Design and Construction of a Pair of Cooperating Autonomous Mobile Robots

A thesis
submitted in partial fulfilment
of the requirements for the Degree
of
Master of Science in
Physics and Electronic Engineering
at the
University of Waikato

by

Andrew Payne

THE UNIVERSITY OF
WAIKATO
Te Whare Wānanga o Waikato

2004

ABSTRACT

Humans have been using machines to assist moving large and heavy items for a long time. When a larger object needs to be shifted, a larger and more powerful machine is built. This significantly limits the operating environment and the number of potential applications, while the costs scale with the machine size. Contrary to this conventional trend is the idea of using multiple robots to cooperatively perform tasks that are difficult for a solitary robot or machine to achieve, such as carrying a long or flexible object. Cooperating robots offer many additional benefits such as increased manoeuvrability and efficiency. This thesis details the design and construction of two autonomous mobile platforms. Designed for outdoor use using a tricycle design, the robot platforms incorporate onboard features including a CPU, power supply, wireless communication, global positioning system (GPS), infrared range finders, and shaft encoders; with provisions for an array of other sensor types to be added. The mobile robots will be used for future research into cooperative behaviour by the Mechatronics Group at the University of Waikato.

ACKNOWLEDGEMENTS

I would like to greatly acknowledge the support, assistance and encouragement received throughout this project from the following people.

Dale Carnegie, my supervisor who has made the entire project possible. Encouragement to begin study at a graduate level has been followed by continual support and guidance from Dale, for which I am deeply grateful.

Assistance in all aspects of the project has been provided by the physics department technical staff. A special thank you goes to Bruce Rhodes and Scott Forbes for their willingness to help, and for their humour making the physics department a great place to study.

Thanks to my fellow graduate students, especially Lucas Sikking and Chris Lee-Johnson, for their ability to discuss different ideas and methods, finding the weaknesses of the circuits I had designed, and generally just having to endure writing a thesis along side me.

My friends, Jenny McFarlane for her continual encouragement and interest in a subject she doesn't understand, and to Paul Read for his enthusiastic assistance with software problems I had.

Finally thank you to my parents for their support throughout my studies.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

1 INTRODUCTION

When moving large or heavy items, the traditional tendency with machinery is to build a large mechanism capable of handling the load. This leads to continuous scaling of the mechanical size and weight of devices being built, with a proportional increase in expense. Trucks are continuously being manufactured with an increasing payload, with vehicles capable of carrying 100 tonne a common occurrence.

Figure 1-1 illustrates an alternative to this conventional trend; the use of cooperative behaviour. Humans are limited by their physical size and strength – and yet are capable of moving heavy furniture and other large items outside their individual strength ability. This is accomplished with the help of a friend to share the load, or a large number of people for a substantial object.

Figure 1-1: Variation between machinery and humans when lifting a heavy load

An alternative to building large machinery is to follow the approached used by humans; to design and build cooperative devices. Requirements of coordination, communication and precision control have restricted this approach in the past. However, new technology addresses these requirements, providing the foundation to allow cooperative interaction to be developed.

1.1 ADVANTAGES OF MULTIPLE COOPERATIVE ROBOTS

Why not just continue the existing mechanical scaling of machinery?

Existing machines are usually specific built, performing only the tasks they are designed for. This incurs the expense of custom designs and manufactured parts. Often the surrounding environment must also be adapted to support the size and weight of the machinery – such as building roads to traverse over.

Robots which are working together have a number of advantages over a single robot performing the same task. This becomes obviously apparent when carrying bulky, heavy, or flexible items such as illustrated in Figure 1-2, or where limited space is available to manoeuvre a large machine.

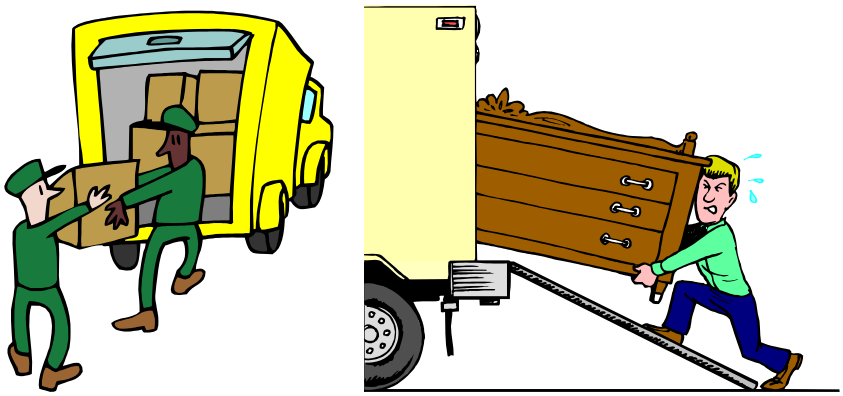


Figure 1-2: Two people performing a task have an advantage over a single person

By working in parallel, cooperative behaviour can increase efficiency and reduce the time required to complete a task. Reliability is increased by introducing redundancy when using a team of robots, while cost is reduced due to the use of smaller simplistic machine designs. Application specific design and manufacturing costs can be removed by fabricating the robots generically to span a range of tasks. Reduced weight necessitates less preparation and upkeep of the working environment, and new environments could be entered such as those with an unstable surface. Complex tasks can be introduced which are too difficult for a single entity to achieve.

1.2 APPLICATIONS

The applications of this project are numerous, especially as each unit can also work alone. Some possible industrial uses include:

- Mining
- Construction
- Forestry application
- Planetary exploration
- Automated manufacturing
- Search and rescue missions
- Cleanup of hazardous waste
- Industrial/household maintenance
- Nuclear power plant decommissioning
- Security, surveillance, and reconnaissance

Although these tasks could be performed by a solitary unit, the benefits of using multiple units are significant.

1.3 PROJECT SPECIFICATIONS

The objective of this project is to develop a pair of mobile robots to investigate autonomous cooperative behaviour. An autonomous unit is one which can independently control itself by making decisions and acting them out. The robots will eventually be required to work together cooperatively to perform a single task, for example carrying a long piece of wood, although the actual cooperative architecture is outside the scope of this project.

The design and development of these robots involves mechanical, electronic hardware and software aspects. Various sensors must be utilised to provide sufficient data for autonomous control. The design must consider the following attributes:

- **Scope** – The design must support future development, providing computational capabilities, power supply and space to include additional components such as sensors
- **Environment** - The robots will be required to operate in an outdoor terrain with a relatively smooth and level surface, such as concrete or mown grass
- **Payload** – Each robot must be able to support any onboard accessories required and an additional external payload of 40 kg
- **Manoeuvrability** – Each unit must be sufficiently agile to allow control in limited space. Precise control of the robot's path will be required
- **Self sufficient** – All required power and computation should be onboard the robot to facilitate independent operation
- **Operating time** – Each robot must be capable of running for one hour while operating at maximum power consumption
- **Efficiency** – As the power supply is local, the complete system should be kept efficient to maintain the required running period
- **Communication** – The robots must have a communication link to allow them to transfer both data and instructions or intentions
- **Software** – The software platform must provide the capability to be extended during future development

To attain the specifications, the following objectives must be met:

- Research existing designs and drive systems
- Locate and acquire mechanical components
- Select electronic equipment for data processing and communication
- Select and obtain sensors required for positioning/obstacle detection
- Evaluate and select a power source
- Design mechanical layout

- Construct chassis, drive and steering systems
- Construct enclosure and mounting for components/sensors
- Develop motor control hardware
- Develop a power output circuit to drive the motors
- Mount and interface sensors through hardware

- Develop software interface with motors and sensors
- Calibrate sensors
- Provide a communication link
- Test and evaluate

The final outcome of the thesis will be two mobile platforms, with the supporting electronics required to provide motion. Simple sensors will provide navigation data and obstacle detection. Software will provide the required interface to drive the motors and read data from the sensors.

1.4 THESIS STRUCTURE

The thesis is presented as follows:

Chapter 2 discusses existing mobile platform designs, drive methods, and the components that the project must incorporate.

Chapter 3 covers the mechanical construction of the robots, including modification of components for this application.

Chapter 4 outlines basic motor control methods, the motor driver control method used and the power drive circuit development.

Chapter 5 discusses the Global Positioning System (GPS) and its limitations.

Chapter 6 details the software written, including sensor interfacing, communication, motor control and the human interface.

Chapter 7 presents the results of the mobile platform and GPS tests. The platforms developed are evaluated and future work is discussed. The entire thesis is then summarised.

2 MOBILE PLATFORM DESIGN AND COMPONENT SELECTION

2.1 PRIOR RESEARCH OF THE MECHATRONICS GROUP

The University of Waikato Mechatronics Group has developed a number of large scale robots. The robots differ significantly from one another, each having been designed and built for a specific environment. The mechatron fields range from underwater reconnaissance and outdoor exploration to an indoor security robot.

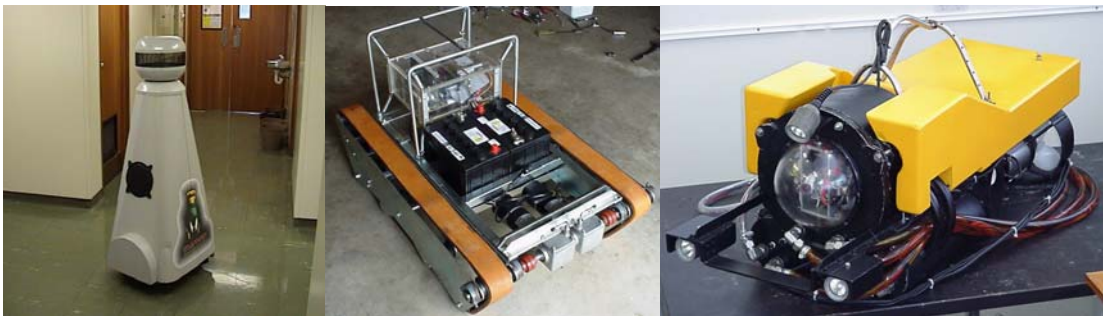


Figure 2-1: LEFT: MARVIN; CENTRE: Tracked vehicle; RIGHT: Underwater ROV

Three of the Mechatronics Group's projects are shown in Figure 2-1. The first, MARVIN, is an indoor security robot designed to patrol the corridors at night interacting with any found occupants or intruders within the building. When traversing multi terrain environments, the tracked vehicle offers high torque and mobility using its drive capabilities. Equipped with an onboard power supply and processing capabilities, it is designed to offer unrestrained passage through rough terrain. Underwater exploration is the forte of the Remotely Operated Vehicle (ROV). Controlled from the surface, the ROV is equipped with an onboard camera, PC and a range of sensors.

These devices excel in their field of expertise, but various limitations deem them not suitable for investigating autonomous cooperative behaviour. Two identical outdoor mobile platforms will be built to facilitate this research.

2.2 DEVELOPED MOBILE PLATFORMS

A small number of suitably designed mobile platforms are made by manufacturers around the world. Current robots which meet the design specifications are mainly used for research or remote bomb disposal and are sparse and expensive. Most robotic arms are designed to sit on a factory floor, are large, heavy and use a substantial amount of power. Whereas mobile robot designs are usually hobby kit sized; designed for use indoors on a smooth floor. A number of manufactured designs which meet the requirements of this project are shown in this chapter, providing features and prices where applicable.

2.2.1 MR-5

The MR-5, shown in Figure 2-2, is a bomb disposal robot built and sold by ESI (Engineering Service Inc.[1]). It is a remote controlled platform, with a manipulator arm consisting of up to eight joints.

The MR-5 specifications are:

Figure 2-2: ESI's MR-5

- Dimensions: 70 × 127 × 79 cm (w × l × h)
- Weight: 250 kg (Standard)
- Incline Ability: 45°
- Battery Power: 3 – 5 hrs
- Maximum Speed: 2.52 km/hr
- Manoeuvrability: 360° turn in a 1.4 m enclosure
- Drive Control: Proportional control of drive motors
- Lift: Horizontal Reach 1.7 m, 20 kg payload with arm fully extended (Figure 2-3)
- Cost: US$80K to $140K depending on the configuration

Figure 2-3: The MR-5 with 1.7 m reach capability

The MR-5 is controlled by a human operator via a remote station. The communication is either wireless RF (Radio Frequency) with a 500 m range (line-of-sight), or cable with a range of 200 m. Tracks can be fitted over the wheels for a rougher environment. Optional sensors include a pan-and-tilt zoom camera providing video and sound feedback to the user through the control station, an infrared camera, and an x-ray mounting assembly.

2.2.2 Bomb Disposal Robot

RoboProbe Technologies Inc.[2] produce a remote controlled bomb disposal robot shown in Figure 2-4. The robot is similar to the MR-5, but is smaller and lightweight. Permanent tracks are used rather than wheels. The robot's specifications are:

- Dimensions: 43.5 × 91.5 × 40.5 cm (w × l × h)
- Weight: 35-45 kg
- Incline Ability: 38° climbing stairs
- Maximum Speed: 2.52 km/hr
- Battery Power: 2 – 3 hrs
- Lift: Horizontal reach 96.5 cm from front edge of platform, 4 kg payload

Figure 2-4: RoboProbe Technologies Inc. bomb disposal robot.

Incorporated sensors are 3 cameras; IR (Infrared) camera with pan-and-tilt capability and IR light source (on a high mounting), colour drive camera with halogen lights (at the base of the arm), and a colour camera mounted on the claw.

2.2.3 MURV-100

The MURV-100 is similar to the MR-5, but very lightweight. Shown in Figure 2-5 fitted with ten wheels, these can be changed for either six large wheels or tracks.
The MURV-100 specifications are:

- Dimensions: 43 × 58 × 30 cm (w × l × h)
- Weight: 23 kg
- Incline Ability: 35°
- Maximum Speed: 0.39 km/hr
- Battery Power: 2 – 4 hrs
- Lift: Horizontal reach 66 cm from front edge of platform, 4.5 kg payload
- Cost: US$35K

Figure 2-5: The lightweight MURV-100

Manufactured by Defenders Network Inc.[3], the MURV-100 offers communication by either a cable or by a standard wireless system, up to 300 m in each case (line of sight required for wireless operation). Optional sensors include a tilt-and-pan camera, a claw camera, and sensors capable of detecting toxic gases, radioactive materials and explosives.

2.2.4 PowerBot

The PowerBot fitted with a PowerArm, Figure 2-6, is designed for research use. A number of sensors are incorporated into the design to complement this, allowing control to be provided by the robot itself rather than from a remote user. It is designed for indoor use, with a small ground clearance and limited traction on rough terrain.

Figure 2-6: PowerBot with PowerArm from Activemedia Robotics [4]

PowerBot specifications:

- Dimensions: 62.5 × 85 × 43 cm (w × l × h, without arm attached)
- Incline Ability: 8.5°
- Maximum Speed: 6 km/hr

- Battery Power: 2 hrs
- Lift: Horizontal Reach 80 cm from front edge of platform, 2 kg payload
- Cost: US$25K to $85K depending on the options chosen.

The PowerBot has a HitachiH8S-based microcontroller onboard, along with shaft encoders and sonar sensors to provide control. An onboard PC, laser scanner, camera and other optional extras are available. Drive is provided to two wheels, with following casters provided for support, implying that the unit's weight must be balanced for effective operation.

2.2.5 Mobile Robotic Arm Kit

Mobile robotic arm kits are available from Lynxmotion[5], Figure 2-7. Although these are only miniature robots (the base is approximately 20 cm × 20 cm), they provide a simple and effective design.

Figure 2-7: Lynxmotion mobile robotic arm kit

The major limitation of using such a unit for development is the physical size of sensors and a controller if mounted onboard. However, they provide a good simulation tool for representing a larger scale design at a low cost.

2.2.6 Other Designs

A range of other designs are available, dependant on the requirements. Some photos of other bomb disposal robots are shown in Figure 2-8.

Figure 2-8: Other bomb disposal robots

2.3 DRIVE SYSTEMS

A number of drive systems are suitable for outdoor motion, each with their own advantages and disadvantages. The preferred system depends on the intended application and its constraints. A summary is shown of common arrangements, discussing the abilities and limitations of each.

2.3.1 Legged Motion

A unit propelled using legs has an adherent advantage of being able to traverse rough terrain, and manoeuvre over obstacles of significant size with respect to its own dimensions.

Figure 2-9: LEFT: Asimo, Honda's humanoid robot; CENTRE: Hamlet, University of Canterbury NZ's hexapod; RIGHT: ARL Monopod II, McGill University, Canada.

As shown in Figure 2-9, a number of leg configurations are possible. These designs are complicated, and demand a large amount of coordination between the motion of each leg. Multiple actuators are generally used to produce the required movements. Manoeuvrability can be limited, as is the speed at which the mechatron can move.

2.3.2 Skid Steering

Skid steering or differential steering uses rigid parallel sets of wheels on either side of the vehicle. To generate a turn, the wheels on each side of the vehicle are driven at different velocities. The different velocities define the turning radius. This method of steering is used by the MR-5 (section 2.2.1) the Bomb Disposal Robot (section 2.2.2), the MURV-100 (section 2.2.3), and also by tracked vehicles such as military tanks and bulldozers (refer Figure 2-10).

Figure 2-10: A bulldozer relies on skid steering to turn

By using a symmetrical design and driving the wheels/tracks on either side at equal velocities in opposing directions, the unit's turning centre will coincide with its geometric centre, shown in Figure 2-11.

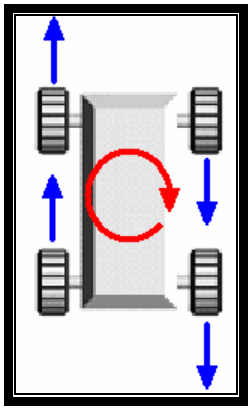


Figure 2-11: Skid steering performing a point turn

This provides high manoeuvrability as the unit is capable of turning on a fixed point. The major disadvantage is the power required to perform a turn, especially on a stationary position, as the wheels must move in a lateral direction as well as moving forward. This increases power consumption, forces that the design must withstand, and the wear on the tyres/tracks. Skid steering is therefore most suitable for applications involving rough terrain, and/or terrain with a loose or soft surface. A skid steered vehicle which is symmetric about its axis has an advantage of equal manoeuvrability in either a forward or reverse direction.

2.3.3 Articulated Steering

To turn a four wheeled vehicle with minimal lateral forces on the tyres, the outside tyres must turn with a larger radius than the inside tyres. Articulated steering accomplishes this by having a piece of the chassis or axle hinged, shown in Figure 2-12. The unit steers by folding the joint, providing each wheel with a tangential direction about the turning centre.

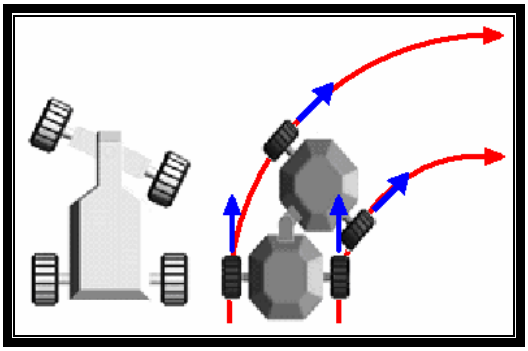


Figure 2-12: LEFT: Articulated axle; RIGHT: Articulated frame

The efficiency of articulated steering is much higher than that of skid steering; however the turning radius is larger, reducing manoeuvrability. The mechanical design is still very simple. Normally drive is applied to two wheels from a single motor. This requires a differential to be used to allow the inside wheel to travel a shorter distance than the outer wheel.

2.3.4 Ackerman Steering

Ackerman steering is another method which overcomes the problem of the inside and outside radii being different; reducing the forces on the tyres. This is the most common type of steering for automobiles. The tyres remain in a fix position, but turn to different angles on each side of the vehicle, illustrated in Figure 2-13.

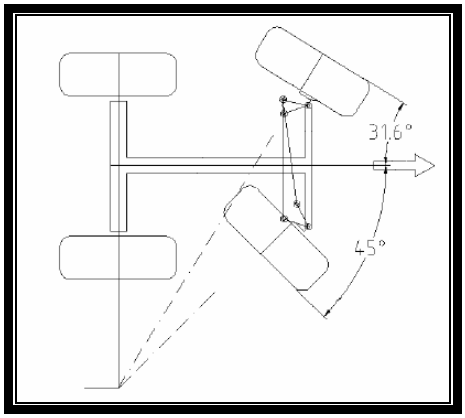


Figure 2-13: Ackerman steering

The coordination of the angles required increases the mechanical complexity of the design. Figure 2-14 shows that as the inside wheel varies over a range of 0-50°, the outside wheel must non-linearly vary over 0-32°. Less clearance around the turning tyres is required compared to the articulated axle design. A rigid chassis is usually more favourable than an articulated chassis.

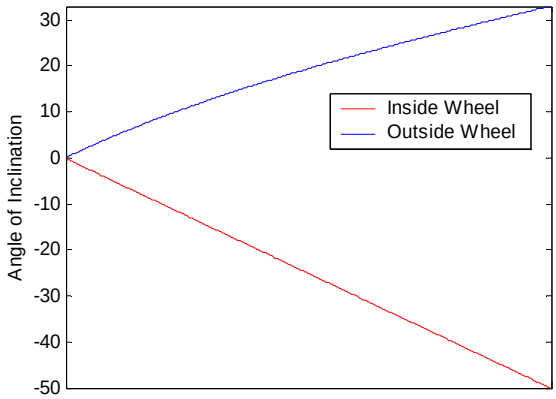


Figure 2-14: A typical variation of inside and outside wheel angles for Ackerman steering

As the tyre angles required are dependant on the chassis dimensions, each vehicle must be uniquely designed. Most mechanical designs are unable to match the requirements exactly, but offer a close estimate.

As the turning radius is large, manoeuvrability is low. By turning the front and rear sets of wheels, the turning radius can be reduced. Another option is to drive "backwards" with the steering wheels at the rear of the vehicle. This is used extensively in forklift design to increase manoeuvrability, refer Figure 2-15.

Figure 2-15: Forklift with Ackerman steering, steered by the rear wheels

2.3.5 Independent Explicit Steering

Independently steering each wheel, shown in Figure 2-16, allows maximum flexibility of movement for a wheeled vehicle. The unit can perform skid, and Ackerman steering techniques. Control and accuracy of the steering angles make this design more complex and expensive; however the manoeuvrability is extremely high, especially in confined spaces.

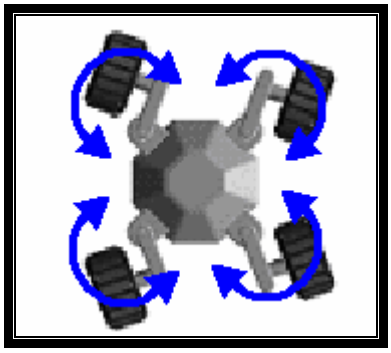


Figure 2-16: Independent explicit steering

A four wheeled vehicle has the potential to perform point turns by turning all of the wheels perpendicular to its geometric centre. This is very efficient as the wheels experience no lateral forces. A unique advantage of independent steering is the ability to "crab steer". This is achieved by setting all of the wheels to face the same direction, whereby the vehicle can then drive sideways with respect to the chassis. Usually all of the wheels are driven, so a four wheel vehicle requires four motors for steering and four motors for driving the wheels.

2.3.6 Wheelchair Configuration

The wheelchair configuration is a form of differential steering, similar to skid steering, where a difference in velocities between the left and right wheels generates a turn. The main difference between the wheelchair configuration and skid steering is that the wheelchair uses passive casters to support the chassis. Shown in Figure 2-17(a) is a standard electric wheelchair with two casters placed in front of the driving wheels. In Figure 2-17(b,c), MARVIN the security robot being developed at The University of Waikato has a caster placed in front of, and behind, the driving wheels.

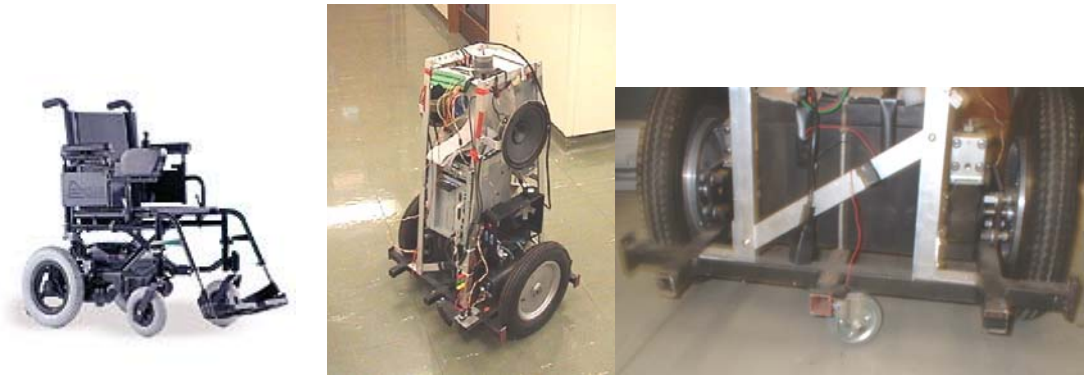


Figure 2-17: LEFT: Standard electric wheelchair; CENTRE: MARVIN the security robot; RIGHT: MARVIN's caster positioning

This configuration takes advantage of the manoeuvrability of the skid steering configuration, while greatly reducing the energy required to perform a turn. By driving the wheels in opposing directions at a common velocity, a point turn is achieved with the turn centre lying at the midpoint of the driving wheels. As MARVIN's geometric centre is at the midpoint of the driving wheels, a turn can be

made on a stationary point. The wheelchair shown in Figure 3-9(a) has the midpoint of the driving wheels at the rear of the unit, requiring a larger turning radius as the front of the chair sweeps around.

The wheelchair design generally is confined to using small casters due to the fact that larger and heavier casters are incapable of pivoting freely. This limits the size and the terrain capabilities of the vehicle. The design is most suited to vehicles with a short length to maintain the advantage of point turns in a minimal area.

2.3.7 Tricycle

The tricycle is a simple design which reduces complexity, components, and cost. A standard configuration uses two fixed-direction wheels at one end of the vehicle, and a steering wheel at the other end.

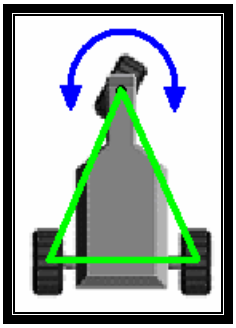


Figure 2-18: Tricycle arrangement. The centre of mass must remain inside the triangle

Normally the pair of fixed wheels is driven, although the steering wheel can be used to drive the vehicle. This design suffers from stability problems, as the centre-of-mass must remain within a triangle whose corners lay on the centre of each wheel (illustrated in Figure 2-18). On a sloped terrain the unit is liable to topple over; however limiting the height minimises this problem. Suspension is not essential for a tricycle design, whereas four wheeled vehicles require it to maintain contact between the ground and all of the wheels on uneven terrain.

2.3.8 Chosen Drive System

A tricycle design was selected for this project as it meets the requirements of:

1) Providing a simple design. Suspension or coordinated steering systems are not required when constructing a simple tricycle.
2) Minimising cost by reducing components required.
3) Being capable of adequately traversing a smooth outdoor terrain.
4) Offering manoeuvrability. It cannot perform point turns; however driving with the steering wheel at the rear will increase manoeuvrability.
5) Stability is offered by keeping the vehicle's height to a minimum. By locating heavy components near the bottom of the design, the centre-of-mass is lowered, reducing the chance of toppling.

2.4 DRIVE COMPONENTS

With the chosen tricycle design, drive will be applied to the fixed pair of wheels to reduce the amount of slippage that would occur if driving just the steering wheel. When driving both wheels, an allowance must be made for turning tight corners, since the inner wheel will travel a shorter distance than the outer wheel, as previously stated in section 2.3.3. Two solutions are available to this problem:

1) Drive each wheel independently using two separate motors.
2) Drive both wheels from a single motor through a differential gear, as used in automobiles. A differential gear is an arrangement of gears in an epicyclic train permitting the rotation of two shafts at different speeds[1].

The first solution requires an additional motor, gearbox and control circuitry. An advantage offered is that by increasing the power to one drive wheel, the robot will experience a rotational force which can assist steering during tight turns in a similar way that a wheelchair configuration steers.

The alternative solution only uses one motor and one reduction gearbox. This increases efficiency while reducing the required control sophistication and circuitry.

[1] *The American Heritage® Dictionary of the English Language, Fourth Edition*

Mechanical complexity is slightly increased by adding a differential gearbox. This was the method chosen.

2.5 DRIVE MOTOR SELECTION

The drive motor must run from a battery supply, preferably either 12 or 24 V DC as these are common voltages when using lead-acid batteries (refer section 2.9). The required output velocity range is approximately from 0 - 1 m/s with the unit fully loaded. The motor must be controllable at low speeds. The gearbox and drive train should be as efficient as possible, with minimal "free play" to aid the controllability. The cost of the motor must be kept minimal for this project.

Two motors, differential gears and axles were acquired, originating from mobility scooters (Figure 2-19). This selection was adequate for this project's requirements, with financial expense minimised compared to high precision new motors. The motors are 24 V DC, 400 W, with an electro-mechanical brake. The electro-mechanical brake can be released by moving a lever, and is not required for this project as it normally functions as a "park" brake. This motor is suitable as it offers the required speed range, and is capable of carrying a 100 kg payload.

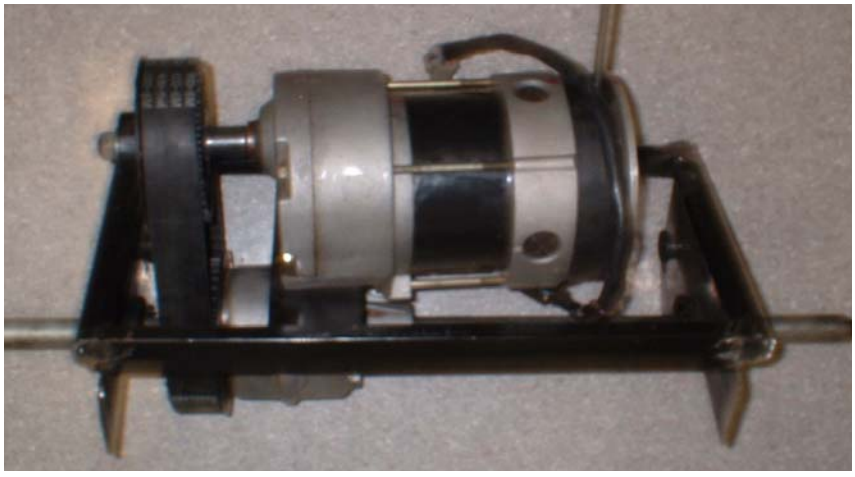


Figure 2-19: Drive motor and differential gear configuration

The rating of this motor is 24 V DC 400 W, therefore the continuous current rating is:
400 W / 24 V = 16.67 A

Experimentally the unloaded current is 3 A, and a significant load brings the current up to 10 A. While operating the robot, 10 A can be expected as the maximum current unless the motor is stalled, i.e. the robot drives against a solid object and cannot move.

2.6 STEERING MOTOR SELECTION

The steering motor changes the orientation of the single wheel, and will be required to travel over a maximum angle range of ±90°. The angle limit used will be determined experimentally once the robot is built, although it is estimated to be ±60° on a high friction surface.

The motor must have sufficient torque to be able to turn, and also hold the wheel in position. A 24 V DC motor is preferred as this will match the drive motor (refer section 2.5), providing a common working voltage, rather than using multiple voltage rails for different components. Size should be minimal to keep the robot's overall dimensions compact. Again cost must be minimised.

2.6.1 Steering Motor Options

One simple solution is to use a stepper motor. A stepper motor moves in small increments and offers high precision angle of movement. The electronic control complexity required to drive a stepper motor is higher, and the torque much lower than a permanent magnet DC motor. The cost of a stepper motor is high, making other alternatives more favourable.

A second solution is to use a servomotor. A servomotor moves to a target position input by a user. It consists of a sensing element, an amplifier and a motor. The target position is entered (usually with a pulse-timed signal) and is compared to the actual position obtained by the sensing element. The error from the comparison is amplified and feed into the motor, driving the motor towards the target angle. Normally a servomotor operates over a ±90° range.

Servomotors are used in hobby remote controlled “toys” such as cars and aeroplanes. These motors operate from a 6 V DC supply and are very small. Precise timing is required by the target position input signals. Position sensing is performed using a potentiometer, and movement uses a proportional control method. The torque provided by these hobby motors range up to 1 Nm. This does not meet the torque requirements of this design, expected to be greater than 10 Nm.

Larger 24 V DC servomotors are available from overseas manufacturers. They offer extremely accurate output positioning and a simple serial input interface. Sensing is provided by optical shaft encoders, and a PID control method is used. The specifications and cost of these units are too high to be considered for this application, in excess of US$3000 for a 24 V DC 125 W position controlled servo with serial (RS-232) input.

A third solution is to use a linear motor configuration, where a linear movement is converted to a circular motion, illustrated in Figure 2-20.

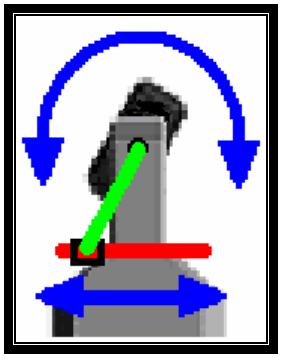


Figure 2-20: Converting a linear movement to a rotational motion

This solution requires a large amount of area for the linear actuator, and is therefore less suitable than the other solutions.

2.6.2 Motor characteristics

As pre-built solutions are expensive, or do not meet the required specifications, a low cost servomotor could instead be built. This requires the addition of a position sensor

to a motor to provide feedback. A basic proportional control method of a servomotor is shown in Figure 2-21.

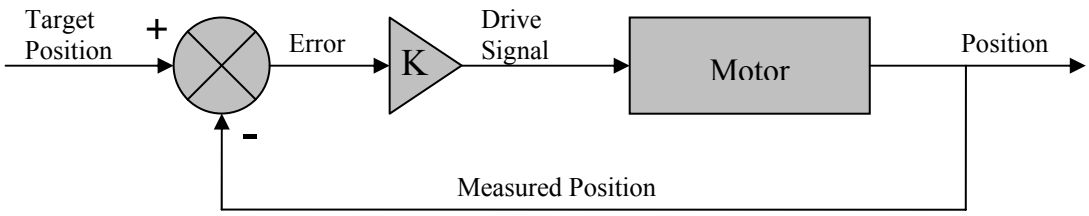


Figure 2-21: Feedback of a proportional controlled servomotor

A target value is input to the system. Subtracted from this is the current position, giving an error. The error is amplified by a constant (K), with the resulting signal driving the motor. The motor moves with power proportional to the error, i.e. a small error gives little power to the motor, a large error results in the motor being given full power. As the motor turns, the position is fed back to the beginning of the controller. This method is simple, although has some disadvantages such as a limited output response and a steady-state error.

A permanent magnet 24 V DC motor and gearing from a truck windscreen wiper was selected as the basis for the servomotor, illustrated in Figure 2-22(a). It was cheap, strong, ran from a 24 V DC supply, the gear reduction ratio was sufficient, and a position sensor could easily be fitted to the output.

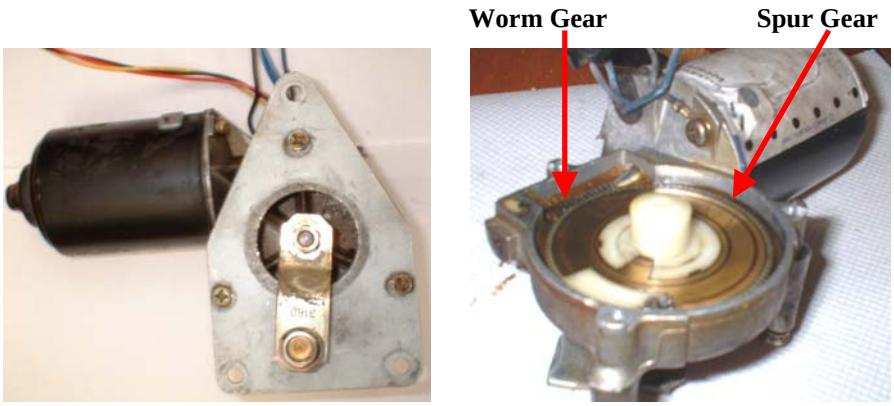


Figure 2-22: (a) 24V Windscreen Wiper Motor; (b) Gear Configuration

The motor shaft comprises a helical worm gear, which drives a spur gear, shown in Figure 2-22(b). The worm gear moves one tooth per revolution, and the spur gear has

99 teeth, giving a 99:1 ratio. The large reduction ratio gives the output an extremely high torque, and makes position movement easier to control as the output shaft moves slowly.

The continuous unloaded current drawn by the motor is approximately 2 A, with the peak current approximately 6 A.

2.7 ONBOARD HARDWARE ARCHITECTURE

A variety of hardware architectures are available to perform the data input/output, communication, control and planning algorithms required by this project. Cost, power consumption, size, and functionality are major factors that influenced the decision.

Some possibilities are:

1) Microcontroller. A microcontroller is small, data input/output (I/O) is performed easily, power consumption is low, but functionality is limited.
2) Handheld PC. A handheld PC is small, uses low power, and offers a reasonable amount of processing power. Bugs and glitches in underdeveloped compilers and drivers can slow down development. Data I/O is difficult.
3) Laptop PC. A laptop computer offers immense processing power compared to the possibilities above. The cost is high, and complex data I/O requires expensive additional hardware.
4) Full-sized PC. A full-sized PC offers the highest processing capabilities while increasing size and power consumption. The cost is much lower than a laptop. Complex data I/O requires additional hardware.

A full-sized PC was chosen for its vast processing capabilities – offering unrestricted development in algorithm processing, such as navigation and path planning. Hardware such as data I/O cards and wireless network cards are widespread, offering a large range of choice with little restriction on compatibility of drivers. This was a lower cost alternative to a laptop.

