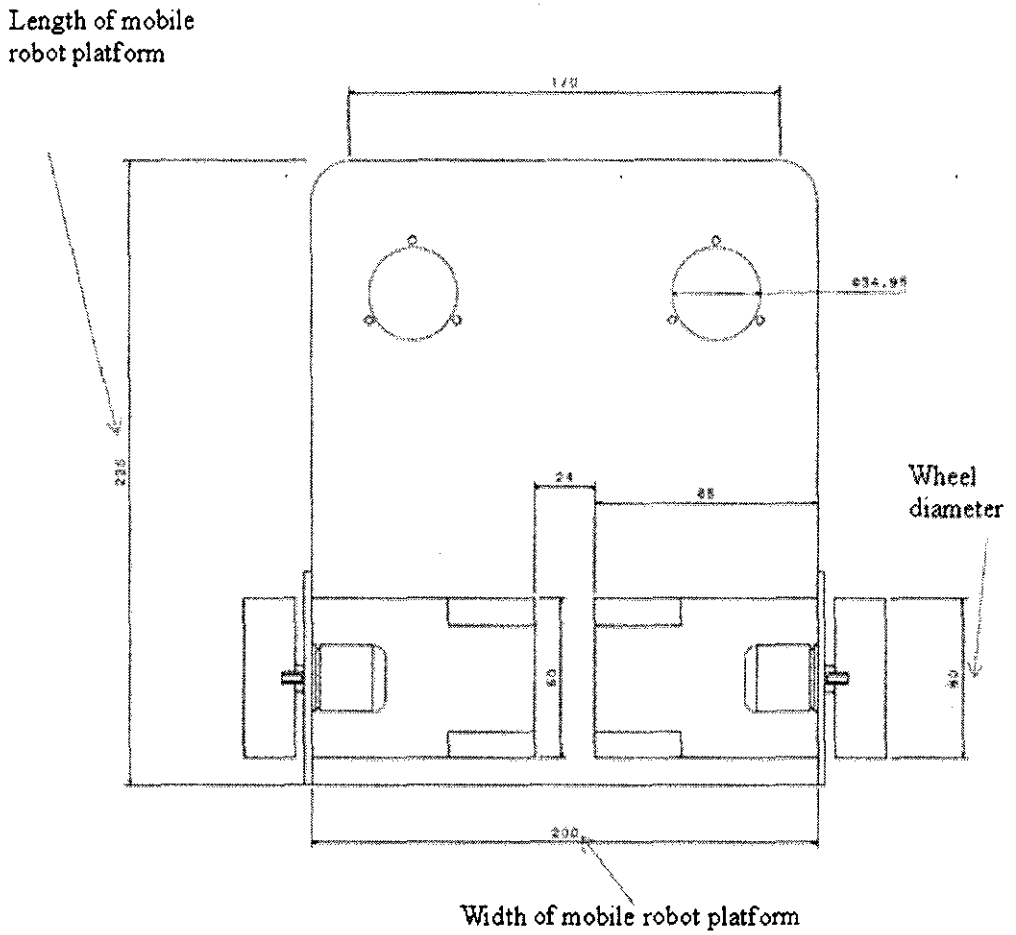

Figure 10: Upper view of mobile robot

Wheel which in 60mm diameter was connected to the DC brushless motor with direct coupling were used. The incremental encoder was mounted parallel with the wheel position. This construction makes the encoder and wheel (motor) easier to be connected directly with the rubber chain. This allow direct output from wheel rotation be recorded by the encoder.

Figure 11 and figure 12 shows the design side view and also the front view of the mobile robot.

