The Development of the Control for an Urban Search and Rescue Robot Manipulator Arm

Bradley Mark Springer
Robotics and Agents Research Laboratory
Department of Mechanical Engineering
University of Cape Town

Supervised by Stephen Marais
Robotics and Agents Research Laboratory
Department of Mechanical Engineering
University of Cape Town

February 2013

The financial assistance of the National Research Foundation (NRF) towards this research is hereby acknowledged. Opinions expressed and conclusions arrived at, are those of the author and are not necessarily to be attributed to the NRF.

Acknowledgements

An immense thank you must first be extended to my supervisor, Stephen Marais, for his enduring support during the completion of this project. Thank you for the guidance and help through it all.

Thank you Tracy for your assistance during this project, especially for your work on the LM3S8962 circuit board design. I am hugely appreciative for all your help.

To my mom: thank you for everything that you do for me; for being my greatest supporter in everything I do. I cannot begin to thank you enough for your encouragement over the last two years. I promise I'll find a job soon.

To Paddy and Sue: thank you for getting me to where I am today. Thank you for the supportive phone calls and endless encouragement through the peaks and valleys of this thesis. I am immensely grateful for everything you have done for me.

Thank you to my friends and family, especially Mike, Dave, Richard, Cameron, JP and Julian in the RARL lab. This project was made all the more enjoyable with your helpful company. Thank you for all the assistance during this time.

And thank you Swanny for always having my back. Thank you for both the encouragement and the "Im proud of you" motivation throughout this time. Brandy Swinger for the win.

The financial assistance of the National Research Foundation (NRF) towards this research is hereby acknowledged. Opinions expressed and conclusions arrived at, are those of the author and are not necessarily to be attributed to the NRF.

Declaration

This thesis is submitted in complete fulfillment of a M.Sc.(Eng) (Mechanical Engineering) at the University of Cape Town.

I know the meaning of plagiarism and declare that all of the work in the thesis, save for that which is properly acknowledged, is my own. Each contribution to, and quotation in, this thesis from the work(s) of other people has been attributed, and has been cited and referenced according to the style for IEEE Transactions. I have not allowed, and will not allow anyone to copy my work with the intention of passing it off as their own.

Signature..... Date.....

Executive Summary

Urban Search and Rescue (USAR) focuses on locating and extracting victims affected by collapsed or damaged structures. These environments pose a serious hazard to rescue personnel through the risk of further collapse, gas leaks or explosions [1]. Robotic systems have shown that they are capable of taking on tasks that are too dangerous or undesirable for humans and have been used in deep-sea, space and nuclear operations [2, 3]. The combination of these applications form the principle upon which USAR robotics is based; USAR robotics allows for the safe execution of exploratory tasks in hazardous urban environments through the use of teleoperated and autonomous robotic vehicles [4].

Examples of robots used in urban search and rescue environments such as after the attack on the World Trade Centre in 2001 and the investigation of the Fukushima Nuclear Power Station in the aftermath of the earthquake in Japan in 2011 are shown below.

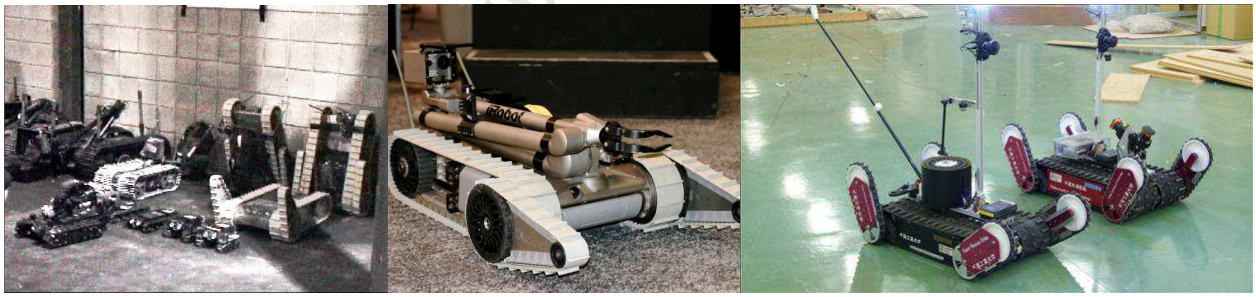

Figure 1: Robots used in USAR tasks. Collection of robots used in World Trade Centre (WTC) attack (left) [5], the Packbot developed by iRobot used in both WTC and Fukushima scenarios (centre) [6] and the Quince 2 robot developed by Chiba Institute of Technology used in the surveillance of the Fukushima Nuclear Power Station [7].

The University of Cape Town (UCT) Robotics and Agents Research Laboratory (RARL) began research into developing a USAR robot in 2006. The final design of the fourth generation USAR robot developed by UCT, named RATEL, can be seen in Figure 2.

Figure 2: UCT's USAR robot (photo courtesy of Richard Whittemore).

This document reports on the research and development of the control for the four degree-of-freedom manipulator arm and pan/tilt system located on the RATEL USAR robot. The report initially discusses control methods used on previously developed teleoperated manipulators in the fields of kinematic modelling, motor control, communications architectures, teleoperative interfaces as well as collision detection and proceeds to discuss the development of the control for the RATEL manipulator.

An inverse kinematic model was initially developed in Matlab to allow for the intuitive control of the end effector of the manipulator. The geometry of the manipulator restricted the movement of the end effector when the first link reached its horizontal limit and thus a dual-axis inverse kinematic algorithm was developed. This approach allowed for the easy and accurate control of the arm.

Proportional - Derivative control laws were developed and coded onto LM3S8962 motor control boards using the LabVIEW Embedded for ARM module to provide the necessary accurate motor position control required of the inverse kinematic calculations. This motor control, together with the inverse kinematic calculations, provided an accurate end effector position. The arm, being controlled by an operator using the inverse kinematic control method to manually draw boxes, can be seen in Figure 3. The pan/tilt system on the arm proved to be difficult to control due to the weight of the sensor payload together with the gearing system used in that section of the arm.

Figure 3: Drawing boxes with the RATEL arm by utilising the inverse kinematics. The image on the left illustrates the ability of the arm to still accurately maintain the desired motion even once the first link is locked at the horizontal and the two link inverse kinematics is implemented.

A TCP/IP communications platform to fulfill the wireless communications requirement of the project was developed. Numerous on-board communication architectures including I^2C, RS-232 and RS-485 were developed to transmit the desired motor positions to the LM3S8962 motor controllers. The most successful of these was the RS-485 communications scheme which produced the required communications stability at an acceptable speed of 13Hz.

Hall effect sensors, triggered on the proximity of magnets located on the gearing of each joint of the arm (see Figure 4), indicated the absolute position of each motor. This information was used to provide the necessary calibration functionality for the arm. Mounts to house all the electronics internally were also developed.

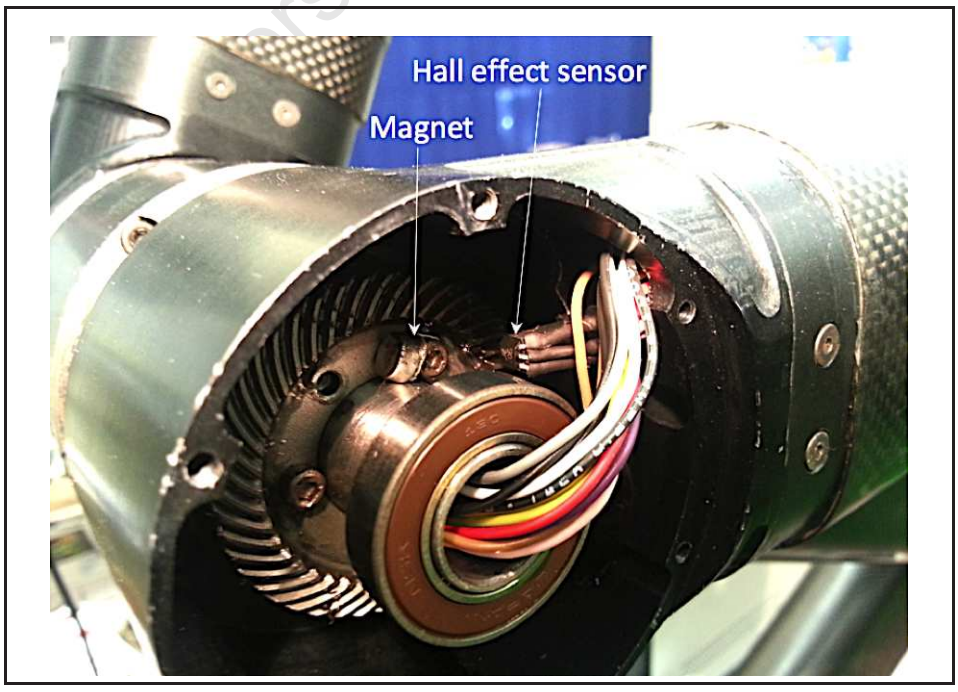

Figure 4: Magnet and hall effect sensor located on the sixth motor gear assembly.

The SpaceNavigator 3D mouse, produced by 3DConnexion, (shown in Figure 5) was used as the teleoperation interface for the user. This device offered control in six degrees-of-freedom thus providing a highly intuitive control interface for the manipulator. The keyboard was used to provide additional control functionality. A graphical representation of the robot (see Figure 6) was developed to provide the user with a 3D image of the remote vehicle which was used in conjunction with an interface control loop making use of both forward and inverse kinematics to improve the response of the robotic arm.

Figure 5: SpaceNavigator 3D mouse produced by 3DConnexion used to control the RATEL robotic arm [8].

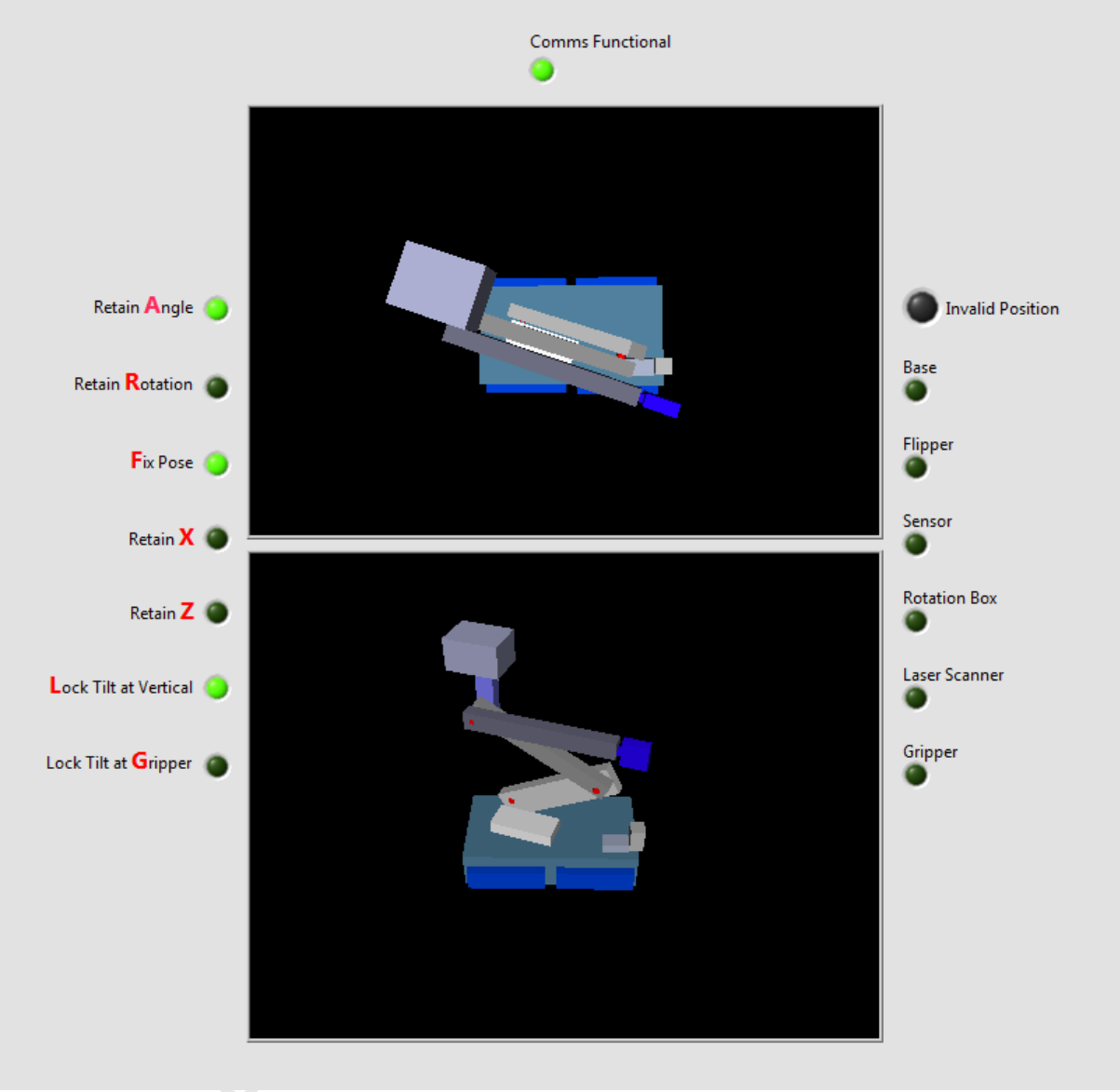

Figure 6: Graphical user interface for the RATEL robotic manipulator including all elements of the robotic arm as well as the flippers and laser scanner located on the base of the robot.

Collision prevention algorithms were developed using the separating axis theorem to prevent the manipulator from colliding with other elements of the robot. The elements of the robot were modeled as simple polygons to allow for fast collision processing while visual and verbal warnings were used to inform the user of any imminent collisions. These algorithms proved to effectively prevent any collisions between the robot elements.

User feedback after testing of the manipulator system showed that the control of the robot arm was easy and intuitive to use. Novice test candidates were able to complete a pick-and-place task (see Figure 7) in only a matter of minutes illustrating the effectiveness of the control of the manipulator.

Figure 7: Pick and place task performed by novice users. The test candidates were required, using only the camera feeds available, to pick up the bottle from position A and place it in position B.

A number of recommendations are made regarding the improvement of the system, most notably modifications to the sensor payload as well as pan/tilt gearing systems to allow for better control of the sensor payload. It is recommended that C code be used on the motor control boards to improve the communications speed while further collision detection techniques, through the use of current monitoring, are also recommended.

Contents

List of Figures

List of Tables

Nomenclature

DAC	Digital to Analogue Converter
DOF	Degree-of-Freedom
FPGA	Field-Programmable Gate Array
GUI	Graphical User Interface
I^2C	Inter-Intergrated Circuit
IC	Integrated Circuit
LabVIEW VI	LabVIEW Virtual Instrument. A collection of LabVIEW code used to perform a specific function. Similar to a 'method' used in C++
NIST	National Institute of Standards and Technology
PID	Proportional-Integral-Derivative
RARL	Robotics and Agents Research Laboratory
SAT	Separating Axis Theorem
TCP/IP	Transmission Control Protocol/Internet Protocol
UCT	University of Cape Town
USAR	Urban Search and Rescue
VHDL	VHSIC Hardware Description Language

Chapter 1

Introduction

1.1 Background

Urban Search and Rescue (USAR) focuses on locating and extracting victims affected by collapsed or damaged structures. These environments pose a serious hazard to rescue personnel through the risk of further collapse, gas leaks or explosions [1]. Robotic systems have shown that they are capable of taking on tasks that are too dangerous or undesirable for humans and have been used in deep-sea, space and nuclear operations [2, 3]. The combination of these applications form the principle upon which USAR robotics is based; USAR robotics allows for the safe execution of exploratory tasks in hazardous urban environments through the use of teleoperated and autonomous robotic vehicles [4].

Examples of robots used in urban search and rescue environments such as after the attack on the World Trade Centre in 2001 and the investigation of the Fukushima Nuclear Power Station in the aftermath of the earthquake in Japan in 2011 are shown below.

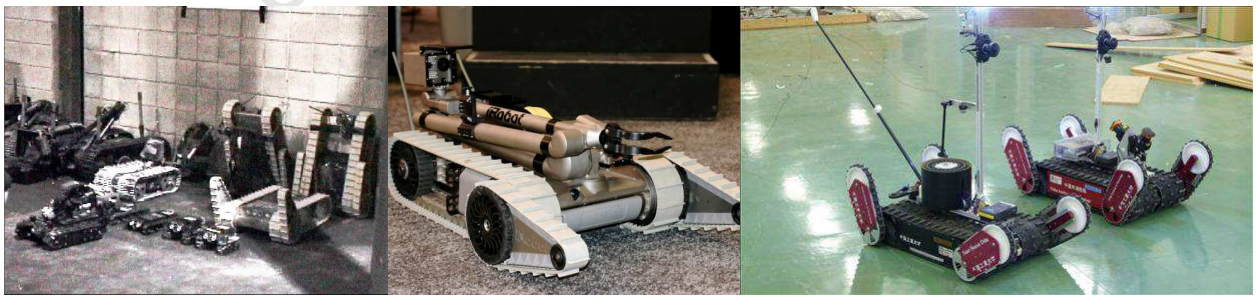

Figure 1.1: Robots used in USAR tasks. Collection of robots used in World Trade Centre (WTC) attack (left) [5], the Packbot developed by iRobot used in both WTC and Fukushima scenarios (centre) [6] and the Quince 2 robot developed by Chiba Institute of Technology used in the surveillance of the Fukushima Nuclear Power Station [7].

In order to accelerate the development and deployment of robotic tools, the National Institute of Standards and Technology (NIST) created the "Reference Test Arenas for Autonomous Mobile Robots" to evaluate the performance

of robotic technologies in completing USAR tasks in simulated urban disaster situations. In these tests, robots were required to navigate through the maze-like arena whilst negotiating obstacles, search for simulated victims and produce a map of the environment. In 2002, the annual RoboCup competition, which previously focused only on soccer playing robots, replicated the test arenas developed by NIST to create the first RoboCup Rescue League which, to this day, remains a fundamental initiative in testing the performance of USAR robotics [9].

In the RoboCup Rescue competition, participating teams are required, via remotely teleoperated and autonomous vehicles, to demonstrate their abilities in mobility, sensory perception, localization and mapping, mobile manipulation, practical operator interfaces and assistive autonomous behaviours to improve operator performance and/or robot survivability. These abilities are evaluated in maze arenas which are comprised of different terrains and challenges of differing levels of difficulty as shown in Figure 1.2 [10].

Figure 1.2: The 2012 RoboCup Rescue arena. Photo courtesy of Richard Whittemore.

In the RoboCup Rescue competition robots are required to:

- negotiate compromised and collapsed structures
- locate victims and ascertain their condition
- produce practical sensor maps of the environment
- establish communications with the victims
- deliver fluids, nourishment and medicines
- place sensors to identify/monitor hazards
- mark or identify the best route to victims
- provide structural shoring for responders.

As well as the requirements of search and rescue robots in the RoboCup Rescue Robot League, there have been initiatives to develop sets of requirements for USAR robots in real-world applications [11]. One such attempt was that of the Department of Homeland Security, the Science and Technology Directorate and the National Institute of Standards and Technology which, in May 2005, endeavoured to document an initial set of requirements for the performance of robots that can support USAR roles and tasks [12]. These requirements were developed by assembling a group of experts in the subject and holding workshops to construct and develop and, ultimately, define the initial requirements for the robots. In the first workshop, the responsibilities of a USAR robot were defined as:

- Reconnaissance
- Primary search
- Structural Assessment
- Stabilization
- Medical
- Rescue
- Monitoring
- Detection of hazardous materials

These coincide well with the eight requirements stated previously in the RoboCup Rescue rules. The succeeding two requirement definition workshops produced a total of 103 requirements for USAR robots which can be found in [12].

The University of Cape Town's (UCT) Robotics and Agents Research Laboratory (RARL) began research into developing a USAR robot in 2006 with the aim of entering the RoboCup Rescue League. The final design of the fourth generation USAR robot developed by the lab, named RATEL, at the RoboCup competition in Mexico in 2012 is shown in Figure 1.3.

Figure 1.3: The Ratel USAR robot (photo courtesy of Richard Whittemore).

The RATEL USAR robot platform is a 502mm by 734mm tracked vehicle, developed by Eugene Dreyer [13] and David Lwabona [14], comprising of independently rotating flippers to facilitate stair climbing and navigation on harsh terrains. The sensor payload on the robot, developed by Cameron Sharpe [15], contains a high resolution camera, thermal camera, CO2 sensor, microphone, speaker, floodlight and spotlight in order to assist in the navigation and victim location tasks required of a USAR robot. Automatic object detection as well as three-dimensional (3D) image creation was developed by Richard Whittemore [16] to further contribute to victim location and hazardous materials detection. Mapping capabilities, developed by Julian Kent [17] and facilitated by the laser scanner and Inertial Measurement Unit (IMU) fixed to the base of the robot, are included in the functions of the robot to produce two-dimensional (2D) maps of the navigated environment. Michael Rieger [18] developed the end effector of the robot. This parallel gripping device includes multiple sensory capabilities to assist the operator in performing gripping tasks.

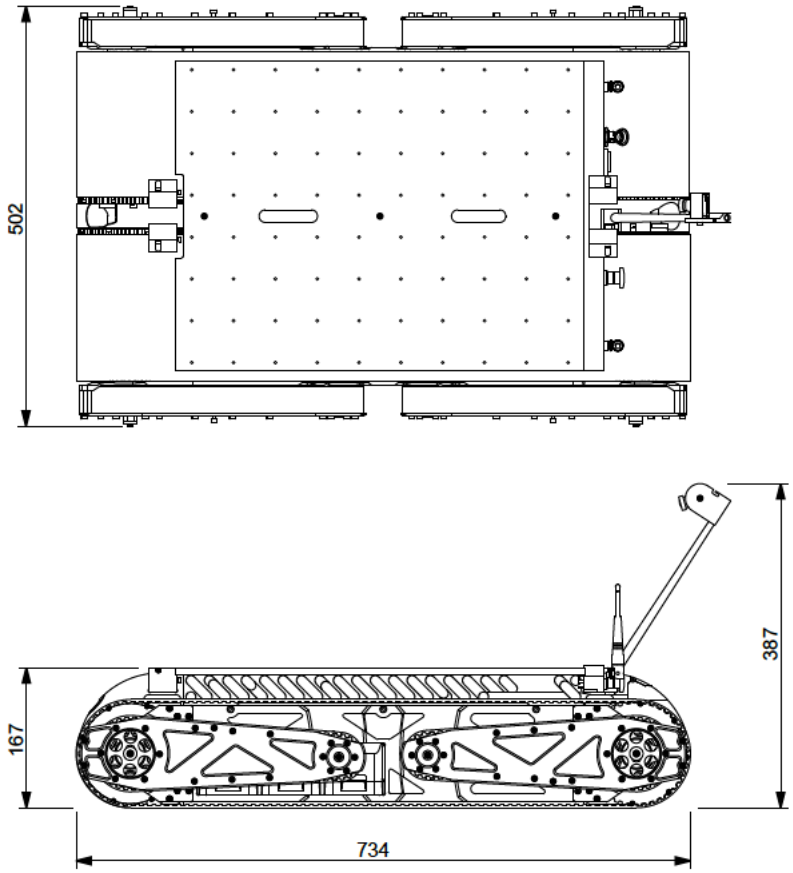

Figure 1.4: RATEL USAR robot platform dimensions [13].

Peter Henson [19] developed the mechanical design for the 4 Degree-of-Freedom (DOF) manipulator arm, including the additional 2 DOF pan/tilt section for the sensor payload, on the robot. The design incorporated the use of three brushless 120W Maxon EC40 motors to be controlled by Maxon DECV 50/5 motor speed controllers (Rotation, Shoulder and Elbow1) and three 20W Maxon EC22 motors to be controlled by Maxon DEC 24/3 motor speed controllers (Pan, Tilt and Elbow2) shown in Figure 1.5.

The objective of this project was to develop the control software and interface in order to effectively control and maneuver the robotic manipulator designed by Henson. The requirements of the project are specified in the following section.

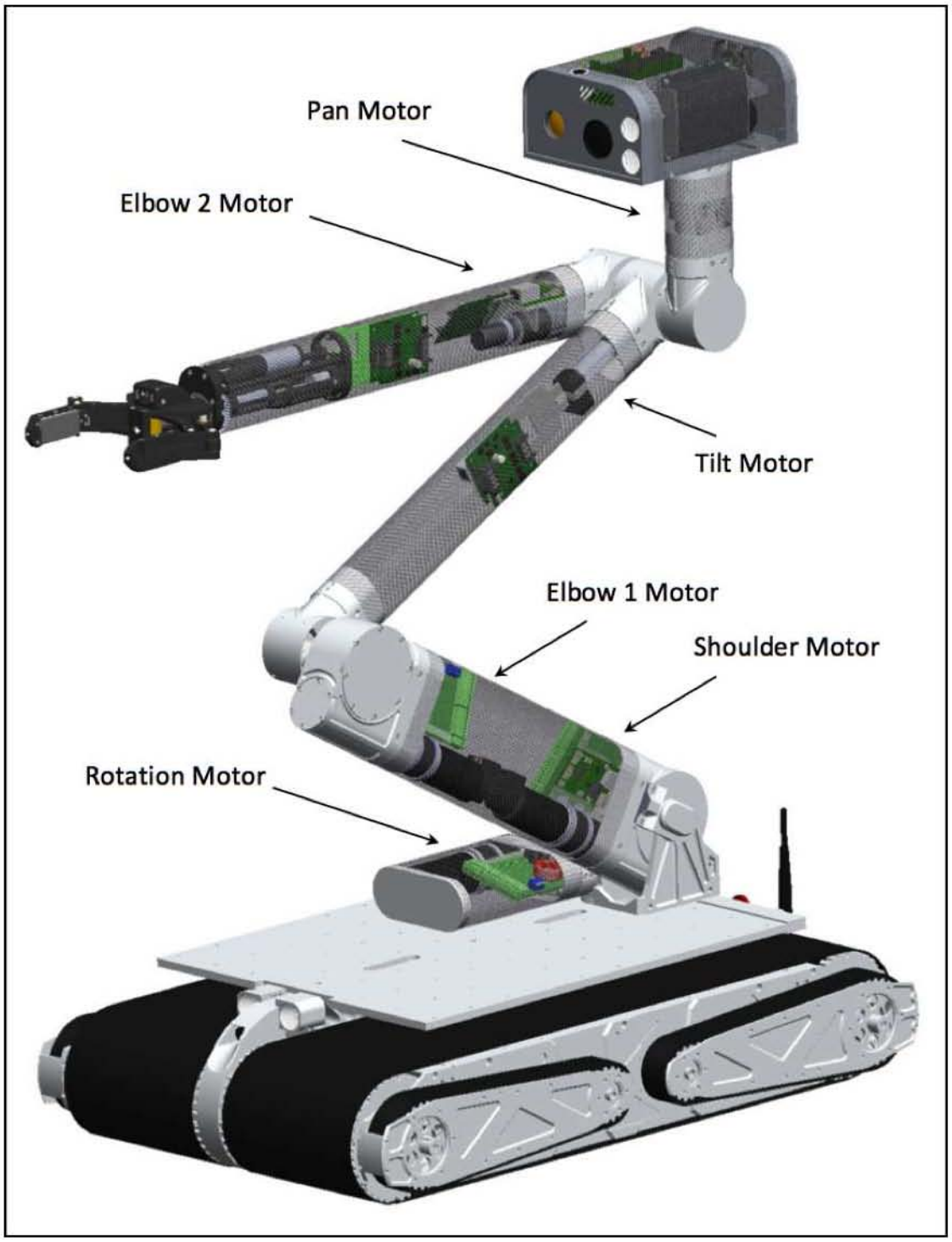

Figure 1.5: SolidWorks model of Ratel USAR robot showing motor assemblies.

1.2 Requirements

In order to fulfill as many of the above-mentioned responsibilities of a USAR robot as possible, the following requirements were specified by Stephen Marais, the head of RARL:

1. The arm was required to be controlled over a wireless network from a remote control station.
2. The control was required to be easy and intuitive to use.
3. The control was to integrate with the gripper/wrist assembly designed by Michael Rieger at RARL.
4. The system was required to avoid collisions with other elements of the robot.
5. The system was required to allow for the calibration of the arm.
6. The movement and control of the system had to be stable enough to easily manipulate the arm and grasp objects with the end effector as well as perform effective surveillance with the sensor payload.
7. The electronics and wiring of the robotic arm were required to be housed internally.

1.3 Plan of Development

Chapter 2 provides background research on robotic manipulators that have been designed in the past. There is particular focus on teleoperation and manipulation, kinematic modelling, collision detection, communication architectures and motor position control.

Chapter 3 provides a functional analysis of the project and tabulates the different functions that the system needed to perform in order to fulfill the requirements stipulated above. This leads to the product specification.

Chapter 4 expands on the kinematic modelling involved in implementing an inverse kinematic control platform for the robotic arm. It then expands on the implementation and complications encountered while creating a forward kinematic model for use in a feedback system for the control of the arm.

Chapter 5 discusses the implementation of motor position control in order for the robotic arm to effectively assume the pose dictated by the inverse kinematic algorithm, discussed above. An investigation into the use of an FPGA - Coldfire interface for position control is initially investigated. Thereafter, the development and use of the LM3S8962 motor control board is expanded upon. Discussion of the derivations of transfer functions and hence Proportional - Derivative (PD) control laws for each of the motors forms the final part of the chapter.

Chapter 6 provides an extensive investigation into three different communication architectures implemented on the arm. Firstly, the implementation of Inter-Integrated Circuit Bus (I^2C) is discussed. When I^2C failed to provide the stable communications scheme necessary for the control of the arm, RS-232 was investigated and is discussed. Finally, the implementation of RS-485 is discussed and expanded upon.

Chapter 7 focuses on the electronics incorporated into the control of the arm. This includes the use of hall effect sensors for calibration of the arm as well as the design of the mounting brackets in order to house the necessary electronic components inside the manipulator.

Chapter 8 investigates the development of the control interface for the robotic arm. This includes the Graphical User Interface (GUI) as well as the control device selected for the manipulation of the arm. The implementation of a Separating Axis Theorem collision prevention algorithm concludes the chapter.

Chapter 9 In this chapter, the testing of the different elements of the system as well as the results obtained are discussed. Testing and the results of motor control, communications, collision prevention and usability testing are discussed.

Chapter 10 concludes the project by relating the results obtained in the test phase to the initial requirements. Recommendations for future work on the system are then given.

Chapter 2

Background Research

This chapter summarises some of the literature relevant to the development of the control for a robotic manipulator on an urban search and rescue robot. The first section explores the RoboCup Arena in more detail with specific focus on the tasks/challenges required of a robotic manipulator. The concept of teleoperation and the devices used in the remote control of robotic manipulators is then discussed. Section three highlights the kinematic models that have been developed to implement user input on the robot arm. This is followed by an investigation into collision detection techniques used in manipulator control. The fourth section considers the communication architectures used to implement the above-mentioned control schemes on a robotic arm and the review concludes with a discussion on motor position control.

2.1 The RoboCup Arena

As mentioned briefly in the previous chapter, the RoboCup Rescue competition allows for the testing and evaluation of rescue robots in maze arenas comprised of different terrains and challenges and of differing levels of difficulty. The RoboCup Rescue rules [20] stipulated the tasks or challenges that a robot would face in the competition arena. One of the tasks or challenges created in order to directly test the abilities of a manipulator arm was the elevated/hidden victim identification tasks, shown in Figure 2.1. These tasks were designed to differentiate between the robots with manipulator sensory capabilities and those with sensors mounted upon the robot frame.

Figure 2.1: Example of an elevated victim identification box at the RoboCup Rescue Competition.

Each victim identification box contained:

- Form (doll or mannequin parts)
- Visual (eye charts and hazardous materials labels)
- Thermal (heating pad)
- Motion (waving cloth)
- Sound (random numbers)
- CO2 (soda or tire cartridges)

The second task created to test manipulator control was the pick and placement of water bottles, walkie-talkies or wooden cubes in designated target areas. The objects were placed at various heights (0m, 0,5m and 1m) and at different depths (0,3m and 0,6m). The robot would be required to pick and place these objects within any one of the victim identification boxes for the accruement of additional points. Figure 2.2 shows an example of this:

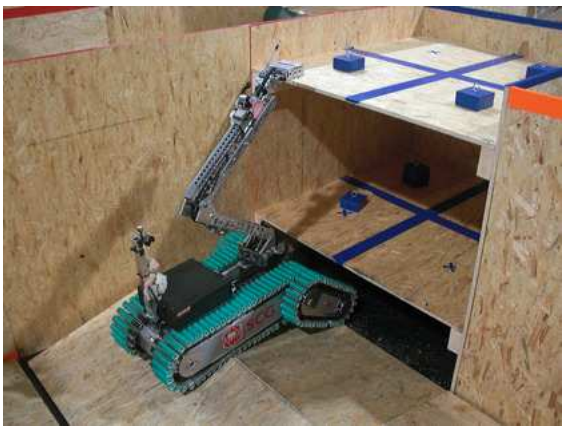

Figure 2.2: Robot performing pick and place task.

The scanning of QR codes was an additional challenge introduced in the 2012 RoboCup Rescue competition. These QR codes were placed in the victim identification boxes as well as in vertical and horizontal cavities around the arena. An additional QR scanning challenge was the pipestar object placed in the rescue arena (see Figure 2.3). This object, comprising of either 25mm or 50mm diameter pipes, had QR codes or eye charts placed in each point and thus stringently tested the dexterity and control of a manipulator.

Figure 2.3: Pipestar code scanning tasks [21].

These tasks, together with the requirements listed in Section 1.2, give a clear perspective of what was required of the control of the robotic manipulator on a USAR robot. It is clear that stable, accurate and dexterous control was required in order to complete the tasks discussed above.

This chapter will now consider the implementation of control in existing robotic arm platforms and the devices used to control robotic arms.

2.2 Teleoperation Interfaces

"Teleoperation is the safe extension of human sensing and manipulation capabilities to a remote location via some sort of vehicle or manipulator." [22]

The term *teleoperation* simply means operating at a distance. Thus, teleoperation allows a user to manipulate a body from a remote position through the physical interaction with a control (master) device and/or interface. Any interaction with the control device is transmitted to the remote (slave) device by means of a communication medium whereby the slave object mimics the user's input commands [23]. Manipulation refers to the way in which a user controls and maneuvers a robotic arm. The ability of the user to accurately coordinate and operate a robotic manipulator is vitally important as there are limitations to the autonomous functions a robot arm can perform. A user-operated robot is able to benefit from the perception, judgment and adaptability of human cognition [2].

In this section, the different interfaces that have been implemented to manipulate robotic arms via teleoperation and the effectiveness of each technique will be discussed . The simpler manipulation interfaces used by robotic arm developers will be looked at first followed by the more advanced haptic manipulation technologies.

2.2.1 Open Loop Control Devices

The movement of a teleoperated robotic arm corresponds to the physical action performed by the user on a control device. What follows is a description of different control devices which have been used to achieve this functionality.

Various interfaces, including joysticks, teach pendants and those used for virtual reality, have been used to control robotic systems [22]. One robotic arm system that utilised a mouse as the predominant control instrument was that of Tunwannarux and Tunwannarux with their five-joint mechanical arm on their CEO Mission II rescue robot. A "status image" of the robotic arm was implemented, shown below, which allowed the user to click or drag-and-drop the robotic arm to the desired position. The interface would interpret the user input and, using inverse kinematics, transmit the relevant joint angles to the robotic arm [24].

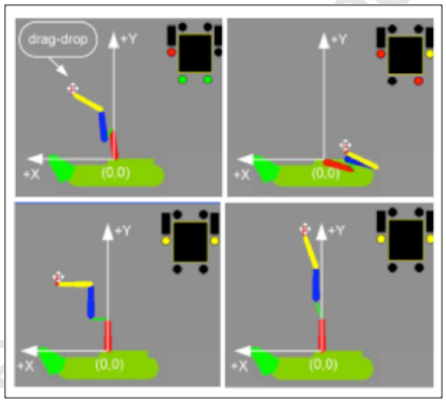

Figure 2.4: The CEO Mission II robotic manipulator GUI showing the drag-and-drop functionality. This allows the user to specify the desired position of the end effector [24].

Many mouse and keyboard teleoperation techniques have been implemented for USAR navigation applications [25, 26, 27, 28]. However, the sources show that because these techniques limit intuitiveness when trying to control a robotic manipulator, alternative manipulator control techniques are far more popular.

An alternative to the use of a mouse is that of a joystick device for manipulator control. NASA, in the development of the Shuttle Remote Manipulator System (SRMS) or Canadarm, utilised two 3 DOF joysticks to control the sophisticated 6 DOF robotic arm system [29]. The joysticks, shown below, control the translational and rotational motion separately, allowing for effective maneuverability of the manipulator [30].

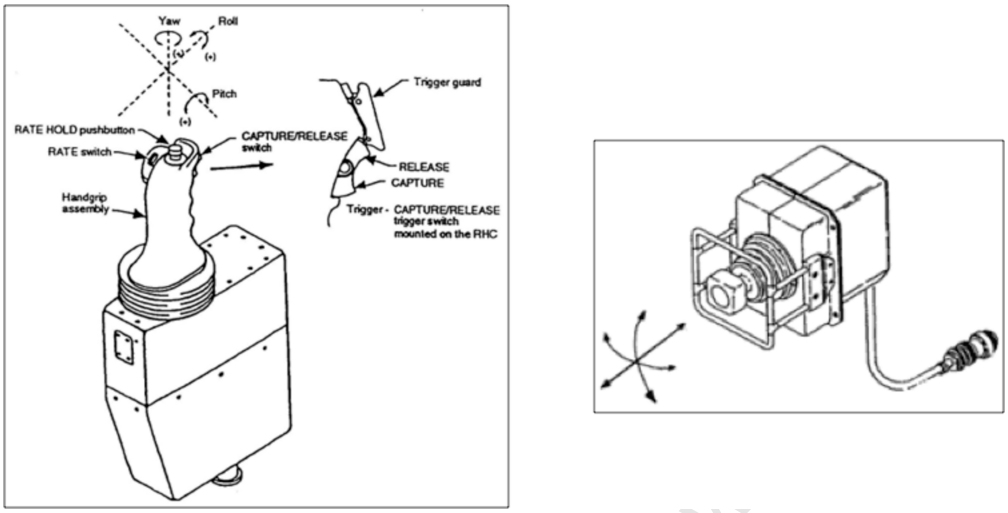

Figure 2.5: On the left, the SRMS translational motion joystick controller and on the right, the SRMS rotational motion joystick controller [31].

Edwards, in his thesis, highlights two commercially available wheelchair-mounted robotic arms which use simple joystick controller instruments; the Manus and Raptor. Both robotic arms use such controllers to provide the users with an easy manipulation method [32]. An example of a joystick is shown in Figure 2.6.

Figure 2.6: Manus joystick controller [32].

As a computer mouse limits the ease of manipulation of a robotic arm, Ryu et al. come to the same conclusion regarding joysticks. They said: "A joystick-type device, which is commonly used for teleoperation, is adequate for navigation, but is inadequate for controlling the manipulator because it does not have an adequate number of degrees of freedom. Therefore, six DOF haptic devices are often applied to a mobile manipulation task because haptic features allow easier operation of this type of task." [33].

Haptic devices, as mentioned above, are an increasingly popular technology being implemented in the teleoperation of robotic manipulators. A discussion as to what haptic technologies are and how they are used in robotic teleoperation follows.

2.2.2 Haptic/Closed Loop Technologies

Many current teleoperation interfaces use visual feedback as the primary method to display the necessary robot control information, such as video, sonar or other sensory feedback, back to the user. The user, having to correlate and fuse all this data mentally, may easily get overwhelmed by this cognitive load. A method to alleviate this is through the use of haptic technologies. Glover et al. describe this process: "Haptic feedback is a mechanical force that is applied through a control device and then perceived by a user". It therefore gives the user a sense of touch relating to the physical environment in which the manipulator is operating [2].

The benefits of haptic feedback in the context of remote teleoperation have been confirmed by a number of sources. Glover et al. show, through three different tests involving the control of a slave robotic arm with an identical master device, that adding haptic feedback in addition to the standard visual feedback allows the user to "feel" what the slave manipulator is experiencing. This significantly improves the user's success rate as well as the time it takes to complete a task [2]. Lee et al, in their haptic experimentation in teleoperating a mobile robot, come to the same conclusion; "Haptic feedback significantly improves operator performance" . They showed that the inclusion of haptic feedback reduced the number of collisions of their mobile robots with the virtual environment without any significant increase in the navigation time [34].

One field in which the benefit of haptic technology has been comprehensively assessed is that of medical teleoperation. An example of this is the work of Wagner, Stylopoulos and Howe who have investigated the effects of force feedback on a blunt dissection task in which subjects used a telerobotic system to expose an artery in a synthetic model. They showed that adding force feedback to the control device significantly reduced the magnitude of the forces applied and reduced the number of errors in comparison to when no force feedback was present. This shows the added dexterity which an operator acquires when equipped with a physical sense of the remote environment [35].

A range of devices have been used in the implementation of teleoperation haptic feedback which could be used for manipulator control. These devices will now be discussed.

The PHANTOM Omni® device, shown below, is a 6 DOF haptic instrument that has been implemented in telerobotic applications.

Figure 2.7: The PHANTOM Omni® haptic device produced by SensAble Technologies, Inc. [36].

The device was originally designed by Massie and Salisbury in 1994 and is now commercially produced, as a line of haptic devices, by SensAble Technologies, Inc. The Omni is the most cost effective haptic device produced by SensAble Technologies, Inc. and offers 6 DOF of position tracking and 3 DOF of force feedback to the user. More advanced PHANTOM devices, such as the PHANTOM 3.0/6DOF, provides the user with the identical position tracking as that of that of the Omni although it includes force feedback in all 6 DOF, incorporating roll, pitch and yaw in addition to the feedback in the x, y and z axes [36].

The miniature robotic arm located on the front of the instrument provides the interface to the user. Encoders track any maneuvering of the device while motors exert any feedback forces on the arm. This allows the users to directly control the remote robotic arm; They implement the movement that they desire on the PHANTOM device and while doing so, physically "feel" any forces acting on the remote arm, thus giving the illusion of contact with objects [37, 38].

In another study, Ryu et al. designed and tested a 6 DOF haptic master for the teleoperation of a mobile manipulator. They argued that even though 6 DOF master devices have greatly improved, they have not been optimised for the purpose of robotic arm teleoperation. The study discusses the fact that in mobile teleoperation, 3 DOF motions are needed for navigation purposes whereas 6 DOF are needed for manipulation control. Conventional 6 DOF devices, which activate all six actuators at once, are therefore undesirable; it would be inefficient to employ manipulator tasks such as the standard planar motion. The design which Ryu et al. adopt incorporates two separable 3 DOF structures mounted on top of each other, as shown in Figure 2.8. This allows the desirable 3 DOF for planar navigation, and, combining with the upper 3 DOF, allows for the optimum 6 DOF of motion for manipulator control [33].

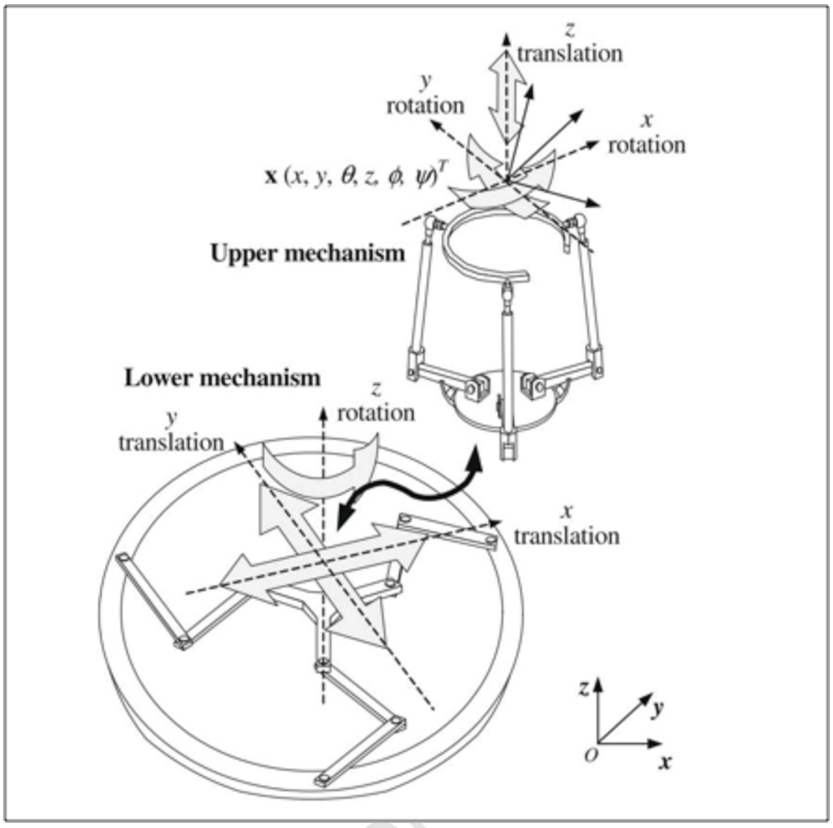

Figure 2.8: Schematic of the haptic master device showing the separable upper and lower mechanisms [33].

The amount of force feedback provided to the user is calculated with respect to the input force from the operator using the kinematic equations used to describe the motion of the haptic device. The device was initially tested using a virtual environment where a USAR robot equipped with a robotic manipulator was simulated. When testing the device virtually, a reaction force was provided to the user in proportion to the depth of penetration of any simulated objects. This, in the real field, protects the manipulator from damage due to any unexpected collisions [33].

A more intuitive haptic teleoperation interface was developed by Glover et al. in which identical master and slave devices were used. They maintained that a haptic master which differs from the remote manipulator could never completely capture the forces acting upon it. Therefore the use of identical master and slave devices is required in order to provide accurate haptic feedback. In comparison to this, Rogers, in his investigation into developing a low-cost teleoperable robotic arm which does not implement haptic feedback, states that the arm and controller need only be kinematically identical in order to create an effective non-haptic teleoperation interface [39]. The manipulators used by Glover et al., shown in Figure 2.9, were the Whole Arm Manipulator (WAM) robotic arm produced by Barrett technology, Inc. These 7 DOF cable-driven robotic arms are human scaled, gravity compensated and back-drivable, allowing for effective haptic control and feedback.

