

DEVELOPMENT OF STRAIGHT LINE ROBOT MOVEMENT

CHING WAI HOONG

UNIVERSITI MALAYSIA PAHANG

UNIVERSITI MALAYSIA PAHANG

BORANG PENGESAHAN STATUS TESIS♦

JUDUL: **DEVELOPMENT OF STRAIGHT LINE ROBOT
MOVEMENT**

SESI PENGAJIAN: 2011/2012

Saya CHING WAI HOONG (881115-08-5677)
(HURUF BESAR)

mengaku membenarkan tesis (Sarjana Muda/~~Sarjana~~ /~~Doktor Falsafah~~)* ini disimpan di Perpustakaan dengan syarat-syarat kegunaan seperti berikut:

1. Tesis adalah hakmilik Universiti Malaysia Pahang (UMP).
2. Perpustakaan dibenarkan membuat salinan untuk tujuan pengajian sahaja.
3. Perpustakaan dibenarkan membuat salinan tesis ini sebagai bahan pertukaran antara institusi pengajian tinggi.
4. **Sila tandakan (√)

☐

SULIT

(Mengandungi maklumat yang berdarjah keselamatan atau kepentingan Malaysia seperti yang termaktub di dalam AKTA RAHSIA RASMI 1972)

☐

TERHAD

(Mengandungi maklumat TERHAD yang telah ditentukan oleh organisasi/badan di mana penyelidikan dijalankan)

☒

TIDAK TERHAD

Disahkan oleh:

(TANDATANGAN PENULIS)

(TANDATANGAN PENYELIA)

Alamat Tetap:

**C43, KAMPUNG BARU,
35350 TEMOH,
PERAK.**

MR. MOHD RIDUWAN BIN GHAZALI
(Nama Penyelia)

Tarikh: **13 JUNE 2012**

Tarikh: **13 JUNE 2012**

CATATAN: * Potong yang tidak berkenaan.
** Jika tesis ini SULIT atau TERHAD, sila lampirkan surat daripada pihak berkuasa/organisasi berkenaan dengan menyatakan sekali tempoh tesis ini perlu dikelaskan sebagai atau TERHAD.
♦ Tesis dimaksudkan sebagai tesis bagi Ijazah doktor Falsafah dan Sarjana secara Penyelidikan, atau disertasi bagi pengajian secara kerja kursus dan penyelidikan, atau Laporan Projek Sarjana Muda (PSM).

“I hereby acknowledge that the scope and quality of this thesis is qualified for the
award of the Bachelor Degree of Electrical Engineering (Electronics)”

Signature : _____

Name : MR. MOHD RIDUWAN BIN GHAZALI

Date : 13 JUNE 2012

DEVELOPMENT OF STRAIGHT LINE ROBOT MOVEMENT

CHING WAI HOONG

A report submitted in partial fulfillment of the requirements for the award of the
degree of Bachelor of Engineering (Electrical and Electronic)

Faculty of Electrical and Electronic Engineering
University Malaysia Pahang

JUNE 2012

“All the trademark and copyrights use herein are property of their respective owner. References of information from other sources are quoted accordingly; otherwise the information presented in this report is solely work of the author.”

Signature : _____

Author : CHING WAI HOONG

Date : 13 JUNE 2012

To my beloved father, mother and brothers

ACKNOWLEDGEMENTS

Initially, the special thank conveys to my helpful supervisor, Mr. Mohd Riduwan Bin Ghazali, for providing me a good opportunity to have a fantastic final year project. His supervision and support truly help me to accomplish my project and the thesis can be presented within the time given. The co-operation, guidance and encouragement are much indeed appreciated.

I would like to extend my appreciation and deep love to my parents and brothers to support me with their sincere love, time and money fostering me to be educated in university. Their cares and encouragements for guiding me to overwhelm my failure and obstacles encountered from completing this project.

My grateful thanks go to the contribution of my faculty and laboratory technicians which provided me a proper experiment environment to test my project's functionality and operation. I appreciate the person in-charge of Component's Store who provided many essential components to roll my hoop to accomplish my task.

Last but not least, I would like to send to high appreciation to all my friends for those who ever help me to complete my project. I sincerely hope that my knowledge and experiences obtained from this project can be shared and beneficial to everyone and society in future. Thank you.

ABSTRACT

This project focuses on the development of straight line movement for a four-wheeled mobile robot. In this project, a DC gear motor is chosen as motion control for two driving wheels and the direction of robot will be controlled by servo motor to two steering wheels. PIC is selected as the brain board controller due to react and respond to the data received from Digital Compass Module to identify and figure out desired position. The implementation of internal PID algorithm is essentially used to restore the system to desired set-point position. Application of Digital Compass Module with the aid of PID control algorithm may command to drive the servo motor to go towards in straight line platform in accordance to the desired set-point direction has been fixed. The robotic hardware has been developed and analyzed successfully. As a result, in despite of unexpected external force varying the desired direction of robot, the robot would still be able to veer back to the original set-point direction to achieve a smooth and stabilized straight line movement.

ABSTRAK

Projek ini tertumpu kepada pembangunan pergerakan garis lurus untuk robot beralih empat roda. Dalam project ini, motor gear arus terus (DC) dipilih sebagai kawalan gerakan untuk dua roda memandu dan arah robot akan dikawal oleh motor servo untuk stering dua roda. Peripheral Interface Controller (PIC) dipilih sebagai pengendali utama untuk tindak balas dan respon kepada data yang diterima dari Kompas Digital Modul untuk mengenal pasti dan menentu kedudukan. Perlaksanaan algorithm Proportional Integral Derivative (PID) dalaman asasnya digunakan untuk mengembalikan sistem kepada kedudukan titik penentuan yang ditetapkan. Aplikasi Kompas Digital dengan bantuan algorithm kawalan PID untuk memacu motor servo ke arah platform garis lurus berdasarkan arah yang ditetapkan. Perkakasan robot berjaya dibangun dan dianalisis. Hasilnya robot mampu mengembali ke arah set penentuan asal namun daya luaran yang tidak dijangka mengubah hala tuju robot yang ditetapkan untuk mencapai pergerakan garis lurus yang lancar dan stabil.

TABLE OF CONTENTS

CHAPTER	CONTENT	PAGE
	TITLE PAGE	i
	DECLARATION	ii
	DEDICATION	iii
	ACKNOWLEDGEMENTS	iv
	ABSTRACT	v
	ABSTRAK	vi
	TABLE OF CONTENTS	vii
	LIST OF TABLES	x
	LIST OF FIGURES	xii
	LIST OF ABBREVIATIONS	xv
	LIST OF APPENDICES	xvi
 1	 INTRODUCTION	
	1.1 Background of Project	1
	1.2 Problem Statement	3
	1.3 Objectives	5
	1.4 Scope of Project	6
	1.5 Expected Outcomes	7
 2	 LITERATURE REVIEW	

2.1	Introduction	8
2.2	Previous Project Work	9
2.2.1	Line_Following Two Wheels Balancing Robot	9
2.2.2	Four-Wheeled Mobile Robot	10
2.2.3	Straight Line Movement Principle	12
2.2.4	Digital Compass Module for Direction Verification	13
2.2.5	Ultrasonic Sensor	14
2.3	Proportional Integral and Derivative (PID) Algorithm Control	15

3

METHODOLOGY

3.1	Introduction	17
3.2	Stability and Straight Line Principles	21
3.2.1	Robot's Controllability and Stability	21
3.2.2	PID Control Algorithm	23
3.3	Software Review	25
3.3.1	CCS C Compiler	25
3.3.2	PICkit 2 V2.61	29
3.4	Hardware Review	35
3.4.1	PIC Brain Board (SK40C) with PIC 18F4550	37
3.4.2	USB ICSP PIC Programmer V2010 (UIC00B)	40
3.4.3	Rainbow Cable	42
3.4.4	DC Gear Motor with Motor Driver IC L293D	43
3.4.5	Servo Motor	47
3.4.6	Digital Compass Module	50
3.5	DC Motor Speed Control	53
3.6	Servo Motor Turning Control	57

3.7	Manipulation of Digital Compass Module to Servo Motor	61
3.8	Obstacles and Wall Detection	65

4

RESULT AND DISCUSSION

4.1	Introduction	45
4.1.1	Four-Wheeled Mobile Robot (FWMR)	70
4.1.2	Full Connection Circuits of Four-Wheeled Mobile Robot	72
4.1.3	Full Sequence of FWMR Operation	73
4.1.4	Full Programming of FWMR Operation	75
4.2	PID Control Algorithm	75
4.2.1	Theoretical Framework	77
4.2.2	Practical Results	79
4.3	Discussion	83

5

CONCLUSION AND RECOMMENDATIONS

5.1	Introduction	85
5.2	Conclusion	86
5.3	Recommendations and Improvements	87
5.4	Additional Elements and Future Tasks	88

REFERENCES	89
-------------------	----

APPENDICES	92
-------------------	----

LIST OF TABLES

TABLE	TITLE	PAGE
3.1	Types of Activities and Respective Description in “File Menu”	27
3.2	Types of Activities and Respective Description in “Compile Menu Ribbon”	28
3.3	Types of Activities and Respective Description in “Help Menu”	29
3.4	Overview of “File” in Menu Bar	30
3.5	Overview of “Programmer” in Menu Bar	31
3.6	Overview of “Tool” in Menu Bar	32
3.7	Overview of “Help” in Menu Bar	33
3.8	Each Part of SK40C with Function Respectively	38
3.9	Each Part of UIC00B Programmer V2010 with Function Respectively	42
3.10	Behavior of Motor Operation with Various Input Conditions	45
3.11	Servo Motor Turning in accordance with Pulse Width	58
3.12	Manipulation of Compass Heading Degree to Servo Motor Turning Degree	62
3.13	MaxSonar EZ1 Pin Out	67
4.1	Respective Parts of Four-Wheeled Mobile Robot	70

4.2	Manipulation of PID Output Range to Servo Motor Turning Degree	76
4.3	PID Constants Adjustment of K_P , K_I and K_D	81

LIST OF FIGURES

FIGURE	TITLE	PAGE
1.1	Block Diagram of Main Components for FWMR	3
2.1	A Line Follower Robot	10
2.2	Front Driven / Steered Ackerman Steering	11
2.3	Front Steered / Rear Driven Ackerman Steering	11
2.4	A Four-Wheeled Mobile Robot in Straight Line Movement	13
2.5	Principle Use of Ultrasonic Sensor	15
2.6	Basic Structure of a system with PID Control	16
3.1	FWMR General System Block Diagram	18
3.2	Overall FWMR's System Flow	19
3.3	Types of Wheeled Mobile Robot	22
3.4	Center of Gravity for FWMR	23
3.5	CCS C Compiler Overview	26
3.6	PICkit TM 2 Programmer Application	30
3.7	Overall Hardware Configuration of FWMR Block Diagram	36
3.8	Top View of SK40C Brain Board Layout	38
3.9	40-Pin PDIP of PIC 18F4550 Configuration	40
3.10	Top View of USB ICSP PIC Programmer V2010's Broad Layout	41
3.11	Rainbow Cable or Programming Cable	42
3.12	DC Gear Motor	44
3.13	Installation of DC Gear Motor to FWMR	45
3.14	IC L293D's Pin Configurations	45

3.15	Connection Circuit of DC Motor and IC L293D to PIC 2	46
3.16	Sequence of DC Gear Motor Operation	46
3.17	Servo Motor (RC C40R)	47
3.18	Installation of Servo Motor to FWMR	48
3.19	Connection Circuit of Servo Motor to PIC 1	48
3.20	Sequence of Servo Motor Steering Direction	49
3.21	Digital Compass Module	51
3.22	Installation of Digital Compass Module HMC6352 to FWMR	51
3.23	Connection Circuit of Digital Compass Module to PIC 1	52
3.24	Sequence of Digital Compass Module Function	52
3.25	Installation of Four Switches to Control DC Motor Speed In FWMR	54
3.26	Connection Circuit of DC Gear Motor, Four Switches and Two LEDs to PIC 2	55
3.27	DC Motor Speed Control by Four Switches and LEDs Indications	56
3.28	Pulse Width for Servo Motor Turning	57
3.29	Turning Degree of Servo Motor C40R	58
3.30	Installation of Servo Motor Steering Control to Two Front Wheels	59
3.31	Connection Circuit of Servo Motor to Two Front Wheels	59
3.32	Sequence of Servo Motor Turning Control	60
3.33	Digital Compass Module to Control Servo Motor Turning Degree	63
3.34	Connection Circuit of Digital Compass Module and Servo Motor to PIC 1	63
3.35	Sequence of Digital Compass Module Heading Degree Manipulation to Servo Motor Turning Control	64
3.36	Side and Bottom View of MaxSonar EZ1	66
3.37	Installation of Ultrasonic Sensor to FWMR	66
3.38	Connection Circuit of Ultrasonic Sensor to PIC 2	67
3.39	Sequence of Ultrasonic Sensor Distance Detection	68
4.1	Four-Wheeled Mobile Robot (FWMR)	70

4.2	Top View of FWMR	71
4.3	Side View of FWMR	71
4.4	Front View of FMWR	71
4.5	Full Connection Circuit of PIC 1	72
4.6	Full Connection Circuit of PIC 2	72
4.7	Full Sequence Flow of PIC 1	73
4.8	Full Sequence Flow of PIC 2	74
4.9	Equation 1 of PID Output in Time Domain	77
4.10	Equation 2 of PID Output after Laplace Transform	78
4.11	Equation 3 of PID Output Application in Programming	78
4.12	Robot Straight Line Movement in Theoretical Application	78
4.13	FWMR Straight Line Movement Performance in Heading Degree of 275 °With and Without PID Control Algorithm	79
4.14	FWMR Straight Line Movement Performance in Heading Degree of 350 °With and Without PID Control Algorithm	80
4.15	Three PID Constants Adjustment in Heading Degree of 275 °	82
4.16	Three PID Constants Adjustment in Heading Degree of 350 °	83

LIST OF ABBREVIATION

PIC	-	Peripheral Interface Controller
PID	-	Proportional Integral Derivative
DC	-	Direct Current
FWMR	-	Four-Wheeled Mobile Robot
LCD	-	Liquid Crystal Display
USB	-	Universal Serial Bus
I ² C	-	Inter Integrated IC
IC	-	Integrated Circuit
I / O	-	Input / Output
LED	-	Light-Emitting Diode
UART	-	Universal Asynchronous Receiver / Transmitter
MCU	-	Microcontroller Unit
EEPROM	-	Electrically Erasable Programmable Read Only
RAM	-	Random Access Memory
SPI	-	Serial Peripheral Interface
CCP	-	Capture, Compare or PWM Mode
PWM	-	Pulse Width Modulation
ADC	-	Analog Digital Converter
RC	-	Radio Control
GND	-	Ground
Tx	-	Transmit
Rx	-	Receive
DCM	-	Digital Compass Module
GPS	-	Global Positioning System

LIST OF APPENDICES

APPENDIX	TITLE	PAGE
A	Program Source Code	92
B	Four-Wheeled Mobile Robot's Specification	109
C	Four-Wheeled Mobile Robot's Overall Circuit Diagram	111
D	Four-Wheeled Mobile Robot's Pictures	113

CHAPTER 1

INTRODUCTION

1.1 Background of Project

Robotic technologies have potential to grow and develop rapidly around this recent century. Robots have been essential and vital which not only applying in engineering field, however for other aspects especially in field of education, medical, industries and even to be used for our daily life basis. By time goes by, there is a lot of mechanical robots have been invented to aid humans for complete their daily life's tasks which difficult to be fulfilled by human beings.

The main purpose of this project is to develop a mobile robot with moving straight line ability without the line-following aids. This mobile robot is a four-wheeled platform robot or car-type drive robot which is common in our surroundings. Four-wheeled mobile robot to be chosen due to its front-wheels drive with a smaller radius compared to differential drive and tricycle-type robot which apply with three wheels. To find a configuration that maximizes the qualities especially regards to controllability, stability and maneuverability, thus four-wheeled robot is right to be applied than two-wheeled or three-wheeled robots due to achieve the requirement of development of straight line movement for a robot.

The motion of this mobile robot will be controlled by a direct current (DC) motor or acts as driving motor to control two driving wheels from behind of the robot. In order to move in straight line, the two speeds of driving wheels must be equal to each other for fulfilling the straight line movement requirement. To stop the robot, speed value of DC motor in the above function must be set to zero. For second property, servo motor is used to connect to the front wheels or named as steering wheels for controlling the displacement of two steering wheels. Displacement is set to be zero to ensure the robot performing the straight line movement. For moving in straight line platform, the robot may often oscillate or veer to one side, therefore servo motor has the responsibility to steer the steering wheels back to the original straight line condition.

Digital compass module is added up due to ensure the fixed angle of direction of the robot moving. If the robot suddenly veered to another angle, this error of angle will be detected and respond to the PIC microcontroller in order to react for adjusting the position of servo motor due to move back to the original angle of direction for the robot moving.

Proportional-Integral-Derivative (PID) control has selected to insert and program to the microcontroller due to its control algorithm can be attributed partly to its robust performance and functional simplicity. A PID algorithm consists of three basic coefficients as the name suggested. These gains are varied to achieve an optimal system response. A system output of heading degree or position of robot is read by Digital Compass Module and then compare the reading with the reference point of desired value has fixed and set into the controller. The difference of the reference and the measured output may result an error value using to calculate the proportional, integral and derivative responses. These three responses are then summed to obtain in the output of the controller which used as input to control the servo motor turning in order that robot may move in straight line.

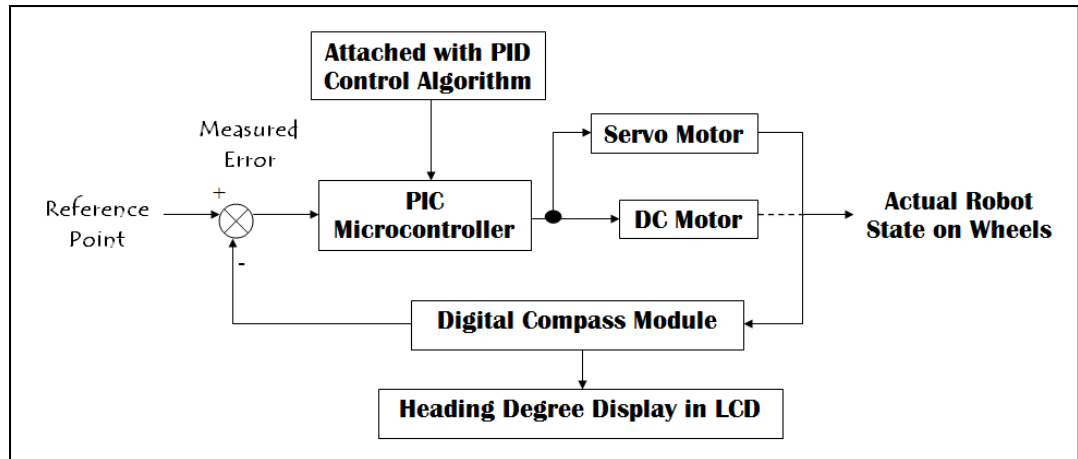


Figure 1.1: Block Diagram of Main Components for FWMR

1.2 Problem Statement

Moving vehicles are a popular type of robot and can be apply as perfect instance for an excellent starting point. Invention of a straight line movement autonomous four-wheeled mobile robot (FWMR) is a relatively simple and straight forward robotic project which approaches to the real moving vehicles especially the cars that we use every day.

Through this project research, four-wheeled mobile robot is my desired choice due to its high balancing condition compared to the others. However, FWMR is a great challenge for the engineers or researches to mainly maintain a straight line movement. There are many aspects require to be considered essentially for the constant rotational speed of a DC motor for driving the back wheels purpose and utilize a servo motor for controlling the displacement of both front wheels due to return back to the straight line condition if FWMR tilted or tend towards other side rather than straight line movement.

As a result, Digital Compass Module is added up which used to ensure the fixes angle of direction of the robot moving. If robot veers to another direction with different angle, response will be sent to controller due to react and adjust the position of servo motor for moving back to the original direction of the robot moving.

Proportional, Integral and Derivative (PID) control is acted as intelligent control requires to be added up due to suit the objectives and aims as mentioned. It is a popular loop feedback-control which to provide suitable algorithm for keeping the FWMR centre always above a certain line for moving straight and the robot will not oscillate a lot along the line and waste any valuable time and battery supply. The term of the PID algorithm have to be explicit especially for the K_p , K_i and K_d are the constants used to vary the effect of Proportional, Integral and Derivative terms respectively. PID algorithm is being used upon the error based on the current position of the motors moved. Therefore, PID control is a critical part which requires being tuned time-by-time for trial due to achieve the less oscillating and tend to straight line movement.

1.3 Objectives

There are three main objectives in this project which are:

1. To control the FWMR of straight line movement by using PIC microcontroller.
2. To ensure the angle of direction of the robot moving in a constant straight direction by the aid of digital compass module.
3. To design and add-up PID control algorithm in PIC microcontroller for achieving the robot straight line movement.

1.4 Scope of Project

According to the objectives of this project, there are few scopes have been highlighted as follow:

1. Design a FWMR is acted as autonomous four-wheeled mobile robot which able to move a straight line with minimizing the oscillation or tending towards other side.
2. With the aid of PIC controller to control the turning degree of the servo motor for two front steering wheels and constant speed control of the DC motor for two back driving wheels after the PIC controller has been fully programmed due to the straight line movement purpose of FWMR.
3. By the aid of Digital Compass Module, the robot will be always ensured with moving in fixed angle of direction for achieving straight line movement.
4. Develop a PID control algorithm with tuning method in accordance with the error value which feedback by Digital Compass Module to the PIC controller to achieve a smooth and stabilized straight line movement by FWMR.

1.5 Expected Outcomes

After interfacing the hardware, software and full correct programming to this project, this FWMR can be constructed and produced successfully in the aspects of mechanical part and its project functionality.

This FWMR can be functioned mainly depends on the main components especially for PIC microcontroller that has been programmed to command controlling the turning degree or angle of servo motor to both front steering wheels and for DC motor of constant speed control to both back driving wheels.

Straight line movement of FWMR needs to be based on the mechanical part especially the equipments of construction of this robot as like the wheels and fabrication of the robot body. Angle of direction or the heading degree for the robot movement can be ensured by Digital Compass Module to guide the robot moving in straight line platform.

Added up with the PID control algorithm which totally able to minimize the oscillation of the robot or tend towards other side during the straight line moving purpose since the integration and derivation of the PID algorithm to recover an error to the original set value that has been programmed to the PIC controller due to give the correction to turning angle of servo motor for steering the robot to the desired set-point direction that has been fixed.

CHAPTER 2

LITERATURE REVIEW

2.1 Introduction

Literature review is prior tool to begin a research project due to understand more vivid regarding with the development of four-wheeled robot in straight line movement in the aspects of robot control technique, mechanical designation and construction, programming development and the reviews may provide researchers some ideal information on the methodologies and technologies which have been applied by other research counterparts around the world. This chapter would convey a detail summary of a mobile wheeled robot with the main aim of straight line movement without the aid of line-follower concept. The present project of development of robot straight line movement with the previous researches will be compared and discussed further on.

A wheeled mobile robot is a wheeled vehicle which is capable of autonomous motion. Autonomous mobile robots are a very interesting subject both in scientific research and practical applications [1]. Wheeled-robot is a human invention and a very popular locomotion concept in man made vehicles. The main attributes of stability are the number and geometry of contact points, the robots centre of gravity,

if the robot is static or dynamic stable and the inclination of terrain. Mobile wheeled-robot is considered as the easy mechanical implementation since there is no need of balance control if the vehicle has at least three or in some case two wheels locomotion. When designed a wheeled robot, the developer has the choice of several different wheel arrangements and wheel types. The combination of wheel type and arrangement is strongly linked and governs the stability, maneuverability and controllability if the robot [2].

2.2 Previous Project Work

2.2.1 Line-Following Two Wheels Balancing Robot

A researcher, Tan Piow Yon (2011) from University Malaysia Pahang has implemented and investigated regarding with two wheels balancing robot with line following capability. His project focuses on the development of a line follower algorithm for a Two Wheels Balancing Robot. Tan Piow Yon (2011) stated that ATMEGA32 has been chosen as the brain board controller to react towards the data received from Balancing Processor Chip in the balance board to monitor the changes of the environment through two infra-red distance sensor to solve the inclination angle problem. With this project research and investigation, infra-red light sensors with internal PID algorithms control are vital due to develop a smooth line follower robot [3].

On the other side, Pratheek (2009) and Jaseung Ku (2005) had the similar opinion that the line-follower robot is able to follow a black or white line on the

ground in accordance with the requirement from the researchers themselves. Due to the project name mentioned, the robot can move in straight line with according to the line-follower. Figure 2.1 expresses that the line-follower robot move in straight based on the black line on the ground.

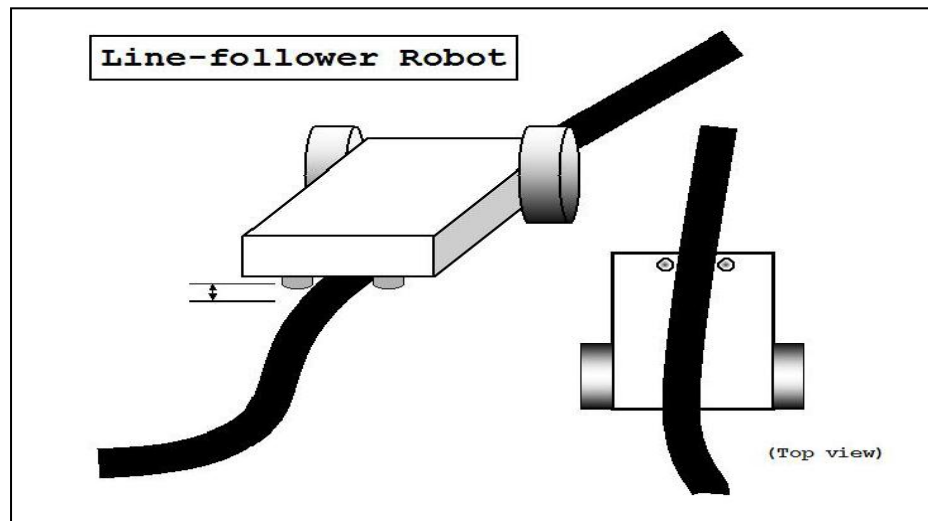


Figure 2.1: A Line Follower Robot (Jaseung Ku, 2005)

2.2.2 Four-Wheeled Mobile Robot

A researcher from Vrije Universiteit Brussel, Kristof Goris (2004-2005) who has investigated regarding to autonomous mobile robot mechanical design mentioned that generally stability of robot can be further improved by adding more wheels, although once the number of contact points exceeds three, the hyper static nature of the geometry will require some form of flexible suspension on uneven terrain to maintain wheel contact with the ground. Kristof Goris stated that balance is not usually a research problem in four-wheeled robot designs, this is because most of the wheeled mobile robot with wheels are in ground contact at all times. Instead of worrying about balance, wheeled robot researcher tends to focus on the problems of traction and stability, maneuverability, and control.

A car type locomotion or Ackerman steering configuration used in automobiles which is very common in the ‘real world’, but not as common in the ‘robot world’. The limited maneuverability of Ackerman steering has a vital benefit which its directionality and steering geometry provide it with very good lateral stability in high-speed turns. Ackerman steering and tricycle steering is designed with a pair of driving wheels and another separated pair of steering wheels. Figure 2.2 and Figure 2.3 indicate that a front driven car drive and its main concurrent is the rear wheel driven car drive configuration.

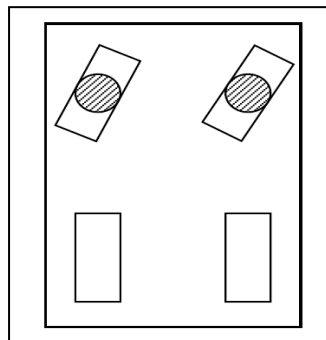


Figure 2.2: Front Driven / Steered Ackerman Steering (Kristof Goris, 2004-2005)

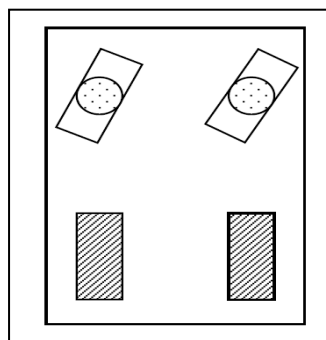


Figure 2.3: Front Steered / Rear Driven Ackerman Steering (Kristof Goris, 2004-2005)

A car type drive is one of the simplest locomotion systems in which separate motors control translation and turning this is an important advantage compared to the differential drive system. The main condition is the turn mechanism must be precisely controlled. A small position error in the turning mechanism can cause large

odometry errors. This simplicity in line motion is the main reason why car type drive is popular for human driven vehicles [4].

2.2.3 Straight Line Movement Principles

According to Ibrahim Kamal (2008), car type drive system consists of two wheels coupled on the same axle at the back and one or two another front wheels capable of steering to control the displacement of the robot. Ibrahim Kamal indicated that this type of wheeled robot has two major advantages which it is easier to control and more accuracy since traction and steering are connected to two independent motors. However, the technique of this car type system is very difficult to implement mechanically [5].

In order for a wheeled-mobile robot to go straight line, both traction and driving motors have to be turning at precisely the same rate and the wheels have to be the same diameter as well. This means that as a wheeled robot travels from A to B, the wheels on both sides travel the same distance in the same time and therefore to go with the same speed. This basic principle has been agreed by K-Junior (2010) and stated that two speed values of the motors to control the two driving wheels have to be equal, if the two wheels have different values, the robot will move on a circular trajectory [6]. In addition, the position-control of servo motor to the front steering wheels need to be set as zero degree or original position-base due to enable the robot to travel in straight line without veering to other side. Figure 2.4 shows that the four-wheeled mobile robot travels from position A to B with applying the basic principle in order to achieve the straight line movement.

