The University of Nottingham

UNITED KINGDOM · CHINA · MALAYSIA

DEPARTMENT OF ELECTRICAL AND ELECTRONIC ENGINEERING

Design, Construction and Control of a Low Cost Intelligent Line Following Robotic Vehicle

Third year project report is submitted in part fulfilment of the requirements of the degree of Bachelor of Engineering.

Abstract:

Building and investigation of optimized line following robotic vehicles being capable of following specified line path drawn on the floor and doing other sophisticated tasks, have always been challenge problems within research of electrical and computer engineering. This project aims to construct the mechanical and electrical system of simple sensor-based line follower with various control algorithms being applied and compared. Beside this, an advanced line follower is developed based on simple line following robot capable of avoiding obstacles, solving simple maze etc. Finally several feasible future works focusing on control system improvement and advanced vision-based line following are investigated and simulated. The experimental results demonstrate that the designed vehicle system is capable of performing basic line following tasks, obstacle avoidance and solving a simple line maze. The analysis of output control response, which are achieved by an on-board data logger being seldom investigated by previous research, also provides an effective demonstration of various control strategies such as on/off control, P, PI, PD and PID control. Finally improvements of sensing and control strategies for future work are proposed and discussed.

Key Words: Control System Design, Electromechanical, Line Following Robotic Vehicle

Table of Contents

Abbreviations and Terminologies:

[1]	AGV:	Autonomous Guided Vehicle
[2]	ADC:	Analogue-to-Digital Converter (ADC)
[3]	AHRC:	Atlanta Hobby Robot Club
[4]	AI:	Artificial Intelligence
[5]	ASCII:	American Standard Code for Information Interchange
[6]	CAD:	Computer Aided Design
[7]	HIS:	Hue Saturation Intensity
[8]	Inc:	Incorporated Company
[9]	IR:	Infrared
[10]	I/P:	Input
[11]	LCD:	Liquid Crystal Display
[12]	LDR:	Line Dependent Resistor
[13]	LFS	Line Following Sensors
[14]	LFSA:	Line Following Sensor Array
[15]	LED:	Light Emitting Diode
[16]	LiPo Battery:	Lithium Polymer Battery
[17]	LUT:	Look-up-table
[18]	MIMO:	Multiple Input Multiple Output Control System
[19]	O/P:	Output
[20]	PID:	Proportional-Integral-Differential
[21]	PWM:	Pulse Width Modulation
[22]	RGB:	Red-Green-Blue scale
[23]	RMS:	Root-mean-square
[24]	RPM:	Revolutions per minute
[25]	RTC:	Real Time Clock Chip
[26]	Rx:	Receiver
[27]	SDHC:	Secure Digital High Capacity
[28]	Tx:	Transmitter
[29]	US:	Ultrasonic
[30]	UBEC:	Universal Battery Eliminator Circuit
[31]	USART/UART:	Universal Synchronous Asynchronous Receiver and Transmitter
[32]	UI:	User Interface

Chapter 1: Introductions and Background Review

1.1. Introduction and Summary of Literature Review

Technology for mobile robot positioning, navigation and control, termed as AVG (Autonomous guided vehicle), aims to develop a mobile robotic vehicle system being capable of finding real-time positions and controlling its movements in different environments, such that the vehicle is able to follow a predefined path or find target destination by evaluating surrounding landmarks, line pattern and signs as references [1, 4]. This technology is becoming popular in recent years, which leads to many innovative research projects such as the Google self-driving car project launched from 2009 [15]. Basically a line following robot, which is defined as a self-operated robotic vehicle system navigated by the pre-designed routes drawn on the floor, could be a perfect platform for investigating this technology. Beside broad applications for educational purpose, such as teaching instruments for demonstrating control theories and fundamental platforms for various robotic competitions (see figure 1 for an example line follower in the competition of AHRC (Atlanta Hobby Robot Club), line following robots are also widely used in human daily life [1, 2]. A good example purposed by Colak and Yildirim [10] is the small line follower carried train used in shopping malls or amusement parks for entertainment purposes, which has advantage of low construction cost (see figure 1.2) . One intuitive reason is that the cost for drawing line pattern is far less than building a rail track. Beside this, Punetha *et al.* [2] have developed a low cost line following robotic vehicle which is capable of carrying and delivering medicine to patients in hospital wards. A conceptual diagram of this equipment can be referred to figure 1.3, which has potential to solve problems of shortage of nurses in medical centres.

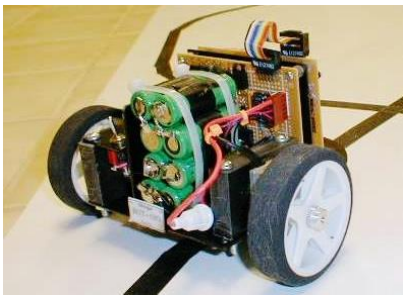


Figure 1.1: Example line follower built by participant of AHRC Competition in 2000 [1]

Figure 1.2: Example line following robot used for shopping mall entertainment purpose [10]

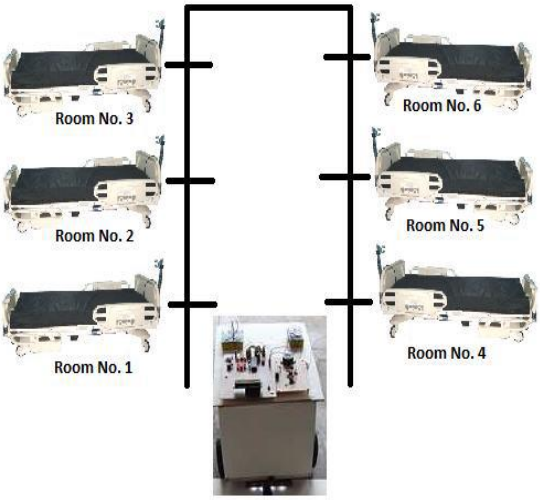


Figure 1.3: Line following robotic vehicle capable of delivering medicines in hospital rooms (image obtained from [2])

To implement a line following robot, Jäntech [11] suggested two main non-linear control strategies which are widely used for robotic motion control. These two approaches are feedback control, in which the control output signals are generated from instantaneous position error such that the vehicle is capable of making responses to output disturbances, and intelligent control, in which the advanced AI (artificial intelligent) computing approaches such as neural networks and ML (machine learning) will be applied to make robot smarter. Commonly, the generic feedback control system, which is the main focus of this project, could be abstracted and described by the block diagram shown in figure 1.4.

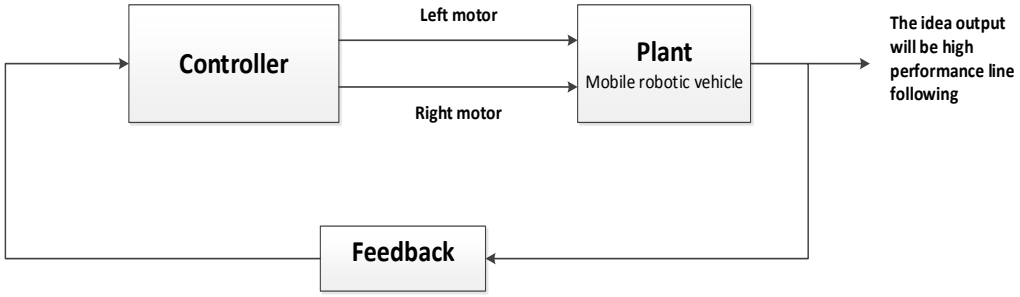


Figure 1.4: Generic block diagram showing the implementations of self-navigated line following robot

In this system, the feedback block aims to find robotic real-time position while the controller aims to control the speeds of steering motors and hence the direction of instantaneous motion, such that the high performance line following behaviours could be achieved. To implement the feedback block, two fundamental methods have been categorized by Dixon and Henlich [1], which can be summarized as sensor-based and vision-based method. Specifically a vision-based line following robot is trained to evaluate various line pattern from the image captured by on-board camera. Although this method has more potential to obtain information of real-time ahead line pattern compared to the sensor-based vehicle by applying customized image processing technique to the captured high resolution image, the complexity of implementing algorithms in microprocessor and processing time are always main concerns. Ismail *et al.* [4] demonstrate a typical method for developing vision-based line following robot, in which the line patterns in the captured image are segmented and extracted using a customized thresholding and filtering algorithm. The steering direction is then calculated

from the sensor array matrix, which is essentially the transformation of RGB components of the original image. Therefore the motors could be controlled by feeding the calculated results to the controller block. In contrast, the sensor-based line following robot takes information gathered by IR (infrared) LFS (line following sensors) which are capable of evaluating shades of colour underneath the sensor, and hence the centroid of steering direction can be located using appropriate algorithms. Clearly although the information of line pattern gathered by this method is far less than vision-based method, the low complexity and simple hardware makes it popular among engineers and hobbyists. This is because in a sensor-based vehicle, the number of values used to find a centroid depends on the number of LFS while with respect to vision based method, the centroid is found from pixels values of captured image, which in turns requires high performance microprocessor that is able to perform more complex calculations with fast speed [4, 5]. Although much research [5, 6, 7] focus on developing sensor-based line following robot has been done in recent years, the problems of how to exploit the capacity for obtaining more line pattern information using a limited number of line following sensors are usually neglected. In this project, these problems will be investigated and different methods for positioning a robotic vehicle will be compared, which is the fundamental of control system design.

Beside a feedback block, the design of the controller block shown in figure 1.4 is critical for implementing line following as well. Although many controller algorithms are used in previous research, a typical algorithm for implementing this controller block is PID (Proportional-Integral-Differential) control strategies, which is widely used in industrial applications and will be investigated detail in this project [7]. Nath *et al.*[7] have given a brief summary about the advantages of this controller whose output in time domain can be briefly described by equation (1), where $e(t)$ is the instantaneous error interpreted by the feedback block.

$$PID_{out} = P_{out} + I_{out} + D_{out} = K_P e(t) + K_I \int_0^t e(\tau) d\tau + K_D \frac{d}{dt} e(t) \qquad (1)$$

From equation (1), the theoretical functions of each of the PID components can be deduced. In general, the P controller reflects how far the robot deviates from the line, which aims to make a robot respond to and correct real-time errors. The I controller gives the sum of error integrated over time, which relate output to previous errors and shows that when output is not enough to correct an error, more change of output is required. The D controller is capable of preventing overshoot and hence reduces wobbling by acting as reverse of integral component. Although the effectiveness of this control algorithm has been proved by many researchers and hobbyists through various experiments qualitatively [5, 6, 7], few of them provide quantitative measurement of control output response so as to demonstrate this conclusion, which will be another focus of this project.

Finally, in order to exploit the capability of constructed line following robot, a maze solving robot, which is capable of finding target reference by walking through a line maze, will be developed on the basis of fundamental line following robot [13]. Although Sakib *et al.*[14] give a brief summary about some popular maze solver algorithms such as walk follower, breadth first-search algorithm and Prim's algorithm, the algorithm with best performances always depends on the specified line maze pattern and robotic architecture, which leads the problems of maze solving becoming a challenge for many engineers and hobbyists in recent years. Despite this challenge, the project will implement the simplest walk follower algorithm, however further advanced maze solver algorithm will be discussed as well.

1.2. Aims and Objectives

The ultimate aim of this project is to construct and develop a fully functional IR sensor based line following robotic vehicle with capable of avoiding obstacle ahead and solving simple line maze. Based on this vehicle, the ideas of sensing external environment, such as locate line position and detect obstacle ahead, and making decisions by 5 basic feedback control strategies will be demonstrated. These control strategies include on/off controller, P controller, PI controller, PD controller and PID controller. In order to achieve the ultimate aim, the project is divided into four minor stages whose aims and minor objectives are summarized as follows:

Stage 1: Mechanical Hardware and Infrastructure Construction

Aim: To construct and develop high performance mechanical prototype vehicle platform equipped with necessary sensors and add-ons;

Minor Objectives:
- Literature review about intelligent line following robot development;
- Design, assemble and construct robotic vehicle and necessary extensions from basic Parallax toolkits;

Stage 2: Simple IR Based Line Following Robot

Aim: To develop fully functional simple line following robot using popular control strategies including on/off control, P, PI, PD, and PID control;

Minor Objectives:
- Establish method for analysing control system performance;
- Design and implement algorithms for achieving 5 different controllers;
- Test and compare performance of different control system;
- Integrate all controllers into single system using a friendly menu system;

Stage 3: IR Based Maze Solver

Aim: To develop basic maze solver based on line following algorithms designed in previous stage;

Minor Objectives:
- Design and develop features of obstacle avoidance and junction recognition;
- Develop algorithms for achieving maze solving feature;

Stage 4: Future Development of Advanced Line Following Robot

Aim: Discussions and proposals of future advanced line following robot;

Minor Objectives:
- Several proposals for improving control performances, such as control of yaw angles;
- Design and simulations of image processing techniques for prospective vision based line following robot;

1.3. Report Structures

This report is divided into 6 chapters. The first chapter provides a brief summary of background reviews and introductions of this project. The next chapter gives a succinct overview of mechanical facilities and infrastructures corresponding to the implementations of first minor stages. After that, the third and fourth chapter present a detailed demonstration of the idea of sensing external environment and making decision while implementing a simple line following robot, which aims to achieve second minor stage. The comparisons and evaluations of effectiveness of different positioning method and control strategies will be given in these two chapters. The fifth chapter is involved with implementing a maze solving algorithms and developing an advanced maze solving robot based on vehicle system developed in previous stage. Finally some possible future works and improvements as well as time management issues will be discussed in the last chapter. Beside these, for future reference purpose, all software project files and some selective demo videos are provided in the attached CD. Details of the files can be referred from the corresponding electronic documentations.

Chapter 2: Hardware Infrastructures and Human Machine Interface (HMI)

In this chapter, the design and constructions of hardware infrastructures will be discussed in detail corresponding to the implementations of the first stage maintained in section 1.2. Discussions, comparisons and evaluations of various choices of component and equipment will be described in terms of criteria of electrical, electronic and mechanical engineering. Finally, a brief overview and introduction of constructed vehicle will be shown.

2.1. Selections of Robotic Platform

Although a customized self-built mechanical robotic platform may be the best choice for this project, the complicated and time consumed mechanical product design process [17] such as product definition, conceptual design and detailed design using CAD (computer-aided design) software can reduce the efficiency for achieving project aims. To overcome this issue, a commercial manufactured robotic shield is used, based on which several advanced extensions will be implemented. Calin [16] gave a concise review about the popular choice of robotic vehicle platforms, in which the Arduino Robot, Pololu 3pi Robot and Parallax Robot are possible choice for implementing this project. A brief evaluations and comparisons of these possible choices are summarized in table 2.1.

Table 2.1: Comparisons of possible robotic vehicle development platforms [18, 19, 20]

Robotic Platform	Arduino Robot	Pololu 3pi Robot	Parallax Robot
Example Image of Robotic Platform			
Number of Steering Wheels	2	2	2
Number of Casters	2	1	1
Microprocessor Unit	Arduino Leonardo	Atmega 328P (without built-in Arduino platform)	Arduino UNO
Disadvantages	• Poor mechanical design; • Uploading problem of control board; • Slow speed;	• Mechanical design is better than Arduino Robot however the distance between vehicle chasis and ground is too small; • Programming with Atmega can be more complex than using Arduino platform;	• Slow speed; • No built-in IR line following sensors; • No built-in UI platforms, e.g. LEDs, LCDs and buttons etc. • Official recommend method for supplying power is extremely inefficient; • One Arduino

		• Future extensions can hardly be installed;	platform may not have enough I/O pins;
Advantages	• Excellent and flexible UI (user interface) platform; • 5 built-in IR line following sensors; • The hardware design is flexible such that future extensions are possible;	• Fast speed; • 5 built-in IR line following sensors;	• Good mechanical design; • Highly flexible system where future extension modules can be easily added;

Generally speaking although the Arduino Robot has an excellent hardware design, in which the microprocessor unit is achieved by separating control board and motor board, the poor mechanical design where the total number of four wheels (two steering wheels and two casters) cannot guarantee that both of steering wheels touch ground can be a serious problem. This disturbance will cause the steering direction being hardly controlled by steering motors when one of steering motors does not touch ground. Additionally, the uploading issue with bootloader of Arduino Leonardo usually makes the program not upload [21]. Although this issue can be resolved either by an external programmer or by resetting system manually at certain exact time before program finishing uploading (i.e. double press the reset button approximately two second before program finishing uploading), these trivial technical problems can reduce project efficiency to a large extent. Another popular choice can be Pololu 3pi Robot, which has both fast speed and acceptable mechanical design, although the vehicle chasis might be too low such that the vehicle might get stuck at certain place when ground is not sufficiently flat. However since the robotic vehicle is not flexible enough where the external extensions can hardly be installed for both of mechanical and electronic reason, the robotic platform may not be an idea choice. A recommended platform for this project could be Parallax Robot, which is used in this project and has excellent mechanical design as well as sufficient considerations for adding external extensions. Although the original model does not include necessary UI platforms or IR line following sensors, these modules could be easily installed and interfaced to original robot externally. According to the manufacturer's documentations [20], a typical parallax robot, as is shown in figure 2.1, only includes an expandable aluminium vehicle chasis, two DC motors for steering purpose, one mechanical caster and a BOE (board of education) containing a 5V regulator for driving two DC steering motors. Based on this, future extensions including a line following sensor array, proximity sensors, data logger and LCD (Liquid Crystal Display) will be installed and interfaced which will be discussed in detail in following sections.

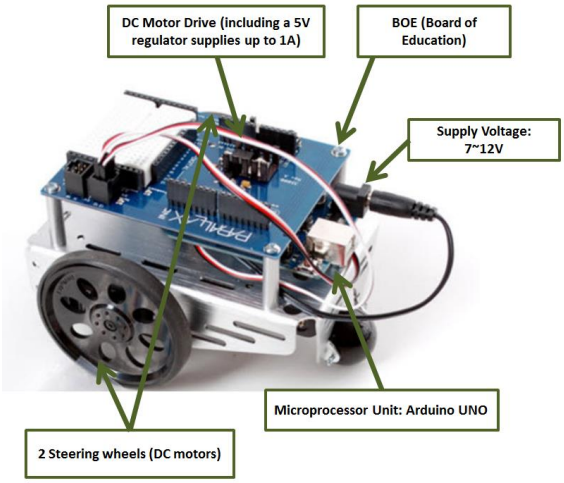
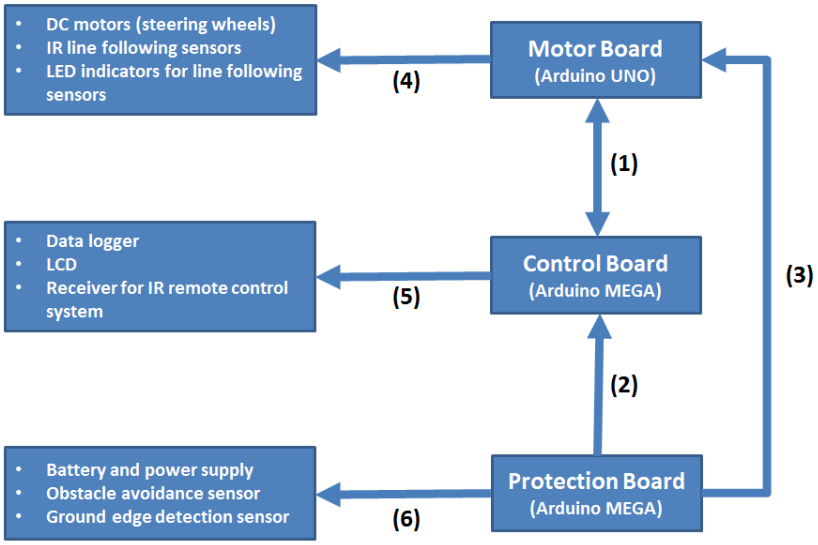


Figure 2.1: Configurations of original Parallax Robot model

2.2. Microprocessors and Communications

Although conventional microprocessors such as Microchip PIC and Atmel AVR can be a possible good choice for developing a high efficient system, the programming complexity may be a serious issue [22]. Instead, since the main focus of this project is designing a robust control system for line following robot, Arduino-based microcontrollers are used, which are favoured by many engineers and hobbyists due to the advantages of fast prototyping. However with respect to future work, the vehicle system may be migrated to conventional microprocessor based platforms for improving performance and efficiency.

In general, one disadvantage of Arduino is that it is not easy to execute multiple processes concurrently. This could be a serious issue for this project since the control of DC steering motors via duty cycle of PWM pulse is time critical event while the data logging process could consume huge amounts of time. Hence, in this project multiple Arduino controllers will be used for achieving virtually multitasking and hence improving overall performance. Specifically the configurations of microprocessors can be referred to figure 2.2 where an Arduino MEGA board is used to implement control board due to the advantages of 4 USART (Universal Synchronous/Asynchronous Receiver/Transmitter) channel for serial communications and two Arduino UNO board are used to achieve motor board and protection board. In this configuration, the motor board aims to control 2 steering wheels and perform line positioning process IR line following sensors. The data read by these sensors will be then be sent to control board and stored in SDHC (secure digital high capacity) memory card by path (1) using simplest UART serial communications. With respect to control board, not only does it store instantaneous line position data sent by motor board, but interface the UI device and the system, i.e. read data sent by IR remote controller and display motor status on LCD. Finally the role of protection board is designed to avoid obstacle collision, battery over discharge and falling from the test bench. For convenience, these events are defined as dangerous events in this project, which will trigger the system making a protection response, such as turning round when meet an obstacle and shutting down entire system when battery is about to over discharge etc.



