

University of Southern Queensland
FACULTY OF ENGINEERING AND SURVEYING

MOBILE MANOEUVRING ROBOT

A dissertation submitted by

Mr. Matthew Free

in fulfilment of the requirements of

Courses ENG4111 and 4112 Research Project

towards the degree of

**Bachelor of Engineering
(Mechatronics)**

Submitted: November, 2006

ABSTRACT

Robotic guidance is a large part of many research and design applications. It is being used more frequently in everyday lifestyles such as vacuum cleaners and lawn mowers, and is slowly being introduced into the automobile industry.

The need for this technology is growing evermore and different aspects and methods are being implemented, from various sensor types to camera machine vision. This dissertation is compiled to explore the use of one of these technologies implemented into a small unit.

This dissertation develops and analyses the use of Mechatronic technology in the use of household applications. The overall objective of this project is to implement a microprocessor based controller in a small mobile robot to achieve straight line driving with implemented obstacle avoidance.

Limitations of Use

The Council of the University of Southern Queensland, its Faculty of Engineering and Surveying, and the staff of the University of Southern Queensland, do not accept any responsibility for the truth, accuracy or completeness of material contained within or associated with this dissertation.

Persons using all or any part of this material do so at their own risk, and not at the risk of the Council of the University of Southern Queensland, its Faculty of Engineering and Surveying or the staff of the University of Southern Queensland.

This dissertation reports an educational exercise and has no purpose or validity beyond this exercise. The sole purpose of the course pair entitled “Research Project” is to contribute to the overall education within the students chosen degree program. This document, the associated hardware, software, drawings, and other material set out in the associated appendices should not be used for any other purpose: if they are so used, it is entirely at the risk of the user.

Prof R Smith

Dean

Faculty of Engineering and Surveying

Certification

I certify that the ideas, designs and experimental work, results analysis and conclusions set out in this dissertation are entirely my own efforts, except where otherwise indicated and acknowledged.

I further certify that the work is original and has not been previously submitted for assessment in any other course or institution, except where specifically stated.

Matthew Phillip Free

Student Number: 0050009628

Signature

Date

ACKNOWLEDGEMENTS

This research was carried out under the principal supervision of Mark Phythian. Many thanks are due for his expertise and advice in the area surrounding the Mobile Manoeuvring Robot

Assistance was also obtained from Mr. Terry Byrnes. Many thanks also to his expertise and advice.

The University of Southern Queensland supplied vital components for the construction of the Mobile Manoeuvring Robot. Many thanks for the use of these components.

Appreciation is also due to Garth and Diana Free for the use of manufacturing tools and materials.

TABLE OF CONTENTS

Contents	Page
ABSTRACT	i
DISCLAIMER PAGE	ii
CANDIDATES CERTIFICATION	iii
ACKNOWLEDGEMENTS	iv
LIST OF FIGURES	viii
LIST OF TABLES	ix
LIST OF APPENDICES	x
NOMENCLATURE	xi
1. CHAPTER 1 – Introduction	1
1.1. Outline of the study	1
1.2. The problem	1
1.2.1. Project Goals	2
1.2.2. Performance Guidelines	3
1.3. Research objectives	3
1.4. Conclusions: Chapter 1	4
2. CHAPTER 2 – Literature Review	5
2.1. Introduction	5
2.2. The Need for Autonomous Applications	7
2.3. Sensing Distances in Applications	8
2.4. Object Avoidance	9
2.5. Dual Motor Drive Systems	11
2.6. Consequential effects and outcomes	12
2.7. Summary: Chapter 2	14
3. CHAPTER 3 – Research Design and Methodology	15
3.1. Safety issues	15
3.1.1. Mechanical Safety Issues	15
3.1.2. During The Electrical System Creation	17

3.1.3.	During the Testing Procedure	17
3.2.	Resource Requirements	17
3.3.	Sustainability	18
3.4.	Methodology	19
3.5.	Justification of Methodology	20
4.	CHAPTER 4 – Data Analysis	21
4.1.	Initial Design	21
4.2.	Second Prototype	22
4.3.	Odometer sensors and interrupt control	23
4.3.1.	Building the odometer sensors	23
4.3.2.	Using interrupts to count odometer readings	24
4.4.	H Bridge Motor Control	27
4.5.	Infrared Distance Sensors	29
4.5.1.	Requirements and Availability	29
4.5.2.	Choice of Sensor	31
4.5.3.	Design and construction of IR Sensor	31
4.6.	The Software	39
4.6.1.	Address Definition	41
4.6.2.	Variable Definition	42
4.6.3.	Main Program	42
4.6.4.	DRFWD Drive Forward Subroutine	43
4.6.5.	Turning Subroutines	44
4.6.6.	Delay	46
4.6.7.	COMP Compare subroutine	47
4.6.8.	KWGISR Key Wake up Port G	48
4.6.9.	IRQISR IRQ Interrupt Service Routine	49
4.6.10.	Initialisation of Ports and Interrupts	50
4.6.11.	Serial Port Communications	52
5.	CHAPTER 5 – Performance and Data Analysis	54
5.1.	Sensor Performance	54
5.2.	Mechanical Performance	57
5.3.	Software Performance	58

5.4.	Overall Performance	60
5.5.	Conclusions	61
6.	CHAPTER 6 Debugging Module	62
6.1.	Microcontroller Not working	62
6.2.	Motors Not Turning	62
6.3.	Not stopping for Objects	63
6.4.	Not driving in a Straight Line	64
6.5.	Stopping for Objects but irregular behaviour	64
7.	CHAPTER 7 Future Work	65
7.1.	Hardware	
7.1.1.	Chassis Design	65
7.1.2.	Sensor Design	65
7.1.3.	Power Source Design	66
7.1.4.	Software Design	66
8.	CHAPTER 8 Conclusions	67
	APPENDICES	69
	REFERENCES	97

LIST OF FIGURES

Number	Title	Page
Figure 2.1:	Army Bomb Deployment	6
Figure 2.2:	Bomb Disposal Robot	6
Figure 2.3:	Increasing technologies	7
Figure 4.1:	Initial Chassis Prototype	21
Figure 4.2:	Second Chassis Prototype	22
Figure 4.3:	Motor setup	23
Figure 4.4:	Chassis design	23
Figure 4.5:	Odometer Sensor	23
Figure 4.6:	Single hole odometer sensor	24
Figure 4.7:	Odometer Sensor Circuitry	24
Figure 4.8:	4013 Flip-Flop configuration	25
Figure 4.9:	Basic H Bridge Operation	27
Figure 4.10:	Sensor Distance Requirements	30
Figure 4.11:	555 Timer Pulsing Sensor Light	32
Figure 4.12:	PCB inside LED Torch	33
Figure 4.13:	LED Torch Complete With Wires	33
Figure 4.14:	Power Supply Filter for Transmitter	34
Figure 4.15:	Position of Sensor Components	36
Figure 4.16:	Sensor mounting on the Robot	37
Figure 4.17:	Data flow for software	40
Figure 5.1:	Sensor spectrum	56
Figure 5.2:	Layered approach to robot design.	58
Figure 5.3:	Straight line testing	59
Figure 5.4:	Results of straight line test	59
Figure 6.1:	Showing IR LED status	63
Figure 8.1:	Working Prototype of Mobile Manoeuvring Robot	68

LIST OF TABLES

Number	Title	Page
Table 3.1:	Cutting aluminium safety	15
Table 3.2:	Grinding and die grinding safety	16
Table 3.3:	Drilling safety	16
Table 3.4:	Lathe safety	16
Table 3.5:	Painting Safety	17
Table 3.6:	soldering safety	17
Table 3.7:	Testing safety	17
Table 3.8:	Mechanical resources	17
Table 3.9:	Electronic Resources	17
Table 5.1:	Sensor range	55

LIST OF APPENDICES

Number	Title	Page
A	Project Specification	70
B	Software Listing	71
C	Software Design Procedure	85
D	Component Data Sheets	91

NOMENCLATURE AND ACRONYMS

The following abbreviations have been used throughout the text and bibliography:-

CRO – Cathode Ray Oscilloscope

DSE – Dick Smith Electronics Australia

IC – Integrated Circuit

IR – Infrared or Infrared Light

LED – light emitting diode

PCB – Printed Circuit Board

PWM – Pulse Width Modulation

USQ – University of Southern Queensland (Toowoomba Campus)

CHAPTER 1

INTRODUCTION

1.1 Outline of the study

The final objective of this project is to implement a successful electronic system into a small mechanical unit in an attempt to make it travel throughout an area whilst avoiding obstacles to evade collisions. In this case the solution will come from the selection of a suitable microcontroller, drive system and sensors.

Further work may be completed to make this device actually perform a practical task such as object retrieval; however this is not essential to the outcomes of this project.

The project objective will be satisfied if the software and hardware configuration is able to control the vehicle within an indoor environment whilst avoiding collision with obstacles.

1.2 The problem

The overall project objective is stated above as *“to implement a microprocessor based controller in a small mobile robot to achieve straight line driving with implemented obstacle avoidance.”*

The project objective will be satisfied if the software and hardware configuration is able to control the vehicle within an indoor environment whilst avoiding collision with obstacles.

This means the unit will have such features as odometer counters for each wheel to aid in turning and driving in straight lines and hopefully record where it is in regards to its starting position. At least three sensors will be used to ‘read’ the path to the front and both sides of the unit. This will be necessary in the path tracking of the unit, i.e. deciding on which way to turn in an attempt to avoid collision.

1.2.1 Project Goals

In order to reach the overall objective of this project it is necessary to consider a set of guidelines or a set of project goals to direct the progress of achievement. They are set to be a set of open statements allowing a lot of movement and flexibility within them but to globally control the direction and outcome of the project.

By pursuing the following goals the project objective should be reached satisfactorily.

1. Define a guideline to assess the performance of the robot to the specifications of the project objective.
2. Build a prototype that is practical to the objective
3. Research the background behind obstacle avoidance and other attempts to achieve similar tasks.
4. Research microcontrollers and sensors to acquire the most practical configuration as to meet the objective.
5. Define an initial design of the configuration and code to be implemented.
6. Construct initial design
7. Test and analyse the results in accordance to the performance guidelines.
8. Re evaluate the design as required to achieve the project objective.

As time permits the following steps will be undertaken to improve on the robot and extend the project outcomes.

- Define a new performance guideline for the practical task completion
- Research more methodology if required
- Re design and construct the robot to achieve the new objective.
- Test and analyse the results in accordance to the new performance guidelines
- Evaluate and change the design as required to achieve new objectives.

1.2.2 Performance Guidelines

The first project goal is to *define a guideline to assess the performance of the robot to the specifications of the project objective.*

For this project to be successful in accordance with the project objective, it is required that the robot perform to the following criteria.

- Travel in a straight line when in a clear area with no manual assistance.
- Stop before colliding with any obstacles that are in the path
- “Decide” which way is more practical to turn and do so.
- Continue on its path in a straight line.

If the robot adheres to this criterion it will be able to drive continuously, avoiding collision on a random path.

1.3 Research objectives

The research in this project will basically revolve around similar projects and concepts developed to achieve similar tasks. The methods that have been used in various areas will be evaluated in comparison with each other and the superior designs analysed and modified to enable a suitable proposal for the manoeuvring robot.

The main research has been compiled into a literature review and is detailed in chapter 2 ‘*Literature Review*’.

1.4 Conclusions: Chapter 1

From the contents of this chapter the overall project objective has become clear and broken down into sizeable sections that can be completed individually. The content of required research has been outlined and a set of performance guidelines has been created to show the true desired outcome. These are all vital parts of the project if the objectives are to be met successfully.

The following chapters will detail each of the sections outlined in chapter one and explain in full the requirements, complications and decisions relating to each aspect. The majority of this dissertation will detail the design, construction and testing of the Mobile Manoeuvring Robot.

CHAPTER 2

LITERATURE REVIEW

The purpose of this literature review is to discover and critically analyse similar concepts to that of the Mobile Manoeuvrable Robot. This will build a suitable framework to commence work and research on the project.

The literature view will comprise of the following aspects:

1. Brief Background on the concept of autonomous control
2. Is there a need for autonomous appliances/applications
3. What is the most suitable method of sensing distances?
4. What methods are best used for object avoidance?
5. Research of previous implementation of dual motor drive systems.
6. Discussions and Implications of autonomous technologies.

2.1 Introduction

The use of automated or autonomous robotic control is not a new concept. Many people have dedicated their careers into designing these machines for many different areas and many different applications. The military uses robots as war machines, such as the TALON robot. The picture below shows a block of C4 explosive being placed in the claw of the robot. Other military uses include safe bomb diffusion by use of robots such as the following TELEROB. The robot can be safely sent into a dangerous environment without the risk of losing human life. This means bombs can be diffused faster and safer.



<http://www.army-technology.com>

Figure 2.1: Army Bomb Deployment



State-of-the-art remote bomb disposal technology, the telerob Explosive Ordnance Disposal and observation robot tEODor. The German Army model is equipped with five cameras and a double shot disruptor type Richmond RE70.

<http://www.army-technology.com>

Figure 2.2: Bomb Disposal Robot

A more closely related topic is the automatic vacuum cleaner. These units commonly use ultrasonic sensors to navigate around the room cleaning as it goes. Boundaries can be set using magnets and the unit is sensitive to drop-offs such as stair wells. A common example of this is Electrolux's Trilobite which is now commercially available at retail stores.

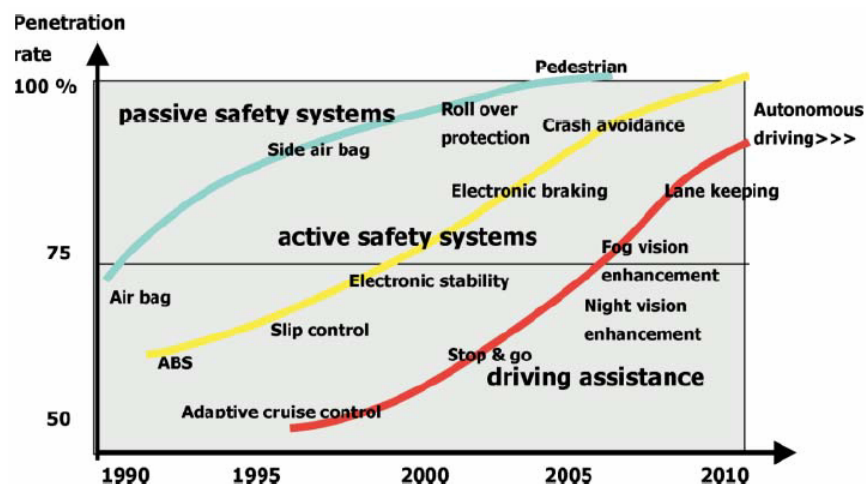
2.2 The Need for Autonomous Applications

Imagine never having to wash, iron, clean or cook again. A robot could do it all for you while you sit back and relax. To some this seems like the perfect world, while to others it seems unethical and truly against their beliefs. The demand for autonomous applications is becoming ever more increasing and public acceptance is also on the increase.

The company 'Poly Micro' argued in their 2004 newsletter that autonomous appliances will grow because of consumers increasing expectations of energy and water saving, noise reduction and overall efficiency and functionality. They also bring to focus that more technology is being accepted within society. The following graph from poly micros 2003 newsletter shows the acceptance of new technology relating to the automation of driving systems within a manned vehicle.

Poly Micro predicts that by 2010 autonomous driving will be largely accepted as practical by society. New cars of today are considered dangerous not to have an airbag whereas 20 years ago this technology was not heard of.

Although this data varies slightly from this dissertation work, the trend of need and acceptance through time is accurately portrayed in Figure 2.3.



<http://www.polymicro-cc.com/site/pdf/POLYMICRO-markets.pdf>

Figure 2.3: Increasing technologies

The article “Appliances of the future’ states also that the need and expectations of this type of technology are growing at an alarming rate. It also mentions that for ideal autonomous work in the future that robots will need emotion such as fear, pride and caution. How can a robot complete great work if it has no pride in what it does, and how can it make decisions without a conscience?

Henrik Christensen suggests that autonomous robotics in the home is a great idea however has not come far enough to really break into the household market. The cost is simply too high in ratio to its functionality.

Overall it is evident that the need and acceptance for autonomous robots in the home is ever growing. While some circumstances of future developments may seem unethical the basic cleaning and entertainment robots are proving to be an accepted and sought after item in the home.

2.3 Sensing Distances in Applications

There are many different methods available for the purpose of sensing distances. These include sensors such as proximity, ultrasonic, infrared, magnetic and various others. Which of these is most suitable for a household application?

The Electrolux Trilobite (vacuum cleaner) uses ultrasonic sensors to navigate around the room. Most sensors work by emitting some form of light or frequency and waiting for that emitted source to ‘bounce back’. Ultrasonic sensors use sound or frequency whereas infrared uses an invisible light source to judge distance or proximity. Infrared light is often also used in data transmission such as remote control for a common television.

The robo-rats website (2001) shows several different methods of sensing distances and details the connections and problems associated with the setups. It shows methods of beam breaking, using infrared emitters and detectors; distance sensors, also using infrared emitters and detectors; and photocell sensing.

From all of these details it seems that a very successful method of judging proximity, and also possibly recognising distance, is to use an infrared LED (Light Emitting Diode) and an infrared detector. When the robot is facing a wall or object within a certain distance (unknown at this stage and variable) infrared light will reflect from that object and be collected in the detector. The higher the voltage or current returned from the receiver, the closer the object must be. To overcome the possibility of natural light, containing infrared, affecting the functionality of this sensor configuration, it will be necessary to clock the infrared emitter at a rate of 36 – 40 kHz. The receiver will also have to be tuned in order to only recognise this frequency of light.

2.4 Object Avoidance

Many applications are beginning to incorporate object avoidance technology into their design. Cars are becoming more autonomous, traction control for example, and this technology is bound to move towards object avoidance. Reversing sensors can alert the driver when an object is close to the rear of the vehicle. It is only a matter of time before a car will be able to recognise which lane it is and manage the steering system to ensure it does not cross any lines. This technology is sure to save many lives from the hazard of driver fatigue.

Object avoidance technology is currently installed into many items such as automatic vacuum cleaners, auto lawn mowers and other items where little human instruction is desired. Electrolux's Trilobite automatic vacuum cleaner uses random driving while avoiding collision in order to cover the floor space.

A project conducted by a student at Niagara Technology University used sensors and hardware to create a object avoiding robot not dissimilar to that of this dissertation. The concept behind this project gave the robot a human like thought process in that it would come to an object, view the left then view right and make a decision of which way was best to turn. If the robot had cornered itself then it would turn around and continue driving.

David Tunnel 2004 suggested that there was no perfect solution available for autonomous object avoiding and hence started a proposal to create an algorithm to

perform this operation. The proposed algorithm is intended to create a logical path for a unmanned vehicle to take by use of sensors, videos and imaging data. ‘Our focus is to develop a object avoidance algorithm called SmartAvoidT that extracts multiple objects/targets out of video/imagery data, establishes individual tracks for each object and maps a path around each object to avoid collisions.’ (Tunnel D 2004). This algorithm would then be implemented into the vehicles navigation system in order to follow the calculated path. The algorithm will be designed to work in all weather conditions such as day, night, rain, smoke or any other condition.

Koren Ward suggests that a successful method of controlling object avoidance is for the robot to ‘learn’ methods and trends associated with traversing different situations. This concept still utilises sensors and logic however has another much more complex learning ability. The unit will record its previous experience with regard to the readings from its sensors and so will be able to make a decision based on what occurred last time. For example if sensor 1 was blocked and sensor 2 was clear then it will perform in the same way as it did last time as long as last time encountered no errors. If errors occurred it will try a different method and compare the results. This is a very complex concept however it would most likely produce the least amount of errors in the long run. A basic logic system may encounter the same error repeatedly, whereas this use of a fuzzy logic learning system should prevent this occurrence.

While there are several methods of controlling a robot to avoid objects many are much too complicated for the general purpose of this dissertation. This dissertation aims to create a base model of an autonomous robot and so the programming of object avoidance will be kept to a simple level. The concept described by the technology student above is the initial concept decided upon for this dissertation. Giving the robot a set of hard coded rules once it reaches an object should give satisfactory performance while maintaining a simpler approach.

The coded instructions might look something close to the following:

START	Drive in a straight line until interrupt from sensor occurs
	Is left sensor clear?

If yes turn left
If no, is right sensor clear?
If yes turn right
If no then reverse out or turn 180 degrees
Continue in a straight line (loop to START)

2.5 Dual Motor Drive Systems

Many systems use dual motors for directional control. A dual motor drive system refers to having one motor for each drive wheel also known as 'differential drive'. Both motors driving forwards will make the unit move forwards; one motor forwards and one motor in reverse will result in the unit turning etc. A well known example of this drive system is in a skid steer (commonly known as a Bobcat). This method gives excellent directional control and 'on the spot' turning allowing for tight corners and superior manoeuvrability. The major problem with this method however is that one motor will always turn slightly faster than the other resulting in the unit driving in a slight curve. This may be due to efficiencies in the motors, gearbox friction, wheel to ground friction and other uncontrollable factors. Some form of odometry for each wheel will be necessary to ensure that the unit manoeuvres in a straight line.