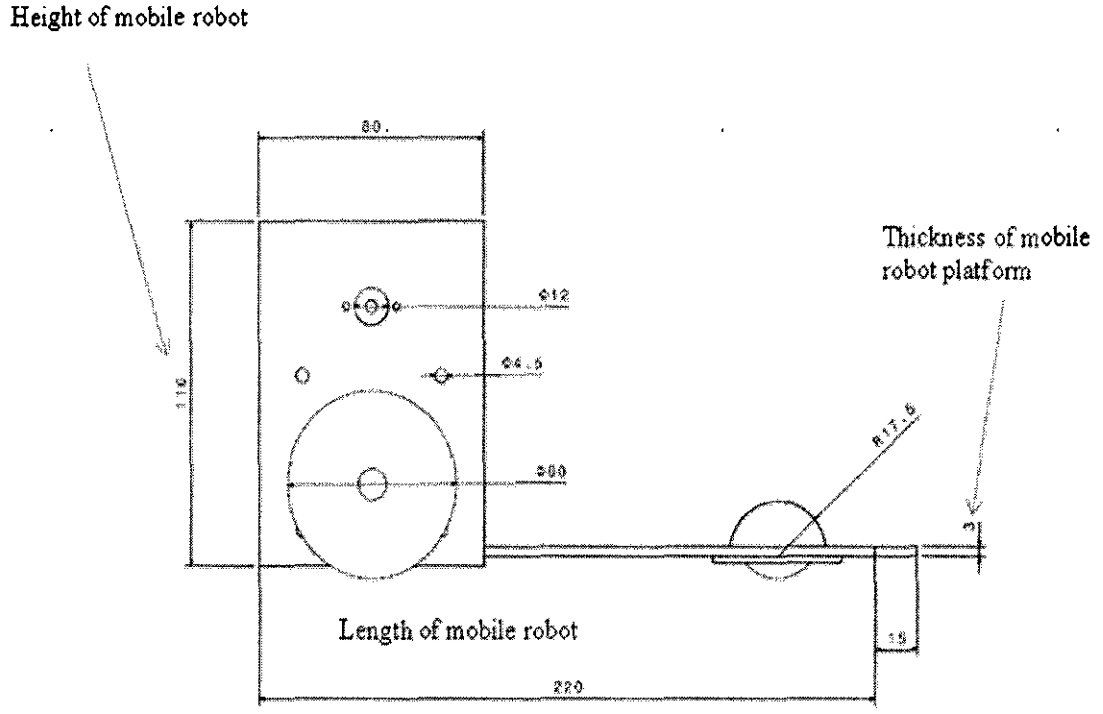

Figure 11: Side view of mobile robot

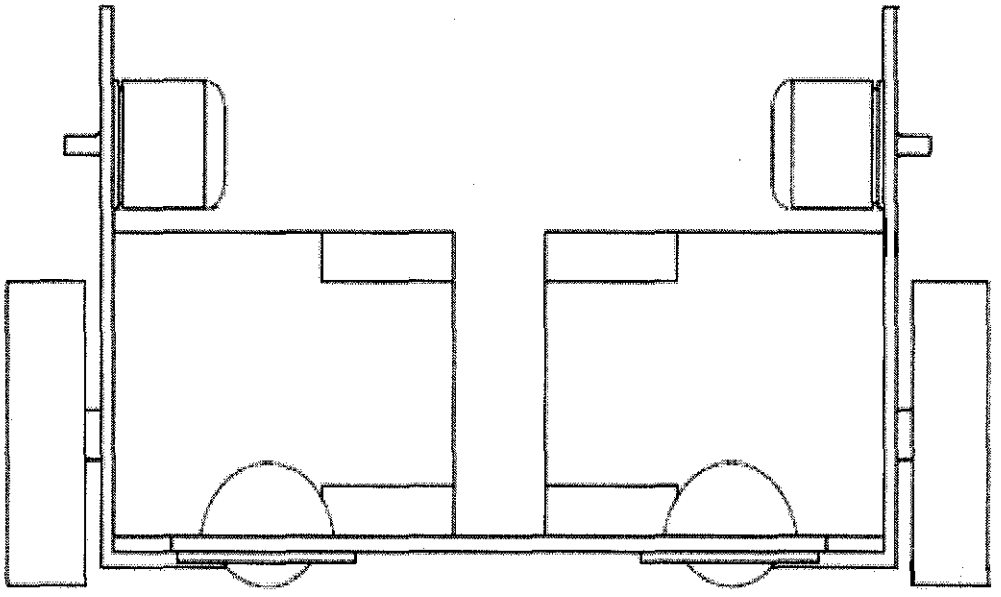

Figure 12: Front view of mobile robot

Two DC brushless motors were use for this design. These motors drive the wheels with specific programmed instruction. The program consists of several

instructions to control the movement of the mobile robot. Both motors were place opposite to each other and with specific gap between for easily to mount and also to balance the robot.

Two castors or free moving wheels were place in front spot of the mobile robot. Each of the castors is mounted in parallel position; this method is to reduce the error of uneven displacement for the mobile robot movement while turning either left or right.

3.2 Control subsystems

3.2.1 AR40B board

In the control subsystem, the controller and also other additional part for controller will be discussed. For the controller, AR40B board have been used as a main board controller, as in Figure 13, there are description on AR40B which have been used for this project.

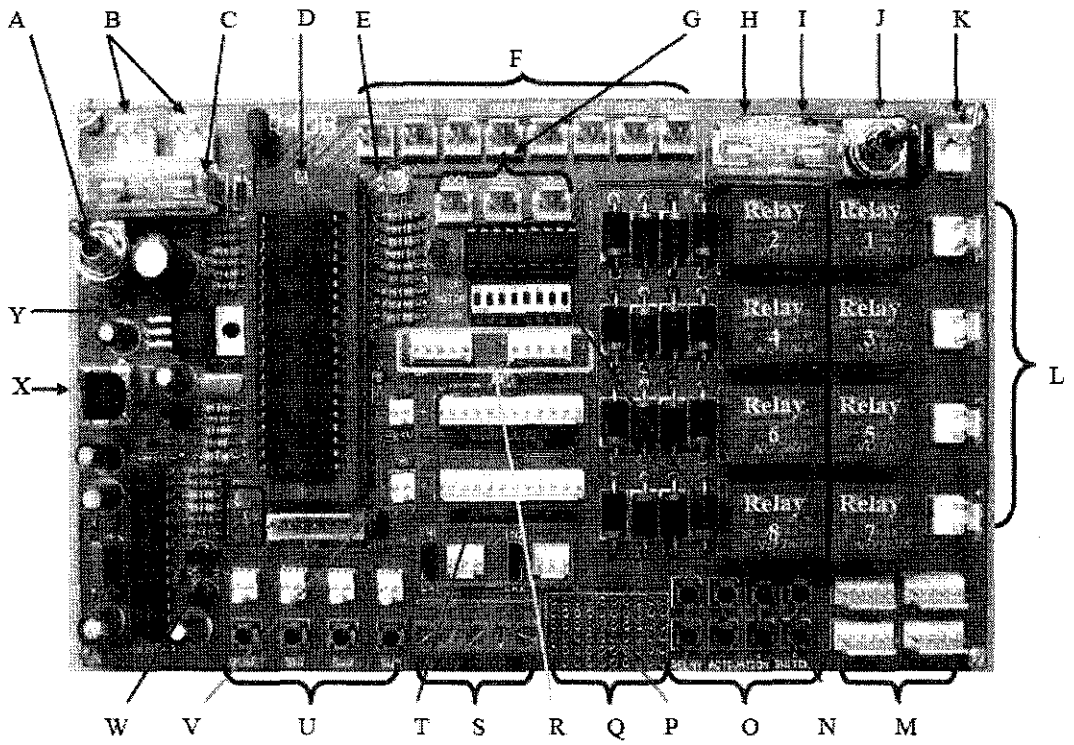

Figure 13: AR40B board layout [10]

Label	Function	Label	Function
A	Switch for main power supply	N	Internal/external relay switch
B	Connector for main power supply	O	Internal relay test switch
C	6 Amp fuse for main power supply	P	Connector for external encoder
D	ZIF socket for 40 pins PIC	Q	Expansion zone
E	User program indicator LED	R	Connector for brush motor driver
F	Connector for sensor/digital input	S	Expansion terminal block
G	Connector for analog input	T	Connector for brushless motor (control pin)
H	10 Amp fuse for motor power supply	U	Reset button and switches
I	Motor power supply indicator LED	V	Connector for brushless motor (power)
J	Switch for motor power supply	W	On board programmer indicator LEDs
K	Connector for motor power supply	X	Connector for USB
L	Connector for brush motor	Y	Main power supply indicator LED
M	Connector for external driver		

Figure 14: AR40B board connector function [10]

AR40B board was specifically design for the autonomous robot either for the mobile robot or the other robots. In the Figure 14 show the description of each section on the AR40B board layout which is shown in Figure 13.