Note:
1. Path (1) is achieved by bidirectional USART communications, which allows the control board to receive and log instantaneous line position data from motor board and the motor board receive configuration data from UI via the control board;
2. Path (2) and (3) allows system response to dangerous events, which will be determined by digital signal sent by protection board;

Figure 2.2: Block diagram illustrating the system configuration

2.3. Battery and Power Supply

Generally the battery and power supply module aims to power the system with a reliable output voltage. The powering method suggested by product supplier is to connect output of battery to Arduino Jack power sockets directly [20]. However the experimental measurements in table 2.2 show that the performance of this powering method is extremely poor which can generate huge amount of heat on device and drains the portable battery quickly. The reason is that this powering method uses the Arduino built-in linear regulator (AN117) to convert input 7 ~ 12 voltages to 5V voltage, which has very low efficiency (see figure 2.3).

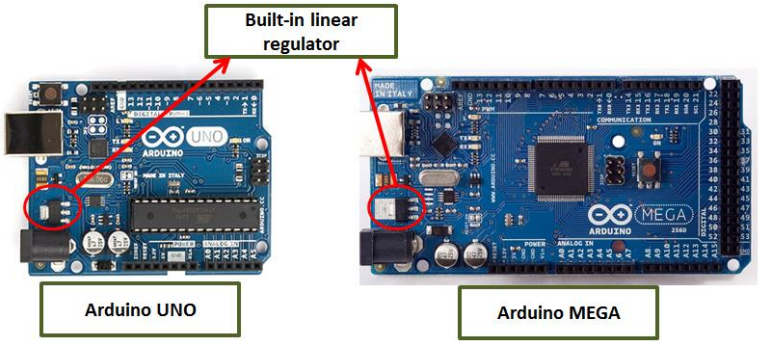


Figure 2.3: Arduino platform used in this project with linear regulators

Therefore in this project, an external UBEC (universal battery eliminator circuit) is used, which is essentially a switched mode DC regulator able to change battery output voltage to 5V for powering systems that has high efficiency (up to 90%) compared to linear regulator (normally below 50%) (see figure 2.4) [22]. In order to evaluate the power consumptions and effectiveness of this method, the power drawn from battery during different mode of operation will be measured which can be shown in table 2.2, which clearly shows that using Arduino built-in linear regulator draws much more current from battery that could generate huge amount of heat on device.

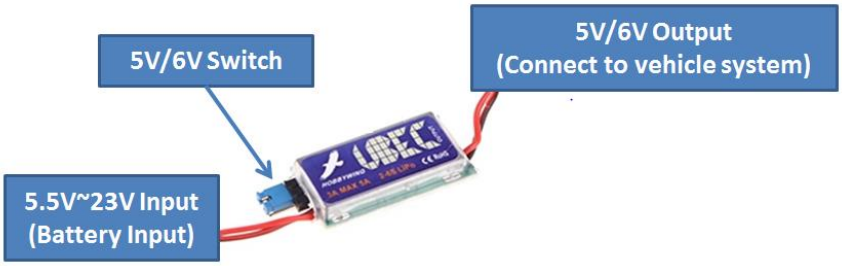


Figure 2.4: Switchable U-BEC 3A 5/6V 3A output used in this project

Table 2.2: Comparisons of power drawn from vehicle systems during different mode

Operation Mode		Arduino Built-in Linear Regulator			External UBEC		
		Current	Voltage	Power	Current	Voltage	Power
Quiescent Mode	Supply Side	3.5mA	7.9V	27.7mW	2.2mA	8.0V	17.3mW
	Load Side	N/A	4.9V	17.2mW	3.5mA	4.8V	17.0mW
	Losses	N/A	N/A	10.5mW	N/A	N/A	0.3mW
	Efficiency	N/A	N/A	62.1%	N/A	N/A	98.3%
Normal Operation Mode	Supply Side	4.2mA	7.9V	33.2mW	2.7mA	8.0V	21.7mW
	Load Side	N/A	4.8V	20.2mW	4.2mA	5.0V	21.0mW
	Losses	N/A	N/A	13.0mW	N/A	N/A	0.7mW
	Efficiency	N/A	N/A	60.8%	N/A	N/A	97.0%

Note:

1. The quiescent mode occurs when system is powered by battery while two wheels are static;
2. The normal operation mode occurs when vehicle system is moving, i.e. two wheels are rotating which theoretically may draw extra current;
3. The approximated power losses of linear regulator can be calculated in (2), where I indicate current drawn from supply.

$$P_{loss} = (V_{in} - V_{out}) \times I \quad (2)$$

On the other side, with respect to battery module, thanks to the advantages of low cost and light weight, the 2S LiPo (Lithium polymer) battery capable of providing 7.4V with minimum capacity of 1300mAh is chosen to power the system. However, typically when cell is over discharged, the battery can hardly be recharged again. This threshold can be marked by alarm line shown in figure 2.5 [24].

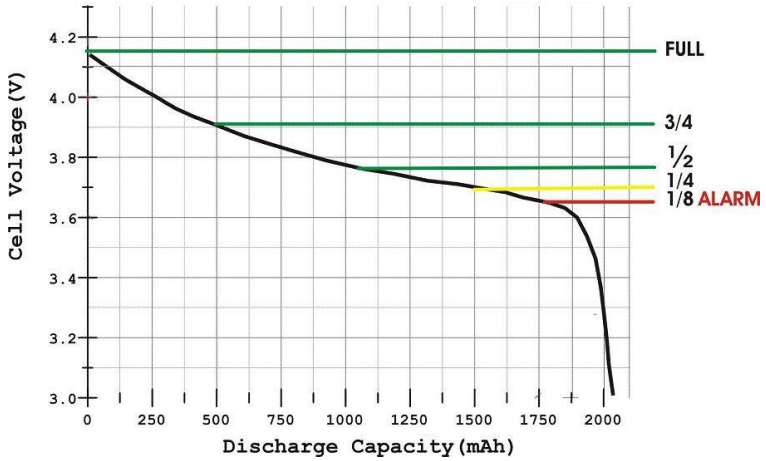
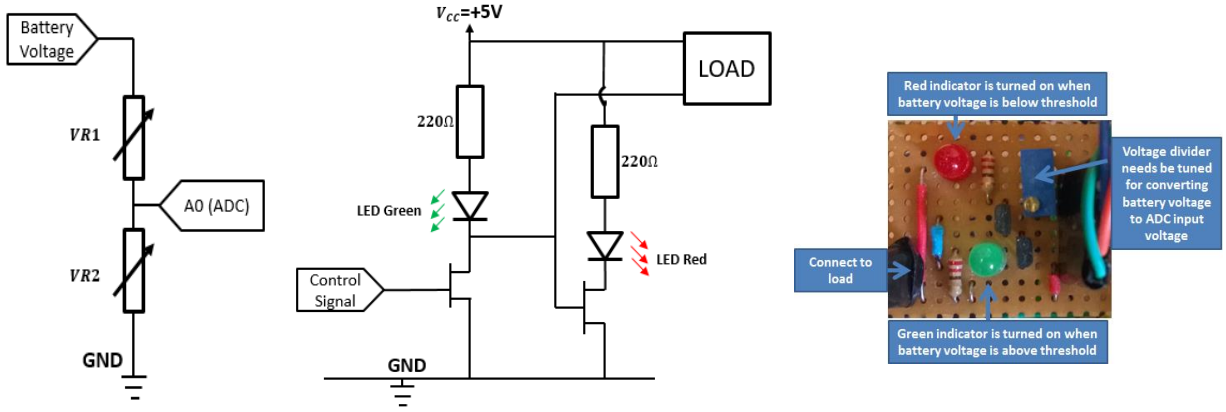


Figure 2.5: Typical LiPo battery discharge profile (1 cell approximation) [24]

Hence in practice, for avoiding over discharging of LiPo battery, a protection circuit is required which aims to measure instantaneous cell output voltage and disable the system once the battery voltage is almost near threshold. Figure 2.6 shows the designed circuit, where the output can be measured by figure 2.6(a) using a simple voltage divider and the load (indicating the rest of vehicle system) can be shouted down by the specified command signal sent from protection board. These two circuits are constructed on the same board to form a complete protection circuit as is shown in figure 2.6(c).



(a) Battery voltage measurement (b) Protection circuit for battery over-discharge (c) Constructed circuit board

Note:
1. The potentiometer can be represented by VR1 and VR2 in figure 2.6(a);
2. The load block in figure 2.6(b) indicates the remaining system including control board, motor board as well as relevant add-ons and sensors;
3. The control signal can be generated by digital pin of protection board;
4. The decision of whether battery output voltage is below threshold is made by protection board;
5. The A0 pin in figure 2.6(a) and control signal pin in figure 2.6(b) are belong to I/O pins of protection board;

Figure 2.6: Circuit module of battery over-discharging protection

2.4. Input User Interface (UI)

The aim of input UI in this project is to control vehicle behaviour such as start, stop and coefficient tuning according to practical demand. This could be a very efficient way for investigating and comparing how vehicle behaves under different control system or under same control system but with different parameters, which is one core purpose of this project. Hence by using this UI, the vehicle can be also served as a teaching tool for undergraduate control engineering course. Although commonly, such as the Arduino Robot shown in table 2.1, the input interface could be simply implemented by installing on the vehicle, which means while pressing the button, the vehicle is not static, this method may not be an efficient way to control robot since it is moving during normal operation. An alternative way to achieve this is to control the robot remotely, which can be achieved by either a hardware remote control system or mobile application controlled robot via Bluetooth etc. In this project, for a demonstration purpose, a simple low cost IR remote controller and receiver are used, which are shown in figure 2.7. By using this pair of device, the commands sent by controller are encoded to a 3 bytes number which will be decoded by receiver. Therefore the received commands can be easily used by using a simple LUT (look-up-table) as is shown in the project software file in attached CD.

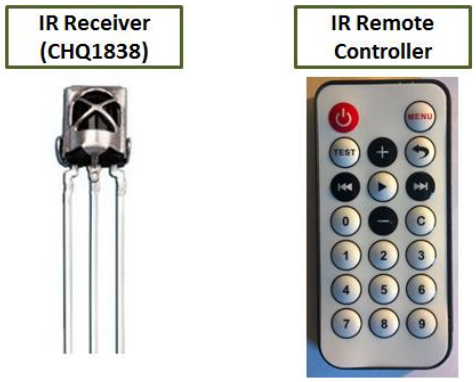


Figure 2.7: IR remote controller and receiver for achieving input UI

2.5. Output User Interface (UI) and Data Logger

One innovative aspect of this project compared to many research [5, 6, 7] is the output UI. These output UI modules mainly includes the LED indicators for LFSA (line following sensor array), LCD screen and a data logger, which is capable of displaying real-time instantaneous states and storing position data for future control analysis.

In the first place the implementation of indicators for the IR line following sensor array aims to provide a visual effect showing which IR sensor detects the black line pattern during each time instants. Since A/D converted results given by each IR line following sensor array will be values ranging from 0 to 1000 where the darker pattern gives a larger value (the detail operation of IR line following sensor can be referred to chapter 3), the idea for converting value measured by line following sensor to LED control signal is to define a threshold and the LED will be turned on once the measured value is larger than predefined threshold. The designed method for implementing this conversion can be divided into two categories which are termed hardware and software method in this report. Generally as is shown in figure 2.8, the hardware method is capable of controlling LED directly from the output of LFSA and deciding whether output is greater than threshold by a comparator while the designed software method is achieved by reading output from line following sensor and control LED

via digital pins of a microcontroller, which is acted as an intermediate medium deciding whether sensor detect the black line pattern.

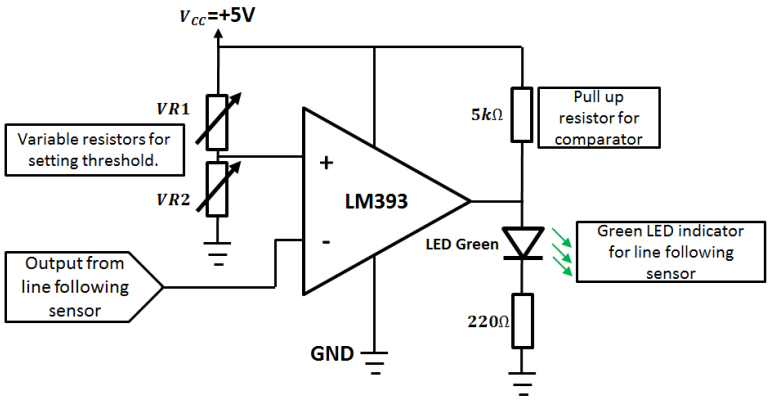
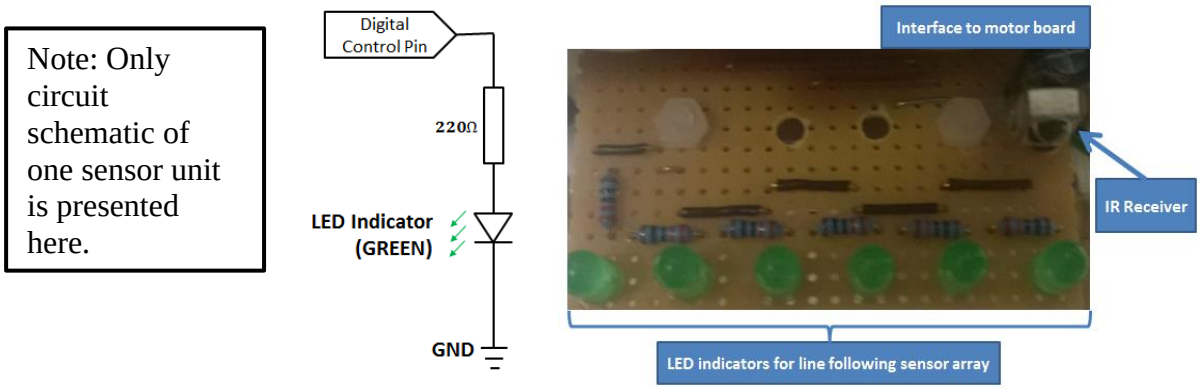


Figure 2.8: Circuit diagram of LED indicators for line following sensor array using hardware method

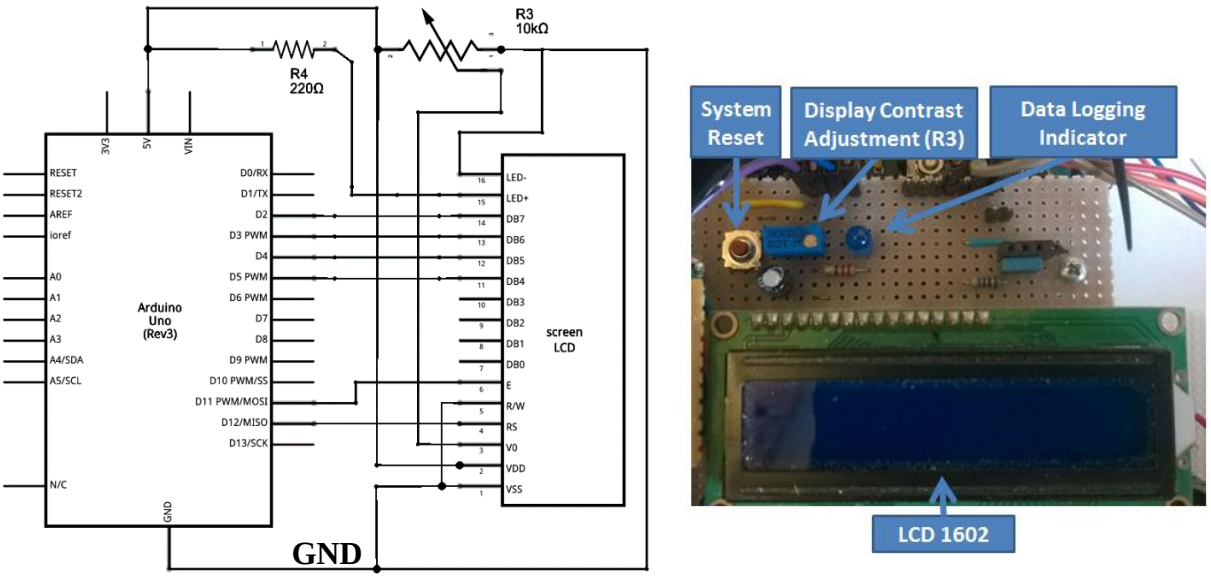
Clearly the advantage of hardware method is that the entire process is independent of the microcontroller and hence will not use the Arduino resources. However one fatal drawback of this method is that the physical size of a conversion circuit is larger than software method and hence it is hard to be physically embedded into vehicle system. Additionally by using this method it is hard to disable LFS indicator when the feature is not needed, hence the power consumption during quiescent mode could be larger than software method. Therefore in this project, the software method that is achieved by driving LFS indicators using digital pin of microcontrollers is used due to the advantages of easy control and small circuit size, which although might consume extra motor board resources. Furthermore this method allows threshold being easily changed by software program which can be useful when experimental environment is changed such as the change of colour of line path. The circuit arrangement of this method is purely a simple control circuit of LED whose schematic can be shown in figure 2.9(a) with the constructed board shown in figure 2.9(b).



(a) Circuit diagram one sensor unit (b) Constructed circuit board

Figure 2.9: Circuit of LED indicators for line following sensor array using software method

A second aspect of UI in this project focuses on the LCD screen, which aims to display real-time system information. In this project the basic 1602 LCD is used being capable of 32 displaying ASCII characteristics in 2 rows [25]. The circuit configuration for driving this LCD can be shown in figure 2.10(a) with the constructed circuit board presented in figure 2.10(b). In this circuit the potentiometer R3 aims to adjust LCD display contrast, which aims to provide a clear display of characters in different environment.



(a) Theoretical circuit diagram (b) Constructed circuit board

Figure 2.10: Driver circuit for LCD 1602 [25]

Finally since one of aims of this project is to investigate the effectiveness of different controllers, the data logger is a necessary part which is capable of storing instantaneously vehicle position data that is used to analyse control output response quantitatively such that the performance of designed controller can be evaluated. Although there are many data storage devices in modern electronic market, the SDHC memory card will be selected for this project due to advantages of fast read/write speed and small physical size. However despite this, the data writing time can be much larger than calculations of various controllers. Therefore in this project, a separated microcontroller, termed as control board, is used for interfacing the LCD and data logger, such that two processes can work in parallel (see figure 2.2 for communications between microcontrollers). As a result, for simplicity purpose, the data logger manufactured by Adafruit is chosen for interfacing SDHC card and rest of systems which also contains a RTC (real-time clock) chip being capable of adding time stamp on each generated file (see figure 2.11) [26].

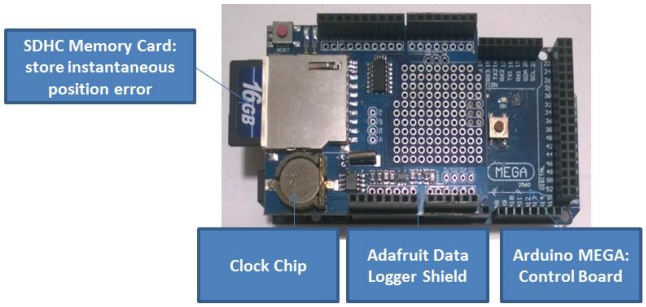


Figure 2.11: Data logger for storing instantaneous vehicle position information

2.6. Design of Menu Systems

Overall the menu system of designed vehicle aims to provide feature working mode selection from IR remote controller and configurations of different controller. To begin with, the main menu will be entered by which one of three modes can be selected as is shown in figure 2.12.

Different mode can be switched by pressing << and >>, and the mode can be entered by pressing confirm button.

(a) On/Off control mode

(b) PID control mode

(c) Maze solver mode

Figure 2.12 LCD displays of menu system

Considering PID control mode, which aims to achieve P, PI, PD, and PID control, the parameters of coefficients of three control components need be entered as is shown in figure 2.13. In this way the effect of different controller with different coefficients may be compared and evaluated. While the on/off controller mode and maze solver mode is able to run directly without customized settings.

(a) Setting base speed

(b) Setting k_p

(c) Setting k_i

(d) Setting k_d

Figure 2.13: LCD displays for PID control mode

When vehicle starts running, data logger will start working automatically, in which the file name can be indicated on screen such as figure 2.14 and figure 2.15. In particular, once unknown errors occur in SDHC memory card, error prompt such as figure 2.16 will prompt. In this case, the memory card needs to be formatted reset to original correct version.

Figure 2.14: LCD display for on/off control mode

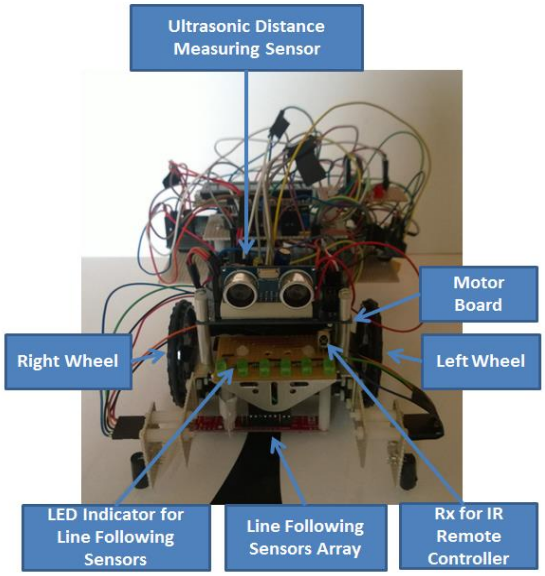
Figure 2.15: LCD display for PID control mode

Figure 2.15: LCD display for maze solving mode

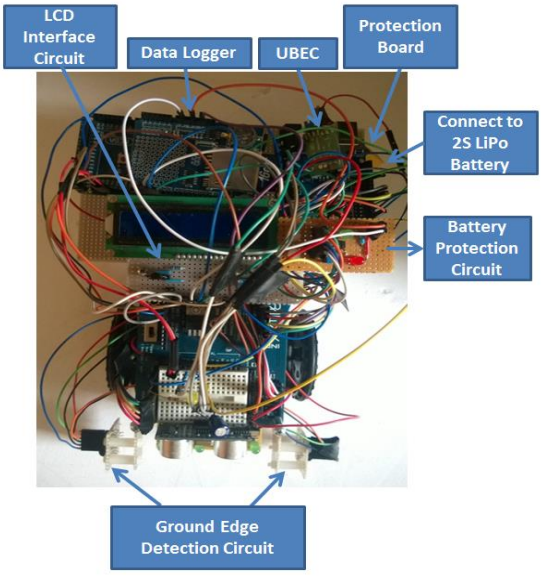
Figure 2.16: Error prompt screen

2.7. Overview of Constructed Infrastructures

The final constructed fully functional robotic vehicle system including front view and top down view can be seen in figure 2.17 with the detailed connections and functions of each pin being listed in appendix 2, which provide detailed guidance for future project reconstructions. Specifically the final robotic vehicle is constructed under the guidance of figure 2.12, which contains three basic modules controlled by motor board, control board and protection board. This constructed platform provides possibilities for investigating line following control systems with automatic obstacle avoiding capabilities and maze solving algorithms which will be demonstrated in following sections. Besides this, the constructed vehicle is also equipped with a fully functional user interface systems that is capable of read command sent by an IR remote controller and display of information by LED indicators or LCD screen.



