"I hereby declare read through this report entitle "Mobile Robot Navigation System: Line Following Robot by using PID algorithm" and found that it has comply the partial fulfillment for awarding the degree of Bachelor of Mechatronic Engineering"

Signature : [illegible]

Supervisor's Name : Dr fahmi bin miskon

DR. MUHAMMAD FAHMI BIN MISKON
PENSYARAH KANAN
Fakulti Kejuruteraan Elektrik
Universiti Teknikal Malaysia Melaka

Date : 26.6.2015

MOBILE ROBOT NAVIGATION SYSTEM: LINE FOLLOWING ROBOT BY USING PID ALGORITHM

LING SING YIP

A report submitted in partial fulfillment of the requirements for the degree of Bachelor of Mechatronic Engineering

Faculty of Electrical Engineering

UNIVERSITI TEKNIKAL MALAYSIA MELAKA

2014/2015

I declare that this report entitle "Mobile Robot Navigation System: Line Following Robot by using PID Algorithm" is the result of my own research except as cited in the reference. The report has not been accepted for any degree and is not concurrently submitted in candidature of any other degree.

Signature : [illegible]

Name : Ling Sing Yip

Date : 26.6.2015

To my beloved mother and father

ACKNOWLEDGEMENT

In preparing this report, I would like to express my sincere appreciation to my supervisor, Dr Muhammad Fahmi Bin Miskon that gave me the opportunity and accepted me to be his supervise. As my supervisor, he gave me encouragement, guidance critics and friendship. Without his support and interest, this project would not have been same as presented here.

I am also indebted to Universiti Teknikal Malaysia Melaka for funding and providing resources and relevant literatures for my project. Then shall give to all my friends who provide me assistance and advice. Other than that, I am also grateful to all my family members.

ABSTRACT

According to the World Health Ranking 2011, it has ranked Malaysia as the top 20 countries with the most death causes of road accidents. Hence, autonomous navigation robot is to compromise and reduce the car accident. Conventional line following robot has slow response and the robot would not be able to follow the line smoothly and sometimes robot tends to move out from the track. This report will present a line following robot by using PID algorithm to track a line. The objective of this paper is to design a high speed and autonomous mobile robot that follows a line accurately by using PID algorithm. In this case, PID algorithm is used to reduce wobbling motion and rectify the error if it leaves the line. This particular robot is an autonomous robot which is able to follow a black line drawn on a white surface while maintaining a smooth tracking motion. Two experimental tests were proposed to achieve the objective of the project. Experiment 1 was conducted to test the performance of line following robot while moving at the high speed while experiment 2 to test the sensitivity of sensor which will impact the accuracy performance of the robot. Result from first experiment shown that the PID controller much faster and accuracy compares to P controller which only taken 10s to complete the track compared to P controller needed 14.11s. For accuracy test, the average error detected by PID controller is -0.7333 while for P controller is -1.0556. By implementing of PID controller closer to the desired value (zero) compare to P controller. Lastly, for experiment 2, it shows that the distance between the surface and sensor will affect the performance in term of accuracy of the sensor. The result shows that the sensor needs to locate 0.6cm from the surface to obtain optimum output.

ABSTRAK

Menurut kajian World Health Ranking tahun 2011, Malaysia telah diiktirafkan sebagai Negara berada dalam kedudukan 20 teratas berdasarkan kemalangan maut berlaku di jalan raya. Oleh hal yang demikian, robot berasaskan konsep navigasi tanpa kawalan manusia wajarnya dicipta untuk mengurangkan kadar kemalangan jalan raya. Kertas kerja ini akan membentangkan tentang robot pengikut garisan dengan menggunakan sistem kawalan PID algoritma. Objektif kertas kerja ini adalah untuk menghasilkan robot pengikut garisan yang berkelajuan tinggi dan tepat ketika mengikut garisan dengan menggukan sistem kawalan PID algoritma. Sistem kawalan PID algoritma digunakan untuk mengurangkan kegoyangan ketika bergerak dan memperbetulkan gerakan jika ianya terkeluar dari garisan. Robot ini merupakan robot yang pintar kerana mampu bergerak mengikuti garis hitam yang dilukis pada permukaan putih serta dapat mengawal pergerakan dengan lancar. Terdapat 2 eksperimen yang di jalankan untuk mencapai objektif kertas kerja ini. Eksperimen yang pertama adalah unutk menguji prestasi robot pengikut garisan ketika bergerak dalam kelajuan tinggi, manakala eksperimen kedua adalah untuk menguji kepekaan sensor yang mempengaruhi ketepatan pergerakan robot. Hasil kajian dari eksperimen pertama menunjukkan bahawa kawalan PID lebih pantas dan tepat berbanding kawalan P dengan mengambil masa 10 saat untuk melengkapkan satu pusingan berbandingkan kawalan P mengambil masa 14.11 saat. Untuk ujian ketepatan, purata ralat yang dikesan menggunakan kawalan PID ialah -0.7333 manakala untuk kawalan P ialah -1.0556. Dengan melaksanakan kawalan PID pada sistem, ia akan menghampiri nilai yang dikendendaki (zero) berbanding dengan kawalan P. Eksperimen terakhir menunjukkan bahawa jarak antara permukaan landasan dan sensor akan mempengaruhi prestasi iaitu kepekaan sensor tersebut. Hasil kajian menunjukkan sensor tersebut mestilah berada di atas paras 0.6cm daripada permukaan landasan untuk memperoleh output yang optimum.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF APPENDICES

CHAPTER 1

INTRODUCTION

1.1 Motivation

The World Health Ranking 2011 has ranked that Malaysia are the top 20 countries with the most death cause by road accident. Road traffic accidents have been identified top cause of death in Malaysia after heart disease, stroke and pneumonia. Based on report by the Department of Road Safety Malaysia, there were increments of 21 percent in the number of vehicles involved in car accidents from year 2005 until 2009. [19]

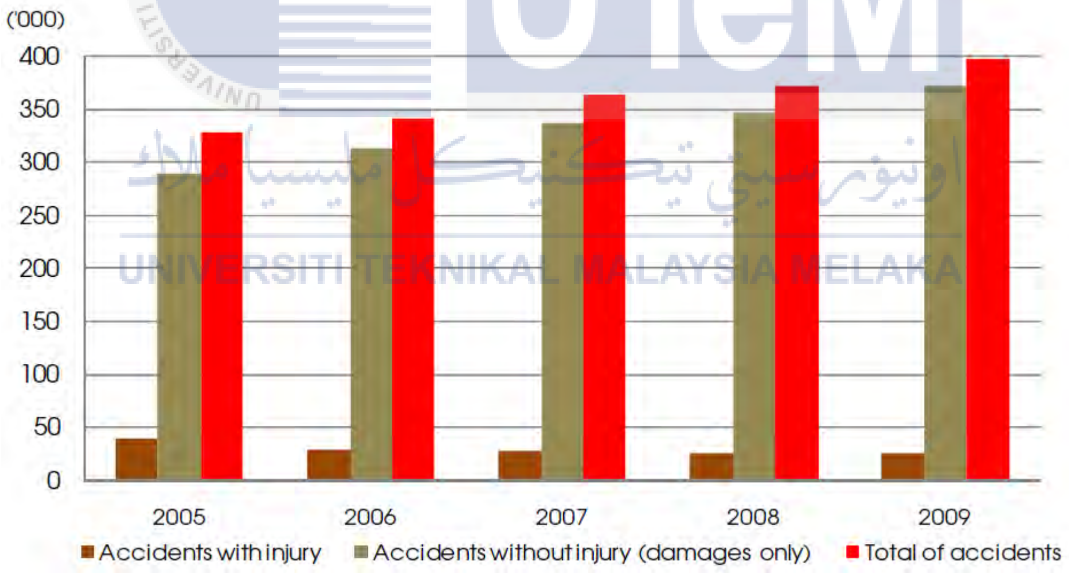


Figure 1.1: Number of accidents, Malaysia, 2005-2009 [1]

The figure 1.1 shows that the number of vehicles involved in accidents has increased in year 2009 (705,623) compared to 2005 (581,082), overall an increase of 21.4 per cent[1]. Almost all categories of vehicles showed an increment in vehicle accidents led by cars & taxis, followed by motorcycles & bicycles, trucks, vans & four wheel drives, buses and others.

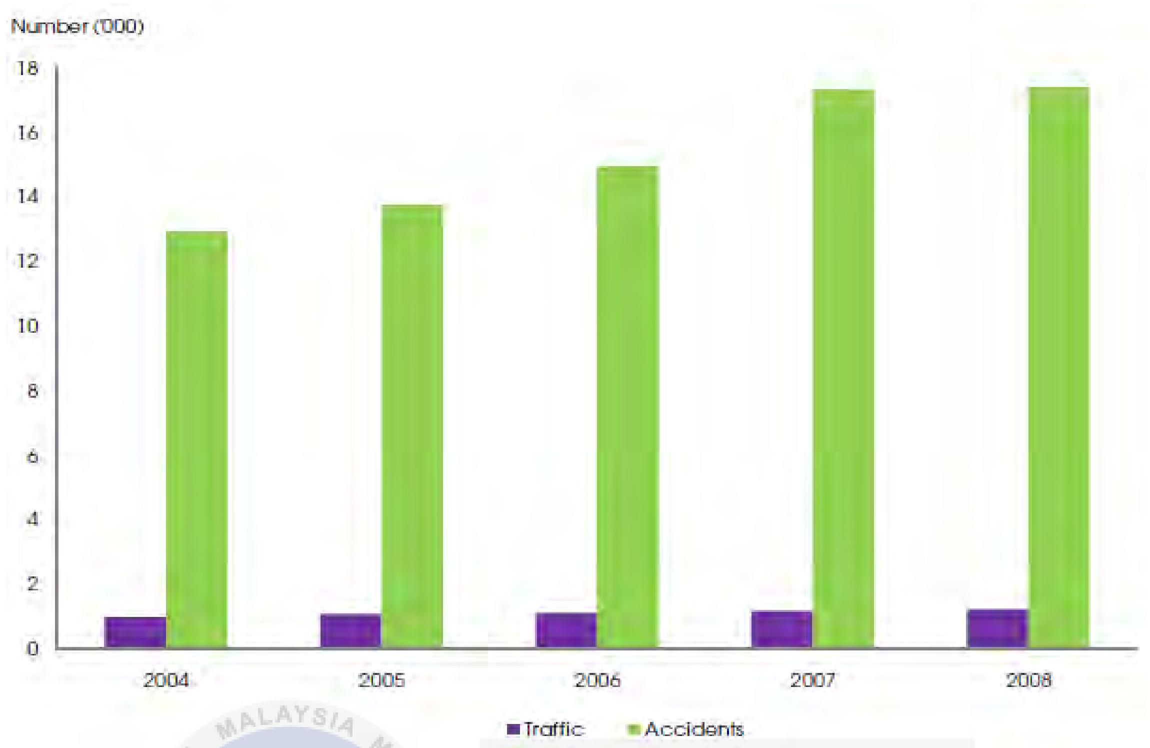


Figure1.2: Number of traffic and highway accidents, Malaysia, 2004-2008 [1]

Highways provide an alternative route to most of the road users. Other than providing a convenient travel option, users may also shorten the travelling time. Until year 2008, there were 24 highways fully-operating in Malaysia. Overall, the traffic volumes have increased to 1,232.6 million in 2008 compared to 1,001.2 million in 2004, an increase of 23.1 per cent. Increased traffic volumes indirectly contributed to the increased number of accidents on the highways. There were a total of 17,369 highway accidents recorded in 2008 compared to 12,949 in 2004, an increase of 34.1 per cent [1].

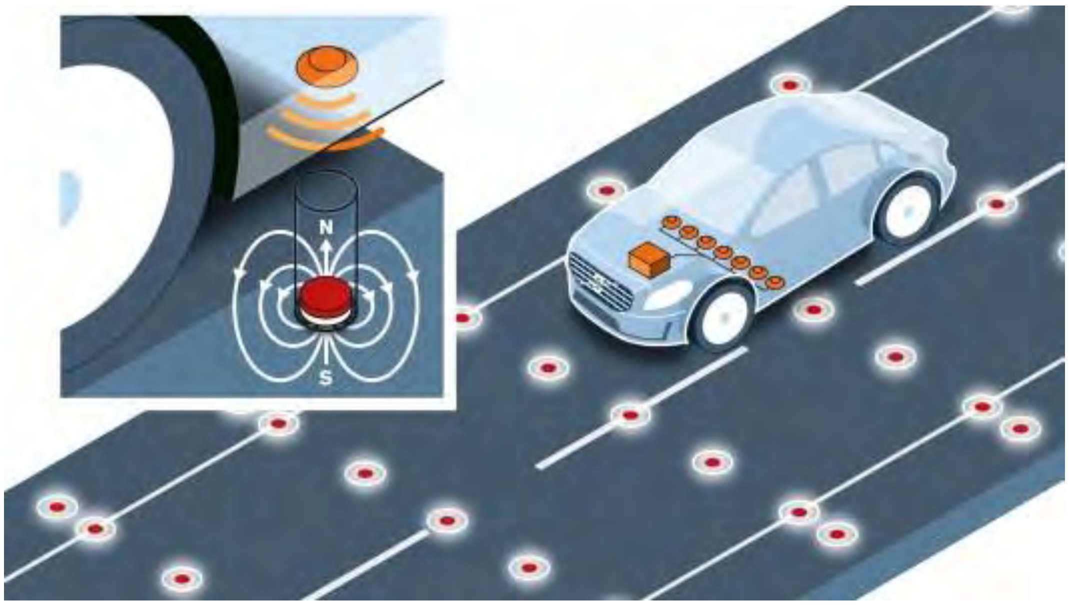


Figure 1.3: Volvo self-driving line following car by use of magnets

Source: https://www.media.volvocars.com/global/en-gb/media/pressreleases/140760/photos

Autonomous navigation robot is to compromise and reduce the car accident. The line following robot is under the category of autonomous navigation robot which can detect visible like detect a line or it can appear invisible like a magnetic field. Line following robot has been a dream of mankind a long time ago. Car manufacturing company, Volvo work on autonomous vehicle with a research project on line following that that use of magnets to keep self-driving cars on the road. Volvo Company pledging to have self-driving autos on 2017 [14]. Incorporating magnet-based positioning in preventive safety systems could help prevent run-off road accidents. Magnets could facilitate accuracy of winter road maintenance, which in turn could prevent damage to snow-covered objects, such as barriers and signs, near the road edge. Upon the researches made, the rate of car accident will be dramatically reduced and the dream of mankind shall be realized in near future. Autonomous car navigation will be a boon to the society. In this project, using the same principle like Volvo Company by replacing the line on the group or track. PID controller will be designed for line following robot to track line.

1.2 Problem Statement

Classical or the line following robot without PID controller has slow response and the robot will not be able to follow the line smoothly and sometimes robots tends to move out from the line. If there is maximum speed beyond which cannot use this classical algorithm, otherwise the robot will overshoot from the line. This problem also will cause the motion of the robot to wobble and its path similar a zigzag pattern, it is wasting valuable time and power supply. Although the mobile robot can follow the line track, its motion and overshoot problem while following in maximum speed needs to be improved. The line following robot faced difficulties to obtain stable and precise navigation. When the speed of the line following robot increases, the robot would not be able to navigate effectively and follow a line precisely. The robot also would not be able to take a sharp turn and tend to move out from the track. These operating problems due to some constraints, slow responses of sensor, time delay and other disturbances. In order to overcome the problem, a better controller is needed to make the robot follow the line accurately, smoothly and without leaving the track of course. The robot will gradually go steer in this project, PID controller is used to improve the motion which forms an effective closed loop system. In developing a PID controller for a line following robot, there are a few requirements which needed to be considered in order to complete the project. Therefore, implementation of PID controller would be able to overcome the problem and increase the performances of the line following robot.

1.3 Project Objective

There are few objectives that need to achieve by end of the project. The objectives of this project are:

i. To design and develop a PID controller for line following robot that able follows a line in high speed and accurately to reduce wobbling motion (oscillation) and be able to rectify the error if it leaves the line.

ii. To test the speed and accuracy of the robot movement by judging its forward motion and overcoming a curvy track turn.

1.4 Scope of projects

i. This project covered the design and develop a PID algorithm controller for a line following robot.
ii. Arduino UNO is selected as microcontroller of the line following robot.
iii. Proportional, integral and derivative controller as a control system of the line following robot.
iv. The speed of the robot is verified by using the PWM module from minimum speed to maximum speed.
v. In this project, the proving of the design method is done by doing lab experiments.
vi. The mobile robot use in this project was small in size, the dimension of the mobile robot was 120mm width and 130mm long.
vii. The mobile robot use 5 sets of IR sensor to communicate with the system.

1.5 Report Outline

This report is about the line following robot by using PID algorithm. In this report, the chapter 1 will cover about the motivation for designing a line following robot by using PID algorithm. The objective and scope of the design will also be stated in this chapter. The theory and basic principle and the review of previous related work of the line following robot by using a PID algorithm will be covered in chapter 2 of the report. This report will also list out the methods and techniques used in this design and fully explained in chapter 3. The preliminary result and the conclusion will be covered in chapter 4 and chapter 5 in this report.

CHAPTER 2

THEORETICAL BACKGROUND AND LITERATURE REVIEW

2.1 Introduction

A Line follower is a machine that can follow a path. The path can be visible like black tape on a white surface (or vice-versa) or it can appear invisible like a magnetic field [2]. The essential method to build a line follower is sensing a line and manoeuvring the robot to stay on course, while constantly tuning its error (wrong moves) using feedback mechanism and forms a simple yet effective closed loop system. Line following robot is mobile robot that widely used in different are, especially in industry field. These robots function as material carrier to deliver the product from one place to another where the conveyor and rail not possible. Apart from line following capabilities, this robot should also capability to navigate. Sensor positioning also plays an importance role in optimizing robot navigation performance. Over the past, there are many method for controller has been develop to increase the performances of the robot in term of navigation such as PID controller, PI controller, neutral network and fuzzy logic.

