DESIGN OF AN AUTONOMOUS MOBILE ROBOT FOR SERVICE APPLICATIONS

Mark de Villiers
207529450

In fulfilment of the degree of Master of Science in Mechatronic Engineering at the School of Mechanical Engineering, University of KwaZulu-Natal.

March 2011

As the candidate's supervisor I agree to the submission of this thesis.

Supervisor: Prof. Glen Bright

Declaration

I ..... declare that

(i) The research reported in this thesis, except where otherwise indicated, is my original work.

(ii) This thesis has not been submitted for any degree or examination at any other university.

(iii) This thesis does not contain other persons's data, pictures, graphs or other information, unless specifically acknowledged as being sourced from other persons.

(iv) This thesis does not contain other persons's writing, unless specifically acknowledged as being sourced from other researchers. Where other written sources have been quoted, then:

a) their words have been re-written but the general information attributed to them has been referenced; b) where their exact words have been used, their writing has been placed inside quotation marks, and referenced.

(v) Where I have reproduced a publication of which I am an author, co-author or editor, I have indicated in detail which part of the publication was actually written by myself alone and have fully referenced such publications.

(vi) This thesis does not contain text, graphics or tables copied and pasted from the Internet, unless specifically acknowledged, and the source being detailed in the thesis and in the References sections.

Signed:

Mark de Villiers

Acknowledgements

I would like to thank my supervisor Professor Glen Bright for his help with this work.
I would like to thank the CSIR for funding this project and degree.
Thank you to all my colleagues for their help, advice and patience.
Special thanks to Mr Russ Ether for help ironing out some mathematical problems.

Abstract

This research project proposes the development of an autonomous, omnidirectional vehicle that will be used for general indoor service applications. A suggested trial application for this service robot will be to deliver printouts to various network users in their offices. The robot will serve as a technology demonstrator and could later also be used for other tasks in an office, medical or industrial environment.

The robot will use Mecanum wheels (also known as Swedish 45° or Ilon wheels) to achieve omnidirectionality. This will be especially useful in the often cramped target environments, because the vehicle effectively has a zero radius turning circle and is able to change direction of motion without changing its pose. Part of the research will also be to investigate a novel propulsion system based on the Mecanum wheel.

The robot will form part of a portfolio of service robots that the Mechatronics and Micro Manufacturing (MMM) group at the CSIR is busy developing. Service robots are typically used to perform Dull, Dangerous or Dirty work, where human presence is not essential if the robot can perform the task reliably and successfully.

Contents

List of figures

Chapter 1.

Introduction

This chapter will give a brief outline of the project, it begins with the motivation for the study, then discusses the scientific contribution of the project and the objectives that should be achieved. It then discusses the project specifications and finally gives some brief chapter overviews.

1.1. Motivation

Worldwide there is currently a large amount of research into mobile robotics. There are a number of different drivers of mobile robotic research including military, search and rescue, space exploration, underwater exploration and service robotics. This project will focus on the field of service robots.

Service robotics itself is a very wide topic and it is seen as a large untapped global market [47]. Service robotics applications range from bar tending , to home vacuum cleaning, to hospital patient transportation [**?**], [29], [48]. The application of service robots specifically in the office environment has not been widely studied in the literature and this project will expand on knowledge in this area.

Service robots are typically used to perform dull, dangerous or dirty work, where human presence is not needed when the robot is performing the task successfully and reliably.

For example, the activity performed by office workers making paper printouts is a task that fits this description. Due to the cost of printers and the pressures of the economic climate many companies use network printers which are located in a central location.

Office workers must send documents to this printer and then leave their desk to go and fetch their printouts. Over a large number of workers this can result in a significant waste of man-hours. If this platform could be used to provide a service to network users that allows delivery of printouts to user's desks, this would save the time spent fetching these printouts.

If this system can be implemented as a low cost platform, it could be put to use not only delivering printouts, but any items that require regular, scheduled or unscheduled distribution. This would allow such a system also to be used on a production line for materials handling or in a hospital for patient file handling.

1.2. Scientific Contribution

The scientific contribution of this project will be to expand the knowledge of office automation. Where items like the personal computer, print/fax/scanner and network connectivity have automated the office environment to a certain extent, there is still work to be done to improve these skilled workers' efficiency. This project will attempt to provide further equipment to allow this improvement to be realised.

To help a robot move as easily as possible within this often unstructured and sometimes dynamic environment, omnidirectional wheels were chosen as a means of locomotion. While the motion control of Mecanum wheel vehicles has been quite well studied, this project will provide a new algorithm for their control.

1.3. Objectives

The main objectives of the project can be divided up as follows:

- Research, design and test a vehicle and its locomotion system such that it will be able to move autonomously under its own power from any starting pose (position and heading) to any other pose, demonstrating its omnidirectionality allowing it to used for office service applications
- Research, design and test a control algorithm for controlling Mecanum wheels

- Research, design and test a sensor system to allow the mobile robot to successfully avoid obstacles
- Research, design and test a suitable navigation technique to allow the robot to navigate according to a map of its environment

1.4. Project Specifications

1.4.1. Mechanical Specifications

Physical Dimensions

The developed robot will be large enough to mount an average desktop printer. It will have enough space to mount batteries that can power this printer and any other necessary hardware. However the robot should be small enough to manoeuvre around relatively cramped offices, hallways, and ultimately fit through doorways. This results in the robot needing to have a width significantly smaller than 813 mm, which is a standard door width at the CSIR offices. As a result the platform base is 358 mm wide without its wheels and bumpers. Once the bumpers have been added the width of the platform is 650 mm, which will only allow the platform 81.5 mm clearance either side to navigate through a door. This will require very good positioning as the robot moves through the door. The length of the robot does not matter as much as the wheels allow it to turn on its own centre to align it with a door.

1.4.2. Motion Specifications

Positioning accuracy

Due the preceding specification the required positioning accuracy must be better than 80mm for the vehicle to navigate a doorway. This will be a challenging specification. However the vehicle will implement obstacle avoidance which will help it navigate smaller openings.

Robot manoeuvrability

This robot will be used in a cramped and unstructured office environment, requiring it to be very manoeuvrable. For this reason omnidirectional, Mecanum, wheels were implemented on the robot. Omnidirectional is defined as "being in or involving all directions", so omnidirectional motion is motion in all directions [5]. This project will demonstrate omnidirectionality using Mecanum wheels. Better understanding how Mecanum wheels work will be aided by viewing the simulation videos on the attached CD [21]. The reader is strongly encouraged to view these videos.

Robot speed

Because the robot is to be used in the office environment it cannot move very fast as this could endanger office workers. As such it should not move faster than the average human walking speed which is about 1.3m/sec [18]

1.4.3. Capacity specifications

Robot battery life

The mobile robot should have enough power capacity for it to operate for an hour without returning to charge. Ideally it would not require recharging for an entire working day, this is not possible with the current design.

Payload capacity

The robot should be able to carry all the hardware required for it to perform its specified task. In this case a printer loaded with paper, which could weigh up to 15 kg. This is not seen as a major problem as the preceding specification will already require the robot to carry heavy batteries.

1.5. Research Publication

The following conference paper was published with respect to the research conducted in this project:

25th International Conference of CAD/CAM, Robotics & Factories of the Future, 13-16 July 2010, "Development of a control model for a four wheel Mecanum vehicle", M. de Villiers G. Bright.

1.6. Brief Chapter Overviews

Chapter 1 introduces the project and outlines the topic discussed in the dissertation. It gives details of the project objectives and specifications.
Chapter 2 gives an outline of literature related to the project topic.
Chapter 3 deals with the mechanical designs made and the modelling of the Mecanum wheel.
Chapter 4 discusses the selection and design of electrical and electronic components used in this project.
Chapter 5 discusses the design and implementation of various software systems including embedded software, PC software and the Player Project.
Chapter 6 discusses the experiments completed to test the work conducted in this project, including wheel tests, motion tests and obstacle avoidance tests.
Chapter 7 concludes the dissertation with a discussion of the findings and a summary of the dissertation.

1.7. Chapter Summary

This chapter has given a brief introduction to the project. It has outlined the motivation for doing the work, which is that there appears to be a gap in the literature pertaining to office service robotic applications. This project aims to make a scientific contribution within this branch of mobile robotics. The chapter went on to briefly explain the specifications of the project and finally gave an overview of the content of the dissertation.

Chapter 2.

Mobile Robots

This chapter will present the work that has been completed by other scholars in fields that are relevant to this project. Mobile robotics spans a broad range of topics, this review will briefly mention some other directions in which mobile robotics is being developed. It will then describe some of the problems that are common to all types of mobile robots and then discuss locomotion and control methods of robots similar to the robot being developed. Then the related methods of perception, localisation and navigation will be explored. Finally the software used for implementation of the above techniques will be discussed.

2.1. Introduction

The word robot is derived from the Czech word *robota* meaning forced labour, it was coined in K. Čapek's play R.U.R., however it has come to mean "a machine capable of carrying out a complex series of actions automatically, especially one programmable by a computer" [59], [2]. There are many systems that fit this description and much research is going into them, from robotic fish to micro robotic flies [42], [37].

There are a large number of institutions carrying out robotics research and it would be impossible to mention the focus topics of all of them. To illustrate these numbers, Vaughan and Gerky provide a map showing numbers of downloads of the Player Project robot server, a popular robotics research tool, which occurred in December 2005 alone [61]. A good overview of the possible research areas is given by Nguyen and Everett discussing robots developed by the Joint Robotics Program in the United States [46]. This research, aimed at military applications, include man portable robots, unmanned (water) surface vehicles,

indoor SLAM (Simultaneous Localisation And Mapping), rugged terrain Unmanned Ground Vehicles and a Robotic Service Pool including many commercial and custom robotic systems. UAVs (Unmanned Aerial Vehicles) is another field of study that attracts a lot of research, Hsiao et. al. discuss some of the main aims of this research which include Surveillance, Autonomous Flight and Navigation [33].

Another variation amongst mobile robots is their level of autonomy. The on-line Oxford dictionary defines Autonomy as "1.Self Government 2.Freedom of action" [1]. Robots may be fully remote control (ie: no autonomy), like various unmanned aerial military vehicles with ground control stations or harnessed deep water exploration vessels [4], [3]. Or they could be semi-autonomous like the Mars exploration vehicles which are locally autonomous, but await higher level commands from a ground control station, which has about a 15 minute transmission delay because of the distance to Mars [9]. Finally they could be near fully autonomous, like Rhino, one of various autonomous tour guides, which was able to function for 6 full days guiding museum goers to various exhibits and only experiencing 6 collisions the 47 hour test period [17].

2.1.1. Research problems within the mobile robotics field

There are many problems specific to mobile robots, which depend on their size and intended environment. Those detailed here will relate only to robots intended to be used in an indoor environment. It can be argued that in this environment the most significant problem is navigation. Navigation is built on four foundations, Perception, Locomotion and Motion control, Localisation and Cognition [50]. This is illustrated in Figure 2.1. In order for the robot to move from one point to another it must first be able to use its sensors to perceive its surroundings and gather relevant data. It must then use this sensor data to localise itself, i.e. it must have knowledge of its current position. Then it must be able to recognise a path from its current position to a goal. Finally it must be able to control its motion to arrive at this goal. The following sections will cover, in more detail, the work conducted by other scholars in these four areas.

2.2. Locomotion and Motion Control

Locomotion and motion control is the business end of a mobile robot. It is where all the calculations about where the robot should go come together and get applied to

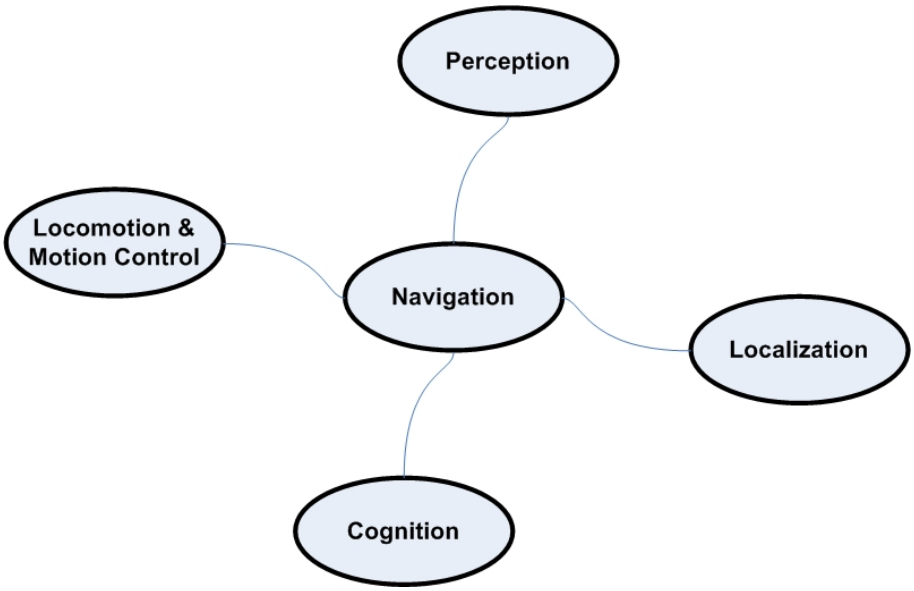

Figure 2.1.: Navigation Foundations [50]

the actuators. These actuators can vary dramatically between mobile robots. The following sections will discuss a few locomotion methods and some of the control methods that apply to them. In ground based mobile robotics there are generally two types of locomotion, legged and wheeled. While the topic at hand is wheeled robotics, legged robots deserve to be mentioned.

2.2.1. Legged robots

Legged mobile robots achieve locomotion using a series of point contacts with the ground. There are two main benefits of this locomotion method. Firstly, it allows a robot to contend with very rugged terrain, and secondly it allows a robot to be highly adaptable. A legged robot may be able to step over a hole or gap in which a wheeled robot would get stuck. A legged robot can move over rough terrain as long as there is enough stable area for it to put its legs down [50].

There are a number of problems with legged robots. These include complexity of the required gait, power consumption and torque requirements for leg actuators, which may at times need to lift the entire weight of the vehicle. Solutions for gait generation of these vehicles can be very interesting. In a hexapod biological cells have been used to learn to produce a gait which moves the robot away from a source of light [58]. For

power consumption there is much research going into leg-wheeled mobile robots, which will give the efficiency of a wheeled system coupled with the adaptability of legs [27].

Due to their advantages, and despite some disadvantages, legged mobile robots are very popular for rugged terrain vehicles like a hexapod designed for use as a mobile autonomous robot for Mars exploration [9]. This robot was used to prove feasibility of such a robot for planetary exploration tasks.In a smooth indoor environment, as specified for this project, there is no need for such complexity and this review will now focus on wheeled mobile robots.

2.2.2. Wheels used on mobile robots

Wheeled mobile robots are by far the most prevalent type of mobile robot. An excellent review of the implications for mobility of different wheeled robot structures is available which discusses the two broad types of wheels, namely conventional or standard wheels and composite or Swedish wheels [19]. This was later expanded to include Spherical wheels [50]. How these wheels are attached and controlled has implications for the manoeuvrability of the vehicle.

Depending on the application, a wheeled mobile robot will have a specific set of wheels. These may be steered or fixed and powered or unpowered. The number of wheels may vary from one, in the case of a single spherical wheel, to vehicles with 6 or more wheels [40]. The number of wheels have implications for stability, power consumption and manoeuvrability.

Tracked vehicles are another form of wheeled locomotion, but the track acts as a continuous surface for the wheels to roll on. This means that the vehicle is able to manoeuvre well in very loose terrains, but because of the skidding requirements for steering this can become a very inefficient form of locomotion [50].

2.2.3. Mobile robot steering

There are variety of ways to connect and control the various types of wheels, to allow vehicle steering. Standard wheel variations include fixed standard wheels, centre orientable wheels and off-centre orientable wheels, these wheels can then be powered or free to rotate [19], [50].

Generally, omnidirectional wheels aren't orientable as they already allow great freedom of motion when fixed, however there has been research with steered omnidirectional wheels [51]. This allows quite a large number of wheel configurations to be realised, with varying degrees of mobility and degrees of freedom. Only a limited number of these will be discussed in this section.

One of the common wheel configurations used in robotics is the differential drive robot, this may or may not include a trailing castor [10]. This set-up is very simple and only requires two actuated wheels, however there are consequences for motion planning and manoeuvrability. This is because of constraints imposed on the motion by the wheels and makes path planning very complex.

Ackerman steering is another well known steering system, this system can be found on all commercially available four wheeled motor vehicles and is also used in robotics, often when a commercial remote or radio controlled vehicle is converted for autonomy. More stringent restrictions apply to the manoeuvrability of these vehicles than for differential drive vehicles.

Most of the possible wheel configurations limit the motion of the vehicle or mobile robot in some way, however there are exceptions. Vehicles that have steering configurations that allow motion in all directions from any pose are omnidirectional vehicles. This can be achieved if all the wheels of the vehicle are steerable or if Swedish or Spherical wheels are used.

To achieve omnidirectional motion, the vehicle in this project used Swedish wheels. The design of these wheels can be very complicated and to achieve the desired motion requires control of individual wheel velocities, but they only require a single actuator per wheel because the wheels are not orientable [50].

2.2.4. Wheeled vehicle modelling

Models of mobile robots can be generated, by specifying constraints applied to the robot by different wheel types in various configurations. This has consequences for the robot's manoeuvrability and degrees of freedom. These constraints are then used to generate state space equations for the robot, equating system inputs (wheel motor torques), to outputs (vehicle position, velocity and acceleration) [19].

Another common way of modelling systems is to generate a freebody diagram of the relevant applied forces, which are due to motor torques. These are then used to equate forces to velocities using Newton's equations of motion. Once again these equations can then be used to generate state space equations which allow control of the vehicle.

Whichever method is used, a suitable set of state space equations must be found. It is then necessary to gather information about the inputs and outputs of the system, so that a suitable control system can be designed. The inputs to the system in this case will be the wheel speeds, proportional to motor voltage, which can be measured using quadrature shaft encoders. The outputs will be the position and velocity of the robot, which can be calculated using dead reckoning, which can be prone to errors, or some other form of position sensing like beacon based sensing or SLAM.

2.3. Locomotion with Mecanum wheels

Mecanum wheels are a specific type of wheel using multiple rollers, arranged around a central hub. When configured correctly they allow the vehicle three full degrees of freedom, in the x, y and rotational directions. This can result in significant savings in manoeuvring time and packaging space when these wheels are used for vehicle mobility [25].

2.3.1. Mecanum wheel design

Mecanum wheels can be designed for optimal smoothness, minimal floor loading and best traction. The optimal design to satisfy these requirements has been found to be a wheel with only six rollers [25]. More recent work on the optimal design of the wheel can be found where the design is optimised for manufacturability [31].

2.3.2. Mecanum wheel control

Generally Mecanum wheels are used in a configuration using four wheels, as this allows excellent mobility. A problem stated often about Mecanum wheels is that they slip on some surfaces, this can be attributed to the small contact surface of the wheel. In contrast to a standard wheel, which has a line contact with the ground at all times, which under compressions expands to a somewhat rectangular shaped patch, a Mecanum

wheel has a very small point or patch that makes contact with the ground (Please see attached CD for a video illustration). Under low friction conditions, such as a very hard surface or where there is a fluid underneath, the wheel may slip, which makes a much used method for navigation, dead-reckoning, unreliable. *Slip* should not be confused with the *roll* which the wheel experiences, which is systemic and allows the vehicle its omnidirectionality.

The problem of slip has been investigated a number of times. One way to deal with slip is to use optical dead-reckoning to correct for the resulting errors [22], [45]. The problem of slip in the forward direction was solved using an extra set of wheels that drop down for forward motion only [26]. This method also claims to get better efficiency in forward motion, and better performance over more rugged terrain, which a Mecanum wheel is unlikely to do. Use of omnidirectional vehicles in rough terrain has also been investigated [60].

A position corrective control method is discussed which corrects for position errors experienced by a Mecanum wheel vehicle [62]. This method is then implemented using dead-reckoning and an external camera as sensor systems allowing the control method to be successfully implemented [49].

The modelling in the above systems follows a free body diagram method. For ease of calculation, this method applies the simplification that the force developed by a wheel is applied through a single point in the centre of the wheels width and on its radius. Only one source was found which contested this method, it points out explicitly that the wheel contact point moves laterally from one side of the wheel to the other as the ground contact is passed along the length of each of the wheel's rollers [31]. This changes the dynamics significantly and makes the control of Mecanum wheels in robotics applications more complicated. The algorithm developed in this work will follow this premise.

2.4. Perception

Perception of a mobile robot deals with its ability to gather information about the environment in which it is moving. This is facilitated by the use of sensors which can be broadly classified as follows [50]:

- **Tactile Sensors** are usually the last line of defence for a robot, they tell the robot when it has touched another object, or when it is in close proximity to another object.
- **Rotational Sensors** are used to measure the rotational position or velocity of a motor or wheel.
- **Heading Sensors** use a fixed reference, like the Earth's magnetic or gravitational field to determine a direction of motion.
- **Beacon Based Sensors** find the position of the robot using a number of reference beacons, which may be active, like a radio beacon, or passive, like a barcode on the floor.
- **Ranging** sensors usually use time of flight to determine the range of an obstacle, this could include ultrasonic, infrared or laser range finders.
- **Vision based sensors** provide a large amount of data about an object, they can be used for monocular vision or stereo vision. Stereo vision allows simpler calculation of an object's position or velocity.

In this project a number of these sensors were applied, including tactile, ultrasonic, rotational wheel encoding and beacon based sensors. These were used together to allow the robot to localise and navigate properly.

2.5. Localisation

Once a mobile robot has received perception information from sensors, like the ones mentioned in the previous section, it is necessary to process this information to allow the robot to know where it is in its environment. There are a number of methods that can be used for the robot to localise itself.

There are a number of different methods available for robot positioning. These include odometry, inertial navigation, use of magnetic compasses, active beacons, GPS, landmark navigation and model matching. These techniques can be split into two sections, relative positioning, which includes odometry and inertial navigation, and absolute positioning, which includes the rest of the methods mentioned below [13].

All of these methods have associated problems and it is common to combine two or more methods to achieve better accuracy. A brief explanation of localization methods follows [13]:

- **Odometry** uses the number of rotations of the wheels to predict the distance that a robot has travelled in a certain amount of time. In a differential drive robot, by monitoring the rotational speeds of each of the driving wheels it is possible to predict the displacement of the robot. However, because of a number of errors which may be systemic or not, the accuracy of these predictions gets worse with time. Systemic errors are ones that occur inside the robot system, like a wheel diameter error, or wheel encoder reading errors. Non-systemic errors occur externally, these could include wheel slippage due to uneven terrain.

- **Inertial Navigation** makes use of accelerometers to keep track of the speed of the robot. Acceleration data is integrated once with respect to time to get vehicle velocity, and is integrated twice to get velocity. Integration drift errors cause this method to become inaccurate with time. It is best used for short periods of time or fused with other localisation methods.

- A **Magnetic Compass** uses the earth's magnetic field to determine the heading of a robot, however this method is effected by proximity to metal objects making it less useful for indoor applications.

- **Active Beacons** are points with known positions which emit energy pulses, such as ultrasonic waves or ultraviolet light. Depending on the system, the detector can then determine its position relative to these beacons, usually using triangulation.

- **GPS** is a global form of active beacon, data is provided by satellites orbiting the earth and relayed using radio waves.
- **Landmark Navigation** is where a detector like a camera or sonar detects known shapes in the environment. These may be predefined markers like barcodes fitted to the floor or wall, or natural landmarks like wall corner edges.
- **Model Matching** is where the robot attempts to fit itself into a predefined model or map. This is also known as map based navigation, but the map can be in the form of, for instance, a previous sonar scan of the environment. A more complicated map based method is Simultaneous Localisation and Mapping (SLAM). This is a much researched navigation method, which allows a robot to be placed in an unknown environment. While moving around, the robot generates a map of the environment and then positions itself within this map.

The chosen localisation methods for this project were odometry and active beacons, as they were relatively simple to implement and much of the hardware was already available before the project started. Both methods use proprietary hardware, the former being the Wheel Watcher quadrature encoder system, the latter being the NorthStar indoor localisation system.

2.6. Navigation

The effectiveness of a mobile robot is dependent on its ability to reach its goal and complete the task it set out to do. This in turn is dependent on its ability to adapt to different situations, represent those situations correctly within its model of the world and then select an appropriate action [20].

Navigation can be split into two different tasks, namely local navigation, which deals with close range obstacles (i.e. obstacle avoidance) and global navigation, which plans paths to the ultimate goal. Methods developed to handle these tasks can vary, but usually must work together for the robot to arrive at its final goal.

2.6.1. Path Planning

As the name suggests, path planning is how the robot determines the best route to get from one point to any other point in the world. A general method for path planning is to use a grid based map, then each possible square radiating from the current position is numbered escalating for squares further away from the origin, see Figure 2.2. This is

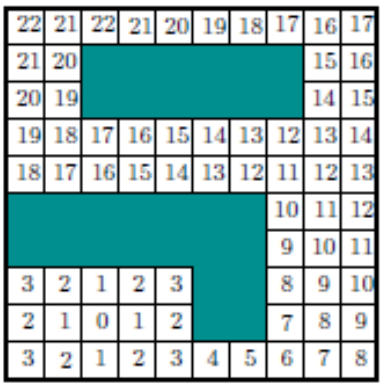

Figure 2.2.: Wavefront propagation [41]

generally known as a wavefront propagation method, but implementations of the method can vary. When a goal square is reached within the map it is then possible to follow the numbered squares back to the initial position and thereby create a plan. A brief outline to the planning algorithm used in this project has been given above, but this is a much more complex problem [41]. It is possible to use wavefront propagation algorithms with obstacle avoidance to form a complete navigation system [8].