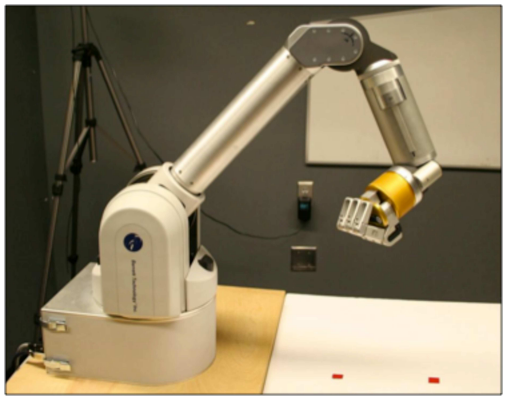

Figure 2.9: The WAM robotic manipulator produced by Barrett Technology Inc. [2].

The implementation of this haptic interface is through the direct manipulation of the master WAM robotic arm. Any force applied to the master arm is then transmitted to the remote arm. In the same way, for effective haptic feedback, any resistance encountered by the remote arm is transmitted back to the master. Glover et al, using this methodology, derived an intuitive control algorithm (see Figure 2.10) for producing the desired driving and feedback torques. In the event of either of the arms not being in the identical position as the other, both arms will attempt to move to the average of their current positions. Therefore, when an arm is moved, the arm that was manipulated torques back towards the position that it previously maintained. The arm that was not moved torques towards the position of the manipulated arm. This provides the teleoperation control of the slave arm while providing the haptic feedback to the user. For example, if the slave arm was to get stuck behind an object, the user operating the master arm would "feel" the effect as the master arm would continuously try to reach the mid-point of the two arms' orientation. Thus a halting force would be exerted upon the master arm's movement [2].

Algorithm 1 Robot Control Loop

1: **while** true **do**
2: Read local joint angles $J_l = l_1, l_2 ... l_n$
3: Receive remote joint angles $J_r = r_1, r_2 ... r_n$ from remote robot over the network
4: Calculate average joint angle vector $A \Leftarrow (J_l + J_r)/2$
5: Use a PID control algorithm to calculate the additional torque that should be applied to each local joint in order to reach the average position A: $T_a \Leftarrow PID(A, J_l)$
6: Transmit the current joint angles of this robot to the remote robot via a network connection: $Transmit(J_l)$
7: Add the additional torque values T_a to the torques $T = t_1, t_2 ... t_n$ that are to be applied to each joint: $T \Leftarrow T + T_a$
8: Apply the torques T to each joint: $ApplyTorques(T)$
9: **end while**

Figure 2.10: Control loop algorithm developed by Glover et al. to simultaneously provide haptic control as well as feedback to the operator by means of position averaging [2].

In this section, the different techniques used in controlling robotic manipulators via teleoperation, ranging from the simplest joystick methods to the more advanced haptic implementations have been analysed. The advantages and disadvantages as well as the applicability of each technique to USAR robotics are important factors to consider

in the development of a USAR teleoperated manipulator. The next section discusses how the information derived from the user input to the control system is implemented on a robotic arm.

2.3 Kinematic Modelling

A crucial step in controlling a robotic manipulator is to acquire and process the commands sent from the chosen manipulation device and "translating" this information into motor commands for each joint. This process is termed kinematic modelling. As Kostic, de Jager and Steinbuch said, ". . . the [kinematic] model is a mapping between the task space (in general, a 6-dimensional space of robot-tip co-ordinates) and the joint space (dimension equal to the number of robot degrees of freedom)" [40]. The process of mapping joint space to task space is called forward kinematics and the opposite is called inverse kinematics [41].

Firstly, a brief explanation of forward kinematics is given. Edlinger et al. implement forward kinematics using the popular Denavit- Hartenberg (DH) notation to compute the orientation of the end effector in respect to the first link of the robotic arm, therefore enabling them to accurately dictate the position of the end effector [42]. Other studies to use the DH notation were that of Chang and Park who used the notation, derived from the implementation of the grid method, in order to acquire an accurate kinematic model for the design of the robotic arm which would realize the task space required of it [43]. Filippi, in his master's thesis, applied the DH convention to produce the forward kinematics model for three different robotic arms, including the Lynx 6, shown below [29].

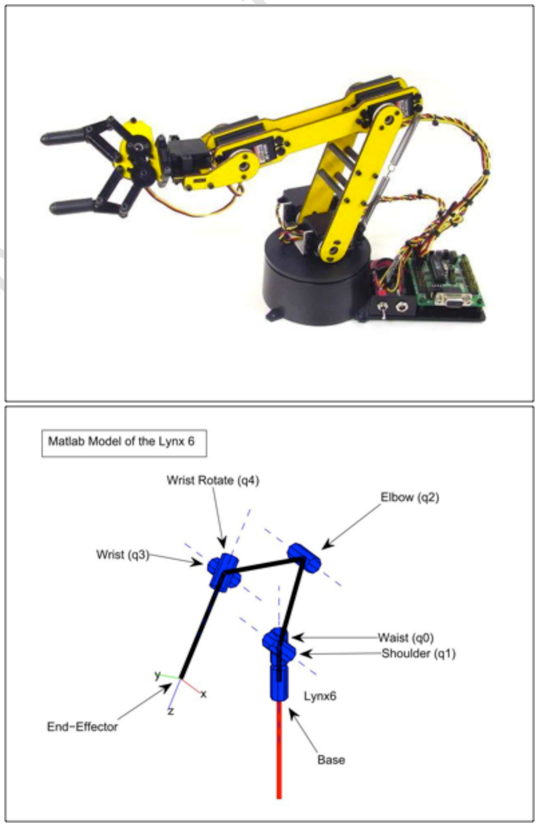

Figure 2.11: The Lynx 6 robotic arm with its kinematic model derived from the Denavit-Hartenberg notation simulated in Matlab by Filippi [29].

Now there is a focus on inverse kinematics. Inverse kinematics increases in complexity with the higher number of degrees of freedom that the manipulator has. The solutions for low DOF manipulators can be calculated using direct analysis, although the calculations become particularly challenging when the number of DOF exceeds that required by the task at hand (termed redundancy). An infinite number of solutions may exist [44]. Tevatia and Schaal state that redundancy, although possibly challenging, is often not disadvantageous to the task at hand; the excess of solutions can be used to optimize certain criteria such as motor current draw or force constraints. Numerical solution methods are therefore commonly used to solve these complex inverse kinematic problems with the optimisation of certain constraints [45].

The implementation of inverse kinematics was referred to in section 2.2.1 with regard to the CEO Mission II robot. This method was adopted to calculate the joint angles of the robot arm from the intuitive robotic arm interface. In this study the team uses geometric methods in order solve the inverse kinematic model [24]. Geometric inverse kinematic methods involve identifying points on the manipulator arm which can be used to express orientation and position by means of a reduced set of joint variables [46]. The Hippocrate robotic arm, a medical purpose robotic arm, is another manipulator to use inverse kinematics. The manipulator employs a direct analytical inverse kinematic model in the control system in order to convert Cartesian coordinates, i.e. probe position, into joint positions [47]. Filippi, in dealing with a high DOF manipulator, opted to use a weighted damped least squares method solution to inverse kinematics in order to provide a satisfactory redundant inverse kinematics solution that aids in the avoidance of joint limits. He implements a control loop incorporating both forward and inverse kinematics, as shown in Figure 2.12, where X is the desired position dictated by the user and X' is the actual position of the end effector of the arm, in order to give the user a higher degree of control of the robot arm [29].

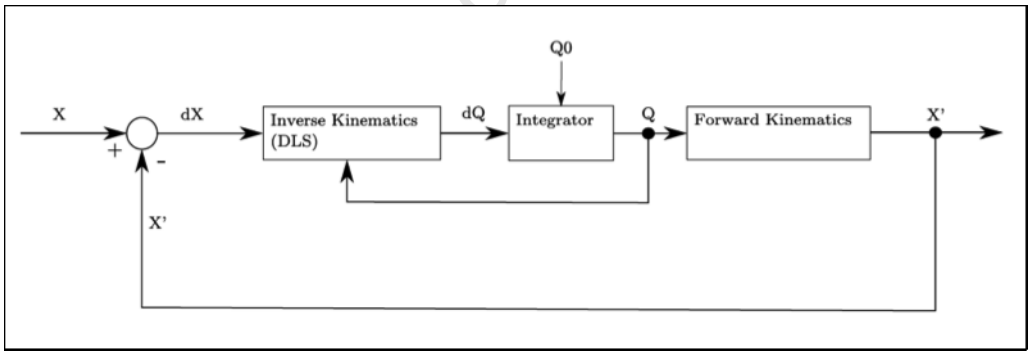

Figure 2.12: Filippi's control loop incorporating both forward and inverse kinematics [29].

A subject closely tied to kinematic modelling is that of collision detection or collision avoidance. This is to ensure that the joint angles calculated by the kinematic model do not cause the robotic manipulator to collide with itself or objects around it.

2.4 Collision Detection

The problem of collision detection has been extensively studied in the field of robotics with a specific focus on the development of collision free trajectories between obstacles in static and dynamic environments [48]. Unlike the

collision detection in these structured environments, an important factor in the scenario of controlling a teleoperated robotic arm is the fact that the motions are not predetermined. Many path-planning and collision-avoidance algorithms are based on the knowledge of future motions. The paradigm for controlling a teleoperated robotic arm must therefore adopt different collision detection algorithms from those implemented in the above-mentioned studies [49].

Most collision detection algorithms are based upon a simulation of the robotic workspace [49]. A particularly unique method of collision detection, using direct sensing and not workspace simulation, was implemented by Chueng and Lumelsky in their development of a "sensory skin" for the motion planning of a robotic manipulator. The authors explain that the skin, which covers the whole body of the robot arm, consists of numerous discrete, active infrared sensors. These sensors detect collisions by processing the amount of light reflected back towards them [50].

In the review of the more popular simulation-based collision detection literature, it became apparent that the implementation of collision detection comprised two parts. Initially, algorithms to detect the bounds of the relevant components were established. This was followed by the actual detection of the intersection or proximity of these bounds. Numerous methods exist in terms of creating 3D models or boundaries for objects. Some of these include axis-aligned bounding boxes, oriented bounding boxes, approximation hierarchies based on S-bounds, spherical shells and octrees [51]. In most of these instances, the boundary of the solid object is determined through the iterative division of initial bounds until a criterion dictating the desired detail of the 3D model is met. An example of the implementation of oriented bounding boxes (OBB tree) is shown below:

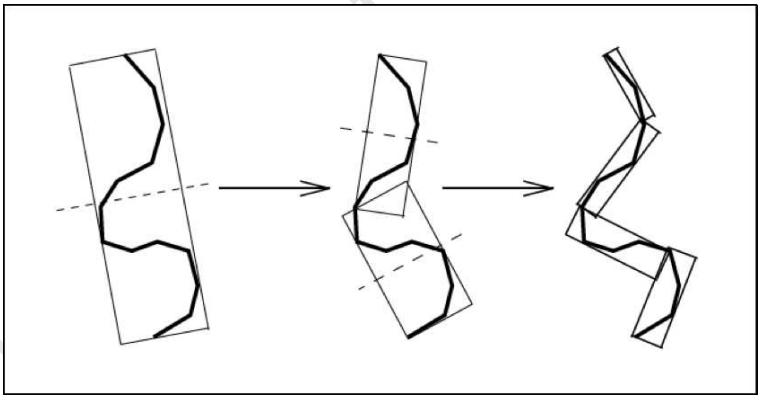

Figure 2.13: Building an OBB tree. Gottschalk, Lin and Manocha explain that the bounded polygon is recursively partitioned and the resulting groups bounded once again [52].

The literature shows that a very popular method to determine the intersection of the 3D box models, developed by the above mentioned algorithms, was through the use of the Separating Axis Theorem (SAT) [53, 54, 52, 51, 55]. Other intersection methods, including that of closest features computation as well as linear programming, exist although Gottschalk et. al. show the computation advantages of the separating axis test compared to the other methods [52]. This is supported by van den Bergen with the statement that "for collision detection of simple polytopes, such as line segments, triangles and boxes, the fastest results are achieved by applying the separating axis theorem" [53].

Once the required joint angles through the use of kinematic modelling and collision detection have been deduced,

this information needs to be communicated to the motors in question. Communication techniques used in the control of robotic manipulators are now looked at.

2.5 Communication Architectures

Numerous communication techniques have been implemented in the development of motor control architectures for robotic systems. An investigation into the development of previous USAR robots competing in the RoboCup Rescue Competition show that serial communications in the form of RS-232 and RS-485 were particularly popular communications methods.

One team to use the RS-232 standard was that of iRAP-PRO [56] for the communication of motor control information in the 2010 RoboCup Rescue competition. The team used a RS-232 serial server in order to communicate the relevant information to the numerous robotic modules. Another robotic manipulator to offer the use of either RS-232 or RS-485 is the commercially available ARM 5E developed by ESA CSIP. In the manufacturing of the robotic manipulator they offer the option of equipping the arm with either of the communication schemes for the control of the brushless DC motors operating the arm [57].

Three teams adopted RS-485 in the development of their USAR robots in the 2010 competition. These included C-Rescue from Japan as well as the Singaporean eeeBot for the control of their robotic manipulator [58, 59]. The third team to adopt RS-485 communications was that of Pasargad with its implementation of an "Internal Bus" to handle the communications to the different modules of the robot. They used a single master, multiple slaves architecture whereby the master board addresses each slave by their particular address and thereafter waits for a response from the slave. Once a response from the slave is received, communication packets are transferred. They stated that an added benefit of the RS-485 daisy-chain architecture is the ease of debugging and maintenance on the robot due to the fact that each system is independently removable [60]. A further robotic manipulator to use the RS-485 communications standard was that developed by Edwards in his design of a wheelchair-mounted robotic arm. He used RS-485 to communicate motor control information to each of the PIC-Servo motor controllers in the robotic arm [32].

Inter-Integrated Bus or I2C is another common communications scheme used in robotic applications. Like RS-485, I2C is daisy-chainable thus allowing for the use of only two wires in the connection of each of the communicating modules. Two studies implemented the use of I2C in the development of robotic manipulators. The first, like the research done by Edwards discussed above, involved the development of a wheelchair-mounted robotic arm. This group implemented an I2C bus in order to communicate with the motor control boards in the manipulator [61]. In a similar way, the second study focused on the development of a robotic arm to catch slugs. In this design, like the first, each uniquely addressed motor control board was connected to the I2C bus and was communicated with by a central master control system [62]. Although not implemented in a robotic manipulator, a study by Lauria, Piguet and Siegwart in the development of an autonomous wheeled climbing robot implemented the identical master - slave I2C bus motor control architecture as discussed above [63].

The final communication technique found implemented in the control of robotic manipulators was that of a Controller Area Network or CAN bus. Tindell, Burns and Wellings state that "Controller Area Network (CAN) is a well-

designed communications bus for sending and receiving short real-time control messages. The bus is designed to connect control systems over a small area (such as automobiles), operating in a noisy environment at speeds of up to 1Mbit/sec." [64]. One RoboCup Rescue team to make use of a CAN bus was SEU-RedSun from China. They implemented two CAN buses for the control of the body of their robot as well as the manipulator motor control as can be seen in Figure 2.14 [65].

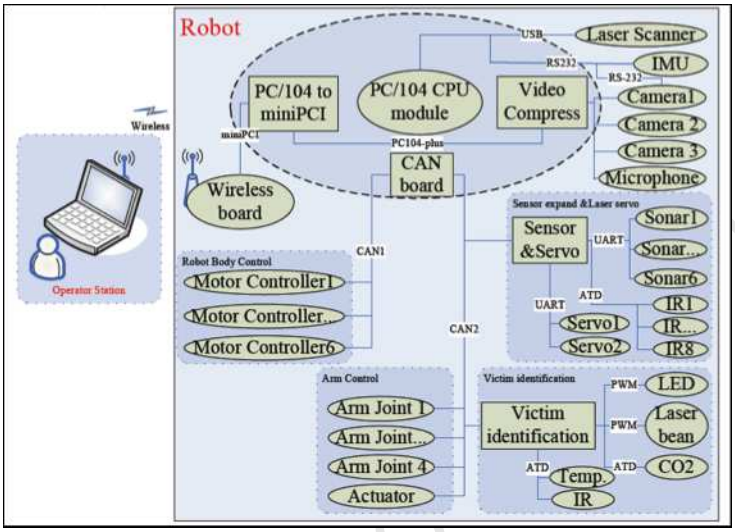

Figure 2.14: Motor control architecture of SEU-RedSun USAR robot. One can see the implementation of both CAN buses for motor control [65].

Asfour, Berns and Dillman were another group to use a CAN bus in their development of a humanoid robot. Their robot executed motor control on C-167 micro-controllers all of which were connected to a CAN bus in order to communicate with the high level processing PC [66].

Once the manipulator control information has been transmitted to the relevant motor control boards, the movement of the particular motors becomes important. The following section discusses different motor control implementations.

2.6 Motor Position Control

The mechanical design of the RATEL robotic arm included the use of quadrature encoders to monitor motor positions. Encoders are a very popular method of monitoring motor position and require decoding in order to extract the relevant motor position information. The C167 micro-controllers, described above, are an example of one solution whereas the use of FPGAs or CPLDs offers another. In the study of the electrical and mechanical properties of the PHANTOM haptic device, the team used a Lattice 3320 FPGA to decode the quadrature encoder signals and determine velocity information. These are based on the number of clock cycles to pass between encoder pulses [67]. The SEU-RedSun RoboCup team employed a CPLD to execute a quadrature decoding algorithm [65]. Edwards, on the other hand, used a PIC-SERVO controller board equipped with a small microprocessor in order to handle any quadrature decoding tasks [32].

The task of the controller devices mentioned above is to analyse all the relevant data and determine if the arm is, in fact, in the position or moving in the way dictated by the operator. If not, it must perform the appropriate control action [67]. This control action is referred to as the control law and has been the subject of much research in the field of robotic manipulators [23].

A popular control law is that of Proportional-Integral-Derivative (PID) control which was used by Glover et al. in their study of robotic teleoperation with haptic feedback [68] (see Figure 2.10). It was also used by Williamson in his study of robot arm control exploiting natural dynamics [69]. The control law was used in the development of the SAUVIM underwater vehicle [70] as well as in the control of the Hippocrate robotic arm [47]. Asfour et al. give a very good explanation for the popular use of PID control in robotic manipulators:

> The implementation of full dynamic control on a robot still remains a challenge to robot scientists and researchers today. It is known that the performance of a robot can be improved with the including of the robot dynamics into its controller. However, the complexity and, more importantly, the lack of knowledge about the dynamic parameters of the robot, lead robots to be controlled mostly by PID control, where the control is done independently for each joint [66].

One method to take the dynamic properties of the manipulator into account is sliding mode control. This is another frequently used control method in telerobotics as it offers robustness against uncertainties and can also deal with time delay; a pertinent problem in the task of telemanipulation [71]. Zhihong and Palaniswami implemented sliding mode control in their control scheme for rigid robotic manipulators with uncertain dynamics. They did this to provide stronger robustness as well as more effective output tracking error convergence in comparison to other feedback control schemes [72]. Su and Leung implemented the same technique although they combined it with an adaptive control scheme in order to provide effective trajectory control of robot manipulators [73]. Guo and Woo state that the classical sliding mode control, such as the methods discussed above, can suffer due to the possible chattering that may be caused by the large gain which needs to be applied to the control scheme. They therefore propose using fuzzy logic in the construction of the control input [74].

One study to use fuzzy logic in their control scheme was that of Asfour, Berns, and Dillmann in their control of the ARMAR humanoid robotic arm. They used a classical feedback position controller technique in which a fuzzy module selects a set of control parameters based on the configuration of the arm, as shown below. The control parameters were established through experimentation [66].

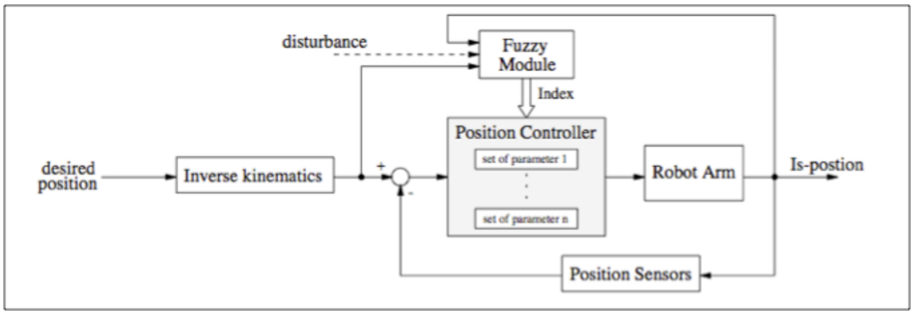

Figure 2.15: The control loop used by Asfour et al. in the control of their humanoid robot ARMAR. One can see the implementation of the fuzzy logic module dictating the control parameters to be used by the position controller [66].

This section has discussed the different aspects of manipulator control and the implementations used in order to achieve the desired effects. The subject of control is an extremely broad topic although the most relevant and widely used techniques have been highlighted above.

2.7 Concluding Remarks

This chapter has outlined the existing information pertinent to this project, namely teleoperation and manipulation, kinematic modelling, collision detection, communication architectures and motor position control. The information gathered allows for the development of the project specification in Chapter 3 which details the application of the above mentioned topics in the fulfillment of the requirements of the project.

Chapter 3

Specification

3.1 Functional Analysis

The knowledge gathered in Chapter 2 allows for the functional breakdown of the RATEL robotic arm. The control for the RATEL robotic manipulator must perform the following functions in order to adequately satisfy the seven requirements stipulated in Section 1.2. The function to be performed, the capability of that function and the requirement which that function satisfies are tabulated below. A functional analysis glossary provides greater detail for each term.

Function	Capability	Requirement Satisfied
Wireless Communications	Allow for the teleoperated control of the robotic arm from a remote control station.	1
Provide an intuitive control interface for the user	Allow for the easy and intuitive control of the arm	2+3
Collision Detection Procedures	Prevent the arm from colliding with elements of the robot	4
Calibration Procedures	Allow for the calibration of the arm	5
Motor Position Control	Allow for the control and feedback of the position of each motor in the arm	2+4+5+6
Arm Communications	Allow for the communication of motor position information to and from motor control elements in the arm	2+3+4+5+6
Kinematic Model	Method to describe how the user is to control each link of the arm	2+6
Electronic Mounts	Allow for the mounting of all the electronic components inside the arm	7

Table 3.1: Functional analysis for the control of the RATEL robotic manipulator.

Wireless Communications needed to be developed in order to allow a user to control the robotic arm from a remote location.

An **intuitive control interface** allows for the easy control of the manipulator. This scope covers the device used to control the arm as well as the interface which the user interacts with while controlling the arm, wrist and gripper.

Collision detection procedures had to be implemented to prevent the arm from colliding with elements of the robot.

Calibration procedures allowed for the zeroing of the motor position values in the arm. This provided the known relative position required of encoder based motor position control.

Motor position control forms the basis upon which the user controls the robotic arm. This critical function provided the platform upon which collision detection, calibration, the ease of use of the arm as well as the stability of the motion of the arm was based.

Motor position control would not be possible without **Arm Communications**. A fast, stable communications scheme needed to be developed between the motor control elements of the robotic arm in order to fulfill the requirements specified.

A **Kinematic model** of the arm needed to be developed in order to translate the input commands from the user into motor positions upon the robotic manipulator.

Finally, **Electronic mounts** provided the functionality to secure all the necessary electronic components inside the arm.

3.2 Project Specification

The above functional analysis leads to this project specification.

3.2.1 Scope

This specification covers the design and implementation of the control for the 6 DOF RATEL robotic manipulator designed by Peter Henson at RARL at UCT.

3.2.2 Characteristics

Kinematic modelling

- An accurate kinematic model should be developed to translate the input commands from the user to motor position information.

Motor Position Control

- Accurate, stable motor position control must be implemented on the robotic arm to ensure ease of use as well as provide the data necessary for collision detection and arm calibration.

Communications

- A wireless communications scheme should be developed to communicate the user commands from the control station to the robotic arm. This must be stable to ensure safe operation of the robot as well as fast enough to allow for real time, easy control of the robotic arm.
- An internal arm communications scheme must developed to communication motor position and control information to and from the control devices of the robotic arm including the wrist/gripper assembly. As specified above, this should be stable to ensure uninterrupted control of the arm as well as fast enough to provide responsive control of the manipulator.

Electronics

- Motor control electronics must be developed to perform the required motor position control in the arm.
- Proximity sensors should be placed on the arm to allow for the calibration of each motor in order to zero the relative motor position. Calibration algorithms would thereafter need to be developed.
- Electronic mounts must be developed to house the necessary control electronics internally.

User Interface

- Collision detection procedures must be implemented to detect and prevent any collisions between the arm and elements of the RATEL robot.
- An easy and intuitive method for the user to control the robotic arm, as well as wrist and gripper, must be used.
- All the information required to control the robot, including the arm geometry as well as any collision detection warnings, should be intuitively presented to the user.

These specifications describe a control system for the RATEL robotic arm that fulfills all of the requirements stipulated in Section 1.2. The implementation of these specifications is now discussed beginning with the development of the kinematic model for the robotic arm in Chapter 4.

Chapter 4

Kinematic Modelling

4.1 Introduction

Kinematic modelling involves the transformation of x, y and z coordinates of the end effector of a robotic manipulator to joint space or joint angles [40]. Inverse kinematics was used as a kinematic modelling technique for the RATEL robotic arm as it offered the user intuitive control over the movements of the manipulator. The process of inverse kinematics means determining the joint angles required in order to attain a desired end effector position. The user would therefore, through the use of an input device, dictate the desired position of the end of the arm. The robotic arm, through the use of inverse kinematic mathematical models, would then need to assume the correct orientation in order to provide the correct end effector position dictated by the user. These inverse kinematic mathematical models will now be discussed.

The RATEL robotic arm, excluding the 2 DOF sensor payload, is a 4 DOF robotic manipulator, represented by Figure 4.1.

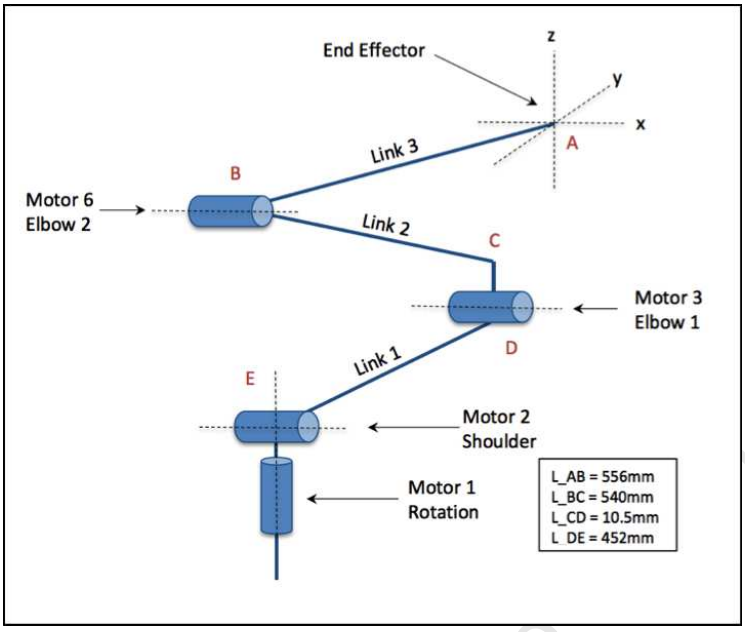

Figure 4.1: Simplified representation of RATEL robotic manipulator.

As the links in the arm are all co-planar, the inverse kinematic equations can be modeled as a 2D problem. The DOFs of the arm are low and thus a geometrical approach (as used by Tunwannarux and Tunwannarux [24]) was adopted to solve the inverse kinematics. Initially, the necessary position of each of the joints was determined. This was followed by the calculation of the relevant joint angles[1].

If the end effector position was dictated only by x and z axis criteria an infinite number of solutions would exist. An additional input was therefore provided to the user to dictate the vector at which the final link should act at a normal to, denoted by $\vec{N}$. This reduces the number of solutions to the desired position to two, as shown in Figure 4.2.

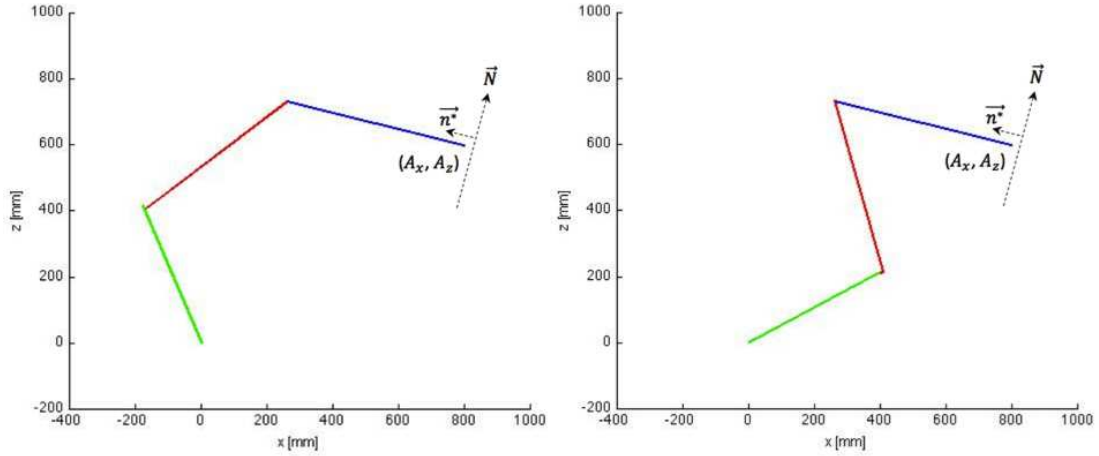

Figure 4.2: Matlab simulation showing the two orientations of the robotic arm resulting in the same desired end effector position and angle, shown at position (800, 600) with $\vec{N}$= (100, 25).

[1]The matlab code for the inverse kinematic calculations can be found on the DVD provided

4.2 Geometric Calculations

It must be noted that the exact distances between the rotation axes of each of the links was used for the inverse kinematic calculations in order to ensure absolute accuracy. Another factor to note was that the centre of the second link was not co-axial with the point of rotation as can be seen in Figure 4.3.

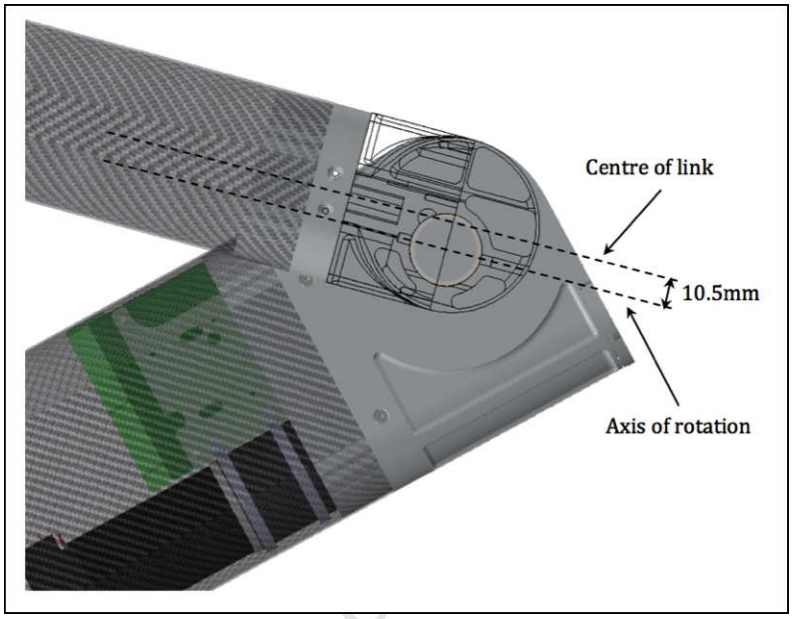

Figure 4.3: The second link axis of rotation compared with centre line.

A method to overcome this in the kinematic modelling was to offset the centre of the second link by 10.5mm. This can be seen, exaggerated, in Figure 4.4.

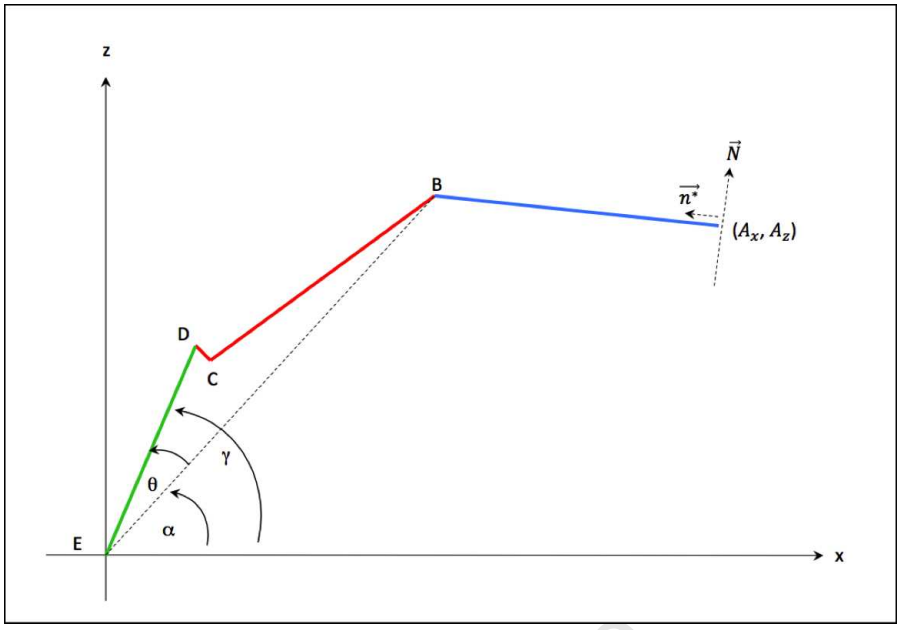

Figure 4.4: The inverse kinematic model of the RATEL robotic manipulator

With an input for A_{xz}, where x and z denote the desired coordinates of the end effector, and the vector upon which the last link should act upon, $\vec{N}$, dictated by the user we can calculate the coordinates of point B_{xz}

$$B_{xz} = A_{xz} + (\vec{n^*} \times |AB|) \tag{4.1}$$

where $\vec{n^*}$ is the norm unit vector of $\vec{N}$. The possible configurations of the joints was taken into consideration when performing the calculations, as can be seen in Figure 4.7, in order to ensure accuracy.

The known lengths of links DE,BE and BD can then be used to calculate the angle θ by means of the cos rule

$$\theta = \frac{arccos(|DE|^2 + |BE|^2 - |BD|^2)}{2 \times |BE| \times |DE|} \tag{4.2}$$

and angle α

$$\alpha = arcsin(\frac{B_z}{|BE|}) \tag{4.3}$$

Now, depending on the orientation of the arm, specifically the location of B with respect to the two possible locations of D, γ could either equal $\alpha + \theta$ or $\alpha - \theta$. This specific calculation provides the two solutions to the inverse kinematic problem, or in other terms, redundancy (see Figure 4.2). As stated by Tevatia and Schaal "...redundancy, although possibly challenging, is often not disadvantageous to the task at hand as the excess of solutions can be used to optimize certain criteria such as motor current draw or force constraints" [44]. It is upon this basis that the solution to γ which offered the smallest angle was always selected as the preferred solution. This means, graphically,

selecting the right hand option over the left in terms of Figure 4.2. This is to minimise the amount of travel that each link of the robotic manipulator would have to move, from the stowed default position, in order to assume the desired position dictated by the user. This prevented large movements of the arm in assuming a desired position and thus aided in the real time control of the arm.

Having calculated the value of γ, the position of D could be determined by a simple trigonometric calculation

$$D_{xz} = E_{xz} + |DE|cos\gamma + |DE|sin\gamma \tag{4.4}$$

The position of C from B was known to be at an angle of $arctan(\frac{|CD|}{|BC|})$ as BCD forms a right-angled triangle by nature of the mechanical design. This was used to shift the position of D by 10.5mm in order to determine the coordinates of C, the final unknown position in the arm.

4.3 Angle Determination

Having deduced the necessary joint positions in space, the joint angles required to attain these positions were calculated. These were then to sent to the motor controllers. In traversing from the calculated joint positions to actual joint angles on the robotic arm, the direction of motion of each of the motors in the arm needed to be taken into consideration in order to determine the correct magnitude of joint angle. Figure 4.5 illustrates the direction of motion of increasing values of motor position.

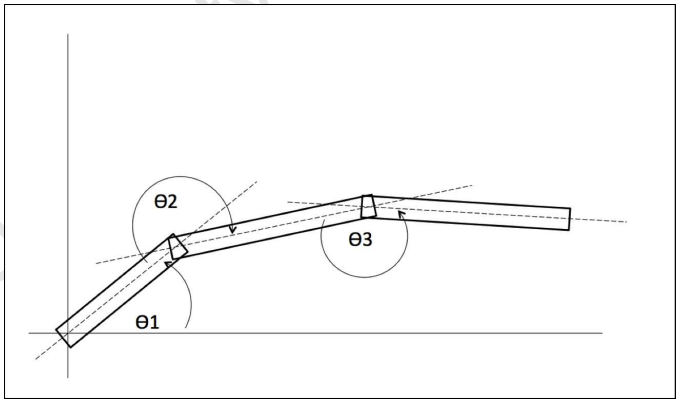

Figure 4.5: Direction of motion of increasing values of joint angle

Therefore, precautions needed to be taken in order to ensure that the correct angle was determined. In terms of $\theta 2$ above, the exterior value rather than the interior value had to be calculated due to the direction of motion of the motor. This was done by analysing the joint positions in space and selecting the appropriate angle magnitudes. This is illustrated by Figure 4.6.

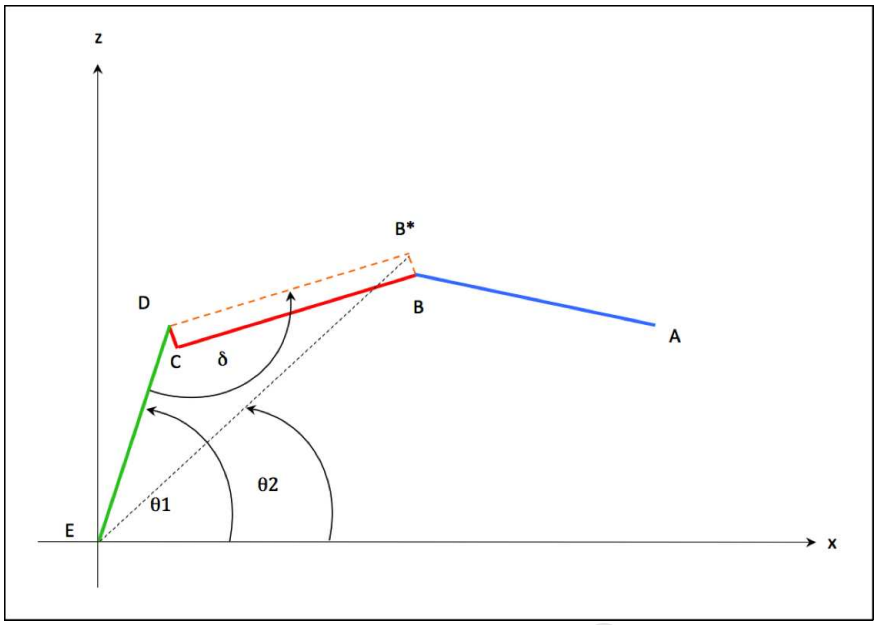

Figure 4.6: Correct angle determination for RATEL .

The angle of rotation that joint D needed to assume, due to the 10.5mm shift in axis position of joint B, was in fact located at a point 10.5mm up from B, B^*. In the instance shown in Figure 4.6, the determination of angle δ using the cos rule applied to triangle EDB* would result in the interior angle being calculated. Thus, $360 - \delta$ would give the required exterior joint angle as dictated by Figure 4.5. This is, however, not the case in certain other configurations, as shown in Figure 4.7, where the cos rule applied to triangle EDB* may automatically yield the desired exterior angle. As joint ED cannot fall below the horizontal due to the rotation constraint of E, four configurations for triangle EDB* exist:

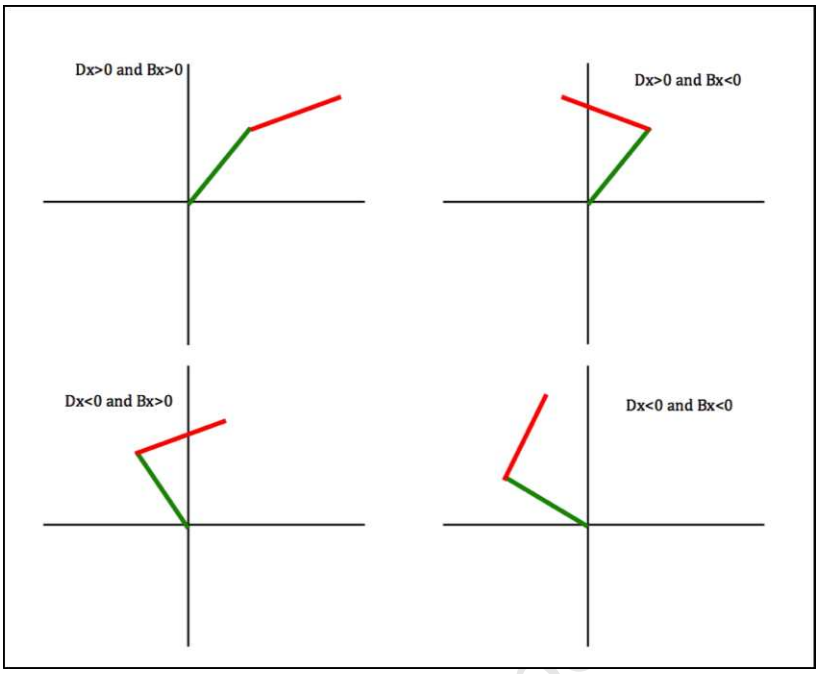

Figure 4.7: The four possible configurations of Links 1 and 2.

The relationship between $\theta 1$ and $\theta 2$ (see Figure 4.6) was therefore analysed for each of the possible configurations in order to determine the correct value of rotation for joint D or Elbow1 as determined by the cos rule.

The identical scenario was present for the combination of the second two links of the arm; joint angle B or geometrically, triangle CBA. The axis of analysis was shifted to joint C and the different combinations of CBA were analysed. However, unlike joint E which has a limit at the horizontal, joint D can rotate into all four axes. CBA can therefore assume a total of ten different poses in a combination of the Z and X axes as opposed to just the four described above. The same method as used above, utilising the magnitude of the angles from the origin to the joints of interest, was used to determine the correct magnitude of CBA for all ten possible scenarios. The Matlab code on the DVD provides a supplement to this section.

4.4 Simulation and Enhancement

With these calculations, the necessary joint angles, or motor positions, in order to assume the desired end effector position were determined. Figure 4.8 shows a simulation of the arm movement maintaining a constant z coordinate with an increasing x coordinate:

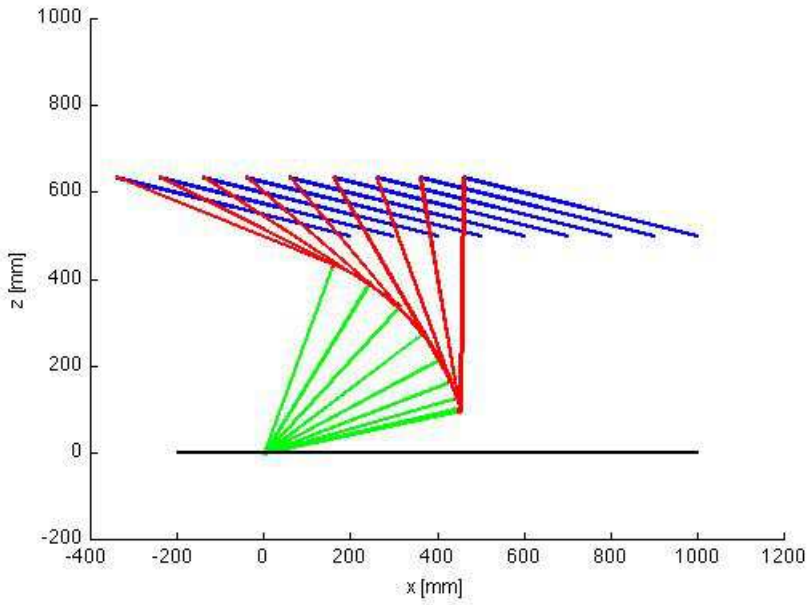

Figure 4.8: Arm movement dictated by inverse kinematic calculations.

After implementing this inverse kinematic technique on the robotic arm it became apparent that the joint limit of the first link at the horizontal (to avoid a collision with the base of the robot) severely restricted the ease of movement of the arm. Any desired end effector position in which the first link was required to assume a position below the horizontal became illegal. Thus, moving the end effector in an increasing x axis direction, as shown above, became very difficult once the first link reached its limit (shown in black in Figure 4.8). One option to overcome this was to allow the user to switch to the previously excluded secondary inverse kinematic solution (see Figure 4.2) which assumed a much larger first link angle. The increased probability of collisions with the rear of the robot together with the lack of intuitiveness of the user having to physically select a new method of operation rendered this solution undesirable. It was decided that a far more practical solution was to lock the first link in place at the horizontal and apply a new inverse kinematic calculation to the remaining two links of the arm.

By locking the first link in place and hence removing a degree of freedom, the ability to dictate the angle at which the last link acts upon the desired end effector position is lost. Only a single solution exists for a 2 DOF manipulator to assume a desired position. Inverse kinematic calculations, identical to those explained above, were therefore performed on the resulting two link manipulator in order to determine the required joint angles.

A simulation of the arm movement, using the new inverse kinematic calculations, once again maintaining a constant z coordinate with an increasing x coordinate is shown in Figure 4.9.

