University of Southern Queensland
Faculty of Engineering and Surveying

Automation of a Load-Haul-Dump (LHD) Unit

A dissertation submitted by

Toni Leigh Partridge

In fulfilment of the requirements of

Courses ENG4111 and 4112 Research Project

Towards the degree of

Bachelor of Engineering (Mechatronics)

Submitted: November, 2006

Abstract

The load-haul-dump (LHD) unit is used extensively in underground mining to perform a variety of tasks within this field. The underground mining environment is extremely dangerous where some LHD accidents have caused major injury and occasional deaths to the operators onboard these vehicles as well as other mining crew, this is why a control system implemented onto these vehicles would be ideal.

This project investigates and finally implements of a tele-operated and automated control system onto an existing scale model of a LHD unit. Since the tele-operated system uses a camera, it was decided that the automated system would also use a simple type of machine vision in its guidance.

University of Southern Queensland

Faculty of Engineering and Surveying

ENG4111 Research Project Part 1 &
ENG4112 Research Project Part 2

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 student's 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.

Professor R Smith
Dean
Faculty of Engineering and Surveying

Certification

I certify that the ideas, designs and experimental work, results, analyses and conclusions set out in this dissertation are entirely my own effort, 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.

Toni Leigh Partridge

Student Number: 0050010241

_____

Signature

_____

Date

Acknowledgements

The author wishes to recognise the support of the following people:

Prof. John Billingsley, University of Southern Queensland

For guidance and advice throughout the duration of the project and preparation of this dissertation.

For the use of his camera interface program, Veye, to use within the project.

Mr Anders Lööf, Chalmers University of Technology

For the use of his microcontroller application program, IOboard.asm, to use within this project.

Table of Contents

List of Figures

Chapter 1

1 Introduction

1.1 Introduction

Within this dissertation the field of control systems and their implementation to existing robots will be examined and finally carried out. This chapter will outline and discuss the aim, methodology and the general overview of the conducted research, design and construction of the control systems.

1.2 Project Aim

The aim of this project is to investigate suitable control strategies for the manoeuvring of a Load-Haul-Dump (LHD) unit within the confines of a mine. Further research is to be done specifically on a tele-operational and fully automated control system to investigate what would be required to implement such a system to a LHD unit. Due to the tele-operational system requiring the use of a camera, the automated system will also utilise this tool by using machine vision as its guidance system. The core of this project will then be the implementation of these two control systems to an existing scale model of a LHD unit.

1.3 Methodology

The following steps are the way in which this project will be undertaken. All of these will be conducted under the supervision of a professional engineer who will assist with guidance and provide some technical information and advice.

Research

Review relevant literature on the following:

- The key features of a typical LHD unit and what they are required to do.
- The implementation of automation. What automation can achieve? What would an automated LHD unit mean to the mining industry?
- The two control strategies: tele-operational and automated systems. What would be the advantages and disadvantages of each of these strategies? What equipment and tasks would be involved in implementing these systems?

This is done by searching databases, libraries and other public information sources.

LHD Model Analysis

Analyse the already constructed LHD model to ascertain how it all operates, as well as the mechanical, electrical and control features. This information was gathered by experimenting with it and testing its boundaries. The following could then be determined:

- Whether the existing mechanical features, circuitry and programming are appropriate for my project
- What equipment, components or programming needs to be re-designed to allow the unit to be tele-operated and automated.

Modelling

Review an appropriate modelling technique for the LHD unit, choose appropriate state variables and derive an algebraic equation for the control of the unit. This will be done by researching different modelling methods through databases, libraries and previous engineering related study. Simulation is used to test these, and to verify the unit will act how one would expect it to according to various inputs. This simulation will be conducted using Microsoft Visual Basic 6.0.

Programming

Construct the communication programs for the unit to run tele-operated and automated. This program will also be coded using Microsoft Visual Basic 6.0, and will include functions to do all of the following:

- Receive the visual from the unit to the computer
- Instruct the unit what direction it should go (for tele-operational), or detect the walls of the tunnel and move accordingly (for automated)

Testing

Finally, the LHD unit will be physically tested to ensure it will operate correctly for both of the control strategies. This will be done by putting together a simulated tunnel for the unit to manoeuvre its way through without encountering any collisions.

1.4 Conclusion

This chapter has introduced the project at hand, implementing a tele-operated and automated control system to an already constructed LHD model unit. Discussed in the next chapter is beneficial background information which will form the basis for latter sections of this report.

Chapter 2

2 Literature Review

2.1 Introduction

This chapter will review literature to establish the need for a suitable control strategy for the widely used Load-Haul-Dump (LHD) unit. After researching and gathering the relevant background information appropriate for this project, it is then possible to implement these strategies to an actual model LHD unit.

2.2 Load-Haul Dump Units

The load-haul-dump (LHD) unit is used extensively in underground mining to perform a variety of tasks within this field. Although their main role is to return and transport the mine's ore from the point of cutting to either dumping points, haulage trucks or crushing station (Tyson n.d.).

The underground mining environment is extremely dangerous where some LHD accidents have caused major injury and occasional deaths to the operators onboard these vehicles as well as other mining crew. These vehicles must be able to travel through narrow winding tunnels which have high temperatures,

dusty and dirty conditions as well as withstanding the occasional collision with the walls of the tunnel.

Typical LHD units, as shown in figure 1.1, are either diesel or electric powered, running on four solid rubber tyres and no suspension. They are made up of an articulated body with the two section connected by a kingpin hitch that can pivot. This articulated setup uses two hydraulic actuators to provide the steering for the unit which provides excellent curve negotiation within the tight winding tunnels.

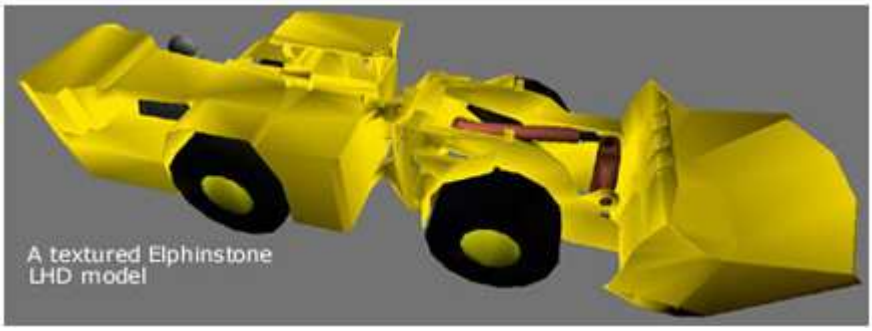

Figure 1.1: Load-Haul-Dump unit

Source: Dunn, P 2004, <www.mirarco.org/projectsmt.php>

The two sections of the unit both have a single axle with non steerable wheels, the front section contains the bucket or scoop, and the back section contains the engine. The profile of the vehicle complies with the cross section of the mining tunnels they are used in, this means they are long, low and relatively wide compared to their height (Tyson n.d.).

2.3 Control Strategies

A control system implemented onto a LHD unit would be ideal for the dangerous mine environments in which they operate, although would prove to be difficult due to the rough, unpredictable state of the mines. Stentz (2001) stated that a

semi-automated system would promote features such as increased productivity, reduced operational costs and improved safety.

There is a wide range of different control strategies that could be implemented onto a LHD unit. The two main categories of these are either infrastructure assisted guidance or an independent vehicle strategy that required no infrastructure construction or modifying of the tunnels (Billingsley 1997, p3). The advantage of the independent vehicle control strategy is that since no changes need to be made to the physical tunnel of the mine the LHD unit is being used in, the vehicle can be used in any mine at anytime.

The four main control strategies that were analysed and compared regarding the LHD unit were manual operation, remote control, tele-operational and fully automated.

2.3.1 Manual Operation

Manual operation is currently the most common strategy used in the mining industry, where a driver is onboard the LHD unit at all times positioned in a closed cabin. The cabin is positioned perpendicular to the direction of travel for viewing where the vehicle is going in the forward and backward direction.

The main disadvantage to this operating method is the concern of the drivers safety, driver fatigue and basic human error. The advantage to this method, compared to remote control and tele-operational methods, is that the unit can travel much faster through the tunnels as the operator is onboard, this is due to remote sensory perception (Robert et al. 2000).

2.3.2 Remote Control

A remote control vehicle means that the operator is out and distanced from the machine but still in the line of sight of the unit to control it via a remote control transmitter.

The advantage to this system is that the operator is located away from the immediate danger the LHD would otherwise put an onboard driver in. Saying this, there is still the risk of injury and accidents due to the line of sight rule and there is still the concern of driver fatigue and human error.

The disadvantage to this system is that the driver is not in any type of cabin area and since they must be in line of sight of the vehicle at all times it usually requires them to stand while controlling the unit. Consequently there is still the risk of injury with driver fatigue and human error still a concern.

2.3.3 Tele-operated Control

A tele-operated control system is similar to a remote controlled system, where the operator is still in full control of the vehicle at all times but there is greater distance between the operator and the device. The operator can be safe and comfortable aboveground controlling the vehicle by an operator interface. This interface consists of two basic components; the vision system and control panel (Hainsworth 2001, p.20).

The main disadvantage to this control system is that the vehicle is not able to be driven as fast as what could be achieved with a driver actually onboard the unit. This is due to the limited remote sensory perception (Robert et al. 2000).

2.3.4 Fully Automated Control

An automated system means the operator is still above ground, but he is playing a supervisory role to the system, hence they are still remote from the danger within the mine. There are a number of ways to implement an automated system the most common include machine vision, sensors and receivers, and GPS. These will be covered later.

The main advantage to this system is that since the vehicle is capable of autonomous steering throughout the underground tunnels, the LHD unit can travel at a greater speed, hence improving productivity whilst still maintaining a safe environment.

2.4 Machine Vision

Machine vision is the acquiring and processing of an image and then the deciphering of information presented in the image for controlling a specific purpose. It uses digital cameras, image processing software and relevant communication between digital input/output devices of the system it is controlling.

The key characteristics of any standard machine vision system is a digital camera with a camera interface program which captures the cameras image and converts the image into an array of numbers. This array represents the pixels of the image and the image is then manipulated or analysed by computer software depending on the application of the system.

2.5 Modelling

To successfully design a control system for any type of vehicle it is essential to have a model the can describe the vehicles position, orientation and other important vehicle parameters at any point in time. As stated by Ridley and Corke (2003), for a LHD unit the basic kinematic model is the most appropriate modelling method for the vehicle, although this can be a challenging task due to the unique articulated structure of the unit.

2.6 Conclusion

This chapter reviewed literature and established a clear view of the Load-Haul-Dump (LHD) unit, the type of modelling to be conducted and also a brief summation on the control strategies to be implemented to the vehicle. Before proceeding to start the project consideration needed to be taken in any predictable issues or problems which may arise undertaking this project, these are looked at in detail in the following chapter.

Chapter 3

3 Project Considerations

3.1 Introduction

Despite dealing with the specific problem at hand, it is also necessary to consider other very important issues associated with this project. These include sustainability, ethical and safety issues which will all be looked at in detail within this chapter.

3.2 Resource Analysis

The three resources required for this project are laboratory access, professional supervision and equipment and component purchasing. Out of these, the most important resources are the first two. If something were to happen with regards to the laboratory access this could cause a slight halt with work on the actual LHD unit. It was predicted that there would still be other work that could be accomplished without access to the actual model. If something were to happen and professional supervision was unable to be given by my supervisor, it was assumed that there would be other professional staff around that would be able to help with any project problems. This following section will look at the three stated resources and identify the purpose, source and its availability.

3.2.1 Laboratory

Purpose

The purpose of the laboratory provides physical access to the actual LHD unit, testing equipment, tools and basically just a work area which remains set up all the time with the conditions required.

Source

The laboratory is located at the University of Southern Queensland's (USQ) Toowoomba campus within the Engineering building, with access being granted by USQ's Faculty of Engineering and Surveying.

Availability

Availability was assured for the duration of the project after filling out the appropriate forms and obtaining an after hours access card and laboratory key.

Alternative

If there was no way to get into the laboratory there would be no alternative for getting access to the unit. It was assumed that work outside of the laboratory would be undertaken instead.

3.2.2 Professional Supervision

Purpose

The purpose of having a professional supervisor was to interact with, share ideas and to obtain assistance and guidance on project aspects.

Source

The professional supervisor has an office located in the Faculty of Engineering and Surveying faculty at USQ's Toowoomba campus. Communication with them was able to be done in person, via email or telephone.

Availability

The supervisor was available for consultations during office hours when an appointment had been made, with communication also able to be made via email and telephone at anytime.

Alternative

If this resource became unavailable it was assumed that there would be other professionals located at university to assist in this area.

3.2.3 Equipment and Components

Purpose

The purpose of this resource is to be able to easily obtain any electrical components, camera, lights or anything else which would be required for the project.

Source

Supplied by USQ's Faculty of Engineering and Surveying.

Availability

The Faculty of Engineering and Surveying Senior Technical Officer was available within business hours. If the equipment or components were not available on-site they would order them in, taking approximately three weeks for delivery.

Alternative

If purchasing problems had arisen, any of the components that were needed were obtainable from any computer / electronic store locally.

3.3 Sustainability Analysis

It was important to consider any potential sustainability issues which may occur during the execution and beyond the completion of this project. Sustainability, as defined by The Institute of Engineers, Australia in its Policy of Sustainability (cited Hood 1998, p3) is:

> "… the ability to maintain a high quality of life for all people, both now and in the future, while ensuring the maintenance of the ecological processes on which life depends and the continued availability of the natural resources needed."

After careful consideration of all sustainability aspects it was concluded that the project work would not negatively affect sustainability. This is due to the following factors:

- No air pollution – as the LHD unit is powered by battery
- No noise pollution – as the LHD unit is driven by electric motors
- No environmental damage – as the LHD unit testing would be carefully conducted without disturbance to the surrounding environment.

There should be no sustainability concern beyond the completion of this project provided that only authorised personnel use the machine with respect towards sustainability.

3.4 Ethical Analysis

It is essential that throughout the execution of this project that The Institute of Engineers, Australia's Code of Ethics (IEAust 2000) is abided by. After carefully considering each of the nine tenets of the code of ethics, the following statements were established.

Before commencing this project the designer and owner promised to abide by the following tenets throughout working on and testing the project.

1) They will ensure that the safety of the community is a major priority.

2) They will act with honour, integrity and dignity throughout the duration of the project.

3) They will always work carefully in their area of competency and if unsure or wanting reassurance on any aspect of the project they will seek assistance from their professional supervisor.

4) They will work with honesty and without discrimination.

5) They will apply all their skills and knowledge in the interests of the project, for self interest and satisfaction and others interested in the outcome.

6) They will inform others of any social or environmental consequences which may arise from their project work.

7) All work and opinions of the project will be given in honesty and on the basis of actual knowledge.

8) They will continue to develop knowledge, skills and experience relevant to what is being done in the project.

3.5 Risk Assessment and Management

This section outlines the safety issues associated with the project. As this project involved the design and construction of a robot, robot safety related problems were specifically looked at. Safety factors and hazards to consider included the safety of humans, the robot itself, and other equipment. All of these were considered during the planning phase for the successful robot.