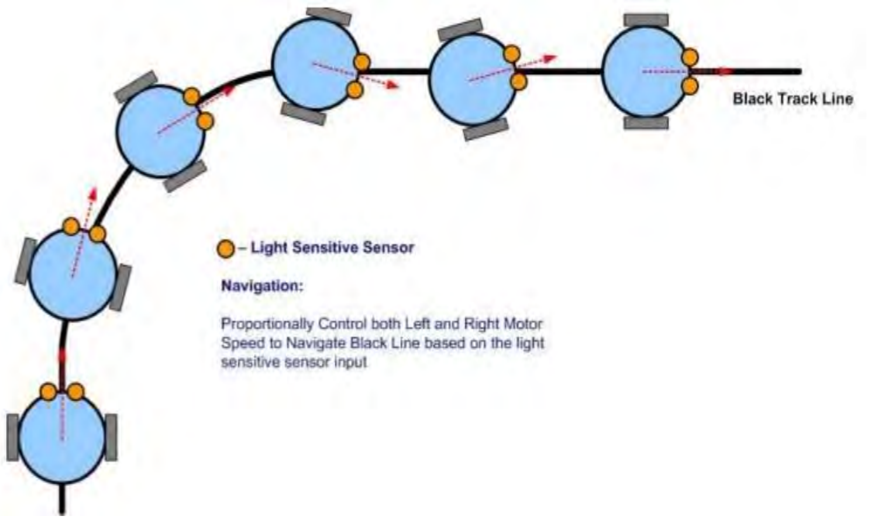

Black Track Line
● – Light Sensitive Sensor
Navigation:
Proportionally Control both Left and Right Motor Speed to Navigate Black Line based on the light sensitive sensor input

Figure 2.1: Line tracking navigation principle on the line follower robot

Source: http://www.ermicro.com/blog/wp-content/uploads/2011/01/op-amp_lfr_01.jpg

2.1.1 History of PID Controller

PID controllers date to 1890s governor design. PID controllers were subsequently developed in automatic ship steering. Elmer Sperry is the person develops the PID type controller in 1911. However, the first published theoretical analysis of a PID controller was done by Russian-American engineer Nicolas Minorsky, in Minorsky in year 1922. In early history, PID controller implemented as a mechanical device. These mechanical controllers use a lever, spring and a mass and were often energized by compressed air [2].

2.2 Theoretical Background of PID algorithm

A proportional-integral-derivative controller (PID controller) is a generic feedback controller. A PID controller calculates an "error" value as the difference between a measured process variable and a desired set point. The controller attempts to minimize the error by adjusting the process control inputs [3].

The PID algorithm accounts the following three things are existing error, the time where the system stay away from the mean position and the possible of overshooting the mean position.

The PID controller calculation algorithm involves three separate constant parameters, and is accordingly sometimes called three-term control: the proportional, the integral and derivative values, denoted by terms P, I, and D. Simply put, these values can be interpreted in terms of time. P depends on the present error, I on the accumulation of past errors, and D is a prediction of future errors, based on current rate of changing.

The PID control scheme is named after its three correcting terms, whose sum constitutes the manipulated variable (MV). The final form of PID algorithm is:

$$u(t) = MV(t) = Kpe(t) + Ki \int_0^t e(\tau)d\tau + Kd \frac{d}{dt}e(t) \quad (2.1)$$

Kp	: Proportional gain, a tuning parameter
Ki	: Integral gain, a tuning parameter
Kd	: Derivative gain, a tuning parameter
e	: Error $= SP - PV$

t : Time or instantaneous time (the present)

τ : Variable of integration; takes on values from time 0 to the present$t$.

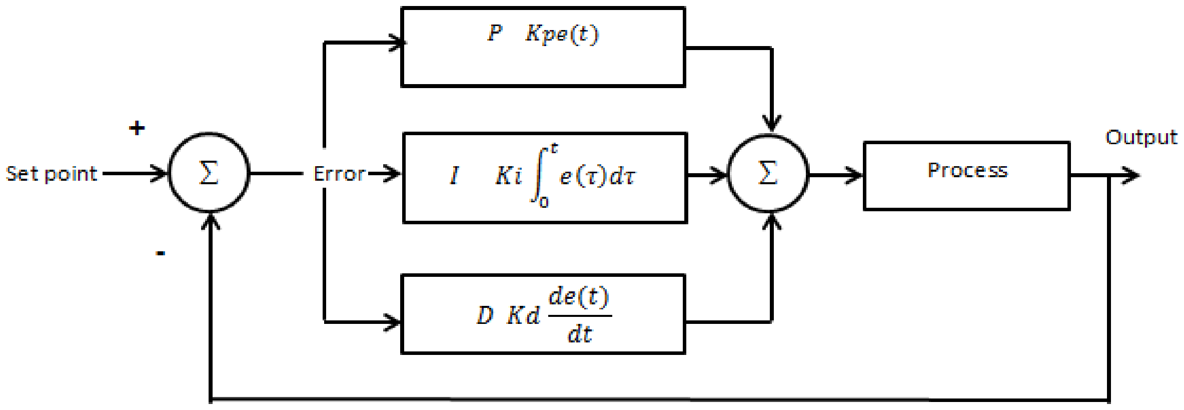


Figure 2.2: A block diagram of a PID controller in a feedback loop

2.2.1 Proportional term

The proportional term produces an output value that is proportional to the current error value. The proportional response can be adjusted by multiplying the error by a constant K_p, called the proportional gain constant [3].

The proportional term given:

$$Pout = Kp\, e(t) \qquad (2.2)$$

2.2.2 Integral term

The integral term is proportional to both the magnitude and duration of the error. The integral in PID controller is the sum of the instantaneous error and give the accumulated offset that should have been corrected previously. The accumulated error is then multiplied by the integral gain K_i and added to the controller output [3].

The integral term given:

$$I\ out = Ki \int_0^t e(\tau) d\tau \qquad (2.3)$$

2.2.3 Derivative term

The derivative of the process error is calculated by determining the slope of the error over time and multiplying the rate of change by the derivative gain Kd. The magnitude of the contribution of the derivative term to the overall control action is termed the derivative gain, Kd [3].

The derivative term given:

$$Dout = Kd\frac{d}{dt}e(t) \qquad (2.4)$$

2.4 Comparison among line following with PID and without PID algorithm

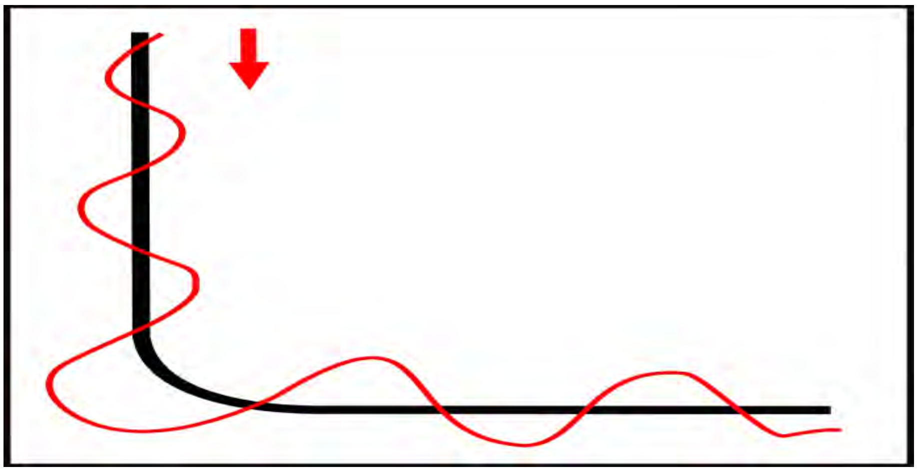


Figure 2.10: Conventional line following robot follows a line

Source: http://www.societyofrobots.com/member_tutorials/node/355

A conventional robot would follow a line as shown in the figure. The figure 2.10 clearly shows that the robot oscillates along the line. It wasted a lot of valuable time and power [5].

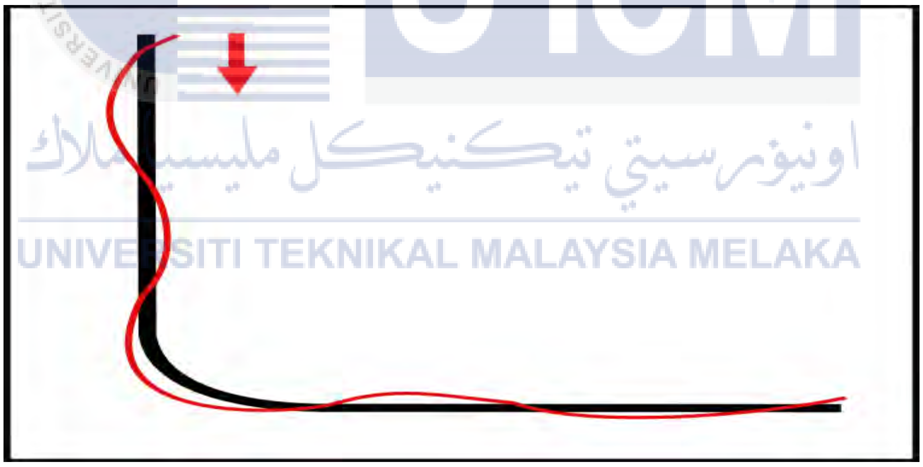


Figure 2.11: Robot movement by applying the PID algorithm

Source : http://www.societyofrobots.com/member_tutorials/node/355

Meanwhile, the figure 2.11 shows the robot movement that wants to achieve with this project. The mobile robot follows the line and keeps in the center. In the straight line, the robot will stabilized gradually while moving straight and be able to follow the line quickly and efficiently [4].

2.5 Available Solutions Trade off from previous study

A line following robot is an autonomous robot which is capable of following the line or track lines on the floor. From the previous research the robot would easily leave the track by using the open loop system. Besides that, this will also make the motion of the robot to keep on oscillating. As a result, PID algorithm is used for smoothening the tracking motion. The PID control system used to sense the line and maneuver the robot to stay on the line and the feedback mechanism correcting the wrong motion, thus might forming an effective closed loop system.

To solve the robot tracking error, Oscar (2008) [9] had implementation Kalman filter in a mobile robot movement in the earlier year. Based on the results of the study, despite the errors in measurement, but the filter still can perform quite well in estimate and position. A modified method by Sivaraj (2011) [8] introduced an automatic steering control by using the Kalman filter and PID controller, which combine Kalman filter and PID. This report proposed control architecture for automatic steering acceleration and braking control of a self-guided vehicle. Kalman filter is the first level of error calculation while PID control algorithm uses to filtered value and calculates the required steer to achieve zero lateral error. The combination of Kalman filter with PID for lateral control reduces trajectory error to a minimum level and adaptive speed control algorithm for longitudinal control provides smooth accelerations over the entire track. This method more reliability than Oscar (2008) method [9], due to the parameter P maintained as lower than Oscar method. The lower the P value indicates the more accurate is the estimated state of the system. Sivaraj (2011) [8] has proved to be better when compared in terms of reliability, accuracy and speed.

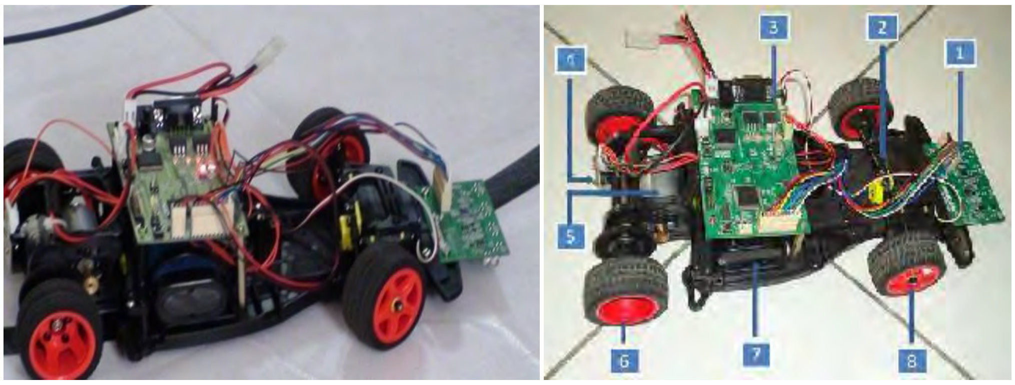


Figure 2.12: Smart car structure by D.Sivaraj [9]

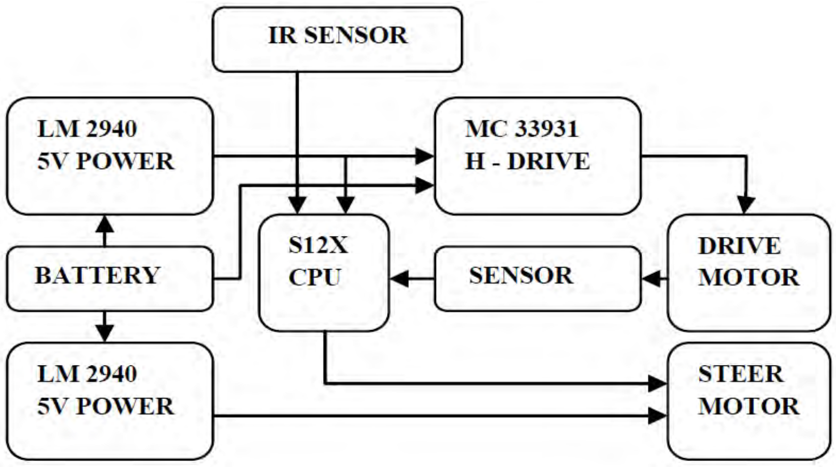


Figure 2.13: Smart car system architecture by D.Sivaraj [9]

The smart car structure by D.Sivaraj is show in Figure 2.12, which the design consists of Sensor array (1) show in the figure, servo motor (2), Controller board with 16-bit MC9S12×[5] (3), encoder for rear wheel drive mechanism (4), DC motor (5), Rear Axle (6), battery (7) and Front axle for front wheel steers mechanism (8).

The research paper by Padhy and Majhi (2006) [10] focused on path tracking problem of a mobile robot by using a PI controller. These methods used an ideal relay in parallel to a conventional PI controller. The gain and phase margin of this method has a good compromise between performance and robustness. On the other hand, Xiuzhi and colleagues (2012) [11] presented a mobile robot guidance approach based on tracking of straight line marked on the ground. PD and PID are designed for mobile robots to track their control over performances when compared with experiment. Xiuzhi and colleagues (2012) [11] clarified that PID performance is not expected as one expected if the measurement error in line's parameter cannot be neglected. PD controller has better performance as demonstrated by the experimental results.

Research paper by Shahril (2012) [6] focused on the development of photoelectric sensor based on intelligent track racing car. The design can recognize a racing track automatically by differentiating the background and the track. From the research, the photoelectric sensor based intelligent track racing car developed in order to investigate the accuracy of the line tracking with low speed and high speed. Similar papers by Sivaraj (2011) [8] also focus on the parameters like accuracy and speed. In order to make the steering more accurate and smooth, the proportional and derivative control mechanism was incorporated into the chosen algorithm. The result proved that Sivaraj (2011) [8] method better than Shahril (2012) [6] in terms of speed and tracking accuracy.

Table 2.1: Comparison of line following robot [[7][8][9][10][11]]

Algorithm/ Method	PID algorithm [7] i. Binary ii. Gray Scale iii. X-Y	Kalman Filter and PID controller [8]	Kalman Filter [9]	PI algorithm [10]	PD and PID algorithm [11]
Performance	Proposed method proves efficient in term of tracking and speed accuracy	Deviation and the tracking efficiency remained stable even at higher speeds	Despite of the error in measurement, the filter can perform quite well in estimating the true position of the robot	Good compromise between performance and robustness	PD controller has better performance
Speed, m/s $V = \frac{d}{t}$	i. 0.99m/s ii. 0.86m/s iii. 0.77m/s	The fastest 12s versus 35% duty cycle The slowest 28s versus 25% duty cycle.	The fastest 15s versus 35% duty cycle The slowest 22s versus 25% duty cycle.	N/A	N/A

Accuracy, (%)	i. 98.62% ii. 82.03% iii. 69.06%	<u>Highest</u> 100% tracking accuracy versus 25% duty cycle <u>Lowest</u> 60% tracking accuracy versus 40% duty cycle	<u>Highest</u> 100% tracking accuracy versus 25% duty cycle <u>Lowest</u> 0% tracking accuracy versus 40% duty cycle	N/A	N/A
Limitation	N/A	N/A	N/A	N/A	-PD and PID controller is commonly used in controlling nonlinear agents; their operation is limited to a comparably narrower operating region.

2.5.1 Microcontroller

According to Prof.Dr.Subhash.P design, [16] the ATmega 16 is a low-power CMOS 8-bit microcontroller. By applying and executing instruction in a single clock cycle, Atmega 16 achieves throughputs approaching 1 MIPS per MHz, which can optimize power consumption versus processing speed. In Zahfi.B and Izham.Z [15] design, the controller chosen is PIC16F877A manufactured by Microchip Inc due to it simplicity and easily obtainable with a reasonable cost incurred. In D.Sivaraj and A.Kandaswamy [8] [9], the line following robot is controlled by Freescale S12X and MC9S12×[5] controller. The controller is high-performance 16 bit microcontroller (MCUs) and digital signal controllers (DSCs) for automotive and industrial application. Anirudh.S and Aravind .K [18] used the Arduino UNO as microcontroller board and Arduino Programming Language for coding it.

Table 2.2: List of detail of the controller from previous study

Title	Algorithm/ Method	Controller	Explanation
Design of Automatic Steering Control and Adaptive Cruise Control of Smart Car [7]	PID algorithm	MC9S12×[5]	• High-performance 16 bit microcontroller (MCUs) and digital signal controllers (DSCs) for automotive and industrial application.
Implementation of AVCS uses Kalman Filter and PID Controller in Autonomous Self-Guided Vehicle [8]	Kalman Filter and PID controller	Free Scale S12X	-
Analysis of Line Sensor Configuration for the Advanced Line Following Robot [15]	Line Following algorithm	PIC16F877A	• Manufactured by Microchip Inc. • Easily obtainable with a reasonable cost incurred.
Develop of Intelligent	Black path	ATemega 16	• Low–power CMOS 8-bit

Line Follower's Robot [16]	follower preference theory		microcontroller. • Approaching 1MIPS per MHz allowing the system designed to optimize power consumption versus processing speed.
Implement of PID control to reduce wobbling in a line following robot [18]	PID algorithm	Arduino Uno Microcontroller	• Brain of the robot • Control the robot to certain degree and artificial intelligence

2.5.2 Sensors

In D. Sivaraj [8] design, he uses an IR sensor array which consists of four SFH4550 Infrared LEDs and seven SFH314 NPN phototransistors. SFH4550 infrared LEDs which provide high radiant intensity, short switching time and narrow emission while SFH314 NPN phototransistors having good radiant sensitive area. A high error detection and correction IR sensor module used by A.Kandaswamy and D.Sivaraj [7] consists of four numbers of Infrared LEDs and eight numbers of NPN phototransistors. The main sensor used in M.Zafri and Izham.Z [15] are ultra-bright red LED combined with a light dependent resistor (LDR). According to them, the LED-LDR combination sensor was low cost and popular alternative choice. The TCRT-LFSM-Digital sensor use by Prodf.Dr.Subhash [16] as a general-purpose proximity or reflectance sensor.The design consists of 5 IR emitter and receiver pairs each phototransistor. The design by Anirudh.S and Aravind.K [18] use the IR transmitter receiver pair for differentiating between the line and the background. The transmitter sends out IR rays while the receiver receives the intensity and calculate it if it is over the line or surrounding.