This issue of non-straight driving is very common among robot builders and has been overcome in several different ways. Many choose to change the drive system to a more car like drive system with a single drive motor and steering mechanism. This is not practical in many cases as it will sacrifice functionality. The problem has been overcome before by pulse-width modulating the two drive motors and counting the edges on the encoders located at each wheel. This seems to be a very successful method.

Knudsen J 2000 refers to a similar method of counting the revolutions of the drive shafts and comparing the number or revs on each side. All of this information is based on LEGO™ components however the methods are still relevant. It also explores the use of a mechanical differential to ensure straight line driving is obtained. This is very similar to a car differential however is used in a different manner. By attaching the two drive motors to where wheels would normally be connected, the differential will not

rotate unless the motors turn at different speeds. By monitoring the differential with a sensor algorithms can be written to control the navigation.

For the purposes of this dissertation the clearest method is to use sensors on each wheel, or better yet on the actual drive motor, to measure the revolutions and therefore distance travelled. An algorithm will be written to ensure both drive wheels rotate at the same speed.

This method will become very useful if time permits to increase the robots performance. If this goes ahead the robot will use a method known as 'dead reckoning' where it calculates its position as a sum of how far and in what direction it has travelled since its origin. Although this is a rather inaccurate approach it is far simpler than any Global Positioning Systems (GPS), especially in such a small application.

2.6 Consequential Effects and Outcomes

The final outcome of this project provides a base model for intelligent household applications such as cleaning, object retrieval or similar. The extent of this project will not extend to a final product suitable for marketing but is just the base concept.

The research and technical outcomes from this project pose no direct ethical or legal dilemmas however if this concept is extended on then some factors may need to be considered.

Many engineers and scientists are still studying the use of autonomous vehicles for many different purposes in many different fields of expertise. From military through to agriculture robots are becoming more frequently relied on and the need for more advanced technology is ever increasing.

Military and federal applications of robots include the use of technology to diffuse bombs or clear areas where the risk of losing human life is too high. Similar to this is the use of robots to track through rubble from collapsed buildings or landslides to search for survivors and perform simple medical checks. NASA uses autonomous robotics for missions such as Mars Rover discovery and other distant missions.

Although most of these cases still rely on human instruction, usually from remote control, they are all still based on a similar content to the technical side of this dissertation.

Future development of this idea of autonomous mobile navigation may result in any number of new designs. Some of these may be:

- Concept cars able to drive without human instruction. May be used for taxi services or courier and mail delivery.
- A similar service to 'Guide dogs for the blind', helping blind members of the community to be self sufficient.
- Automatic cleaning of bathrooms, kitchens and offices. Reducing the need for maids and cleaners.

Some of these future developments pose some ethical questions however. Is it right to make a machine that will put people out of work? What if the automated robot causes damage to humans or property and is not able to be controlled? What if an automated car crashes, killing a family with young children? Who takes the blame for any of these circumstances? It is a very controversial subject with many sides to each argument.

Many movies have been created which discuss the possibility of robotics in the future. Humans create machine with such intelligence that they become uncontrollable. '*The Matrix, 1999*'; '*I Robot, 2004*' and '*Stealth, 2005*' all discuss the implications of errors in machines that have turned into drastic situations. Although these are most likely far fetched extremes, the concept is a possibility. Electronic components, such as solenoids and sensors, can easily fail which might make a car miss a turn or fail to brake. It once again poses questions if it is right to put a human's life in the hands of a machine.

2.7 Summary: Chapter 2

To summarise the literature behind the concept of a mobile manoeuvring robot it is obvious that most of this technology has been invented before. The content that will be covered in this dissertation will be on the basis of autonomous technology and hence will most likely not be an extension into new technology or concepts. The basis for this project is to explore and understand the basic essentials in an autonomous situation that may be used for any industrial or commercial purpose.

CHAPTER 3

RESEARCH DESIGN AND METHODOLOGY

Before commencing any of the construction or completed design of any of the electronic or mechanical components, it was necessary to undertake some certain analysis. A safety analysis and a resource requirements and acquisition table was completed to fully understand the background, direction and precautions to consider whilst the project was under design and analysis.

3.1 Safety Issues

Safety is a relatively low concern in this project. While various factors require some attention, there are no large definite risks such as chemical handling. Saying this however does not mean that there is no cause for concern along the duration of this project. The following risk assessment outlines the primary hazards and methods of reducing the risk.

Constructing anything mechanical brings with it some risks. The construction of this mechanical unit involves cutting, grinding, drilling and using other power tools on products such as aluminium and timber. Bushes are also to be made from industrial grade plastics on a lathe. On the electrical side of this project a lot of soldering and drilling/screwing will also take place. The possible hazards and risks of these operations are listed below.

3.1.1 Mechanical Safety Issues

Cutting aluminium with drop-saw or similar

Hazard	Occurrence likelihood	Consequence H/M/L	Controls to avoid injury	Risk after controls
Loud Noise	High- Aluminium is a noisy material to work with	H – Hearing Damage will occur for multiple cuts	Hearing protection must be worn	L – Small amount of cuts no problem.
Flying debris	High – Hot spatter from saw blade will fly	H – Blinding if caught in eyes, minor burns possible on skin	Eye protection to be worn, Face mask preferable, non-loose – long sleeved clothes	L – Adequately protected
Saw jamb	Medium – If work not secured piece may fly	H- possible loss of fingers in blade or pinching of skin	Clamp work piece when cutting	L – If clamped no problem

Table 3.1: Cutting aluminium safety

When grinding, ensure that the appropriate sized grinder is used. For example, a nine inch angle grinder used on a small piece of aluminium is likely to cause damage to either operator and/or work piece.

Grinding/ Die grinding

Hazard	Occurrence likelihood	Consequence H/M/L	Controls to avoid injury	Risk after controls
Loud Noise	High- Grinding is a noisy operation	H – Hearing Damage will occur for long use	Hearing protection must be worn	L – Small amount of cuts no problem.
Flying debris	High – Hot sparks from blade	H – Blinding if caught in eyes, minor burns possible on skin	Eye protection to be worn, Face mask preferable, non-loose – long sleeved clothes	L – Adequately protected
Grinder jamb	Medium – If work not secured piece may fly	H- possible loss of fingers in blade or pinching of skin	Clamp work piece when cutting, use appropriate grinder	L – If clamped no problem

Table 3.2: Grinding and die grinding safety

When using any power tool ensure correct tools and attachments are used and appropriate user knowledge of the operation is known.

Drilling and other power tools

Hazard	Occurrence likelihood	Consequence H/M/L	Controls to avoid injury	Risk after controls
Loud Noise	Med – most power tools create some loud noise	M - Hearing Damage will occur for multiple cuts	Hearing protection must be worn	L – Small amount of cuts no problem.
Flying debris	High – Hot spatter from saw blade will fly	H – Blinding if caught in eyes, minor burns possible on skin	Eye protection to be worn, Face mask preferable, non-loose - long sleeved clothes	L – Adequately protected
Tool jamb	Medium – If work not secured piece may fly or spin	H- possible loss of fingers in blade or pinching of skin	Clamp work piece during work	L – If clamped problem minimised

Table 3.3: Drilling safety

Lathe work

Hazard	Occurrence likelihood	Consequence H/M/L	Controls to avoid injury	Risk after controls
Flying debris	High – Hot spatter from saw blade will fly	H – Blinding if caught in eyes, minor burns possible on skin	Eye protection to be worn, Face mask preferable, non-loose - long sleeved clothes	L – Adequately protected
High speed operation	M- Chuck and work piece spinning	H- loss of fingers of large cuts and abrasions	Experienced user present, caution used when operating	M- Impossible to remove spinning component
Work piece fly	M- chuck slip or error in clamping	H- building damage or personal injury	Ensure piece clamped, be aware of possible piece release	M- Impossible to reduce risk to a low level.
Saw jamb	Medium – If work not secured piece may fly	H- possible loss of fingers in blade or pinching of skin	Clamp work piece when cutting	L – If clamped no problem

Table 3.4: Lathe safety

Painting

Hazard	Occurrence likelihood	Consequence H/M/L	Controls to avoid injury	Risk after controls
Toxic Fumes	Med – many fumes present	M – Eye and respiratory damage	Eye protection to be worn, fan or dust extraction present, ventilated area	L – Adequately protected

Table 3.5: Painting Safety

3.1.2 During The Electrical System Creation:

Soldering

Hazard	Occurrence likelihood	Consequence H/M/L	Controls to avoid injury	Risk after controls
Toxic Fumes	Med – many fumes present raising upwards	M – Eye and respiratory damage	Eye protection to be worn, fan or dust extraction present, ventilated area	L – Adequately protected
Hot solder dripping	Low	M- Mild burns to skin	Clear work area in standing position. Adequate clothing	L- Caution will mean no burns
Hot iron	Med. Touching will leave instant burns	M – Possible mild burns	Clear work area and use caution	M – unavoidable other than caution used

Table 3.6: soldering safety

3.1.3 During The Testing Procedure:

Although testing seems very straight forward, there are still several hazards to be aware of. User hazards are minimal however damage to hardware and property is still possible.

Testing

Hazard	Occurrence likelihood	Consequence H/M/L	Controls to avoid injury	Risk after controls
Object avoiding fails	M- sensor fail, code error	L – small scratches on objects	Use test blocks which can be hit	L- no damage will be caused
Overload or short circuit microcontroller	M – surge in power or error in connection	H – Expensive microcontrollers	Careful connections, use overload protection	L- microcontroller protected
Power sources	Med – voltage shocks	H – if large voltage touched, electrocution possible	Understand power pack before use. Tape or hide high voltage wires	L- High voltages avoided.

Table 3.7: Testing safety

3.2 Resource Requirements

There are many components to this project which will be sourced from a widely varied field. Due to the nature of this project it will be constructed on a tight budget. This means aesthetics and practicality may suffer to some degree. Most resources will either be scavenged from existing products or borrowed from institutions such as USQ where possible.

The required resources can be categorized into two major fields, mechanical and electronic. The following explains these categories and the planned source.

Mechanical components/resources:

Resource	Description	Requirements	Source	Back-up Plan
Unit chassis	Aluminium 100x100 box section	Light weight	Scrap bin at Patio shop	N/A
Motors (x 2)	12V Low Amperage		Purchase (Bunnings XU1 cordless drills)	N/A
Gear box (x 2)	Windscreen wiper motor (XD Falcon)	Minimal weight, large reduction	Car wreckers	N/A
Trundle wheel	Low friction small trundle wheel	Light weight	Old cupboard wheel	N/A
Fastenings (rivets, screws etc.)	Assorted	N/A	Family Workshop	Hardware Purchase
Wheels	115mm diameter	Light weight	Pram wheels	Hardware Purchase
Bushes etc	Assorted	Low friction, light weight, affordable	Create in workshop	Second hand toys

Table 3.8: Mechanical resources

Electronic components

Although there will be many components required that are unknown at this stage, the following table provides a basic plan for the sourcing of components.

Resource	Description	Requirements	Source	Back-up Plan
Microcontroller	Motorola HC12	---	USQ on loan	Purchase chip and get soldered to PCB
Infrared LED + sensors	Various	---	Electronic shop	Internet purchase
555 timers	---	---	Electronic shop	USQ loan
Wiring	Various colours	---	Electronic shop	USQ

Table 3.9 Electronic Resources

3.3 Sustainability

Unfortunately it will not be practical to keep the robot assembled at the completion of this project. Due to the major component (HC12 microcontroller) being unaffordable to the budget, the robot will be useless after the return of components. Without the microcontroller the rest is rendered futile.

At the conclusion of this project several components will be returned. The microcontroller will remain the property of USQ. The windscreen wiper motors will be beyond repair but may be reused in other applications. Wheels and other mechanical components will be returned to the workshop. The electronic components will be kept for other applications where sensors etc can be used.

3.4 Methodology

The following project goals were created in chapter one and are re-listed in methodology so as to set the basis for the structure of this dissertation. The project work outlined in this dissertation is based entirely around this methodology.

By pursuing the following goals the project objective should be reached satisfactorily and efficiently.

1. Define a guideline to assess the performance of the robot to the specifications of the project objective.
2. Build a prototype that is practical to the objective
3. Research the background behind obstacle avoidance and other attempts to achieve similar tasks.
4. Research microcontrollers and sensors to acquire the most practical configuration as to meet the objective.
5. Define an initial design of the configuration and code to be implemented.
6. Construct initial design
7. Test and analyse the results in accordance to the performance guidelines.
8. Re evaluate the design as required to achieve the project objective.

As time permits

- Define a new performance guideline for the practical task completion
- Research more methodology if required
- Re design and construct the robot to achieve the new objective.
- Test and analyse the results in accordance to the new performance guidelines
- Evaluate and change the design as required to achieve new objectives.

3.5 Justification of Methodology

The methodology above has been chosen carefully to complete the project objectives. The first step of *'Define a guideline to assess the performance of the robot to the specifications of the project objective'* has been chosen to give the robot a set of guidelines to be assessed against. Without having this goal, it will be impossible to determine if the robot has met all requirements. Goal three, *Research the background behind obstacle avoidance and other attempts to achieve similar tasks*, will give background knowledge behind the concept. This can save a lot of time in decision making processes as it may help predict the outcomes based on previous experiences. The design stage is the next logical step and is carried out as goal four. The testing stage is very important and is closely linked with goal one of the performance guideline. The robot must be tested and assessed against these guidelines in order to understand which parts of the robot were successful and which aspects need more work.

The methodology of the time permitting guidelines can be justified in the same manner. It is a very similar process with most of the background information already known.

CHAPTER 4

Construction and Design

4.1 Initial Design

Several ideas were analysed before the initial prototype was built. The mechanical structure of the robot was a major consideration. Wheels versus tracks, mechanical steering versus dual motor drive and other considerations along with shape and dimensions were the chief factors.

Initial analysis of these ideas provided a dual motor, two track setup with a wide wheel - base for stability. This design is shown in figure 4.1 below.

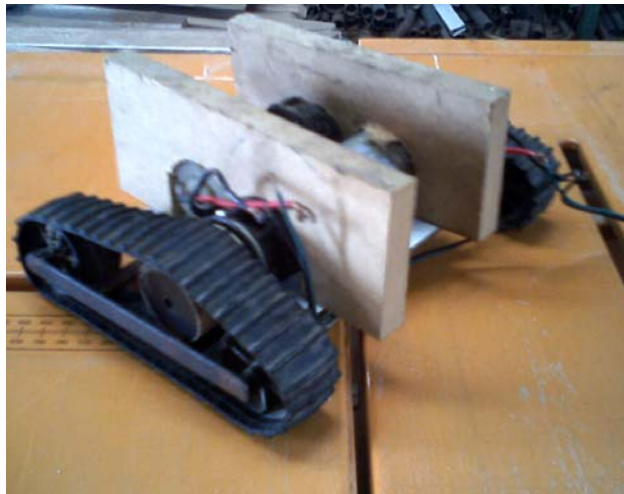


Figure 4.1: Initial Chassis Prototype

This design seemed very practical for this application due to its small centralised arrangement, its excellent turning circle and ability to manoeuvre. The problem occurred however when the drive motor could not gain sufficient traction to the rubber track. Several different ‘drive hubs’ were implemented to attempt to overcome this problem however the rubber tracks were too flexible with a very small coefficient of friction. The setup required to make this arrangement successfully work far outweighed the practicality of the design.

The motor slipping in the tracks would have made this project a near impossibility without a delicate error correction system in place. Because of these difficulties a decision was made to use wheels connected as a direct drive to the motor.

4.2 Second Prototype

The second prototype for the mechanical design consisted of an aluminium chassis with dual drive motors. A simple trundle wheel for stability was used to allow ‘on the spot’ steering and high manoeuvrability.

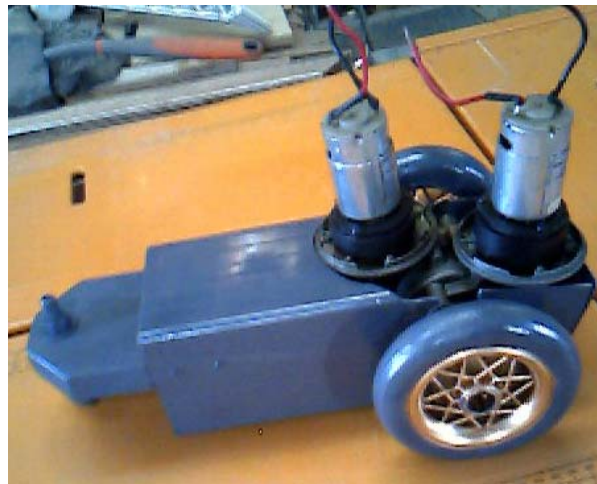


Figure 4.2 Second Chassis Prototype

The drive motors were chosen from a set of affordable ‘XU1’ 12 volt cordless drills. The speed of these motors was far too fast to attach directly to the drive wheels of the robot however, so a gearbox was required. A worm drive seemed the logical arrangement for this to provide high reduction, high torque and very minimal backlash. Two windscreen wiper motors from XD Falcons were purchased from a car yard and modified to suit the desired purpose. Due to the size of this application, batteries will be small and therefore the motors need to be of low power consumption. Windscreen wiper motors draw approximately 5-10 amps and so running two of these would require a rather large battery. By replacing the original motors with the cordless drill motors, the amperage was dropped to approximately 5 amps for both motors running simultaneously under load. Figure 4.3 shows the small motor mounted on the gearbox.



Figure 4.3: Motor setup



Figure 4.4: Chassis design

The chassis is a 100 x 100mm box section of aluminium commonly found on patio posts. The axle holes were formed with a die grinder and are designed for the motors to bolt straight into. The trundle wheel was placed on temporarily until final dimensions are known for extra hardware (batteries, microcontrollers, wiring etc).

4.3 Odometer Sensors and Interrupt Control

4.3.1 Building the odometer sensors

As explained earlier, it is necessary for the Mobile Robot to be able to drive in straight lines between obstacles. For this to occur it is necessary to calculate the turns of each wheel (or motor) to compare against the other. There are various methods available for this; however the easiest and most successful technique is to insert a sensor on each motor to count the revolutions. In this case an infrared LED and receiver have been used to count the revolutions. Figure 4.5 shows how this sensor is implemented.

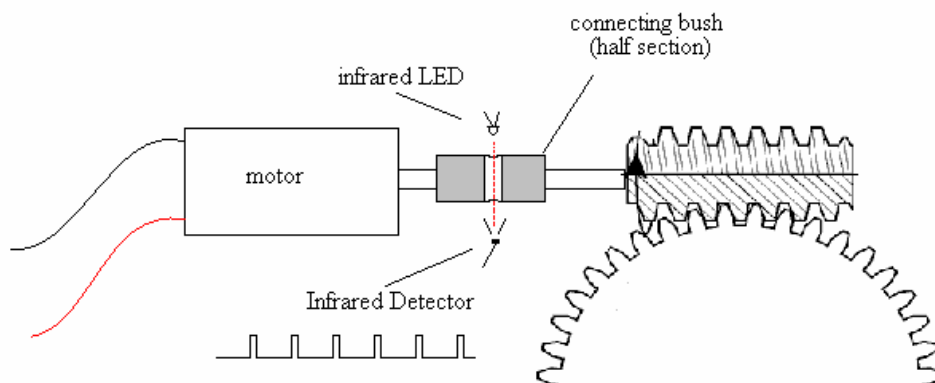


Figure 4.5: Odometer Sensor

The LED (DSE part number Z3235) and Infrared Receiving Diode (DSE part number Z1956) are mounted in an outer casing that mounts the motor to the gearbox. A 4mm hole is drilled through the motor drive shaft which aligns the LED with the Receiving Diode twice for every revolution of the motor. From this circuitry it is possible to obtain a 5 volt drop for every time the sensor is aligned.

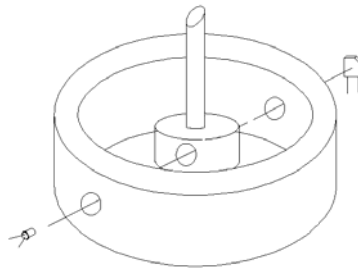


Figure 4.6 Single Hole odometer sensor

To obtain the 5 volt drop it was necessary to fiddle with resistor values on the 5 volt input rail to the receiver. The resistor value was dependent on how much of the hole aligned, the intensity of the infrared beam coming through the hole and also the brightness of the outside light. (Incandescent light bulbs and sunlight contain masses of Infrared.) To overcome the variance of the outside light, a film of white gap filler was applied over the receiver to completely isolate all of the surfaces from any source of IR other than the emitter. A guess and check of values found that when a 68 kilo ohms resistor was placed in series in the power supply for the receiving diode, a five volt drop (5v to 0v) was obtained when the sensor aligned. See circuit below for details.

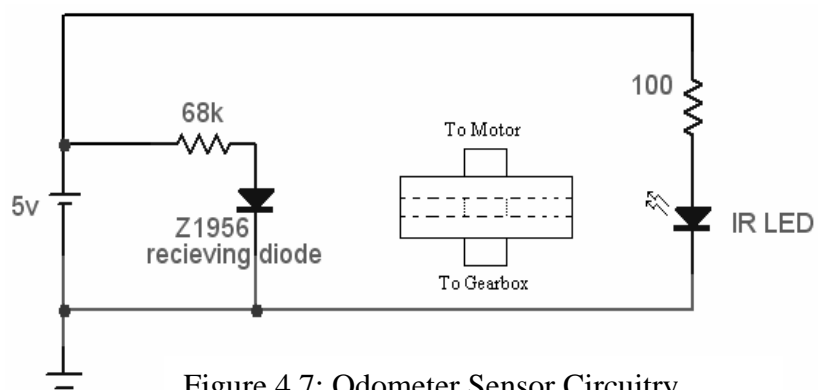


Figure 4.7: Odometer Sensor Circuitry

This circuit can be used to generate an interrupt for use in the microcontroller. It is used on the high to low edge however can be used in either edge sensitive case.