(a) Front view



(b) Top down view

Note:

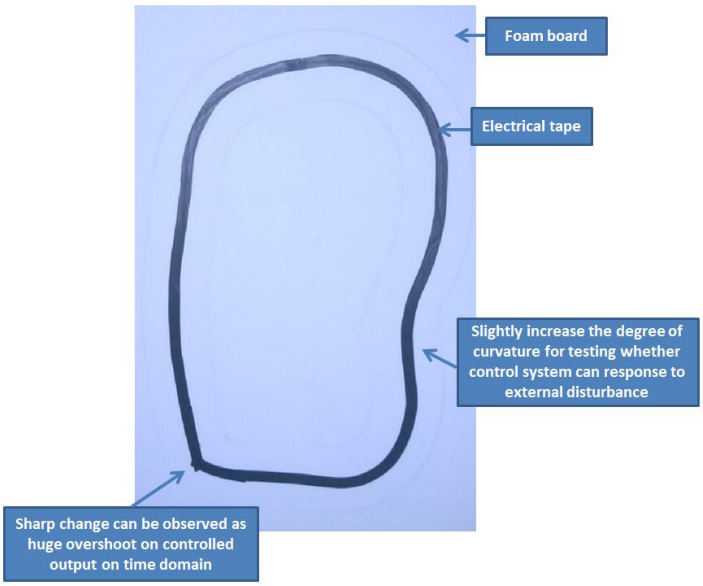
1. The left and right steering wheels are defined by looking at the direction of robotic forward motion;
2. For reference and comparison purpose, the original robotic development platform purchased from Parallax Inc can be referred to figure 2.1.

Figure 2.17: Overview of constructed vehicle system

Chapter 3: Methodologies and Project Test Environment

3.1. Construction of Experimental Board and Line Pattern

The aim of experimental board in this project is to evaluate the performance of designed vehicle control system, such as steady state, overshoot, natural frequency, ability for responding to external disturbance and settling time etc. Therefore the designed experimental board and line pattern for simple line following robot test should have several necessary features and be able to simulate real disturbances. However it should be noticed that further improvements are desired when vehicle system is tested in other environment. An experimental board made for simple line following test can be seen in figure 3.1 being capable of creating sharp change which can be used to test the ability of robot of responding to disturbance and measure average robotic speed. This is because the overshoot created by sharp change can be clearly reflected from controlled output on time domain, by which the period and hence the average speed can be calculated. Beside this, the foam board is selected as the major material for making experimental platform which can reduce sliding friction between vehicle and ground effectively.

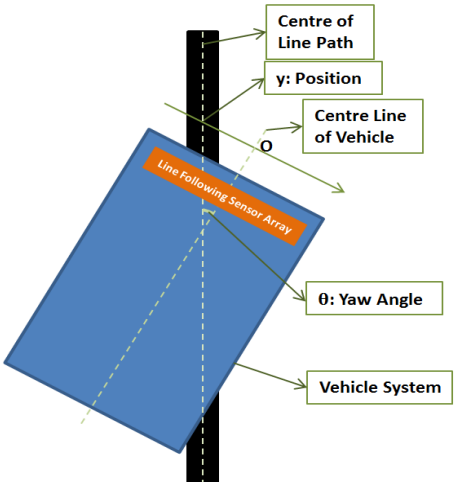


Note: the measurement results show that the whole length of one lap is around 185cm.

Figure 3.1: The experimental boar containing line pattern used in this project

3.2. Methodologies for Analysing Output Responses

The fundamental method for analysing vehicle control system performance is to compare theoretical idea response and measured output response from the data acquired by the data logger. Obviously two parameters can be used for characterising vehicle instantaneous error which are termed yaw angle and position that can be measured using figure 3.2.



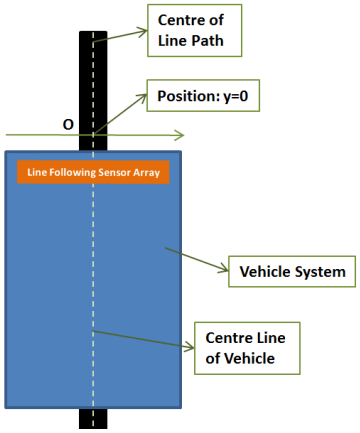
Note:

1. The position of line following sensor array defines the front of vehicle;
2. The yaw angle is defined by the angle between centre of vehicle and centre of line;
3. The position error is the one detected by line following sensors;
4. The range of valid position value is between -25mm to 25mm which is defined by physical length of line following sensor array;

Figure 3.2: Definition of defining instantaneous error information

Cleary in theory the output from control system should be that the vehicle can follow the line smoothly and accurately (see figure 1.74 for block diagram of control system and figure 3.3 for theoretical desired vehicle position). Hence from a mathematical point of view, the output position error in time domain should always be 0 and there should be no components in frequency domain. Although the performance of control system can be evaluated from time domain visually and qualitatively, more accurate results may be obtained and evaluated quantitatively from frequency domain, which is essentially the Fourier transform of signals in time domain. More specifically, these results may include:

- steady state error, which can be approximated by mean value of signal in time domain that is proportional to the DC component in frequency domain [27];
- degree of wobbling, which can be evaluated by harmonics involved. Obviously the ideal controller should have almost no harmonics;



Note: the ideal case occurs when vehicle follows line accurately where position is always 0 and there is no wobbling hence there is no components in ideal time domain.

Figure 3.3: Theoretical desired vehicle position

Beside this, two quantitative statistics results may be calculated using MATLAB and used to evaluate the designed control algorithm in terms of time domain, which includes:

- average value of signal in time domain, which can be approximated by equation (3) that can be used to evaluate steady state.

$$Average = \frac{Area\ enclosed\ by\ output\ respone}{Duration\ of\ measurment} = \frac{trapz(time,position)}{max(time)-\min(time)} \quad (3)$$

where *trapz()* is the MATLAB built-in function for implementing numerical integration using trapezoidal approximation method;

- Root-mean-square value (RMS) (a.k.a. standard deviation), which can be used to evaluate the degree of samples being spread out, i.e. how wobbling the vehicle it is. The calculation of RMS value can be given in (4) according to mathematical definition where T is the duration of measurement;

$$RMS = \sqrt{\frac{\int_0^T (position(t)-average)^2 dt}{T}} = \sqrt{\frac{trapz(time,(position-average).\hat{}2)}{T}} \quad (4)$$

As a result, from above reasoning, the theoretical control output response in time domain and frequency domain can be referred to figure 3.4. Particularly, with respect time domain the desired average value and RMS value should be closed to 0. In this project, these deductions will be used as reference to compare the measured output response for a specific designed control system.

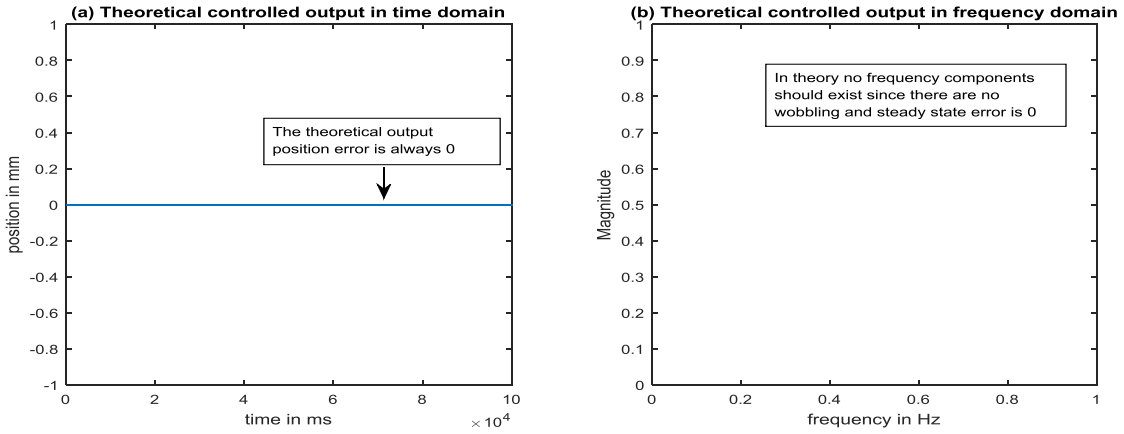


Figure 3.4: Theoretical desired output from vehicle control system

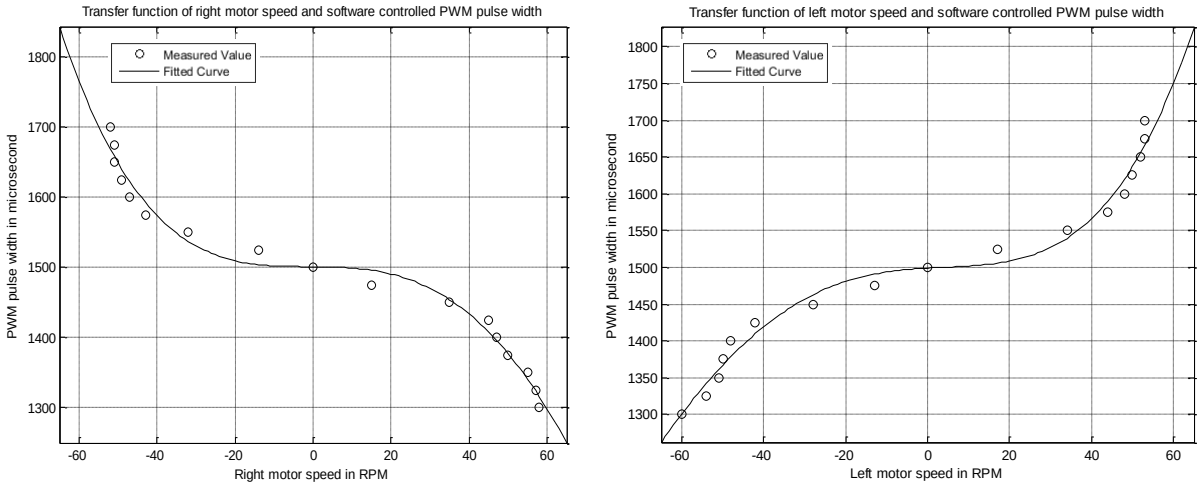
3.3. Wheels Calibrations and Speed Control

In this project, the steering wheels is essentially two servo motors controlled by the motor board and the control of motor speed are determined by pulse width and hence the duty cycle of the control signal, i.e. the duration when control signals stay at high voltage. A suggested method for controlling these servo motors is to use Arduino Servo Library, which is capable of setting up high frequency accurate pulses using interrupt method, i.e. the motor board can focus on other task such as reading information from line following sensors while generating accurate motor control pulses at background. From this reasoning, in order to generate approximately same wheel speeds using identical pulse width, a wheel calibration processes is suggested, which is implemented by setting speed of two motors to 0 manually when fed same pulse width of 1.5ms (1500μs). However, despite this, it can hardly guarantee that each of same pulse width gives same motor speed and the problem is that the desired controlled quantities is the rotation speed of two steering wheels, which control the basic vehicle movement, while, in terms of electrical engineering side, the servo motor used in this project is controlled by pulse with of PWM signal. Therefore in order to control motor speed directly, it will be necessary to explore the transfer function between PWM pulse width and steering motor speeds (i.e. the function for converting input desired speed to suitable PWM

pulse width, which will be used to generate corresponding motor speed), which can be presented in figure 3.5. By using *curve fitting tools* in MATLAB, the transfer function can be approximated in (5) and (6) where pulse width is measured in unit of μs and ω indicates the motor angular speed in rpm.

$$PulseWidth_{Left} = 5.6 \times 10^{-6}\omega^4 + 9.6 \times 10^{-4}\omega^3 - 0.013\omega^2 + 0.3\omega + 1499 \quad (5)$$

$$PulseWidth_{Right} = 3.5 \times 10^{-6}\omega^4 - 1.1 \times 10^{-3}\omega^3 - 3.8 \times 10^{-3}\omega^2 - 0.05\omega + 1501 \quad (6)$$



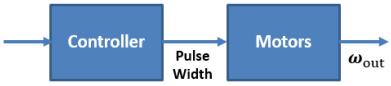
(a) Transfer function of left motor (b) Transfer function of right motor

Note:
1. The positive speed will drive vehicle system to move forward while the negative speed will drive vehicle to move backward. Therefore for driving vehicle to move forward the left motor should rotate in anticlockwise direction while the right motor should rotate in clockwise direction (the definition of left and right steering wheels can be referred to figure 2.12(a));
2. When pulse widths are set to roughly 1500μs the wheels will stop. This fact is used as guidance for calibrating steering motors;

Figure 3.5: Transfer functions of left and right steering wheels

In order to evaluate the performance of speed control, a conventional crude method used in many research [5, 6, 7] is used for reference. The difference of these two control method is whether the actuation signal generated by controller will be processed by transfer function, termed as speed to pulse width converter, mentioned in figure 3.5 (see block diagram in figure 3.6).

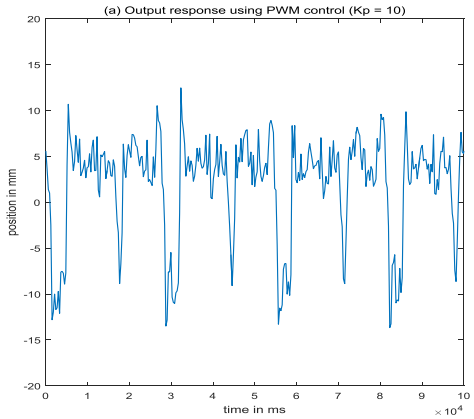
(a) Speed control scheme



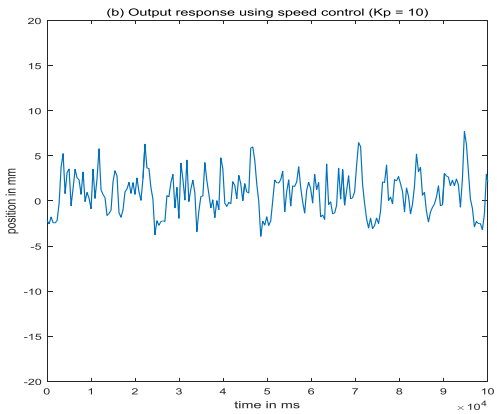
(b) Conventional crude pulse width control scheme

Figure 3.6: Part of block diagram of speed control and crude pulse width control

For comparing output performance, a simple P controller algorithm is used to implement controller block whose output in time domain and frequency domain can be seen in figure 3.7 and 3.8 (the detail of P controller can be referred to section 5.1). Clearly, from the observations and comparisons of statistics in time domain, it is summarized that the output response of speed control has less steady state error and RMS value which is more closed to theoretical output shown in figure 3.4. This result can also be confirmed from the frequency spectrum of speed control method shown in figure 3.8, which has smaller DC content and less high order harmonics. Hence, in the following sections the method of speed control will be used to implement various control algorithms for further investigations.



(a) Conventional pulse width control

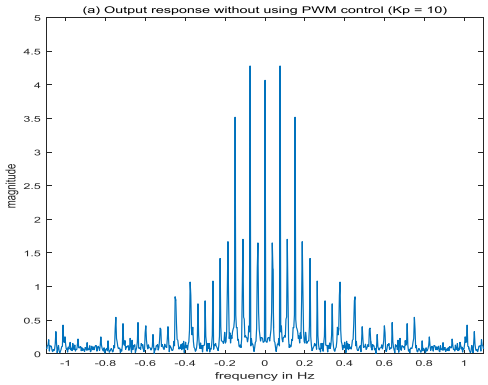


(b) Speed control

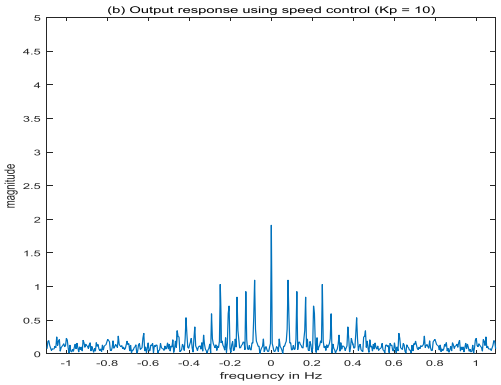
Note:

1. the statistic analysing of pulse width control shows that the mean value is around 2.0360mm and the RMS value is around 5.6068mm;
2. the statistic analysing of speed control shows that the mean value is around 0.9533mm and the RMS value is around 2.2453mm;

Figure 3.7: Comparison of output of speed control and pulse width control in time domain



(a) Conventional pulse width control



(b) Speed control

Figure 4.8: Comparison of output of speed control and pulse width control in frequency domain

Chapter 4: System Design for Sensing External Environment

The system block for sensing external environment is essentially the feedback block shown in figure 1.4 which aims to interpret and quantify the instantaneous vehicle output that will be identified by controller for producing actuation signal. In this project, for designing a MIMO (multiple input and multiple output) control system, the feedback block is designed to consist of three fundamental components which aim to detect the line position, the obstacle ahead and the ground edge of experimental platform for avoiding vehicle fall down from experimental platform.

4.1. Sensing Line Position

4.1.1. Reflectance Line Following Sensor Array

One key aspect for implementing a line following robot is to recognize the instantaneous robotic position accurately, which will be fed to controller that is capable of producing suitable actuation signal. Hasan *et al.*[5] suggested a popular method for sensing line position for sensor based line following vehicle, which uses IR LED and LDR (line dependent resistor) to detect the darkness of ground colour. The basic working principle of such a pair sensor can be referred to figure 4.1 where majority of light can be sensed by LDR if surface is white colour and minority of light can be sensed if surface is black. In this way the darkness of colour of surface can be recognised by intensity of reflected light, where less reflected light indicates higher darkness level corresponding to larger returned value.

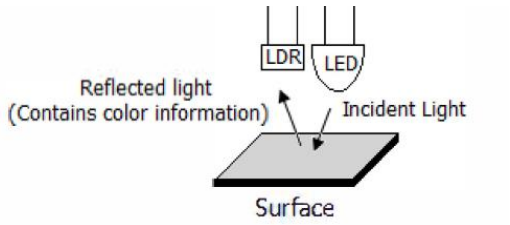
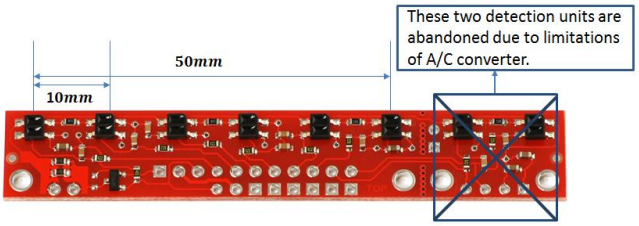


Figure 4.1: Working principle of a pair IR based LFS [5]

In this project the reflectance LFSA manufactured by Pololu will be used which contains 8 LFS shown in figure 4.1 [30]. However, in this project only 6 IR LFS are used due to the limitations of number of A/D converters on the motor board. The measurement of this sensor array can be marked in figure 4.2 with the detection length being 50mm (range of detectable position error is -25mm to 25mm) and distance between two adjacent sensors being 10mm, which determine the detection range and resolution of captured line pattern information. Therefore one challenge for developing a line following robot is to investigate how to exploit as much line position information as possible from the returned values of these 6 LFS, which will be discussed in the following sections.



Note:

1. The 0 of position error is defined at the centre point of line following sensor array (see figure 3.2 for detail explanations);
2. From software point of view, the sensor returned values will be stored in an array, e.g. *boolean[] sensor* or *double[] sensor* where the *sensor[0]* refers the returned value given by most left sensor;

Figure 4.2: Dimensions of IR line following sensor array used in this project

4.1.2. Digital Sensing Method

A conventional method, termed as digital method in this project, for sensing line position using IR line following sensor array is to create a LUT to map the results given by sensor array and measured position error. In this way, the measured LUT can act as the feedback block which is able to transfer the control output to instantaneous position error. In particular the returned values given by each sensor are converted to a two level discrete value where this value is equal to 1 when sensor returned value is larger than predefined threshold (the surface is considered as black line pattern in this case) and equal to 0 if sensor returned value is smaller than threshold (the surface is considered as white ground, i.e. non-line area, in this case). Therefore with this reasoning, the measured LUT can be referred to table 4.1.