3.5.1 Risk Identification

According to Ziskovsky and Addison (cited in Dhillon 1991, p.70) there are three main types of robot hazards. These include impact, trapping and those that occur due to its application.

3.5.1.1 Impact Hazard Identification

Impact hazards include being hit by the robot and/or its moving parts. In the case of this project, there is the risk of someone being struck by the LHD unit whilst it is manoeuvring around.

3.5.1.2 Trapping Hazard Identification

Trapping hazards refer to the movement of the robot in relation to fixed objects. For this project, the risk is the LHD unit becoming stuck between the walls of the test tunnel.

3.5.1.3 Application Hazard Identification

Hazards that occur from the robot's application, in this case, are associated with the operator at the control panel. The operator runs the risk of suffering from eye fatigue when positioned at the video monitor.

3.5.2 Risk Evaluation

3.5.2.1 Impact Hazard Evaluation

If the LHD unit were to strike someone during its operation, this would most likely be caused by human error, software error, or unauthorised and/or

inexperienced access. During the execution of this project, the likelihood of this occurring is only slight. This is due to the fact that the unit will be operated in a model tunnel in which no one will be standing. The consequences from this occurring are predicted to cause only minor injury to the struck person, such as bruising or scratching, due to the unit not travelling at excessive speeds. Only the designer will be regularly exposed to this hazard during the assembly and testing process.

Beyond the completion of the project, the risk will remain the same, unless an unauthorised and/or inexperienced person operates the unit. This would increase the risk to significant, resulting in greater consequences, including minor equipment and component damage as well as minor injury.

3.5.2.2 Trapping Hazard Evaluation

During the execution of the project, the risk of the LHD unit becoming trapped in the model tunnel would most likely be caused by human error, software error, or unauthorised and/or inexperienced access. This risk would have been significant during the developing process and would also have the consequence of minor equipment damage, limited by the fact the unit would not be travelling at high speeds. The exposure to this risk was predicted to be occasional during the initial training process, however, once an individual is experienced in operating the LHD unit, it is expected that the likelihood of this occurring would be very slight.

Beyond the completion of the project, these risks will also be dependant on an unauthorised and/or inexperienced operator.

3.5.2.3 Application Hazard Evaluation

The last risk to consider is the operator’s eye fatigue conditions, which would be caused by operating the LHD unit for excessive periods of time. During the execution and beyond the completion of the project, this risk is very slight due to the fact that it will not be operated for extremely long periods of time. The effect

of this is only minor, resulting in sore eyes and possibly headaches. It is expected that throughout the duration of the project only the designer would be regularly exposed to this.

3.5.3 Risk Control

These identified risks need to be managed appropriately to abide with the Workplace Health and Safety Act 1995, section 32(1) to (3).

3.5.3.1 Impact Hazard Control

The risk of someone being struck and injured during the operation of the LHD unit can be controlled by ensuring that the operator of the unit is authorised and experienced and that any observers are back from the demonstration area.

3.5.3.2 Trapping Hazard Control

To avoid the LHD unit becoming trapped within the tunnel, only an authorised and experienced person should operate the unit.

3.5.3.3 Application Hazard Control

Finally, to avoid operator risk conditions such as eye fatigue, the LHD unit should not be used for long periods of time. However, if such a situation occurs, ensure that an appropriate break away from the monitor is taken. Another way to combat this would be to have other operators able to take over control after designated time limits.

It is recommended that during and beyond the completion of this project, the LHD unit is used in a serious manner at all times. If any of the above risks occur, the unit should be stopped immediately and the issue dealt with appropriately.

3.6 Conclusion

After taking all of these issues into consideration and analysing them before commencing the project, the chance of encountering one of these concerns was reduced. If encountered, the project designer would be prepared for what could happen and be able to deal with it appropriately. Now the project work can commence, the next part to look at is the analysis of the current LHD model.

Chapter 4

4 LHD Model Analysis

4.1 Introduction

In this chapter, the previously designed LHD unit will be introduced and analysed in the sense of its mechanical and electrical workings and also its current control system. From here the suitability and requirements of these were decided for the implementation of the two control strategies.

4.2 The Analysis

The implementation of the two control strategies were done on an existing LHD model unit which was constructed six years ago and is a one fifth scale model of the vehicle which was designed to operate as closely as possible to an actual LHD unit. The unit, shown below in figure 4.1, is approximately 2000mm in length, 500mm wide and 500mm tall.

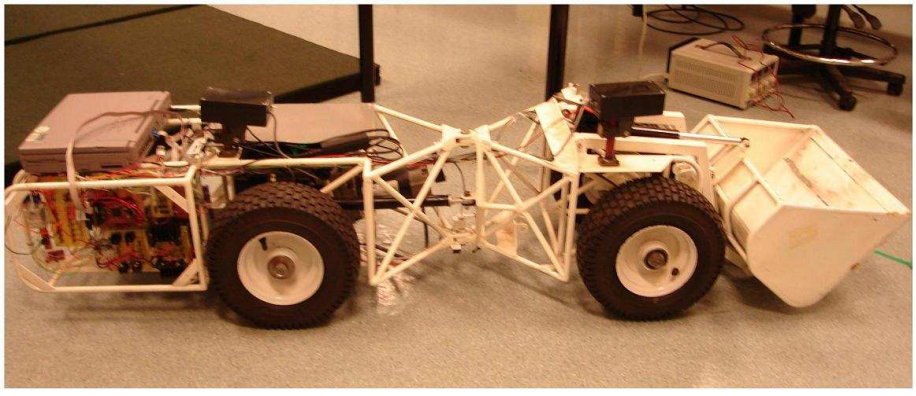

Figure 4.1: The LHD model

Since the model was built many years prior to this project it had previously had a number of different control strategies implemented onto it, some of these included remote control capabilities, line following and GPS capabilities. This meant it was necessary to analyse this current model to determine what was needed or required to be redesigned to implement tele-operationed and automated control system.

The analysis done on the model LHD involved investigating its existing mechanical, electrical and control system, these three aspects are looked at and discussed in detail below.

4.2.1 Mechanical Analysis

4.2.1.1 Driving the LHD

The LHD unit is powered by a DC motor rated at 24V. This drives the right rear axle after going through a simple gear reduction; this can be seen in figure 4.2.

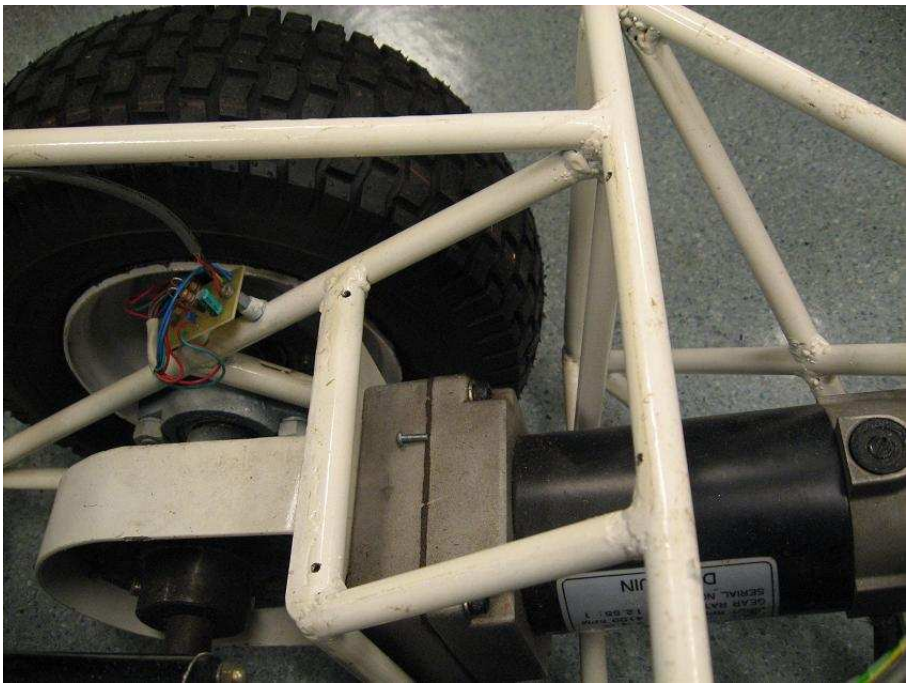

Figure 4.2: LHD motor and gear reduction

4.2.1.2 Steering the LHD

Like real LHD units the model is steered by its centre articulation through two pivot points. This is done by an electric linear actuator which when extended and retracted changes the angle of the pivot and inturn steering it in that corresponding direction as seen in figure 4.3.

To prevent the unit from turning too far and folding up on itself the model is fitted with two micro switches at the bottom of the pivot. When the LHD turns up to that specified point, the switch is engaged and the actuator stops movement in that direction.

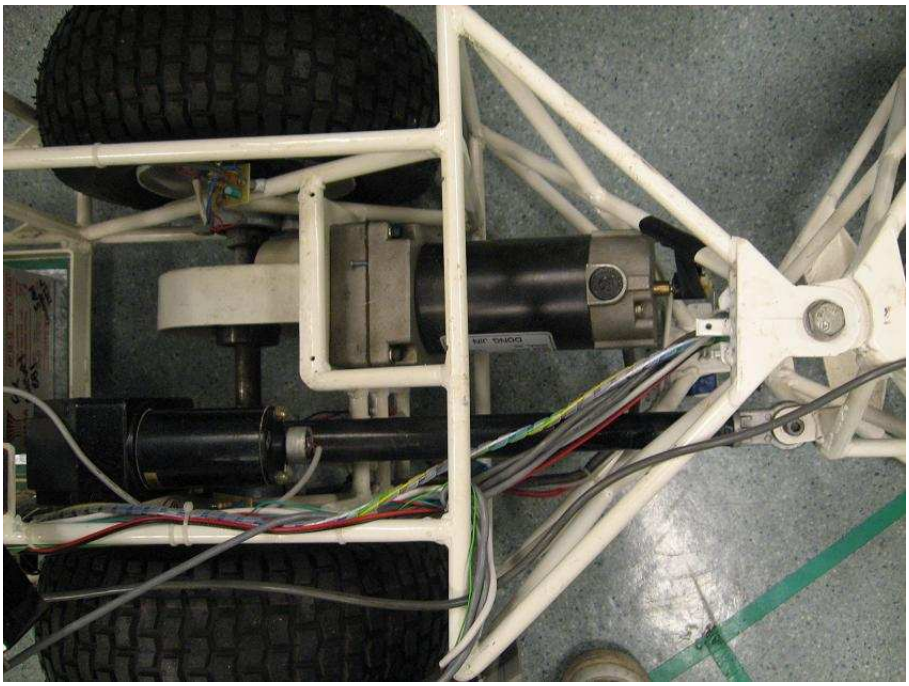

Figure 4.3: LHD unit showing drive and steering setup

4.2.1.3 Bucket of LHD

The bucket is also controlled by electric linear actuators, where one is used to raise and lower the bucket and the other actuator is used to pivot the scoop for loading and unloading.

4.2.2 Electronic Analysis

The electronic analysis examined the power supply, the drive and steering system and the odometry set-up. Figure 4.2 shows the overall circuitry located on the back of the LHD unit.

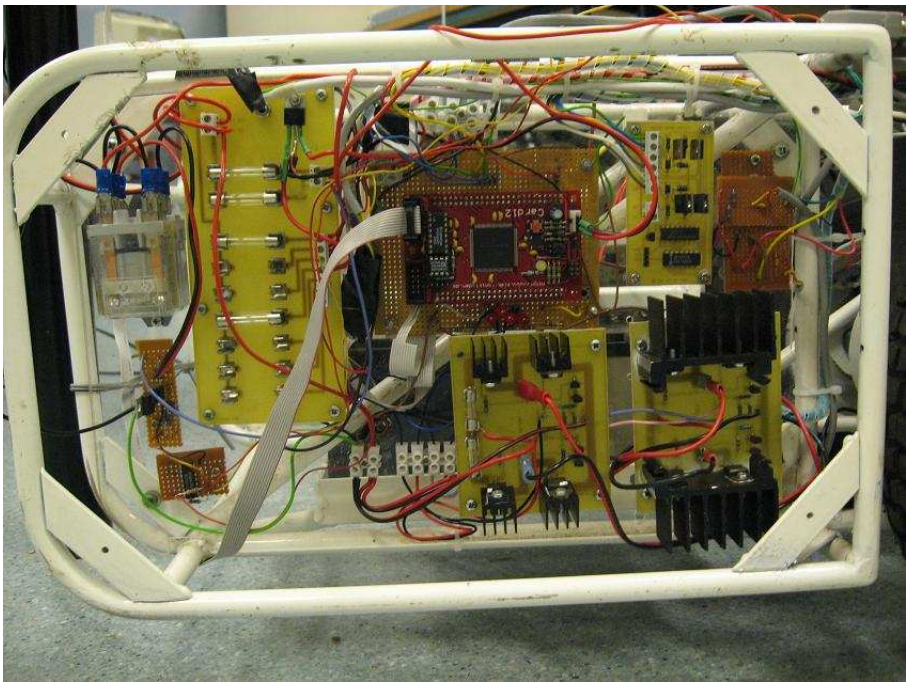

Figure 4.4: LHD unit circuitry

4.2.2.1 Power Supply

The LHD unit runs on two 12V batteries, one supplies power to the circuitry and the other to the mechanics of the system, such as the motor and linear actuators. It was assumed that the purpose of the two batteries was to reduce noise of the system. The battery powering the circuitry first goes through a toggle switch to a voltage regulator to take the 12v voltage and output 5v for the microcontroller.

4.2.2.2 Drive and Steering System

For both the drive and steering application a h-bridge and transducer set-up is used. A two bit logic code is sent out from the microcontroller through four Led

Emitting Diodes (LED)'s which light up according to the signal being transmittedby the h-bridges. Each of these circuits consisted of four power mosfets, two transistors, multiple resistors and heat sinks.

These LED's proved beneficial for testing during the programming stage of the project as it meant the motors did not need to be turned on for running the program, as the output response could be monitored via this light set-up. The LED's output combinations for steering and driving are illustrated in figure 4.4 and 4.5, respectively.

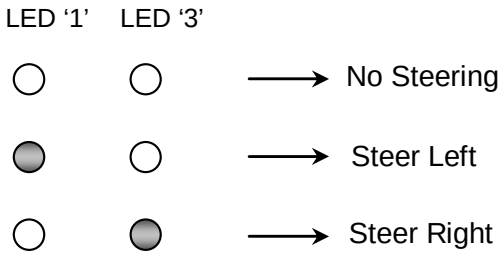

Figure 4.5: LED light up for steering

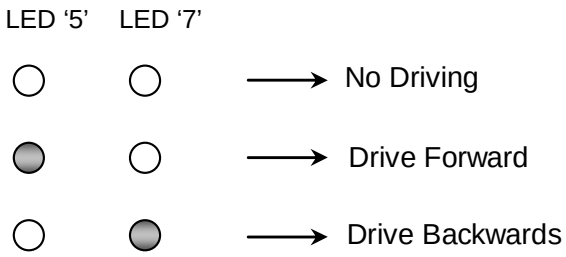

Figure 4.6: LED light up for driving

The drive circuitry uses a h-bridge and according to the logic code it is receiving will either turn the DC motor forwards or backwards. The steering set-up is similar, as it will either extend or retract its linear actuator according to the input from the microcontroller.