The above arrangement of a 4013 Dual D Type Flip Flop is set to work on a rising edge (0 to 5 volts). Using a falling or rising edge will not affect the accuracy of the counting; it will still see two rising edges every rotation of the motor. When the sensor rises from 0v to 5v, the flip flop is triggered and Q NOT changes from high to low which triggers the XIRQ (active low) interrupt. Before the end of the interrupt service routine, a logic high can be sent from PORT A to reset the flip flop and await the next rising edge from the sensor.

This concept did not work when attached to the sensor and microcontroller. When voltages were manually applied to the flip flop slowly it would perform successfully however when the process was applied through the XIRQ interrupt service routine, the logic levels would not change appropriately. The IRQ interrupt was also not working correctly. It was initialised to recognise falling edges but would not activate on the falling edge coming from the sensors. It would however activate if an earth wire was touched to it. It was decided that perhaps noise or other less obvious factors were causing this.

There seemed to be too many problems for the simplicity of this idea so it was deemed appropriate to disregard the use of XIRQ and IRQ for this purpose. The sensor wires were connected to the PORT G key wake up port and instantly an accurate counter was achieved. The major issue with this arrangement was to remember not to use any higher priority interrupts to run a routine which would turn the motors in either direction. This would make the position of the robot an unknown factor.

As a result of this testing the odometer sensors were made successful by simply using an Infrared LED (DSE part number Z3235) and an Infrared Receiving Diode (DSE part number Z1956) connected to the key wake up of PORT G and the software initialised correctly to deal with this. A simple interrupt service routine was written to add one (1) to the counter that recorded the revolutions of the respective motor. These counter values are then available for calculating straight line driving or even position monitoring.

4.4 H-Bridge Motor Control

After having the mechanical unit constructed, encompassing the microcontroller and odometry sensors, it was necessary to decide on a method to control the motors. Two H bridges were sourced from the technician's lab at USQ to allow the microcontroller to handle the large current flows. An H bridge works by a series of four switches which can control the current flow through the motor. The schematic below shows the basic operation of the common H Bridge. The grey arrows represent the current flow through the motor.

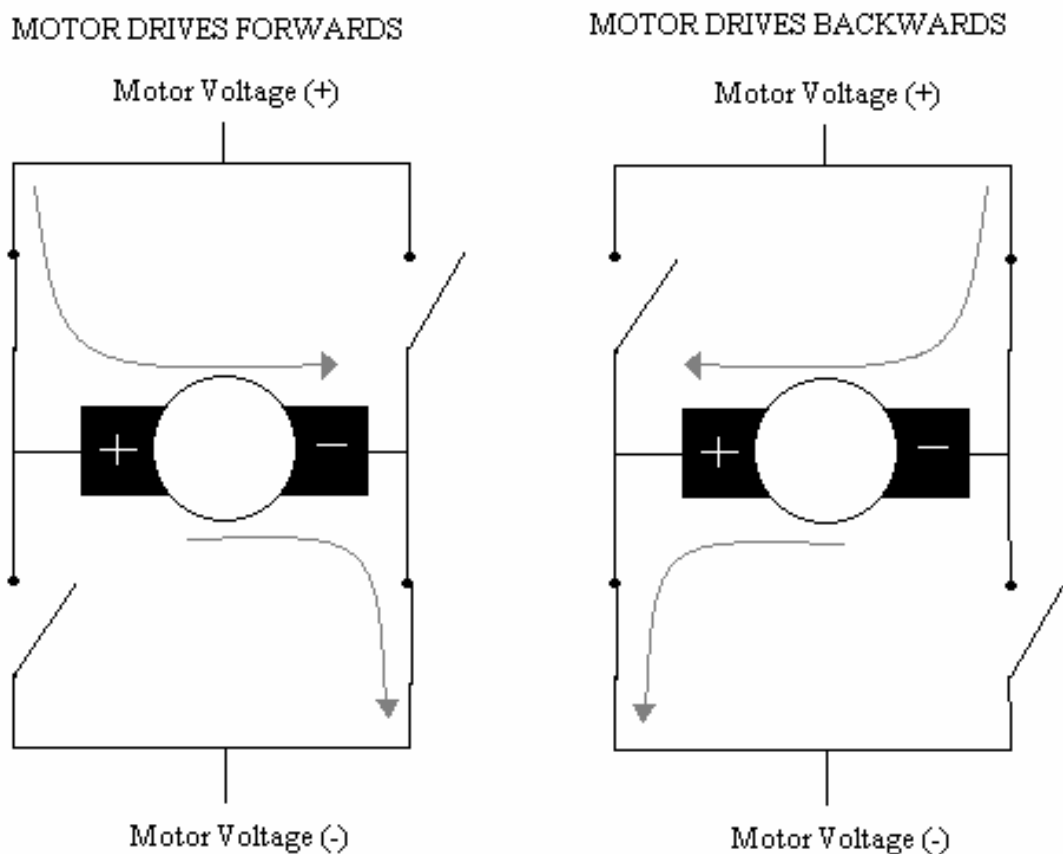


Figure 4.9: Basic H Bridge Operation

By applying a small voltage such as logic high from the Motorola HC12, the switches can be turned on to allow current flow and hence motor operation. Only diagonally opposed switches should be turned on at the one time. If both switches on one side are turned on at the same time a short circuit is made from the positive to negative voltage supply. This is often referred to as shoot through.

Anything that can control a current is suitable for use as the switches in an H bridge such as relays, transistors or MOSFETS. In this case, four MOSFETS have been used with small heat sinks to cater for the 12V motors which are drawing approximately 2.5 amps each at full load.

One excellent advantage of using an H Bridge is that it reduces the voltage and current spikes that occur from the motors starting, stopping or changing direction quickly. With the Mobile Manoeuvring Robot the voltage spikes were still higher than the H Bridge could contain. The cheap motors that were purchased for the drive have very poor characteristics and were resetting the microcontroller every time that the two motors turned on simultaneously. There are several procedures that could have been undertaken to overcome this problem. A more efficient pair of motors may have produced less voltage spike; a large capacitor could have been implemented to provide a spike filter; or the chosen method, to use separate power supplies to run the microcontroller and the motors respectively. A 9 or 10 volt battery is ideal for running the microcontroller as the voltage passes through a LM7805 voltage regulator. This method has the added advantage of being able to have a separate, higher voltage battery to run the motors without having to reduce it significantly to power the microcontroller. Both power supplies must share a common earth for the robot to function correctly.

To control the motor speed requires more than just a logic high input to the H Bridge. Motors will always run at slightly different speeds due to the internal friction losses and other variances in efficiency and so the robot would drive in large circles. Pulse Width Modulation (PWM) is a very successful method of controlling the speed. By applying a chosen frequency to both motors and then adjusting the duty cycle, the motors speed can be controlled. The Motorola HC12 has 4 channels dedicated to pulse width modulation. Bits 0-3 of PORT P can easily be set to output a controllable pulse. This provides a perfect environment to control two motors, one channel for each motor in each direction.

The four PWM pins (PORT P pins 0-3) are connected directly to the H bridge inputs. Simple software initialisation sets the ports ready for PWM and sets a frequency to base the duty cycle around. For the Mobile Manoeuvring Robot a pre-scaler divider of 128 is used to divide the 8MHz processor down to create a slower clock that can be used to

control the frequency. By setting a number between \$01 and \$FF in the PWPER (0-3) registers, the pulse frequency can be set to a proportion of this clock rate. The duty cycle of this frequency can then simply be controlled by storing values between \$00 and \$FF into the PWDTY (0-3) registers. This value represents the proportion of high over low time; hence \$FF/2 would be 50% duty cycle and the motor would be running at half speed, or \$00 would stop the motor.

It is important to not switch the H Bridge switches too fast in this application. Transistors require a small amount of time to switch on or off. It is because of this time that causes a current flow and hence a power loss. Over switching at high frequencies will increase the power consumption and therefore create heat from the MOSFET's. Heat sinks are installed on all of the robots MOSFET's to compensate for the switching power loss.

To conclude on motor control, the Mobile Manoeuvring Robot is controlled by two identical H Bridges which are powered from a 10 – 12 volt source. The motor speed is determined by four separate Pulse Width Modulation Channels with a relatively low but fixed frequency and variable duty cycle.

4.5 Infrared Distance Sensors

4.5.1 Requirements and Availability

The current market offers many types of sensors for use in various applications. A few of the major types of sensors include:

- Optical sensors
- Photo diode sensors
- Fibre optics sensors
- Proximity sensors
- Ultrasonic sensors.

For accurate manoeuvrability in this application it is necessary that the sensors have a long enough range to avoid collision during turning. Figure 4.10 illustrates this need.

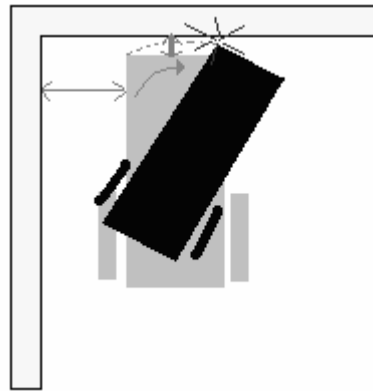


Figure 4.10 Sensor Distance Requirements

The above diagram illustrates that even after the robot has stopped to avoid an object, the turning circle will result in the corner of the robot still proceeding forwards. It is necessary for the sensor to be able to read further than this turning circle. The necessary distance can be reduced significantly by turning the outside wheel backwards so it pulls away from the wall, this however does pose the issue of being able to reverse into objects, as there is no rear sensors.

Also, when in this illustrated position, it would be preferable if the sensors could judge a long distance at the sides of the robot. If the robot were to turn left after stopping in figure 4.10, it would turn directly into a wall and be interrupted by the side sensor half way through a turn. This would make coding of the processor more difficult. Initial experimentation has shown the ideal range of the sensors will be approximately 200mm-300mm but definitely no less than 100mm for the front sensor.

Furthermore, for household applications the obstacle to avoid may be the wall. If a short range sensor is used and placed above the skirting board the wheels may impact this before the sensor realises there is an obstacle. This will result in scratches on the obstacles and possible breakage of the robot. Also, the robot will be jammed until loss of power or failure of components.

4.5.2 Choice of Sensor

Almost all of the current household applications that require distance or proximity sensing utilise Ultrasonic or Infrared Light (IR) sensors. The market available choices are large and vary in specifications and range in price from around \$40 - \$120 each. Many hobbyist robot builders choose to construct IR sensors to save a lot of cost and custom design them to suit the necessary requirements.

For the purpose of the Mobile Manoeuvring Robot, it was decided to construct three separate Infrared Sensors from parts commonly available at electronic stores.

4.5.3 Design and Construction of IR Sensor

The design and construction of the Infrared sensors took a very large portion of the time and effort spent on this project. Lack of information on the chosen parts made it difficult to realise their true operations and many other factors postponed the successful creation of the sensors.

After thoroughly researching other robot builders' attempts to build IR sensors, a commonly available 38 KHz receiver was chosen from Jaycar Electronics along with the same IR LED's used in the odometry sensors. To tune the IR light suitable for the receiver it was decided to use a 555 timer in astable operation to pulse the light at a square wave at 38 KHz with a duty cycle of 50% in accordance with the data sheet of the receiver. This circuit was simple to construct and the schematic shows the detail below in figure 4.11.

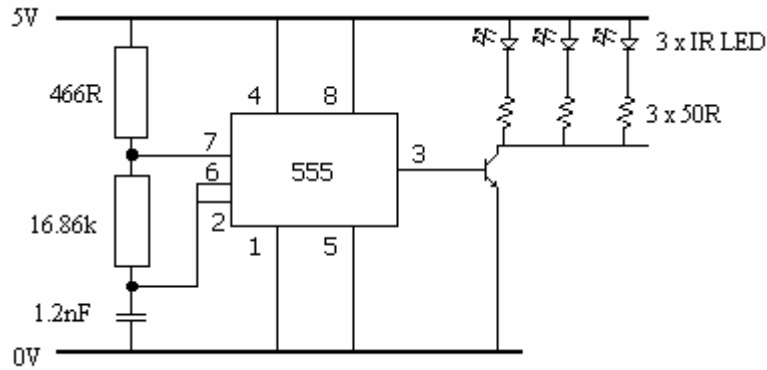


Figure 4.11: 555 Timer Pulsing Sensor Light

Initial calculated results differed substantially from the measured performance. It was necessary to connect a Cathode Ray Oscilloscope (CRO) to accurately measure the waveform output. Slight tuning of resistor values provided an accurate 38 KHz square wave with 50% Duty Cycle.

The transistor is required on the output of the 555 timer to source the current needed to supply the three IR LED's. Each LED can handle 50mA constantly running through them, however due to the 50% duty cycle; the LED will only be on 50% of the time. It can then be safely assumed that the LED's can actually handle 100mA as long as they are only run through the 555 timers pulse.

To contain the IR LED and project the light in the desired direction, it was decided to modify and use some discarded LED keychain torches. After removing the batteries, switch, resistor and white light LED; an IR LED along with the 50 ohm resistor were soldered on to the Printed Circuit Board (PCB). From the PCB, two wires were run out the back of the torch to the 555 timer circuit and then the unit sealed to prevent any IR light escaping which would create undesirable results. Figure 4.12 and 4.13 show the torch modified into an infrared emitter.

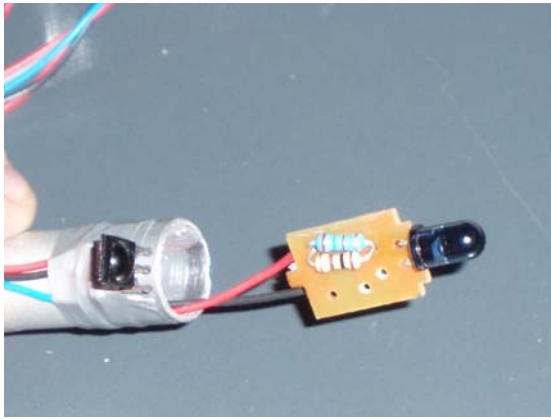


Figure 4.12: PCB Inside LED Torch



Figure 4.13: LED Torch Complete
With Wires

The IR receiving diode should have responded with a voltage drop or even a noticeable resistance change when presented with the torch; however no change was found whatsoever. To check the frequency response of the chip, a function generator was hooked into the base of the transistor at the output of the 555 circuit. The chip was tested for response from 1 Hz up to a few hundred kHz but absolutely no response was found. No datasheet was available for this chip as it was sold solely and had no markings to distinguish it by. After reading other circuits that seemed to contain similar chips, it was found that a pull-up resistor was most likely necessary for correct operation. By adding a 2k2 ohm pull-up resistor to the output of the receiving chip a change of approximately 0.5V could be obtained over a distance of 100mm. This was far from ideal however a comparator circuit would be able to turn this small analog signal into a clear digital response. It wasn't until the comparator circuit had been constructed on a breadboard that it was found that sunlight shining through a window was affecting the response of the sensor. Due to the large complexity of the comparator circuitry to achieve such a simple task, it was decided to scratch this idea and start from a different perspective.

After a lot of searching on the internet for different IR receiving components, a DSE component, part number Z1955, was discovered that claimed to provide a digital signal in relation to the presence of 38 kHz light. This seemed much more practical than the analog signal chips, as measuring distance is not a requirement of the Mobile Manoeuvring Robot. After powering up the new chip and shining the torch off the 555 timer circuit a 5V drop could be obtained over a distance of approximately 500mm.

This was ideal for the purpose of the robot and so more of these chips were obtained. The Z1955 datasheet details a suggested bypassing and filtering circuit containing a few resistors and capacitors. A single circuit board was soldered containing three of these bypassing and filtering circuits and three receiving chips. Testing of these revealed that the first prototype circuit still worked fine, however the two new circuits (identical in every manner) would only work at a distance of approximately 20mm. Due to the fact that one of the receiving circuits worked successfully, the mistake was made of assuming that the IR transmitter circuit was without fault. It wasn't until after many hours of testing and different circuitry being implemented that it was found to that the 555 timer circuit was causing relatively large voltage spikes in the power supply in time with the 38 kHz light pulse. Due to the fact that the emitter and receivers were being powered from the same voltage supply, the noise was affecting the operation of the receiver chips. It is still unsolved why one chip worked and the remaining two failed; however overcoming this problem initiated some desired results.

A group of capacitors wired in parallel on the input of the transmitter circuit provided a noise filter to the power supply. A combination of a large Electrolytic capacitor, medium Tantalum capacitor and a small Monolithic capacitor provide an effective noise filter to combat fast ripples in the power supply. The following schematic details the chosen capacitors and placement in the power supply circuitry.

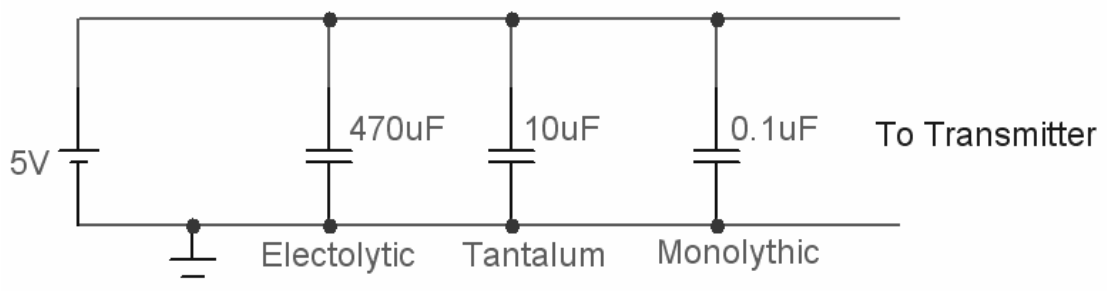


Figure 4.14: Power Supply Filter for Transmitter

An LM7805 voltage regulator is used to supply the 5 volt source from the same battery that supplies the Motorola HC12.

The noise filter, illustrated in figure 4.14, reduced the noise to a negligible level. From this, the Z1955 receiving chips delivered a different behaviour. First experiments on the bread board provided three working receivers over a distance of approximately 300-400mm shown by a 5V drop. As this was suitable performance for the Mobile Manoeuvring Robot, the circuitry was soldered to a circuit board. Testing at a later date showed none of the circuits working again. After another long, strenuous testing period it was discovered that the Z1955 chips were also sensitive to sunlight and incandescent light, despite the claims of the data sheet. If an IR light source was provided to the robot, such as an incandescent bulb, the sensors would work successfully to a distance of approximately 300 mm. Without the additional IR light source the sensors would sometimes work at a distance of 20 – 30 mm.

To overcome this problem it was finally decided to implement some variable brightness IR LED's to supply IR light to the back of the receiving chips. The Z3235 LED can handle at most 50mA constant current. A 1k variable resistor was implemented in series with a 100 ohm resistor so to provide a variable current which could not exceed 50 mA. Calculations prove that the current can be varied between 4.5 and 50mA:

$$\begin{aligned}
 I_{MAX} &= \frac{V}{R_{MIN}} & I_{MIN} &= \frac{V}{R_{MAX}} \\
 &= \frac{(5)V}{(100+0)\Omega} & &= \frac{(5)V}{(100+1000)\Omega} \\
 &= 50mA & &= 4.55mA
 \end{aligned}$$

Where: I = Current flowing through LED [Amps];
 V = voltage drop across LED [Volts];
 R = resistance in series with LED [Ohms].

At 4.5 mA the Z3235 LED is hardly on but does produce some IR light. This variable brightness light provides a method of controlling the sensitivity of the IR receiver and hence a control of the distance sensed. The IR LED's provide enough IR to ensure successful operation of the receiving chips and hence the accomplishment of the distance sensors.

Figure 4.15 gives an overview of how the pulsing IR LED, Receiving diode and variable IR LED are positioned to provide a successful distance sensor.

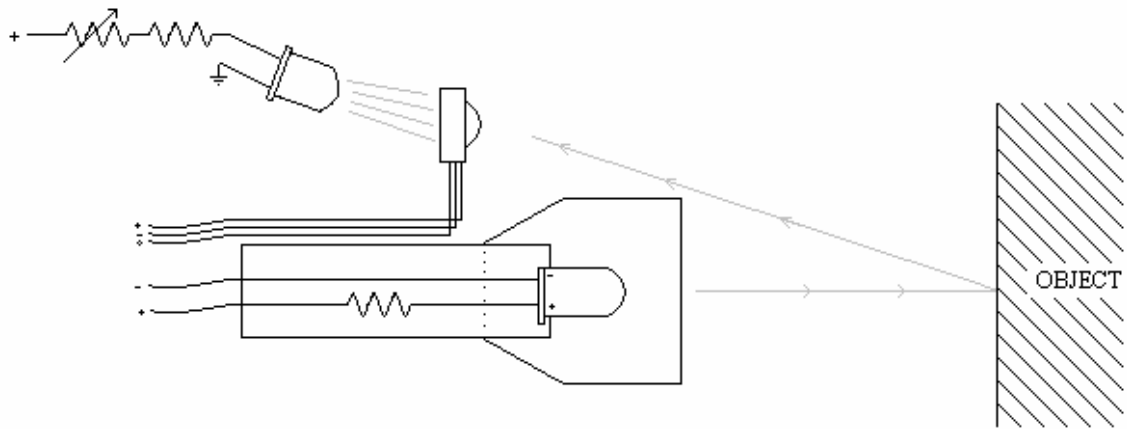


Figure 4.15: Position of Sensor Components

The Z1955 IR Receiver is mounted onto the torch containing the pulsating IR LED. This provides an ideal position to pick up any objects located directly in front of the sensor. A wider sensing range could be created by shortening the outer housing of the torch or using extra pulsating LED's concentrating in the desired direction.