The specifications of the selected system are:

- 800 MHz Celeron
- 256 MB RAM
- 20 GB Hard drive
- Windows XP Professional

2.8 CPU POWER SUPPLY

A suitable power supply is required to run the PC. The power supply must offer the following rails (typical amp rating shown for each rail):

- +12 V @ 7 A
- +5 V @ 25 A
- +3.3 V @ 8 A
- -5 V @ 0.5 A
- -12 V @ 1 A
- +5 V Standby @ 0.75 A

Previously the Mechatronics Group have used an uninterruptible power supply (UPS) to convert the 24 V battery supply into 230 V 50 Hz AC. This directly feeds into a standard ATX power supply of a desktop PC. This is a quick, easy solution but has the following major disadvantages:

- Efficiency is low converting 24 V DC to 230 V AC and then back to 12 V DC and the other required voltages.
- The size of the UPS is large, as illustrated in Figure 2-23.
- The UPS weighs 10.9 kg
- To turn the UPS on it must be connected to 230 V mains supply. For outdoor use, this means that the unit must be started in vicinity of a 230 V supply and remain in a standby condition until power up is required.

Figure 2-23: LEFT: Olympic OTC-1000A UPS next to a 40 cm ruler; RIGHT: ACE-828C Industrial ATX power supply

A better solution is to use an industrial power supply designed for the input voltage available, however these are only available from a small number of manufacturers and suppliers. An ACE-828C industrial ATX power supply was purchased, shown in Figure 2-23, offering an 18 – 32 V DC input range. It has the same dimensions as a standard 230 V ATX supply.

A comparison of efficiency between using a UPS/230 V AC supply and the 24 V DC supply is shown in Table 2-1. The current shown is the amount drawn by each power supply option while the CPU performs a defined task.

	UPS/230 V AC[1]	24 V DC[2]	Increase in Efficiency
Maximum[3]	3.17A	2.20A	30%
Average Working[4]	2.89A	2.06A	29%
Idle[5]	2.21A	1.41A	36%
Standby[6]	0.79A	0.18A	77%

Table 2-1: Comparison of power consumption of CPU supply. Current measured at 24 V DC input.

[1] UPS used was Olympic Model OTC-1000A, with a 230 V AC, A Open Model FSP235-60GT ATX supply.

[2]24 V DC supply manufactured by ICP Electronics, Model ACE-828C.

[3]Maximum defined as highest peak current observed.

[4]Average working defined as average current drawn while performing a scandisk operation on the hard drive.

[5]Idle defined as power on with no processing or disk access operating.

[6]Standby defined as CPU turned off, with only the +5 V standby rail active on the motherboard.

The efficiency increase is not linear, but for this project a 30% increase is expected based on the results shown in Table 2-1. The size and weight of the CPU power supply is significantly reduced as the UPS is not required.

2.9 BATTERY SELECTION

Many varieties of batteries are available on the market, each specialising in different applications with various demands. The requirements are:

- 24 V DC
- Approximately 10-15 Amps continuous current
- Approximately 28 Amps maximum current
- Operation time above one hour during development.

The current is estimated as 2 A continuous, 2.5 A peak for CPU, 5-10 A continuous, 17 A peak for drive motor, 3 A continuous, 8 A peak for steering motor and other electronics.

Lead-acid batteries, such as used in automobiles, are readily available and are an inexpensive option for this application while easily meeting the power requirements during development. However an automobile battery (or starting battery) is designed to provide a large current for a short length of time, with infrequent charging. An alternative *deep cycle* battery is designed to be charged and discharged repeatedly. A deep cycle is defined as a battery being discharged 80% and being recharged to full capacity. The current and deep cycling ability is an attribute of each battery design.

A starting battery has thin lead plates made in a sponge-like fashion to increase their surface area, allowing a very high current to be provided while starting a vehicle. During deep cycling, the lead can be consumed and fall to the bottom of the cell. The battery life is directly related to how deep the battery is cycled. Failure generally occurs after approximately 30-150 deep cycles (80% discharge), although thousands of cycles can be achieved under normal starting conditions (2-5% discharge). If the cycles are reduced to 50% discharge, the battery is expected to last twice as long as 80% discharge cycles.

A deep cycle battery has plates made of solid lead. This allows for discharging and recharging without deterioration of the plates, therefore without reduction in capacity or premature failure. The maximum current rating is reduced. The prices of these batteries are much higher than those of starting batteries.

For final design, deep cycle batteries would be crucial, but during development starting batteries are sufficient due to the nature of testing, i.e. frequent short run periods. The chosen batteries are two 12 V starting batteries in series to provide 24 V. The specifications are given in Table 2-2.

Manufacturer	Champion
Model	E360C Power Plus
CCA	360 Amps
RC	60 Minutes

Table 2-2: Battery specification

New Zealand battery manufacturers have adopted the SAE (Society of Automotive Engineers) standards to provide a rating to each battery. The CCA rating is "*Cold Cranking Amps* which is the discharge load in amperes which a new fully charged battery at -18°C can deliver for 30 seconds and maintain a minimum voltage of 1.2 volts per cell." The RC is the "*Reserve Capacity* (minutes) which is the time in minutes that a new fully charged battery will supply a constant load of 25 amps at 25°C without the voltage falling below 10.5 volts for a 12 volt battery"[6].

For this project, the CCA is irrelevant, but the RC provides the running time of the robot. These batteries will provide approximately two hours of operation when fully charged. To extend the life of the batteries, this should be limited e.g. a maximum of one hour running time will double the life of the battery.

2.10 OTHER COMPONENTS

A number of other essential components were chosen for this project, and are outlined below.

2.10.1 Wireless Network Cards

Communication between the robots and also to a base station will be achieved using Netgear 401 wireless network cards, illustrated in Figure 2-24.

Figure 2-24: Netgear wireless network card

These are a standard 2.4 GHz wireless network card, using the 802.11b protocol. Transfer speeds are up to 11 Mbit/sec with a strong signal.

2.10.2 Data Acquisition (DAQ) Cards

National Instrument's Lab-PC+ card (Figure 2-25) provides data I/O to the CPU.

Figure 2-25: National Instrument Lab-PC+ DAQ card

The LabPC+ card provides:

- 24 Digital Input/Output Lines
- 8 Analogue Input Lines
- 2 Analogue Output Lines
- 3 Counters/Timers

2.10.3 GPS Receivers

GPS positioning is to be achieved using Motorola M12 Oncore receivers, illustrated in Figure 2-26. The GPS receiver is a 12 parallel channel receiver, capable of tracking 12 satellites at once. A position is reported once per second through serial data transmission. GPS is discussed further in chapter 5.

Figure 2-26: Motorola M12 Oncore GPS receiver

2.10.4 Shaft Encoders

Shaft encoders will provide position information from the main drive wheels. The encoders selected are HEDS-5701 panel mount optical encoders. They are optical encoders; providing a 500 count per revolution quadrature output.

Figure 2-27: HEDS-5701 optical shaft encoder

2.10.5 IR Detectors

Sharp GP2Y0A02YK infrared (IR) object detectors will provide a distance measurement to objects within a 20 - 150 cm range, with an accuracy of ±15 cm. By

characterising and filtering each individual detector the accuracy is expected to be increased to ±5 cm.

Figure 2-28: Sharp GP2Y0A02YK infrared object detector

The detectors provide an analogue 0.25 – 2.85 V DC output corresponding to the distance measured, updating every 32 ms. The detectors use a triangulation measuring method (rather than reflected intensity) to allow less variation between different coloured objects being detected.

3 MECHANICAL DESIGN AND ASSEMBLY

The tricycle chassis design must incorporate the dimensions and weights of the known components from chapter 2. The overall size of the mechatron can be minimised by specifically designing the chassis for these components, reducing redundant space. By reducing the size, the manoeuvrability is increased providing a tighter turning radius.

The chassis design must consider the following additional attributes:

- Minimise the height to increase stability
- Incorporate all components into suitable locations (significant weight low down, regularly modified or adjusted components easily accessible)
- Distribute weight evenly within the centre of the wheels (refer section 2.3.7)
- Counterbalance weight from an overhanging arm when supporting the maximum external payload
- Provide adequate ground clearance for outdoor application
- Allow for future expansion, such as the addition of different sensor types.

3.1 PC CASING

The PC motherboard is a standard desktop model. As a standard desktop case is quite large, a custom case had to be built to hold the motherboard, minimising the overall size. The ATX power supply should ideally be built into the same enclosure to reduce the separation from the motherboard.

This was achieved by modifying an existing case, keeping the backplane and existing motherboard and power supply mounting points, as shown in Figure 3-1. All excess area below, and in front of, the motherboard was cut away – leaving a 5 mm clearance around the motherboard. The excess material was then used to make panels to replace the cut away areas.

Figure 3-1: Reduction of a PC ATX case, keeping only minimal area required.

The hard drive is mounted in a bracket above the CPU to allow unobstructed access to the PCI and ISA card slots. The vent on the top and the exhaust fan in the power supply ensure constant airflow through the case, providing cooling.

The overall dimensions of the original case were: 465 × 200 × 455 mm (L×W×H)
reduced to: 365 × 200 × 230 mm (L×W×H)

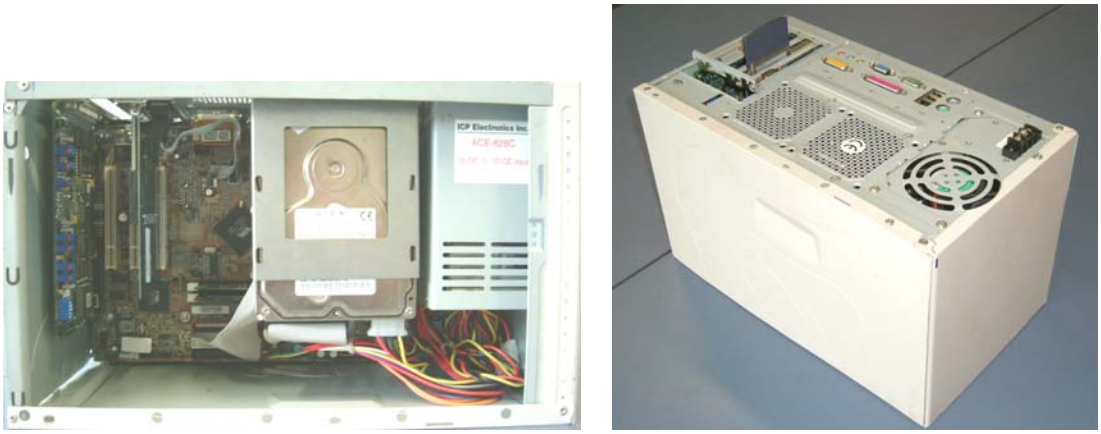


Figure 3-2: Complete PC and ATX power supply housing

3.2 CHASSIS

The chassis design provides mounting and support for all of the major components. Constructed from 16 mm box section, 20 mm and 25.4 mm right angle steel, the chassis provides a robust base.

Designed with the driving wheels at the front, the motor is positioned above the drive axle. This increases tyre traction by applying the weight of the motor onto the two drive wheels. The differential gearbox, which is part of the axle, limits the overall ground clearance of the mechatron to 65 mm. The remainder of the chassis has a ground clearance of 100 mm to prevent the mechatron from bottoming out on uneven terrain.

The manipulator arm (when constructed) needs to be kept short to reduce the torque required to operate it, while being nimble and offering the maximum reach capability possible. To achieve this, the arm will be mounted as low and as far forward as possible. Keeping the manipulator low to the ground aids picking up objects with minimal effort, while the reach capabilities are extended by mounting it near the front. For this design, it corresponds to a mounting directly above the drive motor, approximately 350 mm above the ground.

The batteries provide a significant proportion of the overall weight of the mechatron. Maintaining the centre of gravity within the area defined by the wheels (refer section 2.3.7) requires that the battery placement be in the centre of the robot. However, the batteries will be used to counterpoise the weight of the manipulator and any load it may carry, and therefore is placed near the rear of the design. This placement, combined with the motor above the front axle, distributes the weight over the entire mechatron, maximising stability.

The PC case has been placed in the centre of the design due to the considerable area it requires. Minimal access to the PC is required, as all work can be performed remotely using the wireless network, and hardware connections (such as the serial port) should seldom need to be acessed. By using the terminal server client software built into Windows XP a user can use another (Windows XP) computer to log into the mechatron's PC, and operate it remotely without the need to connect a monitor or keyboard.

The steering wheel is at the rear of the mechatron, with the motor mounted above it. The output from the motor gearbox (refer section 2.6.2) will be used to directly

change the steering angle of the wheel. The chassis must adequately support the wheel and motor mounting.

Any PCBs required to interface with sensors, drive the motors etc., will be supported by an acrylic (non-conductive) tray mounted at the top of the mechatron. This will allow easy access to the circuits during development for debugging, and also allows the addition of new circuits as further sensors are added. A rectangular steel frame is mounted above the PCB area to protect the circuits from being knocked during transportation. Although not completely encased, a cover could easily be added if required.

Figure 3-3 and Figure 3-4 show the layout and major dimensions of the chassis.

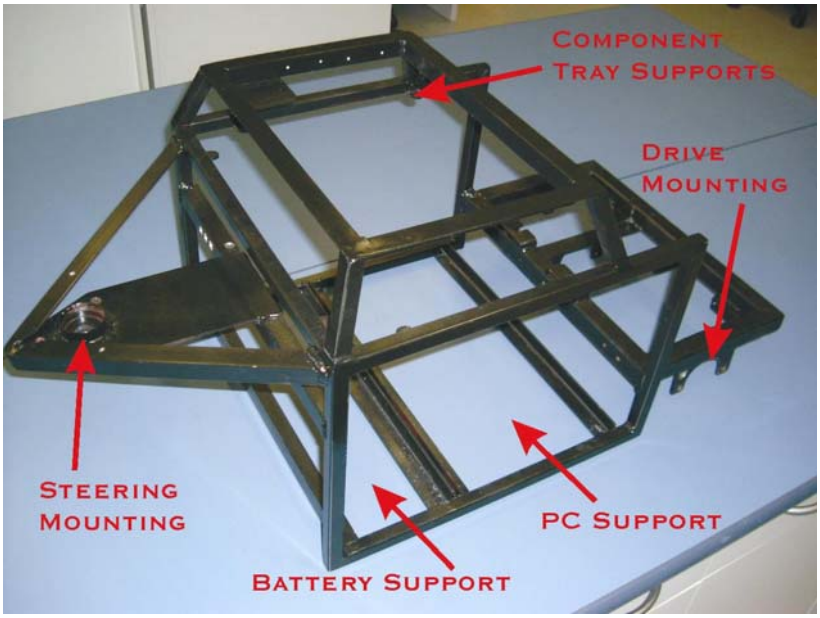


Figure 3-3: Chassis layout

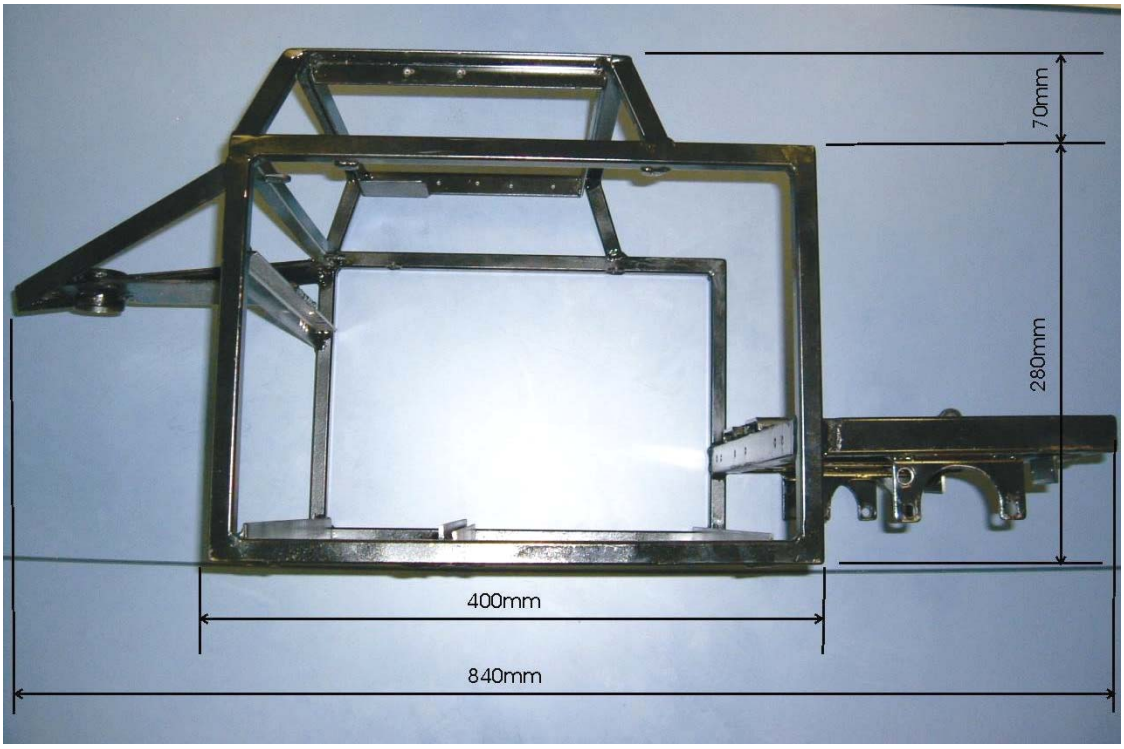


Figure 3-4: Chassis dimensions, Note: chassis width is 405 mm

With the exception of the bearing mounts for the steering (section 3.4.2) which has been brazed, the chassis has been MIG welded together.

The battery supports are made from 25.4 mm right-angle steel which hold the base of each battery along the front and rear edges. With an area of 125 mm × 370 mm, the batteries (each 125 mm × 180 mm), fit in the supports with little movement. No external restraint is required to hold each battery as the environment in which the robot will operate is limited, and therefore will not provide large enough forces to dislodge the batteries.

The PC is mounted in the same manner as the batteries, using right-angle steel across the front and rear of the base. The area available to the PC is 205 mm × 370 mm, which contains the custom built case (section 3.1). Both the batteries and PC can be removed through the side of the chassis rather than out through the top to allow easy access.

The drive and steering components of the chassis are discussed in sections 3.3 and 3.4 respectively.

3.3 DRIVE SYSTEM

The drive system consists of a motor that drives an internal reduction gearbox to a chain. The chain transfers the power to a differential gearbox, which distributes the motion to the two ends of the axle. The motor's internal gearbox uses a two stage 20:1 reduction, illustrated in Figure 3-5. The reduction calculation is shown in Equation 3-1.

$$N = (N_1 / N_2)(N_3 / N_4)$$

$$= (35 / 11)(44 / 7)$$

$$= 20:1 \qquad \textbf{Equation 3-1}$$

where: N = reduction ratio

N_1 = Number of teeth on output shaft

N_2 = Number of teeth on intermediate shaft, driving output shaft

N_3 = Number of teeth on intermediate shaft, being driven by motor shaft

N_4 = Number of teeth on motor shaft

Figure 3-5: 2nd and 3rd stages of the drive motor reduction gearing

A 1:1 ratio is used when transferring power from the motor to the differential gearbox.

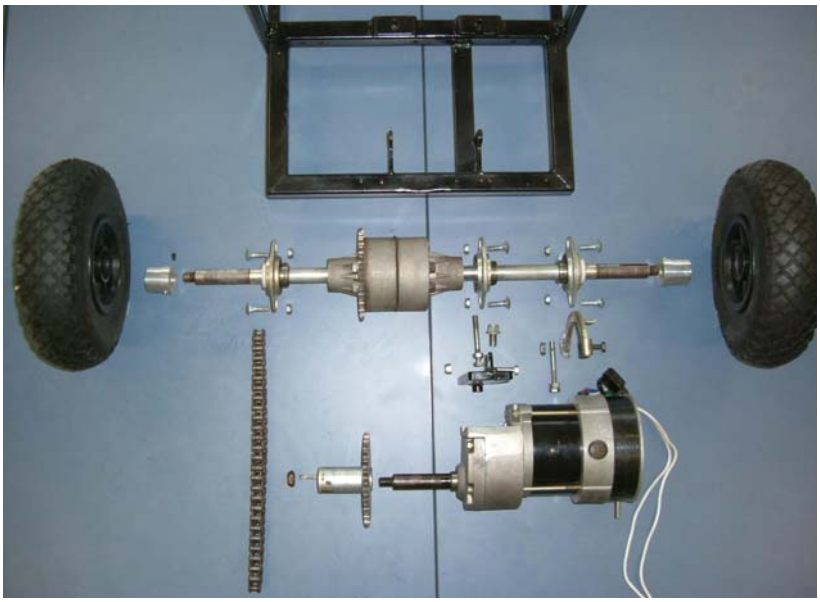


Figure 3-6: Exploded view of drive components

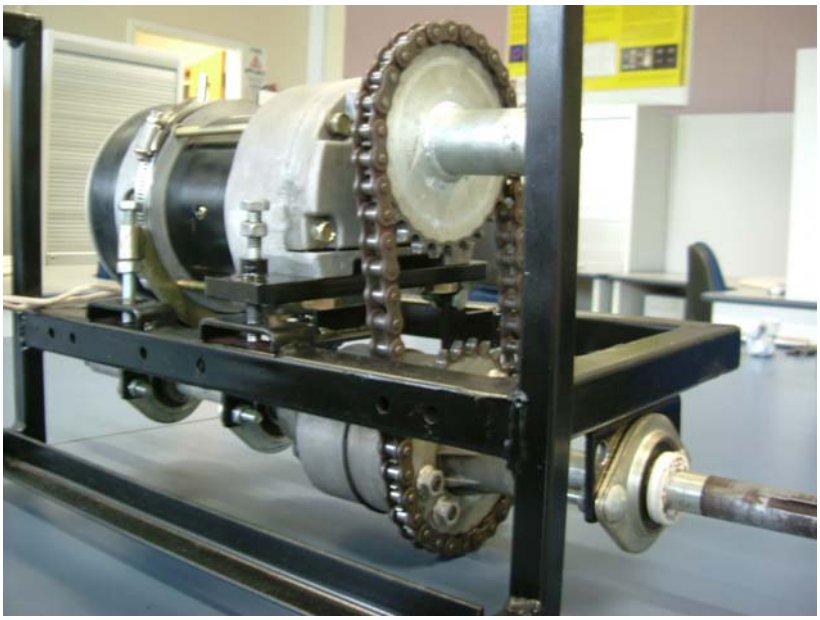


Figure 3-7: Mounted drive system showing chain tensioning method

Figure 3-6 shows an exploded view of the components used for the drive system. The chain is tensioned by adjusting the bolts between the motor mountings and the chassis, visible in the assembled drive system shown in Figure 3-7. The axle is mounted on to the chassis with three support bearings.

Pneumatic rubber tyres with a plastic centre were purchased, suitable for outdoor use with a diameter of 250 mm and a width of 80 mm. The tyres however, did not fit the

axle being used, and therefore aluminium hubs were manufactured to transfer power from the axle to the plastic wheel rims. To prevent the axle from spinning within the wheel, a square key has been fit by cutting a 5 mm wide, 2.5 mm deep grove into both the axle and the hub. When the wheel is positioned into place, a 5 mm square piece of key steel is fit into the cut-out area, visible in Figure 3-8, locking the hub into place. Two screws permanently secure the manufactured aluminium hub inside the plastic wheel.

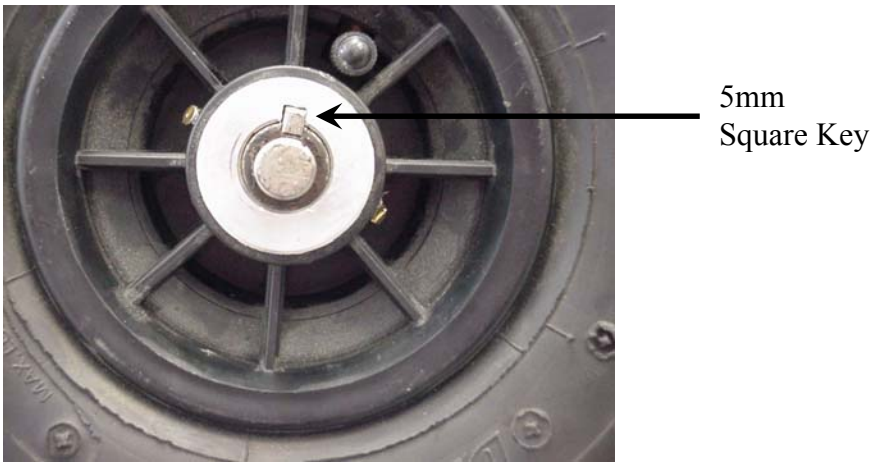


Figure 3-8: A square key prevents the hub from spinning inside the wheel

3.4 STEERING SYSTEM

The steering motor required mechanical mounting to the chassis, and modification to allow a servo motor to be made.

3.4.1 Motor Modification

To provide feedback to a control system, a position sensor needs to be mounted on the steering motor shaft. A potentiometer was chosen as it provided an absolute position, and could easily be connected to either an analogue control circuit or to the LabPC+ card.

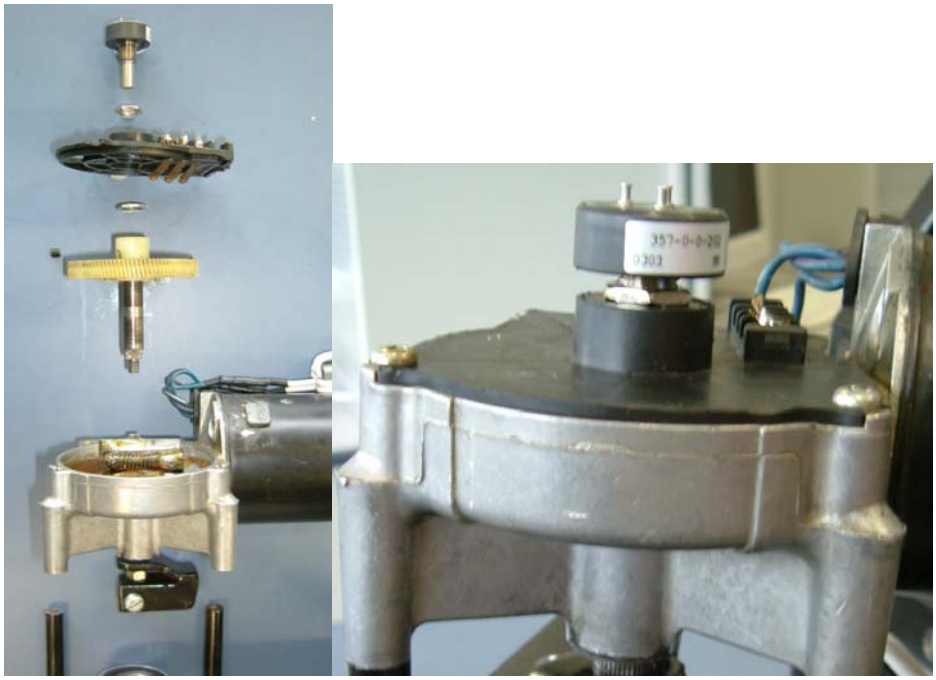


Figure 3-9: LEFT: Exploded view of steering motor and potentiometer; RIGHT: Assembled steering motor with potentiometer mounted

The potentiometer is mounted directly into the top of the output shaft, shown in Figure 3-9. The mounting was achieved by boring out the centre of the spur gear shaft, inserting the spindle of the potentiometer, and holding it in place with a grub-screw through the side of the spur gear shaft. The base of the potentiometer is screwed into the casing of the gearbox.

The wiper motor provides two different speeds through the manufacturing design. This has been accomplished by providing three brushes onto the rotor, shown in Figure 3-10.

Figure 3-10: Wiper motor brush configuration

A standard 180° configuration provides the slow speed setting, and a 115° brush configuration provides the high speed setting. With 24 V DC applied the high speed operates at 44 rpm, the low speed at 36 rpm. As the steering system does not require a high speed output, the slow setting is used which provides a higher torque.

Capacitors have been soldered across each brush inside the motor casing, as shown in Figure 3-11. Poor brush contact, dirt, or worn brushes can cause sparking between the stator and the rotor, producing high frequency noise. This can affect surrounding electronics. The capacitors provide a low impedance path to ground for high frequency noise, suppressing the amount of noise generated. The capacitors used are 100 nF, rated at 250 V.

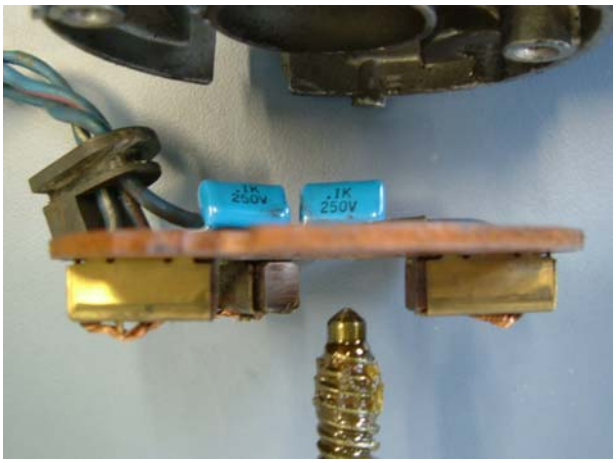


Figure 3-11: Capacitors connected between the brushes reduce the noise generated.

3.4.2 Steering Mechanical Configuration

The steering wheel purchased was the same as the drive wheels (refer section 3.3), but with the addition of bearings mounted within the hub, allowing it to spin freely on a shaft. A bracket was manufactured to support the wheel, with spacers on either side preventing the tyre from contacting the sides of the supporting bracket. The axle is secured with cotter pins on either side.

Figure 3-12: LEFT: Steering wheel mounting bracket; RIGHT: Axle with spacer between wheel and bracket

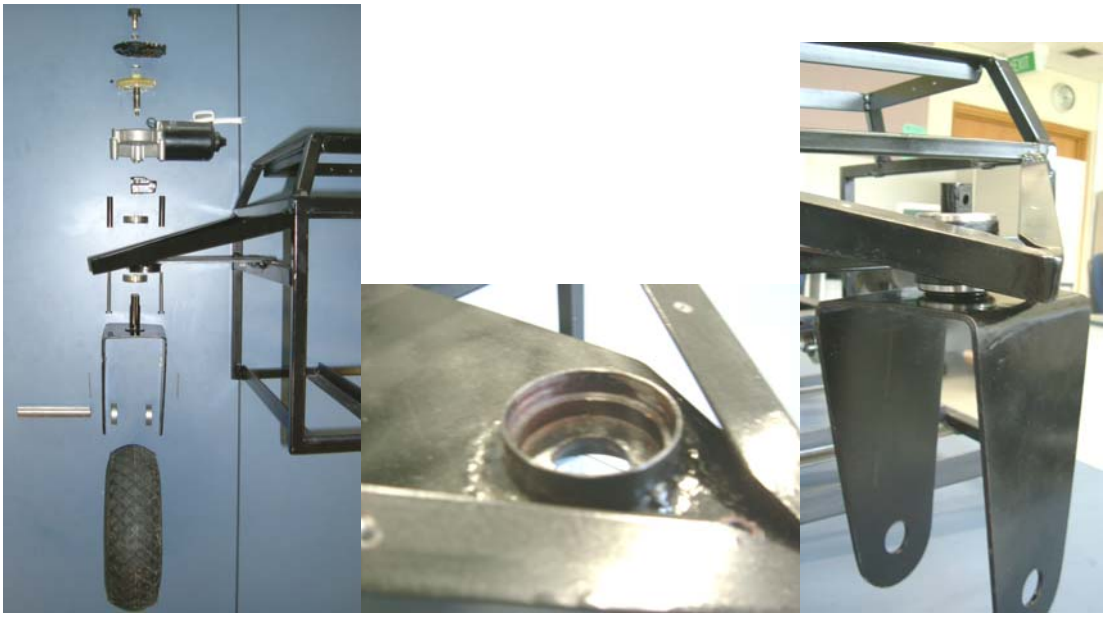


Figure 3-13: LEFT: Exploded view of steering system; CENTRE: Bearing housing for steering arm; RIGHT: Steering arm supported by upper and lower bearings.

The chassis uses two right-angle steel struts in a triangle arrangement to support the steering assembly, which is mounted through a steel plate. Figure 3-13 shows an exploded view of the components used in the steering system. The wheel bracket is mounted on to the chassis through two bearings (5/8" inside diameter, 1-9/16" outside diameter). The housing on the chassis supports the outer rim of each bearing, allowing the steering shaft to apply forces to the inner circumference. A 5/8" custom made shaft is brazed onto the wheel support. The shaft fits through the two bearings,

applying an axial force to the inside of the lower bearing through a flange. Radial support is achieved by mounting the two bearings 25 mm apart. The top of the steering shaft is cut square with a hole through it, allowing torque to be applied by a secured linkage without slippage.

The end of the motor shaft is cone shaped, which increases the difficulty in manufacturing a fitting for it. The original arm was modified to transfer torque from the motor to the steering shaft. This was achieved by reducing the length and welding a fitting on to it, which could be secured onto the squared steering shaft. The linkage can be seen between the motor shaft and the bearing in Figure 3-14. An M5 screw is used to fasten the steering shaft to the linkage, securing the wheel assembly to the chassis. Also shown is the mounting for the motor, made from three 3/8" × 50 mm steel tubes with M6 × 65 mm bolts. The bolts are fixed through the chassis plate and tubes into the motor casing.

Figure 3-14: Adapter to transfer torque between the motor and steering shaft.

3.5 SHAFT ENCODER MOUNTING

To be able to obtain both distance and heading information from odometry, shaft encoders are used to read the distance travelled by each of the drive wheels. The encoder is mounted through the axle bearing support plate on the chassis. Measurements are taken from the drive shaft, using a pulley and belt system shown in Figure 3-15. The pulleys were custom made for this project as manufactured pulleys

or gears do not support the large shaft variation, from 20 mm to 6 mm diameter, with a small external diameter.

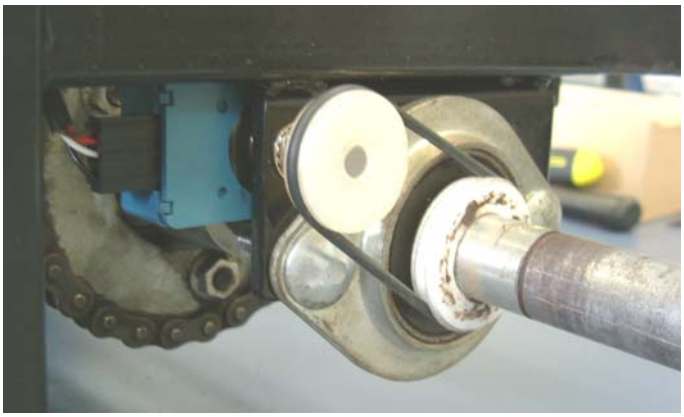


Figure 3-15: Shaft encoder mounting inside drive wheel.

The axle to shaft encoder rotation is not a 1:1 ratio. The axle pulley is 32.0 mm diameter and the shaft encoder pulley is 25.4 mm, providing a ratio of:

$$1:(32.0\text{ mm} / 25.4\text{ mm}) = 1:1.2598 \qquad \textbf{Equation 3-2}$$

Multiplying this ratio by 500 counts per revolution of the shaft encoder (refer section 2.10.4) gives the number of counts per revolution of the axle:

$$500 \times 1.2598 = 629.92 \text{ counts per (axle) revolution} \qquad \textbf{Equation 3-3}$$

As the tyre diameter is 250 mm this provides a resolution of:

$$250\text{ mm} \times \pi / 629.92 \text{ counts per revolution} = 1.247 \text{ mm per count} \qquad \textbf{Equation 3-4}$$

3.6 INFRARED OBJECT DETECTOR MOUNTING

Six infrared object detectors are used to measure the distance from the chassis to surrounding objects. They have been positioned around the chassis to maximise the area covered to prevent collision with environmental surroundings as illustrated in

Figure 3-16. As the mechatron will predominately move in a forward direction, two sensors are mounted on the front to assist with obstacle detection.

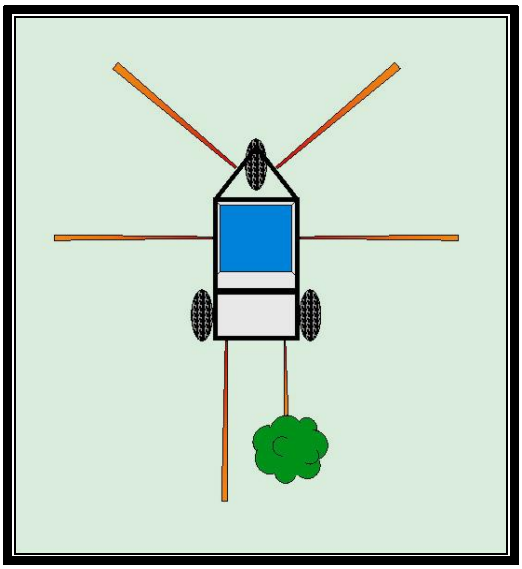


Figure 3-16: Infrared object detector configuration

The "forklift" style steering implies that the rear of the robot will swing outwards during a turn while travelling in forward direction. The two rear detectors are angled to monitor the area that the wheel will pass over during steering, and to also monitor behind the robot while reversing. This configuration however, does have a blind spot directly behind the mechatron – large objects will be detected by the angled sensors but smaller items may not be seen. In most cases the robot will have travelled in a forward direction across the area of concern before any reversing manoeuvre is performed, allowing prior knowledge of the landscape to determine if the intended path intersects with obstacles.

As two IR emitters are to be mounted in a parallel direction, the radiation intensity versus angle was measured (Figure 3-17) to determine if crosstalk would occur between the adjacent sensors. The emitters modulate the light at approximately 970 Hz. If the light from two emitters intersect, the intensity measured by the receiver will beat with a frequency $f1 - f2$, where $f1$ = modulation frequency of one emitter, and $f2$ = modulation frequency of the second emitter. This will provide incorrect measurement readings – reporting objects closer than they actually are during the beat period. From the graph shown in Figure 3-17 it can be seen that angles outside 1°

have a very low intensity. To prevent the infrared light from adjacent emitters beating, the two front emitters have each been mounted on a 1° angle away from the centre, making the intensity in any intersecting area negligible.