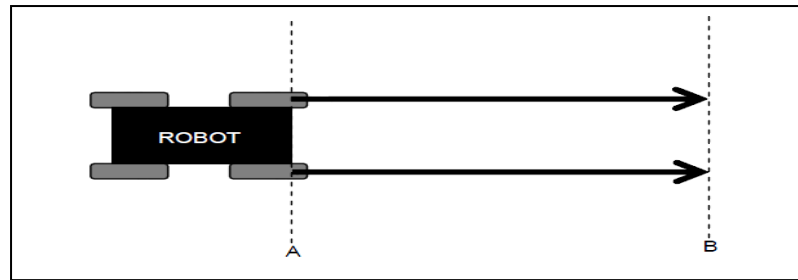


Figure 2.4: A Four-wheeled Mobile Robot in Straight Line Movement

2.2.4 Digital Compass Module for Direction Verification

Digital compass module is reviewed which to be applied in this four-wheeled mobile robot system. This is a great tool for checking the robot often heading to the desired straight movement position in accordance to a set-point direction in case the robot is interrupted or interfered by an obstacle or rugged ground.

The aid of compass module by comparing the compass sensor output with a set-point, then the controller will be able to control the turning angle of servo motor. The servo motor will turn clockwise or counter clockwise [7]. The set-point will be programmed and stated on the first position of the robot set on the ground after the power supply is provided. If there was any interference occurred accidentally which cause the robot veering to right or left of the straight motion, the error of the difference of compass sensor output with the set-point will respond to microcontroller. Following this, the controller will react and control the servo motor to veer the steering wheels back to the original point of straight motion condition. Therefore, the robot may turn and face back to the set point motion.

2.2.5 Ultrasonic Sensor

Ultrasonic sensors or can be known as transceivers when a transmitter is used to send the sound wave signal where a receiver is used to receive the corresponding sound wave signal. It works on a principle similar to radar or sonar which evaluate attributes if a target by interpreting the echoes from radio or sound waves respectively. Ultrasonic sensors generate high frequency sound waves and evaluate the echo which us received back by the sensor. Sensors calculate the time interval between sending the signal and receiving the echo to determine the distance to an object [8].

Khairul (2007) mentioned that ultrasonic sensors typically have a piezoelectric ceramic transducer that converts an excitation electrical signal into ultrasonic energy bursts. The energy bursts travel from the ultrasonic sensor, bounce off objects, and are returned towards the sensor as echoes. Transducers are devices that convert electrical energy to mechanical energy, or vice versa. The transducer converts received echoes into analog electrical signals that are output from the transducer.

The ultrasonic transducer produces ultrasonic signals. These signals are propagated through a sensing medium and the same transducer can be used to detect returning signals. In most applications, the sensing medium is simply air. An ultrasonic sensor typically comprises at least one ultrasonic transducer which transforms electrical energy into sound and, in reverse, sound into electrical energy, a housing enclosing the ultrasonic transducers. Ultrasonic sensors transmit ultrasonic waves from its sensor head and again receive the ultrasonic waves reflected from an object. By measuring the length of time from the transmission to reception of the sonic wave, it detects the position of the object [9].

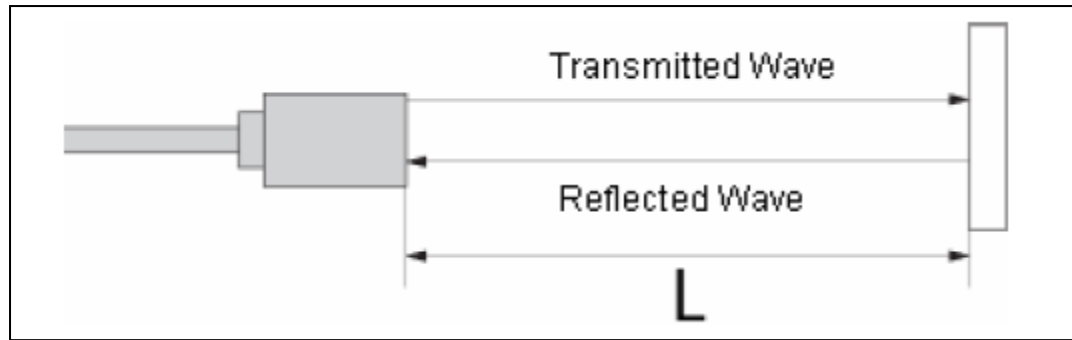


Figure 2.5: Principle Use of Ultrasonic Sensor (Khairul, 2007)

2.3 Proportional Integral and Derivative (PID) Algorithm Control

Proportional Integral and Derivative (PID) algorithm control is generally a control loop feedback mechanism widely applied in robotic and industrial control systems. The popularity of PID controllers can be attributed partly to their robust performance in a wide range of operating conditions and partly their functional simplicity which allows the researchers and engineers to operate them in simple and straightforward manner [10].

A PID algorithm consists of three basic coefficients: Proportional, Integral and Derivative. All these gains are varied to achieve an optimal system response [10]. The ‘Proportional’ value is used to determine the reaction to the current error, ‘Integral’ determines the reaction based on the sum of recent errors and ‘Derivative’ is used for the reaction to the rate at which the error has been changing. By using a sensor to read the system output and compares the reading to a reference point or set point. The difference between the reference and measured output may result to an error value which used to calculate proportional, integral and derivative responses. The weighed sum of these three actions is then used to obtain the output of the controller.

A PID controller works by continuously measuring the output of the robot system and providing corrective input calculated from the PID control algorithm. The PID controller is scalable and tunable. PID scalable means to use the proportional gain or combination of the proportional gain and either the integral or derivative gain (P, PI, PD, or PID control). The system is tunable due to adjust the P, I and D gains to tune the controller for specific system. The basic structure of the system with PID control implemented is shown in Figure 2.6

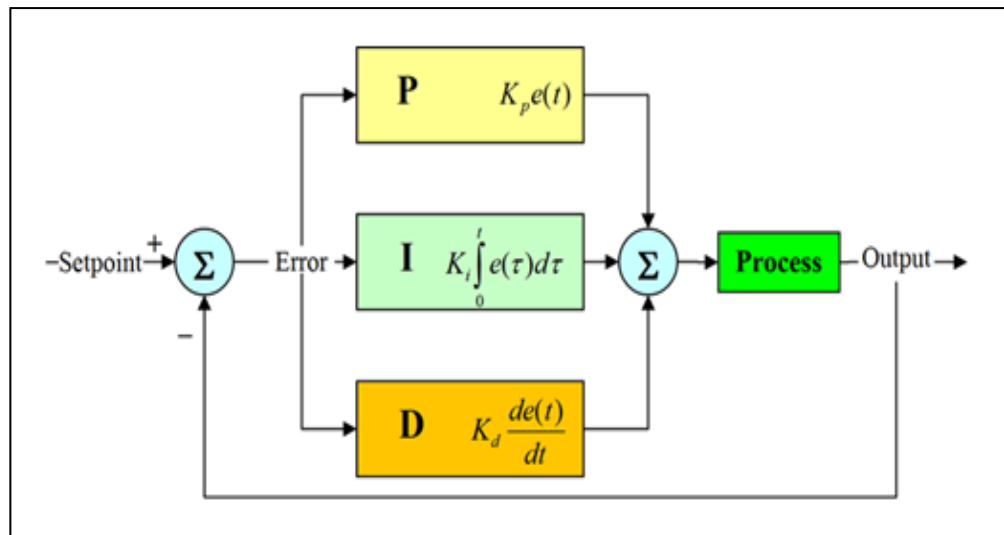


Figure 2.6: Basic structure of a system with PID Control

CHAPTER 3

METHODOLOGY

3.1 Introduction

Research Methodology is an essential chapter which generally gives a guideline and aid to the researchers to solve problems with specific components especially methods, techniques, essential tools, tasks, hardware implementation and software interfacing. The main purpose of this project is to develop a Four-Wheeled Mobile Robot which behaves to perform a straight line movement without the guide or aid of line follower. This project involves with DC motors to manipulate the motion control to the driving wheels or back wheels of robot, whereas servo motor is used to control the steering wheels or front wheels for direction-steering purpose. These both motors are managed and controlled by two PIC brain board controllers respectively. One of the PIC controller is used to control and manipulate the speed or movement of FWMR where for the second PIC will be attached with PID control algorithm due to react the data received from Digital Compass Module by comparing the heading degree of digital compass sensor output and the set-point value which has been programmed in PIC controller. In case the robot tends to other side especially veering to left or right from fixed straight line direction, PIC may command to the servo motor to turn clockwise or counter clockwise due to veer the robot back to the original set-point straight line direction. The FWMR General

System Block Diagram as shown in Figure 3.1 and the Overall FWMR's System Flow as indicated in Figure 3.2.

The Research Methodology of this project is separated to two portions named as Hardware Implementation and Software Implementation. Hardware configuration regards to the interface between the Brain Board Microchip PIC controller to servo motor, DC motor, ultrasonic sensor and Digital Compass Module which including with the brief comprehension and introduction to other essential components. In the software part, all programming and coding development from programming software which sets and programs to the PIC controller to command and control the corresponding essential hardware elements especially DC and servo motors, Digital Compass Module and ultrasonic sensor. In this FWMR project, the Custom Computer Service (CCS) Compiler will be used by applying the C-programming language to set and program into the PIC microchip controller for further action to the robot. Every important part of this project will be discussed and explained further in the following sections.

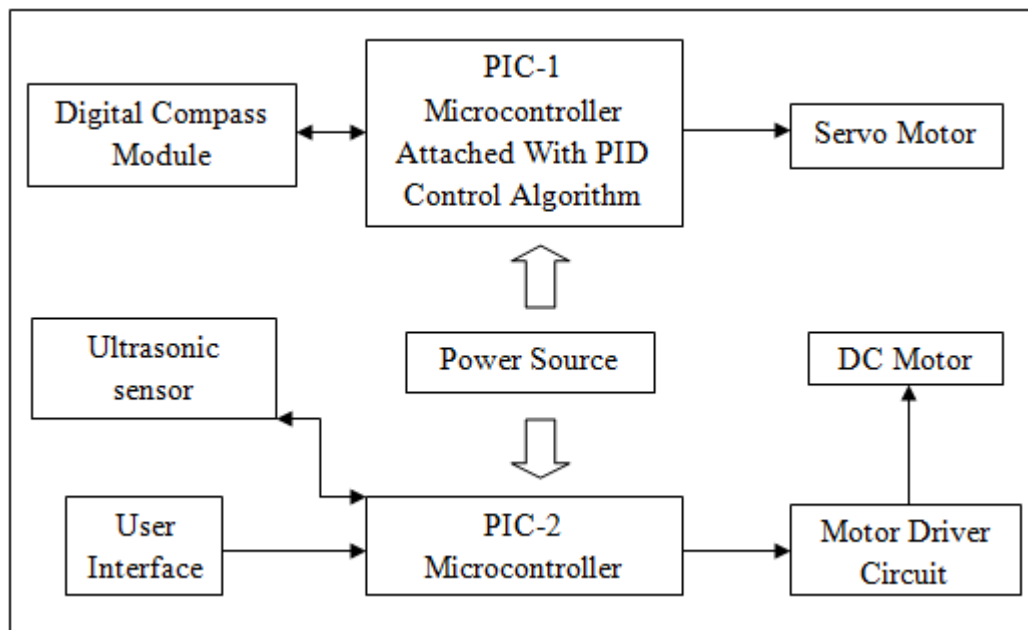


Figure 3.1: FWMR General System Block Diagram

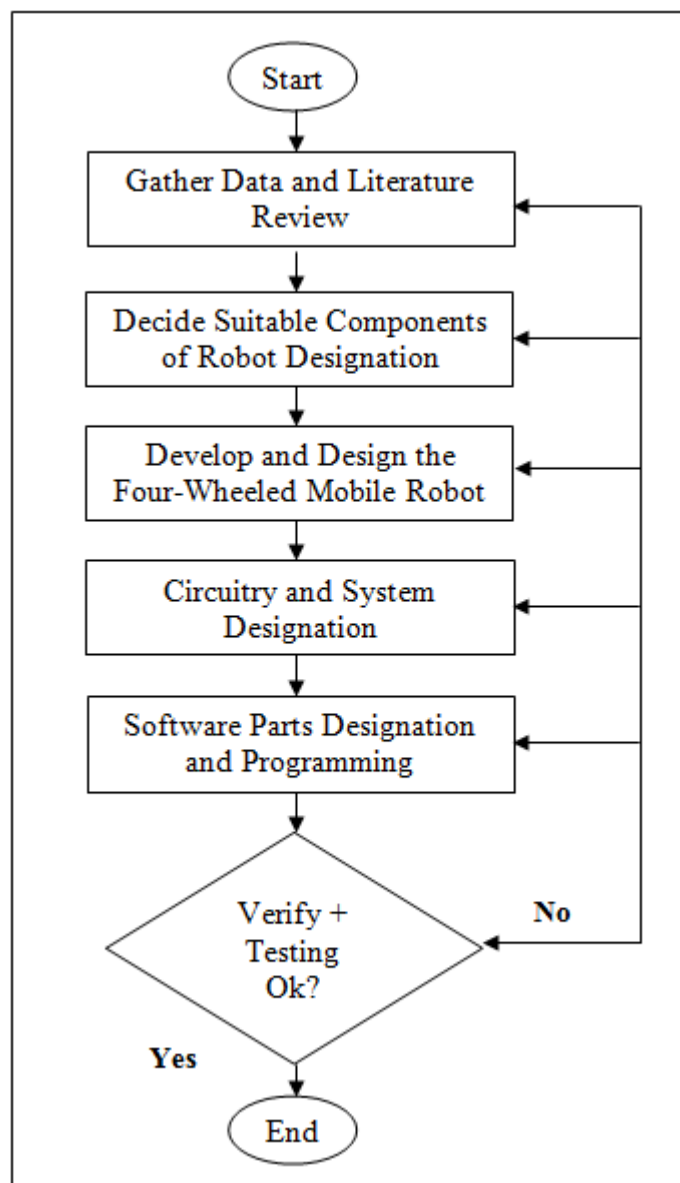


Figure 3.2: Overall FWMR's System Flow

There are three specific topics will be discussed and elaborated detail in following sections:

- i. Stability and Straight Line Principles
 - a. Robot's Controllability and Stability
 - b. PID Control Algorithm
- ii. Software Review
 - a. CCS C-Compiler
 - b. PICKit 2 V2.61
- iii. Hardware Review
 - a. PIC Brain Board (SK40C) with PIC 18F4550
 - b. USB ICSP PIC Programmer V2010 (UIC00B)
 - c. Rainbow Cable
 - d. DC Gear Motor with Motor Driver IC L293D
 - e. Servo Motor
 - f. Ultrasonic Sensor
 - g. Digital Compass Module

For the previous chapter has been discussed and reviewed certain prime hardware and stability's theories of the robot system. In this chapter, three topics as shown above will be further discussed deeply especially for every vital sub-component in hardware implementation with their functionalities and applications respectively. In software part may concentrate to the utilization of CCS C-Compiler which the way to perform experiment and set the C-programming coding to the Brain Board Microchip PIC Controller. Through this project methodology research, researchers and investigators are able to comprehend the functionality, performance and way to utilize all the hardware and software configurations. Therefore,

interfacing between hardware and software parts finally can be accomplished to fulfill this FWMR project system requirement.

3.2 Stability and Straight Line Principles

This sub-topic may advance to discuss the controllability, stability and maneuverability of the robot by complying with all relevant principles and theories to achieve the robot straight line movement performance. Furthermore, the aid of PID control algorithm can be further advanced the project's performance to achieve the straight line movement state which based on the programming set into the controller by applying with the manually tuning method in FWMR.

3.2.1 Robot's Controllability and Stability

Autonomous mobile robot mechanical design is mainly to influence the performance of the robot system that we desire. Thus, the mechanical design to ensure the stability of a robot can be further improved by added up with more wheels. Wheels are the essential tool for an autonomous robot to achieve their controllability and stability since wheels are the contact points with the ground whether it is flat or uneven terrain which still can be able to maintain the robot's stability to escape from robot falling or unbalanced situation to affect the desired performance.

In Figure 3.3 has shown as the types of wheeled-mobile robot which in accordance with their stability and controllability to achieve the moving performance. Car type locomotion is a very common in the 'real world', but not the way in the

‘robot world’. Therefore, Four-Wheeled Mobile Robot to be chosen for implementation is an excellent task to learn and exert our capability to build up a mobile robot which approaches to the ‘real world’ vehicles around us.

Due to achieve the FWMR straight line movement, both traction or driving motors are required to be operating with same rate or speed and firmly the wheels must use the same diameter as well. This brings the meaning that once the robot travel from one starting point to one ending point, both driving wheels must operate and rotate with the same position and speed in regarded time. Therefore, the robot may perform the straight line movement without the effect from external disturbances and interferences.

Center of gravity of the robot plays the main role to ensure that the total weight of the robot is adapted to the center of gravity due to avoid robot skidding to other side or even falling with the unbalanced condition. The center of gravity of a FWMR can be spotted and observed from the X-point in Figure 3.4 in accordance with the height and width of the robot to obtain the point of centre of gravity.

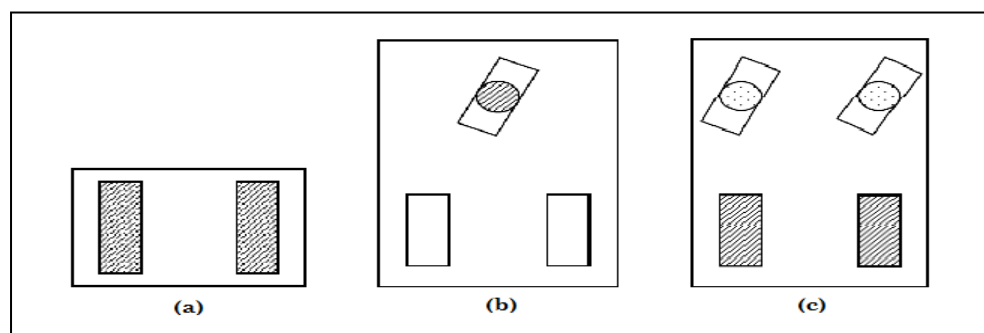


Figure 3.3: Types of Wheeled Mobile Robot – (a) Two-Wheeled (b) Three-Wheeled (c) Four-Wheeled with Front Steering and Back Driving

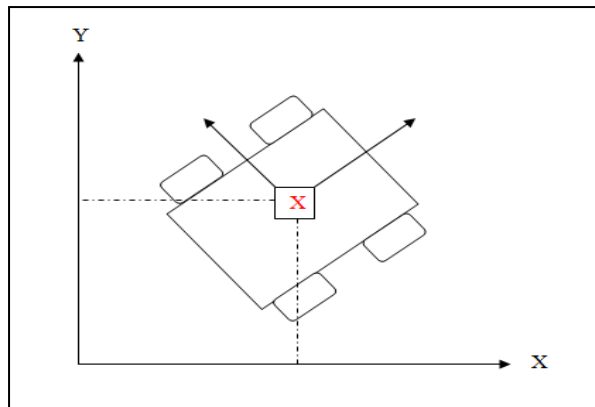


Figure 3.4: Center of Gravity for FWMR

3.2.2 PID Control Algorithm

In accordance to the literature reviewing, PID control algorithm is abbreviation of Proportional-Integral-Derivative algorithm which generally used in control loop feedback mechanism widely applied in robotic and industrial control systems. The Proportional-Integral-Derivative algorithms are well known with its three terms to correct and modify the error occurs in many aspects of robotic and industrial process control. As a result, PID control algorithm has become a basis and advanced control algorithm to the FWMR in order to fulfill the straight line movement performance.

PID control algorithm is a very straight forward algorithm that provides the necessary output system response to control a process. One unique advantage of the PID algorithm is to manipulate the process inputs based on the history and rate of change of the signal. Therefore, the algorithm is suited for linear system modeled process control which gives more accurate and stable control.

PID control algorithm consists of three basic coefficient terms: Proportional, Integral and Derivative. Each term would have its own different task in control process compared to each others. 'Proportional' term is for determining the reaction to the current error, 'Integral' is to ensure the reaction based on the sum of recent errors and 'Derivative' is used for the reaction to the rate at which the error has been changed. All these control terms may have different effects on the system output response.

An error is performing as input to the controller which fed from the robot system output process. The K_P , K_I and K_D are referred as proportional, integral and derivative constants. Each term may bring its equation respectively and finally weighed sum of all these three term actions used to obtain the output of the controller to send to the servo motor for giving a right displacement or turning angle due to achieve the straight line movement purpose on a fixed set-point direction.

In this project case, PID control algorithm will be applied in tunable with trial and error method due to adjust the constant values of each term P, I and D. This brings meaning that to tune the K_P , K_I and K_D to get the best performance of FWMR. Observation to the robot performance due to determine what for the next ought to do. If the robot wobbles in a large value, then reduce the K_P value. Instead, if the robot doesn't aim to straight or goes straight on curves or oscillating, action to increase the K_P should be taken. Tune the K_P value till the robot to be able to move straight smoothly. The term of K_D should be tuned then follow by the value of K_I in order to observe the robot is able to go on straight with no acceleration. The optimum K_P , K_I and K_D values vary a lot even from track to track. Therefore, these optimum values only able to be obtained accurately by testing one-by-one in sequent.

3.3 Software Review

This section generally discusses on programming software that may use in FWMR project and write suitable programming to PIC microcontroller. The way to apply, use and utilize the programming software will be further discussed and elaborated in this section.

3.3.1 CCS C Compiler

In this FWMR project, programming language C will be selected due to its popularity and very widely used in microcontroller especially in PIC. By using C-Language Programming to program the PIC is proper and faster for robot performing some fairly simple or even complex tasks. As a result, Customer Computer Services (CCS) Compiler would be chosen as my project's main communication and interface between the microcontroller with desktop and other operational components especially motors and sensors which are controlled by PIC microcontroller.

CCS C Compiler is very convenient and useful since it supports the Microhip PIC12x, PIC16x, PIC18x and dsPIC devices. The compiler is very close to being 100% ANSI compatible. It supports everything a PIC compiler needs and necessary superset of ANSI C to work with embedded micros, such as fuse and interrupt level supports [15].

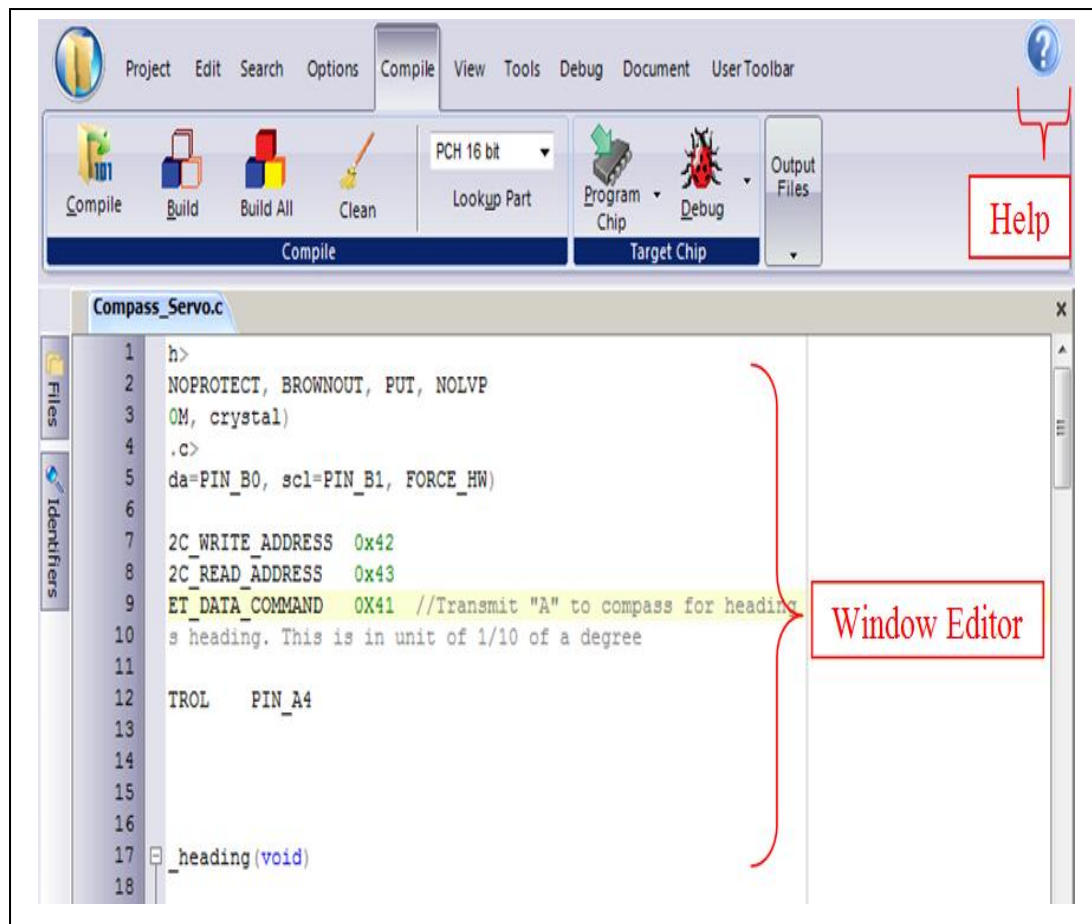


Figure 3.5: CCS C Compiler Overview

The menus and toolbars of CCS C Compiler are set-up in specially organized Ribbons. Each Ribbon relates to a specific type of activity and is only shown when selected. CCS has included a “User Toolbar” Ribbon that allows the user to customize the Ribbon for individual needs. Following is the guide and instructions for users to utilize this software for easier to get started and applied.

1. To utilize the following items, click the icon



Table 3.1: Types of Activities and Respective Description in “File Menu”

Type of Activity	Description of Usage
<i>New</i>	Creates a new File.
<i>Open</i>	Opens a file to the editor. Includes options for Source, Project, Output, RTF, Flow Chart, Hex or Text. Ctrl+O is the shortcut.
<i>Close</i>	Closes the file currently open for editing. Note that while a file is open in PCW for editing, no other program may access the file. Shift+F11 is the shortcut.
<i>Close All</i>	Closes all files open in the PCW
<i>Save</i>	Saves the file currently selected for editing. Ctrl+S is the shortcut.
<i>Save As</i>	Prompts for a file name to save the currently selected file.
<i>Save All</i>	All open files are saved
<i>Encrypt</i>	Creates an encrypted include file. The standard compiler #include directive will accept files with this extension and decrypt them when read. This allows include files to be distributed without releasing the source code.
<i>Print</i>	Prints the currently selected file.
<i>Recent Files</i>	The right-side of the menu has a Recent Files list for commonly used files.
<i>Exit</i>	The bottom of the menu has an icon to terminate PCW.

2. Window Editor is used to type and edit the user desired C programming coding. A program is made up the following four elements in a file:
 - i. Comment
 - ii. Pre-Processor Directive
 - iii. Data Definition
 - iv. Function Definition

Every C program must contain a main function which is the starting point of the program execution. The program can be split into multiple functions

according to their purpose and the functions could be called from main or the sub-functions. In a large project functions can also be placed in different C files or header files that can be included in the main C file to group the related functions by their category.

3. To utilize the Compile Menu Ribbon, click the icon or button

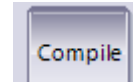


Table 3.2: Types of Activities and Respective Description of “Compile Menu Ribbon”

Type of Activity	Description of Usage
<i>Compile</i>	Compiles the current project in status bar using the current compiler.
<i>Build</i>	Compiles one or more files within a project.
<i>Compiler</i>	Pull-down menu to choose the compiler needed
<i>Lookup Part</i>	Choose a device and the compiler needed will automatically be selected.
<i>Program Chip</i>	Lists the options of CCS ICD or Mach X programmers and will connect to SIOW program.
<i>Debug</i>	Allows for input of .hex and will output .asm for debugging.
<i>C/ASM List</i>	Opens listing file in read-only mode. Will show each C source line code and the associated assembly code generated.
<i>Symbol Map</i>	Opens the symbol file in read-only mode. Symbol map shows each register location and what program variables are saved in each location.
<i>Call Tree</i>	Opens the tree file in read-only mode. The call tree shows each function and what functions it calls along with the ROM and RAM usage for each.
<i>Statistics</i>	Opens the statistics file in read-only mode. The statistics file shows each function, the ROM and RAM usage by file, segment and name.
<i>Debug File</i>	Opens the debug file in read-only mode. The listing file shows each C source line code and the associated assembly code generated.

4. To utilize the Help Menu, click the icon or button



Table 3.3: Types of Activities and Respective Description of “Help Menu”

Type of Activity	Description of Usage
<i>Contents</i>	Help File table of contents.
<i>Index</i>	Help File index.
<i>Keyword at Cursor</i>	Index search in Help File for the keyword at the cursor location. Press F1 to use this feature.
<i>Debugger Help</i>	Help File specific to debugger functionality.
<i>Editor</i>	Lists the Editor Keys available for use in PCW. Shift+F12 will also call this function help file page to quick review.
<i>Data Types</i>	Specific Help File page for basic data types.
<i>Operators</i>	Specific Help File page for table of operators that may be used in PCW.
<i>Statements</i>	Specific Help File Page for table of commonly used statements.
<i>Preprocessor Commands</i>	Specific Help File page for listing of commonly used preprocessor commands.
<i>Built-in-Functions</i>	Specific Help File page for listing of commonly used built-in functions provided by the compiler.
<i>Technical Support</i>	Technical Support wizard to directly contact Technical Support via email and the ability to attach files.
<i>Check for Software Updates</i>	Automatically invokes Download Manager to view local and current version of software.
<i>Internet</i>	Direct links to specific CCS website pages for additional information.
<i>About</i>	Shows the version of compiler(s) and IDE installed.