The final and very important part of the drive and steering circuitry is the velocity and steering angle feedback transducers. The velocity feedback is obtained by attaching a small DC motor onto the armature of the motor to act as a

tachometer. The steering angle feedback was generated by having a small hall-effect sensor located on the pivot of the centre articulation.

4.2.2.3 Odometry

The odometry for the system consists of a wheel encoder on each of the four wheels. These are two phase light gates which monitored a slotted disk that is attached to the axle of each wheel, as shown in figure 4.7. The signals from these switches are then feedback into the microcontroller for analysis.

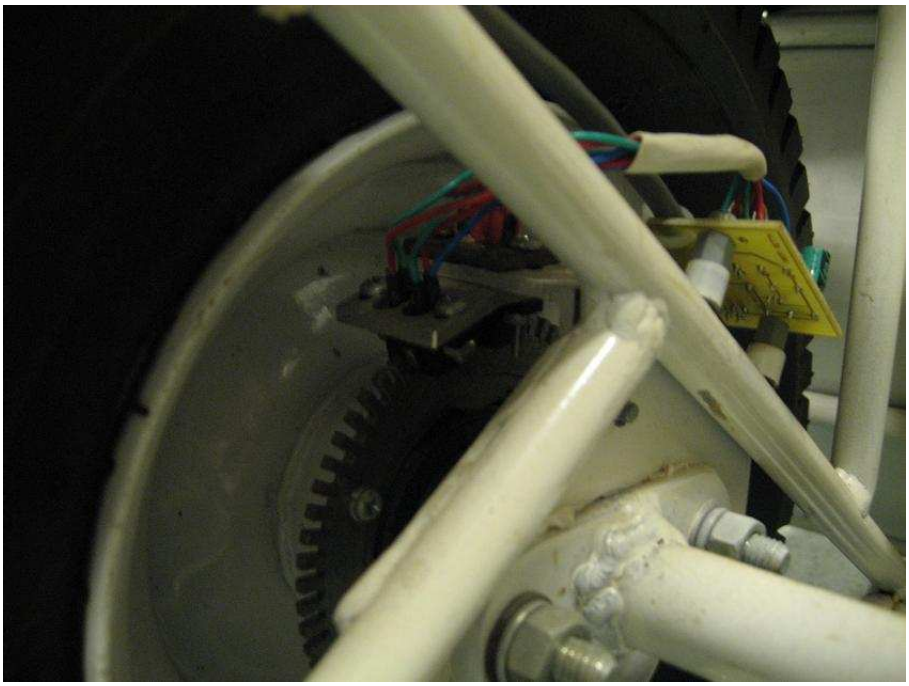

Figure 4.7 Wheel encoder

For this particular project odometry was not used for controlling the unit or monitoring, hence nothing is done with the signal feedback to the microprocessor.

4.2.3 Control Analysis

The basic control structure, shown in figure 4.8, of LHD unit consists of a personal IBM computer, which is located remote from the vehicle, used for delegate a variety of tasks to the onboard microcontroller through a bi-directional RS232 standard serial. The microcontroller onboard the unit is a Motorola HC12 card which deals with the incoming tasks and performs specific operations to achieve them. This involves sending out the relevant commands to either the steering of drive actuators.

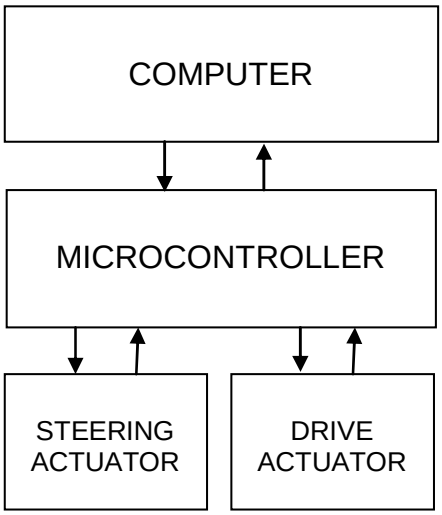

Figure 4.8: Basic control structure

4.2.3.1 Microcontroller

The LHD unit is equipped with a Motorola Card12 controller module. This card is small in size, easy to use and is ideal for low and medium series applications.

The connections made to the HC12 are listed below:

- Power Supply: 5V from voltage regulator
- Serial Port: Connected to computer through RS232 transceiver
- Analogue to Digital Converter Inputs: Using two pins, one for tacho and the other for hall effect sensor
- Port G Outputs: Four pins to the H-bridges which go through four LED’s

The application program for this microcontroller was downloaded from the computer using the terminal program, OC-Console at a rate of 19200 Bd. The program is discussed in greater detail in the following section.

4.2.3.2 Application Program

There were several applications programs on the IBM personal computer which were all used on the LHD at some stage for different applications. These included a line following program, GPS navigator and a remote control strategy. After searching through and deciphering many of these programs, the most appropriate one was found, *IOboard.asm*. It had been previously used when the vehicle was being use with remote control capabilities which meant it would be ideal for both the implementation of the tele-operational and automated control system.

This program, see appendix B, after setting up the interrupts runs constantly until an interrupt to the system was made. The interrupt handler of the program then reads in the bytes from the status register and it checks for any of the six following commands. If none of these conditions are satisfied an error has occurred and an error message is returned.

91 - Sets steering active
90 - Sets velocity active
A1 - Read current velocity value (AD1)
A0 - Read current steering angle (AD0)
81 - Start the real time interrupt (RTI)
80 - Stop the real time interrupt (RTI)

When either the steering or velocity is set active the program then prepares for the next byte received to be the value of that active component. This hexadecimal value received corresponds to different actions to be performed by the steering and velocity commands. This value is sent out through the G-Port

of the microcontroller where it then travels through the LED's (as discussed previously) to the h-bridges to implement the given task.

The current steering and velocity values are able to be read by sending the relevant steering command to the controller. Similarly, the real time interrupts can be manually started and stopped by the use of these commands.

Once the current commands have been dealt with the interrupts are reset to await the next command.

4.3 Decisions Made

After much consideration and testing into the current design of the LHD unit it was decided that the mechanical, electrical and basic control of the vehicle was suitable for the implementation of a tele-operational and automated system.

The additions made to the LHD unit were a light and a simple camera. The light was a LED torch bright enough to produce the light ray to shine out and reflect off the wall of the tunnel, shown in figure 4.8, and was mounted on the front, right corner of the scoop shinning perpendicularly outwards to the direction it is heading.

Figure 4.9: Light shining onto wall

The camera was just a basic Creative webcam which was connected to the computer via USB cable. It was mounted in the centre of front section of the vehicle and positioned facing forwards and slightly to the right, this was so when the unit was approximately 300mm away from the right wall of the tunnel the splosh on the wall from the light beam would be in the centre of the screen, as illustrated in figure 4.9.

Figure 4.10: View of real and computer perspective splosh

4.4 Conclusion

Examining and testing the current LHD unit explained how the unit operated in its mechanical, electrical and control sense, but also how it worked as a whole. It was then possible to determine what was not required for the implementation of the two control strategies. The next chapter will look at the mathematics behind the operation and controlling the LHD unit.

Chapter 5

5 Design and Modelling

5.1 Introduction

To design a control system for any system it is necessary to have a vehicle model that describes the progress of its position and orientation at any point in time. This chapter will look at the systems state variables chosen and the mathematics done in order to describe the LHD unit as a manoeuvring vehicle.

5.2 Variables and Key Features

For determining the mathematics behind the LHD's kinematics only a basic view of the vehicle was used showing only necessary key characteristics. This simply consisted of the front and rear axles of the unit joining at the centre hinge pivot as shown below in figure 5.1.

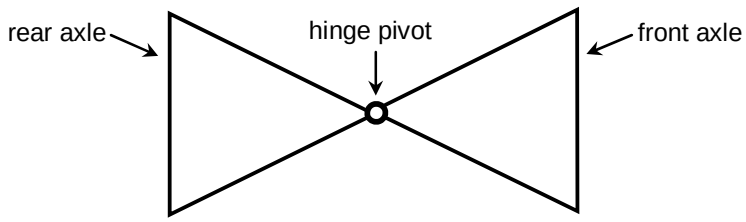

Figure 5.1: Basic model of LHD unit

From this basic model of the LHD unit, variables and key features were chosen which aided in the deriving of the vehicles kinematic structure. Much consideration was taken in selecting these variables as they had to be able to be used throughout the entire project, be easily defined and be able to describe all the crucial characteristics of the vehicle for the manoeuvring. The variables and key features decided on are illustrated and explained below.

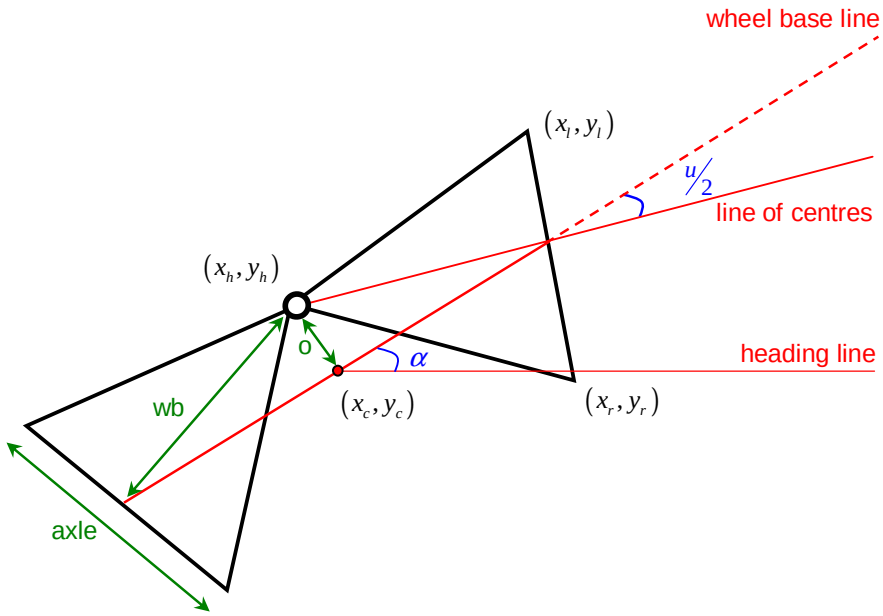

Figure 5.2: Variables and key features of LHD unit

Key Features:

‘Line of Centres’ - Line that joins the centre of the front and rear axles

‘Heading Line’ - Line that indicated the desired direction of travel

‘Wheel base Line’ - Line that goes through hinge pivot and centre of axle

Coordinates:

(x_c, y_c) - Position of the centre of Line of Centres

(x_h, y_h) - Position of the hinge pivot

(x_r, y_r) - Position of the front right wheel

(x_l, y_l) - Position of the front left wheel

Distances:

axle - Length of the axles

wb - Length of the half the wheel base

o - Distance the centre of the Line of Centres is from hinge pivot

Angles:

$u/2$ - Angle between the Line of Centres and the Front Section Line

α - Angle between the Line of Centres and the Heading Line

The most essential key feature identified was the *Line of Centres*, which was chosen for its ability to be used in the description of more than one variable, and in turn making the mathematics much easier to relate these features together. The other important quality of this line is that the coordinates for the centre of this line (x_c, y_c), can be used as the main position variable for LHD unit over the actual hinge point, (x_h, y_h). This is due to the fact that when the steering angle of the vehicle changes, it instantaneously changes the position of the hinge.

5.3 State Variables

State variables were chosen to be able to determine the complete state of the LHD unit and in turn represent the vehicle at any point in time. After much consideration of all the key aspects of the LHD unit there were three state variables chosen, these included the centre point of the imaginary axle line, the heading angle and finally the steering angle of the unit. These three variables are discussed further in detail below.

5.3.1 Coordinates (x_c, y_c)

The first state variable chosen was the x and y coordinates of the centre point of the line of centres (x_c, y_c). This hinge point (x_h, y_h) of the LHD unit always lay perpendicular to this point.

For any point in time this state variable was able to describe the rate of distance travelled in both the x and y direction. From these equations the actual coordinates of the centre point were described for a small time increment.

$$\dot{x} = V.\cos(\alpha) \tag{1.1}$$

$$\dot{y} = V.\sin(\alpha) \tag{1.2}$$

From equations (1.1) and (1.2) calculate x and y

$$x = V.dt \cdot \cos(\alpha) \tag{1.3}$$

$$y = V.dt \cdot \sin(\alpha) \tag{1.4}$$

Hence, when this point is continually changing for every time step, the new position can be determines from the current centre and previous centre point plus the increments from equations (1.5) to (1.6).

$$x_{c_{new}} = x_c + V.dt \cdot \cos(\alpha) \tag{1.5}$$

$$y_{c_{new}} = y_c + V.dt \cdot \sin(\alpha) \tag{1.6}$$

5.3.2 Heading Angle (α)

The next variable was the angle the line of centres line is positioned at; it is described as the angle between the line of centres and the desired heading. This angle was used to describe the heading of the vehicle at any point in time, see figure 5.3.

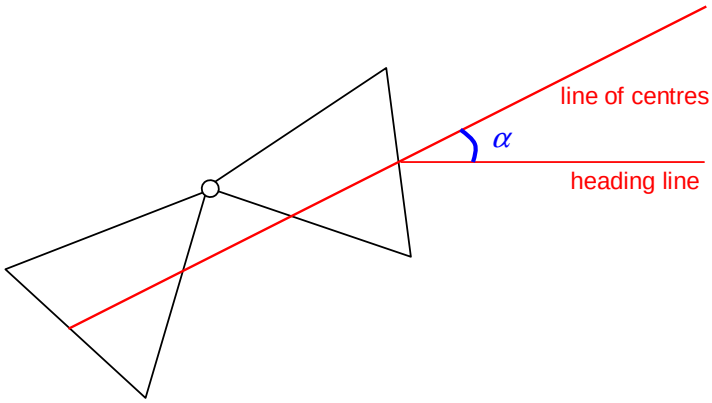

Figure 5.3: Heading angle

As illustrated in figure 5.4, when the vehicle is travelling at a constant speed the front and rear axle centres both move forward in the next time increment the same distance. This obviously changes the line of centres but it can be seen that the line is rotating about a circular path. Hence this state variable can also describe the rate of rotation of the vehicle of at any time.