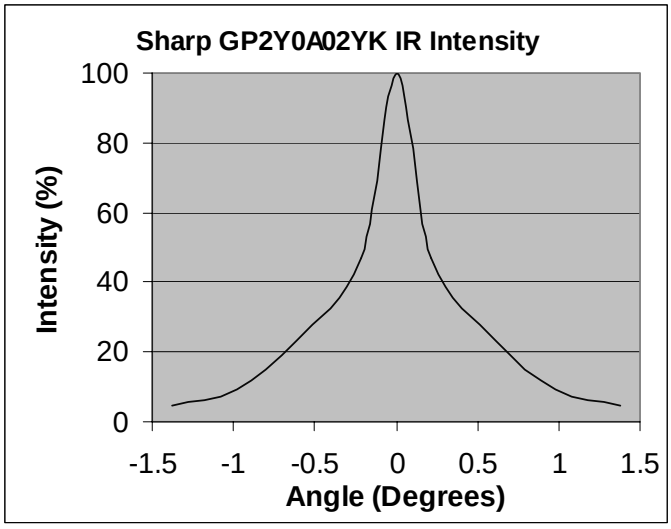


Figure 3-17: Experimentally measured IR object detector emitter intensity versus angle

3.7 ELECTRONIC COMPONENT TRAY

To support the electronic components required by this project, a 285 × 370 mm acrylic tray is mounted above the PC and batteries. Additionally, a rectangular steel frame is mounted 70 mm above the tray encasing the area, providing protection to otherwise exposed sensors and other components.

The tray is mounted 50 mm above the PC to allow access to the serial port and DAQ card. A slot has been cut into the acrylic tray to allow the wireless network card to be removed from the PC, shown in Figure 3-18.

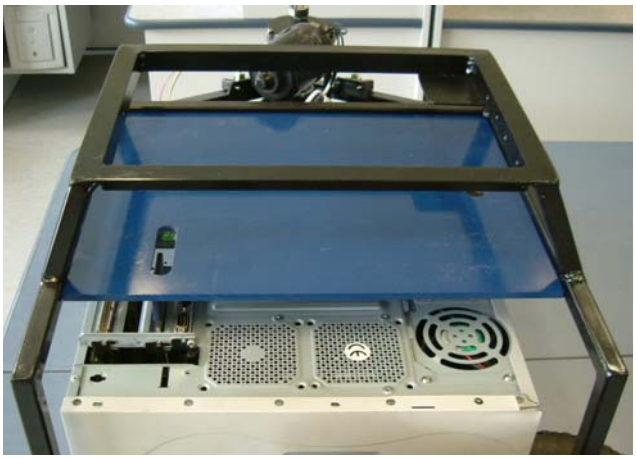


Figure 3-18: Acrylic tray to support sensors and PCB's

A 50 mm × 60 mm flat steel plate is welded 20 mm from the top of the mechatron, inside the PCB area to support the GPS antenna. This provides an unobstructed view of the sky for the antenna which uses a magnetic backing to secure it into place. The receiver can be mounted directly underneath the antenna in this configuration, reducing the area required for the complete GPS unit.

3.8 COMPLETED ASSEMBLY

The complete assembled chassis with the drive systems constructed is shown in Figure 3-19. The mechatron specifications are:

Length:	865 mm	Weight (With PC/Batteries):	58 kg
Width:	600 mm	Min Ground Clearance:	65 mm
Height:	430 mm	Wheelbase:	630 mm
Maximum Speed (with ¾ duty cycle limit):			2 m/s

Figure 3-19: Complete chassis and drive components.

The chassis, when loaded with the PC and batteries, weighs 58 kg. The rear steering wheel supports 24 kg and the driving wheels combined support 34 kg, with the location of the centre-of-gravity shown in Figure 3-20. This is within the triangle generated between the wheels, and offers good stability.

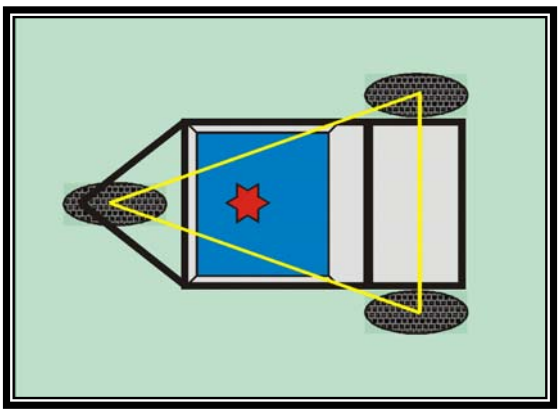


Figure 3-20: Centre of gravity of the loaded mechatron indicated by the star

4 MOTOR CONTROL

4.1 INTRODUCTION

Both of the motors require a control system, and power drive circuit. The current rating and control system requirements of each motor differ. The DC motors allow the direction to be changed by reversing the current flow, so the circuit must be capable of providing this.

A number of methods are available to control the speed of a motor such as continuously rated variable resistors, and multi-tapped windings. An electronic method is used for this application as it allows precise control from a digital controller with high efficiency. Pulse Width Modulation (PWM) provides variable output power by switching the output with a variable duty cycle. The frequency remains fixed, with the conduction time varied to provide the required amount of power to the motor.

One method of reversing the current direction through the motor (therefore reversing the motor's direction) is to mechanically switch the connections using a relay, as shown in Figure 4-1. Due to the high current of the drive motor, the switch contacts would eventually burn out if used for this application.

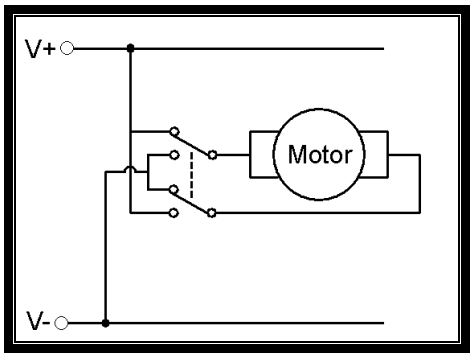


Figure 4-1: Motor direction control using DPDT switch

A second method of reversing motor direction is an H-Bridge. Four switches provide control of the current path through the motor, shown in Figure 4-2. By enabling

switches 1 and 4 the current will flow through the motor from left to right, or enabling 2 and 3 from right to left.

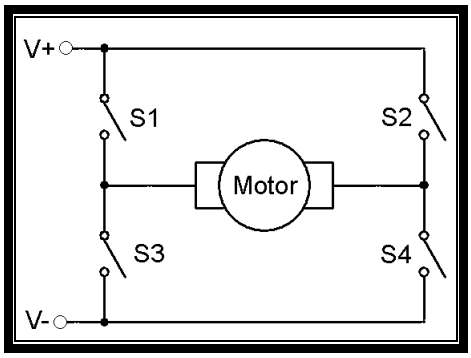


Figure 4-2: H-Bridge motor drive configuration

Braking can also be incorporated into the control system by enabling switches 3 and 4. When the motor is rotating with no current being applied, it acts as a generator. By shorting the motor terminals together, the energy generated is dissipated as heat within the motor itself. This prevents the motor from free-wheeling – providing a braking force proportional to the speed of the motor.

Care must be taken to never enable both the high and low switches on one side of the H-Bridge. If this occurs a large current will flow through the two switches, often destroying them.

4.2 STEERING MOTOR CONTROL

The modified wiper motor, discussed in section 2.6.2, requires both a control system, and a power drive system. The control and driver circuit built (Appendix A.1) uses UC3637 and L298 IC's. The UC3637 is a proportional PWM control IC. The desired position is input to the IC using an analogue signal between 0-10 V, matching the Lab-PC+ analogue output range. This voltage is compared to the voltage from the potentiometer mounted on the motor drive shaft. The error is amplified, and converted to two 30 kHz PWM signals used to control the motor's speed and direction. The signals use 0-24 V logic as this is the supply voltage of the IC.

The L298 is an H-Bridge motor driver IC. The drive signals input use 5 V logic, so the output from the UC3637 is converted from 24 V to 4.9 V using zener diodes. The L298 is capable of driving two individual motors, but for this application the two channels are paralleled for higher current output. The maximum continuous current is rated at 4 A, so current limiting has been included to complement this, which also provides smooth movement of the motor as the acceleration is limited.

The proportional constant had to be high to force the motor to move when small changes were applied. With a low constant, the error was small and the drive signal was limited, for example to 50% of full power, which was not enough to initiate motion from rest. In the PCB design this corresponded to the error amplifier gain set to ten or less. The low constant produced movement towards the target position when a large change was applied (hence large error), as the initial power was enough for motion to commence, with the power decreasing as the target was reached.

By increasing the error amplifier gain (proportional constant) from 10 to 33, this problem is resolved, but another is generated. The system now continues to apply maximum power even when it is moving and is close to the target position. The motor overshoots and reverses direction. This overshoot repeats again and again, slowing decaying. The result is that for small changes of the target position the movement required is obtained, but with large changes the motor must "hunt" for the position.

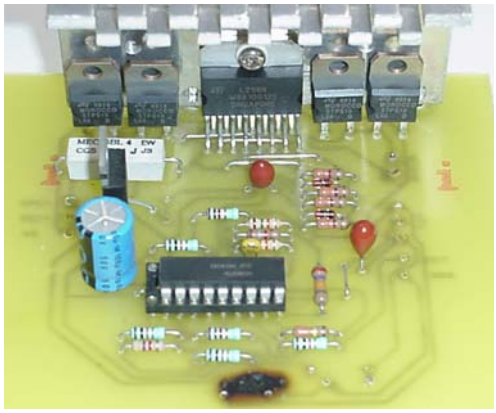


Figure 4-3: Steering motor controller and driver

4.3 DRIVE MOTOR CONTROL

The main drive motor (refer section 2.5) requires a system for control and an output circuit capable of 400 W. The control system would preferably generate a PWM signal to vary the power applied, and use an H-Bridge to provide direction control.

4.3.1 PWM Generation

A number of different methods are available to generate this signal:

1) Use an IC such as shown in section 4.2 to convert an analogue output from the LabPC+ into a PWM signal.
2) Toggle the digital input/output (I/O) lines on the LabPC+ to generate a PWM signal.
3) Toggle the analogue output lines on the LabPC+ between high and low to generate a PWM signal.
4) Use a microcontroller to generate the PWM signal.

A single-ended analogue output signal to control the motor is susceptible to noise from the accompanying drive circuit, and can be difficult to provide precise increments of the required duty cycle. This is most noticeable when the output signal is very small, as any fluctuations become significant.

Toggling the digital I/O outputs generates two problems; i) all of the lines for that port (8 bits) must be set, rather than controlling individual lines, ii) the accuracy of the PC's timer would only allow a low frequency PWM to be generated. Using one bit of the 8 bit port to control a single motor is very inefficient on the limited resources available. The resolution of the PC's timer is very low for this type of application; in this case using the Celeron 800 MHz the timer has a resolution of approximately 10 ms. To maintain an adequate resolution duty cycle, the frequency is then limited below a hundred Hertz. As the inductance of the motor decreases as the drive frequency is increased (see Figure 4-4), over this range, the inductance of the motor is dominant over the resistance. This produces a larger strain on the power output drive

circuit, which must be designed to withstand and control the back emf generated by the motor.

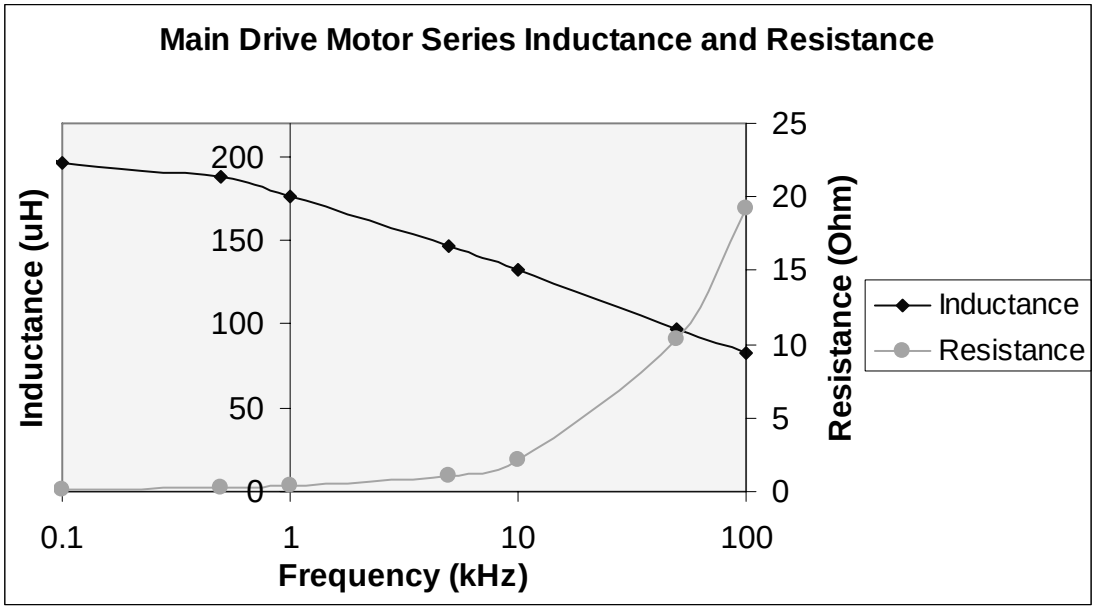


Figure 4-4: Motor inductance vs frequency

Inductance and resistance measured using FLUKE PM6306 programmable automatic RCL meter, with a 2.00 V AC test signal.

The analogue output lines on the LabPC+ can be toggled between high and low to produce a digital output PWM signal, in much the same way the digital lines can be toggled. For the LabPC+ however, a pattern can be stored into an array which is then clocked out by a counter. This allows increased precision, as the output is determined by a counter rather than the software timer, which in turn allows the frequency to be increased. However, the operating frequency is still limited to a few hundred Hertz.

A microcontroller allows a significantly higher frequency than available from the LabPC+, while providing 8 bit resolution. By using a microcontroller a number of additional features can also be incorporated, such as stopping the motors if the PC software fails, rather than continuing with the last instruction given (which the other methods suffer from). This was the technique chosen for this project.

4.3.2 Microcontroller design

The requirements of the microcontroller to provide motor control are:

- Provide an interface to the PC for receiving data
- Provide a high frequency PWM signal output
- Keep the design simple to reduce system complexity and component count.

The microcontroller chosen for this project was the Phillips P89C51RC2. It offers 32K of onboard flash memory, 256 bytes of RAM, and in-system programming. This allows the circuit design to be very simple, reducing component count by using the onboard resources rather than external ROM/RAM IC's etc. An 8051 microcontroller course is taught at the University of Waikato, with only minor variations required to program the P89C51RC2.

As well as PWM generation, a number of additional features are incorporated into the microprocessor due to the processing capabilities available. Provision has been made to control two motors, and input three incremental shaft encoders. Acceleration and direction constraints limit the stress on the power output hardware, and a safety shutdown disables the motors if the PC stops responding.

The PC to microprocessor communication interface chosen was an 8 bit parallel bi-directional bus. This offers a speed advantage over a serial protocol. A serial port is required by the GPS receiver (refer section 5.5.1), and most new PC's only have one serial port available. To utilise another serial port would increase the cost and complexity of the system, requiring a USB to serial converter or similar device. As the microcontroller does not have an onboard analogue to digital converter (ADC), using an analogue control signal would again increase the complexity of the microcontroller circuit.

The parallel communication is designed to operate with the LabPC+ digital I/O lines, configured in mode 2 (bi-directional transfer). Eight data lines are used, with four handshaking lines controlling the flow direction and acknowledgement of the data. A fifth handshaking line is available to generate an interrupt for the microprocessor, but it is not required as the handshaking line will be constantly monitored. Programming

the interface on the PC is straightforward as National Instruments provide the necessary functions to operate the DAQ card and the handshaking lines used. The instruction set to control the microprocessor is shown in Table 4-1. The microprocessor is capable of providing requests and responses to control two motors and three encoders within a control loop time of 10 ms. This is the finest resolution offered by the PC timer.

Data Sent	Data Received if successful	Comments
00000000	11111111	Emergency Stop. Stop both motors immediately.
0000001D 1SSSSSSS	Inverted of Sent Inverted of Sent	Motor 0. Direction defined by D. Speed of motor 0, where SSSSSSS 127 = full speed
0000010D 1SSSSSSS	Inverted of Sent Inverted of Sent	Motor 1. Same as Motor 0 defined above.
00001000	0HHHHHHH 1LLLLLLL	Encoder 0. 14 bit value of encoder count, where HHHHHHH = High 7 bits, LLLLLLL = low 7 bits.
00001001	11110110	Encoder 0 reset to zero.
00010000	0HHHHHHH 1LLLLLLL	Encoder 1. Same as Encoder 0 above.
00010001	11101110	Encoder 1 reset to zero.
00011000	0HHHHHHH 1LLLLLLL	Encoder 2. Same as Encoder 0 above.
00011001	11100110	Encoder 2 reset to zero.
01xxxxxx		Reserved for future use. Adjustable Acceleration limit, Speed limit, Timeout time etc.

Table 4-1: Microprocessor control instruction set.

The commands have been structured (where possible) to remove any ambiguity between instructions and data, therefore reducing synchronisation problems when sending a number of commands/data in succession. This can be seen when comparing the MSB in the table above; an instruction is zero, and a data value is one. Similarly, the encoder data uses the MSB as zero for first byte, then one for the following byte.

When a motor command is received, the desired direction is compared with the present direction. If these match, the response is sent to allow the PC to send the new

PWM value. To change direction, a speed of zero must first be sent to allow the motor to stop. Once sufficient time has allowed the motor to decelerate, the direction can then be changed using the commands in Table 4-1. If this is not done, the microcontroller will reply with `0x7F` rather than the expected negated command, and activate an error LED on the microcontroller PCB. If the user repeats the (incorrect) command, the motor will decelerate to a stop. Hence if a change of direction is consistently sent to the microcontroller, the motor will decelerate itself allowing the change of direction, and then accelerate to the desired speed. The user can still force the direction change – but is restricted by the acceleration limit. This has been done to protect the power output circuit which drives the motor from spurious direction changes, which can cause over-voltage conditions, damaging the MOSFETs.

The microcontroller provides two output signals to each drive circuit; a single data line which provides the required direction, and a positive duty cycle PWM (with an adjustable frequency up to a maximum of 31 kHz).

The microcontroller offers two digital lines which can be used to provide feedback from each motor driver circuit. These are intended for monitoring temperature and current of the output power drive stage. By providing this information back to the microprocessor, the user can define the action to be taken. An example may be to reduce the duty cycle of the PWM during an over-current condition (soft stop), rather than just shutting the output off as most hardware based current limits would function.

Both acceleration and duty cycle limits are available in the microprocessor code. The acceleration limit is useful for providing a smooth transfer of power, reducing the stress on the motor. During the initial development stage of the control system, an unstable system with large oscillations may be capable of damaging the output driver circuit. Once a basic control system is developed, the acceleration limit can be reduced to provide output protection with little restriction on the control system operation. When given a PWM change greater than the set limit, the microcontroller will respond with a successful confirmation. The PWM will then ramp to the desired target rather than stepping the output. The PWM duty cycle can also be restricted during development to limit the mechatron's speed. The microcontroller will reply with a successful confirmation that the target PWM has been received, but will only

ramp up to the limit set. The error LED will activate to show the user that the limit has been exceeded. This helps prevent the user from losing control of the mechatron during development of the control systems, limiting the mechatron's maximum speed.

The shaft encoders used (see section 2.10.4) provide a quadrature output. The output signals are two digital waveforms which are 90° out of phase. By monitoring both output waveforms, the user can gain both direction and angular displacement information. The microcontroller uses a timer positive edge capture interrupt, which captures the current value of timer 2 and generates an interrupt request. The capture lines are used because the microcontroller does not have enough standard external interrupts available. The value captured is irrelevant and therefore ignored. When the interrupt occurs, the state of the second line is queried. If it is high, the encoder is moving in a clockwise direction, or if low, in an anti-clockwise direction, as shown in Figure 4-5. A counter is then either incremented or decremented accordingly.

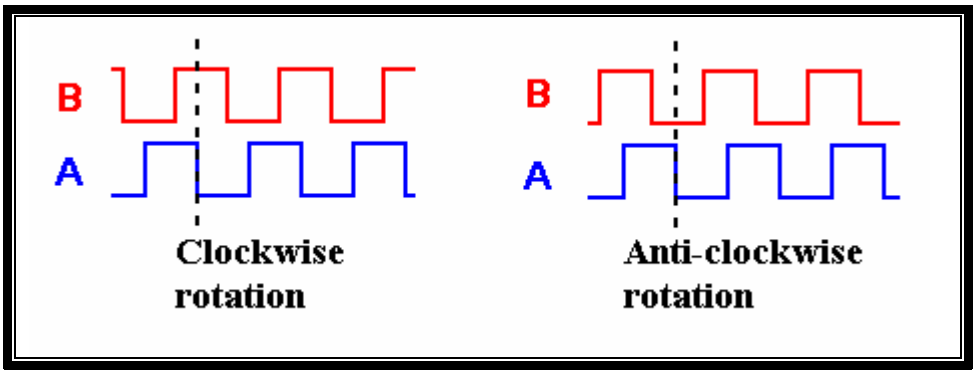


Figure 4-5: Shaft encoder quadrature output

A timeout feature is incorporated with an adjustable delay. This provides protection against PC software/hardware failure, shutting the motors off when no instructions are received with in the set time limit. Cases when this is required are:

- Control software lockup (although an infinite loop may still send instructions).
- Operating system lockup, where the computer is non-responsive.
- Quitting control software without first sending a motor stop instruction.
- PC power supply failure due to flat batteries, where the PC power supply shuts off, or fluctuates causing a reboot.

The timeout shutdown does not check the validity of the instructions, only that they are being received. The timeout is set in milliseconds (default of 500 ms) and can be varied up to one minute. The error LED is activated to show a problem exists.

The microcontroller uses a serial interface to allow In System Programming (ISP). This allows the code to be updated without removing the microcontroller from the mechatron to use a commercial programmer. To activate the ISP the PC control software must be closed, allowing the parallel data lines to default to a logic high state. This is required because a manufacturer error necessitates bits P2.6 and P2.7 to be in a high state to activate the ISP of this microcontroller revision. A standard three wire crossover serial cable connects the DB9 connector and PC. The switch is toggled to download (DL) and the microcontroller reset using a momentary push button switch. Using Phillip's WINISP v2.29, the flash memory can be erased and reprogrammed, using the settings shown in Table 4-2.

Attribute	Value
Chip	P89C51RC2
Port	Com1
Osc (MHz)	33
Vector	FC
Status	00

Table 4-2: Winisp v2.29 settings

The Vector value is a pointer to the ISP firmware in the microcontroller memory. It allows the flash memory to be erased and new code stored, interfaced through a serial communication link. If this vector is incorrectly set, the user cannot connect to the microprocessor using the serial port, and must use a commercial programmer to change the code stored. The Status value determines whether the microprocessor should run the boot loader code to allow serial programming, or the code which has already been stored. A value of 0x00 executes the stored code, and any other value executes the boot loader. The boot loader code can be forced to activate by holding PSEN low, P2.6, P2.7, EA and ALE high when booting (regardless of the status

value). PSEN is pulled low by the download/run switch on the microcontroller PCB. The parallel data lines are connected to P2.6 and P2.7, and are high when inactive. EA is connected to the 5 V supply; and ALE is unconnected which defaults to a high state. When PSEN is released and reset, the microcontroller defaults back to the downloaded code.

4.3.3 Microcontroller problems and limitations

Due to the inability of the microcontroller to source large amounts of output current when the digital lines are in a high state, some data being read by the LabPC+ was being corrupted and read as low. The LabPC+ should be in a high impedance state, but during a read instruction it sinks some current pulling the voltage down, as shown in Figure 4-6. By including a bi-directional buffer, the data lines are capable of sourcing the current required. The buffer direction control operates from one of the handshaking lines, with an insignificant delay between direction changes.

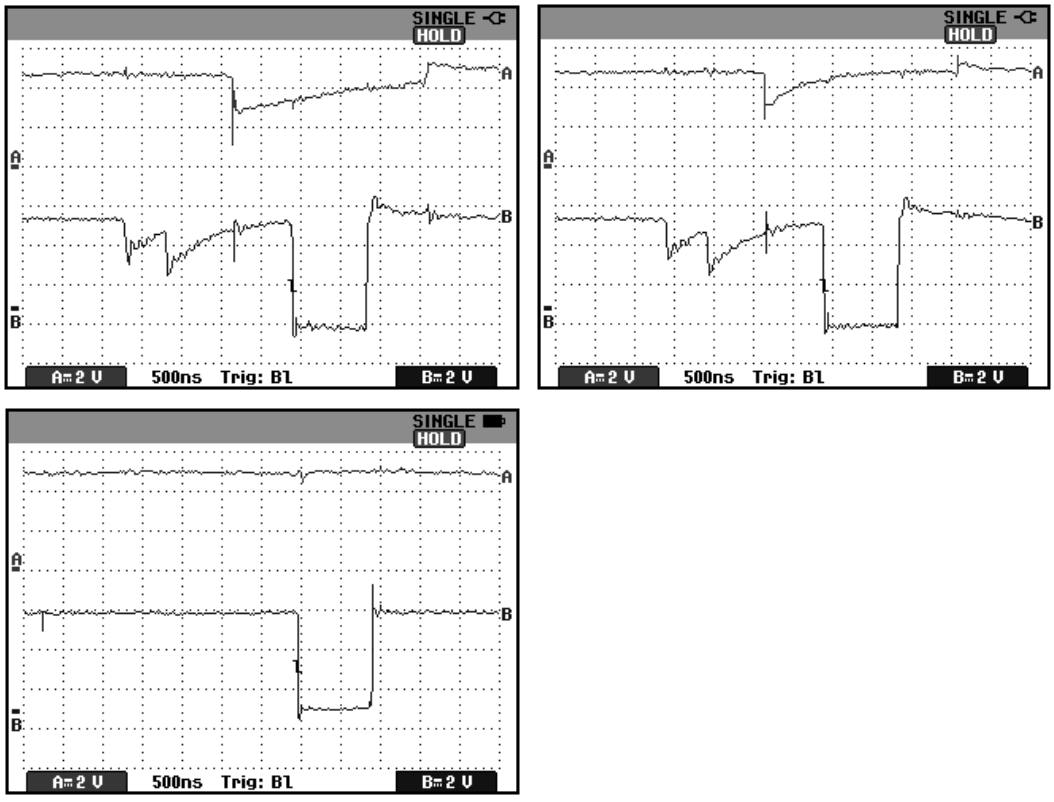


Figure 4-6: Digital line data corruption between the micro and LabPC+. Channel A (top line) shows logic high data, and the channel B (bottom line) specifies when it is available to read (logic low). TOP LEFT: Micro and LabPC+ directly connected together. TOP RIGHT: Pull up resistors added to provide logic high drive. BOTTOM LEFT: A bi-directional buffer providing drive for logic signals.

The buffer also minimised problems due to noise from surrounding electronics, especially the motors which corrupted the digital signals from/to the LabPC+.

The LabPC+ has supporting software functions written for different programming languages included in Nidaq v6.93 from National Instruments. The software offers routines, and examples of their use, for all of the capabilities of the card, ranging from digital and analogue I/O to counter/timer features. The digital I/O examples do not include any bi-directional models, and therefore the code was structured from the uni-directional models and the timing diagrams from the user manual [7]. The function **`DIG_Prt_Status`**, however, does not work correctly when operating in bi-directional mode. The function explanation from the user manual is given in Figure 4-7.

DIG_Prt_Status

status = DIG_Prt_Status (deviceNumber, port, handshakeStatus)

Purpose
Returns a status word indicating the handshake status of the specified port.

Parameters

Direction	Name	Type	Description
Input	deviceNumber	i16	assigned by Measurement & Automation Explorer
	port	i16	digital I/O port number of the port you want the status of
Output	handshakeStatus	i16*	handshake status

Parameter Discussion
port is the digital I/O port number.
Range: 0 or 1 for the DIO-24 and Lab and 1200 Series devices.
2 or 3 for the AT-MIO-16DE-10 and 6025E devices.
0, 1, 3, 4, 6, 7, 9, and 10 for the DIO-96.

handshakeStatus returns the handshake status of the port.
0: A port is not available for reading from an input port or writing to an output port.
1: A unidirectional port is available for reading from an input port or writing to an output port.
2: A bidirectional port is ready for reading.
3: A bidirectional port is ready for writing.
4: A bidirectional port is ready for reading and writing.

Note C programmers: handshakeStatus is a pass-by-address parameter.

Using This Function
DIG_Prt_Status reads the handshake status of the specified port and returns the port status in handshakeStatus. DIG_Prt_Status, along with DIG_Out_Port and DIG_In_Port, facilitates handshaking of digital data between systems. If the specified port is configured as an input port, DIG_Prt_Status indicates when to call DIG_In_Port to fetch the data an external device has latched in. If the specified port is configured as an output port, DIG_Prt_Status indicates when to call DIG_Out_Port to write the next piece of data to the external device. If the specified port is not configured for handshaking, NI-DAQ returns an error code and handshakeStatus = 0.

Figure 4-7: DIG_Prt_Status, from the NI-DAQ function reference help, version 6.9

The function should return a **`handshakeStatus`** value of 3 in an idle condition as data can be written, but no data has been received to read. In practise however, the function only returns values of either 2 or 4 (2 if data has just been written out, but not yet received by the microprocessor, otherwise 4). The implication of this is that there is no way to determine whether new data has been received since the last read instruction other than performing another read instruction and comparing the values received. This in turn restricts the data being sent by the microprocessor to non-repetitive values. As an example of this problem, if the 16 bit value of the encoder counter was zero, the data sent (`0x00` followed by `0x00`) would not be distinguished by the PC as two values. To overcome this problem, rather than re-writing the **`DIG_Prt_Status`** function, the command protocol and data has been reduced to either 7 or 14 bits with the MSB toggling between each byte sent by the microprocessor. This allows the PC to poll for new data, rather than using the **`DIG_Prt_Status`** function. Note: the function performs as expected when operating in uni-directional mode; and this is the last release of NiDAQ (v6.9.3f3) which supports the LabPC+.

The output lines of the microcontroller default to a high state on power up or reset. The first lines of code can set the output lines low, however there is a delay before this code is processed. This causes a problem in that the positive PWM output line defaults to a high state during these conditions. Therefore, if the motors have power connected, they will be activated at full duty cycle. Normally on power-up, the motors are off as the PC is being turned on first. The problem is most obvious when the microcontroller is being reset, or during download mode since the ISP will hold all outputs high. To overcome this problem each PWM signal has been inverted after leaving the microcontroller, shown in Figure 4-8.

Electromagnetic noise from the motors put the microcontroller into an unpredictable state. The code was tested over a 24hr period with no continual errors (occasional communication errors present). With an unshielded motor nearby the microcontroller can stop responding within minutes. To reduce this problem, the motor casing was grounded to the chassis and the negative battery rail. Although the wiper motor is from an automotive application where the chassis is normally the negative connection, the motor brushes are isolated from the casing. By grounding the metal motor case

the noise was greatly reduced. A 0.1µF capacitor was also soldered over the brushes of the motor (inside the motor case) to help suppress the noise due to the brushes sparking on the commutator, refer section 3.4.1. A further solution, if required, would be to fit the microcontroller inside a conductive case to provide further shielding.

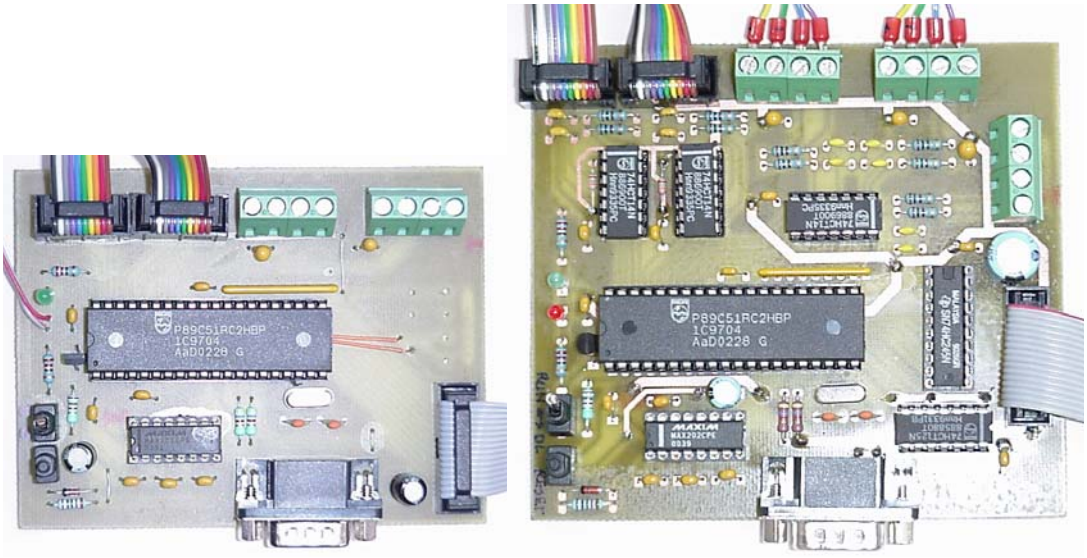


Figure 4-8: Microcontroller PCB. LEFT: Initial design; RIGHT: Improved design with inverters, buffers and low pass filters.

4.4 DRIVE MOTOR H-BRIDGE DESIGN

To power the main drive motor shown in section 2.5, a PWM, H-Bridge circuit was designed [8]. The circuit requirements for this system are:

- H-Bridge, PWM drive up to 31 kHz
- 5 V logic, direction and PWM input signals
- 24 V DC power input and drive output
- 400 W output (approximately 15 A continuous) rating
- Temperature and current sensing as required

The initial design was based on the Devantech Ltd schematic for the MD03 [9] (with authorisation from Devantech). The MD03 is a 5-50 V, 20 A H-bridge motor driver, which uses a PIC16F872 for the user interface and control. The first prototype board design was similar to the Devantech motor driver schematic, shown in Appendix A2, with the control microprocessor changed to the P89C51RC2 (see section 4.3.2).

The circuit operates as two independent halves, each with its own half bridge driver IC. The direction and PWM logic signals are separated into two waveforms using NOR gates, each waveform controlling one half bridge IC (refer Figure 4-9). Depending on the logic level of the direction signal, one side of the bridge remains logic low while the other receives the PWM signal (0-5 V logic).

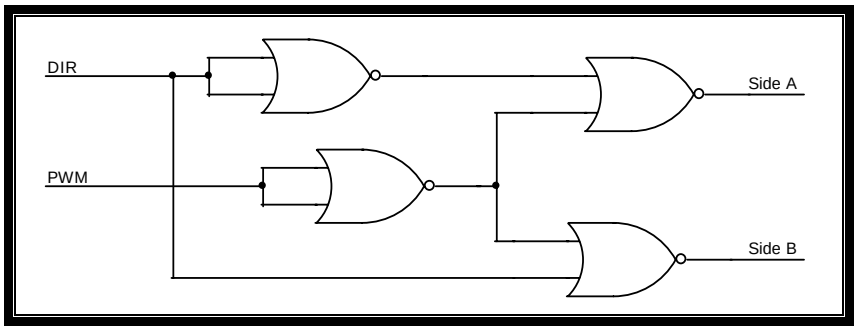


Figure 4-9: Input waveform separation

Each half bridge is driven using ST Semiconductor's L6384. The required supply voltage to operate the L6384 is between 12.5 V and 14.6 V, and must not be a low impedance source. The voltage source is generated from the 5 V logic rail using a three stage charge pump shown in Figure 4-10. This configuration allows the output voltage being driven by the circuit to be independent of the supply voltage to the driver IC's, and therefore could vary between 5-580 V DC by simply choosing appropriately rated components.

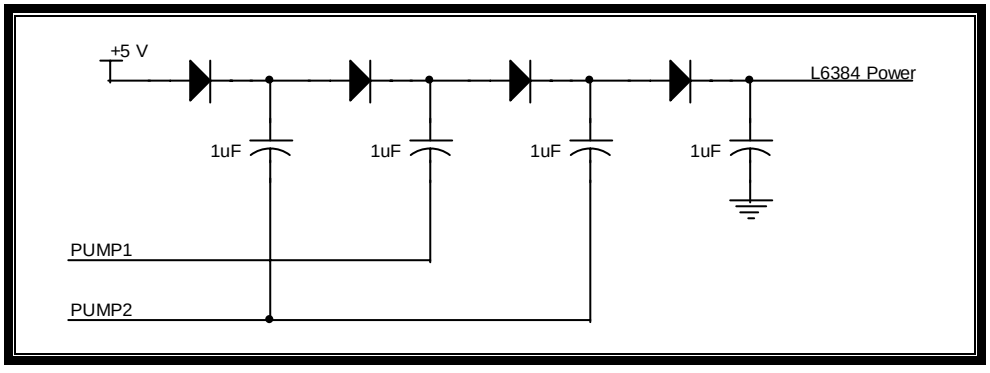


Figure 4-10: Charge pump used to provide input voltage for L6384 IC's

When driving MOSFETs, the Vgs (gate-source voltage) must be >10 V to ensure the MOSFET is turned on sufficiently, not in its linear region of operation. In an H-bridge configuration, two of the MOSFETs are connected to the high side of the load.

To drive these MOSFETs the gate voltage must be larger than the supply voltage; in this case the drive voltage must be > 34 V while using 24 V motors.