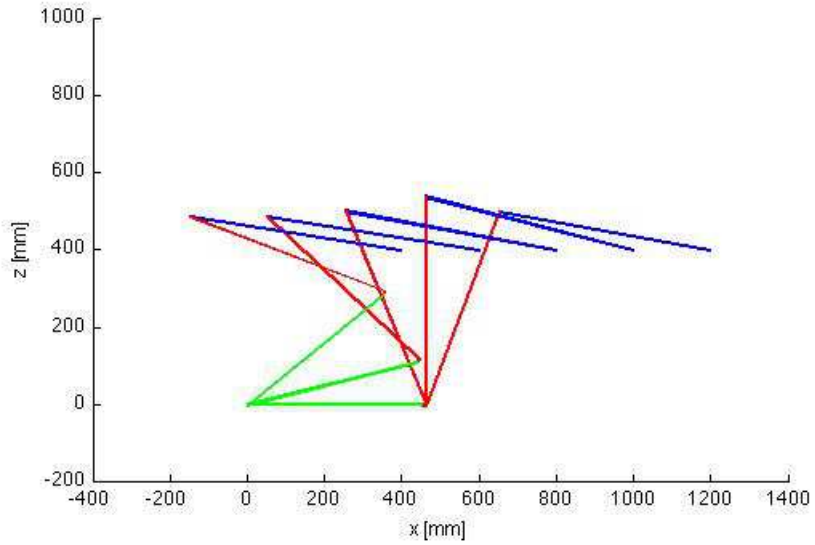

Figure 4.9: Simulation of arm movement with improved inverse kinematic calculations. The angle of the last link (blue) changing in order to assume the desired end effector position once the first link (green) becomes fixed at the horizontal can be seen.

4.5 Summary

These inverse kinematic calculations allowed for easy, intuitive control of the robotic arm. The testing of the manipulator, described in Chapter 9, showed an end effector accuracy of 3.5mm thus illustrating the accuracy of the inverse kinematic calculations. The necessary motor position control in order to rotate the motors to the calculated desired position will now be discussed.

Chapter 5

Motor Position Control

5.1 Introduction

The implementation of inverse kinematics required accurate position control of the motors in the robotic arm. This was to ensure that the end effector of the robotic arm assumed the correct position as dictated by the inverse kinematic equations. Each of the motors used in the arm was equipped with an encoder (refer to the appended DVD for data sheets) in order to monitor the relative position of the motor through the implementation of quadrature decoding.

The motor assembly properties of the arm are shown in Figure 5.1 together with Table 5.1.

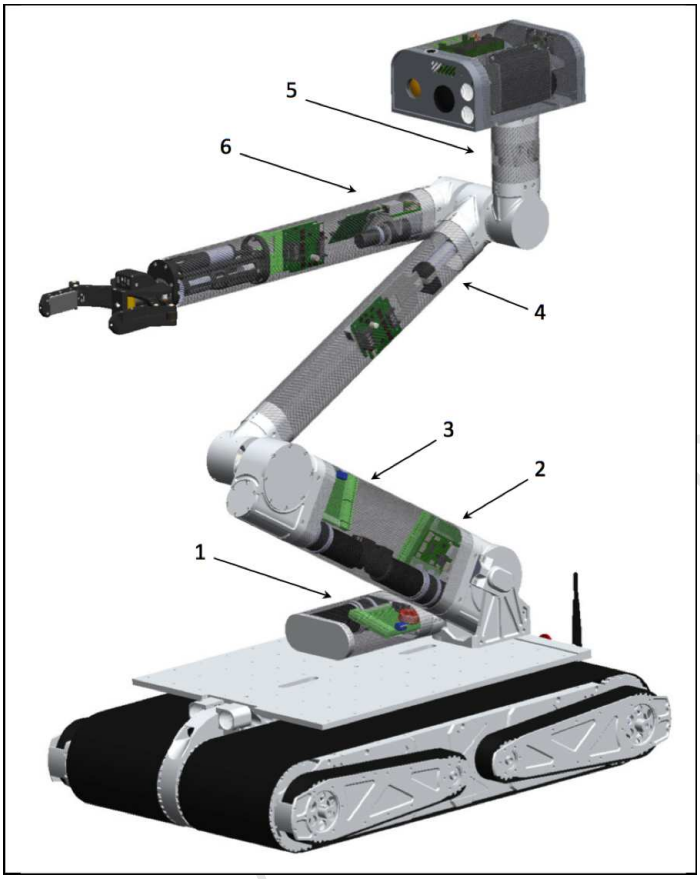

Figure 5.1: Motor numberings of the RATEL manipulator.

Motor Number	Motor	Motor Gearbox Ratio	Encoder	Encoder Counts Per Turn	Motor Controller	Joint Gearing	Joint Gear Ratio
1: Rotation	Maxon EC40 120W	73:1	HEDS 5540	500	DECV 50/5	Worm	36:1
2: Shoulder	Maxon EC40 120W	91:1	HEDS 5540	500	DECV 50/5	Worm	36:1
3: Elbow 1	Maxon EC40 120W	73:1	HEDS 5540	500	DECV 50/5	Worm	36:1
4: Tilt	Maxon EC22 20W	531:1	HEDS 5540	500	DEC 24/3	Helical	2:1
5: Pan	Maxon EC22 20W	531:1	HEDS 5540	500	DEC 24/3	Helical	1:1
6: Elbow 2	Maxon EC22 20W	190:1	MR	512	DEC 24/3	Hypoid	45:1

Table 5.1: Motor assembly properties of the RATEL manipulator.

This chapter discusses the implementation of the required motor position control on each of these joints by first looking at the quadrature decoding used on the FPGA-Coldfire motor control boards. It then looks at the development of the motor position control used on the LM3S8962 motor control boards.

5.2 Quadrature Decoding on the FPGA-Coldfire

Quadrature decoding involves monitoring the signal (Signal A, Signal B and the Index line) output from a quadrature encoder attached to a motor. This is to determine and monitor the position of the motor shaft as well as motor speed. Signals A and B are pulse trains which are phase shifted by 90 degrees, shown in Figure 5.2; A leading B or B leading A depending on the direction of rotation of the motor. By analysing these pulse trains, the number of pulses as well as which signal leads the other, the motor shaft position as well as motor speed can be analysed.

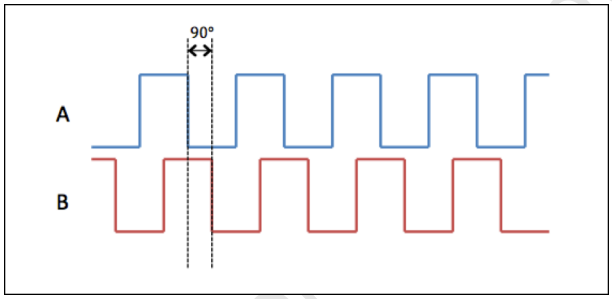

Figure 5.2: Diagram of quadrature encoder output.

5.2.1 Monitoring of Encoder Pulses

The initial device developed to perform quadrature decoding and motor position control in the RATEL manipulator was the FPGA - Coldfire board, shown in Figure 5.3, developed by Tracy Booysen in the RARL lab prior to my arrival. The ProAsic3 A3P060VQ100 was the FPGA used on the board. As the signals in the quadrature encoding pulse train were likely to be in the order of up to 62500 counts per second, the high speed FPGA was used to perform the quadrature decoding. The FPGA would be able to effectively monitor the encoder pulses since the programming (VHDL) on it is realised in high speed hardware connections. The FPGA would then be responsible for tracking the encoder pulses, incrementing or decrementing the motor position count value and index value of the motor shaft, resetting this value if required as well as communicating this information to the Coldfire micro-processor for control of the motor position.

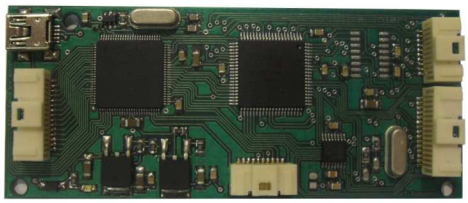

Figure 5.3: FPGA - Coldfire motor control board.

In order to track the pulses derived from the encoder it was decided to monitor Signal A from the encoder as though it were a clock input to the FPGA. On each rising edge of Signal A, Signal B was checked. If Signal B was low the counter was incremented. Alternatively, if Signal B was high, the motor counter was decremented. This simple algorithm was initially coded and simulated in Quartus (shown in Figure 5.4) and thereafter implemented upon the ProAsic3 FPGA (VHDL code appended in DVD provided). The Index pulse (which only pulses once per motor shaft revolution, in both clockwise and anti-clockwise motion) was treated in a similar fashion as a clock input to the FPGA. Upon each positive pulse the index counter was incremented.

Figure 5.4: Quadrature decoding simulation produced in Quartus. "QuadA" and "QuadB" are the simulated input lines to the FPGA. The "Position" variable is the motor position determined by the code upon the FPGA. Note the increment and decrement of the motor position upon a change in motor direction.

5.2.2 Communications with Coldfire

A particularly challenging task of this application was to communicate the relevant motor positions from the FPGA to the Coldfire micro-processor in order to execute motor position control. Two communication options existed to transmit the encoder count information from the FPGA to the Coldfire. The first was via serial communications utilising a RS-232 communications standard upon a MAX3232 IC on the FPGA-Coldfire board. The second was to utilise the physical pin connections between the FPGA and the Coldfire (thirteen in total, refer to the schematic on DVD) and transmit the data in a parallel fashion. Due to the superior speed of parallel communications the latter option was used.

Up to two motors were to be controlled by a single control board (see the design architecture in Figure 6.1). It was decided to utilise the eight pins of Port A of the Coldfire which are directly linked to the FPGA as the data lines (data bus) for communication. As a total of 24 bits needed to be transferred for each motor (16 for the motor position as well as 8 for the index count), the data was sent in packets of eight bits at a time. The remaining five lines were then used as two reset lines in order to reset the encoder and index counts and three status lines to indicate when a data transfer may proceed or has been completed. The three status lines used to indicate to the FPGA what data to place on the eight communication lines were simply named Pin1, Pin2 and Pin3. Pin1 was used as an indicator to the FPGA that the next packet of data was to be placed on the data bus. Pin2 was used to indicate to the FPGA which of the two motors data was to be transferred while Pin3 was the indicator used by the FPGA to inform the Coldfire that data had been loaded onto the data bus. The sequence of data transfer between the FPGA and Coldfire is described below.

1. Coldfire initiates communications by setting Pin1 high while simultaneously setting Pin2 high or low depending on which motor data is to be read. Pin2 low = motor 1. Pin2 high = motor 2.
2. FPGA sets Pin3 high indicating it has received first instruction. No data change upon data lines.
3. Coldfire sets Pin1 high.
4. FPGA applies packet 1 of 3 upon data line, updates internal variables as indication that 1st byte been sent. Sets Pin3 high.
5. Coldfire reads data, sets Pin1 high.
6. FPGA applies packet 2 of 3 upon data line, updates internal variables as indication that 2nd byte been sent. Sets Pin3 high.
7. Coldfire reads data, sets Pin1 high.
8. FPGA applies packet 3 of 3 upon data line, updates internal variables as indication that 3rd byte been sent. Sets Pin3 high.
9. Coldfire reads data.
10. Completion of data transfer.

The communications sequence was synchronised with the external oscillator connected to the FPGA. Upon each rising edge of the oscillator clock the three status lines were analysed and the data bus was updated accordingly. The simulation of this sequence in Quartus is shown in Figure 5.5. The red numbers correlate to the sequence described above.

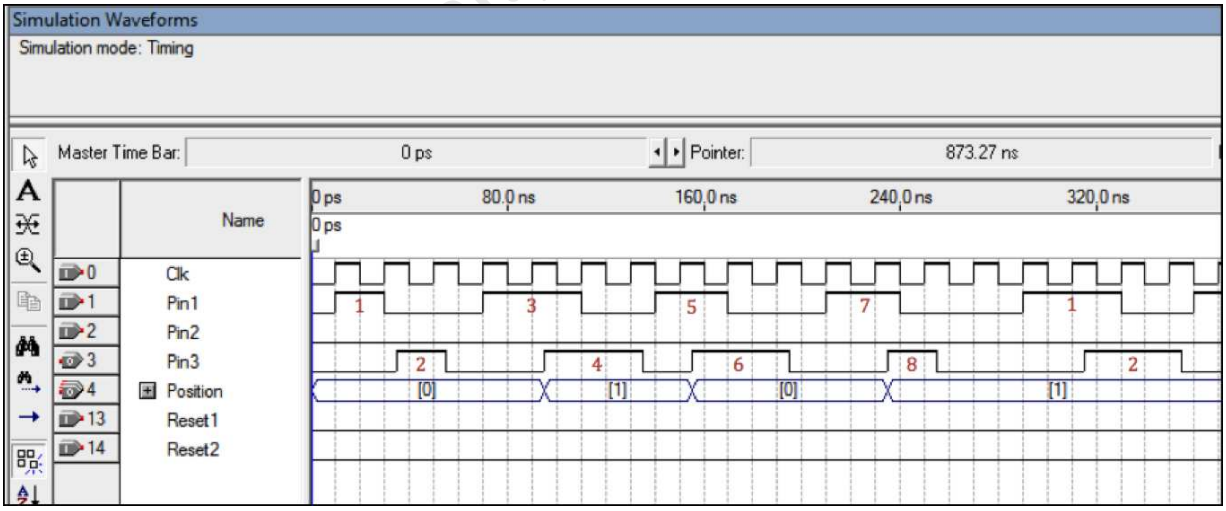

Figure 5.5: The FPGA response to simulated instructions as would be received from the Coldfire micro-processor according to the communications scheme described above. The CLK , Pin1, Pin2 and Reset lines are signals generated to simulate inputs to the FPGA device from the Coldfire. Pin3 and the Position lines shown above are the simulated output from the FPGA. For testing, the value of the motor1 position was set to 1 with an index value of 1.

This communications method, in practice upon the FPGA-Coldfire board, however failed to produce a reliable communications scheme. After numerous tests it became apparent that utilising a repetitive signal from the Coldfire

to the FPGA (setting of Pin1) and implementing variable flags in the FPGA coding to keep track of what data was to be sent (see flow diagram in Figure 5.6) resulted in the FPGA often missing flags. This resulted in incorrect data transmission, or invalid combinations of flags being set thus creating communication lock-ups. It became apparent that the use of variables on the FPGA was an unreliable practice.

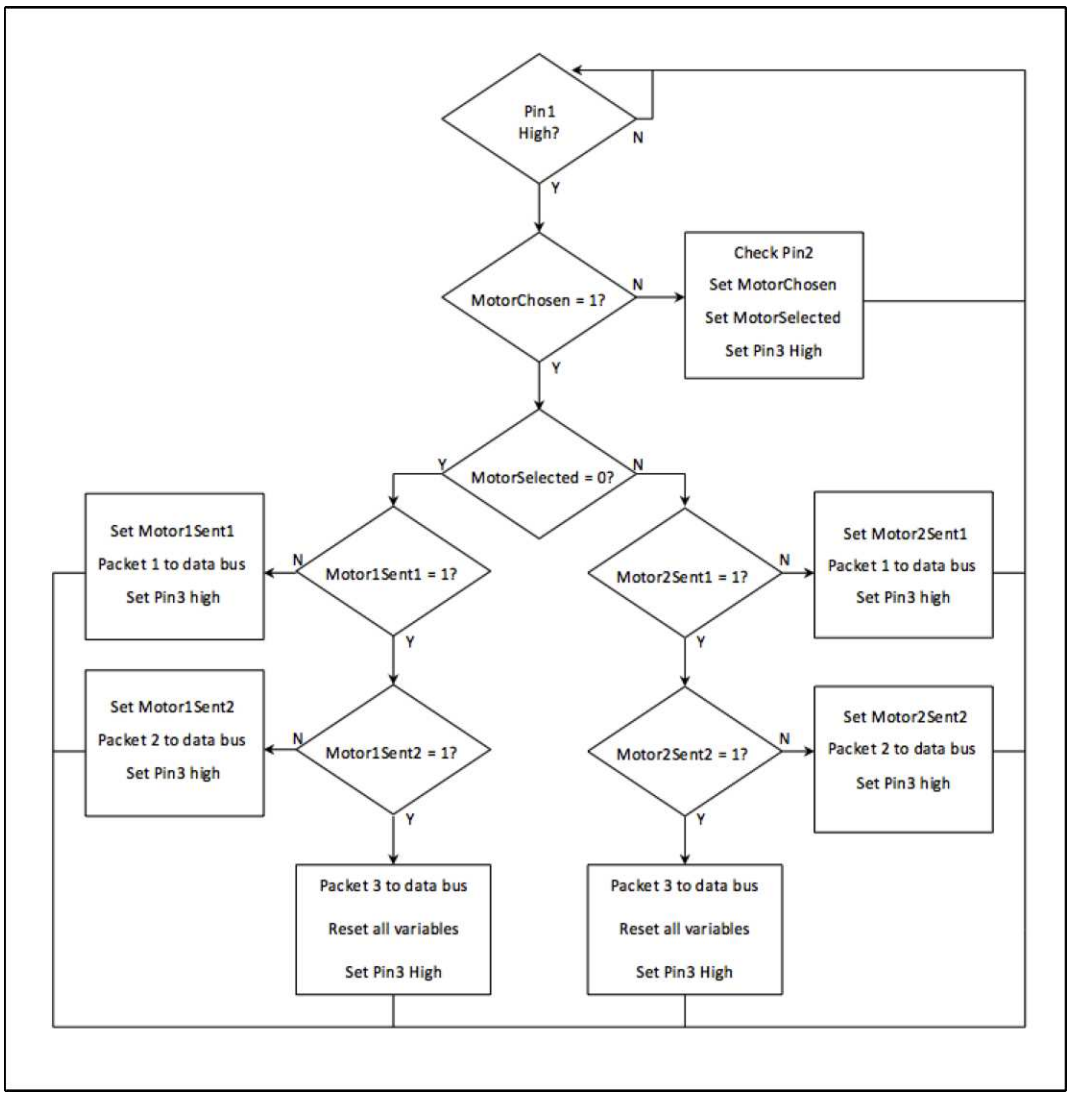

Figure 5.6: Flow diagram of FPGA-Coldfire communications scheme.

In order to avoid the use of variables on the FPGA a new communications system needed to be established. It was decided to once again use the lines directly linked to the Coldfire although in a direct command structure; the Coldfire would tell the FPGA exactly what packet of data to send using specific instructions. This was achieved by reducing the number of data lines from the eight in the previous method to four. This then allowed for four command lines, two reset lines and two status lines. The four command lines were used to indicate to the FPGA, in binary, which four bit packet of data to send[1]. The status lines served to inform each device when data was

[1]The motor position data was split into four bit packets. Packet 1 contained the first four bits of motor position 1, packet 2 contained the second four bits of motor position 1 and so on. The Coldfire would then directly request these four bit packets from the FPGA.

available to read on either the data lines, for the Coldfire, or the command lines for the FPGA. The reset lines were used to initiate a motor position reset. This method completely avoided the use of variables on the FPGA device. The step-by-step implementation of this communication scheme is shown below:

1. The Coldfire implements the binary 'address' on the four data lines. It then sets the "Set" pin high to indicate to the FPGA that there is data to be read. The Coldfire then waits for the FPGA to respond. An adequate wait time was determined to be ten cycles of the processor clock.
2. The FPGA, upon detecting "Set" being high, reads the four command lines to determine which packet of data to load onto the four data lines. It then implements the respective four bits of motor position information onto the data lines.
3. The Coldfire, having waited for the FPGA to respond, reads the data upon the data lines and stores the four bits.
4. This process then continues until the Coldfire has received all the motor position data. The Coldfire then concatenates all the received data into bytes of motor position information.

This communications method greatly reduced the number of variables needed to be stored on the FPGA. The FPGA needed only to check the status of the input lines and act accordingly. The simulation of this procedure upon the FPGA is shown below:

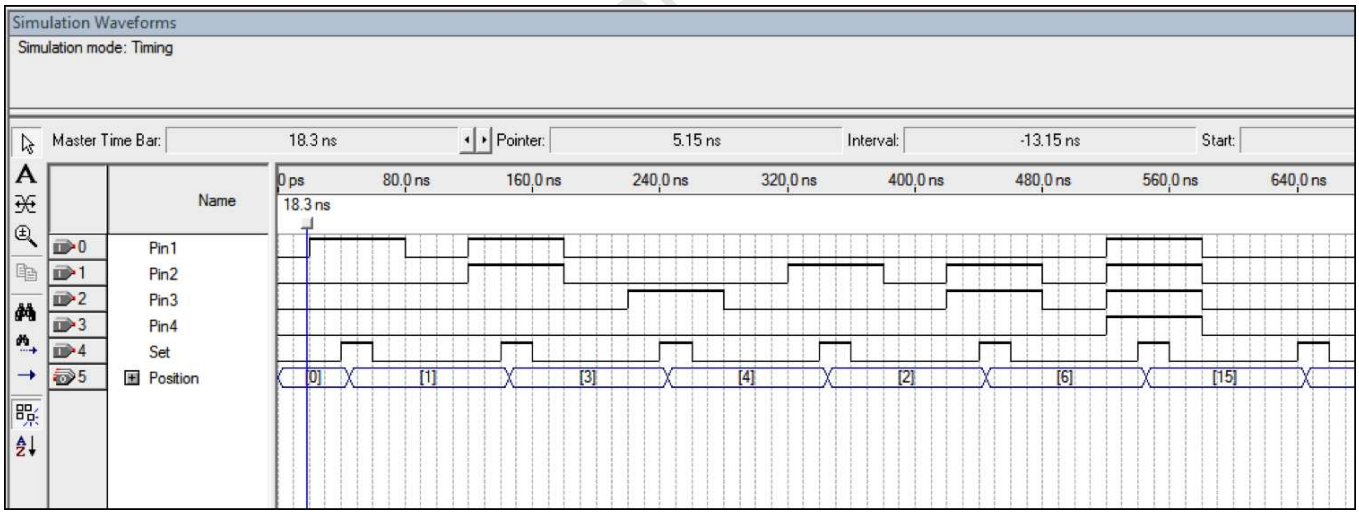

Figure 5.7: FPGA simulation of secondary communications scheme. Pin1 through to Pin4 and Set were manually defined prior to the simulation in order to mimic the commands that the FPGA would receive from the Coldfire. For the benefit of the simulation, the motor position that the FPGA implemented upon the data lines was coded to be equal to the binary address that the Coldfire dictated. Hence, if the Coldfire board requested the four bits in the sixth position, the number six would be returned as the "motor position" bits. One can see a successful simulation of this above.

The above code was implemented on the FPGA using Libero IDE software with the equivalent communications algorithm coded upon the Coldfire microprocessor using Freescale CodeWarrior[2]. The encoder of a motor was connected to the relevant ports on the FPGA and the motor shaft manually rotated in order to test the quadrature

[2]The code for the applications can be found on the DVD provided

decoding and communications scheme to the Coldfire. The output of the motor position in CodeWarrior, shown below, indicates the successful implementation of a quadrature decoder upon the FPGA-Coldfire board.

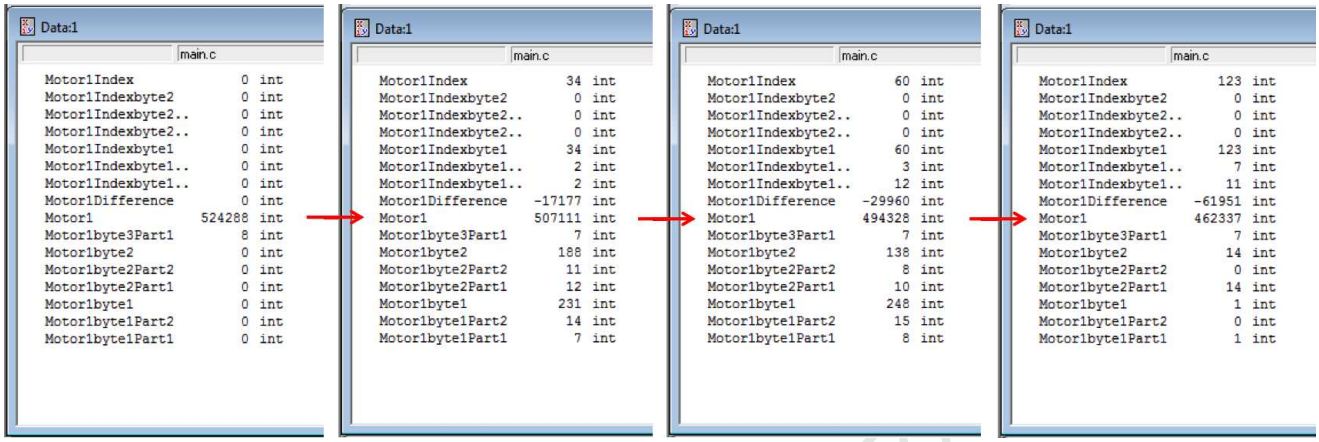

Figure 5.8: Implementation of quadrature decoding upon FPGA-Coldfire motor control board. "Motor1" was set at a default value of 524288 in order to avoid the transmission of signed integers. Any motor position below 524288 was considered "negative" whereas a motor position above this default was considered "positive". One can see the value of "Motor1" decreasing with the rotation of the motor shaft.

The amount of time necessary to transmit the motor position information from the FPGA to the Coldfire was tested in order to determine if the communication speed was fast enough to perform accurate position control on a motor. A 1ms timer interrupt was included in the above code and a counter was incremented after each transmission of both the Motor1 and Motor2 information. At each 1ms interval the counter value was stored in an array for analysis. As the code repeatedly read the motor position information from the FPGA an accurate transmission duration was acquired. The image below of the counter array indicates that 12 transmissions occurred between each 1ms interrupt. This equates to a transmission duration of 0.083ms or 83ns.

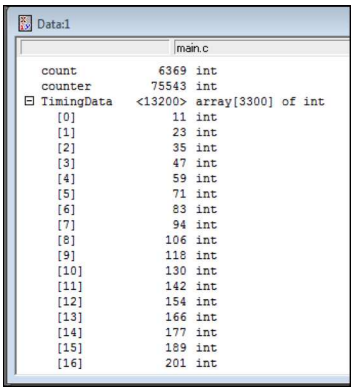

Figure 5.9: Transmission speed of motor position data between FPGA and Coldfire.

Motor control algorithms were implemented on the FPGA-Coldfire board in order to move a motor to a desired position utilising the above quadrature decoding method. Basic PID control was investigated utilising the same methods as described in the following section.

For the purposes of standardisation and ease of motor control implementation RARL changed direction in terms of the micro-processor to be used on the robot. The FPGA-Coldfire board was replaced with the LabVIEW capable LM3S8962 developed by Texas Instruments. The focus of the project therefore moved towards motor control on the LM3S8962 motor control board developed in the RARL lab (expanded on in Chapter 7).

5.3 Motor Control on the LM3S8962

A significant benefit of the LM3S8962 was the on-board quadrature decoder. The LM3S8962 utilises a 4-quad decoder therefore each encoder count is multiplied by 4 (highlighted later in Table 9.1). Focus was therefore directed at accurate position control of the motors in the robotic arm. Due to its overwhelming popularity in the literature, PID control was investigated once the arm was assembled and communications to each of the motor control boards was operational (discussed in Chapter 6).

The control of a single motor in the robotic arm can be represented by the following control loop:

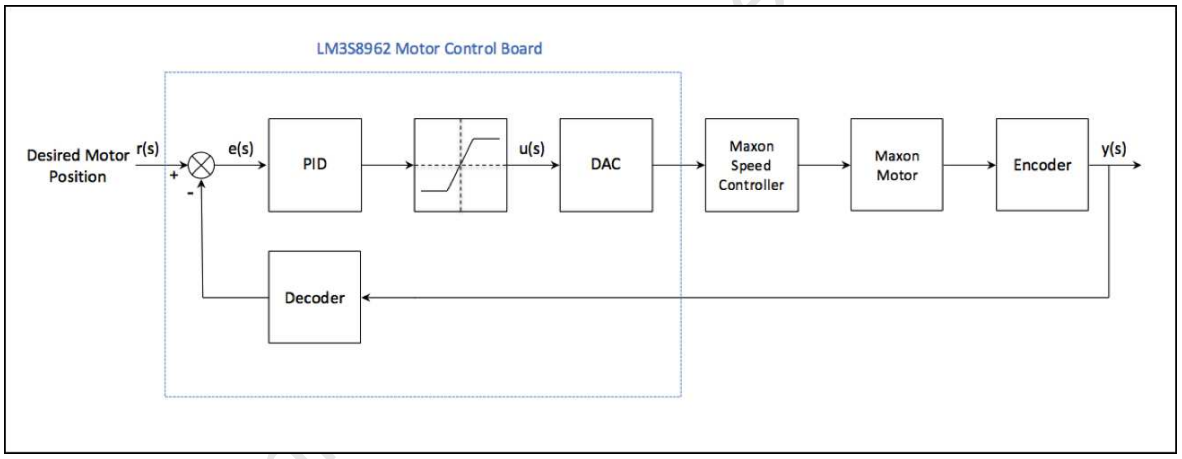

Figure 5.10: Motor position control loop.

In order to determine an effective PID controller and hence implement accurate position control of the motors in the arm it was necessary to attain position transfer functions of each of the motors. This was achieved by inducing a speed step input to each of the motors and analysing the encoder values along with elapsed time. This data was saved to file for manipulation in Microsoft Excel and Matlab.

5.3.1 Motors 1, 2 and 3

Shown below are the Excel plots of Motor 1's speed and position in an anti-clockwise direction after a step input.

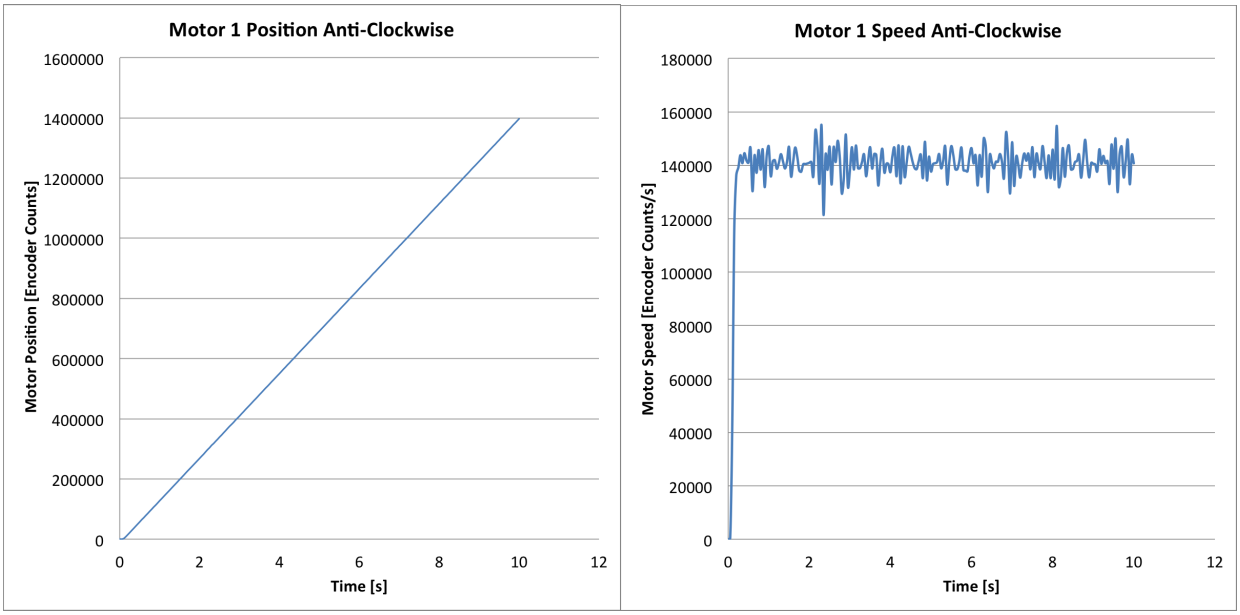

Figure 5.11: Motor 1 speed and position step response. The motor position, in open loop, responds as expected by continuously increasing. The speed shows a curved response as dictated by the Maxon DECV 50/5 speed controller.

The Maxon speed controllers did not allow for direct control of the motor voltage, only the speed setting of the non-linear speed controller. The size of the step input was dictated by the voltage output of the DAC upon the LM3S8962 as well as the output settings of the Maxon speed controllers. Preliminary tests had shown that the minimum motor position stopping overshoot was achieved when the DIP switches on the DECV 50/5 speed controllers were set to the lowest speed (1000-7500 rpm for 1 pole pair motors as being used) together with the highest gain. These settings were therefore adopted for the use of the arm. The maximum speed of the speed controllers (5V which equates to 7500 rpm) proved to be unnecessarily fast for the use of the arm therefore an upper limit of 2.5V was implemented on the DAC output. Since a 12 bit DAC was being used, a step of 2.5V was achieved with a decimal value of 2050. This value, 2050, was used as the step input to the step response analysis below.

The response of the motor speed can be approximated as a First Order System by the equation:

$$g(s) = \frac{A}{1 + sT} \tag{5.1}$$

The gain value, A [Encoder counts per second per bit of DAC output], was calculated to be 69.1054 (since $141666 \div 2050 = 69.1054$) with a 2/3 rise time ($T$) of 0.1357s. Therefore,

$$g(s) = \frac{69.1054}{1 + 0.1357s} \tag{5.2}$$

Matlab was used to model a step input to this function as well as the integral in order to represent the position approximation. These responses, together with the original motor speed and position curves are shown below:

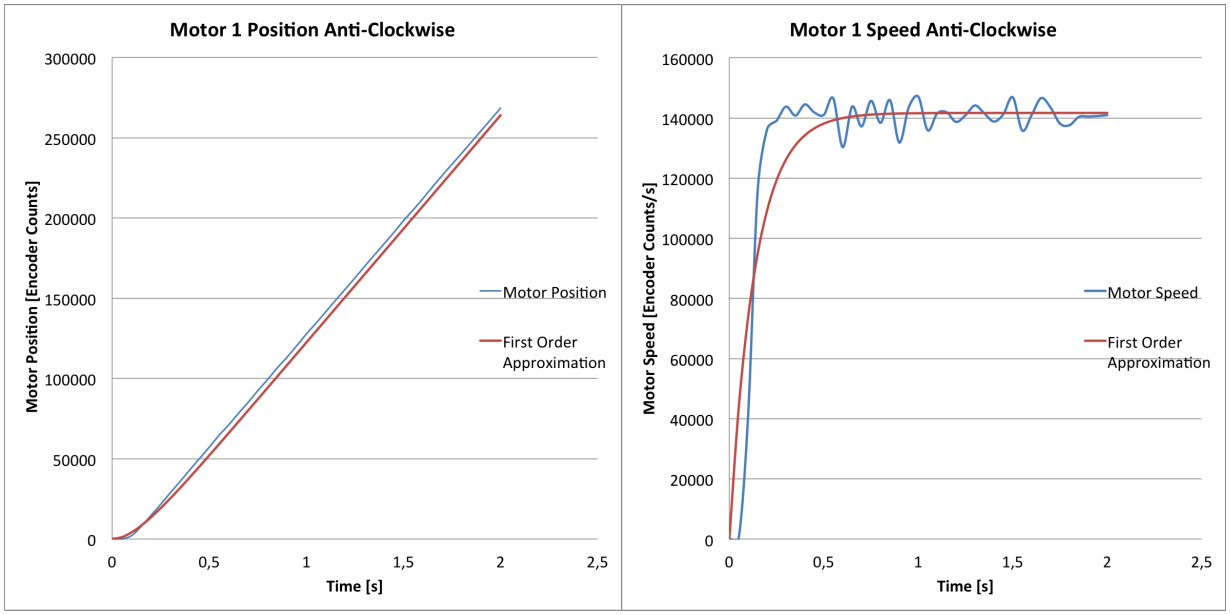

Figure 5.12: Comparison of the original position and speed curves (blue) and the first order approximations (red) described by equation 5.2. It was deduced that in controlling the arm, short durations of movements of the motors would be needed since the user would typically make small adjustments to the desired position of the arm. The first two seconds of the data was therefore analysed.

One can see that the first order approximation of the motor speed represents the system fairly well. It was felt that a critically damped second order approximation may, however, represent the system more adequately. A critically damped second order system with generic transfer function: $g(s) = \frac{k\omega_n^2}{s^2+2\omega_n s+\omega_n^2}$ was therefore investigated where ω_n is the natural frequency is radians per second. The gain value, k, in encoder counts per second per bit of DAC output, was kept at 69.1054 as used previously. An error minimisation Matlab script was implemented in order to cycle through the different values of ω_n in order to determine the value that best described the original motor speed response (Figure 5.11). After doing so, the optimal value of ω_n was determined to be 19.94. This resulted in an approximate speed response of:

$$g(s) = \frac{20476.54}{s^2 + 39.88s + 397.604} \tag{5.3}$$

the speed response once integrated gives the position response:

$$g(s) = \frac{20476.54}{s^3 + 39.88s^2 + 397.604s} \tag{5.4}$$

A plot of these approximations, the original speed and position curves and the first order approximations follows:

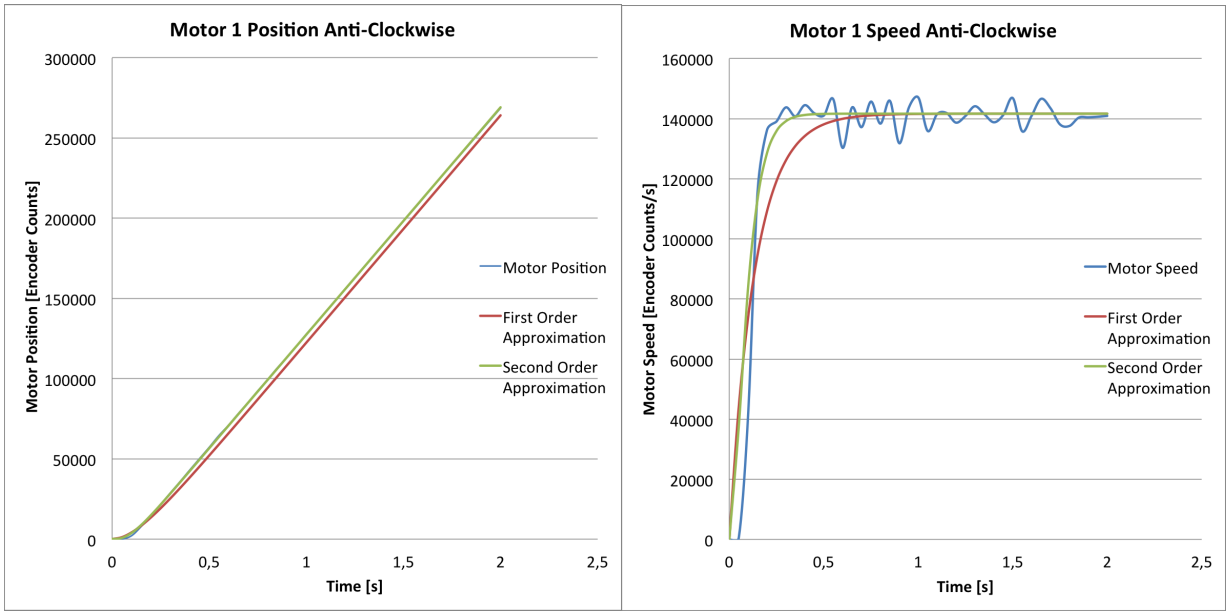

Figure 5.13: Comparison of the original speed curve (blue) with the first order approximations (red) and the second order approximations (green). It can be seen how much better the second order approximation fits compared to the previous attempt. The second order position approximation fits so closely that the original blue line can only be seen briefly at the beginning of the response.

The non-linearity of the motor was then investigated. The speed and position response in both anti-clockwise and clockwise rotations was compared:

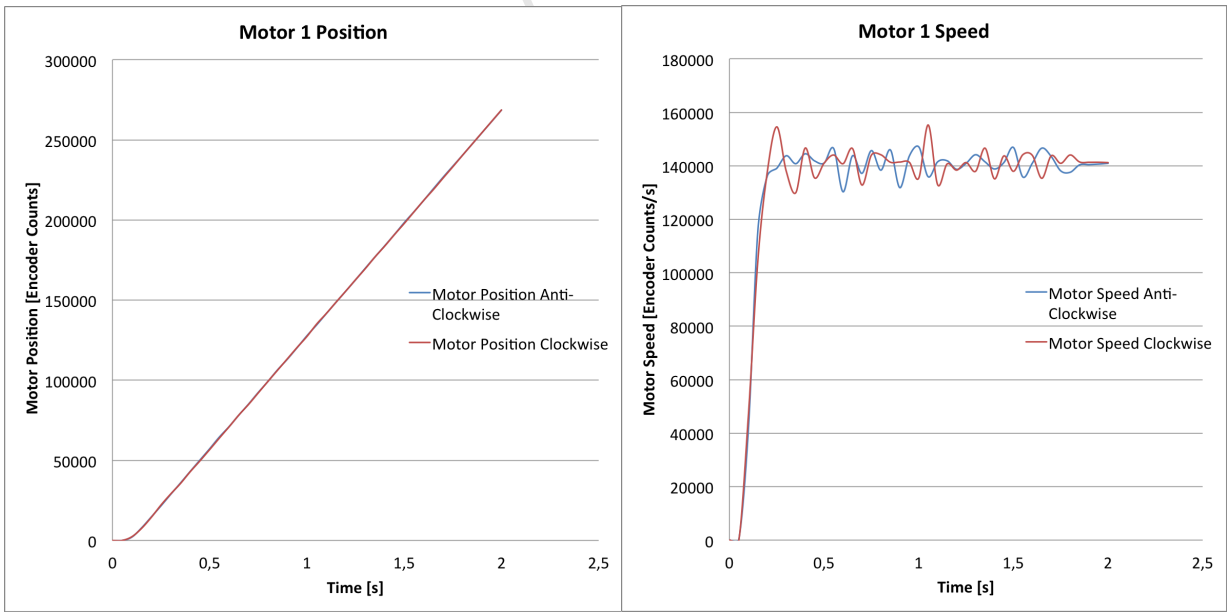

Figure 5.14: Comparison of the motor speed and position responses in both clockwise (red) and anti-clockwise (blue) directions. It can be seen that the responses in both cases in the position plots are identical as one line covers the other. The speed responses are closely matched except for the oscillations when reaching the setpoint. It was therefore concluded that the motor response could be considered linear in both directions.

Once an accurate approximation of the motor characteristics was determined, it was used to design a PID controller

in order to attain the desired system response. Matlab was used to simulate placing the motor position approximation in a closed loop. Applying a varying proportional controller value of Kp=0.01 to Kp = 0.1 yielded the following result:

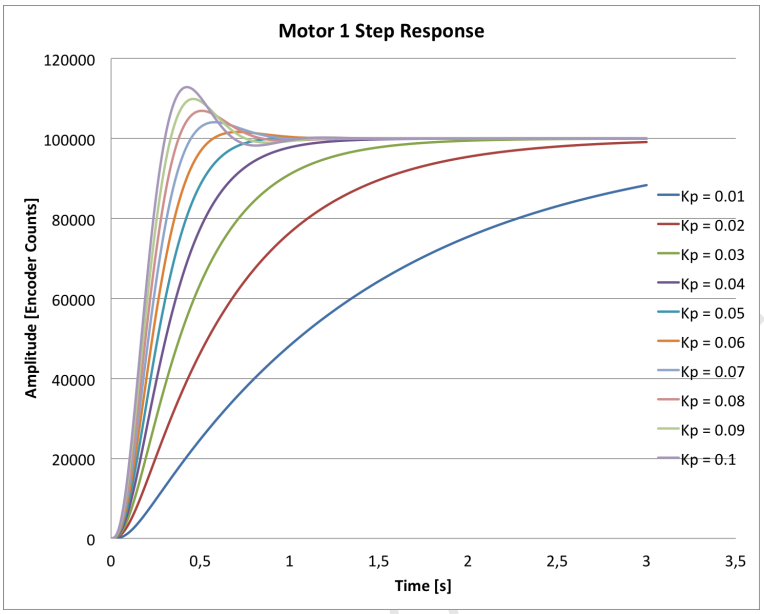

Figure 5.15: Matlab simulation of closed loop motor position response with varying values of Kp. The responses are in ascending order from right to left. The very slow response of Kp=0.01 (blue far right) and the undesirable overshoot of Kp=0.1 (violet far left) can be seen.

From the above plot it was deduced that a proportional controller of value Kp = 0.04 would be an adequate response for the control of the motors. However, it was felt that a less accentuated deceleration would prove to increase the position accuracy. This was due to the fact that the minimum speed dictated by the Maxon speed controllers was 1000rpm. Decelerating to this speed, over a longer duration, would minimise the stopping overshoot when reaching the desired position. Adding a differential term, Kd, to the control law was therefore investigated. An increase in Kp to 0.07 together with the addition of a Kd term of 0.01 improved the response in simulation (Figure 5.16). An Integration term was excluded from the control law as it served to only deteriorate the motor response.

In the actual implementation of the motor control laws, allowances had to be made for the linear approximations that were made in the estimations of the motor responses. Specifically the fact that a linear speed ramp was assumed (0 DAC value = 0 speed) whereas the actual response of the speed controller meant that 0 DAC value = 1000rpm. A scaling factor therefore had to be applied to the calculation of the value sent to the DAC in order to keep the actual response as similar to the linear approximated response as possible. In terms of the equations, since the gain of the first order system was calculated to be 69.1054, a linear approximation of the motor speed was $69.1054 \times n$, where n is the value between 0 and 4095 which is sent to the DAC. The actual speed response (in Encoder Counts/s) is in fact closer to $33333 + (52.91 \times n)$ due to the speed offset. Therefore, a scaling factor needed to be implemented upon n in order to correlate the actual speed response to the linear approximation. This

was done by equating both the linear and actual responses to each other and solving:

$$69.1054n = 33333 + 52.91m \tag{5.5}$$
$$m = -630 + 1.306n$$

Therefore, the actual value sent to the DAC, m, was determined by scaling the calculated value of n by the above equation. As the DAC can only receive values between 0 and 4095, anything outside these bounds was capped to nearest applicable value. The simulated curves, as well as the actual motor position step response curves, to both the simple proportional (P) controller together with the Proportional-Derivative (PD) controller are shown below.