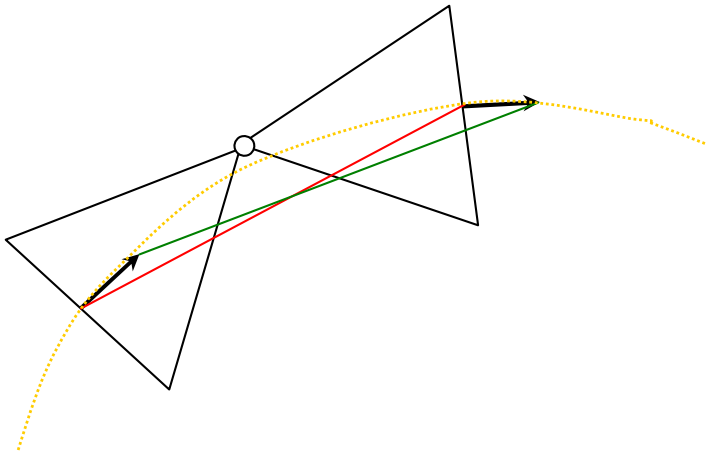

Figure 5.4: Circular rate of rotation of line of centres

The rate of rotation of the system cab be defined as.

$$\dot{\alpha} = \frac{V}{L} \cdot u \tag{1.7}$$

Where L is the length of the line of centres line.

$$L \approx 2 \cdot wb \tag{1.8}$$

Substituting equation 1.8 into 1.7 the following is obtained

$$\dot{\alpha} = \frac{V}{2 \cdot wb} \cdot u \tag{1.9}$$

From equation 1.9, the change in the heading angle can then be described in terms of the time interval, dt.

$$\alpha = \frac{V}{2 \cdot wb} \cdot u \cdot dt \tag{1.10}$$

Therefore, for the simulation purposes, the new steering angle of the unit can be continuously calculated from the below equation, which is just the previous heading angle plus the change in direction, equation 1.10.

$$\alpha_{new} = \alpha + \frac{V}{2 \cdot wb} \cdot u \cdot dt \tag{1.11}$$

5.3.3 Steering Angle (u)

The two above state variables describe both the x and y coordinates and the heading angle, or the position and orientation of the front section of the vehicle at any specific point in time. Next the angle the two axles are positions with respect to each other needed to be defined. This variable was named the steering angle of the LHD unit and is the actual angle of the hinge pivot of the articulated body. As shown below, half of this angle is also the angle between the line of centres and the wheel base line.

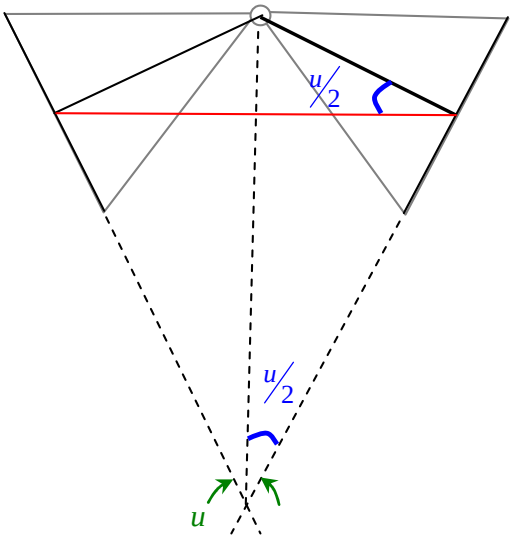

Figure 5.5: Steering angle

5.4 Geometry Calculations

5.4.1 Coordinates of Hinge Point

The hinge point (x_h, y_h) of the LHD unit always stays perpendicular to the centre of the line of centres (x_c, y_c) both its perpendicular distance and the actual position of this point can be determined, as illustrated in figure 5.6.

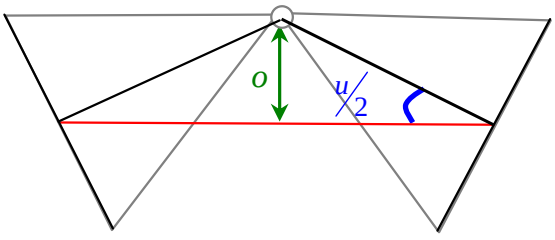

Figure 5.6: Offset distance

The distance the hinge point is offset from the centre of the line of centres is calculated using basic trigonometry as follows:

$$\sin\left(u/2\right) = \frac{o}{wb}$$

$$o = wb \cdot \sin\left(u/2\right) \qquad (1.12)$$

Further geometry calculations were carried out to determine the x and y coordinates of the hinge point using the recently determined offset distance and heading angle of the unit.

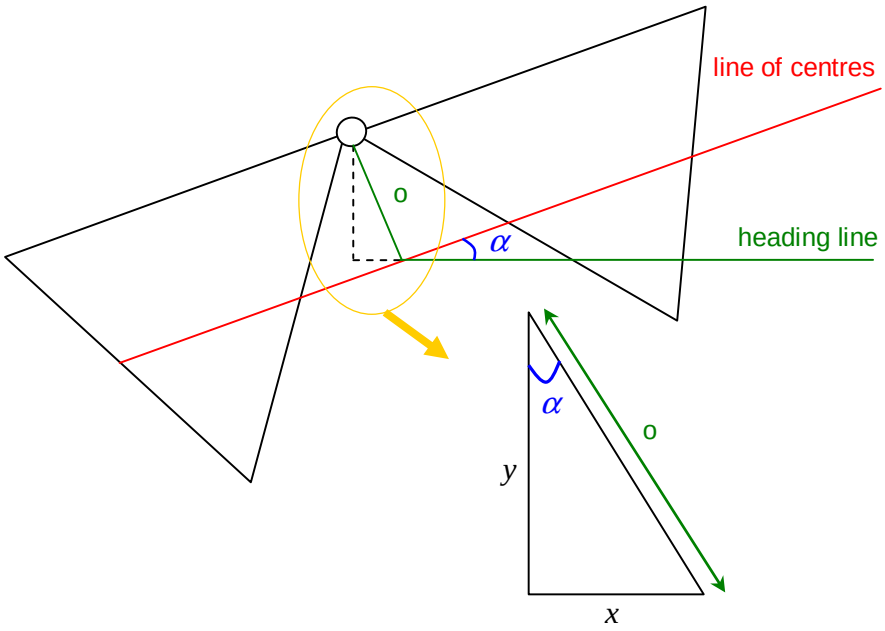

Figure 5.7: Determining coordinates of hinge pivot

The x and y position of hinge with respect to centre of the line of centres.

$$\sin(\alpha) = \frac{x}{o}$$

$$x = o \cdot \sin(\alpha)$$

$$\cos(\alpha) = \frac{y}{o}$$

$$y = o \cdot \cos(\alpha)$$

Therefore, for the simulation purposes, the position of the hinge pivot of the unit can be calculated from the below equation.

$$x_h' = x_c + o \cdot \sin(\alpha) \tag{1.13}$$

$$y_h' = y_c + o \cdot \cos(\alpha) \tag{1.14}$$

5.4.2 Coordinates of Front Axle

The coordinates of the right and left end of the front sections axle are determined purely for the drawing for LHD unit in the simulation. It is calculated from the hinge pivot point up towards the centre of the front axle and finally out to the ends of the axle, each illustrated in figure 5.8.

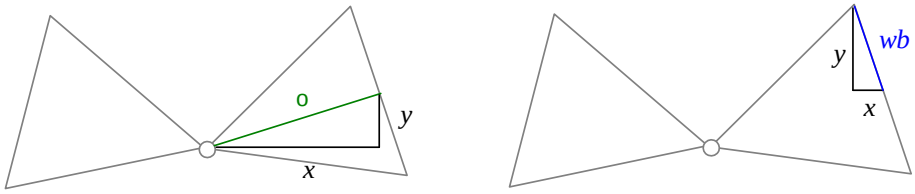

Figure 5.8: Steps in determining front axle coordinates

Left axle corner:

$$x_l = wb \cdot \cos\left(\alpha + u/2\right) - axle/2 \cdot \sin\left(\alpha + u/2\right)$$
$$y_l = wb \cdot \sin\left(\alpha + u/2\right) + axle/2 \cdot \cos\left(\alpha + u/2\right) \tag{1.15}$$

Right axle corner

$$x_r = wb \cdot \cos\left(\alpha + u/2\right) + axle/2 \cdot \sin\left(\alpha + u/2\right)$$
$$y_r = wb \cdot \sin\left(\alpha + u/2\right) - axle/2 \cdot \cos\left(\alpha + u/2\right) \tag{1.16}$$

5.6 Conclusion

This chapter look at all the maths required for knowing and determining the position, orientation and other variables of the system at any point in time. To ensure that these math models were correct a simulation was produced, hence the following chapter will discuss the simulation of the LHD unit.

Chapter 6

6 Simulation

6.1 Introduction

A simulation of the LHD unit was created for the basic testing of the mathematics done behind the manoeuvring of the vehicle. This chapter will look at the reasons behind constructing a simulation, detail into programming with routines and finally the results from the simulation.

6.2 Purpose of Simulation

The main purpose of the simulation was to test that the mathematical model for the LHD unit was correct and that the unit was going to perform how expected. This simulation specifically tested both the basic mobility of the LHD using the derived state variables and also the detection and avoidance system before practically implementing the control strategies to the real model.

6.2.1 Mobility

To test the mobility of the LHD unit only a basic drawing of the unit was used showing only the key features of vehicle. This drawing was then simulated to

drive across the screen at a constant speed while having a basic input system to change the steering angle as it drove along.

The basic features illustrated in the simulation only consisted of the front and rear axles connected to the centre hinge point. This was all that was needed to observe the manoeuvring of the unit was working correctly.

The simulation also illustrated how the rear axle followed the front axle during general driving and manoeuvring, this in turn showed how the articulated steering configuration was ideal for tight turning within a tunnel. This was achieved by leaving a white trial behind the unit of where it had been.

6.2.2 Detection and Avoidance

The vehicles ability to detect and avoid obstacles was an important aspect to simulate as it would examine and determine how the automated system would run before applying it to the actual LHD model.

As the automated system would consist of light rays shining out from the front of the vehicle and hitting the walls of the tunnel the simulation was constructed in a corresponding way. It used rays which headed out in front of the unit and would then detect where it hit the wall, calculate the distance from the wall and finally determine which direction it must move to avoid a collision.

The simulation got the vehicle to then manoeuvre through a short tunnel. This tunnel was drawn onto into the simulation and consisted of reasonably tight winding corners that required the unit to turning both left and right to avoid hitting the wall.

6.3 The Simulation

The simulation was programmed using the Windows programming language Visual Basic 6.0. The following sections will discuss the variables used throughout the program and also a detailed look at each of the routines and subroutines used to produce the simulation. Refer to Appendix B for the simulations complete code.

6.3.1 Variables

The variables used throughout this simulation were declared as single type floating-point numbers which meant they could be either extremely large of small. These variables are listed below.

X Position of Centre of Line of Centres (x_c) -	xmid,	x	
Y Position of Centre of Line of Centres (y_c) -	ymid,	y	
Heading Angle (α) -	angle,	a	
Steering Angle (u) -	steer,	u	
Distance hinge is off line of centres (o) -	offline		
X-Coordinate of Corner of Left Axle -	lx		
Y-Coordinate of Corner of Left Axle -	ly		
X-Coordinate of Corner of Right Axle -	rx		
Y-Coordinate of Corner of Right Axle -	ry		
Right Ray Length -	rright,	rr,	rray
Left Ray Length -	rleft,	lr,	lray
Drawing Colour -	c		
Time -	t		

The constant values which never changed and could be permanently stored within the program are listed below with their values.

Velocity (v) -	v	= 1.0
Length of half the Axle ($axle$) -	haxle	= 1.0
Length of half Wheel Base (wb) -	wb	= 1.5
Change in Time Step -	t	= 0.0001
Value of π -	pi	= 3.1415926

Throughout this simulation program all of these variables were explicitly declared using the *Dim* and *Const* statement before they are actually used anywhere within the simulation. This must be done as the *Option Explicit* statement was added at the start of the program. The use of this statement avoided the error that could have occured if a variable name was written incorrectly somewhere within the program.

6.3.2 Routines

Form_Load Routine

This is the main routine of the simulation which runs as soon as the form is loaded with all the other subroutines executing within this routine. The variables used within this section of the simulation are for the lengths of the right and left light rays, *rright* and *rleft*, respectively.

After the screen is scaled a simple tunnel is drawn onto this form in a wide black line. This thickened line ensured that the replicated light rays would be able to detect either wall without jumping over it during their length increments. Once the tunnel was drawn the drawing width was returned to its regular size for future drawings.

```
Scale (-10, -30)-(30, 10)
Show

DrawWidth = 10
Line (0, 4)-(3, 4)
Line (3, 4)-(9, 1)
Line (9, 1)-(13, -3)
Line (13, -3)-(15, -7)
Line (15, -7)-(18, -9)
```

```
Line (18, -9)-(20, -10)
Line (20, -10)-(25, -10)

Line (0, -4)-(3, -4)
Line (3, -4)-(7, -7)
Line (7, -7)-(9, -11)
Line (9, -11)-(11, -13)
Line (11, -13)-(15, -16)
Line (15, -16)-(17, -17)
Line (17, -17)-(25, -17)
DrawWidth = 1
```

The routine then enters the main loop of the simulation which runs for a set time. It is from this loop that the other subroutines are called from to either obtain important values or to draw the different sections of the LHD. This loop is shown and then discussed in detail below.

```
Do
    rright = 1
    showtruck xmid, ymid, angle, steer, rleft, rright, vbWhite

    If rright > 3 Then
       steer = steer + 3 * dt
    Else
       steer = steer - 3 * dt
    End If

    angle = angle + v * steer / 2 / wb * dt
    xmid = xmid + v * dt * Cos(angle)
    ymid = ymid + v * dt * Sin(angle)

    rright = 0
    showtruck xmid, ymid, angle, steer, rleft, rright, vbBlack

    DoEvents
    t = t + dt

Loop Until t > 30
```

The first task within the main loop was to determine how far away from the walls the LHD unit actually is, hence the routine goes to the subroutine *showtruck* where it has the choice of going to the light ray simulation, *showprobe*. This is done by setting *rright* variable to something greater than zero, as shown below.

```
rright = 1
showtruck xmid, ymid, angle, steer, rleft, rright, vbWhite
```

It then returns from this subroutine with the actual verified right ray value and hence the distance the vehicle is away from the wall. To avoid a collision with the right wall or moving too far away from it and risk hitting the left tunnel wall simple *if* and *else* statements were used for the two different conditions, as illustrated below.

```
If rright > 3 Then
   steer = steer + 3 * dt
Else
   steer = steer - 3 * dt
End If
```

The simulation then goes on to calculate the new values for the system as described earlier in the chapter five. Every time the simulation runs through this loop the *angle*, *xmid* and *ymid* will continually be updated for the correcting steering angle. This is done using the derived equations (1.5),(1.6) and (1.11).

```
angle = angle + v * steer / 2 / wb * dt
xmid = xmid + v * dt * Cos(angle)
ymid = ymid + v * dt * Sin(angle)
```

For these new values the LHD unit is then drawn onto the screen be executing the *showtruck* subroutine but this time jumping to the general *showhalf* routine by the declaration of right is not greater than zero.

```
rright = 0
showtruck xmid, ymid, angle, steer, rleft, rright, vbBlack
```

This is the end of the main loop and hence the time is incremented by *dt* and continues to run through this process repeatedly until the time is greater than thirty.

Showtruck Routine

The aim of this routine is to determine which of the two subroutines, *showprobe* and *showhalf*, to execute. Both are used for drawing the front section of the LHD unit, the only difference is the *showprobe* is also used to find out the distance the vehicle is away from the tunnel wall.