3.2.2 Power supply

The power supplies for the controller board that have been used were 2 lead acid batteries which are 12V. Since 2 brushless motors were used in the project, the board required 24V of power supply to control 2 motors. The power supply connected to the connector of the main power supply at Label A in the Figure 13. The construction of the mobile robot is simple and moderately small, so the batteries that have been used were small enough in size and weight.

3.2.3 Brushless motor

AXH brushless motor in Figure 15 from Oriental Motor was used in the project. It has 30W power rating.

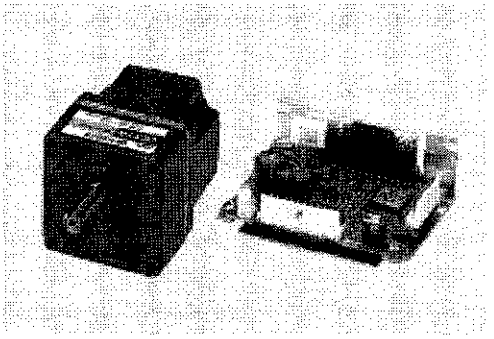

Figure 15: AXH brushless motor with driver [11]

3.3 Navigation Algorithm

Time and distance data collection is easy to evaluate. This technique of data collection and calculation will be used to determine the actual position of the mobile robot and the future movement of the mobile robot in the specific area of experiment. To get the better result for better planning behavior of the mobile robot navigation, the method of calculation in angular displacement of wheel for mobile robot need to accurate and constant. This will produce the desire data to reconstruct and program the instruction for the mobile robot reaching its goal.

The important aspect of controlling the autonomous mobile robot is how to set the algorithm for it to move through the area. The algorithm that has used to control the robot from spot A to spot B can be described Figure 16:

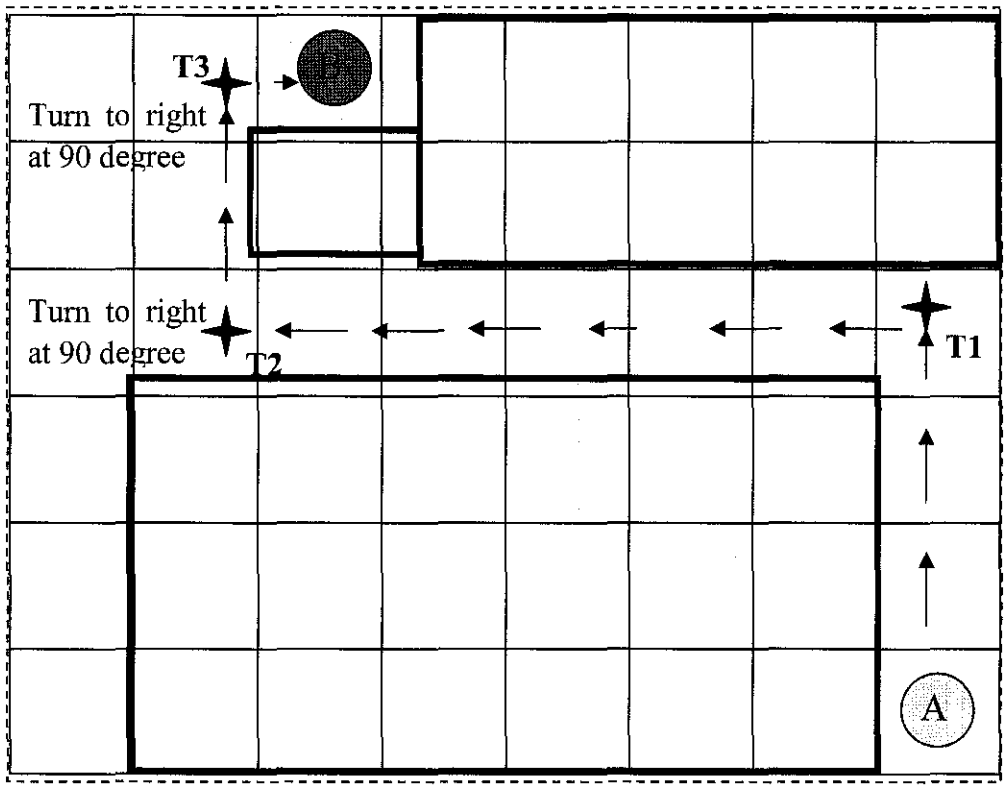

Figure 16: Layout Design

At point A:

- The speed of the left and right motor is set and control. Both motor speed is equal, to ensure that the robot will move forward in straight direction.
- The speed of the motor and the time it is running need to be accurate to make sure the distance its travel is within the desire distance that has been set.

At point **T1:**

- When the robot reaches this point, it will stop for a while.
- Then it will turn to the left by controlling the movement and speed of the right and left motor. To turn 90 degree to the left, the left motor will rotate backward by half of the speed and the right motor will rotate forward by half speed. This will make the robot to turn 90 degree to the left
- Then after checking the position, the robot move forward by moving both of the motors with the same speed for some period of time before stop at point T2.

At point **T2** and **T3**:

- At these points, the robot will make the turn to right by 90 degree before moving forward.
- Controlling the robot turn to the right by 90 degree same as turn to left but instead of moving the left motor reverse, the right motor should move reverse and the left motor rotate forward. The speed of the motor is same and the time of motor movement also must be in the same time.

At point **B**:

- At this point, the motor reach the destination.
- It finishes the task and stops the motor movement.

The control navigation for the movement of the mobile robot using the encoder and motor can be show in the Figure 17:

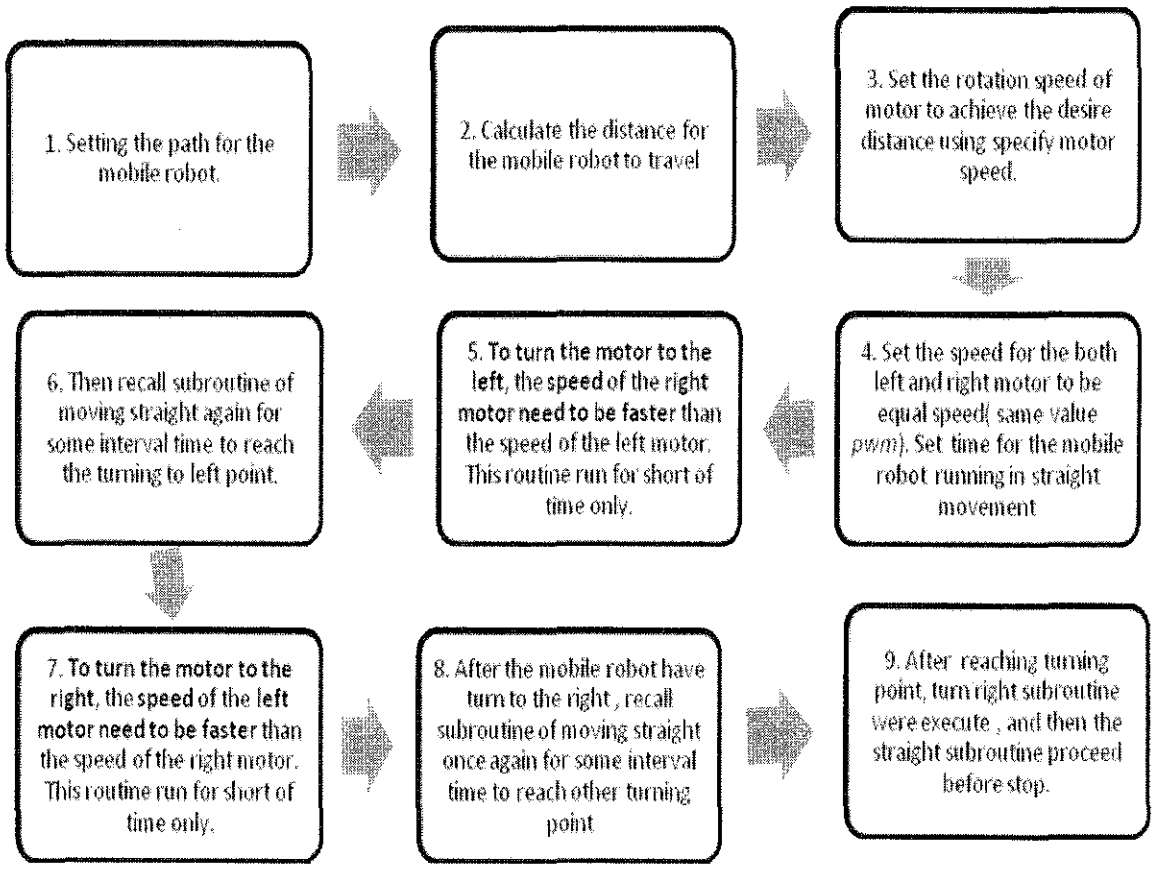

Figure 17: Flow chart of the navigation algorithm

CHAPTER 4
RESULT AND DISCUSSION

In this chapter, the C language programmings of the mobile robot navigation algorithm were discussed.

The mobile robot only use two wheels for the movement, to support the front of the robot, the castor have been use. Using the castor is easy to setup and can make the robot move feely any direction without need to control it. The smoothness of the castor's ball rolling is the factor that also affects the result of the experiment. If the ball's rolling is so rough, this will affect the movement of the mobile robot since so much resistance and also will create the movement impairing for the mobile robot. When this kind of problem occur, the accuracy of the movement can't be fulfill and create undesired result.

4.1 Assembly of the mobile robot

In this subtopic, the assembly of the mobile robot will be discussed regarding the actual assembly of mobile robot and also the design of the mobile robot. There were some difference between the actual and also the design that have come out during early stage. This is due to budget constraint on the mobile robot construction. The assembly process has been done within 4 week time including the body fabricating, controller mounting, additional part sourcing and others.

The Figure 18 shows the actual mobile robot construction.

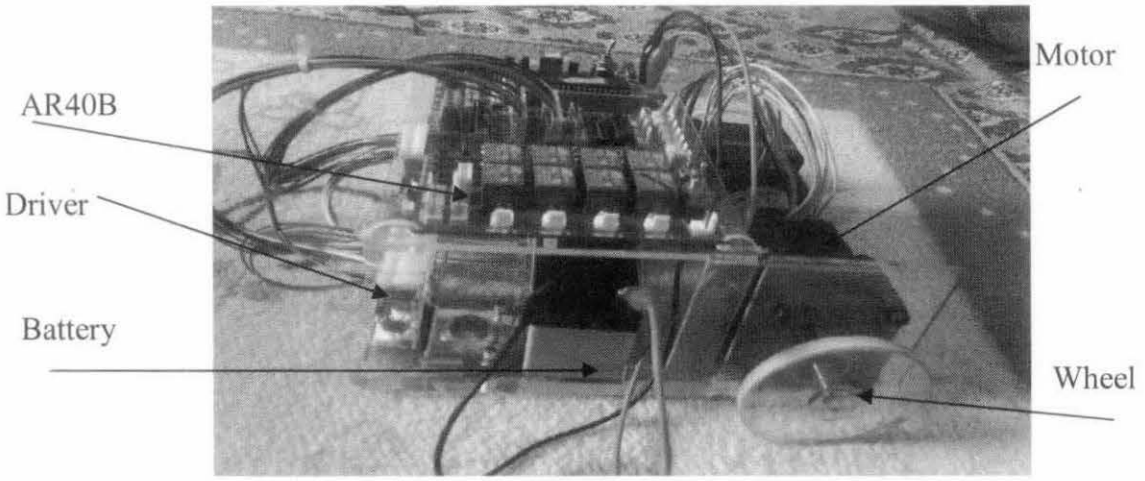

Figure 18: Actual assembly of mobile robot

From the figure 18 above, it shows that there is no incremental encoders were mounted on the mobile robot as in the design figure. This is due to the cost constraint of the project. Each encoder is expensive compare to the budget that has been allocated for the project, so instead using the external incremental encoder; internal encoder was used to calculate the distance travel for the mobile robot.

For the wheel mounting to the motor, direct mount was used as show in Figure 19.