Table 4.1: Digital method for sensing line information using line following sensor array

sensor[0]	sensor[1]	Sensor[2]	Sensor[3]	Sensor[4]	Sensor[5]	Position (y)
0	0	0	0	0	0	=last position
1	0	0	0	0	0	-25mm
1	1	0	0	0	0	-20mm
0	1	0	0	0	0	-15mm
0	1	1	0	0	0	-10mm
0	0	1	0	0	0	-5mm
0	0	1	1	0	0	0mm
0	0	0	1	0	0	+5mm
0	0	0	1	1	0	+10mm
0	0	0	0	1	0	+15mm
0	0	0	0	1	1	+20mm
0	0	0	0	0	1	+25mm

Note:

1. In this table, the sensor value of 1 means the measured darkness is larger than threshold, which indicates the corresponding sensor unit is above line pattern;
2. Although the combinations of 6 sensor values is 64 and there are only 12 cases listed in this table, the rest of situations will not be considered in this sections which involved with more complicated cases such as identifications of junctions;

4.1.3. Analogue Sensing Method

Although the experimental results shows that it is possible to use the simple but popular digital method to implement line position sensing, the resolution of converted line position error may not be satisfied, which is 1 part out of 11 (~9%) of full scale with each position step size being 5mm. This means that the position error during every instant will always be converted to a pre-defined position error value specified in table 4.1 which will generate an inherent quantisation error. As a result, it may be necessary to find a better sensing method to fully exploit the information provided by these 6 line following sensor units.

To overcome this issue, an alternative method, termed as analogue method, is used in this project which can better exploit the instantaneous vehicle position error by calculating the centre of sensed line pattern mathematically. Fosu *et al.* [31] indicate that this method is commonly used in astronomy field which aims to calculate and find the centre of stars from the exposure telescope image. The method for finding the centre of line pattern can be considered as finding the 'centre of masses of line pattern image which can be achieved by calculating weighted average of pixel intensity (pixel index in grey scale) which is indicated by the returned value of each sensor units. This can be summarized mathematically in formula (7) where the coefficient of each sensor value can be considered as the 'weight' of each pixel point and the position error value is in unit of mm.

$$y = \frac{(-25)\cdot sensor[0]+(-15)\cdot sensor[1]+(-5)\cdot sensor[2]+5\cdot sensor[3]+15\cdot sensor[4]+25\cdot sensor[5]}{sensor[0]+sensor[1]+sensor[2]+sensor[3]+sensor[4]+sensor[5]} \quad (7)$$

Another explanation of this method is to model the analogue values of each returned sensor value as the components of a probability distribution function where the higher component indicate the higher level of darkness of surface. Obviously according to the nature of probability function, the centre can be found from mean value, which can be statistically estimated from (7). For more detail demonstrations, an example returned sensor array value was measured and recorded in table 4.2 where the value from LUT is generated by table 4.1 when threshold darkness level is defined as 500 (half of full darkness). Hence the position data given by digital method is 10mm by checking LUT in table 4.1. However in order to find position data using analogue method, the original sensor returned value can be plotted in figure 4.3(a) and hence the image sensed by sensor array can be reconstructed in figure 4.3(b). Based on this plot, a fitting curve using Gaussian function model can be generated as is shown in figure 4.3(b) by which the peak position can be located which is around 7.81mm. However in software coding, the Gaussian curve fitting process will not be implemented which could take huge amount of processing time, instead the position can be calculated statistically using formula (7) where y indicates the centre position (the results can be indicated in (8)).

$$\begin{aligned} y &= \frac{(-25)\cdot sensor[0]+(-15)\cdot sensor[1]+(-5)\cdot sensor[2]+5\cdot sensor[3]+15\cdot sensor[4]+25\cdot sensor[5]}{sensor[0]+sensor[1]+sensor[2]+sensor[3]+sensor[4]+sensor[5]} \\ &= \frac{(-25)\times 44+(-15)\times 47+(-5)\times 55+5\times 624+15\times 332+25\times 42}{44+47+55+624+332+42} \\ &= 6.18mm \end{aligned} \quad (8)$$

Table 4.2: An example returned sensor array information for demonstrating position location using analogue method

Sensor index	0	1	2	3	4	5
Sensor reference position	-25mm	-15mm	-5mm	+5mm	+15mm	+25mm
Original value	44	47	55	624	332	42
Value from LUT	0	0	0	1	1	0

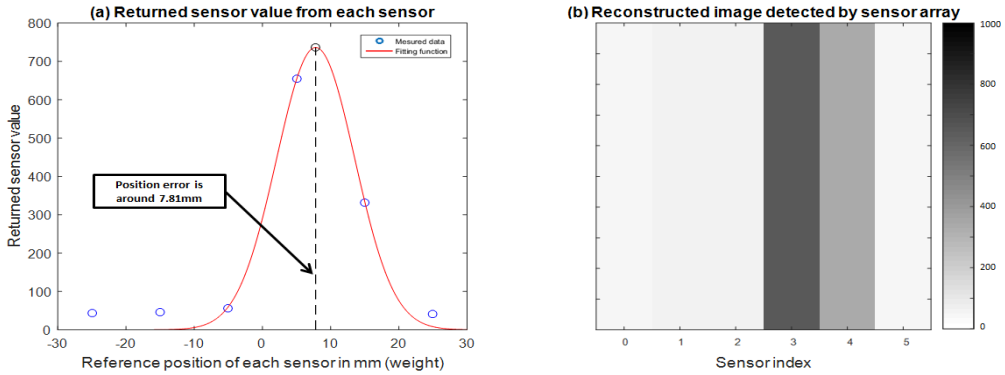
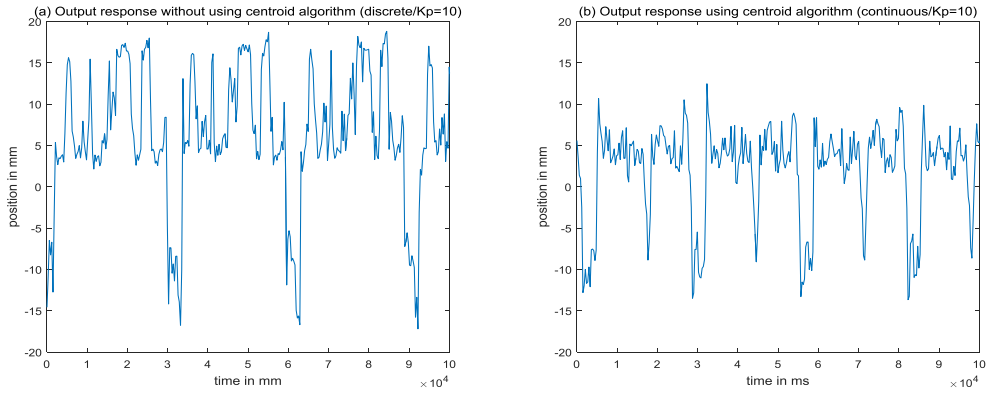


Figure 4.3: Analogue analysis method for locating centre of line pattern

In order to verify the advantage of this method, the control output a simple P controller in time domain and frequency domain will be acquired and can be compared which is shown in figure 4.4 and figure 4.5 respectively.



Note:

1. the statistic analysing of digital method without centroid algorithm being used shows that the mean value is around 6.3991mm and the RMS value is approximately 8.2402mm;
2. the statistic analysing of analogue method with centroid algorithm being used shows that the mean value is around 2.360mm and RMS value is around 5.6068mm;

Figure 4.4: Comparison of output of speed control and pulse width control in time domain

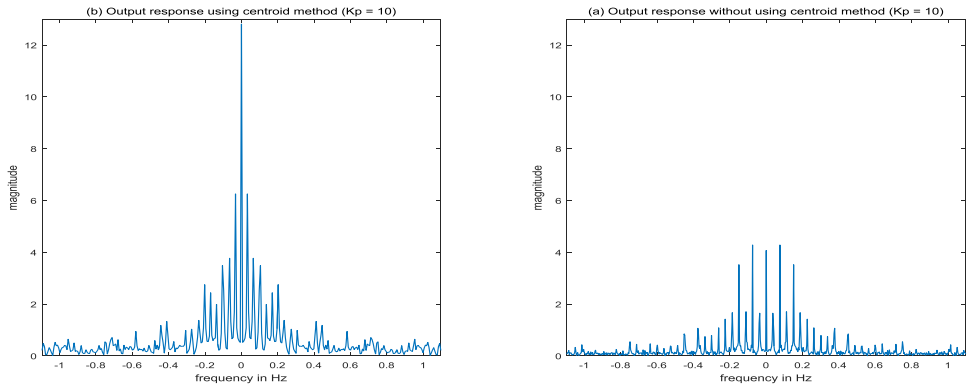


Figure 4.5: Comparison of output of speed control and pulse width control in frequency domain

Obviously results in time domain and frequency domain illustrate that by using the analogue method with centroid algorithm, the vehicle system will have significantly less steady state error and wobbling, and the output results are closer to the desired theoretical output as shown in figure 3.4. Hence it is concluded that the analogue method with centroid algorithm being used can give better performance.

4.2. Sensing Obstacle Ahead

The implementation of obstacle avoidance aims to avoid the vehicle system colliding with obstacles on the line track, which is supposed to be a necessary function in most real-time applications. The core principle of sensing an obstacle ahead is to measure distance between vehicle and obstacle using a proximity sensor. The vehicle will make certain response, such as turning around, to avoid an obstacle once the measured distance is smaller than a 'danger threshold'. Literature [32] summarized 2 popular proximity sensors, termed as IR sensor and US (ultrasonic) sensor used in various mobile robot research. Generally speaking the obstacle detection of IR sensor depends on the properties of reflected IR light which need prior knowledge of obstacle surface such as whether the surface reflects or absorb IR energies. Hence as is indicated in literature [33], most of these sensors is only good for detecting whether an obstacle is within a fixed distance, instead measuring the distance between vehicle and obstacle, although few of them with high cost may be able to measure distance by using time difference between transmitting and receiving IR beans. This means that the dangerous threshold cannot be easily changed if a low cost IR sensor is used. In contrast, the distance sensing of US proximity sensor depends on time interval between ultrasound sending and ultrasound receiving, which is good for measuring distance between vehicle and obstacles, although the response time is not as fast as IR proximity sensor and the surface properties could sometimes cause the error of distance measuring as well, which can produce ghost echo (error distance) and lost echo (cannot receive reflected sonic) (see figure 4.6).

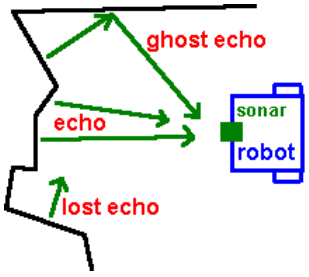
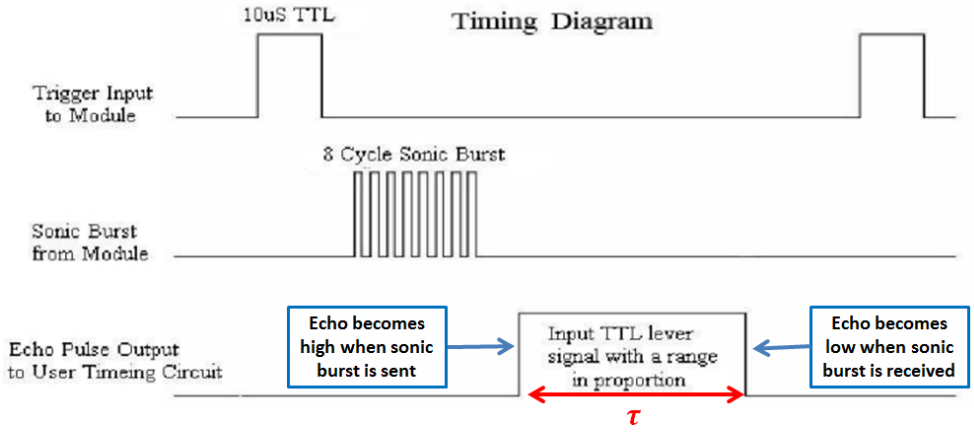


Figure 4.6: Examples that surface of obstacle could cause error of US proximity [33]

In this project a ping US sensor HC-SR04 will be used for obstacle avoidance where the 'danger threshold' may need be modified by software in different cases, while the IR proximity sensor is used for avoiding vehicle falling down from experimental platform where the requirements is to ensure the vehicle is on the ground that will be discussed in next section. The timing diagram of this sensor can be referred to figure 4.7 where the time interval τ indicates the transmission time of sonic burst that is proportional to distance.



Note: τ indicates the high level time of echo pulse output, which is measured in μs.

Figure 4.7: Timing diagram of ping US sensor (HC-SR04) [34]

Hence the distance in cm can be calculated in (9), where the sonic speed in air (with temperature being around 20℃ and density being around $1.2kg/m^3$) can be approximated as 344m/s (34400cm/s). This result can be converted into functions of software code as is shown in figure 4.8 where the loop can avoid error value being returned which can be caused by small ripple of supply voltage when wheels are rotating.

$$distance = \frac{speed\_of\_sound \times time}{2} = \frac{34400cm/s \times (\tau \times 10^{-6})s}{2} \approx \frac{\tau}{58}(cm) \quad (9)$$

```
float distanceSensor(void)
{
   double distance = 0;
   do{
    // generate 10us trig signal for sending sonic burst
    digitalWrite(trig, LOW);
    delayMicroseconds(2);
    digitalWrite(trig, HIGH);
    delayMicroseconds(10);
    digitalWrite(trig, LOW);
    // calculate distance by measuring pulse width of echo signal
    distance = (double)pulseIn(echo, HIGH)/ 58;
   }while(distance < 1.0); // error detection
   if(distance > 400) return 400; // set max detect range to 400 cm
   else return distance; // value in cm
}
```

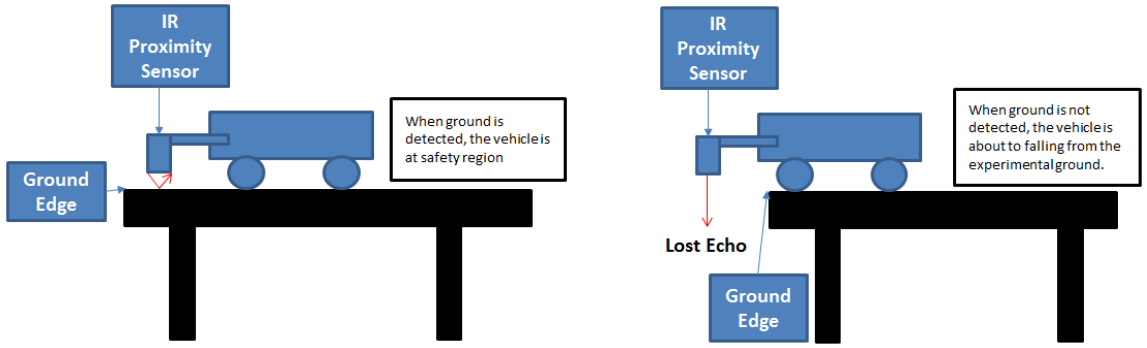
Note: this function is programmed into protection board.

Figure 4.8: Software design for measuring distance using ping US sensor

4.3. Ground Edge Detection/Protection

The ground edge detection aims to ensure the vehicle does not fall down from the experimental platform, which can be achieved by an IR proximity sensor that is good for detecting whether obstacle is ahead (this sensor is seldom used for measuring distance to obstacles). The mechanical installations and scenario of this mechanical protection scheme

can be referred to figure 4.8, where the vehicle is considered as safe if the obstacle/ground is detected by IR proximity sensor (see figure 4.9(b)) and regarded as in danger if the IR sensor cannot detect the ground (see figure 4.9(b)). The experimental result shows that this method works as expected and a simple demo can be referred to the video in attached CD.



(a) Safe case (b) Dangerous case

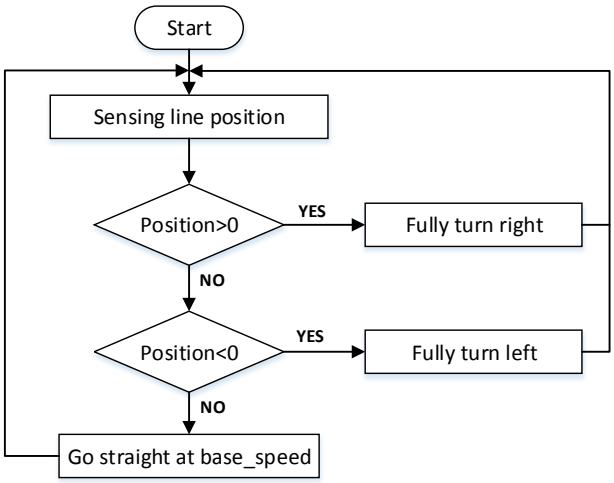
Note:
1. The built in distance threshold IR proximity sensor is around 20cm, i.e. echo lost will occur if distance to obstacles/ground is larger than 20cm;
2. The red arrow refers to the IR light beam;

Figure 4.9: Scenario of ground edge detection

Chapter 5: Design and Evaluation of Control System

5.1. Design and Evaluations of On/Off Controller

On/off controller is one of simplest forms of feedback controller, which is commonly used in refrigerator and heating systems. In this project, this controller can be achieved by the rules that when sensed position of line pattern is larger than 0 (the vehicle is on the left hand side of line pattern), then fully turn right the vehicle; on the other hand, when sensed position of line pattern is smaller than 0 (the vehicle is on the right hand side of line pattern), then fully turn left the vehicle. Hence the top-down design for controller algorithm can be referred to figure 5.1, where the turn right and turn left action is controlled by a conditional block, which is acted as a switch in on/off controller.



Note:

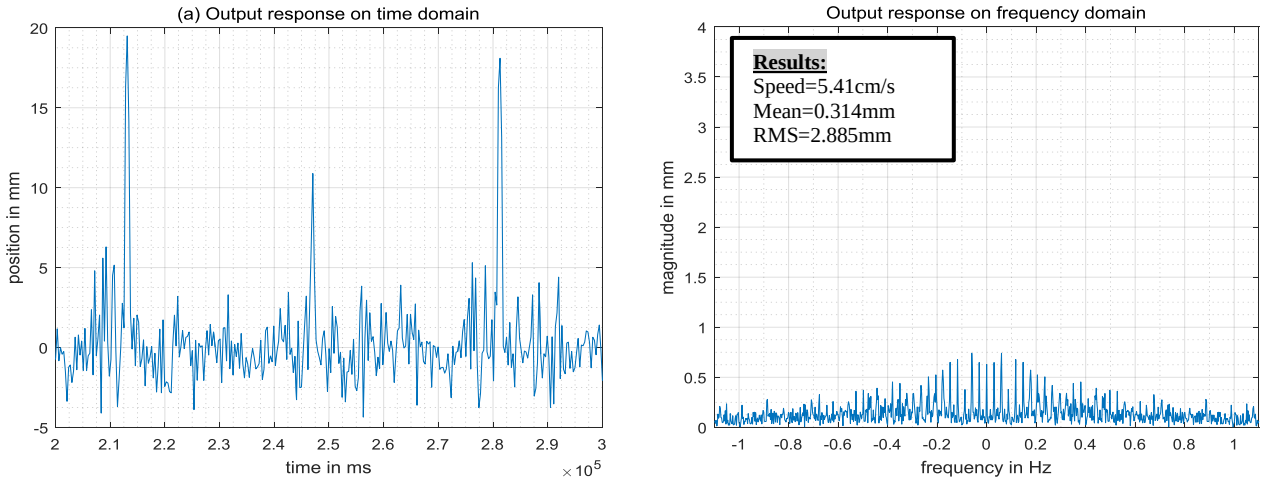
1. The positive value of position indicates that the vehicle is at left hand side of line pattern, which can be referred to figure 3.3.
2. Fully turn right means the left steering wheel is moving forward anti-clockwise while the right steering wheel is static, such that overall, the vehicle is moving forward;
3. The *base_speed* is the vehicle rotational speed when there is no position error and vehicle is following the line pattern perfectly (in this project, this value is set to 50rpm, which is slightly smaller than maximum speed limitations. Hence the maximum vehicle speed can be estimated by (10));

$$vehicle\ speed = \frac{50rpm \cdot \pi \cdot wheel\_diameter}{60s} = \frac{50rpm \cdot \pi \cdot 6.5cm}{60s} \approx \frac{17.02cm}{s} \quad (10)$$

Figure 5.1: Top-down design of on/off controller

By implementing this controller, the output response in time domain and frequency domain as well as relevant quantified parameters including average speed, mean and standard deviation can be referred to figure 5.2. Obviously, from figure 5.2, it can conclude that although the steady state error of controlled output is almost satisfied, the vehicle can is running forward with relative high wobbling (oscillations) which may cause the slow speed. This can be either observed from spikes (overshoot) in output response in time domain, or the relative large of high harmonics in frequency domain or the RMS value (standard deviation) calculated statistically which is around 2.9mm. This result is basically agree with the theoretical output of on/off controller where the persistence huge oscillations of output

controlled variable make it less popular in being applied in controlling mechanical engineering applications.