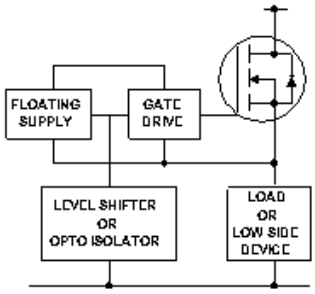
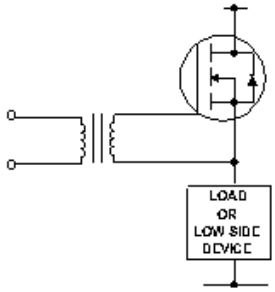
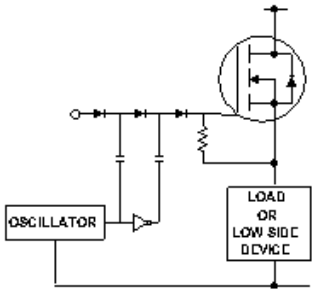
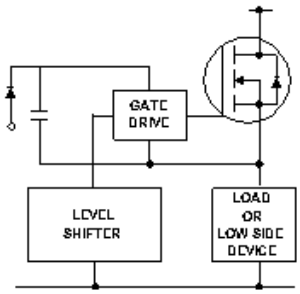
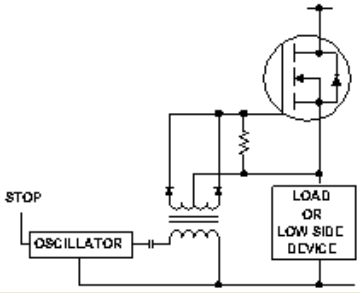
METHOD	BASIC CIRCUIT	KEY FEATURES
FLOATING GATE DRIVE SUPPLY		Full gate control for indefinite periods of time. Cost impact of isolated supply is significant (one required for each high side MOSFET). Level shifting a ground referenced signal can be tricky: Level shifter must sustain full voltage, switch fast with minimal propagation delays and low power consumption. Opto isolators tend to be relatively expensive, limited in bandwidth and noise sensitive.
PULSE TRANSFORMER		Simple and cost effective but limited in many respects. Operation over wide duty cycles requires complex techniques. Transformer size increases significantly as frequency decreases. Significant parasitics create less than ideal operation with fast switching waveforms.
CHARGE PUMP		Can be used to generate an "over-rail" voltage controlled by a level shifter or to "pump" the gate when MOSFET is turned on. In the first case the problems of a level shifter have to be tackled. In the second case turn on times tend to be too long for switching applications. In either case, gate can be kept on for an indefinite period of time. Inefficiencies in the voltage multiplication circuit may require more than two stages of pumping.
BOOTSTRAP		Simple and inexpensive with some of the limitations of the pulse transformer: duty cycle and on-time are both constrained by the need to refresh the bootstrap capacitor. If the capacitor is charged from a high voltage rail, power dissipation can be significant. Requires level shifter, with its associated difficulties.
CARRIER DRIVE		Gives full gate control for an indefinite period of time but is somewhat limited in switching performance. This can be improved with added complexity.

Table 4-3: Methods of providing high-side gate drive [10]

A number of different method are available to generate the required Vgs, a summary of common methods is shown in Table 4-3. The L6384 uses a bootstrap configuration.

Current monitoring is performed using a sense resistor, refer Figure 4-11 . A low power, rail-to-rail op-amp is connected to the resistor using differential inputs to accurately measure small voltages. The op-amp integrates and amplifies the voltage from the sense resistor to provide an average of the current use (rather than fluctuations from each pulse). A second op-amp is configured as a comparator, with a resistor divider setting a threshold for the current limit. The output is fed back to the microprocessor as a digital 0-5 V signal, as the op-amp operates from a 5 V supply.

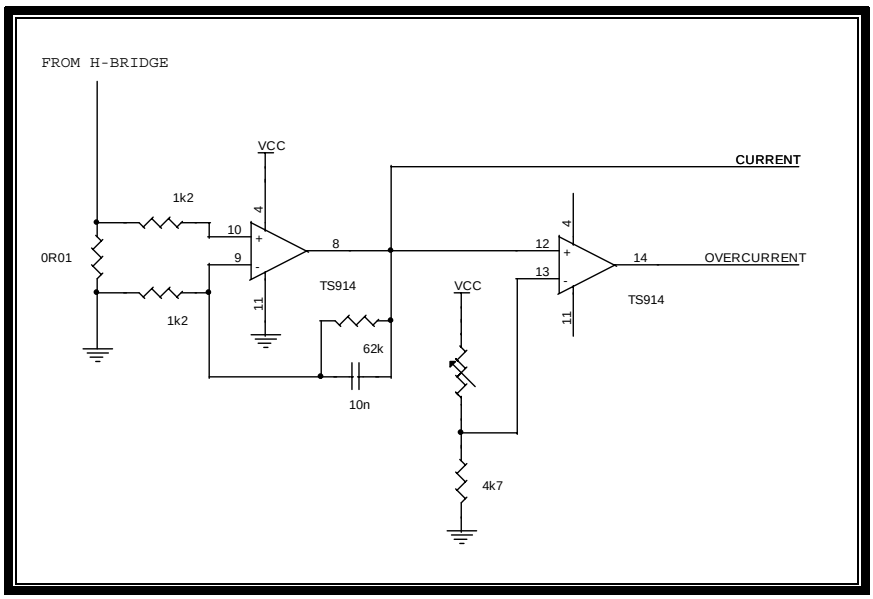


Figure 4-11: Motor current sense circuit

The temperature sensing is performed in much the same way as the current sensing, but uses a diode as the sense element. The sense resistor transfers heat to the diode (which is positioned underneath the current sense resistor). This gives a representation of the heat of the MOSFETs as the resistor will heat up with continual high currents. Again op-amps amplify the signal then convert it to a digital representation against a set threshold.

This design worked, but had two major limitations:

1) The power consumption by the L6384s is too high for the charge pump generating their supply voltage (with a 31 kHz PWM signal);
2) The duty cycle is limited by the need to refresh the bootstrap capacitor.

The charge pump providing the supply voltage to operate the L6384s can only provide a limited amount of current. The required current depends on the switching frequency of the PWM signal, as most power is used to drive the MOSFET gates. With the 31 kHz PWM frequency, the current consumption was higher than that supplied by the charge pump. The result shown in Figure 4-12 is the high side MOSFET gate signal. The supply voltage decreases until an under-voltage (UV) lock out activates, at which point the IC shuts off. With no current consumption, the supply voltage rapidly rises above the UV lockout threshold, so the IC activates again, repeating the pattern.

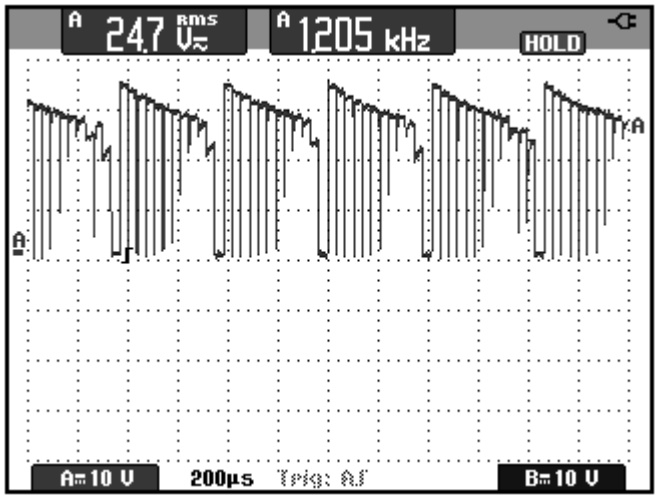


Figure 4-12: High side Vgs with power supply problem

To overcome this problem, a new power rail was generated from the 24 V DC supply rather than from the 5 V logic supply. A linear regulator reduces the voltage to 15 V, which is then passed through a red LED to further reduce it to 13.2 V (creating the supply voltage in the 12.5 – 14.6 V range required). The LED also provides a status indicator for the board, dim if power is on, and bright if the circuit is being driven. This change restricts the circuit operation to output power voltage in the range 18-35 V DC.

The second problem occurs due to the L6384 drive method using a bootstrap capacitor. During logic low input, the capacitor is charged to Vcc (~13 V). When the logic input is high, the capacitor floats up on the output voltage to provide the gate voltage for the MOSFET. If the capacitor does not receive sufficient time to charge during the low input cycle, the gate voltage decreases as shown in Figure 4-13. To maintain Vgs at 10 V, a maximum duty cycle of 80% can be used at this frequency. Note that at the lower voltages the MOSFET will turn on, but will be in a linear region of operation. This limits the current flowing through the transistor, and produces a large amount of heat. If the circuit is allowed to operate like this for more than a few seconds, the transistor will overheat and damage itself.

By reducing the operating frequency of the PWM signal, the duty cycle range can be made wider. For example, a 0-80% PWM range at 28 kHz is equivalent to 0-95% range at 7 kHz. This will *always* have an upper limit, and component destruction is risked if it is exceeded.

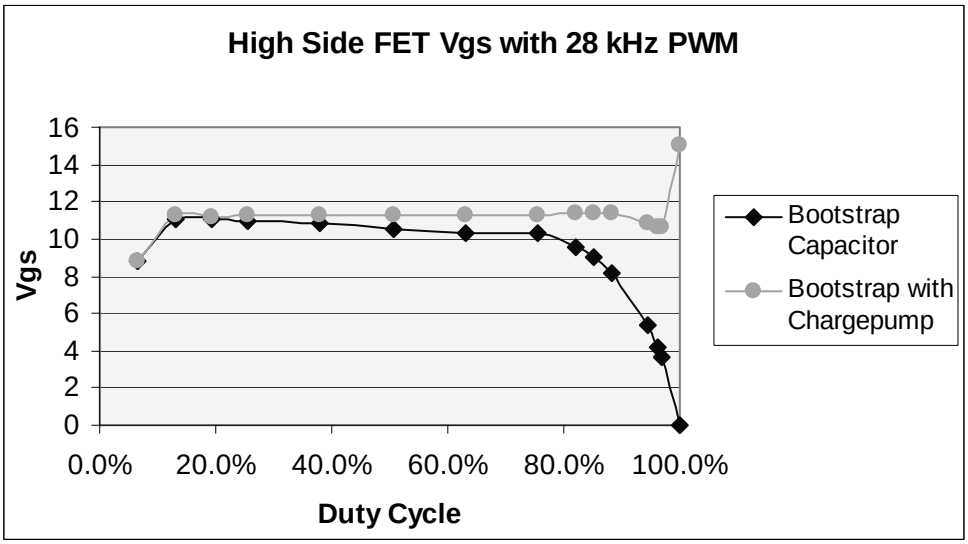


Figure 4-13: Vgs comparison using bootstrap capacitor and additional charge pump

A solution for this problem is to combine the bootstrap method with the charge pump method shown in Table 4-3. By combining these techniques together, the capacitor charge can be maintained over the entire operating range – including 100% duty cycle, but at the expense of component count and PCB size. The schematic diagram in Figure 4-14 shows the implemented charge pump for each side of the bridge.

Operating from a 555 timer IC, the charge pump frequency has been set to 70 kHz to ensure the capacitor is charged multiples times on every cycle of the PWM signal. The charge pump operates from a 15 V source provided by a zener diode, floating under the output voltage from the half bridge. Only one charge pump operates at a time using this configuration as the other half bridge output is at 0 V.

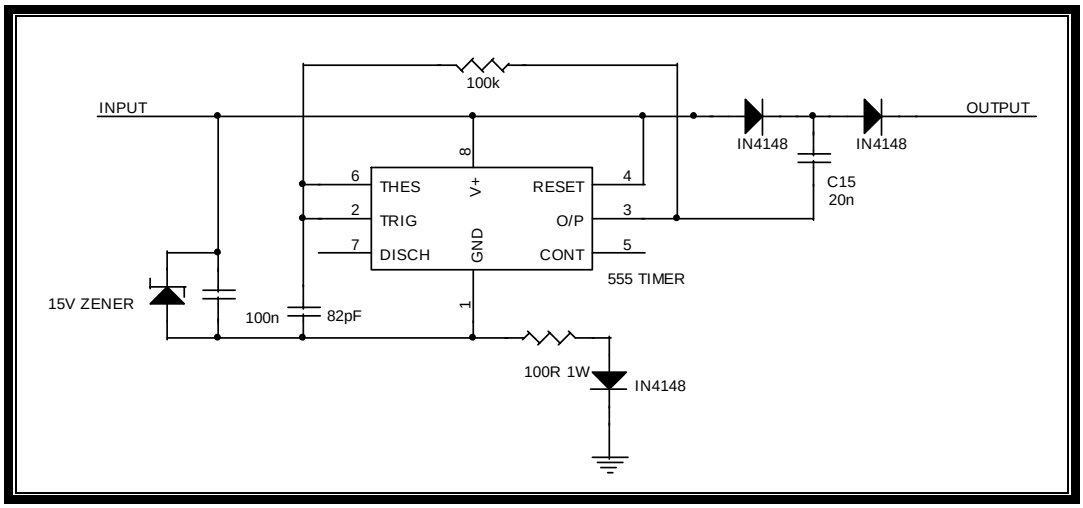


Figure 4-14: Charge pump used to maintain bootstrap capacitor charge

The effect of the added charge pump is shown in Figure 4-13, maintaining the gate voltage required to operate the MOSFETs safely. Note that the diode to ground allows the oscillator to continue running for short periods after the input goes low (operating from charge stored in the 100 nF capacitor). Therefore charge is continuously pumped into the bootstrap capacitor during the time the half bridge is active. This differs from using a charge pump to drive the gate of the MOSFET (refer Table 4-3) where the pump is only active during the high stage of the input cycle.

Due to the inductance of the motor being driven, large back emf voltages can be generated. These voltages can exceed the maximum operating limits of the MOSFET, breaking down the insulating layer to the gate. Insulated Gate Bipolar Transistors (IGBTs) were investigated to replace the MOSFETs as they offer increased durability, especially with high operating temperatures and high current conditions. MOSFETs have a positive temperature coefficient, as the junction temperature increases so does the drain-source resistance. The increased resistance then produces more heat (Power = Current^2 × Resistance), which in turn further increases the resistance. The

MOSFET quickly exceeds the maximum operating temperature at which point it is either destroyed or damaged (the soft solder bonding the silicon chip to the substrate melts and bubbles out through the top of the transistor, reducing the transistor's ability to conduct the heat away from the junction). IGBTs have a negative temperature coefficient, so as the temperature is increased the collector-emitter voltage decreases, producing *less* heat.

The IGBTs available are made for high power applications, and are rated between 600 - 1200 V, with a $Vce_{(sat)}$ of 1.7 - 4 V. The $Vce_{(sat)}$ is the voltage across the transistor output when it is turned on (saturated). This voltage drop is much higher than generated when using MOSFETs over the required current range (MOSFETs have an on resistance rather than voltage) and therefore increased cooling must be added to the circuit to remove the extra heat generated using IGBTs. A comparison is shown in Table 4-4 for an available IGBT and MOSFET, showing their efficiency over the current range required, assuming a constant $Vce_{(sat)}$ for the IGBT and a constant $Rds_{(on)}$ for the MOSFET (no variation with current, temperature).

Transistor	Output Current	IGBT IRG4BC40W	MOSFET STP60NE06-16
Conduction loss (W):	@ 5 A	10.25	0.33
	@ 10 A	20.50	1.30
	@15 A	30.75	2.93
Switching loss @ 20 kHz (W):	@ 5 A	6.80	0.42
	@ 10 A		0.84
	@15 A		1.26

Table 4-4: Comparison of available IGBT and MOSFET

STP60NE06-16: $R_{ds(on)} = 0.013\ \Omega$, $T_{sw} = 175$ ns

IRG4BC40W: $V_{ce(sat)} = 2.05$ V, $E_{ts} = 0.34$ mJ

Equations 4-1 and 4-2 show the formulae used to estimate the power losses [11]. For small current applications, MOSFETs are preferred as R_{ds} is normally very small, so the power loss during conduction is also small. In high voltage (>1000 V), high current applications, IGBTs would be favoured over MOSFETs. As the normal motor

operating current is approximately 4-8 A, at 24 V, MOSFETs are used in the motor driver design for their increased efficiency.

$$P_{con} = I^2 \times R_{ds(on)}$$

$$P_{sw} = I_d \times V_{ds} \times t_{sw} \times f_{sw} \qquad \textbf{Equation 4-1}$$

where: t_{sw} = total switching time ($t_r + t_f$ from datasheet)

$$P_{con} = V_{ce(on)} \times I_c$$

$$P_{sw} = E_{ts} \times f_{sw} \qquad \textbf{Equation 4-2}$$

where: E_{ts} = total switching losses (from datasheet)

To avoid damaging the MOSFETs three components have been included:

1) Free wheeling diodes
2) R-C Snubber
3) Transient voltage suppression diodes

During the dead time between switching the high and low MOSFETs on one side of the H-bridge, no current path is available from the motor to either ground or Vcc. During this time the motor inductance keeps current flowing, which must be then conducted through the reverse diode of the MOSFET as it is the only path available. To reduce the reverse current handled by each MOSFET diode, Schottky diodes have been added across each transistor. These are known as free-wheeling diodes, which provide a method of transferring energy back into the batteries when the motor is not being driven.

A resistive – capacitive (RC) snubber network has been added across each MOSFET to provide damping and also to restrict the rate-of-rise of the output of the transistor. The primary use for the snubber is to reduce parasitic ringing caused by the system inductance by damping the resonant frequency. The effect of this can be seen in Figure 4-15 where the high frequency oscillation has been removed.

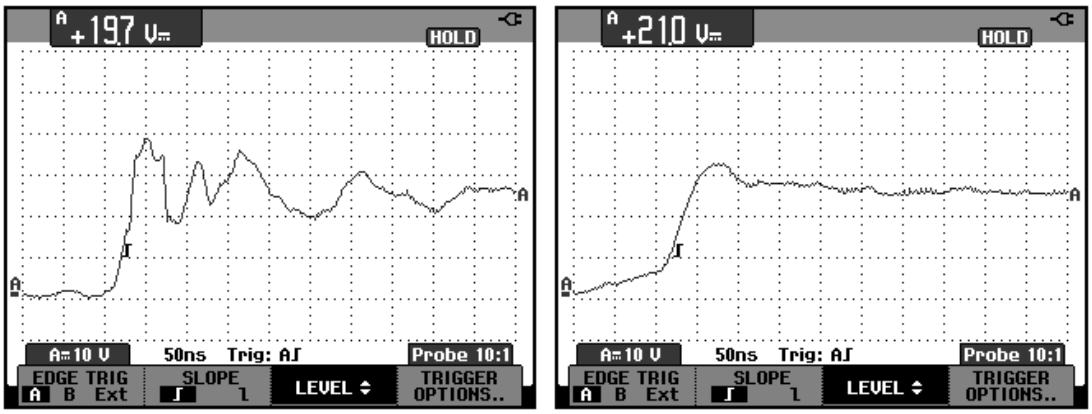


Figure 4-15: Vds of MOSFET, LEFT: Without snubber; RIGHT: With simple RC snubber

The rate-of-rise of the voltage on the MOSFET output is limited by the snubber, limiting the turn-on and turn-off times. By switching the MOSFET slowly, the switching losses increase, but the noise generated is reduced.

A very high frequency, high voltage transient (such as caused by the motor brushes arcing) can travel back from the motor to the MOSFETs. The spike can enter the drain and be coupled into the gate through the internal capacitance. This can cause a MOSFET in its off state to spuriously turn on, or if enough energy is coupled, the gate voltage rises above its maximum rating and the MOSFET is destroyed. The snubber helps to absorb these spikes and protect the transistor.

Transient suppression diodes have been added across each MOSFET to clip any excess voltage caused by back emf. The transient voltage suppressor chosen has the following specifications:

- Manufacturer and part no: Multicomp, part 1.5KE33CA
- Stand off voltage: 28.20 V
- Breakdown voltage: 31.40 V (min) – 34.70 V (max)
- Maximum clamping voltage: 45.7 V
- Power rating: 1500 W

These transient suppressors complement the power supply used (two 13.6 V automotive batteries in series), and the maximum Vds rating of the MOSFETs (55 V).

4.4.1 H-Bridge driver problems

Problems were experienced with the performance of the motor driver which was unrelated to the circuit schematic design. By modifying the PCB layout these problems were minimised or eliminated.

4.4.1.1 Inductive loop

Under certain load conditions, one high side MOSFET turned on unexpectedly causing a "shoot through" (where Vcc is shorted to ground through the two transistors on one side of the bridge). The circuit was analysed, with the most likely cause being the stray inductance of the loop between the gate-L6384-source of the high side MOSFET as shown in Figure 4-16.

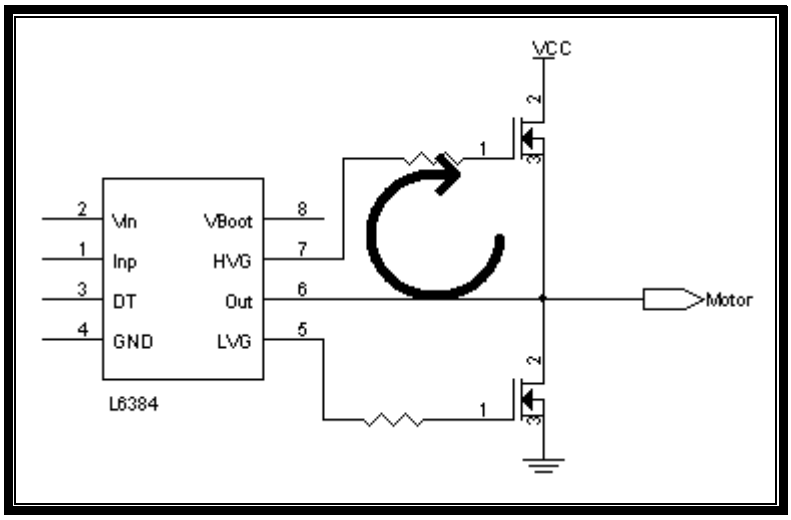


Figure 4-16: Problem caused by inductance of gate-L6384-source loop

The physical separation of the L6384 driver and the MOSFET was reduced, and the track width connecting the gate to the IC was increased. This reduced the stray inductance and prevented the problem from re-occurring on the next prototype board.

4.4.1.2 Ground return path

A problem occurred where the high side MOSFET oscillated during its on period, rather than following the input of the L6384 driver. This is shown in Figure 4-17,

where channel B is the input from the microcontroller, and channel A is the gate voltage on the high-side MOSFET.

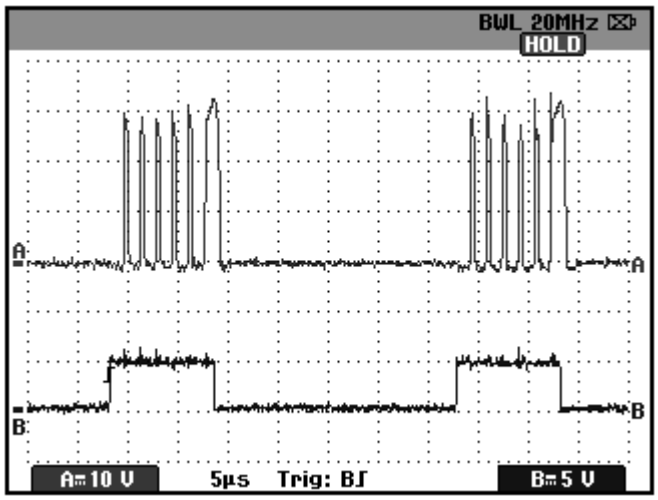


Figure 4-17: High side MOSFET gate oscillation problem

The cause of this problem was a common ground plane used by the logic IC, driver IC and MOSFETs. Current flows through the ground plane when the transistor is turned on to complete a loop back to the battery connector. Due to the resistance of the ground plane this produces a voltage (voltage = current × resistance). The voltage lifts the ground reference of the driver IC, effectively making the PWM input appear to be logic low, switching off the MOSFET. The plane returns to zero voltage potential as no current is flowing in to it, starting the cycle again. An attempt to capture the logic input to the driver IC is shown in Figure 4-18 however, it is not a true representation as the oscilloscope probe capacitance reduces the problem when connected to the PCB.

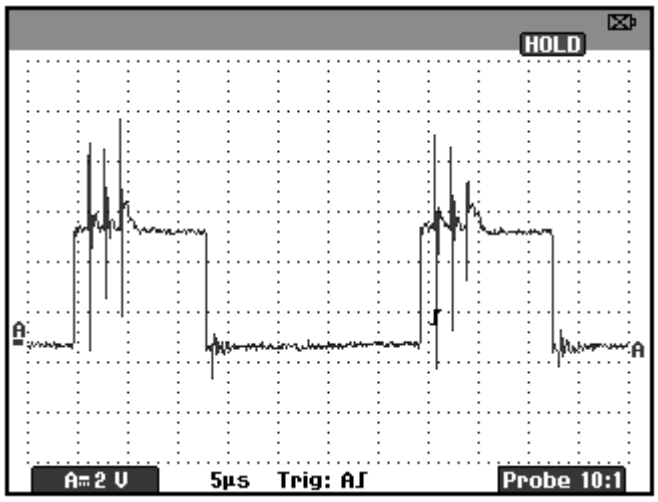


Figure 4-18: Logic input seen by driver IC

To remove the problem, a defined current path was made for the ground connection of the MOSFETs back to the power connector rather than returning through the ground plane. A comparison of two different prototype PCBs in Figure 4-19 and Figure 4-20 show the possible current return paths. The logic inputs to the driver IC's are no longer affected by the MOSFET switching in the right hand PCB layout.

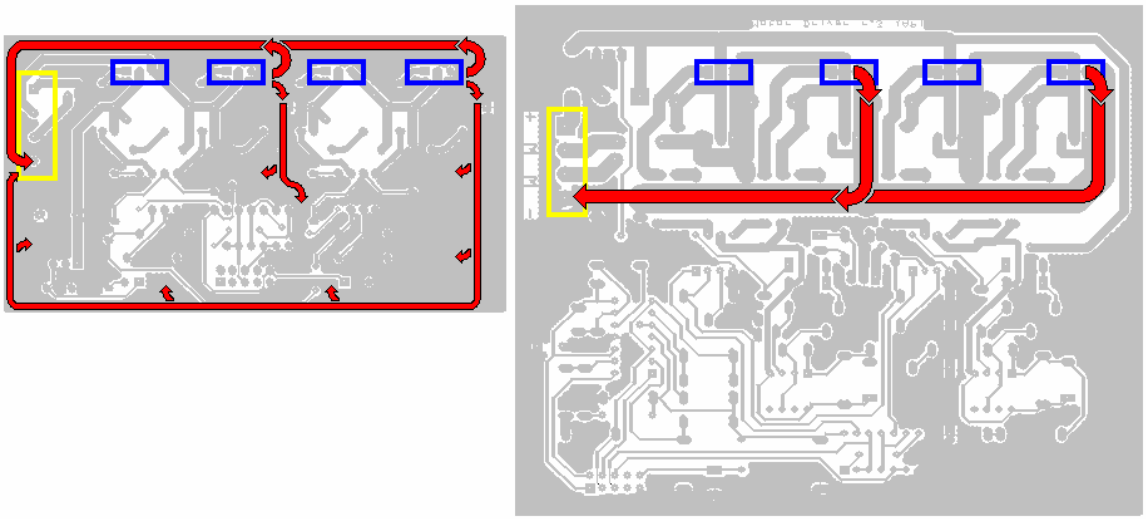


Figure 4-19: Ground return path from MOSFET's back to power connector. LEFT: Through entire plane; RIGHT: Through defined path

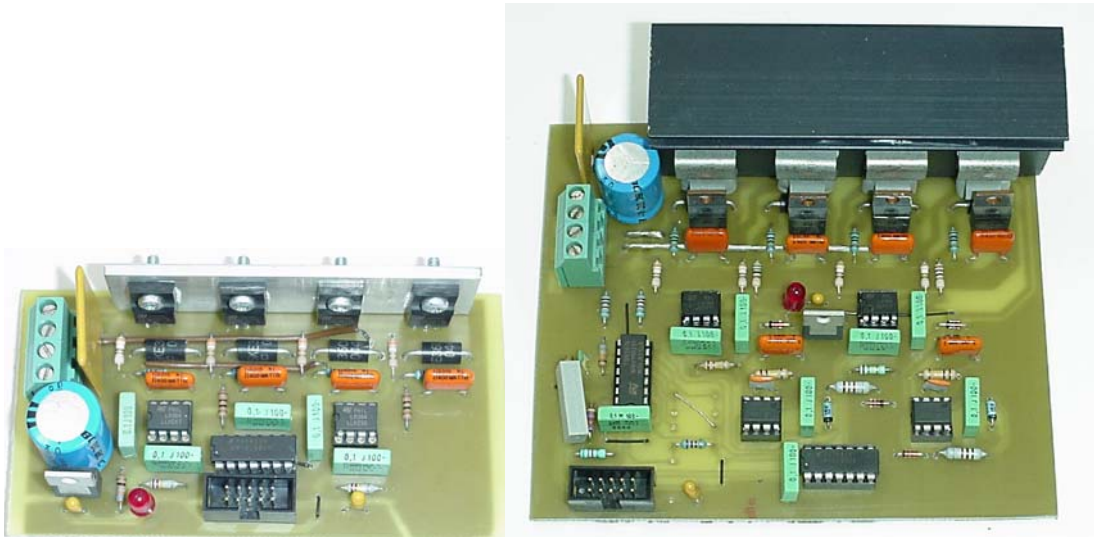


Figure 4-20: LEFT: Prototype PCB with ground loop problem; RIGHT: Final motor driver board

4.5 IMPROVED STEERING MOTOR CONTROL

As a PC control interface and high current output circuit was designed for the main drive motor, the steering motor controller/driver was replaced with an identical board. This removed the current handling and proportional control limitations outlined in section 4.1.

The potentiometer was connected to 0 and 5 V, with the wiper providing a position dependant voltage to the analogue input of the LabPC+. This position feedback allows a Proportional-Integral-Derivative (PID) control system to be developed through software on the PC (section 6.3.4), rather than just the proportional control previously used.

4.6 MOTOR DRIVE CONCLUSION

The microcontroller and power drive circuits have been successfully used on both robots built, and also on MARVIN, another mechatron at the University of Waikato (Figure 4-21).

Figure 4-21: MARVIN indoor security robot

MARVIN's software programming language and data acquisition card are different to those used in the cooperative robot project. The software is a combination of MATLAB and LABVIEW, and the DAQ card is a National Instruments 6025E. The PC interface to operate the microcontroller was programmed by fellow masters student Chris Lee-Johnson with minimal effort despite the system differences. MARVIN is driven in a wheelchair configuration using two 24 V DC motors.

The power driver circuit is capable of operating motors with different characteristics, as demonstrated on MARVIN. The current rating for MARVIN's motors are lower than that of drive motor used for this project (section 3.3) and the inductance and loading characteristics are significantly different. The circuit has performed well during many hours of testing MARVIN's navigation system by fellow Masters students Lucas and Chris. The input voltage range is limited to 17 – 25 V DC, but with a small PCB modification it is expected that the circuit could drive a wide range of DC motors.

5 GLOBAL POSITIONING SYSTEM (GPS)

Accurate positioning of the mobile robots is fundamental to the success of this project. To work together, each robot must have a precise knowledge of its own position, as well as the position of any other interacting robots. One form of sensor used to provide this is GPS.

5.1 BACKGROUND

5.1.1 Established Navigation Techniques

Navigation and positioning are tasks that are crucial to many activities including hiking, surveying, aviation, defence, and vehicle tracking, using a wide range of methods. Some established techniques include:

- Landmarks: Using points at known locations, a current position can be calculated by measuring distances from these points. This only works in a confined area where known points have been generated. A disadvantage is that some landmarks could be damaged or moved.
- Dead Reckoning: Using a compass to determine direction, and measuring the distance travelled from a previously known position, a new position can be calculated. This process is complicated when in the air, on water, or on rough terrain, and any errors accumulate. Accuracy is limited by the tools used to make the distance/heading measurements.
- Celestial: Using known star configurations and the current time, a position of limited accuracy can be calculated. This method is complicated and limited to use only at night in good weather.
- OMEGA: Optimized Method for Estimated Guidance Accuracy is a system based on radio direction beacons. This limits availability to the range of the signals and accuracy. The signals are also subject to radio interference.

- LORAN: A LOng-RAnge Navigational system in which position is determined by an analysis involving the time intervals between pulsed radio signals from two or more pairs of ground stations of known position. Coverage is limited to areas that have been setup with transmitters (mostly coastal), and the signals are easily jammed or disturbed both intentionally or by random noise.
- SatNav: SATellite NAVigation. Measurements are based on low-frequency Doppler measurements so are sensitive to small movements of the receiver, limiting receiver movement. Only a small number of satellites are available, so updates are infrequent.

5.1.2 GPS System Design

GPS is an outdoor positioning system developed by the U.S. Department of Defense (DOD) beginning in 1973, to provide precise worldwide positioning. The system involves 24 satellites 20 000 km above the earth, which circle the earth twice a day, and five ground base stations which monitor and manage each satellite's position. The system cost approximately $12 billion to set up, and is capable of providing a three dimensional position anywhere in the world.

The design was based around multiple requirements from the DOD. To:

- offer 24 hour, all weather positioning
- be suitable for all platforms used by the DOD, including aircraft (helicopter and plane), ship, land (vehicle and foot), and space (missiles)
- provide real-time positioning
- limit precision for unauthorised users. This ensures enemy/civilians are unable to use the system to full precision
- be resistant to interference and/or jamming (intentional and unintentional)
- be passively received by users, i.e. not require a response to be transmitted from the user. This prevents tracking of signals which could be used to give away the current position of soldiers
- be able to support an unlimited number of users
- be low power and low cost for users

- offer redundancy provisions to ensure the system is fully operational at all times

5.2 How GPS Works

5.2.1 Trilateration

GPS works on the principle of trilateration. This is analogous to triangulation without the use of angles. Satellites send out a radio signal, and the receiver measures the time delay for each signal to reach that position on earth. The time delay is converted into a distance, and the position is calculated from the intersection point of multiple measurements from known satellite positions, illustrated in Figure 5-1. This distance is known as a pseudo-range, as it is not accurately measured.

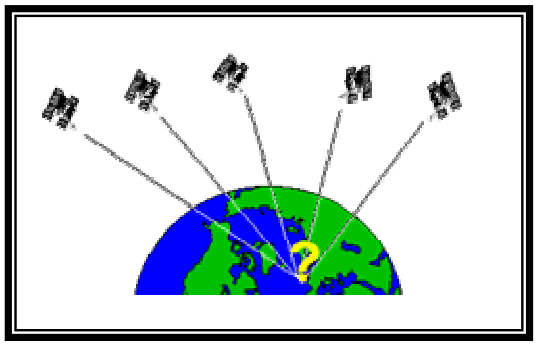


Figure 5-1: Global position using satellites [12]

A two dimensional position is achieved by using two satellites, each with a measured pseudo-range radius sphere around it. The intersection of the spheres will provide two possible positions for the location of the receiver. From these two points, one can usually be eliminated as a non valid answer, as in Figure 5-2, where one point lies on earth, the other in space. For a 3-dimensional position, three satellites again produce two possible points, of which one would be chosen.

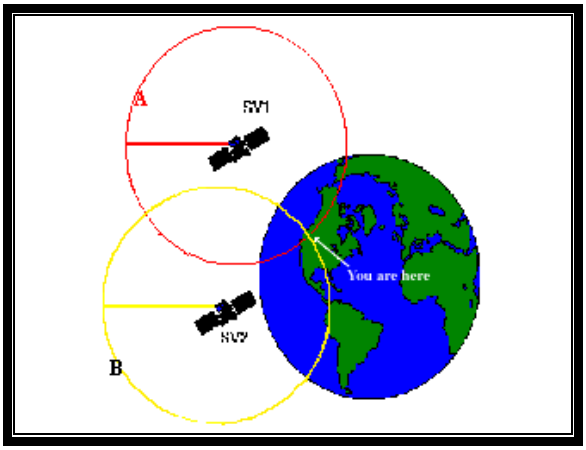


Figure 5-2: Two intersecting spheres provide two possible positions in 2D [16]

This method relies on some fundamental features to provide accurate positioning. First, precise time must be known by the receiver and the satellites to be able to measure the time for the signal to reach the receiver from the satellites. Radio waves travel at approximately 300 000 km/s (the speed of light), so an error of one millisecond in the measured time would create an error of 300 km of the computed distance. Secondly, the exact position of the satellites must be known. Finally, any delays or other errors must be corrected for.

The GPS system is divided into three groups:

i) The Space Segment
ii) The Control Segment
iii) The User Segment.

5.2.2 The Space Segment

The space segment consists of 24 satellites orbiting the earth at an altitude of 20190 km. The satellites orbit on six different equally spaced planes, at an inclination of 55° with respect to the equatorial plane with an orbital period of 11 hours 58 minutes, as shown in Figure 5-3. The DOD guarantees 24 operational satellites 70% of the time, and 21 satellites 98% of the time. This constellation provides the user with a minimum of five satellites visible from any point on the earth; typically each satellite can be seen for approximately five hours above the horizon.

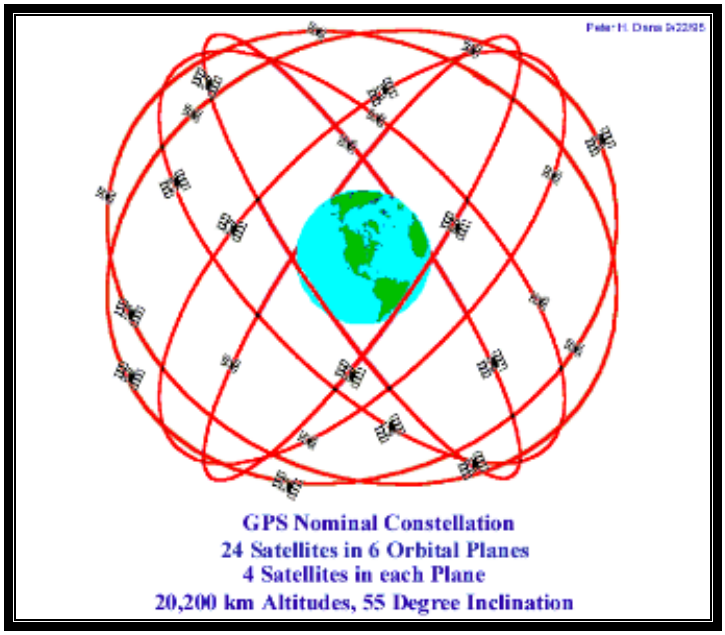


Figure 5-3: GPS satellite constellation [14]

The main reason for using this configuration rather than geo-stationary satellites (satellites which stay in the same relative position above the earth as the earth rotates) is that geo-stationary satellites orbit 36,000 kilometres above the equator. This would provide very limited satellite visibility to users at the planets poles, whereas by using six orbital planes the globe is covered evenly offering the minimum four visible satellites to any point on the earth at any time. The second problem that would occur with using geo-stationary satellites is that the geometrical accuracy of the position would be greatly reduced because all of the satellites would lie in the same plane. To provide an accurate position, measurements need to be made from numerous angles so that the error in the resulting intersection point is reduced.