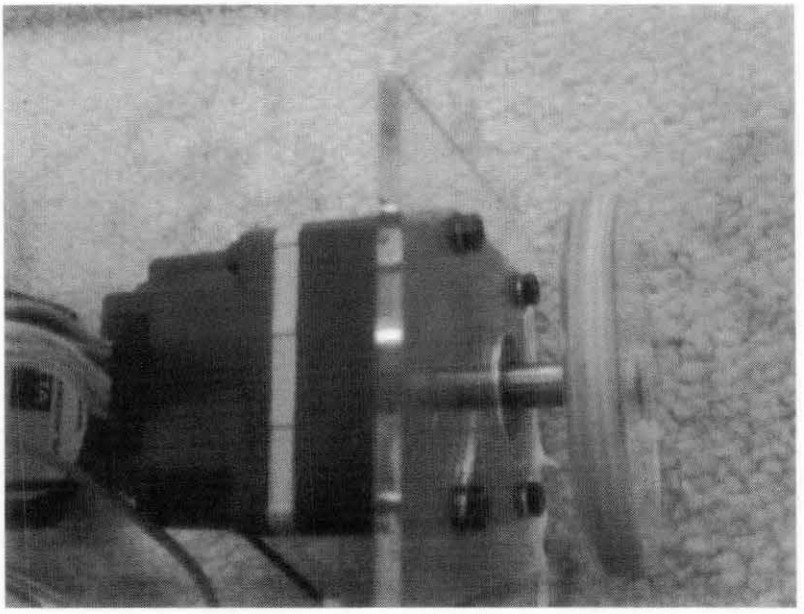

Figure 19: Wheel mounted to the motor

Each of the motor needs a driver to control the motor movement, in Figure 20 show the driver of the motor that have been used.

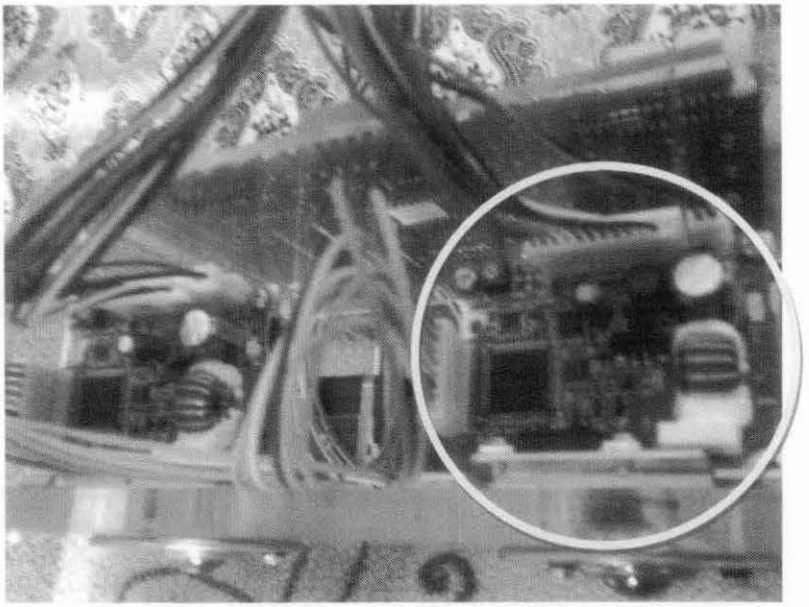

Figure 20: Motor driver

AR40B board were mount on second base of the mobile robot because the lower base of the mobile robot was placed with the two batteries and proper mounting place needed for the AR40B board to avoid the problem with the packaging of the wire and also to avoid signal interfere with the brushless motor. The AR40B board was mounted can be shows in Figure 21.

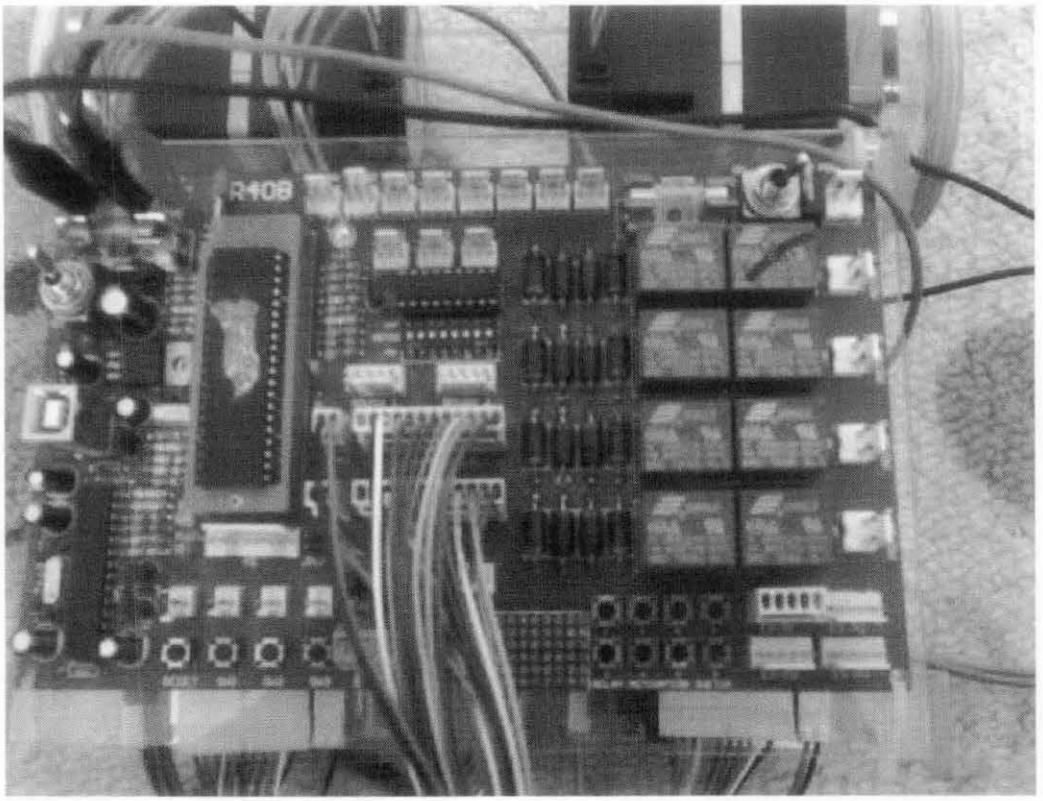

Figure 21: Mounting of AR40B board on second base

4.2 Programming of the mobile robot navigation

There were several types of coding to control the movement of the mobile robot. For the wheel control application, this is the coding that been used to control the motor rotation of the mobile robot.

This subroutine instructs the controller to control the movement of the left and right motor. *Pwm1* refer to right motor signal input, and *pwm2* is the signal input for the left motor. By adjusting the value for the motor signal, the movement of the motor can be controlled. From above code, both signal for left and right motors were same, this cause the mobile robot move straight since there is not different between left and right motor rotation.