2.6.2. Obstacle Avoidance

Obstacle avoidance can also be seen as local path planning. A robot must recognise that there is an obstacle blocking its path and must find a way around it. There are a number of techniques to do this and some have been written directly into Player, the chosen robot control code. These include the Vector Field Histogram and Nearness Diagram methods.

The Vector Field Histogram continually monitors the environment surrounding the robot. It then creates a polar histogram which describes the "obstacle density" in any direction. Once this histogram has been calculated, the algorithm decides on the most expedient direction in which to move based on the obstacle density value. It does this by choosing the direction that is closest to the goal with the smallest obstacle density [14].

A newer algorithm, the Nearness Diagram, uses a decision tree that evaluates the robot's current situation. Decisions are made depending on how near obstacles are and where they are with respect to the goal. Situations are classified as safe or not depending on whether an obstacle is within a predefined range and direction or not, and different actions are planned for different situation types [43].

2.7. Software development for mobile robots

Much of the work that must go into the development of a mobile robot is involved with the development of software. All the environmental data that is collected by the robot sensors must be processed, sometimes by an embedded microprocessor and sometimes by a more powerful PC.

Once the main decision making program has decided on a course of action, it must be implemented using the robot actuators. Depending on the robot architecture this may require another embedded processor which controls the motors, or there may be a direct serial link from the PC to the motor controller. All of these items need software to be written and this can result in significant research effort going into software development.

Most of this required software is similar to many different robot projects and research goals. A number of research groups have recognised this and there are numerous Robot Development Environments (RDEs) available. A good review was found, which evaluates nine open source, freely available, RDEs [38]. These vary in capability, usability and

impact. A very popular RDE is the Player Project, it was originally developed by researchers at the University of Southern California [30]. It now has an active community of developers and users worldwide [61].

The CSIR Mechatronics and Micro-Manufacturing research group, has chosen to implement Player as a group wide RDE so that knowledge, experience and software can be shared amongst researchers and projects. This choice was made due to the wide use of Player in the robotics community and because the software is open source, freely available and relatively easy to use.

2.8. Mobile Robots for Service Applications

Service robots have huge growth potential and can be applied in many diverse environments and situations [47]. Office service robot are capable of delivering faxes in an office environment, using only ultrasonic sensors for navigation [11]. MOPS, a Mobile Post Distribution System is another example of an office service robot, it uses pattern recognition to read the door numbers of the people it is distributing postal items to [57].

Quite a famous robot is the Roomba, an autonomous vacuum cleaner, which is readily available as a consumer item. Such a machine can have interesting effects on the social environment, for example allowing people who are not traditionally involved in housework to become more involved [29]. Similair social effects would no doubt also be a consideration for an office environment employing such machines.

Service robots have also been used to help disabled people become capable workers on a factory floor, with the aid of a wheelchair that also functions as a mini forklift [36]. The Intelligent Hospital Service Robot performs simple tasks which free up staff to do more complex work, these tasks include chart distribution, clothing distribution and leading patients around the hospital [48].

There are a wide variety of service robots available, ranging from fax delivery robots to robots helping with rehabilitation of disabled people. This project will focus on indoor service applications aimed towards the office environment.

2.9. Conclusion

The above literature has briefly discussed all the necessary components to make a working mobile robot. Construction and modeling of the vehicle to fit the desired specifications will require application of mechanical engineering. This is an electrically powered system, the actuator and sensor devices will need to be integrated using electronic design methods. Most of the electronic components require some kind of software to gather data or give commands. The full system requires overall commands to be given by planning software. The design of this mobile robot requires the integration of mechanical, electronics and software components, as such it is a typical application of a mechatronics system [34].

Chapter 3.

Mechanical Design, Construction and Modelling

3.1. Introduction

This chapter will discuss the mechanical tasks completed in this project. This is a mechatronics project which must include a mechanical component. Mechanical challenges have been a significant part of this project. The basic design of the platform was inherited from a previous student working for the CSIR, but much of the finer details were completed as part of this project. The aluminium wheel itself was designed by Unathi Diniso, and four of these wheels had been manufactured before the project, in its current form, was started. The platform base, which is a simple box, was also designed by him, as were the mountings for his chosen motors.

All subsequent mechanical work was completed during this project, this includes the design of mountings for wheel encoders, ultrasonic sensors, the printer, the NorthStar indoor navigation system and the touch screen user interface. Design of a bumper ring and bump sensors, and the design of a test set-up to enable the wheel to be tested against the model developed in Section 3.5.

The most significant mechanical work done in this project is the mathematical modelling of the wheel itself. The model is developed from first principles and is of a generic nature that would allow its use for wheels with non-standard angles, i.e. not 45°. To test the developed mathematical model the wheel tester allows a single wheel to be run on a treadmill. The treadmill is able to change the apparent angle of motion of the wheel relative to the wheel axis.

3.2. Wheel Design

The wheel used in this project was based on the split wheel design suggested by the original inventor, Bengt Ilon , and was made using solid aluminium rollers without including rubber treads due to cost constraints, see Figure 3.1 [35]. The physical design of the wheels were not part of this project.

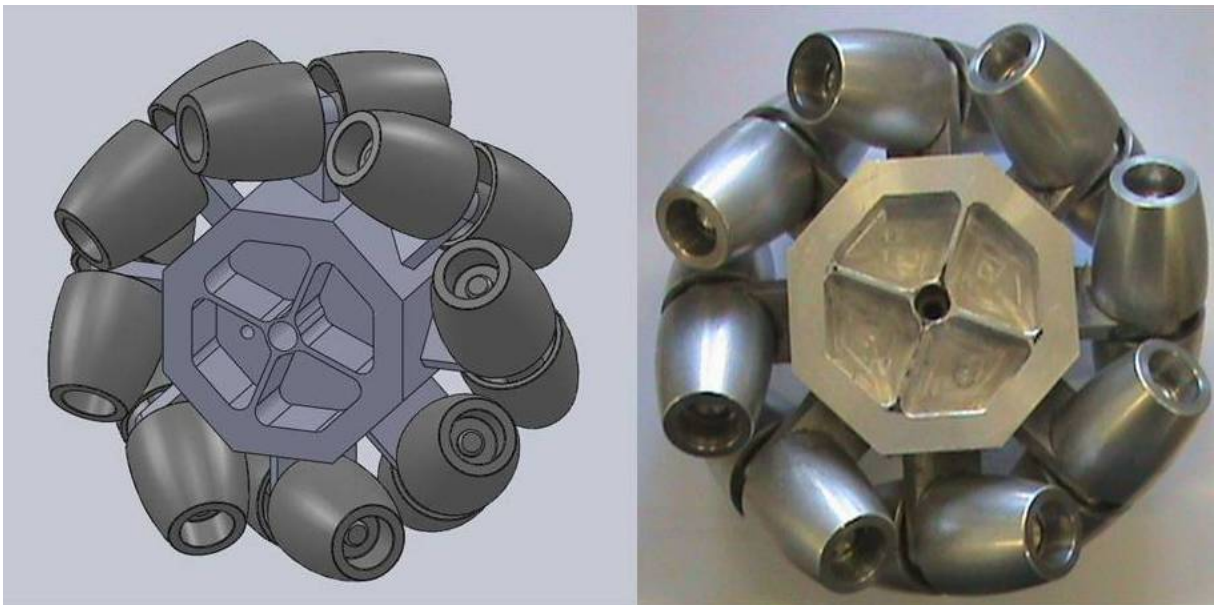

Figure 3.1.: The Mecanum wheel used in this project

3.3. Construction and Vehicle Assembly

The vehicle system is made up of numerous parts, each part mounted firmly and correctly. Items which required simple mounting include the batteries, the single board computer, the micro-controller and the motor controllers. Mounting these required little more than to drill holes in the side of the platform where they were best positioned for access and functionality.

The box shaped shell used as the platform base limited the available options of how to attach items to the robot. The batteries were positioned at the bottom and centre as these are the heaviest items present. The motor controllers were positioned on the sides, each between a pair of wheels. The rest of the items were placed where logic and space permitted. A general view of the outside of the platform is shown in Figure 3.2.

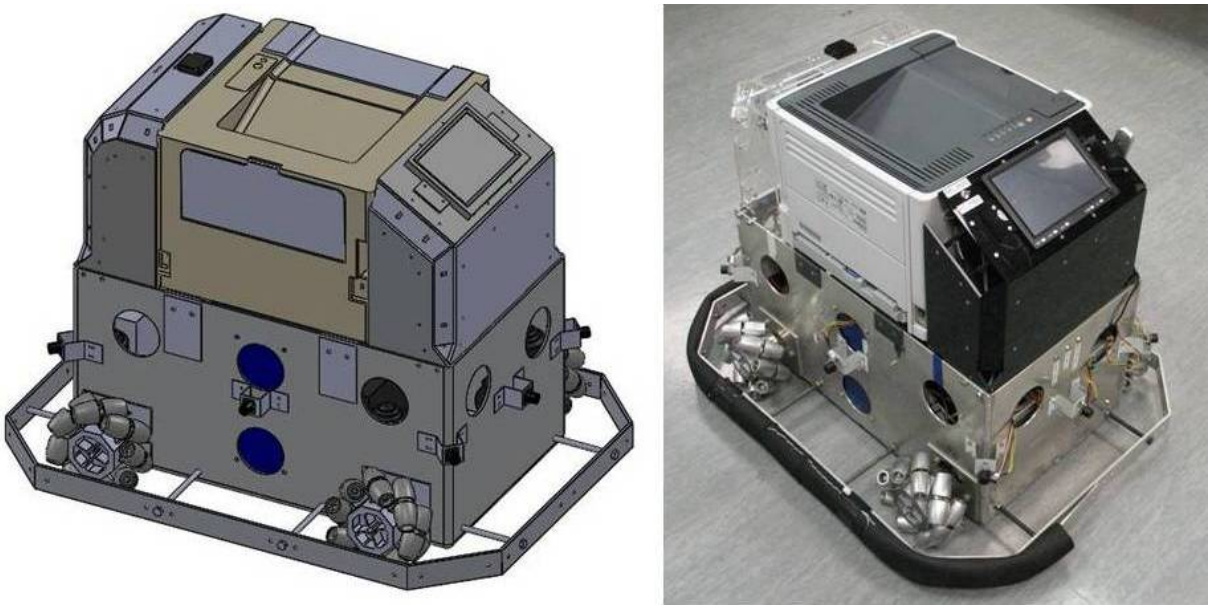

Figure 3.2.: Fully assembled vehicle

A lot of work went into the design of other mechanical hardware which needed to be in place for the platform to work correctly. The rest of this section will describe these parts and drawings of these parts will be included in the Appendix A.

3.3.1. Component placement on robot

An interior view of the robot is given in Figure 3.3 showing most of the components mounted on the robot.

3.3.2. Bumper Design

The bumper was designed to act as a last line of defence for the robot if it approaches an obstacle and this obstacle is neither mapped, nor observed by the robot's sensors. The bumper is a firm octagonal ring which stops the platform when it bumps into anything at floor level. As it is, it cannot protect the robot from items at table level, but to a greater extent the platform is low enough to avoid these. The design of the bumper ring can be seen in Figure A.10.

The first attempt at bumpers was a set of flat metal plates, spring loaded with leaf switches as contacts. However once the platform was properly mobile it was noticed that

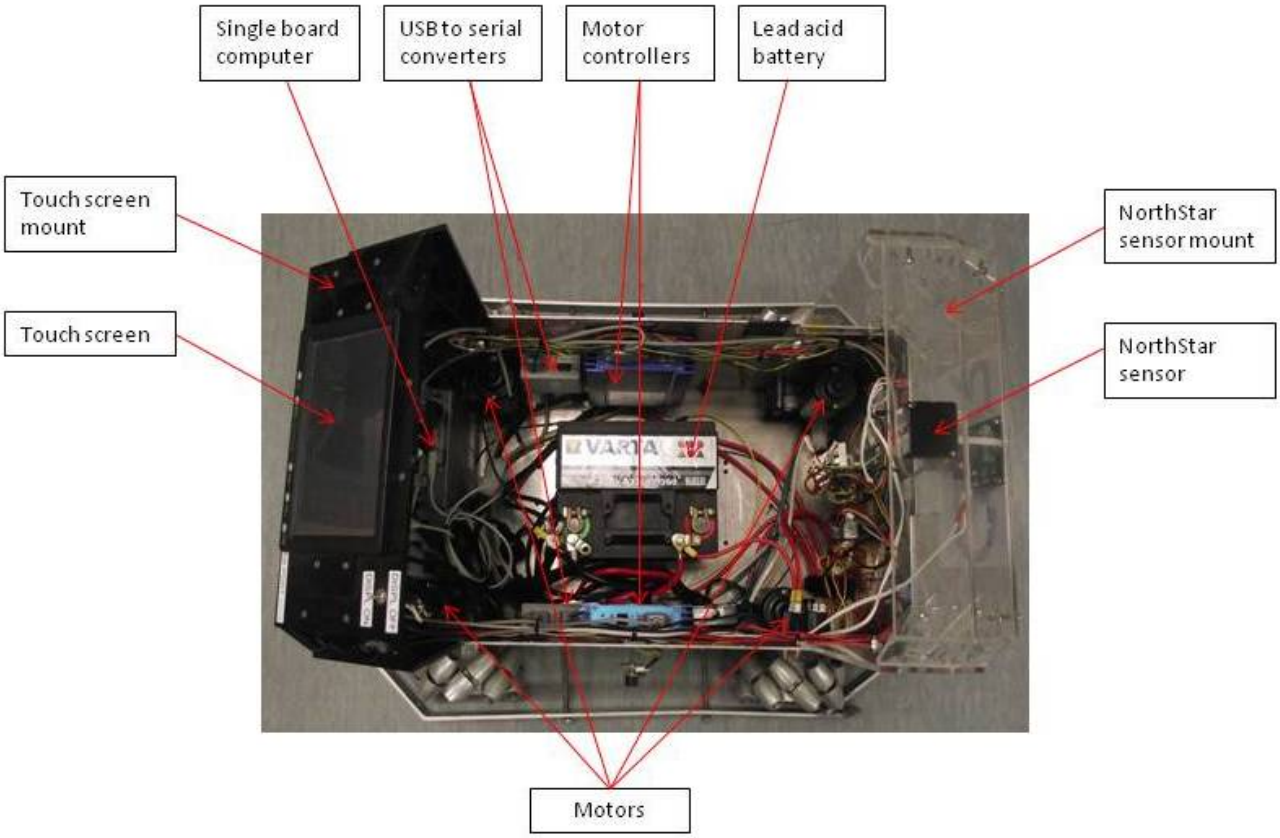

Figure 3.3.: Top view of components in the robot

they were definitely not suitable as they caught on objects when the platform approached obstacles at an acute angle. This did not allow the bump switch to be activated before a collision and often resulted in damage to the bumpers themselves or to the object with which the robot collided

Another solution was suggested using a thin piezoelectric film made of polyvinylindine fluoride. This film was then laminated in plastic and inserted into a rubber bumper which was in turn attached to the octagonal ring. Some extra circuitry was required and will be discussed in Section 4.1.2.

Ultimately touch sensing was abandoned as a sensing method. Once correctly implemented the other sensors proved sufficient to avoid obstacles. The bumper ring remained attached and kept the robot wheels from being damaged by items below the sensors.

3.3.3. Wheel encoder mounts

The wheel motors were mounted on the side walls of the platform. The wheels themselves were mounted on the motor shafts. The platform shell wasn't initially designed to have wheel encoders, so mountings had to be added on that were thin enough to fit between the wheels and the side wall of the platform. The mounts were designed to match the chosen wheel encoders and not to interfere with the wheels themselves. Each of the four wheels had its own wheel encoder. See Figure A.11.

3.3.4. Ultrasonic sensor mount

Eight ultrasonic sensors were mounted around the platform to allow obstacle avoidance. They were equally spaced at 45° increments, all the way around the vehicle as this is an omnidirectional vehicle. The actual mount was made of 0.5 mm sheet steel bent either end to fit on the flat sides of the vehicle or on the corners of the vehicle. See Figure A.12.

3.3.5. Other mounts

The printer was mounted in the centre of the vehicle using two simple 2 mm mild steel plates bolted to non-functional parts of the bottom of the printer. These plates were then bent to fit correctly over the edge of the walls of the platform. See Figure A.13.

The North Star sensor needs to be mounted at the highest point on the platform allowing the best view of the projected light spots on the ceiling. This means that it needed to be at least as high as the printer. A perspex mounting assembly was designed, laser cut and assembled. See Figure A.14.

The use of a touch screen was an idea that occurred later on in the project. It is a small 8 inch touch sensitive screen which will be used as an interface with users. Another simple mounting assembly, similar to the North Star mount was constructed to house the 8 inch touch screen. The mount rests on the edge of the main structural shell. See Figure A.15.

3.4. Wheel tester design

The idea for a wheel tester came from the need to test the proposed model for controlling Mecanum wheeled vehicles and a need in a separate project to compare different wheel types. The mechanical design of the tester can be split into 3 sections: the frame, the treadmill and the motor mount. All of these sections provided their own challenges.

The frame design went through a number of iterations, from a perspex mock-up, to an angle iron design, which was not constructed and finally to an aluminium extrusion design, which was very successful. The treadmill design also started as a perspex mock-up, but was constructed using aluminium for the final iteration. There were a number of problems experienced with this design which will be discussed in a following section. The motor mount was quite simple but as problems were solved with the treadmill minor changes also had to be made with the motor mount. See Figure 3.4 for a design drawing of the tester and 3.5 for a picture.

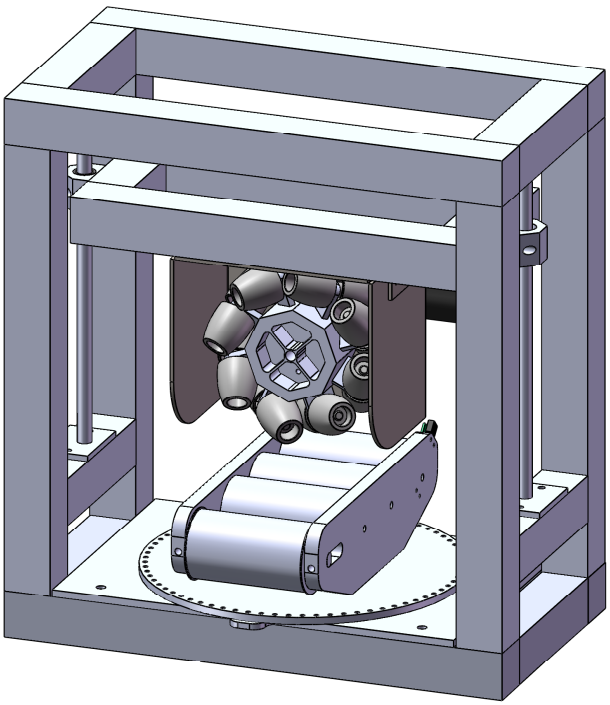

Figure 3.4.: Design of Wheel tester

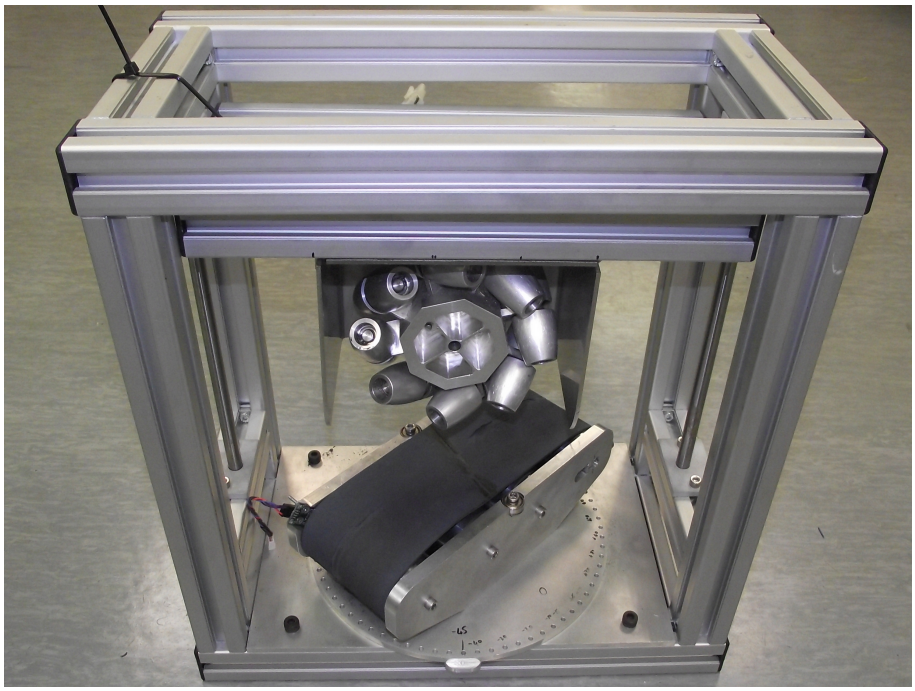

Figure 3.5.: Wheel tester

3.4.1. Frame

The perspex mock-up proved not to be rigid enough to use as a final version, but it was a very good exercise as it gave a good feel for the scale and rigidity required. The frame was redesigned using aluminium extrusions. The result is a simple box shaped frame with linear bearings on each side attached to motor mount. This allows the wheel freedom to move in the vertical direction and allows loading of the wheel to better simulate the real loading conditions it is used under. The added weight giving the wheel better traction on the belt and requiring more torque from the wheel motor. See Figure A.17 for drawings of the frame.

3.4.2. Treadmill

The initial idea for the treadmill was to use the parts of a belt sander, but destroying an expensive piece of equipment in the hopes that it might be the solution proved too much to risk. It was then decided to make the treadmill out of perspex and thick plastic bar, with the base dimensions made to fit a 600 mm $\times$ 100 mm belt from a belt sander.

Initial tests showed that neither the belt nor the roller assembly were fit for purpose. Finding non abrasive sanding belts proved difficult and eventually it was decided to use a rubber belt. The treadmill itself was finally designed using four aluminium rollers mounted on bearings and held together with aluminium side plates.

Numerous changes had to be made to the treadmill as problems were discovered and corrected. These problems included:

- Wheel Encoder Recess – Omission of a recess in which to mount a wheel encoder for comparison between wheel speed and belt speed. Further machining was required to rectify this.
- Belt run off – The rubber belt on which the wheel rolled moved to one side so that it started rubbing on the side-plates. This caused undesired friction which would affect the results and often caused the belt to stop moving entirely.

  The first attempt to prevent this was to indent the end roller slightly so that the belt ran in a groove, but this failed to solve the problem. The next solution was to machine recesses and mount bearings into the treadmill side-plates which would allow the belt to roll on them, instead of rubbing. This solution was mildly successful, but at times the belt would simply move over top of the bearing and rub on the sides of the bearings instead of their rounded surface. It was then necessary to make plates above and below the bearing to ensure that the belt ran on the bearings only.
- Encoder disk recess – After running a number of tests it was noticed that the treadmill speed, recorded by the wheel encoder, would oscillate between zero and a maximum, in a very regular fashion. It was then observed that the treadmill motion was relatively constant and this motion was not reflected by the sensors. Once the treadmill was dismantled it was found that the printed encoder disk was running too close to the side-wall and some of the dark, non-reflective markings had been rubbed off. The solution was simply to increase the depth of the recess in the side-wall that was already there for this very purpose.

Once these changes had been made the tester produced satisfactory results which will be discussed further in Section 6.1. For the mechanical drawings of the Treadmill please see Appendix A.18. Figures 3.6, 3.7 and 3.8 show the treadmill at different test angles.

Figure 3.6.: Treadmill at 0°

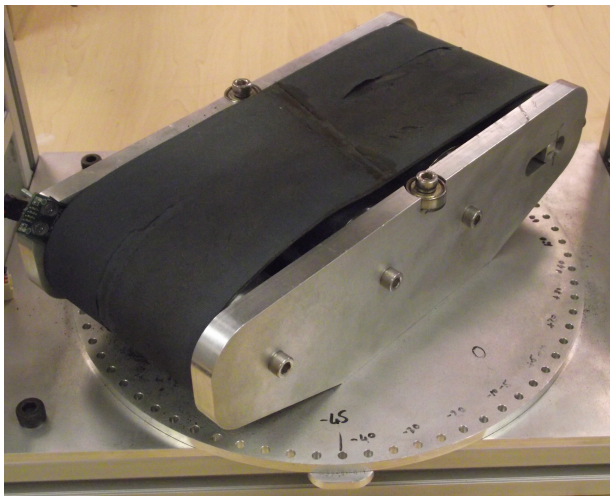

Figure 3.7.: Treadmill at 45°

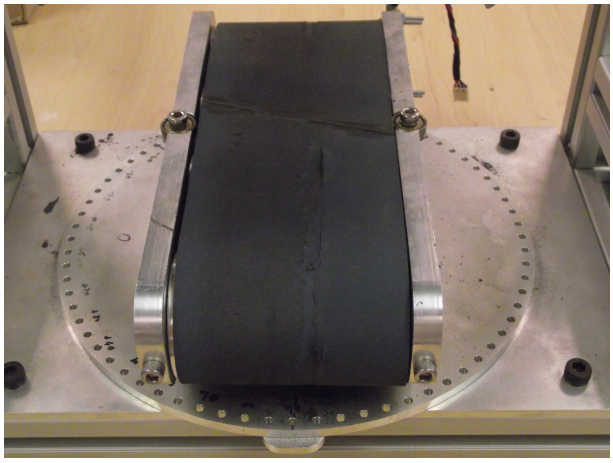

Figure 3.8.: Treadmill at 90°

3.4.3. Wheel and Motor Mount

The wheel tester used the same motor as is used on the platform itself. An encoder was attached to keep track of wheel rotational speed. Aluminium extrusions were used to make a bridge between two linear bearings, attached by smooth steel shafts to the main frame. The mount, made of laser cut stainless steel, was then bolted underneath this bridge. There were also some minor adjustments that needed to be made to this design. As changes were made to the treadmill it was found that the wheel mount interfered with some of the new hardware on the treadmill. This was remedied by cutting off a portion of the mounting assembly, which was not functional.