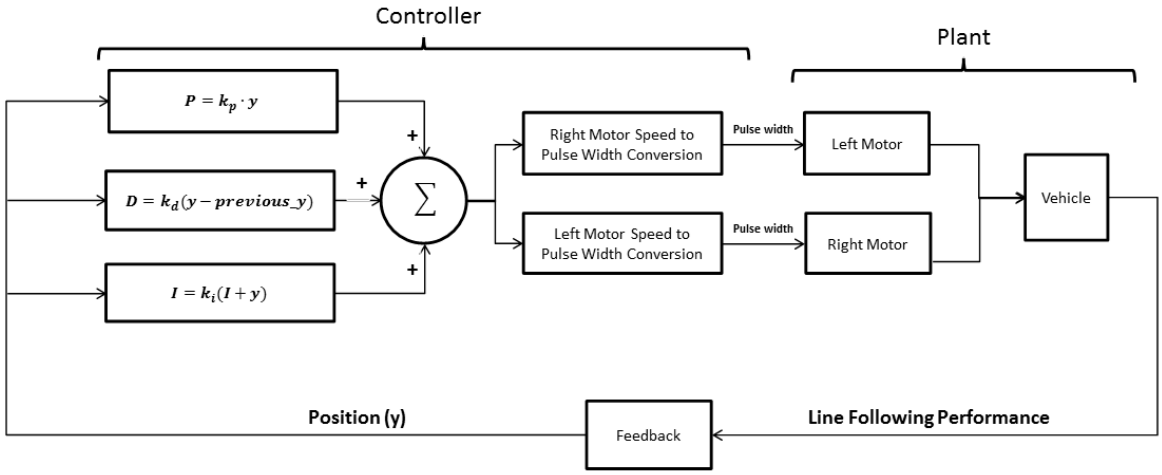


(a) Output response on time domain (b) Output response on frequency domain

Figure 5.2: Output response in time domain using on/off controller

5.2. Design of PID Controller

Although the line following behaviour can be achieved by on/off controller, the performance of controlled output is not satisfactory with slow speed and high oscillations. Obviously the ideal desired controlled output should have small steady state error which is measured by small mean value, small oscillations which are indicated by small RMS value (standard deviation), and high speed which can be indicated by small time interval for running one lap. Theoretically an optimized PID controller can be regarded as a combination of P, D, and I component that is able to achieve this where the P controller aims to reduce achieve basic line following function, the D controller aims to reduce wobbling and oscillations and I controller aims to reduce steady state error. Hence the control diagram can be generalised in figure 5.3, which takes advantages of speed control introduced in section 3.3 (see speed to pulse width converter blocks in figure 5.3).



Note:
1. The speed fed to left and right motors are *base_speed+PID_output_value* and *base_speed-PID_output_value*;
2. The P, PI, and PD controller can be considered as the special case when corresponding coefficients/gain is zero;

Figure 5.3: Generalised block diagram for PID controller

Obviously, the most challenging part for designing optimized PID controller is to find appropriate coefficient/gain for each component. Although there are many methods for finding these coefficients, in this project a popular method summarized by Sitoula [41] is used, which is based on the theoretical effectiveness of each components. In this method, the gain of proportional component (k_p) will first be tuned, which allows the robot to follow the line. The second step is to tune k_d, which aims to reduce oscillations of robotic vehicle. And finally the value of k_i would be tuned which aims to reduce steady state error. This process will be detail discussed in the following sections. However, it should be noted that the tested coefficients of PID value cannot be over large, which will cause huge calculation burden on microcontroller that would reduce sampling frequency and result in aliasing issues. This means that the output position logged in SD card cannot reflect the vehicle wobbling correctly. Additionally the calculations with overlarge values can cause the results to become invalid, which can be caused by the intrinsic numeric limits of float/double value type.

5.2.1. Tuning Coefficient of P Controller (k_p)

By setting coefficient of k_i and k_d to 0 and base speed to 50rpm, the general controlled output response when k_p varies from 0 to 400 can be measured (in this test, the controlled response when k_p is 1, 10, 20, 30, 40, 50, 70, 120, 180, 240, and 350). To reflect the output at different k_p, 6 output responses in time domain are particularly selected, which are shown in figure 5.4. In addition, the statistical parameters for measuring output performance including speed, mean and RMS at different k_p are plotted in figure 5.5.

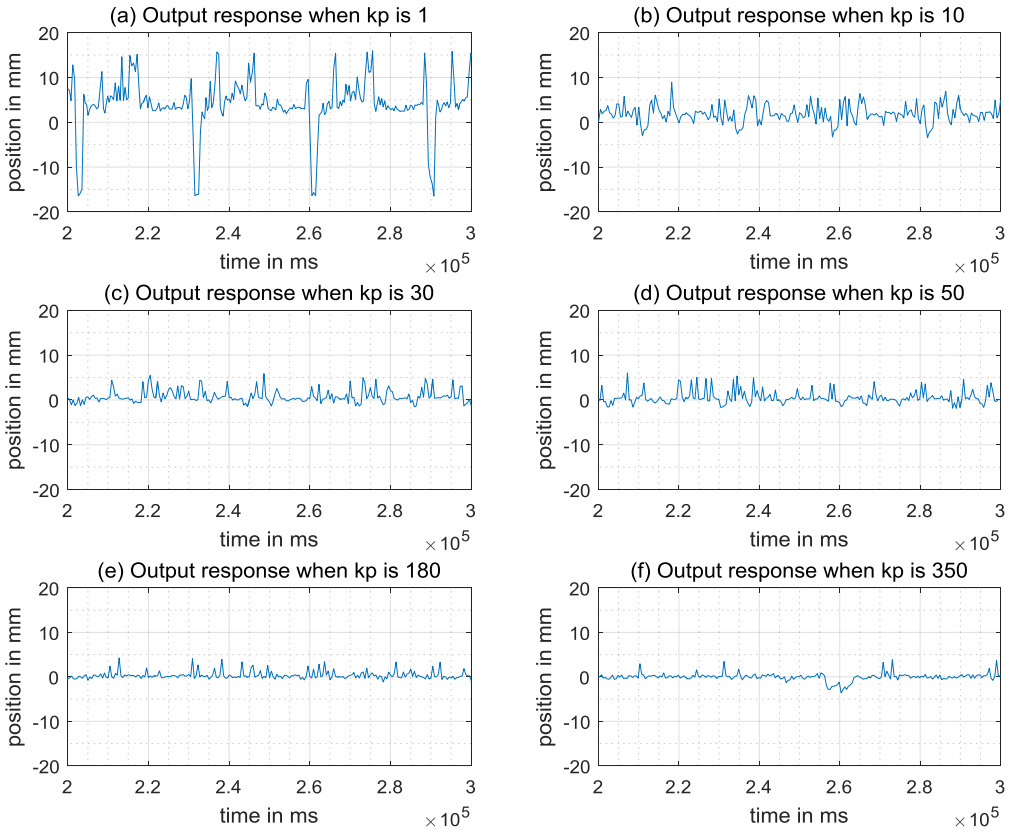


Figure 5.4: Output response in time domain for tuning coefficient of P controller (k_p)

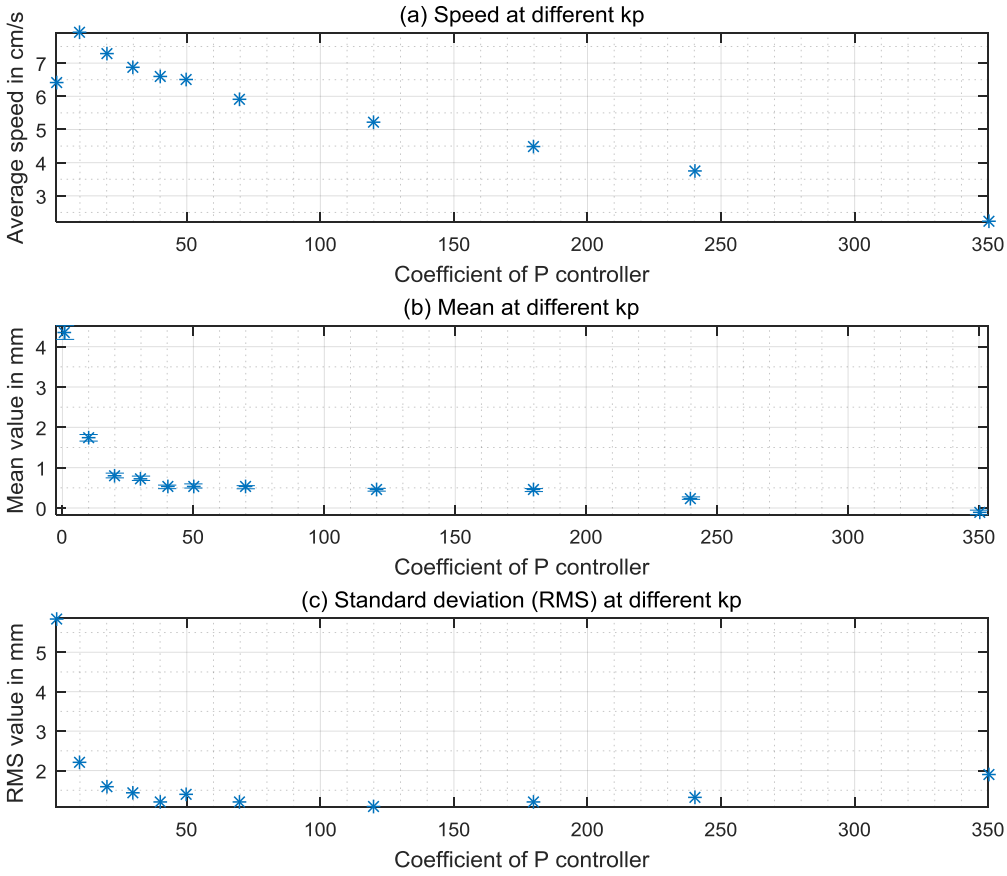


Figure 5.5: Speed, mean and RMS at different k_p

Theoretically the speed should be as large as possible while the rms and mean should be as small as possible that is closed to 0. According to this principle, figure 5.5 shows that the speed reaches as a peak when k_p is at around 10 at which the mean and standard deviation do not reach bottom. However, these issues are supposed to be solved by adding D controller component which theoretically aims to minimize wobbling (i.e. minimise rms value) and I controller which theoretically aims to eliminate steady state error (i.e. the mean value in this case). The tuning of these two controller components will be discussed in the following sections.

5.2.2. Tuning Coefficient of D Controller (k_d)

From the previous section, the approximated optimal coefficient of P controller can be located at around 10, which will be used as control system design. Similar, the coefficient of the D controller can be also found using same method. By setting coefficient of k_p and k_i to 10 and 0, and base speed to 50rpm, the output control response at different k_d are recorded in figure 5.6 and the statistical parameters for measuring output performance are summarized in figure 5.7 as well.

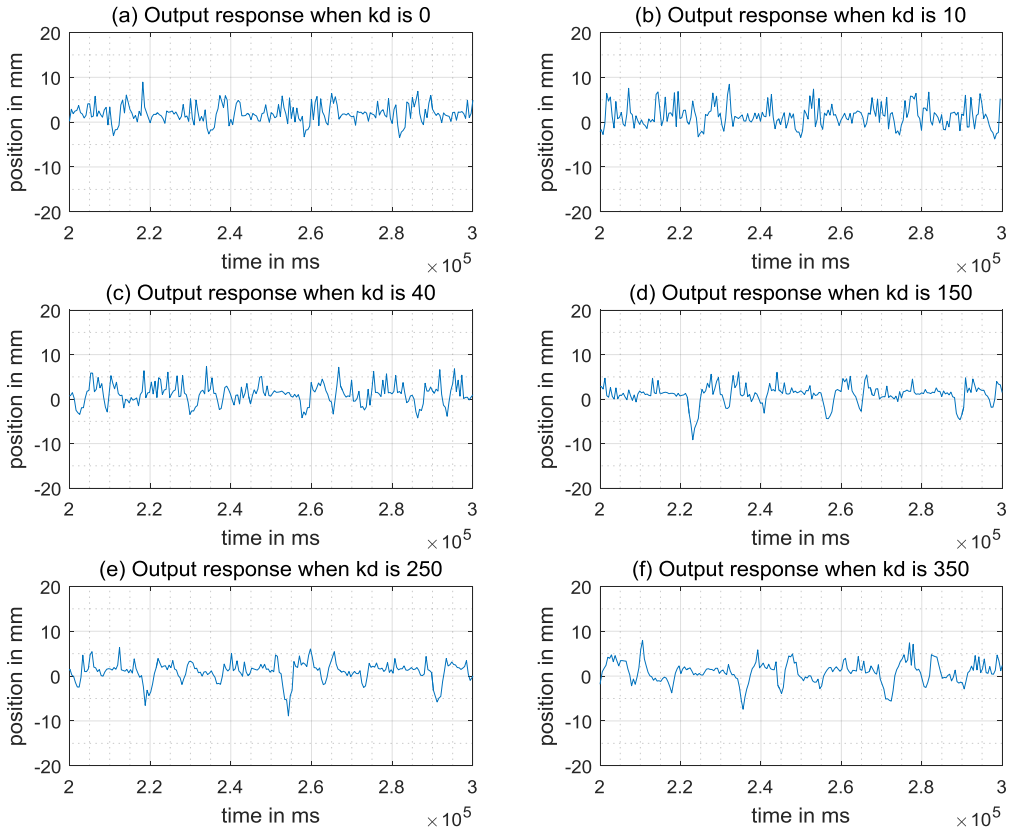


Figure 5.6: Output response in time domain for tuning coefficient of D controller (k_d)

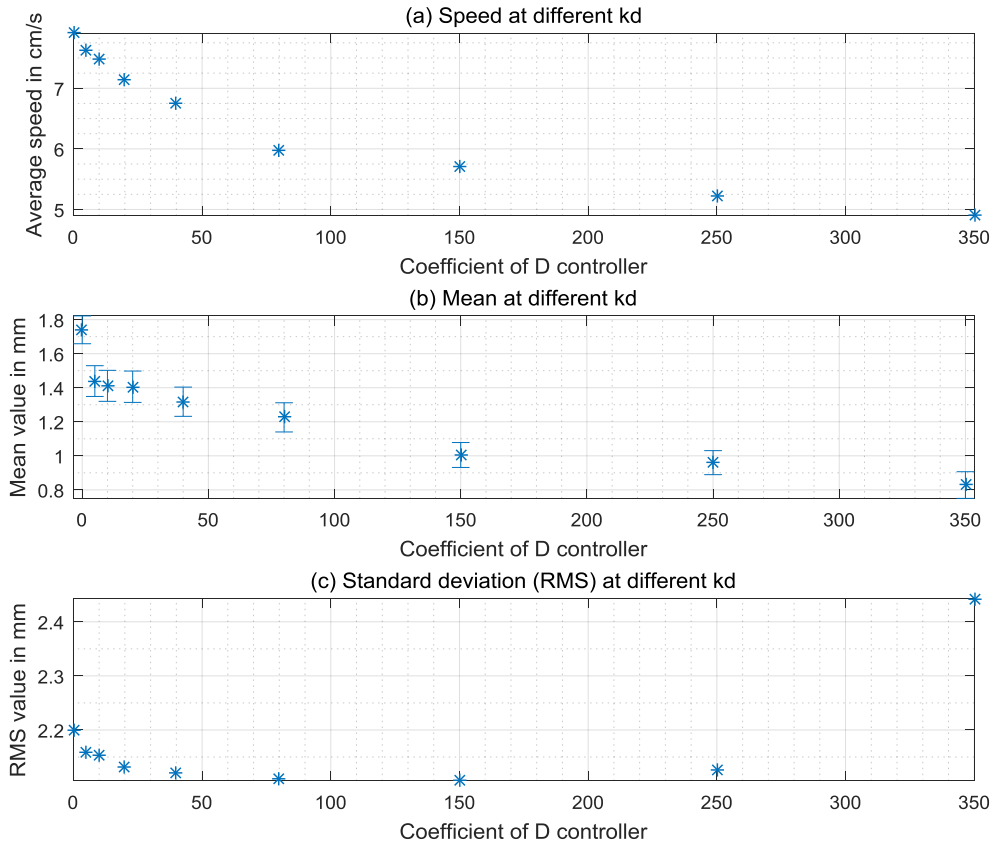


Figure 5.7: Speed, mean and RMS at different k_d

Since theoretically the D controller aims to reduce the degree of wobbling, more focus will put on analysis of figure 5.7(c), where the standard deviation shows the how robot wobbles at different k_d. However, some attentions may need be paid on speed of vehicle at different k_d, which can be shown in figure 5.7(a). Figure 5.6 and figure 5.7(a) illustrate that with increasing of k_d, the speed of vehicle will reduce and in this situation the robotic vehicle may over-respond to the error of position causing unwanted overshoot on output response (see figure 5.5(d – f)). To compromise the speed and standard deviation, the D coefficient is selected to be 20 (relative high speed and small RMS). Although the mean value when k_d is 20 is relatively large, it can be reduced by I controller implemented in following section.

5.2.3. Tuning Coefficient of I Controller (k_i)

The final step for tuning PID controller is to find optimal coefficient of I controller, which aims to minimize the steady state error that is indicated by average value. Obviously this coefficient cannot be overlarge which may cause system overresponse to position error. The output response on time diagram of selected k_i are referred to figure 5.8 and the parameters for evaluating system performance are presented in figure 5.9. Although it may be hard to evaluate the output response on time domain at different coefficient, the optimal k_i could be easily located in figure 5.9 which is around 0.007 at which the speed reaches peak value and average value is most closed to 0. However with further increasing of this value, the speed and mean value will reduce continually indicating the system is now over responding to error which obviously is unwanted.

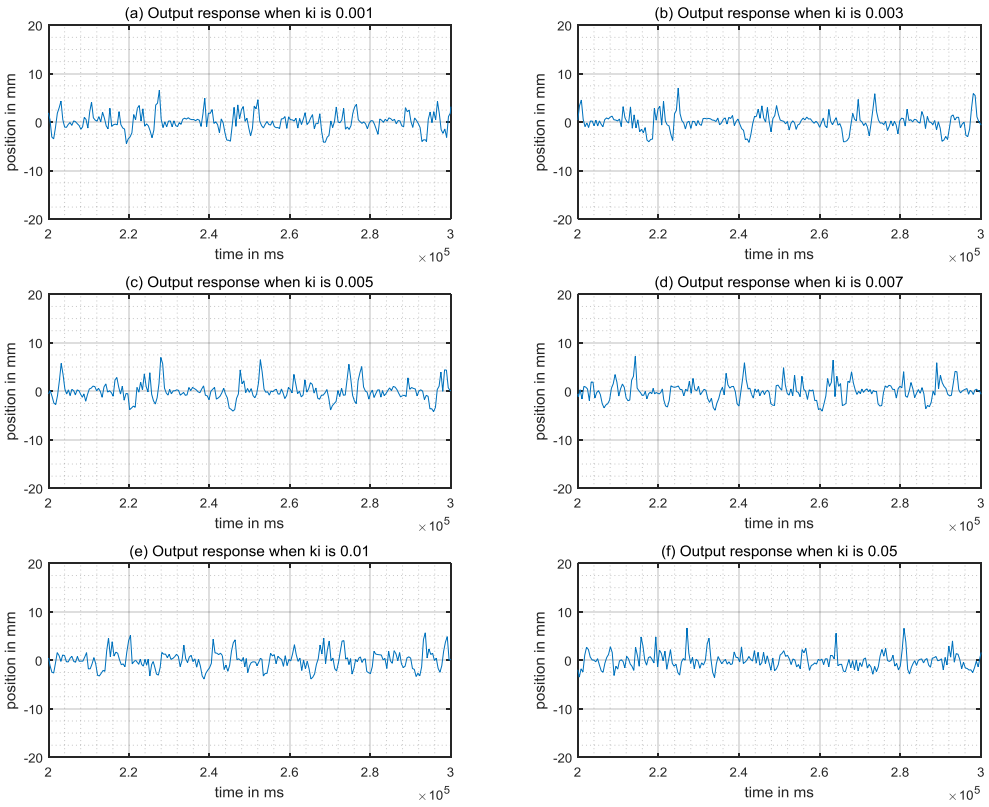


Figure 5.8: Output response in time domain for tuning coefficient of I controller (k_i)

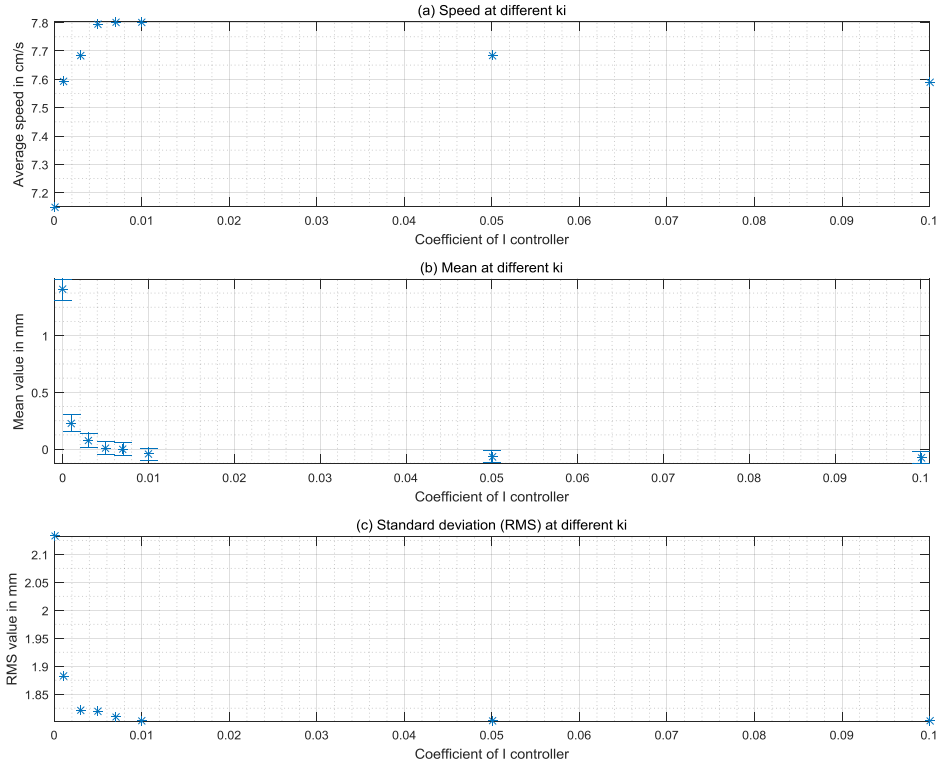
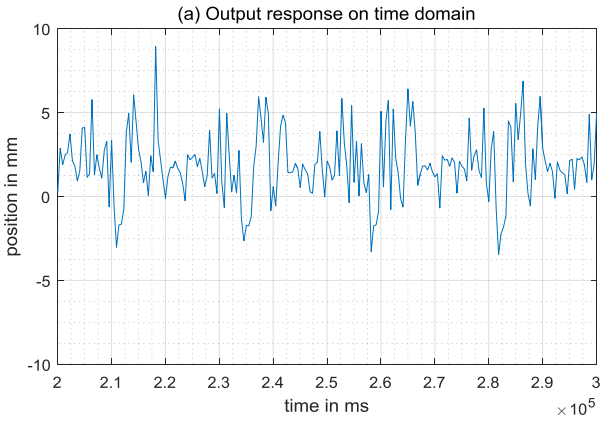


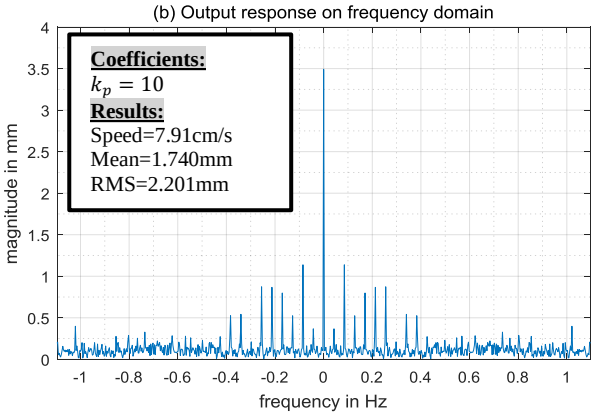
Figure 5.9: Speed, mean and RMS at different k_i

5.3. Design, Evaluations and Comparisons of P, PI, PD, and PID Controller

From section 5.2, the optimal coefficients of PID controller for the designed test environment have been found which are 10, 20 and 0.007. In this section, the output response P, PI, PD, and PID controllers are compared which can be designed by assigning optimal coefficients to related parameters and setting unrelated parameters to 0. The measured output responses of P, PI, PD and PID controller on time domain and frequency domain are referred to figure 5.10 to figure 5.13. For demonstrating the superiority of PID control strategies, the fundamental on/off controller will also be compared which can be seen in figure 5.2.