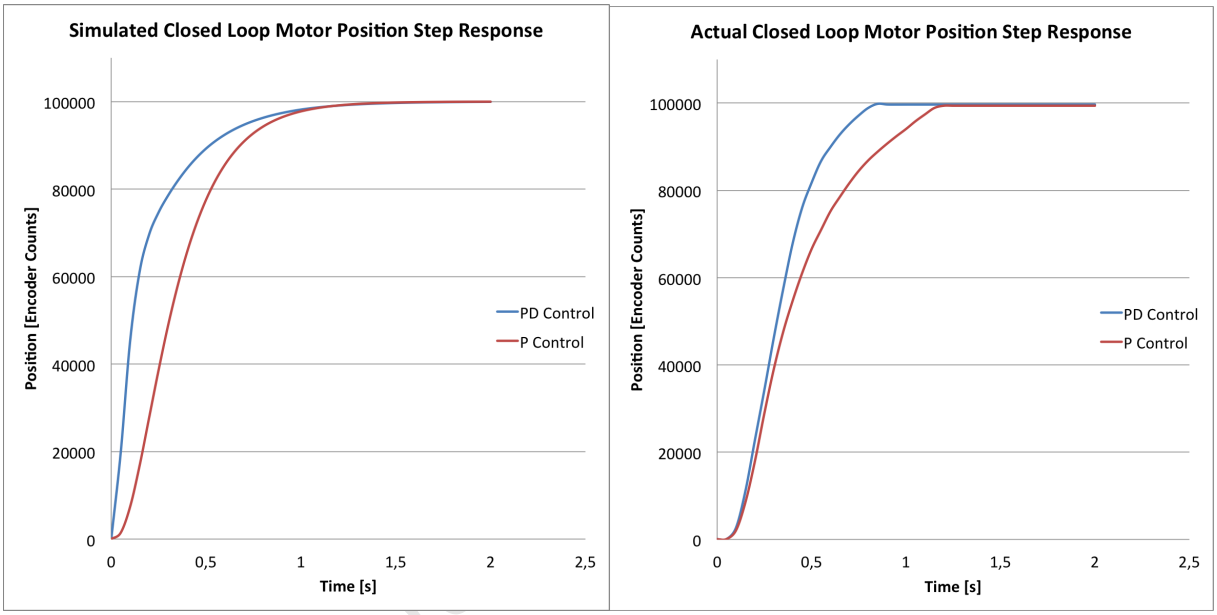

Figure 5.16: Comparison of a simple P controller (red) to PD controller (blue) in both simulation (left) as well as implementation (right).

Large differences can be seen in the simulated responses compared to the actual motor position step responses. Rate limits were excluded from the simulations in order to show the difference between the actual motor response and the ideal. Three factors play a role in this: firstly, the inherent ramping of the motor speed caused by the Maxon speed controller creates an initial discrepancy between the simulated and actual step responses. The second factor is the maximum speed of the motor. In simulation, no maximum speed is assumed which creates much steeper step responses compared to the actual response, where maximum speed is a factor. Lastly, and most significantly, the simulations assume that the minimum speed of the motor is 0 whereas in reality the minimum speed is 1000rpm. The simulation therefore shows far more gradual decelerations in motor speed whereas the actual response is limited by the minimum speed of the motor. When reaching the setpoint, a more abrupt stopping of the actual response is created by this minimum motor speed. Due to the faster settling time and the adequate deceleration of the motor speed, the PD controller was selected to be used for motor control.

Once this control law was established a comparison of the first three motors (as they are all the same motor with the same speed controller) was performed. This was to determine whether the same control law may be applied to

all these motors. Below is a plot of the position and speed of all three motors in both a clockwise and anti-clockwise direction:

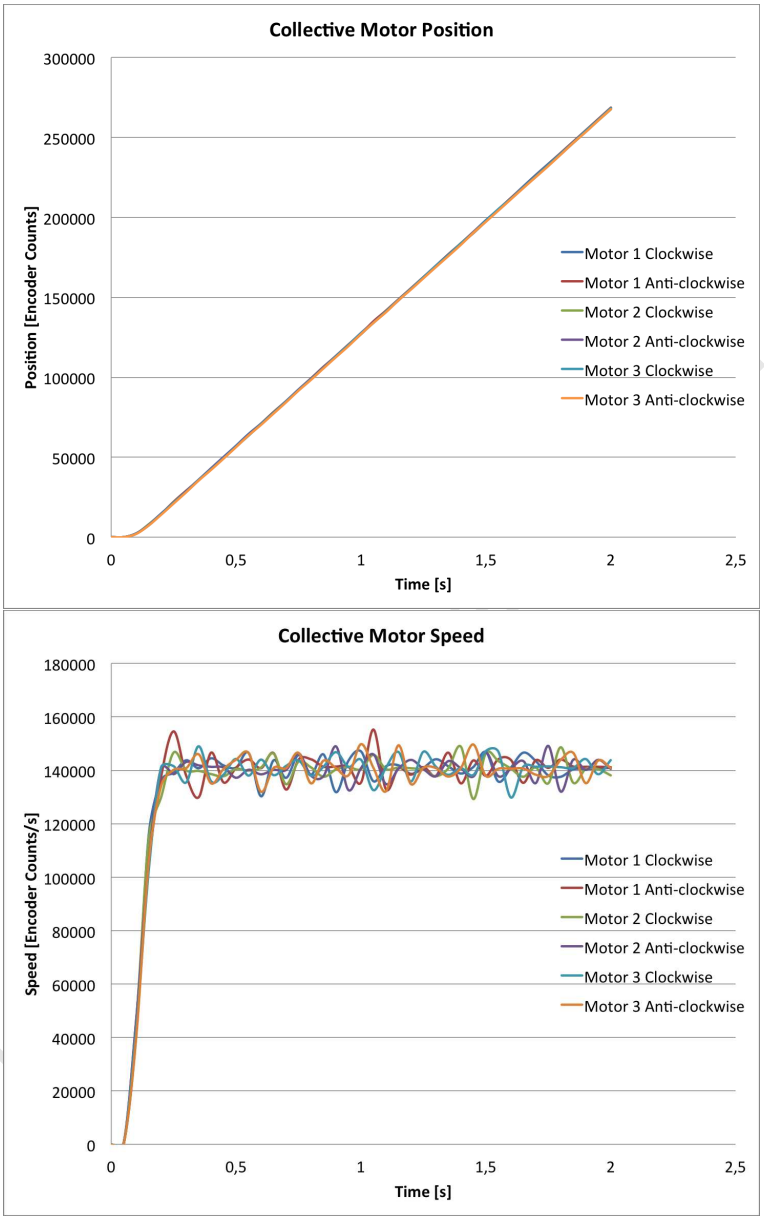

Figure 5.17: Collective plot of position and speed step responses of first three motors in the robotic arm. The close similarity of all the plots justifies the use of the same control law for all three motors.

5.3.2 Motors 4, 5 and 6

Pan Motor

The same control procedures that were implemented for the first three motors were carried out on the pan motor. Initially, the linearity of the pan motor in a clockwise and anti-clockwise direction was investigated:

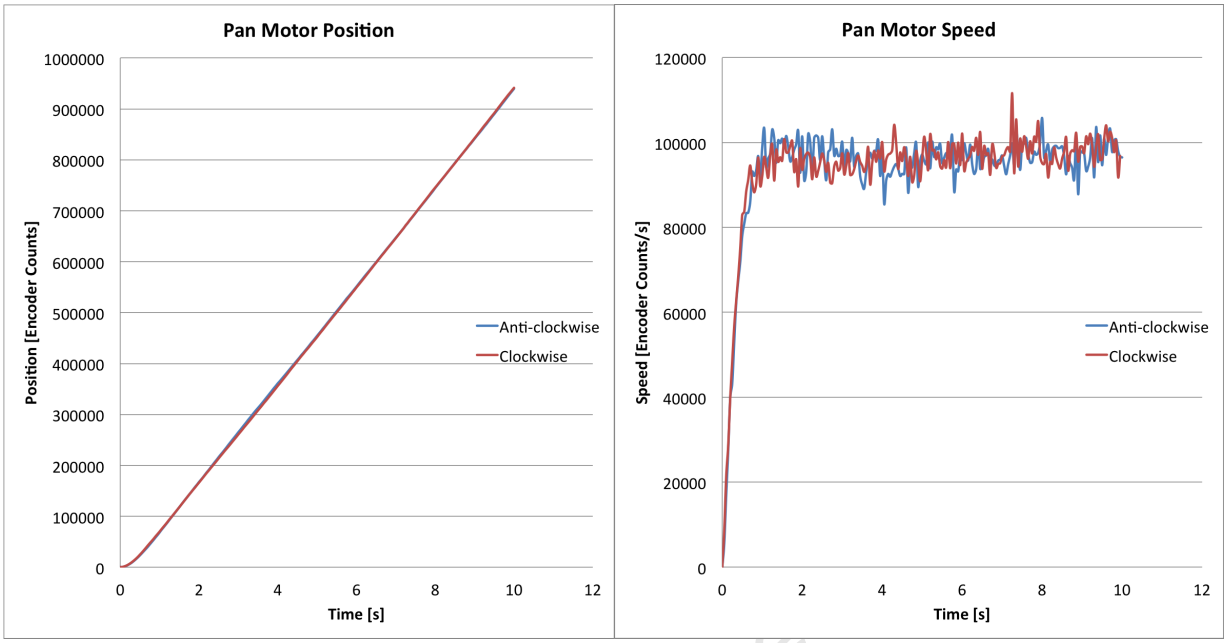

Figure 5.18: Pan motor position and speed step responses. It can be seen that the responses are very similar in both directions. A slight wavering can be seen in the speed response of the motor. As this response was acquired with the tilt arm at an angle, the weight of the payload creates a varying moment about the point of rotation which affects the speed response of the motor.

A second order approximation to the speed response was once again calculated. The gain of the system was determined to be 93.59. The same error minimization Matlab script was then implemented in order to determine the best second order approximation to the pan motor speed step response. Running this algorithm provided a best case ω_n value of 7.18. The approximate transfer function for the speed of the pan motor is therefore:

$$g(s) = \frac{4824.79}{s^2 + 14.36s + 51.5524} \tag{5.6}$$

The position approximation is simply the integration of this:

$$g(s) = \frac{4824.79}{s^3 + 14.36s^2 + 51.5524s} \tag{5.7}$$

Plots of the speed and position step responses in comparison to these approximations are shown below:

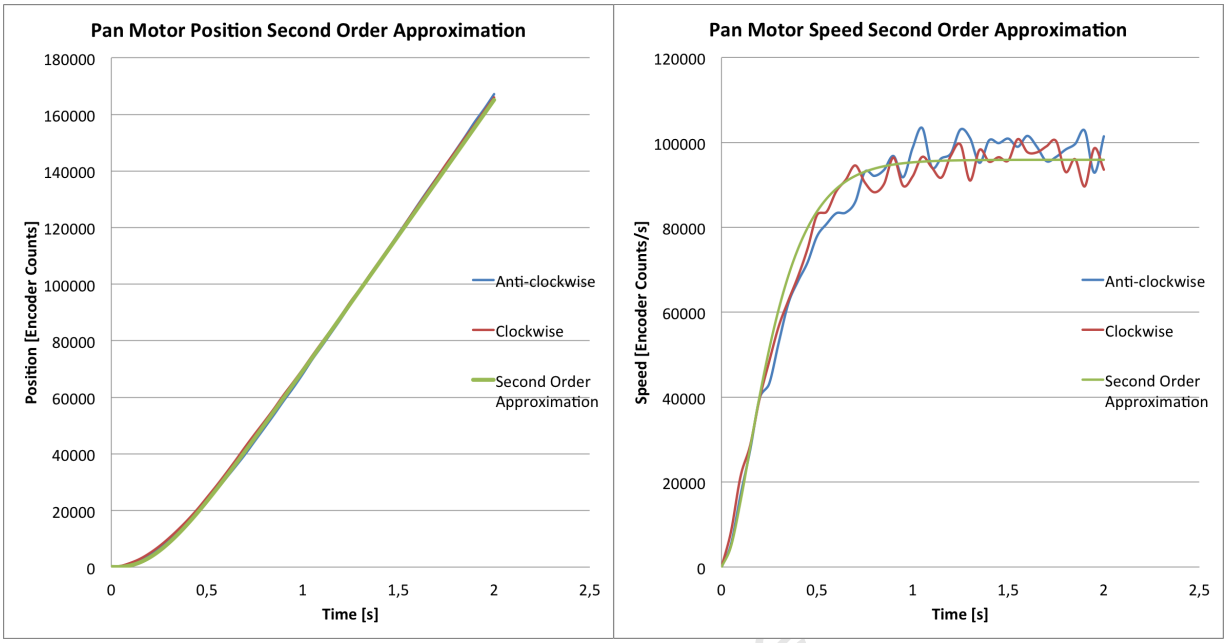

Figure 5.19: Comparison of position and speed step responses to the linear approximations. A good fit is seen in both cases.

As done previously, Matlab was then used to develop a controller for this motor. Once again, a PD controller (P=0.02 and D = 0.008) was selected as the best option. A non-linearity compensation equation was implemented on the system as done for the first three motors. Shown below is the comparison of the simulated PD controlled response with that of the actual motor position response:

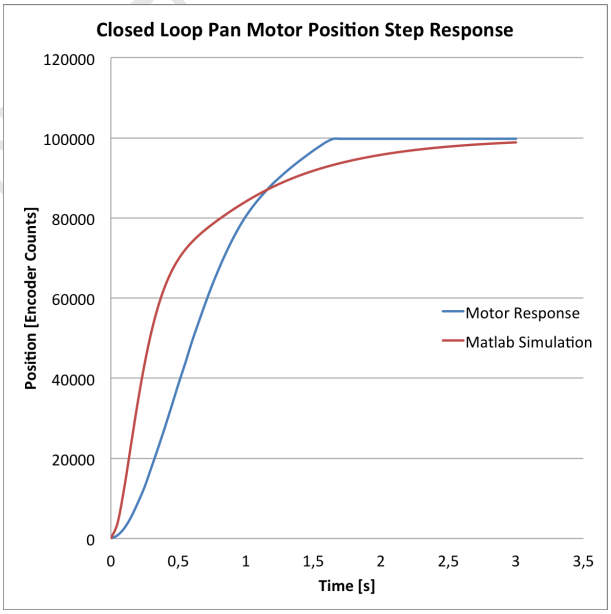

Figure 5.20: Comparison of the PD controlled pan motor in implemented closed loop as well as simulated closed loop. It can be seen that a fairly large discrepancy exists between the simulated (red) and actual (blue) motor responses. The reasons for this are identical to those explained in Figure 5.16. This PD controller delivered an adequate response time together with decent accuracy.

Tilt Motor

The position control of the tilt motor proved to be very difficult due to the instability of the system. The pan and tilt motor housing designs did not incorporate self-locking worm gear systems like that of the other arm links. The sensor payload therefore proved to be too heavy for the tilt motor[3] and would fall uncontrollably when rotated past a certain point (clarified below). The tilt motor speed curves (Figure 5.21) illustrate this.

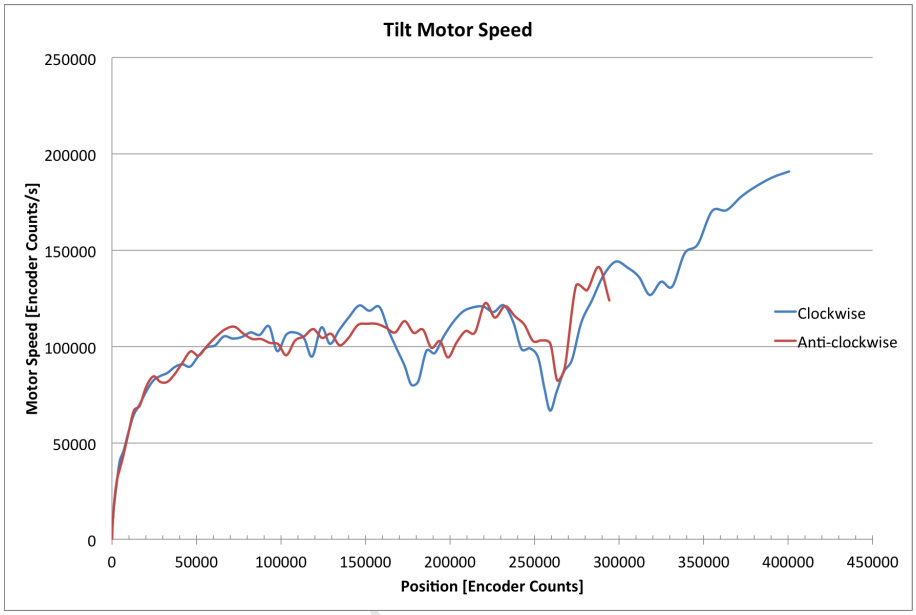

Figure 5.21: The speed of the tilt motor from a vertical position compared against motor position.

In both cases, a sharp acceleration of the motor can be seen at approximately the 250 000 encoder counts position, indicating the loss of control of the sensor payload. The sharp variations in speed leading up to the falling point indicate that the Maxon 24/3 speed controller was trying to slow down the speed of the motor as it began to fall. As the position of the payload tends towards the horizontal, the weight of the payload proved to be excessive; it could not be controlled by the motor and the payload fell. The motion of the payload in the anti-clockwise direction was halted in order to prevent it from colliding with the other links of the arm. Due to the similarity of motion in a clockwise direction, it was felt that the same conclusions could be drawn for both directions.

Due to the instability of the tilt motor movement, it was decided to implement a limit on the range of motion. As can be seen from the plot above, the point at which the instability of the system begins is at approximately 130 000 encoder counts from the vertical. This equates to 22 degrees. A limit of 20 degrees in both directions was therefore implemented in order to keep the tilt system within a stable range of motion.

The above plot was used to determine a second order system to represent the tilt motor movement. The gain of the system was determined to be 101.4 and when applying the same error minimisation algorithm as used previously,

[3]The motor in question was the stronger replacement of the original motor which P. Henson originally found to be too weak to control the tilt of the sensor payload.

a best fit ω_n value of 6.05 was yielded. The transfer functions to describe the system are shown below.

$$g(s) = \frac{3711.50}{s^2 + 12.1s + 36.6025} \tag{5.8}$$

$$g(s) = \frac{3711.50}{s^3 + 12.1s^2 + 36.6025s} \tag{5.9}$$

The following plot illustrates the comparison between the second order approximation and the actual motor responses:

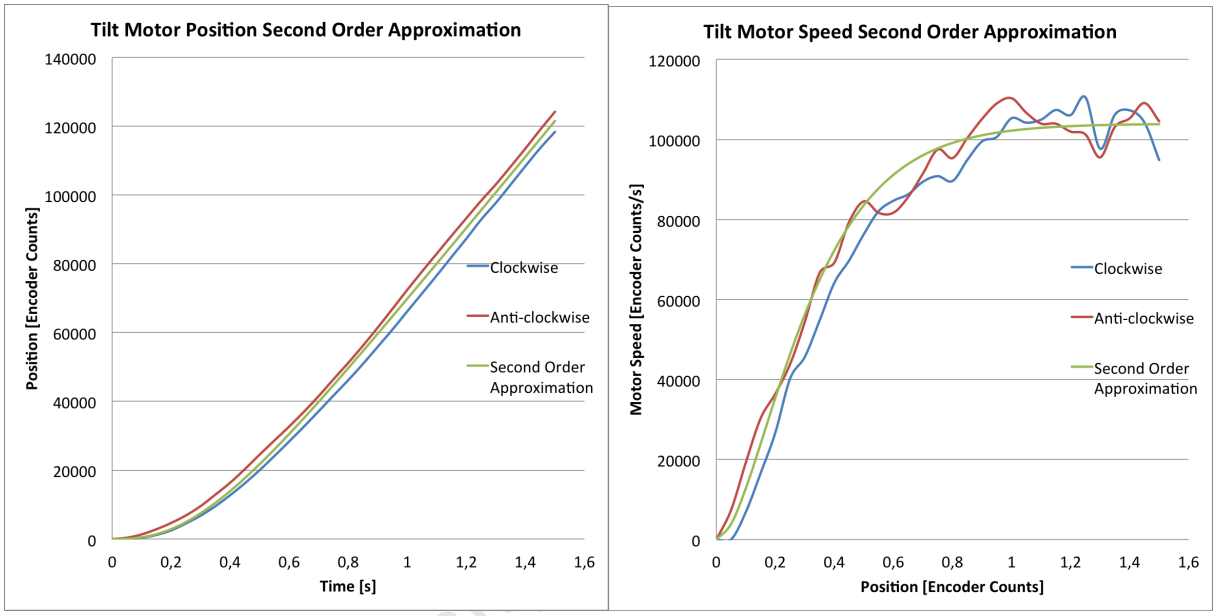

Figure 5.22: Comparison of the tilt motor position and speed step responses to the linear second order approximations. A slight discrepancy between the responses of the motor in a clockwise and anti-clockwise direction can be seen. An analysis of the same occurrence in the following section (section 5.3.2) resulted in the same control law being applied in both directions. A best fit approximation to both directions was therefore implemented on this motor.

This approximation was then used to develop a controller for the motor. Due to the close similarity of this system to that of the pan motor it is not surprising that the same controller values were derived for the control law; P = 0.02 and D = 0.008.

Second Elbow

The plots of the second elbow (sixth motor) are shown below:

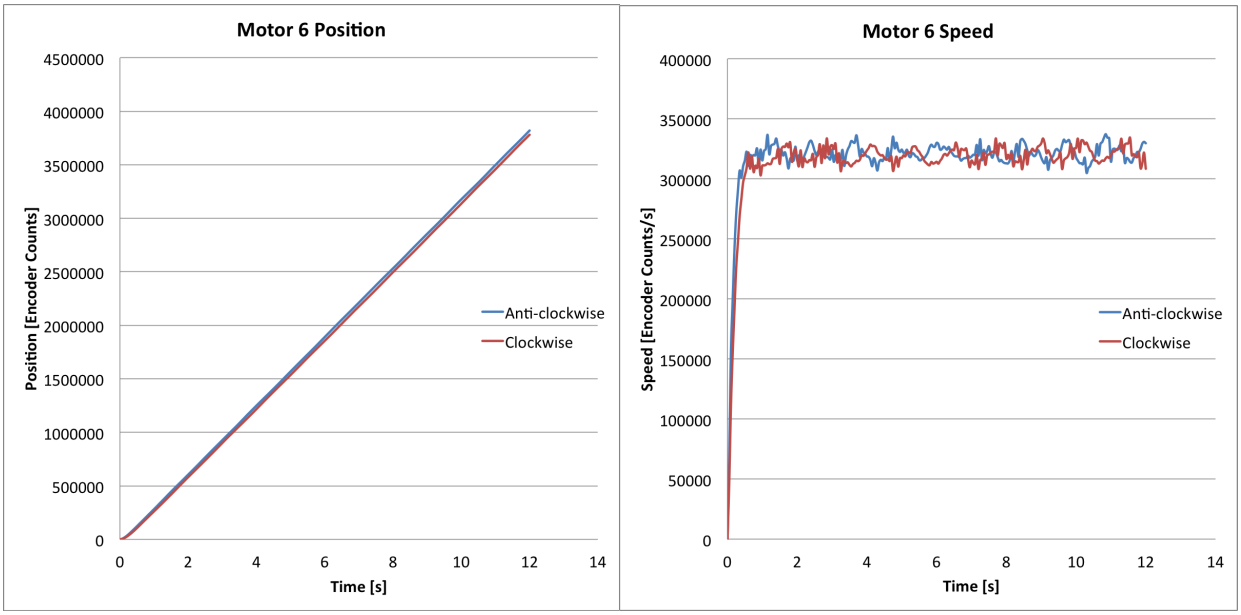

Figure 5.23: Motor 6 speed and position step response.

The motion in an anti-clockwise direction is slightly faster than clockwise. In an anti-clockwise direction the motor is effectively lifting the gripper, thus causing the slower speed. The difference in motor speed is more pronounced when looking at the first two seconds of the motor step response:

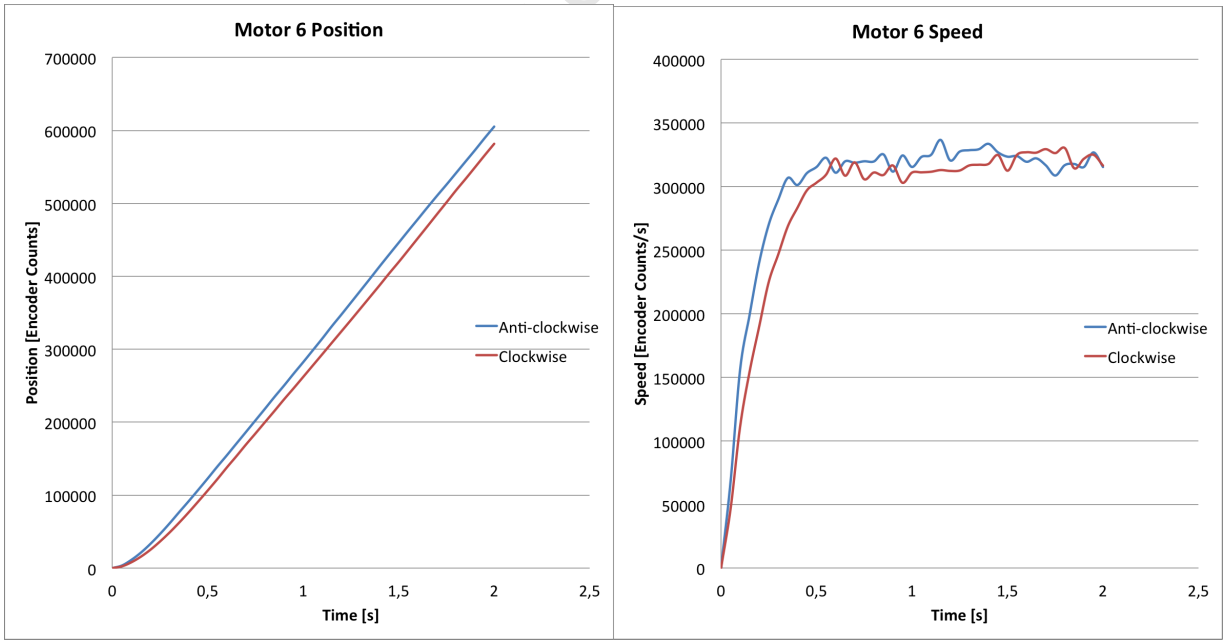

Figure 5.24: First two seconds of Motor 6 position and speed step responses. A clearer illustration of the difference in responses based on the direction of movement can be seen.

Due to this discrepancy in motor response characteristics the control for the motor was developed separately for each direction of motion. A second order system was approximated for both the clockwise and anti-clockwise directions

of the motor. Firstly, clockwise: the gain in a clockwise direction was calculated to be 77.417 and the Matlab error minimisation algorithm implemented upon the clockwise speed response indicated that the most suitable value of ω_n was 11.78. The anti-clockwise gain equalled 77.62 with a best ω_n value of 18.79. Below are the comparisons of actual motor responses to the approximations:

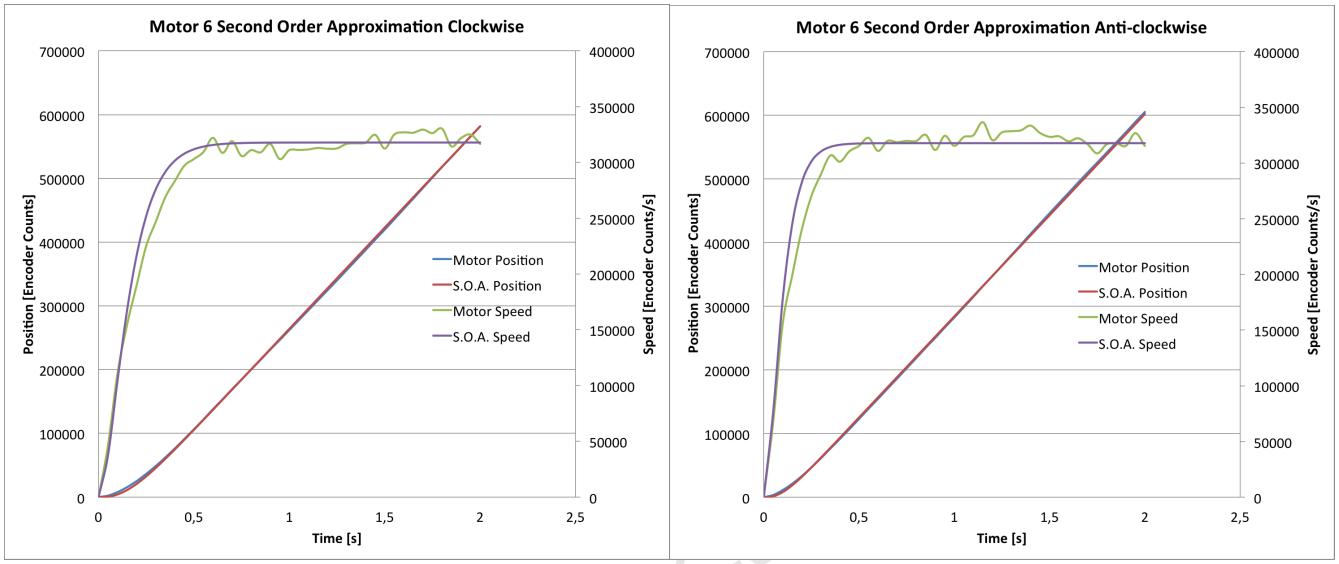

Figure 5.25: Comparison of motor response to second order approximations in both clockwise and anti-clockwise directions. A good fit in all instances can be seen.

The transfer functions for the clockwise direction are therefore:

$$g(s) = \frac{10743}{s^2 + 23.56s + 138.77} \tag{5.10}$$

for the speed and

$$g(s) = \frac{10743}{s^3 + 23.56s^2 + 138.77s} \tag{5.11}$$

for the position. The anti-clockwise equivalents are:

$$g(s) = \frac{27404.8}{s^2 + 37.58s + 353.064} \tag{5.12}$$

and

$$g(s) = \frac{27404.8}{s^3 + 37.58s^2 + 353.064s} \tag{5.13}$$

These transfer functions were used in Matlab to calculate the best controllers for the system. When calculating these controllers, it became clear that using the same PD controller (P=0.05, D = 0.01) in both the clockwise and anti-clockwise cases resulted in the best response. The following plot illustrates the similarity of the motor responses:

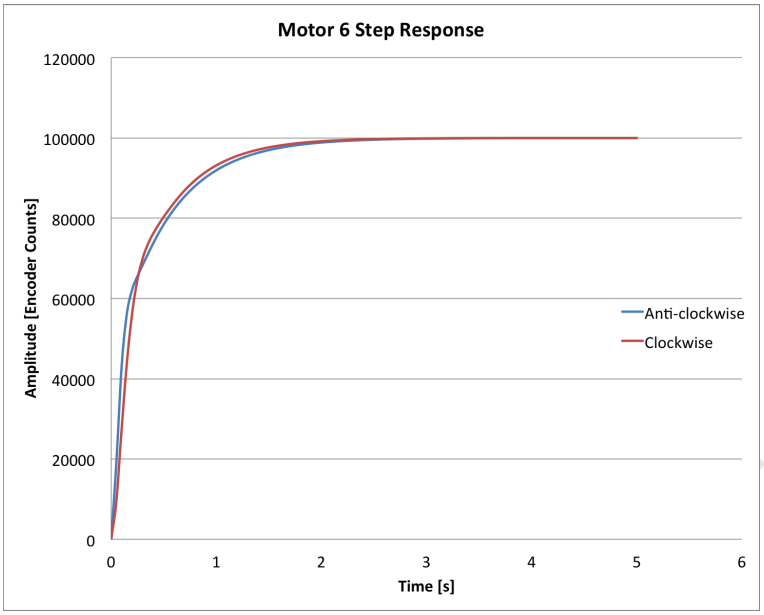

Figure 5.26: Matlab simulated motor 6 PD controlled position step response in both a clockwise and anti-clockwise direction. A strong similarity between both responses can be seen.

The comparison of the simulated closed loop response to that of the actual response highlights the same discrepancies discussed in the previous sections:

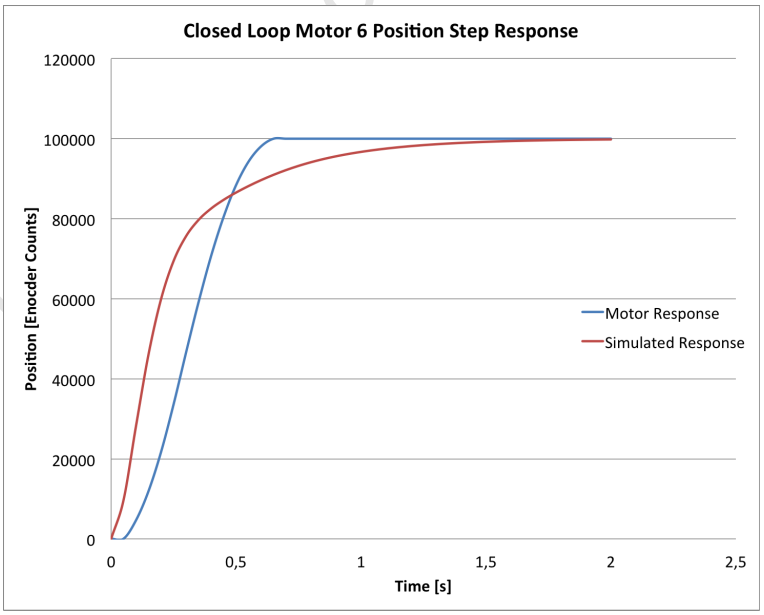

Figure 5.27: Comparison of the PD controlled motor in implemented closed loop in an anti-clockwise direction (blue) as well as simulated closed loop (red).

When implementing the control law in closed loop, it became apparent that the motor system did not hold the link in a fixed position upon reaching a setpoint when moving in a clockwise direction; effectively when attempting to lift the gripper. This is illustrated by the following plot:

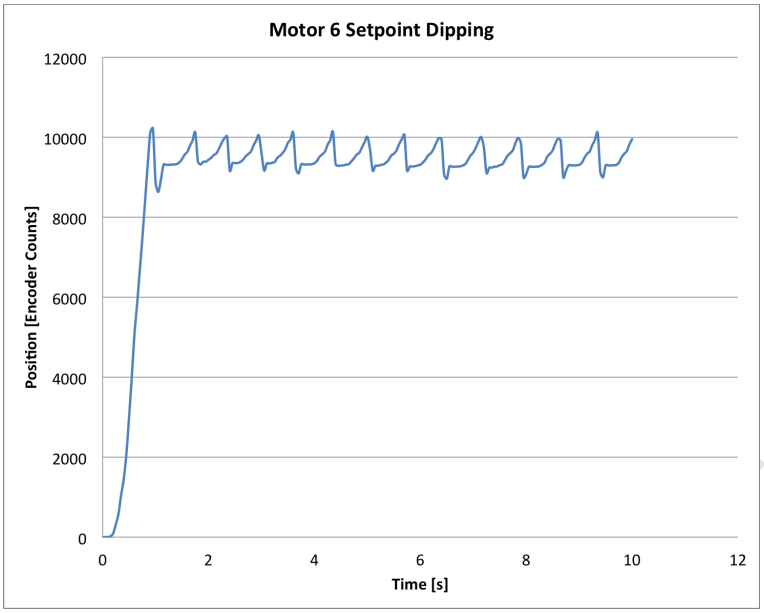

Figure 5.28: Motor 6 setpoint dipping upon moving in a clockwise direction.

It can be seen that upon reaching the setpoint of 10 000 encoder counts, due to the motor's inability to hold the position, the motor position drops sharply. This action occurs repeatedly. Two factors play a part in this. Firstly, the last joint of the arm is the only joint (with the exception of the pan/tilt system) that does not utilise a worm gear system. Instead it utilises a hypoid gear system. This apparently offers a less acceptable performance in holding the link in place as this did not occur in any of the other gearing systems. Secondly, the weight of the gripper creates a large moment about the joint of the final link. Although the sixth motor is heavily geared (190:1), this moment may be too large for the motor to hold. The combination of these factors caused the motor position to dip upon reaching the desired setpoint.

In all the other motor control cases a positive and negative deadband was implemented to compensate for overshoot in both motor directions. Therefore, the setpoint was effectively the desired motor position minus the deadband area. In the case of the sixth motor this was unfeasible as the motor would reach the setpoint and thereafter dip out of the deadband area thus causing this repetitive dipping action. A method to overcome this motion was to dictate a much smaller deadband area if the motor was moving compared to the standard deadband area if the motor was stationary. Therefore, the motor would reach the smaller deadband area whilst moving, stop its motion and thereafter "fall" to within the stopped deadband area thus eliminating the dipping cycle shown above.

The typical motor control algorithm used on the LM3S8962 motor control board can be represented by the following flow diagram:

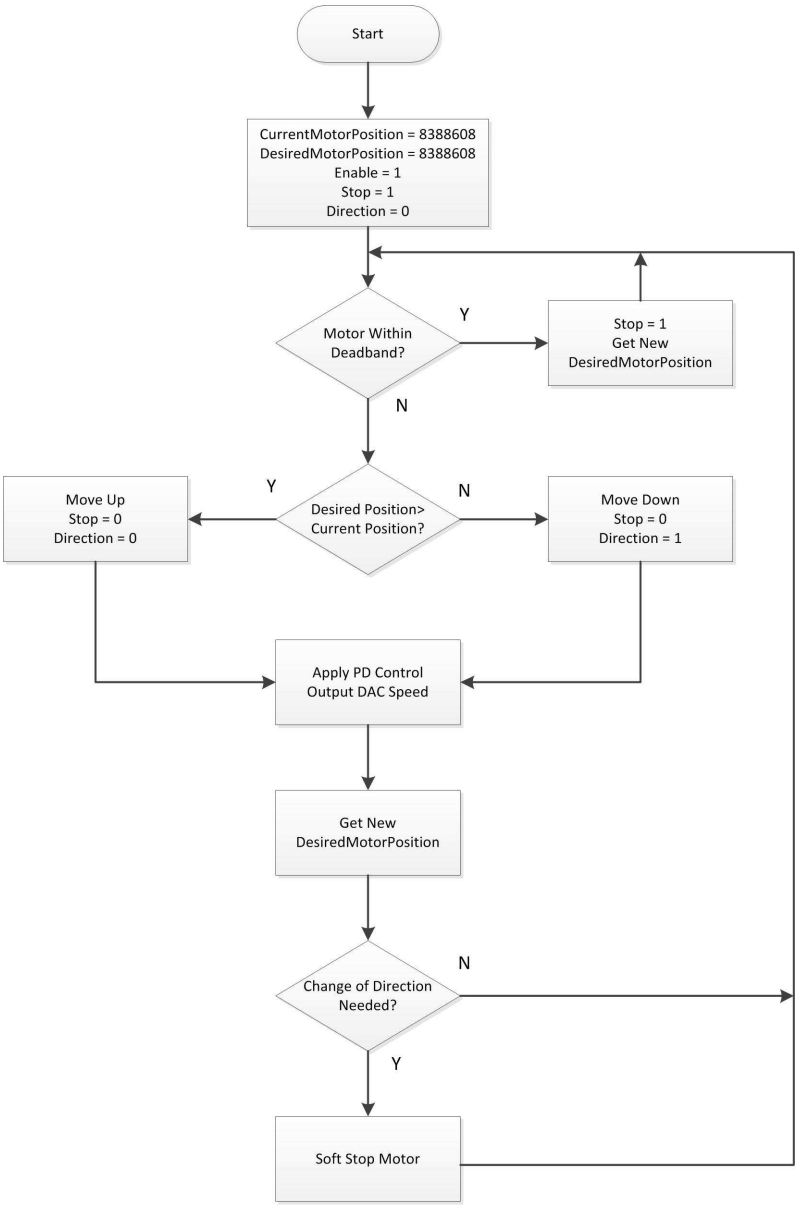

Figure 5.29: Motor control flow diagram as used on the LM3S8962 motor control boards.

The PD control law for each motor as calculated above can be represented as follows:

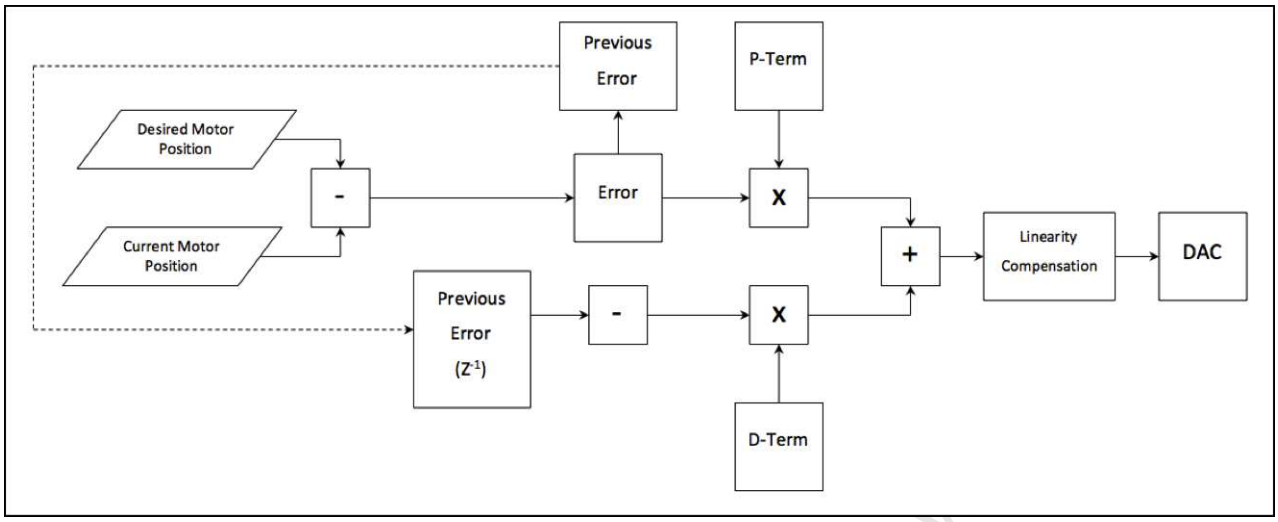

Figure 5.30: PD control law flow diagram.

5.4 Summary

These motor control algorithms were coded on the motor control boards[4] in the arm and performed as expected offering stable, accurate motor position control. The testing and results of the motor position control can be found in Chapter 9.

The motor control boards in the arm required a communications scheme in order to receive the desired motor positions. The following chapter discusses the different communication schemes implemented in order to send the desired motor positions to each of the motors.

[4]refer to DVD for LabVIEW code

Chapter 6

Communications

6.1 Introduction

The desired motor positions are derived from the inverse kinematic calculations as shown in Chapter 4. In order to transmit these to each of the motors in the RATEL robotic manipulator, a stable communications architecture needed to be developed between the user and the robot as well as between the different motor control boards in the RATEL arm. The general layout of the necessary communication architecture is shown in Figure 6.1.

The high number of slave devices and the severe lack of wiring space in the RATEL manipulator dictated that a daisy-chain communications protocol between the motor control boards in the robotic arm be used. The I²C communication protocol, such as that used by [63, 62, 61], was the first communications scheme used in order to send and receive motor control information to and from the motor control boards in the robotic arm. This was initially coded on the Coldfire-FPGA boards (see Figure 5.3) and later on the LM3S8962 boards (see Figure 7.1).

6.2 I²C Communications on the Coldfire-FPGA Board

The I²C code[1] used on the Coldfire-FPGA boards was implemented in an interrupt routine as suggested by the MCF51JM128 reference manual [75] according to the flow diagram shown in Figure 6.2. Two Coldfire-FPGA boards were used to test the communication protocol. One was set as the master while the other was set as an addressable slave. The slave board included the quadrature decoding algorithm discussed in Section 5.2 in order to move a motor to a desired position as shown in Figure 6.3. This code can be found on the DVD provided. The system was able to successfully move the motor to set positions transmitted to the slave board from the master device. Only limited testing of this system was executed before the Coldfire-FPGA boards were replaced with the LM3S8962 motor control boards for ease of motor control and lab standardisation purposes.

[1] Coded in Freescale Codewarrior

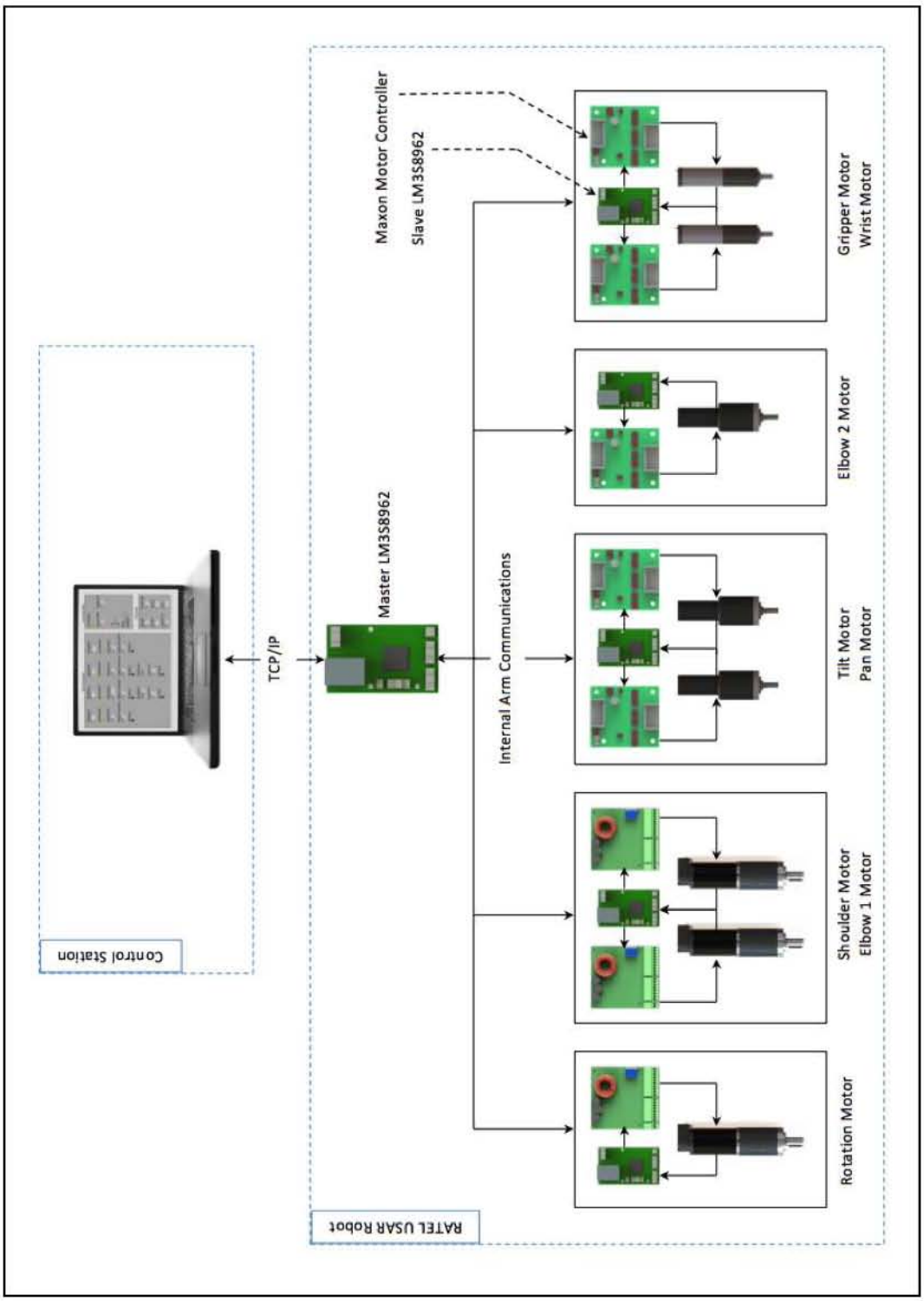

Figure 6.1: General communications architecture of the RATEL robotic manipulator.