Table 2.3: List of detail of the sensor from previous study

Title	Algorithm/ Method	Sensor	Explanation
Design of Automatic Steering Control and Adaptive Cruise Control of Smart Car [7]	PID algorithm	Interfaced with IR sensor array	• Consists of 4 infrared LEDs, which provide high radiant intensity • 8 numbers of NPN phototransistor which having good radiant sensitive area.
Implementation of AVCS using Kalman Filter and PID Controller in Autonomous Self-Guided Vehicle [8]	Kalman Filter and PID controller	-Four SFH4550 Infrared LEDs -Seven SFH314 NPN phototransistor	• SFH4550 infrared LEDs provide high radiant intensity, short switching time and narrow emission. • SFH314 NPN phototransistor having good radiant sensitive area.
Analysis of Line Sensor Configuration for the Advanced Line Following Robot [15]	Line Following algorithm	Sensor with Slave controller	• Ultra bright red LED combined with a light dependent resistor (LDR) • LED-LDR combination sensor chosen for low cost and popular alternative choice.
Development of Intelligent Line Follower's Robot [16]	Black path follower preference theory	TCRT-LFSM-Digital sensor	• The general purpose proximity and reflectance sensor • Module consists 5 IR emitter and receiver pairs each phototransistor.
Implement of PID control to reduce wobbling in a line following robot [18]	PID algorithm	IR transmitter receiver pair	• Transmitter sends out IR ray • Receiver the intensity and calculate if it is over the line or the surrounding.

2.5.3 Actuators

According D.Sivaraj [8], the actuator that used are servo motor and DC motor. Servo motor with front axle for front wheel steer mechanism while the DC motor used for rear wheel drive mechanism. Next, the motor selected by M.Zafri and Izham.Z [15] is the VEXTA 15W DC Motor with motor driver cards. This type of motor operates at 24V DC and the operating torque from 0.4lb/inch to 260 lb/inch. In Prof. Dr. Subhash [16] design, two DC motors are selected. The motor will convert electrical energy into mechanical energy through the concept of rotating for wire coil to produce the EMF. The actuators used by Anirudh. S and Aravind. K [18] brush DC motors with a reduction gearbox which the motor can operate at 60 RPM and have sufficient torque to carry the robot.

Table 2.4: List of detail of the actuator from previous study

Title	Algorithm	Actuator	Explanation
Design of Automatic Steering Control and Adaptive Cruise Control of Smart Car [7]	PID algorithm	Servo motor DC motor	• Servo motor as a steering mechanism.
Implementation of AVCS using Kalman Filter and PID Controller in Autonomous Self-Guided Vehicle [8]	Kalman Filter and PID controller	Servo motor DC motor	• Servo motor use for front wheel steer mechanism. • DC motor which includes real axle and an encoder for rear wheel drive mechanism.
Analysis of Line Sensor Configuration for the Advanced Line Following Robot [15]	Line Following algorithm	VEXTA 15W DC motors	• Motor operates at 24V DC • Operating torque from 0.4lb/inch to 260 lb
Develop of Intelligent Line Follower's Robot [16]	Black path follower preference theory	DC motor	• Two DC motor use in the robot • The motor converts electrical energy into

			mechanical energy
Implement of PID control to reduce wobbling in a line following robot [18]	PID algorithm	Brushed DC motor with a reduction gearbox	• Operate at 60 RPM which have sufficient torque to carry the robot.

2.6 Summary

Based on the Table 2.2, 2.3 and 2.4, there are 5 examples of design in term of controller, actuator and sensor used. The algorithm that will be applied in this project is by using the PID algorithm. According to Anirudh, S. [18] design, by implementing the PID into their design, the robot following the line more efficiency with very less wobbling. PID algorithm also can reduce the overshooting of the robot while robot trying to return to the line.

Different sensor is used in each of the designs. However, the performance of the entire sensor used is almost the same, but costly due to its high accuracy. The sensor used for this project is IR line following sensors which built inside Arduino robot. Based on the design of A. Kandaswamy and D. Sivaraj. [7], the IR sensor module is used. The reflected IR induces a great diminishing effect on the output voltage from the white surface in comparison from the black surface.

Based on the Table 2.4, it shows that the previous designs use a DC motor as the actuator. Before starting the project, ensure the DC motor has sufficient torque to carry the load. DC motor is used to control the wheel. The motion that can achieve is moving forward, backward and turning left or right. The motion will be controlled by the microcontroller.

Refer to all the available trade off from the previous study, it's difficult to compare different method done by other people numerically because the test and the prototype in different of design. However, we can see the performance and design one by one. From there, we can approximate what will be the performance. Nevertheless, in this project will compare by implementing different of method by myself.

For this reason, this project will use a PID algorithm and the performance will be tested to see whether it can really follow line in high speed and accuracy.

CHAPTER 3

RESEARCH METHODOLOGY

3.1 Introduction

The project is started by conducting a literature review. Literature review consists of the previous research and the method is used as a guide for this chapter. This chapter consists of the setup of experiment and simulation of this project. The experiment setup is used to test the speed performance and the sensitivity of the sensor, to make sure the robot can move at high speed and follows a line accurately.

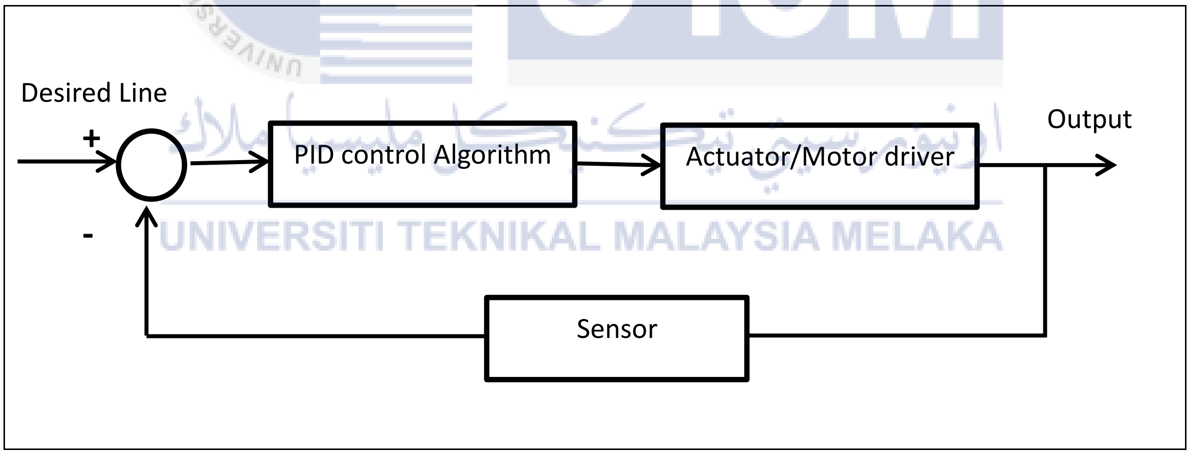


Figure 3.1: Closed loop system for line following robot

The figure 3.1 shows a closed loop system for line following robot. The sensor used in this project is infrared (IR) sensor, which could detect the black line on the white surface. The output signal from the sensor will be sent as a feedback to the microcontroller. Next, it will use PID algorithm to execute and calculate the error that were produced and give a proper signal to an actuator or motor drive.

3.2 Project Methodology Flowchart

Firstly, it is needed to conduct research and study about the literature review related from journal and relevant paper. Figure 3.2 shows the flowchart of the methodology to conduct the project.

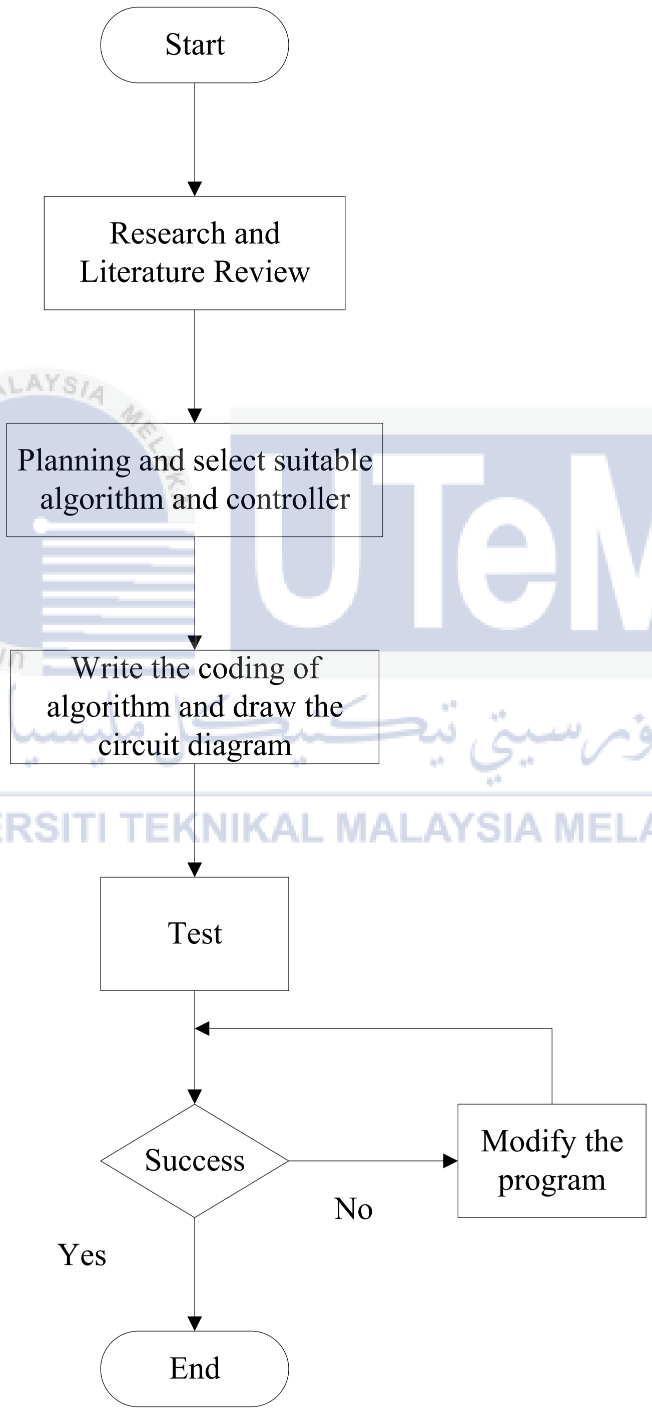

Start
Research and Literature Review
Planning and select suitable algorithm and controller
Write the coding of algorithm and draw the circuit diagram
Test
Success
No
Modify the program
Yes
End

Figure 3.2: Methodology of Project

3.3 The Process Flow diagram

In this part, the overall process flow of this project is described thoroughly. It consists of hardware design and software design. In hardware part, acrylic is selected as the platform of line following line and the platform will be constructed. Meanwhile, in software construction, the circuit for Arduino, motor driver and sensor is designed using the Proteus 7 Professional. The different algorithm program for the line following robot will be developed, compiled and test. When the program successfully executes, the line following robot will be tested under several experiments and the result will come out and analyzed.

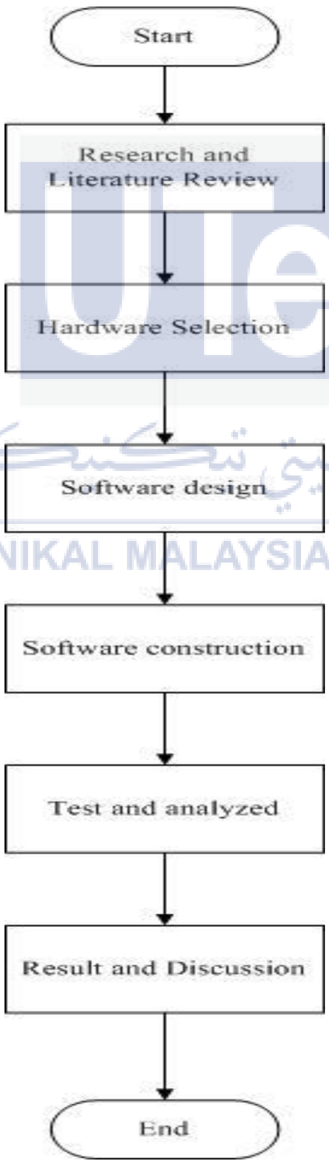


Figure 3.3: Flow chart of Overall Process of Project

3.4 Hardware Development

This section discusses all about the hardware development.The hardware plays an important role in deciding the speed and accuracy of tracking the line for the robot, which want to achieve the objective of this project. Hardware development includes the construction circuit of microcontroller, sensor array board and the design of robot platform. The line following mobile robot consist of several components which is Arduino UNO (microcontroller), gear motor, wheels, resistor, potential meter, IR sensor, ball casters and other miscellaneous component.

3.4.1 Microcontroller

The Arduino UNO is a microcontroller board which is a small integrated circuit that containing of processor, I/O peripherals and memory. The processor use for Arduino UNO is ATmega328. The total digital input/output pins is 14 and 6 of them can be used as PWM outputs. Besides that, it also consists of 6 analog inputs, a USB connection, a 16 MHz ceramic resonator, an ICSP header, a power jack, and a reset button. It contains about everything needed to support the microcontroller. The Arduino microcontroller easier to use compared to other controller. [12][13]

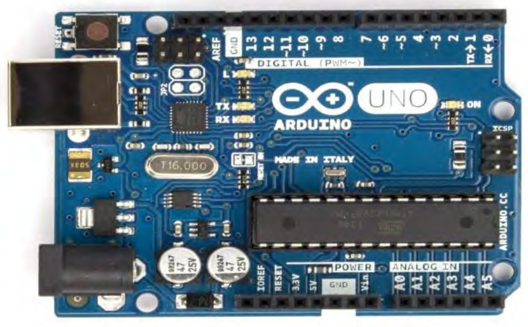


Figure 3.4: Arduino Uno

Source: http://www.arduino.cc/en/Main/ArduinoBoardUno

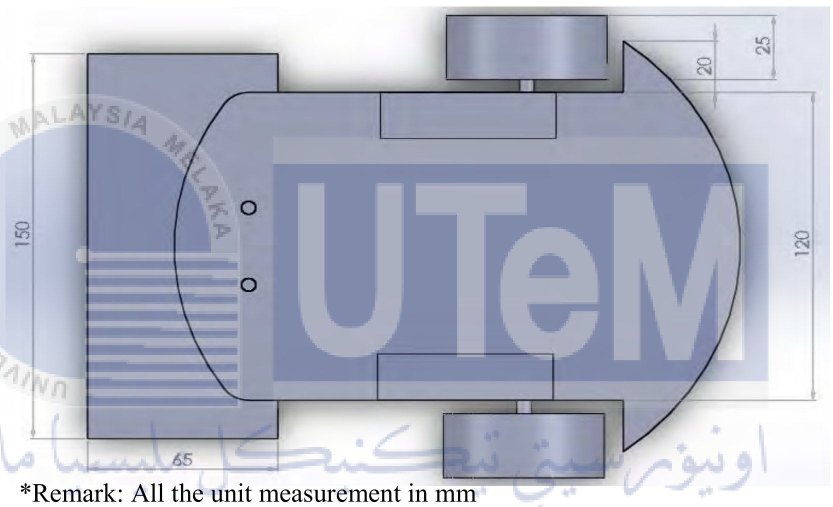
3.4.1.1 Physical Characteristic

Table 3.1: Summary of Arduino Uno

Microcontroller	ATmega328
Operating Voltage	5V
Input Voltage (recommended)	7-12V
Input Voltage (limits)	6-20V
Digital I/O Pins	14 (of which 6 provide PWM output)
Analog Input Pins	6
DC Current per I/O Pin	40 mA
DC Current for 3.3V Pin	50 mA
Flash Memory	32 KB (ATmega328) of which 0.5 KB used by bootloader
SRAM	2 KB (ATmega328)
EEPROM	1 KB (ATmega328)
Clock Speed	16 MHz
Length	68.6 mm
Width	53.4 mm
Weight	25 g
Microcontroller	ATmega328

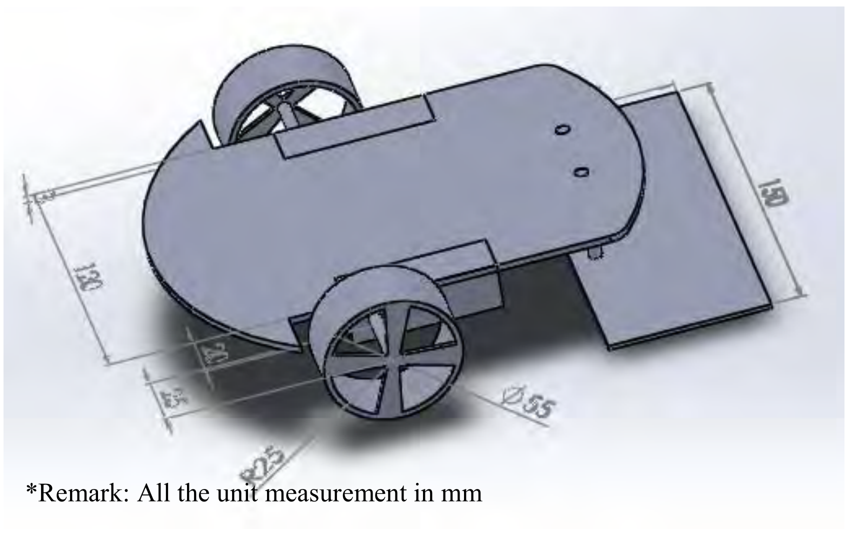
3.4.2 Robot Platform

Acrylic was chosen as the base of a line following robot due to its characteristic of light, strong and insulator to the electronic component. All of the components will be mounted on the base. The body structure is a small in design and. Two DC gear motors were used for controlling speed and steering bolt. Before building up the hardware, the sketch of the diagram was designed by using Solidwork software. The platform of the robot was small in size, which is 120mm wide and 130mm long. The diameter tires of the robot were 55mm and the rim 25mm in radius.