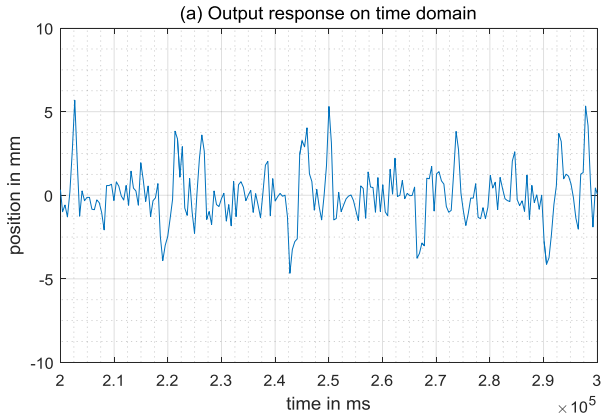


(c) Output response on time domain

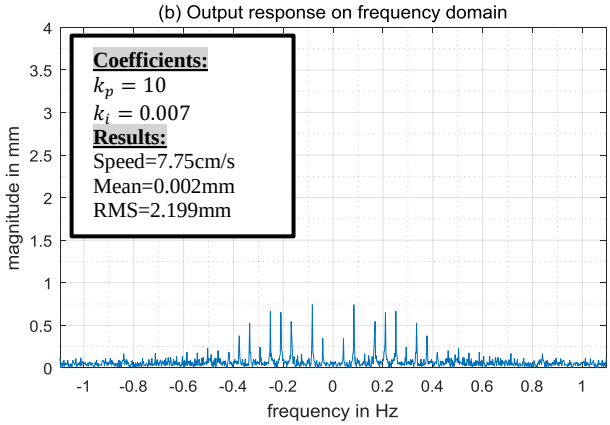


(d) Output response on frequency domain

Figure 5.10: Output response of P controller

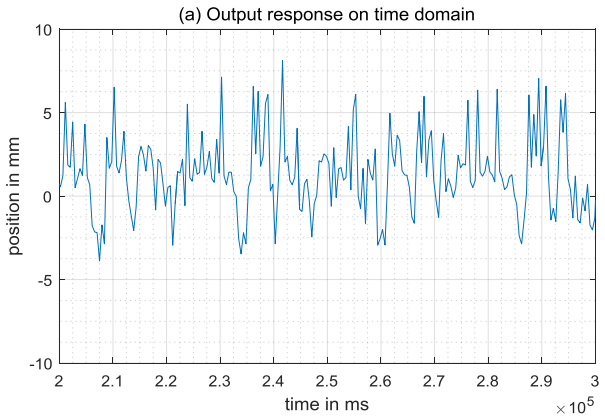


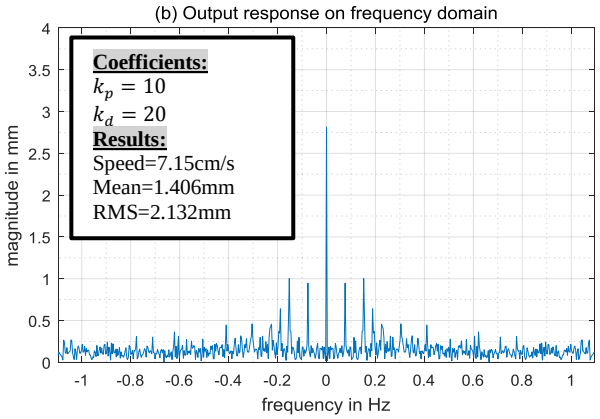
(a) Output response on time domain



(b) Output response on frequency domain

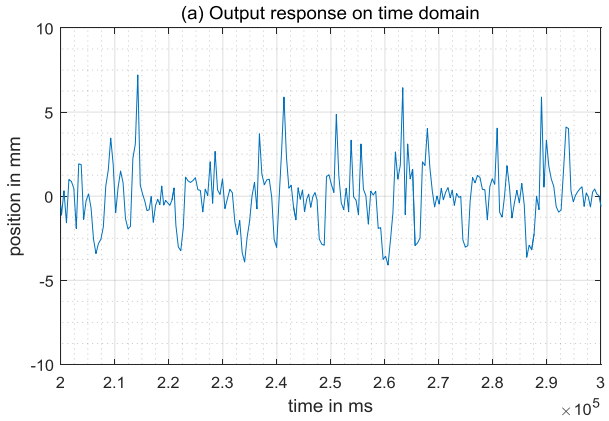
Figure 5.11: Output response of PI controller

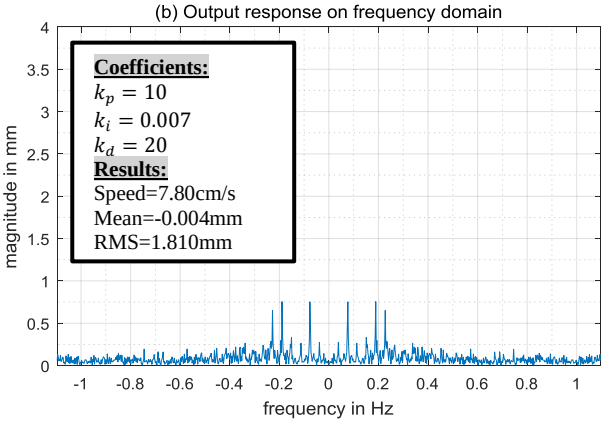




(a) Output response on time domain (b) Output response on frequency domain

Figure 5.12: Output response of PD controller





(a) Output response on time domain (b) Output response on frequency domain

Figure 5.13: Output response of PID controller

By using the idea of variable control method, the strategies for building best control system can be concluded as follows by comparing experimental results shown in figure 5.10 to figure 5.13. Further observing and comparing details of visual effects of each control strategies can be referred to demo videos in attached CD.

- **The performances of PID type controllers are better than on/off controller:**
  The advantages of PID type controllers including P, PI, PD, and PID controllers are mainly reflected by less system wobbling/oscillations while vehicle is running. This can be demonstrated by comparing output responses in frequency domain of figure 5.10 to figure 5.13 with figure 5.2 where the output response of on/off control has less high harmonics than tuned PID type controllers.

- **Minimize mean value for reducing steady state error:**
  The strategy for minimizing steady state error can be demonstrated by comparing figure 5.10 and figure 5.11 with figure 5.12 and figure 5.13. This comparison illustrates that the purpose of I component is to reduce the steady state error which can be demonstrated by almost no DC component of frequency spectrum of PI and PID controller.

- **Minimize RMS value for reducing degree of wobbling:**
  The strategy for reducing degree of wobbling can be demonstrated by comparing figure 5.10 and figure 5.12 with figure 5.11 and figure 5.13. This comparison illustrates that the

role of D component is to reduce system wobbling which can be demonstrated by the small quantity of higher harmonics of PD and PID controlled system.

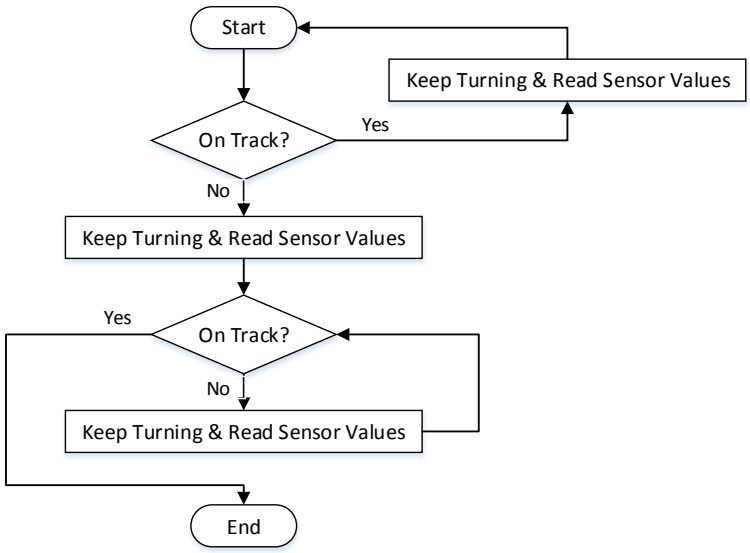
- **Increase average speed of vehicle:**
  The experimental results show that the speed of designed vehicle is highly depended on whether system is over-responding to the position error which is determined by the coefficients of P, I and D components. Basically, the vehicle which is over-responded to the position error can be considered as a non-optimized controller with relative slow speed, since more time will be wasted for system to over-respond to output error.

Chapter 6: Advanced Line Following Vehicle

Based on the built simple line following robot, the advanced line follower improved in this chapter aims to develop functions for responding obstacles ahead and searching target in a simple maze. These functions can be considered as necessary in many real line follower applications introduced in first chapter, which will be discussed later. However points need be noticed is that the algorithms discussed in this chapter can only be used to demonstrate the principal ideas, which still need be improved if put into practical applications and the practical scenario is always much more complicated than that described here.

6.1. Obstacle Avoidance

The feature of obstacle avoidance allows a vehicle to make response once an obstacle ahead is detected. In this project this response was implemented by suspending the line following algorithm and turning the vehicle around. As described in the introduction section, the feedback control strategies are the main focus of this project, by which the system is capable of responding to external disturbance and eliminating noise. Hence while vehicle is turning around, a navigation system is required for telling the robot what position it is in every time instant. In many research such as the mini line follower robotic system developed by Osorio [39], a digital compass module is used for feeding back instantaneous position information. However since one goal of this project is exploit limited information provided by limited low cost sensor infrastructures, the line pattern detected by line following sensor array would be an ideal reference sign for indicating the instantaneous vehicle position. The designed algorithm for turning the vehicle around can be described by flow chart shown in figure 6.1 which will be called once the obstacle is detected. Generally this algorithm can be summarized as the fact that the steering wheels will keep rotating in same direction for driving vehicle turning around until the sensor array sees the line for the second time.



Note:
1. The 'keep turning' behaviour indicates when two steering wheels are rotating in same rotational direction such that the vehicle is driven to turn around;
2. The vehicle will keep turning when first see the line pattern, while it will exit routine when line following sensors see the line twice;

Figure 6.1: Top down design for implementing obstacle avoidance

The experimental results show that this algorithm works in most cases whose visual effects could be referred to the demo video of attached CD. However failure may occur when the vehicle is running on a badly designed surface where the sliding frictions (disturbance) may cause vehicle go out of track and it cannot see the line for second time. This means that using

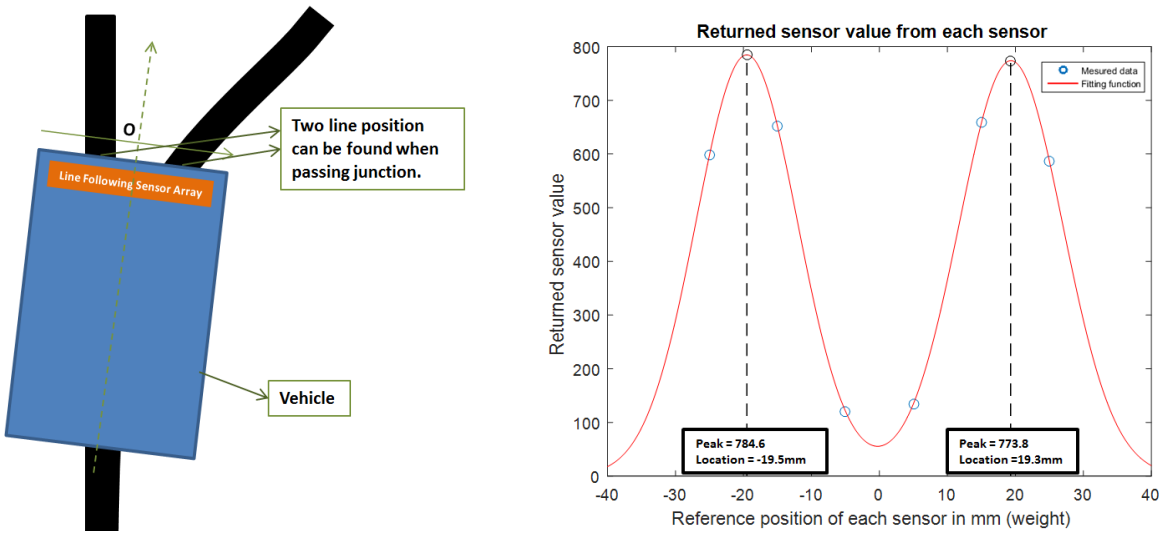
line pattern as a reference for turning vehicle can only be considered as a crude navigation method and a compass module may be used when a more accurate obstacle avoidance feature is needed.

6.2. Simple Maze Solving Robot

Design of maze solving algorithms is continually a popular topic in computer science. Typically, in different environment and testing maze, the algorithms can have huge difference and there are many kinds of maze solving algorithms developed. In this project, however, a simple maze solver with only two diverges is developed for searching for a target by trying each path sequentially in line maze.

6.2.1. Junction Recognition

Junction recognition is always an essential feature in a maze solving vehicle which aims to distinguish normal line track pattern and a junction line pattern. In this project, a specially designed junction pattern with two diverges is used which can be considered as the simplest case in maze pattern. Obviously, the most challenging part for achieving this feature is to distinguish these two cases from the 6 returned sensor values. This can be demonstrated by example returned sensor values and corresponding positions can be explained in figure 6.2 whose result is modelled by a linear combination of two Gaussian functions, by which the position of centre of line patterns can be located at -19.5mm and 19.3mm. Obviously these results are different from the one calculated by centroid method when vehicle is on single line whose sensor returned value is modelled by single Gaussian function and the result is around -0.04mm.



(a) Vehicle position (b) Returned sensor values

Figure 6.2: Returned sensor values at different position

Hence in order to identify whether the vehicle is on the junction, the rule that *whether two end sensors detects the black line pattern while one of the other sensors detect the white pattern* is used. This means that when returned values of two end sensors are larger than predefined threshold while one of returned values of middle sensors are smaller than threshold, the vehicle will fully turn left/right until no more than one end sensor sees the black line pattern (the option of turning right or left may be determined by a specific maze

solving algorithm). In this way a simple line following algorithm may be applied after vehicle running on the single line path.

6.2.2. Maze Solving Algorithms

The maze solving algorithms aims to drive the robotic vehicle to find the target by searching and checking different possibilities. Although there are many type of line maze pattern in current research, in this project, a specific maze pattern containing only one type of junctions with only one converges (input) and two diverges (output) are used, which can be shown in figure 6.3 where the green blocks marked by obstacle indicates 'dead end', i.e. the corresponding path is not correct and another try is needed, and the big black region refers to the target.

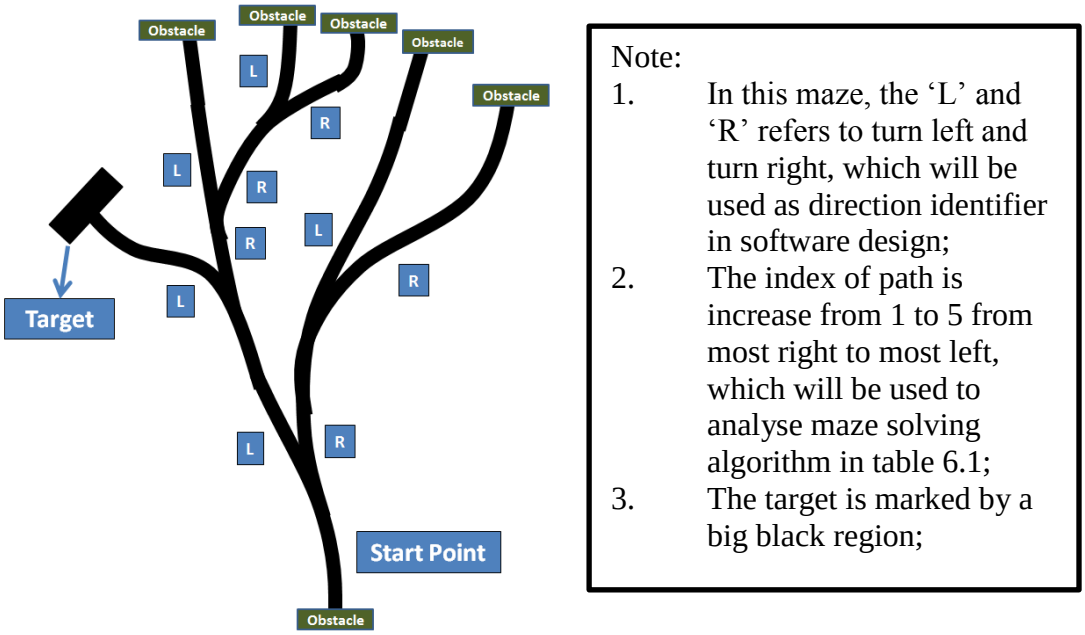


Figure 6.3: Simple tree structure line maze for developing maze solving algorithm

Generally speaking, the maze solver is designed based on the previously developed simple line following controller, obstacle avoidance feature and junction recognition algorithms. As is shown in figure 6.3, the robotic vehicle starts from blue block marked 'start point'. When recognising a junction pattern, the vehicle will *always turn right* first and try the corresponding path. If the path has a dead end, the vehicle will turn around and go back to the start point. This process will be repeated and continue such that the vehicle can search all paths. Once the target region is detected the vehicle will stop running. Hence according to this rule, the turning directions at each junction can be summarized in table 6.1.

Table 6.1: Summary of turning direction at each junction

Path	First Junction	Second Junction	Third Junction	Fourth Junction
1	Right	Right	Undefined	Undefined
2	Right	Left	Undefined	Undefined
3	Left	Right	Right	Right
4	Left	Right	Right	Left
5	Left	Right	Left	Undefined
6	Left	Left	Undefined	Undefined

By observing table 6.1, a simple rule for determining whether turning right or left can be summarized that *the vehicle will only turn left once all previous paths of next junctions are turning left or undefined (i.e. not turning right)*. For example, at second junction of path 6, the vehicle will turn left, which occurs at the condition when third junction of path 5 is turning left. Hence a memory will be needed for avoiding the vehicle searching the same path with dead end twice that can reduce maze solving efficiency, i.e. spending more time for solving maze. Specifically, this logic can be described by figure 6.4 and figure 6.5 respectively where functions of *obstacle_ahead()* and *on_the_junction()* can be implemented based in section 6.1 and 6.2.1 respectively and the array of *Maze_Memory[]* allows vehicle to remember the path tried previously whose element will be initialized to 'R', indicating always try right direction first when passing the junction.

```
bool Maze_Solver(void){
  // first detect whether there is an obstacle ahead
  if(obstacle_ahead()){
    direction = !direction;
    turn_around();
    if(direction == false) {
      maximum_index = junction_index; // record the maximum junction index
      junction_index = 0; // reset junction index and vehicle will back to origin
    }
   }

  if(on_the_juction() == true && direction == true){
     if(Maze_Memory[junction_index]=='R'){
      if(change_turning_direction(junction_index)) Maze_Memory[junction_index] = 'L';
      while(on_the_junction()) fully_turn_right(); // fully turn right until not on the junction
     }
     else{
      if(change_turning_direction(junction_index)) Maze_Memory[junction_index] = 'R';
      while(on_the_junction()) fully_turn_left(); // fully turn left until not on the junction
     }
  }
  else{
    pid_line_following(); // if not on junction then follow line using PID algorithm
  }
  return true;
}
```

Note:

1. The flag of direction indicates whether the vehicle is leaving or backing start point. In this project, the junction recognition function will be disabled when direction is false (backing to the start point);
2. This function can be called by statement *while(Maze_Solver())* in main function;
3. The presented code aims to explain the methodologies for achieving target search process, hence the routines of target finding process is ignored here;

Figure 6.4: Pseudo code of maze solving algorithm

```
bool change_turning_direction(int junction_index){
  for(unsigned int k = maximum_index - 1; k > junction_index; --k)
    if(Maze_Memory[k] == 'R')  return false; // if one of previous path of next junction is 'Right'
  return true;
}
```

Figure 6.5: Pseudo code of functions for determine whether vehicle need change turning direction when pass same junction in next trial

The final experimental results shows that this maze solving algorithm works as expected which can drive vehicle to search each path efficiently. However, some failures may occur due to the less satisfied mechanical design of caster wheel where the sliding frictions makes the vehicle cannot be fully controlled by steering wheels. This problem may be overcome by replacing the originally manufactured caster wheel with a universal caster wheel such as the one shown in figure 6.6.