3.3.2 PICKit 2 V2.61

The PICKit 2 Programmer application allows user to program all supported devices listed in the PICKit 2 Readme file. The programming interface appears as shown in Figure 3.6. Its controls are listed in the following sections.

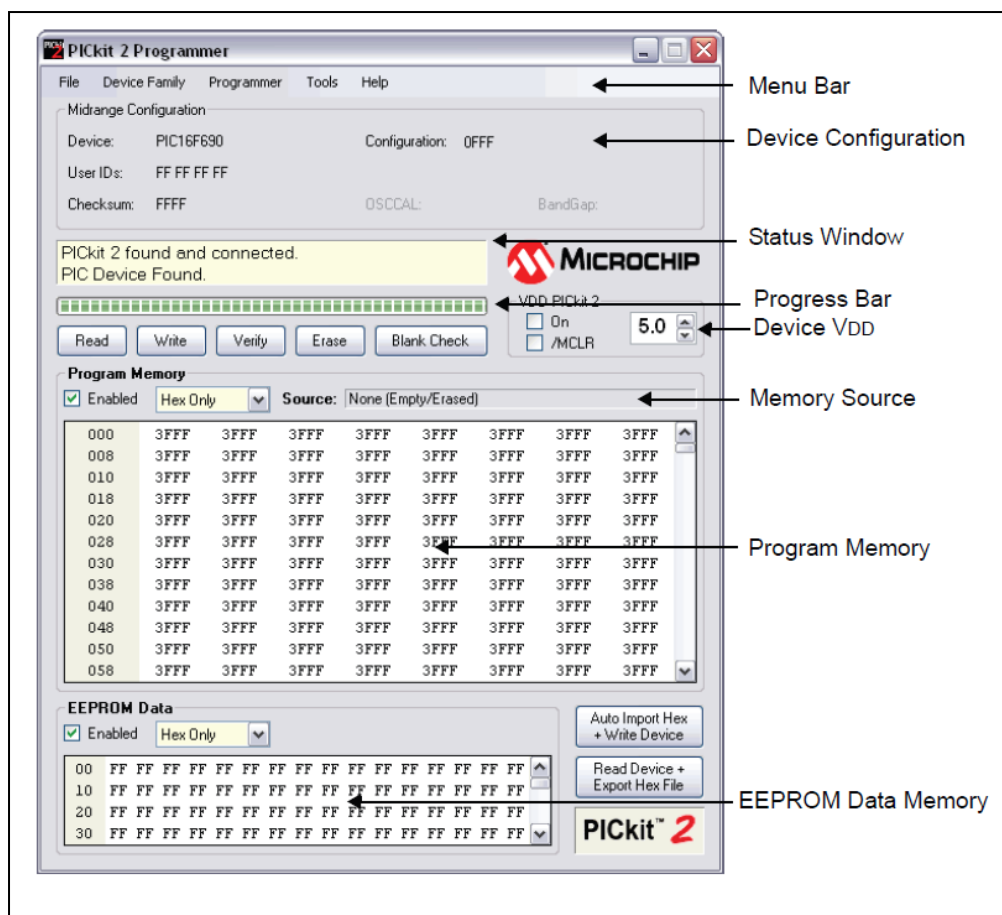


Figure 3.6: PICkit™ 2 Programmer Application

a. **Menu Bar**

The Menu bar selects various functions of the PICkit2 Programmer application. A summary of the functions are:

i. **File**

Table 3.4: Overview of “File” in Menu Bar

Content	Description
<i>Import Hex</i>	Import a hex file for programming. The hex file format INHX32 is supported.
<i>Export Hex</i>	Export a hex file read from a device. The hex file is created in the INHX32 format.

<i>File History</i>	Up to the last four hex files opened are displayed with their file path. These recent hex files may be selected to quickly import them. Note that the file history will initially be blank on a new installation until a hex file is imported.
<i>Exit</i>	Exit the program.

ii. Device Family

Select a device family to search for a connected device in that family. Selecting the device family of the current part will clear all device data. Some families which cannot be auto-detected (such as Baseline) will bring up a drop down box from which support devices may be selected.

iii. Programmer

Table 3.5: Overview of “Programmer” in Menu Bar

Content	Description
<i>Read Device</i>	Reads program memory, data EEPROM memory, ID locations and configuration bits.
<i>Write Device</i>	Writes program memory, data EEPROM memory, ID locations and configuration bits.
<i>Verify</i>	Verifies program memory, data EEPROM memory, ID locations and configuration bit read from the target MCU against the code stored in the programming application.
<i>Erase</i>	Performs a Bulk Erase of the target MCU. OSCCAL and band gap values are preserved on parts with these features.
<i>Blank Check</i>	Performs a Blank Check of program memory, data EEPROM memory, ID locations and configuration bits.
<i>Verify on Write</i>	When checked, the device will be immediately verified after programming on a Write (recommended). When unchecked, the device will be programmed but not verified on a Write.
<i>Hold Device in Reset</i>	When checked, the MCLR (V_{PP}) pin is held low (asserted). When unchecked, the pin is released (tri-stated), allowing an external pull-up to bring the device out of Reset.
<i>Write on PICkit Button</i>	When checked, a Write operation will be initiated by pressing the PICkit 2 push button.

iv. Tools

Table 3.6: Overview of “Tools” in Menu Bar

Content	Description
<i>Enable Code Protect</i>	Enables code protection features of the microcontroller on future Write operations.
<i>Enable Data Protect</i>	Enables data protection feature of microcontrollers with data EEPROM memory on future Write operations.
<i>Set OSCCAL</i>	Allows the OSCCAL value to be changed for devices where it is stored in the last location of Program Memory.
<i>Target V_{DD} Source</i>	<p><i>Auto-Detected</i> – The PICKit 2 will automatically detect whether the target device has its own power supply or needs to be powered by the programmer on each operation.</p> <p><i>Force PICKit 2</i> – The PICKit 2 will always attempt to supply V_{DD} to the target device.</p> <p><i>Force Target</i> – The PICKit 2 will always assume the target has its own power supply.</p>
<i>Calibrate V_{DD} & Set Unit ID</i>	Opens a wizard that steps the user through calibrating the PICKit 2 V _{DD} supplied voltage so it is more accurate, and optionally assigning a Unit ID to identify between multiple PICKit2 devices.
<i>Fast Programming</i>	When checked, the PICKit 2 will attempt to program the device as fast as possible. When unchecked, the PICKit 2 will slow down ICSP communication. This may be helpful for targets with loaded ICSP lines.
<i>Check Communication</i>	Verifies USB communication with PICKit 2 and ICSP communication with a target device by attempting to identify the connected device by its device ID.
<i>UART Tool</i>	Puts the PICKit 2 in UART Mode and opens a terminal-like interface for communication with a PIC MCU device program through the USART pins.
<i>Troubleshoot</i>	Opens a wizard to help with troubleshooting connectivity from the PICKit 2 to the target device. This is most useful where the programmer is unable to detect the target device at all.
<i>Download PICKit 2 Programmer Operating System</i>	Performs a download of the PICKit2 operating system (firmware)

v. Help

Table 3.7: Overview of “Help” in Menu Bar

Content	Description
<i>PICkit 2 User's Guide</i>	Attempts to launch the user's guide PDF.
<i>44-Pin Demo Board Guide</i>	Attempts to launch the 44-Pin Demo Board User's Guide PDF.
<i>LPC Demo Board Guide</i>	Attempts to launch the Low Pin Count Demo Board User's Guide PDF.
<i>PICkit 2 Programmer on the web</i>	Opens www.microchip.com/pickit2 in the default web browser.
<i>Readme</i>	Opens the PICkit s Readme.txt file
<i>About</i>	Opens a dialog with the PICkit 2 Programmer application version, device file version and firmware version.

b. Device Configuration

The Device Configuration window displays the device, User ID, Configuration Word and Checksum. It also displays OSCCAL and Band Gap for parts with those features. For baseline (12-bit core) devices and serial EEPROM devices, user must select the device from the Device drop-down menu. All other part family devices will be detected by their device ID and the part name will be displayed on the Device line.

c. Status Window

The status window displays text status of the operations in progress. If an operation is successful, the status window will display a green background. If an operation fails, the status window will display red. If an operation alerts a caution, the status window will display yellow.

d. Progress Bar

The progress bar displays the progress of an operation

e. Device V_{DD}

The PICkit 2 V_{DD} may be turned on and off by clicking the checkbox “On”/ the voltage may be set in the box on the right either by typing it directly or using the up / down arrows to adjust it a tenth of a volt at a time. The maximum and minimum allowed voltages will vary depending on the target device.

f. Device /MCLR State

When the box is checked the target device will be held in Reset. When unchecked, the target circuit is allowed to pull /MCLR up to V_{DD} to release the device from Reset. This function can be used to prevent a device from executing code before and after programming.

g. Memory Source

The Source bar displays the source of the currently loaded device data. If read from a hex file, it will display the hex file name. If read from a device, it will display the part name. *None (Empty / Erased)* indicates the buffers are empty, and it will display *Edited* once Program Memory or Data EEPROM Memory has been edited in the window.

h. Program Memory

Program code can be loaded into the PICkit2 Programmer application by selecting File>Import HEX to import a hex file or by clicking Read to read the

device memory. The original of the code is displayed in the Source block. The Program Memory window displays the program code in hexadecimal. The code may be edited in the window.

i. Data EEPROM Memory

Similar to Program Memory above, data EEPROM code can be loaded into PICkit 2 Programmer application by selecting File>Import HEX to import a hex file or by clicking Read to read the device memory. The origin of the code is displayed in the Source block. The Data EEPROM Memory window displays the program code in hexadecimal. This code may be edited in the window.

3.4 Hardware Review

This part is to review and discuss the entire relevant hardware configurations which would utilize in FWMR project. The Overall Hardware Configuration of FWMR Block Diagram as shown in Figure 3.7. All the important components and devices included for the FWMR project as mentioned following:

- i. PIC Brain Board (SK40C) with PIC 18F4580
- ii. USB ICSP PIC Programmer V2010 (UIC00B)
- iii. Rainbow Cable
- iv. DC Gear Motor 12V with Motor Driver IC L293D
- v. Servo Motor (RC C40R)
- vi. Ultrasonic Sensor (SN-LV-EZ1 or MaxSonar EZ1)
- vii. Digital Compass Module (HMC6352)

All these components act as basic necessities due to construct a FWMR to achieve the straight line movement purpose. Therefore, every sub-part of the components will further discussed in respective following section.

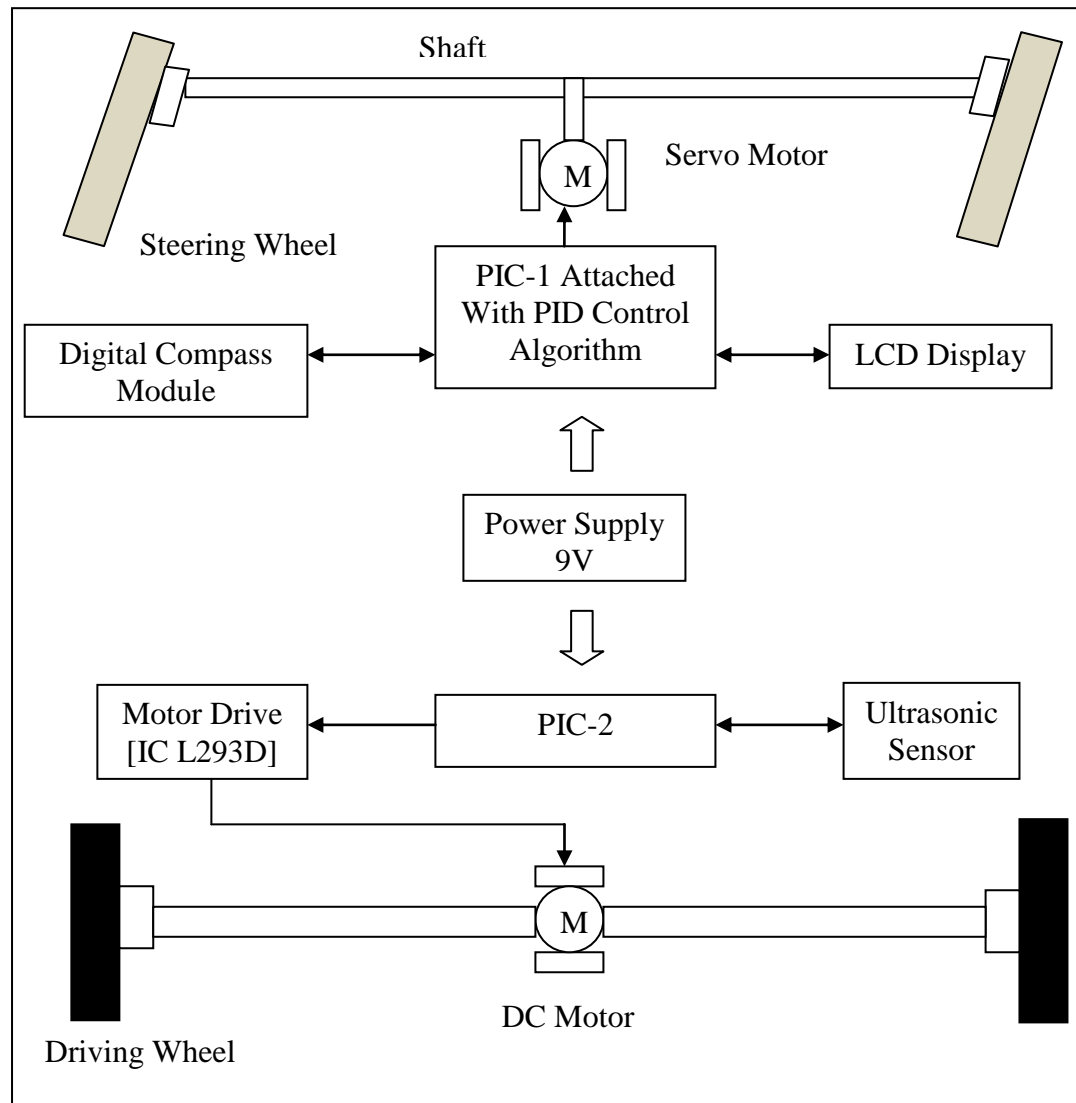


Figure 3.7: Overall Hardware Configuration of FWMR Block Diagram

3.4.1 PIC Brain Board (SK40C) with PIC 18F4550

SK40C has been selected as the FWMR's PIC microcontroller Brain Board which can be used to program directly without anymore to use external programmer to program the PIC microcontroller. Users are able to utilize the function of PIC by directly plugging in the I/O components [11]. With the aid of UIC00A connector on board or UIC00B ICSP PIC Programmer, programming can be directly set for developing the project with this kit right away. It offer plug and use features as mentioned following:

- a. ICSP connector for UIC00A or UIC00B – simple and fast method to load program
- b. Perfectly fit for 40 pins 16F and 18F PIC
- c. 2 × programmable switch
- d. 2 × LED indicator
- e. Exchangeable Crystal with 20MHZ
- f. Existing pad for 16 × 2 characters LCD display
- g. UART connection to interface with other controller or even computer
- h. All 33 I/O pins are nicely labeled
- i. Maximum current is 0.5A
- j. USB on board

In Figure 3.8 is shown as the top view of board layout for the PIC Brain Board of SK40C and Table 3.8 indicated all the parts in the Brain Broad which has been specified with regarding 'Label' respectively. All these 'Label' parts may include their function or usage respectively in the Table 3.8.

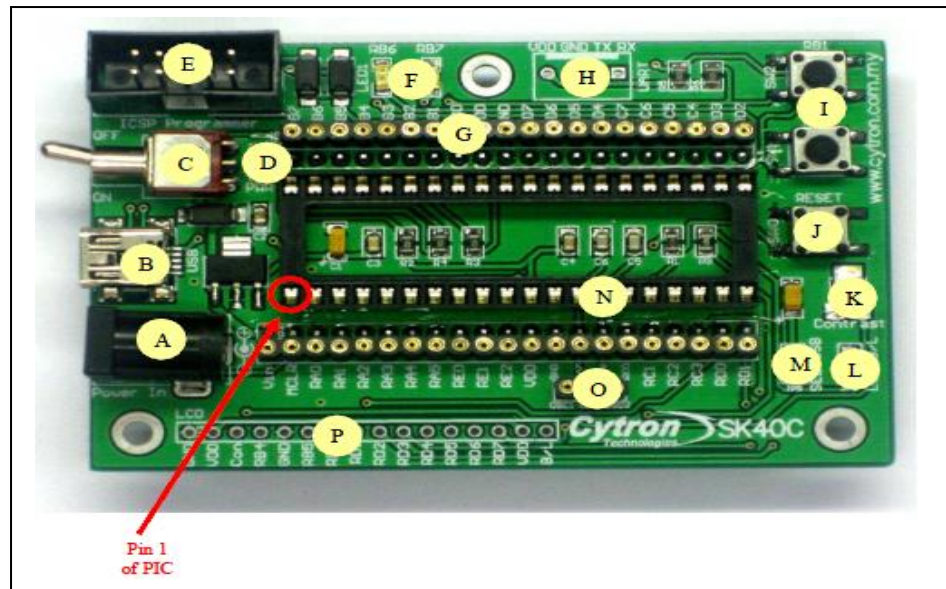


Figure 3.8: Top View of SK40C Brain Board Layout

Table 3.8: Each Part of SK40C with Function Respectively

Label	Function	Label	Function
A	DC Power Adaptor Socket	I	Programmable Push Button
B	USB Connector	J	Reset Button
C	Toggle Switch for Power Supply	K	LCD Contrast
D	Power indicator LED	L	JP8 for LCD Backlight
E	Connector for UIC00A Programmer	M	JP9 for USB
F	LED Indicator	N	40 Pin IC Socket for PIC MCU
G	Header Pin and Turn Pin	O	Turn Pin for Crystal
H	UART Connector	P	LCD Display

The SK40C of PIC Brain Board Start-up Kit comes without placing the microcontroller into the board due to provide the freedom for the user or researcher to choose a desired PIC type. In this FWMR project case, PIC 18F4550 has been

selected as the robot brain controller to react the output values obtained from the encoders and Digital Compass Module, then with the aid from PID control algorithm to justify and modify the error value, then sum up to a response for further action.

PIC 18F4550 is an ideal microchip controller for low power with nano-Watt and connectivity applications that benefit from the availability of three serial ports: FS-USB (12M bit/s), I²C and SPI (up to 10M bit/s) and asynchronous (LIN capable) serial port (EUSART). Large amounts of RAM memory for buffering and Enhanced Flash program memory make it ideal for embedded control and monitoring applications that require periodic connection with a (legacy free) personal computer via USB for data upload/download and/or firmware updates [12].

PIC 18F4550 configuration has been shown in Figure 3.9 to indicate overall 40 pins with each pin location purpose. The features or parameters of PIC 18F4550 brain microcontroller indicated as follow

Pin Count: 40

- a. Program Memory (KB) 32 Flash
- b. CPU Speed (MIPS): 12 RAM Bytes 2,048
- c. Data EEPROM (bytes): 256
- d. Digital Communication Peripherals: 1-A/E/USART, 1-MSSP (SPI/I²C)
- e. Capture/Compare/PWM Peripherals: 1 CCP, 1 ECCP
- f. Timers: 1 × 8-bit, 3 × 16-bit
- g. ADC: 13ch, 10-bit
- h. Comparators: 2
- i. USB (speed, compliance): 1, Full Speed, USB 2.0
- j. Operating Voltage Range (V): 2 to 5.5

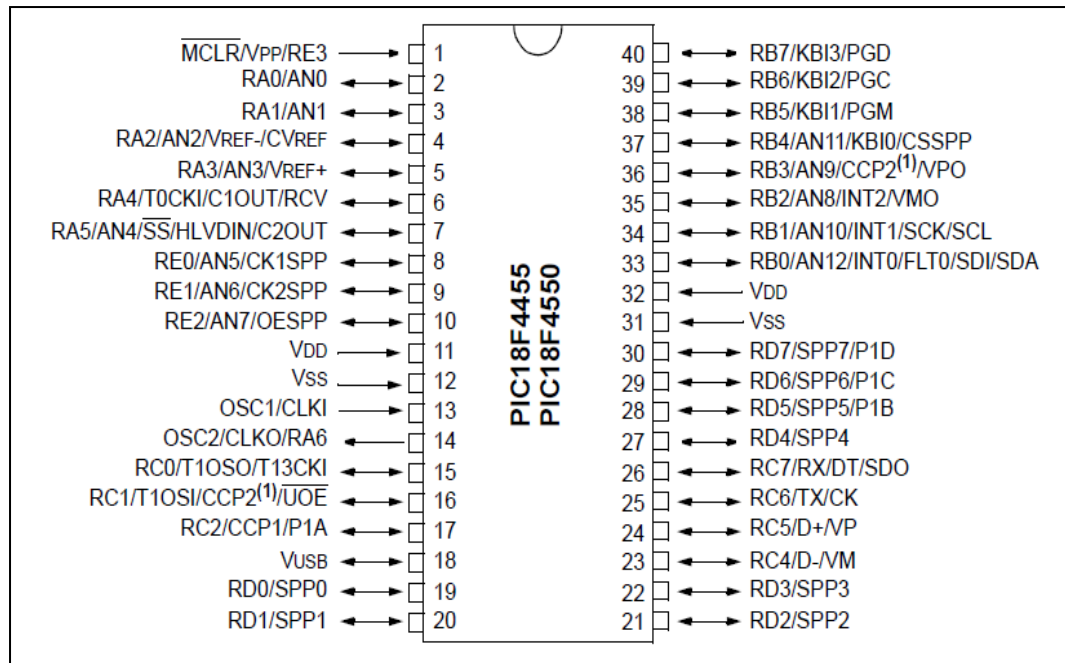


Figure 3.9: 40-Pin PDIP of PIC 18F4550 Configuration

3.4.2 USB ICSP PIC Programmer V2010 (UIC00B)

UIC00B is an enhanced version of UIC00A with offering low cost yet reliable and user friendly PIC USB programmer solutions for developers and researchers. It is designed to program popular Flash PIC MCU which includes most of the PIC family. Besides 8-bit, it can also program 16-bit and 32-bit PIC MCU. On board ICSP in abbreviation of 'In Circuit Serial Programming' connector offers flexible methods to load program, UART Tool and Logic Tool. It supports on board programming which eliminate the frustration of plug-in and plug-out of PIC MCU. This also allow user to quickly program and debug the source code while the target PIC is on the development board. Since USB port is commonly available and widely used on Laptop and Desktop PC, UIC00B is designed to plug and play with USB connection. This programmer obtained its power directly from USB connection, thus there is no external power supply is required, making it a truly portable programmer.

This programmer is ideal for field and general usage. UIC00B is designed with capabilities and features indicated as follow [13]:

- a. Compatible with Window XP, Vista and 7
- b. Compatible with Microchip's PIC Kit 2
- c. Powered directly from USB port
- d. No external power required for UIC00B to function
- e. Compatible with PIC Kit 2's Logic Tool and UART Tool
- f. UIC00B supports on-board programming which eliminates the need of plug-in and plug-out of PIC MCU
- h. Allow user to modify the program without removing the PIC from the development board
- i. This programmer comes with mini USB cable and rainbow cable

Figure 3.10 is shown as the top view of board layout for USB ICSP PIC Programmer V2010 and Table 3.9 indicated all the parts in the UIC00B Programmer which has been specified with regarding 'Label' respectively. All these 'Label' parts may include their function or usage respectively in the Table 3.9.

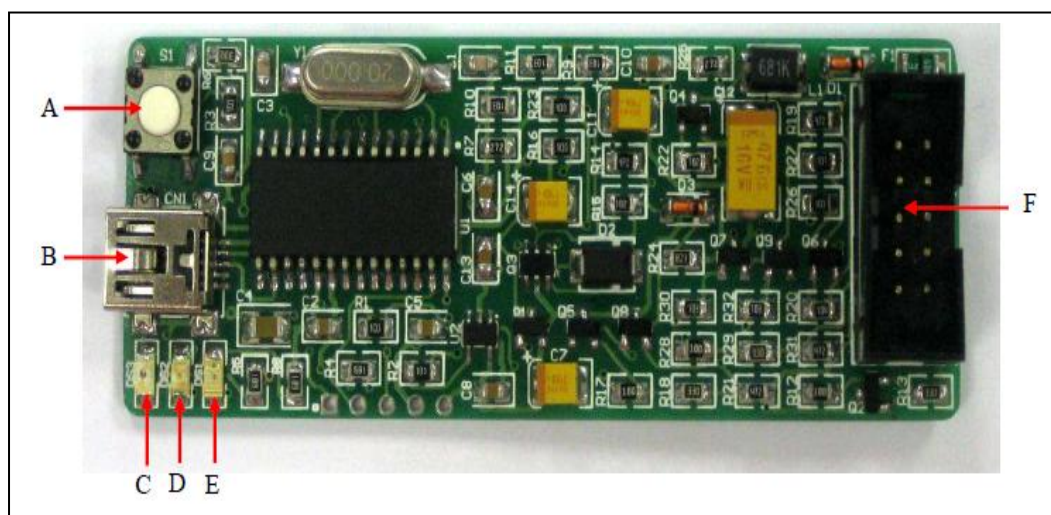


Figure 3.10: Top View of USB ICSP PIC Programmer V2010's Broad Layout

Table 3.9: Each Part of UIC00B Programmer V2010 with Function Respectively

Label	Function	Label	Function
A	Switch to Initiate Write Device Programming	D	Target Indicator LED (Orange)
B	Mini USB Port Socket	E	Busy Indicator LED (Red)
C	Main Power Supply Indicator LED (Green)	F	IDC Box Header for Programming Connector

3.4.3 Rainbow Cable

This is a programming cable which especially applies to connect one side of programming cable to box header of UIC00B programmer and other side to box header of development board or target device to be program. This tool can give a big help to program a PIC microcontroller plugged-in the SK40C Brain Board in fast speed.



Figure 3.11: Rainbow Cable or Programming Cable

3.4.4 DC Gear Motor with Motor Driver IC L293D

The prime aim by using DC gear motor in the FWMR project is to apply for the robot driving purpose in order that it is able to move from a starting point to an ending point. Figure 3.12 is shown as the type of DC gear motor with its outlook and Figure 3.13 to indicate the installation of DC gear motor to two back driving wheels in FMWR. DC gear motor is a vital tool due to rotate two driving wheels at the back of FWMR with high torque output and fast RPM which is totally compatible and suit to use in robot system. As a result, it is very essential to know how to control a DC gear motor effectively with a microcontroller. The specification of DC gear motor as shown below:

- i. Voltage Supply: DC 12V
- ii. Output Power: 1.1Watt
- iii. Rated Speed: 1.3RPM
- iv. Rated Torque: 127.4mN.m
- v. Sample Application: mobile robot, educational robot, etc.

DC gear motor is electromechanical device that converts electrical energy into mechanical energy that used to move or start-up the robot system movement. It has two wires or pins where used to connect with the power supply, then the shaft may rotates. The direction of rotation can be reversed by just reversing the polarity of input of the motor.

Microcontroller ports are not powerful enough to drive the DC gear motor directly. Thus, a motor driver IC L293D chip is one of the types of Integrated Circuits (ICs) which is very easy and safe to use. The pin configuration as shown in Figure 3.14 and Table 3.10 shown as the behavior of motor to be functioned for various input conditions. The connection circuit of DC gear motor and IC L293D to PIC microcontroller as displayed in Figure 3.15.