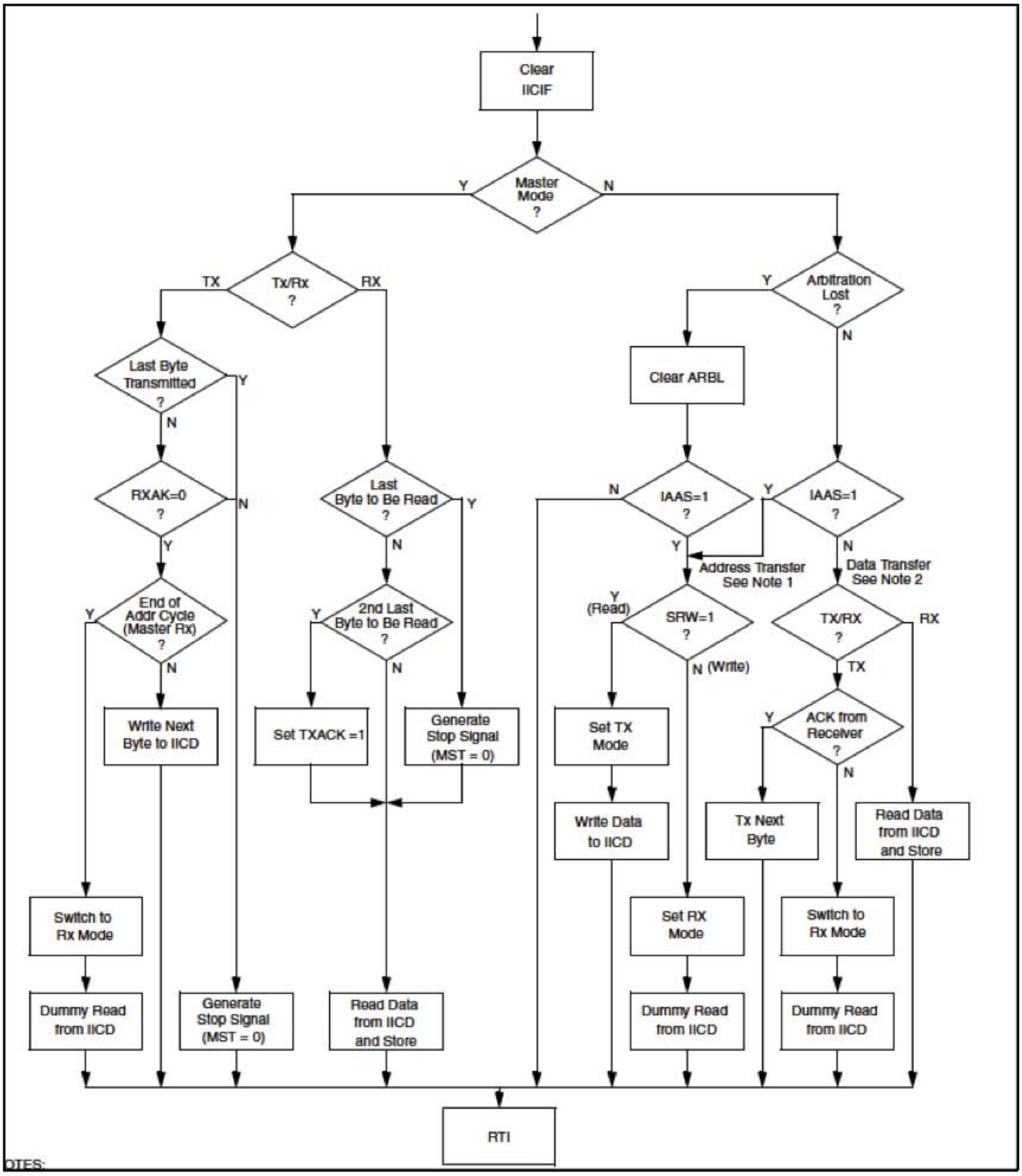

Figure 6.2: I²C interrupt routine as stipulated in the Coldfire reference manual [75].

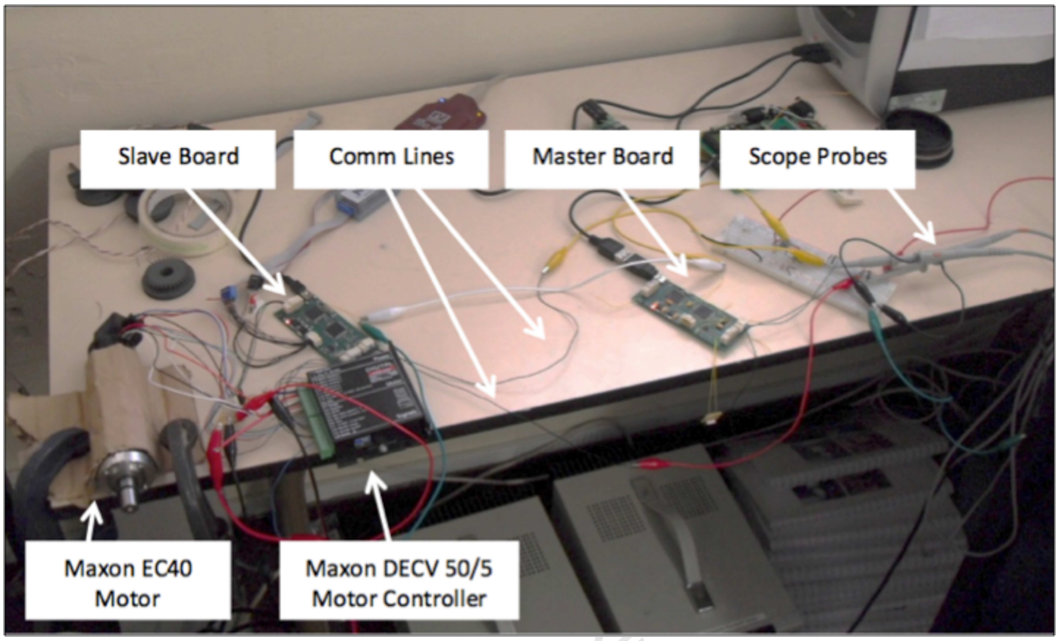

Figure 6.3: I²C and motor test setup for the Coldfire-FPGA board.

6.3 I²C on the LM3S8962 Board

Like the Coldfire microcontroller, the LM3S8962 had an integrated I²C module and thus the same communication protocol was used on the new motor control boards coded in LabVIEW Embedded for Arm Microcontrollers. The I²C communication architecture can be seen in Figure 6.4.

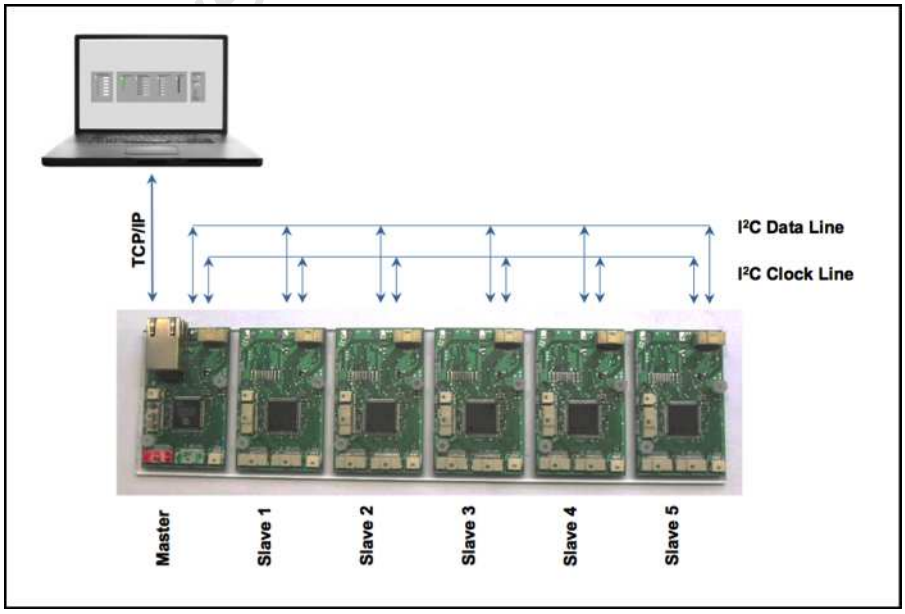

Figure 6.4: The I²C communication architecture as used on the LM3S8962 motor control boards.

The master board was responsible for handling the TCP/IP communications to and from the control station as well as communicating this information, over I²C, to the relevant slave boards. This was completed in a sequential order. Initially the master board would receive all the motor positions via TCP/IP and thereafter address and transmit the relevant information to each of the slave boards. Each slave board would reply with motor information and the master would then, after having addressed all the boards, transmit this information back to the user via TCP/IP.

On the slave boards, LabVIEW did not offer an interrupt routine for the processing of the I²C communications like that found on the Coldfire boards. Therefore, the code (see appended DVD), after giving the slave board an address with which the master board could communicate, would continuously check the I²C buffer for new information. This occured upon each repetition of the motor control loop. If information present upon the I²C lines was addressed to the board in question, the communications code would extract the packets of motor control data and reply to the master board with motor information. The motor information transmitted up the arm (to the slave boards) was simply the desired motor position of up to two motors[2]. The information transmitted down the arm, back to the master board and subsequently to the user, included a boolean value to indicate whether the slave board was functional, the present motor position, motor current and the state of the hall effect sensors (further discussed in Section 7.3).

The preliminary testing of the communications scheme (see Chapter 9) showed a stable architecture with a satisfactory communication speed of 20Hz. Placing the communications code in the same loop as the motor control code, however, made the communication speed unfavourably dependent on the processing time of the motor control loop as shown in Table 9.6. When testing the communication scheme in the arm it became apparent that the increased motor noise, due to the close proximity of the communication lines to the motor lines, even with the use of twisted pair as well as shielded cable and ferrite beads, caused the communications to regularly fail. It was therefore decided to look at other communication options between the LM3S8962 motor control boards. The use of a CAN bus, as used by [64, 59, 76], was initially investigated although soon discarded due to the lack of functionality in coding an LM3S8962 as a CAN slave board. An RS-232 protocol was therefore implemented.

6.4 RS-232 on the LM3S8962 Board

The design of the third generation LM3S8962 motor control boards, as used in Section 6.3, incorporated a MAX232 IC, connected to the LM3S8962 Universal Asynchronous Receiver/Transmitter (UART) ports, in order to accommodate RS-232 communications between devices. It was decided to use this functionality to create a new communications scheme in the robotic arm. RS-232, unlike I²C, is not a daisy chain based communications scheme and operates on the basis that communicating devices have a direct, single connection with each other. The LM3S8962 unfortunately only had two UART ports and thus a direct communication link between all five slave devices was physically impossible. The use of a RS-232 server, as used by iRAP-PRO [56], was an option. However, the lack of wiring space in the arm and the limited number of available wires in the slip rings used in the manipulator restricted the use of numerous communication lines. An improvised mock daisy chain architecture was therefore created by utilising both serial ports of each LM3S8962 slave board in order to transmit motor control information up and

[2]Slave boards 2, 3 and 5 each controlled two motors whereas boards 1 and 4 each only controlled a single motor as can be seen in Figure 6.1

down the robotic manipulator from one slave board to another in an almost "leap-frog" fashion. A simplified model of the RS-232 communications architecture can be seen in Figure 6.5.

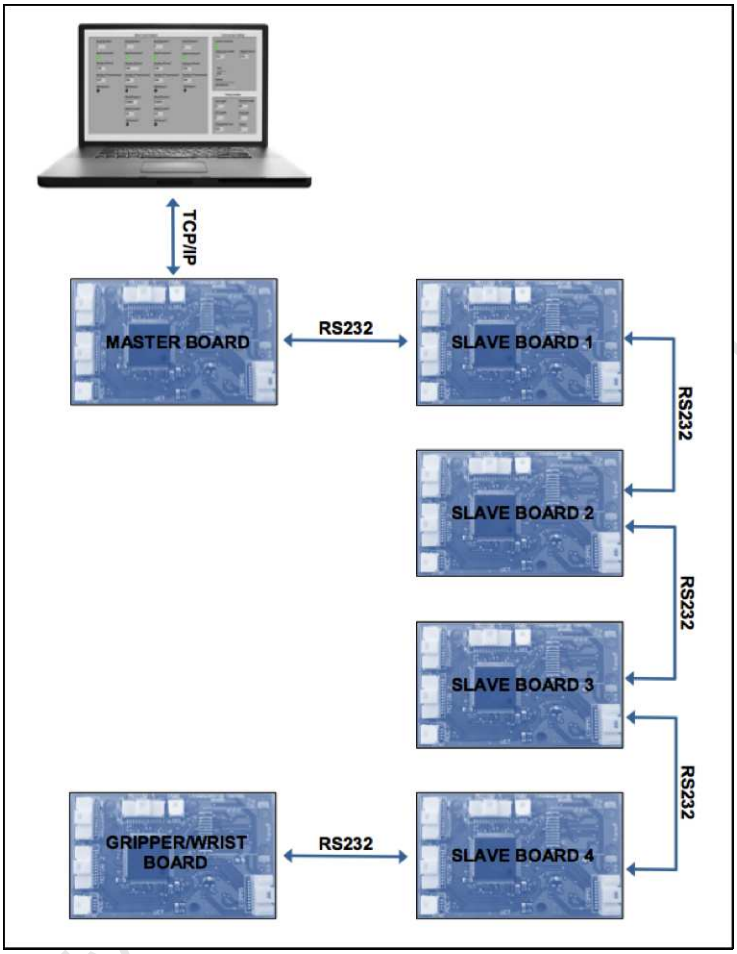

Figure 6.5: Simplified model of the RS-232 communication scheme showing client PC, master LM3S8962 communication board and slave LM3S8962 motor control boards.

Instead of the master board addressing a slave board directly, as was the case with I²C, the two serial ports on the LM3S8962 were used to pass motor control information, bidirectionally, from board to board. This allowed two devices to communicate directly with each other at one time (adherence to the RS-232 protocol standard) yet created a daisy chain system whereby each board could receive and transmit motor control information to/from the master board.

The RS-232 communications code was placed in a 50ms timer interrupt[3] on each LM3S8962 board. Three reasons motivated this decision: firstly, it was decided to handle the communications independently of the motor control code on the slave boards in order to optimize both the motor control and communications processing. The communications speed dependency created by placing the communications in the same motor control loop, as done in the I²C architecture, was deemed to be impractical for the system. Placing the communications in a parallel loop to the motor control code was investigated but this severely affected the processing speed of the microcontroller. A

[3]Faster speeds were investigated although 50ms proved to be the most stable.

timer interrupt was therefore found to be the optimal solution; it allowed for an independent communication stream whilst minimally affecting the processing speed of the microcontroller and hence the speed of the motor control loop. Secondly, the leap-frog communication architecture used in this scheme ran optimally by having a regular interval to move data up and down the arm. With each interrupt, due to the synchronisation of all the boards to the 50ms interval, it was guaranteed that there would be information present at the serial buffer for each slave board to read. This, therefore, prevented the situation where a board would wait for information to be available and reduce the overall speed of communications. The timer interrupt, therefore, proved to optimise the communication speed of the system. Lastly, referring to the master board, it was discovered that performing the TCP/IP to RS-232 conversion in a sequential manner, as described in section 6.3, could affect the communication speed to the motor control boards. Any disturbance in the TCP/IP communications would have a direct impact on the communications up the arm. As a result of this, it was decided to perform the TCP/IP and RS-232 communications independently on the master board which called for the use of a timer interrupt.

The technique used to send information up the arm differed to that for sending information in the opposite direction (down the arm, back to the master). Up the arm, each slave board needed to extract the relevant pieces of motor position information. Down the arm, however, the information was directed only at the user, or master board, and thus each slave board could ignore what the other motor control boards had sent. The method to send data up the arm will be looked at first. With each interrupt, the master board would send the entire packet of motor position information for all slave boards, prepended and appended with start and stop bits to define the packet boundaries, to the first slave board. The information passed from board to board was always encoded using a byte stuffing algorithm (code available on DVD provided), developed by Richard Whittemore [77], in order to ensure that there was zero ambiguity between the start and stop bits and the actual motor position data. The slave board, with each occurrence of the interrupt, would initially check that there were at least 42 bytes (the size of an entire packet of data) present at the serial buffer before searching for the start bit. LabVIEW does not offer a serial read timeout therefore precautions needed to be taken in order to ensure that the code did not attempt to read serial data if no data was present at the serial port. Thereafter, the slave board would read each byte checking for the start of the packet boundary. Each byte read was immediately transmitted to the next slave board up the arm[4]. Once the slave board located the start character, each byte was inserted into a string array for decoding and written to the following slave board until the stop byte was identified. The relevant motor positions in this decoded string array were then extracted.

After completing the necessary transmissions up the arm in the timer interrupt, the slave boards would then look at the information present in the other direction. Each slave board would read each byte transmitted down the arm looking for the stop character (except the last board which would initiate the communications down the arm). As before, each byte read was then immediately written to the following board in the chain. Once the stop character was found it was removed, the current slaves encoded motor position information was written and a new stop character was appended to the packet. The data packet therefore increased in size after each slave board down the arm appended its own data until the complete packet, containing information from all the slave boards, was written to the master board to transmit back to the user. The flow diagram to illustrate this communications process can be seen in Figure 6.6.

[4]Except for the last slave which had no board to transmit up to.

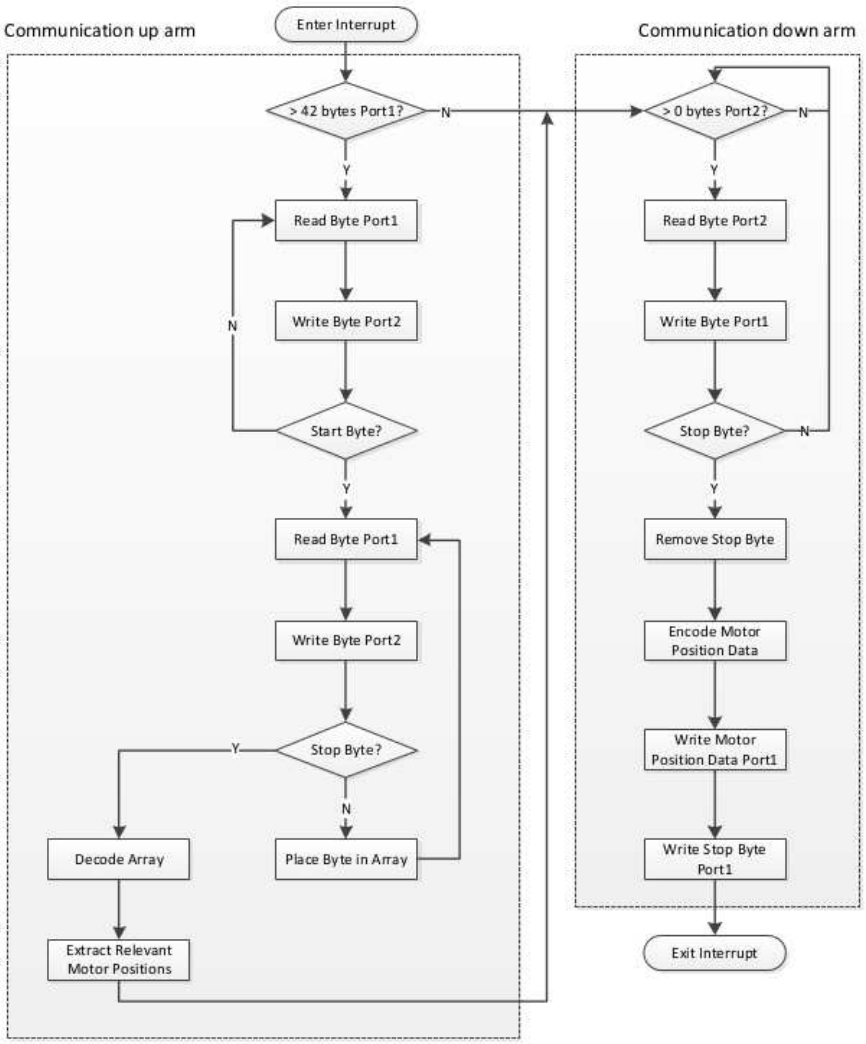

Figure 6.6: RS-232 timer interrupt communications scheme.

Four default values as well as a summation check of all the motor positions was used as an error checking precaution to ensure that the motors in the arm did not move to an incorrect position. The testing of the communications scheme (see Chapter 9) showed a stable communications architecture with a communications speed of just under 20Hz - a direct correlation to the 50ms timer interrupts used. There were, however, complications with the communications scheme. Firstly, the 20Hz communications speed was the rate at which each board received data. However, due to the reliance of the communications scheme on each of the boards, the speed at which a particular board received new data from the user, as well as the rate at which the user received data from that board, was dependent on how far up the communication chain the slave board in question was. It took five iterations of the communications interrupt for the gripper/wrist board (the fifth board in the chain) to receive data from the user as the information needed to be passed between all four prior slave boards. The gripper board, as well as all the other slave boards, did receive a communications packet every 50ms although the design of the communications system meant that, in the worst case, an unfavourable delay of 250ms occurred. Secondly, the reliance of the

communications scheme on each of the boards meant that debugging the system became a challenge. With I²C, each board could be communicated with independently and any errors could be immediately identified. In contrast, the co-reliant RS-232 architecture was more complicated in that if there was an error or if a board was faulty, the problem could not be narrowed down to a specific slave device.

The RS-232 communications scheme, even with the communications delay, offered satisfactory control of the arm. The debugging and independent communication benefits of a pure daisy chain protocol were, however, too appealing to ignore and thus the RS-485 communications protocol was investigated.

6.5 RS-485 on the LM3S8962 Board

The RS-485 transmission standard is a communication scheme that is robust against electrical noise and can be configured in a daisy chain architecture [78]. It was on this basis that this communications standard was used in the RATEL robotic manipulator; it offered the same flexibility offered by the I²C daisy chain yet was robust against motor noise. The LM3S8962 motor control boards were therefore re-designed to incorporate RS-485 ICs, instead of the previous versions MAX232 ICs, and make use of the same UART ports.

A half-duplex RS-485 architecture was developed using a single master, multiple slaves system similar to that used in the I²C communications scheme discussed in Section 6.3. A simplified diagram of the communication architecture in the arm can be seen in Figure 6.7.

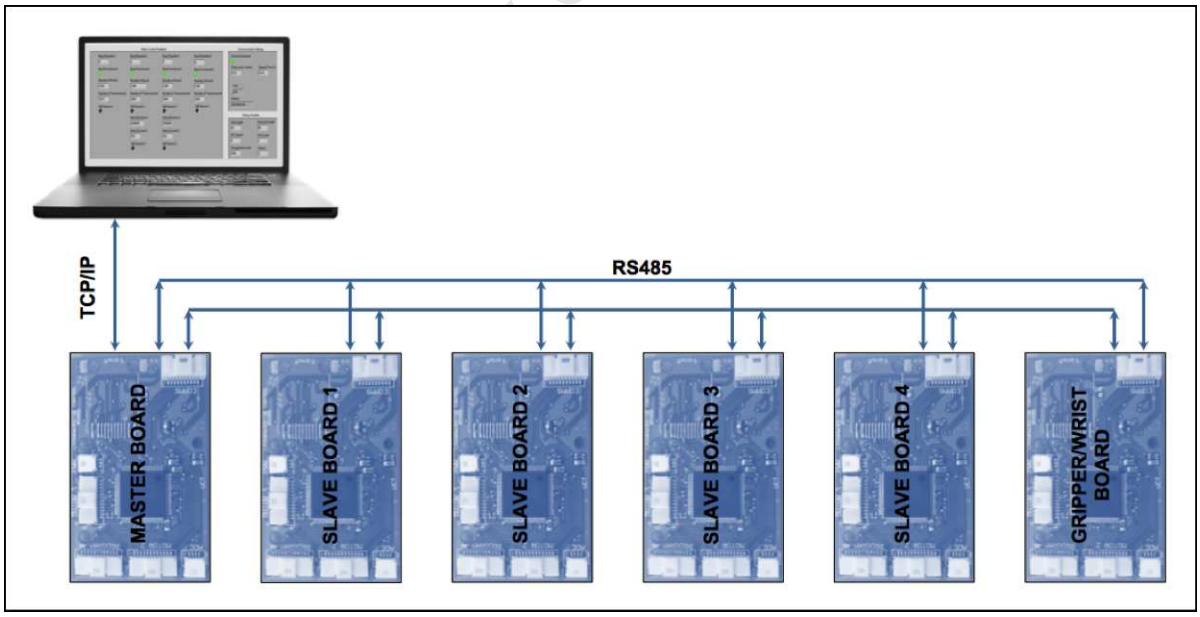

Figure 6.7: Simplified diagram of the RS-485 communications architecture.

The RS-485 communication protocol only allows for a single device to write data at a time dictated by a manually set enable/disable pin on each RS-485 IC. Therefore, an iterative address and reply communications scheme was implemented in which the master board would address a particular slave board. The slave whose address matched

the master's transmission would then receive the data and reply with information of its motor position. The other slave boards would simply ignore any data not sent to their address, including that written back to the master by other slave devices.

The timing of the communications sequence was crucial in creating a stable communications scheme. Most notably, the enable/disable pin of the RS-485 IC needed to be set high for long enough to allow all the data to be written without interruption yet short enough so as to not create unnecessary delays in the communications scheme. Figure 6.8 shows two incorrect timings used in the communications scheme and the eventual correctly timed transmission.

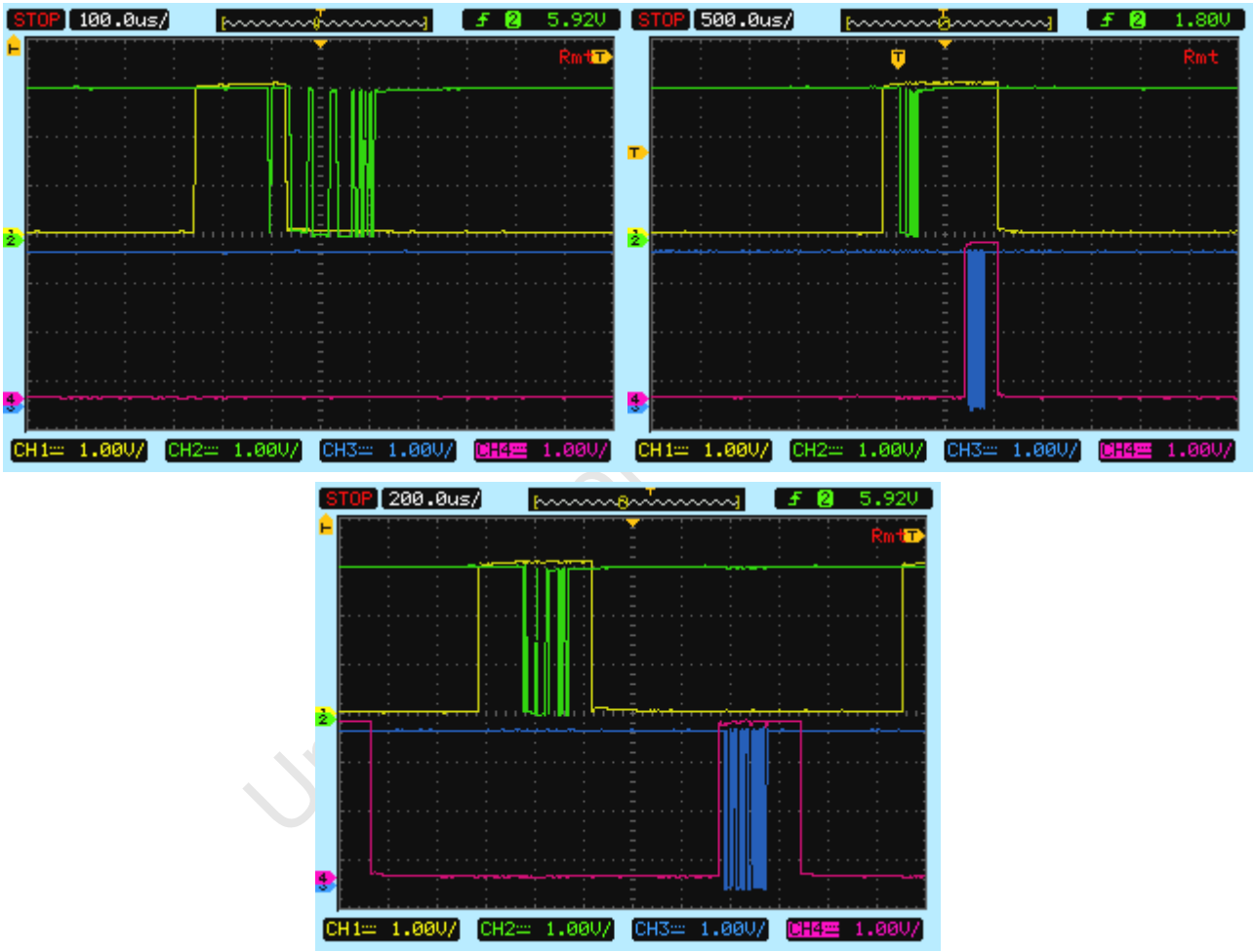

Figure 6.8: RS-485 transmission timing. Top left shows the master enable/disable pin (yellow) cutting off the duration of the data transmission (green) and hence, no response from the slave. The top right figure shows the master enable pin exceeding the duration of the slave response thus causing errors. The lower figure shows a correctly timed transfer from master to slave and a slave response back to the master board.

The benefits of creating a communications scheme that is independent of the motor control code, as discussed in section 6.4, once again provided the motivation to place the slave RS-485 communications in an interrupt of its own. Due to the architecture of the communications scheme in which the slave board only needs to respond when addressed by the master, a timer-based interrupt, as used for RS-232, was deemed to be inferior for the current application. Instead, an interrupt triggered on the presence of data at the serial port was investigated.

LabVIEW Embedded for Arm Microcontrollers does not offer the use of serial interrupts. Therefore a manual interrupt was created by looping a wire from the RX (receive) line of the RS-485 communications to an I/O pin of the LM3S8962. A falling edge on the RX line provided the indication to the slave board that data to be read was present. The slave board would then enter the I/O interrupt and proceed to read the information present on the RS-485 transmission lines.

There were complications in the implementation of this interrupt routine. It was discovered that the time taken for the code to enter the I/O interrupt was in excess of $50\mu s$ (see Figure 6.9) which caused several interrupts to be queued, a significant waste of processing power. Another consequence of the queued interrupts was a stack overflow error caused by the excess number of interrupts stored in the microcontroller's memory stack which would render the board useless until a restart occurred. A solution to the problem was to decrease the baud rate at which the communication scheme was run and introduce delays into the communications code in order to minimise the number of queued interrupts. This slowed the overall communications speed by approximately 5Hz but allowed for a stable system.

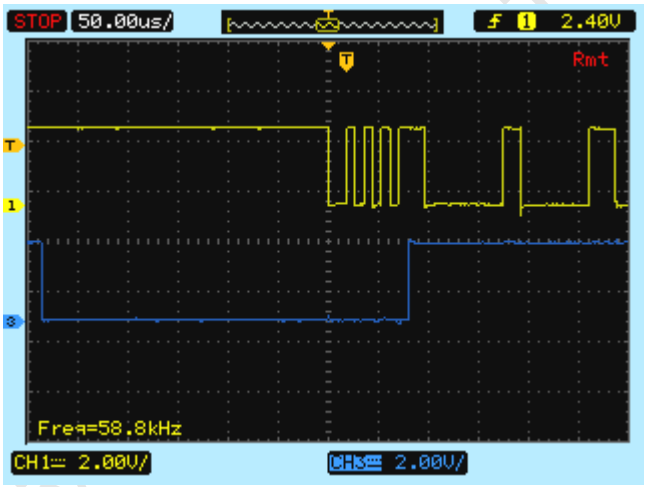

Figure 6.9: RS-485 interrupt initiation delay. Yellow (upper trace) indicates the data present on the RX line of the slave board. The rising edge of the blue line (lower trace) indicates the point at which the slave eventually enters the interrupt routine; sufficient time for several interrupts to queue.

It was decided to keep the size of the packet transmissions in the RS-485 protocol to a constant 20 bytes. The length of the transmission delays could then remain at a constant value in order to simplify the communication scheme. A checksum error catching technique similar to that used for RS-232 was used to guarantee the validity of the data transmission at the slave device. The data packet transmission up the arm, from the master, therefore took the following format:

Information	Purpose	Length [bytes]
Address	Specific to a slave board	2
Function	Dictate normal operation or execute calibration procedure	2
Motor 1 Position	Desired position for motor 1	4
Motor 2 Position	Desired position for motor 2	4
Checksum1	Addition of both motor positions	4
Checksum2	The number 12345	4

Table 6.1: RS-485 transmission packet up arm.

The reply transmission from each of the slaves was structured as follows:

Information	Purpose	Length [bytes]
Board Number	Indicate board number	1
Board Functional	Indication that the slave board is operational	1
Motor 1 Position	Position of motor 1	4
Motor 1 Current	Motor 1 current draw	4
Motor 1 Hall Sensor	Indication of hall effect trigger motor 1	1
Motor 2 Position	Position of motor 2	4
Motor 2 Current	Motor 2 current draw	4
Motor 2 Hall Sensor	Indication of hall effect trigger motor 2	1

Table 6.2: RS-485 transmission packet down arm. Dummy information was inserted into the motor 2 positions for slave boards only controlling a single motor.

It was decided to place the TCP/IP and RS-485 communications in parallel loops on the master board to allow the communications up the arm to operate independently of the TCP/IP network. Much like the system used by team Pasargad in the 2010 RoboCup Rescue league [60], the master board would iteratively address each slave with the most current information transmitted from the user over TCP/IP and thereafter wait for a response. If the slave board did not reply within an allotted time (300 cycles of a no operation for loop) the transmission was ignored. The board functional indicator was manually set to false and the next slave in the system was addressed.

6.6 Summary

This chapter has looked at three different communication protocols used in order to transmit motor positions to each of the motor control boards in the RATEL manipulator arm. Inadequacies in the former two communication schemes resulted in RS-485 being the preferred choice. Testing of the RS-485 communications scheme (see Chapter 9) showed a stable yet slower (13Hz compared to 20Hz developed using RS-232) communications speed than that produced by the RS-232 protocol discussed above. However, the lack of a communication delay of up to 250ms, like that incurred by the RS-232 architecture, together with the debugging benefits of the RS-485 system however made it a far superior communications scheme.

The development of the electronics necessary to make these communications schemes possible, as well as other manipulator arm functions, will now be discussed in Chapter 7.

Chapter 7

Electronics

7.1 Introduction

Effective control of the RATEL manipulator was not possible without the development of key electrical components in the system. This chapter firstly looks at the development of the LM3S8962 motor position control boards and then discusses the use of hall effect sensors in the arm in order to calibrate the position control of the motors. The chapter concludes with a look at the mounts used to house all the electrical elements in the arm.

7.2 The LM3S8962 Motor Control Boards

The LM3S8962 motor position control boards were developed in the Robotics and Agents Research Laboratory by Tracy Booysen with assistance from several students involved in the development of the RATEL robot. Figure 7.1 shows the different generations of boards that have been developed. A brief description of the functionality of each generation follows.

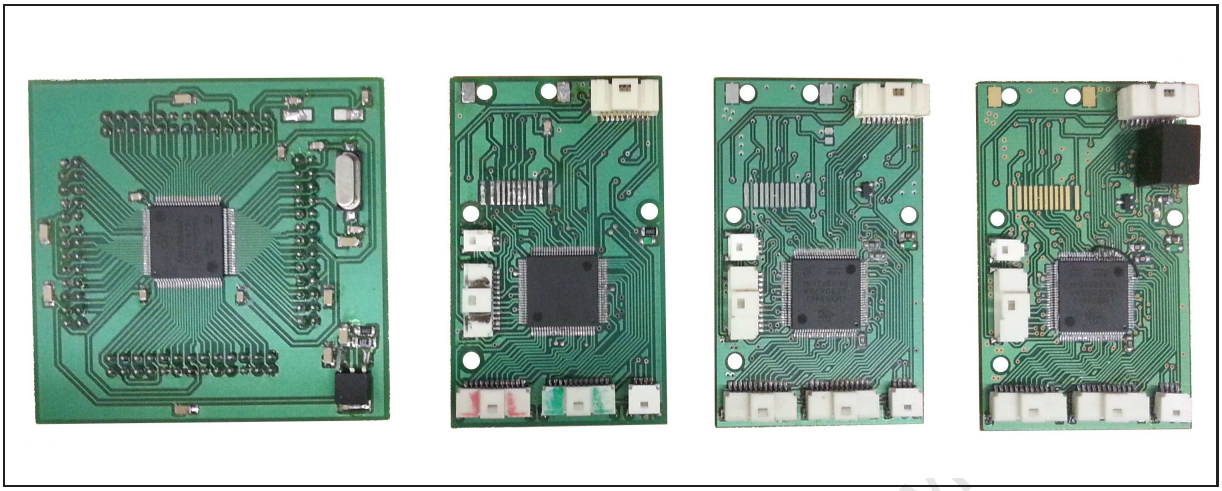

Figure 7.1: Development of the LM3S8962 motor position control boards. Left to right shows the progression from the first generation to the final fourth generation.

Generation 1: This generation provided the pure functionality test phase of the development of the boards. Simple I/Os were provided to test the functionality of the LM3S8962 micro-controller.

Generation 2: The second board developed was the first board to offer all the motor control functionality needed to control motors in the arm. This included the addition of a Digital to Analog Converter (DAC) to dictate the speed and current draw of the motors in question.

Generation 3: The DAC used on the generation 2 control boards was changed to simplify the boards. A 12V port (in addition to the 5V) was added to provide a stable reference voltage for the DAC. A more notable change on this board was the protection circuity added to the quadrature decoding lines to prevent an over-voltage spike (regularly produced by the quadrature encoders on the Maxon EX22 motors) causing any damage to the processor.

Generation 4: The two previous generations of LM3S8962 motor control boards were equipped with RS-232 functionality. In the fourth generation, this was replaced with the RS-485 IC. The fourth generation board was also powered off 18V instead of 5V. An on-board DC to DC converter (20V to 9V) was placed on the board and regulated down to power the LM3S8962. There were two reasons for this. Firstly, assembly of the robotic arm showed a fault in the voltage supply to the motor control boards in the end effector. The accumulated resistance in the length of wire necessary to assemble the manipulator dropped the 5V supply voltage to below the required 4.7V necessary to run the LM3S8962 micro-processors. The micro-processor in the gripper/wrist assembly was thus rendered useless. As an initial solution, a 32V to 5V DC to DC converter was placed midway up the arm (in the second link below the pan/tilt electronics assembly) to boost the LM3S8962 supply voltage. The inclusion of the on-board DC-DC converter resolved this issue. Secondly, testing of the robotic arm showed a significant disturbance to the LM3S8962 upon a sharp stop or change in direction of the smaller Maxon motors. The board affected would become unresponsive and negate any motor control functionality. Further investigation showed that the sudden change in direction or stopping of a motor would affect the supply voltage (see Figure 7.2) to the LM3S8962 and thus cause the board to brown out. The addition of significant decoupling capacitors proved to diminish the issue however the on-board DC to DC converter was included in the fourth generation motor control board to prevent such faults occurring as the converter smoothed the input voltage to the LM3S8962.

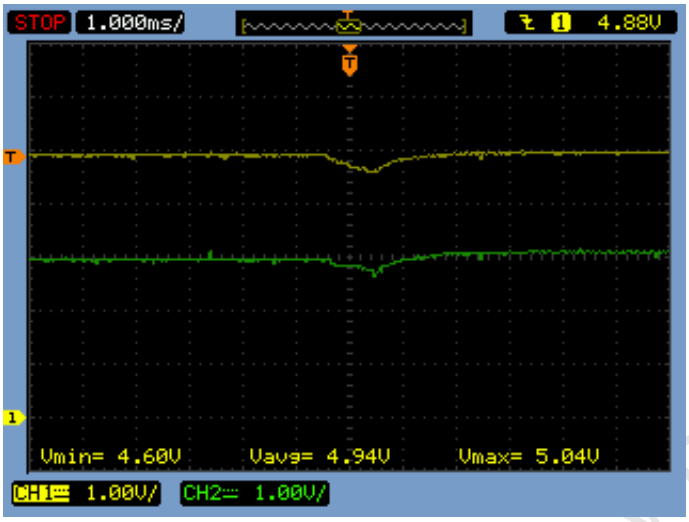

Figure 7.2: Brown out voltage of LM3S8962 motor control board. Yellow shows the 5V supply voltage to the LM3S8962 board. Green indicates the 3.3V supply to the micro-processor. The supply to the board dipping to 4.6V (indicated by Vmin in the bottom left) at the occurrence of the motor undergoing a sudden stop can be seen.

7.3 Hall Effect Sensors

Quadrature decoding is a relative position control technique. The position of the motor is therefore only relative to the initial position, from when power is provided to the system, and not to the absolute geometry that the arm is in. Magnets, fixed to the gear of each joint, and Hall effect sensors (see Figure 7.3) were therefore positioned on each joint of the arm to calibrate (or zero) the manipulator links to ensure that the motor position in the control scheme correlated accurately to the actual position of the arm.

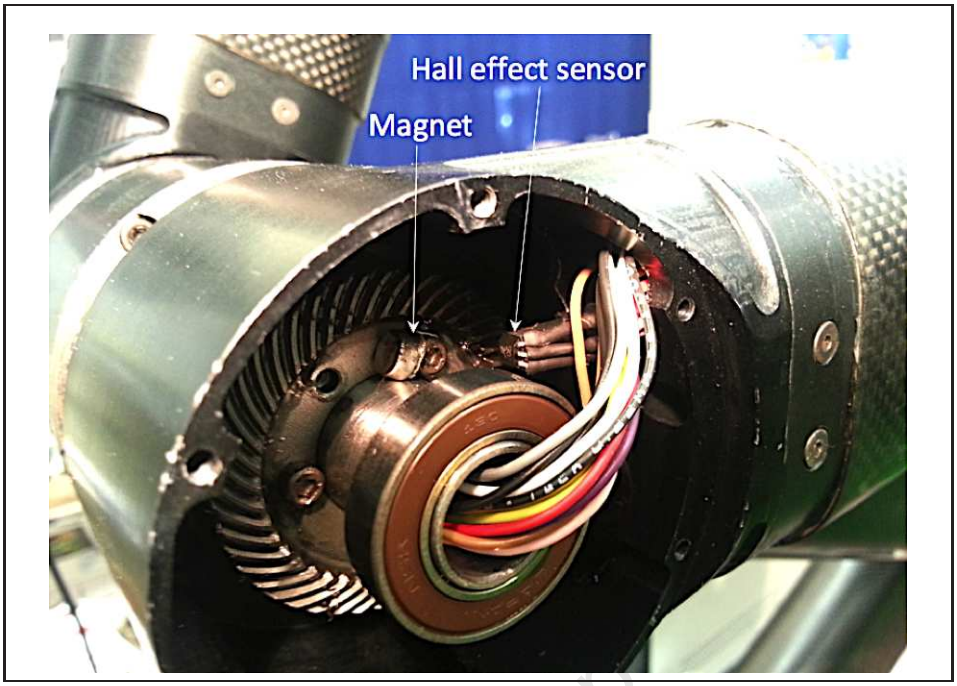

Figure 7.3: Magnet and Hall effect sensor located on the sixth motor gear assembly as can also be seen in Figure 7.10.

The Hall effect sensors used were Allegro A1101 unipolar digital Hall Effect switches. The sensors have a preset threshold and switch on when a sufficiently strong magnetic field is experienced. As there was only a single Hall effect sensor per manipulator joint (a result of Henson's design), prior approximate knowledge of where each Hall effect triggered was needed to move each link to a position above where the magnet was situated, as shown in Figure 7.4(a). Each link was thereafter independently, automatically zeroed by allowing the user to select which motor to calibrate. The motor control board, upon receiving the command to calibrate from the user, would then execute the calibration code described below. This functionality was added to the testing front panel of the RS-485 communications scheme (see Figure 7.5) where each link of the arm could be moved independently. This allowed the user to easily move the arm to the required general pre-calibration position.

The code on the motor control boards used to calibrate each link (see "RS-485 communications" on the DVD provided) was activated by altering the "function" value of the data packet sent to that slave board (see Table 6.1 in Section 6.5 as well as the motor control code on the DVD). Once the calibration sequence was activated, the motor moved towards the Hall effect trigger point at a medium speed. Upon the Hall effect being triggered, the motor would stop and proceed in the opposite direction at the slowest possible speed until the Hall effect sensor was out of range of the magnet. Thereafter the motor would move in the original direction, again at the slowest speed to ensure trigger accuracy, until the Hall effect triggered once more. The trigger locations of the Hall effect sensors, due to their placements on the gearing of the arm joints, were not at the exact desired zero positions of the arm. Therefore, once an accurate trigger position was established, the motor was moved the known distance to the zero position from the accurate trigger location. The calibration process can be seen in the video on the DVD as well as in Figure 7.4. Once a link had been calibrated, the functionality of the calibration button was disabled in order to remove the possibility of the button being accidentally pressed. This would potentially cause massive damage to the robot if the link went "looking" for the Hall effect trigger point once it was below the trigger location.