The satellites transmit signals on two different carrier frequencies: L1 (1575.42 MHz) and L2 (1227.60 MHz), which are received by the *user segment* (section 5.2.4). These signals can penetrate clouds, plastic and glass, but are easily reflected or blocked by solid objects. They can also be blocked by foliage depending on density and water content of the leaves and branches. The signals provide the ranging codes, and the navigation message required for positioning by the user segment.

5.2.3 Control Segment

The control segment consists of five ground monitoring stations, refer Figure 5-4. Located in Hawaii, Ascension Island, Diego Garcia, Kwajalein, with the master control station at Schriever Air Force Base (AFB), formally Falcon AFB, in Colorado Springs, these stations provide satellite health monitoring, telemetry, tracking, command and control, satellite orbit and clock offset computations, and data up-linking.

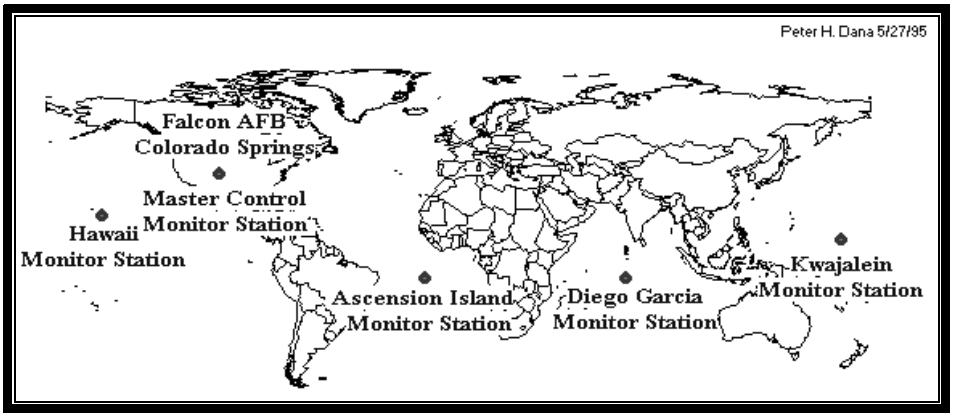


Figure 5-4: GPS control and monitor station network [14]

The monitoring stations track the satellites as they pass overhead by measuring distances to them every 1.5 seconds. This data is then smoothed using ionospheric and meteorological information before points are generated and sent to the master control station every 15 minutes.

The data is processed to estimate the ephemeris (an accurate description of each satellite's position and orbit), satellite clock corrections, as well as assessing the health status of the satellites and determining if any re-positioning may be required. The ephemeris errors are caused by gravitational pulls from the moon, sun and planets, atmospheric drag, pressure of solar radiation and the variable gravitational field from the earth arising from the variation of the solid earth and ocean tides.

The new data is then transferred to the satellites via the up-load stations (Ascension Is., Diego Garcia and Kwajalein). Due to the worldwide spread of the control

stations, all GPS satellites are tracked 92% of the time. The space and control segments are the responsibility of the U.S. Air Force Space Command, Colorado.

5.2.4 User Segment

The user segment is the major commercial segment, and is the area of main interest. It involves the applications using the positioning system including:

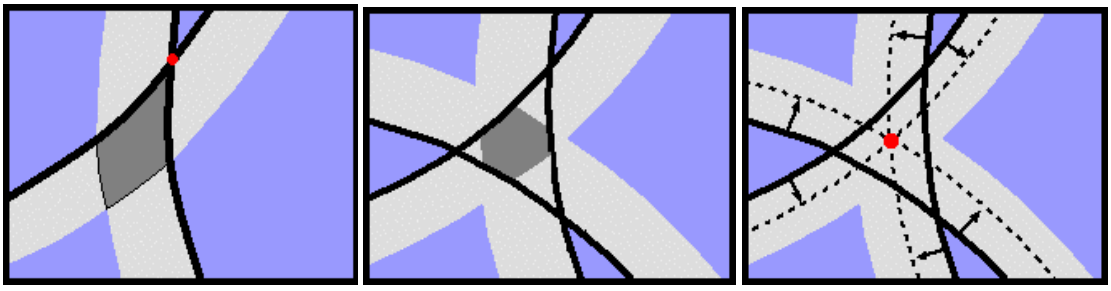
- Land, sea, air and space navigation and tracking. This includes tasks such as route navigation, collision avoidance, vehicle tracking, search and rescue etc. The accuracy and cost are generally low, although high speed results are usually required.
- Surveying and mapping. This requires high accuracy, and usually uses specialised hardware and software. Real-time is not always required.
- Military applications. These applications are often similar to civilian uses, but require very high system reliability.
- Recreational use. Low cost and easy-to-use receivers are required.
- Specialised uses. This includes precision timing, atmospheric studies etc. The requirements vary, but are usually demanding, i.e. high precision position/time, real-time applications etc., which require specifically developed products.

New applications are being created continuously to take advantage of the properties GPS can offer. To most users the control and space segments become almost transparent to the operation of the receiver.

5.2.5 Time Keeping

The precise time is achieved using atomic clocks, using cesium and rubidium, in each of the satellites. Although these clocks had never been tested in space before GPS, they remained stable, offering long term frequency stability. These precise clocks cost hundreds of thousands of US dollars each, and the weight is more than 20 kg, making them impractical to be required by the receivers. Quartz crystal clocks are generally used in the receiver equipment. They offer excellent short-term stability,

but drift over long periods of time. As a work-around to this problem, the receivers track another satellite (four satellites for a 3D position) to resolve the fourth variable: time.



(a) (b) (c)

Figure 5-5: (a) 2-D position and error with 2 satellites; (b) 2-D position and error with 3 satellites; (c) 2-D position and error reduced with 3 satellites

As shown in Figure 5-5(a), a two dimensional position can be achieved using two satellites (if the other intersection is ignored due to a ridiculous answer – e.g. in space or inside the earth). This puts the user at the position marked by the red dot. However, a very small timing error in the signal delay calculated results in a large distance error, shown in light grey. This error now means the user could be anywhere inside the dark grey area. Figure 5-5(b) shows the introduction of a third signal, and shows that a timing error has occurred as the three signals do not intersect at a common point. The position could be anywhere inside the dark grey area. Figure 5-5(c) shows the adjustment of the calculated distances, with respect to a constant time error. Note that all signals are adjusted by the same time correction (the same distance), to produce an accurate position marked by the red dot. The errors in the time have virtually been eliminated. This procedure is repeated using four satellites to produce a three dimensional position.

5.2.6 Satellite Signals

Modulated onto the two carrier frequencies, refer section 5.2.2, are two *ranging codes* and the *navigation message*. The ranging codes are:

- The C/A code (Coarse Acquisition or Clear Access code) designed for civilian use with limited accuracy.

- The P code (Precise or Private code) designed for military and other authorised users.

The C/A code consists of a Pseudo Random Noise (PRN) sequence of 1023 bits, defined for each individual satellite. The bits appear as random noise, but are chosen to have good correlation properties. The C/A code is modulated onto L1 at a rate of 1.023 MHz, and repeats every millisecond. An identical code is produced in the receiver to match that of the tracked satellite. The received pattern will be retarded due to the time delay of the signal flight, where the delay is proportional to the distance travelled, illustrated in Figure 5-6. The receiver correlates the received pattern with that generated internally; when the two codes align they fully correlate, but have a low correlation with any other alignment.

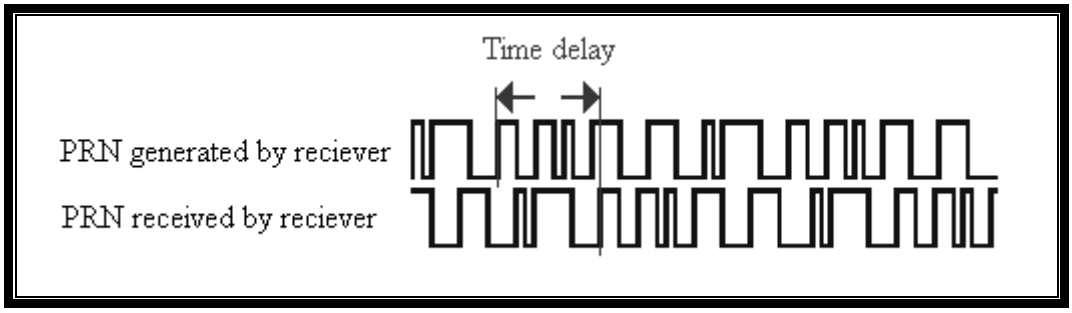


Figure 5-6: Retardation of the received signal is proportional to distance travelled

As the code is only one millisecond long, this provides a signal 300 km long. This means that the measured distance is a distance of 0 – 300 km plus an integer value of 300 km. This integer can easily be calculated as the resulting intersection of signals from multiple satellites will not coincide if it is incorrect. This is known as integer ambiguity. Note that all the satellites use the same carrier frequencies, but have different C/A codes. This system is used in most civilian uses, and is known as the Standard Positioning Service (SPS).

The Precise Positioning Service (PPS) uses the P code, and works in the same way that the C/A code does. It uses approximately 10^{14} bits, which are modulated at 10.23 MHz, therefore the code doesn't repeat for approximately 267 days. Each satellite uses a one week portion of the code, and is reset each week. The increased frequency allows for a more accurate alignment, and therefore a more accurate position. The major difference between these two codes is that the P code is transmitted

simultaneously on both L1 and L2. As the delay caused by the ionosphere is a function of frequency, by making measurements on both frequencies this source of error can be eliminated. This provides a major advantage over the SPS, as atmospheric delays cause errors that are variable and are difficult to predict. Also, the length of the P code ensures that there is no integer ambiguity associated with the measurement.

The navigation message is also modulated onto both L1 and L2 at a rate of 50 Hz. The entire message comprises 25 frames, each frame being 1500 bits long, taking a total of 12.5 minutes to broadcast. Each frame contains five sub frames, which are sent every 30 seconds, consisting of:

1:	Satellite clock corrections.
2 and 3:	Precise satellite orbital data (ephemeris data)
4 and 5:	System data.

The clock and orbital data repeat every frame to allow a system with some data to initialise much more quickly than waiting for the entire message to repeat, only requiring 30 seconds for the frame, compared with 12.5 minutes for the complete data set.

Clock corrections are an offset for that satellite's clock. Ephemeris data parameters describe the individual satellite's orbit over a short section of the total orbit. Ephemeris data is normally updated hourly, but can be used for up to four hours with little error added. Other data includes:

- Almanac – approximate orbital data of all the satellites over a long period of time (can be used for months)
- Satellite health status
- Ionosphere delay estimation parameters, to try and reduce errors where the system is unable to cancel the error out (standard low cost single unit)
- UTC (Coordinated Universal Time)/ system time parameters
- Status messages

5.3 ERRORS/LIMITATIONS

GPS calculated positions experience errors from many different sources. To achieve an accurate position, these errors must be reduced as much as possible. Some typical error sources and approximate position errors introduced are shown in Table 5-1:

Source	Uncorrected Error Level
Ionosphere	10 metres
Troposphere	1 metre
Measurement Noise	2 metres
Ephemeris Data	1 metre
Satellite/Receiver Clock Drift	1.5 metres
Multipath	0.5 metre
Selective Availability	100 metres

Table 5-1: GPS error sources

5.3.1 Ionosphere

The ionosphere is the upper part of the atmosphere (70 km – 400 km above the earth's surface). The ionosphere contains charged particles, which have the effect of slowing down the code (group delay), and speeding up the carrier (phase advance). This results in the calculated pseudo-range being larger than the actual distance. The intensity of this effect depends on three major factors; the elevation of the satellite, the time of day, and the geomagnetic latitude of the receiver. Satellites at low elevation angles experience more delay since the signal must pass through a greater section of the ionosphere. Due to solar radiation, the error peaks during the day, and reduces at night. A typical night time delay would be five times less than that measured during the day. The delay is also increased at the geomagnetic poles and equator.

5.3.2 Troposphere

The troposphere is the lower half of the atmosphere (0 – 70 km). It affects the satellite signals by delaying both the code and carrier phase, and causes refraction. This refraction is dependant on atmospheric pressure, temperature, and humidity.

5.3.3 Measurement Noise

These errors are due to the processes used within the receiver. They are dependant upon the antenna used, the ADC method, the correlation process used, and the software and hardware design.

5.3.4 Ephemeris Data

The satellite ephemeris is predicted by measurements taken around the world, returned to base, modelled, and sent to the satellites. Small errors occur due to imperfect modelling as well as the delay in the procedure.

5.3.5 Satellite/Receiver Clock Drift

A one nanosecond error in the satellite clock results in a pseudo-range error of approximately 30 cm. Although atomic clocks are used, a nanosecond error can accumulate over a three hour period. This error must be offset. To achieve this, the control segment monitors these errors and sends a correction with the navigation message.

The receiver only uses a quartz crystal clock, which has limited accuracy. This error is reduced by tracking an additional satellite (refer section 5.2.5).

5.3.6 Multipath

Multipath errors as shown in Figure 5-7 occur when a signal is reflected off a nearby object, building or the ground before reaching the antenna.

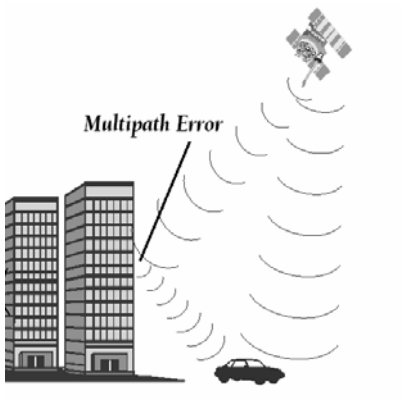


Figure 5-7: Reflection of signals can provide incorrect range measurements [15]

5.3.7 Selective Availability (SA)

Although the P code was designed to be much more accurate than the C/A code, in practice the accuracy of the C/A code was better than expected. To limit civilian accuracy, the DOD introduced selective availability (SA). This involved purposely introducing errors into the orbit data and/or the satellite clock frequency over a long time period. The result was a reduction in the accuracy from approximately 25 metres to 100 metres. Selective Availability was introduced on July 4, 1991, and turned off on May 2, 2000 due to a decision by the DOD. Stated in a White House Press Release, the decision was to "encourage acceptance and integration of GPS into peaceful civil, commercial and scientific applications worldwide; and to encourage private sector investment in and use of U.S. GPS technologies and services." The US DOD reserves the right to continue selective availability if hostility warrants the need for it at any time.

Other limitations include:

- Dilution of Precision (DOP)
- Signal Availability
- Anti-spoofing (AS)

5.3.8 Dilution of Precision (DOP)

Dilution of precision is a factor limiting the accuracy of the position due to the geometric positioning of the satellites. If all the satellites being used for a position calculation are grouped close together, the error in the position is larger than when the satellites are positioned further around the sky, shown in Figure 5-8.

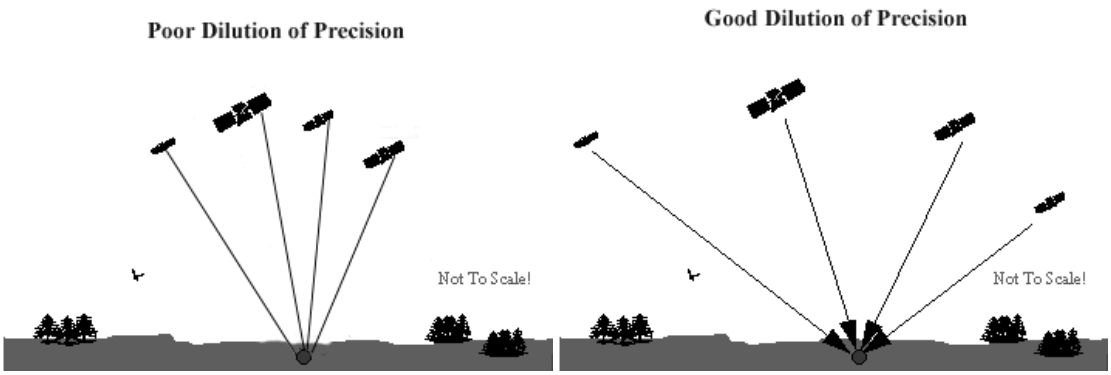


Figure 5-8: Satellite positioning affects dilution of precision

Dilution of precision can be expressed in many forms depending on the geometry used i.e.
- GDOP - geometrical dilution of precision
- HDOP - horizontal dilution of precision
- TDOP – time dilution of precision etc.

The value of the DOP can help indicate the quality of the position to the user (the lower the value, the higher the potential position accuracy).

5.3.9 Signal Availability

The signals require line-of-sight to reach the receiver, and this is not always possible. Obstructions such as buildings, terrain, and a wet canopy can prevent the receiver from tracking certain satellites. This is especially true when the receiver is trying to receive a good DOP as scattered satellites across the entire sky are required. This is the reason GPS is not effective indoors.

5.3.10 Anti – spoofing

Anti-spoofing is another limitation introduced by the DOD (similar to SA) on January 1994. The P code is encrypted using a secret W code to produce the Y code. This Y code is then broadcast on L1 and L2 in place of the P code. Only authorised users have the key required to decode the Y code back into the P code required for precise positioning. This is intended to prevent "spoofing" military GPS receivers by transmitting false P code signals from other satellites, or jamming the signal using a ground based transmitter. This also reduces the use of both frequencies to civilian users as the code is unknown.

5.4 GPS ENHANCEMENTS

As technology improves, advancements have been made to the basic receiver hardware and/or receiver configuration. The reported position accuracy is enhanced by reducing errors and increasing measured resolution.
These include:

- Carrier aided tracking hardware
- Differential GPS configuration
- Post processing techniques
- Dual frequency tracking hardware
- Carrier phase tracking hardware / Real-Time Kinematics (RTK)

5.4.1 Carrier Aided Tracking

Carrier aided tracking is a process in which the carrier phase is used to apply an exact alignment of the C/A code in the correlation process performed by the receiver. This generates an increase in the measured accuracy between the satellite and receiver by removing any 'slip' in the alignment of the signal received and signal generated.

5.4.2 Differential GPS (DGPS)

Differential GPS (DGPS) is a simple idea of using two receivers which will both experience the same errors if they are placed close together. A "base station" is positioned at a fixed known position. Since the base position is known, and each satellite position is known from the ephemeris, the actual distance between each satellite and the base can be calculated. The base station measures the pseudo-range to each satellite using the techniques discussed. The difference between the pseudo-range and the actual range measurements becomes a correction factor due to the errors from each satellite. As the base and rover receivers are close together, within a few hundred kilometres, the signals travel through the same area of the atmosphere, so the same errors are experienced by both receivers. These correction factors are then applied by the second "rover" receiver to provide a more accurate position.

For real-time operation, the corrections must be sent directly to the rover unit(s). This is normally done using a radio link from the base station. Due to the movement of the satellites, and changes in their clocks, corrections vary rapidly with time and so a constant update is required. Most errors are removed using DGPS as shown in Table 5-2:

Source	Uncorrected Error	After DGPS
Ionosphere	10 metres	Mostly Removed
Troposphere	1 metre	All Removed
Measurement Noise	2 metres	Not Removed
Ephemeris Data	1 metre	All Removed
Satellite/Receiver Clock	1.5 metres	All Removed
Multipath	0.5 metre	Not Removed
Selective Availability	100 metres	All Removed

Table 5-2: DGPS error sources

DGPS removes errors that are common to both receivers; measurement noise and multipath are not removed as these errors are local to the receiver, not due to the satellites or atmosphere.

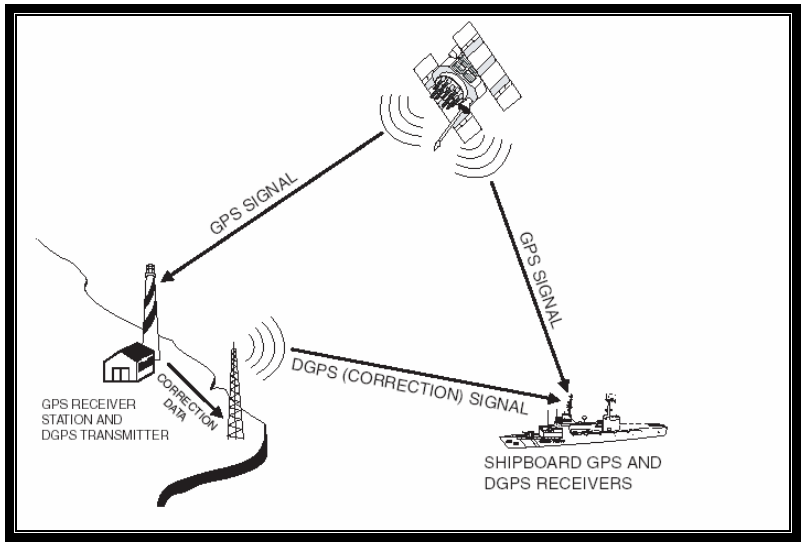


Figure 5-9: DGPS configuration as used by the US coast guard [15]

The US Coast Guard installed the Maritime Differential GPS (DGPS) service, a radio-beacon DGPS network for continental US coastal waters, illustrated in Figure 5-9. The network provides free corrections to a large area of the US, allowing high accuracy to users that have radio equipment to receive the corrections, rather than having to setup their own base station. These corrections eliminate the SA error that was purposely put in by the DOD. This elimination was possibly one reason the DOD have currently disabled SA.

As not all applications require real-time results, e.g. surveying where data can be collected and processed at a later time, the data link can be eliminated from the system. Instead the rover must store time data as well as position data, and the base must store time and correction data. Corrections can then be applied for that particular time, and the position accuracy increased. This is known as *post-processed DGPS* (see section 5.4.3). This has the advantages that communication equipment is not required, and also quality assurance measures can be implemented.

As the errors are individual to each satellite signal, and many satellites are often available for a receiver to track, some form of collaboration is required between the base and rover. Two different methods of DGPS are available:

1) Block Shift Method.
2) Range Corrections Method.

5.4.2.1 Block Shift Method

The Block Shift method is easy to implement, but has serious limitations. It works by using two receivers that are forced to use the same satellites. The base station selects four satellites to use, and calculates a three dimensional position from the satellites. As the actual location is known, a correction (ΔX, ΔY and ΔZ) is calculated from the difference between the satellite-calculated position, and the actual site. The rover receiver uses the same satellites selected by the base station to also compute a position. The ΔX, ΔY and ΔZ are sent from the base to the rover, which are then added to the rover receiver's position to remove any common errors. This process is illustrated in Figure 5-10.

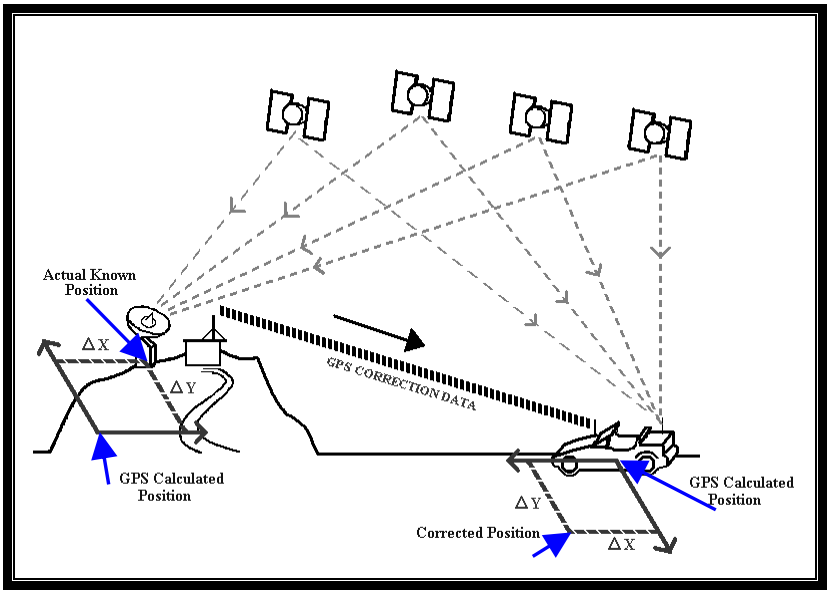


Figure 5-10: Block shift DGPS

The main limitations come from the fact that both receivers *must* use the same satellites. In areas where buildings/land/foliage could obstruct a line-of-sight to any of the required satellites, the rover would not be able to correct its position at all. If only a limited number of rover units were in use, the rover could communicate to the base station and change the selected satellites being used for the fix.

5.4.2.2 Range Corrections Method

The more commonly used method is the range correction method. Rather than making corrections to the co-ordinates, the correction is applied to the pseudo-ranges *before* computation of the position is performed. The base station position is known, as is each satellite's position (from the ephemeris), so the range from each satellite to the base station can be calculated. The base station measures the pseudo-range to each satellite. Corrections are then generated between the actual and pseudo-ranges, which are sent to the rover receiver, shown in Figure 5-11. The rover applies the corrections to the pseudo-ranges it measures, and then calculates its position.

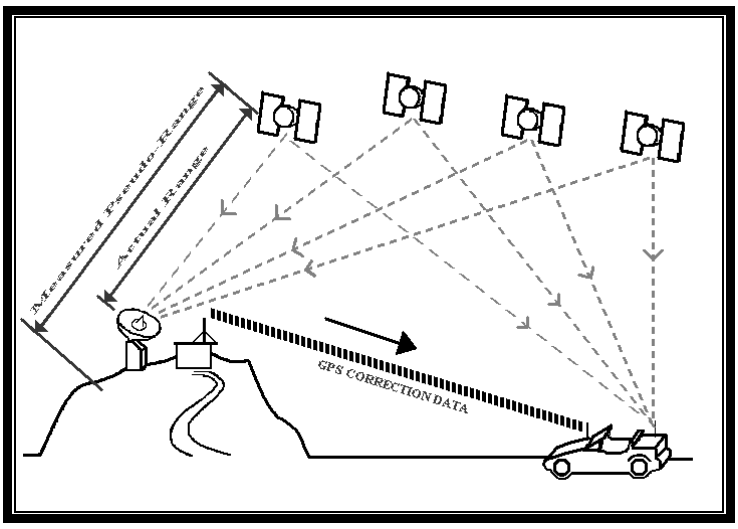


Figure 5-11: Pseudo-range correction DGPS

This technique reduces the limitations of the block shift method, as individual corrections are generated for all satellites visible by the base station. This allows the rover to select any combination of four satellites that are within the base station's view.

5.4.3 Post Processing Techniques

Post processing techniques are accomplished following the recording of the data from the receiver with its associated time stamp. For applications not requiring real-time results this has some advantages:

- Communication equipment is not required to broadcast/receive the corrections.

- Quality control can be implemented to ensure required data accuracy is achieved.
- A base station does not necessarily need to be owned as corrections can be acquired from an 'outside' source.
- Position accuracy can be increased.

Position accuracy can be increased using post processing as the ephemeris provided in real-time in the navigation message (refer section 5.2.6) is an estimate of the position that each satellite will be at. The actual position can be obtained from the DOD on the day following the data recording. The pseudo-range corrections can also be generated from multiple base stations at the same time and "averaged" together, rather than from a single location, increasing their reliability.

5.4.4 Dual Frequency Tracking Hardware

As described in section 5.2.6, by using dual frequency measurements, ionospheric errors can be eliminated. These are used in military applications, but since AS (refer section 5.3.10) has been in place, the general public does not have access to use the Y code on L1 and L2. Manufacturers have developed complex partial solutions to this problem, using the knowledge that the same Y code is broadcast on both L1 and L2. By receiving both of the codes on L1 and L2, a correlation can be made between the two raw "undecoded" codes, providing a time difference due to the ionosphere delay. This can then be used to reduce the errors in measured distance between the satellite and the receiver in the same way that the military units do.

5.4.5 Carrier Phase Tracking / Real-Time Kinematics (RTK)

Carrier phase tracking uses the phase of the carrier signal to provide distance measurements. The length of one bit of the C/A code when transmitted is approximately 300 m; whereas the wavelength of the carrier signal L1 (1575.42 MHz) is approximately 19 cm. By determining the phase of the received signal, a resolution of millimetre magnitude can be calculated. This has the same problem as the C/A code: integer ambiguity. A measurement between 0-19 cm can be calculated, but it

must be added to an unknown multiple of 19 cm. This idea can also be used on L2 (wavelength 24 cm).

The Real-Time Kinematics (RTK) method provides real-time, very accurate positioning. The base station and rover must both lock-on to a number of satellites, and resolve the wavelength ambiguity. The rover then moves, tracking the changes in carrier phase relative to any phase changes experienced by the base station. Using the carrier phase, very small movements can be measured.

The initialisation of the receivers is a slow complex process, and the time required can vary depending on multiple factors:

- The confidence level required for correct integer estimation.
- The number of visible satellites.
- Whether a single or a dual frequency receiver is being used.
- The strength of any multipath signal.

A high confidence level requires a longer period of time as additional data must be received. The number of available satellites, and hence available data, influences the ease of calculating the ambiguity, e.g. the required time is approximately five times longer if six satellites are used compared to eight satellites. Using both frequencies (L1 and L2) offers the same result as having more available satellites. Multipath is the main reason for incorrect integer estimation, so less interference due to multipath will result in less time to accurately measure the ambiguities. Multipath is directly related to the surroundings (section 5.3.6) and also to the antenna's ability to reject it. Expected times for initialisation can range from 5 minutes to 40 minutes.

If either the base station or the rovers lose the signal from a tracked satellite, the receiver must reinitialise the satellite carrier phase ambiguity. This limits the canopy and surroundings this method can operate in. The velocity of the rover is also limited since the receiver must be capable of tracking the changes in the carrier phase without jumping an entire wavelength.

As a general rule, for a short baseline of 10 km or less, 8 satellites are required if using single frequency receivers or 6 satellites are required if using dual frequency

receivers. For a long baseline, 8 satellites are required using dual frequency receivers. These values refer to satellites that are visible to *both* the base and rover. Due to the configuration of the satellites (section 5.2.2) GPS only provides 8 satellites 32% of the time. To reduce this limitation, manufacturers are building receivers which can operate using GPS (NAVSTAR) as well as GLONASS, a Russian system which is very similar, but less common than NAVSTAR. It uses its own satellites with the same pseudo-range principle as NAVSTAR; the main variation is that each satellite uses a separate frequency.

Despite the limitations and cost of the equipment, RTK can offer millimetre resolution positioning.

5.5 EXPERIMENTAL SETUP

5.5.1 GPS Receiver and Interface

The GPS receivers purchased were Motorola M12 Oncore (section 2.10.3). The M12 was selected as:

- It is designed to interface directly to a serial port
- It is designed for automotive use, is small, with a small antenna and quick initialisation on power up
- Power consumption is low
- 12 parallel channels mean that 12 satellites can be tracked at once
- Pseudo-range corrections can be generated and received
- Satellites can be ignored
- A range of settings and features can be specified, such as filtering used, 3D/2D position, mask angle, almanac and ephemeris input and output
- Full receiver, satellite, and position status is available

The M12 was shipped as a "plain unit", requiring a suitable power supply and serial interface to connect the unit to a PC. The power requirements of the M12 are 2.75 to 3.2 V DC; 50 mVp-p ripple (maximum). As only a 3.3 V supply is offered within a PC, a regulator had to be designed for this voltage. The serial interface used 0-3 V

levels at 9600 baud, 8N1 (8 data bits, No parity, 1 stop bit). These levels had to be converted to RS232, namely 0 V converted to between 5 V and 15 V, and 3 V converted to between -5 V and -15 V. A circuit was designed (Appendix A5) and fabricated (Figure 5-12(a)) to meet these needs.

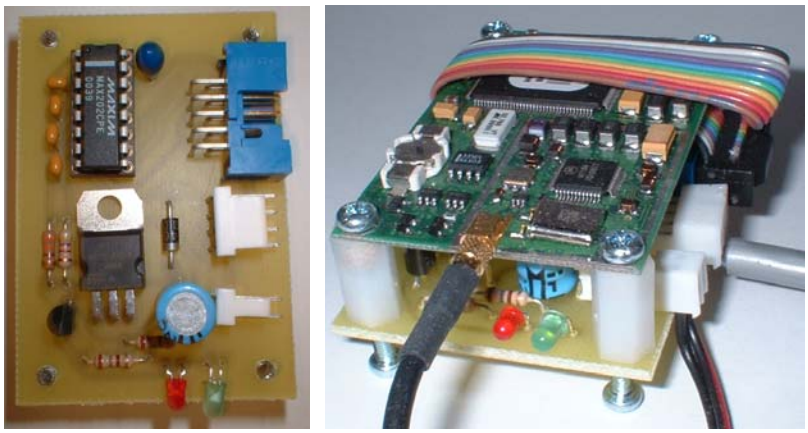


Figure 5-12: LEFT: Power supply and serial interface; RIGHT: M12 Receiver mounted on interface PCB

The interface PCB connects to the M12 through a 10 pin header. It has an input range of 6 V – 40 V DC, with polarity protection. LEDs show power on (green) and a one pulse per second (red) signifying that the M12 is connected and running. The two serial ports are computer level compatible, using RS232 voltages. The M12 mounts directly on top of the interface PCB, which has the same dimensions (60 mm × 40 mm) and same mounting configuration, shown in Figure 5-12(b).

5.5.2 Software

Software written compares various methods of operating the GPS receivers; a standalone configuration, pseudo-range correction DPGS, and blockshift DGPS.

The standalone method uses built-in software filtering to minimise atmospheric errors, with estimated ionosphere correction values sent with the navigation message.

The pseudo-range correction method uses two receivers, a base station, and the rover. Documentation on using the M12 as a base station is not supplied by Motorola in the

M12 manual even though this is a feature offered, therefore the M12 may not be in its optimal configuration for this method. The initialisation process used involved:

Base Station:

- Disable ionosphere/troposphere corrections
- Set the mask angle to 10° to ignore satellites low on the horizon
- Disable the position filter
- Set the position to a known location
- Enable the position hold so that position remains fixed
- Output corrections every 10 second

Rover:

- Disable ionosphere/troposphere corrections
- Set and hold height to provide a 2D position output
- Apply corrections from base station

The software reads the corrections from the master station and applies them to the base station every ten seconds. The rover reports its position every second, which is recorded and displayed.

The accuracy of the corrections is directly related to the accuracy of the position that the base station is set to. The base station position was set by averaging the standard GPS position over many hours, rather than by surveying the site.

The block shift method initialises each receiver to turn off ionosphere/troposphere corrections, and set the receiver height to generate 2D positioning. The satellite position message is retrieved from the receiver every 60 seconds to provide the positioning of all the visible satellites, and the selection of which to use. The data is plotted to the screen and logged. The three "highest in the sky" satellites are selected. This selection was chosen as:

- It provides simple criteria
- All receivers are likely to have a line of sight to the satellites
- Atmospheric errors are reduced compared to low angle satellites
- Multipath is reduced

A disadvantage is that the DOP is high because the satellites are grouped together.

The main problem experienced with this method is the time required to acquire the satellites by the receiver. By adding the satellite number to the ignore list, the receiver stops tracking that satellite rather than just ignoring it in the position calculations. This means that when the satellite's selection changes, the receiver must acquire the new satellite's signal, delaying the time before a position can be provided. If an obstruction blocks *any* of the required satellites, a position cannot be given, and the receiver reports *bad geometry*. During this time the output does not change from the last known position.

The degree to metre conversion is performed using the following method:

- Latitude: 1 minute of latitude is 1842 m and is constant over the world.
- Longitude: 1 minute of longitude is 1855 m at the equator. As the position gets closer to the poles, longitude separation goes to zero. Assuming a spherical earth, at -38° latitude, the 1 minute of longitudinal separation will be:

$$1855 \text{ m} \times \cos(38^\circ) = 1462 \text{ m} \qquad \textbf{Equation 5-1}$$

5.5.3 GPS Results

Experimental results from the different positioning methods available are given in section 7.2. The results are analysed with respect to this project's requirements.

6 SOFTWARE

6.1 SOFTWARE OVERVIEW

A range of different software routines are required to interface sensors, provide feedback and control to the user, and to control the motors. Visual C++ programming language was chosen as the platform for the software development for the following reasons:

- Network support is natively integrated. Due to the nature of this project communication is fundamental. The use of third party software routines is not desirable as anomalies may exist.
- Real-time applications can be built. The capability to multi-thread tasks allows different parts of the program to run simultaneously rather than sequentially. This is advantageous for running a critical control loop, while polling sensors in the background at various update rates.
- Software routines are available from National Instruments for the LabPC+ DAQ card.
- It is a very well known and powerful programming language which will support future developments for this project.

An overview of the hardware is shown in Figure 6-1. Each component uses a different data transfer protocol, requiring individual software to interface each part of the complete system. A control system is also required to interpret the data received and control the motion of the mechatron.