The specifications of IC L293D have mentioned as following:

- Wide Supply-Voltage Range: 4.5V to 36V
- Separate Input-Logic Supply
- Internal ESD Protection
- Thermal Shutdown
- Output Current 600mA for L293D Per Channel
- Output Clamp Diodes for Inductive Transient Suppression

3.4.4.1 Hardware Review of DC Motor and IC L293D



Figure 3.12: DC Gear Motor

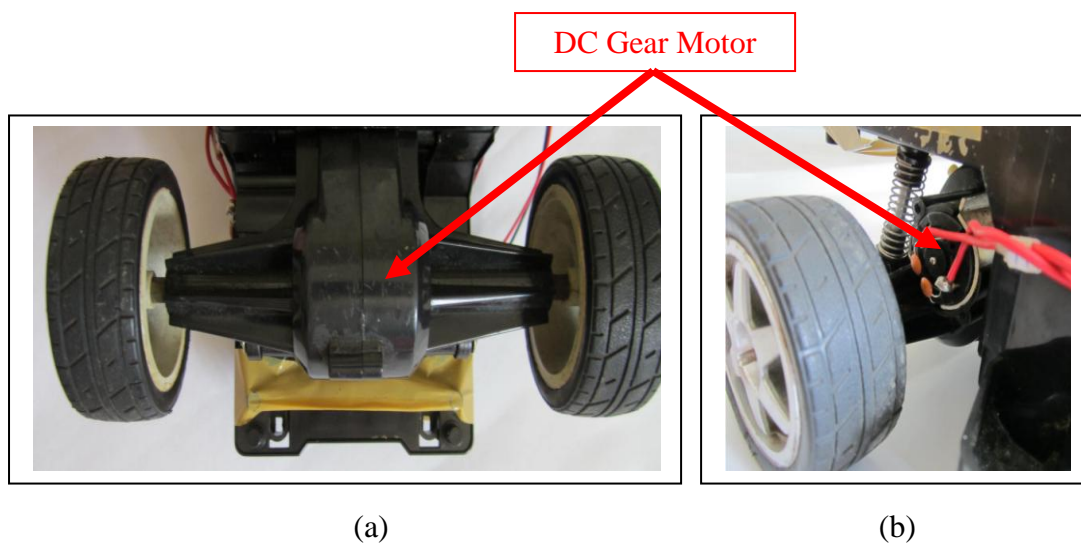


Figure 3.13: Installation of DC Gear Motor to FWMR

(a) Bottom View

(b) Side View

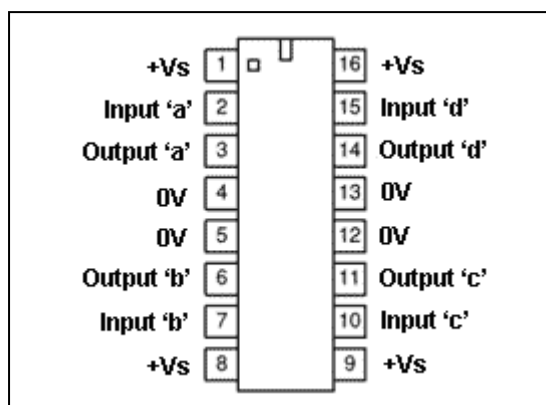


Figure 3.14: IC 293D's Pin Configurations

Table 3.10: Behavior of Motor Operation with Various Input Conditions

Condition	A	B
Stop	Low	Low
Clockwise	Low	High
Anti-Clockwise	High	Low
Stop	High	High

3.4.4.2 Connection Circuit of DC Gear Motor and IC L293D to PIC 2

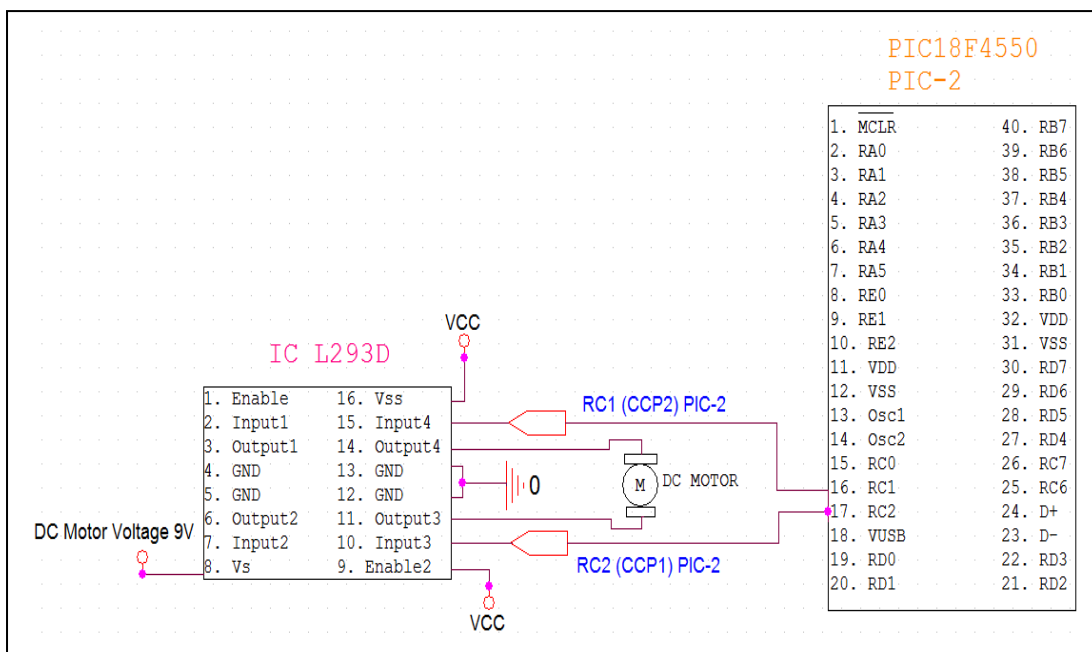


Figure 3.15: Connection Circuit of DC Motor and IC L293D to PIC 2

3.4.4.3 Sequence of DC Gear Motor Operation

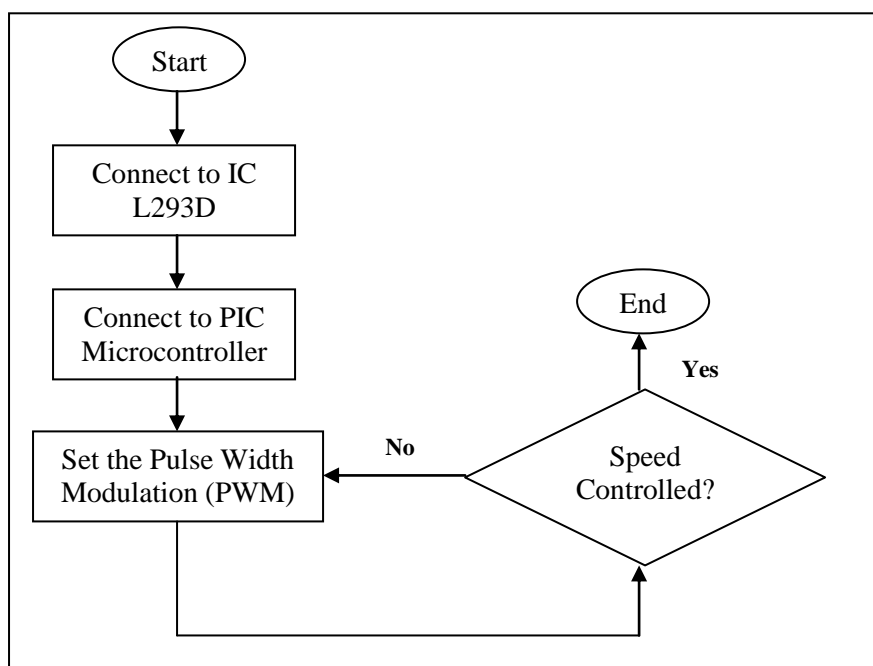


Figure 3.16 Sequence of DC Gear Motor Operation

3.4.4.4 Programming of DC Gear Motor

* Refer to Appendix A (1)

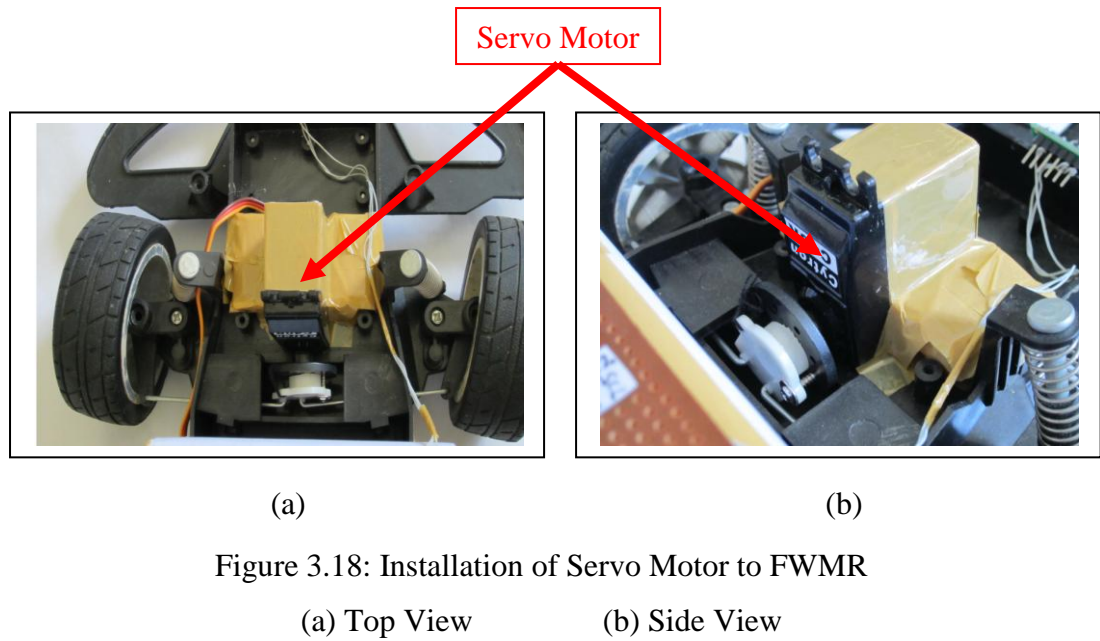
3.4.5 Servo Motor

Servo motor is often sold as a complete module which is used within a position-control or speed-control of feedback control system. Figure 3.17 as shown to indicate the device of one types of servo motor. Motors which applied in a servomechanism must have well-documented characteristics for speed, torque, and power. A servo system has a difference with other motors' application which in that its position feedback is continuous while the motor is running. In this FWMR project, servo motor plays a vital role in order to displace with 0 ° displacement for achieving the robot straight line movement purpose. If the robot tends to other side by certain external factors, servo motor may take responsibility to veer the wheels of the robot back to the original set-point direction position after getting the command from the PIC controller.

3.4.5.1 Hardware Review of Servo Motor



Figure 3.17: Servo Motor (RC C40R)



3.4.5.2 Connection Circuit of Servo Motor to PIC 1

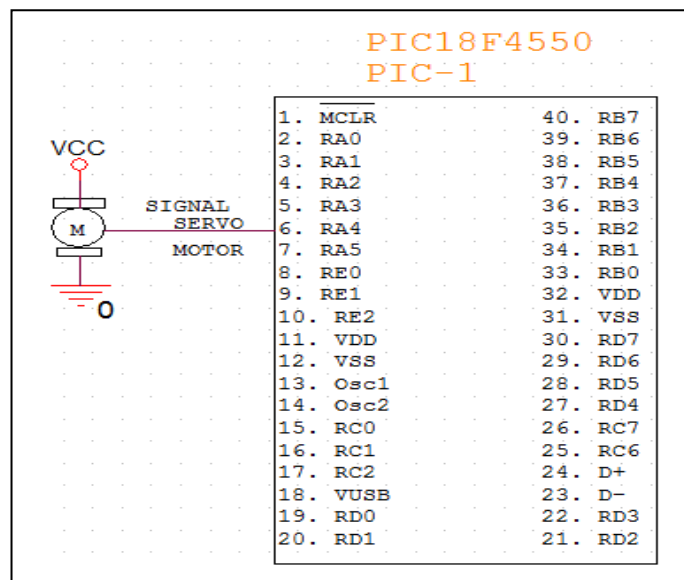


Figure 3.19: Connection Circuit of Servo Motor to PIC 1

3.4.5.3 Sequence of Servo Motor Operation

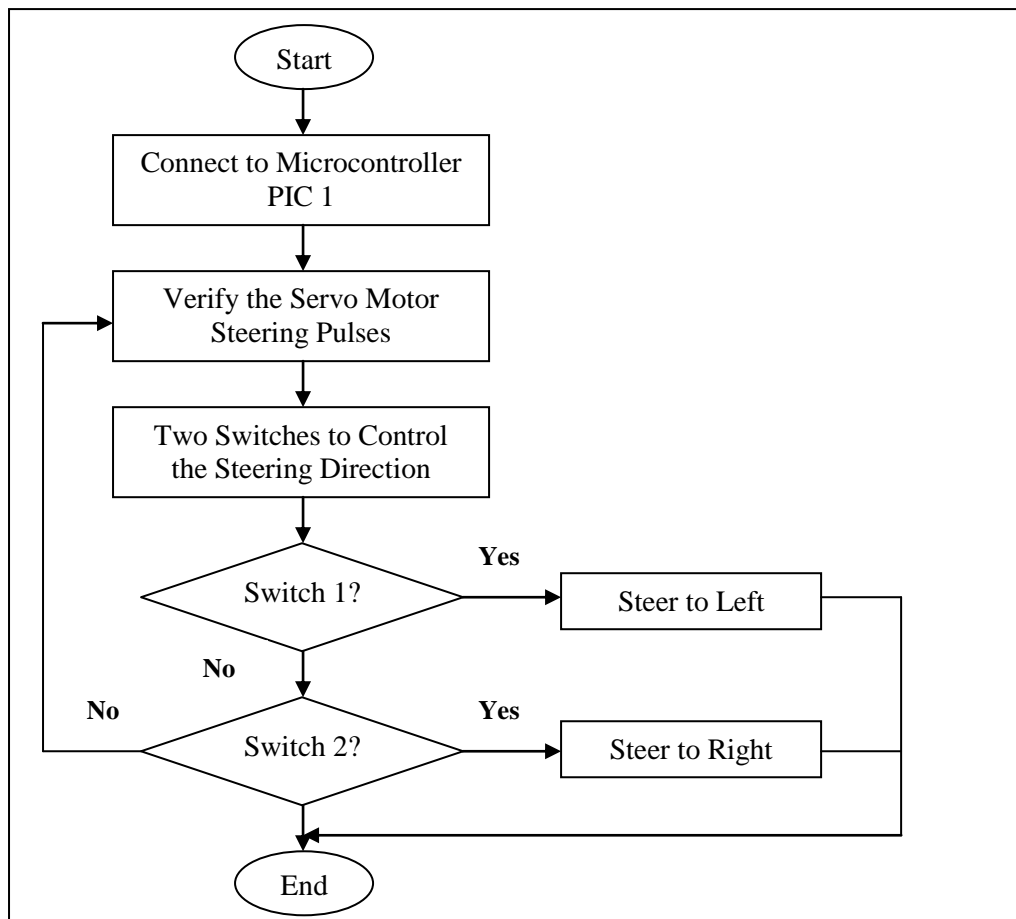


Figure 3.20: Sequence of Servo Motor Steering Direction

3.4.5.4 Programming of Servo Motor Steering Operation

* Refer to Appendix A (2)

3.4.6 Digital Compass Module

In order to ensure the fixed angle of direction of the robot moving, Digital Compass Module (DCM) is necessary to be added up to make sure the angle direction during the robot is moving which to be matched with the original set point motion direction. This module has combined 2-axis magneto-resistive sensors with the required analog and digital support circuits and algorithms for heading computation. Figure 3.21 indicated that a type of breakout board is fully integrated DCM. The specification of Digital Compass Module HMC6352 will be shown as following:

- a. Full integration of 2-axis magnetic sensors and electronics
- b. Firmware included
- c. Small Surface Mount Package (6.5 x 6.5 x 1.5mm, 24-pin LCC)
- d. Low Voltage Operation (2.7 to 5.2V)
- e. I²C 2-wire serial interface
- f. Wide Magnetic Field Range
- g. Supply current:
 - * Sleep Mode ($V_{\text{Supply}} = 3.0\text{V}$) – typical in 1 μA , maximum in 10 μA
 - * Steady State ($V_{\text{Supply}} = 3.0\text{V}$) – typical in 1mA, maximum in 10mA
 - * Steady State ($V_{\text{Supply}} = 5.0\text{V}$) – typical in 2mA, maximum in 10mA

With the aid of Digital Compass Module, the task and aim have been set to achieve with has been shown in Figure 3.24. This operation flow can be expressed detail regarding the way of operation and the main purpose requires to be fulfilled in this FWMR project system.

3.4.6.1 Hardware Review of Digital Compass Module



Figure 3.21: Digital Compass Module

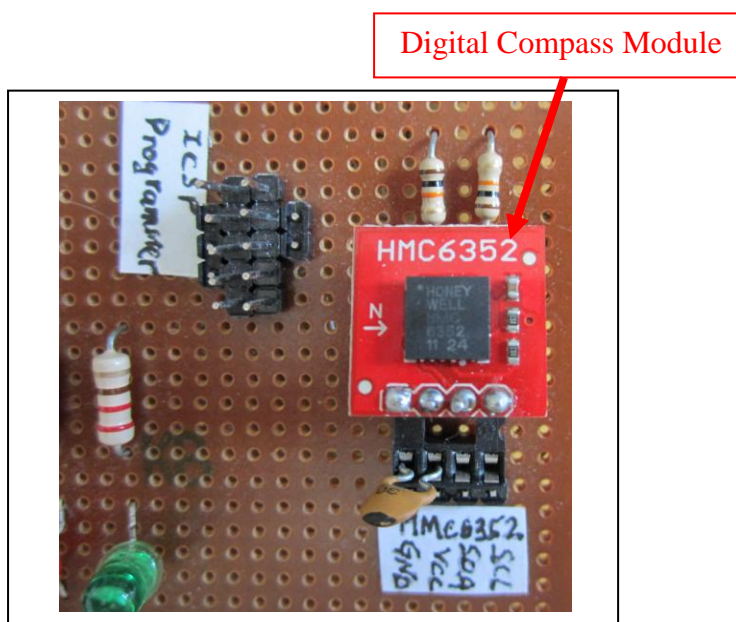


Figure 3.22: Installation of Digital Compass Module HMC6352 to FWMR

3.4.6.2 Connection Circuit of Digital Compass Module to PIC 1

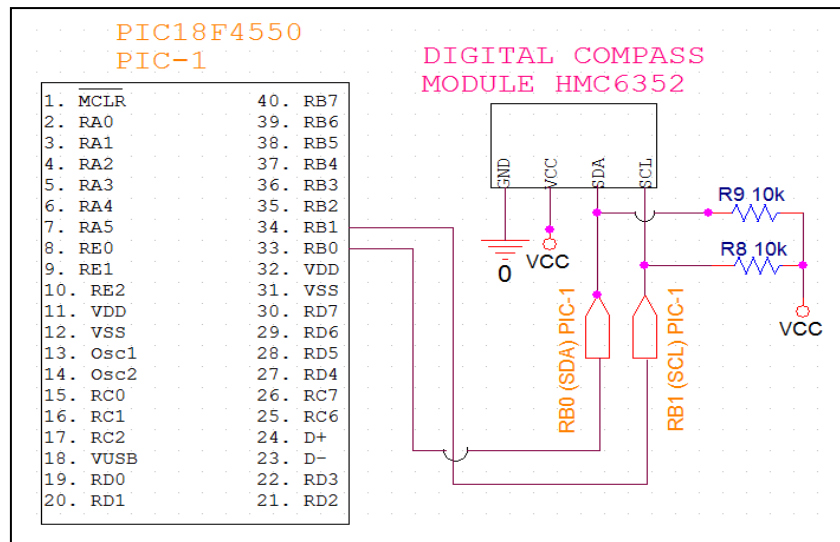


Figure 3.23: Connection Circuit of Digital Compass Module to PIC 1

3.4.6.3 Sequence of Digital Compass Module Function

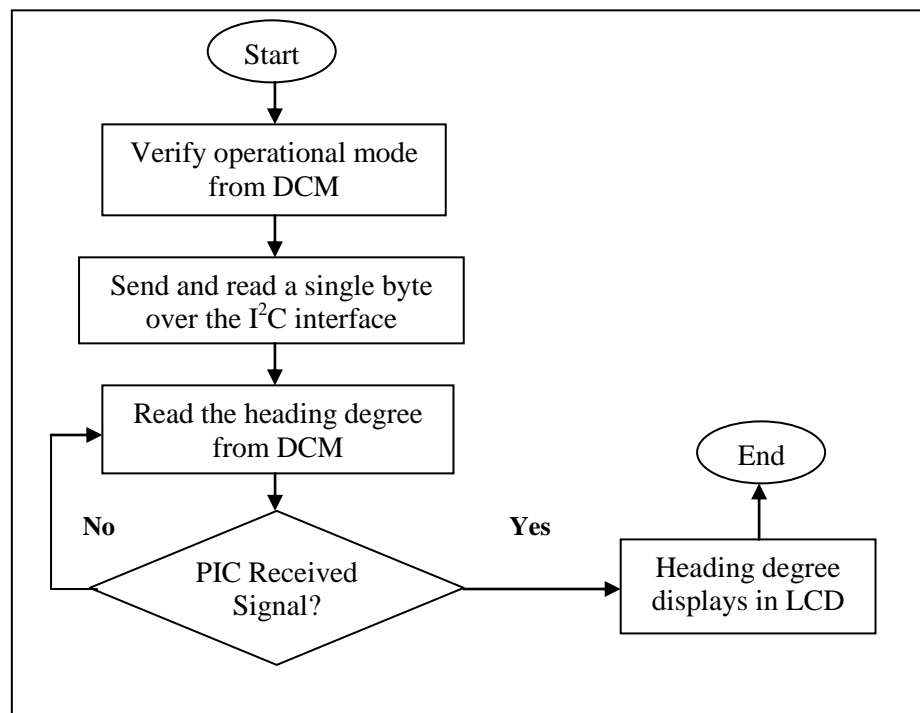


Figure 3.24: Sequence of Digital Compass Module Function

3.4.6.4 Programming of Digital Compass Module Operation

* Refer to Appendix A (3)

3.5 DC Motor Speed Control

A DC geared motor has been chosen in this FWMR project due to achieve its motion for forward and backward purposes. Four switches will be utilized to control and manipulate the DC motor speed rotation to confirm the robot either to be moving fast or slow in forward and backward motion. A programming coding with suitable pulse width modulation or PWM pulse will be set and programmed to the PIC 2 and build up the connection circuit to a motor drive IC L293D to control the DC motor motion speed. When the DC motor is operating, an indicator of green LED will be turned on simultaneously to indicate the robot is moving. In contrast, when an indicator of red LED is turned on to be explained as the robot halted.

3.5.1 Hardware Review of DC Gear Motor, Four Switches and Two LEDs

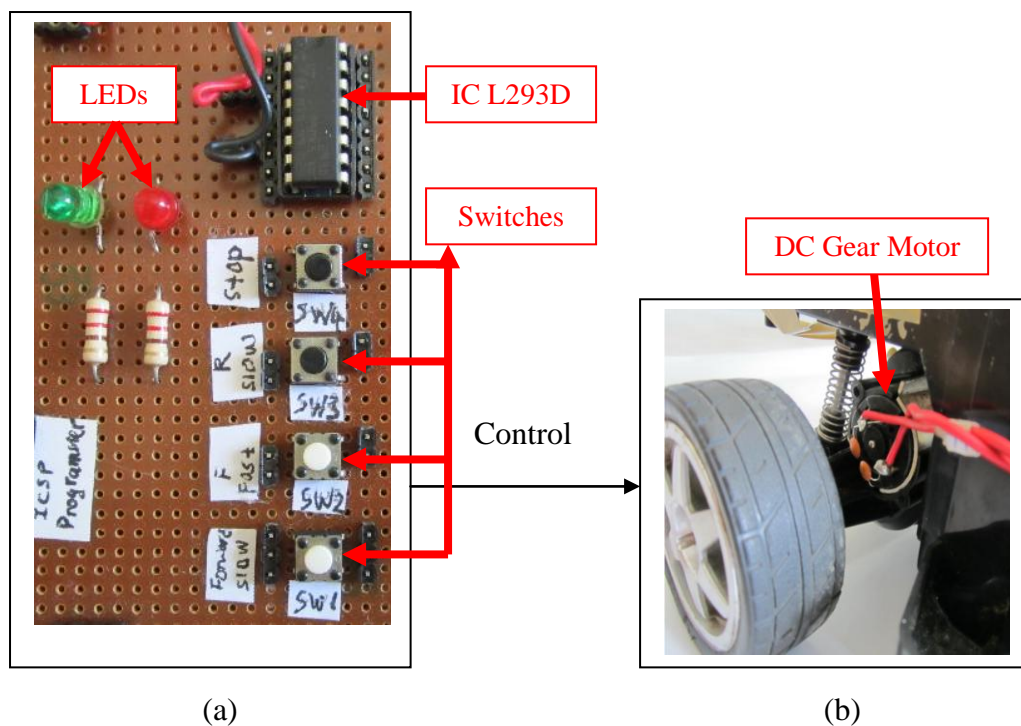


Figure 3.25: Installation of Four Switches to Control DC Motor Speed in FWMR

(a) Four Switches in Board (b) DC Gear Motor

3.5.2 Connection Circuit of DC Gear Motor, Switches and LEDs to PIC 2

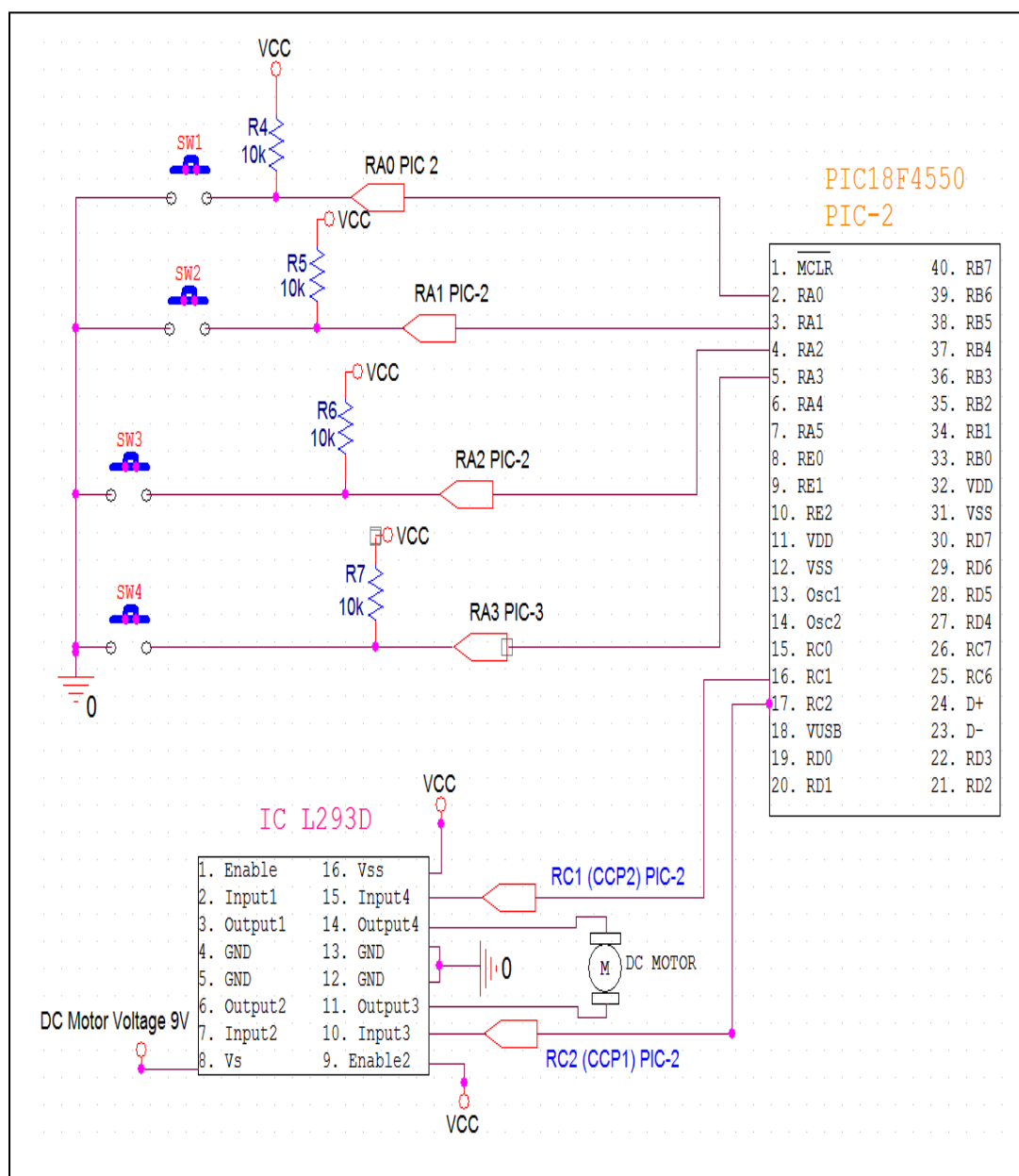


Figure 3.26: Connection Circuit of DC Gear Motor, Four Switches and Two LEDs to PIC 2.

3.5.3 Sequence of Functions for DC Motor Speed Control

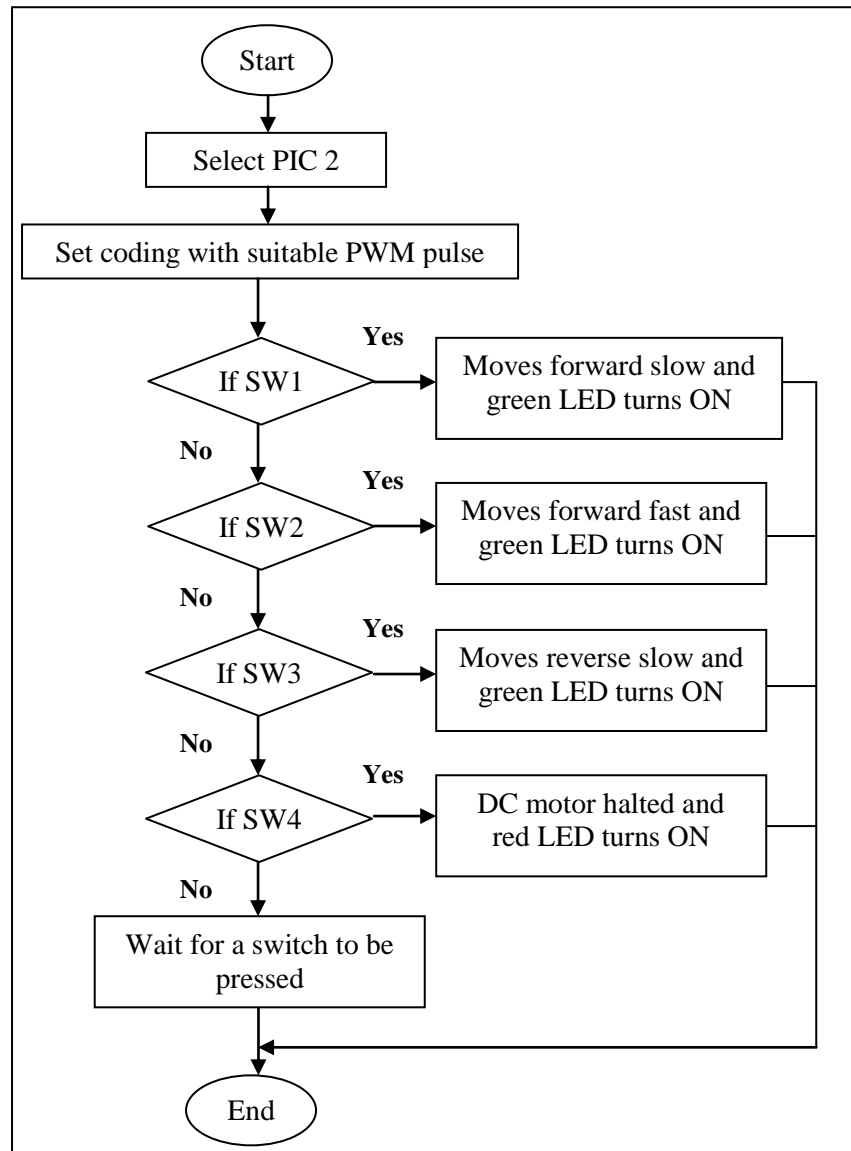


Figure 3.27: DC Motor Speed Control by Four Switches and LEDs Indications

3.5.4 Programming of DC Motor Speed Control by Four Switches and LEDs Indications

* Refer to Appendix A (4)

3.6 Servo Motor Turning Control

Wheeled-robot's steering function is very important due to veer and maintain in certain desired direction. Therefore, radio control (RC) servo motor C40R will be selected to apply in my project for controlling the steering purpose of two front wheels of FWMR. Servo is controlled by sending it a pulse of variable width and its rotation angle is determined by the duration of pulse that is applied to the signal wire. This is called Pulse Width Modulation (PWM) and the servo expects to obtain a pulse every 20ms.