Before decided on which routine to employ it first takes the variables from the main routine and transforms them into its own values of *x*, *y*, *a*, *u*, *lr*, *rr*, and *c.* From here the offline distance of the line of centres to the hinge point is calculated from equation (1.12), and is used to calculate the coordinates of the centre hinge point from the equation (1.13) and (1.14).

```
If rr > 0 Then
   showprobe x+offline*Sin(a),y-offline*Cos(a),a+u/2,lr,rr,c
Else
   showhalf x+offline*Sin(a),y-offline*Cos(a),a+u/2,c
End If

showhalf x+offline*Sin(a),y-offline*Cos(a),a-u/2+pi,c
```

The factor used to decide whether to go to the subroutine that does the light ray calculations is dependent on if the right ray value, rr, is greater than zero. The value of this variable is set in set up in the *Load_Form* routine.

The final part of this program has the execution of the *showhalf* subroutine, although this one is for drawing of the rear section of the LHD due to the addition of pi to the angle. This means it draws the refection of the front section of the vehicle with respect to the angle.

Showprobe Routine

The purpose of this subroutine was to draw the front section of the LHD, send out the replica light rays towards the walls of the tunnel and then obtain the length of these rays from the point of shining to the contact with the wall. The length of these rays are then taken back to the loop within the *Form_Load* routine where the appropriate steps are taken to control the LHD unit accordingly.

At the start of this subroutine all the variables from the previous routine are redefined as *x*, *y*, *a*, *lray*, *rray*, and *c*. New variables were then introduced for the simple geometry of the front half of the vehicle; these were *lx*, *ly*, *rx*, *ry*, *ca*, and *sa.* The last variable type introduced within this subroutine is the Boolean variable for *doneright* and *doneleft* where the value should either be true or false.

To begin with, the program calculated the right and left coordinates for both corners of the front section of the LHD unit using the equations (1.15) for the left and (1.16) for the right corner.

Within the routine the sine and cosine parts are first calculated on their own according to the angle, this was done to save calculating both of them each time they were used. This section of the program is shown below:

```
ca = Cos(a)
sa = Sin(a)

lx = wb * ca - axle * sa
ly = wb * sa + axle * ca
rx = wb * ca + axle * sa
ry = wb * sa - axle * ca
```

The initial values for the variables were then defined before entering the tunnel wall detection loop. This set both the right and left ray to a starting length of 0.1 and then the Boolean variables to false to indicate no wall was detected.

The tunnel detection loop calculates the first point, which is just an x, y coordinate, on the form to look at with respect the current light ray length. The simulation looks to see if that point is coloured black, if it is the Boolean variable for that corresponding ray is then set to true and if not it is reset to zero.

These equations were then put into the simulation loop as shown.

```
Do
   If Point(x+rx+(ca+sa)*rray,y+ry+(-ca+sa)*rray)=vbBlack Then
      doneright = True
   End If
```

```
      If Point(x+lx+(ca-sa)*lray,y+ly+(ca+sa)*lray)=vbBlack Then
         doneleft = True
      End If

      rray = rray + 0.2
      lray = lray + 0.2

   Loop Until doneleft Or (lray > 5) And doneright Or (rray > 5)
```

This loop continues to loop, each time incrementing the ray length, by 0.2, until either a tunnel wall is detected or there is obviously no wall close enough for the ray to hit.

A coloured dot was then drawn on the screen for anywhere where the tunnel of the wall was detected. This served no purpose apart from a visual indication of which and where each light ray was picking up the black drawn line of the tunnel. A seen in the code below, if the particular ray was not the one that detected the wall it would be set back to zero.

```
If doneright Then
   PSet (x+rx+(ca+sa)*rray, y+ry+(-ca+sa)*rray), vbGreen
Else
   rray = 0
End If

If doneleft Then
   PSet (x+lx+(ca-sa)*lray, y+ly+(ca+sa)*lray), vbRed
Else
   lray = 0
End If
```

The last part of this subroutine actually drew the front section of the LHD unit from the earlier calculated axle corner values.

Showhalf Routine

This routine is used for drawing either the front or rear section of the LHD unit, it all depends on the angle received from the *showtruck* subroutine. This has all the same drawing mathematics and simulating operations as *showprobe* routine except for the fact that it does not have anything to do with the light rays.

It transforms the variables from the previous routine to *x*, *y*, *a*, and *c* and then goes onto calculate the corner axle coordinates. From here the coordinate points are joined using the following code.

```
ForeColor = c

Line (x, y)-Step(lx, ly), c
Line -Step(rx - lx, ry - ly), c
Line -(x, y), c
End Sub
```

The *ForeColor* command is also obtained from the previous routine and can be either black or white. The black is for the current instantaneous picture of the LHD where the white is the 'rub-out' making of where the unit has been.

6.4 Simulation Results

The following figure, Figure 6.1, shows the basic tunnel that was drawn within the program, as well as the basic LHD model that was continually being redrawn for ever step in time for the ever changing steering angles.

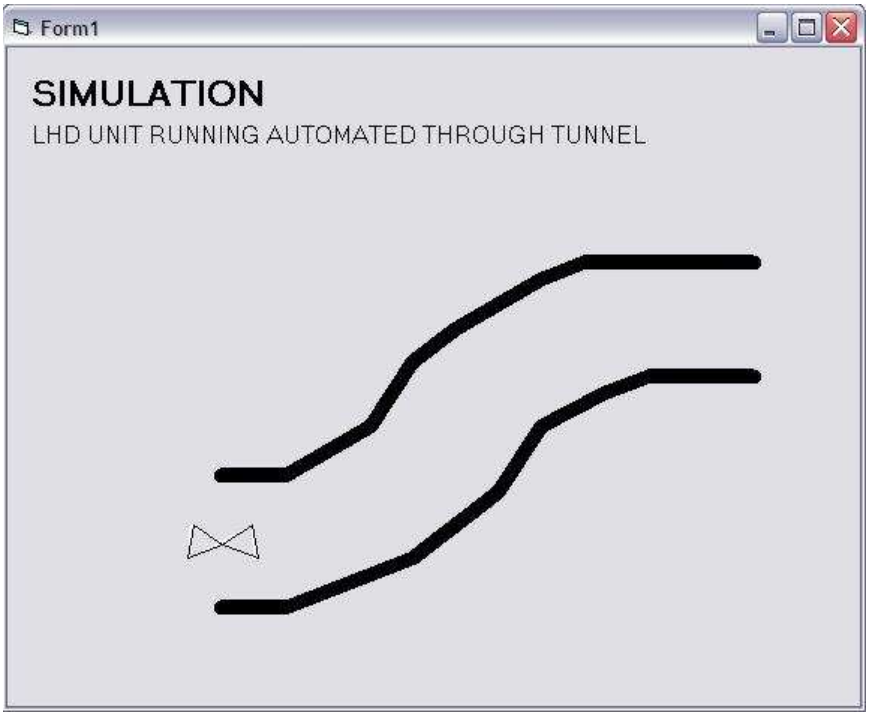

Figure 6.1: Tunnel and LHD model simulation visual

The vehicle was required to manoeuvre itself through this winding tunnel via its right light ray detection system without touching the wall at any stage. When the simulation was run for the automated testing the following result was obtained.

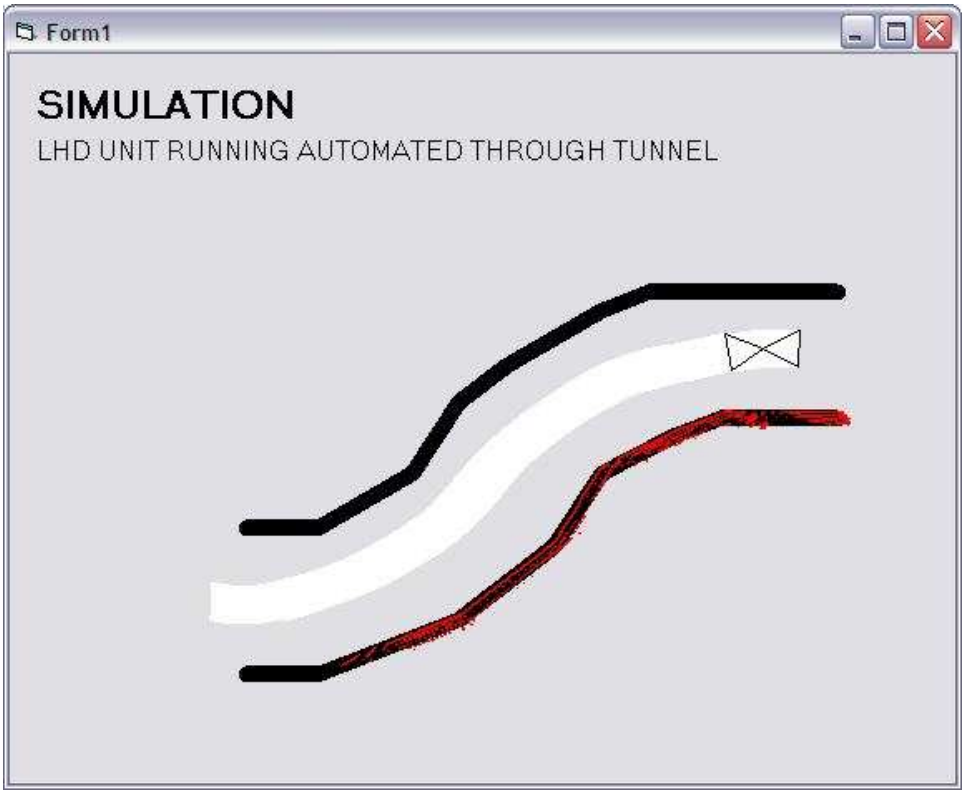

Figure 6.2: Final simulated results

This final simulation shows the LHD unit arriving at the other end of the tunnel and as it can be seen by the white trailed line of where the vehicle had travelled, there were no wall collision and it stayed a reasonable distance away from the tunnel walls for the whole trip. The simulations also showed, in red, every point the right light ray detected that wall.

6.5 Conclusion

The simulation proved that the mathematical models derived for manoeuvring the LHD unit, as well as obstacle detection and avoidance were all correct as the vehicle acted in a manner that was predicted. This proved that the control system was ready to be implemented onto the actual LHD model. The next

chapter will now look at programming the communication programs for the implementation of control strategies.

Chapter 7

7 LHD Communication Program

7.1 Introduction

The communication program is one of the major parts of this project and for each of the control strategies being implemented to the LHD unit, a separate yet similar communication program is used. This chapter will look at the programs objectives and a detailed look at how they were constructed and programmed.

7.2 Program Objectives

The purpose of the monitor program for both the tele-operation and automated system is to basically communicate with the microcontroller to send specific tasks that need to be done by the LHD unit.

7.2.1 Tele-operational Objectives

The following is a list of the objectives that the tele-operation monitor program is required to undertake to operate successfully:

- Start the microcontroller application program running
- Receive and display the image from the camera
- Wait for an operators input for specific task
- Send appropriate information to microcontroller to achieve this task

7.2.2 Automated Objectives

The following is a list of the objectives that the automated monitor program is required to undertake to operate successfully:

- Start the microcontroller application program running
- Receive and display the image from the camera
- Analyse image to determine vehicle location within tunnel
- Determine the action to be taken to according to the vehicles location
- Send appropriate information to microcontroller to achieve this task

7.3 The Communication Program

The communication programs for both control strategies were also written in Windows Visual Basic 6.0. In this section the variables used throughout the program, both when it is tele-operational and automated, will be given and also a detailed look at each of the routines and subroutines use.

7.3.1 Basic Program

Both the communication programs for the tele-operational and automated mode are the same basic arrangement with only a few changed being made in some of the subroutines. The basic program and user interface is used for both of these control strategies is discussed in detail below.

7.3.1.1 Components

MSComm

The communication programs communicate to the microcontroller by utilising Visual Basic's MSComm control. This control is added to the form and allows serial data transfer via the computers serial ports. Within the properties of this

component the serial port to be used in the communication is specified as Port 1 and the baud rate is set to 19200 to match the speed of the microcontroller.

Veye

Both of the control strategy programs use a camera to obtain an image and place this image onto the form for the operator to view. This was done using a Veye control. This component is used as the camera interface for grabbing frames of image data and storing it as an array. This array of data is the able to be accessed for processing elsewhere in the program.

7.3.1.2 Variables

The variables used throughout both communication programs are listed below.

Code to be sent for steering input -	setsteer
Code to be sent for speed input -	setspeed
Code to be sent for the RTI to be stopped -	stopint
X-Position on image -	i
Y-Position on image -	j
X-Position of splosh -	xsplosh
Y-Position of splosh -	ysplosh
Width of image -	w
Height of image -	h
Red pixel -	r
Green pixel -	g
Blue pixel -	b
Previous red pixel -	oldr
Image frame number -	fno
Steering correction -	correct

Incoming byte from microcontroller - bb()

These variables were declared within the program using either the *Public* or the *Dim* statement depending on whether or not they need to be used just within one section of the program or more than one.

7.3.2.3 Routines

Form_Load Routine

This routine starts up as soon as the form is loaded on the screen and uses the previously defined variables *bb()* and the three microcontroller conditions; *setsteer*, *setspeed* and *stopint*.

The hex numbers for steering, speed and the stopping of the real time interrupt are set corresponding to the application program setting on the microcontroller, as illustrated below.

```
setsteer = Chr(&H91)
setspeed = Chr(&H90)
stopint = Chr(&H80)
```

Following this the scroll bars that were places on the form for both the steering and the speed are set up with initial starting value, their minimum and maximum range of movement and finally the size of the steps in the scrolling when the user clicks the scroll arrow. For the steering scroll bar the initial value is set to the centre of the bar with movement in the left and right direction, taking only double steps in each direction. The values for these were determined from a trial and error method with the LHD unit to determine which code value needed to be set to the get the steering actuator to stay stationary, retract for a right turn and extend for a left turn.

The steering values determined are shown below:

Stationary -	(centre)	159
Right -	(minimum)	155
Left -	(maximum)	159

For the speed scroll bar, as the LHD was only required to drive in the forward direction the initial position of the scroll bar would also be the same as the minimum value and when it moved to the maximum in a single step it would drive forward.

The velocity values determined are shown below:

Stopped -	(minimum)	115
Forward -	(maximum)	116

These scroll bars were set up as follows:

```
Steer.Min = 155
Steer.Max = 159
Steer.Value = 157
Steer.SmallChange = 2

Velocity.Min = 116
Velocity.Max = 115
Velocity.Value = 115
Velocity.SmallChange = 1
```

The next section of the routine was concerned with preparing the computer for serial communication and then reading in the information coming from the microcontroller and displaying it for the user. This commenced with opening the serial port for communication by setting its condition to *true*, followed by setting the port so that it would only read one byte at a time.
[write about]

Gocamera_Click Routine

This routine is responsible using the array of image data obtained and stored by the Veye frame grabber control and displaying the image on the form and for the automated program, to detect where the walls of the tunnel are. It uses the variables *i* and *j* for each position along the image, *r, g* and *b* for the pixel properties, and *fno* for the number of frames currently shown.