*Remark: All the unit measurement in mm

Figure 3.5: Top view of the mobile robot



*Remark: All the unit measurement in mm

Figure 3.6: Side view of the mobile robot

3.4.3 Sensor array board

Each of the IR sensors comprised of 2 LEDs. The black color LED is the phototransistor while the blue color LED is the IR emitter. The IR emitter it will send a continuous beam of infrared light to the ground and it will reflect back and receive by infrared detector. The infrared detector is the phototransistor which uses to active the transistor base.

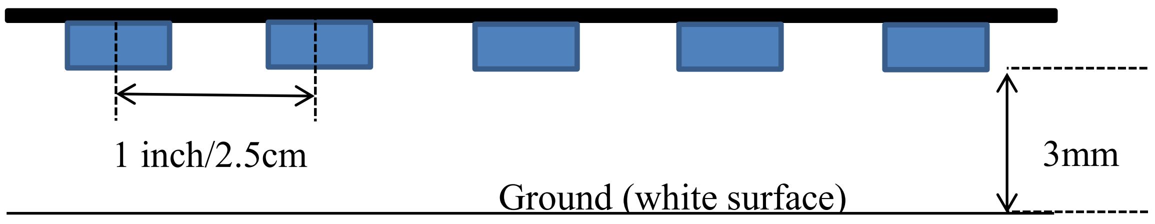


Figure 3.10: Placement of IR sensor

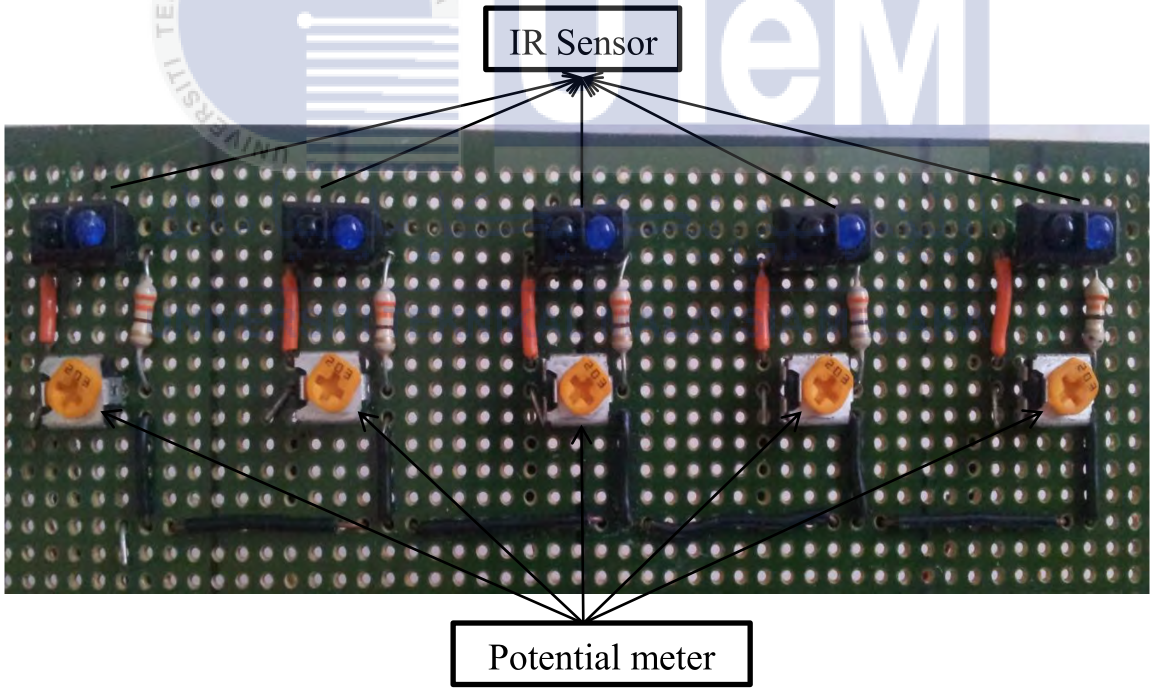


Figure 3.11: Back view of the sensor circuit board

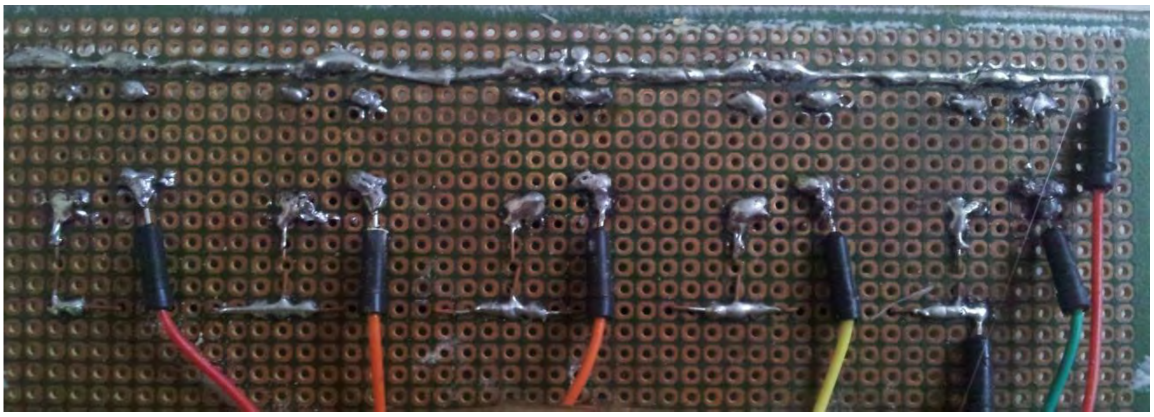


Figure 3.12: Front view of the sensor circuit board

The line following robot consists of 5 IR sensors and the sensor's gap equivalent to each other. The spaced between each sensor is 1" which based on the wide black line on a white surface. The sensor spacing allows more than 2 of the sensor to detect the line which can increase the precision and accuracy of the sensor array. The sensor values are read and send to the controller to determine the line position.

Table 3.2: The full range of values of sensor

Binary Value	Value
00001	4
00011	3
00010	2
00110	1
00100	0
01100	-1
01000	-2
11000	-3
10000	-4
00000	-5 or 5 (depending on the previous value)

The range value measured position between -5 until 5. The value 0 represents the robot sensor centered over the line mean that the robot will move in a straight line. Value of -4 means the robot is moved to the right of the line while value 4 mean the robot move to the left of the line.

3.4.4 Circuit Design

Proteus 7 Professional software was chosen to design and develop the circuit and schematic diagram. Every of the component has their own function to design the complete circuit.

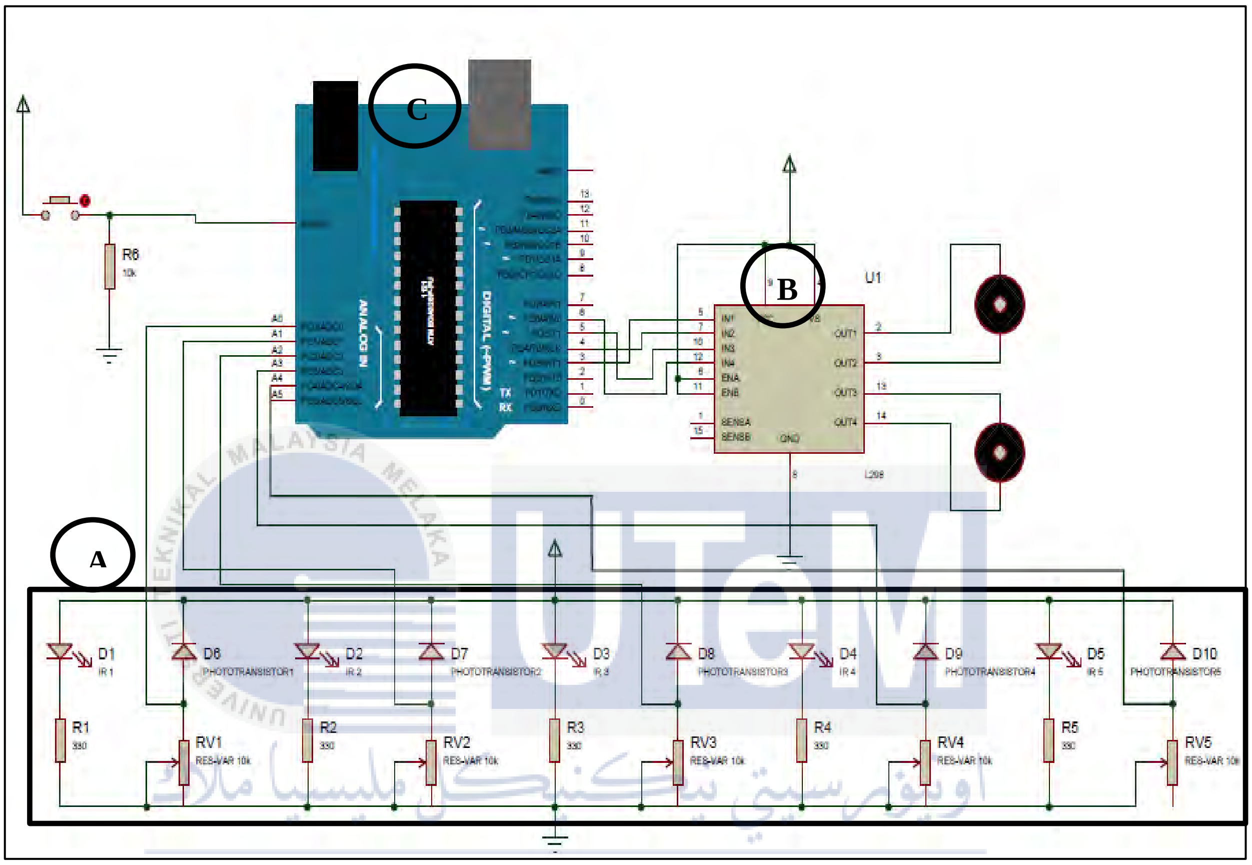


Figure 3.13: Circuit diagram

The power supply of 5V to Arduino UNO was used to power up the entire component in the system. The sensor array gets 3.3V from Arduino port which enough to supply for all sensors. The sensor array board (circle A) show in the figure 3.13 consist of 5pair of the IR sensor, the resistor and potential meter. In this case, the potential meter/variable resistor is used to calibrate the entire IR sensor to make sure the reading outcome is accuracy and consistency.

For the motor drive and Arduino circuit (refer to circle B and C in the figure), the motor drive is used to power up the voltage supply to motor and controlled the rotation of the motor. The 6V battery is connected to motor drive L298D and use to power up the motor if necessary.

3.5 Track Specification

The track has been designed to test performance of the robot. The track consists of right turn and move straight, which can be easily used to determine the performance of the robot. The robot can move in bidirectional which are clockwise and anticlockwise. The track has 2.9 m long and 2.5 cm width. The track is using black line which sticks to a white surface. The starting point is marked on the track since it is easier to take the times per one course/track. Starting point also named as an end point, which the robot starts its course at that point and also end at that point. This is mean one complete track/course.

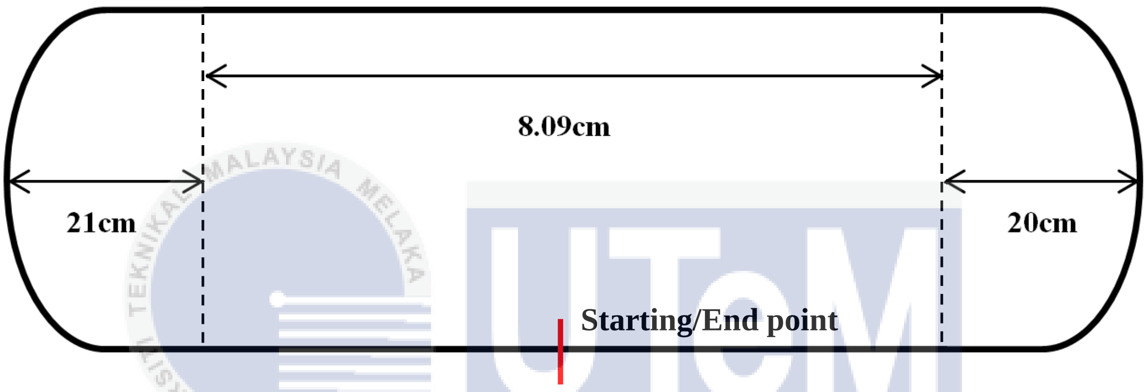


Figure 3.14: Track of experiment

3.5.1 Calculation of the track distance

The track distance is derived from the calculation of the perimeter and circumference. The length of the arc,

When the radius of the half circle is 21cm,

Arc length $= (n°)/360 \times 2\pi r$

$=180/360\times 2\pi r$

$=180/360\times 2\pi\ (21)$

$=1/2\times 2\pi\ (21)$

$=65.67 \cong 66\text{cm}$

When the radius of the half circle is 20cm,

Arc length $= (n°)/360 \times 2\pi r$

$=180/360 \times 2\pi r$

$=180/360 \times 2\pi\ (20)$

$=1/2 \times 2\pi\ (20)$

$=62.83\text{cm}$

Total length of the track $= (80.9 \times 2) + 65.97 + 62.83$

$= 290.6\text{cm}/\ 2.9\text{m}$

3.6 Tuning PID Control

Tuning of PID is the important part to build a PID control. By tuning the Kp, Ki and Kd to come out and get the best result. By referring to the characteristics of P,I and D controller Table 3.4, the optimum value could be determined.

The turning process is started by setting all the constant value to zero. The first value will be adjusted proportionally constant, Kp. The value gives an approximate value can initialize by 1 and seeing the robot performance. If robot overshoot and doesn't follow the line, reduce the value and tune till the robot smoothly follows the line. Meanwhile, if the robot cannot turn left and right, increase the value Kp until can follow a line. It was observed that the robot goes with no acceleration in a straight line. Next, value of Kd is increased until the movement is stable. After tuning the Kd, the next value of to be tuned is Ki. Ki is increased slightly until the robot shift to the center of the straight line and then accelerate, so that the robot follows the line smoothly.

Table 3.3: The characteristic of P, I and D controller [3]

CL Response	Rise Time	Overshoot	Settling Time	S-S Error
K_P	Decrease	Increase	Small Change	Decrease
K_i	Decrease	Increase	Increase	Eliminate
K_d	Small Change	Decrease	Decrease	Small Change

3.7 Basic Programming Equation for Line Following Robot

In this section,it will explain over the usage of several important equations related to the coding involved.

a) Calculate the error

$$Error = Setpoint\ value - Current\ value \tag{3.1}$$

The PID controller reads the sensor position and the position of the sensor determine on which sensor detects the black color line. Based on the sensor position, the error would be calculated. The higher the error value means the further the position of sensor away from the target.

b) Determine the adjust speed of the motor

$$Integral = integral + error \tag{3.2}$$

$$Derivative = error - last\ error \tag{3.3}$$

$$MotorSpeed = Kp * Error + Kd * (Error - LastError); \tag{3.4}$$

$$LastError = Error; \tag{3.5}$$

$$RightMotorSpeed = RightBaseSpeed + MotorSpeed; \tag{3.6}$$

$$LeftMotorSpeed = LeftBaseSpeed - MotorSpeed; \tag{3.7}$$

Integral from the equation above means the sum of error. The function of integral is to correct the previous error while the function of the derivative is to correct the future error. The motor speed will be used to calculate the left and right motor PWM. The equation above is important for writing the coding of the controller and the coding will attach at the appendix.

3.8 Experiment Setup

There are two types of experiment conduct of this project, which was tested performance (speed and accuracy) and test signal sensor for line detection. Each of the experiments would be explain briefly in this section.

3.8.1 Experiment 1: Sensitivity of sensor

The sensor is an important part which affects the accuracy performance of the robot. This experiment is used to test the sensitivity of the sensor. To get the effectiveness of the sensor, the output voltage is measured by varying the distance from the surface was conducted in this experiment. By testing the sensor, height adjustment is a method to affect the output of the sensor.

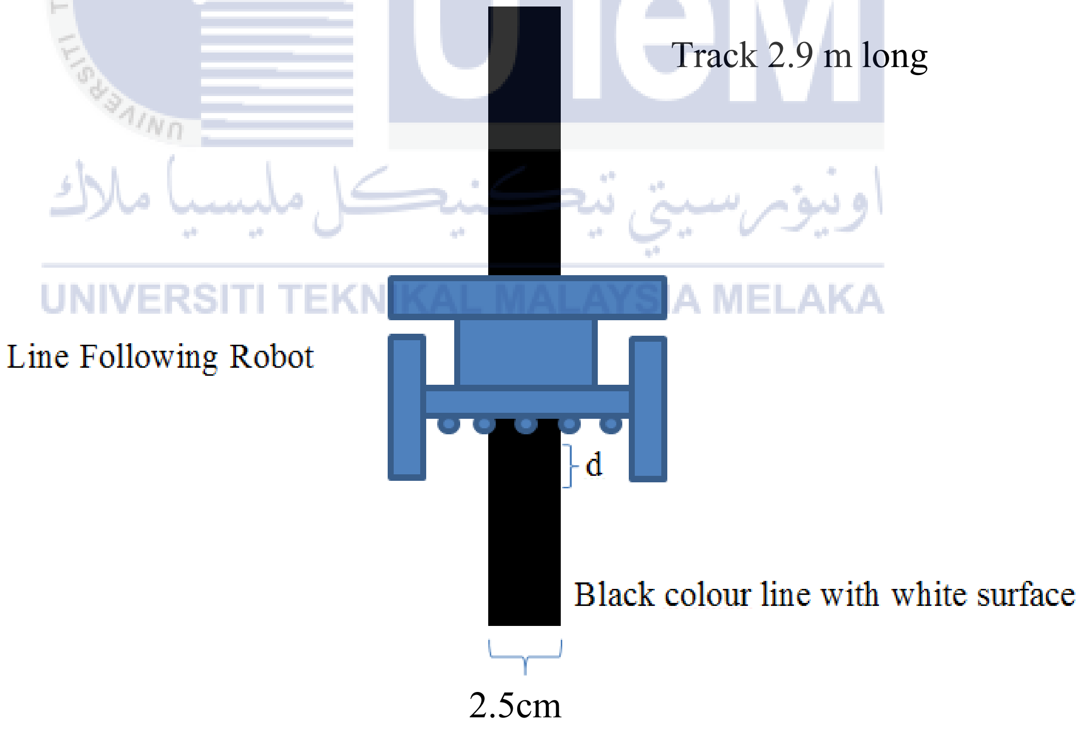


Figure 3.15: Experiment setup for test sensitivity of sensor

3.8.1.1 Circuit Design of Line Sensor

The circuit is designed and drawn by using Proteus Professional version 7.0 software. The line sensor consists of an IR transmitter and a receiver. The transmitter sends the IR rays and the receiver will receive it. As light energy hits the surface, part of the light will scatter or absorb and the rest of the energy is reflected. Different surface scatters, reflects and absorbs light in different positions. In the experiment, the line sensor tested on two different surfaces which on black and white color.