Figure 6.6: An example universal caster wheel

A second thing needs be noticed is that the designed algorithm can only be applied in specific case (see figure 6.3). Hence for more complicated situation, an improved maze solving algorithms are desired.

6.3. Vision-Based Line Following Robot

As shown in previous and introduction sections, one key aspect for developing line follower control systems is to sense and locate instantaneous line position accurately. Although in this project, this process can be achieved by 6 IR line following sensor units with optimized algorithm (sensor based), the maximum information of line position that can be exploited is limited. A popular method to extent this limitation is to update robotic vehicles to vision based systems. This means that the line position is detected by analysing and processing an image captured by on-board a low cost camera, which obviously contains much more information than data captured by line following sensors. However, as stated in summary of literature review, this method may also have some disadvantages, such as demand for a high performance microprocessor, complex algorithms for implementing image processing, and low sampling frequency, which may affect the design of a digital controller. Elhady *et al*. [36] demonstrated the basic idea of image processing technique so as to find a line position from a captured image, which can be summarized in the block diagram of figure 6.7.

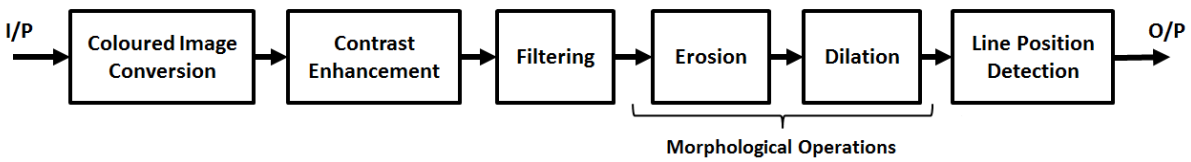
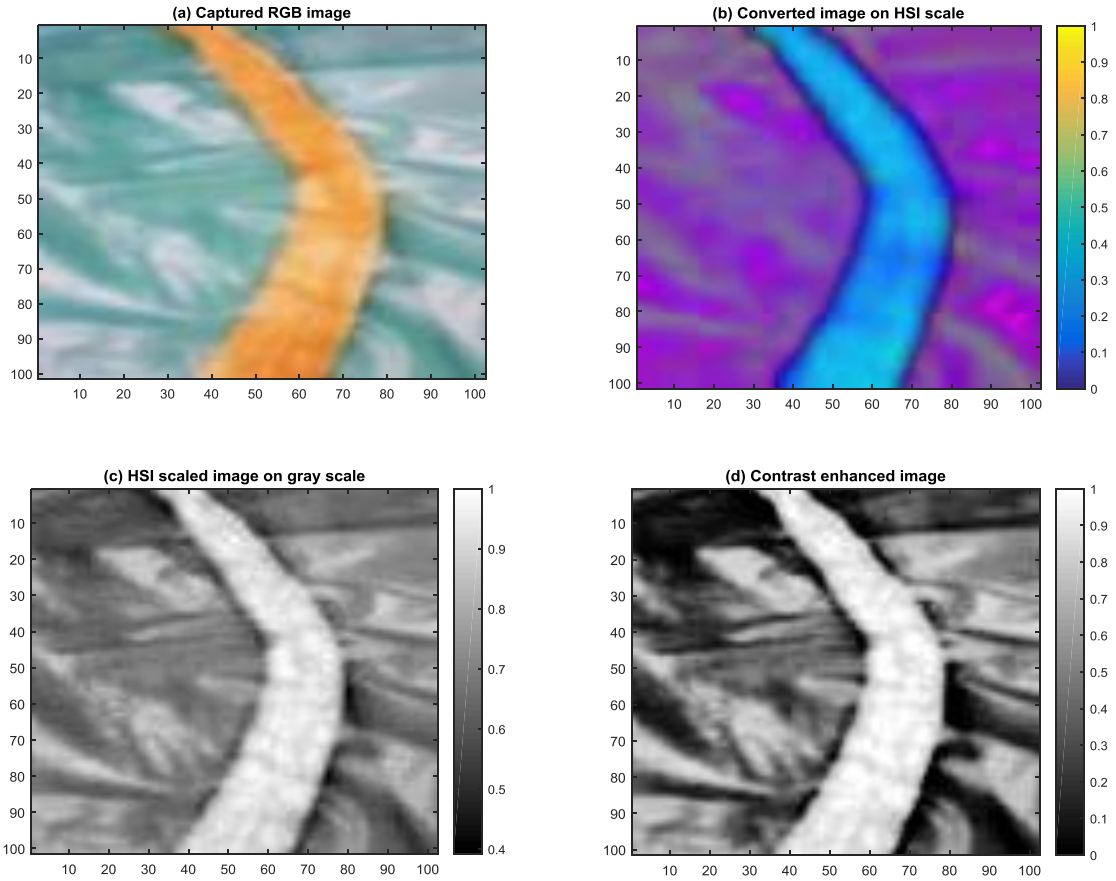
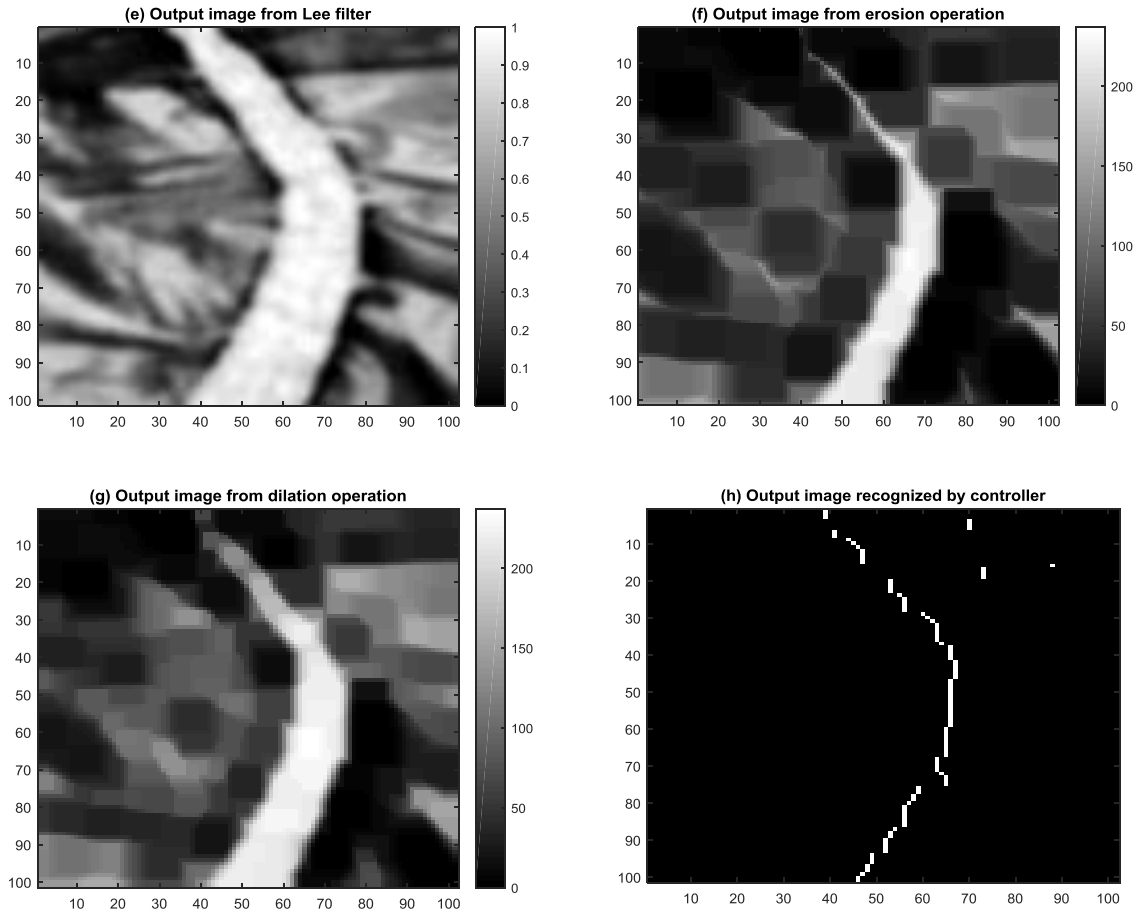


Figure 6.7: Typical image processing process for design algorithm of vision based line following robot

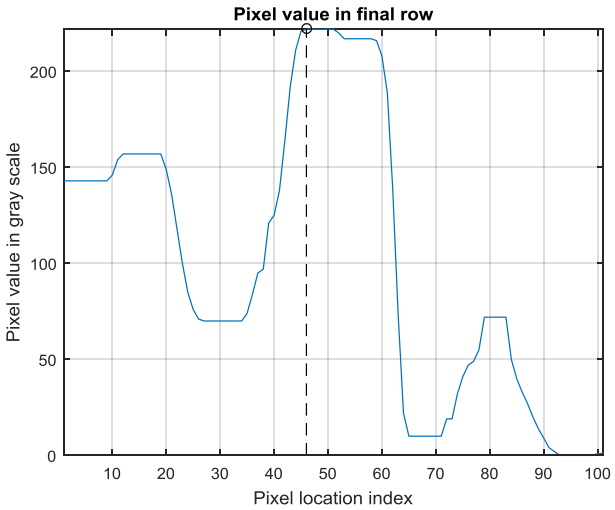
In this project, some software simulations for illustrating this method have been achieved. To demonstrate this image processing technique, a low quality RGB image [36] with complex dark background and lighter line pattern is used to simulate the image captured by a low cost on-board camera which is shown in figure 6.8 (a) (this is slightly different to experimental board used in this project.). The first step is to convert original RGB image data to Hue Saturation Intensity (HSI) scale where the useful colour intensity can be extracted from the original data (see figure 6.8(b)). After that the contrast of grey scale of image is enhanced using histogram equalisation method by which the edge between background and line pattern is enhanced (figure 6.8(d)). The third step is to implement image filtering which aims to reduce background while preserving edge information (see figure 6.8(e)). Literatures [36 – 38] suggest a simple image processing filter, termed as Lee filter, commonly used in processing radar image. The filter is used in this step due to its simple algorithm. Then morphologic operations including erosion which aims to darken the background pixels and dilation which aims to make the centre of line patterns larger are applied to locate and enhance the centre of line pattern (see figure 7.3(f) and figure 7.3(g)). Finally, the position of centre of line pattern of each row can be approximated by finding location of peak pixel values in each row (see figure 7.4 as an example of analysing of final row of pixels). By using this method the centre of line pattern of each row can be marked and the route can be detected as is shown in figure 7.3(h). In contrast, in sensor based line follower developed in this project, the location of centre of line pattern is approximated from mean value of distributions of returned sensor values.





Note:
1. The simulation of image processing technique for future vision based line following robot shown in these figures are implemented in MATLAB;
2. The implementation of Lee filter is achieved in MATLAB code contributed by Mianowski [37];

Figure 6.8: Output image of each block by using image processing technique in figure 6.7



Note:
1. The index (location) of peak is at 46, which can be considered as the detected centre of the line pattern;
2. The plot of centre of each row of pixels can be referred to figure 6.8(h)

Figure 6.9: Example analysis of locating centre of line pattern using final row of pixels in figure 6.9(g)

Chapter 7: Discussions of Future Works and Review of Project Time Management

7.1. Discussions of Future Works

In this project the fundamental aims and objectives have been achieved. And a fully functional IR sensor based line following robotic vehicle capable of avoiding obstacles and solving simple line mazes has been developed and the effectiveness of various controllers have been demonstrated and compared. However several aspects could be improved. These future works are categorized to the improvements of mechanical infrastructures, the user interface systems and advanced strategies for sensing line positions and vehicle system control. Although these three aspects were not implemented in this project, several background reviews and simulations demonstrate the possibilities of these future works.

7.1.1. Improvements of Mechanical Infrastructures

Although the developed line follower is able to perform fundamental line following using on/off control and PID type control strategies, some failures may occur when vehicle performs advanced line following tasks such as obstacle avoidance and maze solver. As is stated in previous sections, this may be result from the less satisfied mechanical design of original Parallax robotic platform, such as non-universal caster wheels and small platform space for equipping necessary sensors and add-ons etc. These mechanical issues can make huge impacts on performance of controlled output and hence the improvements of design of mechanical infrastructures can be considered as one of options for future work.

7.1.2. Improvements of User Interface Systems

Another future work of this project may focus on the improvement of user interface systems which are important part of hardware infrastructures. In this project the input user interface is implemented via an IR receiver (CHQ1838) and an IR remote controller as is shown in section 2.4. However a better method maintained by Pahuja and Kumar [35] is to replace this user input system by Bluetooth infrastructure. In this way the command can be sent from a customized GUI application on laptop/Mobile phone, where the user interfaces can be more flexible than purchased IR remote controller. Literature [35] also summarized two popular HC series Bluetooth modules, which are HC-05 where master and slave mode can be switched and HC-06 where master and slave mode cannot be switched. Hence for displaying vehicle real-time information, the module of HC-06 with decoder is suggested to be interfaced with a control board whose connections are essentially equivalent to series port line connections between Arduino controllers.

7.1.3. Improvements of Strategies for Sensing Line Positions and Controlling Vehicle System

The final aspect of future work aims to improve the sensor-based line position sensing and control strategies. Although in this project, the vehicle is able to follow a predefined line path with fast speed, small steady state error and small wobbling using PID control strategies, further improvements may be achieved by using a better strategy for sensing and locating instantaneous vehicle position with respect to line pattern. As is stated in figure 3.2, the vehicle position at any time instants can be characterized by position (y) and yaw angle (θ) whose desired values are both 0 as is stated in ideal vehicle positions shown in figure 3.3. However in this project, only position quantity is considered and controlled which is

calculated by averaging 6 sensor returned values with yaw angle being omitted. Hence the analysis of yaw angle can be regarded as part of future work. Basically from the prospective of mathematical theory, it could be predicted that yaw angle can be characterised by variance of 6 returned sensor values, which measures how sample distributions spread out (obviously the desired variance of sensor array values should be as small as possible). Example sensor array values are shown in figure 7.1 when the position is around 0mm (ideal case) while the yaw angles are equal to 0° (desired angle) and 75° (undesired angle) respectively, which has almost identical mean but significant differences of variance. As a result, not only is the future improved vehicle able to control position, but can control yaw angle simultaneously.

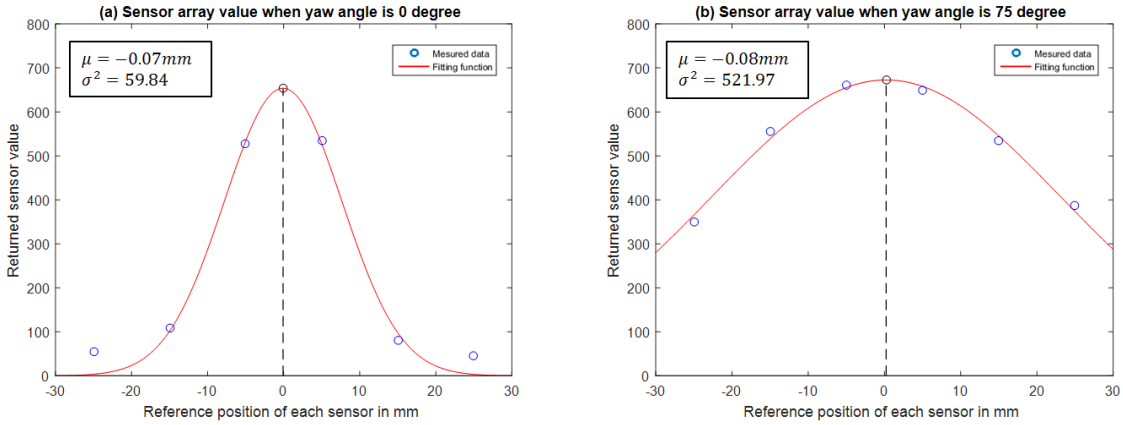


Figure 7.1: Example showing the relations between yaw angle and variance

Another method to further enhance line location positioning is to update the line follower to a vision based line following robot which is able to sense more environmental information for making different responses. A possible technique for processing captured image by on-board camera can be referred to section 6.3. However a problem that may be introduced is the low performance of the microprocessor. Since huge image information need be processed for an improved line follower, a better processing platform instead of Arduino with faster speed and capability of parallel programing such as a Raspberry Pi is required.

7.2. Review of Time Management

As demonstrated in the first chapter, this project has been divided into four minor stages followed by several secondary aims and objectives. For reference purpose, the comparisons of initial time plan and recordings of actual project progress can be seen in figure A4.1 and figure A4.2 in Appendix 4 presented in the form of Gantt chart where the task items in actual Gantt chart marked by red texts is the ones added latter based on initial plan. Generally speaking, by reviewing and comparing the two Gantt charts, it can be concluded that most of tasks have been finished and the ultimate aim and secondary objectives have been achieved. However since the vehicle development platform was not confirmed and purchased during planning stages, the stages of development obstacle avoidance has been combined into development of IR line follower and maze solver. Additionally, considering the remaining time before project due, literature reviews and simulations on image processing techniques for future vision based line following robot is implemented. Finally in order to prevent impacts of external reasons on project efficiency such as delay of courier delivery of components for unknown reasons, many tasks on actual Gantt chart were conducted simultaneously (in parallel), such as second and third stages, as well as final stages, which has proved to be an effective time management method for enhancing project efficiency.

Chapter 8: Conclusions

The fundamental aims and objectives developed based on project specifications of appendix 1 have been successfully achieved and completed with developed line following robot being capable of performing simple line following and several advanced line following. Considering future applications, since the vehicle is equipped with fully functional UI as well as control systems, it can be served as an educational tool for institutions for demonstrating and comparing different control strategies for control engineering courses student. Beside this, this project can also be used as a starting point for investigating problems of more advanced line following system, robotic and computer visions and machine learnings etc.

In summary, this project was started by developing mechanical and electrical infrastructures of line following robot, the optimal robotic development platform, microprocessors, user interface facilities and various add-ons are selected. After that, a simple line following robotic vehicle system with various control strategies including on/off controller, P, PI, PD, and PID controller have been successfully achieved and output responses of each controller were compared and evaluated. In particular, the coefficients of optimal PID controller were successfully tuned. After that some additional advanced functions such as obstacle avoidance, simple maze solver algorithms, and advanced line position sensing and system controls have been discussed, simulated and investigated. These are considered as possible research proposals for future line follower project.

Although the fundamental aims and objectives of this project are completed, some possible future works were emphasized in this report as well. These include the improvements of mechanical infrastructures, user interface systems especially the command input system, and strategies for enhancing line position sensing and vehicle system control.

Generally speaking the achieved tasks in each stages corresponding to minor project aims, objectives and deliverables can be summarized as below. In particular, some of important deliverables including visual effects of simple line following using different controllers, obstacle avoidance as well as ground edge detection/protection can be also referred to the demo videos in attached CD.

Achieved Tasks in Different Stages:

- **Stage 1: Mechanical Hardware and Infrastructure Construction**
    - Literature reviews and comparisons of different robotic platforms;
    - Design, investigations and construction of basic mechanical and electronic infrastructures including line following sensor arrays, power supply modules, data logger, obstacle avoidance detectors and various LED indicators;
    - Development of friendly HMI based on IR remote controller/decoder/receiver and LCD screen;
    - Development of friendly menu systems (improved in stage 2);
    - Construction of experimental platforms including line pattern for future use;

- **Stage 2: Simple IR Based Line Following Robot**
    - Comparisons and investigations of 2 line sensing methods;
    - Design, develop and evaluate different control strategies;

- **Stage 3: IR Based Maze Solver**
    - Development of fundamental algorithms for obstacle avoidance;

    - Design and develop algorithms for solving simple maze;
    - Literature reviews on advanced maze solver algorithms;

- **Stage 4: Future Development of Advanced Line Following Robot**
    - Proposals on advanced remote vehicle control strategies;
    - Proposals on sensing and locating line pattern by considering yaw angle;
    - Proposals and simulations on image processing strategies of future vision based line follower;

Achieved Deliverables:

- Constructed robotic vehicles with additional facilities for line following, obstacle avoidance, data logging and user interface;
- Simple IR based line following robot with different line position sensing and control strategies;
- Advanced maze solving vehicle with capable of obstacle avoidance;
- Fundamental image processing methods for implementing future vision based line follower;

References

[1] J. Dixon and O. Henlich, "Mobile robot navigation", *Information systems engineering (year 2),* Imperial College London, June 1997, unpublished.

[2] Dale's Homemade Robots, "Line follower robot", October 1999, unpublished.

[3] D. Punetha *et al.*, "Development and Application of Line Following Robot Based Health Care Management System", *International Journal of Advanced Research in Computer Engineering and Technology (IJARCET),* vol. 2, no. 8, pp. 2446 – 2450, August 2013.

[4] A.H. Ismail *et al.*, "Vision-based system for line following robot", Department of Electrical and Electronic Engineering, Universiti Putra Malaysia, *IEEE Symposium on Industrial Electronics and Applications (ISIEA 2009),* Kuala Lumpur, Malaysia, vol. 2, pp. 642 – 645, October 2009.