Firstly the cameras picture width and height are determined so that the image can be collected and each pixel byte can be received for that image dimension. The variable for holding these bytes is then declared ready for receiving the image data, this code is shown below.

```
w = Camera.pwidth
h = Camera.pheight

ReDim picbytes(2, w - 1, h - 1) As Byte
```

The routine then enters a loop which it continues to go through until the program is stopped. The purpose of this loop is to get the image from the camera and place it on the form

```
Do
  framen.Text = fno
  fno = (fno + 1) And 1023

  Camera.OnTop
  Camera.SnapToArray picbytes()
  DoEvents

For j = 1 To h - 1
       For i = 1 To w - 1
          r = picbytes(2, i, j)
          g = picbytes(1, i, j)
          b = picbytes(0, i, j)

          pic.PSet (i, h - j), RGB(r, g, b)
        Next
        DoEvents
    Next
```

```
Loop Until running = False
```

This is done by stepping through each x and y position of image from the camera and recording that points red, green and blue pixel number for that location. This is done, as shown below, by using two *For* command for each x, y direction, to step from the first position of the image, which is the top right, to final position, the bottom left corner. This was done from knowing the width and height of the picture and hence recognising that the bottom left corner would be one minus these values.

```
For j = 1 To h - 1
       For i = 1 To w - 1
          r = picbytes(2, i, j)
          g = picbytes(1, i, j)
          b = picbytes(0, i, j)
```

The image is then painted onto the picture box of the form by using the *RGB* command that returns a single value for the red, green and blue pixel bytes. This then goes across the picture box of the form in the same manner the above section did and paints each point the *RGB* value.

```
pic.PSet (i, h - j), RGB(r, g, b)
```

This is all done while counting the number of frames that have been currently displayed on the form. The loop is exited when the stop button is pressed where it then shuts down the camera and the routine is finished.

Go8000_Click Routine

This routine was for starting the actual microcontroller application program. It corresponded to a button that was placed on the form so that when pressed, it would jump to the user program located at $8000. The button meant that the user did not have to physically type in the address of the program, instead this

routine sent this “G 8000” and the carriage return out through the serial port, as shown below.

```
Private Sub go8000_Click()
ser.Output = "G 8000" + vbCr
End Sub
```

Endit_Click

This routine is used to operate the stop button on the form, when clicked on the Boolean expression for running is set to false which stops the camera routine from looping and then exits the program altogether.

```
Private Sub endit_Click()
running = False
End
End Sub
```

Velocity_Change

When the velocity scroll bar of the form is changed this routine runs, shown below. It opens up the serial port for sending information and then sends the velocity command to let the application program know that the following number will be a velocity value. The value that the scroll bar was moved to then corresponds to a value in which the vehicle will steer.

```
  If ser.PortOpen = True Then
    ser.Output = setspeed + Chr(Velocity.Value)
  End If
End If
```

Ad0_Click and Ad1_Click

These two routines are both for the two buttons for reading into the computer the current status of the LHD units analogue to digital converters. The first one,

AD0 is the current steering angle of the unit, where the AD1 is the current velocity it is travelling at. These routines simply send the appropriate commands set by the application program out via the serial connection, shown below.

```
Private Sub ad0_Click()
ser.Output = Chr(&HA0)
End Sub

Private Sub ad1_Click()
ser.Output = Chr(&HA1)
End Sub
```

7.3.2.4 User Interface

The form of the program is where all the controls and the visual aids of the program are put, this makes up the user interface. The basic communication program has the following basic form, figure 7.1. From this basic form the communication programs for each control strategy could be built on top of this.

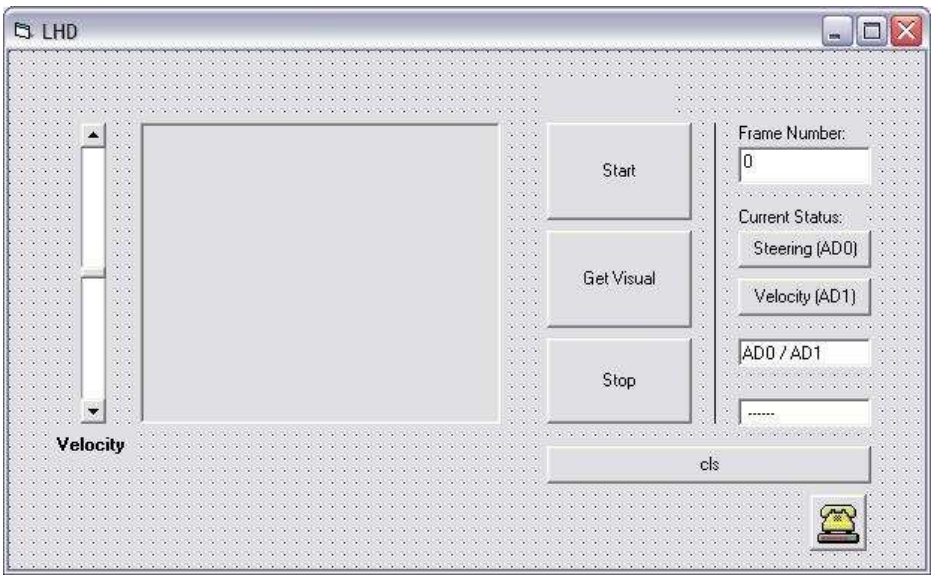

Figure 7.1: Basic form layout

The main components on the user interface consists of the picture box where the camera image is displayed, the simple velocity scroll bar for setting the speed of the vehicle and several buttons and text boxes over the right hand side of the screen. The three most used and important buttons were set large and

close to the image display, these include the program ‘Start’ button, the ‘Get Visual’ button, and the ‘Stop’ button.

The smaller buttons are less important to the user and are used mainly for checking the status of the vehicle. This includes the display of the current frame number from the camera, buttons for obtaining the analogue and digital current values for steering angle and velocity, with corresponding text box for displaying these results.

7.3.2 Tele-operational Program

7.3.2.1 Variations to Routines

Steer_Change

This is an extra routine that is added to the basic program to obtain an input from a user into the system. It acts identical to the *Velocity_change* routine but sends a steering command instead so that the application program knows that the next byte is for the change in steering.

7.3.3 Automated Program

7.3.3.1 Variations to Routines

Form_Load Routine

Since this program completely automates the steering operation for the vehicle the set up for the steering scroll bar can be removed. The velocity scroll bar was left there so that the operator can still tell the LHD to go or stop.

Gocamera_Click Routine

The changes made to the basic camera routine were made to get the program to examine the incoming red pixel bytes and determine where the light ray was shinning on the wall of the tunnel. The location of this splosh would be the lightest spot within the image, which corresponds to the largest pixel value. This involved the introduction of three new variables, *xsplosh* and *ysplosh* used for the location of the light ray on the right wall, and *oldr*, which is the red pixel value of the previous lightest image location.

Within the main loop of this routine an *If* statement was used to test each pixel byte with the value of the previous byte. Only when the new value was greater than the old value would it enter the statement, shown below.

```
If r > oldr Then
  xsplosh = i
  ysplosh = j
  oldr = r
End If
```

The x and y coordinate of this brightest pixel was saved into the variable location *xsplosh* and *ysplosh*, respectively, and the current red byte is then saved as the old value for the next loop.

For each frame taken the y coordinate of the light splosh is compared with two extreme values. If it lies outside of these two extremes it means the steering must be corrected or a collision with the wall will occur. These two extremes are illustrated below.

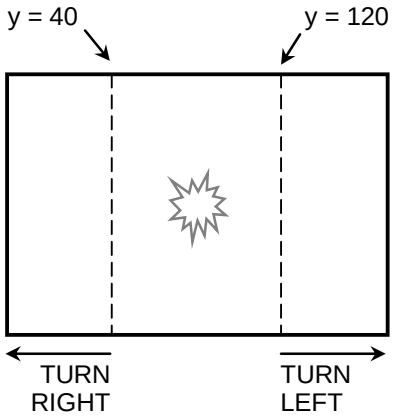

Figure 7.2: Screen extremes with corrections

As figure 7.2 illustrates, when the splosh of light is located before the y position of 40 a steering correction must be made to get the vehicle to turn right, similarly if it is positioned past 120 it must turn left to get the slosh back near the centre of the screen.

This was done by determining the situation, if it was outside of these values the appropriate steering code was loaded, this was then sent out through the serial port for a short period of time and then the steering stopped. This code is shown below:

```
If ysplosh > 120 Then
  correct = 155
End If
If ysplosh < 40 Then
  correct = 159
End If

ser.Output = setsteer + Chr(correct)
pause 2
ser.Output = setsteer + Chr(157)
```

Pause Subroutine

This is a subroutine that is added just to the automated program and runs off the camera routine. It is used to create a pause in time so that the steering of

the unit would be only for a short period of time before straightening up. This routine is given below.

```
Sub pause(a As Single)
Dim t As Single
t = Timer + a
Do
DoEvents
Loop Until Timer > a
End Sub
```

7.4 Conclusion

This chapter looked in detail at the two communication programs for the tele-operated and automated control strategies. These programs are then tested and their results are then evaluated in the following chapter.

Chapter 8

8 Testing and Evaluation

8.1 Introduction

This chapter will discuss the final testing of the LHD unit when it is operating in both the tele-operated and automated mode. It will look at the requirements of the robot for the project and compare them with the actual results achieved.

8.2 Tele-operated Mode

8.2.1 Testing Objectives

For the testing of the tele-operational mode, four are three main requirements which must be met from the time the program is started to when it is stopped. These objectives are listed below:

- Display the user interface
- Display image from the onboard camera
- Wait for user to input either velocity or steering
- Respond accordingly

Testing of the LHD unit was undertaken within a hallway outside the laboratory, where there was plenty of room to manoeuvre as well as obstacles to try and avoid.

8.2.2 Results Achieved

Testing involved running the tele-operational communication program and driving the LHD unit, figure 8.1, up and down the hallway avoiding obstacles and utilising all the controls on the user interface, figure 8.2.

Hence the LHD unit was about to demonstrate that it was capable of meeting all the, above, objectives specified for the system.

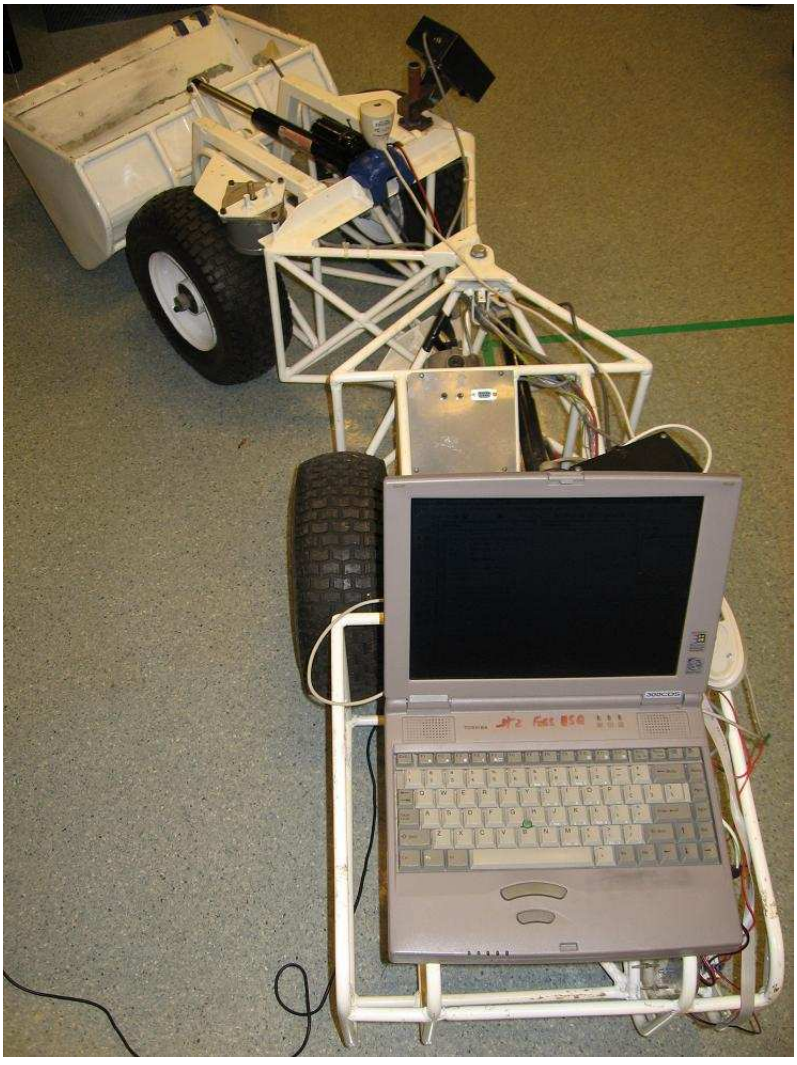

Figure 8.1: LHD unit set up in tele-operational mode

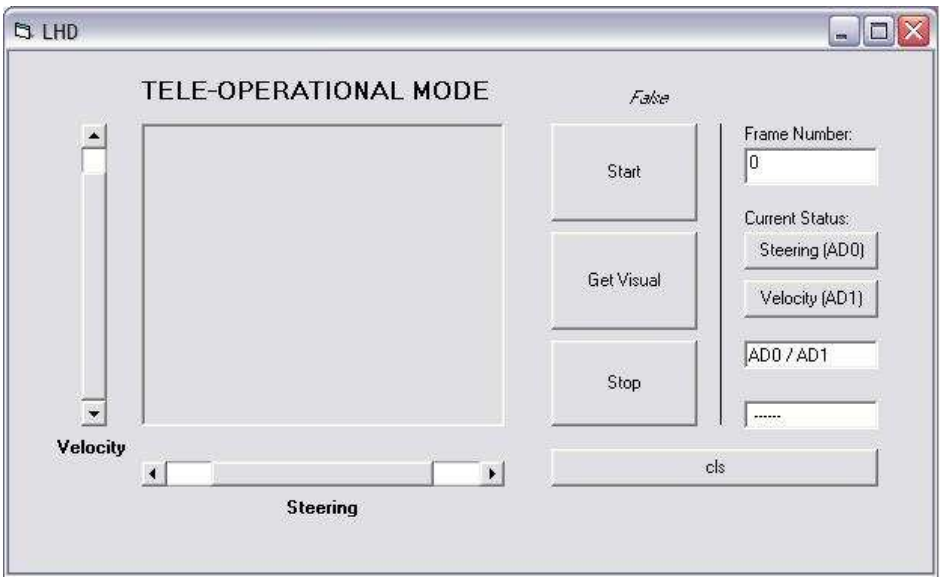

Figure 8.2: Tele-operational Interface

8.2.3 Possible Improvements

After the testing phase the major improvements that could be improved to the system was the acquiring and displaying of image frames, as the current program reads in each frame slower that what the user would prefer. Although the program works a system with faster frame grabbing would make the system more reliable, improving the accuracy and inturn making the system a lot safer.

8.3 Automated Mode

8.3.1 Testing Objectives

From the time the program is started to the time it is switched off, the automated method must be able to satisfy the following objectives:

- Display user interface
- Display image from the onboard camera
- Wait for user input of velocity and respond accordingly
- Detect where the right hand wall is and determine what action to take

The testing of the automated communication program was also undertaken within a hallway outside the laboratory where it was not bright enough to interrupt with the detection of the right wall. It then let the LHD unit out into the run up close to the right hand side of the wall on its own to test the control systems.