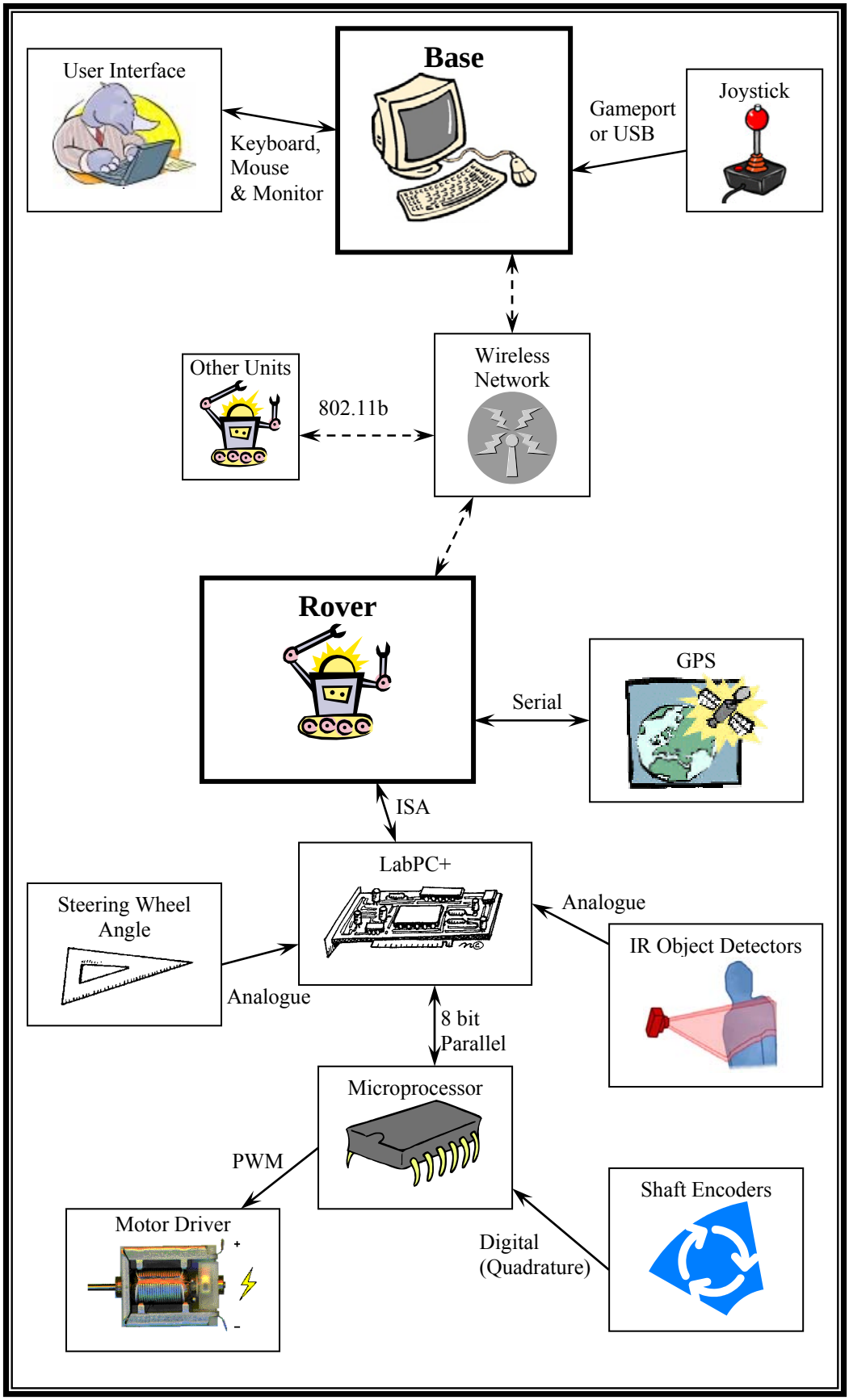


Figure 6-1: Overview of hardware

6.2 COMPONENT INTERFACE

6.2.1 GPS

The GPS requires a serial port to send and receive data. The code shown in Figure 6-2 creates a handle to the serial port by calling **`CreateFile()`**. The buffers are set using **`SetupComm()`**, followed by **`SetCommState()`** which configures the data transfer parameters. Finally the timeouts are set using **`SetCommTimeouts()`**.

```
HANDLE          hCom;
DCB             m_dcb;
COMMTIMEOUTS    m_CommTimeouts;

hCom = CreateFile("COM1",GENERIC_READ | GENERIC_WRITE,0,NULL,OPEN_EXISTING,0,NULL);

SetupComm(hCom, 512, 512);

m_dcb.BaudRate = 9600;
m_dcb.ByteSize = 8;
m_dcb.Parity = NOPARITY;
m_dcb.StopBits = ONESTOPBIT;
m_dcb.fAbortOnError = TRUE;

SetCommState(hCom, &m_dcb);

m_CommTimeouts.ReadIntervalTimeout = 10;
m_CommTimeouts.ReadTotalTimeoutConstant = 10;
m_CommTimeouts.ReadTotalTimeoutMultiplier = 10;
m_CommTimeouts.WriteTotalTimeoutConstant = 50;
m_CommTimeouts.WriteTotalTimeoutMultiplier = 10;
SetCommTimeouts (hCom, &m_CommTimeouts);
```

Figure 6-2: Initialisation of the serial port

Data is written to the serial port using the routine **`WriteFile()`**, where **`lpBuffer`** is the data to write, and **`nNumberOfBytes`** is the number of bytes to write. Similarly **`ReadFile()`** reads data from the serial port, and stores it in the array **`sResult`** (refer Figure 6-3).

```
WriteFile(hCom,lpBuffer,nNumberOfBytes,&iBytesWritten,NULL);

ReadFile(hCom, sResult, nNumberOfBytes, &iBytesRead, NULL);
```

Figure 6-3: Reading and writing to the serial port

The Motorola M12 is initialised by sending the instruction: ***@@Ha****r**C*<*CR*><*LF*>[19].

Where: *@@Ha* = Ascii command

r = Output rate in seconds;

C = checksum.

<*CR*><*LF*> = carriage return, line feed

This instructs the GPS receiver to output a continuous position/status message every *r* seconds. The checksum is generated by performing an exclusive OR on each byte of the message.

The response from the M12 is a 154 byte long message, containing position, time, satellite status, and receiver status data. The information required is extracted from the message and stored in local variables.

Figure 6-4 shows an example of the code, extracting four bytes of data and restructuring it to form the current longitude position. The position is given in milliarcseconds (mas) with a range of -648000000 to 648000000. Each byte is left shifted appropriately, and then added together to give a 32 bit value. Negative values are accounted for by storing this value into a **`signed __int32`** variable. If the most significant bit (MSB) of a signed integer is set, the value will be negative. By ensuring the integer variable is 32 bits in length the GPS receiver can control this bit, rather than having to manually check the MSB and modify the value to be negative. The four byte value is divided by 3600000 to scale from mas into degrees.

```
__int32 temp;

temp =  ((unsigned char)(command[15])<<24)+((unsigned char)(command[16])<<16)+
        ((unsigned char)(command[17])<<8)+(unsigned char)(command[18]);

data.fil_longitude = (double)(temp) / 3600000;
```

Figure 6-4: Retreiving longitude position from M12 message

6.2.2 LabPC+

Provided by National Instruments, NiDAQ v6.9.3f3 provides the interface for the PC to the LabPC+. This is the last NiDAQ release supporting the LabPC+. Installed with Visual C++ support, all routines are provided by including the libraries into the compilation.

Note: This restricts compilation/execution to machines with NiDAQ installed.

6.2.3 Microcontroller

A parallel communication interface is used to send bi-directional data between the microcontroller and the LabPC+. Using four handshaking lines, each device has a request-to-send (output) and acknowledge (input) signal. To send data, the PC sets the request-to-send line. The microcontroller acknowledges the request, allowing the PC to drive the data lines. After a short delay the microcontroller reads the value. When the acknowledge line is released, the PC releases the data lines and then the request signal. The transfer is identical in the opposite direction, using the other two handshaking lines. This method is half-duplex as data cannot be sent in both directions simultaneously.

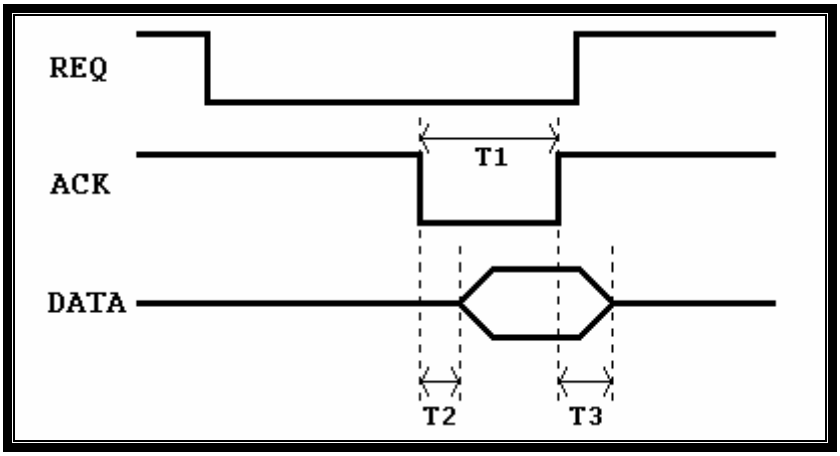


Figure 6-5: Handshaking protocol and timing constraints between microcontroller and LabPC+

	Minimum	Maximum
T1	300ns	-
T2	-	300ns
T3	20ns	250ns

Port A is configured as a bi-directional port using the **`DIG_Prt_Config()`** routine. Reading and writing is performed using **`DIG_Out_Port()`** and **`DIG_In_Prt()`** respectively, refer Figure 6-6. The instruction set to control the microcontroller is given in section 4.3.2.

```
iDevice = 1;
iPort = 0;
iMode = 1;
iDir = 2;

DIG_Prt_Config(iDevice, iPort, iMode, iDir);

DIG_Out_Prt(iDevice, iPort, Command);

DIG_In_Prt(iDevice, iPort, &State);
```

Figure 6-6: Commands to send/receive data from microprocessor

where	Command	=	data to write
	State	=	data received

6.2.4 IR Object Detectors

The Sharp GP2Y0A02YK object detectors provide an analogue output, refer section 2.10.5. The six outputs are connected to the LabPC+ which has an eight channel, 12 bit Analogue to Digital Converter (ADC). The voltage is converted and read from the LabPC+ using **`AI_VRead()`**.

```
AI_VRead(iDevice, Chan, Gain, &dVoltage);
```

Figure 6-7: Reading IR voltage from LabPC+

where	Chan	=	channel being read (0-7)
	Gain	=	channel gain (1 = -5V to 5 V range)
	dVoltage	=	voltage returned

The voltage received is converted into a distance using Equation 6-1. The constants have been determined by curve fitting an equation to the graph from the manufacturer's datasheet. Figure 6-8 shows the original graph and the polynomial used to estimate the distance.

$$D = \frac{0.008271 + 939.6V}{1 - 3.398V + 17.339V^2}$$ **Equation 6-1**

where	D	=	distance (cm)
	V	=	voltage

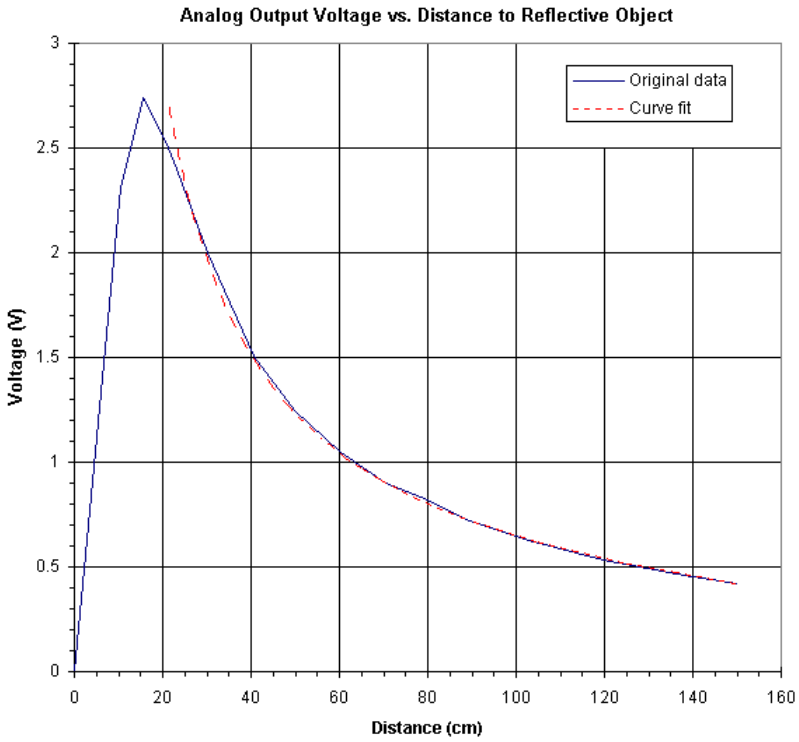


Figure 6-8: Curve fit used to convert voltage to distance for GP2Y0A02YK

6.2.5 Steering Wheel Position

The analogue position of the steering wheel is received through the LabPC+, and uses the same function as the IR Object Detectors (section 6.2.4) to read the voltage. The steering potentiometer is connected to a 0-5 V DC supply to ensure it is always within the operating range of the ADC.

6.2.6 Network Communication

Either through the 2.4 GHz wireless network card (refer section 2.10.1), or the onboard 100baseT Network Interface Card (NIC), a network interface is required to provide communication between the mechatrons. The 100baseT can be used during software development on a stationary mechatron to provide enhanced speed and accessibility using the existing wired network.

The network communication uses a stream socket. This is a connection based byte stream with the benefit of reliable and sequenced data transmission. An alternative is to use a datagram method, where the data flow is connectionless and is transmitted to any listening sockets. The data may not be sequenced or unduplicated, and is not guaranteed to be delivered.

Two halves are used to provide the network stream, a server and a client, refer Figure 6-9. The server (on each mechatron) is started when the software is first run. It listens for any incoming connections on a user defined port. The client (on the control base) generates a connection to the server. Data can be sent in either direction and is initiated using the `Send()` command. The complementary unit receives the data using a callback function. A callback function is activated when an event happens rather than having to explicitly call it from within the main code. This allows data to be read when it is received, rather than constantly polling to see when data is available.

The server program can accept connections from more than one client, allowing flexibility in future development. By not limiting the number of connections, data could be passed between each mechatron as well as to the base, using a Mesh rather than Star topology (refer Figure 6-10). This is achieved by dynamically creating the connection, and using a linked list to store all of the known connections.

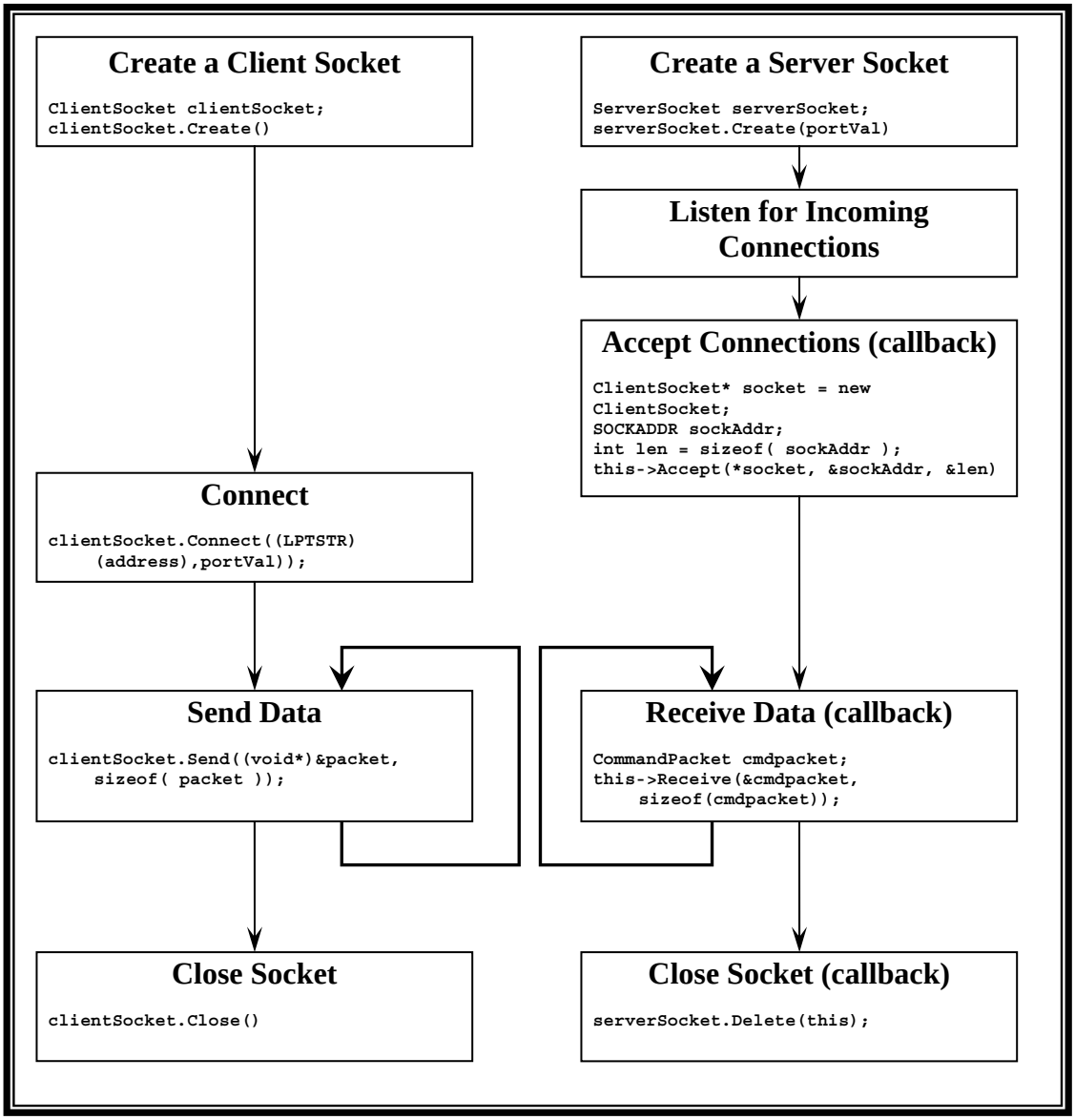


Figure 6-9: Flow diagram for network communication

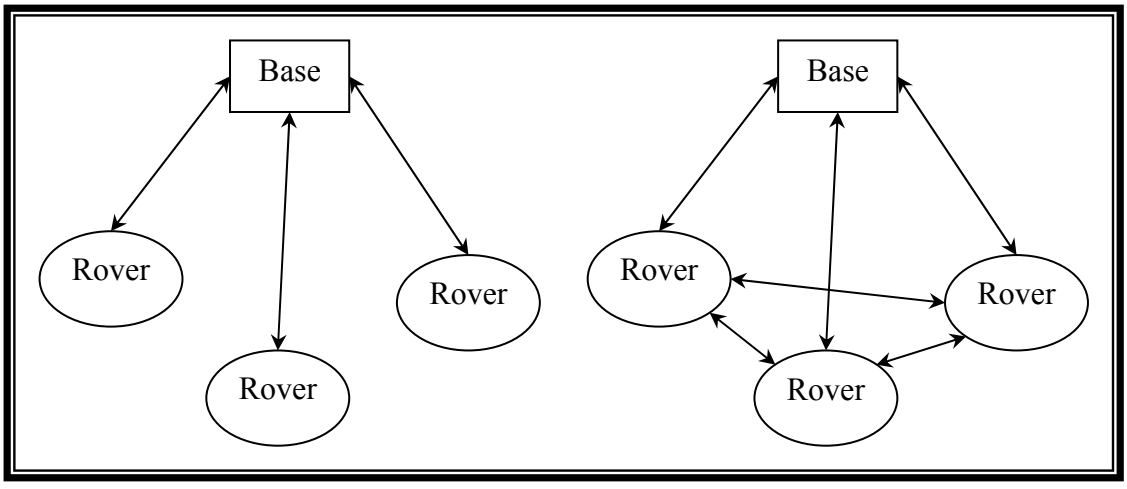


Figure 6-10: LEFT: Star network topology; RIGHT: Mesh network topology

6.2.7 Joystick

A joystick can provide simple control during development. This will allow the user to override the control system and remotely manoeuvre the mechatron when required, with minimal complexity.

When requested, the joystick is enabled using:

```
joySetCapture(hDlg, JOYSTICKID1, NULL, FALSE);
```

This call passes all messages from the joystick to the main dialog window. The main dialog then uses callback functions to deal with each message (Figure 6-11). Each time the joystick changes position, or the button is pressed/released, a function is called.

```
// Called when a joystick button is pressed
LRESULT CmasterDlg::OnJoystickButtonDown(WPARAM wParam, LPARAM lParam)
{
        if (joyEnabled)
                stopBut.SetState( TRUE );
        return 0;
}


// Called when a joystick button is released
LRESULT CmasterDlg::OnJoystickButtonUp(WPARAM wParam, LPARAM lParam)
{
        if (joyEnabled)
                stopBut.SetState( FALSE );
        return 0;
}


// Called when the joystick is moved
LRESULT CmasterDlg::OnJoystickMove(WPARAM wParam, LPARAM lParam)
{
        if (joyEnabled)
        {
                steerSlider.SetPos( (int)(((float)LOWORD(lParam) /
                        abs(joystick.jc.wXmax - joystick.jc.wXmin)) * 100) );
                speedSlider.SetPos( (int)(((float)HIWORD(lParam) /
                        abs(joystick.jc.wYmax - joystick.jc.wYmin)) * 100) );
        }
        return 0;
}
```

Figure 6-11: Callback functions from joystick input

6.2.8 User Interface

Two different user interfaces have been designed: one for use by the base that is controlling the mechatrons, the other on the mechatron itself.

6.2.8.1 Base User Interface

The base provides a client connection to each mechatron. It allows control of the mechatron's speed and direction through the use of two sliders, and an emergency stop button is provided. The mechatron will report back its current position, which is then displayed on screen. A joystick (either USB or analogue) can be used to control the sliders and hence the mechatron's movements. A screenshot of the base station is shown in Figure 6-12.

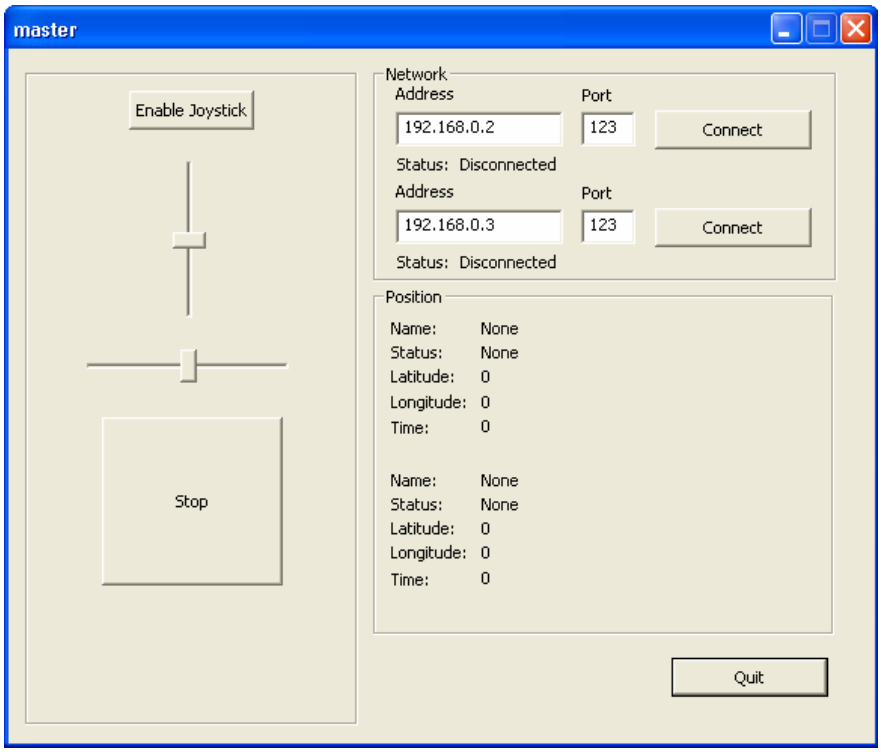


Figure 6-12: Base station user interface

The base has very little information passed back to it. This has purposely been done to encourage all data processing to be performed by each rover itself, with appropriate responses performed onboard. As an example, if an object is detected in front of the

mechatron - rather than reporting this to the base and waiting for a response, the robot should decelerate the motor itself to avoid collision.

6.2.8.2 Rover User Interface

To assist development, all data received by the sensors is displayed locally on the mechatron. Any settings or variables can be viewed and altered with minimal effort.

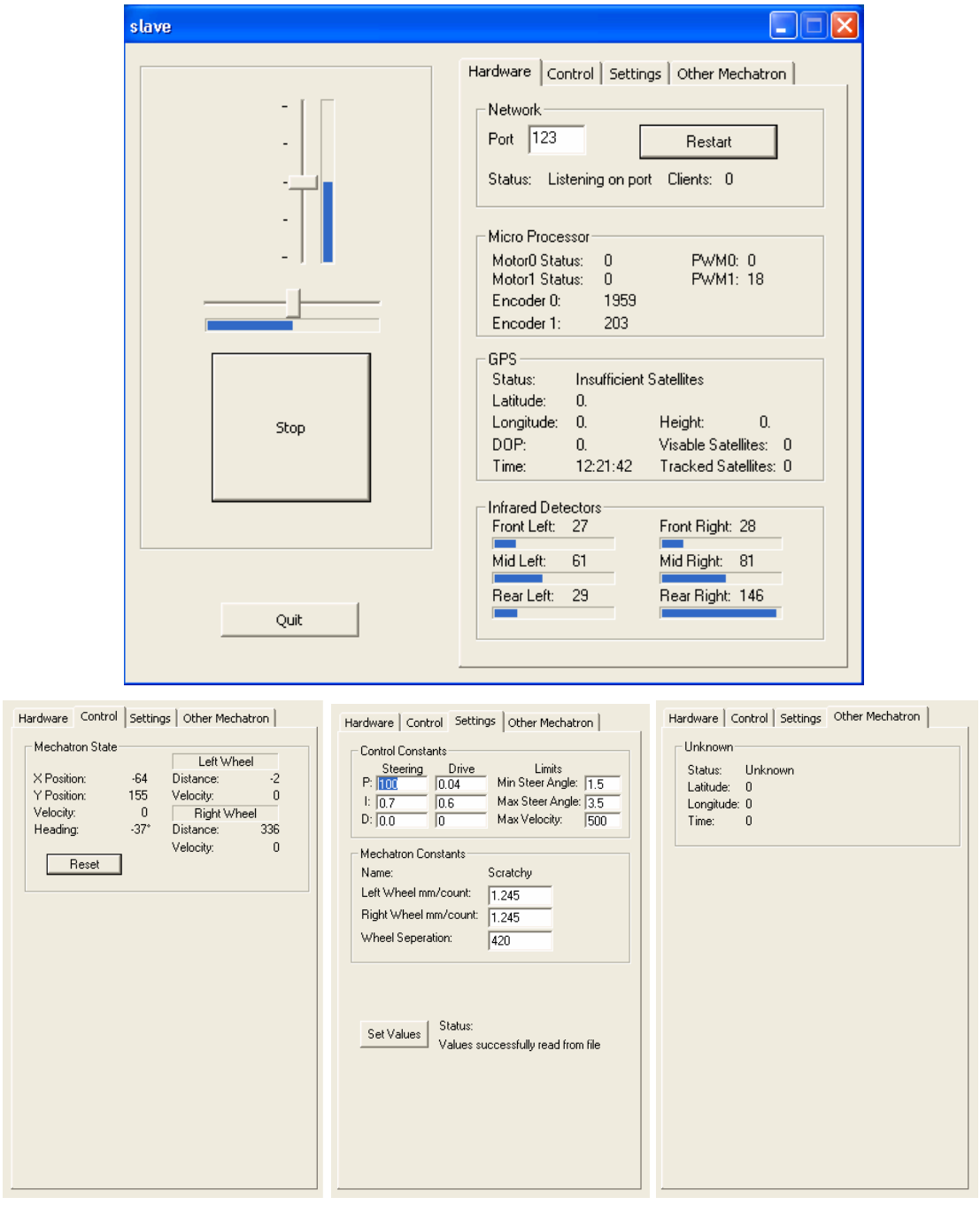


Figure 6-13: Rover user interface

The left-hand half of the dialog provides control to the motors. If commands are being received through the network, the sliders will move as the base sliders are moved. Otherwise the sliders can be moved by the user, providing a debugging interface. The progress bar next to each slider shows the mechatron's state, the current steering wheel angle, and the speed of the drive wheels. The stop button provides an emergency halt to both of the motors and can be used at any time.

The right-hand side of the dialog is broken into a number of tabs, providing information on different parts of the system.

The hardware tab provides network, microprocessor, GPS and IR object detector data. This tab is primarily used to provide low-level data from the sensors and to indicate any problems. The network port can be changed and restarted to listen on the new port. The status of the network port is shown, along with the number of clients connected. The microprocessor section shows the current PWM signals being sent from the PC, and the response received (indicating any errors). The raw encoder values received are shown, again indicating any errors. The GPS shows the status of the receiver, the 3D position, the number of satellites available and the time. The time has been included to allow for logging of data on separate robots for later comparison. Using progress bars, the IR object detectors show a graphical representation of the distance measured to any surrounding obstacles. A numerical representation is also shown with centimetre units.

The control tab is used to provide processed information to the user. The distance and speed of each drive wheel is shown, along with a calculated position and heading determined from this data. A large area of this tab has been provided for future development.

The settings tab provides on-the-fly alterations to the control system constants. This includes the PID constants for both the drive and steering systems, and physical constants including the values used to convert encoder counts into a distance. These are provided to allow the user to "tweak" the values for each individual mechatron.

Each mechatron provides GPS data back to the base. This data is relayed so that each unit knows its own position as well as the position of others around it.

6.3 SYSTEM

A number of software routines are required to link the system together and provide control of the mechatron. This involves making sense of the data received from the user and sensors, filtering sensor data, and controlling the motors.

6.3.1 Timing

All routines which require constant updates at a set rate are called by a timer. The timer is started by calling **`SetTimer(`*`nIDEvent, nElapse, NULL`*`)`**, and stopped using **`KillTimer(`*`nIDEvent`*`)`**, where:

nIDEvent	=	A value to identify the timer
nElapse	=	The timeout value in milliseconds

When a timer is activated, a message is set to the main dialog window which calls a callback function, **`OnTimer(nIDEvent)`**. A switch statement identifies which timer has been activated, and calls the routines required. Three timers are used; the first controls a safety timeout; the second polls sensors, controls the motor output and updates the display; the third polls the GPS.

Set with a timeout of 1000 ms, the first timer is used as a safety feature. When controlling the mechatron remotely through the base program, if a communication error occurs the robot will continue with the last speed/steering angle received. This could occur by exiting the base program without stopping the robot first, the wireless network failing by exceeding its range, or any other problem which will prevent the messages being sent correctly. The timer is stopped and restarted using the **`KillTimer()`** and **`SetTimer()`** functions every time a control message is successfully received over the network. If no messages are received within the time set, the **`OnTimer()`** callback function stops the drive motor, bringing the mechatron to a halt.

Note that this feature does not prevent the rover software from driving the motors directly during testing, and is only active when the base software is connected.

The second timer defines the system control loop time. Activated every 50 ms, the infrared detectors, steering wheel position, and shaft encoder values are read. The control systems (refer section 6.3.4) calculate the new PWM values for the motors, and send it to the microcontroller. The user interface (refer section 6.2.8.2) is updated. This routine can easily be performed by another (less frequent) timer to reduce the system load on the processor required by constantly redrawing to the screen; however during development it can be advantageous to visually detect transients in sensor data which can be concealed by a low update rate.

The third timer activates once every second. The GPS position/status/data message is read from the serial port buffer, and interpreted. If any network connections are established, the mechatron will report its GPS position to every connection.

6.3.2 IR Object Detector Filtering

The analogue signals from the IR object detectors (section 3.6) fluctuate due to electrical noise and ADC inaccuracies, especially when the output voltage is small due to no objects being detected. To remove the transients from the readings, a sliding median has been used. This was chosen as any brief transients will not have any effect on the resultant output, unlike a sliding mean. The response time of the detectors is slowed down by this process, so the number of samples should remain small.

The software used is based on vectors. Similar to an array, vectors can store a number of elements. They have the advantage of being able to change size dynamically, and are also optimised for operations such as adding/removing elements and sorting.

```
//Constants for curve fit
const double A = 0.008271, B = 939.6, C = -3.398, D = 17.339;
double voltage;
const int number_of_samples = 10;

using namespace std ;
typedef vector<int> INTVECTOR;

static INTVECTOR IRFrontLeftVector;
INTVECTOR SortVector;

if(IRFrontLeftVector.size() > number_of_samples){
        IRFrontLeftVector.erase(IRFrontLeftVector.begin());
}

voltage = DAQ.ReadADC(4,1);
IRFrontLeftVector.push_back((A + B*voltage) / (1 + C*voltage + D*voltage*voltage));

SortVector = IRFrontLeftVector;
sort(SortVector.begin(), &SortVector[SortVector.size()]);
IRFrontLeft = SortVector.at(SortVector.size()/2);
```

Figure 6-14: Sample code of sliding median used by IR object detectors

Figure 6-14 shows the code used to generate the sliding median. If the number of samples is larger than the value stored in **`number_of_samples`**, the oldest value is removed from the vector by erasing (**`.erase()`**) the first element (**`.begin()`**). The new voltage is read by the LabPC+ DAQ card (refer section 6.2.4), and converted to a distance using a curve fit equation (refer section 3.6). The new distance reading is stored onto the end of the vector using the **`.push_back()`** command.

The vector is copied into another temporary vector, on which the **`sort()`** function is called, sorting the elements into ascending order. The median value is now in the centre of the list, retrieved by the command **`SortVector.at(SortVector.size()/2);`**, where **`.at()`** returns the value in the stored and **`.size()`** returns the size of the vector.

All references to elements within the vector, or the vector size, are dynamic rather than hard-coded. This allows the number of samples to be altered to increase either the response time or the filtering level. This code also has an advantage that during start-up the output values are accurate – an array would require all of the elements to be filled before the output correctly reflects the input.

6.3.3 Shaft Encoders

The shaft encoders are interfaced through the microcontroller which returns a 14 bit count value (section 4.3.2). The software must account for counter overflow/underflow every time the wheel travels over twenty metres:

2^{14} counts × 1.247 counts/mm (refer Equation 3-4) = 20431 mm **Equation 6-2**

Due to the unshielded nature of the parallel communication link between the PC and microcontroller, noise generated from surrounding motors can corrupt the data being received. This occurrence is infrequent, approximately one per thousand data transmissions, but causes problems if not dealt with adequately. The PC can detect an incidence when it occurs by checking the microcontroller response with expected response (Table 4-1). The problem can be corrected by repeating the transmission, returning to normal operation with an insignificant delay. An incorrect value sent to the microcontroller has minor consequence as:

i) the command is not understood and therefore not actioned, or
ii) the motor will accelerate within the acceleration limit until the next instruction correctly sets it. Due to the slow response time of the motor, the effect will be minimal.

Receiving shaft encoder values from the microprocessor does not use the negated response of the other commands (Table 4-1), and therefore is susceptible to unrecognised errors. Improved communication error checking, such as a checksum, could remove this problem but at the expense of increased overhead and reduced speed. Instead of increasing the microcontroller interface complexity, a windowing technique is used by the PC.

By knowing the maximum speed of the mechatron, the conversion from encoder counter to distance, and the sampling rate of the encoders, a maximum allowable change can be calculated. Any extreme changes of the encoder value outside this limit can be assumed to be caused by a communication error, and therefore ignored.

An example of this is:

Maximum speed:	2 m/s
Encoder to distance constant:	1.247 mm per count
Sampling period:	50 ms

A maximum allowable encoder change per sample would then be:

$$\frac{2\ m/s}{1.247\ mm/count} \times 50\ ms = 80\ counts/sample \qquad \textbf{Equation 6-3}$$

This causes the encoder value to remain constant during the period of the error, affecting the current velocity and heading measurements. Multiple errors in succession can also put a valid encoder value outside the allowable window.

The code shown in Figure 6-15 uses the preceding encoder values to estimate the new value, minimising disruption to the control (section 6.3.4) and navigation (section 6.3.7) systems. Vectors are used to store the encoder values (see section 6.2.4). If the change is outside the allowable window, the code checks for overflow and subtracts 2^{14} (counter resolution) to correct the problem. Underflow is similarly corrected by adding 2^{14}. If the problem still exists, the new value is deleted and replaced by:

$$\text{value}(t\text{-}1) + (\text{value}(t\text{-}1) - \text{value}(t\text{-}2)) \qquad \textbf{Equation 6-4}$$

where: t = time

During the unknown period, constant velocity is assumed.

```
int EncoderChange[2];
const int encoder_window = 200;

using namespace std ;
typedef vector<int> INTVECTOR;
static INTVECTOR Encoder0Vector;

Encoder0Vector.push_back(DAQ.Encoder0(FALSE));

EncoderChange[0] =Encoder0Vector.back()-Encoder0Vector.at(Encoder0Vector.size()-2);

if(abs(EncoderChange[0]) > encoder_window)
{
        if((Encoder0Vector.back() > (16384-encoder_window/2)) &&
                (Encoder0Vector.at(Encoder0Vector.size()-2) < (encoder_window/2)))
        {
                EncoderChange[0] -= 16384;
        }
        else if ((Encoder0Vector.back() < (encoder_window/2)) &&
           (Encoder0Vector.at(Encoder0Vector.size()-2) > (16384-encoder_window/2)))
        {
                EncoderChange[0] += 16384;
        }
        else
        {
                Encoder0Vector.pop_back();
                EncoderChange[0] = Encoder0Vector.back() -
                        Encoder0Vector.at(Encoder0Vector.size()-2);

        Encoder0Vector.push_back(Encoder0Vector.back()+EncoderChange[0]);

        if(Encoder0Vector.back() > 16384)
                Encoder0Vector.back() -= 16384;
        else if(Encoder0Vector.back() < 0)
                Encoder0Vector.back() += 16384;
        }
}
```

Figure 6-15: Filtering encoder values

This correction method ensures any transient readings cause negligible disturbance to the system.

6.3.4 PID Control System

A number of different control systems can be implemented in software. These include:

- Fuzzy logic
- Proportional-Integral-Derivative (PID)
- Neural network
- Fuzzy-neuro

Fuzzy logic generalises the system inputs into sets. The controller uses a list of "rules" to determine the action to take. This action is finally converted into a response which controls the output variable.

Proportional-Integral-Derivative (PID) controllers are based on mathematical models in which the control system is described using one or more differential equations that define the system response to its inputs.

Neural networks use a number of nodes to process data simultaneously, combined using weightings to determine the output. The system can "learn" by adapting the weightings from each node to reduce the error, learning from past patterns. This can be combined with fuzzy control to produce a fuzzy-neuro control system.

A PID system is chosen to control the motors as it is a well known and documented classical control system which performs well for the required application. Easily adjusted, the control system can provide a quick response, with little instability or steady state error. As discussed in section 4.2, proportional control alone is inadequate for this application, so a Proportional-Integral-Derivative (PID) system will be used. Note that as the system is software driven, it can be reduced to PI control by zeroing the derivative constant if required.

A proportional controller will generate an output which is proportional to the error as mentioned in section 2.6.2. The output will always have a steady-state error as the output will never match the (non-zero) target. A small proportional constant will make the response slow and the steady-state error large. By increasing the constant, the response will become quicker and the steady-state error will reduce, however; the output will eventually become unstable, oscillating about the target.