3.5. Modelling of the Platform

Much of this section and Section 6.1 are based on a conference paper presented at the *25th International Conference of CAD/CAM, Robotics & Factories of the Future* [24]. Modelling of a Mecanum wheel vehicle for the purposes of robotic control has been investigated in other works [56], [28], [51]. These works are based on the simplification that defines the contact point of the wheel with the ground to be positioned on the circumference of the wheel, but in the centre of the wheel's width. When the wheel in motion is examined closely it is observed that the contact point moves from one side of the wheel to the other as the contact point moves down the angled roller, this motion is demonstrated by the three simulation videos on the attached CD. Some papers do not mention this movement at all others have mentioned it but it is not analysed mathematically [28], [56]. A more recent paper pointed this out, but the analysis of the model and resulting equation is different to the one proposed in this paper [31].

3.5.1. Wheel Forces

A Mecanum wheel is a wheel with a number of rollers arranged around the hub at a predefined angle θ, usually $\pm\frac{\pi}{4}$ radians (45°), to the rotational plane of the wheel as can be seen in Figure 3.1. When the wheel rotates there is a force applied down the roller axis, generated by the torque applied to the motor shaft, so this force acts at angle θ to the wheel plane. The rollers rotate freely about their respective axes as they make contact with the ground.

In Figure 3.9, a local co-ordinate system is defined with two orthogonal axes, x and y. Positive rotation is defined as anticlockwise around the origin. The wheels are made and attached such that the forces produced by the wheels act in the directions of the dotted lines in Figure 3.9. These wheel forces can then be split into x and y components and summed to get the following equations (3.1),(3.2) defining the total force applied to the vehicle, the subscript T is applied to show that these are total forces:

$$F_{Tx}\mathbf{i} = \sum_{w=1}^{4} F_{xw}\mathbf{i} \tag{3.1}$$

$$F_{Ty}\mathbf{j} = \sum_{w=1}^{4} F_{yw}\mathbf{j} \tag{3.2}$$

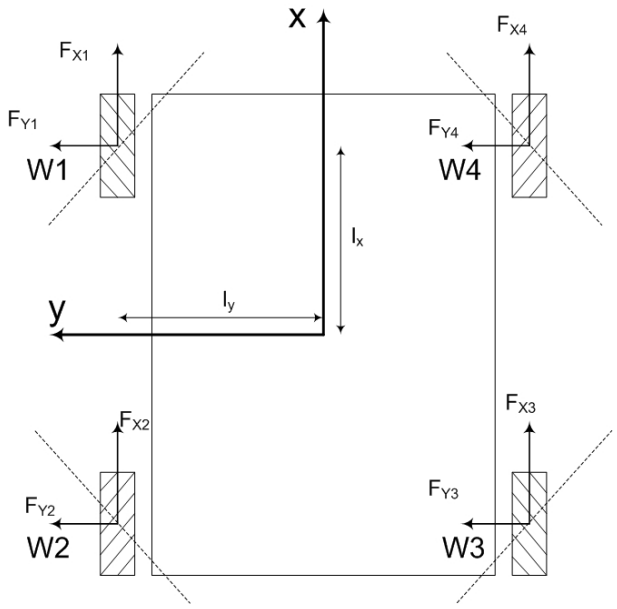

Figure 3.9.: Forces acting on a Mecanum wheel vehicle

Where **i** and **j** are unit vectors in the x and y directions respectively and w represents the wheel number. If there is pure translation, i.e. no rotational motion, then the direction of motion, which makes an angle α with the x-axis can be shown to be,

$$\alpha = \arctan \frac{F_{Ty}}{F_{Tx}}$$

If there is rotational motion then moments are taken around the vehicle centre at (0,0), which results in an equation for torque, τ, which will result in rotation of the vehicle. The resulting torque can be expressed with the following equation,

$$\tau = (-F_{x1} - F_{x2} + F_{x3} + F_{x4})l_y + (F_{y1} - F_{y2} - F_{y3} + F_{y4})l_x$$

Where l_x and l_y are half the wheel base and track respectively and are the radii at which the forces are applied by the wheels.
For the dynamic case Newtons Second Law allows the following equations to be applied:

$$\begin{aligned} m\ddot{x} &= F_x - c_x\dot{x} \\ m\ddot{y} &= F_y - c_y\dot{y} \\ I_0\ddot{\phi} &= T - c_z\dot{\phi} \end{aligned}$$

$c_{x,y,z}$ are the damping factors for motion in the three dimensions, assuming that in the range of operation the system damping factor is constant and is comprised primarily of frictional damping components. m is the mass of the vehicle and I_0 is the moment of inertia of the vehicle. Because this is not a spring system there is no displacement term and in the "steady state" case all accelerations (translational and rotational) are zero:

$$\begin{aligned}
\dot{\hat{x}} &= \tfrac{F_x}{c_x} &&= \frac{F_{x1} + F_{x2} + F_{x3} + F_{x4}}{c_x} \\
\dot{\hat{y}} &= \tfrac{F_y}{c_y} &&= \frac{F_{y1} + F_{y2} + F_{y3} + F_{y4}}{c_y} \\
\dot{\hat{\phi}} &= \tfrac{T}{c_z} &&= \frac{(-F_{x1} - F_{x2} + F_{x3} + F_{x4})l_y + (F_{y1} - F_{y2} - F_{y3} + F_{y4})l_x}{c_x}
\end{aligned}$$

In the steady state case the velocities are shown with a caret to differentiate them from the case when the vehicle is accelerating.

This implies that in the steady state the vehicle velocity and, by implication, the wheel velocities, are proportional to the forces applied by each wheel:

$$V \propto \sum_{w=1}^{4} \underline{F_w} \tag{3.3}$$

and so, because this is a rigid body, for translational motion

$$V = V_w \qquad\qquad w = 1 \rightarrow 4 \tag{3.4}$$

3.5.2. Modelling a Single Wheel

Consider Figure 3.10, the dotted lines represent the path taken by the contact point of the wheel with the ground when the vehicle is moving straight forward, note that this contact point moves back and forth along the wheel axis, for an illustration of this movement, please see the video folder of the attached CD. In the figure the horizontal dotted lines are discontinuities where the contact point transfers from one roller to the next, however for the purposes of calculation the vector origin is put halfway along this line. The diagonal dotted line is in the same direction as the roller axis. The vertical

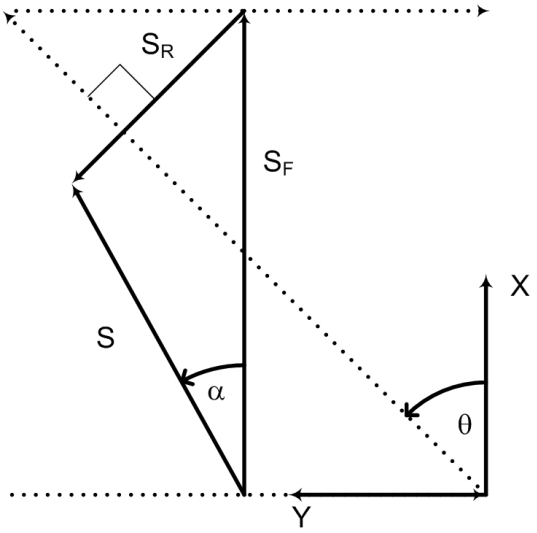

Figure 3.10.: Wheel displacement vectors

solid line represents the displacement vector due only to wheel rotation S_F, in the positive x-direction. The solid line S_R represents the displacement due to rolling, in the direction orthogonal to the roller axis. S represents the resultant displacement of the wheel.

The angle α is the direction of the resultant displacement vector, and is measured from the x-axis, θ is the angle that the roller makes with the x-axis. The above displacement vectors translate to the velocity vectors when differentiated with respect to time, resulting in Figure 3.11, where ω is the wheel rotational velocity. Positive rotation results in a positive velocity vector $\omega r\mathbf{i}$ along the x-axis, r is the wheel radius and $\mathbf{i}$ is a unit vector in the x direction. Consider that the components along the roller axis of the $\omega r\mathbf{i}$ and $\mathbf{V}$ vectors are equal, hence

$$\omega r \cos\theta = V\cos(\theta - \alpha) \tag{3.5}$$

$$\omega r = V\frac{\cos(\theta - \alpha)}{\cos\theta} \qquad \theta \neq \frac{\pi}{2} + n\pi; n \in \mathbb{Z} \tag{3.6}$$

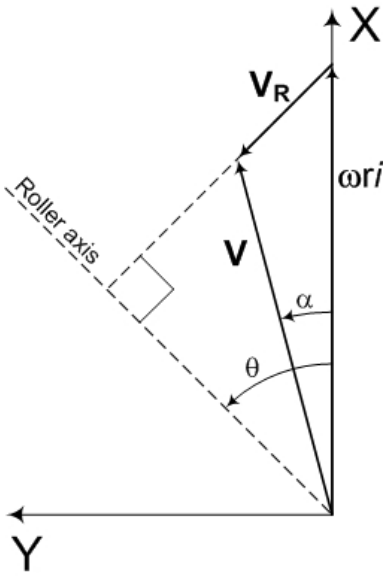

Figure 3.11.: Wheel velocity vectors

When $\theta = \frac{\pi}{2} + n\pi$, the rotation of the wheel results in no translational motion. When $\theta = n\pi$ the wheels become "standard" 90° omni-wheels, but in the configuration of a Mecanum vehicle this will allow only forward motion, omnidirectionality is lost. Additionally any external force in the y direction will result in motion in that direction as the rollers will allow uncontrolled motion.
Rearranging (3.6) results in an equation for the magnitude of the velocity in the direction α:

$$V = \omega r \frac{\cos(\theta)}{\cos(\theta - \alpha)} \qquad \theta \neq \frac{\pi}{2} + n\pi; n \in \mathbb{Z} \tag{3.7}$$

When $(\theta - \alpha) = \frac{\pi}{2}$, the rotational speed of the wheel must be zero, but any value of translational motion is valid as this is driven by other wheels on the vehicle.

3.5.3. Translational Motion

The equations for the forces acting on the vehicle and the vehicle velocity can be used to calculate the wheel velocities (ω) required to move the vehicle in any particular direction α at any particular speed V. Assume that there is enough motor torque to overcome friction in the drive train and rolling friction, and that the vehicle motion is in steady state. It is known from (3.3) that $V_w \propto F_T$ and from (3.4) that, $V_w = V$, because this is

a rigid body. Using (3.6) we get an equation for wheel rotational velocity ω_w:

$$\omega_w = \frac{V\cos(\theta_w - \alpha)}{r_w \cos\theta_w} \qquad (\theta - \alpha) \neq \frac{\pi}{2} + n\pi; n \in \mathbb{Z} \tag{3.8}$$

3.5.4. Rotational Motion

It is now possible to define equations for rotation of the vehicle around any point in the plane. In this case the angle α defining the direction of motion of an individual wheel is no longer the same. It varies depending on the position of the centre of rotation of the vehicle. For (3.8) to be applied an additional calculation must be made to determine α_w, $w = 1 \rightarrow 4$.
At an arbitrary point in time it is desired that the velocity of the origin of the vehicle

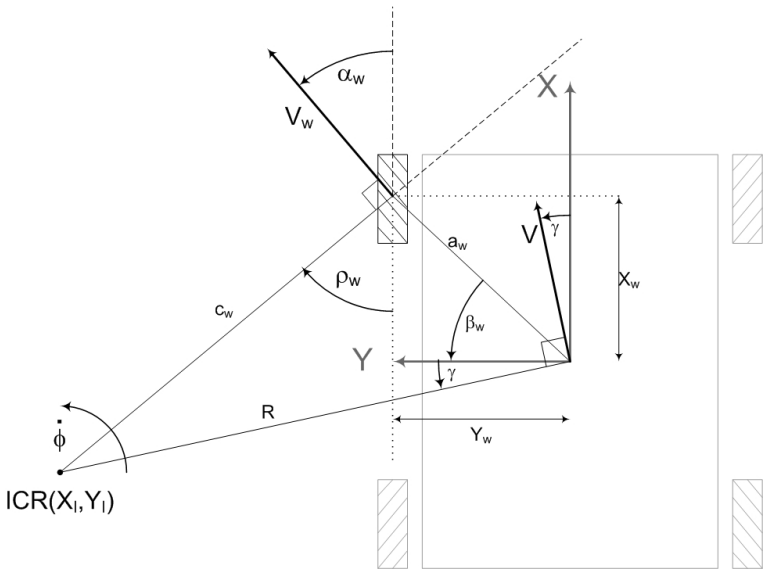

Figure 3.12.: Vehicle and ICR geometry

be $V = \mathbf{i}V_x + \mathbf{j}V_y$ and the rotational velocity be $\dot{\phi}$ around the resulting Instantaneous Centre of Rotation (ICR) [50]. The radius to the ICR in Figure 3.12 is then:

$$R = \frac{V}{\dot{\phi}} \tag{3.9}$$

and γ is the angle the radius makes with the y axis defined by:

$$\gamma = \arctan\frac{V_y}{V_x} \tag{3.10}$$

which allows us to define the position of the ICR:

$$X_I = -R\sin\gamma \tag{3.11}$$

$$Y_I = R\cos\gamma \tag{3.12}$$

Consider Figure 3.12, we again define a local reference frame with its origin at the centre of the vehicle. For the purposes of the geometric calculation we now fix the contact point of the wheel with the ground at the centre of the wheel's width.
Each wheel is then defined by its position X_w and Y_w and the orientation of its rollers θ. Let β_w be the angle from the y-axis to the ground contact point of wheel w, then

$$\beta_w = \arctan\frac{X_w}{Y_w} \tag{3.13}$$

Let a_w be the distance from the origin to the contact point of wheel w, then

$$a_w = \sqrt{X_w^2 + Y_w^2} \tag{3.14}$$

In the case of a Mecanum vehicle all a_w should be equal and constant as the geometry of the vehicle is fixed and symmetrical. β_w can be pre-calculated and will remain constant. Using (3.9), (3.10) ,(3.13) and (3.14) it is now possible to calculate the distance from the wheel contact point to the ICR. Let c_w be this distance

$$c_w = \sqrt{a_w^2 + R^2 - 2a_wR\cos(\beta_w + \gamma)} \tag{3.15}$$

The resultant wheel velocity V_w must act perpendicular to the line c_w which makes an angle ρ_w with the x-axis, note that in Figure 3.12, ρ_w is negative, which follows since in the illustrated case $X_I - X_w$ will be negative

$$\rho_w = \arctan\left(\frac{Y_I - Y_w}{X_I - X_w}\right) \tag{3.16}$$

$$\Rightarrow \alpha_w = \tfrac{\pi}{2} + \rho_w \tag{3.17}$$

Given (3.9), the instantaneous resultant velocity of the wheel, V_w, orthogonal to the line c_w, can be shown to have a magnitude of:

$$|V_w| = c_w|\dot{\phi}| \tag{3.18}$$

Equation (3.18) can now be substituted into (3.8) to calculate ω_w, resulting in

$$\omega_w = \frac{c_w \dot{\phi} \cos(\theta_w - \alpha_w)}{r \cos \theta_w} \qquad \theta \neq \frac{\pi}{2} + n\pi; n \in \mathbb{Z} \tag{3.19}$$

Which is a general equation giving wheel velocity ω_w in terms of the position of the ICR (X_I, Y_I) and rotational velocity around this point, ultimately in terms of velocity $\mathbf{V}$ and rotational velocity $\dot{\phi}$. This equation was then applied in software to calculate the velocities of each of the wheels for any desired motion.

3.6. Chapter Summary

This chapter has described the mechanical aspects of the project. The chapter begins with a brief description of the wheel design implemented in this project. It then describes how the vehicle was assembled including basic mounting tasks. Then the wheel tester is described, which is used to test Mecanum wheel performance. Finally a mathematical model is developed which will allow proportional control of all the wheels depending on the desired motion.

Chapter 4.

Electronic and Electrical Design and assembly

This chapter will discuss the electronics and electrical systems which are included in the robot system. These systems include sensors, electronics and wireless communications, microcontrollers, Computer, DC and AC electrical power systems. Most of the electronics was bought off-the-shelf, but there was some circuitry that had to be built to integrate the systems and interface with certain sensors. Most of the interfacing was performed using a micro-controller board which was able to gather analogue and digital data, and to communicate with the computer and other devices using a serial interface. All the onboard electronics was powered using a lead acid battery.

4.1. Sensors

To achieve a level of autonomy a mobile robot requires sensors to know where it is and what is close to it. A number of sensors were implemented in this project. Some for obstacle avoidance and some for localisation. The next few sections will discuss the electronics required to interface with these sensors.

4.1.1. Ultrasonic sensors and interface board

The ultrasonic sensors that were used on this project were the MaxSonar EZ1 Sonar range finder boards. There are three methods to interface with these boards. The first possible method is an RS232 based communication protocol. The drive voltage levels are 0-5V, standard RS232 levels are +5 to +15 volts (logic low input) -5 to -15 volts (logic high input) [32]. The next possible interface method is an analogue voltage output between 0 and 2.55 volts, with 10mV per inch scaling factor. The last method is a pulse width output, the output is scaled as 147μs per inch.

The RS232 method was chosen as it is less susceptible to noise, and easier to implement for multiple sensors. Eight sonars were used as the platform is omnidirectional and required a sonar to face in as many directions as possible. In order to feed this directly to the micro-controller and to isolate the sonars from one another a transistor circuit was built. This circuit inverts the signal and allows less current to be directly drawn from the sonar sensor. Circuit boards were then cut out with an in-house milling machine, and populated. See Figure 4.1.

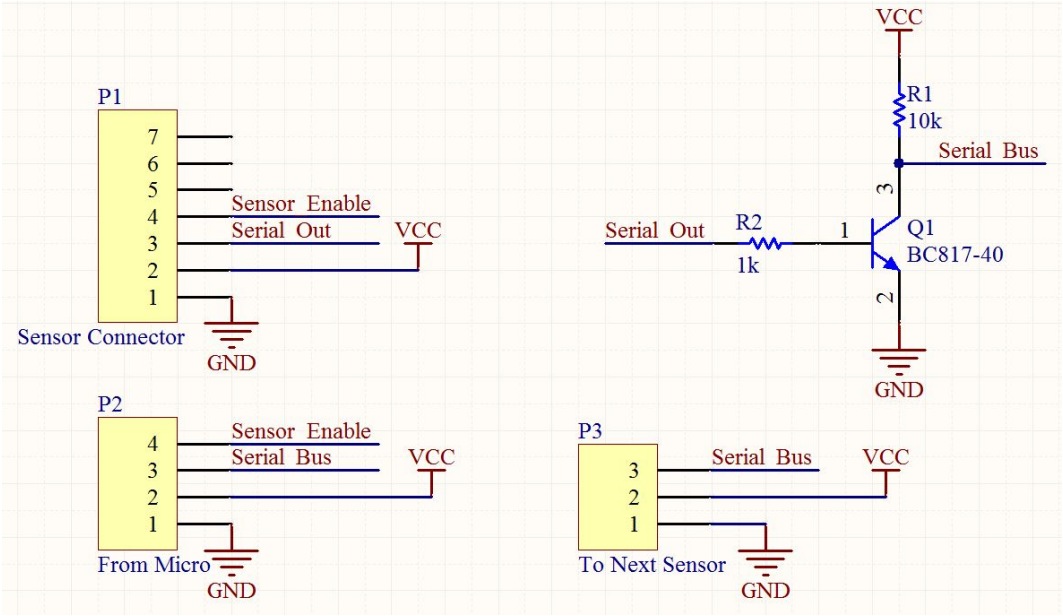

Figure 4.1.: Ultrasonic Interface board schematic

Multiple sensors were used but connected to a single RS232 serial bus. Enable wires were connected to each sensor, allowing only one sensor to use the serial bus at a time. Power lines were routed to each sensor.

4.1.2. Touch sensor boards

Originally simple touch sensors were used, which entailed simple flat metal plates contacts and springs, however these proved not to be successful. It was decided rather to go with PVDF (Polyvinylidene Fluoride) film which is a piezo-electric film. See Section 3.3.2 for the mechanical construction. A simple flip flop circuit was designed by another student, Yaseen Suleman, for use with the bumpers, which were finally made with PVDF film. The circuit uses two opamps to filter and amplify the signal from the piezo film. Then it uses a flipflop to capture only the first spike and a simple RC circuit to reset the flipflop for the next spike. Each board accepts sensor signals from 2 sensors and transmits these to the micro-controller board via the wiring loom. This means that the touch sensor bus includes power to each of eight boards and 14 digital return wires for each attached sensor.

4.1.3. Wheel Encoders

Wheel encoders were used for localisation implementing a method called dead-reckoning. In this project the wheel encoders used were Nubotics Wheel Watcher WW01 boards. These boards use two infrared transceivers to provide quadrature encoding. The electronic connections for the wheel encoders were very simple, they needed only two power wires and two signal wires, however all the connections were routed to the micro-controller board as at the time of assembly it was not certain which decoding method would be used later in software.

4.1.4. Infrared Scanner

Navigation experiments were conducted using an infrared range scanner. The scanner is significantly cheaper than the more widely used laser scanners, discussed in the next section. A Hokuyo PBS 03JN scanner was used, it uses a serial RS232 interface to connect to the Single Board Computer (SBC). This scanner has a range of 3m and 180° which is sufficient for indoor use. Its angular resolution of 1.8° is somewhat less than the resolution of a laser scanner, but is also suitable for low speed indoor applications. It has a scan frequency of 10Hz. Use of this scanner proved troublesome as the software drivers for it were not compatible with other software used on the robot.

4.1.5. Laser Scanner

Ultimately a Sick LMS 200 laser scanner was used as the primary sensor on this robot. It has a range of 80m and 180°, with an angular resolution up to 0.25°, it has a scan rate of 75Hz. It also uses a serial RS232 interface to connect to the computer. Device driver software was specifically written for this scanner in the software used on the robot, which meant it worked seamlessly.

4.2. NorthStar localisation system

Although this should possibly have been in the preceding Sensors section, it was decided to include it separately as it is comprised of two separate sections. The NorthStar localisation system is a beacon based indoor system comprised of two distinct parts namely the projectors and the receiver board. The receiver sees projected light spots on a wall or roof and returns the calculated position to a computer.

4.2.1. NorthStar Receiver

The NorthStar receiver board is a proprietary sensor produced by Evolution Robotics. All the computation is encapsulated within the receiver. From an electronics point of view it is a very simple system to integrate, the receiver has its own electronics and returns position and direction information via a serial cable. The receiver requires a 9V power connection and an RS232 connection. Unfortunately most of the other systems in the vehicle run on 12V or 5V so a power supply had to be assembled specially. Due to availability a LM317 IC was used to make the simple 9V supply.

4.2.2. NorthStar Projector

The NorthStar projector is the beacon for the NorthStar system. The system was bought with only two projectors which results in a rather limited operating area. Added projectors were relatively expensive to buy but the schematics were provided for the projector boards.

The NorthStar Projector works by projecting a relatively large infrared spot onto the roof of the room that it is in. Because this spot is so large it results in significant

errors in the position readings supplied by the reciever. Another problem with the design is that it uses columnar Fresnel lenses which are very difficult to procure. A decision was taken to modify the projector to face down from the roof or wall, with light shining directly on the detector from the infrared LEDs. Five LEDs were used instead of four suggested in the NorthStar schematics to provide better coverage from each projector. The schematic diagram for this board is included in Figure B.1

4.3. Control board

A generic micro-controller board implementing an ATMEGA128 micro-controller was used to interface with the ultrasonic sensors, wheel encoders and bump sensors. This board interfaced with the SBC via a serial cable. The generic micro-controller board is a printed circuit board with all the necessary hardware on board for the micro-controller to work, so it has a crystal, power supply, serial converter and programming connector. Its input/output ports go to connectors so that it is possible to connect any desired hardware to these ports. There is also some development space on the board to mount any extra hardware. The boards have been manufactured for use on multiple projects in the CSIR MMM research group. In this case the added hardware was a number of connectors to gather all the data from the sensors. There are fourteen inputs for the bump sensors, eight outputs for the ultrasonic sensors and two inputs for each of the four wheel encoders.

4.4. Single Board Computer

A Single Board Computer (SBC) was used to do all the higher level computation required for this project. From an electronics point of view this computer is very similar to an ordinary desktop PC. The chosen computer was part of an the IEI ECW181AS3 embedded computer system. The system was chosen specifically for its industrial enclosure, solid-state hard drive and wide input DC power supply. The power supply can accept voltage from 9-36V DC, which is perfect for a battery powered device.

4.5. Wireless Connection

An integral part of this project was that the robot should have network connectivity to allow print users to print to the mobile robot from their desks. This made a wireless connection point necessary. The first attempt was to use a Netgear wireless dongle, which is connected to the computer via a USB port, however the interface proved to be troublesome in Ubuntu Linux, which was the chosen Operating System.

A wireless router was then connected to the SBC via an ethernet cable, and also powered by 12V DC. The robot then had its own wireless network to which users could connect, later this will need to be connected to the wider network to allow multiple users to connect. Another reason why wireless network connection was necessary was to control the robot while developing and testing software, it was possible to use Virtual Network Computing (VNC) to remotely control the SBC, which made it possible to allow the robot to move around while monitoring its progress from another, stationary, computer.

The wireless router which was available onsite is a 3Com Office Connect Wireless Access Point. Later in the project this 3Com access point was also replaced by a Linksys USB dongle, which is recommended for use on the CSIR wireless network. A network technician helped to set this up on the robot computer.

4.6. Printer

The printer that was sourced for this project had to have a relatively large paper tray, to prevent the need for frequent paper refills. The result was that unlike the SBC a suitable DC powered printer was not sourced as all the DC powered printers investigated were very small and for mobile use and did not have large paper trays. The printer that was purchased was a 220V AC powered printer, a HP Laserjet P2015, which interfaces to the SBC via a USB cable and does not require drivers for the Ubuntu operating system.