The sensors are mounted to point in each possible direction of travel. This includes a front, left and right sensor. The code is designed to stop the travel of the robot if anything activates one of the sensors and then perform a task specific to the status of each sensor. The figure below shows the mounting of the sensors.

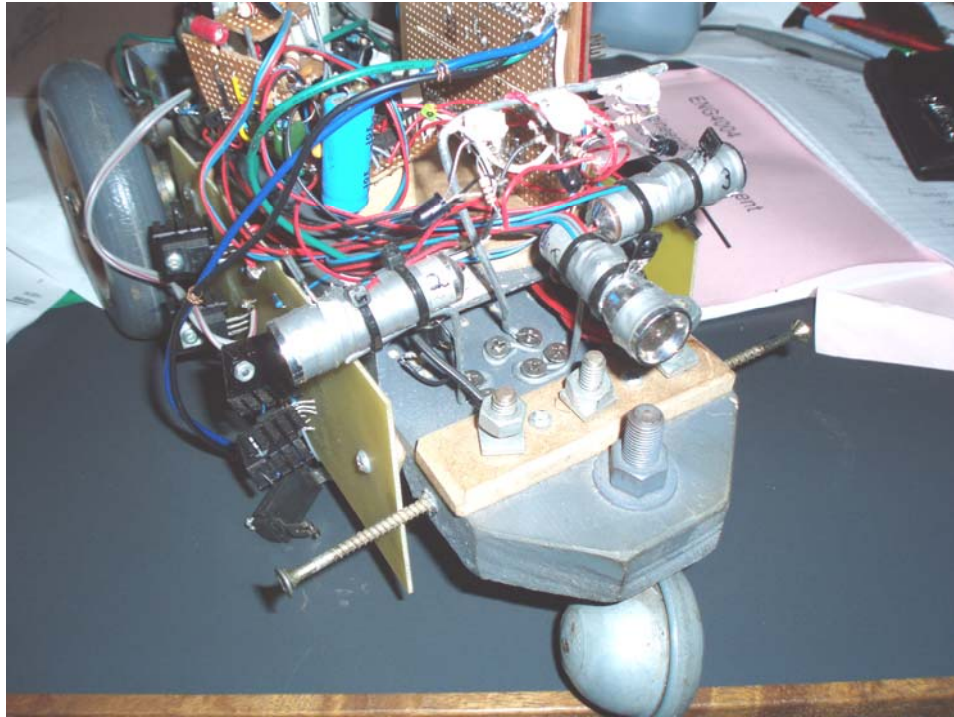


Figure 4.16 Sensor mounting on the Robot

There were two main options to communicate between the sensors and the microcontroller. Either the status wire from the receiver could be connected as an interrupt to the microcontroller; or it could be connected as a data line to any available port. Due to the fact that the robot's guidance is more interested in knowing the current status of each of the sensors rather than knowing exactly when each is triggered, using the data line approach is far more effective. One advantage of this method is that noise filtering can be used by checking the sensor a number of times to ensure there is actually an object in front of the sensor and that the signal was not just caused by noise. After checking the sensors, the robot can make an informed decision on which direction to travel. Not using interrupts also allows checking of each of the sensors during any part of the code or in any interrupt service routine.

Each of the sensors is connected to PORT A (pins 0-2). After originally connecting the wires directly into the port so that when the sensor was active it would read logic zero and non-activated would read logic one, it was deemed more practical if the data was inverted. More accurate calculations can be made on this data and the code will be a lot easier to understand and debug. To invert this data, a HD741S00P (quad two-input NAND gate) IC was used with the inputs of each individual NAND tied together to

provide inverters. The logic signal from the sensors was then applied through these joined inputs and the output provided an inverted signal of the input.

The program code simply checks the status of each sensor in a loop by loading PORT A into an accumulator and performing logic decisions on the value loaded. The microcontroller now reads a logic one for an activated signal and a logic zero for a non-activated sensor. Hence, if all sensors are activated the register will read 00000111 or if no sensors are activated it will read 00000000.

To summarise the distance sensors used on the Mobile Manoeuvring Robot the following points can be observed.

- Infrared LED's (DSE part number Z3235) clocked at a 38 kHz square wave with an even mark to space ratio are used for the transmission of the sensors light.
- Buffered Infrared receiver chips (DSE part number Z1955) with the recommended filtering and bypassing are used to pick up the transmitted light.
- Extra IR LED's with variable brightness are used to saturate the IR receivers to ensure successful performance
- The logic outputs of the sensors are inverted to communicate effectively with the Motorola HC12
- Sensor status is read in by the microcontroller as data on PORT A.

In retrospect to designing custom IR sensors for the simple use of obstacle detection, it would be far more practical to spend the money on tuned sensors that work without error. The robots sensors work reliably now; however an enormous amount of time was spent on designing these which would have been more effectively used on the objective and practicality side of the design.

4.6 The Software

The software used to control the Mobile Manoeuvring Robot was created in assembly language for the Motorola HC12. This allowed direct communication with the microcontroller and convenient access to all of its ports and functions. The software used to control the heading of the robot follows a relatively simple structure with small subroutines for each separable function or routine. Interrupts are used for the odometer counting and a tight loop in the main function keeps track of the direction of travel and sensor readings.

A software design procedure (available in Appendix C) was used to analyse the problem and deal with suitable specifications and provide an appropriate software structure. The software design procedure encompassed the major points of software design; such as objectives, user information and program structure. The objective of the software was stated to be *“To control the mechanical hardware of a two drive-wheel robot in order to avoid collision with obstacles. The software must be able to control the unit to travel in a straight line between objects and is based on assembly level language of the Motorola 68HC12 microcontroller.”* It was necessary to base the software structure around this as well as encompassing the user information. The user information is listed below.

User Information

The code has five sensors that input to the software. Three of these are obstacle avoidance sensors read in as data through port A and the other two are concerned with odometer readings read in as interrupts through port G. The program starts at \$0400 with data starting at \$0200. The stack pointer must be set in a clear location such as \$0700 and the program counter to \$0400 to start the procedure.

When run, the program will output through the serial port the status of the odometer readings and also controls the speed and direction of both motors to achieve the desired performance. The program loops continuously to give real time control the mechanical components

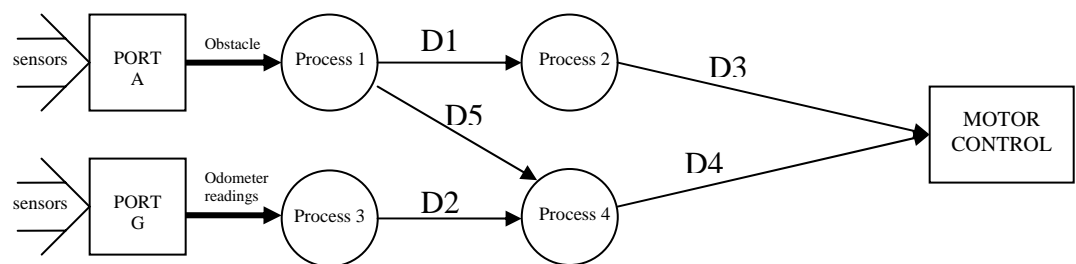
From these listed objectives and user information, while also including any other specific requirements, a structure was decided upon to control the Mobile Manoeuvring

Robot. It is in the form of a tight loop main routine that branches out when individual activities require to be completed. The following outline was created to establish the necessary activities to be completed in the main loop and the appropriate related subroutines.

While driving do the following:

- *Read the odometer sensors and increment odometer counters*
 - *Check odometer counters and adjust speed for straight driving*
- *Watch obstacle sensors in a tight loop act accordingly when sensors triggered.*
 - *If left is clear, turn left otherwise turn right*
 - *If left and right blocked, reverse*
- *Continue on a straight path*
- *Output data through serial port for debugging purposes*
- *Repeat process continuously*

The data will flow as follows, coming from the five IR sensors and being manipulated and processed to obtain the required drive on the motors. There are four main processes that take part in this data processing.



D1 = Signal of which direction to turn
 D2 = odometer counters
 D3 = motor information for turning
 D4 = motor information for drive straight
 D5 = jump to process 4 when required

Figure 4.17: Data flow for software

Process 1:

This is the main section of the program. It runs a tight loop checking for data from the sensors on port A and also checking the odometer readings to initiate a compare function to maintain straight line driving.

Process 2:

Process 2 is a large subroutine called every time a turn is needed. It accesses data from port A. It checks sensors and turns relative to their status. After the motors have been adjusted to gain desired performance the code returns to the main loop, process 1.

Process 3:

This is an interrupt service routine that determines which odometer sensor has triggered and increments the respective counter. The counter is a byte stored in memory and is cleared regularly to avoid overflow issues.

Process 4:

This process is a subroutine used to maintain straight line driving. It is called from the main program loop and works by comparing the odometer counters and adjusting motor speeds to maintain a constant even speed between the motors. This results in even wheel speed hence straight driving.

The above set-out of structure and processes has allowed the following code to be written and implemented into the Mobile Manoeuvring Robot. The next sections will break the code down into segments and subroutines and detail the operation of each function. From this, the code can be easily understood and modified to suit similar purposes. A full code listing is available in Appendix B.

4.6.1 Address Definition

```

“ ;ADDRESS DEFINITION
PORTA      equ $00   ;Port A Data
PORTB      equ $01   ;Port B Data
DDRA       equ $02   ;Port A Data Direction
...
...
...
ADR16L     equ $1FD   ;
ADR17H     equ $1FE   ;
ADR17L     equ $1FF   ;
.....”

```

This section of code is simply written to show and name all of the available addresses of ports etc inside the Motorola HC12. Credit for this section of code goes to Mr Terry Byrnes. The rest of the program then references these address names for ease of reading, understanding and debugging. For instance the code to output hexadecimal 10 would be

```
MOV B #10, PORTA
```

This is much easier to understand than a code that says

```
MOV B #10, $00
```

Without knowing the addresses of every single port it would be near impossible to decipher the assembly language code.

4.6.2 Variable Definition

The next step in the code is variable definition. This simply defines all of the variables that will be used throughout the program and defines a location for their storage in memory. The variables start at address \$0200 and list onwards from this address.

```

;*****VARIABLE DEFINITION*****
                                ORG $0200                                START DEFINITION AT ADDRESS $0200
ODMA    DC.B    0
ODMB    DC.B    0                                ;DEFINE VARIABLES
COUNT  DC.B    0

```

4.6.3 Main Program

After variable definition is the main program control loop. This section is responsible for initialising all of the hardware and ports. The bottom tight loop is responsible for ensuring the robot drives straight and also that the sensors are checked frequently for object detection.

```

;*****MAIN PROGRAM..CONTROL LOOP*****
;NOTE: NO ACTION WILL OCCUR UNTIL THE IRQ INTERRUPT IS ACTIVATED UNLESS
FRONT SENSOR IS TRIGGERED

                                ORG    $0400                                ;START PROGRAM STORE AT $0400
MAIN    LDS    #$0700                                ;SET STACK POINTER

```

```

        CLR      COPCTL      ;DISABLE COMPUTER OPERATING
                               NORMALLY WATCHDOG
        JSR      INITSCO     ;INITIALISE SERIAL PORT
        JSR      INITKWG     ;INITIALISE PORT G
        JSR      INITPWM     ;SET UP PULSE WIDTH MODULATION ON
                               PORT P
        JSR      INITIRQ     ;INITIALISE IRQ INTERRUPT
        JSR      INITPA
WAIT    BRCLR    PORTA,$01,SKP ;CHECK FRONT SENSOR AND IF ON
                               BRANCH TO TURNL
        JSR      TURNL
SKP    LDAA     COUNT
        CMPA    #$5A        ;HAS MOTOR A COUNTED xx TIMES YET
                               (xx/2 REVOLUTIONS)
        BMI     WAIT        ;IF NOT ,WAIT AND CHECK AGAIN
        JSR     COMP        ;IF SO JUMP TO COMP (compare function)
        BRA     WAIT        ;AFTER COMP GO BACK AND WAIT FOR
                               ANOTHER xx/2 REVOLUTIONS

```

The main loop code runs indefinitely until the reset button on the microcontroller is pressed or the power source is interrupted. Firstly it checks port A to determine whether the front sensor is triggered. If port A shows the front sensor has been triggered the program branches out of the main loop to the turn left subroutine. This subroutine will be explained further on. If the sensor does not show to be triggered the main loop continues on to check how many times it has been since the odometer was serviced. It does this by loading the number of revolutions motor A has turned and comparing it with a set number, currently \$5A. This number can be adjusted to alter the performance of the robot. If motor A has not turned \$5A counts, the main program branches back to the start of the loop and continues checking the sensor followed by the odometer counters. Once motor A has turned \$5A counts, the main loop branches to the subroutine 'COMP' which compares motor A counts with motor B counts and adjusts the motor speeds accordingly to achieve straight line driving. This subroutine will also be detailed later.

4.6.4 DRFWD Drive Forward Subroutine

This subroutine is very basic and is called in order to start the robot in a forwards motion. Bytes relative to the desired speed are moved into the PWDTY address that

applies to the motor to be run. The byte moved is between \$00 and \$FF and is the percentage high time of the duty cycle of the pulse used to control the speed. A value of \$FF will be full time on, hence maximum speed, and \$00 will be full time low, hence not moving at all. The value \$A0 has been chosen from physical observation to define the desired speed as smooth but reasonable pace.

```
*****DRIVE FORWARDS*****
DRFWD  MOVB    #$A0, PWDTY0    ;DUTY CYCLE FOR CHANNEL 0|
        MOVB    #$A0, PWDTY2    ;DUTY CYCLE FOR CHANNEL 2| **DRIVE
        ;FORWARDS
        RTS                ;RETURN FROM SUBROUTINE
```

4.6.5 Turning subroutines

The subroutines used to turn in either the left or right direction are very similar and hence only one subroutine will be detailed. The TURNL code is explained in the following.

Before the robot will turn left, several sensors are checked to ensure the turn signal was not generated by noise and also to ensure there are no objects in the direction of turn. Firstly the front sensor is checked repeatedly to ensure it is definitely set and that the signal was not generated by noise. This is done by loading a number into index register X and then looping that number of times, checking the sensor and decrementing X each loop. If at any time the sensor does not read triggered, the instruction to turn left will be aborted and the program will return to the main loop. Once the loop has been executed X times and is guaranteed to be a legitimate turn request, the left sensor is checked to ensure there is room to turn. If the left sensor is clear the code proceeds with a left turn. If not clear the code branches to the turn right subroutine.

In order to turn left, a duty cycle value is written to the left wheel PWDTY register in order to turn it backwards. The odometer for motor A is cleared and watched in a tight loop until it reaches the specified number of turns relative to a right angle turn. Running the motors from an 11V power supply proved decimal 90 counts to be a 90 degree turn. This takes into account the wind down time of the motor (the inertia continues the wheel turning for a small time after voltage is turned off). Once the robot has come to a halt after turning, it continues on a forward path. Between all of the direction changing

commands a delay is implemented to reduce some of the jerky motion. The delay subroutine is used to create this delay.

```

;*****TURNING CODES*****
TURNL   LDX     #$0A00
LP      BRCLR  PORTA,$01,FALSE   ;/
        DEX                      ;LOOP TO CHECK SENSOR $0A00 TIMES
        DEX                      ;TO ENSURE NOT JUST NOISE
        BNE    LP                  ;/
        BRCLR  PORTA,$04,SKR      ;IF LEFT SENSOR(3) BLOCKED, TURN
        JSR    TURNR              ;RIGHT INSTEAD
SKR     LDAA   #'L'
        JSR    TXBYTE
        MOVB   #$00,PWDTY0        ;/
        MOVB   #$00,PWDTY2        ;/STOP MOTORS
        JSR    DELAY              ;WAIT FOR DELAY PERIOD
        MOVB   #$00,ODMA          ;/
        MOVB   #$00,ODMB          ;/CLEAR ODOM COUNTERS
        MOVB   #$80,PWDTY1        ;TURN MOTOR B BACKWARDS
CONTL   LDAA   #$00
        LDAB   ODMB
        CPD    #90                 ;SET NUMBER RELATIVE TO A 90 DEGREE
        BNE    TURN               TURN
        BLT    CONTL              ;WAIT FOR TURN TO COMPLETE
        MOVB   #$00,PWDTY1
        JSR    DELAY
        JSR    DRFWD              ;CONTINUE FORWARDS
FALSE   RTS                       ;RETURN FROM SUBROUTINE

```

The backup subroutine is called only after the code has deemed it impossible to move forward or turn in either direction. The code is designed to only reverse long enough to clear one side of the robot and allow it to turn in that direction.

Initially it ensures both motors are stopped, and then turns both wheels backwards at the same speed as it normally drives forward. The code then drops into a loop which waits for either side sensor to clear. First it checks the left sensor and turns left then exits the loop if clear; followed by the right sensor and turns right and exits if it is clear. If neither is clear then the loop continues until either one of the directions is vacant. After

the robot has turned in either direction the backup subroutine is exited and the main loop continues.

```

BACKUP ;LDAA    #'K'                ;SEND K THROUGH SERIAL PORT
        ;JSR    TXBYTE              ;(DEBUG)
        ;MOVB   #$00,PWDTY0         ;/
        ;MOVB   #$00,PWDTY2         ;/STOP MOTORS (DRIVING FORWARDS)
        ;JSR    DELAY               ;WAIT DELAY PERIOD
        ;MOVB   #$A0,PWDTY1         ;/
        ;MOVB   #$A0,PWDTY3         ;/REVERSE STRAIGHT
WFCLR  BRCLR   PORTA,$02,OUTL       ;IF LEFT NOT SET BRA TO OUTL
        ;BRCLR  PORTA,$04,OUTR      ;IF RIGHT NOT SET BRA TO OUTR
        BRA     WFCLR               ;IF L&R SET, BRA WFCLR. WAIT FOR L OR
                                   R TO CLEAR
OUTL   JSR     TURNL                ;TURN L WHEN L SENSOR IS CLEAR
OUTR   JSR     TURNR                ;TURN R WHEN R SENSOR IS CLEAR
        JSR     DELAY
        RTS                          ;RETURN TO MAIN LOOP

```

4.6.6 Delay

The delay code works with a loop inside a loop decrementing a value slowly. The inner loop increments index register X by one for every loop, until it ‘rolls over’ from \$FFFF to \$0000. When index register X reaches zero, the outer loop value (stored in index register Y) decrements by one. Thus, the delay is the time it takes to increment X \$FFFF times, multiplied by the number initially stored into index register Y. To adjust the delay time it is easiest to adjust the initial value in index register Y.

```

;*****DELAY*****
;CODE TO ADD A DELAY TO EASE THE SWITCHING OF MOTORS
;GIVES APPROX 2 SEC DELAY. CAN SHORTEN TIME BY INCREASING INITIAL Y
                                   VALUE
DELAY LDY     #$FFDC                ;INITIAL VALUE FOR Y
DELA  LDX     #$0000                ;INITIAL CONDITION OF X
DEL   INX
        BNE   DEL                  ;IF Z NOT = 0 BRANCH TO DELAY
        INY
        BNE   DELA
        RTS

```

4.6.7 COMP Compare Subroutine

The entire purpose of the COMP subroutine is to compare the speeds of each motor and ensure they run in parallel. This ensures straight line driving which was one of the major project guidelines. This function is called by the main loop every time the odometer for motor A reaches its defined value. When this occurs, it writes a new line through the serial port then compares ODMA (odometer of motor A) with ODMB (odometer of motor B). If ODMA is larger than ODMB then motor A is spinning too fast, therefore the subroutine branches to the label 'A2FAST' where the motor speeds are adjusted. Likewise if ODMB is larger than ODMA the subroutine branches to the label 'B2FAST'.