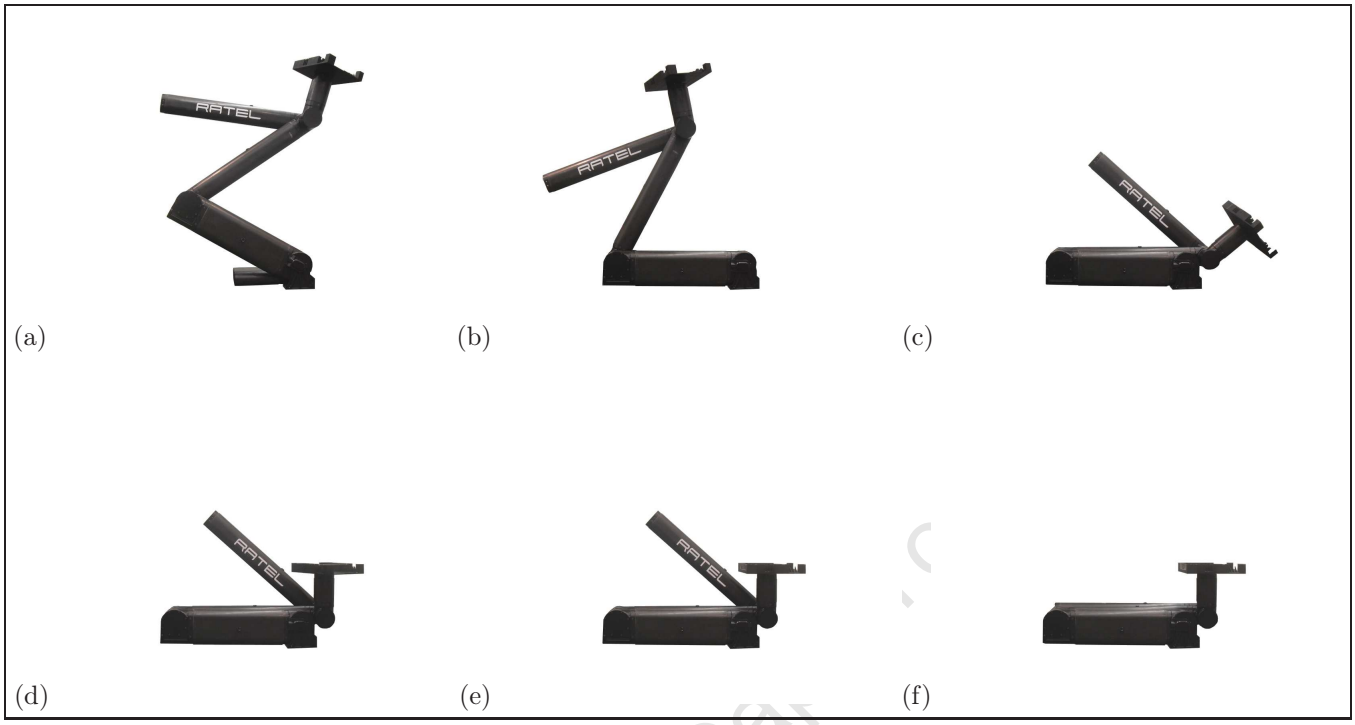

Figure 7.4: Calibration process of RATEL robotic arm. (a) Default position. (b) Link 1. (c) Link 2. (d) Tilt. (e) Pan. (f) Link 3. Only the base of the currently disassembled sensor payload was used to display the calibration procedure.

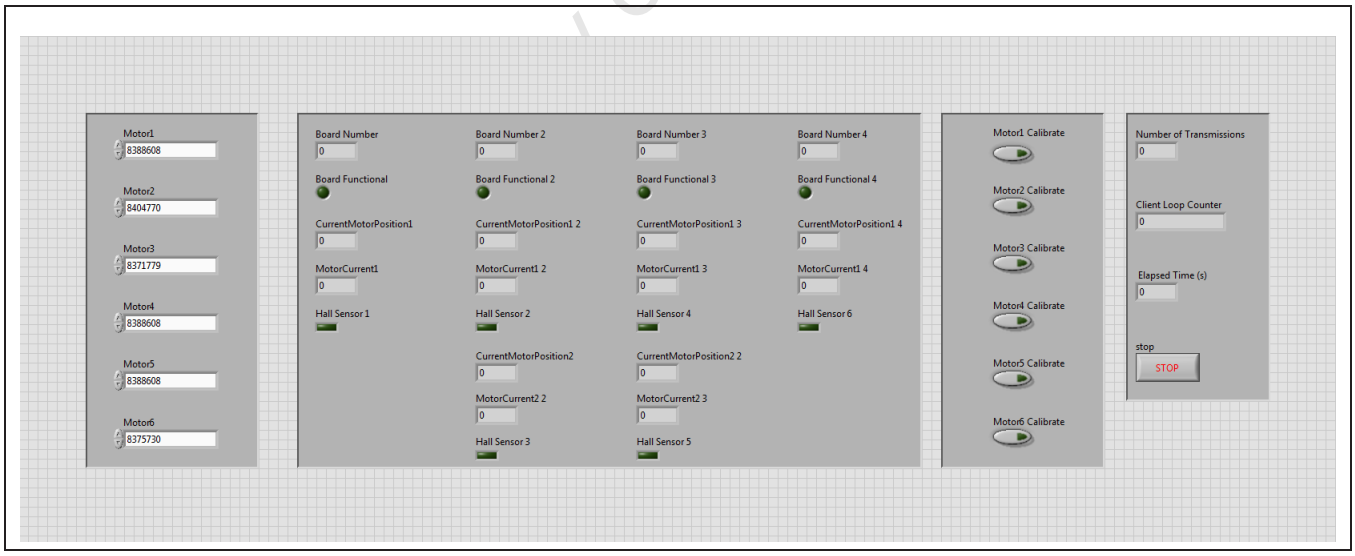

Figure 7.5: Testing/calibration front panel for the RS-485 communications scheme. The calibration functionality can be seen second from the right.

The Hall effect sensors needed to be securely positioned in order to prevent any movement during the operation of the robot. If moved, inaccurate trigger positions may be produced which would jeopardise the accuracy of the calibration algorithms. Mounts to fasten the Hall effect sensors as well as the motor control electronics were therefore developed.

7.4 Electronic Mounts

7.4.1 Motors 1, 2 and 3

Peter Henson [19] designed the identical motor housings for the first three motors with enough space to accommodate an LM3S8962 as well as the Maxon DECV 50/5 motor speed controller. Henson designed preliminary perspex mounts for the electronics however these proved to be too brittle and impractical for use in the arm. They were therefore redesigned. The perspex fasteners designed by Henson were broadened and cut from aluminium to eliminate the brittleness of the previous design. The Hall effect arm was also broadened and redesigned so as to not interfere with the gearbox functionality. The electronics mount was completely redesigned to simplify the mounting of the DECV 50/5 and LM3S8962 onto either side of the plate. The assembly as well as an exploded view of the first three electronic mounts can be seen in Figure 7.6[1].

[1] SolidWorks parts and drawings can be found on the appended DVD

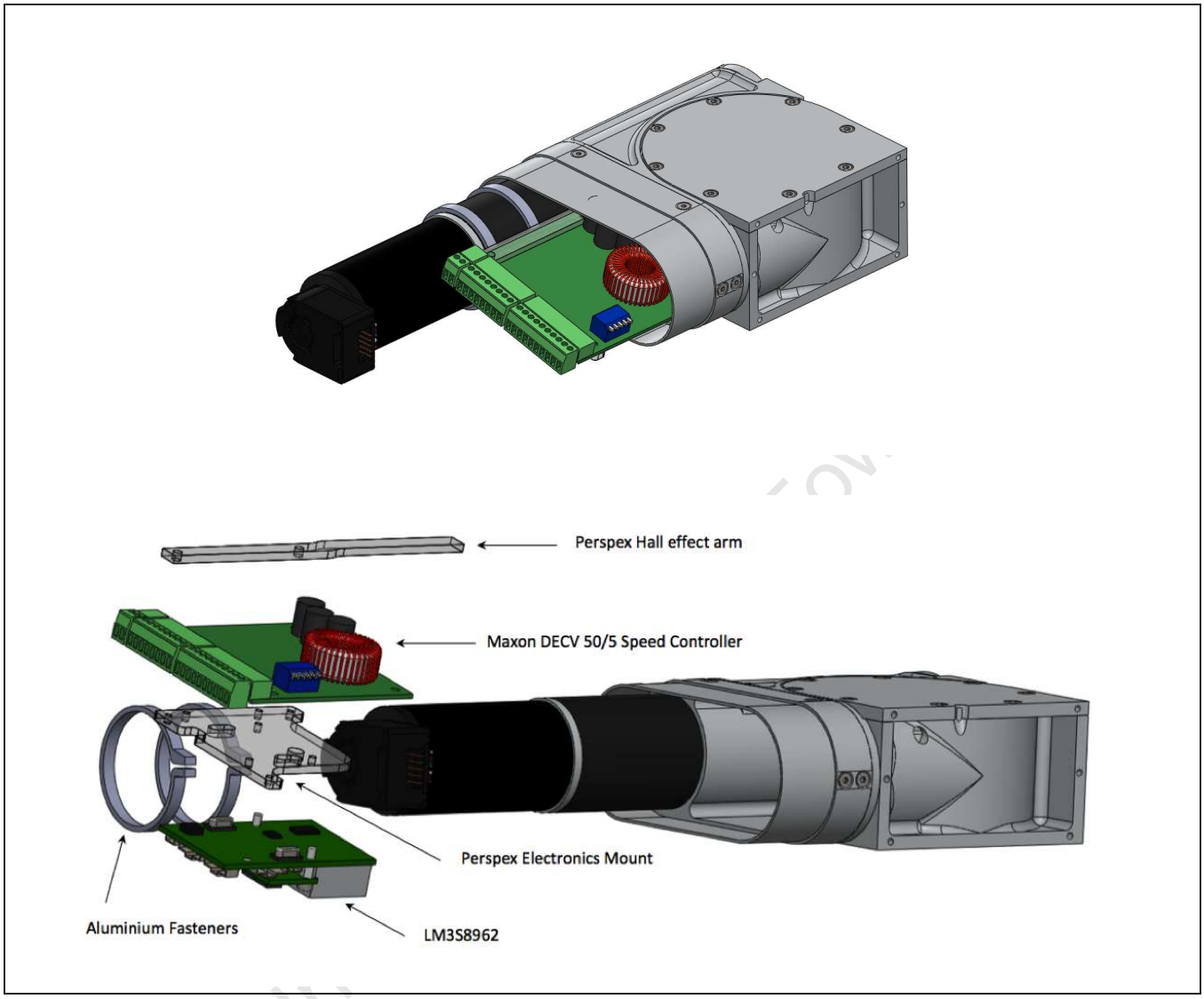

Figure 7.6: Electronics assembly for motors 1, 2 and 3.

The completed assembly can be seen in Figure 7.7

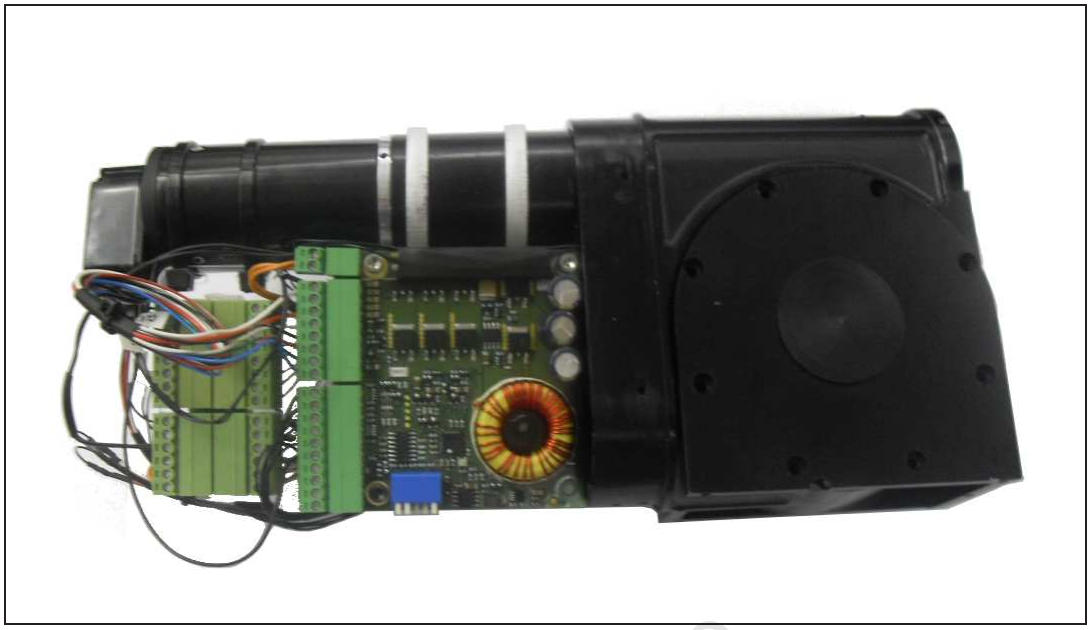

Figure 7.7: Completed electronics assembly of Motor 1. The intermediate connection board, discussed in Section 7.4.3, can be seen to the left of the DECV 50/5 mounted on its own perspex plate.

These mounts effectively fastened the electronics to the motor body thus creating a fixed position for the Hall effect sensors.

7.4.2 Pan/tilt Motors

The original pan/tilt electronics mount designed by Henson did not incorporate the use of the Maxon DEC 24/3 speed controllers and was therefore discarded[2]. The redesigned electronics mount can be seen in Figure 7.8

[2]The original motors used for the pan/tilt system, discussed in Henson's thesis, had incorporated electronics and thus did not require the use of speed controllers

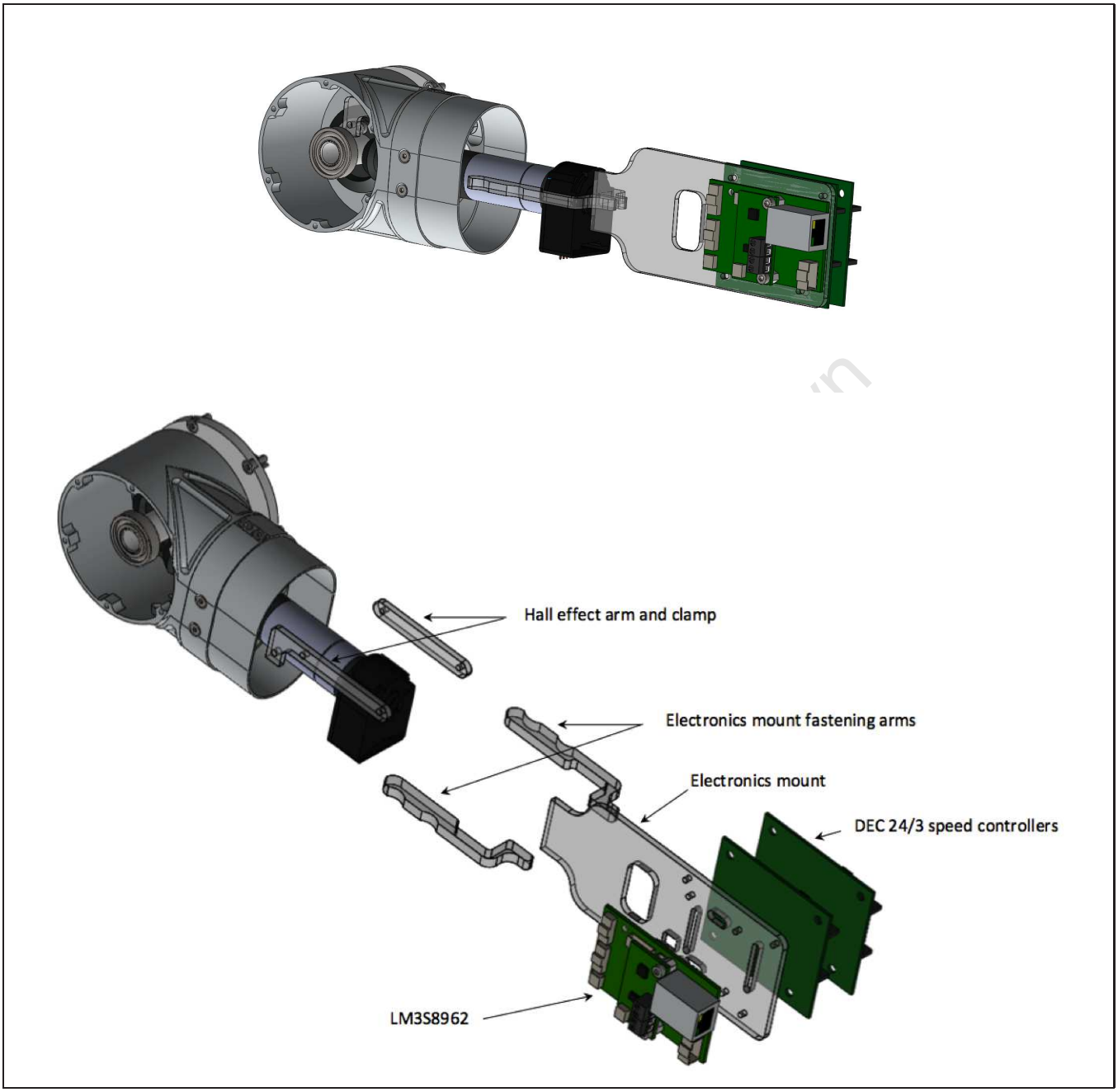

Figure 7.8: Pan/tilt electronics assembly.

As done for the first three motors, the tilt motor Hall effect sensor was glued to the Hall effect arm in order to detect the magnet located on the joint gear. The body of the motor housing was used to effectively clamp the Hall effect arm providing the required fixed sensor position. The electronics mount fastening arms were bonded to the electronics mount using tensol and thereafter cable-tied to the motor body providing the mount for the DEC 24/3 speed controllers and the LM3S8962. The pan Hall effect sensor was positioned inside the pan motor housing in order to detect the magnet located on the internal gear system. This can be seen in Figure 7.9.

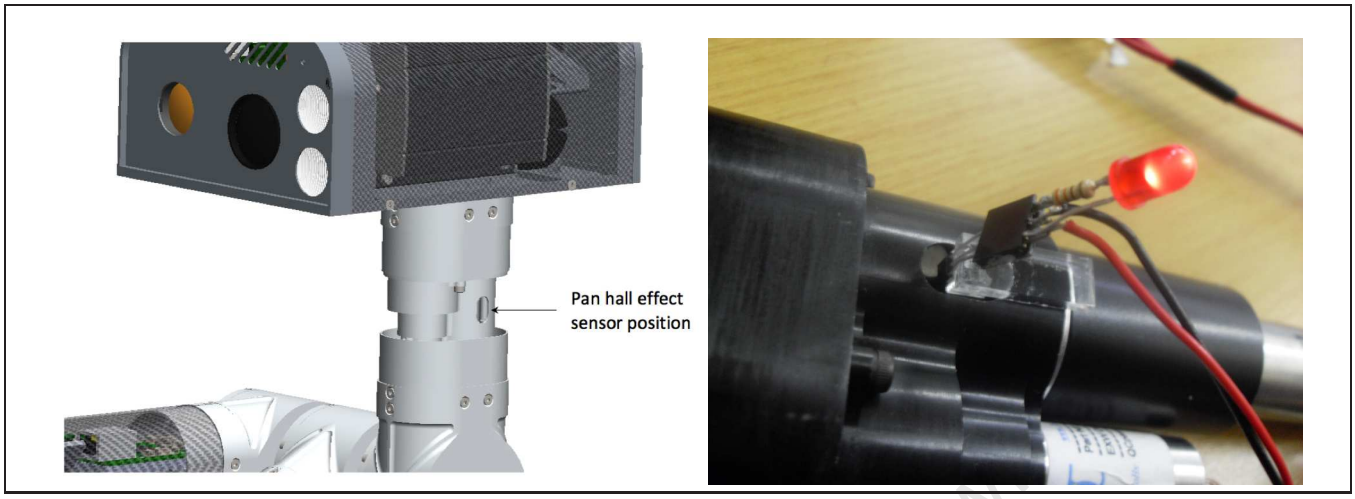

Figure 7.9: Positioning of the pan Hall effect sensor. The image on the right shows the simple LED testing rig, positioned on the pan Hall effect sensor, used to verify the activation of each of the Hall effect sensors in the arm.

7.4.3 Motor 6

The space restrictions in the last link of the arm, caused by a combination of the sixth motor and the gripper/wrist assembly, dictated that the electronics assembly be as compact as possible. The final design is shown in Figure 7.10.

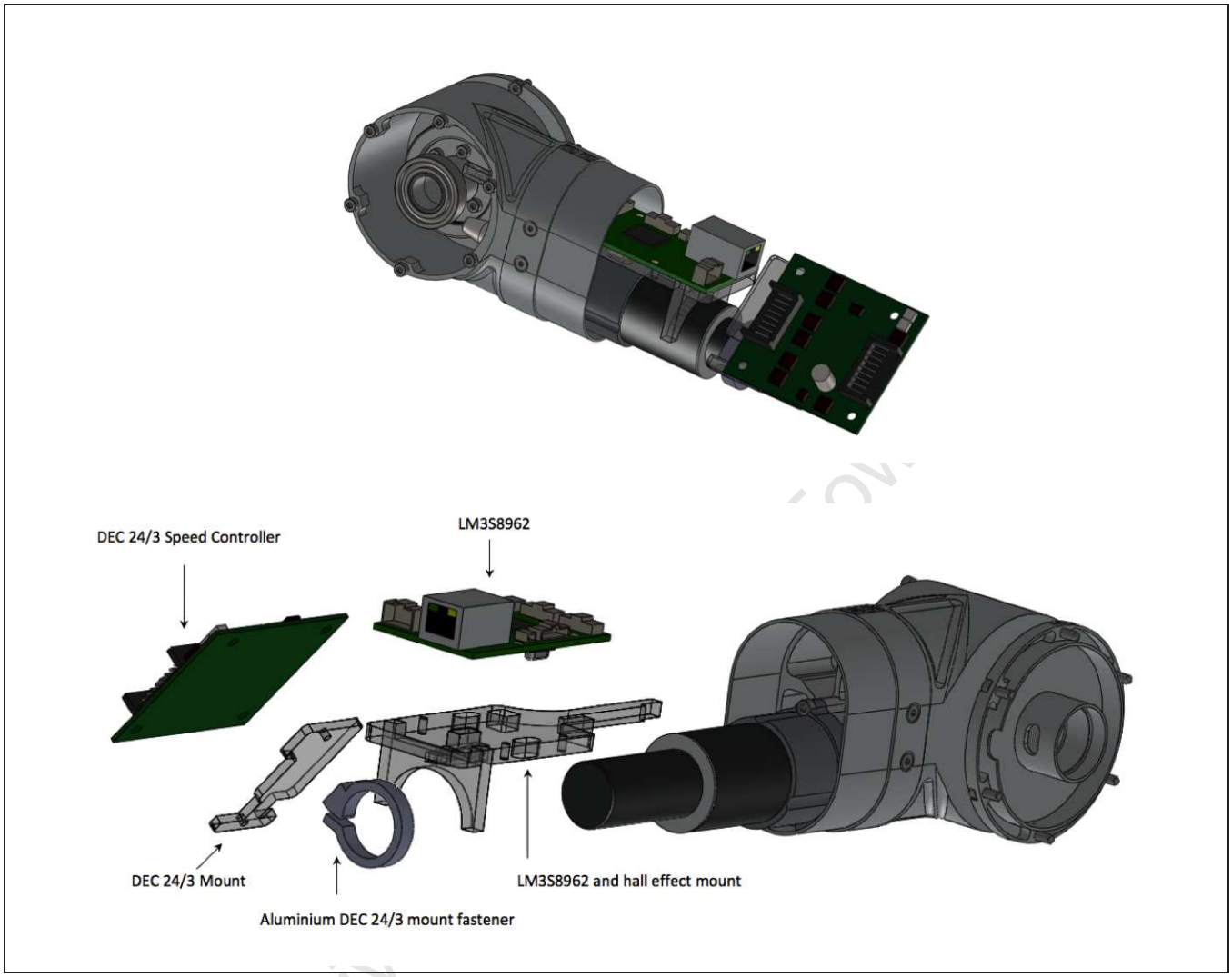

Figure 7.10: Motor 6 electronics assembly.

The LM3S8962 was mounted above the motor body to minimise the space used by the electronics assembly. The three piece perspex LM3S8962 and Hall effect mount, bonded together with tensol and fastened with cable ties, was designed to snuggly fit the motor and thus create the stable positioning for the Hall effect sensor. The Maxon DEC 24/3 speed controller was screwed to its own perspex mount and fastened to the motor body with an aluminium bracket similar to that used for the first three motors.

The wiring of the arm was thereafter completed (The complete wiring diagram as well as the slip ring wiring breakdown can be found in Appendix A). Intermediate connection boards, as shown in Figure 7.7, comprising of removable screw terminal blocks were developed to allow for easy separation of each link of the robotic arm.

7.5 Summary

This chapter has looked at the development of the electrical components needed to control the RATEL manipulator. The LM3S8962 motor control boards provided an effective platform on which to implement motor control algorithms. The Hall effect sensors allowed for the calibration of the arm thus fulfilling the requirement stated in Section 1.2 and the perspex and aluminium mounts provided the secure housings needed to accommodate the electronics internally. With the manipulator fully functional, the operator interface to control the arm will now be discussed.

Chapter 8

Teleoperation Interface and Collision Prevention

8.1 Introduction

The teleoperation interface consists of the elements that the operator uses to control the robotic arm. It includes the physical instruments that the operator manipulates to send movement information to the arm and the Graphical User Interface (GUI) which provides the operator with the necessary system information to effectively control the robotic manipulator. The development of these systems will be discussed in this chapter. The chapter concludes with the development of the collision prevention algorithms used to avoid collisions between the robotic arm and other elements of the robot.

8.2 User Interface

8.2.1 3D Mouse

Ryu et al. state that,

> A joystick-type device, which is commonly used for teleoperation, is adequate for navigation, but is inadequate for controlling the manipulator because it does not have an adequate number of degrees of freedom. Therefore, six DOF haptic devices are often applied to a mobile manipulation task because haptic features allow easier operation of this type of task [33].

It was on this basis that the SpaceNavigator 3-dimensional (3D) mouse, developed by 3DConnexion (see Figure 8.1), was used to control the robotic manipulator. The 3D mouse, although not a haptic device as referred to by

Ryu et al., provided control over 6 DOF and thus offered a highly intuitive interface for the user to control the robotic arm.

Figure 8.1: SpaceNavigator 3D mouse produced by 3DConnexion used to control the RATEL robotic arm [8].

As the RATEL robotic manipulator acts in a single plane, the pan left/right and roll axes were ignored. The zoom axis was used to adjust the x-axis of the inverse kinematics calculations (intuitively, extend and retract the arm). The pan up/down was used to move the end effector position up and down (z-axis adjustment) while the tilt axis was used to alter the angle of attack of the third link of the arm. Finally, the spin axis was used to rotate the arm left and right.

8.2.2 Keyboard

Together with the 3D mouse, the computer keyboard was used to provide additional functionality in controlling the manipulator. The number pad arrow keys (numerals 8,6,4 and 2) were used to control the pan/tilt of the sensor payload while the central arrow keys were used (by Michael Rieger [18]) to control the wrist and gripper assembly. To help the operator control the manipulator, keys were provided to lock different axes of arm movement. Five locks were used to control the arm movement, as listed below:

- A: Retain Angle. Used to lock the angle at which the last link acted upon the desired end effector position.
- R: Retain Rotation. Used to lock the rotation of the arm.
- F: Fix Pose. Used to lock all axes of arm movement and only allow rotation.
- X: Retain X. Used to lock the x-axis movement of the arm, thus only allowing the end effector to move up and down.
- Z: Retain Z. Used to lock the z-axis (or height) movement of the arm and thus only allowing the end effector to extend or retract at a constant height.

By locking any of the axes, the code (see "Graphical User Interface" on the DVD provided) would retain a constant value of the axis variable sent to the inverse kinematic calculations (see Chapter 4) and thus effectively "lock" the axis at the desired position. Each key was selected for its reference to a letter in the function (highlighted in blue) to allow ease of association with the action it performed. The keys were also selected for their location; as they are clustered together to the left of the keyboard, it is easier to select a lock function than it would be if the keys were spread out over the keyboard. Two more keys were used to lock the tilt of the sensor payload at certain positions, as shown below:

- G: Lock tilt at Gripper. Lock the sensor payload to focus at the gripper.
- L: Lock tilt at the Vertical. Lock the sensor payload to focus at the horizontal.

The angle of the sensor payload was dependent on the angle of the first two links of the arm. These were therefore taken into account to calculate the necessary tilt angle in order to produce the desired lock position. This functionality kept the sensor payload focused on a desired location whilst the arm was moving. This, therefore, allowed the user to concentrate only on controlling the end effector of the arm and not worry about manually moving the sensor payload to remain in video shot of the desired position thus making the control of the arm far easier.

In addition to the lock keys, preset positions were included to allow the operator to quickly select a default position for the arm. These were set as numeral keys 1, 2 and 3. Use of the number 1 sent the arm to the stowed, home position. Use of the number 2 sent the arm to the 'active position'. Both of these are shown in Figure 8.2. Number 3 re-aligned the rotation of the arm back to the centre of the robot.

Figure 8.2: Preset positions of arm. On the left, the home position of the arm set by key 1. On the right, the active position set by key 2.

8.2.3 Graphical User Interface

As the robot was to be out of sight of the user, a graphical representation of the arm needed to be created. A 3D representation of the arm was originally designed in Matlab, as shown in Figure 8.3.

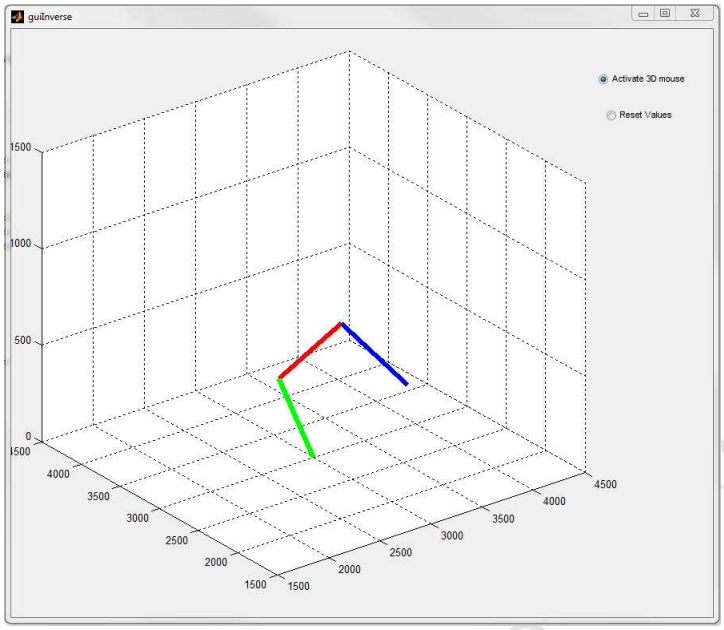

Figure 8.3: 3D representation of arm developed in Matlab.

A freeware Matlab driver [79] was used to communicate with the 3D mouse and update the desired end effector positions. However, it was later decided to use the LabVIEW Robotics Toolkit to create a 3D image of the robotic manipulator as this would better represent the current position of the arm. The final interface can be seen in Figure 8.4.

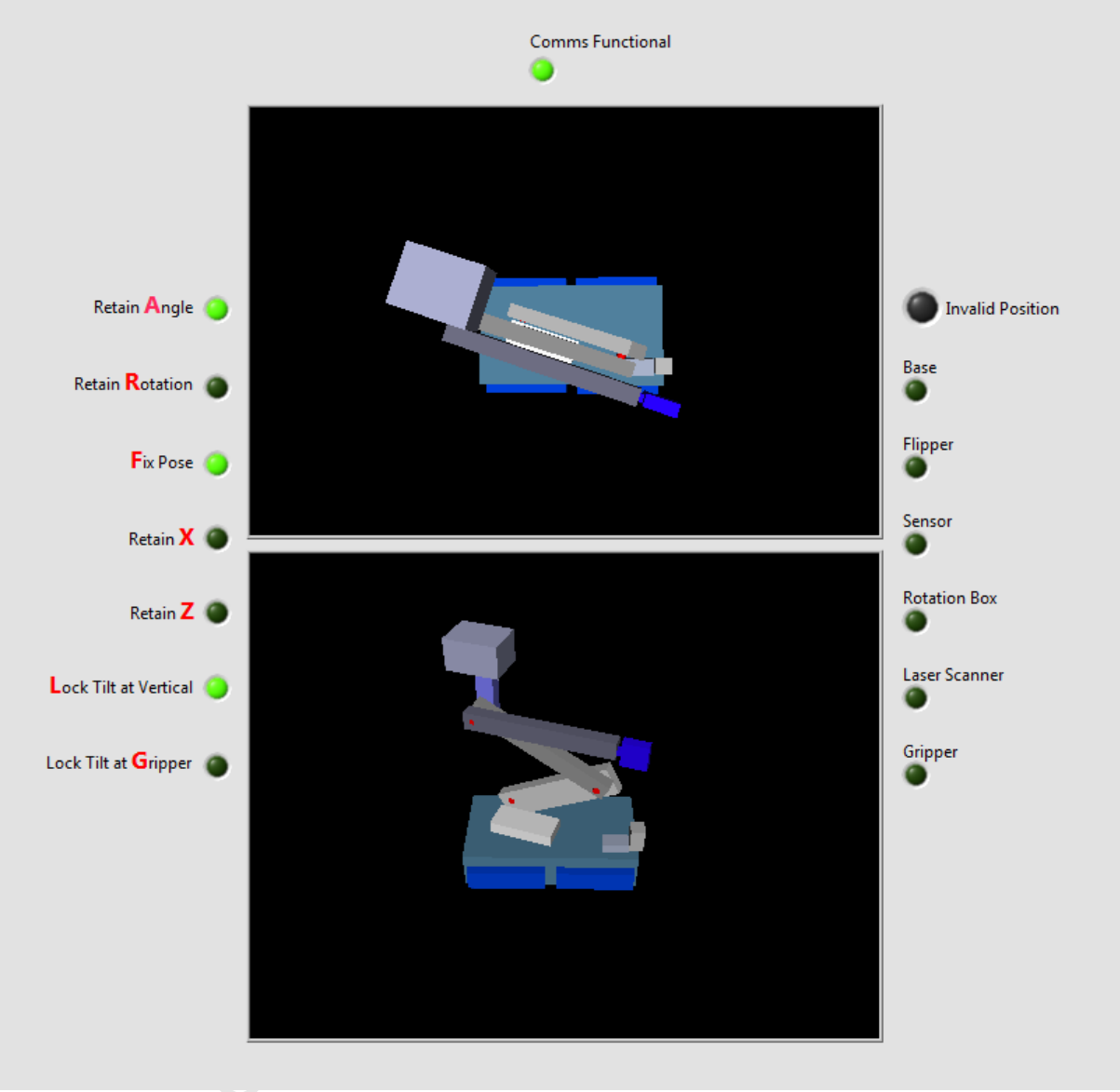

Figure 8.4: GUI for the RATEL robotic manipulator including all elements of the robotic arm as well as the flippers and laser scanner located on the base of the robot. On the left, can be seen LED lights which indicate to the user the axes of movement that have been locked. The right-hand side contains indicators to inform the user of any collisions that have been prevented (discussed in Section 8.3).

8.2.4 User Interface Control Loop

In the master's thesis presented by Filippi [29], a control loop was used which incorporated inverse as well as forward kinematics to improve the control of the robotic manipulator (see Figure 2.12). A similar method was used to control the RATEL robotic arm. Due to the sensitivity of the 3D mouse, it became apparent that the arm would overshoot the desired position once the user had stopped manipulating the input device. The desired end effector position, received from the 3D mouse, would continuously increase while the user was moving the device. Once the arm had reached the required position, the user would release the mouse and expect the arm to stop. The desired position of the arm, however, due to the continuous input to the 3D mouse, differed to the position which the arm

currently assumed. Thus the arm would continue to move, without user input, until it reached the desired position. Even with precise scaling of the 3D mouse input values, this characteristic made controlling the arm difficult. An operator would intuitively want the arm to stop immediately once the input device was released. It was therefore decided to use forward kinematics, in combination with the inverse kinematic calculations discussed in Chapter 4, to scale the desired value of the end effector based on the current orientation of the arm. A simplified control loop of this method can be seen in Figure 8.5.

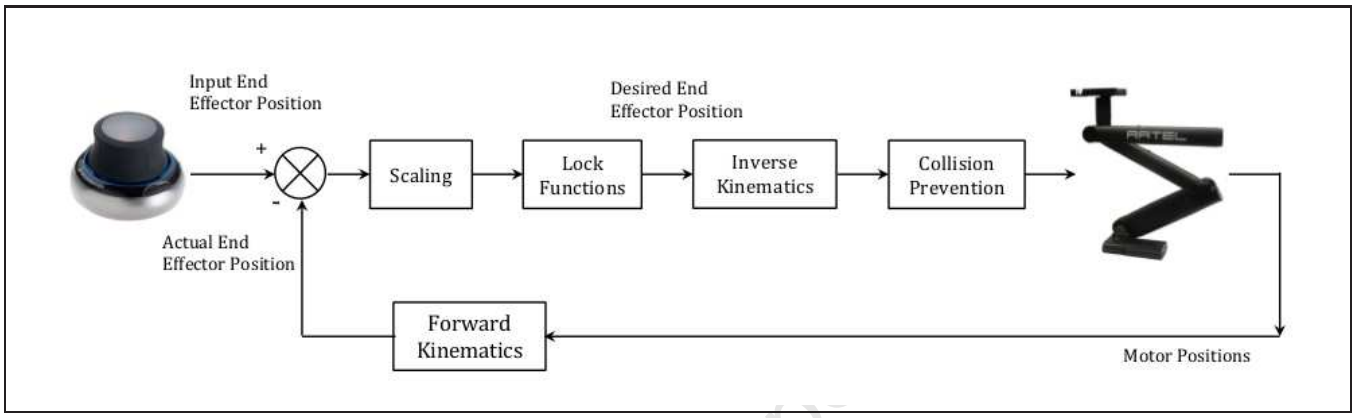

Figure 8.5: Inverse and forward kinematics control loop.

The inclusion of forward kinematics in the control loop allowed for the continuous comparison of the input end effector position to the current end effector position of the robotic arm, thus preventing any overshoot as previously experienced. If the difference between the current end effector position and input position (for all four variables of motion: X-axis, Z-axis, base rotation and angle of final link) exceeded a setpoint (derived through experimentation) the value sent to the inverse kinematic calculations was coerced to a much smaller value[1], thus limiting any overshoot in the desired position. If the difference between input and actual end effector positions was within the determined setpoint, the original value was sent to the inverse kinematic equations (albeit coerced to a larger value to further improve the response of the arm).

The values derived from the scaling calculations were then passed through a check to determine whether any of the lock functionalities (discussed above in 8.2.2) had been activated. If so, the value previously calculated was discarded for the value stored at the time when the lock key was pressed. This desired end effector position was thereafter sent to the inverse kinematic calculations together with the collision prevention algorithms. If no collision was detected at the desired position, the motor positions derived from the inverse kinematics were transmitted to the motors. If the desired position would cause a collision to occur, the motor position values were discarded and the last legal position of the arm was sent to the motors. This, therefore, prevented any collisions from occurring.

A more detailed flow chart of the code can be seen in Figure 8.6.

[1]The size of this value, together with the setpoint, was a compromise between developing immediate response control of the arm and preventing the arm jerking whilst moving. A smaller setpoint and coercion value would result in a faster response. It would, however, cause the arm to jerk since the difference between the input and actual end effector positions would exceed the setpoint more often. A larger setpoint and coercion value would provide smooth arm motion but would deteriorate the response of the arm.

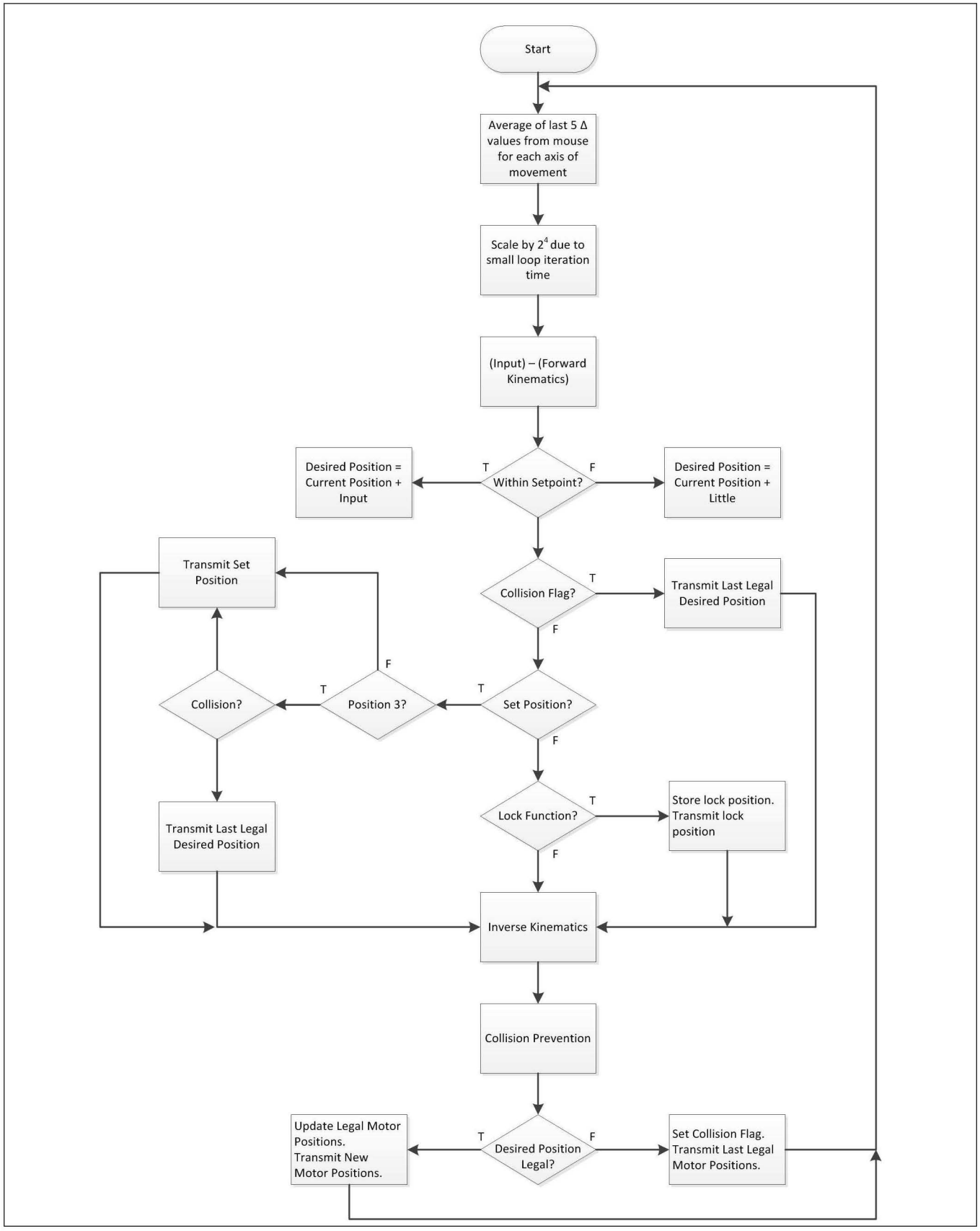

Figure 8.6: User interface code flow diagram.

8.3 Collision Prevention

8.3.1 The Separating Axis Theorem

The Separating Axis Theorem (SAT) was used as the collision detection algorithm for the RATEL manipulator on the basis of the speed and effectiveness of the method as discussed in 2.4. The theorem states that two convex polygons do not intersect if there exists a separating axis between them [52]. This proximity detection algorithm therefore uses projections onto axes to determine whether the bounds of simplistic shapes (line segments, triangles and boxes) intersect. This is graphically represented in Figure 8.7.

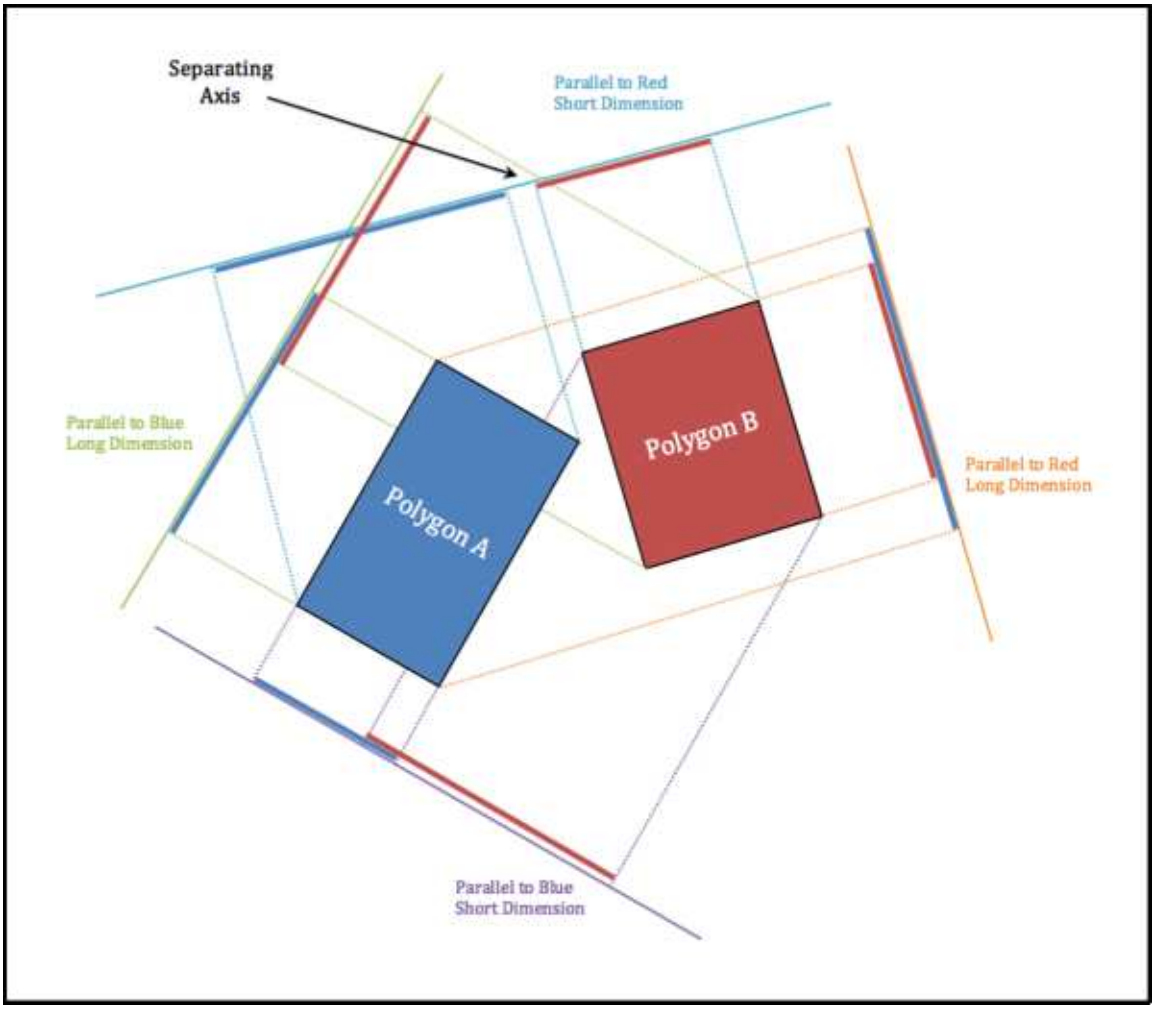

Figure 8.7: Graphical representation of the Separating Axis Test. The boundaries of the polygons projected onto the two axes of each polygon can be seen. A separating axis can be seen on the "Parallel to Red Short Dimension" axis.