An attempt was made to disassemble the printer to bypass the rectifier and modify the printer to take DC, but modern printers are very tightly packed with electronics and mechanical hardware and there was too great a risk of destroying the brand new printer in the process. It was then decided to buy an inverter to convert DC to AC power, the inverter output was wired to the printer to provide power.

4.7. Inverter

Unfortunately the first inverter bought, which was supposed to have a peak power output of 1000W and a steady state power output of 500W, was not able to power the printer at startup, a larger 1000W output inverter was bought to deal with the larger power requirement. The inverter output was then wired to the mains power input of the printer, allowing the printer to effectively be run off battery power.

4.8. Motor Controllers

The motor controllers which were available at the beginning of this project were Roboteq AX2550 dual channel switch mode motor controllers. After examination they were found to be more than capable of what the platform required. The mecanum wheel platform requires individual speed control of every wheel, which meant that four motors needed to be controlled. The AX2250 controls 2 motors at a time, with multiple input modes, including standard hobbiest RC controls, 0-5V control or serial control. It is a high power controller supplying up to 4.8kW which is more than enough for this application. It can accept supply voltages of between 12 and 40 volts and can supply up to 120A. It uses a Pulse Width Modulated H-Bridge configuration, using Power MOSFET transistors to switch efficiently.

4.9. USB to Serial Port Converter

There were five devices used in this project that required the use of serial ports, namely, two motor controllers, the laser/IR scanner, the microcontroller board and the NorthStar receiver. The SBC has only 2 serial ports making it necessary to find a serial expander board. At first an eight port PC104 serial expander was investigated, but the support for this device in the Ubuntu operating system was inadequate. Another inhouse project had found a need for a USB to serial converter. As the USB protocol allows for a large number of ports to be branched off the original port it was possible to use this converter while using a four port USB hub to ensure there were enough ports for the other hardware items that needed to be connected to the SBC.

4.10. Extra USB Ports

The above USB to serial port converters need the full 500mA available from a single USB port as does the USB wireless dongle. It was decided to use the SBC expanded USB capability. Two extra USB ports were made available this way and solved the power problem.

4.11. Battery

Currently the system uses one 35Ah 12V battery to drive the motors, SBC, printer and all other electronic hardware. The micro board needed very good power filtering because the noise from the high current PWM motor drivers was found to interfere with the micro-controller. The battery is currently charged through connectors on the front panel of the platform, a later improvement will see the platform have a charging station at which it can remain while not in use.

4.12. Battery Monitor Board

A battery monitor board was designed for use on the platform, but it has not yet been implemented or tested, the circuit schematic can be viewed in FigureB.2. The circuit uses a Texas Instruments BQ2013H gas guage IC which monitors the current into and out of the batteries. It then interfaces with a micro-controller to transmit this information to the SBC which will be able to make decisions about whether to return to the planned recharge station or whether it can continue on towards the next user. There is some experimentation that needs to be completed with regards this board as the batteries are significantly larger than the recommended size for use with this IC, however this was taken into account and it is still believed that with the correct configuration this IC could be used to monitor the batteries on-board the vehicle.

4.13. Chapter Summary

This chapter has discussed all the electrical and electronic components required for use with this robotic platform. It also discussed the design work that was required to ensure that all the components interface correctly. Some circuit design was undertaken for use with specific sensors, including the sonar sensors, the bump sensors and a gas gauge battery monitor. Some care had to be taken when supplying power to the electronic components as the switching noise from the motor controllers interfered with the performance of some items. Datasheets of all the components used are available on the attached CD.

Chapter 5.

Software design

5.1. Introduction

The control of an autonomous mobile robot requires complex software to allow the robot to make the correct decision in all situations that may be encountered. This software exists on two levels, PC software and embedded software. The high level PC based control software enables the robot to answer general questions like "where should I go next?" and "how can I get around that obstacle". Low level embedded software determines things like motor speeds to achieve a certain vehicle velocity, and the interpretation of data received from sensors.

In this case the higher level software exists on the Single Board Computer (SBC) described in Chapter 4. Player is widely used by other researchers worldwide and is very portable across different hardware platforms, making it a strong choice for robot control. For these and other reasons a decision was taken within the CSIR MMM research group to standardise on robot control software using the Player/Stage robot server. This made its selection for use with this project easy.

Some of the sensing and actuation hardware was able to interface directly to the PC and Player. Those items that could not interface directly had to interface via the microcontroller board described in Section 4.3. Embedded software was written to run on this microcontroller board to allow the data to be captured and transmitted in the correct format.

This chapter will describe the software that was implemented, on the computer, the micro-controller and any other systems that were required during testing. It will start

by giving some more relevant background to the Player Project and then giving more detail about the drivers that were written for this project and the Player components that were implemented. It will then discuss the embedded software that was written to gather data from some specific sensors. Finally it will discuss the software that was written for a specific test set-up, namely the wheel tester.

5.2. Embedded Software

Embedded code was written in the C Programming language to capture data from the four wheel encoders, the bump sensors and the eight sonar sensors. This data was then sent via serial cable to the SBC where it was used by the "WSB" (Wheel encoder Sonar Bumper) Player driver. This code was written to run on the Atmel ATmega 128 microprocessor [15]. Embedded code was also written for the wheel tester set-up to capture data from the two wheel encoders and send it to the test control program, discussed later in this chapter.

5.3. The Player Project

Player is an open source robot control server. It allows messages to be passed back and forth from client programs to device drivers via specific interfaces. Consider Figure 5.1, it illustrates how the Player server works. A client program passes commands to the server and receives data about relevant sensors and actuators. The server processes these commands and forwards them, configured to meet the interface requirements, to the relevant device driver. The device drivers, when possible, communicate directly with the sensors and actuators.

Player must run on Linux, and Ubuntu was chosen as the preferred Linux distribution. This was due to its ease of use and because it is a CSIR wide installation. This made installation of the Player server a lot easier for a user with minimal prior Linux experience. Ubuntu has a package installer called Synaptic, which allows one to select which pre-compiled package to install. Synaptic then checks which other packages are necessary to run the selected item and installs them at the same time. With any other Linux distribution it would be necessary to manually source each of these packages, then compile and install them which can become quite difficult. What's more with Player, using any

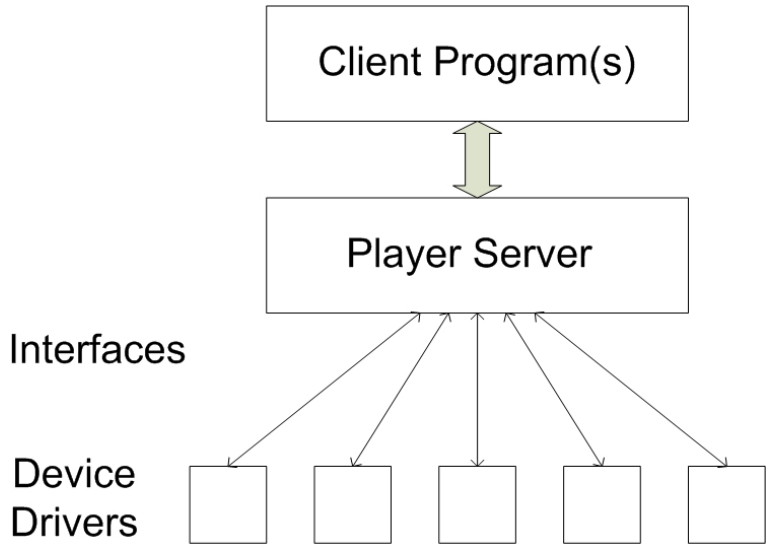

Figure 5.1.: The simplified Player architecture

other Linux distribution, it is necessary to compile from source code, including those parts that are desired most. For a Linux amateur the ease of Ubuntu made it the logical choice.

There is a downside that was realised late in the project, this is the unavailability of source code, which must be downloaded separately if necessary. If the source was compiled manually it would be available to be viewed. Being able to view the source code was necessary at times to better understand how the Player server worked.

5.3.1. Player Abstractions

In a mechatronic system, a device is a piece of hardware, a sensor or actuator, that is controlled or communicated with using electrical signals [12]. A device driver is a piece of software that performs two basic tasks. Firstly it must communicate with or control a device and secondly it must interface correctly with whichever piece of software requires the use of that device [23]. The Player Project helps robot designers develop these drivers in a structured manner.

Player is an easily reusable piece of code because the abstractions it uses to allow the code to be generic and portable [61]. Player splits these into four separate abstractions, namely:

- the "Player Abstract Device Interface (PADI)"

- the “Message Protocol”
- the “Transport Mechanism”
- and the “Implementation”

These abstractions will be discussed below [61].

Player Abstract Device Interface

Player borrows from Unix based Operating Systems, using the “device-as-file” abstraction and the “driver/interface” model. The device-as-file abstraction allows the interface to a device to mimic files with open, close, read, write and ioctrl (Input, Output control) commands. This makes writing the interfaces to devices much easier and clearer [61].

The driver/interface model, builds on the above by allowing similar devices to be grouped together by functionality. The interface specifies exactly how the data to and from a specific device group must be formatted. The piece of code that converts the data sent to or received from a device is called a device driver. Once a driver is written to comply to a specific interface, the higher level code is completely isolated from the specific control signals to and from the device [61].

Both of the above abstractions make it possible to write control code that works on various different robots. For example, two similar robots, having different motor controllers, could be controlled with the same client program, once the correct drivers have been written.

The Message Protocol

If data must be sent to, or received from, a device, the Player server will send a request message to that device. Player specifies how messages should be compiled and how they can be accessed. A number of predefined message handling functions have been written to deal with messages. These messages must comply with the specific interface structures of the driver to which they are being sent [61].

The Transport Mechanism

Although other transport mechanisms can be used with Player, this project used the standard TCP (Transmission Control Protocol) client/server method. This method borrows from internet data transmission methods and defines the robot control code as a client which connects to the Player server via a TCP socket. This allows the client program to be written in any programming language that supports TCP sockets and allows the client to be run from a remote computer if desired. [61]

Implementation

All of the above mechanisms are implemented in the Player server. When the Player server is started a configuration file is specified. This file determines which drivers should be run, and which interfaces they should use. This interface defines what kind of messages the driver can send and receive via the server. The server then communicates with the client program via a TCP socket tranport mechanism [61]. The drivers and clients will be discussed in more detail in the next section.

5.4. Player drivers and clients

The Player Project makes writing robot control code relatively easy. But it is still necessary to learn how to use it, and then to write the necessary code that will allow it to be used with the current project. A client program had to be written to test the robot functionality. The hardware used for this project was mostly custom and not yet supported by the Player community. This required some driver code, which ended up being the bulk of the Player code written.

5.4.1. Motion Driver

The motion driver was written to control the wheels according to the model derived in Section 3.5. A large amount of time and effort went into getting this code to work and some simple client-side code had to be written concurrently to allow testing. While Player may be relatively easy for an experienced programmer, it was found that for a comparative novice there was very little easily available and understandable information.

The motion driver code was written to use a **position_2d** interface and accepts only velocity commands from client programs (or abstract drivers). On the hardware side it must communicate with the Roboteq motor controllers via two serial ports. It regularly publishes velocity data to the Player server which forwards this to any interested parties, including client programs and other drivers. A further improvement to the code will be to add direct wheel encoder feedback, however this has not yet been implemented.

5.4.2. Micro-controller driver

The wheel encoders, ultrasonic sensors and bumpers, described in Section 4.1 are accessed via an Atmel AVR micro-controller. This micro has a single serial interface making it necessary to write a single driver to communicate with all the sensors connected to the micro. Wheel encoder data is usually used directly by the motion control driver, as such Player does not have a specific wheel encoder interface and an **actarray** interface was used. The ultrasonic sensors used the **sonar** interface, and the bumpers used the **bumper** interface. This driver publishes relevant messages to the Player server using all three of these interfaces [16].

5.4.3. NorthStar driver

NorthStar, described in Section 4.2 is similar to a GPS in that it gives position and heading information. It made sense initially to use the **GPS** interface to write this driver, but later the generic **opaque** interface was implemented. There was some mismatch experienced in the co-ordinate systems implemented in the motion driver and the NorthStar driver. This required some rework of both drivers, but was a good lesson learned in communications between software developers [52].

5.4.4. Abstract Player drivers

Player drivers can also be written to perform higher level functions. These drivers generally pass messages to and from other drivers. They are generally used to implement complex algorithms which are commonly used amongst the robotics community. Good examples of such algorithms are the obstacle avoidance, navigation and path planning algorithms that are available, and were implemented in this project.

Vector Field Histogram

The Vector Field Histogram obstacle avoidance algorithm works using a polar range finding sensor system, which may be a laser range finder or a number of ultrasonic sensors arranged around a mobile robot [14]. In this project ultrasonic sensors were used initially but were ultimately replaced in favour of the Sick laser scanner.

The algorithm uses data from the sensor(s) to generate a histogram grid, which is a grid of cells a predefined distance from the robot. Each data point increments the value of the grid cell which intersects the centreline of the sensor and the range finding. While this is quite a slow method it is claimed to be quite robust. The grid moves as the robot moves through the environment, it has been described as a moving window, allowing data that is no longer valid to the robots current position to be discarded.

Next the algorithm uses the histogram grid to create a polar histogram of the robot's surroundings. It does this by mapping the currently active histogram regions onto the polar histogram. The polar histogram is comprised of a number of sectors arrange around a predefined vehicle centre point. Once it knows where the obstacles are, it can decide on the best path which the robot should take to avoid these objects in a direction that is closest to the goal. The Vector Field Histogram is programmed into one of Player's abstract drivers called the "vfh" driver [14] [7]. The implementation of this obstacle avoidance method was relatively easily. It worked better in larger environments than narrow corridors.

Nearness Diagram

Nearness Diagram obstacle avoidance uses a different method. It defines a security region around itself and then uses range findings to make decisions based on the decision tree shown in Figure 5.2. Criterion 1 is applied first to decide on the proximity of objects. If there are objects in the security region then Criterion 2 is applied. This criterion has two possible outcomes, objects within the security region on both sides of the gap closest to the goal, and objects on only one side of the gap closest to the goal. The algorithm then plans motion to avoid the obstacles while moving closer to the goal.

If Criterion 1 determines that there are no objects in the security region, then the robot either applies Criterion 3 or Criterion 4. Criterion 3 means that there are no major obstacles between the robot and the goal. Criterion 4 means that there are obstacles

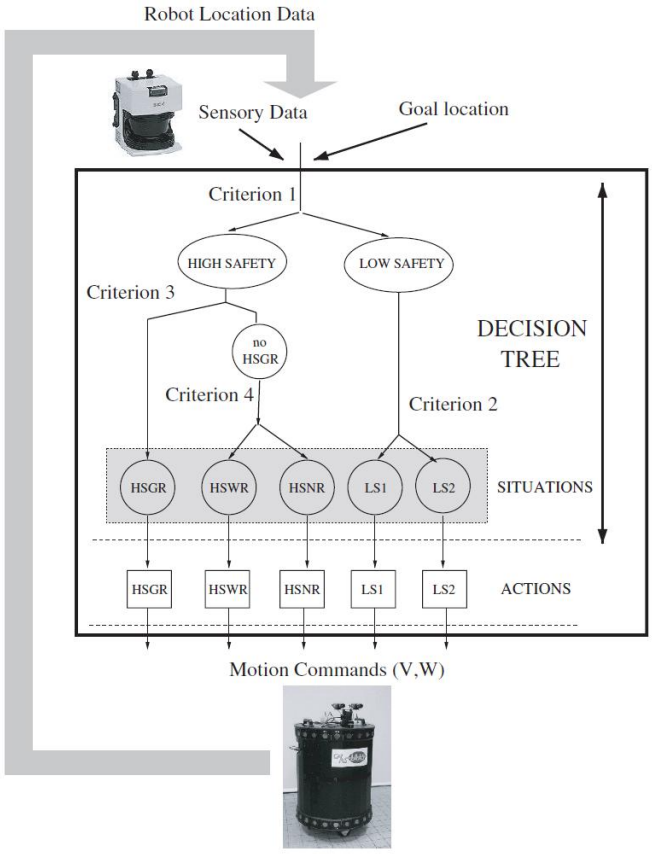

Figure 5.2.: The Nearness Diagram Decision Tree [43]

but they are outside the security area and there are obstacles between the robot and the goal. The two separate results of Criterion 4 are if there is a wide or narrow gap to move through towards the goal. Different actions are taken by the robot depending on which conclusion about the situation is reached [43], [44].

The Nearness Diagram algorithm is also programmed as one of Player's abstract drivers, called the "nd" driver [6]. Problems were experienced with the implementation of thes driver. When the source code was opened to investigate why it didn't behave as expected, the comments were found to be in Spanish, this code was then abandoned as a possible algorithm and tests were performed only with the "vfh" algorithm.

Wavefront

The Player "wavefront" path planning driver is based on the wavefront propagation method. As discussed in Section 2.6.1, this method uses a grid based map to allocate a value to each cell of the grid depending on the distance from the source of the waves, in

this case the mobile robot. It is then possible to trace the shortest route back to the source following the steepest value gradient. This is used to find the fastest route to a goal. If sensor data is received that prevents the robot from following this path then it is necessary to use an obstacle avoidance algorithm and replan a path to the goal [8], [39].

Wavefront accepts data from a "map" interface which it uses for the grid method described above. It also accepts data from a "planner" interface which allows a client proxy to provide goal information to the wavefront driver itself. The driver then sends data to the hardware via a "position_2d" interface. This driver worked well, but it was initially unclear whether the map was being correctly implemented. Further testing proved that the map does improve the robot's performance.

Map

The Wavefront driver requires a map to plan how to get from a start position to a goal position. This plan was drawn from building plans and represented as a Portable Network Graphics (.png) file. The driver looks for boundaries between dark and light areas, which represent walls and then plans a path from where the robot currently is on the map to the goal position.

5.4.5. Client Programs

An initial client program had to be written to check the functionality of the motion driver. This was a simple "goto" client, which took x,y and rotational speed commands. This allowed the correct functioning of the omnidirectional wheel control to be demonstrated. It was originally written with the **player_cpp** library, but was later changed to the **player_c** library as this allowed better control.

Player provides a simple Application Programming Interface (API) in a number of programming languages, including C,C++, Python and Java. This project utilised the C and C++ libraries. They allow simple proxies to be created allowing communication with each of the interfaces being used. In this sense a proxy allows a client to connect to an interface through the Player server. Once this is applied there are simple class accessor functions which allow read and write commands to and from the interfaces and via the devices. For example, to control the motion controller directly it would be necessary to create a **position_2d_proxy** and then use a "SetSpeed()" command to set the speed of

the vehicle. The Player server would send a position2d velocity command to the motion driver, which would then send the appropriate command to the motion controller.

The final client was written to deliver printouts to network users. The SBC runs a print server for the on board printer. When an item is printed by a network user, the job is sent wirelessly to the on board print server. The co-ordinates of the associated user is looked up. The client program subscribes to a planner proxy and sends the co-ordinates to this planner, which then decides on a path to get to the user who has just made a printout. The planner sends a waypoint command message to the "vfh" driver which decides if there are obstacles in the way, and moves the robot via the motion driver, towards the goal.

5.5. Running Player

The Player executable is run with a configuration (config) file as a parameter. This config file tells the Player server which drivers should be run and what interfaces they provide or require. A typical config file would include a driver controlling the motion of the robot via a **position_2d** interface. It would also have a driver communicating with an obstacle avoidance or navigation sensor, providing an interface like **sonar**.

If obstacle avoidance is not performed by the client program then a driver like vfh would be used for this purpose and would "provide" one **position_2d** interface and "require" one **position_2d** interface. This is so that a client program, giving goal commands, can interface with the obstacle avoidance driver as if it was the motion controller.

Once the Player server is running it is necessary to run a client program which connects to the server via a TCP socket as explained in Section 5.3.1. A C or C++ client then uses proxies to access data from a specific driver, ultimately allowing it to control the overall motion of the robot.

5.6. Software to run the wheel tester

The software for the wheel tester, described in Section 3.4 was written in two parts. The first part is the embedded code written in C, running on an ATmega 88 microprocessor,

which interprets quadrature encoder data received from the test wheel and the treadmill, and sends it via a serial port to a PC running Windows. A PC program was written to gather serial data from the micro-controller and the inline digital ammeter, a Fluke 8845A. This is then compiled into a database entry running on MySQL server. [53]

5.7. Laser interface software

Testing required the use of a laser scanner to provide an external position reference as comparison to the onboard data gathered by the robot. This software simply gathered scan data from a Sick laser scanner and wrote it to a data file for post processing. [54]

5.8. Laser data post processing

Once laser data was captured from a motion test it was necessary to extract the relevant position data from all the scans. As will be expanded on in Chapter 6, a plate was mounted on the robot to provide a surface which the laser scanner use as position reference. Each scan must then be examined for a number of points which are in a straight line, this could range from very few points when the robot is at an angle to the scanner, to very many when it is side on. The program then plots this data as arrows indicating position and direction. [55]

5.9. Chapter Summary

This section has discussed the issues relating to programming a mobile robot, including the navigation algorithms that have been implemented. There are generally two types of program implemented on the robot, embedded and PC based. The embedded software was written in the C programming Language. The PC based software was written to interface with the Player server, either as device drivers or as client programs. This chapter also discussed software required to run tests and post process the data resulting from these tests.

Chapter 6.

Testing and Validation

This chapter will describe the tests that were completed to validate the objectives of the project. The first set of tests performed were to confirm that the model, developed in Section 3.5, behaves as predicted. The next set of tests show that the implementation of this model allows omnidirectional motion and that the platform can move in this manner. Tests were then performed to show the obstacle avoidance capability of the platform and finally to show the functionality of the path planning algorithm.

A number of motion tests were performed to determine the functionality of the platform. It was decided to use a SICK LMS 200 laser scanner to perform data capture during all motion tests. A plate was attached to the vehicle to allow the laser scanner to pick up its position. Data was then post-processed to produce position and direction data, which was then mapped to show the motion of the vehicle.

6.1. Testing the Wheel Model

The Mecanum wheel was tested under load using the test platform described in Section 3.4. This platform allowed the wheel to be driven with exactly the same motor that is used on the platform. It allowed a number of weights to be loaded onto the wheel to simulate the loading conditions experienced on the ground, and to generate enough traction for the wheel to turn the treadmill. The wheel was then lowered onto the treadmill. The treadmill was mounted on a turntable which can vary the angle made between the belt and the rotational plane of the wheel. Quadrature encoders then measured the rotational speed of the wheel and the treadmill. This data was then compared to the values predicted in the model.

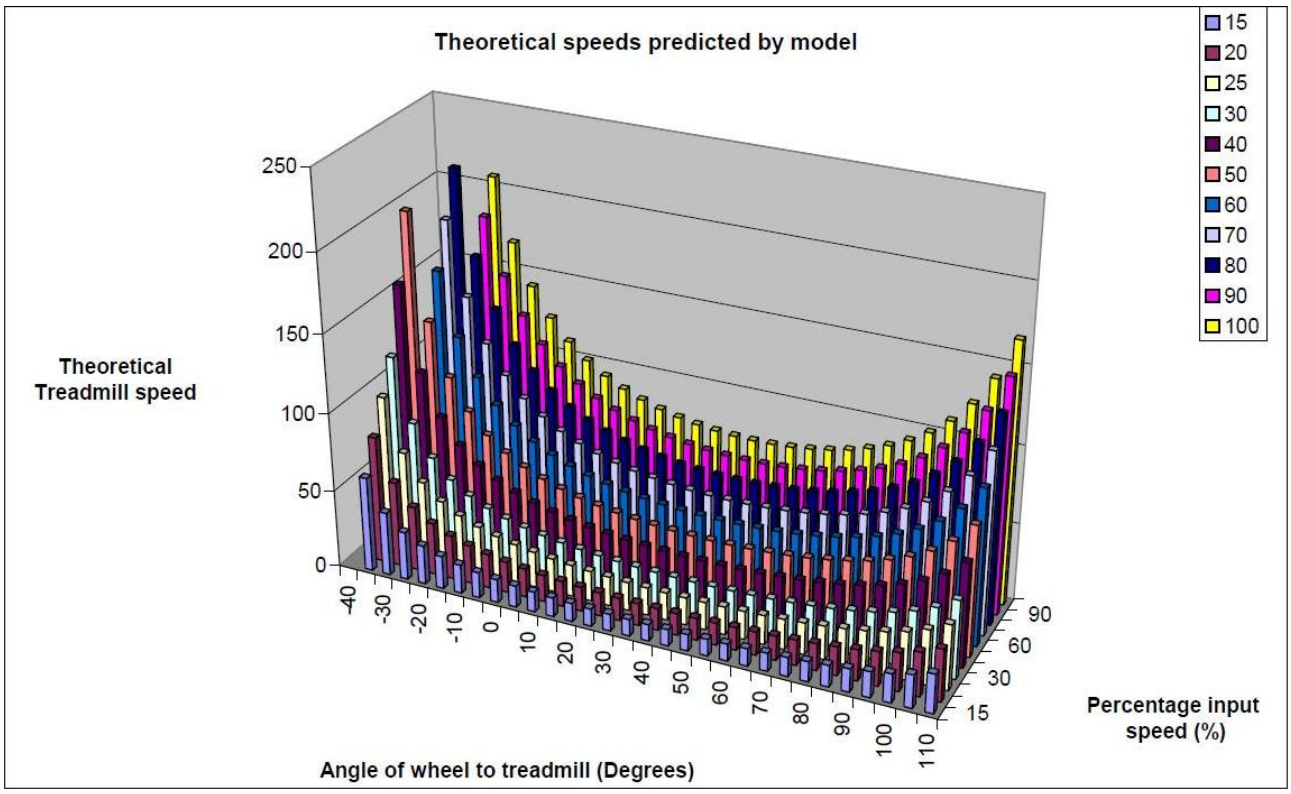

Figure 6.1.: Theoretical resultant treadmill speeds

The resulting data is three dimensional, as it varies according to the input speed and the input angle of the wheel to the treadmill. This is illustrated in Figure 6.1, which gives the theoretical results of the application of equation (3.7) developed in Section 3.5, this is the example to which the results should be matched. Figure 6.1 is a discrete representation of a surfaced defined by the relationship between input speed, wheel angle and the resulting treadmill speed. Some of the data points in Figure 6.1 were removed to view the data more closely as the defining equation tends to infinity as $\cos(\theta - \alpha) \to 0$.