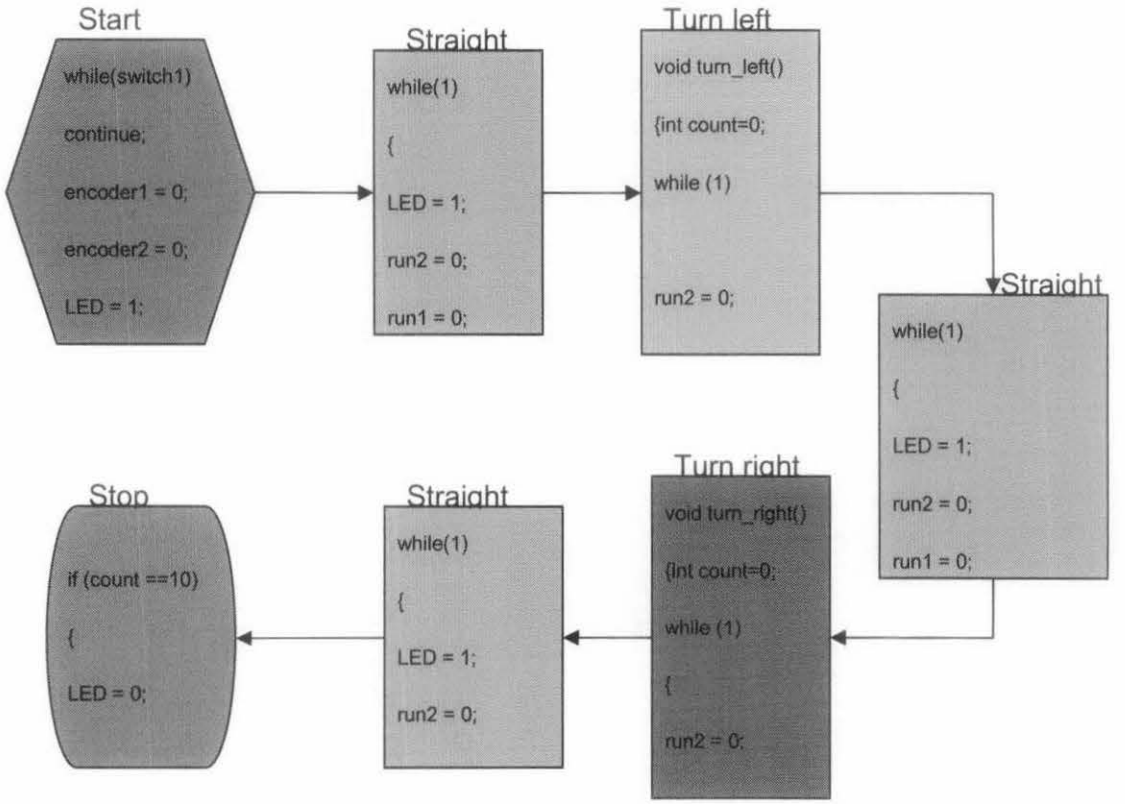

Figure 22: Programming flowchart

4.2.1 Moving straight

For the initial, the straight movement instruction was instructed to the mobile robot.

void straight ()

{int count=0;

```
LED = 1;
run2 = 0;
run1 = 0;
pwm1 = 80;
pwm2 = 80;
ccw1 = 0;
ccw2 = 1;
delay (20000);
```

From the programming code, it shows that both *pwm1* and *pwm2* were equal; this is the main instruction to make the mobile robot move straight. Both of the motor were moving in opposite direction, which can only make the mobile robot move forward. This instruction run for several second before it end.

4.2.2 Turning left

After reaching the turning point, the microcontroller will recall the routine for the turning to the left movement. This was executed after the moving straight subroutine was done.

```
void turn_left()
{
ccw1=0;
run1=0;     //enabling BLM
ccw2=0;
run2=0;     //motor not brake
LED = 0;
pwm1 = 70;
pwm2 = 50;
}
```

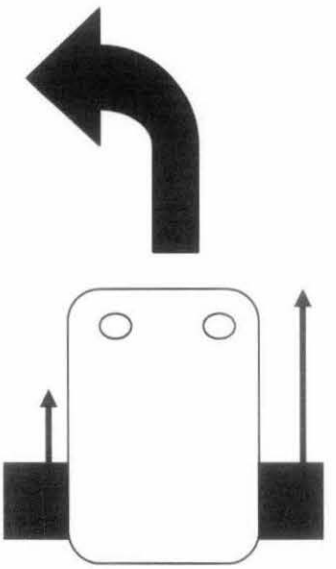

Figure 23: Mobile robot turns right

For mobile robot to turn to the right or left, the signal of the motor need to be adjusts accurately to prevent over-turn. Increasing the value of the *pwm1* cause the rotation of the left motor turn faster and by decreasing the value for *pwm2* causes the right motor rotation became slower, this instruction cause the mobile robot turn to the left.

4.2.3 Turning right

After turning left, the microcontroller recall the instruction for the moving straight and stop at turning point where *void turning_right* need to be execute for the mobile robot to turn to the right.

```
void turn_right()
{
ccw1=1;
run1=0;     //enabling BLM
ccw2=0;
run2=0;     //motor not brake
LED = 0;
pwm1 =50;
pwm2 = 70;
}
```

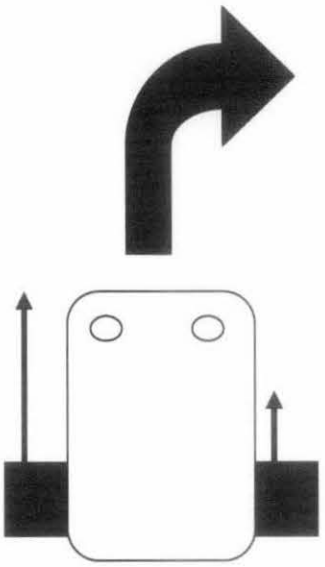

Figure 24: Mobile robot turns left

Programming for subroutine of mobile robot turning to the right is different from the subroutine for turning to the left. But there were just minor change in the programming regarding the *pwm1* and *pwm2* value. By changing the value of the *pwm*, it changes the mobile robot movement to the right.