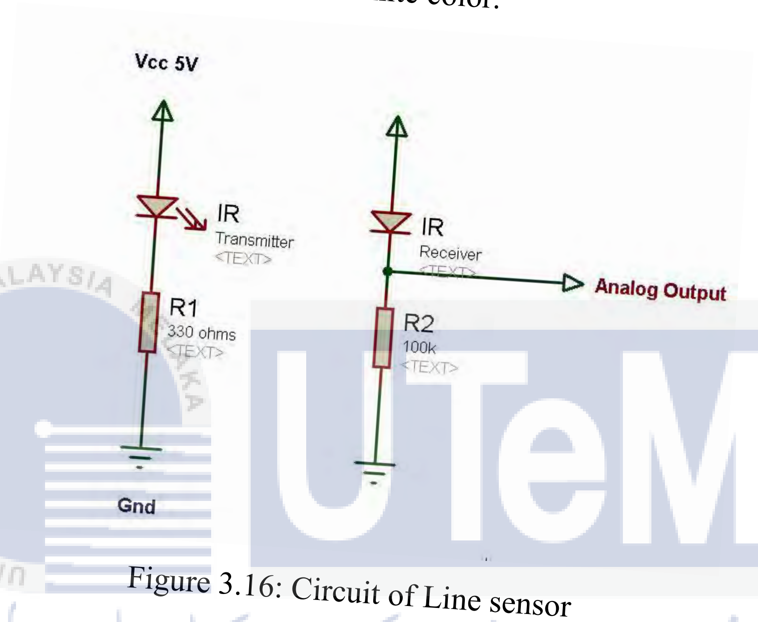


Figure 3.16: Circuit of Line sensor

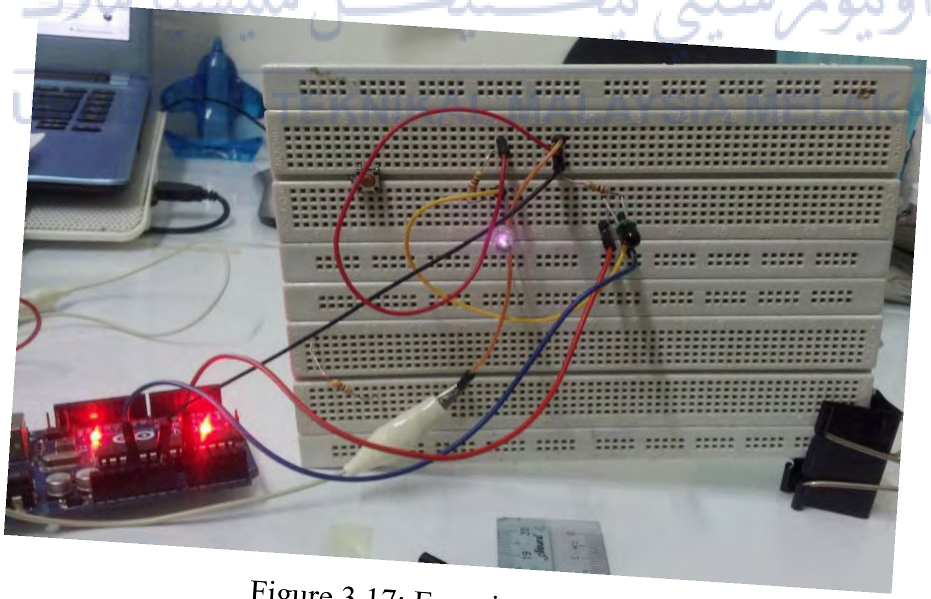


Figure 3.17: Experiment Setup

3.8.1.2 Experiment Process Flow Diagram

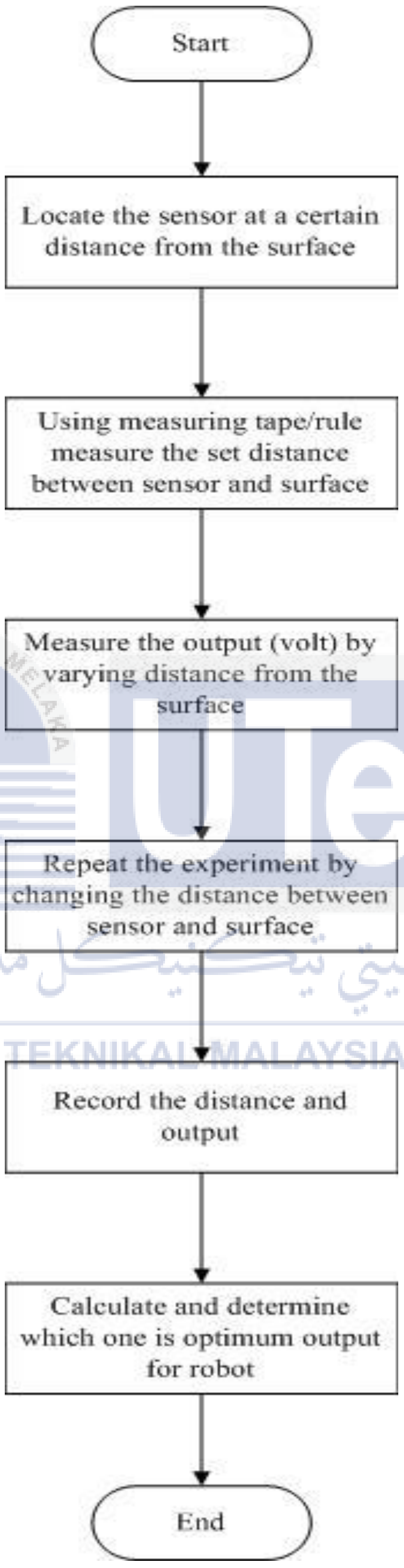

Start
Locate the sensor at a certain distance from the surface
Using measuring tape/rule measure the set distance between sensor and surface
Measure the output (volt) by varying distance from the surface
Repeat the experiment by changing the distance between sensor and surface
Record the distance and output
Calculate and determine which one is optimum output for robot
End

Figure 3.18: Flow Chart for Sensor Experiments Procedure

To get the effectiveness of the sensor, the output voltage is measured by varying distances from the surface. The sensor is needed to locate a certain distance from the surface and the distance is adjusted to obtain the optimum output. In other words, the experiments are repeated by varying the distance. The optimum output indicates that the sensor achieves better performances in terms of response and accuracy.

3.8.2 Experiment 2: Performance Analysis (Compare between PID and P Controller)

This experiment is conducted to test the performance of the line following robot. The robot speed was tuning from minimum until maximum to determine the duration of robot taken to finish a complete course and to analyse the accuracy of robot follow the track.

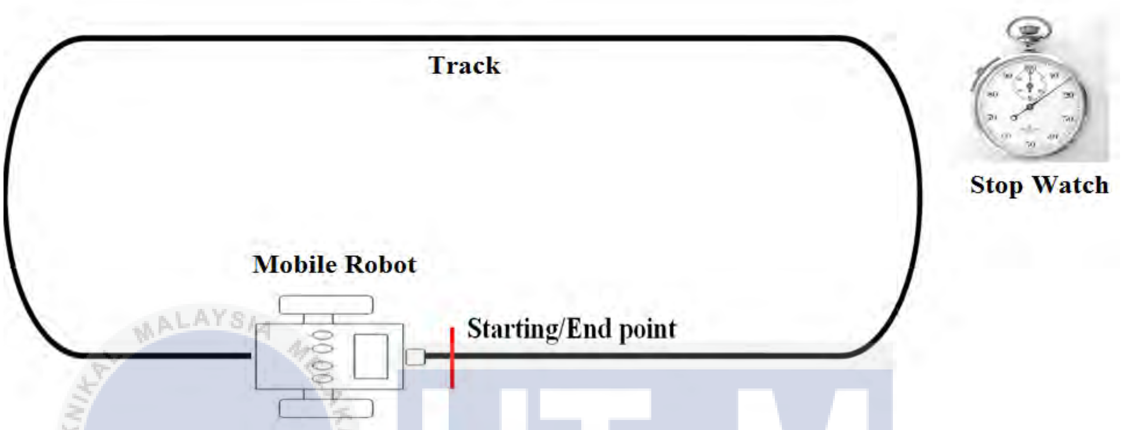


Figure 3.19: Experiment Setup

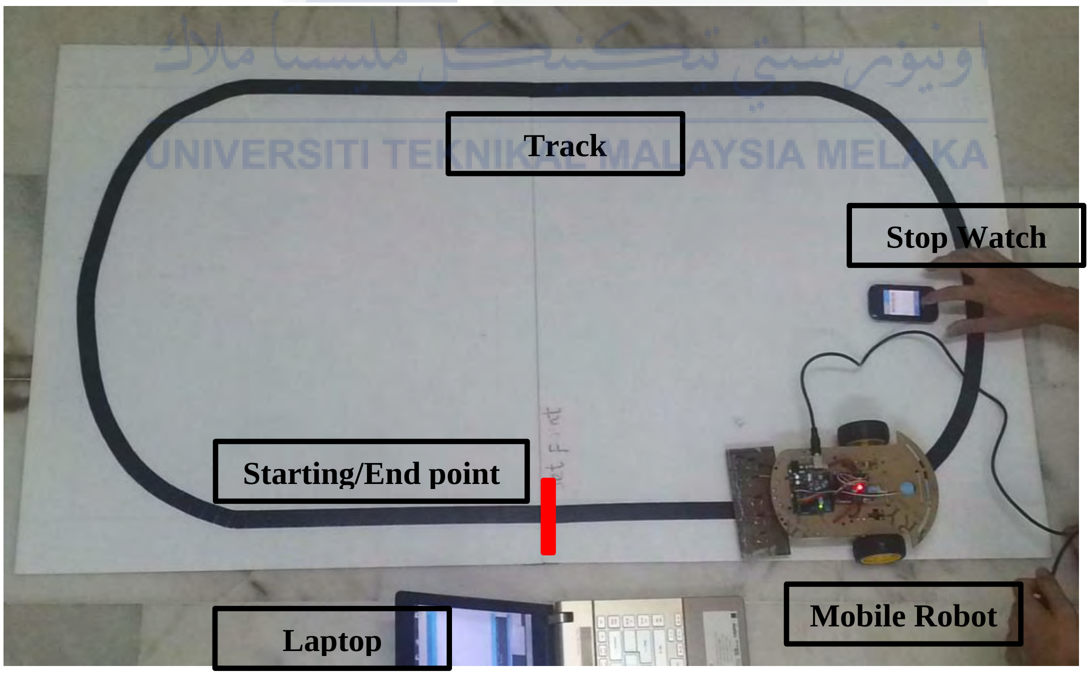


Figure 3.20: Real experiment Setup

The figures 3.20 show how to set up the experiment. The equipment used in this experiment includes a line following robot, a laptop and a stopwatch to calculate the time taken. The term of speed refers to the distance or how far an object travels in a given time interval. The speed of the robot is tested for minimum to maximum speed by using PWM. By using this experiment, it can determine at which speed the robot can navigate efficiency and accuracy. If the robot can navigate efficiency and accuracy, means that it can track the line with high speed and without leaving the line and correct the error if the robot leaves the line.

3.8.2.1 Experiment Process Flow Diagram

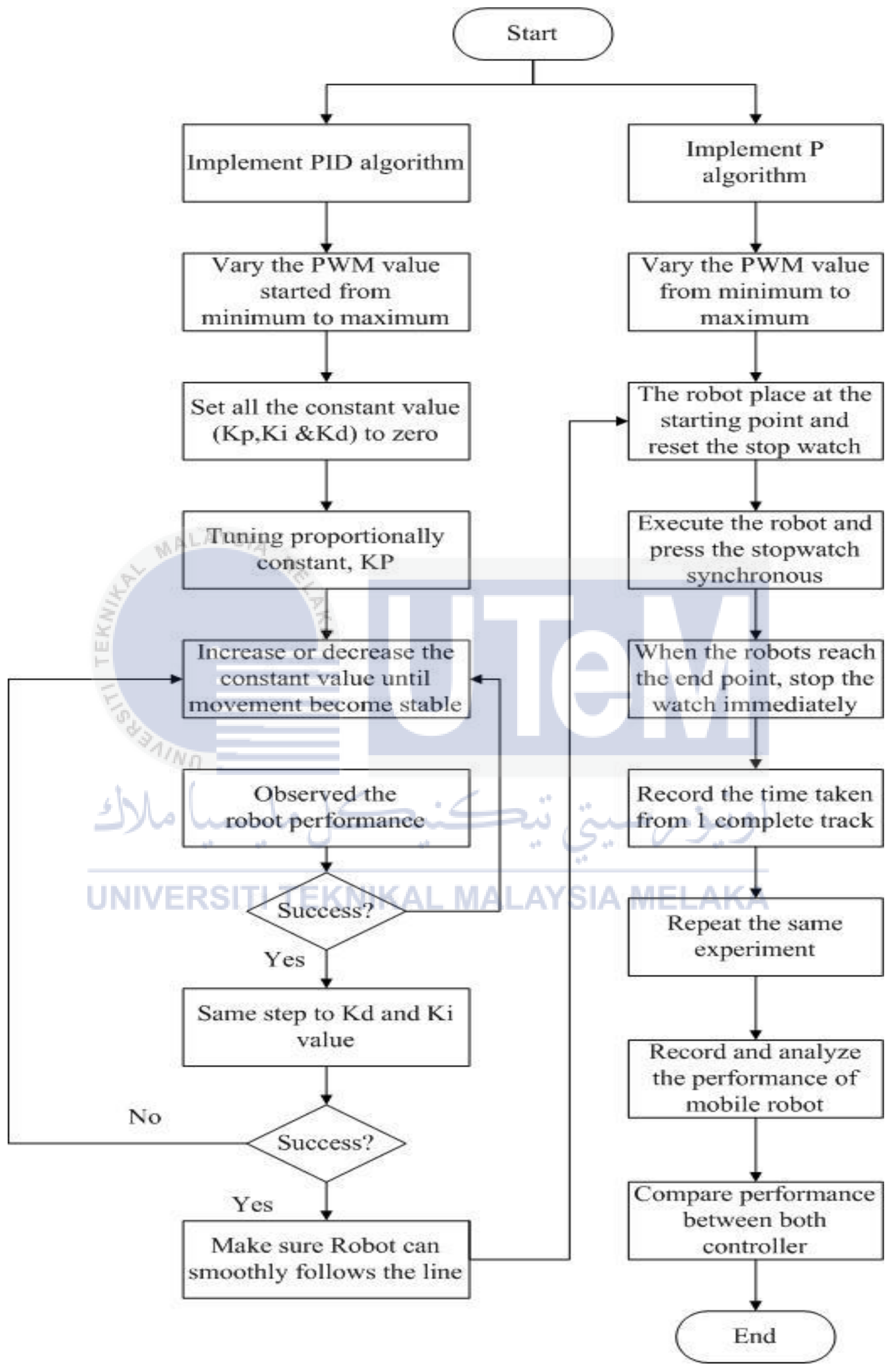

Start
Implement PID algorithm
Vary the PWM value started from minimum to maximum
Set all the constant value (Kp,Ki &Kd) to zero
Tuning proportionally constant, KP
Increase or decrease the constant value until movement become stable
Observed the robot performance
Success?
Yes
Same step to Kd and Ki value
No
Success?
Yes
Make sure Robot can smoothly follows the line
Implement P algorithm
Vary the PWM value from minimum to maximum
The robot place at the starting point and reset the stop watch
Execute the robot and press the stopwatch synchronous
When the robots reach the end point, stop the watch immediately
Record the time taken from 1 complete track
Repeat the same experiment
Record and analyze the performance of mobile robot
Compare performance between both controller
End

Figure 3.21: Flowchart of Performance Analysis Experiment Procedure

The coding of two algorithms (P and PID) has been done to test the performance. The experiment needs to divide into 2 parts which is for PID controller and P controller. Although the controller was different, but the experiment procedure was same. Firstly, the track has white surface and a black line lies in its central park. A starting point and end point are set on the track to observe and analyze the performance of the robot. By using PWM the speed of the line follower robot is varying from minimum speed to maximum speed. The line following robot placed on the track to complete the course/track and the time is taken by using a stopwatch. The start button is pressed when the robot starts moving from the starting point and then stop button is pressed when the robot goes over the endpoint. The time taken for one course is taken and calculated. The error and system data also recorded. The steps are repeated over 3 laps. Lastly, repeated the same experiment by change another controller. From all the data get, sort out all the data into different categories and analysis all the information get. The performance analysis due to the movement of robot in different speed. Here is the equation had been used to come out the result.

Speed performance equation:

i. $$Velocity\left(\frac{m}{s}\right) = \frac{Distance}{Time\ taken\ to\ compete\ 1\ lap/track} \quad (3.8)$$

ii. $$Average\ velocity\left(\frac{m}{s}\right) = \frac{V1+V2+V3}{3} \quad (3.9)$$

iii. $$Average\ time\ taken\ to\ complete\ the\ track\ (s) = \frac{t1+t2+t3}{3} \quad (3.10)$$

3.9 Precaution on replicability and reliability issues

Pulse Width Modulation (PWM) technique is used where its signal is generated in microcontroller and the PWM signal will send to the motor driver to vary the voltage supply to motor to acquire desired speed. Through the experiment, Pulse Width Modulation (PWM) signal used to control motor speed at desired speed.

The cable used to connect between microcontroller and laptop also will be an issue that needs to take care. The cable is used to easy monitoring and transfer the serial print result and data between the computer and the controller. Apart from this, the cable also as transmission of power supply to power up the whole system and due to the reason, the cable should be long in length. When the robot state operates, must be always pay attention and make sure the cable not obstruction and trip over the mobile robot.