6.1.1. Equipment and test set-up

To run the wheel tests, a significant amount of equipment was employed this included the wheel test jig discussed in Section 3.4, motor controllers, meters and a PC. A full list of this equipment is provided in Appendix D.1. A set-up procedure was followed so that the tests were repeated correctly for each iteration. For the full set-up procedure please see Appendix D.2. On the attached CD there is a video taken while the tester was running.

6.1.2. Test Method

The PC software running the wheel tests allows one to set a number of parameters. These were included to ensure the data capture was repeatable and accurate. All the data was sent to a MySQL server where it could be retrieved for later analysis. The

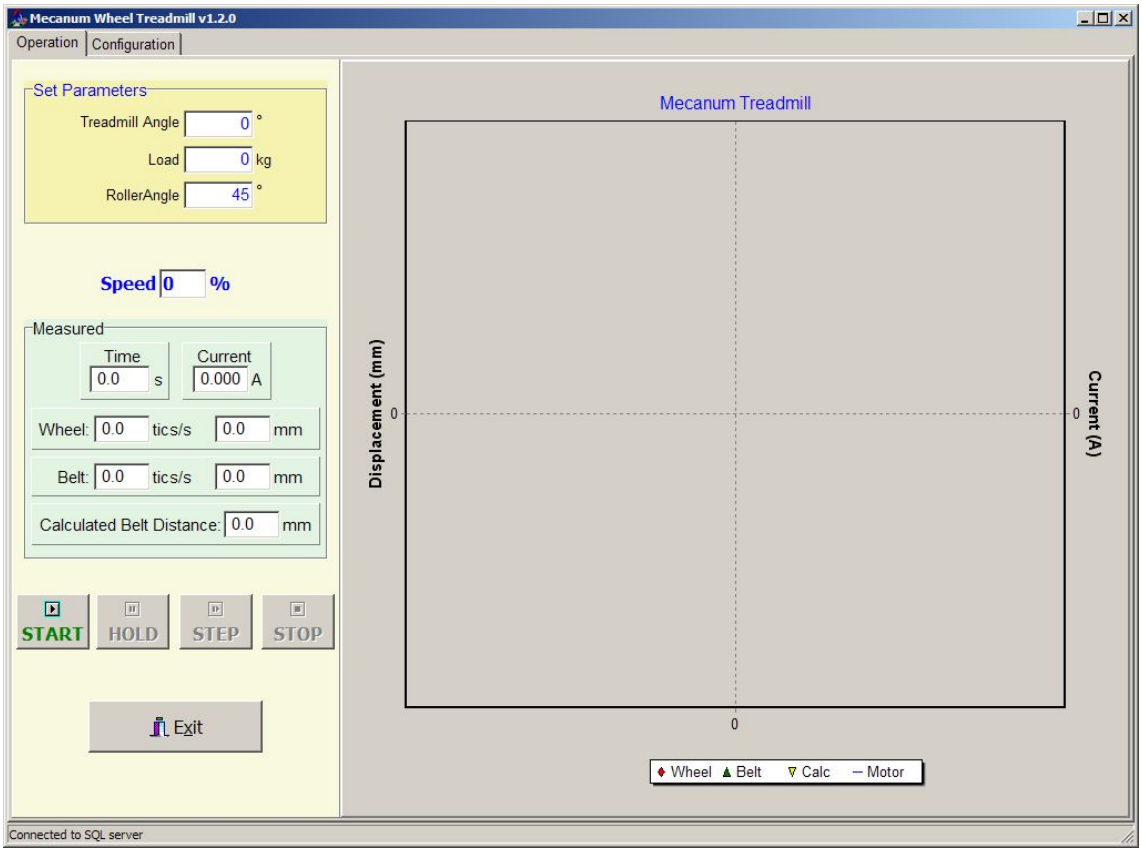

Figure 6.2.: First tab of the Mecatread program

first tab of the program, seen in Figure 6.2, gives real-time feedback of the current test being performed. The second tab, seen in Figure 6.3, allows Logging Parameters and Configuration Variables to be set for each test. Both these items will be discussed in the following sections.

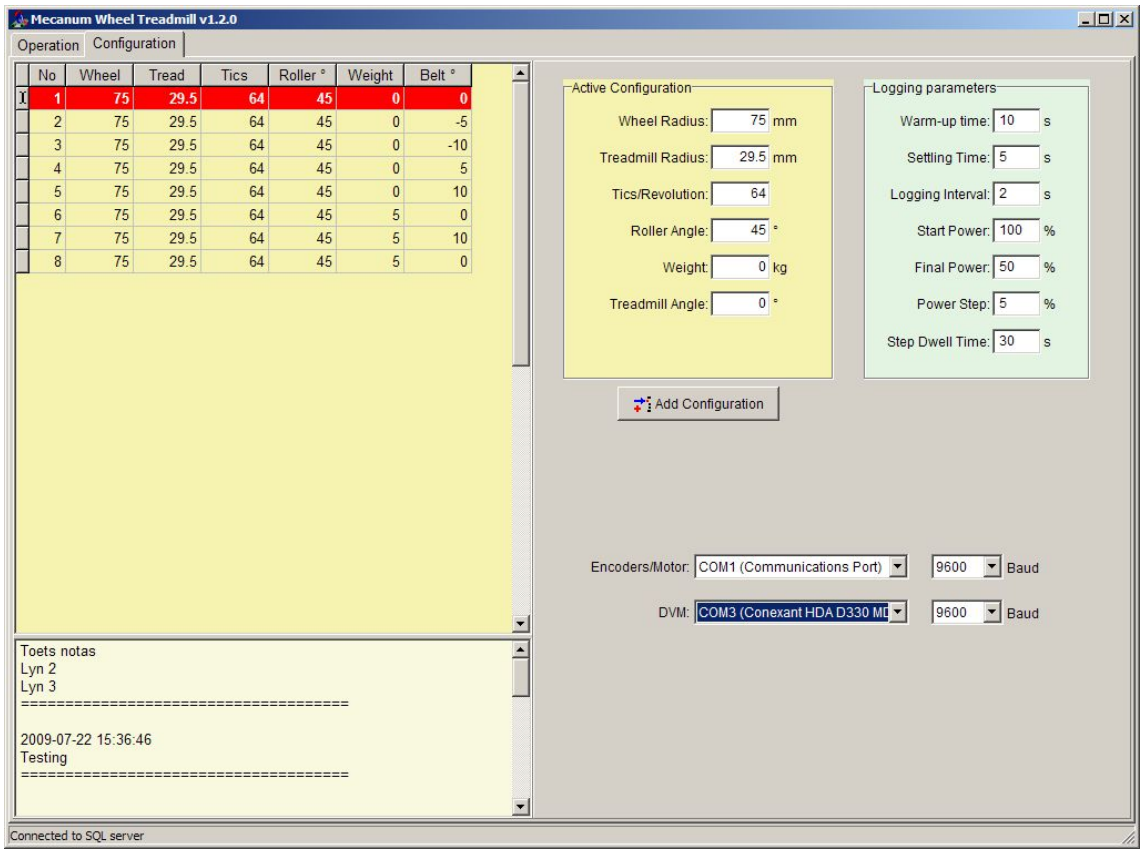

Figure 6.3.: Second tab of the Mecatread program

Logging Parameters

Logging Parameters define how the test is run, how much time is spent at each voltage level and what the difference between each voltage level is.

- Warm-up time (s) - This determines how much time is spent warming up the belt and all other equipment before each test
- Settling time (s) - This is a period of time spent between each voltage step that allows any transients to die away before data is logged
- Logging interval (s) - The embedded controller captures and averages data from the wheel encoder continuously, the logging interval defines how frequently this data is logged to the database
- Step dwell time (s) - This defines the time spent at a particular voltage level

- Start Power (%) - - The motor controller varies the average voltage applied to the motor from 0 to 12V using a PWM signal. The input to the controller is a percentage of the maximum possible voltage, 12 volts, applied to the motor. Start power determines at what percentage the testing should start
- Final Power (%) - This is the percentage where the test should end
- Power Step (%) - This is an absolute value percentage step, which allows the final voltage to be greater or smaller than the start power. These three values will determine how many steps there are in a full test run.

Configuration Variables

Configuration Variables are the details about how a test is set up, most of these variables are fixed in this case, but for possible future tests or changes to the test equipment all these items were made variable in the software.

- Wheel Radius - The radius of the wheel running on the treadmill, in this case a constant
- Treadmill Radius - The radius of the treadmill, including the belt, running over the roller with the quadrature encoder which is constant
- Ticks/Revolution - This describes the resolution of the quadrature encoder and is constant
- Roller Angle - This is the angle the wheel rollers make with the axis of the wheel. The tester was designed to test other composite wheels, which may have different roller angles, in this case it is 45°.
- Weight - This describes the loading on the wheel, as more weight is added there is more traction between the wheel and treadmill. Tests were only performed at 5 kg, but later tests could include larger weights.
- Treadmill Angle - This describes the angle the treadmill makes with the wheel rotational plane. This is varied in 5° increments to carry out the tests.

Once the correct configuration and test parameters have been selected and a test is started, the program automatically varies the voltage supplied to the motor from a maximum velocity to a minimum velocity or vice versa. During initial testing it was

found that the belt used on the treadmill performed better when warm, so an initial warm-up period was included in the test and the wheel speed was varied from fast to slow.

Voltage to the motor was stepped down in 5% increments for each angle setting of the treadmill. Once the motor no longer provided enough torque to turn the wheel on the treadmill the test was stopped and the angle of the treadmill was changed. The treadmill angles were changed in 5° increments, from -40 to 110 degrees, any further angle changes caused damage to the belt. For each data point the time, treadmill displacement and wheel displacement were logged to the database.

6.1.3. Results

The results of the wheel tests are shown below in Table 6.1, this data is captured directly from the wheel encoders mounted on the treadmill. This can then be compared with the data calculated using Equation (3.7) and the corresponding input wheel speed measured with the wheel encoder mounted on the wheel, shown in Table 6.2. Correlation co-efficients between these values, for each of the angles tested, are calculated and shown in Table 6.3. The overall correlation co-efficient for these two data sets is 0.82.

Table 6.1.: Captured Treadmill Speed Data

	Percentage Motor Power										
Angle	50%	55%	60%	65%	70%	75%	80%	85%	90%	95%	100%
-30	28.7	44.4	63.7	78.7	117.3	151.1	170.8	170.8	197.4	193.9	149.7
-25	54.9	91.7	112.1	157.5	201.8	253.5	309.2	309.2	360.2	414.9	181.4
-20	98.3	116.9	152.8	170.2	197.3	247.9	301.9	301.9	356.1	417.7	290.6
-15	83.5	122.0	186.0	236.1	269.3	351.7	421.2	421.2	553.3	598.9	438.9
-10	113.2	162.0	229.8	292.1	347.9	417.1	494.2	494.2	655.6	690.2	586.9
-5	113.4	169.3	232.3	295.8	366.9	433.2	511.0	511.0	658.4	714.2	688.2
0	112.4	170.3	223.9	286.8	352.1	415.6	477.0	477.0	641.6	687.1	721.8
5	204.2	240.4	289.3	363.2	411.7	487.5	560.3	560.3	709.6	756.3	695.5
10	195.3	210.1	259.7	327.9	380.6	452.3	525.4	525.4	690.3	713.2	777.0
15	134.6	169.0	212.2	280.8	319.4	385.7	444.6	444.6	591.4	614.9	684.4
20	134.6	169.0	212.2	280.8	319.4	385.7	444.6	444.6	591.4	614.9	684.4
25	99.0	133.1	179.2	225.0	261.2	312.1	372.6	372.6	537.8	577.6	623.2
30	98.8	124.5	160.9	202.7	243.1	284.0	320.9	320.9	429.2	461.5	582.2

Table 6.2.: Calculated Treadmill Data

	Percentage Motor Power										
Angle	50%	55%	60%	65%	70%	75%	80%	85%	90%	95%	100%
-30	41.8	64.4	95.5	129.1	184.6	229.9	290.7	290.7	419.4	464.0	445.2
-25	64.1	106.8	134.6	191.5	237.0	303.7	382.0	382.0	500.8	545.6	472.8
-20	102.1	134.0	166.3	208.6	246.7	304.3	364.1	364.1	499.4	544.6	521.5
-15	77.4	115.6	170.6	224.2	270.2	326.7	392.9	392.9	529.5	566.1	548.2
-10	95.2	137.8	193.4	254.1	301.8	368.4	438.6	438.6	584.3	611.5	571.7
-5	100.0	150.3	206.3	264.8	332.0	393.6	461.9	461.9	597.4	643.1	617.8
0	112.7	170.4	226.0	291.5	359.5	427.1	505.0	505.0	667.3	723.5	645.9
5	242.7	285.6	344.5	431.3	489.8	579.8	668.3	668.3	848.4	907.2	733.3
10	280.6	301.0	372.5	471.0	546.8	649.7	755.5	755.5	997.4	1031.7	930.4
15	102.1	134.0	166.3	208.6	246.7	304.3	364.1	364.1	499.4	544.6	521.5
20	112.7	170.4	226.0	291.5	359.5	427.1	505.0	505.0	667.3	723.5	645.9
25	242.7	285.6	344.5	431.3	489.8	579.8	668.3	668.3	848.4	907.2	733.3
30	280.6	301.0	372.5	471.0	546.8	649.7	755.5	755.5	997.4	1031.7	930.4

Table 6.3.: Correlation co-efficients (r) for each test angle

Angle	-30	-25	-20	-15	-10	-5	0	5	10	15	20	25	30
r	0.91	0.89	0.95	0.97	1.00	1.00	0.99	0.99	0.98	0.99	0.98	0.95	0.94

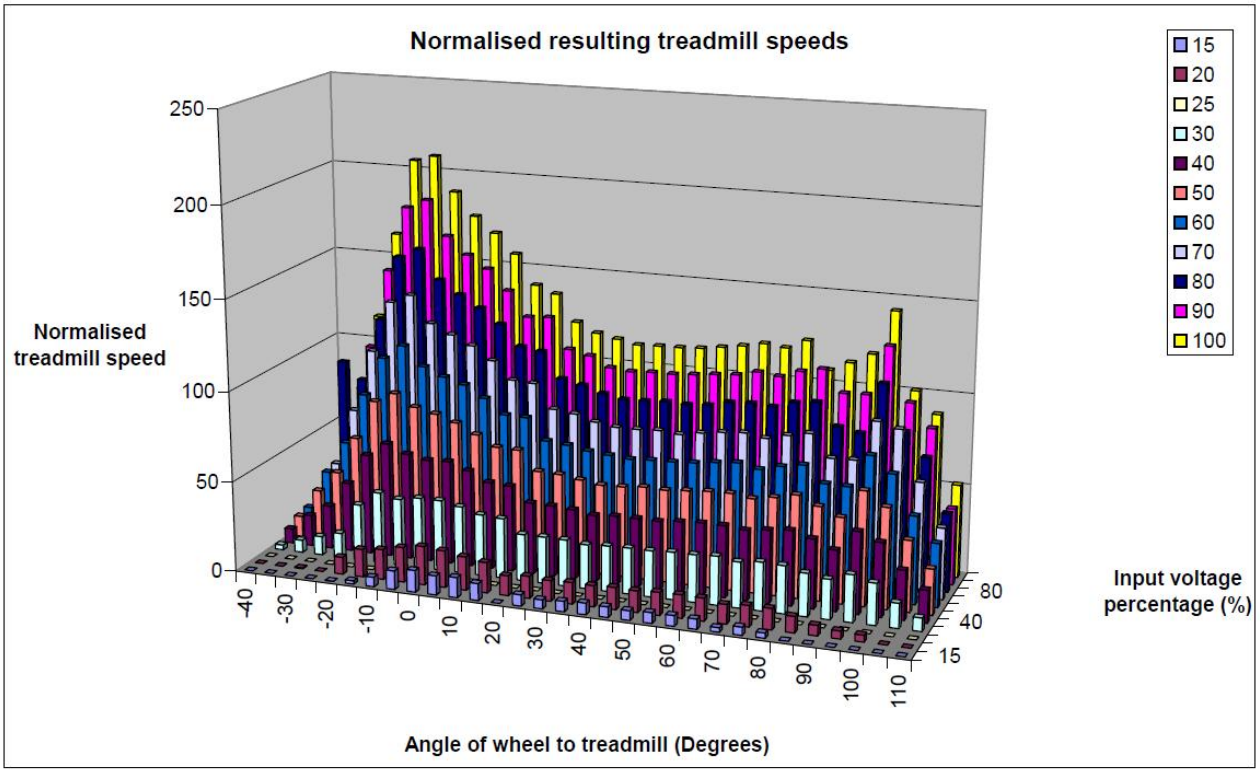

Figure 6.4.: Results of wheel tests

The data has been summarised in Figure 6.4. The columns in the figure represent discrete points on a surface, the surface produced by the response of the treadmill depending on the percentage of full motor input voltage. This is similar to the surface represented by Figure 6.1, but due to the errors listed below it is not identical.

6.1.4. Errors

- Control variable - The input control variable for these tests was the average voltage supplied to the motor, by the PWM controller. As a result the velocity for each average voltage level changes depending on the load applied to the motor. A better way to run the tests would be to have velocity feedback on the motors to ensure consistent velocity inputs.
- Belt stiffness - the stiffness of the belt is identified as a variable, as the belt heats up it rotates easier on the treadmill. An attempt to mitigate this effect was to run the belt for some time before recording test data.
- Belt manufacture - the belt is not perfectly continuous, the joint in the belt creates a tendency for the belt to be oval in shape, this results in differences in resistance to rotation as the belt moves over the rollers.
- Wheel slip - Especially at angles where the rollers of the wheel are moving near perpendicular to the direction of motion of the treadmill, the wheel has a tendency to slip, which will result in errors in the results.
- Belt rub - Although much effort was made to prevent the belt rubbing on the treadmill walls, at times it still did result in significant friction on the belt.

6.1.5. Conclusion

The resulting surface seen in Figure 6.4 is for a single wheel rolling on a treadmill. When compared to the ideal case shown in Figure 6.1 it is apparent that as the treadmill angle gets closer to the asymptotes, at -45° and +135°, the actual readings diverge from the ideal model. This is because the rollers are tending towards moving perpendicular to the belt surface and cannot provide any force in the direction of motion of the belt. This behaviour can be compared to screwing a screw into wood, if one imagines that the pitch on consecutive screws get longer and longer, it will be more and more difficult to screw into the wood. Similarly it gets more and more difficult to rotate the wheel as the roller axis gets closer to perpendicular to the belt.

In a complete, four wheel system, when the direction of motion is perpendicular to a particular wheel's roller axis, there is force applied to this wheel by other wheels oriented in different directions. This will result in displacement of the wheel with no rotation of

the wheel. In other words this is where the wheel rollers will be free-wheeling. It is not possible to simulate this with the tester used in this experiment. It is concluded that the derived formula for motion of a Mecanum wheel behaves as expected when compared to gathered real data.

6.1.6. Recommendations

A better way to have run the tests would have been to power the treadmill at a predefined speed and measure the resultant motion of the wheel. This was only recognised after the test platform was constructed. If the treadmill were powered instead of the wheel, at -45° there would be very little force transfer from the treadmill to the wheel, it would only be proportional to the amount of rolling friction created by the roller currently in contact with the belt.

6.2. Motion Test Equipment and Set-up

The tests discussed in Sections 6.3 and 6.4 required the same equipment and test set-up. To run the tests using the robot a few items of equipment needed to be available. These included an external Sick laser scanner and a Windows PC. For a full list of equipment used in these tests please see Appendix D.3.

The robot must be started up and a start-up check was performed to ensure proper functionality, this procedure can be found in Appendix D.4. With start-up complete, the system is ready to run Player and any client program that has been written to control the robot. The client program will define one of the specific tests that were performed.

Before testing could commence it was necessary to set up the external laser scanner. This scanner captured position data from the robot as it moved. This data was used as a comparison to the on board position data that the robot generated. The full laser set-up procedure is included in Appendix D.5. Once set up was complete, the PC was ready to log range data supplied by the laser scanner.

6.3. Testing the motion control software

A locus, described in [62] and shown in Figure 6.5, was used to test the motion driver and show that it is capable of omnidirectional motion. This includes forward and sideways motion, rotation on the spot, rotation about another centre and motion in any desired direction.

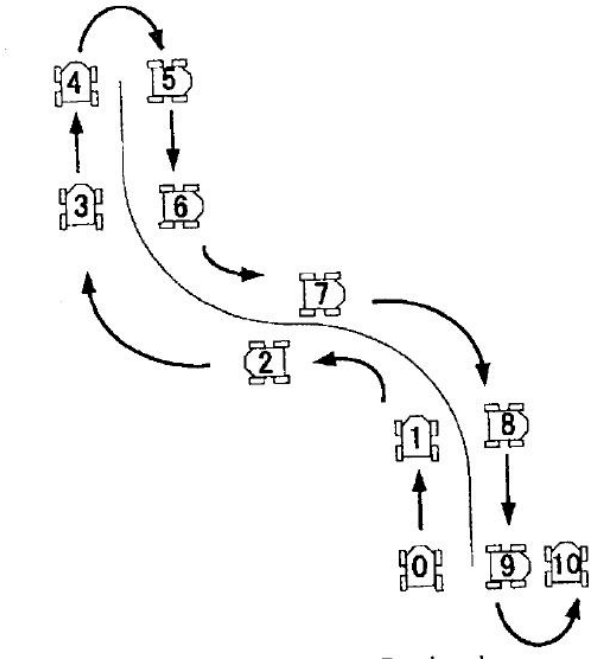

Figure 6.5.: Benchmark Locus [62]

Motion Number	Motion Description
0	Forward motion
1	Left hand turn off centre
2	Right hand turn off centre
3	Forward motion
4	Right hand turn around centre
5	Right hand straight line motion
6	Transition from right hand motion to forward motion
7	Transition from forward motion to right hand motion
8	Right hand straight line motion
9	Left hand turn on centre

Table 6.4.: Locus Motions

Table 6.4 gives a description of which motion is expected at which numbered point in Figure 6.5.

6.3.1. Test Method

The platform was set up to run a Player client program designed specifically to follow the locus shown. There was no motion feedback, this was a purely open-loop motion test. A vertical piece of sheet metal was attached laterally onto the platform, as seen in Figure 6.6 to give a large surface area for the laser scanner to gather range data. The robot and Laser Scanner were set up according to Section 6.2. Via VNC (Virtual Network Computing) the Player server was run using the "Sebs-current.cfg" config file. The system is now ready to run the locus tests, the procedure to run these tests follows:

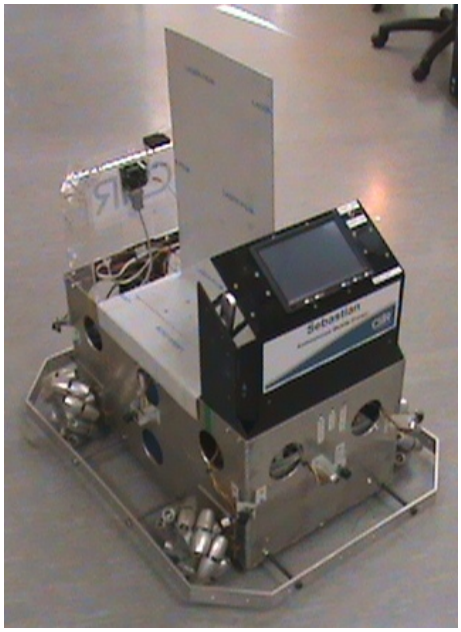

Figure 6.6.: Platform with position plate attached

- Run "robot-player Sebs-current.cfg" in one PuTTY communication terminal
- Run the SickLMS.exe program and select "stream"
- Select an appropriate file name and press start
- Run the "./locus_test" Player client program in the other PuTTY terminal

The robot will now move along the locus described in Figure 6.5.

6.3.2. Results

The laser scanner returned successive scans at the chosen scan rate. These scans are made up of range findings at a chosen angular resolution, in this case 1°. To remove the relevant data from the set, only the data between an observed maximum and minimum angle was considered, anything outside these angles was discarded. Only relevant data remained. Further range and angular restrictions were imposed on the data, effectively extracting only the motion data.

To find the centre of the plate the average angle and range of each filtered scan was taken, using $x = r\cos(\theta)$ and $y = r\sin(\theta)$, each centre position of the vehicle could then be plotted as it followed the locus. To find the direction of the robot motion, a line was fitted through each set of data points, which resulted in a gradient(m) that was converted into an angle where $\theta = \arctan(m)$. While the angle of the plate could be found this could be 180° out of phase with the actual heading of the vehicle.