A2FAST works on a simple rule. Before simply reducing the duty cycle in the PWDTY register it is necessary to check whether that duty cycle is above a certain threshold. Without this check, the robot would eventually slow to a stop. If the duty cycle is lower than this threshold, in this case \$65, instead of slowing Motor A it will instead increase the speed of motor B slightly. Likewise the routine B2FAST works on the same threshold, if the duty cycle drops below the threshold, it will speed motor A instead of slowing motor B. Of course if the duty cycle is above the threshold it will simply reduce the speed of the respective motor.

```
;*****COMPARE SPEEDS*****
;CODE TO COMPARE MOTOR A AND B SPEEDS AND CONTROL H BRIDGE

COMP JSR      NEWLINE      ;START NEW LINE ON SERIAL PORT
      LDAA    ODMA         ;LOAD MOTOR A COUNT INTO ACC A
      CMPA   ODMB         ;COMPARE MOTOR A WITH MOTOR B
      BPL    A2FAST       ;IF MOTOR A FASTER THAN MOTOR B
      BMI    B2FAST       ;IF MOTOR A SLOWER THAN MOTOR B
;-----
A2FAST JSR    NEWLINE      ;CASE MOTOR A IS FASTER THAN B
      LDAA   #'a'         ;TX BYTE THROUGH SERIAL (DEBUG)
      JSR   TXBYTE
      LDAA   PWDTY2      ;LDAA SPEED OF MOTOR A
      CMPA  #$65         ;IF MOTOR A IS SLOWER THAN $65
      BMI   FASTB        ;SPEED UP MOTOR B
```

```

        SUBA    #$01                ;IF MOTOR A IS FASTER THAN $C5
        STAA   PWDTY2              ;SLOW MOTOR A SLIGHTLY
        BRA    DONE                ;EXIT
FASTB  LDAA   PWDTY0              ;/
        ADDA   #$01                ;|SPEED B UP ONE
        STAA   PWDTY0              ;/
        BRA    DONE

;-----
B2FAST JSR    NEWLINE            ;CASE MOTOR B IS FASTER THAN A
        LDAA   #'b'
        JSR    TXBYTE            ;TX BYTE THROUGH SERIAL (DEBUG)
        LDAA   PWDTY0            ;LDAA WITH SPEED OF MOTOR B
        CMPA   #$65              ;|
        BMI    FASTA            ; | IF MOTOR B GOING TOO SLOW, SPD UP
        A
        SUBA   #$01              ;/
        STAA   PWDTY0            ;|SLOW MOTOR B SLIGHTLY
        BRA    DONE
FASTA  LDAA   PWDTY2              ;/
        ADDA   #$01              ;| SPEED A UP BY ONE
        STAA   PWDTY2            ;/

DONE  MOVB   #$00, COUNT        ;/
        MOVB   #$00, ODMA        ;| CLEAR COUNTERS
        MOVB   #$00, ODMB        ;/

        RTS                      ;RETURN AND WAIT FOR NEXT COMPARE

```

4.6.8 KWGISR Key Wake Up Port G

This is the code written for the interrupt service routine which is responsible for incrementing the odometer counters every time the microcontroller receives a signal from the odometer sensors. Basically, when the interrupt port is triggered by a falling edge on any port G pin (5v down to 0v), the code checks each connected pin (pins 0 and 1) to determine which sensor triggered the interrupt. If pin 0 triggered the interrupt then it loads ODMA and adds 1, if it was pin 1 that triggered it loads ODMB and increments it. If both pin 0 and 1 appear to not have triggered the interrupt then the subroutine will simply exit and the main program will continue to run.

```

;*****KEY WAKE UP PORT G *****
;ADDS 1 TO COUNTER FOR CORRESPONDING MOTOR
;***SENSOR G 0*****
KWGISR      BRCLR      KWIFG,$01,NOTG0      ;CHECK BIT 0
            LDAA      ODMB      ;IF MOTOR B, LDA WITH ODOM COUNT B
            ADDA      #$01      ;ADD 1 TO ODMB COUNT
            STAA      ODMB      ;STORE BACK TO ODMB
            LDAA      #'B'
            JSR       TXBYTE      ;WRITE TO SERIAL PORT (DEBUG)

;***SENSOR G 1*****
NOTG0 BRCLR  KWIFG,$02,NOTG1      ;CHECK BIT 1
            LDAA      ODMA      ;IF MOTOR A, LDA WITH ODOM COUNT A
            ADDA      #$01      ;ADD 1 TO COUNT
            STAA      ODMA      ;STORE BACK TO ODMA
            LDAA      COUNT
            ADDA      #$01
            STAA      COUNT
            LDAA      #'A'
            JSR       TXBYTE      ;WRITE TO SERIAL PORT (DEBUG)
;*****FALSE INTERRUPT*****
NOTG1 MOVB   #%00000011,KWIFG ;CLEAR WAKE UP FLAGS

RTI      ;RETURN FROM INTERRUPT

```

4.6.9 IRQISR IRQ Interrupt Service Routine

This interrupt service routine waits for the falling edge on the IRQ interrupt. This is input to the microcontroller from a push button switch which is intended to start the operation of the robot. When the button is pressed, the service routine delays before branching to the DRFWD subroutine which initiates a forward motion. The delay provides some software switch bounce as this interrupt is set to be edge sensitive only.

```

;*****IRQ SERVICE ROUTINE*****
;THIS INTERRUPT STARTS THE ROBOT WHEN BUTTON IS PRESSED
IRQISR JSR    DELAY      ;PROVIDES SOME SWITCH BOUNCE
        JSR    DRFWD     ;BEGIN DRIVING
        RTI    ;RETURN FROM INTERRUPT

```

4.6.10 Initialisation of Ports and Interrupts

There are several different subroutines used to initialise ports and define interrupt control. One exists for each of the following: Pulse width modulation, port P; Port A; Key wake up Port G and the serial port. It is essential that these initialisation subroutines are run at the start of the program before any actual running commences. This ensures the interrupts are set ready to work, the serial port is communicating and that all input and output is taken care of.

The Pulse Width Modulation (PWM) is the method of controlling the speed of the motors. The Motorola HC12 has four channels devoted to PWM and can easily be set using the following code. Each of the bytes moved into the PWM control registers provides a certain setting required for the desired use of the PWM channels. Clear explanation of each bit can be found in the data sheet for the Motorola HC12. First specification set is the pre-scaler. This is used to slow down the clock for use as the PWM. Extremely high frequencies are not desired as they create excess power loss and heating through the H Bridge switching. Used here is a pre-scaler of 128 which means the frequency of the PWM will be 128 times smaller than the frequency of the microcontroller clock. The PWPOL byte takes care of which clock to use for each channel and also the polarity of the duty cycle for each cycle. PWEN simply enables all the selected lines to be output regardless of the data direction register. PWCTL sets port P to have an active pull up device enabled.

```

;*****
;SETUP PWM Channels

INITPWM MOVB    #%00111111,PWCLK    ;PRESCALER OF /128
          MOVB    #%00001111,PWPOL    ;PPOL
          MOVB    #%00001111,PWEN     ;ENABLE ALL OUTPUTS
          MOVB    #%00000010,PWCTL    ;PULL UP DEVICE ENABLED

SETPWM MOVB    #$FF,PWPER0           ;/
          MOVB    #$FF,PWPER1         ;/PERIOD FOR PWM
          MOVB    #$FF,PWPER2         ;/
          MOVB    #$FF,PWPER3         ;/

```

```

RTS                                ;RETURN TO MAIN

```

Initialisation for port A simply involves setting the three used pins for data input and setting all of the pins to initially start with logic low. This ensures no sensor readings take place until the sensor actually detects an object.

```

*****INITIALISE PORT A*****
INITPA MOVB    #%11111000,DDRA    ;SET PORT A BITS FOR OUT AND IN 0 =
                                   INPUT
      MOVB     #%00000000,PORTA   ;ENSURE PORT A INPUTS ARE ZEROES
                                   WAITING FOR ACITVE HIGH
RTS

```

Initialising port G for key wake up behaviour involves setting conditions in four separate registers. DDRG sets the data direction for port G, the used pins need to be data input while the rest can be either. Writing to KWIFG clears the wake up flags on the desired pins and KWIEG enables the selected pins to be key wake up interrupts. This means that as soon as the edge is detected the interrupt will automatically service. The PUCR register is used to enable or in this case disable all of the pull up resistors on selected ports. As no pull up resistors are desired on any ports in use it is simplest to write all zeroes to this register. Storing the opcode for the jump command into the pseudo vector notifies the interrupt what to do when the interrupt is triggered. Also mentioned here is the address to jump to. This is the interrupt service routine that was outlined above. It is essential that the interrupt mask bit is cleared after these registers have been initialised.

```

*****INITIALISE PORT G FOR KEY WAKE UP*****
INITKWG MOVB   #%11111100,DDRG    ;BITS 0-1 AS INPUT
      MOVB     #%00000011,KWIFG   ;CLEAR WAKE UP FLAG 0-1
      MOVB     #%00000011,KWIEG   ;ENABLE KEY WAKE UP INTERRUPTS
      MOVB     #%00000000,PUCR    ;SET PORT G, H IRQ AND XIRQ FOR PULL
                                   DOWN (DISAHLE ALL PULL UPS(ALL 0'S))
      LDAA    #$06                ;JMP COMMAND OPCODE
      STAA    $07B8               ;RTI PSEUDO VECTOR
      LDD     #KWGISR             ;ADDRESS TO JUMP TO, KWG ISR
      STD     $07B9
      CLI                                ;CLEAR INTERUPT MASK
RTS

```

The serial port requires having the baud rate set and also to enable the transmitter. It is also adequate to enable the transceiver which accomplishes the same task. The baud rate value is chosen and then a corresponding number is found from the baud rate generation table in the HC12 data sheet. Loading accumulator A with the contents of SCODRL clears the flags inside that register and readies the serial port for use.

```

;*****SERIAL PORT*****
; TO USE THE SERIAL PORT, SET BAUD RATE TO 19200
; BAUD. THE VALUE HERE IS A 16 BIT DIVISOR.

INITSC0 LDD    #26                ; VALUE FROM BAUD RATE GENERATION
                                           TABLE
           STD    SC0BDH
           LDAA   #$0C            ; ENABLE TRANSCEIVER
           STAA   SC0CR2
           LDAA   SC0DRL          ; CLEAR FLAGS
           RTS

```

4.6.11 Serial Port Communications

Two simple subroutines were written for serial port communication. Although not necessary to the correct functioning of the Mobile Manoeuvring Robots performance, they provide an excellent debugging facility. Characters can be sent to a computer screen to understand which parts of code are being run and hence follow the real time branching and code order.

NEWLINE simply loads the ASCII value of a carriage return and uses the TXBYTE subroutine to transmit it to the screen. This is followed by an ASCII line feed being sent in a similar matter. On the computer screen this looks the same as hitting an enter key in a text editor.

```

;*****SERIAL PORT WRITING COMMANDS*****
; SEND A CARRIAGE RETURN AND LINE FEED TO SCREEN
NEWLINE LDAA   #$0D                ;LOAD CARRIAGE RETURN FOR ASCII
           JSR    TXBYTE            ;TRANSMIT TO SCREEN

```



```

LDAA    #$0A        ;LOAD LINE FEED ASCII
JSR     TXBYTE      ;TRANSMIT TO SCREEN
RTS

```

The TXBYTE subroutine checks the serial ports status register to await the clear flag before sending the data stored in accumulator A through the serial port. It is essential to await the clear flag before sending data to avoid data contentions.

```

; TRANSMIT A CHARACTER STORED IN ACCUMULATER A

```

```

TXBYTE    BRCLR    SC0SR1, #$80    ;WAIT FOR CLEAR FLAG
          STAA     SC0DRL          ;SEND DATA THROUGH SERIAL PORT
          RTS

```

```

;*****

```

CHAPTER 5

PERFORMANCE AND DATA ANALYSIS

No specific data was obtained from the results of this dissertation. This is due to the fact that it is a simple performance based outcome. If GPS tracking or similar had been used to control direction there would be headings and position plots to consider. The only way to assess the performance of the Mobile Manoeuvring Robot is to visually assess the performance. This performance however can be broken down into the various aspects of the robot such as mechanical, software, electrical and so on. This chapter discusses the performance of each individual aspect of design and in some cases suggests recommendations to overcome some of the poorer performance aspects.

5.1 Sensor performance

The sensors used in the Mobile Manoeuvring Robot ended up working successfully. A reasonable range was achieved (explained later) and a clean voltage drop was obtained for communication with the microprocessor. Overall these are the only requirements necessary for operation of the robot.

To greatly assess the performance of the sensors a simple test was performed to determine the range of object detection. This involved sensor height range, width range and distance range. Basically an object was held to the side of the sensors view and then moved slowly across the sensors view until the response triggered. By lining this point up to rulers laying in each direction orthogonal to each other (x , y , z axes) a point was recorded in 3d space. This was performed in every direction until each extremity of the sensors range was known. These points can then be plotted using computer software to give a clear indication of the effective distance of the sensors. The object of choice was a small white block of timber which gave an effective reflection of light. The following table of data was obtained from physical experimentation with one of the sensors.

	Coordinate		
Position	X	Y	Z
1	0	200	0
2	50	180	0
3	-55	180	0
4	0	180	65
5	0	180	-50
6	35	180	35
7	-35	180	-35
8	-35	180	50
9	35	180	50
10	45	180	30
11	-45	180	30
12	45	180	-20
13	-50	180	-20
14	20	180	60
15	-20	180	60
16	20	180	-40
17	-25	180	-40

TABLE 5.1: Sensor range

Using computer software to plot these results shows the range the sensor can detect. The following MATLAB code was produced to plot these points as vectors from the origin in three dimensional space.

```

x = [0 50 -55 0 0 35 -35 -35 35 45 -45 45 -50 20 -20 20 -25];
y = [200 180 180 180 180 180 180 180 180 180 180 180 180 180 180 180 180];
z = [0 0 0 65 -50 -35 -35 50 50 30 30 -20 -25 60 60 -40 -40];
loop = 1;
for loop = 1:17;
    plot3([0,x(1,loop)], [0,y(1,loop)], [0,z(1,loop)])
    hold on;
    loop = loop + 1;
end
plot3([0,x(1,1)], [0,y(1,1)], [0,z(1,1)], 'r')

```

Figure 5.1 shows the MATLAB 3D plot which depicts roughly the boundary an object must enter to be detected by the sensor. The red line represents the perpendicular sensing distance and the blue lines show the extremities of the reflected light.

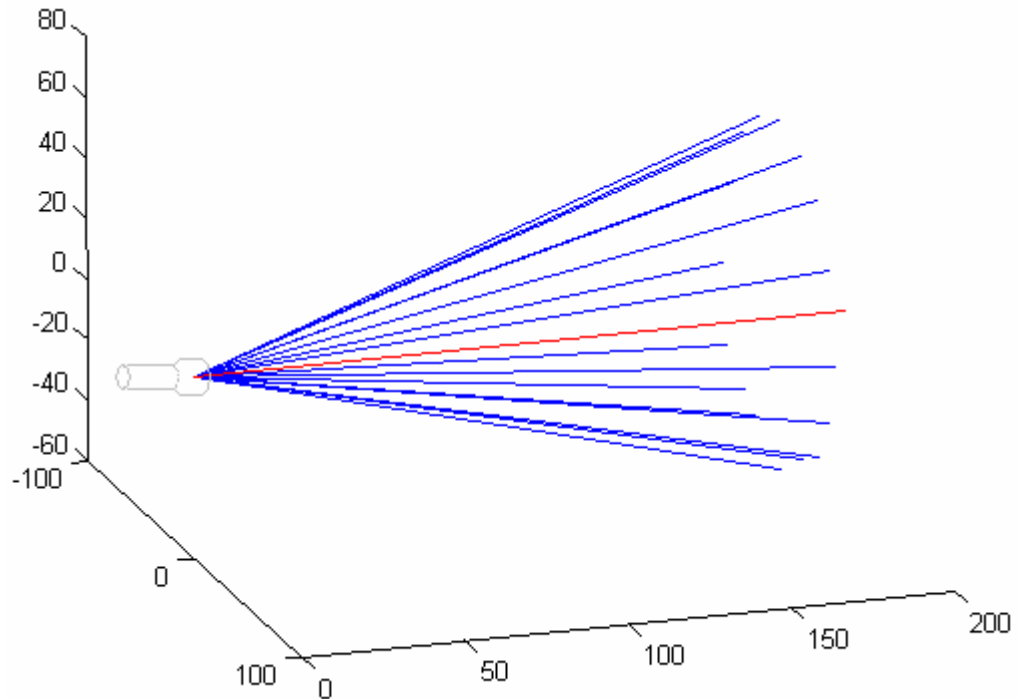


Figure 5.1 Sensor spectrum

This is a practical spread of detectable range. If pointed towards the driving surface the sensors will detect any object that comes within the driving direction of the robot.

After the large time and effort that went into the design, construction and testing explained in chapter 4, it has become evident that the price of commercial sensors is very affordable. If the sensors had been purchased as a whole unit it is presumable that few troubles would have occurred from the vision aspect for the duration of the project. This in turn would have allowed a much greater development of the practical side of the robot. Dead reckoning guidance could have been implemented and a useful purpose such as object retrieval could have been created. For sensors in similar applications it is easy to see that it is worth spending extra money and having a small, compact sensor that can be relied on relatively heavily.

5.2 Mechanical performance

The chassis used for the duration of this project served its purpose sufficiently. It was possible to mount all the required equipment, however room for additional components is exhausted. The major problem with the chassis design at this point is the lack of room for a suitable battery. The battery used can sustain operation for only a short period of time and so either more cells or a new design of battery is required. For this to take place more room is required on the chassis. Also for extending any operation of the robot, such as a retrieval arm, would require chassis space to mount and control.

The only successful way to continue work on this project, including extended operation and achieving smoother running, is to re-design and construct the chassis. A more successful approach to this than the prototype used would be to construct the chassis in a series of layers. Each layer would contain a separate part of operation for the robot. By designing in this manner there is always more room for additional components by adding additional layers until all objectives have been met. After the robot is successful to the point of putting into operation it could be finally re-designed into a more compact unit suitable for practical use or manufacturing. This would encompass such factors as aesthetics, practicality and weight reduction.

Figure 5.2 below shows the application of the layered board system. The bottom layer here contains the drive wheels, motors and H Bridges. It may also be practical to mount the microcontroller here if there is room. Layers are joined by ribbon cables or wires and are bolted together for rigidity. It is important to consider the spacing of the layers as if the robot becomes too tall, stability will suffer greatly.

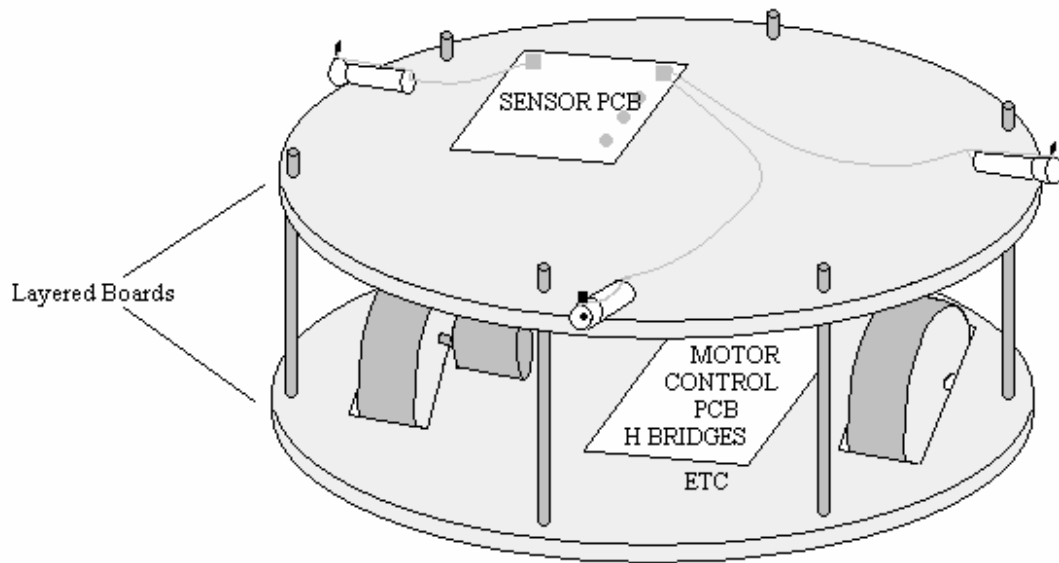


Figure 5.2: Layered approach to robot design.

This design allows greatly for troubleshooting as each layer can be pulled apart separately to give room for soldering, viewing and modifying. It would also be advisable to allow a lower layer for the battery storage as they are likely to be one of the heavier components and mounting them high would affect the robots stability.

5.3 Software Performance

The software as explained above follows a very simple structure. Because of this there are few errors that affect the performance of the Mobile Manoeuvring Robot. As a whole, the software completed the overall objective of the project and so should be recognised as a success.

As a formal test for the straight line driving performance, a pen or chalk can be attached to the unit to track the trajectory of the robot. To achieve this, a chalk stick was taped to the rear of the robot. Ideally it should be mounted midway between the drive wheels however this was not practical for the experiment. The robot was then placed on a clear and clean slab of concrete and set to run. The result was a line with slight curvature at the beginning, but as a whole very straight. The slight curvature at the start is due to the motors starting at full speeds and slowly equalizing. During this time the robot will turn a slight, hardly noticeable curve. Figure 5.3 shows the testing in progress and figure 5.4 shows the results of the experiment.