Fully charged the battery/power bank before start the experiment. It will lose power during conduct the experiment and it will affect the performance test of mobile robots. So, by using rechargeable battery has been suggested.

Since the robot is intended use for lab test, mean that must put on relatively smooth and flat surface. Avoid running the mobile robot on rough surface or might scrape or damage the underside of mobile robots. Besides that, always check the tire pattern, once the tires worn, it must be replaced immediately, so it doesn't affect the performance of the robot.

CHAPTER 4

RESULT AND DISCUSSION

4.1 Introduction

In this section, the results have generated from the experiment. From the experiment, the data have been collected and the analysis and discussion is made to show the significance of the result. The results are divided into four main parts to analyze all the results and data which are the result to test the sensitivity of sensor, performance analysis of P controller/ without PID controller, performance analysis of PID controller as well as the comparison of performance between P and PID controller.

4.2 Result of experiment to test the sensitivity of the sensor

The experiment is conducted to determine the sensitivity of the infrared sensor to obtain the effectiveness and performance of the sensor, detection surface IR sensor plays an important role and it will affect the output of the sensor.

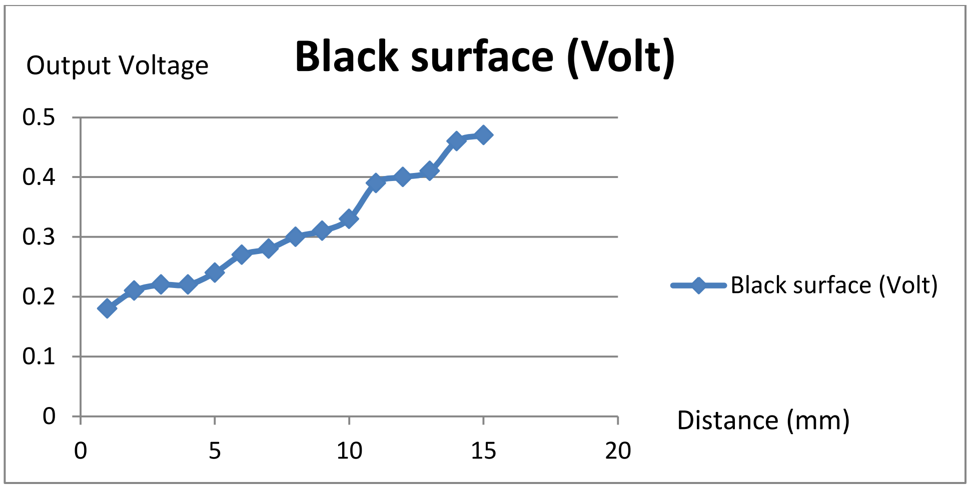


Figure 4.1: Output voltages vs. Distance on black surface

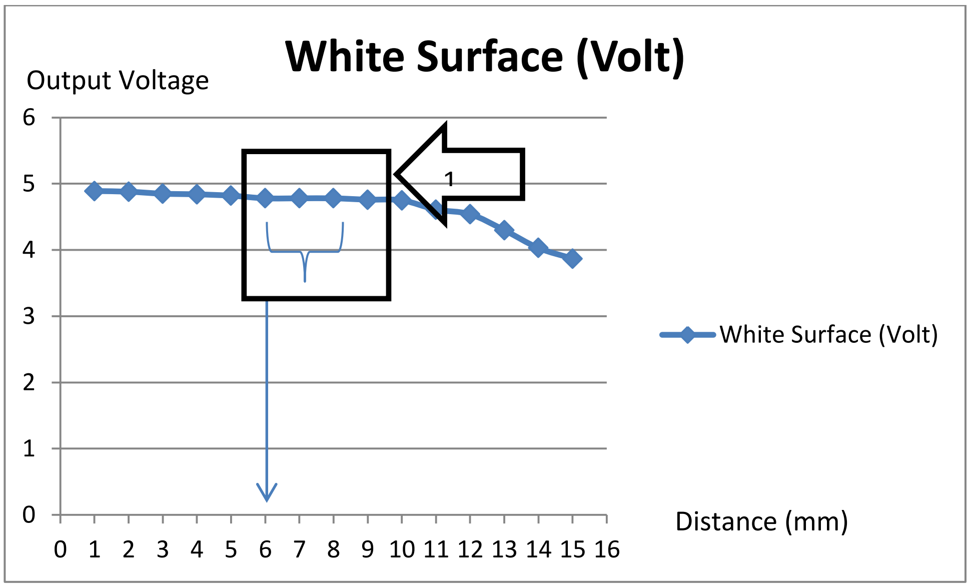


Figure 4.2: Output voltages vs. Distance on White surface

In order to obtain the effectiveness and sensitivity of the sensor, measure the output voltage by varying distances from the surface which show in the table in appendix A. From the figure 4.1, the output voltage increases with distance in black surface while for the figure 4.2, it shows that the white surface induces a great diminishing effect on the output voltage.

When the light falls on white surface the resistance of the sensor decreases while at the black surface it will result in higher resistance. Different surface will have different reflection intensity. A black surface will absorb more light than a white surface while white surface will reflect the light.

The figure 4.2 shows that when the sensor at the 0.6cm distance from the white surface, the output voltage remains constant until 0.8cm. This means that the sensor needs to locate 0.6cm from the surface to obtain optimum output.

As a result, the distance between the surface and the sensor will affect the performance in terms of response and accuracy of the sensor. The output of voltage value also will decrease as the further the distance of the object from the infrared sensor. IR sensor is more suitable to differentiate the white surface and black surface. Hence, the track must uses black line which lies on a white surface.

4.3 Comparison between the performance of PID and without PID Controller

Based on the figure 4.3, 4.4 and 4.5 shows the different type of performance analysis carry out by conduct the experiment using P and PID controller.

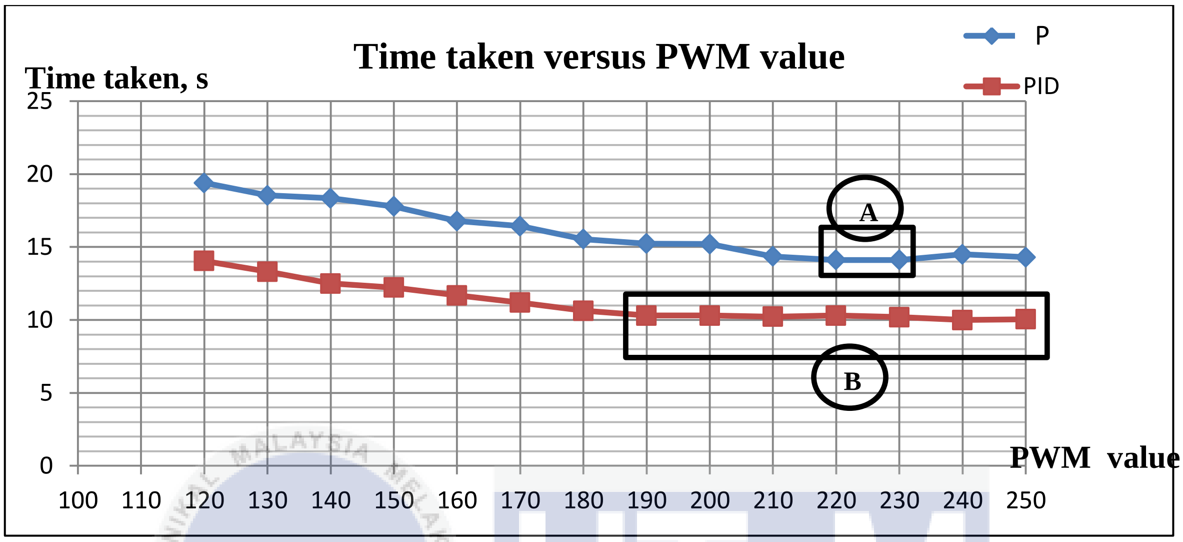


Figure 4.3: Time taken versus PWM value

From the figure 4.3, it demonstratively shows that the time taken for implementation of PID controller shorter compare to P controller. The graph shows a steady decrease between these two lines. The shortest time taken to achieve by the implement of PID taken 10s to complete the track compared to P controller needed 14.11s. The total differences between PID controller and P controller are 4.11s. In contrast, the longest time taken for PID controller was 14.05s at 120 values of PWM compare to 19.4s to without PID controller.

For the P controller, refer to the blue color line as shown in the figure 4.3. The shortest time taken to complete the track is 14.11s during 220 value of PWM. Although 220 values of PWM not the maximum value (max value of PWM = 255). By using a higher PWM value, it does not mean that it will take the shortest time to complete the track because when the PWM value is higher, the line following robot will move faster and more error will make. The robot takes a long time in terms of overshoot when taking a turn even move in a straight line. However the ideal PWM value in this case is between 220 and 230 (referring to circle A in the figure). The time taken of both value is 14.11s and 14.12s which the difference only between 0.01s. As a result, the PWM between 220 and 230 was most suited to the robot behavior and characteristics.

On the other hand, the red color line in the figure which is the PID controller. Refer back to the figure 4.3, the taken becomes constant and stable when the value of PWM reached 190 (referring to circle B in the figure). The higher the PWM value means more of the voltage supply to the motor and the speed of the motor will be increased. In this case, the time taken to complete the track became stable and after 190 value of PWM. This is because starting from 190 PWM is the ultimate limit that the robot characteristic and behavior can reach.

Lastly, from the comparison between using PID controller and without PID controller, it clearly shows that PID controller used less in time to complete the track because it will improve the movement of robot automatically so that the error can also be reduced at the same time. Less of the error means that less overshoot and it consequently save the time during overshoot period. This is why the reason of PID controller faster than without PID controller.

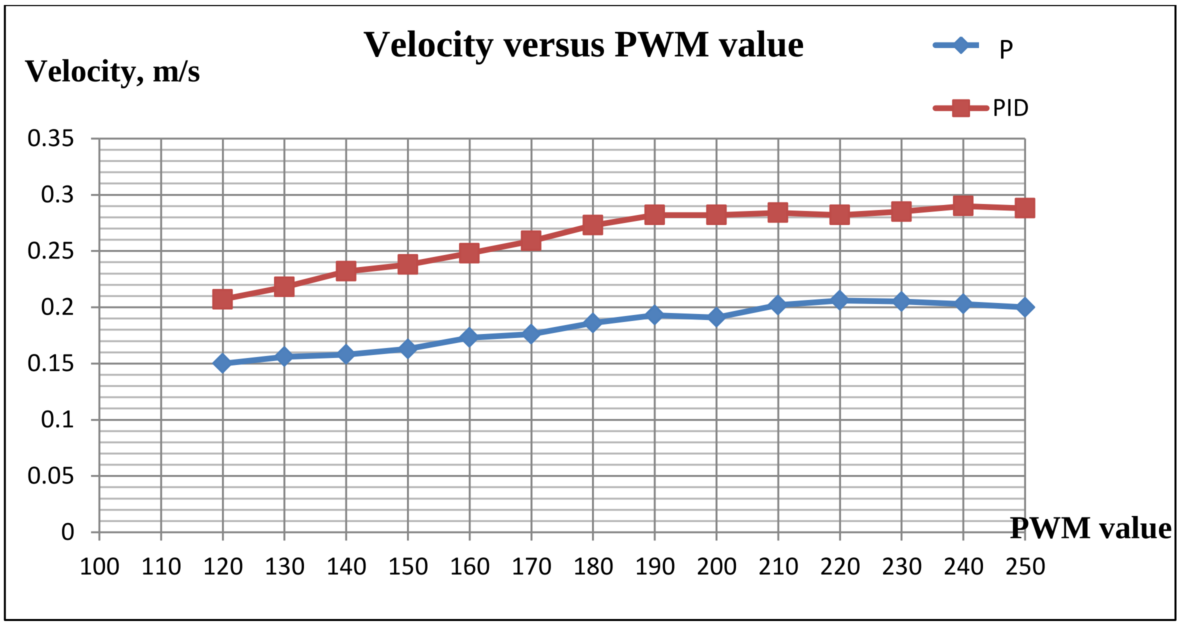


Figure 4.4: Velocity versus PWM value

Based on the figure 4.4, it denotes that the velocity with PID controller higher than P controller. The maximum velocity can be achieved when implement of PID controller, which is 0.290 m/s. On the other hand, for the P controller, the maximum velocity is 0.206 m/s. The different of maximum velocity can be attained between both of the controller are 0.084m/s.

For the P controller, the velocity is increased from 120 values of PWM until 220 value of PWM. After over 220 values of PWM, the velocity gradually reduces until 250 value of PWM. The lowest PWM used in this experiment is 120 because if the value of PWM less than 120, the line following robot is unable to move due to insufficiency of the voltage supply. In this case, 220 values of PWM were the most ideal and suitable for the robot characteristic and the track because it can achieve the highest velocity comparable with other values of PWM. It means that the robot makes a less error when follow the black line and less overshoot when taking a corner. Fundamentally, if the PWM value increases, the velocity will also increase. However, in this case the velocity decreases after 220 value of PWM because when velocity becomes higher, it will produce more errors when taking the corner even the straight line. The greater amount of error means the controller need more time to calculate and computing to return the target (black line).

Over the next, as can be seen from the graph the red line which is the PID controller, it shows the higher the PWM value, the higher the velocity of the mobile robot. The fluctuating value of the velocity became stable after reaching 190 values of PWM. If the value of PWM lower than 120, the robot unable to move due to not enough voltage to make the motor to rotate. In this case, the time taken to complete the track became stable and after 190 PWM. This is because starting from 190 PWM is the ultimate limit that the robot characteristic and behavior can reach.

By implementing the PID controller, the line following robot can follow the line and keep moving in the center thanks to its automatic calculate and minimize the error detected. When the mobile robot move in the straight line, the mobile robot will gradually stabilized move straight and able to follow the line quickly and efficiently. That is why the reason mobile robot with PID controller faster than without PID controller. Mobile robot without PID controller make more error, therefore the controller requires more time to calculate and execute the error.

Error Versus Time (When PWM = 120)

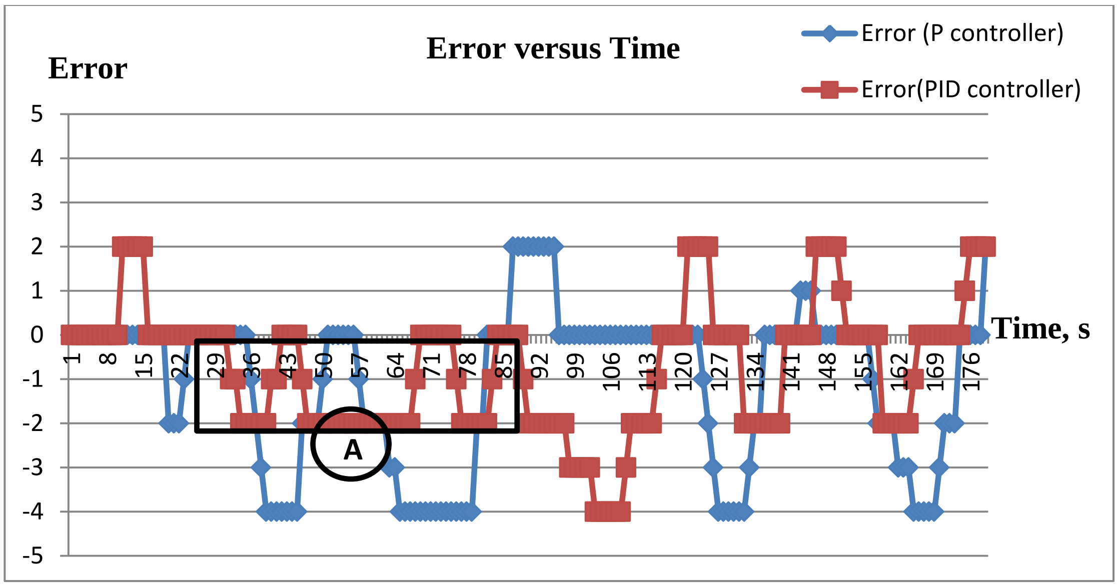


Figure 4.5: Error versus Time

Refer to the figure 4.5, it shows the error versus time by using two different types of controller during 120 value of PWM. The blue color line shows the error without PID controller while the red color was implementing of PID controller. As can be seen from the graph, both the lines went up and down widely. To implement of PID controller, the oscillation of error is smaller than without PID controller. It clearly shows at the point A at figure 4.5. For P controller, the error is in a large wavy contrast with implementing PID controller. The error of both controllers moves as a sinusoidal wave. The maximum error is 2 while the minimum error is -4. The peak to peak value between the highest and lowest is 6.

It can be seen from the figure 4.5, negative error is more than positive error and this is due to the robot is moving in a clockwise direction. When the mobile robot turns in clockwise, the left sensor will majority detected the black line and sends back to the controller to execute and compute the data.

Error in this experiment stands for the difference between the current position and the target. The error can be zero, positive and negative and the possible measurable range are between 5 until -5. The error was depending on the position of the sensor. In order to measure the error, 5 IR sensors are used to take the position input of the robot. (Refer back to the Chapter 3).

Zero value of PWM represents that the center sensor of the robot was detected and the robot follows the track move in a forward direction. The positive side of the graph means the right side sensor was detected and the robot turns in left direction. In contrast, the negative side of the graph means that the left side sensor was detected and the robot turns in the right direction. The average error of PID controller is -0.7333 while P controller is -1.0556. The desired target set was zero value mean it detect the center sensor of mobile robots. The error was PID controller was more close to the desired value compare to P controller. In fact, the result proved that PID controller is more accurate than P controller.

Based on this case, the error is standing for the difference between the current position and the target. By using PID controller, it will automatically calculate and minimize the error detected. The response and reaction of PID controller are more sensitive to overcome the detected error and back to the target.

4.4 Performances Analysis of PID Controller

Based on the figure 4.6, 4.7 and 4.8 shows the other performance analysis is carried out by conduct the experiment using PID controller.