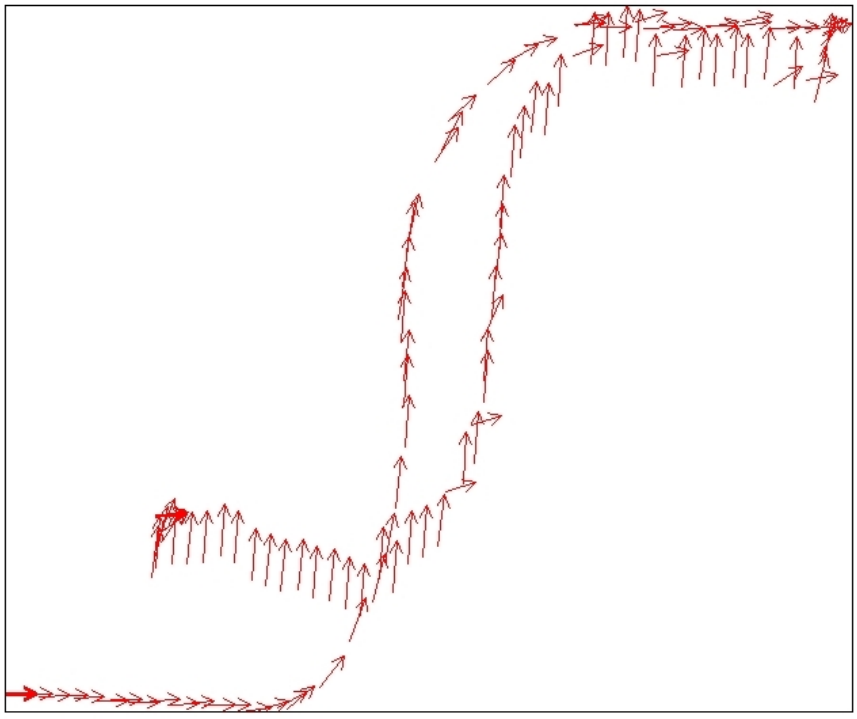

Figure 6.7.: Results of wheel tests

The resulting data is graphed in Figure6.7. The test began at the bottom right corner and ended further up and to the left of the start point. When compared to the locus described in Figure6.5, it can be seen that the robot follows the locus with some errors,

but as this is an open loop test it is not surprising. The direction appears to be incorrect while the vehicle is moving sideways from the top most point of the graph. This direction should be towards the right, however there was not enough information to determine the heading, only the angle of the plate.

6.3.3. Errors

Possible errors included:

- Movement of the target plate on the platform
- Differences between speed of forward and lateral motion

6.3.4. Conclusion

The above section tested the various motion capabilities of the Mecanum wheel robot. These motions included, forward and sideways motion and all directions in between as it transitioned from sideways motion to forward and back to sideways motion. It also showed that the robot is capable of rotating around its own center and other Instantaneous Centres of Rotation. The motion did not follow the locus exactly, which should have resulted in its return to the starting position, this was expected as this was not a closed loop control test. The result does demonstrate the omnidirectional capabilities of the platform and its motion control software.

6.4. Testing the obstacle avoidance software

At least two different obstacle avoidance algorithms are available for use with the Player Project, namely the “nd” and “vfh” methods discussed in Section 5.4.4 . While both of these were attempted for use on the platform only the “vfh” algorithm worked reliably. The reason for “nd” not working was very difficult to diagnose as the software is written with Spanish comments resulting in it being abandoned as an option for use as an obstacle avoidance method.

A path planner called “wavefront” was used on the platform. For the obstacle avoidance tests a false map was given to the planner in the shape of a large circular room.

This resulted in paths being planned straight through obstacles that were set up in the robot's path. This meant that the obstacle avoidance algorithm was fully responsible for getting the robot to the goal.

6.4.1. Test method

A cross shaped obstacle was put in the robot's path as can be seen in Figure 6.8. Poles were attached at the centre and edges of the cross. These allowed the laser scanner data to be referenced to the position of the cross. Note that the boxes did not need to be taped closed as the robot's laser scanner was lower than their upper edge and the external laser scanner used the poles as position reference, higher than the box lids. The four NorthStar sensors were placed above each quadrant of the cross. A map of the test environment was prepared and is shown in Figure 6.9. It is not used in this test, but it is included here for clarity. The cross shown in Figure 6.8 can be seen on the map.

Figure 6.8.: Layout of the obstacle

Four NorthStar projectors were placed on the ceiling above the cross, see Figure 6.10. Points directly below the NorthStar projectors were chosen as goal positions. The order in which the goal positions were sent to the robot, meant that it kept taking left turns. Consider Figure 6.11, if for example the robot was moving from point 4 to point 1, then next goal would be point 3.

Because the Wavefront algorithm requires a map, a generic "map" was provided. This was not the map shown in Figure 6.9. The map used is a large circle with the origin in

Figure 6.9.: Map of test environment

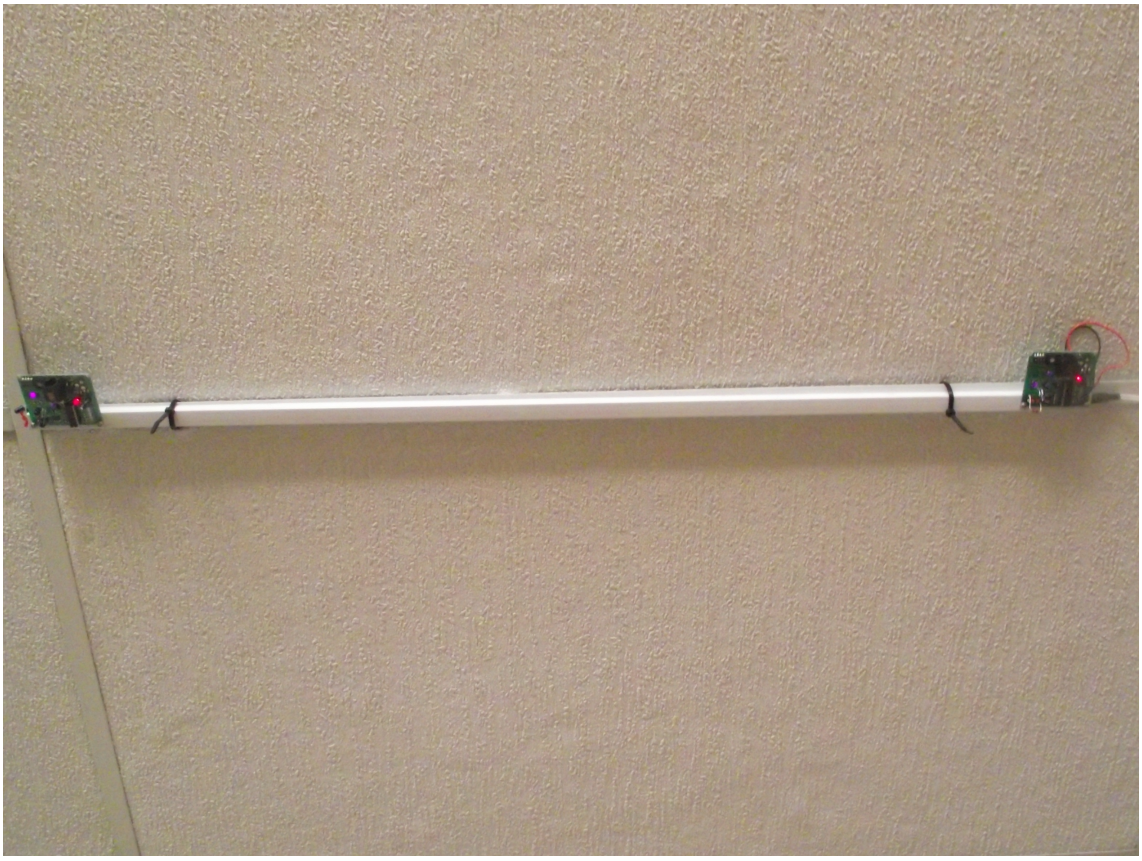

Figure 6.10.: NorthStar Projector Pair mounted on the ceiling

the center and does not include the obstacle that is really in the robot's path, causing it to plan a straight line to the next goal. This forces the robot to avoid the obstacle in the centre of the room using only the obstacle avoidance software.

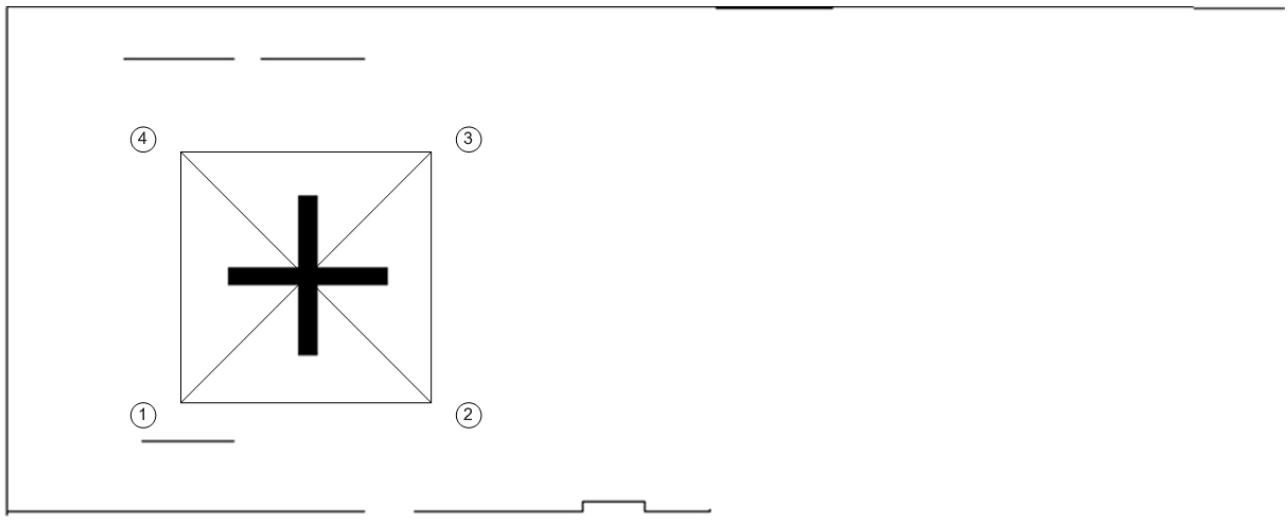

Figure 6.11.: Obstacle avoidance goals and paths superimposed on the map

A Sick LMS 200 laser scanner was once again used to gather external position data from the robot using an in-house program that records the data to a .csv file. The same metal plate used in Section 6.3 was used in this test, see Figure 6.6. Another program, ScanParser.exe, was then used to convert the data into a position vector plot as will be seen in the Results section. Data was also recorded internally by the robot allowing a comparison to be made between the two.

6.4.2. Results

The following figures are excerpts from the tests performed. Red arrows indicate the position and heading of the robot, the green dots indicate background points. The position of the cross shaped obstacle, discussed in Section 6.4.1 can be clearly seen in Figure 6.12. For each set of results there are two figures, the first figure is the data gathered from the laser scanner and is more reliable. The second figure in the set is the data that was recorded on the platform and is based on the Northstar data received.

Figures 6.12 and 6.13 show the motion of the platform from the start position through goal number 1 to goal 2. The platform gets to these goals quite easily as there is no real obstacle in its way.

Figures 6.14 and 6.15 show the motion of the platform from, position 2 along the diagonal to position 4. This takes much longer as the platform must find its way there

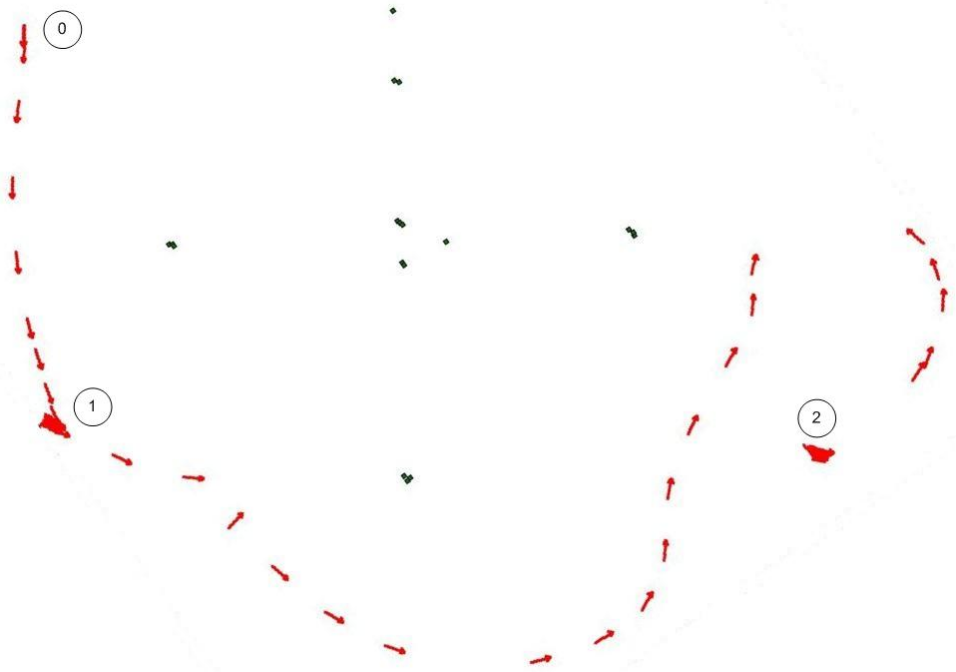

Figure 6.12.: Motion from start through goal 1 to goal 2, laser data

Figure 6.13.: Motion from start through goal 1 to goal 2, on board data

using only obstacle avoidance. Because there is no planning it overshoots the goal more than once, before finding it.

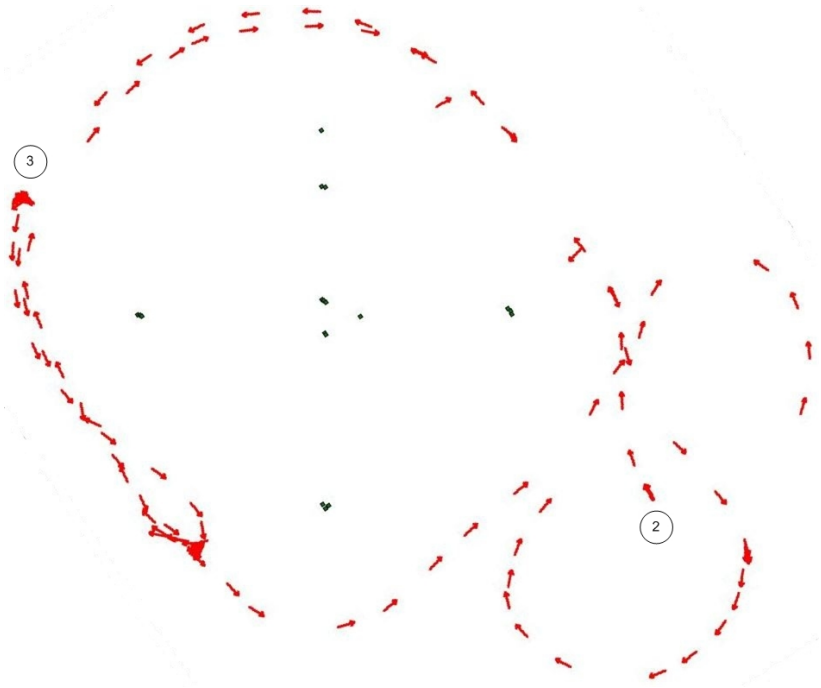

Figure 6.14.: Motion from start through goal 1 to goal 2, laser data

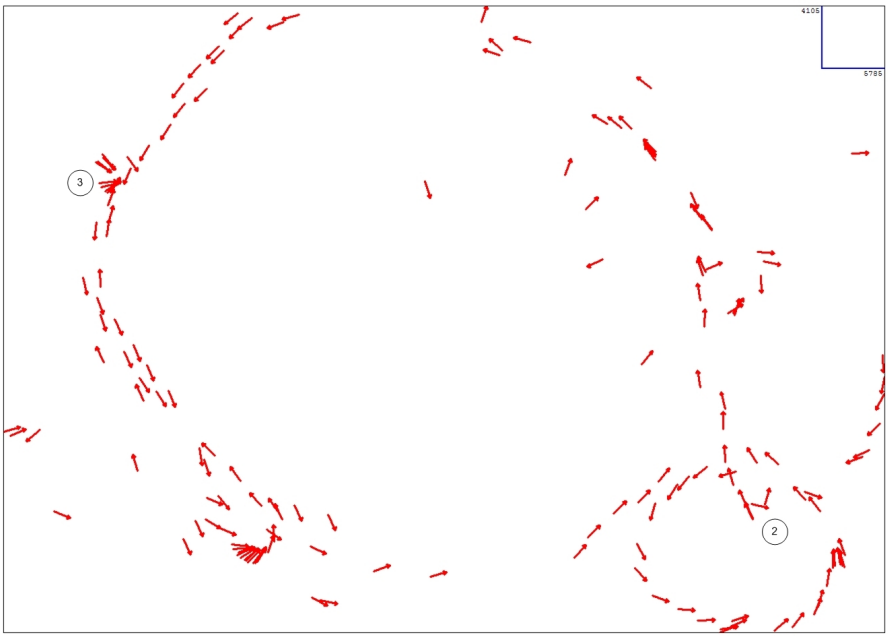

Figure 6.15.: Motion from start through goal 1 to goal 2, on board data

6.4.3. Errors

Errors were experienced with the interpretation of the laser data and with the positioning of the robot.

- **Laser Data Error** - the target for the laser is a flat plate and thus does not provide any directional data. This is addressed in software, which assumes that the direction of motion of the platform, at the start of a test, is in a forward direction. However where the plate rotates through the end on position towards the laser, only a very few points are seen and the direction of motion may be lost. An example of this can be can be clearly seen in Figure 6.12. Directly above the position 2 marker, there are a number of arrows missing and the direction of motion appears to change. When compared to Figure 6.13 it can be seen that the direction of motion does not change. This cannot be helped, but the figures are still useful for comparison between the laser data and the on board data.
- **NorthStar Error** - the NorthStar sensor appears not to be the most stable source of positional data. At certain points in time the sensor perceives an incorrect position for the robot, this can be seen in the few arrows which are significantly off path in Figure 6.15, when compared to the laser data.

6.4.4. Conclusion

Based on the above figures, and more data that was captured showing similar behaviour, it can be seen that the platform is capable of successfully avoiding obstacles in its path. It is not as successful at reaching the goal smoothly, but this is to be expected as the path planning algorithm has effectively been disabled. The difference between the position that the robot reports and the position which the laser scanner captures can also be seen, which is explained by NorthStar inaccuracy.

6.5. Testing the path planning software

The path planning software was tested using the same environment as for the preceding obstacle avoidance tests. However this time the path planner was given a map of the environment shown in Figure 6.9 and Figure 6.11. The room extends much further to the right of the figure, but only the part that the robot was likely to move in was mapped. The lines inside the map are flat surfaces (overturned tables) which were placed to avoid other furniture behind them which could have been damaged by a robot malfunction. To the bottom right of the cross is a gap, which indicates an open door. The Sick Laser scanner was used as an external position reference.

6.5.1. Test Method

The robot was given the exact same commands as in the obstacle avoidance section, to allow comparison of the speed to reach goals. The test set-up was precisely the same, except for the addition of the map in Figure 6.9. This addition is made by modifying the configuration file that is called when the Player server starts.

6.5.2. Results

The following figures are excerpts from the full set of data recorded:

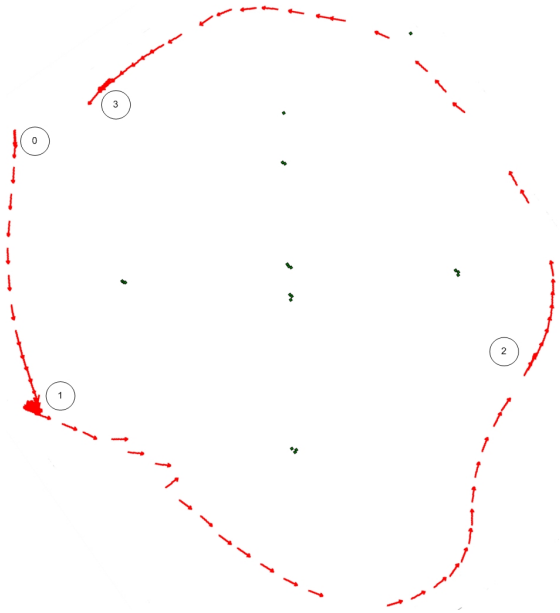

Figure 6.16.: Laser Scanner Data - Start to goal 3

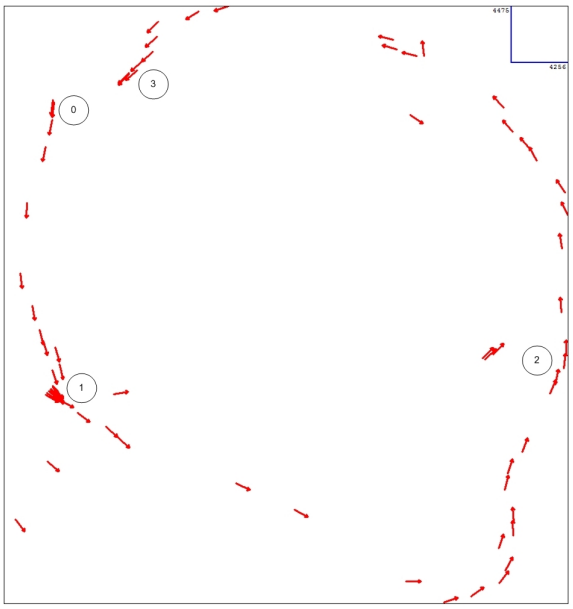

Figure 6.17.: On board Data - Start to goal 3

It can be seen that the motion of the platform is much smoother than the tests performed in Section 6.4 without the map. It can still be observed that the NorthStar

data is at times inaccurate, but the planner is still able to guide the platform to its goals sooner and more smoothly.

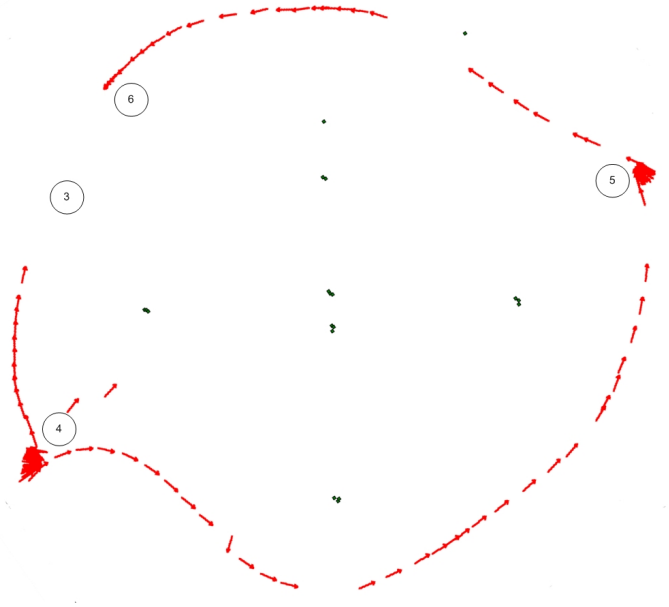

Figure 6.18.: Laser Data - Goal 3 to Goal 6

Unfortunately the first few data points in this scan were with the platform positioned directly towards the laser scanner, which meant that not enough data points were found to show the position of the platform.

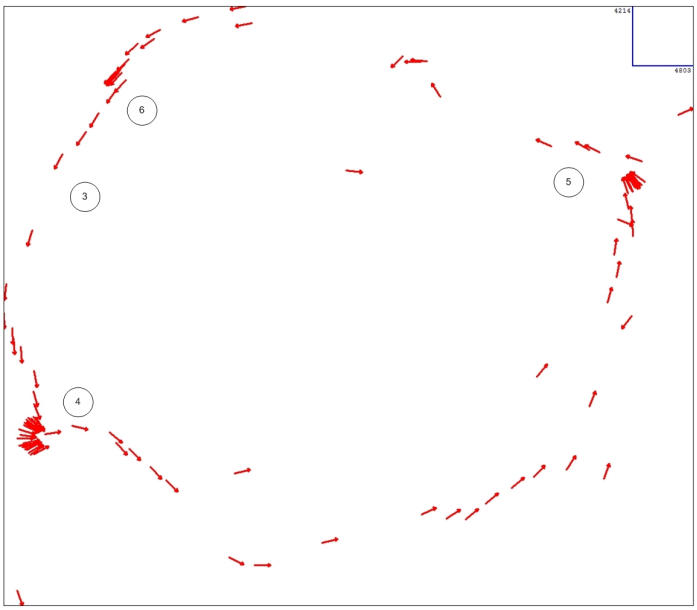

Figure 6.19.: On board Data - Goal 3 to Goal 6

6.5.3. Errors

The errors in this section will include all those discussed in Section 6.4.3. Another probable error is inaccuracy of the map which could be due to a possible mismatch between the map and the features seen by the laser scanner on board the platform. This may be in the form of items missing from the map, or items in the map, which are positioned at different co-ordinates to those expected by the path planning algorithm.

6.5.4. Conclusion

Based on the above data it can be seen that the platform can navigate smoothly to a goal position, if given an accurate map of its surroundings, using the Wavefront path planning algorithm provided in Player. It is also much more successful at reaching it's goals than when a map is not supplied.

6.6. Chapter Summary

This chapter has presented the results of the tests performed using the robotic platform. Wheel tests were performed to show that the actual response of Mecanum wheels follow the model developed in this project. A locus test was performed to show that the robot is capable of omnidirectional motion with the developed model implemented. An obstacle avoidance test was completed, which showed that the robot was able to successfully avoid obstacles using the "vfh" algorithm. A path planning test was completed using a map of the same environment as the obstacle avoidance tests, illustrating the success of the algorithm and the time saving that such an algorithm can produce.

Chapter 7.

Conclusion

7.1. Summary of Findings

A literature study was made into the various aspects of mobile robot technology, gathering information about Locomotion, Perception, Localisation and Navigation. A robotic platform was then designed and built using Mecanum wheels as the method of locomotion. A new mathematical model was developed to control these wheels. Electronics were installed on the robot to provide power for locomotion, sensors for Perception and computing power for Localisation and Navigation. Software was written to interface with the Player server which was chosen as the robot control code. This included both PC and embedded software. Tests were performed to show that the developed model performed correctly and that the system was able to navigate around obstacles with and without a map.