4.3 Discussion

- Testing has been done only to test the validation of the navigation algorithm in C language. For the further testing such as accuracy, precision and other testing aspect need to be conduct in different type of testing method.
- The result shows that it is easier to test and also to debug the navigation system for mobile robot with using the C language as a command instruction.
- Testing for accuracy of control system using C language required more detail set of instruction rather than direct command. With more detail instruction will give the mobile robot step by step movement testing.
- Accuracy of movement for mobile robot movement greatly influence by type of area, component used (wheel, etc.) and others.
- Design for mobile robot need to be more robust if it were used in uneven layout of floor.
- Suitable test area for different testing type.

CHAPTER 5
CONCLUSION

5.1 Conclusion

Implementation of algorithm navigation for mobile robot using internal sensor were easier if accuracy of the sensors were accurate. This allows the robot to more precisely control the position and navigate along the path. Although much effort have been exerted to do research on these project, the information on how to control autonomous mobile robot using the internal sensor still not inadequate and the control navigation of autonomous mobile robot using the internal sensor need to be more flexible and precisely control on complex area or space.

From the studies have been done, the implementation of the algorithm for navigation system for mobile robot need to be more detail to achieve the objective of the study. Implementation of close-loop position control and navigation algorithm using the C language was focusing only on controlling the motor. The project also reveals some weakness and also advantage of using this technique compare to other algorithm technique. This algorithm rely too much on the wheel rotation, this needs more improvement for more accurate data feedback to be implement on more complicated and complex task.

The construction of the mobile robot is simple and without using the external sensor invite several problems that actually can be avoid such as mobile robot dynamic movement, and also problem regarding part that have been used such as wheel type, wheel position and others. Due to this, to make much more complicated algorithm approach is not limited by those constrain.

5.2 Recommendation

As a recommendation, in term of robot construction, the external sensor was needed to make the navigation algorithm more robust and versatile in navigating the mobile robot along the unknown area. For the C language programming, to make the senseless mobile robot more independent and autonomous, the algorithm construction need to more detail and more accurate archive the target point.

REFERENCES

[1] Towards Principled Experimental Study of Autonomous Mobile Robots (Appears in ISER95.To appear in Autonomous Robots.) Erann Gat

[2] http://en.wikipedia.org/wiki/Autonomous_robot

[3] H. R. Everett, "*Sensors for Mobile Robots.*"A. K. Peters, Ltd., Wellesley,Spring ,1995.

[4] Incremental optical encoder disk <http://mitros.org/p/projects/encoder/>

[5] Incremental optical encoder signal schematic <http://www.engr.colostate.edu/~dga/mechatronics/figures/9-16.gif>

[6] L. Feng1, J. Borenstein2, and H. R. Everett3, "*Sensors and Methods for Autonomous Mobile Robot Positioning"* December 1994

[7] Control theory

<http://www.colincampbellonline.com/thesis/control_theory.html>

[8] Masahnni Takano, Shunichi Odaka, Takahiro Tsukishinia, Ken Sasaki

'STUDY ON MOBILE ROBOT NAVIGATION CONTROL BY INTERNAL AND EXTERNAL SENSOR DATA WITH ULTRASONIC SENSOR' *IEEE/RSJ International Workshop on Intelligent Robots and Systems '89, Sep. 4-6, 1989,Tsukuba, Japan*

[9] Microchip Technology Inc, published on 2003 "*PIC16F877XA data sheet*", pg 5.

[10] Cytron Technology Sdn. Bhd, published on Nov 2007, "*AR40B autonomous robot control board instruction manual*" Ver. 1.3

[11] Oriental General Motor Catalog, published on 2003/2004 " *Brushless Dc Motor Systems, AXH series*"

APPENDIX A

AR40B BOARD CODE

```
#include <pic.h>

#define run1          RC5          //
#define ccw1          RC4          //
#define pwm1          CCPR1L       //
#define encoder1      TMR0 //
#define run2          RE2          //
#define ccw2          RE1          //
#define pwm2          CCPR2L       //
#define encoder2      TMR1L        //
#define switch1             RE0          //
#define switch2             RB6          //
#define switch3             RB7          //
#define ADC_DATA ADRESH       //
#define sen1          RB0          //
#define sen2          RB1          //
#define sen3          RB2          //
#define sen4          RC3          //
#define sen5          RB4          //
#define sen6          RC6          //
#define sen7          RA2          //
#define sen8          RA5          //
#define relay1        RD0          //
#define relay2        RD1          //
#define relay3        RD2          //
#define relay4        RD3          //
#define relay5        RD4          //
#define relay6        RD5          //
#define relay7        RD6          //
#define relay8        RD7          //
#define LED           RB3          //
#define GPO1          RB5          //
#define GPO2          RC7          //
//================================== AR40B PIN ASSIGNMENT
==========================
/*;PIN Des        func            PIN   Des       func
;1      MCLR       Reset button    40    RB7       SW3
;2      RA0        ADC1            39    RB6       SW2
;3      RA1        ADC2            38    RB5       GPO1
;4      RA2        SEN7            37    RB4       SEN 5
;5      RA3        ADC3            36    RB3       LED
;6      RA4        ENCODER2        35    RB2       SEN 3
;7      RA5        SEN8            34    RB1       SEN 2
;8      RE0        SW1             33    RB0       SEN 1
;9      RE1        CCW1            32    VDD       5V
```

```
;10     RE2         RUN1            31      VSS         GND
;11     VDD         5V              30      RD7         Relay 8
;12     VSS         GND             29      RD6         Relay 7
;13     OSC1        Crystal         28      RD5         Relay 6
;14     OSC2        Crystal         27      RD4         Relay 5
;15     RC0         ENCODER2        26      RC7         GPO2
;16     RC1         PWM1            25      RC6         SEN6
;17     RC2         PWM2            24      RC5         RUN2
;18     RC3         SEN4            23      RC4         CCW2
;19     RD0         Relay 1         22      RD3         Relay 4
;20     RD1         Relay 2         21      RD2         Relay 3
*/
void init_io ( void ) ;
void init_serial ( void ) ;
void setup_brushless ( void ) ;
void setup_brushed ( void ) ;
void setup_relays ( void ) ;
void setup_adc ( void ) ;
unsigned char read_adc ( unsigned char channel) ;
void setup_pwm(void);
void setup_encoder(void);
void delay (unsigned long i);

// subroutines

void init_io ( void )
{
        TRISA = 0b11111111;
        TRISB = 0b11010111;
        TRISC = 0b01001001;
        TRISD = 0b00000000;
        TRISE = 0b00000001;
        PORTA = 0b00000000;
        PORTB = 0b00000000;
        PORTC = 0b00110000;
        PORTD = 0b00000000;
        PORTE = 0b00000110;
}

void setup_brushless ( void )
{
        run1 = 1;       //stop all DC brushless motor
        ccw1 = 1;
        pwm1 = 0;       //speed = 0
        run2 = 1;       //brake motor
        ccw2 = 0;
        pwm2 = 0;
}
```

```
void setup_brushed ( void )
{
        run1 = 0;       //stop all DC brushed motor
        ccw1 = 0;
        pwm1 = 0;
        run2 = 0;
        ccw2 = 0;
        pwm2 = 0;
}
void setup_relays ( void )
{
        PORTD = 0b00000000;
}
void setup_adc ( void )
{
        ADCON0 = 0b10000000;    //Left justified, RA0, RA1 and RA3 as ADC
input
        ADCON1 = 0b01000100;    //Fosc/32, channel 0, ADC not active, ADC off
}
unsigned char read_adc ( unsigned char channel )
{
        switch (channel)
        {
                case 1:
                ADCON0 = 0b10000001;
                break;
                case 2:
                ADCON0 = 0b10001001;
                break;
                case 3:
                ADCON0 = 0b10011001;
                break;
                default:
                ADCON0 = 0b10000000;
        }

        delay(5);                       // delay for a while
        ADGO = 1;                       // start conversion
        while (ADGO) continue;
        ADON = 0;                       // Off ADC module
        return ADRESH;                  // return ADC result

}
void setup_pwm(void)
{
        PR2 = 255;
        TMR2ON = 1;//enable timer 2
        pwm1 = 0;
        pwm2 = 0;
        CCP1CON = 0b00111100;
        CCP2CON = 0b00111100;}
```

```
void setup_encoder(void)
{
        T0CS = 1;               //timer0 as counter
        T0SE = 0;               //increment on rising edge
        PSA = 1;                //assign prescale to watch dog, not timer0
        encoder1 = 0;           //clear timer0
        T1CON = 0b00000111;
                                        //no prescale
                                        //osc off
                                        //no sync
                                        //external pulse from T1CKI
                                        //enable timer1
        TMR1H = 0;
        TMR1L = 0;              //clear timer1
}
void delay (unsigned long i)
{
        for (; i>0; i-=1);
}
```