Figure 5.3: Straight line testing

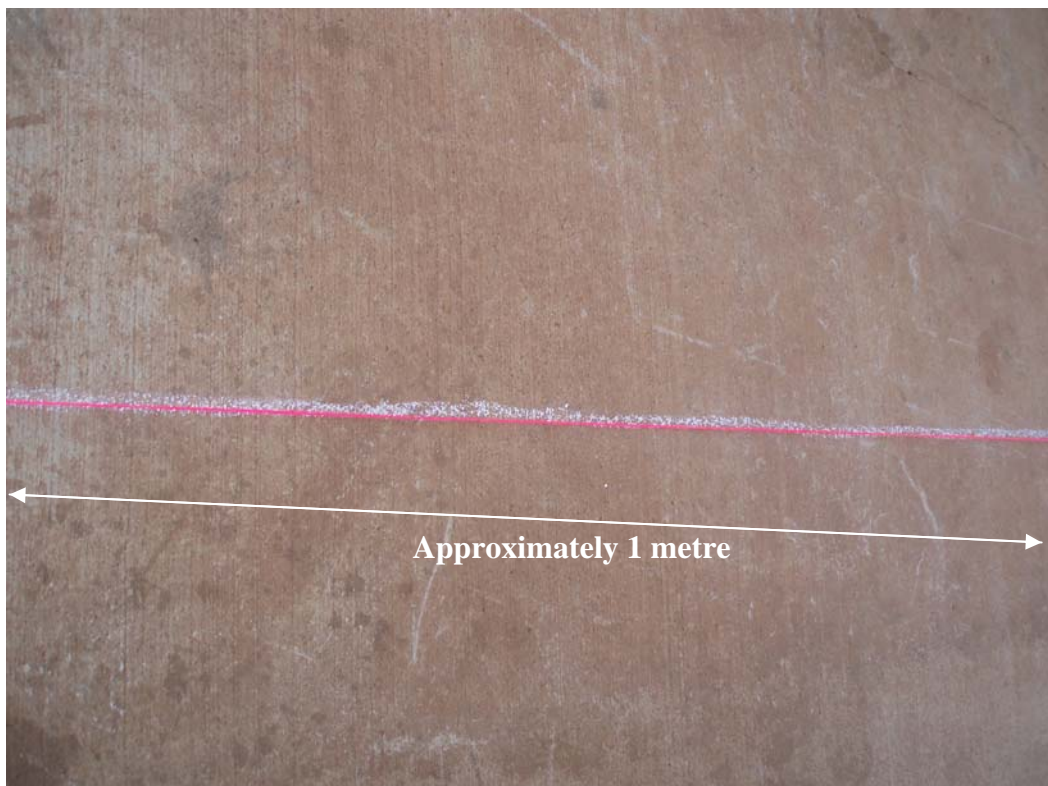


Figure 5.4: Results of straight line test

Figure 5.4 shows the results of the straight line test. This section of line is approximately one metre from the start of the test and approximately one metre long. The chalk line has been drawn over to make more visible and a string line pulled tight to lie beside it as a control. It can be seen that as a whole, the robot drives in a very straight line with only slight variances in either direction. These variances may be due to the inaccuracies of chalk on concrete or the slight changing in motor speeds and loads. Either way this test proves the robot to adequately meet the straight line driving performance objective.

5.4 Overall Performance

As a whole the robots performance is best assessed against the performance guidelines set out in the project specification. Whether or not improvements can be made, the performance guidelines are the real assessment criteria of the Mobile Manoeuvring Robots performance. The robots performance in relation to the initial guidelines is outlined below in detail.

Performance guideline one:

- *Travel in a straight line when in a clear area with no manual assistance.*

The Mobile Manoeuvring Robot encompasses odometer sensors and interrupt control to achieve straight line driving. When in a clear area free from objects, the robot maintains a constant speed and direction. Hence this performance guideline is met adequately.

Performance guideline two:

- *Stop before colliding with any obstacles that are in the path*

Once the robot is in motion, any detectable object is avoided. Issues with this aspect of performance come into affect with small objects that are missed by the relatively narrow range of the sensors. Obstacles such as chair legs and tables are adequately avoided and as it was not defined what size obstacle is necessary to be avoided; the performance in this sector is adequate. Increased sensor spread and range would however improve this performance aspect.

Performance guideline three:

- *“Decide” which way is more practical to turn and do so.*

The robot follows a strict simple decision making process in turning operations. This operation is detailed in the code section of this dissertation. If there is not room for the robot to turn in any one direction, the robot will not attempt the turn and reverse until a clear path is found. This process ensures that the third performance guideline is met satisfactorily.

Performance guideline four:

- *Continue on its path in a straight line.*

Similarly to performance guideline one, the robot will continue on a straight path after any turn or reverse decision. The Mobile Manoeuvring Robot adheres to this performance guideline.

Time permitting:

Due to time constraints and early issues with hardware design, the time permitting objectives were not completed. This however, does not hinder the specified performance of the robot.

5.5 Conclusions

While several aspects of the Mobile Manoeuvring Robot have ample room for improvement and extension, the overall success of its performance is noteworthy. The robot adheres to all of the performance guidelines set out in initial project specifications and performs satisfactorily. Many aspects need improvement such as the purpose for this technology (e.g. object retrieval) and restructuring the chassis, however this is not encompassed within the project definition or requirements. The robot performs as such that it can move throughout an indoor environment whilst avoiding obstacles. This is the overall project specification and so must be recognised as a success.

CHAPTER 6

DEBUGGING MODULE

The Mobile Manoeuvring Robot has been designed and constructed in such a way for ease of troubleshooting. An example of this is that the software has been written to output certain characters through the serial port which can be used to gain understanding of the actual performance of the robot. The following sections are written to detail the methods of debugging or troubleshooting the robot when performance is not desirable.

6.1 Microcontroller Not Working

There are two main reasons for the possible case of the microcontroller not turning on, either a poor power source or a short circuit on the voltage source lines. The orange LED on the card12 will determine whether the HC12 is working or not. If the orange LED is not lit, check the polarity and connection of the battery terminals. If these are connected properly and securely then it is likely to be either a flat battery, faulty voltage regulator, or a short circuit from any of the loose wires.

Firstly check the battery, it is easiest to either recharge or connect a different battery to eliminate this problem. The LM7805 voltage regulator requires approximately 7 to 12 volts at the input to ensure correct operation with a 5 volt output. To test the voltage regulator, a multimeter can be connected to the output leg. Typically this value will read 5.04 volts when working correctly. If this voltage is degrading slowly then the battery is flat or faulty. If there is 5 volts present on this track and is not broken away from the microcontroller then it is highly likely there is a short somewhere on the robot. Visually checking wires and contacts is the only way to find most shorts.

6.2 Motors Not Turning

If the microcontroller is on and there is no mechanical output from the robot when the push button is depressed, then it is likely there is no power supplied to the H-Bridges. This power comes from a separate supply to the microcontroller and should be 10 – 12 volts. It is connected directly to the top and bottom of the H Bridges. Also ensure all wires are connected securely between the H-Bridges and the microcontroller and also to the motors. Ensure H-Bridges are not damaged or burnt out however this is unlikely to

be the cause of problem. To ensure the push button has worked and been acknowledged by the microcontroller, connect the serial port to a computer and run a monitor program such as OC Console. Depress the button; a character 'Q' should be displayed on the screen if it has worked. A small delay follows this character send before power is sent to the H-Bridges.

6.3 Not Stopping for Objects

The main cause for the robot not stopping at objects is for the sensors to have failed. The sensors are powered by the same supply as the microcontroller and so should always be working if the microcontroller is still on. To test the sensors, view the IR LED through a digital camera. If the LED is on it will glow a purple shade and if not on it will remain black. Figure 6.1 shows cases of the LED both on and off.

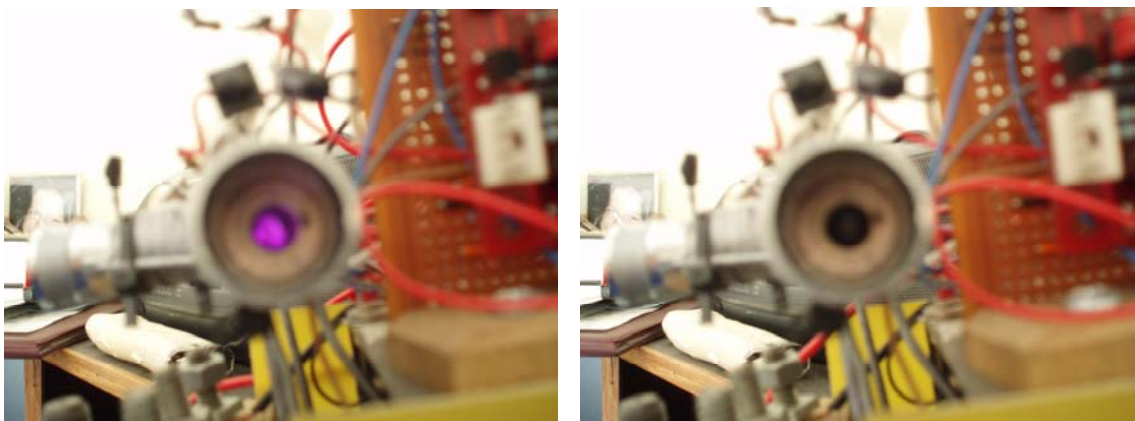


Figure 6.1: showing IR LED status. (Left: LED on // Right: LED off)

If the sensors are not working, check the power supply at the LM7805 voltage regulator on the sensor PCB. This should behave the same as the regulator for the microcontroller detailed above. If 5 volts is present and supplied to the PCB, check all wires and connections.

If the sensors are working check the operation of the receivers. Connect a multimeter to the left pin of the receiver (Blue wire). This pin should read 5 volts when there is no IR light present, and close to 0 volts when it receives reflected IR light. If this behaviour does not occur check all wiring and components.

6.4 Not Driving in a Straight Line

If the robot is moving but fails to drive in a straight line it is most likely caused by a fault in the odometer sensors. To check these, connect the serial port to a computer and run a monitoring program such as OC Console. Connect a voltage to one of the motors to make it spin. As the motor spins, either the character 'A' or 'B' should be repeated on the screen. ('A' for motor A; 'B' for motor B). This character is sent to the screen for every half turn of the motor armature. If these characters are not being sent to the screen, either the IR LED, receiver or microcontroller interrupt is failing. Check the voltage coming from the receiver. The receiver output should read 5 volts until the hole in the motor shaft is aligned, when it should drop to approximately 0 volts. If this is occurring and the character is still not being written to the screen disconnect the wire from port G and apply a voltage edge (5 volts down to 0) directly onto the port G pin. If this does not send a character to the screen the problem lies within the software. Check the INITKWG and KWGISR subroutines.

6.5 Stopping for Objects but Irregular Behaviour

If the robot is not behaving as explained in any of the above sections, the problem is most likely within the software. During construction sometimes changing hardware required different properties in the code. For instance if the source used to drive the motors was changed from 10 volts up to 12 volts, the number of odometer counts required for a 90 degree turn was altered slightly. Due to the greater inertia of the motor spinning faster, the wheel would continue to turn longer after the power was cut.

Generally, all of the driving subroutines can be operated separately. If major debugging is needed, it is possible to comment out the 'BACKUP' and 'TURNR' subroutines along with the branch instruction in 'TURNL'. This will then make the robot behave the same for every circumstance. Every time an object is detected the robot will turn left. Evidently this is impractical however it provides a good base to start adding additional code without worrying about all of the loops that are involved with the driving sequence. The 'TURNR' can then be implemented after the turning left sequence is working correctly, followed by the 'BACKUP' subroutine until the complete driving sequence is functioning correctly.

CHAPTER 7

FUTURE WORK

This chapter summarises the future work that may be carried out on the Mobile Manoeuvring Robot. Future work has been detailed in each individual section such as chassis design and software; however this chapter aims to provide the possible future work in a neat summary.

7.1 Hardware

7.1.1 Chassis Design

As described earlier, the current chassis, consisting of an aluminium box section with two worm drive gearboxes mounted inside, is barely adequate for the desired purpose of the Mobile Manoeuvring Robot. The design has very limited space with the current components as it is required to support:

- Two gearbox drives with motors
- Two H Bridge cards
- One microcontroller card
- Three obstacle sensors
- Sensor PCB
- Noise filter
- Power Supply

For future work to continue, the chassis needs to be extended or re-constructed. The layered design explained in chapter 5 (performance and data analysis) is deemed the most adequate.

7.1.2 Sensor Design

The sensors require very little extra work as they are adequate as they are. If the purpose of this robot is extended to achieve more practical tasks, the sensors may need to be adjusted to give a wider range. This may be more suitable for detecting smaller or less obvious obstacles in the path. This extension may be completed by removing the

outer of the torch so that the clocked IR light spreads further rather than being concentrated to a point. Another method would be to add more clocked light LED's to transmit a greater quantity of light, and hence objects will reflect more light to the receiver. This is not deemed an important aspect of future work however may become important if the robots performance criteria change.

7.1.3 Power Source Design

Currently, the Mobile Manoeuvring Robot has two power sources. One battery powers the microcontroller and IR sensors, and the other powers the drive motors. The main purpose of this concept is to eliminate voltage spikes from the drive motors resetting the microcontroller. This problem could be overcome however by adding filters (capacitors) to the power source or the drive motors. For the purposes of the Mobile Manoeuvring Robot this was not dealt with as the dual power source was not an issue. For future work it may be desirable to reduce the amount of battery carried on the robot to lessen the weight and bulkiness of the system.

7.2 Software Design

The structure of the software depends greatly on the desired performance of the Mobile Manoeuvring Robot. If future work only extends on the base concept there are several areas where improvement can be made.

- Ramping functions on the drive motors instead of starting them at full speed. This will reduce voltage spikes and make the travel of the Robot much smoother.
- Dead reckoning guidance. This will give the robot much more objective. If the robot 'knows' its starting position (zero odometer counts) and also a desired finishing position, it will have a more logical turning sequence, rather than simply avoiding the obstacles.

CHAPTER 8

CONCLUSIONS

As discussed in chapter 5 (Performance and Data Analysis), the performance of the Mobile Manoeuvring Robot was a success. All of the performance guidelines were met including:

- Travel in a straight line when in a clear area with no manual assistance.
- Stop before colliding with any obstacles that are in the path
- “Decide” which way is more practical to turn and do so.
- Continue on its path in a straight line.

All of these criteria are met satisfactorily however there is room for improvement in several areas as also explained in chapter 5. Before implementation into any practical situation the following factors need improvement.

- Chassis re-design to allow more components e.g. batteries, robotic arms.
- Software improvements depending on the desired task.

Although unlikely to be the entire solution to any problem, the code and hardware associated with the Mobile Manoeuvring Robot would provide a good base or inner structure into many autonomous applications. Depending on the application, it may require minor or major adjustments to the software to suit. For example an autonomous vacuum cleaner may use this software, but turn 180 degrees and offset every time an obstacle is detected, instead of simply turning 90 degrees. This would require only several extra lines of code.

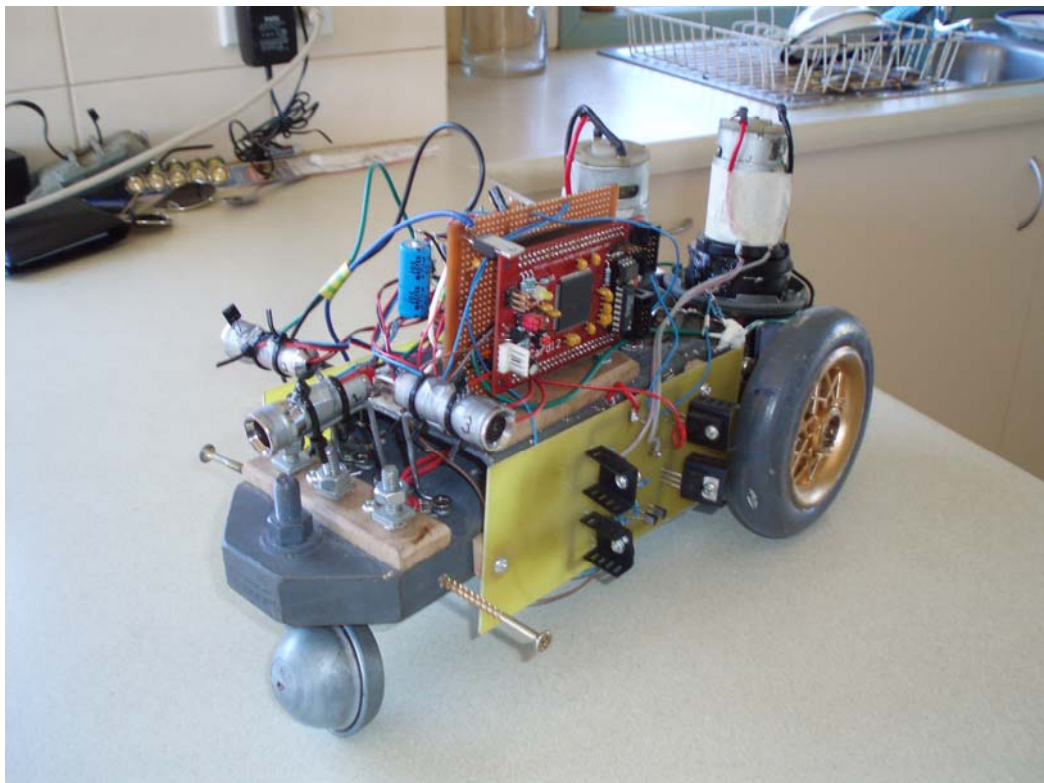


Figure 8.1: Working Prototype of Mobile Manoeuvring Robot

Figure 8.1 shows the prototype robot to date without the power supplies attached.

Overall, the Mobile Manoeuvring Robot was a success in every aspect of performance however would need substantial work to implement into a practical application.

APPENDICES

Appendix A – Project Specification

Appendix B – Software Listing

Appendix C – Software Design Procedure

Appendix D – Component Data Sheets

Appendix B – Software Listing

This is the code stored in the Motorola HC12 's memory which is executed on the event of the 'start' push button.

;ADDRESS DEFINITION

```
PORTA      equ $00  ;Port A Data
PORTB      equ $01  ;Port B Data
DDRA       equ $02  ;Port A Data Direction
DDRB       equ $03  ;Port B Data Direction
PORTE      equ $08  ;Port E Data
DDRE       equ $09  ;Port E Data Direction
PEAR       equ $0A  ;Port E Assigment
MODE       equ $0B  ;Mode
PUCR       equ $0C  ;Pull Up Control
RDRIV      equ $0D  ;Reduced Drive
INTRM      equ $10  ;RAM Position
INTRG      equ $11  ;Register Position
INITEE     equ $12  ;EEPROM Position
MISC       equ $13
RTICTL     equ $14  ;Real Time Interrupt Control
RTIFLG     equ $15  ;Real Time Interrupt Flag
COPCTL     equ $16  ;COP Control
COPRST     equ $17  ;Arm/Reset COP Timer
INTCR      equ $1E  ;Interrupt Control
HPRIO      equ $1F  ;Highest Priority Interrupt
BRKCT0     equ $20  ;
BRKCT1     equ $21  ;
BRKAH      equ $22  ;
BRKAL      equ $23  ;
BRKDH      equ $24  ;
BRKDL      equ $25  ;
PORTG      equ $28  ;Port G Data
PORTH      equ $29  ;Port H Data
DDRG       equ $2A  ;Port G Data Direction
DDRH       equ $2B  ;Port H Data Direction
KWIEG      equ $2C  ;
KWIEH      equ $2D  ;
KWIFG      equ $2E  ;
KWIFH      equ $2F  ;
SYNR       equ $38  ;
```

```

REFDV      equ $39  ;
PLLFLG     equ $3B  ;
PLLCR      equ $3C  ;
CLKSEL     equ $3D  ;
SLOW       equ $3E  ;
PWCLK      equ $40  ;
PWPOL      equ $41  ;
PWEN       equ $42  ;
PWPRES     equ $43  ;
PWSCAL0    equ $44  ;
PWSCNT0    equ $45  ;
PWSCAL1    equ $46  ;
PWSCNT1    equ $47  ;
PWCNT0     equ $48  ;
PWCNT1     equ $49  ;
PWCNT2     equ $4A  ;
PWCNT3     equ $4B  ;
PWPER0     equ $4C  ;
PWPER1     equ $4D  ;
PWPER2     equ $4E  ;
PWPER3     equ $4F  ;
PWDTY0     equ $50  ;
PWDTY1     equ $51  ;
PWDTY2     equ $52  ;
PWDTY3     equ $53  ;
PWCTL      equ $54  ;
PWTST      equ $55  ;
PORTP      equ $56  ;
DDRP       equ $57  ;
ATD0CTL0   equ $60  ;Reserved
ATD0CTL1   equ $61  ;Reserved
ATD0CTL2   equ $62  ;ATD Control 2
ATD0CTL3   equ $63  ;ATD Control 3
ATD0CTL4   equ $64  ;ATD Control 4
ATD0CTL5   equ $65  ;ATD Control 5
ATD0STAT0  equ $66  ;ATD Status
ATD0STAT1  equ $67  ;ATD Status Low Byte
ATD0TESTH  equ $68  ;ATD Test
ATD0TESTL  equ $69  ;ATD Test Low Byte
PORTAD0    equ $6F  ;Port AD Data Input