All the objectives of the project were achieved:

- A platform was researched, designed and tested that demonstrates omnidirectional motion and is capable of moving autonomously from one pose to another, under its own power.
- A unique Mecanum vehicle control method was derived from first principles and a test platform was designed and built to verify this method.
- A sensor system was implemented and tested that allowed the robot to avoid obstacles.
- A navigation system was implemented that allowed the robot to move from one point to another using a map of its environment.

7.2. Discussion of Problems

Problems were mainly experienced with software, or software interfaces to hardware. These problems are summarised below:

- It was found that the ultrasonic sensors did not work with the obstacle avoidance software. This may be due to insufficient numbers of sensors placed around the platform. This possibility was not experimented with
- The infrared scanner also did not interface well with the obstacle avoidance software, this could be due to the low sample rate when compared to a laser scanner or the lower resolution. Ultimately a laser scanner was implemented for obstacle avoidance
- The documentation for the Player server was not always clear and one of the drivers was commented in Spanish which made debugging impossible
- Minor mechanical problems were also experienced with the wheel tester

None of the above problems were fatal, but a significant amount of time was spent attempting to get the software working with the infrared scanner. When the laser scanner was attempted in its place the robot performed well.

7.3. Conclusions

This study has shown the development of an autonomous mobile robot. The robot has excellent manoeuvrability due to its implementation of Mecanum wheels for locomotion. A new algorithm allows the robot to move with full omnidirectionality, but the obstacle avoidance algorithms implemented by the Player server do not allow this to be fully utilised. Obstacle and path planning was successfully implemented on the developed platform.

7.4. Summary of contributions

This project made contributions in three different fields, namely, Mechanical, Electronics and Software engineering.

As a mechanical contribution a robotic platform was designed and constructed implementing Mecanum wheels to provide locomotion. A new general algorithm was developed for composite wheels which is used to control the Mecanum wheels. Tests performed show that this algorithm works well.

The contribution to electronics was made in the form of integration of the sensor systems so that they communicate effectively with the on board computer. The software contribution included a Player driver to control the motion of the vehicle, which implemented the algorithm mentioned above. Numerous other items of PC software and embedded software were written to conduct tests and to post process the data resulting from these tests.

7.5. Suggestions for further Research

A new Player driver providing obstacle avoidance capabilities to omnidirectional vehicles could be written in the future. Currently available obstacle avoidance drivers are written for differential drive robots. Such an algorithm should have positive implications for the motion capabilities of the robot.

More work could be done using more ultrasonic sensors on the current platform. This may enable it to navigate without the use of an expensive laser scanner. A better Player driver could also be written for the infrared scanner so that this low cost sensor can be used on the platform.

Bibliography

[1] Askoxford: autonomy. http://www.askoxford.com/concise_oed/autonomy?view=uk, August 2009.

[2] Askoxford: robot. http://www.askoxford.com/concise_oed/robot?view=uk, August 2009.

[3] Underwater vehicles : Woods hole oceanographic institution. http://www.whoi.edu/page.do?pid=8423, August 2009.

[4] Denel dynamics - division of denel pty (ltd.). http://www.deneldynamics.co.za/, September 2010.

[5] Mirriam webster. http://www.merriam-webster.com/dictionary/omnidirectional, September 2010.

[6] Nearness diagram. http://playerstage.sourceforge.net/doc/Player-2.1.0/player/group__driver__nd.html, September 2010.

[7] Vector field histogram. http://playerstage.sourceforge.net/doc/Player-2.1.0/player/group__driver__vfh.html, September 2010.

[8] A. Al-Jumail and C. Leung. Wavefront propagation and fuzzy based autonomous navigation. *International Journal of Advanced Robotic Systems*, Volume 2, Number 2:093–102, 2005.

[9] P. Arena, P. Di Giamberardino, L. Fortuna, F. La Gala, S. Monaco, G. Muscato, A. Rizzo, and R. Ronchini. Toward a mobile autonomous robotic system for mars exploration. *PLANETARY AND SPACE SCIENCE*, 52:23–30, 2004.

[10] P.H. Batavia and I. Nourbakhsh. Path planning for the cye personal robot. In *IEEE RSJ International Conference on Intelligent Robots and Systems*, volume 1, pages 15–20, 2000.

[11] M. Beetz and M. Bennewitz. Planning, scheduling, and plan execution for autonomous robot office couriers. *Integrating Planning, Scheduling and Execution in Dynamic and Uncertain Environments*, 1998.

[12] Robert H. Bishop, editor. *The Mechatronics Handbook*. CRC PRESS, 2002.

[13] J. Borenstein, H.R. Everett, L. Feng, and D. Wehe. Mobile robot positioning: Sensors and techniques. *Journal of Robotic Systems*, 14:231–249, 1997.

[14] J. Borenstein and Y. Koren. The vector field histogram - fast obstacle avoidance for mobile robots. *IEEE Journal of Robotics and Automation*, Vol 7, No 3:278–288, 1991.

[15] P Bosscha. Atmega wheel encoder, sonar, bumper embedded software. 2009.

[16] P Bosscha. Wsb player driver. 2009.

[17] W. Burgard, A.B. Cremers, D. Fox, D. Haehnel, G. Lakemeyer, D. Schulz, W. Steiner, and S. Thrun. Interactive museum tour-guide robot. In *5th National Conference on Artificial Intelligence, AAAI*, 1998.

[18] J.M. Burnfield and C.M. Powers. *Orthopaedic Physical Therapy Secrets*, chapter Normal and Pathologic Gait. Hanley and Belfus, 2006.

[19] G. Campion, G. Bastin, and B. D'Andrea-Novel. Structural properties and classification of kinematic and dynamic models of wheeled mobile robots. *IEEE Transactions on Robotics and Automation*, 12, No. 1:47–62, 1996.

[20] R Chatila and S Lacroix. Adaptive navigation for autonomous mobile robots. *7th International symposium on Robotics research, Munich*, none:450–458, 1995.

[21] S. Coetzee. Solidworks simlations.

[22] JA Cooney, WL Xu, and G Bright. Visual dead-reckoning for motion control of a mecanum-wheeled mobile robot. *MECHATRONICS*, 14(6):623–637, JUL 2004.

[23] J. Corbet and A. Rubini. *Linux Device Drivers*. O'Reilly Media, second edition, 2001.

[24] M. de Villiers and G. Bright. Development of a control model for a four wheel mecanum vehicle. In *Procedings of the 25th International Conference of CAD/CAM, Robotics & Factories of the Future*, 2010.

[25] S.L. Dickerson and B.D. Lapin. Control of an omni-directional robotic vehicle with mecanum wheels. In *National Telesystems Conference*, pages 323–328, 1991.

[26] O. Diegel, A. Badve, G. Bright, J. Potgieter, and N.S Tlale. Improved mecanum wheel design for omnidirectional robots. In *Proc. 2002 Australian Conference on Robotics and Automation*, 2002.

[27] G. Endo and S. Hirose. Study on roller-walker (multi-mode steering control and self-contained locomotion). In *IEEE International Conference on Robotics and Automation (ICRA)*, volume 3, pages 2808–2814, 2000.

[28] L. Ferriere, B. Raucent, and G. Campion. Design of omnimobile robot wheels. In *Conference Proceedings of the 1996 13th IEEE International Conference on Robotics and Automation.*, volume 4, pages 3664–3670, 1996.

[29] J. Forlizzi and C. DiSalvo. Service in the domestic environment: A study of the roomba vacuum in the home. In *Conference of HRI 2006: 2006 ACM Conference*

on Human-Robot Interaction, 2006.

[30] B. Gerkey, R. Vaughan, K Stoy, and A. Howard. Most valuable player: A robot device server for distributed control. In *Proceedings of the IEEE/RJS International Conference on Intelligent Robots and Systems*, 2001.

[31] A. Gfrerrer. Geometry and kinematics of the mecanum wheel. *COMPUTER AIDED GEOMETRIC DESIGN*, 25:784–791, 2008.

[32] P Horowitz and W Hill. *The Art of Electronics*. Cambridge University Press, 1995.

[33] F. Hsiao, Y. Lai, and H. Tenn. The development of an unmanned aerial vehicle system with surveillance, watch, autonomous flight and navigation capability. In *Proceedings of the 21st Bristol UAV Systems Conference*, 2006.

[34] T.R. Hsu. Mechatronics. an overview. *Components, Packaging, and Manufacturing Technology, Part C, IEEE Transactions on*, 1997.

[35] Bengt Ilon. Wheels for a course stable self propelling vehicle movable in any desired direction on the ground or some other base, 1975.

[36] Jung Won Kang, Bong Sung Kim, and Myung Jin Chung. Development of assistive mobile robots helping the disabled work in a factory environment. In *Proceeding of the 2008 IEEE/ASME International Conference on Mechatronic and Embedded systems and applications*, pages 426–431, 2008.

[37] Michael Karpelson, Gu-Yeon Wei, and Robert Wood. Milligram-scale high-voltage power electronics for piezoelectric microrobots. In *2009 IEEE International Conference on Robotics and Automation, ICRA '09*, 2009.

[38] J. Kramer and M. Scheutz. Development environments for autonomous mobile robots: A survey. *Autonomous Robots*, 22:101–132, 2007.

[39] J.C. Latombe. *Robot Motion Planning*. Springer, first edition, December 1990.

[40] T. Lauwers, G. Kantor, and R. Hollis. One is enough. In *12th International Symposium on Robotics Research San Francisco,*, 2005.

[41] S. M. LaValle. *Planning Algorithms*. Cambridge University Press, 2006.

[42] J. Liu, I. Dukes, and H. Hu. Novel mechatronics design for a robotic fish. In *International conference on Intelligent Robots and Systems*, 2005.

[43] J. Minguez and L. Montano. Nearness diagram (nd) navigation: Collision avoidance in troublesome scenarios. *IEEE Transactions on Robotics and Automation.*, 20(1):45–59, 2004.

[44] J. Minguez, J. Osuna, and L. Montano. A divide and conquer strategy based on situations to achieve reactive collision avoidance in troublesome scenarios. In *IEEE International Conference on Robotics and Automation Robotics and Automation (ICRA)*, 2004.

[45] K. Nagatani, S. Tachibana, M. Sofue, and Y. Tanaka. Improvement of odometry for omnidirectional vehicle using optical flow information. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 468–473, 2000.

[46] H.G. Nguyen and H.R. Everett. Joint robotics program (jrp)-supported efforts at the space and naval warfare systems center, san diego - art. no. 623024. In *Unmanned Systems Technology VIII*, 2006.

[47] R.D. Schraft and G. Schmierer. *Service Robots*. AK Peters, 1st edition, 2000.

[48] M.Y. Shieh, J.C. Hsieh, and C.P. Cheng. Design of an intelligent hospital service robot and its applications. In *IEEE International Conference on Systems, Man and Cybernetics*, 2004.

[49] A. Shimada, S. Yajima, P. Viboonchaicheep, and K. Samura. Mecanum-wheel vehicle systems based on position corrective control. In *IECON 2005: Thirty-First Annual Conference of the IEEE Industrial Electronics Society, Vols 1-3*, pages 2077–2082, 2005.

[50] Illah R. Nourbakhsh Siegwart R. *Introduction to Autonomous Mobile Robots*. Number 026219502X in ISBN-10. The MIT Press, 2004.

[51] J. B. Song and K. S. Byun. Design and control of a four-wheeled omnidirectional mobile robot with steerable omnidirectional wheels. *Journal of Robotic Systems*, 21(4):193–208, 2004.

[52] P Terblanche. Northstar player driver. CSIR MMM, 2009.

[53] P Terblanche. Wheel tester software. CSIR MMM, 2009.

[54] P Terblanche. Laser interface software. CSIR MMM, 2010.

[55] P Terblanche. Laswer post processing software. CSIR MMM, 2010.

[56] N. Tlale and M. de Villiers. Kinematics and dynamics modelling of a mecanum wheeled mobile platform. In *15th International Conference on Mechatronics and Machine Vision in Practice (M2VIP)*, 2008.

[57] N. Tschichold-Gurman, S.J. Vestli, and G. Schweitzer. Service robot mops: First operating experiences. *Robotics and Autonomous Systems*, 34:165–173, 2001.

[58] S. Tsuda, K. Zauner, and Y. Gunji. Robot control with biological cells. *BioSystems*, 87:215–223, 2007.

[59] J. Čapek. Rossum's universal robots, 1920. Prague, CZ **Note**: Although used in a play written by K Čapek, he attributes the word to his brother.

[60] Iagnemma K. Udengaard, M. Kinematic analysis and control of an omnidirectional mobile robot in rough terrain. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2007.

[61] R. Vaughan and B. Gerkey. Really reusable robot code and the player/stage project. In *Software Engineering for Experimental Robotics*, 2006.

[62] P. Viboonchaicheep, A. Shimada, and Y. Kosaka. Position rectification control for mecanum wheeled omni-directional vehicles. In *The 29th Annual Conference of the IEEE Industrial Electronics Society*, 2003.

Appendix A.

CAD drawings of the platform, components and test equipment

Drawings of the Robot Frame

This section shows a progression of drawings of the platform as it is assembled.