APPENDIX B

SUBROUTINE CODE FOR THE BRUSHLESS MOTOR

```
#include "AR40B.h"
#include <pic.h>

__CONFIG ( 0x3F32 ) ;
unsigned char temp;
void turn_left();
void turn_right();
void straight();
void straight2();
void main (void)
{
        int count=0;

        init_io();
        setup_brushless();
        setup_pwm();
        setup_relays();
        setup_encoder();
        setup_adc();
        LED = 0;


        while(switch1) continue;
        encoder1 = 0;
        encoder2 = 0;
        LED = 1;
                        run2 = 0;
                        run1 = 0;
                        pwm1 = 80;
                        pwm2 = 80;
                        ccw1 = 0;
                        ccw2 = 1;
        delay(20000);

        while(1)
        {
                        LED = 1;
                        run2 = 0;
                        run1 = 0;
                        pwm1 = 120;
                        pwm2 = 120;
                        ccw1 = 0;
                        ccw2 = 1;
```

```
                    if (encoder1 >= 100)
                            {
                            count++;
                            encoder1=0;
                            }

                    if (count ==8)
                            {
                            LED = 0;
                            turn_left();
                            while(1)continue;
                            }

        }
}

void turn_left()
{int count=0;
                while (1)
                {
                        run2 = 0;
                        run1 = 0;      // enable both brushless motor
                        pwm1 = 70;    //
                        pwm2 = 50;
                        ccw1 = 0;
                        ccw2 = 0;
                        delay (10000);
                        pwm1 = 120;
                        pwm2 = 120;
                        if (encoder1 >= 100)
                        {
                        count++;
                        encoder1=0;
                        }
                        if (count ==3)
                        {
                        LED = 0;
                        straight();
                        //while(1)continue;
                        }
                        }

}

void straight ()
{int count=0;

                        LED = 1;
                        run2 = 0;
                        run1 = 0;
                        pwm1 = 80;
```

```
                pwm2 = 80;
                ccw1 = 0;
                ccw2 = 1;
        delay(20000);

        while(1)
        {
                LED = 1;
                run2 = 0;
                run1 = 0;
                pwm1 = 150;
                pwm2 = 150;
                ccw1 = 0;
                ccw2 = 1;


                if (encoder1 >= 100)
                {
                count++;
                encoder1=0;
                }
                if (count ==15)
                {
                LED = 0;
                run2 = 0;
                run1 = 0;
                delay (20000);
                turn_right();
                while(1)continue;
                }

        }
}

void turn_right()
{int count=0;
                while (1)
                {
                        run2 = 0;
                        run1 = 0;
                        pwm1 = 50;
                        pwm2 = 70;
                        ccw1 = 1;
                        ccw2 = 1;
                        delay (10000);
                        pwm1 = 150;
                        pwm2 = 150;
                        if (encoder1 >= 100)
                        {
                        count++;
                        encoder1=0;
```

```
                    }
                    if (count ==3)
                    {
                    LED = 0;
                    straight2();
                    //while(1)continue;
                    }
                    }

}

void straight2 ()
{int count=0;

                    LED = 1;
                    run2 = 0;
                    run1 = 0;
                    pwm1 = 80;
                    pwm2 = 80;
                    ccw1 = 0;
                    ccw2 = 1;
        delay(20000);

        while(1)
        {
                    LED = 1;
                    run2 = 0;
                    run1 = 0;
                    pwm1 = 120;
                    pwm2 = 120;
                    ccw1 = 0;
                    ccw2 = 1;


                    if (encoder1 >= 100)
                    {
                    count++;
                    encoder1=0;
                    }
                    if (count ==10)
                    {
                    LED = 0;
                    run2 = 0;
                    run1 = 0;
                    delay (20000);
                    run2 = 1;
                    run1 = 1;
                    while(1)continue;
                    }

        }
```