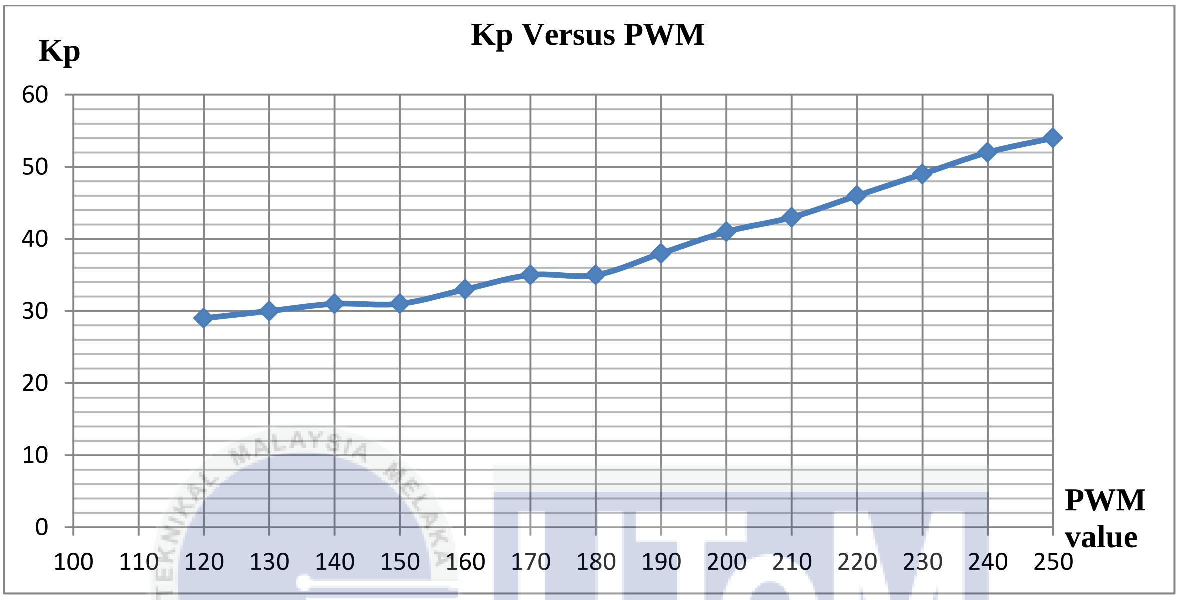


Figure 4.6: Kp Versus PWM

Figure 4.6 shows the positive relationship between Kp and PWM. When the PWM value increases, the Kp value also increases. The PWM use in the experiment is from the 120 and increase by 10 to 250 values of PWM. The maximum Kp used in the experiment is 54 during 250 value of PWM and the minimum value of Kp is 29 at 120 values of PWM.

The Kp times error is equal to proportional which is used to determine how far the robot from the line. For an example, the robot is moved to the right, to the little right or to the extreme right. Kp is the fundamental term to be tuned first and calculate the other like Ki and Kd. The higher the PWM means that the robot will move faster and it will procedure more error. Kd is to respond of the error the higher the error make greater value of Kd to turning as shown in the figure 4.6.

Tuning Kp required manual trial and error to get the desired behavior which can fix to the robot. The goal to tuning the Kp is to get the robot follow the line first even the robot move wobbly. Increasing the Kp when the robot cannot navigate a turn or sluggish and decreasing the Kp when the Kp value loses or move out the track.

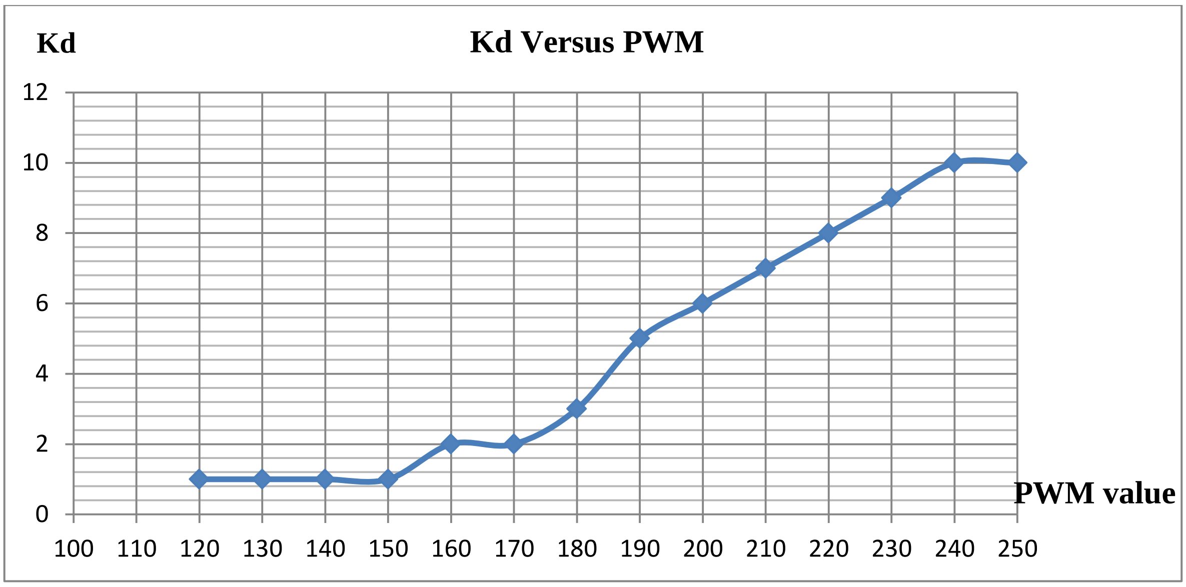


Figure 4.7: Kd Versus PWM

The figure 4.7 shows the Kd increase as the value of PWM increased. The Kd maintain stable from the early starting which is 120 to 150 of value of PWM and increase 1 value of Kd at 160 PWM and stable again between 160 and 170.After 170 value of PWM, it increases sharply until 240 value of PWM and lastly maintain stable at 250 values of PWM.

The robot may keep wobbly or oscillating after tuning Kp and Ki. The derivative is to reduce the wobbling effect over time. The faster the robot side-to-side means the higher the derivative value is. The higher the PWM value means that the faster the speed. When the speed increase, the error also increase, so a great value of Kd is needed to overcome the problem.

Left versus Right motor speed (When PWM value = 120, implement of PID)

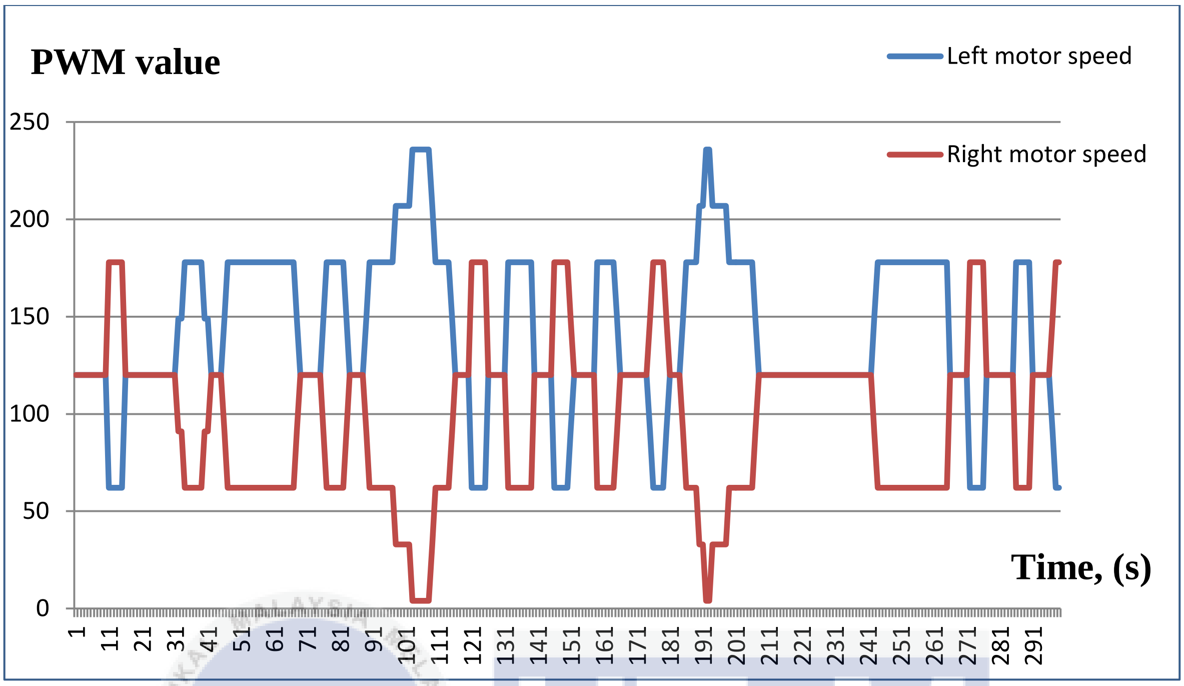


Figure 4.8: Left versus Right motor speed

Refer from the figure 4.8, the graph is shown a large fluctuation between left and right motor speed. The value of PWM equal to 120 was selected because it can clearly show that inverse relationship between left and right motor PWM. When both of the motors value of PWM are 120, it indicates that the robot moving forward direction. If one side of the motor's PWM is high while another side is low it represent that the robot was taking a turn. Refer to the figure 4.8 during 110s, the robot taking an extreme sharp turn, so one side of the motor will turn in the full value of PWM while another side motor almost stops to let the mobile robot to taking a turn. The changes of PWM depend on the sensor array position error. If the volatility changing of error became larger, it means that the value of PWM fluctuation between left and right motor also increases. As a result, both of the motor coordinates well with each other so that the line following robot can keep moving in the line without move out away the track. When turning a corner, both of the motor speed will adjust it by increase one side and decrease other side to prevent the robot move out from the track.

CHAPTER 5

CONCLUSION AND FUTURE WORK

5.1 Conclusion

As a conclusion, the objectives of this project achieve. Based on the objective the project, two of the experiments is carried out to determine the performance analysis which emphasis on speed and accuracy test. The first experiment is to test the sensitivity of the IR sensors. In order to get the effectiveness and sensitivity of the sensor, the output voltage is measured by varying distances from the surface. The output voltage value is successfully collected and the result shows that the distance between the surface and sensor will affect the performance in terms of response and accuracy of the sensor. The result of experiments shows that sensor at 0.6cm until 0.8cm the output voltage remains constant at the white surface. This means that the sensor need locate 0.6cm from the surface to obtain optimum output. The output voltage value decreases as the object detected is farther away from the IR sensor. IR sensor is more suitable to differentiate the white surface from the black surface. Hence, the track must use black line while the surface is in white colored.

For the second experiment, the speed and accuracy become a major concern on this line following robot. Before using the PID controller, the problem that faced was the robot did not follow the line smoothly and keep oscillation. It wasn’t efficiently and need occupy more time to following the line, so by implementing the PID algorithm it can improve the problems faced. There are two of the algorithms used to compare in chapter 4 which are P controller and PID controller to analysis the performance. Refer to figure 4.3, 4.4 and 4.5, the result had shown that the by doing the comparison, PID controller much faster and accurately compare to P controller. The shortest time taken to achieve by the implement of PID taken 10s to complete the track compared to P controller needed 14.11s. On the other hand, The maximum velocity can be achieved by implementing of PID controller is 0.290 m/s faster than 0.206 m/s of P controller. From the result, it clearly shows that by implementing PID controller, the performance of speed much faster than P controller. The PID controller reduces the overshooting of the robot while trying to return to the line and this is the reason why times taken for PID faster than P controller. The line following robot also been developed in order to investigate the accuracy performance within slow speed and high speed. As is shown illustrated by the figure 4.5, the average error detected by PID controller is -0.7333 while for P controller is -1.0556. The desired target set was zero value mean it detect the center sensor of mobile robots. The result shows that the PID controller closer to the desired value compare to P controller. In fact, it proved that PID controller is more accurate than P controller.

While implementing PID, the lesson learned from the experiment was tuning and setting up the constant value of PID. It makes a better understanding of the impact of the value of P, I and D toward the line following robot behavior. First, all of the value as 0 and then made the changes by either increasing or decreasing the value, and seeing the changes in the characteristic and behavior of the robot. The constants PID value had to be changed many times before arrived at the perfect values. After setting up P constant, follow by setting the value of I and D Constant. By observation and result implementing the proportional part, the wobbling of the robot was lesser than before, but it was still a big issue need to be overcome. After implementing the proportional part, tuning the integral and derivative value from the coding, and the robot was now following the line much more efficiently with very less wobbling.

5.2 Recommendation for future research

In the future, the line following robot is expect to upgrade the hardware and software to a more advance level to improve performance which faster and accuracy of the robot. The motor of the robot can change to high performance and high RPM DC motor which can run at high speed. Besides that, the IR sensor can replace by Polulu QTR-8RC Reflectance Sensor Array which has 8 IR LED/phototransistor pairs to making it a great detector for a line-following robot. The sensor array has a greater reflection and sensitivity, which can perform well in PID line following robot.

For the software part, instead of manual tuning method to turn the PID, using modelling of Matlab also can be regarded as a best method. Manual tuning methods for PID are time-consuming and iterative, and if used on hardware to turn, it can cause damage to the component. By using Matlab modelling, it's able to automatically tune PID controllers to achieve the optimal system design.

REFERENCES

[1] World Health Organization, On The Road, The driver,vechicles and accident, 2011

[2] S. Bennett, Development of the PID Controller, IEEE control system, 1939.

[3] Norman S.Nise, Control Systems Engineering,six edition, 2011

[4] B. C. Robot, PID Line Following Robot, pp. 1–19, 2014.

[5] Enigmerald, PID Tutorials for Line Following , [online]. Available at: http://letsmakerobots.com/content/pid-tutorials-line-following [accessed 16 January 2014]

[6] M. G. S. Shah, a. B. Elmi, a. Nazir, and S. Shukri, Photoelectric Sensor Based Intelligent Track Racing Car, Procedia Eng., vol. 41, no. Iris, pp. 588–592, 2012.

[7] D. Sivaraj, K. R. Radhakrishnan, and T. J. Krishanth, "Design of Automatic Steering Control and Adaptive Cruise Control of Smart Car PSG College of Technology PSG College of Technology," no. Icvci, pp. 12–17, 2011.

[8] D. Sivaraj, V. Rajasekar, P. B. Sankarganesh, and G. Manikandan, Implementation of AVCS using Kalman Filter and PID Controller in Autonomous Self Guided Vehicle, vol. 27, no. 2, pp. 1–8, 2015.

[9] O. L. Casanova, M. Iaeng, F. Alfissima, and F. Y. Machaca, Robot Position Tracking Using Kalman Filter, vol. II, pp. 2–6, 2008.

[10] P. K. Padhy and S. Majhi, "Robot Path," pp. 0–3, 2006.

[11] X. Li, S. Jia, J. Fan, L. Gao, and B. Guo, Autonomous Mobile Robot Guidance Based on Ground Line Mark, pp. 1091–1095, 2012.

[12] Arduino,Arduino Robot Overview [online], Available at: http://arduino.cc/en/Main/Robot [accessed 25 October 2014]

[13] Arduino,Arduino Robot, Manual Book. pp. 2-3, 2014

[14] Volvo Car Group, "Volvo Car Group tests road magnets for accurate positioning of self-driving cars", Retrived March 11, 2014

[15] M. Z. Baharuddin, I. Z. Abidin, and S. S. K. Mohideen, Analysis of Line Sensor Configuration for the Advanced Line Follower Robot, pp. 1–12, 2006.

[16] P. S. P. Rasal, Development of Intelligent Line Follower's Robot, vol. 7, pp. 15–17, 2013.

[17] J. Su, C. Lee, H. Huang, S. Chuang, and C. Lin, An intelligent line-following robot project for introductory robot courses, vol. 8, no. 4, pp. 455–461, 2010.

[18] A. S. Nath and T. Malik, IMPLEMENTATION OF PID CONTROL TO REDUCE WOBBLING IN A LINE FOLLOWING ROBOT, pp. 531–535, 2013.

[19] New Straits Times, Makaysia ranked 20th in road deaths, July 9, 2014

Appendix A: Experiment 1

Table: Output Voltage Varying Distance from the Surface

Distance (mm)	Black surface (Volt)	White Surface (Volt)
1	0.18	4.89
2	0.21	4.88
3	0.22	4.85
4	0.22	4.84
5	0.24	4.82
6	0.27	4.78
7	0.28	4.78
8	0.30	4,78
9	0.31	4.76
10	0.33	4.75
11	0.39	4.61
12	0.40	4.54
13	0.41	4.30
14	0.46	4.03
15	0.47	3.87

Appendix B: Result Without PID Controller / P Controller

Table: Time taken versus PWM value

PWM value	Time taken for 1 lap, s			Average time taken, s
	T1	T2	T3	
120	19.35	19.39	19.45	19.40
130	19.17	18.19	18.28	18.54
140	18.57	18.13	18.33	18.34
150	17.54	18.62	17.19	17.78
160	16.85	16.74	16.74	16.78
170	16.56	16.66	16.08	16.43
180	15.39	15.40	15.85	15.54
190	15.19	15.01	14.94	15.24
200	15.52	15.39	14.69	15.20
210	14.36	14.26	14.45	14.36
220	13.90	14.44	13.98	14.11
230	13.95	14.56	13.86	14.12
240	13.97	14.28	14.65	14.30
250	14.82	14.48	14.19	14.50

Table: Velocity versus PWM value

PWM value	Velocity (m/s)			Average Velocity
	V1	V2	V3	
120	0.150	0.150	0.149	0.150
130	0.151	0.159	0.159	0.156
140	0.156	0.160	0.158	0.158
150	0.165	0.156	0.169	0.163
160	0.172	0.173	0.173	0.173
170	0.175	0.174	0.180	0.176
180	0.188	0.188	0.183	0.186
190	0.191	0.193	0.194	0.193
200	0.187	0.188	0.197	0.191
210	0.202	0.203	0.201	0.202
220	0.209	0.201	0.207	0.206
230	0.208	0.199	0.209	0.205
240	0.208	0.203	0.198	0.203
250	0.196	0.200	0.204	0.200

Appendix C: Result Impementing of PID controller

Table: Time taken versus PWM value

PWM	Time taken for 1 lap, s			Average time taken, s
	T1	T2	T3	
120	13.84	13.89	14.42	14.05
130	13.05	13.8	13.07	13.31
140	12.53	12.61	12.37	12.5
150	11.9	12.05	12.73	12.23
160	12.3	11.49	11.29	11.69
170	10.93	11.33	11.31	11.19
180	10.99	10.5	10.41	10.63
190	10.45	10.09	10.37	10.3
200	10.32	10.97	9.65	10.31
210	9.79	10.23	10.66	10.23
220	10.58	10.67	9.64	10.3
230	10.11	9.92	10.54	10.19
240	10.28	10.04	9.69	10
250	9.85	10.2	10.09	10.05