For doing this testing, there are 3 pulses will be concerned initially. A 1.5 ms pulse will make the motor turning to the 0° position or considered as neutral position, where 2.0 ms is the maximum pulse width to turn to -40° of left and the minimum pulse width of 1.0 ms for 40° turning to right. These turning degrees can be observed and discovered vividly from Figure 3.28.

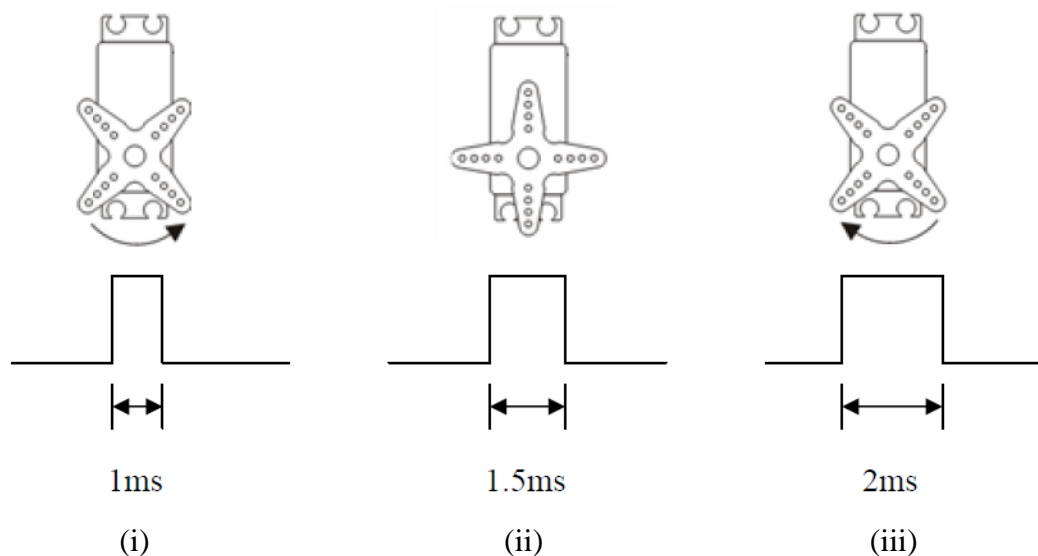


Figure 3.28: Pulse Width for Servo Motor Turning – (i) 1ms pulse to turn right 40°
(ii) 1.5ms pulse to set in neutral position (iii) 2ms pulse to turn left 40°

According to this concept of pulse has been mentioned, the servo can be tested and obtained more turning degrees within the pulse of 1.0 – 2.0 ms range which to be shown in Figure 3.29 and Table 3.11 due to indicate that it is not only rigid for maximum and minimum turning.

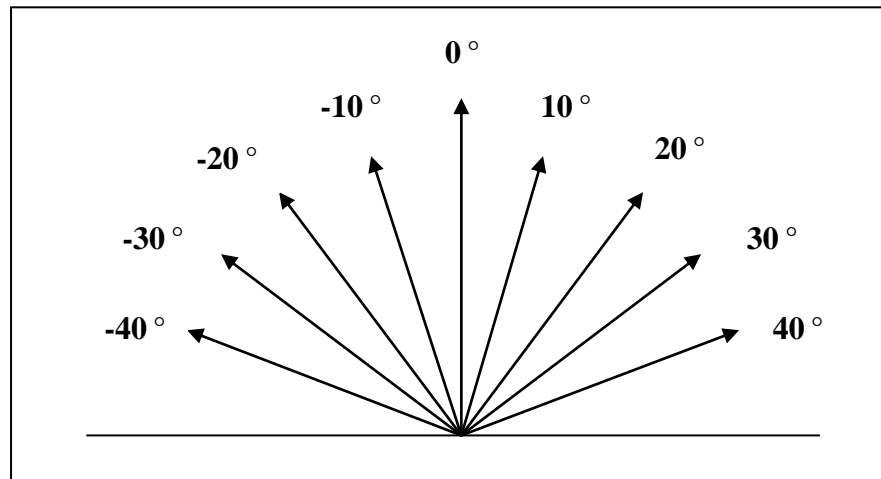


Figure 3.29: Turning Degree of Servo Motor C40R

Table 3.11: Servo Motor Turning in accordance with Pulse Width

Pulse Width (ms)	Servo Motor Turning Degree (°)
2.000	-40
1.875	-30
1.750	-20
1.625	-10
1.500	0
1.375	10
1.250	20
1.125	30
1.000	40

3.6.1 Hardware Review of Servo Motor Turning Control

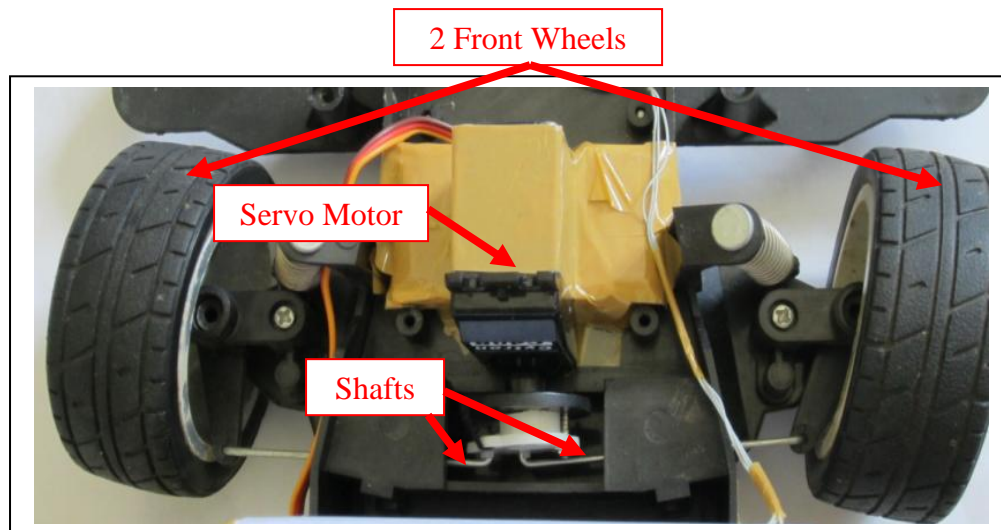


Figure 3.30: Installation of Servo Motor Steering Control to Two Front Wheels

3.6.2 Connection Circuit of Servo Motor to PIC 1 with Two Front Wheels

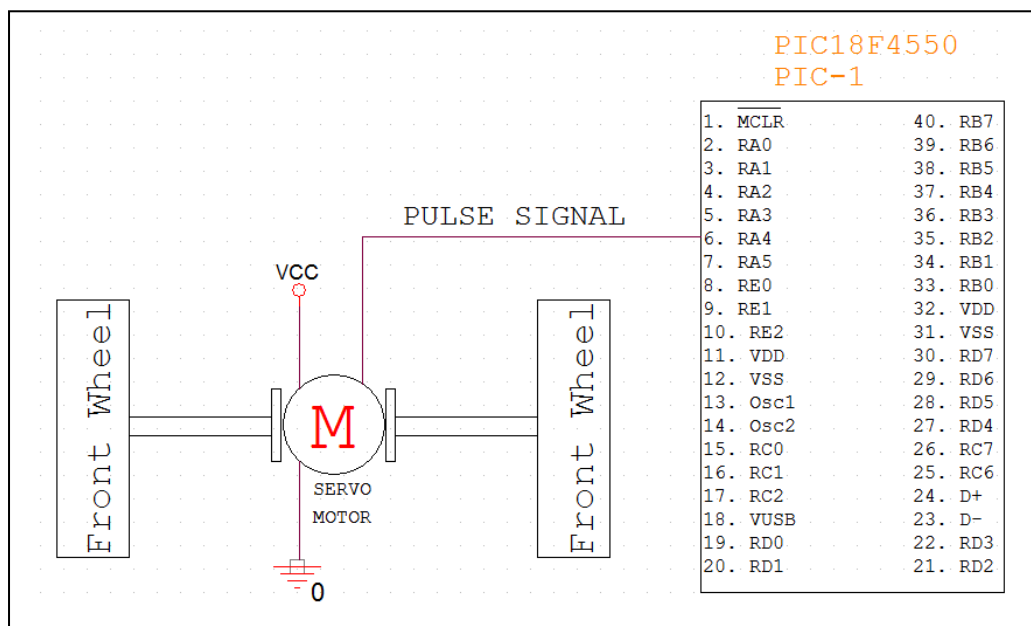


Figure 3.31: Connection Circuit of Servo Motor to Two Front Wheels

3.6.3 Sequence of Servo Motor Turning Control

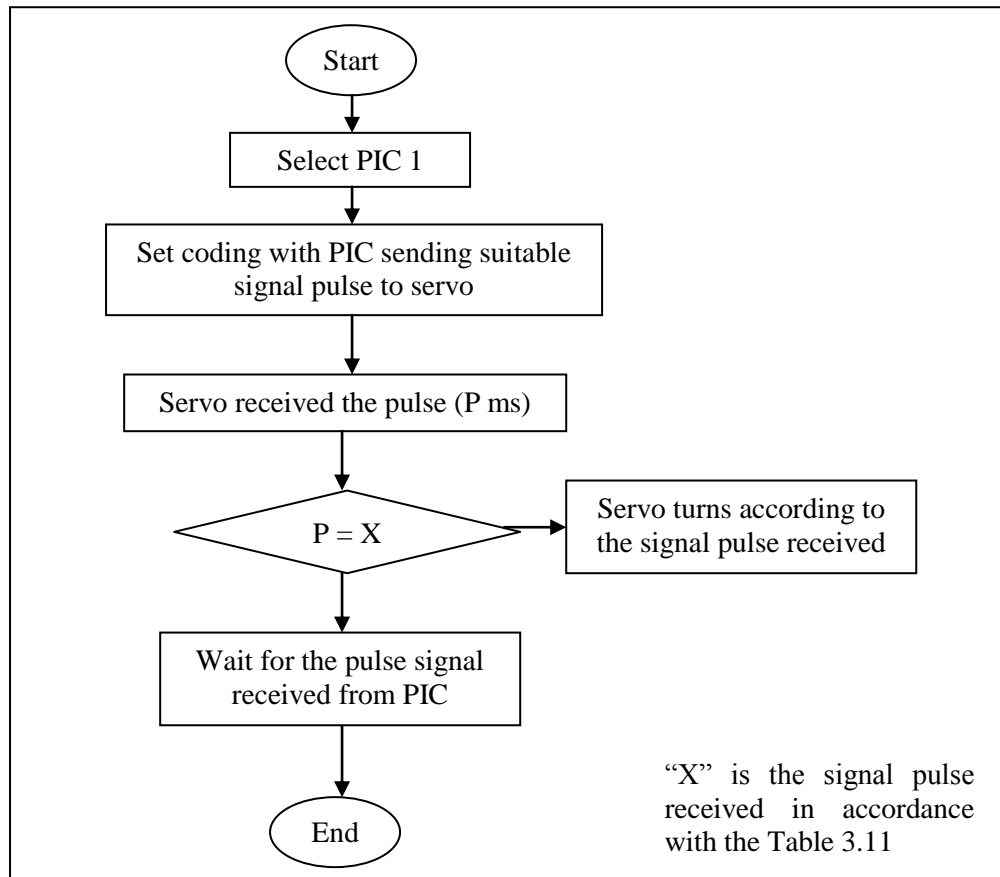


Figure 3.32: Sequence of Servo Motor Turning Control

3.6.4 Programming of Servo Motor Turning Control

* Refer to Appendix A (5)

3.7 Manipulation of Digital Compass Module to Servo Motor

Digital Compass Module is a vital component sensor in order to verify the position of robot moving. In this sub-section, the further testing would be done in accordance to the previous 3.6 sub-section for controlling the servo motor turning degree. With the aid of PIC 1, this microchip controller would be used to receive the data from compass module, then directly feed the data to the LCD to indicate the current position of the robot should be. As a result, the position or direction of the robot can be verified through the observation from the LCD.

A set-point of direction angle or heading degree will be set in coding. Following this, some valuable conditions of heading degree from compass sensor to manipulate the turning degree from servo motor have to be set in order to ensure the FWMR is able to move straight in one set-point direction. In this project, the set-point angle will be specified to North or $359^\circ (\approx 0^\circ)$ of heading degree. Therefore, no matter which initial position of the robot is placed, compass module may give the heading data to the PIC and base on the condition have been set in coding, the robot may find out the desired set-point angle by turning the servo motor with corresponding turning degree to two front wheels. As a result, the robot straight line platform will be carried out to move towards to the north direction for achieving the requirement.

Although the straight line movement requirement has been fulfilled, nonetheless, the robot would move in unstable state because of the compass module is always checking the robot moving direction. The slightly direction change of the robot during its movement would directly cause the heading degree detected by the compass to be varied. Because of this situation, FWMR would move in vibration mode or considered as “sinusoidal waveform” movement based on the fixed set-point direction. Therefore, PID control algorithm will be implemented to reduce and minimize the oscillation of the robot during the straight line movement purpose. This implementation will be further discussed in the following sub-section.

When set-point angle = 359° , the condition of range regarding to the heading degree to adjust and verify the appropriate turning degree has been expressed in Table 3.12 which to indicate the manipulation of heading degree from Digital Compass Module to the servo motor turning degree with specific condition has been done.

Table 3.12: Manipulation of Compass Heading Degree to Servo Motor
Turning Degree

Compass Heading Degree	Servo Motor Turning Degree
$U = 359$	0° , in neutral position, 1.500 ms pulse
$354^\circ \leq U \leq 358^\circ$	$+10^\circ$, turn right, 1.375 ms pulse
$340^\circ \leq U \leq 353^\circ$	$+20^\circ$, turn right, 1.250 ms pulse
$325^\circ \leq U \leq 339^\circ$	$+30^\circ$, turn right, 1.125 ms pulse
$180^\circ \leq U \leq 324^\circ$	$+40^\circ$, turn right, 1.000 ms pulse
$35^\circ \leq U \leq 179^\circ$	-40° , turn left, 2.000 ms pulse
$20^\circ \leq U \leq 34^\circ$	-30° , turn left, 1.875 ms pulse
$5^\circ \leq U \leq 19^\circ$	-20° , turn left, 1.75 ms pulse
$0^\circ \leq U \leq 4^\circ$	-10° , turn left, 1.625 ms pulse

3.7.1 Hardware Review of Digital Compass Module Manipulation to Servo Motor

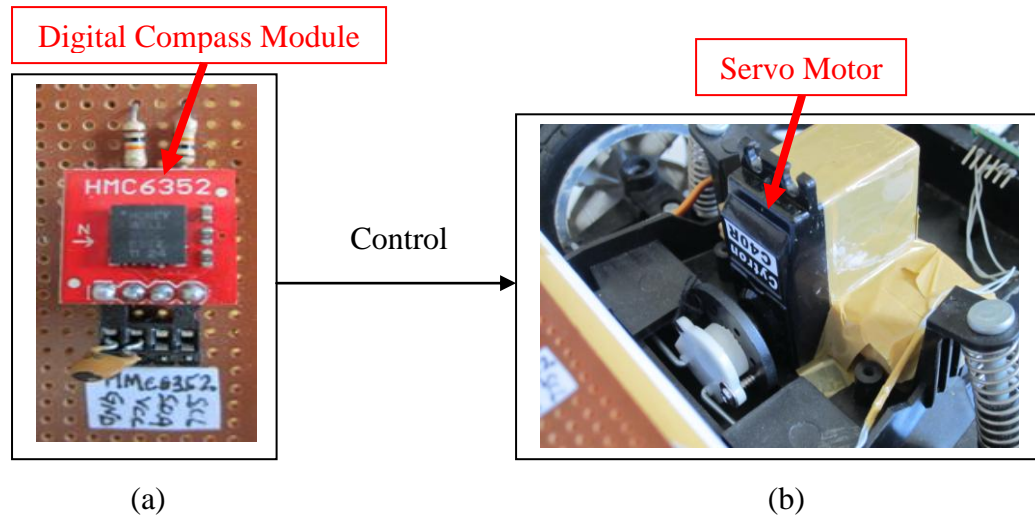


Figure 3.33: Digital Compass Module Heading Degree to Control Servo Motor Turning Degree

(a) Digital Compass Module (b) Servo Motor

3.7.2 Connection Circuit of Digital Compass Module and Servo Motor to PIC 1

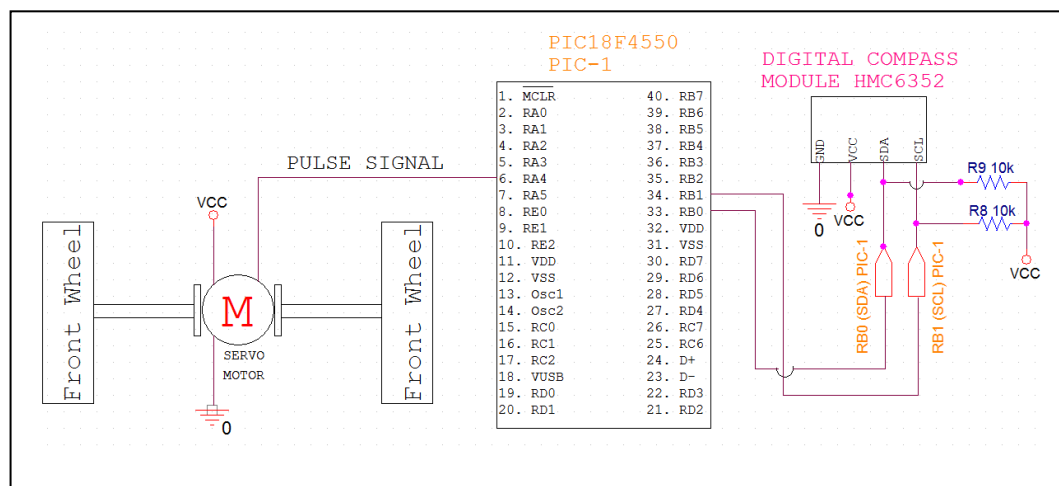


Figure 3.34: Connection Circuit of Digital Compass Module and Servo Motor to PIC 1

3.7.3 Sequence of Digital Compass Heading Degree to Manipulate Servo Motor Turning Degree

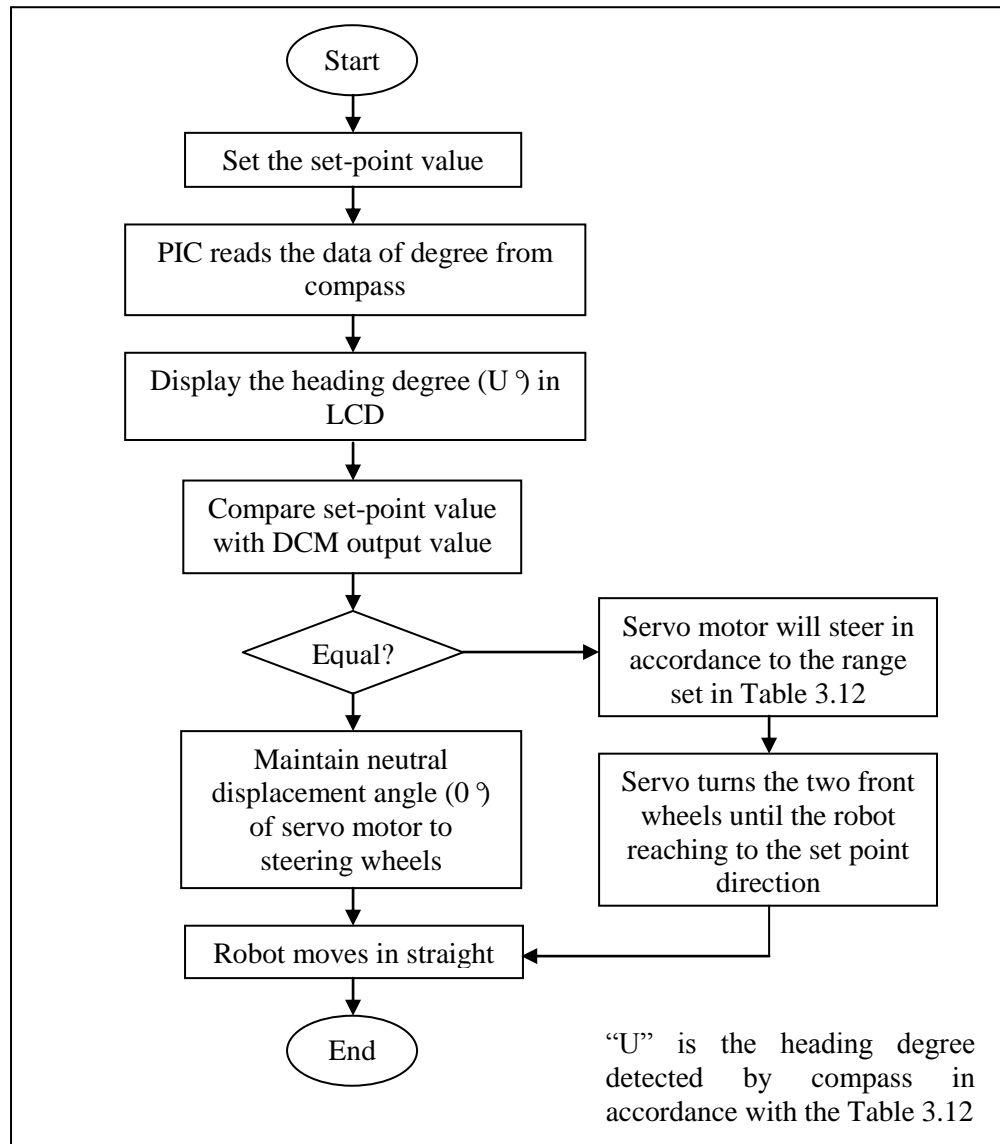


Figure 3.35: Sequence of Digital Compass Module Heading Degree Manipulation to Servo Motor Turning Control

3.7.4 Programming of Compass Heading Degree Manipulation to Servo Motor Turning Degree

* Refer to Appendix A (5)

3.8 Obstacles and Wall Detection

MaxSonar EZ1 is a ultrasonic sensor which offers very short to long-range detection and ranging, in an incredibly small package with ultra low power consumption. The MaxSonar EZ1 detects objects from 0-inches to 254-inches and provides sonar range information from 6-inches out to 254-inches with 1-inch solution. Objects from 0-inches to 6-inches range as 6-inches. The interface output formats included are pulse width output, analog voltage output, and serial digital output [14].

In this FWMR project, ultrasonic sensor is required due to avoid the robot crashing the front obstacles especially walls and poles. This is an extra component to affix in front of the robot due to the safety purpose to protect the system and circuitry of all the component elements from the risk of damage and malfunction after the robot crashing to a wall or a large obstacle. The ultrasonic transducer would produce the ultrasonic signals from time-to-time when an appropriate power supply or voltage provides to EZ1. Program a suitable coding to fix certain distance detection is vital in order that these signals are propagated through a sensing medium and EZ1 can be used to detect returning signals which bouncing back from obstacles. The condition will be set in coding when the distance detected is less than or equal to the fixed set-distance, therefore the PIC microcontroller may command to the DC motor to be halted state and stopped for further moving to avoid from crashing to the obstacles.

A desired condition has been set in coding due to verify and figure out the distance between the ultrasonic sensor and an obstacle. If the distance of the ultrasonic sensor to an obstacle is less than or equal to the fixed distance value, then the PIC 2 will react and command the DC motor to be stopped or halted.

3.8.1 Hardware Review of Ultrasonic Sensor

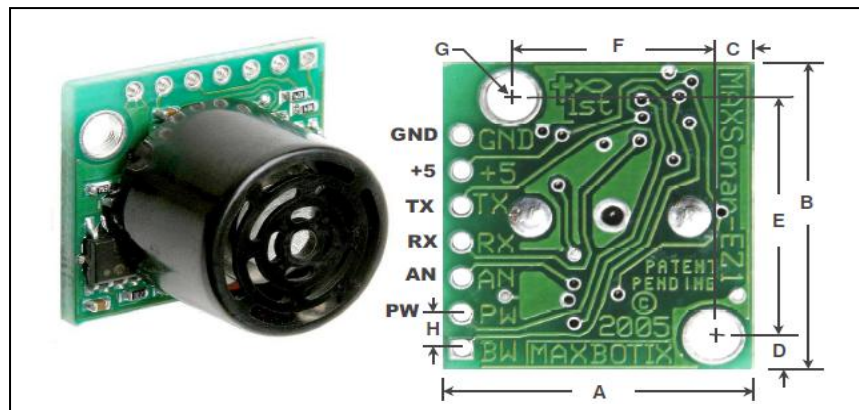
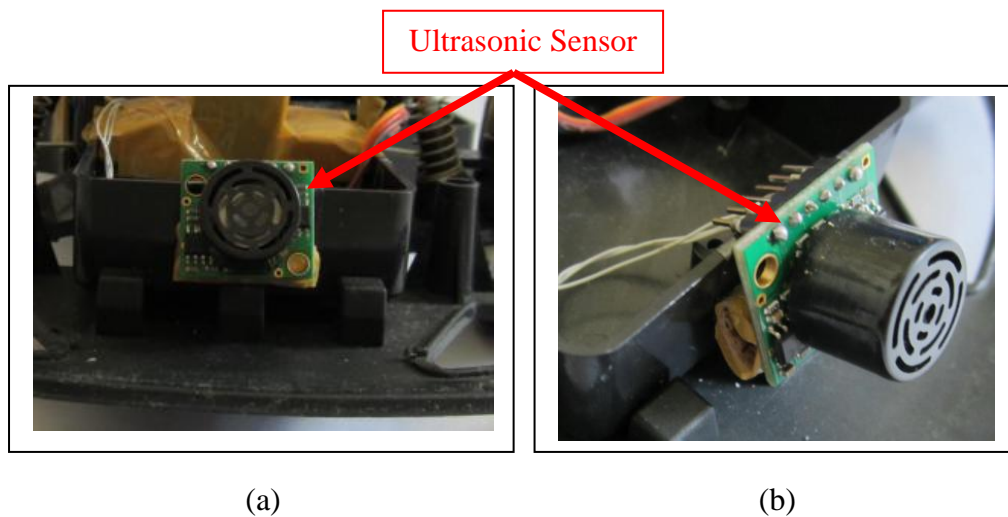


Figure 3.36: Side and Bottom View of MaxSonar EZ1



(a)

(b)

Figure 3.37: Installation of Ultrasonic Sensor to FWMR

(a) Front View

(b) Side View

Table 3.13: MaxSonar EZ1 Pin Out

Pin	Connection and Description
GND	Return for the DC power supply. Must be ripple and noise free for best operation
+5V	Requires 5Vdc +/- 0.5Vdc. Current capability of 3mA capacity recommended
TX	Delivers asynchronous serial with an RS232 format, except voltage are 0-5V
RX	This pin is internally pulled high. The EZ1 will continually measure range and output if RX data is left unconnected or held high
AN	Outputs 0 to 2.55 volts with a scaling factor of 10mV per inch.
PW	This pin outputs a pulse width representation of range
BW	Reserved

3.8.2 Connection Circuit of Ultrasonic Sensor to PIC 2

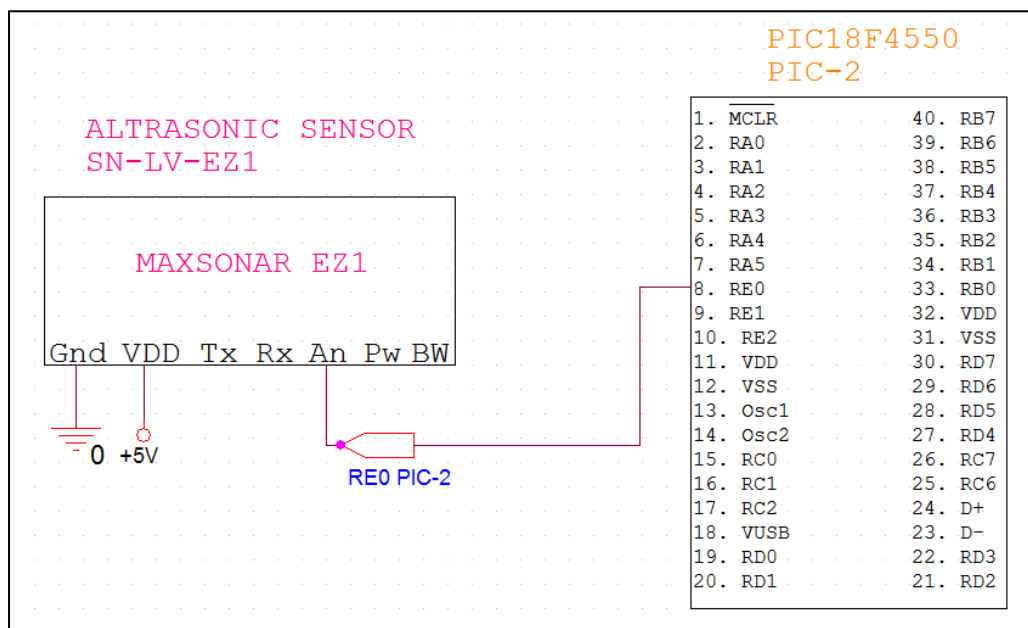


Table 3.38: Connection Circuit of Ultrasonic Sensor to PIC 2

3.8.3 Sequence of Ultrasonic Sensor Function

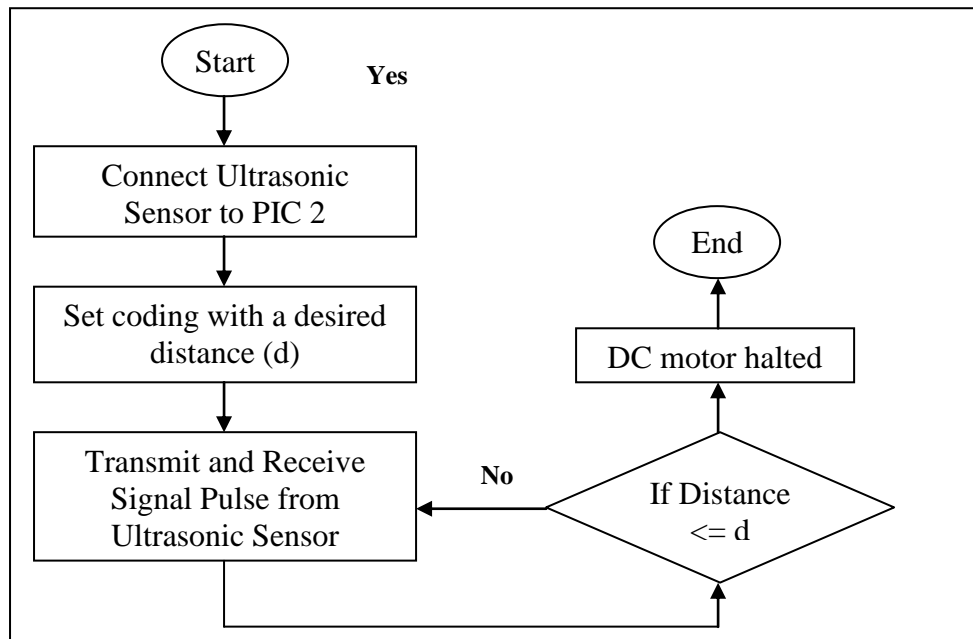


Figure 3.39: Sequence of Ultrasonic Sensor Distance Detection

3.8.4 Programming of Ultrasonic Sensor Operation

* Refer to Appendix A (6)

CHAPTER 4

RESULT AND DISCUSSION

4.1 Introduction

This chapter plays the main role to observe and analyze all relevant testing has been done for the development of straight line movement for four-wheeled mobile robot. In experiments and testing, the data and responses from every essential component tools will be collected and recorded for further analysis and interpretation. In this project, result will be impressed and emphasized in certain main testing components especially DC motor, servo motor, ultrasonic sensor, digital compass module and the addition of PID control algorithm.