Integral control provides a method to remove the steady-state error, effectively summing the past error values until the output matches the target. At this point the input to the integrator will be zero, therefore the integral will stop changing and the output will remain constant (assuming both the target and disturbance inputs remain static). The integral term will slow the response of the output during sudden input changes, and will also increase output overshoot.

Derivative control is used to reduce the overshoot, and improve the transient response of the system. It can make the system more stable, however it must be used with caution as any noise in the error signal is amplified [20].

Mathematically the terms can be written as follows:

proportional control: $u(t) = Ke(t)$ **Equation 6-5**

integral control: $u(t) = \frac{K}{T_I}\int_0^t e(\eta)d\eta$ **Equation 6-6**

and derivative control: $u(t) = KT_D\dot{e}(t)$ **Equation 6-7**

where: u = output control variable
e = tracking error (target – current output)
K = proportional gain
T_I = integral time
T_D = derivative time

To implement these equations in a digital computer, they are approximated to an algebraic equation. Combining the three equations and using Euler's method gives:

$$u(k) = u(k-1) + K\left[\left(1 + \frac{T}{T_I} + \frac{T_D}{T}\right)e(k) - \left(1 + 2\frac{T_D}{T}\right)e(k-1) + \frac{T_D}{T}e(k-2)\right]$$ **Equation 6-8[21]**

where: T = sampling interval

This equation assumes rectangular integration. Also note that the integral term is inverted, so a small T_I constant will have a large integral effect.

This equation is used by both the steering and drive motors to control the angular position and velocity respectively. The user interface allows the constants to be modified to tune the response of the system. As previously stated; the derivative constant can be zeroed if PI control is preferred due to noise being amplified by the derivative term.

6.3.5 Network Packet Identification

To allow flexibility of data transferred through the network, a packet system has been used. This simplifies the sending and receiving of data, and has been configured to allow different data types/lengths to be sent.

Two different packets have been created; the first is a command to a rover containing speed and steering information, the second is position information sent to/from each unit. The packet structure is shown in Figure 6-16. The first entity is a variable which contains a constant number used to identify each packet type; a command is given a value of one, a position a value of two. To send data, the struct variable is created; the values are stored in it appropriately, and the packet is sent using the **`Send()`** function discussed in section 6.2.6.

```
struct CommandPacket
{
        char type;
        char x, y;
        bool but1;
};

struct DataPacket
{
        char type;
        char name[10];
        __int16 GPSStatus;
        double latitude;
        double longitude;
        char time[9];

};
```

Figure 6-16: Network data packet structure

The receive callback function (refer section 6.2.6) must now determine the type of data being received, as the contained variable types and length will vary. This is performed by "peeking" at the first byte of data being received without removing it from the input buffer, refer Figure 6-17. A switch statement uses this byte to determine the correct type/length of data to receive and which routine to call to process the new instruction. Note that the "peeking" of data must not remove it from the incoming buffer; otherwise the subsequent read will fail as the byte is part of the packet. The alternative solution of sending the type, followed by the data (as two

separate entities) is not desirable as it increases the processor workload, and could lead to synchronisation issues.

```
char packettype;

this->Receive(&packettype, sizeof( packettype ), MSG_PEEK);

switch (packettype){
        case 1:
                CommandPacket cmdpacket;
                this->Receive(&cmdpacket, sizeof( cmdpacket ));

                slaveDlg.OnClientSocketRecvCommand( &cmdpacket );
                break;

        case 2:
                DataPacket datapacket;
                this->Receive(&datapacket, sizeof( datapacket ));

                slaveDlg.OnClientSocketRecvData( &datapacket );
                break;
        }
```

Figure 6-17: Receving multiple types of data

The configuration used is extremely flexible, allowing up to 255 different message/data types to be sent through one connection. Each data type is also very flexible, and can easily be changed by modifying the packet structure, adding or removing variables.

6.3.6 File I/O

File input and output is used to store variables unique to each mechatron. When the program is started the variables used in the rover's settings tab, refer section 6.2.8.2, are read from a text file. The values can be manually changed during operation and therefore are stored back into the text file on exit of the program so that they are available for subsequent runs.

This is much more desirable than having to change hard-coded values and recompile the software each time, simplifying the task of adjusting variables such as the distance constant used by the shaft encoders; which may vary as the tyre pressure decreases over time.

A text file has been chosen over a binary file so that the user can directly make changes to the contents without having to run the software. The file contains descriptors of each value; illustrated in Figure 6-18, allowing the file to be modified with ease.

```
/* File to store all constants on exit of program */

Steering Proportional Constant:      10
Steering Integral Constant:          0.7
Steering Derivative Constant:        0.0

Drive Proportional Constant:         0.04
Drive Integral Constant:             0.6
Drive Derivative Constant:           0.0

Steering Minimum Angle Limit:        1.5
Steering Maximum Angle Limit:        3.5

Maximum Velocity Limit:              500

Mechatron Name:                      Itchy
Left Wheel mm per count:             1.245
Right Wheel mm per count:            1.245
Wheel Seperation:                    420
```

Figure 6-18: Settings.txt file used to store mechatron settings

If the file does not exist or is corrupt, the program will default to values hard-coded into the software, and report an error to the user interface.

6.3.7 Dead Reckoning

The data from the shaft encoders can be used to calculate position and heading information using dead reckoning. Dead reckoning is the process of using a known position, combined with heading, and distance (or speed and time) measurements to calculate a new position. By iterating the calculations using the previous results, the position can be constantly updated providing a continuously known position. As dead reckoning provides a relative change in position, the starting position must be known to achieve an absolute position. As any errors are accumulative, the accuracy of the absolute position decreases over time.

The software uses two measurements on each iteration of the control loop to update the position; the change in distance, and the change in heading.

The change in distance is determined by taking the average of the distance travelled by each of the drive wheels since the last reading. This calculates the distance travelled by the centre of the mechatron.

The change in heading is approximated using trigonometry. A triangle is generated using the known wheel separation dimension (physical chassis dimension), and the difference between the distances travelled by each drive wheel during the delay since the last reading, refer Figure 6-19.

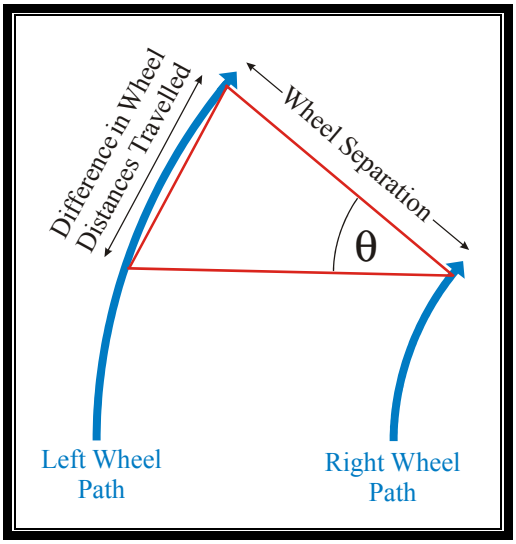


Figure 6-19: Trigonometry approximates the heading change during a turn

The angle is accumulated every iteration to maintain a continuously updated knowledge of the absolute heading. The angle is only approximated as: i) the outside edge of the triangle is assumed to be straight, but is actually an arc (refer Figure 6-19); ii) the triangle is assumed to have a right-angled corner. Both of these limitations become negligible as the angle decreases. To reduce the angle, the frequency of the measurements can be increased. The system limits the maximum frequency due to the resolution of the system timer, refer section 6.3.1, and also through the communication interface between the microcontroller and PC. The sampling interval has been chosen to be the same as the control loop period, 50 ms, but can be decreased if the performance is poor due to the approximation limitations.

The distance and heading information is converted to an X-Y coordinate system by accumulating of the changes each cycle:

$$\Delta X = \sin(\phi) \times \Delta D$$

$$\Delta Y = \cos(\phi) \times \Delta D$$

where: ϕ = absolute heading

ΔD = change in distance

The results of this method of navigation are discussed in section 7.1.2

7 TESTING AND CONCLUSION

7.1 EXPERIMENTAL TESTING

Before testing of the mechatron could commence, the mechanical, electronic and software parts of the project had to be completed and individually tested. The mechanical construction was tested by applying large amounts of weight to check the chassis strength and stability. The motor drivers and controller developed was tested on a high power resistive load, on free wheeling motors and finally on MARVIN. The software was verified by running tests on each individual part while the mechatron was stationary. An oscilloscope verified the I/O timing and analogue voltages to the DAQ card.

7.1.1 Mobility

The mechatron control system limits the velocity and the steering angle values. The steering wheel is capable of operation over a ±70° range on a high friction surface, allowing the mechatron to complete a 360° turn within a 1.5 m enclosure. During this manoeuvre, one drive wheel remains stationary while the mechatron sweeps around it.

With a ¾ duty cycle limit within the microcontroller code, the maximum velocity of the mechatron is 2 m/s. This limit is implemented to prevent damage to the mechatron and motor drive circuit if part of the system fails. If the shaft encoder belt broke, the control system would increase the power to the motor to account for the "stopped" wheel; the ¾ duty cycle limits the effect this would have before the user stops the mechatron. Similarly, if the mechatron drove into a solid object and could not move, the control system would increase the power to the motor. The current drawn through the power circuit during this time can damage it; the ¾ maximum duty cycle limits the damage caused by the continuous current overheating the MOSFETs.

7.1.2 Localisation

To determine the accuracy of both the GPS and dead-reckoning navigation techniques implemented, a mechatron was driven over a known path, illustrated in Figure 7-1. The path used was defined by the markings of a rectangular sports field of dimensions 55 m × 35 m.

Figure 7-1: Outdoor voyage over a known set path

The mechatron path started and ended at the same location so that any recorded data should form a closed loop when plotted. The reported position of each navigation method was logged to file every second. The results of multiple runs are shown in Figure 7-2 and Figure 7-3. A scale is shown in the top left corner of the first plot, allowing a degree to metre conversion to be performed. The latitude and longitude axes are scaled independently.

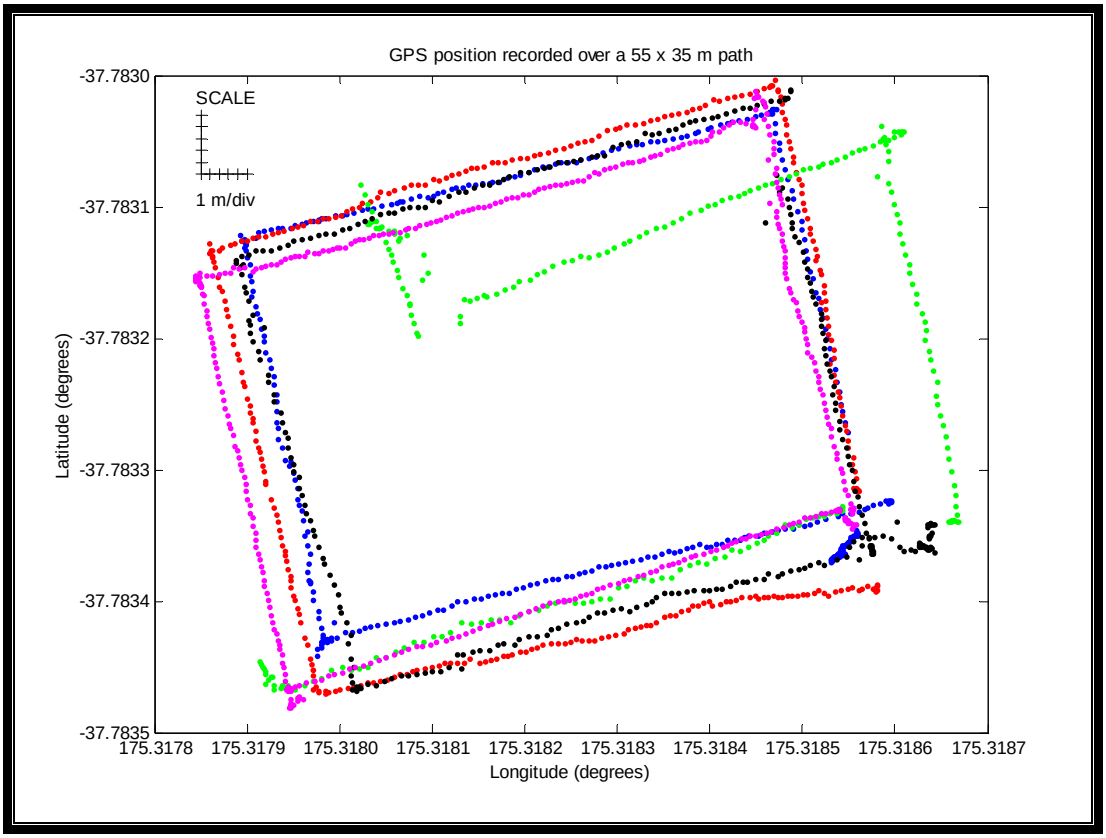


Figure 7-2: GPS Reported Position over a rectangular path

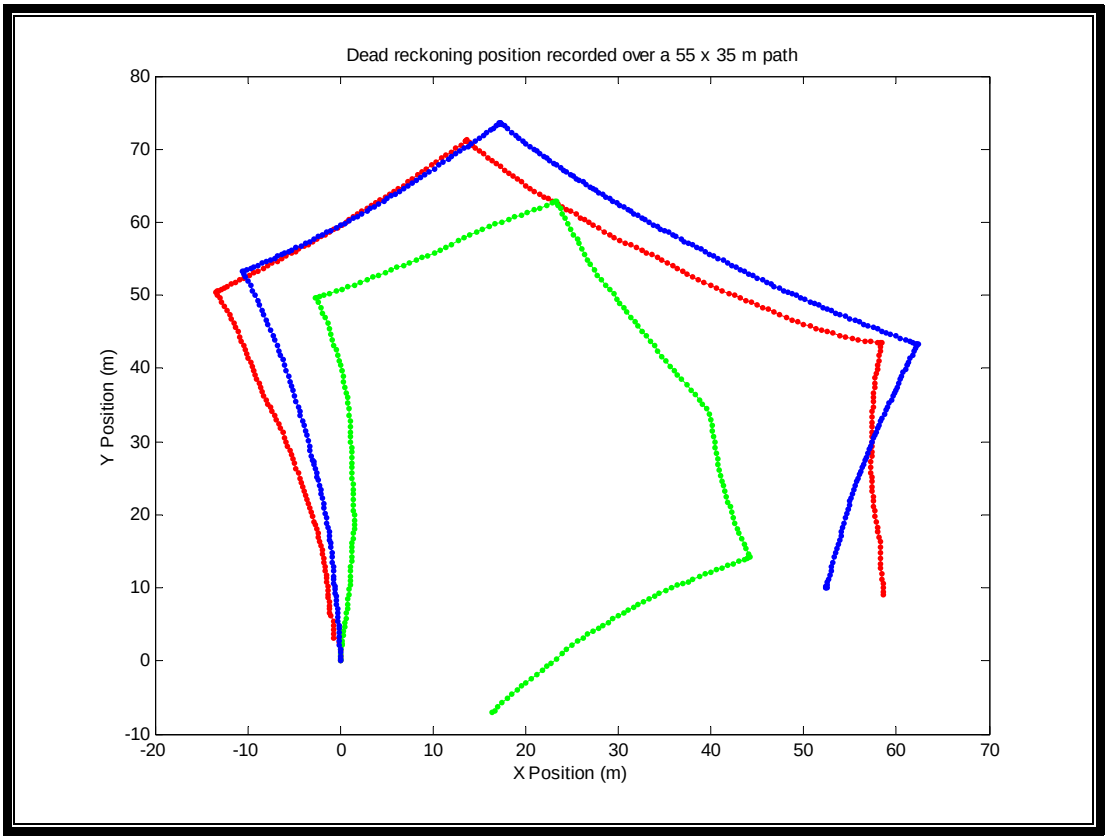


Figure 7-3: Dead reckoning position using shaft encoders

The GPS positioning shows an enclosed path around the perimeter of the field. The receiver was obstructed during the green plot, with the position unavailable during the obstruction time and a low accuracy position available during the "recovery" time.

Each individual run plots a rectangular shape. The rectangle is not always closed however, as the start and end positions are not aligned (bottom right hand corner of each plot in Figure 7-2). The start-to-end measured distance (the same position) was:

Blue:	7.81 m
Red:	8.85 m
Green:	10.27 m
Black:	6.33 m
Magenta:	0.94 m

This result shows that the position can rapidly drift with time, as each run was performed in 7-10 minutes. By superimposing multiple runs together, the position error drift over time is visible. Ignoring the green plot as the GPS status informs the user of a problem, the maximum error between known positions over multiple runs was 8.85 m. The loss of position information shown in the green plot implies that the mechatron must be capable of integrating multiple navigation methods to continue operation during GPS obstruction. The error is too large for the robots to operate in close vicinity of each other (<10 m) using a standard GPS position.

The dead reckoning technique using shaft encoders (Figure 7-3) provides accurate measurements over short distances (<5 m), but accumulated error limits the use over longer distances. The start-to-end distance of each run was:

Blue:	53.37 m
Red:	59.32 m
Green:	17.87 m

The green run shows a 'kink' in the third section of the path, contrary to the trend of consistently drifting left while travelling in a straight path. This abnormality is due to an error in the distance read from a shaft encoder, which has caused the heading

calculation to be incorrect. The software was since modified to filter out this problem (section 6.3.3).

While traversing in a straight line, the reported position drifts to the left, due to the incorrect calibration of the constant used to convert the encoder count to distance. Correcting for this error, the mechatron can calibrated to the current conditions:

Tyre pressure:	18 psi
Loading:	PC, Batteries, and electronic PCBs
Left Wheel:	1.245 mm/count
Right Wheel:	1.236 mm/count

However, preceding each testing session recalibration would be required if any variations in the mechatron's tyre pressures or weight loading have occurred. An uneven operating surface will also cause deviations in the heading resulting in inaccurate positioning, irrespective of the encoder constants.

Left Wheel	Right Wheel	Average	Error
180 149 mm	181 517 mm	180 833 mm	0.463 %
179 048 mm	182 515 mm	180 782 mm	0.434 %
180 216 mm	181 575 mm	180 896 mm	0.498 %
177 081 mm	179 651 mm	178 366 mm	0.908 %

Table 7-1: Shaft encoder based measurements over a 180 m path

Table 7-1 shows that the encoder measurements were accurate within 1% of the total distance travelled, using the path in Figure 7-3, despite the constants requiring further calibration as previously stated.

To increase the robustness of the dead reckoning navigation system an alternate method of sustaining the heading angle needs to be used. A magnetic compass or gyroscope could be used to provide this heading data, allowing a much larger tolerance on the error of the wheel constants while maintaining an accurate position.

The position reported using dead reckoning is relative not absolute, and so must be integrated into a system with other navigation techniques such as GPS to provide continuous positioning information over a long period of time. As the short range accuracy of the shaft encoders is very high; a weighted system should be used to combine different navigation systems. As an example; when the mechatron is stationary, the shaft encoders are the highest accuracy system so the GPS data should be averaged during this time to increase the precision of the absolute position. Due to time constraints, this has not yet been implemented.

7.2 IMPROVING LOCALISATION WITH DGPS

To determine any increase in position accuracy when using differential GPS (DGPS) techniques, as discussed in chapter 5, two M12 GPS receivers were used to report one position. Achieved by keeping the receivers stationary, the reported position was recorded over time with any variation therefore due to GPS error. The test was repeated using pseudo-range corrections and also a block shift method.

7.2.1 Comparison of DGPS Methods

Each method was executed over a six hour period to give a fair representation over continuously changing satellite coverage. Figure 7-4 shows the results from each test. The receiver remained stationary, so the points plotted should be dense, showing little variation over time. To graphically show the distribution of the reported position, the mean is shown with a star and the standard deviation shown as an ellipse over the data.

The first plot (top left) shows the variation of one receiver in a default mode of operation. Corrections are made for atmospheric conditions using data received in the GPS message and the output from the receiver is three dimensional. The standard deviation has been converted into metre units (rather than degrees) to aid comparison, and was 2.12 m and 3.40 m in the latitude and longitude directions respectively.

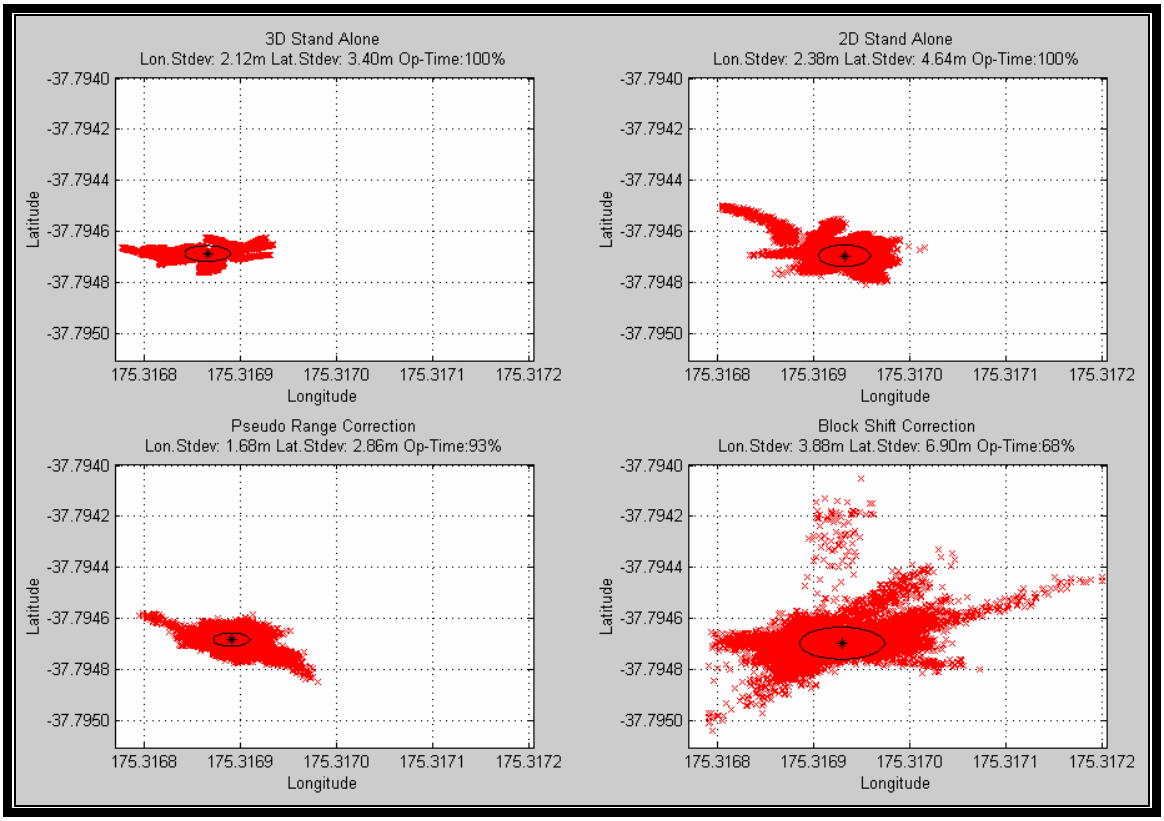


Figure 7-4: Comparison of different GPS methods on a stationary point

The 2D single-receiver method produced an unexpected result, shown in the top right of Figure 7-4. The position varied over a larger area than that of the 3D stand alone process, with standard deviations of 2.38 m and 4.64 m. This is probably due to a discrepancy between the height the receiver was set to, and the actual height.

The pseudo-range and block shift methods were executed on a 2D position, reducing the number of satellites required by the receiver. Comparison of these methods is therefore performed with the single-receiver providing a 2D position.

The pseudo-range correction method shown in the bottom left of Figure 7-4, and significantly reduced the variation of the 2D position. The standard deviations are approximately 30% lower than those of a single receiver, recorded at 1.68 m and 2.86 m. To provide continuously updated data as the satellites shift with respect to the earth, the correction messages were sent every ten seconds. The operational time is shown above each plot, and has reduced to 93% of the six hour period.

The corrections for the satellite ranges are sent in two serial messages to the GPS receiver and must be received back-to-back. The receiver is configured to send a position message back to the PC every second. To successfully apply the pseudo-range corrections, both messages must be sent within this one second period, as shown in Figure 7-5. If this is not done, the receiver will use half of the correction data, only applying pseudo-range corrections to some of the visible satellites. To overcome the problem, the corrections must be re-sent.

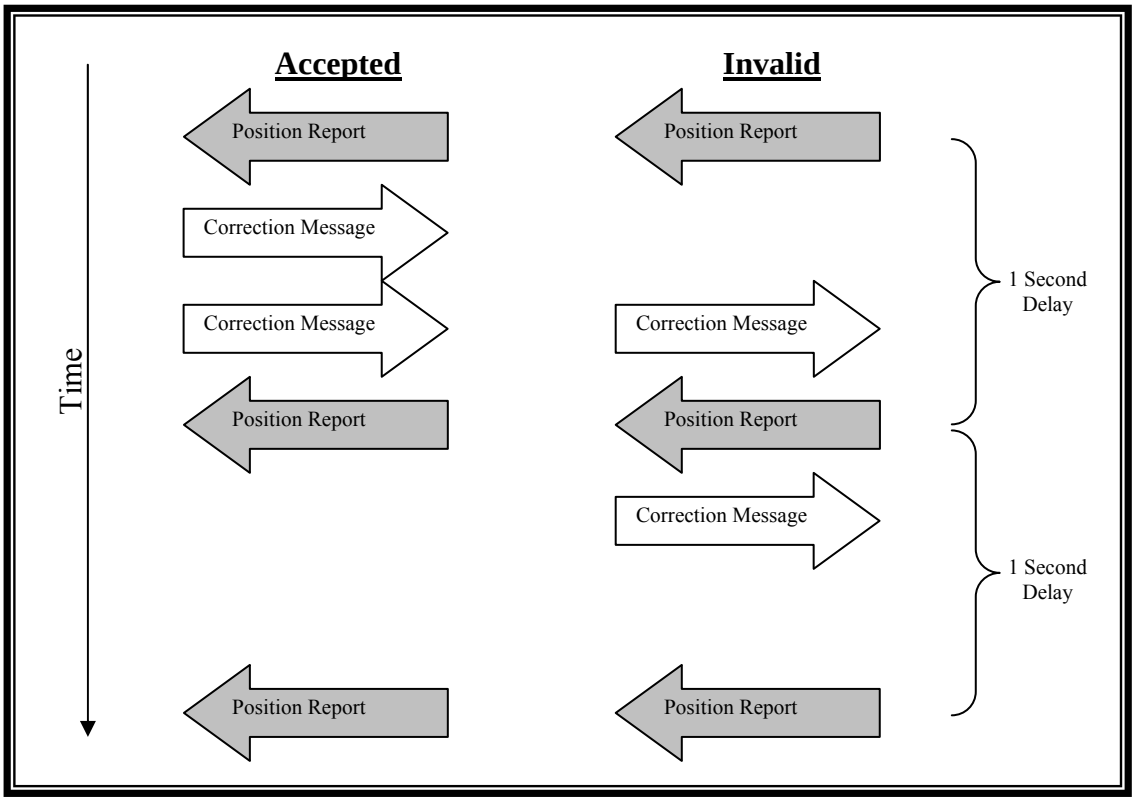


Figure 7-5: Pseudo-range correction message timing constraints

Another unexpected result was that the block shift method produced the most unreliable 2D positioning, shown in the bottom right of Figure 7-4. The standard deviation of the position was approximately 50% worse than that achieved by a single receiver, at 3.88 m and 6.90 m. This is possibly because the satellite selection used (section 5.5.2) limited the dilution of precision (DOP). The operational time over the 6 hour period was only 68%; for large periods of time no position was available due to the delay required for the M12 receiver to track a new satellite.

Since the receiver returns a DOP variable indicating the quality of the position reported, the mechatron could increase the weighting of the GPS position when the DOP is low. The effect of limiting the DOP was investigated on each method to provide a higher accuracy position. The DOP is shown for the six hour period in Figure 7-6. Only the operational time data is plotted, so the pseudo-range and block shift methods do not span the entire six hour period.

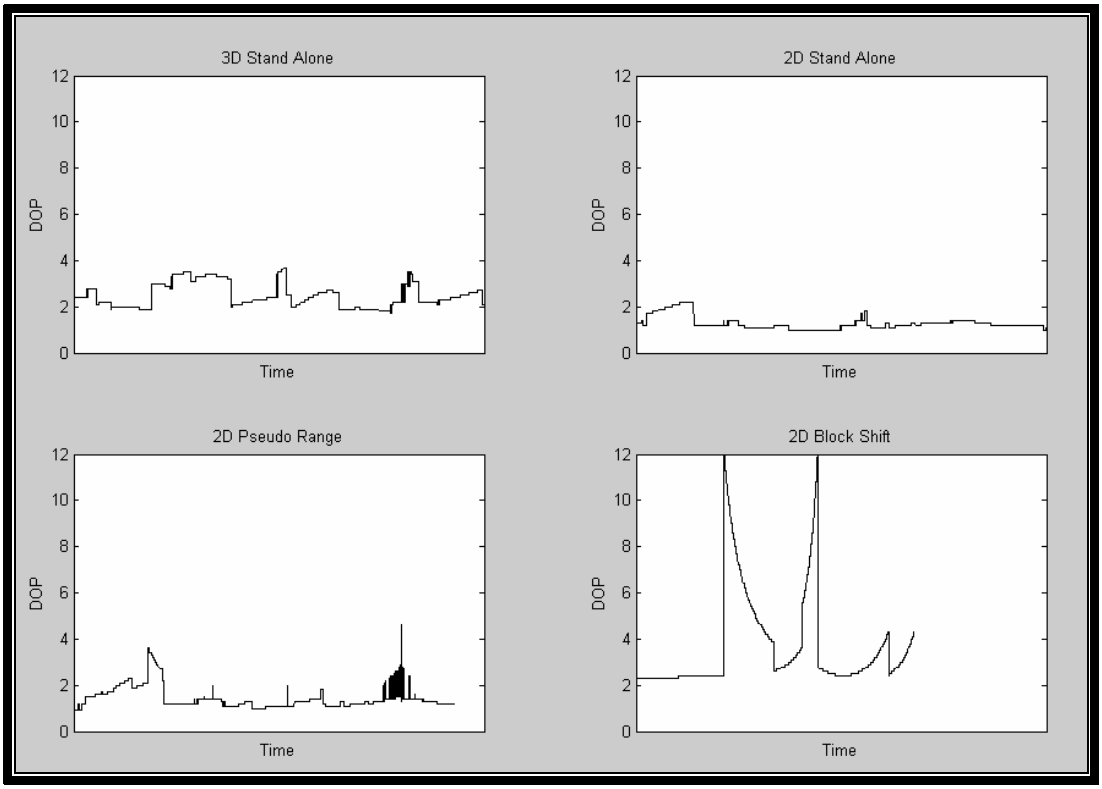


Figure 7-6: The level of DOP using different methods

The DOP results are as expected; the 2D stand alone and pseudo-range methods are lower than the 3D method as fewer satellites are required, and the block shift method had a high value due to the satellite selection (section 5.5.2).

Figure 7-7 shows the effect of limiting the DOP to a value of 2.45 (the highest DOP of the 2D stand alone method). The effects are significant for the 3D stand alone and block shift methods which experience high DOP values. The result is an increase in the precision of the position at the expense of the operating time. The block shift method accuracy increased the latitude standard deviation by 70% and the longitude by 20%, but reduced the operational time significantly. The pseudo-range method

accuracy was slightly increased with very little loss of operating time. For any method used, the mechatron software should use a DOP dependant weighting, as shown by these results.

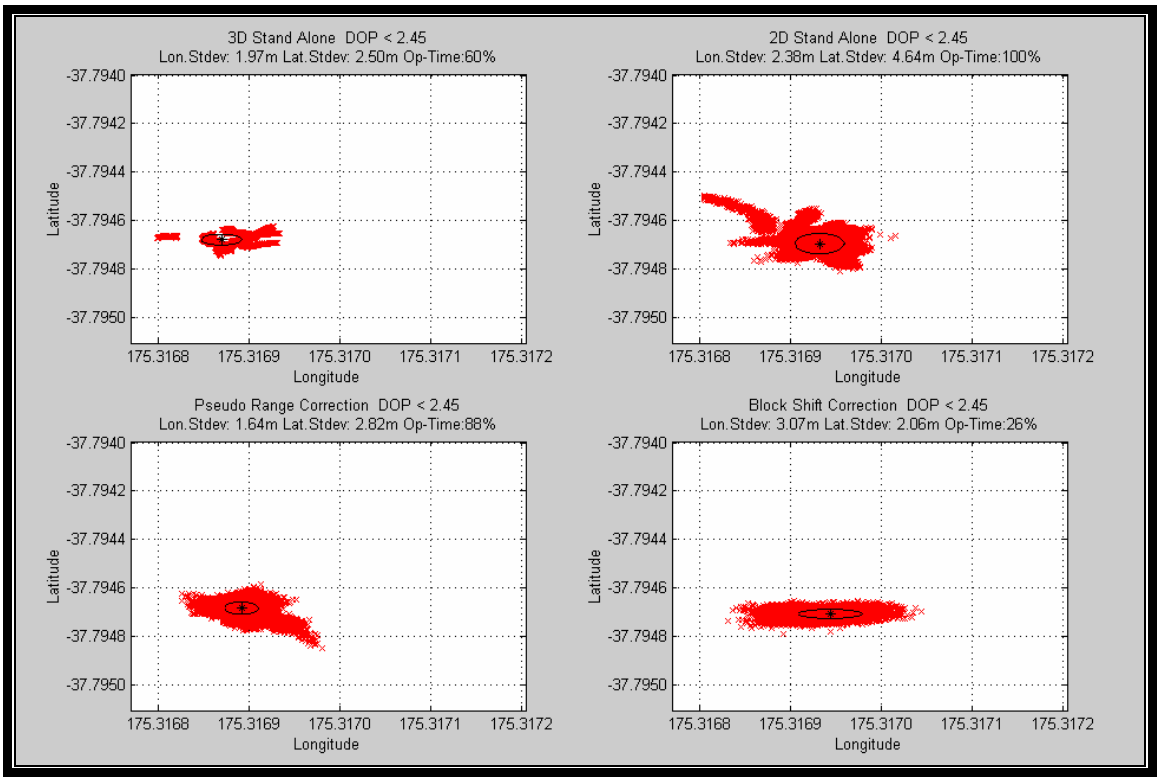


Figure 7-7: Comparison of different GPS methods on a stationary point, limiting the DOP

A consideration of the pseudo-range DGPS method is the correction update rate used. As time passes, the offsets become obsolete due to the constant change of moving satellites.

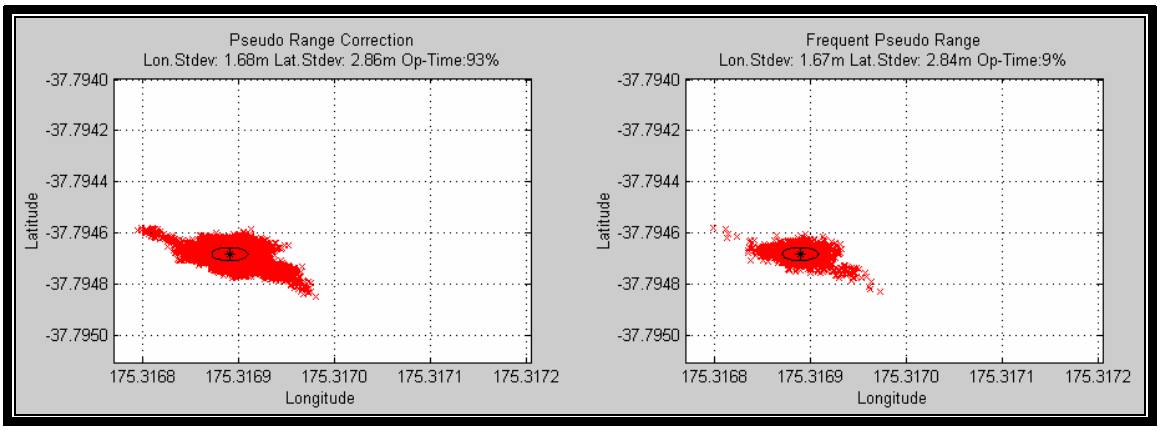


Figure 7-8: Increasing the frequency of corrections with the Pseudo-Range Correction Method

The approach used to determine the effect of the update rate was to only plot the position returned by the GPS receiver directly after sending the correction message. This was done for two reasons:

i) The position becomes improperly compensated if the two halves of the data are not received together as discussed above. If data is sent every second, the PC software is unable to verify the response messages as being correctly received.

ii) The data from the previous run can be used, covering a six hour period without repeating the experiment.

Figure 7-8 shows the result of updating the correction data directly before each GPS position is received. The longitude standard deviation decreased by 0.6%, and the latitude by 0.7%. The ten second update period used is therefore acceptable as the increased complexity and overhead generated by a one second period offers negligible improvement on the position accuracy.

7.2.2 DGPS Conclusion

The highest accuracy of the Motorola Oncore M12 is achieved using pseudo-range corrections, although this adds significant complexity compared to a stand-alone method. As the operational time needs to be continuous for this project, the maximum DOP cannot be limited as this causes periods of time with no position information being available. However, this could be used as a weighting factor with other localisation methods.

The expected standard deviation of the position when using pseudo-range corrections is approximately 1.68 m in a longitudinal direction and 2.86 m in a latitudinal direction. This is sufficient for the robot navigation while working large distances apart, but close range cooperative behaviour requires higher precision positioning. Further investigation into the position returned by the M12 receiver is not possible as the pseudo-range measurements are processed onboard the receiver and are not available to the user. Carrier phase capable receivers, and/or dual frequency receivers could provide the accuracy required, but the cost of this hardware is above the budget

of this project, greater than US$10 000. Other sensor types must be investigated to provide the required close range positioning information.

7.2.3 Infrared Object Detectors

The infrared object detectors are unable to operate in an outdoor environment. Even in overcast weather the detectors report a very short range to an obstacle in an open surrounding. It is assumed that the detector is being saturated with ambient light, and is therefore unable to detect the modulated signal. Figure 7-9 shows the detectors reading ranges of 19 – 32 cm without any surrounding objects. The receivers operate correctly in an indoor location. To overcome this problem, optical filters need to be investigated to allow light of wavelength 850 ± 70 nm [19] to be received while blocking other wavelengths. If the results are still not acceptable, another ranging method such as ultrasound could be used.