Knowing the dimensions and orientations of each polygon, scalar values for the projection of each point of the polygons to each axis are calculated by taking the dot product of the axis vector and the point in question. If the largest "Polygon A" result is smaller than the smallest "Polygon B" result (or vice-versa) then a separating axis exists and the polygons do not intersect.

8.3.2 Determining Robot Bounds

Elements of the RATEL robot were therefore simulated using simplistic polygons to represent the worst case dimensions of the particular robot elements (see the appended DVD for Matlab code). Motor position feedback from the robot was used to modify the simulations to calculate whether the bounds of any robot elements intersected, indicating a collision. To represent the RATEL as the required 2D problem, vertical "slices" of the robot body were taken (see Figures 8.8 and 8.9) based on the angle of rotation of the base arm motor together with the distance from the centre of rotation of the arm segment in question. Three slices were tested for each link of the arm (left, right and centre) to account for any possible collisions.

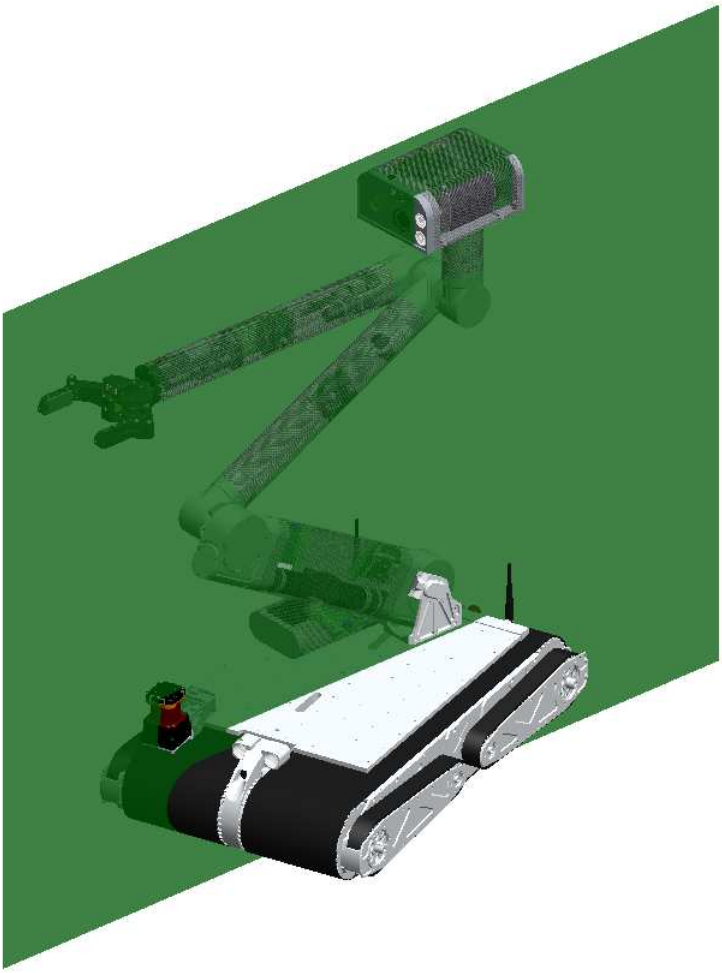

Figure 8.8: A theoretical vertical slice of the RATEL robot taken at the left wall of the first link.

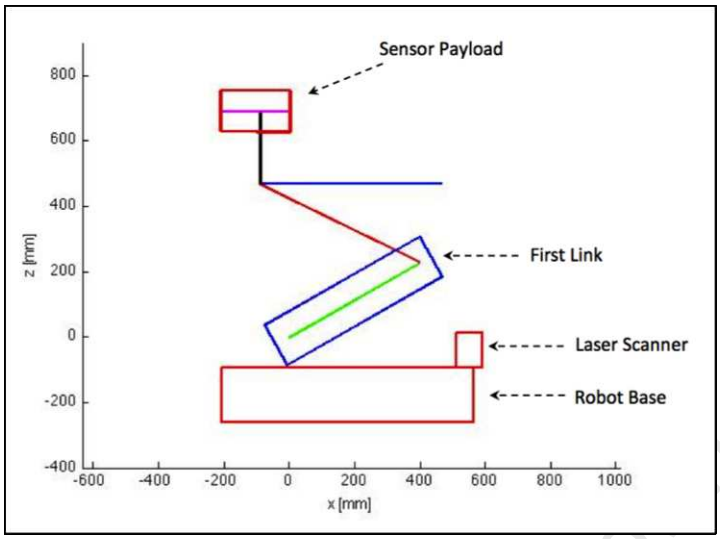

Figure 8.9: Matlab graphical 2D representation of the SAT polygons derived from a slice of the robot dimensions as shown in Figure 8.8. The polygons used for the collision tests are shown as the red and blue boxes representing the sensor payload, the first link of the manipulator arm, the robot base and the laser scanner.

As the sensor payload intersection was independent of the arm rotation, the pan and tilt angles were used to slice the sensor payload accordingly. It must be noted that the inverse kinematic calculations used the points of rotation of each of the links and not the actual bounds of the links to determine the joint angles required to provide the desired end effector position. The simulations of each of the links were therefore extended by the necessary amount past the points of rotation to allow for accurate collision testing. This can be seen in the difference in length between the green line (representing the line between the points of rotation) and the blue box (representing the actual bounds of the link) in Figure 8.9.

The laser scanner located on the front of the base of the robot was modelled using two boxes to account for the worst-case dimensions. The flippers of the robot, due to their rotation, were modelled on the worst-case width of the orientation (see Figure 8.10) in order to determine the point of intersection of the boundary. Once the points of intersection were determined, these were used, together with the angle of the flipper, to determine the accurate height of the resulting 2D polygon to be used for the separating axis test. The 2D representation of the front right flipper in this orientation, being sliced by the right hand exterior of the first link of the arm is shown in Figure 8.10.

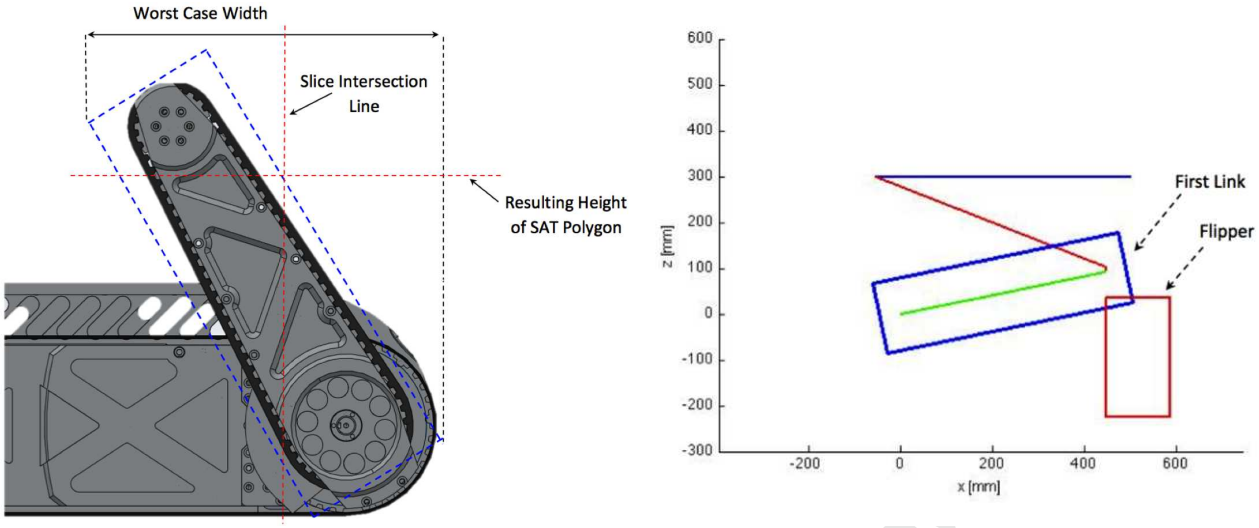

Figure 8.10: Left: Determining the height of the SAT polygon. Right: 2D view of flipper collision with the first link of the arm.

The final collision possibility was the gripper. The same collision tests (excluding the sensor payload) therefore needed to be done based on the orientation and position of the gripper. Due to the varying length and width of the device based on the distance between the gripping arms, the worst-case dimensions were used for the purposes of the collision tests. The worst case height and width were used based on the rotation of the wrist. Slices of the robot body, as done for the manipulator, were therefore taken on these maximum bounds of the gripper (left, centre and right as done previously for each link of the arm). The boundary of the gripper, from the front view, as well as the SAT between the gripper and robot elements can be seen in Figure 8.11.

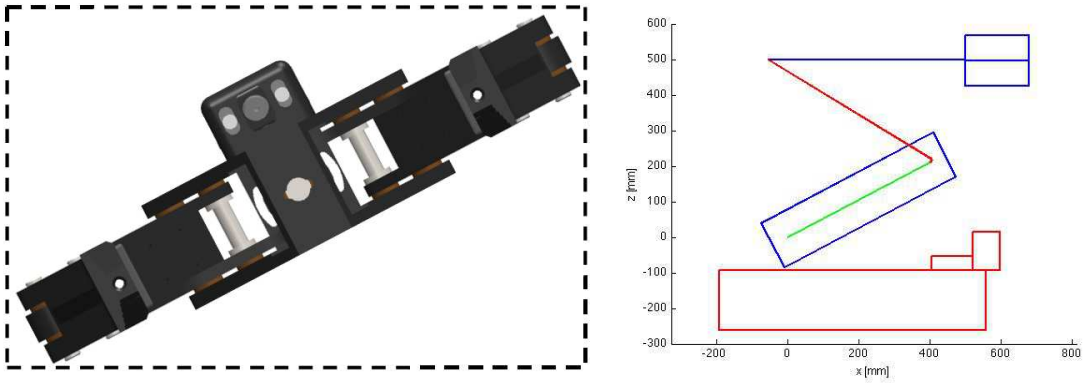

Figure 8.11: Left: Representation of gripper boundaries based on maximum width and height (Image courtesy of Michael Rieger). Right: Graphical representation of the SAT between the gripper and robot elements calculated in Matlab.

An additional collision prevention scheme was implemented for the set position functionality to rotate the arm back to the zero rotation position (activated by pressing key #3, see Section 8.2.2). Initially, the collision prevention scheme would only analyse the desired position of the arm. In this case, when the numeral 3 key was pressed, the desired position was at the zero rotation point at which there were no flippers or laser scanner to collide with. The arm would therefore rotate from its current position, regardless of what might lie in its path, since no collision was

detected. It was therefore decided, once the operator activated this preset position, to segment the path between the current rotational position and the desired position into 50 steps. The collision prevention algorithms were then applied to each step. If all the steps provided a clear path, the arm was allowed to rotate to the zero position. If not, the arm would remain stationary and a warning was emitted to the operator.

If any collisions were detected, the correlating indicator LED would be illuminated on the GUI (see Figure 8.4) to inform the operator of the error. It was found, however, that the operators often did not pick up on the warning lights due to the fact that they were focused on the video images to control the arm (see Figure 9.30). Since the operational instruments lacked any haptic feedback to further notify operators of an impending collision, an additional auditory warning system was used. A freeware LabVIEW Virtual Instrument (VI) (which made use of built in LabVIEW ActiveX functions together with Microsoft SDK speech capability) [80] was used to issue a verbal warning to the operator over a speaker. As an example, if a collision with the sensor payload was detected, a message stating "collision, sensor payload" was emitted to the operator.

8.4 Summary

In this chapter the teleoperative interface as well as collision detection algorithms have been dealt with. The implementation of the 3D mouse and keyboard as the teleoperative interface has been discussed with the addition of the GUI developed in LabVIEW. The control loop making use of both inverse and forward kinematics was also highlighted. The use of a SAT collision detection algorithm was then discussed with particular focus on the methods used to describe the boundaries of the different robot elements.

With the interface completed and collision prevention successfully implemented on the RATEL manipulator, testing of the system could commence. Chapter 9 discusses the testing procedures used and the results obtained from such tests.

Chapter 9

Testing and Results

9.1 Introduction

Once the system was completed, the RATEL manipulator needed to be tested. This chapter discusses the testing procedures used to verify the system against the stated requirements of the project. The chapter initially discusses testing of the motor control used in the robotic arm followed by the testing of the different communications schemes. This is followed by collision prevention testing and the chapter concludes with the usability testing.

9.2 Motor Control

9.2.1 Motor Position Control Accuracy

The accuracy of the motor position control directly affects the ability of the arm to assume the pose dictated by the inverse kinematics calculations and hence the accuracy of the collision detection algorithms. The accuracy of the PD motor control, as derived in Chapter 5, was therefore tested in order to determine the effectiveness of the control scheme.

Motor Stopping Overshoot

Due to the minimum speed of the motor controllers it was impossible for the motor position control to be absolutely accurate; overshoot was an inherent factor in the system. Due to this overshoot, an accuracy deadband needed to be included in the motor control code. The magnitude of each motor's overshoot was tested in order to determine the appropriate size of the deadband.

To test the stopping overshoot of the motors a setpoint desired value was transmitted to the relevant LM3S8962 motor control board via RS-232 (see Section 6.4). The board, upon receiving the value, would then move the motor

to the desired position whilst applying the PD control law, dictated in Chapter 5. The position of the motor was continuously sent back to the client PC for analysis. This test was run ten times for each setpoint value to attain an accurate average. The average overshoot of the first three motors in both a clockwise and anti-clockwise direction is tabulated below:

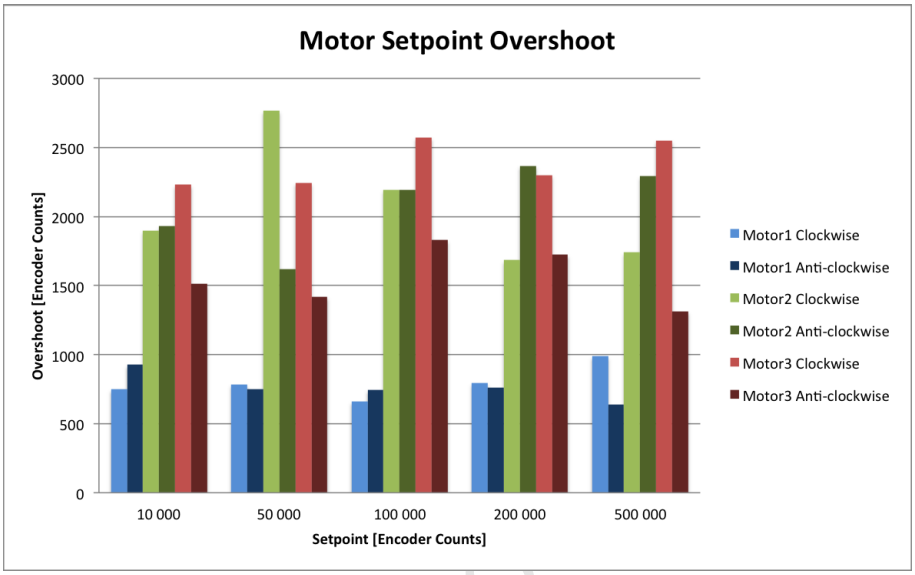

Figure 9.1: Stopping overshoot of motors 1, 2 and 3. As is discussed in Table 9.1, an overshoot of 2500 encoder counts equates to approximately 0.17 degrees.

The results indicate that the magnitude of the setpoint plays no part in the size of the overshoot. It remained uniform for each motor regardless of the distance covered. Secondly, a large difference in the size of the overshoot can be seen between Motor 1 and the next two motors. This is due to the larger forces acting upon these two links compared to that of Motor 1. The shoulder motor (Motor 2) has the moment of the weight of the robot arm acting upon it. This is also the case for the third motor. These larger forces cause a larger momentum which increases the stopping overshoot. It can also be seen that the overshoot of the third motor is larger in a clockwise direction than in an anti-clockwise direction. This is related to the previous statement in that the momentum of the third motor is larger in a clockwise direction as this is when the heavy sensor payload is "dropping". In the opposite direction, the third motor is effectively lifting the sensor payload and hence the stopping overshoot is reduced. The worst case scenario would therefore need to be adopted in order to account for all cases. From the above results it was decided to implement a deadband of 1500 (+750 to -750 to account for movements in both directions) for Motor 1 and a deadband of 3600 (+1800 to -1800) for motors 2 and 3.

The stopping overshoot of the pan and tilt motors was tested and is shown below:

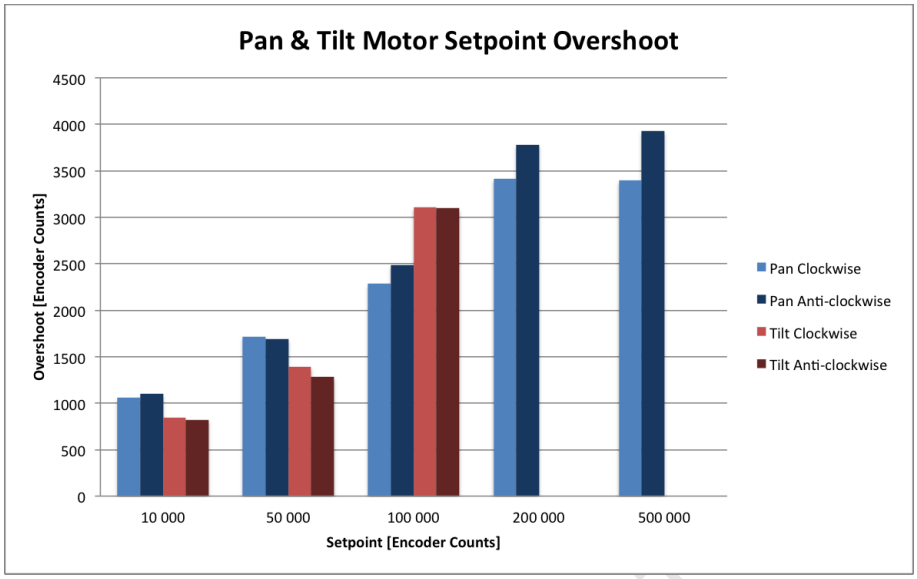

Figure 9.2: Stopping overshoot of the pan and tilt motors. The testing of the tilt motor overshoot was performed from vertical and thus the range of motion was limited to 100 000 encoder counts to not exceed the tipping point of 20 degrees, as clarified in Chapter 5.

Unlike the first three motors, the overshoot of the pan and tilt motors can be seen to be dependent on the size of the setpoint. A best case fit of 4000 encoder counts was therefore implemented for the deadband of each motor.

As described in Chapter 5, the final motor proved to be difficult to control due to the weight of the gripper affecting the ability of the motor to hold the desired position. The extent to which the motor position dipped in a clockwise direction and the motor setpoint overshoot in an anti-clockwise direction was tested:

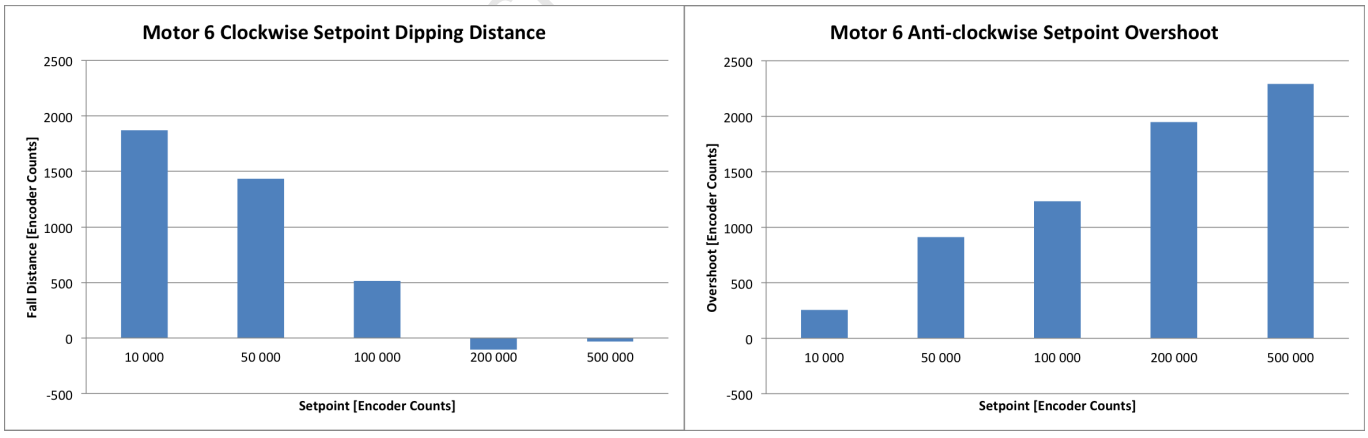

Figure 9.3: Dipping and overshoot error of sixth arm motor. The dipping error (left) can be seen to be negative at the larger setpoints. This is in fact overshoot due to the larger momentum of the link of the arm eradicating any dipping of the motor. As can be seen on the right, much like the pan and tilt motors tested above, the overshoot of the sixth motor increases with setpoint magnitude. From these results, a dynamic deadband of 500 encoder counts was implemented with a static deadband of 3000 in order to remove the dipping cycle discussed in Chapter 5.

Motor Accuracy

Having deduced deadband sizes for each of the motors, the overall accuracy of the motor control was tested.

In the case of Motor 1, with the implementation of the deadband, once the motor gets to within 750 counts of the desired value the stop pin is enabled. As the overshoot is expected to be approximately 750 counts, a fairly accurate final value is expected. As above, the test was run ten times for each setpoint in order to attain an accurate result. The results of the absolute accuracy are shown in Figure 9.4.

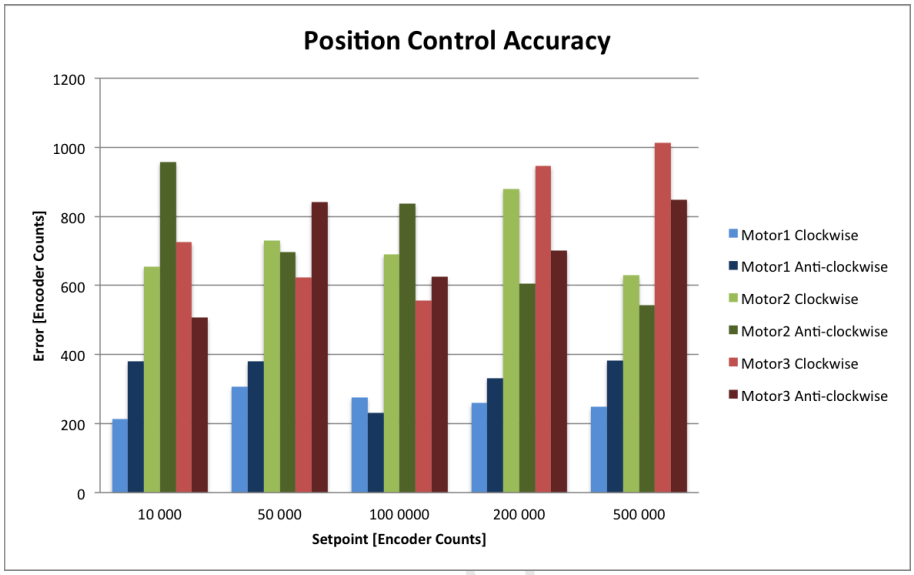

Figure 9.4: Position control accuracy of the first three motors. The absolute value of the error was used therefore making a negative (undershoot) error equivalent to a positive (overshoot) error. It can be seen that the average error of Motor 1 remains below 400 encoder counts. The error of the second two motors can be seen to be approximately double that at 800 encoder counts.

Table 9.1 shows the encoder counts calculations for each link of the arm. This shows that, due to the gearing of both the motor and worm gears in the arm, one degree of movement is equivalent to 14600 encoder counts for both the first and third motors. The second motor, due to the larger motor gearing, equates one degree of movement to 18200 encoder counts. The above results indicate that, on average, Motor 1 will stop within 400 encoder counts of the desired position. This correlates to an accuracy of 0.03 degrees. Motor 2's average accuracy was determined to be 0.05 degrees while motor 3 was equal to 0.06 degrees.

Motor Number	Encoder Counts / Turn	Gearbox Ratio	Decoder Multiplication	Joint Gear Ratio	Counts / Joint Revolution	Counts / Degree	Joint Degrees / Count
1: Rotation	500	73:1	4	36:1	5256000	14600	6.85×10^{-5}
2: Shoulder	500	91:1	4	36:1	6552000	18200	5.49×10^{-5}
3: Elbow 1	500	73:1	4	36:1	5256000	14600	6.85×10^{-5}
4: Tilt	500	531:1	4	2:1	2124000	5900	1.69×10^{-4}
5: Pan	500	531:1	4	1:1	1062000	2950	3.39×10^{-4}
6: Elbow 2	512	190:1	4	45:1	17510400	48640	2.06×10^{-5}

Table 9.1: Encoder counts per degree calculations for each link of the arm.

The accuracy of the pan and tilt motors was similarly tested:

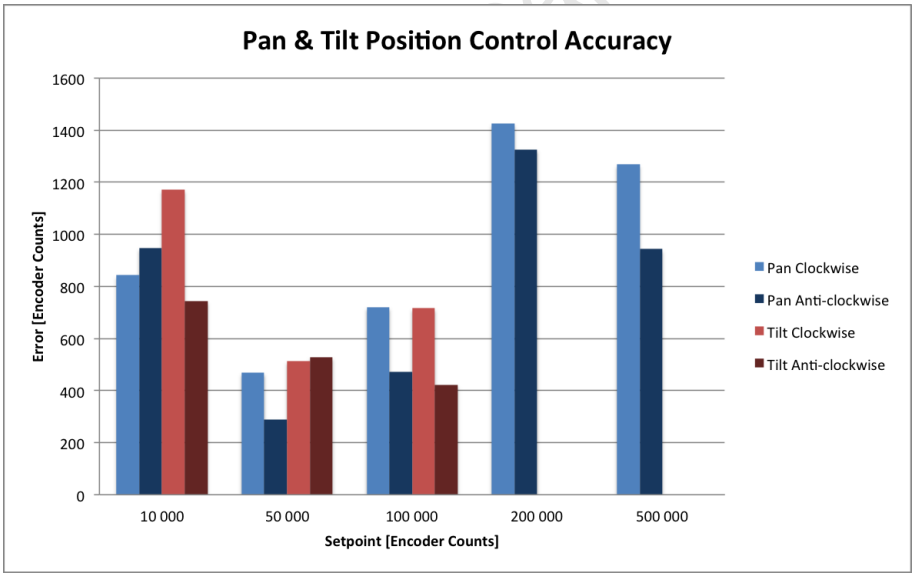

Figure 9.5: Position control accuracy of pan and tilt motors.

One can see that the average accuracy of the pan/tilt motors was determined to be approximately 1200 encoder counts. The pan system had no external gearing and thus had a resulting accuracy of 0.4 degrees. The tilt system, with a 2:1 gearing ratio, had an accuracy of 0.2 degrees.

Finally, accuracy testing upon the final, sixth, motor was executed and the results are shown below:

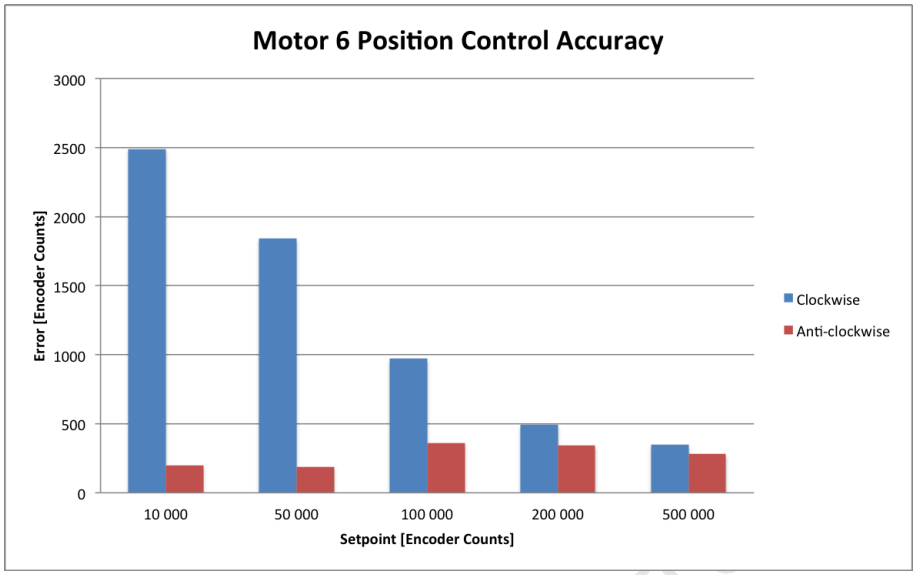

Figure 9.6: Motor 6 position control accuracy. A large difference between the accuracy of the link when moving in a clockwise direction compared to an anti-clockwise motion can be seen. The dipping of the motor position due to the weight of the gripper acting upon the hypoid gear clearly has a large effect upon the accuracy of the link when moving small distances.

The effective accuracy of motor 6, at the worst case error of 2500 encoder counts shown above, was determined to be 0.05 degrees. Motor 6 and the hypoid gearing are both highly geared at 190:1 and 45:1 respectively resulting in the high accuracy even at a large encoder count error.

The overall accuracy of the arm is tabulated below:

Motor #	Worst Case Error [Encoder Counts]	Worst Case Accuracy [Degrees]	Average Error [Encoder Counts]	Average Accuracy [Degrees]
1	750	0.05	300	0.03
2	1800	0.1	800	0.05
3	1800	0.12	800	0.06
4	2000	0.33	1200	0.2
5	2000	0.67	1200	0.4
6	3000	0.06	2500	0.05

Table 9.2: Motor position control accuracy.

The arm at full extension has a reach of 1.548m. A worst case error of the rotation motor, Motor 1, acting at this full extension length equates to an error of 0.0014m or 1.4mm as dictated by the equation

$$L = \frac{\alpha \pi r}{180}$$

where L is the length of arc, α is the angle in degrees and r is the radius or, in this case, length of extension of the arm. This was calculated for each link of the arm individually:

Link Number	Motor Error [degrees]	Link Length [mm]	Link Error [mm]
1	0.1	452	0.78
2	0.12	540	1.13
3	0.06	556	0.58

Table 9.3: Individual arm link accuracy.

These individual errors, compounded, dictate that the end effector should move to within 3.89mm of the desired value.

9.2.2 End Effector Accuracy

The accuracy of the end effector was tested as an indication of the combination of the inverse kinematic modelling accuracy together with the motor control accuracy.

A laser pointer was fixed to the top of the last link of the arm and shone on a white board adjacent to the system, as shown in Figure 9.7, in order to test the overall accuracy of the position of the end effector.

Figure 9.7: Laser pointer mounted on robotic arm in order to monitor movement error.

With each movement of the arm, a fine-point marker pen was used to mark the position of the laser pointer on the white board, as shown above right. These points were then used to determine the distance travelled between movements in order to calculate the position error of the end effector. The end effector was set to a default position of x=468mm, z = 468mm and thereafter sent to different locations whilst maintaining a constant x and then z axis coordinate in order to accurately measure the amount of error. The tables below indicate the accuracy of the end effector.

Desired End Effector Position [mm]	Actual End Effector Position [mm]	Error [mm]
290	287	3
343	341	2
404	403	1
510	510.5	0.5
573	574	1
636	637.5	1.5
695	696.5	1.5

Table 9.4: End effector position whilst maintaining a constant z axis position.

Desired End Effector Position [mm]	Actual End Effector Position [mm]	Error [mm]
329	329	0
416	415.5	0.5
522	523	1
568	569	1
616	617	1
670	670.5	0.5
729	730	1

Table 9.5: End effector position whilst maintaining a constant x axis position

Figure 9.8 indicates the desired movement in comparison to the actual movement of the arm in a sequence of motions. In each alternating movement, the x-axis and z-axis coordinates were locked in order to accurately measure the amount of error.

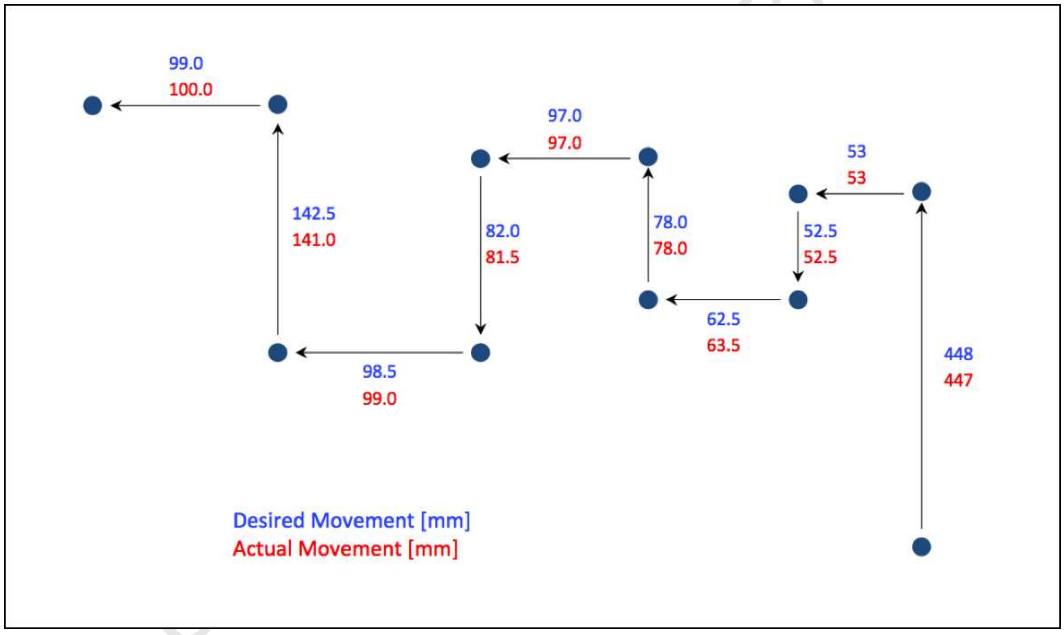

Figure 9.8: Comparison of desired and actual movement magnitudes of the end effector.

It can be seen from the above results that the z-axis motion is consistently within 1.5mm of the desired movement. The x-axis motion is slightly less accurate with a maximum error of 3mm. This yields an overall accuracy of approximately 3.5mm[1].

The drift of the manipulator was another accuracy concern. This was tested by repeatedly sending the end effector to two set positions, shown in Figure 9.9, whilst monitoring the position of the laser pointer, which was mounted on the arm (as above), on a white board placed five meters away in order to exaggerate any drift errors. The position of the laser pointer, upon repeat movements to both set positions (marked 1 and 2), is shown in Figure 9.10.

[1]This may be due to zeroing inaccuracies of the initial position of the arm, slight backlash errors as well as mechanical construction inaccuracies compared to the ideal dimensions of the arm.

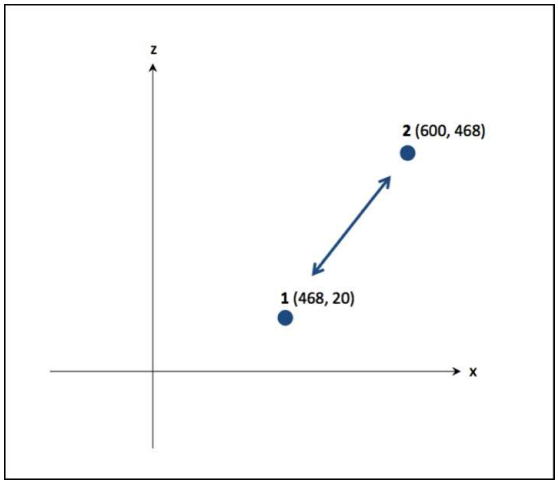

Figure 9.9: Set positions for drift test.

Figure 9.10: End effector drift indicated by laser pointer projected onto white board. Left to right shows initial the position followed by position after five movements and after ten movements.

It can be seen that there is the slightest amount of error in the horizontal axis of the end effector. The error is, however, not consistently moving away from the initial position which indicates minimal drift in the system.

9.2.3 Arm Motion

The motion of the arm, utilising the inverse kinematics, was then tested. The control scheme allowed the user to dictated the x and z coordinate of the end effector as well as angle of attack of the last link of the arm upon that x and z coordinate. Therefore, by locking the end effector in position while changing the angle at which the arm acted upon this position, the end effector could rotate around a stationary point in space. This was tested and the results shown in Figure 9.11.

Figure 9.11: Ratel arm maintaining x and z coordinate with differing angle of attack.

The ability of the arm to maintain an x or z coordinate was then tested. This was done by fixing a pen to the end of the last link of the arm and using the control interface (3D mouse and keyboard) to draw a box on the white board used above. Two boxes were drawn, both of which are displayed below:

Figure 9.12: Drawing boxes with the RATEL arm by utilising the axis lock functionality of the inverse kinematics. The image on the left illustrates the ability of the arm to still accurately maintain the desired motion even once the first link is locked at the horizontal and the two link inverse kinematics is implemented as discussed in Section 4.4.

A slight deviation from the desired line of motion in both images on the bottom line of the boxes can be seen. This is due to the separate links of the arm not reaching the incremental desired positions at exactly the same time, thus causing a brief discrepancy in the end effector position. If there was an unlimited maximum speed and zero

minimum speed for the motors then the PD control rise times of the motors would allow all the links to reach the desired positions at the same time. There are, however, limits to the speeds of the motors and thus, in cases where some links have to move a large distance compared to the smaller distance of others, temporary discrepancies in the lines of motion did occur. In the case on the right, the last link of the arm, because it had a lower maximum speed than the other two links, had to "catch up" with the motion of rest of the arm. This therefore caused a dip in the line of movement of the end effector. It must be noted that these brief discrepancies from the desired line of motion only occurred when the arm was in the process of moving. The desired end effector position, once all the links reached their desired positions, maintained the accuracy discussed in 9.2.2. The images shown in Figure 9.12 together with the videos appended on the DVD, illustrate the smooth motion of the arm when moving to a desired position.

9.3 Communications

9.3.1 I²C Communication Protocol

The I²C communication protocol was the initial communication scheme implemented to send and receive motor control information to and from the LM3S8962 motor control boards in the robotic arm. Due to the instability of the communication scheme upon final implementation in the arm (discussed further below) only preliminary testing was performed. These tests are now discussed.

Communication speed

The communication speed is an important factor in the control of a robotic arm due to the response time of the motors to user inputs. A highly delayed response due to a slow communication speed would make controlling the device difficult and inaccurate. This test therefore determines whether the communications scheme performs according to the desired specifications in order to effectively control the robotic arm.

To test the communication speed the LM3S8962, boards were mounted on a perspex test rig and an I²C daisy chain passed between the devices to create the necessary I²C communication architecture as shown in Figure 9.13.

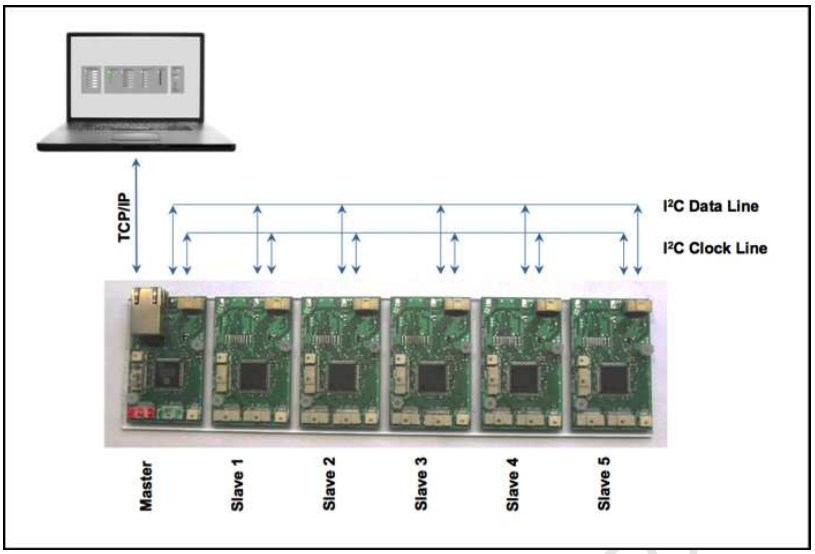

Figure 9.13: Simplified model of the I²C test setup showing client PC as well as Master and Slave LM3S8962 motor control boards.

The client PC program was then run and the information returned from the master board was monitored. The sequential procedure in which the I²C communication and TCP/IP communication was executed on the master LM3S8962 board meant that the master board would have to complete an I²C communication to all slave boards (read and write) before initiating a TCP/IP communication back to the client PC. Therefore, an accurate representation of the communication speed was to increment a counter each time the client PC program received any information. This occurred on each iteration of the program loop. The program loop counter was therefore monitored and divided by the total elapsed time in order to calculate the communication frequency. The testing front panel is shown below.

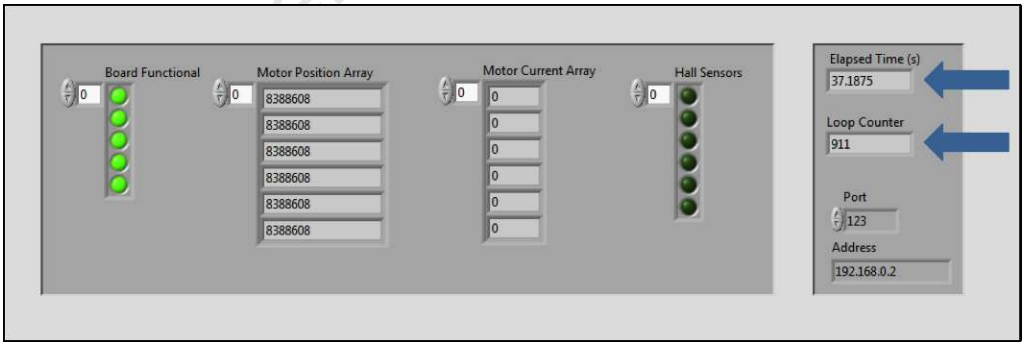

Figure 9.14: The front panel of the I²C testing client PC program. The fields in question are indicated by the two arrows.

It became apparent that the speed of the communications scheme was dependent on the amount of processing that each slave board had to do. This was because the slave boards would perform the motor control code and then I²C code in a sequential manner. Therefore, if the desired motor position was equal to the current motor position, very little processing needed to occur and the program loop would cycle faster which, in turn, would make the I²C communications run faster. On the other hand, if the desired motor position was not equal to the current motor position, more processing needed to occur (Quadrature decoding, PID control) which would slow down the I²C

communications. As no motors were connected to the test system the current motor positions remained at the default value of 8388608. Differing values of the desired motor positions were then set on the slave boards in order to test the effect on the communications speed. The test was run six times with the number of boards "aligned" with the desired motor position decreasing each time.

Boards Aligned with Desired Position	Loop Iterations	Elapsed Time [s]	Frequency [Hz]
5	8049	305.96	26.31
4	7169	285.73	25.09
3	9247	387.42	23.87
2	4295	189.65	22.65
1	5728	267.32	21.43
0	4740	234.54	20.21

Table 9.6: Communication speed results of I²C testing with varying amount of boards in which the Desired Motor Position equalled the Current Motor Position. The proportional decrease in communication speed with a decrease in the number of "aligned" boards can be seen.

The 26Hz communication speed correlates to the oscilloscope image of the I²C data and clock lines shown below:

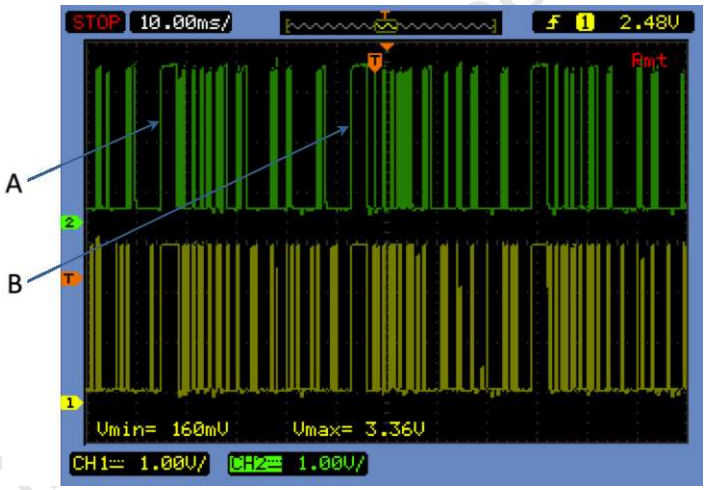

Figure 9.15: Oscilloscope image of the I²C Data and Clock lines. "A" and "B" indicate the start of successive communication transfers between all five slave boards. The scope is set at 10ms/division so one can deduce that a complete transfer takes approximately 38ms. This confirms the 26Hz communication speed.