The analysis and discussion of each part of result will be specified and elaborated in detail. All interpretations in accordance to the data collected and transforms to relevant graphs or charts due to assess the effectiveness, stability, performance, efficiency and straight line precision of the robot. The comparison between the actual performances manipulated by compass's heading degree and the PID control output will be fully observed and explored to achieve the straight line platform with minimal sinusoidal waveform existed due to obtain the best result and performance.

4.1.1 Four-Wheeled Mobile Robot (FWMR)

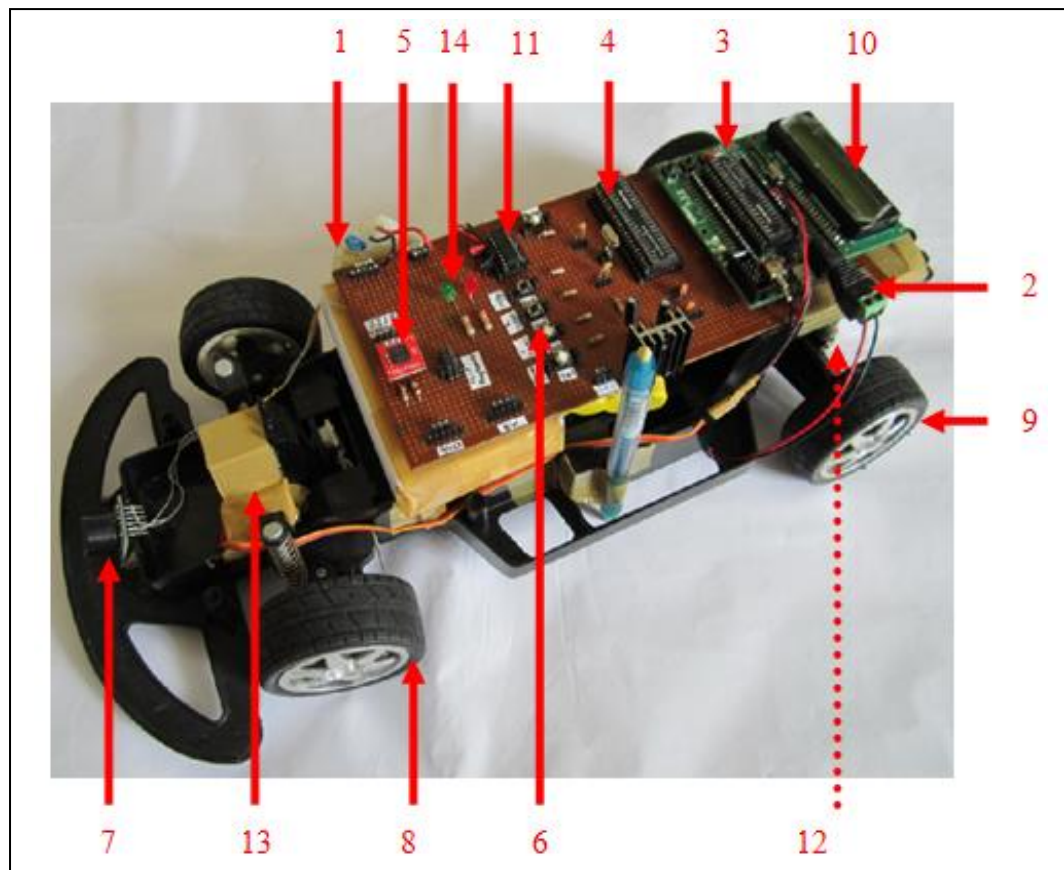


Figure 4.1: Four-Wheeled Mobile Robot (FWMR)

Table 4.1: Respective Parts of Four-Wheeled Mobile Robot

Label	Component	Label	Component
1	Power Supply 1 with 9V	8	Driving (Front) Wheels
2	Power Supply 2 with 9V	9	Steering (Back) Wheels
3	PIC 1	10	LCD
4	PIC 2	11	IC L293D
5	Digital Compass Module	12	DC Motor
6	Four Switches	13	Servo Motor
7	Ultrasonic Sensor	14	LEDs (Green and Red)

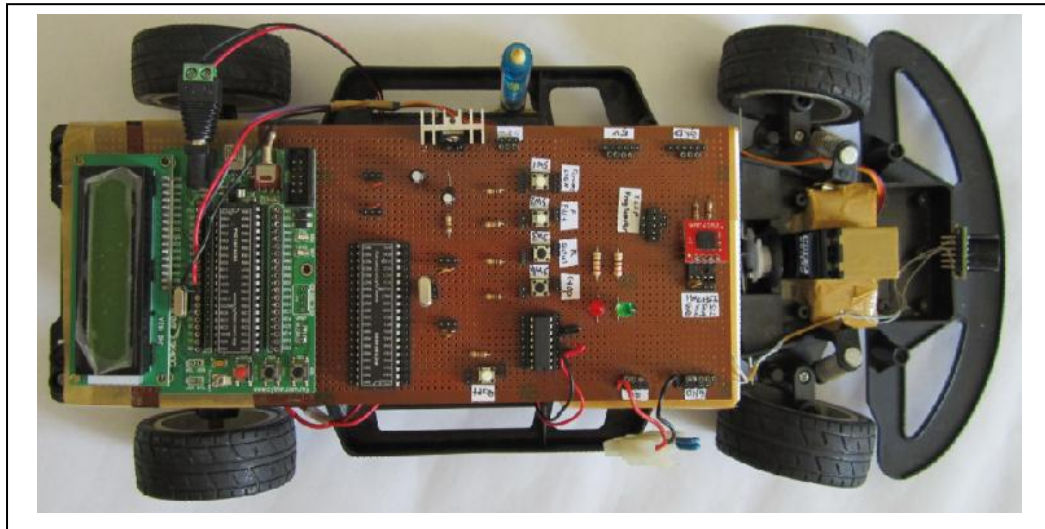


Figure 4.2: Top View of FWMR

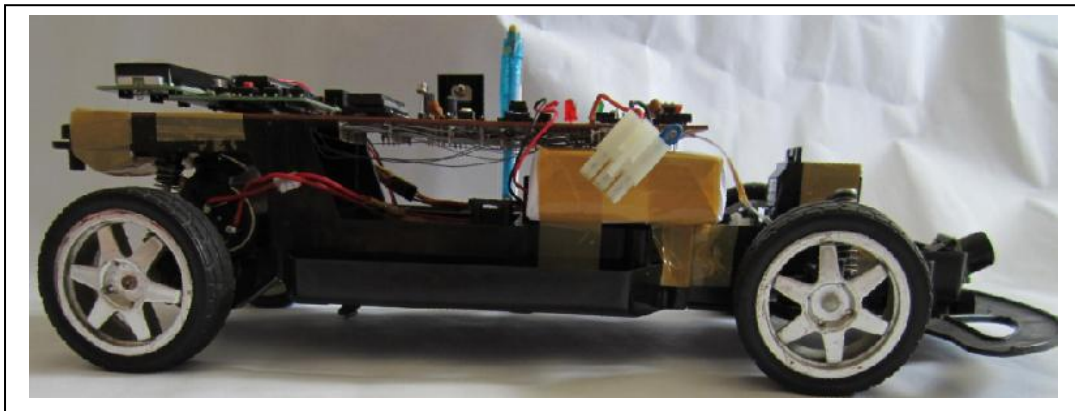


Figure 4.3: Side View of FWMR

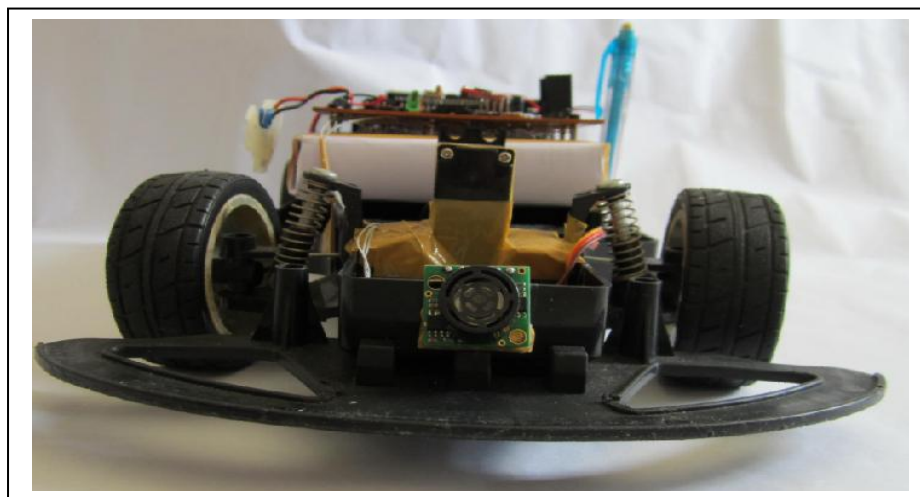


Figure 4.4: Front View of FWMR

4.1.2 Full Connection Circuits of Four-Wheeled Mobile Robot

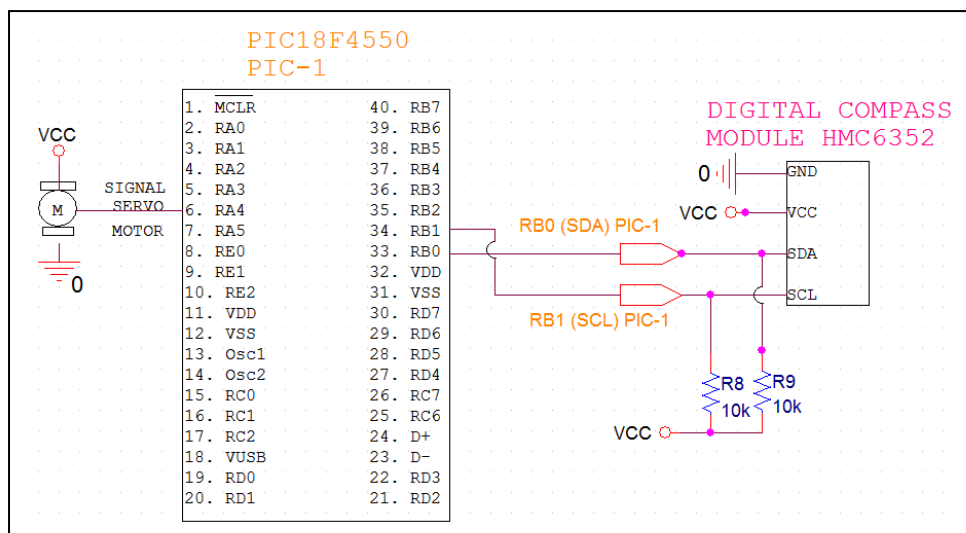


Figure 4.5: Full Connection Circuit of PIC 1

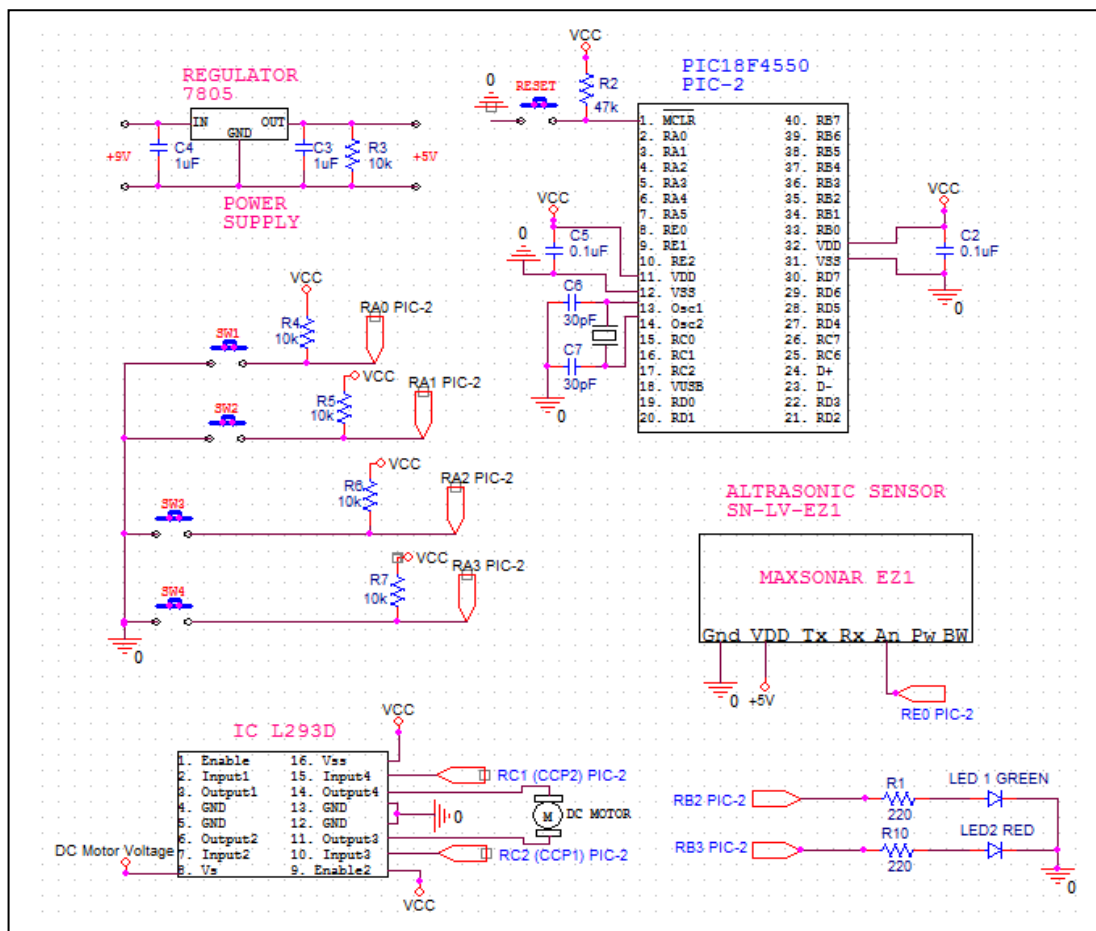


Figure 4.6: Full Connection Circuit of PIC 2

4.1.3 Full Sequence of FWMR Operation

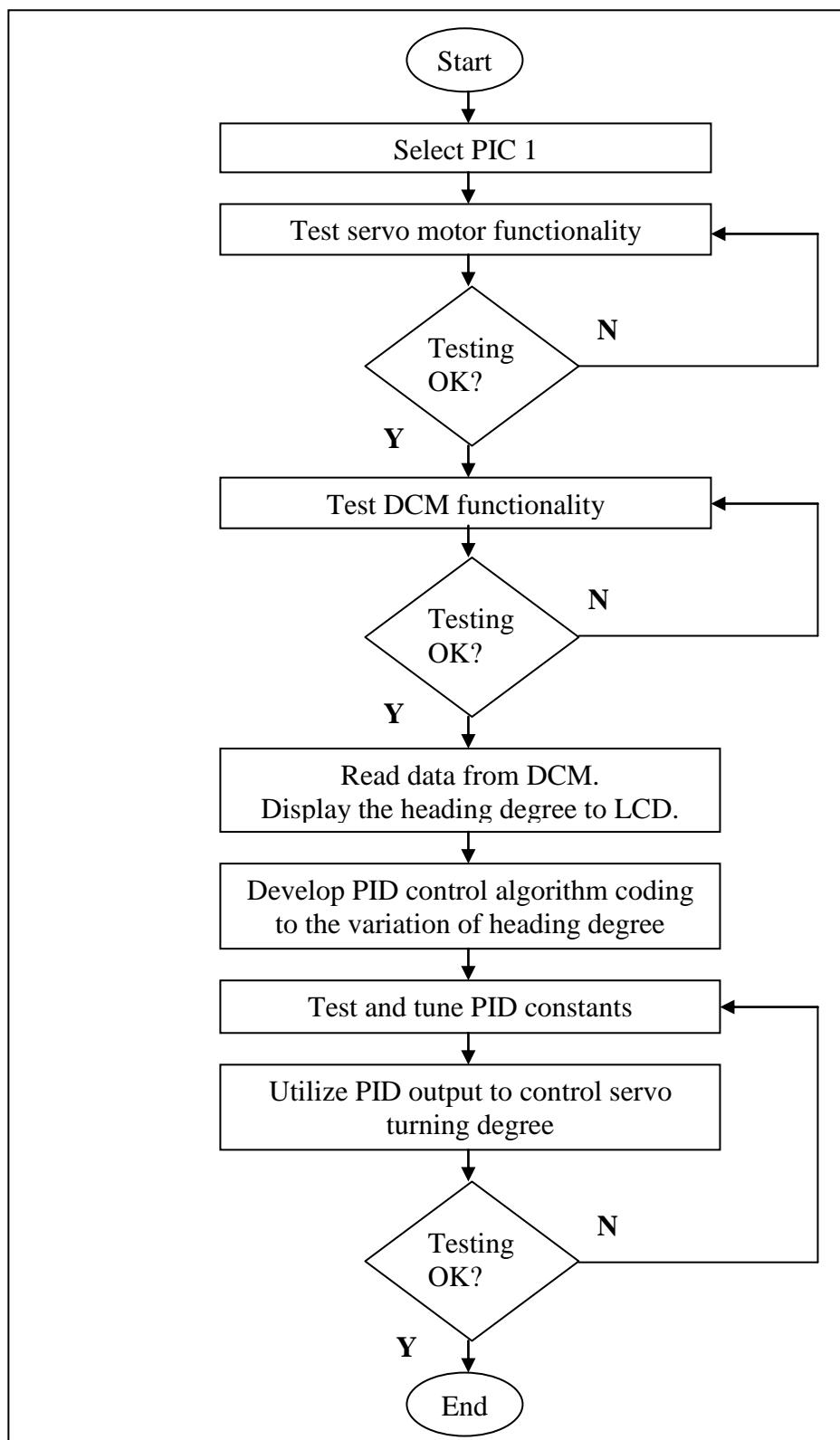


Figure 4.7: Full Sequence Flow of PIC 1

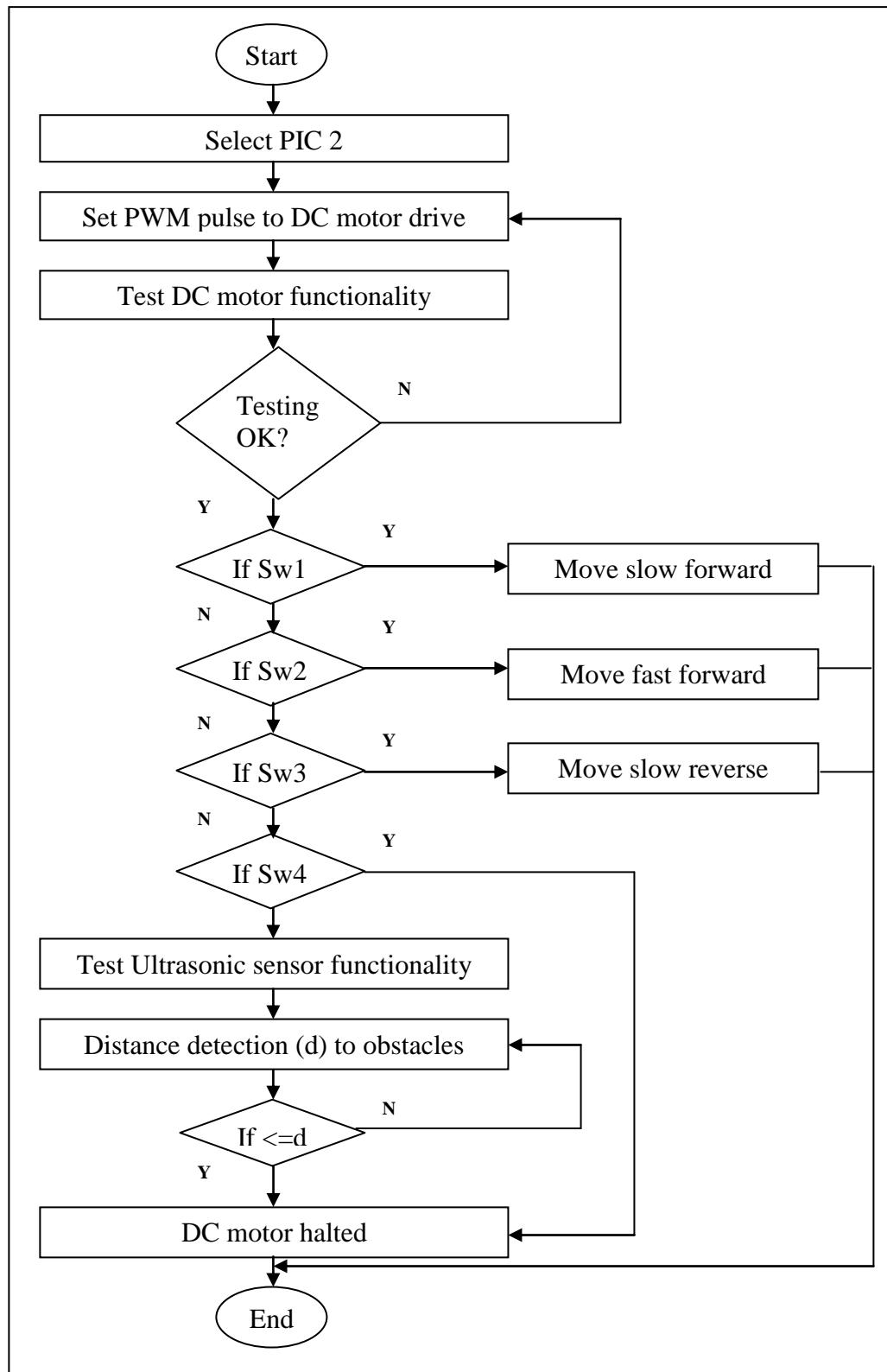


Figure 4.8: Full Sequence Flow of PIC 2

4.1.4 Full Programming of FWMR Operation

* Refer to Appendix A (7) and (8)

4.2 PID Control Algorithm

PID control algorithm is implemented due to the instability of compass heading degrees to control the servo motor turning degrees which cause large oscillation and veering to other side during the FWMR moving straight in accordance to the fixed set-point direction. The prime advantage of adding up the PID algorithm is to manipulate the process inputs based on the previous errors changes which is suited to apply in this robot straight line movement process in order to give more stable and accurate control to the servo motor turning.

The three basic coefficient terms play the prime role with each term would contribute its own task coherently to stabilize the robot from large oscillation and maintain a smooth straight line motion. As mentioned in section of Methodology, “Proportional” is the first coefficient term used to determine the reaction to the current error, “Integral” to be the second term to sum up the current and previous errors for the next reaction, and the last term of “Derivative” is used for the reaction to the rate of the error changed. Each term would have its own constant as K_P , K_I and K_D , to scale, adjust and tune the terms in order to reach to an optimal PID output result to enable FWMR moving a smooth straight line platform according with the fixed set-point direction by minimizing the oscillation.

The differentiation between the robot system with and without the PID algorithm will be further discussed in following sub-portions base on the graphs

indications. In additional, a suitable tuning and adjustment to these three PID constants are essential task due to obtain a better PID output result. A reasonable and proper comparison will be accomplished according to the graphs drawn from the data recorded through practical testing. Full interpretation will be stated in each graph in the following sub-sections.

PID output result will be used to manipulate the servo motor turning degree instead of heading degree from compass. A full and distinct PID output range has been expressed in Table 4.2 and each PID output range to respond the PIC microcontroller due to react the respective servo motor turning degree of FWMR.

Table 4.2: Manipulation of PID Output Range to Servo Motor
Turning Degree

PID Output	Servo Motor Turning Degree
$P = 0$	0 °, in neutral position
$1^{\circ} \leq P \leq 5^{\circ}$	+10 °, turn right
$6^{\circ} \leq P \leq 19^{\circ}$	+20 °, turn right
$20^{\circ} \leq P \leq 34^{\circ}$	+30 °, turn right
$35^{\circ} \leq P \leq 179^{\circ}$	+40 °, turn right
$180^{\circ} \leq P \leq 324^{\circ}$	-40 °, turn left
$325^{\circ} \leq P \leq 339^{\circ}$	-30 °, turn left
$340^{\circ} \leq P \leq 354^{\circ}$	-20 °, turn left
$355^{\circ} \leq P \leq 370^{\circ}$	-10 °, turn left

4.2.1 Theoretical Framework

Before going to the result sections, a theoretical framework is a collection interrelated concepts which guides to determine what things will be measured and statistical relationships to be looked for. Theoretical frameworks are obviously critical in deductive. According to this research, the theoretical framework must be very specific and well-thought out before the tasks of carrying out the robot straight line movement analysis to get the corresponding result.

The equation of PID algorithm is essential to be understood for inserting this equation to FWMR's program coding to manipulate the robot in order to complete the process smoother, faster and efficient. Equation 1 expresses that the PID output will be obtained in accordance with time domain function. On the other hand, Equation 2 shows that the Equation 1 has been transformed using Laplace Transform to become a frequency domain function equation prime to apply in programming for manipulation purpose. Equation 3 indicates that the three main coefficients with K_P , K_I and K_D can be tuned and adjusted to achieve optimal robot movement response.

$\text{PID Output} = u(t) = P e(t) + I \int e(t) dt + D \frac{d}{dt} e(t)$	
Where	
P	= Coefficient of proportion in time domain
I	= Coefficient of integral in time domain
D	= Coefficient of derivative in time domain
e(t)	= Tracking error in time domain

Figure 4.9: Equation 1 of PID Output in Time Domain

$$\text{PID Output} = u(t) = Pe(s) + I \frac{1}{s} e(s) + Dse(s)$$

Where
 s = Laplace element of s domain

Figure 4.10: Equation 2 of PID Output after Laplace Transform

$$\text{PID Output} = u(t) = K_p \text{Proportion} + K_i \text{Integral} + K_d \text{Derivative}$$

Where
 Proportion = current error
 Integral = cumulative of current and previous errors
 Derivative = current error – previous error

Figure 4.11: Equation 3 of PID Output Application in Programming

Figure 4.12 indicates that the desired result base on the theoretical application of PID algorithm into the FWMR's operational system. The robot can be gradually stabilized to go straight and the sinusoidal waveform can be reduced to enable the robot moving straight efficiently.