8.3.2 Results Achieved

The automated mode, shown running in figure 8.2, was put in the hallway with the light point from the light ray initially being in the desired, centre of the screen, as seen in figure 8.3.

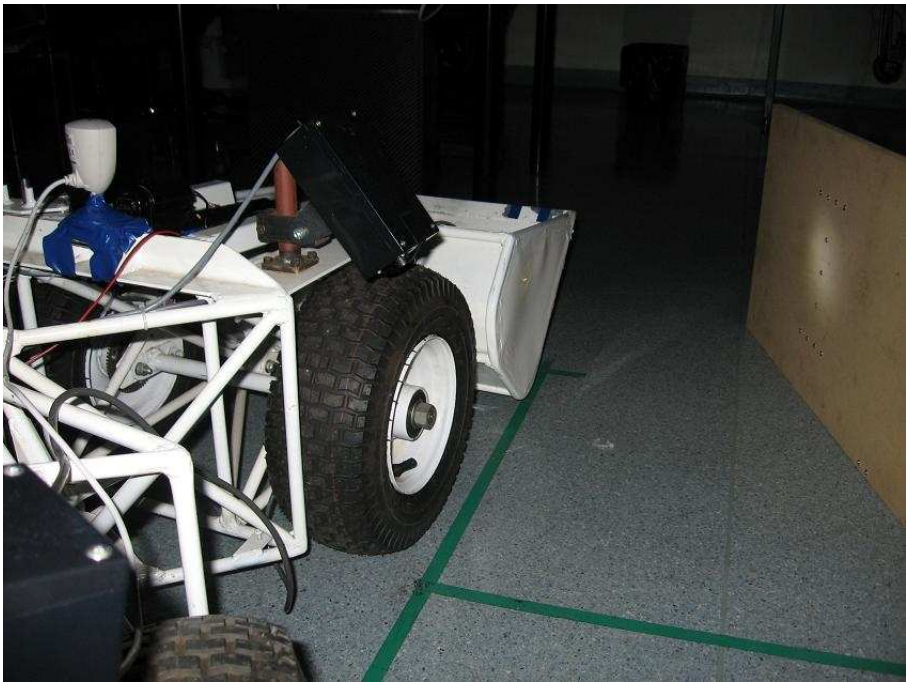

Figure 8.2: LHD Operating in automated mode

Figure 8.3: Light spot detection

As the unit is driving along monitoring the right wall it is continually correcting its steering and more often then not ends up in a bouncing type effect getting extremely unstable. It was still safe to say that the automated met the requirements of the system.

8.3.3 Possible Improvements

Similar to the tele-operational system one of the major improvements to the system would be the receiving the frames of image a lot faster so that there is less delay in the system making it more reliable and much safer.

The other improvement that could have been made to this system is the bouncing effect that the current program has happening.

8.4 Conclusion

This chapter looked at the testing of the LHD unit when it was running its tele-operated and automated communication program. From this testing it could be determined whether or not each of the programs met the main objectives of the simulation. The following and final chapter is the overall conclusion for this project.

Chapter 9

9 Conclusion

9.1 Introduction

This chapter will conclude the overall project by summing up and looking at the project achievements and any further work that could be worked on at the completion of this project.

9.2 Project Achievements

The objective of investigating and implementing suitable control strategies for the manoeuvring of a LHD unit, and implementation of these strategies to a model LHD have been achieved in this project.

This was accomplished by following the clear set objectives presented in the Project Specifications, see Appendix A. These included:

- Research the field of remote vehicle operation, including in the mining field today.
- Research other remote control / automatic LHD units being used in the mines. What else can be achieved with automatic / enhanced control.
- Study the existing control circuitry and programming of the model LHD unit. Determine its suitability for this project.

- Research further into the implementing of a remote control and semi-automated control system into the unit.
- Design and implement suitable control strategies for navigation, communication and machine vision into the LHD unit.
- Devise and apply a tele-operational and semi-automated control system for the LHD unit.

9.3 Future Work

It would be ideal for the automated control strategy not to just be guided off a single wall but off both of them. This would mean that the vehicle would be able to easily determine the centre of the tunnel and the control system would be a lot more stable. This would require more research into the mathematical modelling of this system before being about to practically implement it.

9.4 Conclusion

The two most ideal control strategies for a LHD unit, tele-operational and automated, were able to be implemented onto a previously designed LHD robot successfully.

In conclusion, although this exact type of control strategy would be implemented to a real LHD unit, from working on this project and the model, it is possible to understand how it could successfully adapted and implemented to a real machine.

References

Corke, P 2004, ‘Robots for the Real World’, TOPIC, The Quarterly Newsletter of the CSIRO ICT Centre, Issue 6.

Dhillon, B 1991, Robot Reliability and Safelty, Springer-Verlag, New York.

Freidonvich, L unpub. 2005, Modelling for Load-Haul-Dump Vehicles, lecture notes, Umea University.

Hainsworth, D 2001, ‘Teleoperational User Interfaces for Mining Robotics’, Autonomous Robots, vol.11, p.p.19-28.

Hood, D 1998, ‘Sustainable Innovation in Engineering Education and Practice’, Waves of Change.

IEAust – see The Institute of Engineers, Australia.

LHD Automation: DAS the way to go 2003, online article, ferret.com.au, viewed 2 March 2006 <http://www.ferret.com.au/articles/70/0c018670.asp>

Ridley, P & Corke, P 2003, ‘Load Haul Dump Vehicle Kinematics and Control’, Journal of Dynamic Systems, Measurement, and Control, vol.125, p.p.54-59.

Robert, J Duff, E Corke, P Sikka, P Winstanly, G Cunningham, J 2000, ‘Autonomous Control of Underground Mining Vehicles using Reactive Navigation’, in Proceedings of IEEE International Conf. on Robotics and Automation, San Francisco, CA, p.p.3790-3975

Stentz, A 2001, ‘Robotic Technology for Outdoor Industrial Vehicles’, Unmanned Ground Vehicle Technology III, proceedings of The International Society for Optical Engineering, vol.4364, p.p.192-199.

The Institute of Engineers, Australia 2000, ‘Code of Ethics’

Tyson, J n.d., To see or not to see…That is the question, Ergonomics Australia On-Line, viewed 28 February 2006, <http://www.uq.edu.au/eaol/oct97/tyson/tyson.html>

Pack, D Barrett, S 2002, 68HC12 Microcontroller Theory and Application, Prentice Hall, New Jersey

Levin, P 2005, Excellent Dissertations, McGraw-Hill Education, Berkshire

Spong, M Hutchinson, S Vidyasagar, M 2006, Robot Modelling and Control, John Wiley & Sons Inc, America

Petty, M.K., Billingsley J, Tran-Cong T, ‘Autonomous LHD loading’, 4th Annual Conference on Mechatronics and Machine Vision in Practice, (M2VIP '97), 1997.

Appendix A

Project Specification

University of Southern Queensland

Faculty of Engineering and Surveying

ENG 4111/2 Research Project
PROJECT SPECIFICATION

FOR: **Toni Leigh PARTRIDGE**
TOPIC: Automation of a Load-Haul-Dump (LHD) Unit
SUPERVISOR: Prof. John Billingsley

SPONSORSHIP: USQ – Faculty of Engineering & Surveying

PROJECT AIM: The aim of this project is to investigate a suitable control strategy for the mechanical operation of a Load-Haul-Dump Unit, and then implement these strategies to an existing scale model of the LHD.

PROGRAMME: **Issue A, 27th March 2006**

1. Research the field of remote vehicle operation, including in the mining field today.

2. Research other remote control / automatic LHD units being used in the mines. What else can be achieved with automatic / enhanced control.

3. Study the existing control circuitry and programming of the model LHD unit. Determine its suitability for this project.

4. Research further into the implementing of a remote control and semi-automated control system into the unit.

5. Design and implement suitable control strategies for navigation, communication and machine vision into the LHD unit.

6. Devise and apply a tele-operational and semi-automated control system for the LHD unit.

As time permits:

7. Investigate and employ other navigation control strategies into the system (ie. Compass).

8. Investigate and employ other appropriate machine type vision into the system.

AGREED: _____ (Student) _____ (Supervisor)

(dated) ____ / ____ / ____

Appendix B

HC12 Application Program

```
;====================================================================
; File:        IOboard.asm
; Func:        Work as analog input, digital output seriel port
;              and controll board for steering and speed
; Author:      Anders Lööf
; e-mail:      f00anlo@dd.chalmers.se
;====================================================================

             CPU 68HC12
             PADDING OFF

REGBASE equ  $0000

COPCTL  equ  REGBASE+$0016  ; COP Control Register
PORTH   equ  REGBASE+$0029  ; Port H Register
DDRH    equ  REGBASE+$002b  ; Port H Data Direction Register

;TIME INTERUP REGISTER
RTICTL  equ  $0014
RTIFLG  equ  $0015

;SERIEL REGISTER

SC0BDH  equ  $00C0
SC0BDL  equ  $00C1
SC0SR1  equ  $00C4
SC0DRL  equ  $00C7
SC0CR2  equ  $00C3

;OUT PORT REGISTER

DDRG    equ  $002A
PortG   equ  $0028

;PULSE COUNT REGISTERS

PACTL           equ  $00A0  ;16 bit Pulse Accumulator A Control Register
PAFLG           equ  $00A1  ;Pulse Accumulator A Flag Register
PACAHI          equ  $00A2  ;Pulse Accumulator count registers hi
PACALO          equ  $00A3  ;and lo

;ADC REGISTER

ATD0CTL2        equ  $0062
ATD0CTL3        equ  $0063
ATD0CTL4        equ  $0064
ATD0CTL5        equ  $0065
ATD0STAT0       equ  $0066
ATD0STAT1       equ  $0067
ADR00H          equ  $0070
ADR01H          equ  $0072


RAMTOP  equ  $05FF
CODE    equ  $8000
RAM     equ  $0200

;variables
             org  RAM
AD0          RMB  1             ;variable for ADC0 (velocity)
AD1          RMB  1             ;variable for ADC1 (steering)
```

```
TC              RMB     1               ;variable for time Counter
DESV            RMB     1               ;desired Velocity
DESS            RMB     1               ;desired Steering
SETVEL          RMB     1               ;variable for knowing if next incomeing byte is a
parameter velocity or command
SETSTE          RMB     1               ;variable for knowing if next incomeing byte is a
parameter steering or command
XPOS            RMB     4               ;x coordinate
YPOS            RMB     4               ;y coordinate
HEADING         RMB     2               ;heading
OLDPULSE        RMB     2               ;old Number of pulses
NEWPULSE        RMB     2               ;new Number of pulses
PULSEDIF        RMB     2               ;pulse differance

THETA           RMB     1               ;steering angle

                ORG     CODE
main            lds     #RAMTOP
                clr     COPCTL
                sei                     ;disables interrupts
                jsr     initSC0         ;setup serial comunication
                jsr     initclk         ;setup RTI
                jsr     initintr        ;setup interupt
                jsr     initadc         ;setup adc
                movb    #$FF,DDRG       ;set portG as output
                movb    #$50,PACTL      ;setup pulse accumulator
                cli                     ;enables interrupts

                movb    #$80,DESV       ;set desired velocity to 128
                movb    #$80,DESS       ;set desired steering to 128

loop            nop                     ;main loop to not lose track of PC
                bra     loop


SerialInt       ldaa    SC0SR1          ;load status register 1
                anda    #$20            ;mask out RDRF
                lbeq    endSerialInt    ;if RDRF not set go to end
                ldaa    SC0DRL          ;load byte recived

                ldab    SETVEL          ;load staus variable set velocity
                cmpb    #$01            ;if set vel =1 the incomening byte should be
loaded to desired velocity variable
                bne     stepOversetvel ;
                staa    DESV            ;store desired vel
                movb    #$00,SETVEL     ;clear set vel variable
                jmp     endSerialInt    ;jump to end of interupt handler

stepOversetvel ldab     SETSTE          ;load status variable for set steering
                cmpb    #$01            ;if set store incoming byte as desired steering
                bne     stepOversetste ;
                staa    DESS            ;storing recived byte
                movb    #$00,SETSTE     ;clear set steering variable
                jmp     endSerialInt    ;jump to end of interupt handler

stepOversetste cmpa     #$a0            ;compare with $a0 comand for retrive AD0
                bne     stepOvera0      ;if not equal step over
                jsr     getad0          ;call getad0
                jmp     endSerialInt    ;jump to end ov interupt handler

stepOvera0      cmpa    #$a1            ;retrive AD1
```

```
                bne     stepOvera1
                jsr     getad1
                jmp     endSerialInt

stepOvera1      cmpa    #$11            ;set led 1 high
                bne     stepOver11
                jsr     set1h
                jmp     endSerialInt

stepOver11      cmpa    #$10            ;set led 1 low
                bne     stepOver10
                jsr     set1l
                jmp     endSerialInt

stepOver10      cmpa    #$31            ;set led 3 high
                bne     stepOver31
                jsr     set3h
                jmp     endSerialInt

stepOver31      cmpa    #$30            ;set led 3 low
                bne     stepOver30
                jsr     set3l
                jmp     endSerialInt

stepOver30      cmpa    #$51            ;set led 5 high
                bne     stepOver51
                jsr     set5h
                jmp     endSerialInt

stepOver51      cmpa    #$50            ;set led 5 low
                bne     stepOver50
                jsr     set5l
                jmp     endSerialInt

stepOver50      cmpa    #$71            ;set led 7 high
                bne     stepOver71
                jsr     set7h
                jmp     endSerialInt

stepOver71      cmpa    #$70            ;set led 7 low
                bne     stepOver70
                jsr     set7l
                jmp     endSerialInt

stepOver70      cmpa    #$81            ;command for start RTIintrupt
                bne     stepOver81
                jsr     timeron
                jmp     endSerialInt

stepOver81      cmpa    #$80            ;command for stop RTIintrupt
                bne     stepOver80
                jsr     timeroff
                jmp     endSerialInt

stepOver80      cmpa    #$90            ;command for set Set velocity active
                bne     stepOver90
                movb    #$01, SETVEL    ;set status variable high
                jmp     endSerialInt

stepOver90      cmpa    #$91            ;command for set Set steering active
                bne     stepOver91
```

```
                movb    #$01, SETSTE   ;set status variable high
                jmp     endSerialInt

stepOver91      ldaa    #$45           ;An error has ocured send error message
                bsr     senda
                ldaa    #$52
                bsr     senda

endSerialInt    rti

readad          brclr   ATD0STAT0,$80,readad  ;loop until adconvert complet
                movb    ADR00H,AD0            ;move adresult into variable AD0
                movb    ADR01H,AD1            ;move adresult into variable AD1
                rti


senda           nop
                ldab    SC0SR1  ;load status register 1
                andb    #$80    ;mask out TDRE
                beq     senda   ;wait until TDRE is set
                staa    SC0DRL  ;send register a via serial
                rts

getad0          ldaa    #$a0
                bsr     senda
                ldaa    AD0     ;load adresult from variable
                bsr     senda   ;branch to send register a
                rts

getad1          ldaa    #$a1
                bsr     senda
                ldaa    AD1     ;load adresult from variable
                bsr     senda   ;branch to send register a
                rts

set1h           ldab    PortG   ;load PortG status
                orab    #$02    ;set 2:nd bit high
                stab    PortG   ;store result
                ldaa    #$02    ;set reply message
                ;bsr    senda   ;send replymessage
                rts


set1l           ldab    PortG   ;load PortG status
                andb    #$FD    ;set 2:nd bit low
                stab    PortG   ;store result
                ldaa    #$FD    ;set reply message
                ;bsr    senda   ;send replymessage
                rts

set3h           ldab    PortG   ;load PortG status
                orab    #$08    ;set 4:th bit high
                stab    PortG   ;store result
                ldaa    #$08    ;set reply message
                ;bsr    senda   ;send replymessage
                rts

set3l           ldab    PortG   ;load PortG status
                andb    #$F7    ;set 4:th bit low
                stab    PortG   ;store result
                ldaa    #$F7    ;set reply message
```

```
              ;bsr    senda   ;send replymessage
              rts

set5h         ldab    PortG   ;load PortG status
              orab    #$20    ;set 6:th bit high
              stab    PortG   ;store result
              ldaa    #$20    ;set reply message
              ;bsr    senda   ;send replymessage
              rts

set5l         ldab    PortG   ;load PortG status
              andb    #$DF    ;set 6:th bit low
              stab    PortG   ;store result
              ldaa    #$DF    ;set reply message
              ;jsr    senda   ;send replymessage
              ;jsr    senda   ;send replymessage
              rts


set7h         ldab    PortG   ;load PortG status
              orab    #$80    ;set 8:th bit high
              stab    PortG   ;store result
              ldaa    #$80    ;set reply message
              ;jsr    senda   ;send replymessage
              ;jsr    senda   ;send replymessage
              rts

set7l         ldab    PortG   ;load PortG status
              andb    #$7F    ;set 8:th bit low
              stab    PortG   ;store result
              ldaa    #$7F    ;set reply message
              ;jsr    senda   ;send replymessage
              ;jsr    senda   ;send replymessage
              rts

timeron
              movb    #$84,RTICTL    ;enable RTIinterupt
              ldaa    #$05
              jsr     senda
              jsr     senda
              rts

timeroff
              movb    #$04,RTICTL    ;disable RTIinterupt
              ldaa    #$06
              jsr     senda
              jsr     senda
              rts

RTIint        bset    RTIFLG,#$80           ;Acknowledge interrupt
              inc     TC                    ;increas time counter by 1
              jsr     CalcPulseDif          ;send distance
              jsr     CalcNewPos
              jsr     ADJVel                ;adjust velocity
              jsr     ADJSte                ;adjust steering
RTIend        rti

CalcNewPos
              rts

CalcPulseDif
```

```
                ldaa    PACAHI                  ;load pulse accumulator Hi
                ldab    PACALO                  ;load pulse accumulator       Lo
                std     NEWPULSE                ;store it as new pulsecount
                subd    OLDPULSE                ;subtract old pulse count
                std     PULSEDIF                ;store it as pulse diff
                ldd     NEWPULSE                ;move new pulse count to
                std     OLDPULSE                ;old
                rts

ADJSte                                          ;adjust steering
                ldaa    DESS                    ;load desired steering angle
                cmpa    AD1                     ;compare with actual steering angle
                beq     nosteering              ;if equal bransh to no steering
                bls     steerleft               ;if dif neg bransh to steer left
                jsr     set1h                   ;set out 1 high
                jsr     set3l                   ;set out 3 low

                bra     endsteering             ;jump to end
steerleft       jsr     set1l                   ;steer left set out 1 low
                jsr     set3h                   ;set out 3 high

                bra     endsteering             ;jump to end
nosteering      jsr     set1l                   ;set out 1 low
                jsr     set3l                   ;set out 3 low
endsteering
                rts                             ;return from sub

ADJVel                                          ;adjust velosity
                ldaa    DESV                    ;load desired velocity
                cmpa    #$80                    ;compare to zero vel
                beq     noaccel                 ;if 0 vel des bransh to no acceleration
                bgt     posdesvel               ;if desired > 0 bransh to positive desired
velocity
                cmpa    AD0                     ;compare desired vel with actual in case
des vel <0
                bgt     noaccel                 ;if desierd vel > actual vel and desired
vel <0 turn off motors
                jsr     set5l                   ;set out 5 low
                jsr     set7h                   ;set out 7 high for reverse
                bra     endvelocity             ;jump to end
posdesvel       cmpa    AD0                     ;compare desired velocity with actual
                blt     noaccel                 ;if desired vel< actual vel and popsitive
desired velocity turn motors of
                jsr     set5h                   ;set out 5 high for acceleration
                jsr     set7l                   ;set out 7 low
                bra     endvelocity             ;jump to end
noaccel         jsr     set5l                   ;set out 5 low
                jsr     set7l                   ;set out 7 low
endvelocity     rts


;Initsialising procedures

initclk         movb    #$84,RTICTL     ;set interupt time to 8.196ms
                movb    #$00,TC         ;reset time counter
                rts

initintr        ;RTIintr
                ldaa    #$06            ;load jump comand
                staa    $07EB           ;store at RTI interup adress
                ldd     #RTIint         ;load interupt handler adress
```

```
                std     $07EC           ;store at RTI interup adress
                ;Seriel event
                ldaa    #$06            ;load jump comand
                staa    $07C4           ;store at SCI0 interup adress
                ldd     #SerialInt      ;load interupt handler adress
                std     $07C5           ;store at SCI0 interup adress
                ;ADconversion compleated
                ldaa    #$06            ;load jump comand
                staa    $07BE           ;store at ADC interup adress
                ldd     #readad         ;load interupt handler adress
                std     $07BF           ;store at ADC interupt adress
                rts

initadc         movb    #$82,ATD0CTL2   ;enable adc
                movb    #$00,ATD0CTL3   ;one conversion each sequence
                movb    #$61,ATD0CTL4   ;8 bit accuarcy
                movb    #$30,ATD0CTL5   ;scan adconverter
                rts

initSC0         ldaa    #$2C            ;add flag for interupt enabled
                staa    SC0CR2          ;enables reciver interupt
                clr     SC0BDH          ;clear controllregister should be clear anyway
                movb    #$1A,SC0BDL     ;set Baud Rate 38400
                rts
```