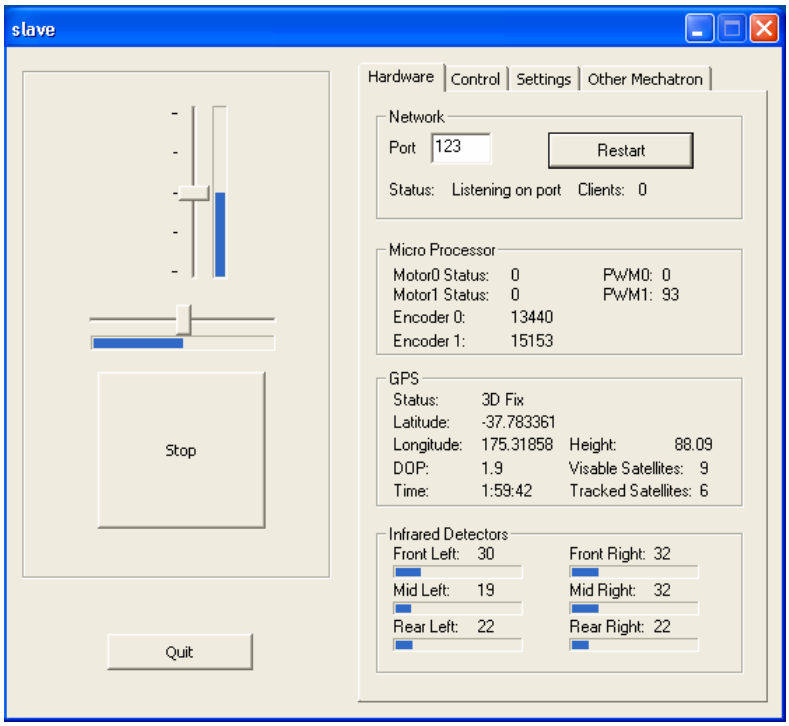


Figure 7-9: Rover user interface operating in an outdoor environment

7.3 CONCLUSION

7.3.1 Mechatron and system evaluation

The chassis is robust, and houses the required components. It is capable of supporting a very large payload (in excess of 80 kg) and provides very good stability due to the low height. The steering system allows high manoeuvrability, capable of turning within a 1.5 m enclosure.

The motors are controlled by a microcontroller interfaced to the PC. This has proven to be extremely beneficial due to the independence from the PC software, allowing safety features such as a timeout to stop the motors when the PC software stops responding. The motor driver circuits are capable of providing the output power required when operating the motors, tested up to 10 A continuous current.

The power drive circuits cannot handle the back emf generated by the motors during rapid changes of direction or very heavy acceleration. To overcome this problem, the acceleration is limited to reduce the stress on the power MOSFETs (the addition of voltage clamping components did not provide adequate protection). This implies that the power drive circuit must be used with the developed microcontroller, as a different method of PWM generation can over-stress the circuit.

The operator can directly control the mechatron from the onboard computer, through a virtual desktop using Windows XP terminal server, or using the remote base software written. The remote software allows a joystick to be used on the local machine to ease control of moving the mechatron.

The software is written in visual C++, providing a powerful and flexible base for future work to continue development. Data is received through the DAQ card, providing a high update rate to monitor the sensors for obstacles. The control loop period of 50 ms allows suitable motor control, providing a quick response to changing inputs. The data from the shaft encoders provides a resolution of 1.245 mm, and accuracy within 1% on outdoor terrain. The shaft encoder interface also provides

direction information; if the counters on the DAQ card were used, only distance measurements would be available and the direction would have to be predicted by other methods (such as the direction the motor is being driven). The heading calculated from the shaft encoder data alone is insufficient for localisation, but could be combined with a compass to provide a more accurate dead reckoning navigation system.

The motors are controlled using a PID control system. The drive motor uses a PI system (the derivative constant is zeroed) providing a satisfactory output response. The steering motor uses a PID system, with a very large derivative constant required to break the initial motor friction. The steering wheel suffered from a poor response when using the initial proportional controller built. This has been overcome by using the software PID control system.

The mechatrons distribute their position information through the wireless network, allowing the base and each mechatron continuously to know the current position of both of the robots. The software allows for future expansion, capable of accepting additional network connections if more robots are built as well as allowing different data to easily be sent through the wireless network using the foundation provided. The network system used confirms that all data sent is correctly received, ensuring integrity. However, if another system generates heavy network traffic (such as streaming video from the rover to the base) the data tends to accumulate and then send all at once. The effect of this is that the rover becomes unresponsive due to the reduced update rate. If future development requires heavy network traffic, a priority system could be introduced to overcome the limitation.

The rover user interface provides a large amount of information to assist with debugging, and calibrating the system. The physical constants can easily be adjusted to compensate for an increase in weight or change of tyre pressure, and also to further tune the control systems used to operate the motors.

The infrared object detectors, as manufactured, are not suitable for outdoor use. Outdoor operation may be possible by adding appropriate infrared optical filters to the receiver.

The GPS receiver is capable of providing an absolute position with standard deviations of 1.68 m and 2.86 m in the longitude and latitude directions, when using pseudo-range corrections. For applications where the mechatrons are distant from one another, this can be used for localisation.

7.3.2 Future Work

A manipulator needs to be built to allow the mechatrons to pick up objects. By constructing an arm with five or six degrees of freedom (independent joints) the mechatrons will be capable of picking up and carrying a range of different objects. The high number of movements available will provide the capability to traverse uneven terrain and turn tight corners under load, while minimising the arm stress.

The software written works under restricted test conditions, but exception/error handling needs to be increased to provide a reliable system as different conditions are met. As an example; if the serial port is in use by another device (such as the programming software for the microcontroller) the program will fail to run rather than disabling the GPS receiver.

The infrared detectors need to be modified to filter ambient light for outdoor use. If this is not possible, an alternate system such as ultrasound should replace the current IR detectors.

The addition of a compass would allow dead reckoning localisation to be performed with increased accuracy. The effect of using a magnetic compass in near vicinity of two electric motors needs to be investigated before this can be purchased. If a magnetic compass is not suitable, further investigation into alternate methods of measuring the heading needs to be done. This could include the use of an optical gyroscope to indicate changes in the heading angle.

Development of outdoor navigation, mapping and localisation is required, operating with an absolute position. The mechatrons should collaborate to produce a map of the

environment, using data collected from multiple robots in the same surrounding. An autonomous control system is required to provide the capability for each mechatron to navigate to a target position.

Cooperative behaviour should be investigated with respect to the control system. Two main variations of behaviour are possible; a master controller which instructs all of the robots' movements and tasks, making all decisions; or a distributed system where each unit is "smart", deciding its own tasks and collaborating with surrounding units.

Cooperative behaviour between the two mechatrons can be implemented once these other advancements are made to the project. The robots will eventually be capable of autonomously navigating to an object, each collaboratively selecting an end, before picking up and carrying the object to a new target location.

7.3.3 Summary

Two tricycle based robots have been built to investigate cooperative behaviour at the University of Waikato. All of the project objectives have been met, and the software written has exceeded the project requirements. The robots have been designed with the capability of eventually being fully autonomous, containing an onboard power source and processor.

The tricycle chassis is made from steel, with a 24 V DC 400 W motor driving the front wheels, and a 24 V DC wiper motor turning the steering wheel. Driven with the steering at the rear (like a forklift), each mechatron can turn within a 1.5 m enclosure, and has a top speed of 2 m/s. The chassis contains a power supply, two 12 V automotive starter batteries in series, and an onboard processor. The CPU is a desktop motherboard with an 800 MHz Celeron processor. An industrial ATX power supply provides the voltage rails required by the PC, operating from an input range of 18 – 32 V DC. A custom built case encloses the PC, hard drive, and ATX power supply. Communication is provided using a Netgear 301 wireless network card, using a standard 802.11b protocol. Data input and output is achieved using a LabPC+ data acquisition card manufactured by National Instruments.

Each mechatron has an onboard Motorola Oncore M12 GPS receiver to provide position information. An absolute position can be achieved with a standard deviation of 1.68 m in the longitude direction, and 2.86 m in the latitude direction. Shaft encoders are connected to the two driving wheels to make distance measurements, and also to provide velocity information to the system. The shaft encoders are interfaced to provide a measurement resolution of 1.247 mm, with an accuracy of ±1% when traversing an outdoor grass surface. Infrared detectors are incorporated to determine the presence of obstacles up to 1.5 m, but require modification to be capable of operating in an outdoor environment.

Power circuits have been developed to run the motors, capable of driving the 24 V DC 400 W motor. Using an H-Bridge MOSFET configuration, the motor direction can be changed, and the speed of the motor is precisely controlled using a PWM signal. The circuit allows for current and temperature monitoring; the controller can reduce the power or stop the motor under different problem conditions.

A Phillips P89C51RC2 microcontroller has been used to control the motors, and store the shaft encoder values. This allows a fail safe system to operate; if the PC stops responding or the code is exited, the microcontroller will timeout and shut off the motors. The maximum duty cycle and acceleration can also be set. This is used to limit the back emf from the motor when changing direction or accelerating heavily. The direction of rotation of the shaft encoders is determined by the microcontroller, providing direction and distance measurements to the control system.

Software interfaces have been written for all of the different communication protocols. This ranges from serial, parallel and wireless radio data transmission, to analogue sensors and a user interface. The sensor data is continuously updated at a 20 Hz rate, and PID control systems are implemented to operate the motors. A separate program has been written to act as a base station, sending instructions to the mechatrons remotely. A joystick driver in the base software allows ease of control during development.

The rover software written reports its position back to the base station. The base then transmits the data to the other mechatron when applicable. The result of this is that each mechatron and the base are constantly aware of each robot's position. The communication software written allows any data to be sent, such as the mechatron's intentions, to allow future development.

The pair of robots built provides an ideal platform for future research into autonomous cooperative behaviour by the Mechatronics Group at the University of Waikato.

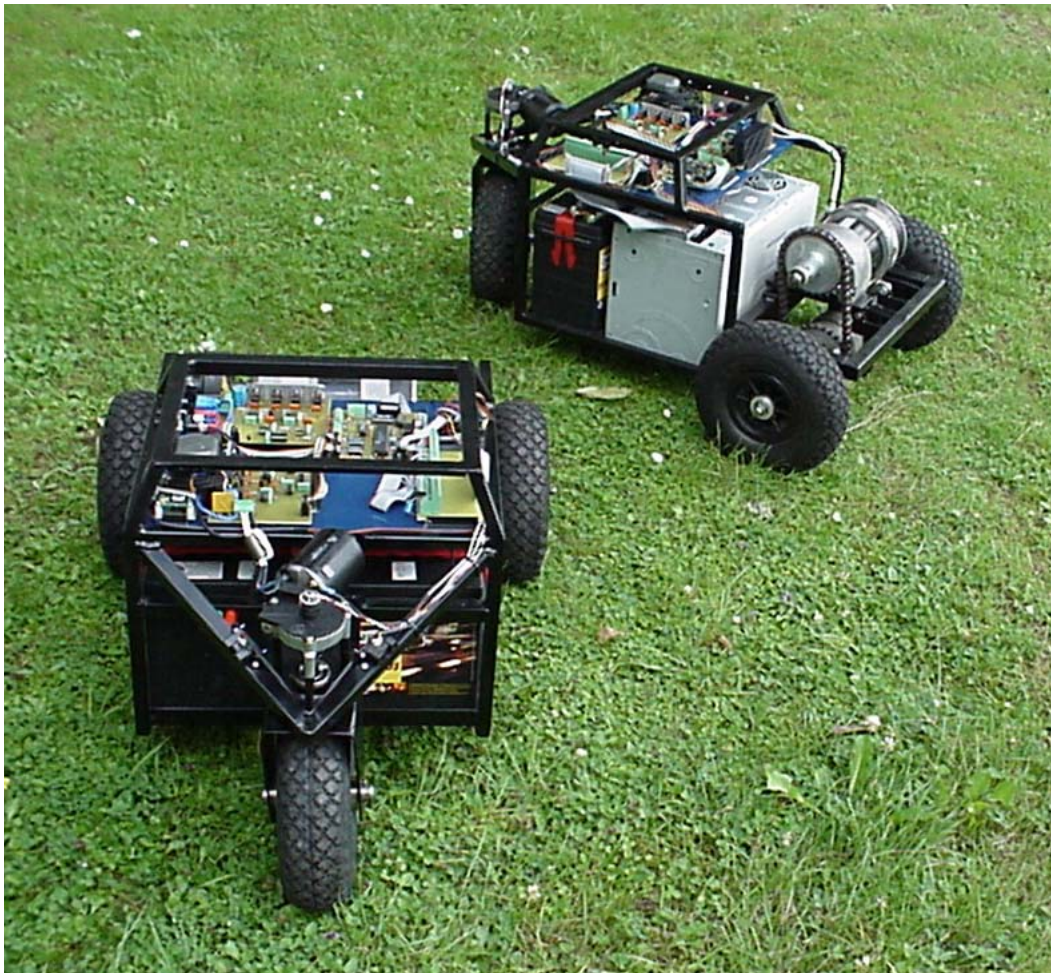
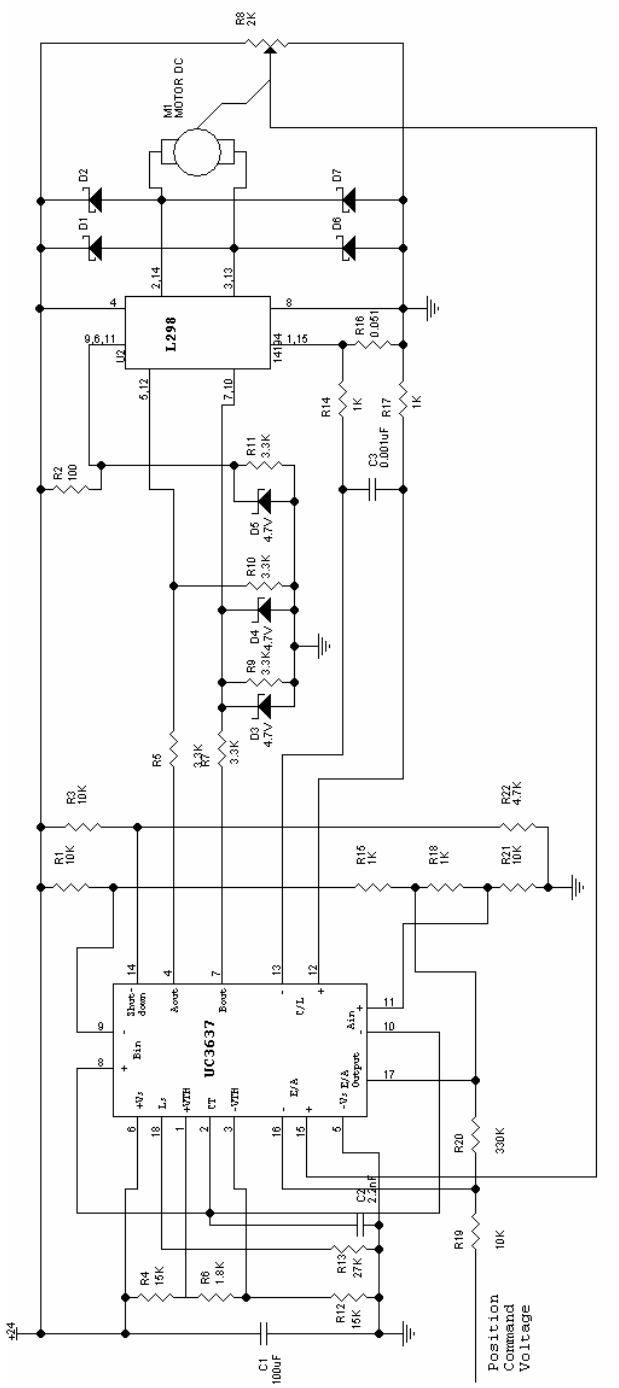


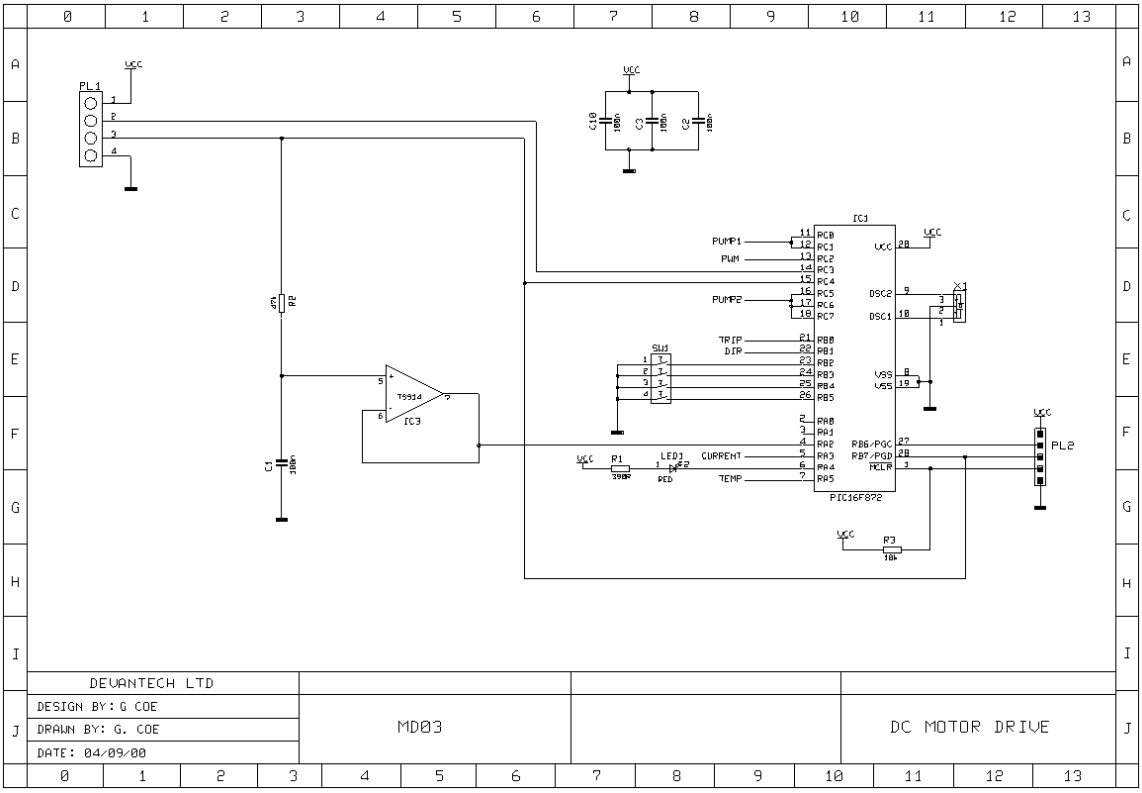
Figure 7-10: Completed mobile platforms for future development of cooperative behaviour at the University of Waikato

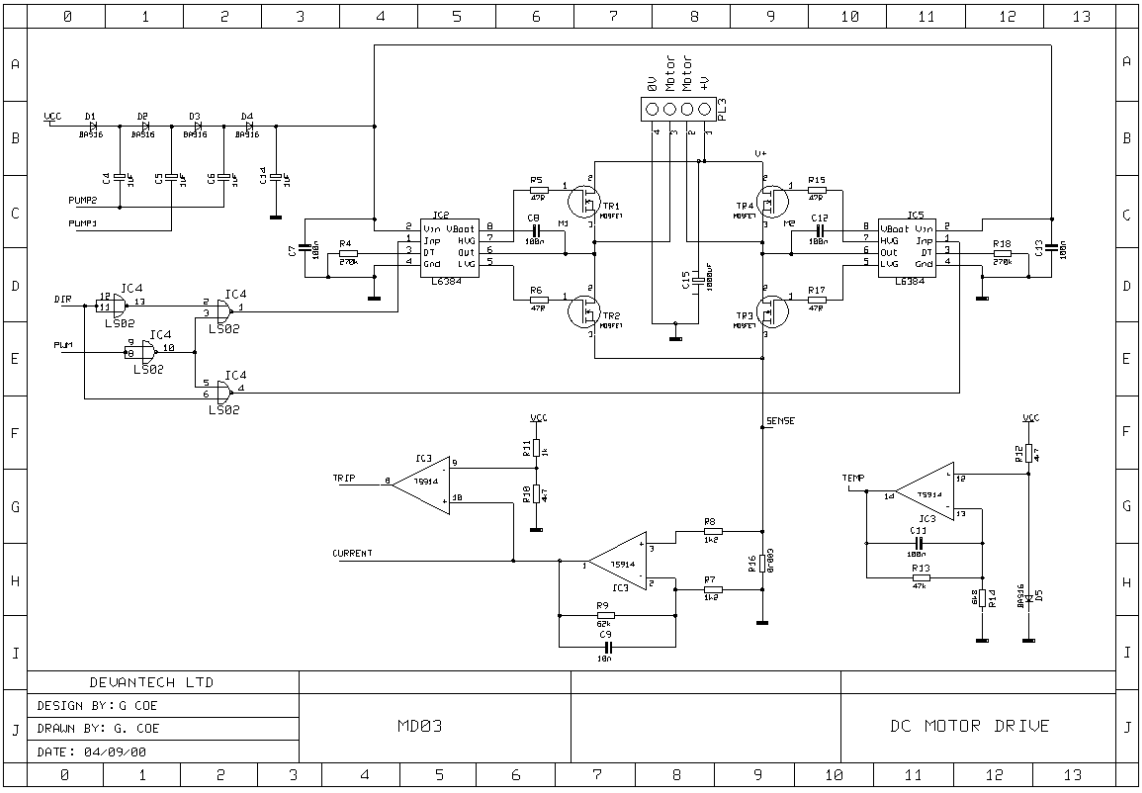
APPENDIX A: HARDWARE

A.1 PROPORTIONAL SERVO MOTOR DRIVER

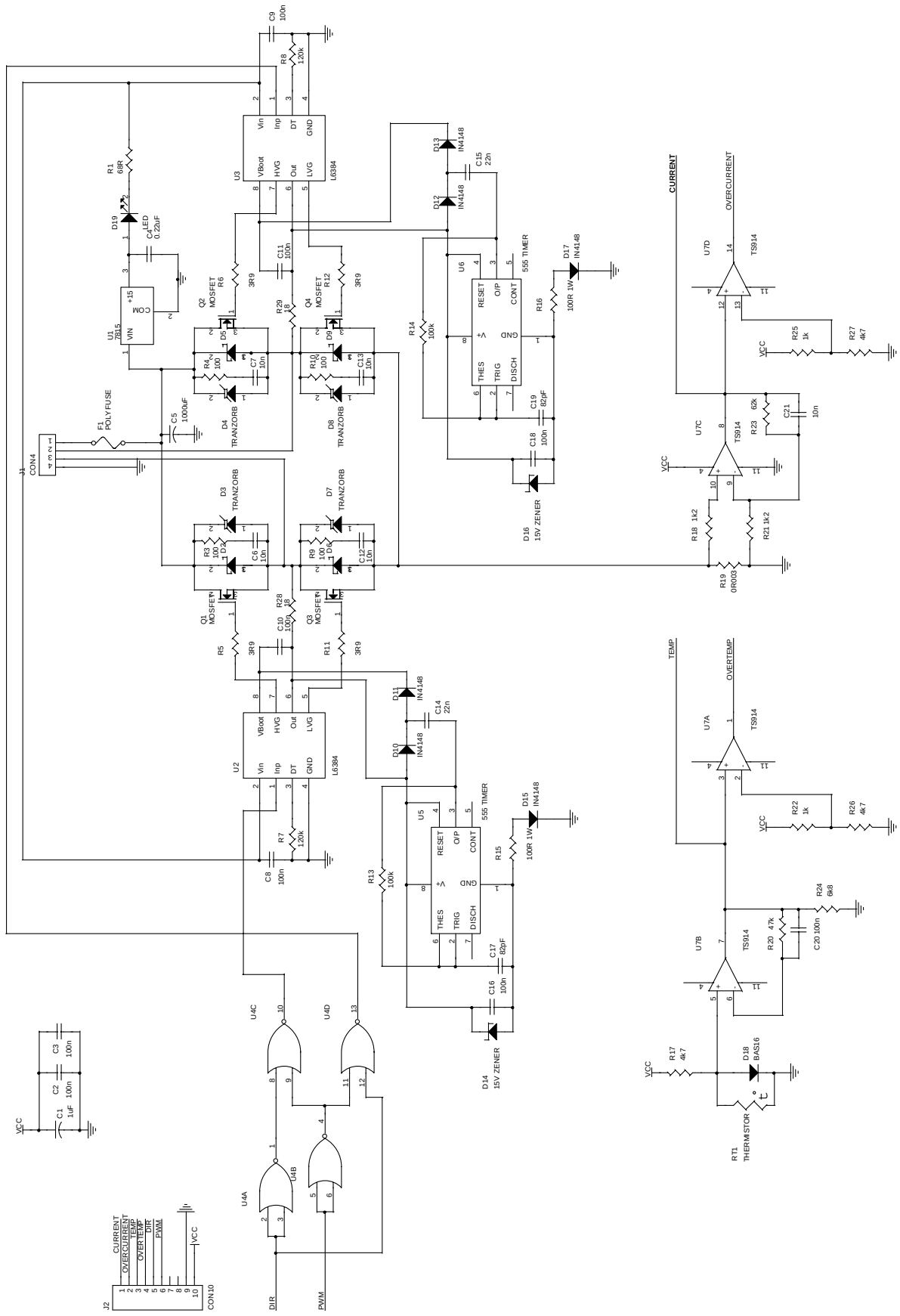


A.2 MD03 DEVANTECH MOTOR DRIVER SCHEMATIC

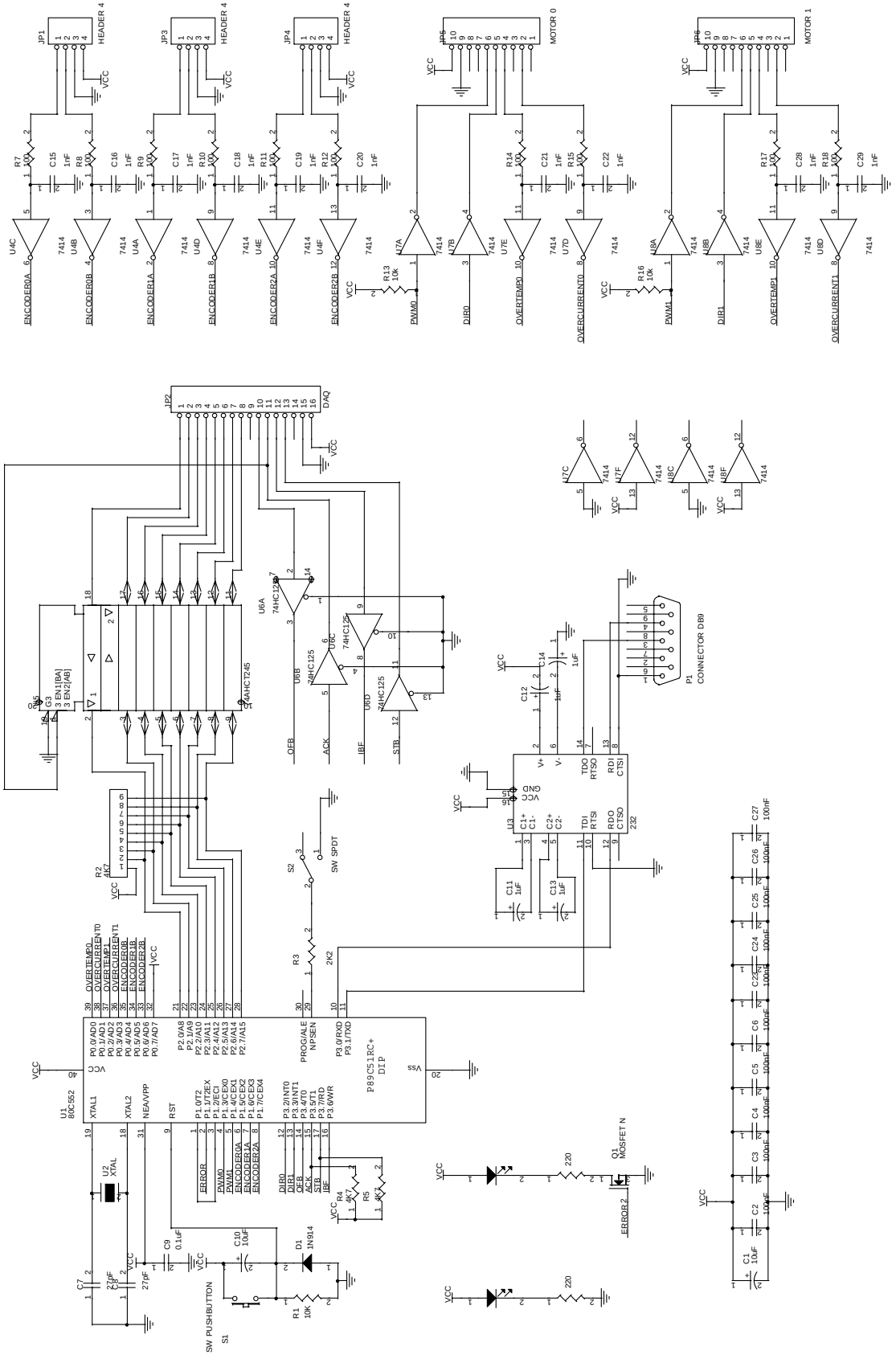




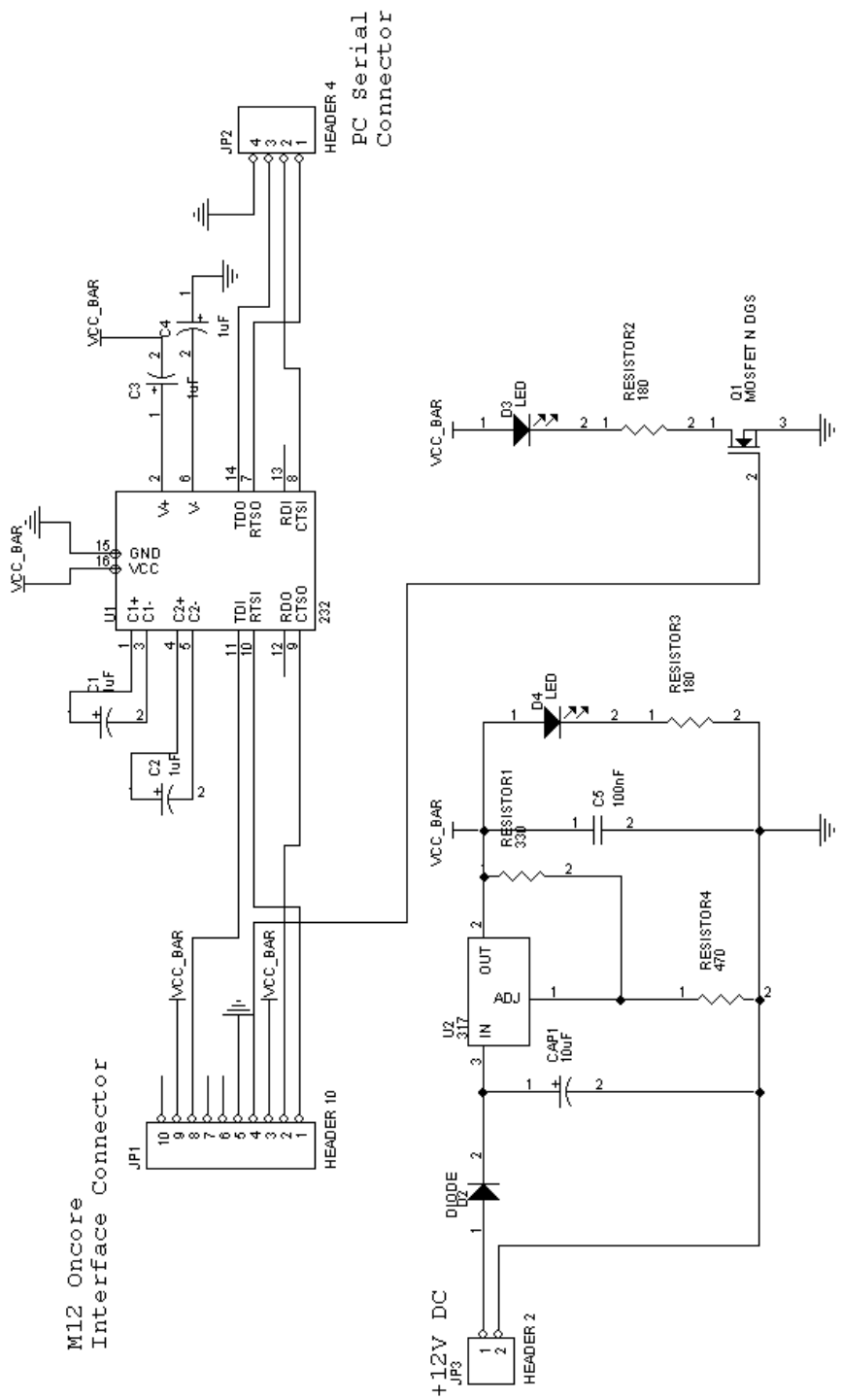
A.3 MOTOR DRIVER SCHEMATIC



A.4 MOTOR MICROCONTROLLER SCHEMATIC



A.5 MOTOROLA M12 ONCORE GPS RECEIVER INTERFACE



APPENDIX B: CD CONTENTS

The attached CD contains the following:

- **Outdoor Operation Videos**

- **Photo Gallery**

- **User Manual**

- **Software**

  Code:

  - Microcontroller C code
  - Base Station Visual C++ code
  - Rover Visual C++ code

  Supporting Software:

  - Demo version of HI-TECH 8051 compiler v8.00
  - WinISP v2.29
  - WinOncore12
  - NiDAQ v6.9.3

- **Datasheets**

  - DAQ - LabPC+ card user manual
  - GPS - Motorola M12 Oncore user guide supplement
  - Microcontroller - P89C51RC2H datasheet
  - Shaft encoder - HEDS-5701 datasheet
  - Object detector – GP2Y0A02YK datasheet
  - IC components: - Microcontroller PCB
    - Motor driver PCB
    - GPS Interface PCB

GLOSSARY

802.11b	Wireless network protocol
ADC	Analogue to Digital Converter
Control segment	Part of a GPS system used to monitor and control the satellites and their signals
DAQ	Data Acquisition
DOD	Department of Defense
DOP	Dilution of Precision. Geometrical limitation to position accuracy.
Ephemeris	A table giving the coordinates of a celestial body at a number of specific times during a given period.
GLONASS	Global Navigation Satellite System
GPS	Global Positioning System
IGBT	Insulated Gate Bipolar Transistor
I/O	Input/Output
MOSFET	Metal Oxide Semiconductor Field Effect Transistor
MSB	Most Significant Bit
Navigation message	A message sent from satellites containing various parameters
NAVSTAR	Navigation satellite timing and ranging, official US DOD name for GPS
NIC	Network Interface Card
PID	Proportional Integral Derivative.
PPS	Precise Positioning Service.
Pseudo-range	Distance calculated using timing to generate the distance. Not geometrically measured
PWM	Pulse Width Modulation
Ranging codes	The code sent from a satellite, used to determine the distance to that satellite.
SA	Selective Availability. DOD added error for civilian users. Currently NOT active.

Space segment	Part of a GPS system which involves the satellites and their orbits
SPS	Standard Positioning Service.
UPS	Uninterruptible Power Supply
TRANSIT system	uses Doppler shift measurements alone to determine the user's position.
User segment	Part of a GPS system which involves users locating a position.
Vector	Software structure similar to an array, capable of storing multiple values

BIBLIOGRAPHY

[1] Engineering Services Inc.
http://www.esit.com

[2] RoboProbe Technologies Inc.
http://www.roboprobe.com

[3] Defenders Network Inc.
http://www.defend-net.com

[4] ActivMedia Robotics
http://www.activrobots.com

[5] Lynxmotion
http://www.lynxmotion.com

[6] Exide, **"***Battery Specifications Section 2",*
http://www.exide.co.nz/pdf/section2.pdf

[7] National Instruments, *"Lab-PC+ User Manual",* June 1996 Edition

[8] Payne, A., and Carnegie, D., *"Design and Construction of a Pair of Tricycle Based Robots to Investigate Cooperative Robotic Interaction"*, Proceedings of the Tenth Electronics New Zealand Conference, 2003.

[9] Robot Electronics, 20 A H-Bridge MD03
http://www.robot-electronics.co.uk/images/md03sch2.gif

[10] IR Application Note AN-978, *"HV Floating MOS-Gate Driver ICs"*
http://www.irf.com/technical-info/an978/an-978p17.htm

[11] Chao, C., *"Using WARP Speed IGBTs In Place Of Power MOSFETs at Over 100 kHz Converter Applications"*
http://www.irf.com/technical-info/designtp/dtwarp.html

[12] *"Introduction to the Global Positioning System for GIS and TRAVERSE"*
http://www.cmtinc.com/gpsbook/index.htm

[13] Trimble Navigation Limited
http://www.trimble.com

[14] Dana, P., *"Global Positioning System Overview"*, 2000
http://www.colorado.edu/geography/gcraft/notes/gps/gps_f.html

[15] Garmin Corporation, "GPS GUIDE for beginners", 2000
http://www.garmin.com/manuals/GPSGuideforBeginners_Manual.pdf

[16] Mercator GPS Systems *"GPS Tutor"*
http://mercat.com/QUEST/gpstutor.htm

[17] Ashjaee, J. & Ashjaee, N., *"A GPS Tutorial"*,1998
http://www.topconps.com/gpstutorial/

[18] Spilker Jr J., *"GPS Signal Structure and Performance Characteristics"*, 1980. Global Positioning System, The Institute of Navigation, Vol 1.

[19] Motorola, *"M12 Oncore User's Guide Supplement"*

[20] Graham, R., *"FAQ on PID Controller Tuning"*, 2004
http://www.tcnj.edu/~rgraham/PID-tuning.html

[21] Franklin, G. F, Powell, J. D, Workman M, *"Digital Control of Dynamic Systems"*, 3rd Edition, 1998.