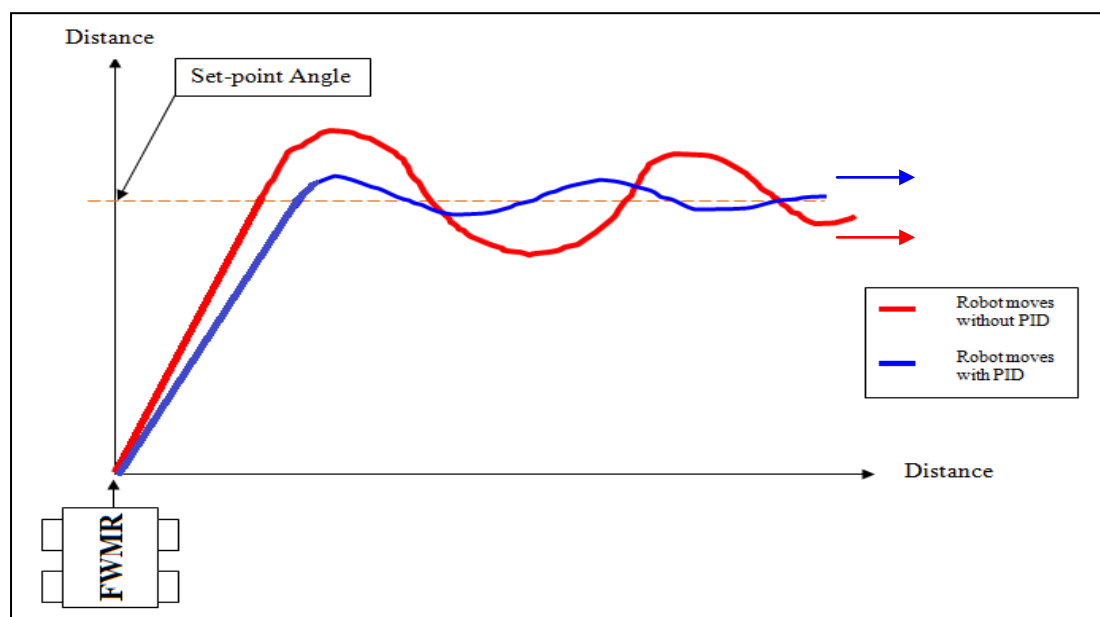


Figure 4.12: Robot Straight Line Movement in Theoretical Application

4.2.2 Practical Results

Figure 4.13 and Figure 4.14 indicate the FWMR straight line movement in accordance with the robot heading angle of 275° and 350° . The graphs obtained from these two figures can be verify and figure out the difference by applying PID and without PID output range to manipulate the servo turning degree to achieve the straight line platform performance to the set-point angle with $359^\circ \approx 0^\circ$ (or to North).

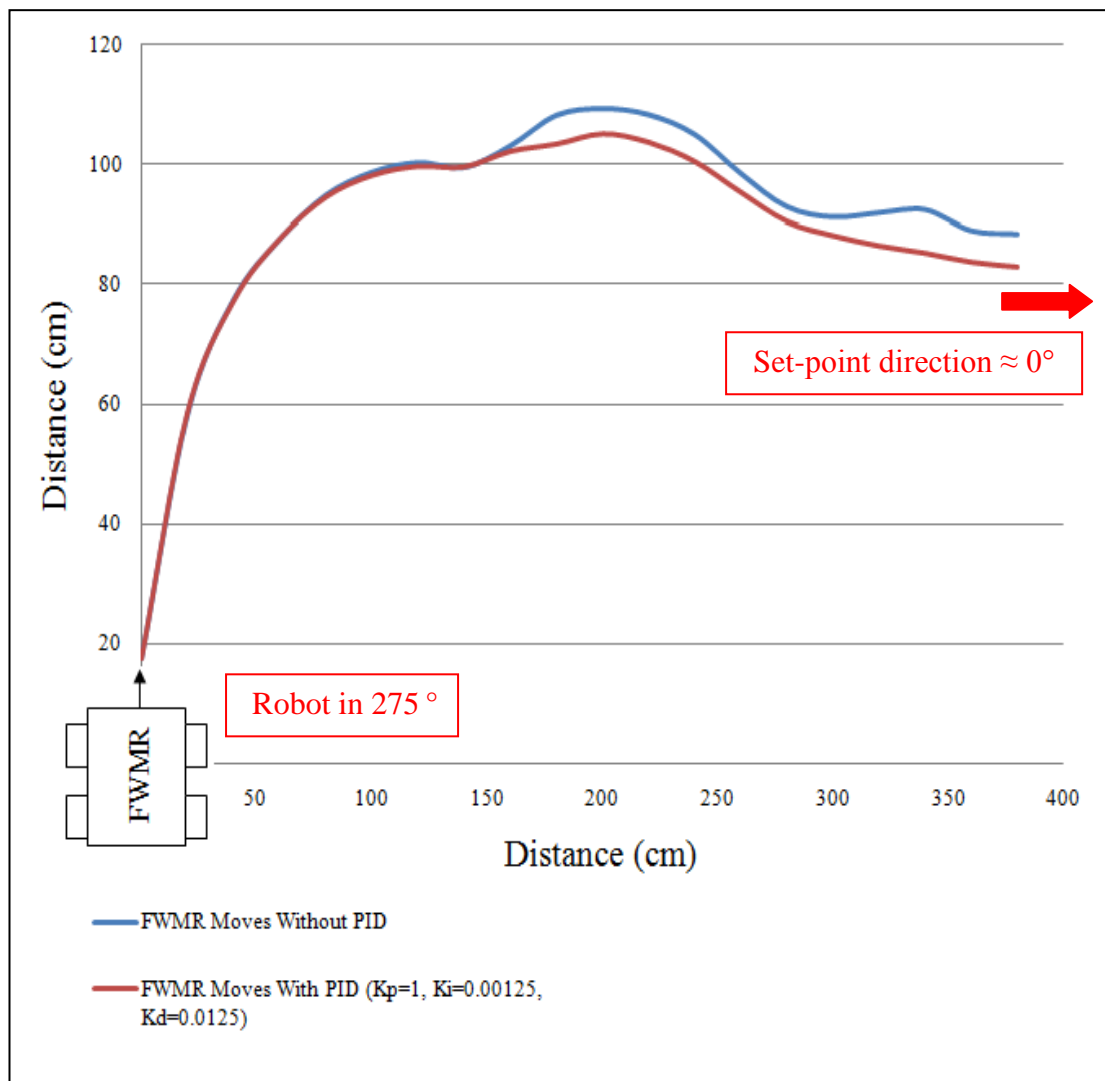


Figure 4.13: FWMR Straight Line Movement Performance in Heading Degree of 275° With and Without PID Control Algorithm

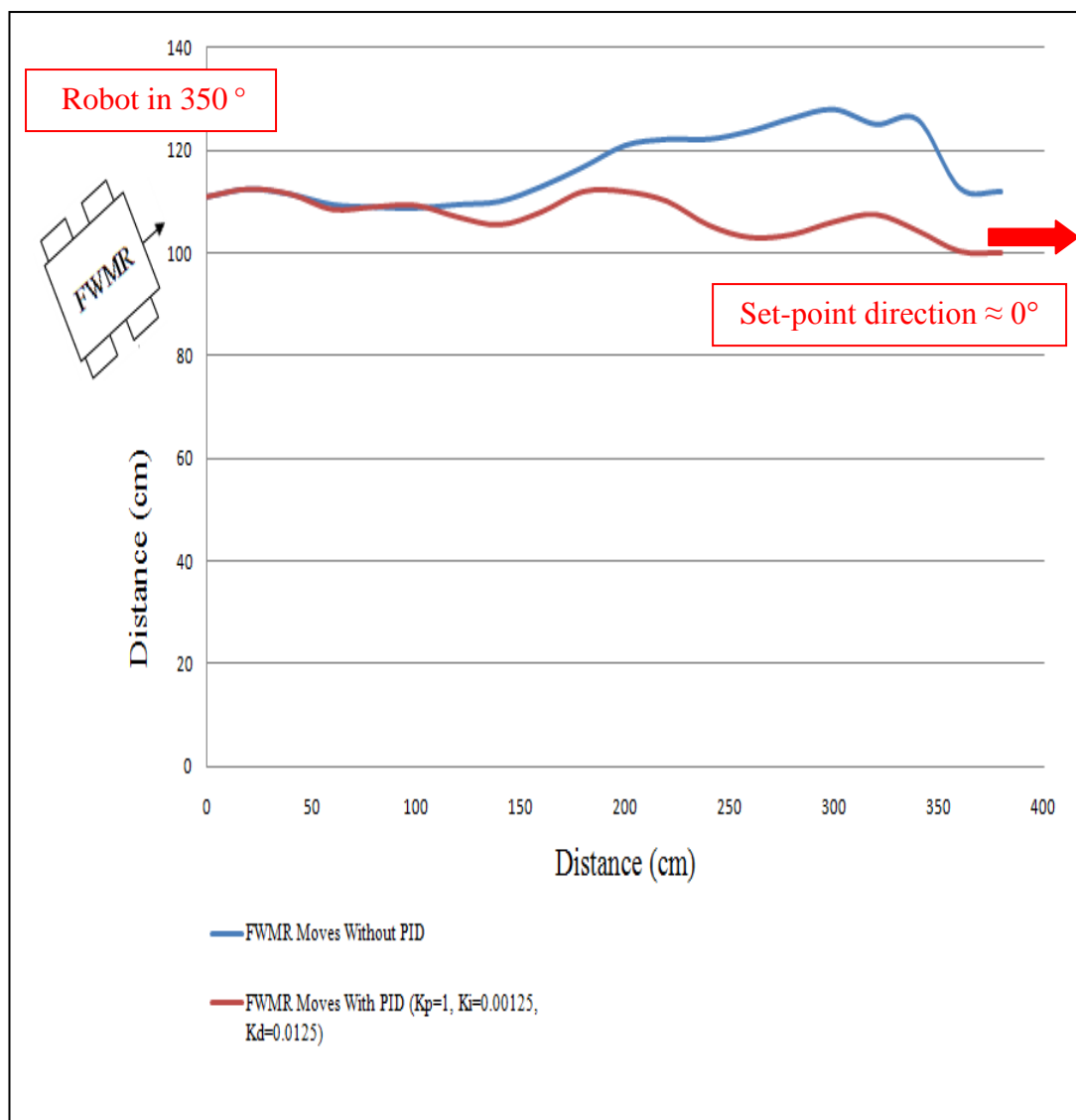


Figure 4.14: FWMR Straight Line Movement Performance in Heading Degree of 350° With and Without PID Control Algorithm

Three coefficient terms of PID can be adjusted and tuned with its respective constant in order to reach to the smoother straight line movement by FWMR. In this case, K_P , is set as “1” for the direct reaction to the current error. In the process of tending straight to the set-point angle, robot may overshoot the set-point direction and move to the other side. Therefore, K_I and K_D have to be adjusted to reduce the oscillation during its motion tending to desired set-point direction. Table 4.3 expresses the adjustment of K_P , K_I and K_D with different values.

Table 4.3: PID Constants Adjustment of K_P , K_I and K_D

Testing	K_P	K_I	K_D
a.	1	0.0005	0.005
b.	1	0.00125	0.0125
c.	1	0.002	0.02

The PID output result is not able to manipulate the servo turning degree if the PID constants values are set to be too large since this output value would exceed the condition ranges and robot is unable to identify the correct direction to go forward. In addition, the PID output control would be similar to the compass heading degree control to servo motor turning if the PID constants values have been set to be very small. According to the Table 4.3, K_P will be set in 1 where K_I and K_D are adjustable and tunable in order to obtain better result which may achieve the closest point for the robot to overshoot to the set-point angle to implement its better performance to straight line movement. Figure 4.15 and Figure 4.16 express that three PID algorithm constants are adjusted in accordance to the Table 4.3 with the robot heading angle of 275° and 350° .

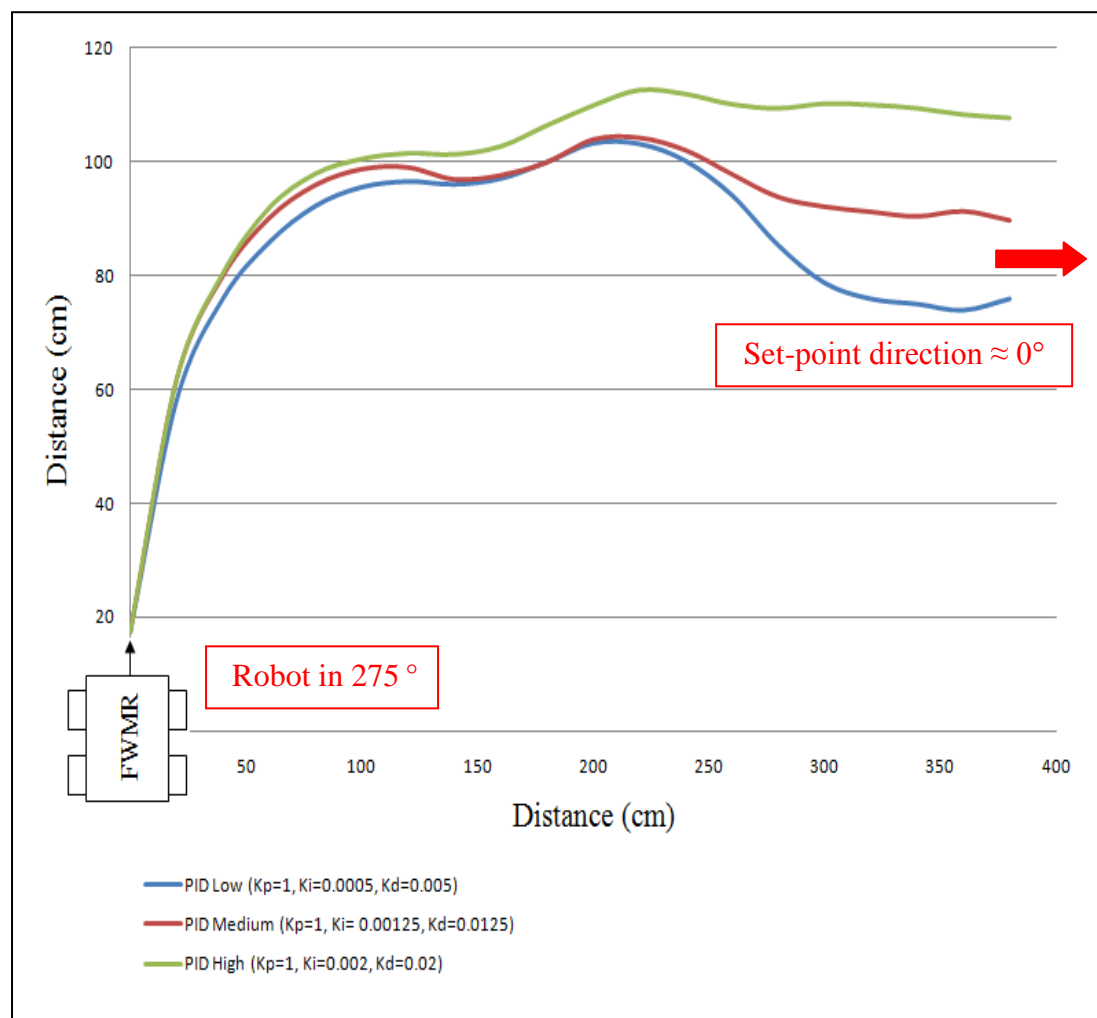


Figure 4.15: Three PID Constants Adjustment in Heading Degree of 275°

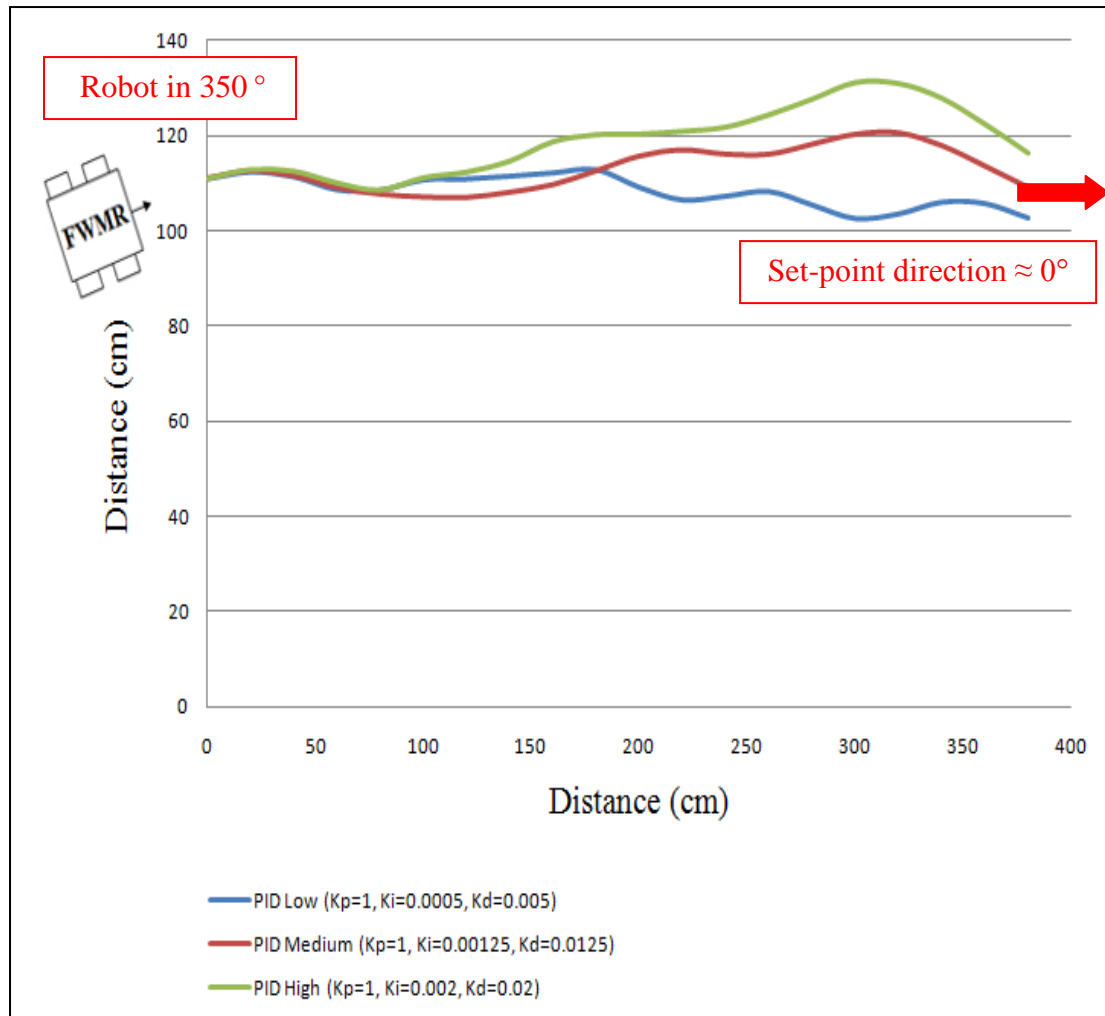


Figure 4.16: Three PID Constants Adjustment in Heading Degree of 350°

4.3 Discussion

According to the graphs obtained in Figure 4.11 and Figure 4.12, FWMR can move smoother for achieving the straight line movement platform after the utilization of PID algorithm into this robot system. Obviously, movement of FWMR will be very unstable and a large oscillation will be happened if the coding system without applying PID algorithm control.

Three testing have been done by varying the values of PID constants values and the robot performances have been shown in Figure 4.13 and Figure 4.14. The graphs imply that the smaller or higher values of K_I and K_D will affect the robot tends to the set-point direction with better solution or vice versa. I would select the “PID Medium” ($K_P = 1$, $K_I = 0.00125$, $K_D = 0.0125$) to set into the FWMR coding system in order to reduce the oscillation and to be more accurate for the robot tending to North or $359^\circ \approx 0^\circ$ direction.

Nonetheless, the FWMR may not be able to achieve the smoother straight line movement in despite of the PID algorithm has been inserted to the coding system. This circumstance is because of the environment effects and also with the components limitations. A very large testing space must be used in order for the robot to achieve better straight line movement and the data collected would be in high accuracy. High sensitivity of Digital Compass Module is one of the main reasons to depress the FWMR movement performance.

Some of the issues may affect the robot straight line movement performance which indicated in detail as following:

- A small area or space is not able to observe the movement performance and obtain the accurate data
- External force especially with the friction and external interrupts to the robot
- Rotational speed of DC motor
- Ground roughness
- Sufficiency of power supply
- Compass Module position detection to be failed because of the electromagnetic effect from other components or electronic devices, internet wireless signal and surrounding full of plants or forest

CHAPTER 5

CONCLUSION AND RECOMMENDATIONS

5.1 Introduction

Conclusion and recommendations are essential section to tie or integrate the various issues and researches that have completed from previous chapters to make comments upon the meaning for all of the portions due to fulfill the requirements of FWMR straight line movement performance. All testing and analysis for each stage may guide to ensure and ascertain the key of success to develop the robot to function and operate correctly. There are some points will be introduced and elaborated in this chapter as well as recommendations for improvements, research with additional elements and future investigation will be further discussed and explained. The platform will be provided to minimize the problems encountered to the next researchers continuing for the future works and investigations.

5.2 Conclusion

In conclusion, this project can be managed to develop the Four-Wheeled Mobile Robot to move in straight line platform in combination the software and hardware parts to ensure that it could fulfill and meet the requirements and objectives. The electrical and mechanical hardware can be successfully installed and set to build up a Four-Wheeled Mobile Robot to achieve the satisfaction stage. Apart from this, the software part consists of C-Language coding is vital to thoroughly manipulate the robot functionality and operation to meet the straight line movement performance. PID control algorithm has been appended to the coding instead of utilizing heading degree of compass to control the servo turning to minimize the oscillation of the robot to tend to other side during its straight line performance to the fixed set-point angle. Nonetheless, this robot straight line movement is not able to perform with higher accuracy during its repeating movement operation by using PID algorithm and the robot is easily affected by the issues as well as the external force, rotational speed of DC motor, ground roughness, sufficiency of power supply and sensitivity of compass module. As a result, some appropriate recommendations will be provided to enhance this robot movement system and implement the future investigations explicitly.

5.3 Recommendations and Improvements

This Four-Wheeled Mobile Robot straight line movement development is recommended to be undertaken again with better experience and expert in manipulation of PID control algorithm set into the coding system. The PID auto-tuning technique can be investigated and utilized instead of manually trial-and-error method to set those three PID coefficient constants in the coding.

PID algorithm is essential to build up and develop the robot to be capable to move in straight line platform. As a result, auto-tuning technique is rightly to tune the constants of K_P , K_I and K_D until the robot reaches the most corresponding values to minimize the oscillation during its straight line movement to the desired set-point direction.

To observe and investigate the robot movement, the investigator should place the robot which is free of any wireless signal transmission and the surrounding with fewer plants due to avoid the affects of the compass module position detection. In addition, the space must be sufficiently large to enable the robot to move longer and the robot movement performance can be observed explicitly due to improve the data accuracy and precision.

The designation and construction of mechanical hardware of this Four-Wheeled Mobile Robot ought to make in smaller size and weight. The robot with smaller size and weight can be easier for the DC gear motor to rotate the wheels in forward or backward manner. Moreover, this improvement can reduce the robot sliding or veering to other side during its straight line movement to a fixed set-point direction and directly to enhance the capability of the robot to meet the point of straight line movement.

5.4 Additional Elements and Future Tasks

High inaccuracy and instability will be encountered if the robot is dependent solely on Digital Compass Module to manipulate the servo motor turning degree. Therefore, several elements or features can be further added that may improve the effectiveness, efficiency, accuracy and operability of the Four-Wheeled Mobile Robot and better solution or method to achieve the robot's straight line movement purpose.

A GPS sensor can be applied and implemented to the FWMR due to enable the robot to search, identify and verify the desired set-point direction to be reached. Furthermore, this type of sensor for use in machine or robot controls to provide pace and distance measurement. Hence, this may enhance the robot system to verify the robot current position correctly instead of just using the compass module to verify the robot direction and even used to manipulate the servo motor turning degree as to reach the desired result of development of straight line movement.

To identify the front objects or obstacles, a vision sensor can be used in the FWMR to allow the robot autonomously navigating and figuring out the objects or obstacles. The capability of this vision sensor can perform to verify the shape and size of the obstacles then used to respond to the microcontroller to implement further action whether to avoid the small obstacles or halt the robot as to protect from crashing to the large objects or walls.

REFERENCES

- [1] Gyula Mester (2006). "Motion Control Of Wheeled Mobile Robots," SISY 2006 Serbian-Hungarian Joint Symposium On Intelligent Systems. Department Of Informatics, Polytechnical Engineering College, Marka Oreskovica 16, 24000 Subotica, Serbia.

- [2] Sven Bottcher (2006). "Principles Of Robot Locomotion," Seminar 'Human Robot Interaction'.

- [3] Tan Piow Yon (2011). "Two Wheels Balancing Robot with Line Following Capability," World Academy Of Science, Engineering and Technology 79 2011, Bachelor Degree of Electrics (Electronics) in University Malaysia Pahang, Faculty of Electrical and Electronics.

- [4] Kritof Goris (2004-2005). "Autonomous Mobile Robot Mechanical Design," Vrije Universiteit Brussel, Faculteit Ingenieurswetenschappen Vakgroep Werktuigkunde.

- [5] Ibrahim Kamal (2008). "Small Robot Drive Trains," Website Of IKA LOGIC Electronics Solutions.
Citing Internet Sources URL: ikalogic.com/tut_mech_1.php

- [6] K-Junior (2010). "Moving Straight and Detecting Obstacles," K-Team S.A Rue Galilee 9, Switzerland, in Collaboration with Department of Electrical Engineering Politechnica University of Bucharest, Romania.

- [7] Zhao Zhang (2009). "Automatic North-Facing Robot with Compass Module And Closed-Loop Control," Missouri Western State University 4525 Downs Drive, Saint Joseph, MO 64507.