```

```

ADR00H      equ $70 ;A/D Converter Result0
ADR00L      equ $71 ;
ADR01H      equ $72 ;A/D Converter Result 1
ADR01L      equ $73 ;
ADR02H      equ $74 ;A/D Converter Result 2
ADR02L      equ $75 ;
ADR03H      equ $76 ;A/D Converter Result 3
ADR03L      equ $77 ;
ADR04H      equ $78 ;A/D Converter Result 4
ADR04L      equ $79 ;
ADR05H      equ $7A ;A/D Converter Result 5
ADR05L      equ $7B ;
ADR06H      equ $7C ;A/D Converter Result 6
ADR06L      equ $7D ;
ADR07H      equ $7E ;A/D Converter Result 7
ADR07L      equ $7F ;
TIOS        equ $80 ;Timer Input Capture/Output Compare Select
CFORC      equ $81 ;Timer Compare Force
OC7M       equ $82 ;Output Compare 7 Mask
OC7D       equ $83 ;Output Compare 7 Data
TCNT       equ $84 ;Timer Counter
TCNTL      equ $85 ;Timer Counter Low Byte
TSCR       equ $86 ;Timer System Control
TQCR       equ $87 ;Reserved
TCTL1      equ $88 ;Timer Control 1
TCTL2      equ $89 ;Timer Control 2
TCTL3      equ $8A ;Timer Control 3
TCTL4      equ $8B ;Timer Control 4
TMSK1      equ $8C ;Timer Interrupt Mask 1
TMSK2      equ $8D ;Timer Interrupt Mask 2
TFLG1      equ $8E ;Timer Interrupt Flag 1
TFLG2      equ $8F ;Timer Interrupt Flag 2
TC0        equ $90 ;TIC/TOC 0
TC0LOW     equ $91 ;TIC/TOC 0 Low Byte
TC1        equ $92 ;TIC/TOC 1
TC1LOW     equ $93 ;TIC/TOC 1 Low Byte
TC2        equ $94 ;TIC/TOC 2
TC2LOW     equ $95 ;TIC/TOC 2 Low Byte
TC3        equ $96 ;TIC/TOC 3
TC3LOW     equ $97 ;TIC/TOC 3 Low

```

```

TC4          equ $98  ;TIC/TOC 4
TC4LOW       equ $99  ;TIC/TOC 4 Low Byte
TC5          equ $9A  ;TIC/TOC 5
TC5LOW       equ $9B  ;TIC/TOC 5 Low Byte
TC6          equ $9C  ;TIC/TOC 6
TC6LOW       equ $9D  ;TIC/TOC 6 Low Byte
TC7          equ $9E  ;TIC/TOC 7
TC7LOW       equ $9F  ;TIC/TOC 7 Low Byte
PACTL        equ $A0  ;Pulse Accumulator Control
PAFLG        equ $A1  ;Pulse Accumulator Flag
PACN3        equ $A2  ;Pulse Accumulator Count
PACN2        equ $A3  ;
PACN1        equ $A4  ;
PACN0        equ $A5  ;
MCCTL        equ $A6  ;
MCFLG        equ $A7  ;
ICPACR       equ $A8  ;
DLYCT        equ $A9  ;
ICOVW        equ $AA  ;
ICSYS        equ $AB  ;
TIMTST       equ $AD  ;Timer Test
PORTT        equ $AE  ;Timer Port T Data
DDRT         equ $AF  ;Timer Port T Data Direction
PBCTL        equ $B0  ;
PBFLG        equ $B1  ;
PA3H         equ $B2  ;
PA2H         equ $B3  ;
PA1H         equ $B4  ;
PA0H         equ $B5  ;
MCCNTH       equ $B6  ;
MCCNTL       equ $B7  ;
TC0H         equ $B8  ;
TC0HLOW      equ $B9  ;
TC1H         equ $BA  ;
TC1HLOW      equ $BB  ;
TC2H         equ $BC  ;
TC2HLOW      equ $BD  ;
TC3H         equ $BE  ;
TC3HLOW      equ $BF  ;
SC0BDH       equ $C0  ;SCI 0 Baud Rate

```

```

SC0BDL      equ $C1    ;SCI 0 Baud Rate Low Byte
SC0CR1      equ $C2    ;SCI 0 Control 1
SC0CR2      equ $C3    ;SCI 0 Control 2
SC0SR1      equ $C4    ;SCI 0 Status 1
SC0SR2      equ $C5    ;SCI 0 Status 2
SC0DRH      equ $C6    ;SCI 0 Data
SC0DRL      equ $C7    ;SCI 0 Data Low Byte
SC1BDH      equ $C8    ;SCI 1 Baud Rate
SC1BDL      equ $C9    ;SCI 1 Baud Rate Low Byte
SC1CR1      equ $CA    ;SCI 1 Control 1
SC1CR2      equ $CB    ;SCI 1 Control 2
SC1SR1      equ $CC    ;SCI 1 Status 1
SC1SR2      equ $CD    ;SCI 1 Status 2
SC1DRH      equ $CE    ;SCI 1 Data
SC1DRL      equ $CF    ;SCI 1 Data Low Byte
SP0CR1      equ $D0    ;SPI 0 Control 1
SP0CR2      equ $D1    ;SPI 0 Control 2
SP0BR       equ $D2    ;SPI 0 Baud Rate
SP0SR       equ $D3    ;SPI 0 Status
SP0DR       equ $D5    ;SPI 0 Data
PORTS       equ $D6    ;Port S Data
DDRS        equ $D7    ;Port S Data Direction
PURDS       equ $D9    ;
EEMCR       equ $F0    ;EEPROM Module Configuration
EEPROT      equ $F1    ;EEPROM Block Protect
EEPROM      equ $F3    ;EEPROM Control
FEE32LCK    equ $F4    ;
FEE32MCR    equ $F5    ;
FEETST      equ $F6    ;
FEE32CTL    equ $F7    ;
FEE28LCK    equ $F8    ;
FEE28MCR    equ $F9    ;
FEETST1     equ $FA    ;
FEE28CTL    equ $FB    ;
CMCR0       equ $100   ;
CMCR1       equ $101   ;
CBTR0       equ $102   ;
CBTR1       equ $103   ;
CRFLG       equ $104   ;
CRIER       equ $105   ;

```

CTFLG	equ \$106 ;
CTCR	equ \$107 ;
CIDAC	equ \$108 ;
CRXERR	equ \$10E ;
CTXERR	equ \$10F ;
CIDAR0	equ \$110 ;
CIDAR1	equ \$111 ;
CIDAR2	equ \$112 ;
CIDAR3	equ \$113 ;
CICMR0	equ \$114 ;
CIDMR1	equ \$115 ;
CIDMR2	equ \$116 ;
CIDMR3	equ \$117 ;
CIDAR4	equ \$118 ;
CIDAR5	equ \$119 ;
CIDAR6	equ \$11A ;
CIDAR7	equ \$11B ;
CIDMR4	equ \$11C ;
CIDMR5	equ \$11D ;
CIDMR6	equ \$11E ;
CIDMR7	equ \$11F ;
PCTLCAN	equ \$13D ;
PORTCAN	equ \$13E ;
DDRCAN	equ \$13F ;
RxFG	equ \$140 ;
Tx0	equ \$150 ;
Tx1	equ \$160 ;
Tx2	equ \$170 ;
ATD1CTL0	equ \$1E0 ;
ATD1CTL1	equ \$1E1 ;
ATD1CTL2	equ \$1E2 ;
ATD1CTL3	equ \$1E3 ;
ATD1CTL4	equ \$1E4 ;
ATD1CTL5	equ \$1E5 ;
ATD1STAT0	equ \$1E6 ;
ATD1STAT1	equ \$1E7 ;
ATD1TESTH	equ \$1E8 ;
ATD1TESTL	equ \$1E9 ;
PORTAD1	equ \$1EF ;
ADR10H	equ \$1F0 ;


```

ADR10L      equ $1F1 ;
ADR11H      equ $1F2 ;
ADR11L      equ $1F3 ;
ADR12H      equ $1F4 ;
ADR12L      equ $1F5 ;
ADR13H      equ $1F6 ;
ADR13L      equ $1F7 ;
ADR14H      equ $1F8 ;
ADR14L      equ $1F9 ;
ADR15H      equ $1FA ;
ADR15L      equ $1FB ;
ADR16H      equ $1FC ;
ADR16L      equ $1FD ;
ADR17H      equ $1FE ;
ADR17L      equ $1FF ;

```

```

;*****

```

```

;MOTOR DIRECTION CONTROL

```

```

;pwm channel 0 = motor B CLOCKWISE
;pwn channel 1 = motor B ANTI CLOCKWISE
;pwm channel 2 = motor A ANTI CLOCKWISE
;pwm channel 3 = motor A CLOCKWISE

```

```

;*****VARIABLE DEFINITION*****

```

```

                ORG $0200                START DEFINITION AT ADDRESS $0200
ODMA            DC.B    0
ODMB            DC.B    0                ;DEFINE VARIABLES
COUNT         DC.B    0

```

```

;*****MAIN PROGRAM..CONTROL LOOP*****

```

```

;NOTE: NO ACTION WILL OCCUR UNTIL THE IRQ INTERRUPT IS ACTIVATED UNLESS
FRONT SENSOR IS TRIGGERED

```

```

                ORG    $0400                ;START PROGRAM STORE AT $0400

MAIN            LDS    #$0700                ;SET STACK POINTER
                CLR    COPCTL                ;DISABLE COMPUTER OPERATING
                                                NORMALLY WATCHDOG

```

```

        JSR      INITSC0      ;INITIALISE SERIAL PORT
        JSR      INITKWG      ;INITIALISE PORT G
        JSR      INITPWM      ;SET UP PULSE WIDTH MODULATION
                                ON PORT P
        JSR      INITIRQ      ;INITIALISE IRQ INTERRUPT
        JSR      INITPA

WAIT    BRCLR    PORTA,$01,SKP ;CHECK FRONT SENSOR AND IF ON
                                BRANCH TO TURNL

        JSR      TURNL

SKP     LDAA     COUNT
        CMPA     #$5A          ;HAS MOTOR A COUNTED xx TIMES YET
                                (xx/2 REVOLUTIONS)
        BMI     WAIT          ;IF NOT ,WAIT AND CHECK AGAIN
        JSR     COMP          ;IF SO JUMP TO COMP (compare function)
        BRA     WAIT          ;AFTER COMP GO BACK AND WAIT FOR
                                ANOTHER xx/2 REVOLUTIONS

```

,*****DRIVE FORWARDS*****

```

DRFWD   MOVB     #$A0, PWDTY0   ;DUTY CYCLE FOR CHANNEL 0|
        MOVB     #$A0, PWDTY2   ;DUTY CYCLE FOR CHANNEL 2| **DRIVE
                                FORWARDS
        RTS          ;RETURN FROM SUBROUTINE

```

,*****TURNING CODES*****

```

TURNL   LDX      #$0A00
LP      BRCLR    PORTA,$01,FALSE ;|
        DEX          ;LOOP TO CHECK SENSOR $0A00 TIMES
                                TO ENSURE NOT JUST NOISE
        BNE     LP          ;|
        BRCLR   PORTA,$04,SKR  ;IF LEFT SENSOR(3) BLOCKED, TURN
        JSR     TURNR        ;RIGHT INSTEAD
SKR     LDAA     #'L'
        JSR     TXBYTE       ;TRANSMIT L TO SCREEN (DEBUG)
        MOVB    #$00, PWDTY0   ;|
        MOVB    #$00, PWDTY2   ;|STOP MOTORS
        JSR     DELAY        ;WAIT FOR DELAY PERIOD
        MOVB    #$00, ODMA     ;|

```

```

MOV B #0,ODMB ;CLEAR ODOM COUNTERS
MOV B #80,PWDTY1 ;TURN MOTOR B BACKWARDS
CONTL LDAA #00
LDAB ODMB
CPD #90 ;SET NUMBER RELATIVE TO A 90
DEGREE TURN
BLT CONTL ;WAIT FOR TURN TO COMPLETE
MOV B #0,PWDTY1
JSR DELAY
JSR DRFWD ;CONTINUE FORWARDS
FALSE RTS ;RETURN FROM SUBROUTINE
;-----
TURNR ;BRCLR PORTA,$02,SKB ;IF RIGHT SENSOR(2) BLOCKED,
BACKUP INSTEAD
JSR BACKUP
SKB LDX #0A00 ;
LP2 BRCLR PORTA,$04,FALSE2 ;
DEX ;LOOP TO CHECK SENSOR(3) $0200
TIMES TO ENSURE NOT JUST NOISE
BNE LP2 ;
LDAA #'R' ;SEND 'R' THROUGH SERIAL PORT
JSR TXBYTE ;FOR DEBUG PURPOSES
MOV B #0,PWDTY0 ;
MOV B #0,PWDTY2 ;STOP MOTORS (DRIVING FORWARDS)
MOV B #0,ODMA ;
MOV B #0,ODMB ;CLEAR ODOM COUNTERS
JSR DELAY ;WAIT FOR DELAY PERIOD
MOV B #80,PWDTY3 ;TURN MOTOR A BACKWARDS
CONTR LDAA #00
LDAB ODMA
CPD #90 ;SET NUMBER RELATIVE TO A 90
DEGREE TURN
BLT CONTR ;WAIT FOR TURN TO COMPLETE
MOV B #0,PWDTY3
JSR DELAY ;WAIT DELAY PERIOD
JSR DRFWD ;CONTINUE FORWARDS
FALSE2 RTS ;EXIT AND RETURN FROM SUBROUTINE
;-----
BACKUP ;LDAA #'K' ;SEND K THROUGH SERIAL PORT

```

```

;JSR      TXBYTE      ;(DEBUG)
;MOVB    #$00,PWDTY0  ;|
;MOVB    #$00,PWDTY2  ;|STOP MOTORS (DRIVING FORWARDS)
;JSR     DELAY        ;WAIT DELAY PERIOD
;MOVB    #$A0,PWDTY1  ;|
;MOVB    #$A0,PWDTY3  ;|REVERSE STRAIGHT
WFCLR    BRCLR    PORTA,$02,OUTL  ;IF LEFT NOT SET BRA TO OUTL
;BRCLR   PORTA,$04,OUTR  ;IF RIGHT NOT SET BRA TO OUTR
BRA      WFCLR        ;IF L&R SET, BRA WFCLR. WAIT FOR L
                        OR R TO CLEAR
OUTL     JSR      TURNL          ;TURN L WHEN L SENSOR IS CLEAR
OUTR     JSR      TURNR          ;TURN R WHEN R SENSOR IS CLEAR
;JSR     DELAY
RTS
;RETURN TO MAIN LOOP

```

```

;*****DELAY*****
;CODE TO ADD A DELAY TO EASE THE SWITCHING OF MOTORS
;GIVES APPROX 2 SEC DELAY. CAN SHORTEN TIME BY INCREASING INITIAL Y VALUE

```

```

DELAY    LDY      #$FFDC        ;INITIAL VALUE FOR Y
DELA     LDX      #$0000        ;INITIAL CONDITION OF X
DEL      INX
;BNE     DEL      ;IF Z NOT = 0 BRANCH TO DELAY
INX
;BNE     DELA
RTS

```

```

;*****COMPARE SPEEDS*****
;CODE TO COMPARE MOTOR A AND B SPEEDS AND CONTROL H BRIDGE

```

```

COMP     JSR      NEWLINE        ;START NEW LINE ON SERIAL PORT
;LDAA   ODMA      ;LOAD MOTOR A COUNT INTO ACC A
CMPA    ODMB      ;COMPARE MOTOR A WITH MOTOR B
BPL     A2FAST    ;IF MOTOR A FASTER THAN MOTOR B
BMI     B2FAST    ;IF MOTOR A SLOWER THAN MOTOR B
;-----
A2FAST   JSR      NEWLINE        ;CASE MOTOR A IS FASTER THAN B
;LDAA   #'a'
;JSR    TXBYTE    ;TX BYTE THROUGH SERIAL (DEBUG)

```

```

        LDAA    PWDTY2           ;LDAA SPEED OF MOTOR A
        CMPA    #$65            ;IF MOTOR A IS SLOWER THAN $C5
        BMI     FASTB           ;SPEED UP MOTOR B
        SUBA    #$01            ;IF MOTOR A IS FASTER THAN $C5
        STAA    PWDTY2         ;SLOW MOTOR A SLIGHTLY
        BRA     DONE            ;EXIT
FASTB   LDAA    PWDTY0           ;|
        ADDA    #$01            ;|SPEED B UP ONE
        STAA    PWDTY0         ;|
        BRA     DONE

;-----
B2FAST JSR     NEWLINE          ;CASE MOTOR B IS FASTER THAN A
        LDAA    #'b'
        JSR     TXBYTE          ;TX BYTE THROUGH SERIAL (DEBUG)
        LDAA    PWDTY0         ;LDAA WITH SPEED OF MOTOR B
        CMPA    #$65            ;|
        BMI     FASTA          ;| IF MOTOR B GOING TOO SLOW, SPD
                                UP A
        SUBA    #$01            ;|
        STAA    PWDTY0         ;|SLOW MOTOR B SLIGHTLY
        BRA     DONE
FASTA  LDAA    PWDTY2           ;|
        ADDA    #$01            ;| SPEED A UP BY ONE
        STAA    PWDTY2         ;|

DONE   MOVB    #$00, COUNT     ;|
        MOVB    #$00, ODMA     ;| CLEAR COUNTERS
        MOVB    #$00, ODMB     ;|

        RTS                    ;RETURN AND WAIT FOR NEXT
                                COMPARE

;*****KEY WAKE UP PORT G *****
;ADDS 1 TO COUNTER FOR CORRESPONDING MOTOR
;****SENSOR G 0*****
KWGISR BRCLR   KWIFG,$01,NOTG0 ;CHECK BIT 0
        LDAA    ODMB           ;IF MOTOR B, LDA WITH ODOM COUNT
                                B
        ADDA    #$01           ;ADD 1 TO ODMB COUNT
        STAA    ODMB          ;STORE BACK TO ODMB

```

```

        LDAA    #'B'
        JSR     TXBYTE           ;WRITE TO SERIAL PORT (DEBUG)

;****SENSOR G 1*****
NOTG0    BRCLR  KWIFG,$02,NOTG1           ;CHECK BIT 1
        LDAA    ODMA           ;IF MOTOR A, LDA WITH ODOM COUNT
        ADDA    #$01           ;ADD 1 TO COUNT
        STAA    ODMA           ;STORE BACK TO ODMA
        LDAA    COUNT
        ADDA    #$01
        STAA    COUNT
        LDAA    #'A'
        JSR     TXBYTE           ;WRITE TO SERIAL PORT (DEBUG)

;*****FALSE INTERRUPT*****
NOTG1    MOVB   #%00000011,KWIFG ;CLEAR WAKE UP FLAGS

        RTI                    ;RETURN FROM INTERRUPT

;*****IRQ SERVICE ROUTINE*****
;THIS INTERRUPT STARTS THE ROBOT WHEN BUTTON IS PRESSED
IRQISR   JSR     DELAY           ;PROVIDES SOME SWITCH BOUNCE
        JSR     DRFWD           ;BEGIN DRIVING
        RTI                    ;RETURN FROM INTERRUPT

;*****
;SETUP PWM Channels

INITPWM  MOVB   #%00111111,PWCLK ;PRESCALER OF /128
        MOVB   #%00001111,PWPOL ;PPOL
        MOVB   #%00001111,PWEN  ;ENABLE ALL OUTPUTS
        MOVB   #%00000010,PWCTL ;PULL UP DEVICE ENABLED

SETPWM   MOVB   #$FF,PWPER0     ;|
        MOVB   #$FF,PWPER1     ;|PERIOD FOR PWM
        MOVB   #$FF,PWPER2     ;|
        MOVB   #$FF,PWPER3     ;|

        RTS                    ;RETURN TO MAIN

```

*****INITIALISE PORT A*****

```
INITPA      MOVB      #%11111000,DDRA    ;SET PORT A BITS FOR OUT AND IN 0 =
                                                    INPUT
            MOVB      #%00000000,PORTA  ;ENSURE PORT A INPUTS ARE ZEROES
                                                    WAITING FOR ACITVE HIGH

            RTS
```

*****INITIALISE PORT G FOR KEY WAKE UP*****

```
INITKWG     MOVB      #%11111100,DDRG    ;BITS 0-4 AS INPUT
            MOVB      #%00000011,KWIFG   ;CLEAR WAKE UP FLAG 0-4
            MOVB      #%00000011,KWIEG   ;ENABLE KEY WAKE UP INTERRUPTS
            MOVB      #%00000000,PUCR    ;SET PORT G, H IRQ AND XIRQ FOR PULL
                                                    DOWN (DISAHLE ALL PULL UPS(ALL
                                                    0'S))
            LDAA      #$06                ;JMP COMMAND OPCODE
            STAA      $07B8              ;RTI PSEUDO VECTOR
            LDD       #KWGISR            ;ADDRESS TO JUMP TO, KWG ISR
            STD       $07B9
            CLI                                     ;CLEAR INTERUPT MASK
            RTS
```

*****SERIAL PORT*****

```
; TO USE THE SERIAL PORT, SET BAUD RATE TO 19200
; BAUD. THE VALUE HERE IS A 16 BIT DIVISOR.
```

```
INITSCO     LDD       #26                ; VALUE FROM BAUD RATE
                                                    GENERATION TABLE
            STD       SC0BDH
            LDAA      #$0C                ; ENABLE TRANSCEIVER
            STAA      SC0CR2
            LDAA      SC0DRL              ;CLEAR FLAGS
            RTS
```

*****SERIAL PORT WRITING COMMANDS*****

```
; SEND A CARRIAGE RETURN AND LINE FEED TO SCREEN
```

```
NEWLINE     LDAA      #$0D                ;LOAD CARRIAGE RETURN FOR ASCII
            JSR       TXBYTE              ;TRANSMIT TO SCREEN
            LDAA      #$0A                ;LOAD LINE FEED ASCII
            JSR       TXBYTE              ;TRANSMIT TO SCREEN
            RTS                          ;RETURN
```

;-----

; TRANSMIT A CHARACTER STORED IN ACCUMULATER A

```
TXBYTE      BRCLR    SC0SR1, #$80    ;WAIT FOR CLEAR FLAG
             STAA    SC0DRL          ;SEND DATA THROUGH SERIAL PORT
             RTS
```

.******.*

Appendix C – Software design procedure

MOBILE MANOEUVRING ROBOT SOFTWARE DESIGN

Objectives: To control the mechanical hardware of a two drive-wheel robot in order to avoid collision with obstacles. The software must be able to control the unit to travel in a straight line between objects and is based on assembly level language of the Motorola 68HC12 microcontroller.