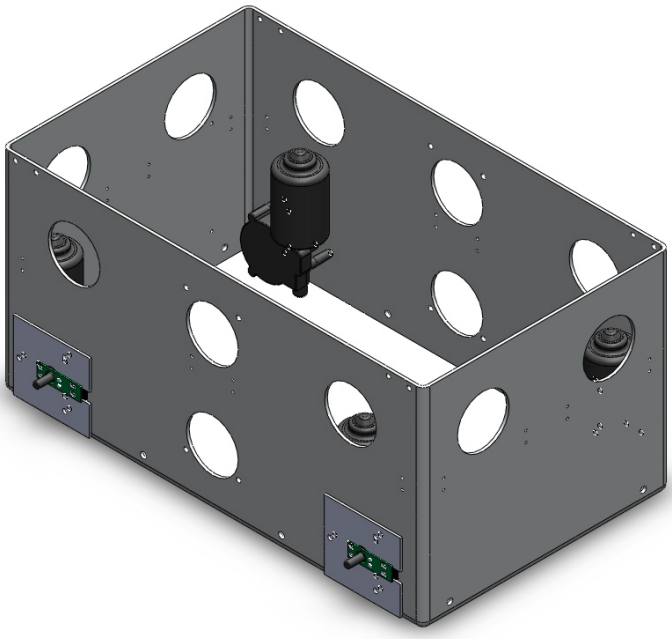

Figure A.1.: Platform base box with motors and wheel encoders attached

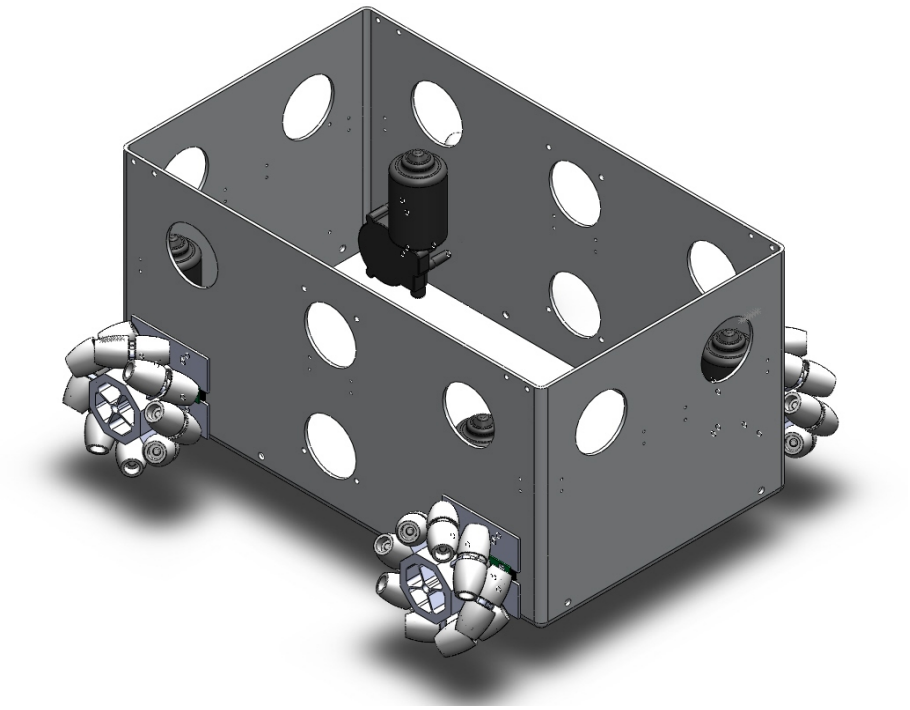

Figure A.2.: Platform base with wheels attached

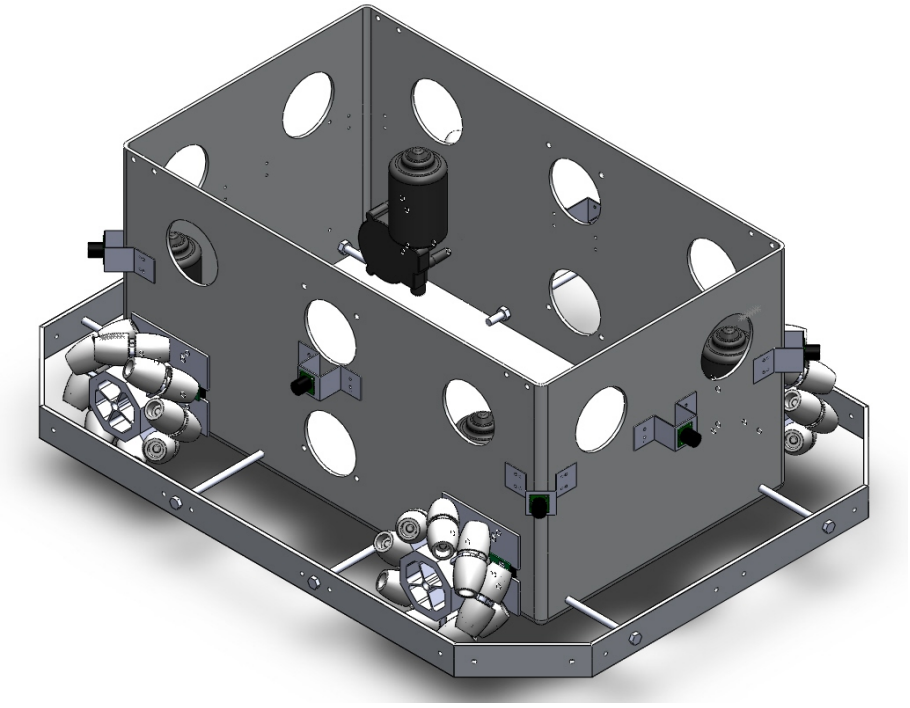

Figure A.3.: Platform base with bumper ring and ultrasonics sensors attached

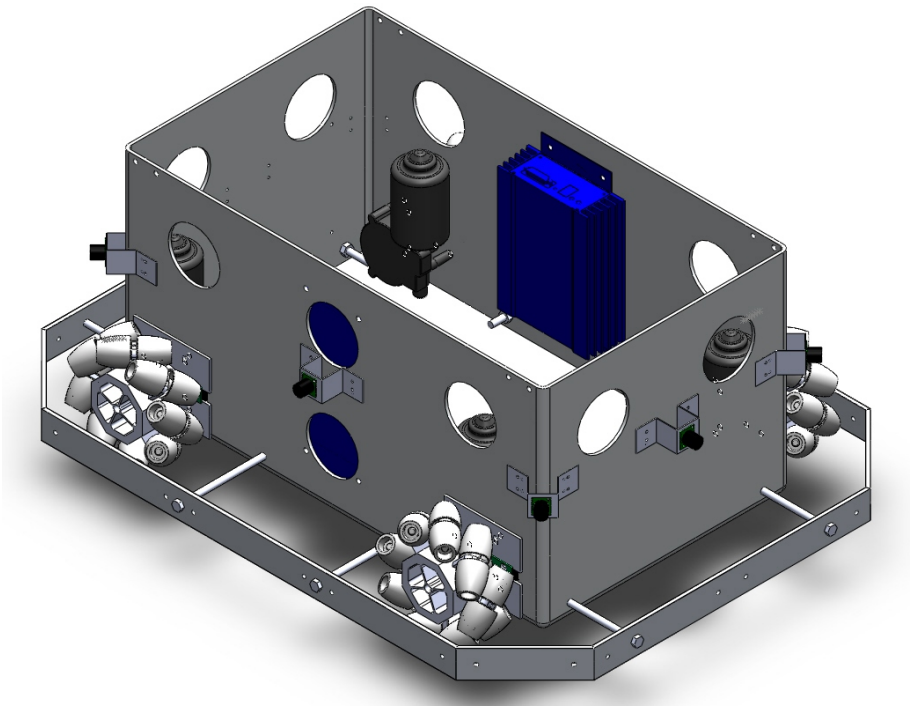

Figure A.4.: Platform base with motor controllers attached

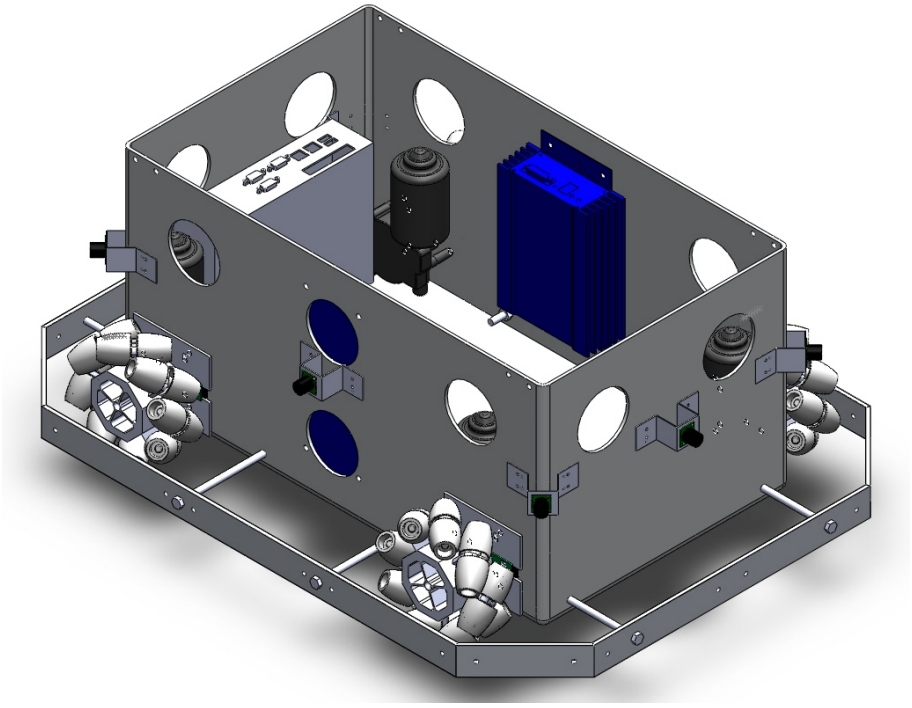

Figure A.5.: Platform base with Single Board Computer attached

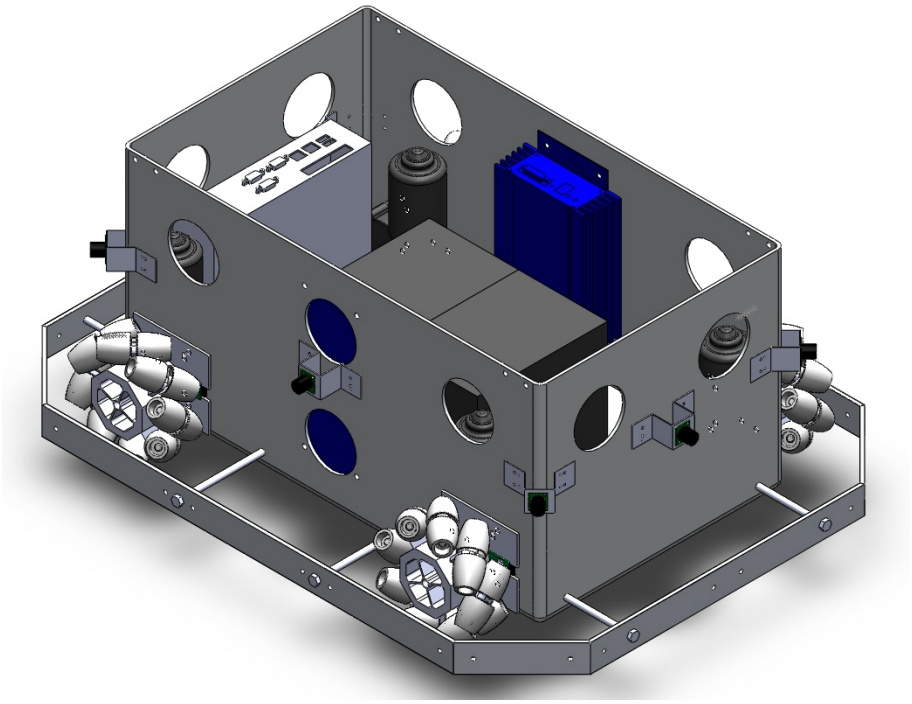

Figure A.6.: Platform base with batteries installed

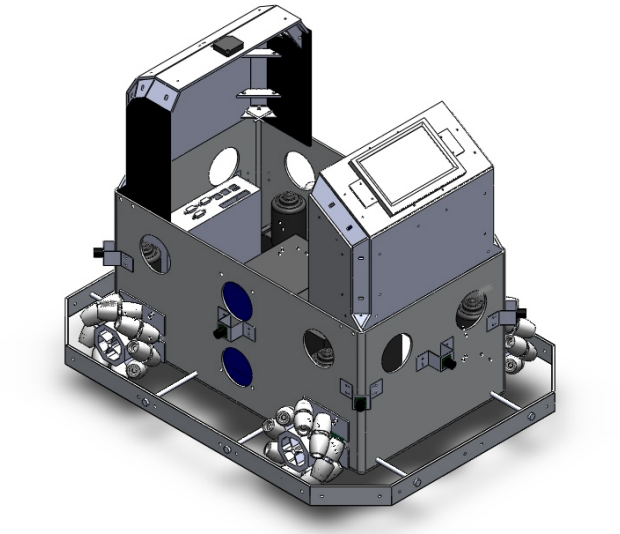

Figure A.7.: Platform base with NorthStar and touch screen mounts attached

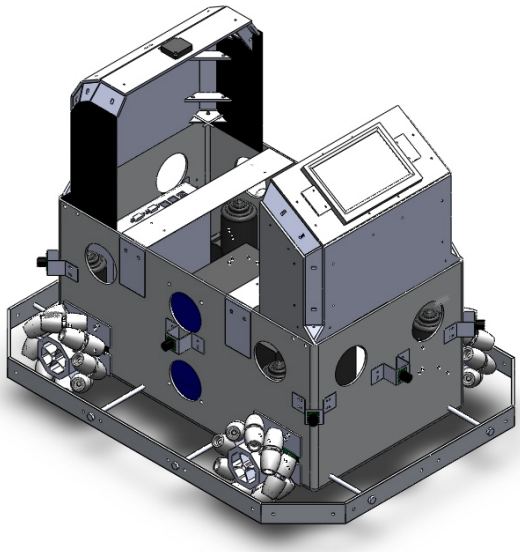

Figure A.8.: Platform base with accessory mount attached

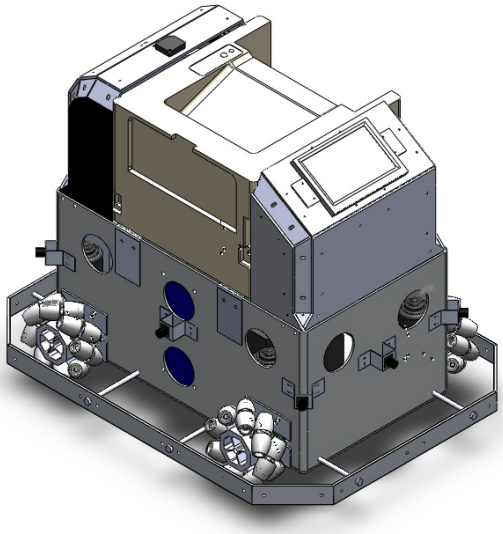

Figure A.9.: Platform base with printer attached

Drawings of Components

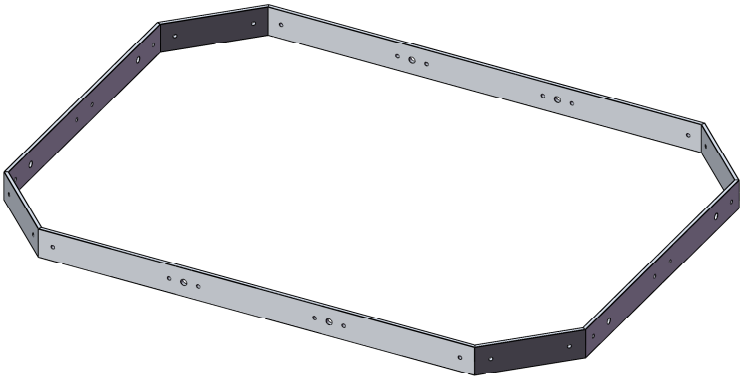

Figure A.10.: Bumper Ring

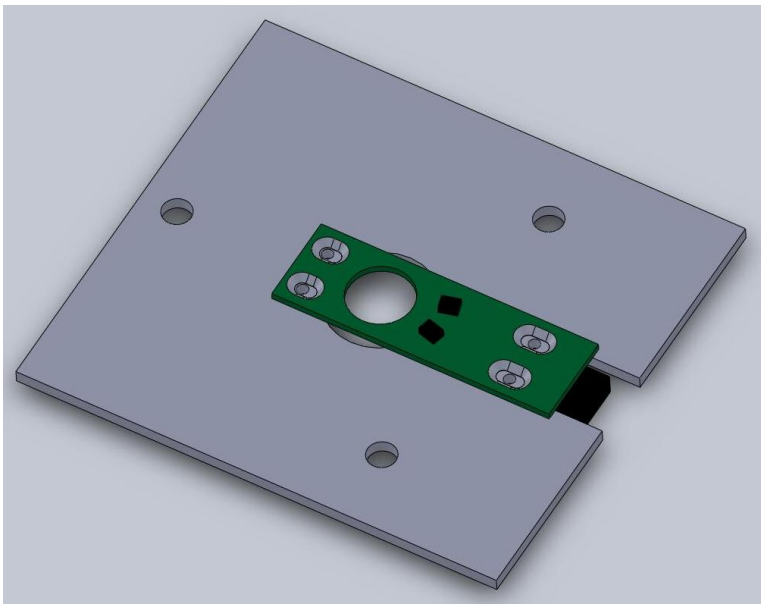

Figure A.11.: Wheel Encoder Mount

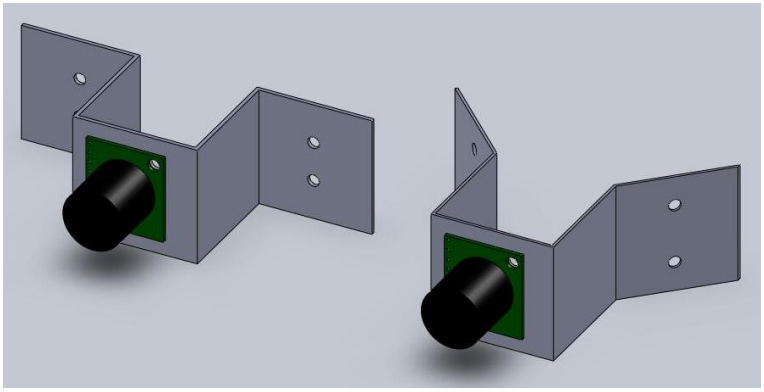

Figure A.12.: Ultrasonic sensor mounts

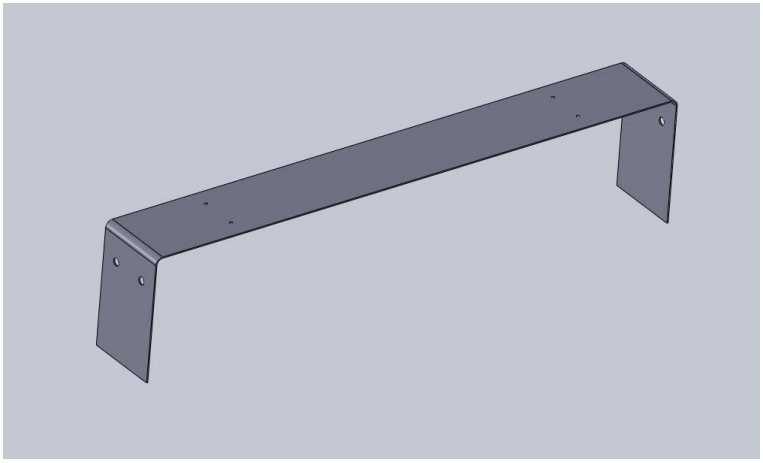

Figure A.13.: Printer Mounting Plate

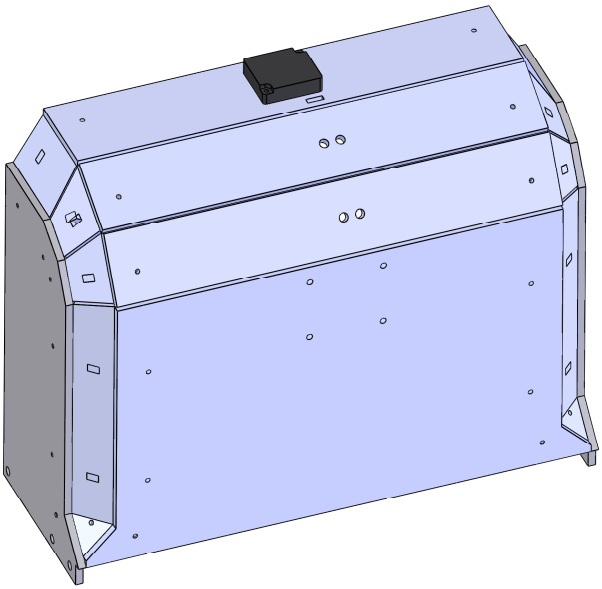

Figure A.14.: North Star Sensor Mount

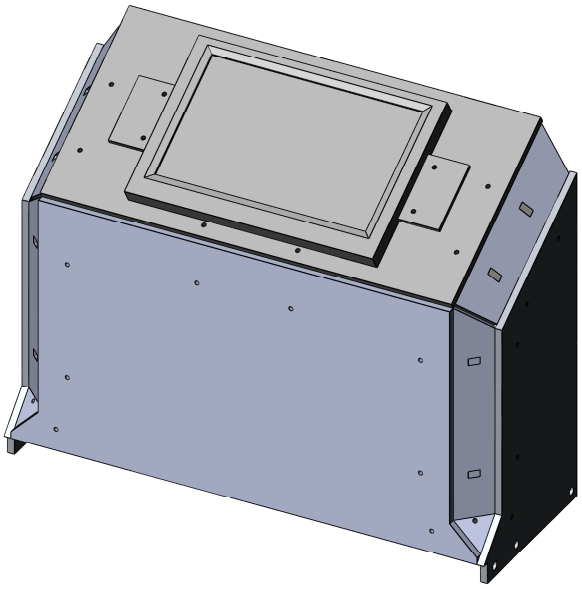

Figure A.15.: Touch Screen Mount

Drawings of the Wheel tester

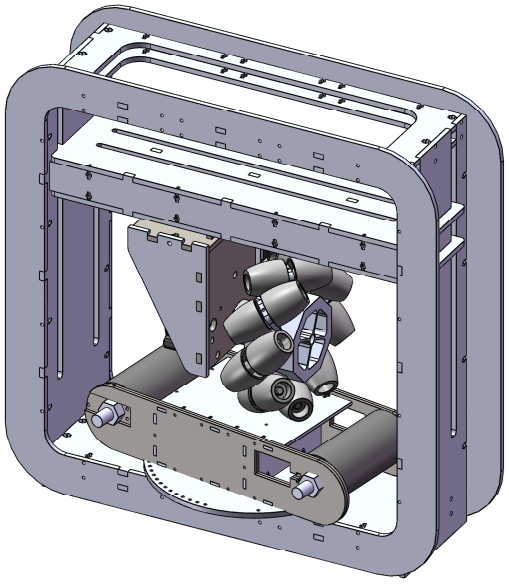

Figure A.16.: Perspex Wheel Tester

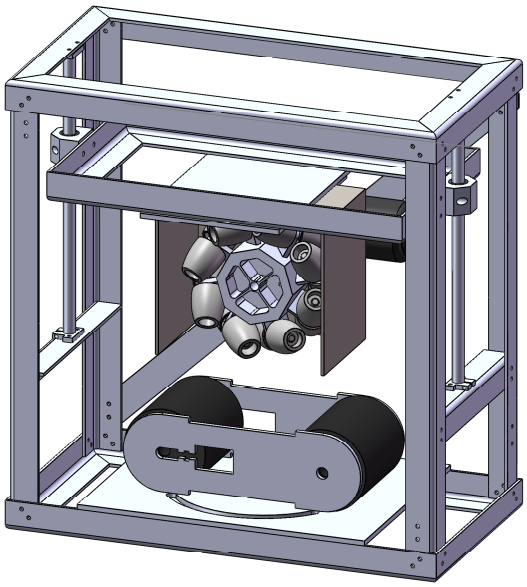

Figure A.17.: Tester design using angle iron frame

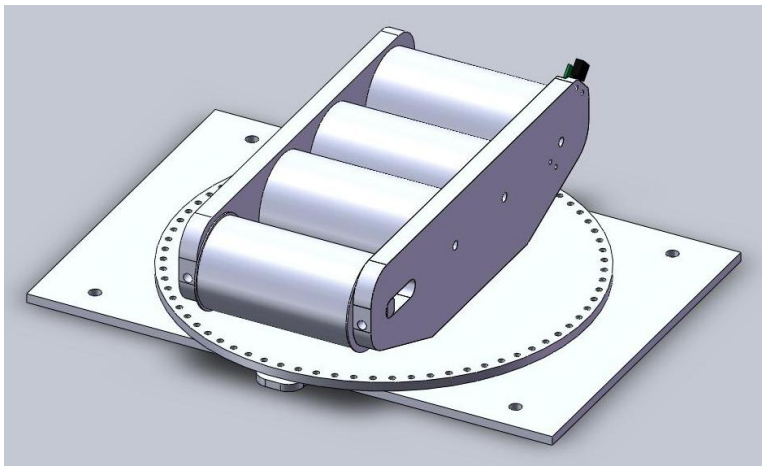

Figure A.18.: Treadmill

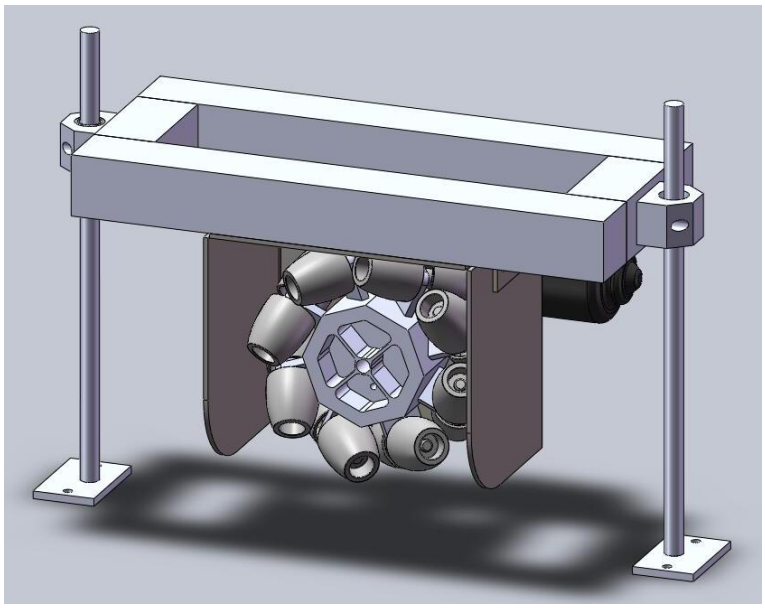

Figure A.19.: Tester Motor Mount

Appendix B.

Electronic Datasheets

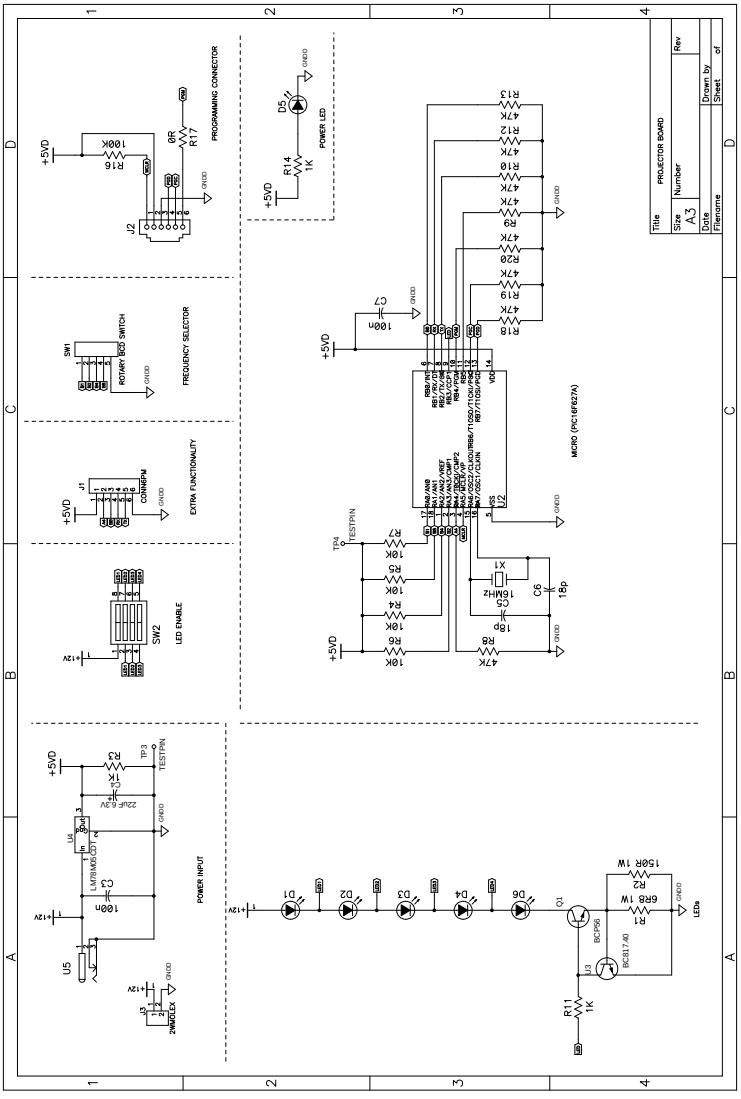

Figure B.1.: Projector Schematic

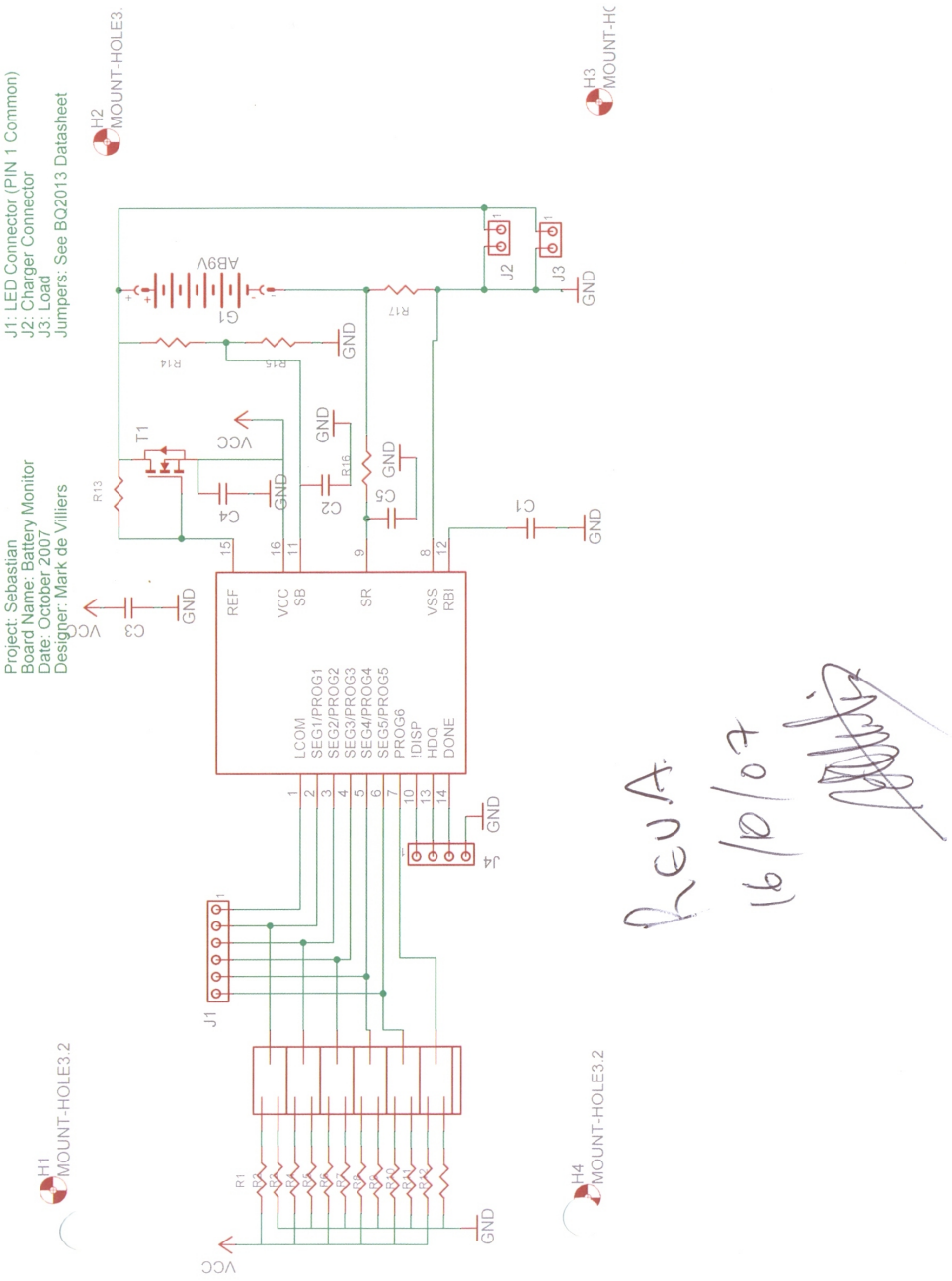

Figure B.2.: Gas guage schematic

Appendix C.

Software Listings

Interface Board Embedded Software

- comms.c
- comms.h
- global.h
- hw.c
- hw.h
- makefile
- ngui.c
- ngui.h
- Sebastian.c
- Sebastian.h
- support.c
- support.h
- wheelinterpretation.c

Player Drivers and Clients

A number of clients and drivers were written to run on the mobile robot. They are listed below, the make files have been included so that these items can be recompiled if desired.

Motion Driver

This driver performs the calculations converting motion commands received from client programs into specific wheel velocities.

- Mtn_current.cc
- makefile

Micro-controller Driver

This driver handles all communications to and from the PC to the wheel encoders, ultrasonics and bump sensors.

- WSBSensors.cc
- makefile

NorthStar Driver

The NorthStar Driver captures all NorthStar sensor data and reports the vehicle's position based on data received from the NorthStar sensor.

- northstar.cc
- makefile

Player Client File

- Mtn_client.c
- makefile

Wheel tester software

The Software used to run the wheel tester included PC software to manage the wheel speeds and record data, and Embedded software gather data from the wheel encoders.

Embedded Software

Wheel tester embedded software included the following files which are included on the attached CD:

- global.h
- hw.c
- hw.h
- ngui.c
- ngui.h
- support.c
- support.h
- uart.c
- uart.h
- WheelTester.c

PC Software

Wheel tester PC software included the following files included on the attached CD:

- AddDlg1.cpp
- AddDlg1.dfm
- AddDlg1.h
- DMod1.cpp
- DMod1.dfm

- DMod1.h
- MecaTread.cbproj
- MecaTread.cpp
- MecaTread.dsk
- MecaMain1.cpp
- MecaMain1.dfm
- MecaMain1.h
- NotesDlg1.cpp
- NotesDlg1.dfm
- Notes.Dlg1.h
- TreadDlg1.cpp
- TreadDlg1.dfm
- TreadDlg1.h

Laser Interface Software

- SickLMS.exe

Laser Post Processing Software

- ScanParser.exe
- ScanParser.ini

Appendix D.

Test Equipment and Set-up

D.1. Wheel test equipment

- In-house wheel test jig with turntable and treadmill, weighted with 5 kg
- "Red Scorpion XL" Motor controller board
- In-house wheel encoder and serial interface board
- AFX 650.667 DC Regulated power supply
- In-house USB-Serial converter box
- Fluke 8845A desktop multimeter
- Serial cable for Fluke connection to USB-Serial converter
- USB cable from USB-Serial converter to PC
- A Windows PC with the "mecatread.exe" program installed

D.2. Wheel Test Set-up Procedure

- Connect the Fluke 8845A to the USB-Serial box on the right hand serial port of that unit, using the serial cable
- Connect the controller board to the USB-Serial box on the left hand serial port of that unit
- Connect the USB-Serial box to one of the PC USB ports, make sure the PC has recognised the USB-Serial converter and the serial ports are registered on the system
- Connect the Scorpion board to the power supply using the attached connectors
- Turn on the 13.8 V regulated power supply.
- Make sure the controller board is powered - LEDs on
- Make sure the red motor controller board is on - LEDs on
- Open the “mecatread.exe” program
- Follow the directions in the program to set up the Fluke 8845A
- Select the USB Serial ports, if they are the wrong way around the program will generate a warning
- Select the desired test parameters
- Select the correct configuration

D.3. Motion Test Equipment

- Robot Platform - Working robot platform running Player on the on-board Single Board Computer (SBC)
- SICK LMS200 Laser Scanner
- “SickLMS.exe” program to gather data from the laser scanner
- 24 V DC power supply
- Networked Windows PC having one 9 pin serial port

- A Virtual Network Computing (VNC) program like "Real VNC" to allow wireless control of the robot

D.4. Robot Start up Check

- Ensure the robot battery is properly charged
- Ensure the the wireless dongle is connected to the SBC via USB
- Ensure that the Roboteq motor controllers are connected to the USB-to-serial converter
- Ensure that the USB-to-serial converter is connected to the SBC
- Ensure that the micro-controller is connected to the SBC via serial cable
- Ensure that the NorthStar receiver is connected to the SBC via serial cable
- Power up the on board Ubuntu SBC
- Power up the Roboteq motor controllers
- Power up the micro-controller
- Power up the NorthStar receiver
- Power up the on board LCD to ensure that Ubuntu starts up correctly on the SBC
- Open WinSCP (Windows Secure CoPy) – a free open source FTP (File Transfer Protocol) program
- Connect WinSCP to the Ubuntu box using the IP address: sebs-rover.csir.co.za
- Open 2 PuTTY terminal emulators

D.5. Laser Set-up

- Set the DC power supply to 24V, ensure that it is not current limited below 1 A
- Switch the power supply off
- Position the Laser Scanner in a manner that helps isolate the motion data, i.e.: attempt to keep the robot motion within as small an angular range as possible to help isolate the data during post processing
- Connect the power supply to the Laser Scanner making sure the polarity is correct
- Connect the Laser Scanner serial connector to the Windows PC serial port
- Switch the power supply on
- Start the "SickLMS.exe" program