- [8] Wikipedia. "Ultrasonic Sensor."
Citing Internet Sources URL: http://en.wikipedia.org/wiki/Ultrasonic_sensor

- [9] Khairul A'Alam Bin Abdul Ghani (2007). "Obstacle Avoidance Mobile Robot," University Malaysia Pahang

- [10] Wikipedia. "PID Controller."
Citing Internet Sources URL: http://en.wikipedia.org/wiki/PID_controller

- [11] User's Manual of SK40C Development Board Start Up Kit. Available at:
http://www.cytron.com.my/usr_attachment/SK40C_Users_Manual.pdf

- [12] Datasheet of Microchip PIC 18F Series. Available at:
http://www.cytron.com.my/datasheet/IC/MCU/18f2455_2550_4455_4550.pdf

- [13] User's Manual of UIC00B ICSP PIC Programmer. Available at:
http://www.cytron.com.my/usr_attachment/UIC00B_&_UICS_Users_Man

- [14] Datasheet of MaxSonar EZ1. Available at:
<http://www.cytron.com.my/datasheet/sensor/LVEZ1.pdf>

- [15] C Compiler Reference Manual July 2011. Available at:
www.ccsinfo.com/downloads/ccs_c_manual.pdf

- [16] Pratheek (2009). "An Advanced Line Following Robot with PID Control,"
Society of Robots with Robot Tutorials. Available at:
www.societyofrobots.com/member_tutorials/book/export/html/350

- [17] Jaseung Ku (2005). "A Line-Follower Robot," Available at:
online.physics.uiuc.edu/courses/.../Robot_project_jaseung_.pdf

APPENDIX A
Program Source Code

(1) Source Code of DC Gear Motor with IC L293D

```

//*****
*Title: Programming of Four-Wheeled Mobile Robot PIC Brain Board *
*Program Author: Ching Wai Hoong *
*Software: PIC C Compiler (CCS) *
*PIC Type: PIC Microchip 18F4550 *
*PIC-2 *
//*****

#include <18F4550.h>
#fuses NOLVP, NOWDT, HS, NOPBADEN
#use delay(clock=20M, crystal)

#define L293D_2 PIN_B3
#define L293D_7 PIN_B4
#define L293D_10 PIN_B1
#define L293D_15 PIN_B2

void main ()
{
    setup_adc_ports(NO_ANALOGS);
    set_tris_b(0x00);
    output_b(0x00);
    delay_ms(50);

    while(true)
    {
        //Motors Run Forward
        output_high(L293D_2); //L293D input1 set as 1//
        output_low(L293D_7); //L293D input2 set as 0//
        output_high(L293D_10);
        output_low(L293D_15);
        delay_ms(5000);

        //Motors Stopped
        output_low(L293D_7); //L293D input2 set as 0//
        output_low(L293D_2);
        output_low(L293D_15);
        output_low(L293D_10);
        delay_ms(5000);

        //Motors Run Backward
        output_high(L293D_7); //L293D input2 set as 1//
        output_low(L293D_2);
        output_high(L293D_15);
        output_low(L293D_10); //L293D input1 set as 0//
        delay_ms(5000);
    }
}

```



```

//Motors Stopped
output_low(L293D_7); //L293D input2 set as 0//
output_low(L293D_2);
output_low(L293D_15);
output_low(L293D_10);
delay_ms(5000);
}
}

```

(2) Source Code of Servo Motor Steering Operation

```

//*****
*Title: Programming of Four-Wheeled Mobile Robot PIC Brain Board *
*Program Author: Ching Wai Hoong *
*Software: PIC C Compiler (CCS) *
*PIC Type: PIC Microchip 18F4550 *
*PIC-1 *
//*****

#include <18f4550.h>
#fuses NOLVP, NOWDT, HS, NOPBADEN, NOPROTECT
#use delay(clock=20M, crystal)

#define L293D_1 PIN_B2
#define SW1 PIN_B0
#define SW2 PIN_B1

void main ()
{
    set_tris_b(0x03);
    output_b(0x00);
    delay_ms(50);

    while(true)
    {
        if(!input(SW1)==1)
        {
            //Motors Run Forward 40 degree
            output_high(L293D_1);
            delay_ms(2);
            output_low(L293D_1);
            delay_ms(18);
        }
        if(!input(SW2)==1)
        {

```

```

//Motors Run Reverse 40 degree
output_high(L293D_1);
delay_ms(1);
output_low(L293D_1);
delay_ms(19);
}
}
}

```

(3) Source Code of Digital Compass Module Operation

```

//*****
*Title: Programming of Four-Wheeled Mobile Robot PIC Brain Board *
*Program Author: Ching Wai Hoong *
*Software: PIC C Compiler (CCS) *
*PIC Type: PIC Microchip 18F4550 *
*PIC-1 *
//*****

#include <18f4550.h>
#fuses HS, NOWDT, NOPROTECT, BROWNOUT, PUT, NOLVP
#use delay(clock=20M, crystal)
#include <flex_lcd.c>
#use i2c(master, sda=PIN_B0, scl=PIN_B1, FORCE_HW)

#define HMC6352_I2C_WRITE_ADDRESS 0x42
#define HMC6352_I2C_READ_ADDRESS 0x43
#define HMC6352_GET_DATA_COMMAND 0X41 //Transmit "A" to compass
for heading
// Read the compass heading. This is in unit of 1/10 of a degree
// from 0 to 3599

int16 HMC6352_read_heading(void)
{
    int8 lsb;
    int8 msb;

    i2c_start();
    i2c_write(HMC6352_I2C_WRITE_ADDRESS); //0x42 is the address that we
need to communicate with to write information to the device
    i2c_write(HMC6352_GET_DATA_COMMAND); //"A" or 0x41 is sent to
magnetometer to get a response that pertains to what direction it is facing
    i2c_stop();
    delay_ms(10);

    i2c_start();
    i2c_write(HMC6352_I2C_READ_ADDRESS); //0x43 is the address that to
communicate with to read information from compass

```

```

    msb=i2c_read(); //Response 1 for MSb data after reading the data from compass
    lsb=i2c_read(0); //Response 2 for LSb data after reading the data from compass
    i2c_stop();
    return((int16)lsb | (int16)msb << 8);
}
void main()
{
    Int16 heading;
    while(true)
    {
        heading=(HMC6352_read_heading())/10;
        lcd_init();
        printf(lcd_putc, "\fHeading degree:");
        printf(lcd_putc, "\n    %LuDE", heading);

        delay_ms(1000);
    }
}

```

(4) Source Code of DC Motor Speed Control

```

//*****
*Title: Programming of Four-Wheeled Mobile Robot PIC Brain Board *
*Program Author: Ching Wai Hoong *
*Software: PIC C Compiler (CCS) *
*PIC Type: PIC Microchip 18F4550 *
*PIC-2 *
//*****

#include <18F4550.h>
#fuses NOLVP, NOWDT, HS, NOPROTECT, BROWNOUT, PUT,
#use delay(clock=20M, crystal)

#define SW1      PIN_A0
#define SW2      PIN_A1
#define SW3      PIN_A2
#define SW4      PIN_A3
#define GREEN_LED PIN_B2
#define RED_LED  PIN_B3

void DC_MOTOR(void)
{
    if(!input(SW1)==1)
    {
        output_high(GREEN_LED);
        output_low(RED_LED);
        //Motors Run Forward
        set_pwm1_duty(110);
    }
}

```

```

    set_pwm2_duty(0);
}

else if(!input(SW2)==1)
{
    output_high(GREEN_LED);
    output_low(RED_LED);
    //Motors Run Forward
    set_pwm1_duty(150);
    set_pwm2_duty(0);
}

else if(!input(SW3)==1)
{
    output_high(GREEN_LED);
    output_low(RED_LED);
    //Motors Run Reverse
    set_pwm1_duty(0);
    set_pwm2_duty(110);
}

else if(!input(SW4)==1)
{
    output_low(GREEN_LED);
    output_high(RED_LED);
    //Motors stopped
    set_pwm1_duty(0);
    set_pwm2_duty(0);
}
}

void main ()
{
    int anvolt;
    set_tris_b(0x03);
    set_tris_c(0x00);
    set_tris_a(0b00001111);

    output_low(RED_LED);
    output_low(GREEN_LED);
    output_low(PIN_C2); // Set CCP1 output low
    output_low(PIN_C1); // Set CCP2 output low
    setup_ccp1(ccp_PWM); // Configure CCP1 as a PWM of pin c2
    setup_ccp2(ccp_PWM); // Configure CCP2 as a PWM of pin c1
    setup_timer_2(T2_DIV_BY_16, 150, 1); // Setup for 1220.7Hz
    set_pwm1_duty(0); // 0% duty cycle
    set_pwm2_duty(0); // 0% duty cycle
    delay_ms(10);
    while(true)
    {
        DC_MOTOR(); }}

```

(5) Source Code of Manipulation of Digital Compass Module to Servo Motor

```

//*****
*Title: Programming of Four-Wheeled Mobile Robot PIC Brain Board *
*Program Author: Ching Wai Hoong *
*Software: PIC C Compiler (CCS) *
*PIC Type: PIC Microchip 18F4550 *
*PIC-1 *
//*****

#include <18f4550.h>
#include HS, NOWDT, NOPROTECT, BROWNOUT, PUT, NOLVP
#include <flex_lcd.c>
#include <flex_lcd.c>
#include i2c(master, sda=PIN_B0, scl=PIN_B1, FORCE_HW)

#define HMC6352_I2C_WRITE_ADDRESS 0x42
#define HMC6352_I2C_READ_ADDRESS 0x43
#define HMC6352_GET_DATA_COMMAND 0x41 //Transmit "A" to compass
for heading
// Read the compass heading. This is in unit of 1/10 of a degree
// from 0 to 3599
#define SERVO_CONTROL PIN_A4

unsigned int a;
int16 heading;

int16 HMC6352_read_heading(void)
{
    int8 lsb;
    int8 msb;

    i2c_start();
    i2c_write(HMC6352_I2C_WRITE_ADDRESS); //0x42 is the address that we
need to communicate with to write information to the device
    i2c_write(HMC6352_GET_DATA_COMMAND); //"A" or 0x41 is sent to
magnetometer to get a response that pertains to what direction it is facing
    i2c_stop();
    delay_ms(10);

    i2c_start();
    i2c_write(HMC6352_I2C_READ_ADDRESS); //0x43 is the address that to
communicate with to read information from compass
    msb=i2c_read(); //Response 1 for MSb data after reading the data from compass
    lsb=i2c_read(0); //Response 2 for LSb data after reading the data from compass
    i2c_stop();
    return((int16)lsb | (int16)msb << 8);
}

```

```

void SERVO_MOTOR(void)
{
    ////////////////////control the servo motor turning
    if(heading==359)
    {
        for(a=0; a<10; a++) //servo in neutral position
        {
            output_high(SERVO_CONTROL);
            delay_us(1500);
            output_low(SERVO_CONTROL);
            delay_us(18500);
        }
    }
    else if(heading<=19) //servo -20degree turns left
    {
        for(a=0; a<10; a++)
        {
            output_high(SERVO_CONTROL);
            delay_us(1750);
            output_low(SERVO_CONTROL);
            delay_us(18250);
        }
    }
    else if(heading<=34) //servo -30degree turns left
    {
        for(a=0; a<10; a++)
        {
            output_high(SERVO_CONTROL);
            delay_us(1875);
            output_low(SERVO_CONTROL);
            delay_us(18125);
        }
    }
    else if(heading<=179) //servo -40degree turns left
    {
        for(a=0; a<10; a++)
        {
            output_high(SERVO_CONTROL);
            delay_us(2000);
            output_low(SERVO_CONTROL);
            delay_us(18000);
        }
    }
    else if(heading<=324) //servo +40degree turns right
    {
        for(a=0; a<10; a++)
        {
            output_high(SERVO_CONTROL);
            delay_us(1000);
            output_low(SERVO_CONTROL);

```

```

        delay_us(19000);
    }
}
else if(heading<=339)           //servo +30degree turns right
{
    for(a=0; a<10; a++)
    {
        output_high(SERVO_CONTROL);
        delay_us(1125);
        output_low(SERVO_CONTROL);
        delay_us(18875);
    }
}
else if(heading<=358)           //servo +20degree turns left
{
    for(a=0; a<10; a++)
    {
        output_high(SERVO_CONTROL);
        delay_us(1250);
        output_low(SERVO_CONTROL);
        delay_us(18750);
    }
}
else
{
    for(a=0; a<10; a++)           //servo in neutral position
    {
        output_high(SERVO_CONTROL);
        delay_us(1500);
        output_low(SERVO_CONTROL);
        delay_us(18500);
    }
}
}

void main()
{
    set_tris_b(0b00000011);
    set_tris_a(0x00);
    output_low(SERVO_CONTROL);

    while(true)
    {
        heading=(HMC6352_read_heading())/10;    //get 3 digits value from the
        digital compass
        lcd_init();
        printf(lcd_putc, "\fHead Deg=%Lu", heading);    //display heading degree
        error and error value based on robot position
        SERVO_MOTOR();
    }
}
```

(6) Source Code of Manipulation of Obstacles and Wall Detection

```

//*****
*Title: Programming of Four-Wheeled Mobile Robot PIC Brain Board *
*Program Author: Ching Wai Hoong *
*Software: PIC C Compiler (CCS) *
*PIC Type: PIC Microchip 18F4550 *
*PIC-2 *
//*****

#include <18f4550.h>
#define adc = 8
#define HS, NOWDT, NOPROTECT, BROWNOUT, PUT, NOLVP
#define delay(clock=20M, crystal)
#include <flex_lcd.c>

#define LED PIN_B3

void main()
{
    int anvolt;
    set_tris_b(0x00);
    setup_adc_ports(NO_ANALOGS);
    setup_adc(ADC_CLOCK_INTERNAL);
    set_adc_channel(5);

    while(true)
    {
        delay_us(100);
        output_low(LED);
        anvolt = read_adc();
        lcd_init();
        printf(lcd_putc,"inches=%u",anvolt);
        if(anvolt<=3)
        {
            output_high(LED);
        }
        delay_ms(300);
    }
}

```


(7) Source Code of Four-Wheeled Mobile Robot with PID Control Algorithm

```

//*****
*Title: Programming of Four-Wheeled Mobile Robot PIC Brain Board *
*Program Author: Ching Wai Hoong *
*Software: PIC C Compiler (CCS) *
*PIC Type: PIC Microchip 18F4550 *
*PIC-1 *
//*****

#include <18f4550.h>
#include HS, NOWDT, NOPROTECT, BROWNOUT, PUT, NOLVP
#include <flex_lcd.c>
#include <flex_lcd.c>
#include <flex_lcd.c>
#include <flex_lcd.c>

#define HMC6352_I2C_WRITE_ADDRESS 0x42
#define HMC6352_I2C_READ_ADDRESS 0x43
#define HMC6352_GET_DATA_COMMAND 0X41
//Transmit "A" to compass for heading
// Read the compass heading. This is in unit of 1/10 of a degree from 0 to 3599
#define SERVO_CONTROL PIN_A4
#define Kp 1
#define Ki 0.00125 //can be adjustable
#define Kd 0.0125 //can be adjustable
#define setpoint 359

unsigned int a;
int16 error, error1, error2, error3, error4, heading;
float proportional, derivative;
float PID, output, output2;

int16 HMC6352_read_heading(void)
{
    int8 lsb;
    int8 msb;

    i2c_start();
    i2c_write(HMC6352_I2C_WRITE_ADDRESS);
    //0x42 is the address that we need to communicate with to write information to the
    //device
    i2c_write(HMC6352_GET_DATA_COMMAND);
    //"A" or 0x41 is sent to magnetometer to get a response that pertains to what
    direction it //is facing
    i2c_stop();
    delay_ms(10);

    i2c_start();
    i2c_write(HMC6352_I2C_READ_ADDRESS);
    //0x43 is the address that to communicate with to read information from compass

```

```

msb=i2c_read();
//Response 1 for MSb data after reading the data from compass
lsb=i2c_read(0);
//Response 2 for LSb data after reading the data from compass
i2c_stop();
return((int16)lsb | (int16)msb << 8);
}
float pidcal(void)
//PID control algorithm
{
    static float pre_error = 0;
    static float integral = 0;

    error = setpoint - heading;
    //check error with PID algorithm
    proportional = error;
    integral = integral + error;
    derivative = error - pre_error;
    output = (Kp*proportional)+(Ki*integral)+(Kd*derivative);
    //sum up three terms of PID

    error4 = error + 4;
    error3 = error + 3;
    error2 = error + 2;
    error1 = error + 1;

    if(output == 0)
    {
        output2=0;
    }
    else if(output <= error1)
    {
        output2=error1;
    }
    else if(output <= error2)
    {
        output2=error2;
    }
    else if(output <= error3)
    {
        output2=error3;
    }
    else if (output <= error4)
    {
        output2=error4;
    }
    else
    {
        output2=error4;
    }
}

```

```

    }
    pre_error = error;
    //previous error saved in PIC
    return(output2);
}
void SERVO_MOTOR(void)
{
    ////////////////////control the servo motor turning
    if(PID==0)
    {
        for(a=0; a<10; a++)
            //servo in neutral position
            {
                output_high(SERVO_CONTROL);
                delay_us(1500);
                output_low(SERVO_CONTROL);
                delay_us(18500);
            }
    }
    else if(PID<=5)
        //servo +10degree turns right
        {
            for(a=0; a<10; a++)
            {
                output_high(SERVO_CONTROL);
                delay_us(1375);
                output_low(SERVO_CONTROL);
                delay_us(18625);
            }
        }
    else if(PID<=19)
        //servo +20degree turns right
        {
            for(a=0; a<10; a++)
            {
                output_high(SERVO_CONTROL);
                delay_us(1250);
                output_low(SERVO_CONTROL);
                delay_us(18750);
            }
        }
    else if(PID<=34)
        //servo +30degree turns right
        {
            for(a=0; a<10; a++)
            {
                output_high(SERVO_CONTROL);
                delay_us(1125);
                output_low(SERVO_CONTROL);
                delay_us(18875);
            }
        }
    }
}

```

```

    }
}
else if(PID<=179)
//servo +40degree turns right
{
    for(a=0; a<10; a++)
    {
        output_high(SERVO_CONTROL);
        delay_us(1000);
        output_low(SERVO_CONTROL);
        delay_us(19000);
    }
}
else if(PID<=324)
//servo -40degree turns left
{
    for(a=0; a<10; a++)
    {
        output_high(SERVO_CONTROL);
        delay_us(2000);
        output_low(SERVO_CONTROL);
        delay_us(18000);
    }
}
else if(PID<=339)
//servo -30degree turns left
{
    for(a=0; a<10; a++)
    {
        output_high(SERVO_CONTROL);
        delay_us(1875);
        output_low(SERVO_CONTROL);
        delay_us(18125);
    }
}
else if(PID<=353)
//servo -20degree turns left
{
    for(a=0; a<10; a++)
    {
        output_high(SERVO_CONTROL);
        delay_us(1750);
        output_low(SERVO_CONTROL);
        delay_us(18250);
    }
}
else if(PID<=370)
{
    for(a=0; a<10; a++)
        //servo -10degree turns left

```

```

        {
            output_high(SERVO_CONTROL);
            delay_us(1625);
            output_low(SERVO_CONTROL);
            delay_us(18375);
        }
    }
    else
    {
        for(a=0; a<10; a++)
            //servo in neutral position
            {
                output_high(SERVO_CONTROL);
                delay_us(1500);
                output_low(SERVO_CONTROL);
                delay_us(18500);
            }
    }
}

void main()
{
    set_tris_b(0b00000011);
    set_tris_a(0x00);
    output_low(SERVO_CONTROL);

    while(true)
    {
        heading=(HMC6352_read_heading())/10;
        //get 3 digits value from the digital compass

        PID = pidcal();

        lcd_init();
        printf(lcd_putc, "\fHDeg=%Lu,e=%Lu", heading, error);
        //display heading degree error and error value based on robot position
        printf(lcd_putc, "\nPID=%f",PID);
        //PID algorithm output value by trial&error to adjust Kp, Ki and Kd

        SERVO_MOTOR();
    }
}

```

(8) Source Code of DC Gear Motor Control with Ultrasonic Sensor

```

//*****
*Title: Programming For Four-Wheeled Mobile Robot PIC Brain Board *
*Program Author: Ching Wai Hoong *
*Software: PIC C Compiler (CCS) *
*PIC Type: PIC Microchip 18F4550 *
*PIC-2 *
//*****

#include <18F4550.h>
#define adc = 8
#define fuses NOLVP, NOWDT, HS, NOPROTECT, BROWNOUT, PUT,
#define use delay(clock=20M, crystal)

#define SW1      PIN_A0
#define SW2      PIN_A1
#define SW3      PIN_A2
#define SW4      PIN_A3
#define GREEN_LED PIN_B2
#define RED_LED  PIN_B3

void DC_MOTOR(void)
{
    if(!input(SW1)==1)
    {
        output_high(GREEN_LED);
        output_low(RED_LED);
        //Motors Run Forward
        set_pwm1_duty(110);
        set_pwm2_duty(0);
    }
    else if(!input(SW2)==1)
    {
        output_high(GREEN_LED);
        output_low(RED_LED);
        //Motors Run Forward
        set_pwm1_duty(150);
        set_pwm2_duty(0);
    }
    else if(!input(SW3)==1)
    {
        output_high(GREEN_LED);
        output_low(RED_LED);
        //Motors Run Reverse
        set_pwm1_duty(0);
        set_pwm2_duty(110);
    }
    else if(!input(SW4)==1)

```

```

    {
        output_low(GREEN_LED);
        output_high(RED_LED);
        //Motors stopped
        set_pwm1_duty(0);
        set_pwm2_duty(0);
    }
}
void main ()
{
    int anvolt;
    set_tris_b(0x03);
    set_tris_c(0x00);
    set_tris_a(0b00001111);

    output_low(RED_LED);
    output_low(GREEN_LED);
    output_low(PIN_C2); // Set CCP1 output low
    output_low(PIN_C1); // Set CCP2 output low
    setup_ccp1(ccp_PWM); // Configure CCP1 as a PWM of pin c2
    setup_ccp2(ccp_PWM); // Configure CCP2 as a PWM of pin c1
    setup_timer_2(T2_DIV_BY_16, 150, 1); // Setup for 1220.7Hz
    set_pwm1_duty(0); // 0% duty cycle
    set_pwm2_duty(0); // 0% duty cycle
    delay_ms(10);

    setup_adc_ports(NO_ANALOGS);
    setup_adc(ADC_CLOCK_INTERNAL);
    set_adc_channel(5);
    delay_us(100);

    while(true)
    {
        DC_MOTOR();

        anvolt = read_adc();
        if(anvolt<=4)
        {
            set_pwm1_duty(0);
            set_pwm2_duty(0);
            output_high(RED_LED);
            output_low(GREEN_LED);
            delay_ms(200);
        }
    }
}

```

APPENDIX B

Four-Wheeled Mobile Robot's Specification

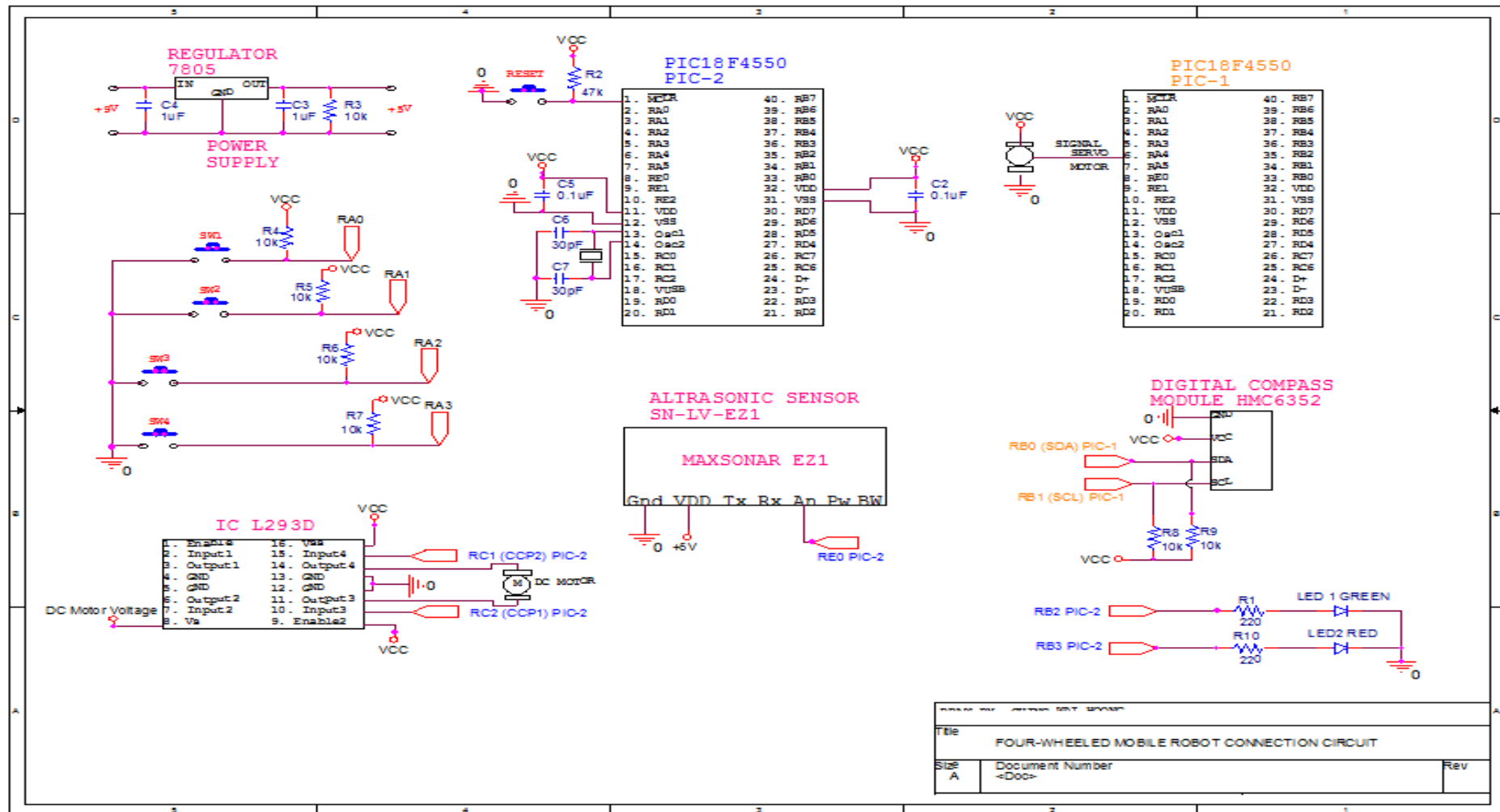
The specification of Four-Wheeled Mobile Robot

Microchip Controller	<ul style="list-style-type: none"> • Programmable PIC 18F4550 microchip controller to program desire source coding to respond, react or manipulate the sensors, switches and other devices. • PIC-1 is used to respond to the PID algorithm output result to manipulate the servo turning degree. • PIC-2 is used to respond to switches for controlling the DC motor rotational speed and react to distance detection of ultrasonic sensor.
Robot's Motion	<ul style="list-style-type: none"> • 1 9V DC geared motor to control two back-wheels. • Rotational speed and forward or backward manner controlled by motor driver IC L293D.
Robot's Turning	<ul style="list-style-type: none"> • 1 5V RC servo motor to adjust and verify two front-wheels turning degree.
Position and Direction Verification	<ul style="list-style-type: none"> • Digital Compass Module is utilized to identify and figure the robot moving direction with pointing to the desired set-point direction angle.
Distance Detection	<ul style="list-style-type: none"> • Ultrasonic sensor is used to verify certain distance repelling from obstacle or wall due to respond to controller to halt the DC motor from crashing tremendously to ruin the robot.
Communications	<ul style="list-style-type: none"> • I²C communication is two wire serial interfaces to communicate to the compass sensor that requires a host device or controller to properly command and function.
Power Source	<ul style="list-style-type: none"> • 8 x AA batteries, Ni-Cd, 1000mAh, 9.6V for supplying to PIC-1 of DC motor rotational speed control. • 1 x 9V recharger batter, Nickel-Metal Hydride, 175mAh
Dimension	<ul style="list-style-type: none"> • 40cm long x 20cm wide x 11cm high
Weight	<ul style="list-style-type: none"> • Approximately 2kg

APPENDIX C

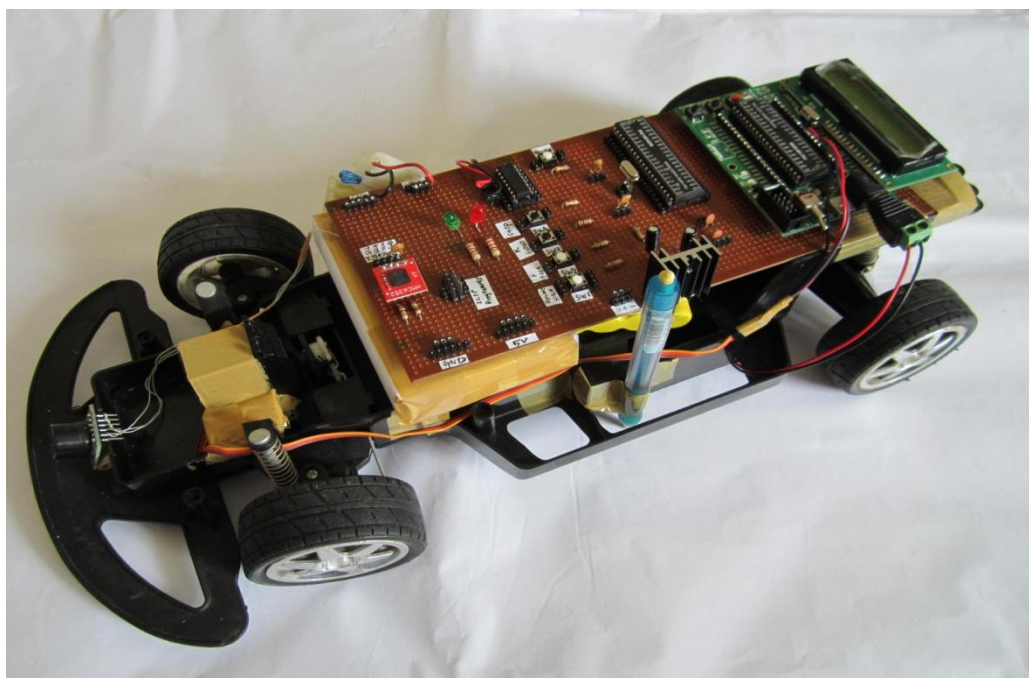
Four-Wheeled Mobile Robot's Overall Circuit Diagram

FWMR's Overall Circuit Diagram

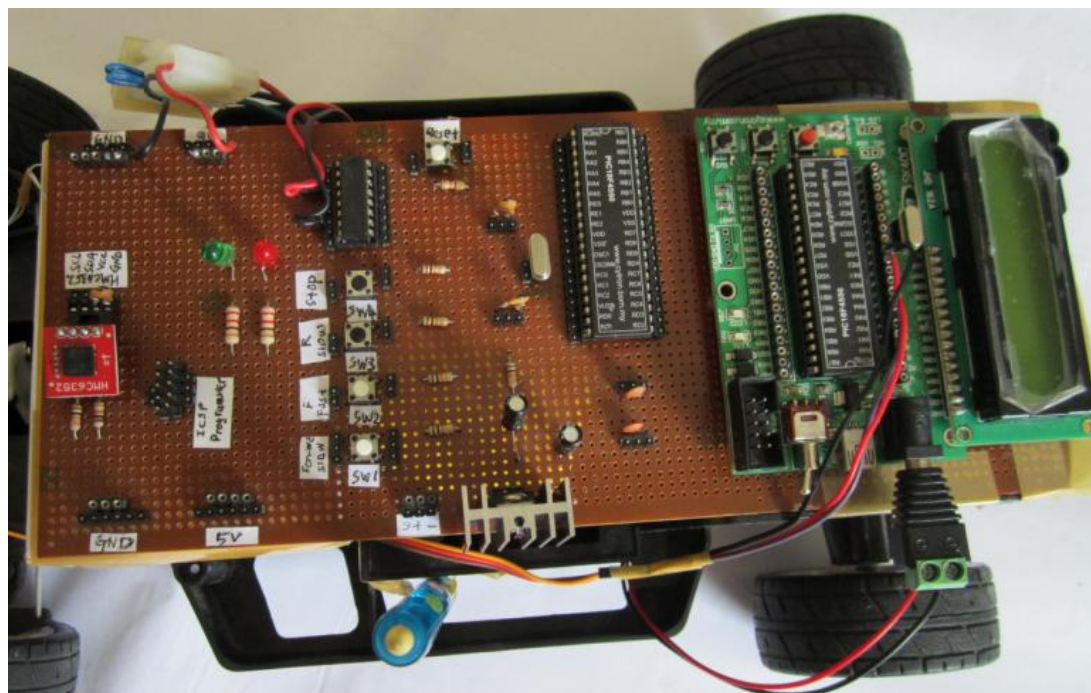


APPENDIX D

Four-Wheeled Mobile Robot's Pictures



Four-Wheeled Mobile Robot



Overall Connection Circuitry of FWMR



Digital Compass Module



Distance Detection of Ultrasonic Sensor