Obstacle avoiding

While driving do the following:

- Read the odometer sensors and increment odometer counters
 - Check odometer counters and adjust speed for straight driving
- Watch obstacle sensors in a tight loop act accordingly when sensors triggered.
 - If left is clear, turn left otherwise turn right
 - If left and right blocked reverse
- Continue on a straight path
- Output data through serial port for debugging purposes
- Repeat process continuously

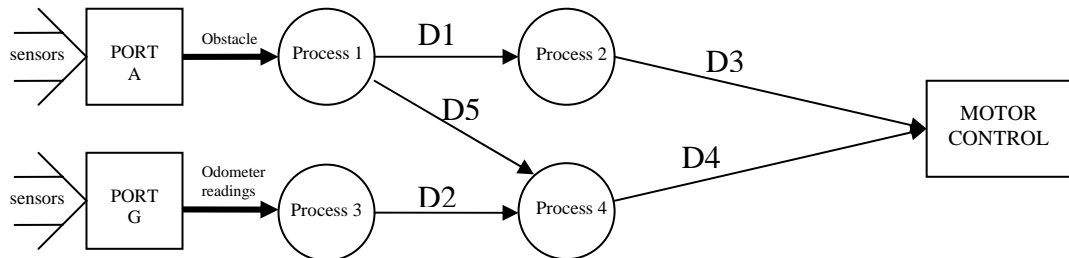
User Information:

The code has five sensors that input to the software. Three of these are obstacle avoidance sensors read in as data through port A and the other two are concerned with odometer readings read in as interrupts through port G. The program starts at \$0400 with data starting at \$0200. The stack pointer must be set in a clear location such as \$0700 and the program counter to \$0400 to start the procedure.

When run, the program will output through the serial port the status of the odometer readings and also controls the speed and direction of both motors to achieve the desired performance. The program loops continuously to give real time control the mechanical components

System Model

The following data flow diagram shows the basic flow of values through the system.



D1 = Signal of which direction to turn
D2 = odometer counters
D3 = motor information for turning
D4 = motor information for drive straight
D5 = jump to process 4 when required

Process 1:

This is the main section of the program. It runs a tight loop checking for data from the sensors on port A and also checking the odometer readings to initiate a compare function to maintain straight line driving.

Process 2:

Process 2 is a large subroutine called every time a turn is needed. It accesses data from port A. It checks sensors and turns relative to the status of them. After the motors have been adjusted to gain desired performance the code returns to the main loop, process 1.

Process 3:

This is an interrupt service routine that determines which odometer sensor has triggered and increments the respective counter. The counter is a byte stored in memory and is cleared regularly to avoid overflow issues.

Process 4:

This process is a subroutine used to maintain straight line driving. It is called from the main program loop and works by comparing the odometer counters and adjusting motor speeds to maintain a constant even speed between the motors. This results in even wheel speed hence straight driving.

Software Specification

Execution structures, Pseudocode

MAIN set the stack pointer in a clear area
 Clear computer watchdog
 Initialise serial port
 Initialise Kew Wake up on port G
 Initialise IRQ Interrupt
 Initialise port A

Start loop check sensor for objects
 Branch to turn if necessary
 Check odometer counters and branch to compare if necessary
 Repeat loop.

DRFWD start both wheels driving forwards
 Return to main

TURNL ensure signal not just noise
 if left blocked, jump to turn right
 transmit an L through the serial port
 Stop the motors
 Turn motor B backwards until robot turned 90 degrees
 Stop turning
 Continue in straight path (jump to DRFWD)
 Return to main

TURNR ensure signal not just noise
 if right blocked, jump to backup
 transmit an R through the serial port
 Stop the motors
 Turn motor A backwards until robot turned 90 degrees
 Stop turning
 Continue in straight path (jump to DRFWD)
 Return to main

BACKUP	transmit 'K' through serial port Stop the motors Delay Turn both motors backwards
Start loop	if left sensor clear, turn left then go forwards again If right sensor clear, turn right then go forwards again Repeat loop Return to main
DELAY	load a large number Decrement it slowly Return to main
COMP	new line through serial port Load accumulator with odometer count A Compare with odometer count B If B bigger than A branch to B2FAST If A bigger than B branch to A2FAST
A2FAST	write 'a' through serial port Compare motor A speed with a constant If faster than constant slow it slightly If slower than constant branch to FASTB
FASTB	Increase motor B slightly Return to main
B2FAST	write 'b' through serial port Compare motor B speed with a constant If faster than constant slow it slightly If slower than constant branch to FASTA
FASTA	Increase motor A slightly Return to main

Software Module Description (Major modules)

Module Name: **MAIN**
Purpose: To run a tight loop continuously monitoring all sensors and odometer readings.
Called by: N/A
Calls: INITSCO, INITKWG, INITPWM, INITIRQ, INITPA, TURNL, COMP
Validation: PORT A OBSTACLE SENSORS
Registers used: A and memory location COUNT

Module Name: **DRFWD**
Purpose: Initialise driving forward
Called by: IRQISR, TURNL, TURNR
Calls: N/A
Validation: N/A
Registers used: N/A

Module Name: **TURNL and TURNR**
Purpose: To turn the robot in the desired direction by 90 degrees
Called by: MAIN
Calls: TXBYTE, DELAY, DRFWD
Validation: Outputs to PWDTY registers
Registers used: A, B, and X

Module Name: **BACKUP**
Purpose: Reverse the robot in a straight line
Called by: TURNR
Calls: DELAY, TURNR, TURNL
Validation: outputs to PWDTY registers
Registers used: A, B, X

Module Name: **DELAY**
Purpose: To create a pause in the program.
Called by: IRQISR, TURNL, TURNR, BACKUP
Calls: N/A
Validation: N/A
Registers used: X and Y

System Implementation

Data dictionary

Constants and variables used in this code are listed below.

- | | |
|---------|-----------------------------------|
| - ODMA | odometer counter of motor A |
| - ODMB | odometer counter of motor B |
| - COUNT | variable used in compare function |

Conversion procedures

The assembly language code has been written in a text editor such as notepad or MiniIDE and built into a listing file using MiniIDE. Data has been transferred to the microcontroller using the program OC Console.

Load Map

- Data is to be loaded at \$0200.
- Program starts at \$0400
- Stack pointer set to clear area at \$0700

Conclusions

The system performed adequately to the required specifications. The software responded as desired for all sensor inputs including odometer readings. As a result of the success, the robot is able to drive at a constant speed in a straight line between obstacles.

The robots performance could be improved by adding additional code in many of the functions. One such instance is the adding of ramps in the PWDTY registers to 'ease' into the desired speed. If dead reckoning guidance is implemented, this code will require extensive modification.


Appendix D – Component Data Sheets

The following data sheets are the main components used in the construction of the Mobile Manoeuvring Robot. Not all data sheets will be supplied such as the transistor sheets which are not specific to the purpose of the Mobile Manoeuvring Robot.

The Microcontroller, Motorola MC68HC12 data sheet will not be included as it is far too extensive for the purpose of listing the data sheets. The MC68HC12 data sheet can be downloaded from various sources on the internet.


Z 3235 IR LED

The Z3235 IR LED is a Dick Smith Electronics part available at Dick Smith Electronic stores. This component is used in all of the IR sensors, both clocked and constant light sources.



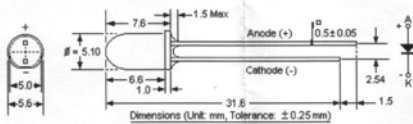
A.B.N. 34 000 908 716

Cat No. Z 3235
Infrared Light Emitting Diode



9 315999 030497

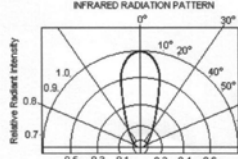
General Description:
The Z-3235 is a high intensity Gallium Aluminium Arsenide (GaAlAs) Infrared Light Emitting p-n junction semiconductor diode mounted in a special dark blue plastic filter package that reduces interference from visible light. At 940nm, it is spectrally matched to Z 1956, Z 1953, Z 1954 and Z 1955 types infrared receiving diodes. Applications include photo detectors, smoke detectors, night vision devices, infrared illuminators for CCD cameras and Camcorders, photoelectric "eyes", automatic or remote control systems, and alarm beam sensors.



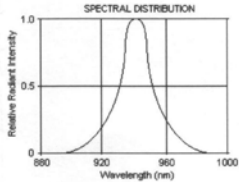
ELECTRO-OPTICAL CHARACTERISTICS				Absolute	Unit	Ta = 25°C
Parameter	Symbol	Min	Typ	Maximum		Test Conditions
Radiant Power	IE	14	-	28	mW/cm ²	IF = 20mA
Peak Emission Wavelength	λP	-	-	940	nm	IF = 20mA
Spectral Line Half Width	Δλ	-	-	50	nm	IF = 20mA
Forward Voltage	VF	-	1.2	2.0	V	IF = 20mA
Reverse Current	IR	-	-	100	μA	VR = 5V
Rise Time	TR	-	-	200	nS	PW = 10μs, S, DC = 10%, IF = 20mA
Fall Time	TF	-	-	500	nS	PW = 10μs, S, DC = 10%, IF = 20mA
Power Dissipation	PD	-	-	150	mW	VR = 5V
Continuous Forward Current	IFC	-	-	50	mA	-
Peak Forward Current	IFP	-	-	1	A	PW = 10μs, S, DC = 10%, IF = 20mA
Reverse Voltage	VR	-	-	5	V	-
Viewing Angle	2θ 1/2	-	20	-	°	-
Operating Temperature Range	Topr	-45	+25	+100	°C	-
Storage Temperature Range	Tstp	-45	+25	+100	°C	-
Lead Soldering Temperature	Tsolt	-	-	+250	°C	15 Secs 1.6mm from the base of the case

PW = Pulse Width DC = Duty Cycle

INFRARED RADIATION PATTERN



SPECTRAL DISTRIBUTION




Notes:
1. All specifications are subject to change without notice.
2. All drawings remain the copyright '2000 of Dick Smith Electronics Pty Ltd.

ZA8966

Z 1956 Infrared Receiving Diode

The Z 1955 IR Receiving Diode is a Dick Smith Electronics part available at Dick Smith Electronic stores. This component is used in the odometer counting sensors to receive the IR light.

Cat No. Z 1956
Infrared Receiving Diode



9 315999 029835

Features:

- * High output
- * Fast response
- * Wide viewing angle
- * High sensitivity
- * Reliable
- * High cut off frequency
- * Low cost
- * Small package

Applications include:

- * Remote controls
- * Light meters
- * Photo detectors
- * Smoke detectors
- * Cameras
- * Exposure meters
- * Automatic control systems
- * Robotics

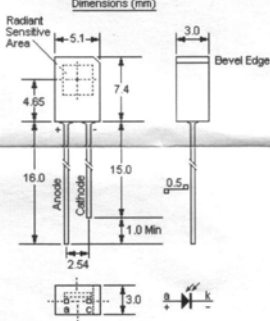
General Description:
The Z 1956 is an Infrared Receiving Diode packaged in a special dark plastic filter that is designed to reduce the effects of visible light and allows the passage and detection of infrared. The virtually black coloured device is spectrally matched to the Z 3235 Infrared Transmitting Diode.

ABSOLUTE MAXIMUM RATINGS Ta = 25°C

Parameter	Maximum Rating	Unit
Power Dissipation	150	mW
Reverse Breakdown Voltage	30	V
Operating Temperature Range	-45°C to +100°C	
Storage Temperature Range	-45°C to +100°C	
Lead Soldering Temperature *	250°C for 5 Seconds	

* 1.5mm from base of component body
VR = Reverse Voltage, IR = Reverse Current
Ee = Irradiance, IS = Photocurrent

Dimensions (mm)



ELECTRICAL-OPTICAL CHARACTERISTICS Ta = 25°C

Parameter	Symbol	Min	Typ	Max	Unit	Conditions
Reverse Breakdown Voltage	VBR	30	-	-	V	IR = 100uA, Ee = 0 mW/cm ²
Reverse Dark Current	ID(R)	-	-	30	nA	VR = 10V, Ee = 0 mW/cm ²
Reverse Light Current	IL	1.7	2.0	-	uA	VR = 5V, Ee = 5 mW/cm ²
Open Circuit Voltage	Voc	-	400	-	mV	λ = 940nm, Ee = 0.5 mW/cm ²
Short Circuit Voltage	Isc	-	70	-	uA	VR = 5V, Ee = 0.05 mW/cm ²
Rise Time	Tr	-	50	-	nS	VR = 10V, RL = 1kΩ
Fall Time	Tf	-	50	-	nS	VR = 10V, RL = 1kΩ
Capacitance	CT	-	25	-	pF	R = 3V, Vf = 1MHz, Ee = 0 mW/cm ²
Peak Wavelength	λP	-	940	-	nm	-
Viewing Angle	2θ 1/2	-	140	-	° (degrees)	-

Notes:
1. All specifications are subject to change without notice.

7A8063A

Also included in this appendix are the important pages on Interrupt vectors used for the CARD12.

6. Monitor Program TwinPEEKs

Software Version 1.6

Serial Communication

The monitor program TwinPEEKs communicates using the first RS232 interface ("SER0") at **19200 Baud**. The settings are: 8N1, no hardware or software handshake, no protocol.

Autostart Function

After Reset the TwinPEEKs Monitor checks, whether Pins PH6 and PH7 are connected or not. If there is a connection the Monitor jumps to address \$8000 (*in the case of a 912DG128: \$4000*). In this way it is possible to automatically start a user program without changing the Reset Vector, which resides in the protected Flash Boot Block area.

Redirected Interrupt Vectors

The interrupt vectors reside at the end of the 64KB memory map, which falls in the area of the protected Flash Boot Block. In order to process interrupts of an application program, the interrupt vectors are redirected to a RAM area by the Monitor. The method is similar to the behaviour of the HC11 in Special Bootstrap Mode.

The user's application program installs an interrupt vector by placing a jump instruction at the RAM pseudo vector location.

Example: Follow these steps to use the SPI interrupt:

```
ldaa #$06          ; JMP Opcode
staa $07C7         ; SPI Pseudo Vector
ldd  #isrFunc      ; Jump Address
std  $07C8         ; SPI Pseudo Vector + 1
```

The following listing shows, which interrupt is redirected to which pseudo address in RAM by the monitor program:

```

FFC2 :                                org $ffc2                                ; 912D60 *and* 912DG128
FFC2 : 07A6                            dc.w  RAMTOP-90                            ; CGM LLH
FFC4 : 07A9                            dc.w  RAMTOP-87                            ; MSCAN Tx
FFC6 : 07AC                            dc.w  RAMTOP-84                            ; MSCAN Rx
FFC8 : 07AF                            dc.w  RAMTOP-81                            ; MSCAN Err
FFCA : 07B2                            dc.w  RAMTOP-78                            ; Pulse Accu B Overflow
FFCC : 07B5                            dc.w  RAMTOP-75                            ; MDCU
FFCE : 07B8                            dc.w  RAMTOP-72                            ; KWUG, KWUH
FFD0 : 07BB                            dc.w  RAMTOP-69                            ; MSCAN Wake Up
FFD2 : 07BE                            dc.w  RAMTOP-66                            ; ATD
FFD4 : 07C1                            dc.w  RAMTOP-63                            ; SCI1
FFD6 : 07C4                            dc.w  RAMTOP-60                            ; SCI0
FFD8 : 07C7                            dc.w  RAMTOP-57                            ; SPI
FFDA : 07CA                            dc.w  RAMTOP-54                            ; Pulse Accu Input Edge
FFDC : 07CD                            dc.w  RAMTOP-51                            ; Pulse Accu Overflow
FFDE : 07D0                            dc.w  RAMTOP-48                            ; Timer Overflow
FFE0 : 07D3                            dc.w  RAMTOP-45                            ; TC7
FFE2 : 07D6                            dc.w  RAMTOP-42                            ; TC6
FFE4 : 07D9                            dc.w  RAMTOP-39                            ; TC5
FFE6 : 07DC                            dc.w  RAMTOP-36                            ; TC4
FFE8 : 07DF                            dc.w  RAMTOP-33                            ; TC3
FFEA : 07E2                            dc.w  RAMTOP-30                            ; TC2
FEEC : 07E5                            dc.w  RAMTOP-27                            ; TC1
FEEE : 07E8                            dc.w  RAMTOP-24                            ; TC0
FFF0 : 07EB                            dc.w  RAMTOP-21                            ; RTI
FFF2 : 07EE                            dc.w  RAMTOP-18                            ; IRQ
FFF4 : 07F1                            dc.w  RAMTOP-15                            ; XIRQ
FFF6 : 07F4                            dc.w  RAMTOP-12                            ; SWI
FFF8 : 07F7                            dc.w  RAMTOP-9                             ; Illegal Opcode
FFFA : 07FA                            dc.w  RAMTOP-6                             ; COP Fail
FFFC : 07FD                            dc.w  RAMTOP-3                             ; Clock Monitor Fail
FFFE : F000                            dc.w  main                                 ; Reset

```

This table is valid for both HC912D60 and HC912DG128, despite the fact that the highest used RAM address is \$3FFF with the DG128 compared to \$07FF with the D60. In addition, the DG128 has five more interrupt vectors:

```

FFB8 :                                org $ffb8                                ; 912DG128 *only!*
FFB8 : 3F97                            dc.w  RAMTOP-105                           ; MSCAN1 Tx
FFBA : 3F9A                            dc.w  RAMTOP-102                           ; MSCAN1 Rx
FFBC : 3F9D                            dc.w  RAMTOP-99                            ; MSCAN1 Err
FFBE : 3FA0                            dc.w  RAMTOP-96                            ; MSCAN1 Wake Up
FFC0 : 3FA3                            dc.w  RAMTOP-93                            ; IIC

```

REFERENCES

Polymicro 2004, Application Areas and Markets for polymer Micro-Optics, viewed 9 May 2006, POLYMICRO, <http://www.polymicro-cc.com/site/pdf/POLYMICRO-markets.pdf>

Media Limited SPG 2006, 'Explosive Ordnance Disposal and Mine Clearance Gallery', viewed 9 May 2006, <<http://www.army-technology.com>>

Electrolux, 'Welcome to the Electrolux Trilobite', viewed 9 May 2006, <<http://www.electrolux.com.au/node142.asp>>

2002, 'Appliances of the Future', Ricoh Journal, viewed 9 May 2006, http://www.jnd.org/dn.mss/appliances_of_the_fu.html

Christensen H 2002, 'Intelligent Home Appliances', viewed 9 May 2006, <<http://www.nada.kth.se/~hic/hic-papers/isrr-01.pdf>>

2001, 'Robo-Rats Electronics', viewed 10 May 2006, <http://groups.csail.mit.edu/drl/courses/cs54-2001s/interface.html>

'Project', Viewed 10 May 2006, <http://www.technology.niagarac.on.ca/students/d/mdelange/index.html>

Tunnel D 2004, 'SmartAvoidT - A Portable, Scalable Object Avoidance Solution for all Day/Night/Weather/Smoke Environments', viewed 10 May 2006, http://www.navysbir.com/04_3/49.htm

Ward K, 'Learning Mobile Robot Behaviours by Discovering Associations between Input Vectors and Trajectory Velocities,' viewed 10 May 2006, <http://www.uow.edu.au/~koren/Papers/AI97.pdf>

Lee F Lin L, 'Don't Worry, Be Happy,' viewed 10 May 2006,
<http://robots6270.mit.edu/contests/2001/robots/10/www/>

Knudsen J 2000, 'The Straight and Narrow,' Viewed 10 May 2006,
<http://www.oreillynet.com/pub/a/network/2000/05/22/LegoMindstorms.html>

Bitsoi H et al 2001, 'F.U.B.A.R', viewed 11 July 2006,
www.ee.nmt.edu/~wedeward/EE382/SP01/group3.pdf

McCoy T et al 2004, 'Infrared Communications Link with Voice and Data' viewed 19
July 2006, www.home.people.net.au/~tmccoy

Leppard D, 'Using B32's Interrupts IRQ and XIRQ' viewed 5 August,
<http://www.seattlerobotics.org/encoder/200008/dougl.html>