Appendix C

Simulation Program

Tele-Operational

```
Public setspeed As String
Public setsteer As String
Public stopint As String
Public running As Boolean

Private Sub ad0_Click()                     'read current AD0 - steering angle
ser.Output = Chr(&HA0)
End Sub

Private Sub ad1_Click()                     'read current AD1 - velocity
ser.Output = Chr(&HA1)
End Sub

Private Sub cls_Click()                     'clear screen button
cls
End Sub

Private Sub endit_Click()                   'stop program button
running = False
End
End Sub

Private Sub Form_KeyPress(KeyAscii As Integer)
inkey$ = Chr$(KeyAscii)                     'gets set everytime key is hit
Print inkey$;
End Sub

Private Sub Form_Load()
Dim bb() As Byte

 setsteer = Chr(&H91)                       'steering hex code
'hex numbers to be sent for:
 setspeed = Chr(&H90)                       'velocity hex code
 stopint = Chr(&H80)                        'stop RTI hex code

Show
cls                                         'clears screen
Camera.OnTop                                'put camera frame ontop of form
runn.Caption = running

Steer.Min = 155                             'set up steer scroll bar
Steer.Max = 159
Steer.Value = 157
Steer.SmallChange = 2

Velocity.Min = 116
Velocity.Max = 115
Velocity.Value = 115
Velocity.SmallChange = 1

ser.PortOpen = True                         'opens port for communication
ser.InputLen = 1                            'input length - one byte at a time

Do
  DoEvents

  If ser.InBufferCount > 0 Then             'length of input is more than one character
     bb() = ser.Input                       'byte returned from serial connection?
```

```
      char2.Text = char1.Text
      char1.Text = Hex$(bb(0))

      a$ = Chr$(bb(0))                      'zero-th element - only one we are concerned
with
      If g < 2 Then                         'g=1 for decimal mode, g=2 for hex mode
         Print a$;                          'decimal - print
      Else
         n = n + 1                          'so code doesnt go off screen
         If n > 10 Then                     'only go 10 characters across
            Print vbCr
            n = 0
         End If
         Print Hex$(bb(0)); " ";            'hex - print (followed by space)
      End If
  End If

Loop                                        'quit program


End Sub

Private Sub go8000_Click()                  'run HC12 program
ser.Output = "G 8000" + vbCr
End Sub

Private Sub gocamera_Click()
Dim i As Integer, j As Integer, r As Integer, g As Integer, b As Integer
Dim fno As Integer, oldr As Integer, xsplosh As Integer, ysplosh As Integer

gocamera.Enabled = False                    'can not use button twice

w = Camera.pwidth                           'image width
h = Camera.pheight                          'image height

ReDim picbytes(2, w - 1, h - 1) As Byte     'ready to recieve image data

running = True                              'program running

Do
  framen.Text = fno                         'frame number displayed
  fno = (fno + 1) And 1023                  'count up for each frame

  Camera.OnTop                              'put camera frame ontop of form
  Camera.SnapToArray picbytes()             'fills array with the image data

  DoEvents
    For j = 1 To h - 1                      'move up? screen for each pixel
       For i = 1 To w - 1                   'move across screen for each pixel
          r = picbytes(2, i, j)             'red bytes
          g = picbytes(1, i, j)             'green bytes
          b = picbytes(0, i, j)             'blue bytes

          pic.PSet (i, h - j), RGB(r, g, b) 'colour that point
       Next
       DoEvents
    Next

Loop Until running = False                  'continue till stop button pressed
Camera.shut                                 'close OCX
```

```
End

End Sub

Private Sub Steer_Change()                  'steer input
Dim DESS As Single

If Steer.Value <> 158 Then
  If ser.PortOpen = True Then
    ser.Output = setsteer + Chr(Steer.Value)  'code to send to HC12 - activate code for
steering & scroll bar steer angle input
  End If
End If

Debug.Print Steer.Value

End Sub

Private Sub stoprti_Click()
ser.Output = Chr(&H80)
End Sub

Private Sub Velocity_Change()               'velocity input
Dim DESV As Single

If ser.PortOpen = True Then
  ser.Output = setspeed + Chr(Velocity.Value) 'code to send to HC12 - activate code for
velocity & scroll bar velocity input
End If

'Debug.Print Velocity.Value

End Sub
```

Appendix D

Communication Program

Automated

```
Public inkey$                                   'public - can be seen by all parts of
program
Public setspeed As String
Public setsteer As String
Public stopint As String
Public a$
Public running As Boolean

Private Sub ad0_Click()                         'read current AD0 - steering angle
ser.Output = Chr(&HA0)
End Sub

Private Sub ad1_Click()                         'read current AD1 - velocity
ser.Output = Chr(&HA1)
End Sub

Private Sub cls_Click()                         'clear screen button
cls
End Sub

Private Sub endit_Click()                       'stop program button
running = False
End
End Sub

Private Sub Form_KeyPress(KeyAscii As Integer)
inkey$ = Chr$(KeyAscii)                         'gets set everytime key is hit
Print inkey$;
End Sub

Private Sub Form_Load()
Dim bb() As Byte

 setsteer = Chr(&H91)                           'steering hex code
'hex numbers to be sent for:
 setspeed = Chr(&H90)                           'velocity hex code
 stopint = Chr(&H80)                            'stop RTI hex code

Show
cls                                             'clears screen
Camera.OnTop                                    'put camera frame ontop of form
runn.Caption = running

Velocity.Min = 116
Velocity.Max = 115
Velocity.Value = 115
Velocity.SmallChange = 1

ser.PortOpen = True                             'opens port for communication
ser.InputLen = 1                                'input length - one byte at a time

Do
  DoEvents

  If ser.InBufferCount > 0 Then                 'length of input is more than one character
     bb() = ser.Input                           'byte returned from serial connection?

      char2.Text = char1.Text
      char1.Text = Hex$(bb(0))
```

```
      a$ = Chr$(bb(0))                        'zero-th element - only one we are concerned
with
      If g < 2 Then                           'g=1 for decimal mode, g=2 for hex mode
         Print a$;                            'decimal - print
      Else
         n = n + 1                            'so code doesnt go off screen
         If n > 10 Then                       'only go 10 characters across
            Print vbCr
            n = 0
         End If
         Print Hex$(bb(0)); " ";              'hex - print (followed by space)
      End If
  End If

Loop                                          'quit program


End Sub

Private Sub go8000_Click()                    'run HC12 program
ser.Output = "G 8000" + vbCr
End Sub

Private Sub gocamera_Click()
Dim i As Integer, j As Integer, r As Integer, g As Integer, b As Integer
Dim fno As Integer, oldr As Integer, xsplosh As Integer, ysplosh As Integer, correct As
Integer


gocamera.Enabled = False                      'can not use button twice

w = Camera.pwidth                             'image width
h = Camera.pheight                            'image height

ReDim picbytes(2, w - 1, h - 1) As Byte       'ready to recieve image data
running = True                                'program running

Do
  framen.Text = fno                           'frame number displayed
  fno = (fno + 1) And 1023                    'count up for each frame

  Camera.OnTop                                'put camera frame ontop of form
  Camera.SnapToArray picbytes()               'fills array with the image data

  DoEvents
    For j = 1 To h - 1                        'move down screen for each pixel
       For i = 1 To w - 1                     'move across screen for each pixel
          r = picbytes(2, i, j)               'red bytes
          g = picbytes(1, i, j)               'green bytes
          b = picbytes(0, i, j)               'blue bytes

          pic.PSet (i, h - j), RGB(r, g, b) 'colour that point

          If r > oldr Then                    'use red bytes to detect splosh
            xsplosh = i                       'x position of splosh
            ysplosh = j                       'y position of splosh
            oldr = r                          'update oldr for comparison
          End If

        Next
        DoEvents
```

```
    Next

sploshx.Text = xsplosh                      'display results on screen
sploshy.Text = ysplosh

If ysplosh > 120 Then                       'check if outside maximum
  correct = 155                             'turn left
End If
If ysplosh < 40 Then                        'check if outside minimum
  correct = 159                             'turn right
End If

ser.Output = setsteer + Chr(correct)        'steering correction
pause 2
ser.Output = setsteer + Chr(157)            'straighten up

Loop Until running = False                  'continue till stop button pressed
Camera.shut                                 'close OCX
End

End Sub

Private Sub Steer_Change()                  'steer input
Dim DESS As Single

If Steer.Value <> 158 Then
  If ser.PortOpen = True Then
    ser.Output = setsteer + Chr(Steer.Value)  'code to send to HC12 - activate code for
steering & scroll bar steer angle input
  End If
End If

Debug.Print Steer.Value

End Sub

Private Sub stoprti_Click()
ser.Output = Chr(&H80)
End Sub

Private Sub Velocity_Change()               'velocity input
Dim DESV As Single

If ser.PortOpen = True Then
  ser.Output = setspeed + Chr(Velocity.Value) 'code to send to HC12 - activate code for
velocity & scroll bar velocity input
End If

'Debug.Print Velocity.Value

End Sub

Sub pause(a As Single)
Dim t As Single
t = Timer + a
Do
DoEvents
Loop Until Timer > a

End Sub
```