[5] M.S. Islam and M.A. Rahman, "Design and fabrication of line follower robot", Department of Electrical and Electronic Engineering, Rajshahi University of Engineering and Technology, *Asian Journal of Applied Science and Engineering*, vol. 2, no. 2, pp. 598 – 603, 2013.

[6] R.T. Vannoy II, "Designing and Building a line following robot", ITT-Technical Institute, n.d., unpublished.

[7] A.S.Nath *et al.*, "Implementation of PID control to reduce wobbling in a line following robot", SRM University, Chennai, India, *International Journal of Research in Engineering and Technology (IJRET),* vol. 20, no. 10, pp. 531 – 534, October 2013.

[8] J. Walker, "Project specifications of Robotic Vehicle Construction and Control", Department of Electrical and Electronic Engineering, University of Nottingham, 2015 (a copy of this document can be referred to appendix)

[9] TeachFeast (n.d.) "Line Follower (PD Control)", *Techfeast*, n.d., unpublished.

[10] I. Colak and D. Yildirim, (2009) "Evolving a Line Following Robot to Use in Shopping Centres for Entertainment", *Industrial Electronics, 2009. IECON '09. 35th Annual Conference of IEEE*, pp. 3803 – 3807, November 2009.

[11] M. Jäntech, "Non-linear Control Strategies for Musculoskeletal Robots", Ph.D. Dissertation, Lehrstuhl für Echtzeitsysteme und Robotik, Technische Universität München, München, Germany, October 2013.

[12] J.A. Pandian *et al.*, "Maze Solving Robot Using Freeduino and LSRB Algorithm", Anna University, T.N. India, *International Journal of Modern Engineering Research (IJMER),* pp. 92 – 100, 2013.

[13] R.T. Vannoy II, "Design a Line Maze Solving Robot, Teaching a Robot to Solve a Line Maze", *Documentations of presentation from Pololu Robotics and Electronics*, April 2009, unpublished.

[14] S. Sakib *et al.*, "Maze solving algorithm for line following robot and deviation of linear path distance from non-linear path", Department of Electrical and Electronic Engineering, Bangladesh University of Engineering and Technology, *16th International Conference Computer and Information Technology*, Khulna, Bangladesh, pp. 478 – 483, March 2014.

[15] Google, (n.d.) “Google Self_Driving Car Project”, [Online], Available at: <https://www.google.com/selfdrivingcar/> [Accessed on 5 February 2016]

[16] D. Calin (2014), “47 Programmable Robotic Kits”, *Intro-robotics*, [Online], Available at: <http://www.intorobotics.com/47-programmable-robotic-kits/> [Accessed on 3 March 2016]

[17] D.G. Ullman, “The Mechanical Design Process”, McGraw Hill Higher Education, 4th ed., pp. 81 – 95, 2010.

[18] Official documentations for Arduino Robot, [Online], Available at: <https://www.arduino.cc/en/Main/Robot> [Accessed on 5 February 2016]

[19] Official documentations for Pololu 3pi Robot, [Online], Available at : <https://www.pololu.com/product/975> [Accessed on 28 February 2016]

[20] Official documentations for Parallax Robot, [Online], Available at: <https://www.parallax.com/product/32335> [Accessed on 28 February 2016]

[21] “Arduino Robot Control Board – Uploading Issues With Bootloader And AVRISPmkII”, Arduino, [Online], Available at: <http://forum.arduino.cc/index.php?topic=207908.0> [Accessed on 9 February 2016]

[22] R. Light, “Notes from lecture 10 of embedded computing (H63ECH)”, Department of Electrical and Electronic Engineering, University of Nottingham, 2015.

[23] Battery University (n.d.) “Is Lithium-ion the idea battery”, [Online], Available at: <http://batteryuniversity.com/learn/article/is_lithium_ion_the_ideal_battery> [Accessed on 5 March 2016]

[24] R. Schmidt (June 2015) “’Smart’ batteries not so smart”, [Online], Available at: <http://www.inspirepilots.com/threads/smart-batteries-not-so-smart.3612/> [Accessed on 02 March 2016]

[25] Official Documentations from Arduino, (n.d.), “Hello World”, tutorial on applications of LCD screen, [Online], Available at: <https://www.arduino.cc/en/Tutorial/HelloWorld> [Accessed on 06 March 2016]

[26] B. Earl, “Adafruit data logger shield”, *Adafruit learning system*, January 2015, unpublished.

[27] S.W. Smith, “Chapter 13: Continuous signal processing”, in *The Scientist and Engineer’s Guide to Digital Signal Processing*, San Diego, CA, the United States of America :California Technical Published, December 1997, pp. 243 – 255.

[28] Official documentations from MathWorks, [Online], Available at: <http://uk.mathworks.com/help/matlab/ref/trapz.html?requestedDomain=www.mathworks.com> [Accessed on 19 March 2016]

[29] Official tutorial and documentations of Parallax Arduino Robot [Online], Available at: <http://learn.parallax.com/node/179> [Accessed on 05 October 2015]

[30] Official documentations of QTR-8RC Reflectance Sensor Array, *Pololu Robotic and Electronics*, n.d. unpublished.

[31] C. Fosu *et al.*, “Determination of Centroid of CCD Star Images”, *ISPRS XXXV Congress*, Istanbul, Turkey, 2004.

[32] T. Mohammad, "Using Ultrasonic and Infrared Sensors for Distance Measurement", *International Journal of Mechanical*, Aerospace, Industrial, Mechatronic and Manufacturing Engineering, vol. 3, no. 3, 2009.

[33] Society of robots, "Infrared vs. ultrasonic", [Online], Available at: <http://www.societyofrobots.com/member_tutorials/node/71> [Accessed on 18 October 2015]

[34] Official data sheet documentation of ping US sensor (HC-SR04), n.d. unpublished.

[35] R. Pahuja and N. Kumar, "Android Mobile Phone Controlled Bluetooth Robot Using 8051 Microcontroller", *International Journal of Scientific Engineering and Research*, vol. 2, no. 7, pp. 14 – 17, July 2014.

[36] W.E. Elhady *et al.*, "Implementation and evaluation of image processing techniques on a vision navigation line following robot", *Journal of Computer Science*, vol. 10, no. 6, 2014.

[37] G. Mianowski, (June 2010), Contributed MATLAB code for implementing Lee filter, MathWork file exchange, [Online], Available at: <http://www.mathworks.com/matlabcentral/fileexchange/28046-lee-filter> [Accessed on 13 March 2016]

[38] X. Wang, "Lee Filter for Multiscale Image Deniosing", *2006 8th International Conference on Signal Processing*, Beijing, China, vol. 1, 2006.

[39] C.R. Osorio *et al.*, "Intelligent line follower mini robot system", *International Journal of Computers, Communications and Control*, vol. 1, no. 2, pp.73 – 83, 2006.

[40] A. Visioli, "Practical PID Control – Advances in Industrial Control (AIC)", Springer-Verlag, London, UK, 2006, pp. 1 – 22.

[41] A. Sitoula (June 2012), "PID Tutorials for Line Following", *Lets Make Robots* [Online], Available at: <http://letsmakerobots.com/node/39972> [Accessed on 5 November 2015]

APPENDIX 1: INITIAL PROJECT SPECIFICATIONS

Note: the document of this page is original suggested project specifications which can be referred to [8].

Robotic Vehicle Construction and Control

Project Supervisor: Dr. J.Walker

Robotics is a growing area with more and more applications being develop. One important area is autonomous or robotic vehicles, where the key component is efficient and safe control of the vehicle.

Project Aims

The aim of this project is to construct a very simple robot vehicle and to investigate the effectiveness of different control strategies to control the vehicle to be able to follow a given route, avoid obstacles and some other task.

The outcome of this project could be useful in illustrating to undergraduate students the different performance of on/off, proportional (P), proportional-integral (PI) and proportional-integral-differential (PID) control methods as well as the importance of choosing the correct control parameters, e.g. to obtain a stable output.

Objectives

To meet the initial project aims the following objectives might be appreciated

1. Construct a robotic vehicle from a standard kit including servo motors and an ARDUINO board capable of changing the speed of the motors
2. Given an ability to program the ARDUINO board to monitor the vehicles sensors and control the vehicle motors
3. Adapt the code developed in 2 to enable the vehicle to be controlled in a simple way to accomplish some tasks
4. Investigate the effectiveness of more sophisticated control algorithms
5. Think about hoe to judge the effectiveness of the different control methods and whether this project could be used as a teaching aid to illustrate the importance of understanding control theory

Skills Needed

An ability to program the ARDUINO board (simple C type programming) and some knowledge of control (such as you learned in H62SPC).

References

The robotic car kit recommended is manufactured by Parallax and contains an on-board ARDUINO board see http://www.parallax.com/product/32335

Note: The fundamental aims and objectives in this specification have been achieved within scheduled time with some additional advanced aims and objectives being achieved and discussed.

APPENDIX 2: ARDUINO CONTROLLER CONNECTIONS FOR VEHICLE SYSTEM CONSTRUCTION

The purpose of this appendix is to provide detail descriptions for connections between Arduino controllers and various add-on modules, which will be used as supplementary material for section 2.7, the overview of constructed infrastructures. Beside this, the contents in this appendix can also be used as the guidance for project reconstructions in the future (the functions of each analogue/digital pins on motor board, control board and protection board can be referred to table A2.1, 2.2, and 2.3).

Table A2.1: Descriptions of pins of motor board

Arduino I/O pins	External Modules
A0 to A5	QTR-8 line following sensors (only 6 sensors are used)
D0 (Rx)/D1(Tx)	Serial USART channel for communicating with user interface board
D3 and D4	Trig and echo pin for ultrasonic distance measuring sensor
D5 to D10	LED indicators showing whether line position is detected by sensors
D11	Micro server motor for controlling position of distance measuring sensor
D12	Servo motor for controlling right wheels
D13	Servo motor for controlling left wheels

Table A2.2: Descriptions of pins of control board

Arduino I/O pins	External Modules
D2 to D5	Connected to data pin 7, 6, 5, 4 of LCD1602
D26, D28	Connected to RS and enable of LCD1602
D32	LED indicator for data logging
D18 (Tx1)/D19(Rx1)	Serial USART channel for communicating with motor control board
D33	Connected to IR remote receiving unit (CHQ 1838)

Table A2.3: Descriptions of pins of protections board

Arduino I/O pins	External Modules
D2 to D5	Connected to data pin 7, 6, 5, 4 of LCD1602
D26, D28	Connected to RS and enable of LCD1602
D32	LED indicator for data logging
D18 (Tx1)/D19(Rx1)	Serial USART channel for communicating with motor control board
D33	Connected to IR remote receiving unit

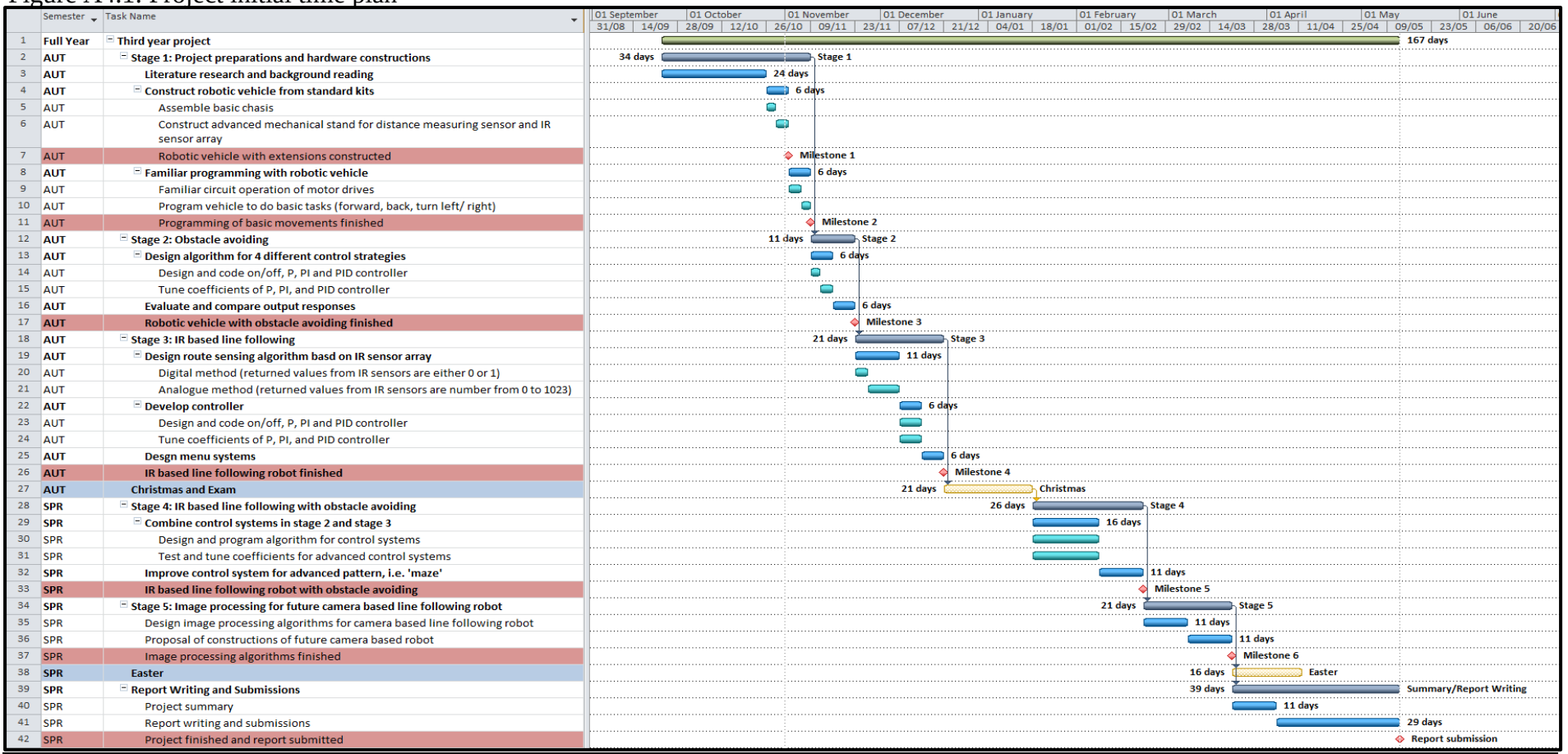
APPENDIX 3: SOFTWARE BACKUP FILES

The detail contents of this appendix can be referred to the attached CD. These software attached files include:

- Arduino IDE (Integrated Development Environment) version 1.6.5;
- Library files:
  - FloattoString
  - IRRemote
  - LiquidCrystal
  - Qtr-sensors-arduino
  - SD-Adafruit
  - Servo
- Line following robotic vehicle software file:
  - Motor board;
  - Control board;
  - Protection board;
- Simulations of image processing technique (vision based);
- Tools for analysing output response:
  - Application for generating time response;
  - Application for deriving frequency response;
  - Application for calculating statistical parameters from time domain;
- Wheel calibrations and tests;
- Demo videos of selected different achieved features;
- Project planning document (interim report);

APPENDIX 4: COMPARISONS OF INITIAL TIME PLAN AND ACTUAL PROJECT PROGRESS

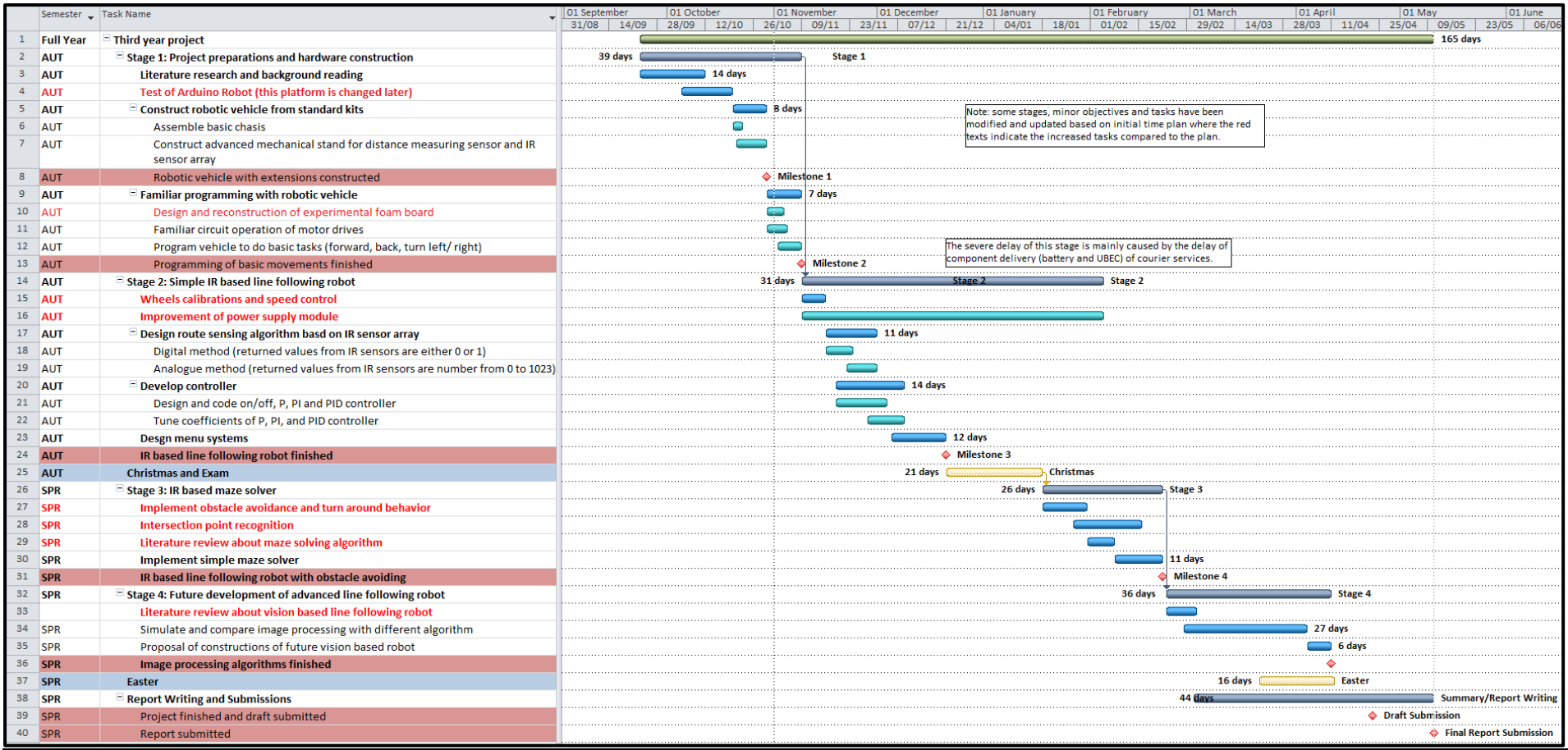
Figure A4.1: Project initial time plan



	Semester	Task Name
1	Full Year	Third year project
2	AUT	Stage 1: Project preparations and hardware constructions
3	AUT	Literature research and background reading
4	AUT	Construct robotic vehicle from standard kits
5	AUT	Assemble basic chasis
6	AUT	Construct advanced mechanical stand for distance measuring sensor and IR sensor array
7	AUT	Robotic vehicle with extensions constructed
8	AUT	Familiar programming with robotic vehicle
9	AUT	Familiar circuit operation of motor drives
10	AUT	Program vehicle to do basic tasks (forward, back, turn left/ right)
11	AUT	Programming of basic movements finished
12	AUT	Stage 2: Obstacle avoiding
13	AUT	Design algorithm for 4 different control strategies
14	AUT	Design and code on/off, P, PI and PID controller
15	AUT	Tune coefficients of P, PI, and PID controller
16	AUT	Evaluate and compare output responses
17	AUT	Robotic vehicle with obstacle avoiding finished
18	AUT	Stage 3: IR based line following
19	AUT	Design route sensing algorithm basd on IR sensor array
20	AUT	Digital method (returned values from IR sensors are either 0 or 1)
21	AUT	Analogue method (returned values from IR sensors are number from 0 to 1023)
22	AUT	Develop controller
23	AUT	Design and code on/off, P, PI and PID controller
24	AUT	Tune coefficients of P, PI, and PID controller
25	AUT	Desgn menu systems
26	AUT	IR based line following robot finished
27	AUT	Christmas and Exam
28	SPR	Stage 4: IR based line following with obstacle avoiding
29	SPR	Combine control systems in stage 2 and stage 3
30	SPR	Design and program algorithm for control systems
31	SPR	Test and tune coefficients for advanced control systems
32	SPR	Improve control system for advanced pattern, i.e. 'maze'
33	SPR	IR based line following robot with obstacle avoiding
34	SPR	Stage 5: Image processing for future camera based line following robot
35	SPR	Design image processing algorithms for camera based line following robot
36	SPR	Proposal of constructions of future camera based robot
37	SPR	Image processing algorithms finished
38	SPR	Easter
39	SPR	Report Writing and Submissions
40	SPR	Project summary
41	SPR	Report writing and submissions
42	SPR	Project finished and report submitted

Initial Time Plan of Third Year Project	**Explanations of Graphics and Symbols**
Project: Design, Construction and Control of a Low Cost Intelligent Line Following Robotic Vehicle	Overall projects Major aims and stages
Name: Chen Chen	Secondary objectives
Supervisor: Dr. John Walker	Main tasks
Moderator: Dr. Steve Sharples	Holidays, exams and vacations
Date: 10 May 2015	Milestones

Figure A4.2: Actual project progress



Actual Progress of Third Year Project	**Explanations of Graphics and Symbols**
Project: Design, Construction and Control of a Low Cost Intelligent Line Following Robotic Vehicle	Overall projects
	Major aims and stages
Name:	Secondary objectives
Supervisor:	Main tasks
Moderator:	Holidays, exams and vacations
Date: 10 May 2015	Milestones