Table: Velocity versus PWM value

PWM value	Time taken for 1 lap, s			Average Velocity
	V1	V2	V3	
120	0.21	0.209	0.201	0.207
130	0.222	0.21	0.222	0.218
140	0.231	0.23	0.234	0.232
150	0.244	0.241	0.228	0.238
160	0.236	0.252	0.257	0.248
170	0.265	0.256	0.256	0.259
180	0.264	0.276	0.279	0.273
190	0.278	0.287	0.28	0.282
200	0.281	0.264	0.301	0.282
210	0.296	0.283	0.272	0.284
220	0.274	0.272	0.301	0.282
230	0.287	0.292	0.275	0.285
240	0.282	0.289	0.299	0.29
250	0.294	0.284	0.287	0.288

Table: Kp, Ki and Kd value versus PWM value

PWM value	Kp	Ki	Kd
120	29	0	1
130	30	0	1
140	31	0	1
150	31	0	1
160	33	0	2
170	35	0	2
180	35	0	3
190	38	0	5
200	41	0	6
210	43	0	7
220	46	0	8
230	49	0	9
240	52	0	10
250	54	0	10

Table: Result by implementing PID controller when the PWM value set at 120

Error (PID controller)	Error (P controller)	PID	Left motor speed	Right motor speed	Direction
0	0	0	120	120	Forward
0	0	0	120	120	Forward
0	0	0	120	120	Forward
0	0	0	120	120	Forward
0	0	0	120	120	Forward
0	0	0	120	120	Forward
0	0	0	120	120	Forward
0	0	0	120	120	Forward
0	0	0	120	120	Forward
0	0	0	120	120	Forward
2	0	58	62	178	Right turn
2	0	58	62	178	Right turn
2	0	58	62	178	Right turn
2	0	58	62	178	Right turn
2	0	58	62	178	Right turn
0	0	0	120	120	Forward
0	0	0	120	120	Forward
0	0	0	120	120	Forward
0	0	0	120	120	Forward
0	-2	0	120	120	Forward
0	-2	0	120	120	Forward
0	-2	0	120	120	Forward
0	-1	0	120	120	Forward
0	0	0	120	120	Forward
0	0	0	120	120	Forward
0	0	0	120	120	Forward
0	0	0	120	120	Forward
0	0	0	120	120	Forward
0	0	0	120	120	Forward

0	0	0	120	120	Forward
0	0	0	120	120	Forward
-1	0	-29	149	91	Left turn
-1	0	-29	149	91	Left turn
-2	0	-58	178	62	Left turn
-2	0	-58	178	62	Left turn
-2	-1	-58	178	62	Left turn
-2	-2	-58	178	62	Left turn
-2	-3	-58	178	62	Left turn
-2	-4	-58	178	62	Left turn
-1	-4	-29	149	91	Left turn
-1	-4	-29	149	91	Left turn
0	-4	0	120	120	Forward
0	-4	0	120	120	Forward
0	-4	0	120	120	Forward
0	-4	0	120	120	Forward
-1	-2	-29	149	91	Left turn
-2	-2	-58	178	62	Left turn
-2	-2	-58	178	62	Left turn
-2	-2	-58	178	62	Left turn
-2	-1	-58	178	62	Left turn
-2	0	-58	178	62	Left turn
-2	0	-58	178	62	Left turn
-2	0	-58	178	62	Left turn
-2	0	-58	178	62	Left turn
-2	0	-58	178	62	Left turn
-2	0	-58	178	62	Left turn
-2	-1	-58	178	62	Left turn
-2	-2	-58	178	62	Left turn
-2	-2	-58	178	62	Left turn
-2	-2	-58	178	62	Left turn
-2	-2	-58	178	62	Left turn
-2	-2	-58	178	62	Left turn

-2	-3	-58	178	62	Left turn
-2	-3	-58	178	62	Left turn
-2	-4	-58	178	62	Left turn
-2	-4	-58	178	62	Left turn
-2	-4	-58	178	62	Left turn
-1	-4	-29	149	91	Left turn
0	-4	0	120	120	Forward
0	-4	0	120	120	Forward
0	-4	0	120	120	Forward
0	-4	0	120	120	Forward
0	-4	0	120	120	Forward
0	-4	0	120	120	Forward
0	-4	0	120	120	Forward
-1	-4	-29	149	91	Left turn
-2	-4	-58	178	62	Left turn
-2	-4	-58	178	62	Left turn
-2	-4	-58	178	62	Left turn
-2	-2	-58	178	62	Left turn
-2	-2	-58	178	62	Left turn
-2	0	-58	178	62	Left turn
-1	0	-29	149	91	Left turn
0	0	0	120	120	Forward
0	0	0	120	120	Forward
0	0	0	120	120	Forward
0	2	0	120	120	Forward
0	2	0	120	120	Forward
-1	2	-29	149	91	Left turn
-2	2	-58	178	62	Left turn
-2	2	-58	178	62	Left turn
-2	2	-58	178	62	Left turn
-2	2	-58	178	62	Left turn
-2	2	-58	178	62	Left turn
-2	2	-58	178	62	Left turn

-2	0	-58	178	62	Left turn
-2	0	-58	178	62	Left turn
-3	0	-87	207	33	Left turn
-3	0	-87	207	33	Left turn
-3	0	-87	207	33	Left turn
-3	0	-87	207	33	Left turn
-3	0	-87	207	33	Left turn
-4	0	-116	236	4	Left turn
-4	0	-116	236	4	Left turn
-4	0	-116	236	4	Left turn
-4	0	-116	236	4	Left turn
-4	0	-116	236	4	Left turn
-4	0	-116	236	4	Left turn
-3	0	-87	207	33	Left turn
-2	0	-58	178	62	Left turn
-2	0	-58	178	62	Left turn
-2	0	-58	178	62	Left turn
-2	0	-58	178	62	Left turn
-2	0	-58	178	62	Left turn
-1	0	-29	149	91	Left turn
0	0	0	120	120	Forward
0	0	0	120	120	Forward
0	0	0	120	120	Forward
0	0	0	120	120	Forward
0	0	0	120	120	Forward
2	0	58	62	178	Right turn
2	0	58	62	178	Right turn
2	0	58	62	178	Right turn
2	-1	58	62	178	Right turn
2	-2	58	62	178	Right turn
0	-3	0	120	120	Forward
0	-4	0	120	120	Forward
0	-4	0	120	120	Forward

0	-4	0	120	120	Forward
0	-4	0	120	120	Forward
0	-4	0	120	120	Forward
-2	-4	-58	178	62	Left turn
-2	-3	-58	178	62	Left turn
-2	-2	-58	178	62	Left turn
-2	-2	-58	178	62	Left turn
-2	0	-58	178	62	Left turn
-2	0	-58	178	62	Left turn
-2	0	-58	178	62	Left turn
-2	0	-58	178	62	Left turn
0	0	0	120	120	Forward
0	0	0	120	120	Forward
0	0	0	120	120	Forward
0	1	0	120	120	Forward
0	1	0	120	120	Forward
0	1	0	120	120	Forward
2	0	58	62	178	Right turn
2	0	58	62	178	Right turn
2	0	58	62	178	Right turn
2	0	58	62	178	Right turn
2	0	58	62	178	Right turn
1	0	29	91	149	Right turn
0	0	0	120	120	Forward
0	0	0	120	120	Forward
0	0	0	120	120	Forward
0	0	0	120	120	Forward
0	0	0	120	120	Forward
0	-1	0	120	120	Forward
0	-2	0	120	120	Forward
-2	-2	-58	178	62	Left turn
-2	-2	-58	178	62	Left turn
-2	-2	-58	178	62	Left turn

-2	-3	-58	178	62	Left turn
-2	-3	-58	178	62	Left turn
-2	-3	-58	178	62	Left turn
-1	-4	-29	149	91	Left turn
0	-4	0	120	120	Forward
0	-4	0	120	120	Forward
0	-4	0	120	120	Forward
0	-4	0	120	120	Forward
0	-3	0	120	120	Forward
0	-2	0	120	120	Forward
0	-2	0	120	120	Forward
0	-2	0	120	120	Forward
0	0	0	120	120	Forward
1	0	29	91	149	Right turn
2	0	58	62	178	Right turn
2	0	58	62	178	Right turn
2	0	58	62	178	Right turn
2	2	58	62	178	Right turn

Appendix D

Gantt chart of Research Activities

Project Activity	Sep-14				Oct-14				Nov-14				Dec-14	
PSM 1	Wk 1	Wk 2	Wk 3	Wk 4	Wk 5	Wk 6	Wk 7	Wk 8	W k9	Wk 10	Wk 11	Wk 12	Wk 13	Wk 14
Topic discussed with lecture and the project understood	🟨													
The details collected regarding the topic and literature review studied	🟨	🟨	🟨	🟨	🟨	🟨	🟨	🟨	🟨	🟨	🟨	🟨	🟨	🟨
Experiment design and simulation					🟨	🟨	🟨							
The proposal completed for PSM 1 and prepared for presentation								🟨	🟨	🟨				
Started doing Hardware and coding for the controller											🟨	🟨	🟨	🟨

PSM 2	Jan-15	Feb-15	Mar-15	Apr-15	May-15	Jun-15
Continue doing hardware and coding for the controller	🟨					
Coding tested and analysis		🟨				
Report for PSM 2 completed			🟨	🟨	🟨	
PSM 2 Presentation						🟨

Appendix E: Programming for Line Following Robot with PID controller

```
        float Kp =0;
        float Ki = 0;
        float Kd = 0;
float error=0, P=0, I=0, D=0, PID_value=0;
float previous_error=0, previous_I=0;
int sensor[5]={0, 0, 0, 0, 0};
int initial_motor_speed = 255;

void read_sensor_values();
void calculate_pid();
void motor_control();

void setup()
{
 pinMode(3,OUTPUT); //Left Motor Pin 1
 pinMode(4,OUTPUT); //Left Motor Pin 2
 pinMode(5,OUTPUT); //Right Motor Pin 1
 pinMode(6,OUTPUT);  //Right Motor Pin 2
 Serial.begin(9600); //Enable Serial Communications
}

void loop()
{
   read_sensor_values();
   P = error;
   Serial.print(error);
   Serial.print('\t');
   Serial.print(P);
   I = I + error;
   Serial.print('\t');
   Serial.print(I);
```

```
D = error - previous_error;
Serial.print('\t');
Serial.print(D);
PID_value = (Kp*P) + (Ki*I) + (Kd*D);
Serial.print('\t');
Serial.print(PID_value);
previous_error=error;
int left_motor_speed = initial_motor_speed-PID_value;
if(left_motor_speed>255)
{
  left_motor_speed=255;
}
int right_motor_speed = initial_motor_speed+PID_value;
if(right_motor_speed>255)
{
  right_motor_speed=255;
}
Serial.print('\t');
Serial.print(left_motor_speed);
Serial.print('\t');
Serial.print(right_motor_speed);
if (right_motor_speed > left_motor_speed)
{
  Serial.print('\t');
  Serial.println("Right turn");
}
else if (right_motor_speed < left_motor_speed)
{
  Serial.print('\t');
  Serial.println("Left turn");
}
 else if (right_motor_speed = left_motor_speed)
{
    Serial.print('\t');
    Serial.println("Forward");
   }

   analogWrite(3,left_motor_speed);   //Left Motor Speed
   analogWrite(5,right_motor_speed);  //Right Motor Speed
   //following lines of code are to make the bot move forward
   /*The pin numbers and high, low values might be different
   depending on your connections */
   digitalWrite(4,LOW);
   digitalWrite(6,LOW);
}

void read_sensor_values()
{
  int value = 100;
  sensor[0]=analogRead(A0);
  sensor[1]=analogRead(A1);
  sensor[2]=analogRead(A2);
  sensor[3]=analogRead(A3);
  sensor[4]=analogRead(A4);

if((sensor[0]>value)&&(sensor[1]>value)&&(sensor[2]>value)&&(sensor[3]>value)&&(s
ensor[4]<value))       //00001//
  error=4;
  else
if((sensor[0]>value)&&(sensor[1]>value)&&(sensor[2]>value)&&(sensor[3]<value)&&(s
ensor[4]<value))  //00011//
  error=3;
  else
if((sensor[0]>value)&&(sensor[1]>value)&&(sensor[2]>value)&&(sensor[3]<value)&&(s
ensor[4]>value))  //00010//
  error=2;
```

```
 else
if((sensor[0]>value)&&(sensor[1]>value)&&(sensor[2]<value)&&(sensor[3]<value)&&(s
ensor[4]>value))  //00110//
 error=1;
 else
if((sensor[0]>value)&&(sensor[1]>value)&&(sensor[2]<value)&&(sensor[3]>value)&&(s
ensor[4]>value))  //00100/
 error=0;
 else
if((sensor[0]>value)&&(sensor[1]<value)&&(sensor[2]<value)&&(sensor[3]>value)&&(s
ensor[4]>value))  //01100//
 error=-1;
 else
if((sensor[0]>value)&&(sensor[1]<value)&&(sensor[2]>value)&&(sensor[3]>value)&&(s
ensor[4]>value))  //01000//
 error=-2;
 else
if((sensor[0]<value)&&(sensor[1]<value)&&(sensor[2]>value)&&(sensor[3]>value)&&(s
ensor[4]>value))  //11000//
 error=-3;
 else
if((sensor[0]<value)&&(sensor[1]>value)&&(sensor[2]>value)&&(sensor[3]>value)&&(s
ensor[4]>value))   //10000//
 error=-4;
 else
if((sensor[0]>value)&&(sensor[1]>value)&&(sensor[2]>value)&&(sensor[3]>value)&&(s
ensor[4]>value))  //00000//
   if(error==-4) error=-5;
   else error=5;
}
```

Appendix F: Programming for Line Following Robot without PID Controller

```
void print_sensor_values();
int errorfinding();
int left = 0;
int right = 0;
int r = 0;
int l = 0;

void setup()
{

 pinMode(3,OUTPUT); //Left Motor Pin 1
 pinMode(4,OUTPUT); //Left Motor Pin 2
 pinMode(5,OUTPUT); //Right Motor Pin 1
 pinMode(6,OUTPUT);  //Right Motor Pin 2
 Serial.begin(9600); //Enable Serial Communications
}

void loop()
{
   digitalWrite(4,LOW);
   digitalWrite(6,LOW);
   analogWrite(3,left);
   analogWrite(5,right);
   int error = errorfinding();
   switch(error)
  {
    case  '-4' :  analogWrite(3,left-4*l);analogWrite(5,right+4*r); break;
    case  '-3' :  analogWrite(3,left-3*l);analogWrite(5,right+3*r); break;
    case  '-2' :  analogWrite(3,left-2*l);analogWrite(5,right+2*r); break;
    case  '-1' :  analogWrite(3,left-l);analogWrite(5,right+r); break;
    case  '0'  :  analogWrite(3,left);analogWrite(5,right); break;
   case  '1'  :  analogWrite(3,left+l);analogWrite(5,right-r); break;
   case  '2'  :  analogWrite(3,left+2*l);analogWrite(5,right-2*r); break;
   case  '3'  :  analogWrite(3,left+3*l);analogWrite(5,right-3*r); break;
   case  '4'  :  analogWrite(3,left+4*l);analogWrite(5,right-4*r); break;
   default    :  analogWrite(3,left); analogWrite(5,right);
  }

  print_sensor_values();

}

 int errorfinding()
  {
  int error;
  int value = 100;

if((analogRead(A0)>value)&&(analogRead(A1)>value)&&(analogRead(A2)>value)&&(a
nalogRead(A3)>value)&&(analogRead(A4)<value))       //00001//
 {
  error=4;
 }
  else
if((analogRead(A0)>value)&&(analogRead(A1)>value)&&(analogRead(A2)>value)&&(a
nalogRead(A3)<value)&&(analogRead(A4)<value))  //00011//
  {
   error=3;
  }
  else
if((analogRead(A0)>value)&&(analogRead(A1)>value)&&(analogRead(A2)>value)&&(a
nalogRead(A3)<value)&&(analogRead(A4)>value))  //00010//
  {
  error=2;
  }
  else
if((analogRead(A0)>value)&&(analogRead(A1)>value)&&(analogRead(A2)<value)&&(a
nalogRead(A3)<value)&&(analogRead(A4)>value))  //00110//
  {
   error=1;
  }
  else
if((analogRead(A0)>value)&&(analogRead(A1)>value)&&(analogRead(A2)<value)&&(a
nalogRead(A3)>value)&&(analogRead(A4)>value))  //00100/
  {
  error=0;
  }
  else
if((analogRead(A0)>value)&&(analogRead(A1)<value)&&(analogRead(A2)<value)&&(a
nalogRead(A3)>value)&&(analogRead(A4)>value))  //01100//
  {
  error=-1;
  }
  else
if((analogRead(A0)>value)&&(analogRead(A1)<value)&&(analogRead(A2)>value)&&(a
nalogRead(A3)>value)&&(analogRead(A4)>value)) //01000//
  {
  error=-2;
  }
  else
if((analogRead(A0)<value)&&(analogRead(A1)<value)&&(analogRead(A2)>value)&&(a
nalogRead(A3)>value)&&(analogRead(A4)>value))  //11000//
  {
  error=-3;
  }
  else
if((analogRead(A0)<value)&&(analogRead(A1)>value)&&(analogRead(A2)>value)&&(a
nalogRead(A3)>value)&&(analogRead(A4)>value))   //10000//
```

```
 {
 error=-4;
 }
 else
if((analogRead(A0)>value)&&(analogRead(A1)>value)&&(analogRead(A2)>value)&&(a
nalogRead(A3)>value)&&(analogRead(A4)>value))  //00000//
  {
   error=5;
   if(error > 5)
   {
    error=5;
   }
  else if (error < -5)
   {
    error=-5;
   }
  }

  return error;
 }

void print_sensor_values()
{
 int error;
 Serial.println("\nA0 \tA1 \tA2 \tA3 \tA4");
 Serial.println("---------------------------------");
 for (int i = 0; i < 10 ; i++)
 {
 Serial.print(analogRead(A0));
 Serial.print("\t");
 Serial.print(analogRead(A1));
 Serial.print("\t");
 Serial.print(analogRead(A2));
```

```
  Serial.print("\t");
  Serial.print(analogRead(A3));
  Serial.print("\t");
  Serial.print(analogRead(A4));
  Serial.print("\t");
  Serial.println(error);
  }
  delay(1000);
}
```