Communication Stability

As there are a number of slip rings in the robotic arm the question of whether the I²C communications would be affected by such devices came to light. Two slip rings were therefore placed in series in between the master and first slave board with the two I²C lines as well as all power lines (36V, 18V, 12V and 5V) connected in order to simulate the operational conditions in the arm. A Maxon DECV 50/5 speed controller as well as a Maxon EC40 motor were connected to the first slave board in order to create any possible motor noise disturbances on the communication lines. The communications scheme was then run, as before, for considerable durations while moving the motor sporadically in order to determine if the slip rings cause the communications scheme to fail. The test was run three times as indicated by the following table:

Elapsed Time [s]	Communications still active?
1548	Yes
1345	Yes
1630	Yes

Table 9.7: Preliminary I²C stability test in order to determine whether the addition of slip rings and motor control into the system causes the communications to die. In all cases the communications were still active after at least 20 minutes.

A secondary concern was that the length of wires in the arm may affect the stability of the communication protocol. The I²C lines were therefore extended with two 10m lengths of wire in order to create a worst case scenario for the communication lines. The above test was repeated with the same results.

These preliminary tests did not, however, indicate the sensitivity of the communications scheme to increased motor noise. With the implementation of the communications scheme in the arm, the increased motor noise developed by the smaller Maxon EC22 motors and the Maxon 24/3 speed controllers caused the I²C communications scheme to regularly fail. These failures did not occur when only the lower sections of the arm were assembled but began when connecting the pan/tilt motors[2] to the system. The slip ring in the third joint of the arm was removed in an attempt to eliminate the failures however they reoccurred. It was therefore deduced that the increased motor noise together with the proximity of the communication lines to the motor electronics affected the stability of the communications scheme to the extent that it was deemed completely unusable in the robotic arm.

9.3.2 RS-232 Communication Protocol

The RS-232 communication protocol was implemented on the robotic arm as an alternative to the failed I²C communications scheme. A simplified model of the communication architecture in the robotic arm is shown below:

[2]This is the first occurrence up the arm of the smaller motors and speed controllers

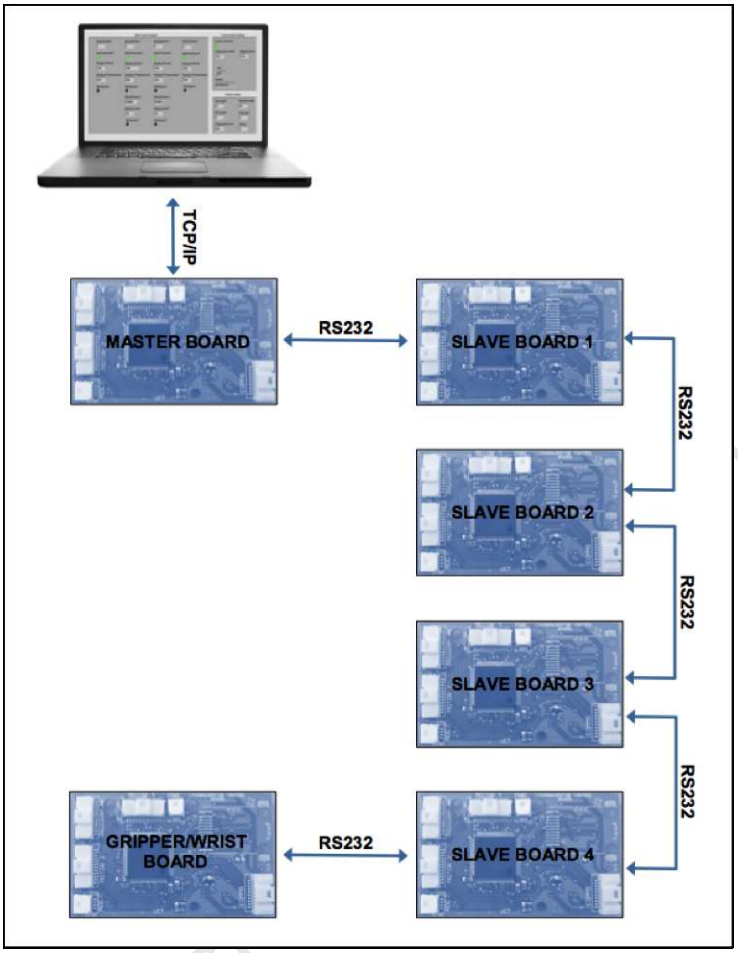

Figure 9.16: Simplified model of the RS-232 communication scheme showing client PC, master LM3S8962 communication board and slave LM3S8962 motor control boards.

The communication speed and stability of the communication protocol was tested. A complication in carrying out these tests was that the communication scheme was based on strict 50ms timer interrupts on each board. This characteristic, together with the fact that the motor control boards were powered off two different voltage lines (32V and 5V) due to the DC to DC converter placed mid-way up the arm, meant that the timing of physically plugging in the different voltage lines affected the stability of the communication scheme. It was apparent that the 32V and 5V needed to be plugged in simultaneously in order for the communications scheme to run as expected[3]. Bearing this in mind, the testing procedures were executed and will now be discussed.

Communication Speed

The communication speed of the RS-232 protocol was identified as the frequency at which the information transmitted from any given LM3S8962 motor control board in the robotic arm arrives at the client PC. This was tested by continuously rotating the wrist of the end effector and monitoring the indicated position of this wrist motor on

[3]A solution to this problem was to place a DC to DC converter upon each motor control board as discussed in Section 7.2

the front panel of the testing GUI. The frequency of communication was determined by incrementing a counter each time the position of the wrist motor changed (as this indicates a successful transmission) and dividing this by the time elapsed during the test. This is shown below:

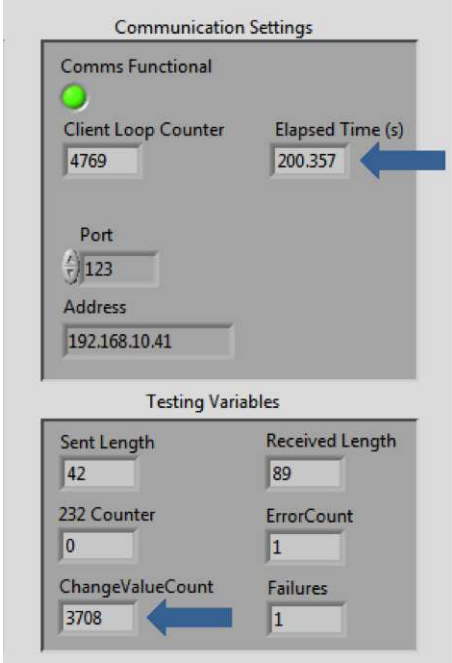

Figure 9.17: Screen capture of the communication section of the testing GUI. The lower arrow indicates the counter which increments with each successful transmission from the end effector motor control board while the upper arrow indicates the elapsed time. The counter value was divided by the elapsed time in order to determine communication frequency.

The test was executed five times. The results are tabulated below:

Counter Value	Elapsed Time [s]	Resulting Communication Frequency [Hz]
2367	118.651	19.95
5300	265.268	19.98
7428	371.71	19.98
8594	430.029	19.98
12274	618.386	19.85

Table 9.8: Results of the RS-232 communications speed test. It can be seen that the resulting communications speed is very close to 20Hz which correlates with the 50ms timer interrupts used in the communications scheme.

Communication Stability

The stability of the communication scheme was identified as a combination of the accuracy of the data that was transmitted as well as the robustness of the communication scheme against disturbances[4]. Firstly, the data accuracy of the communications scheme was tested by monitoring the amount of incorrect data that was received by the motor control boards in the arm. As a precaution, error checking functionality had been implemented upon the

[4]Disturbances in this case refers to any variation in TCP/IP communications speed from the PC to the master LM3S8962 board as this is a characteristic in utilising wireless TCP/IP links.

motor control boards in order to ensure that the motors did not move to an incorrect position, shown in Figure 9.18. This error checking code was used to increment a counter if any of the data that was received was faulty.

In each test the RS-232 communications were initiated and the arm links manipulated in order to produce any error-causing motor noise upon the communication lines. The error counter, together with the total number of transmissions, was then transmitted back to the client PC for monitoring as shown in Figure 9.19.

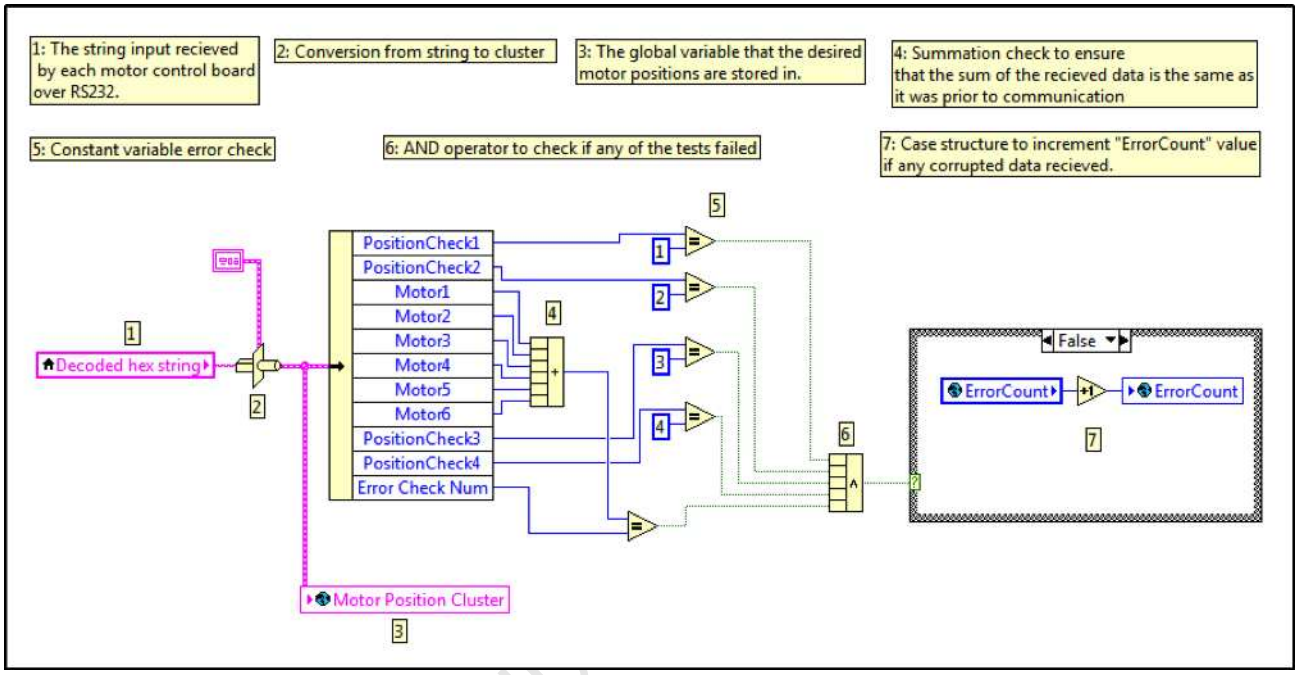

Figure 9.18: Error checking code used on the LM3S8962 motor control boards. The code checks the validity of four specific bytes in the packet that is received ("PositionCheck1" to "PositionCheck4") as well as a summation check of all the motor positions that are sent up the arm. If any of these are not what is expected the "ErrorCount" variable increments. This variable is then transmitted back to the PC.

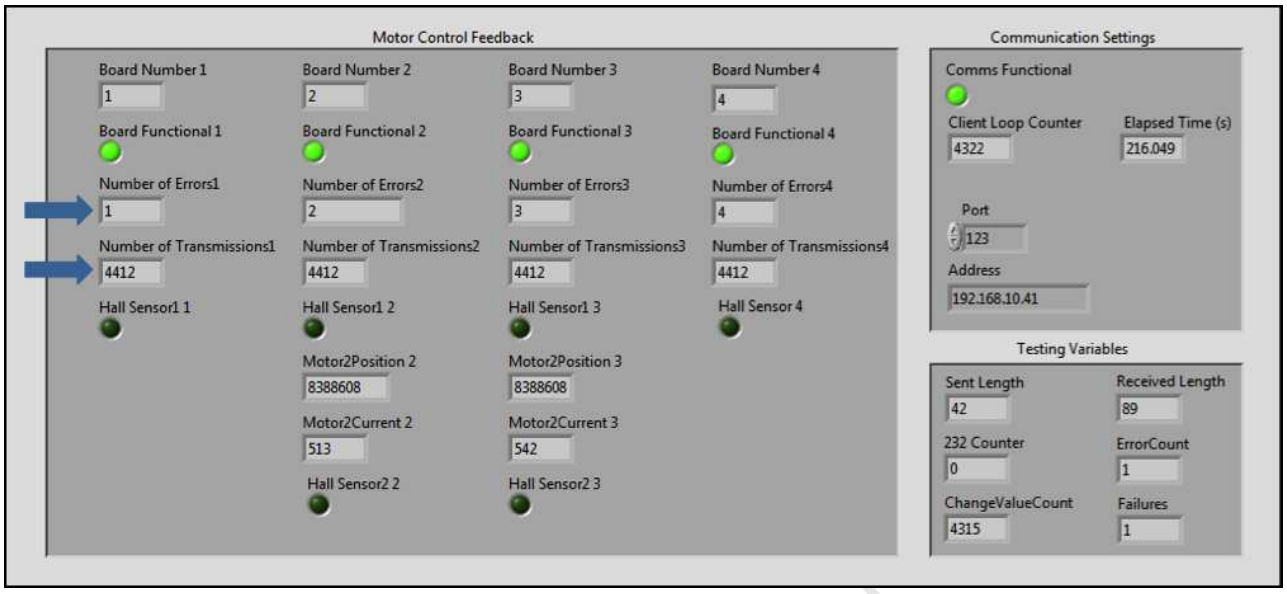

Figure 9.19: The Motor Control Feedback panel of the testing GUI. The fields in question (Number of Errors and Number of Transmissions) can be seen highlighted by the two arrows.

The "Number of Errors" fields above were calculated by determining the number of times that the "ErrorCount" variable communicated from the control boards changed. This was due to the fact that when powering the boards, as a result of using timer interrupts to execute the communications scheme , the boards would start trying to communicate. Without starting the client program on the PC, the slave boards would continuously transmit "false" data and the "ErrorCount" variable would continuously increase. The "Number of Errors" field on the front panel would therefore start off at an incorrect value. The change in value of this variable, and not the variable itself, was therefore monitored in order to provide an accurate result of the validity of the data transmitted. The test was run five times and the results tabulated:

Elapsed Time [s]	Number of Transmissions	Number of Errors	Inaccuracy
Board 1:			
360.205	7371	1	0.01%
399.415	8061	2	0.02%
430.029	8695	2	0.02%
618.386	12452	2	0.02%
800.027	16093	1	0.006%
Board 2:			
360.205	7330	2	0.02%
399.415	8055	3	0.04%
430.029	8685	2	0.02%
618.386	12448	3	0.02%
800.027	16093	2	0.01%
Board 3:			
360.205	7289	4	0.05%
399.415	8049	4	0.05%
430.029	8675	3	0.03%
618.386	12445	4	0.03%
800.027	16093	3	0.02%
Board 4:			
360.205	7289	5	0.06%
399.415	8049	5	0.06%
430.029	8675	4	0.04%
618.386	12445	5	0.04%
800.027	16093	4	0.02%

Table 9.9: RS-232 data accuracy test results. In all instances, the number of errors occurred at the beginning of transmissions tests and did not increment at all during the tests themselves. This indicates that, once the communication is running, there are zero errors in the data transmissions. An interesting thing to note is the increasing value of errors proportional to the board number. This is due to the fact that the communication architecture is reliant on each of the boards, linked together, to transmit data up and down the arm. The information present on the fourth board therefore takes four communication cycles to get to the master board. The third board takes three communication cycles; the second board takes two cycles and so on. The number of communication iterations that the incorrect data is sent up the arm is therefore proportional to the number that the board is in the chain. Hence the ascending value of errors.

The robustness of the communication scheme was then tested. This was done by implementing a variable timing delay in the client program and varying this manually with a range of delay values. This was to simulate any delays that may occur as a result of the wireless TCP/IP signal. The manual delays ranged from a length of 50ms to 1s in order to simulate worst case conditions. The values returned from the slave boards, as above, were then monitored in order to determine whether the delays had any effect on the RS-232 communication scheme. After varying the timing delay for several minutes it became clear that any interference with the TCP/IP had no effect on the communications up the arm; no additional errors occurred since the master board re-sent the last received information in every 50ms interrupt. The only effect was therefore a delay in the new information the master board could transmit.

9.3.3 RS-485 Communication Protocol

The fourth generation of the LM3S8962 motor control boards incorporated on-board DC to DC voltage converters as well as RS-485 communication capabilities instead of RS-232, as included in the previous generations. The speed

and stability of the RS-485 communications scheme was tested is now discussed.

RS-485, as discussed previously in Section 6.5, is daisy-chainable like I^2C. The resulting simplified model of the communications scheme inside the robotic arm is as follows:

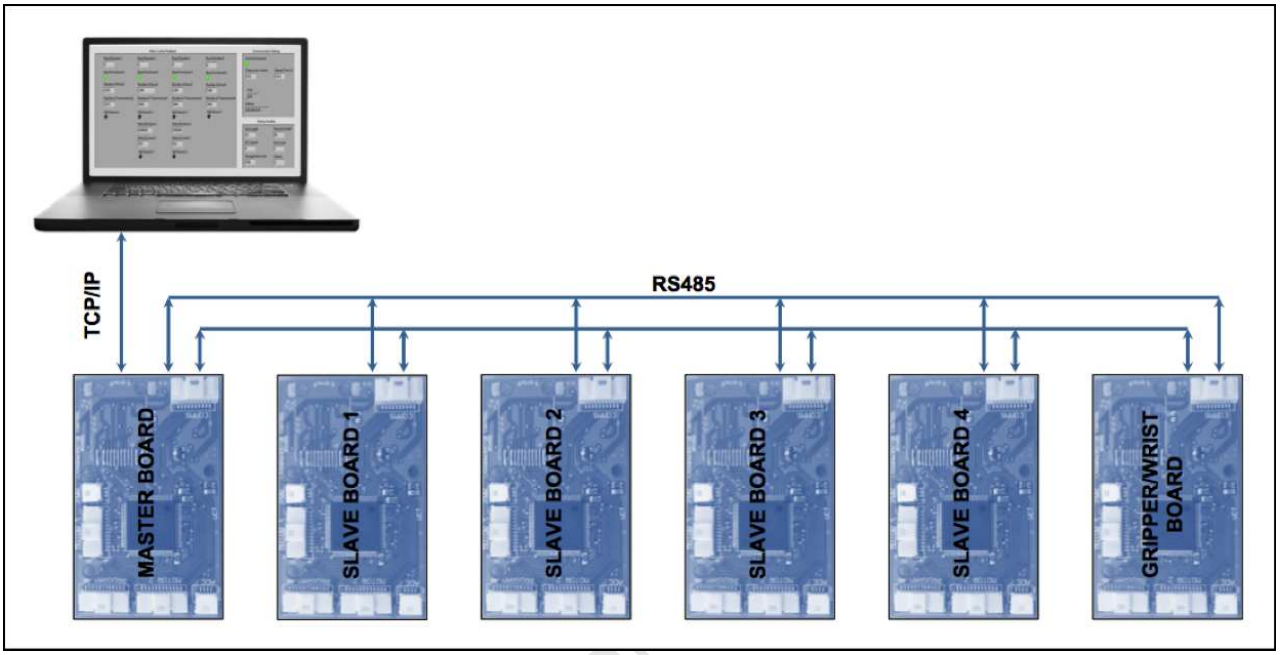

Figure 9.20: RS-485 communications architecture.

Communications Speed

The communications speed, as in the previous sections, was identified as the frequency at which a packet of data is transmitted to and from a slave board. The master board performed the TCP/IP and RS-485 communications in parallel while loops. Therefore, an accurate indication of the RS-485 communications speed was to increment a counter on each iteration of the RS-485 communications loop. Each iteration represented a bi-directional packet transfer to a single slave board. Thus, dividing the total number of iterations by five (as there are five slave boards) and dividing this by the total elapsed time would result in the overall communications speed. The testing interface is shown below with the relevant fields "Number of Transmissions" and "Elapsed Time" shown on the right.

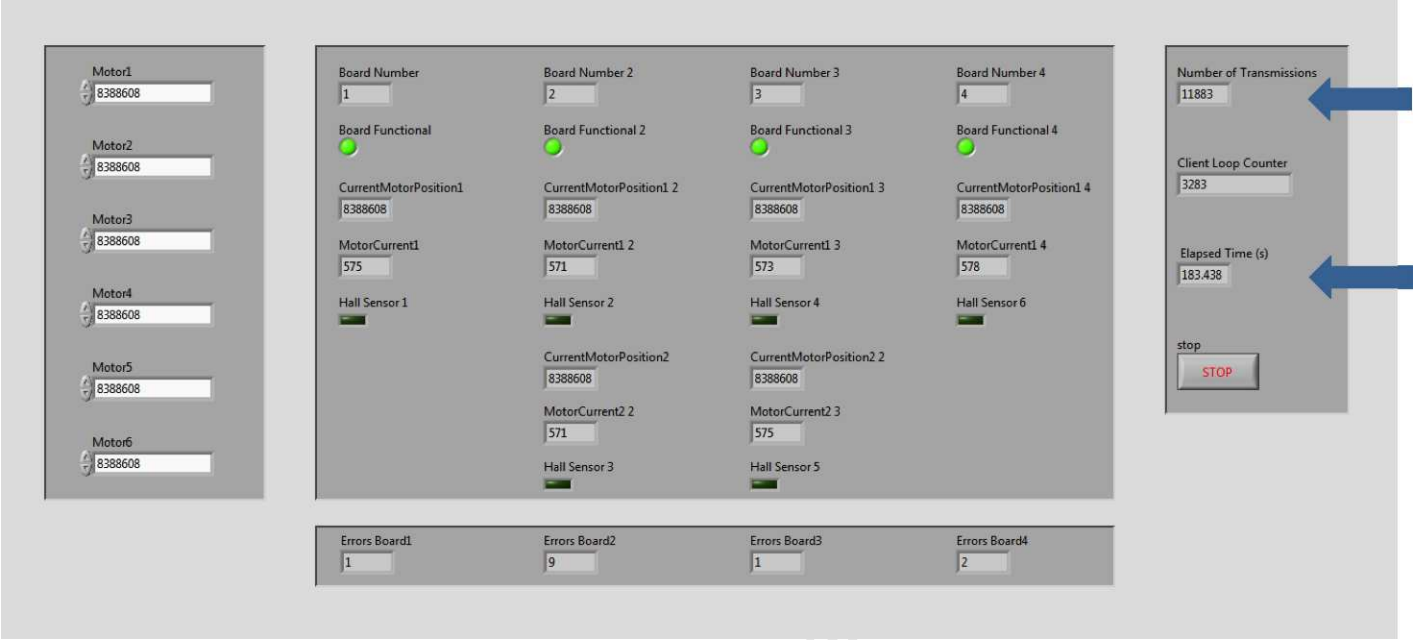

Figure 9.21: RS-485 Testing front panel.

The program was run five times and the resulting communications speed tabulated:

Elapsed Time [s]	Number of Transmissions	Board Transmissions	Communications Speed [Hz]
208.049	13693	2738	13.16
156.374	10170	2034	13.01
328.031	21078	4215	12.85
334.295	21556	4311	12.90
480.395	31330	6266	13.04

Table 9.10: RS-485 communication speed test results.

It can be seen that the resulting communication speed is, on average, 13Hz.

Communication Stability

The stability of the RS-485 communications scheme was tested in three ways. Firstly, much like the error checking functionality implemented upon the slave boards in the RS-232 communication scheme, a similar checksum was implemented upon the RS-485 slave boards. A counter was then incremented whenever faulty data was transmitted to each slave board. The testing front panel was executed and the arm motors moved sporadically to create any error causing motor noise. The test was run three times and the error counter data was transmitted back to the client for monitoring.

Elapsed Time [s]	Number of Transmissions	Number of Errors	Inaccuracy
Board 1:			
518.271	6772	0	0%
149.651	1949	0	0%
239.456	3124	0	0%
Board 2:			
518.271	6772	0	0%
149.651	1949	0	0%
239.456	3124	0	0%
Board 3:			
518.271	6772	0	0%
149.651	1949	0	0%
239.456	3123	0	0%
Board 4:			
518.271	6772	0	0%
149.651	1949	0	0%
239.456	3123	0	0%

Table 9.11: RS-485 data accuracy test results.

It can be seen that no false data was transmitted to any of the slave boards.

Another factor affecting the stability of the RS-485 communications scheme was the effect that the timing of the transmissions from the master board had on the ability of the slave boards to receive and analyse the sent packets. It became clear that if the data packets were transmitted to the slave boards at too swift an interval, the slave board, due to the complications in triggering the interrupt upon each RS-485 transmission, would sporadically fail to recognise the address bytes and thus ignore transmissions aimed for that particular board. The speed of communication (tested above) was thus based on a compromise between the overall speed of transmission and the frequency of transmission errors occurring. The sequence of communication from the master board was to transmit to a slave and then wait for the slave to reply with motor control data. If the slave did not reply within a certain window it was deduced that an error had occurred and the following slave board was addressed. The frequency of these errors was monitored and tabulated below:

Elapsed Time [s]	Number of Transmissions	Failed Transmissions	Error
Board 1:			
1482.93	18306	18	0.1%
126.164	1620	1	0.06%
377.04	4834	8	0.17%
Board 2:			
1482.93	18306	18	0.1%
126.164	1620	5	0.3%
377.04	4834	1	0.02%
Board 3:			
1482.93	18306	19	0.1%
126.164	1620	2	0.13%
377.04	4834	11	0.2%
Board 4			
1482.93	18306	24	0.13%
126.164	1620	1	0.06%
377.04	4834	2	0.04%

Table 9.12: RS-485 transmission failures.

It can be seen that, at the worst case, 0.3% of the transmissions sent to a slave board did not get detected. This equates to three transmissions out of every 1000 and thus this only has the slightest effect on the overall effective speed of transmission to a particular board.

The last factor affecting the stability of the communications scheme was any disturbances incurred by the TCP/IP platform. As done in the RS-232 testing, a variable timing delay was inserted into the client testing program and manually adjusted to monitor the effect on the RS-485 communications speed. In the worst case, with a delay of 1s implemented, the slowest RS-485 communications speed was determined to be approximately 8Hz. The communications resumed functionality at 13Hz when the delay was removed.

9.4 Collision Prevention

The collision prevention algorithms used in the system played a crucial role in ensuring that no damage was caused to the manipulator or to the other elements on the robot. Testing of the collision prevention algorithms took place on the table upon which the manipulator was fixed due to the disassembled state of the base of the RATEL USAR robot at the time of testing. Representations of the elements of the robot (the robot flippers and the laser scanner system) were manufactured from perspex and positioned on a one-to-one scale print of the base of the robot (see Figure 9.22) to ensure testing accuracy.

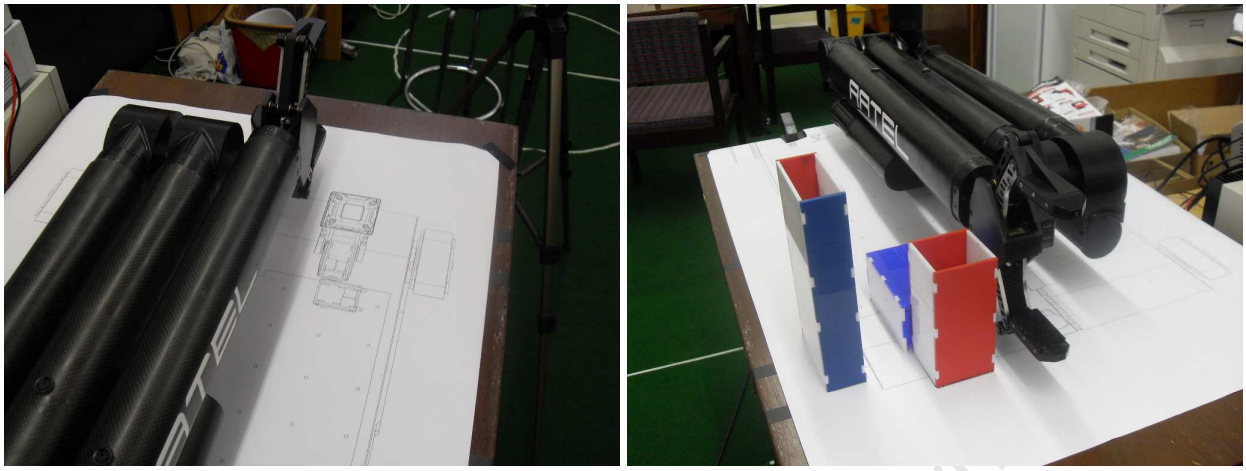

Figure 9.22: Collision testing setup. On the left is the scale print of the base of the robot. On the right is shown the perspex models of a flipper of the robot positioned vertically upwards for ease of testing together with the two elements of the laser scanner system (motors and laser scanner).

The heights of the perspex flipper and laser scanner models were altered to account for the mounting height difference (13mm) between the arm on the table and the arm on the actual robot. The manipulator was then moved towards the robot elements to verify the functionality of the collision prevention procedures. Figures showing the successful collision prevention of the manipulator with elements of the robot follow (Videos of each of these can be found on the DVD).

9.4.1 Manipulator with Flippers

The angle of the flippers (a value to be received from the control system for the base of the robot) was manually set to be 90 degrees and the manipulator was thereafter driven towards the flippers in a variety of poses to test the collision prevention of the different links of the arm. Figure 9.23 shows the collision prevention of the arm with the front left flipper. Video clips of the collision prevention tests can be found on the DVD provided.

Figure 9.23: Collision prevention of manipulator with front left flipper.

The same procedure on the front right flipper is shown in Figure 9.24

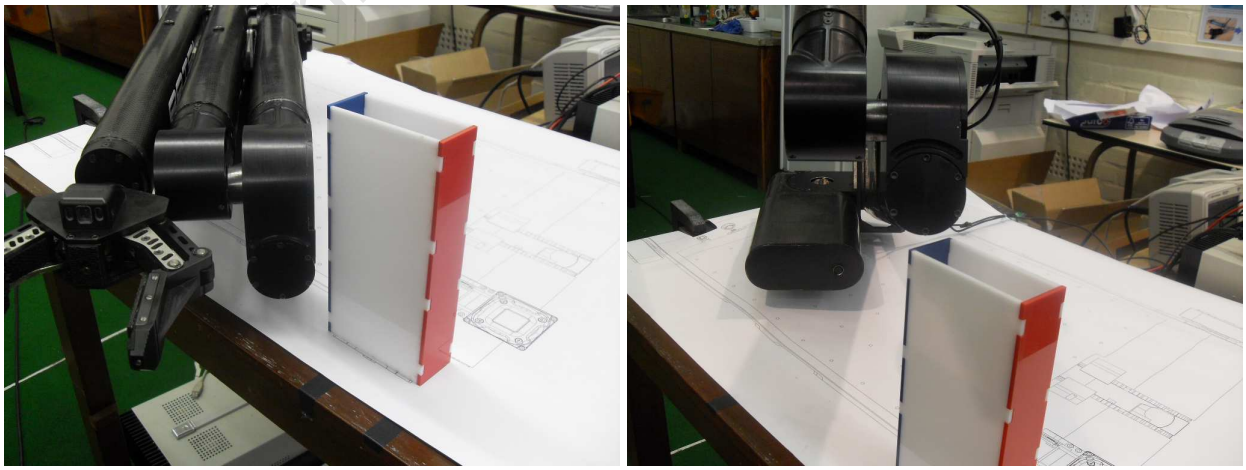

Figure 9.24: Collision prevention of manipulator with front right flipper.

The identical process was carried out on the rear flippers with the same successful results.

9.4.2 Gripper with Flippers

The angle of the gripper was manually set to 0 degrees and thereafter driven towards the flippers in a variety of poses to test the collision prevention of the gripper. Figure 9.25 shows the collision prevention of the gripper with the front right flipper.

Figure 9.25: Collision prevention of gripper with robot flippers.

9.4.3 Manipulator with Laser Scanner

The manipulator was driven towards the laser scanner in a variety of poses to test the collision prevention of the arm with the laser scanner. Figure 9.26 shows the successful collision prevention of the first link with the laser scanner from right, left and above.

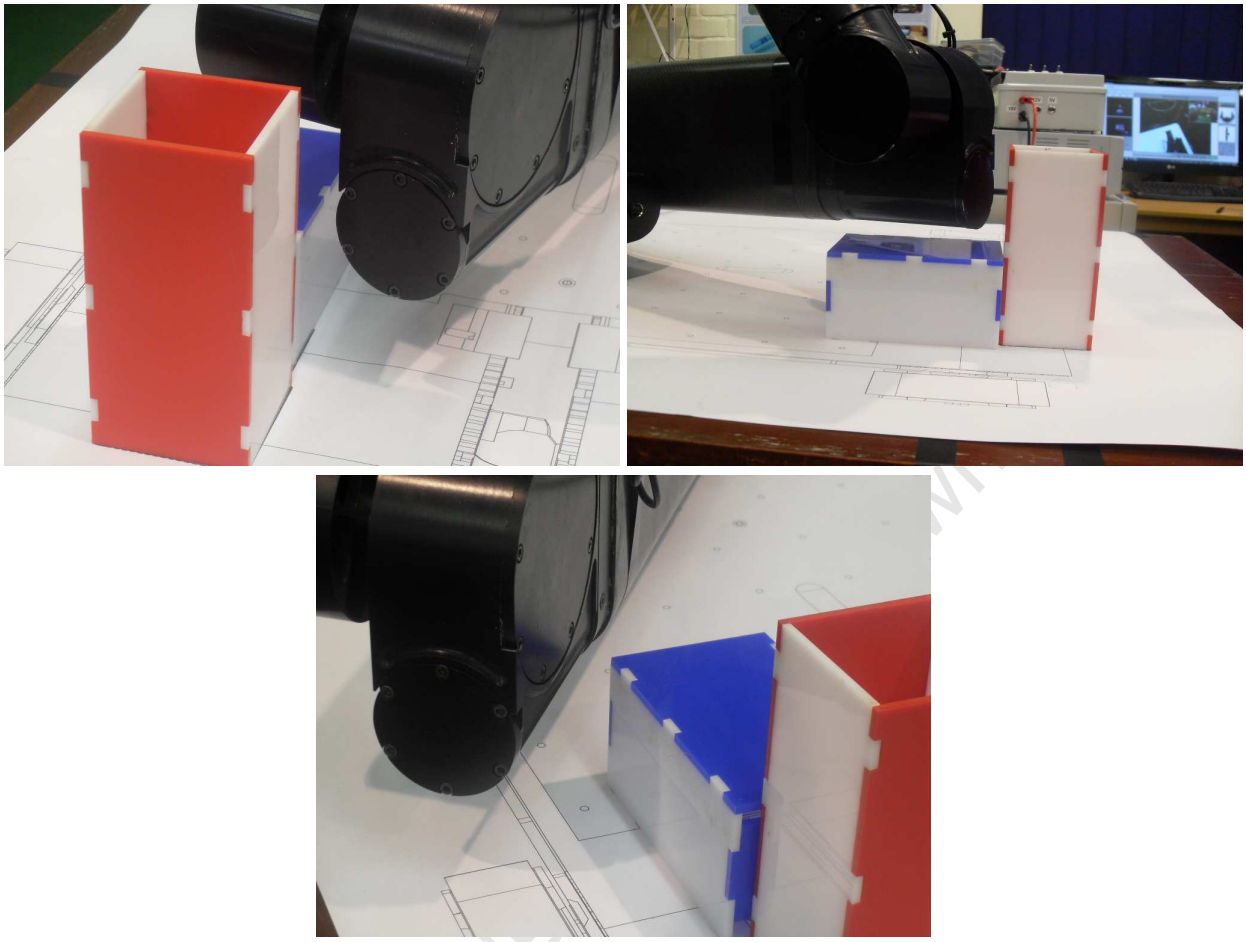

Figure 9.26: Collision prevention of manipulator with laser scanner

9.4.4 Gripper with Laser Scanner

The gripper was then driven towards the laser scanner. The collision prevention of the gripper, rotated at 270, 90 and 0 degrees, and the laser scanner can be seen in Figure 9.27.

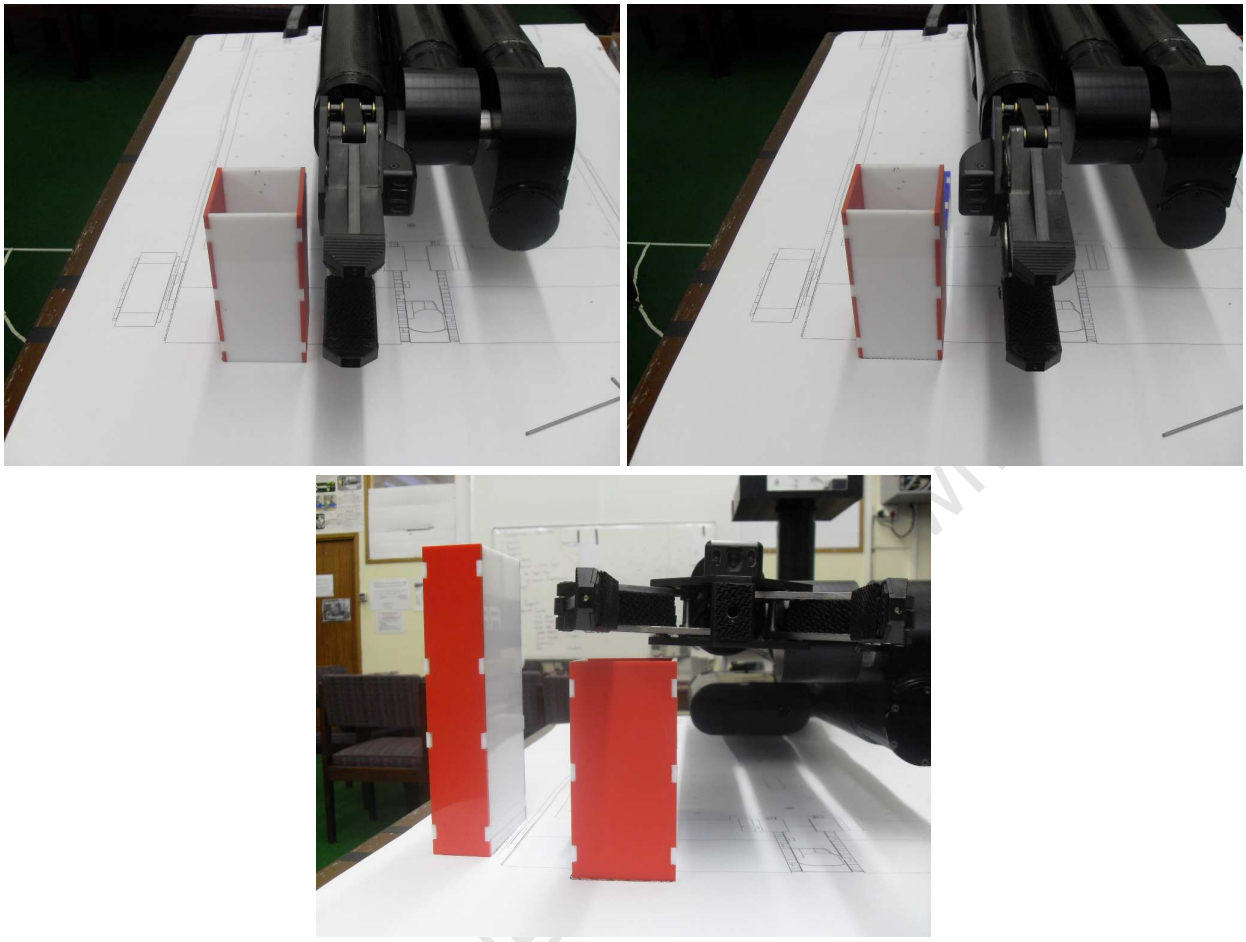

Figure 9.27: Collision prevention of gripper with laser scanner

9.4.5 Sensor Payload

The collision prevention of the sensor payload was then tested. The base of the disassembled sensor payload was used to test the collision prevention algorithms. The results can be seen in Figure 9.28 as well as on the DVD.

Figure 9.28: Collision prevention of the sensor payload.

9.5 Usability Testing

The ease of use of the system was a fundamental requirement of the project. This requirement was tested by allowing novice users to complete a task with the robotic arm and thereafter rate the ease of use of the complete system according to certain criteria in the form of a questionnaire (see Appendix B). The task executed by the novice users was a typical pick and place scenario as would be found in the RoboCup Rescue arena (see Figure 9.29).

Figure 9.29: Pick and place task performed by novice users.

The test candidates were seated with their backs to the robot arm to limit any direct sight of the task. The user was then required to maneuver the manipulator, using only the test GUI (see Figure 9.30), to grasp the water bottle in position A (the current position in Figure 9.29) and place it as close to the centre of the box in position B in as short a time as possible.

Figure 9.30: Testing GUI provided to the user including all necessary information regarding the arm and gripper systems. The main video feed was provided by a webcam fixed to the pan/tilt arm (see Figure 9.29) to mimic the video feed from the currently disassembled sensor payload.

The procedure in which the user executed the task was carried out as follows:

1. The user was briefed on the operation of the arm and gripper and shown how to operate the system.
2. The arm was placed in the active position (position 2) with all lock keys enabled and the tilt locked at the horizontal.
3. The user then completed the task described above. Upon placement of the bottle, the user moved the arm back to the active position at which time the task was completed.

The task was completed three times by each user. The first attempt served as a training run in which the user was heavily supervised. The second attempt required the user to complete the task on his/her own with supervision offered where necessary or when asked for. The third attempt was a timed, completely individual task with no assistance offered. Upon completion of the third attempt, the user's accuracy as well as time taken to complete task was recorded (Video recordings of each of the users attempts can be found on the attached DVD). The user was then required to complete the RATEL Manipulator Arm Usability Questionnaire to provide feedback on the ease of use of the system[5].

A total of nine candidates were selected to participate in the test. None of the candidates had ever operated the arm before, and three had zero prior knowledge of the manipulator. Table 9.13 shows the results of each of the user's attempts at completing the task.

User Number	Time [m:s]	Accuracy [mm]
1	2:22	21
2	3:23	16
3	2:59	10
4	4:47	8
5	2:23	18
6	2:54	12
7	3:09	46
8	3:31	11
9	3:10	9
Average:	3:11	17

Table 9.13: Pick and place task results.

The questionnaire that each user completed (copies of which can be found on the DVD) consisted of six criteria which the user was required to grade out of 5 (1 being inadequate, 5 being very effective). The results of the questionnaire can be seen in Table 9.14.

[5]The users completed the questionnaire anonymously in order to not influence their feedback

User Number	Representation of Arm	Collision Warnings	3D Mouse	Lock Key Functionality	Intuitiveness	Overall Ease Of Use
1	5	5	4	5	4	4
2	5	4	5	5	4	4
3	4	4	3	5	4	4
4	4	4	3	5	4	4
5	5	5	5	4	4	4
6	5	4	4	5	5	4
7	4	3	5	5	4	4
8	4	4	3	3	4	4
9	4	5	4	4	5	5
Average	4.44	4.25	4	4.55	4.22	4.11

Table 9.14: Pick and place task user feedback.

A comment that was present upon three of the questionnaires was that the rotation of the arm proved to be too sensitive when the arm became extended. Another comment suggested providing the user with an option to select the speed of the arm. A factor to note was that the bottle was only knocked over once in all user attempts to complete the task.

Having acquired all the necessary testing results, Chapter 10 discusses the conclusions that can be drawn about the system.

Chapter 10

Conclusions and Recommendations

10.1 Conclusions

10.1.1 Motor Control

The motor control used in the RATEL robotic manipulator, in combination with the inverse kinematic calculations, provided the stable and accurate arm motion required of the project. A novice operator was able to easily perform a pick and place task illustrating the effectiveness of the motor position control algorithms. The non-zero minimum motor speed allowed by the Maxon speed controllers did negatively impact the accuracy of the position control (creating only slight overshoot) however, for the application, an overall end effector accuracy of 3.5mm was deemed more than adequate.

The control of the pan/tilt system proved to be only partially successful. The weight of the sensor payload together with the pan/tilt gearing system meant that, even with PID position control, control of the sensor payload was lost after a certain tilt rotation point. Limiting this rotation point, however, did allow for successful sensor payload control. The second elbow of the arm also proved to be difficult to control as shown in Figure 5.28. However, by altering the deadband size depending on the motion of the link, accurate position control was achieved.

Calibration of the arm was successfully implemented through the use of Hall effect sensors. Including this functionality in the testing/calibration front panel allowed for quick and easy 'zero-ing' of the manipulator.

10.1.2 Communications

A TCP/IP communications scheme was carried by a Wi Fi (IEEE802.11a) network as required by the Robocup Rescue Specifications between the interface PC and master LM3S8962 board creating the required wireless communications standard upon which the RATEL robot was to be controlled.

The task of creating a stable, fast communications scheme between the motor control boards inside the arm proved to be a challenge. The electrically noisy environment inside the arm severely affected the initial attempts at creating an I^2C communications channel. The rescue RS-232 communications scheme proved to allow for the control of the manipulator but, although fast and stable, was unusable for the application due to its dependance on neighbouring motor control boards and the transmission delays up and down the arm. The final RS-485 communications scheme used on the re-designed LM3S8962 motor control boards provided the required communications stability in the arm. The results showed that the RS-485 communications scheme was slower than the RS-232 alternative, however, due to the 'leap-frog' way in which the RS-232 communications were executed, RS-485 provided a faster overall communications speed. The debugging benefits of the daisy-chain architecture used for RS-485 reaffirmed it as the superior protocol.

In developing the RS-485 communications scheme, the lack of a serial interrupt in the LabVIEW Embedded for Arm Microcontrollers functionality, and hence the manual processing of serial data, proved to greatly affect the communications speed as well as reliability. Precise timing delays needed to be inserted into the code in order to slow down the overall communications speed and hence provide the stability needed. Even so, the RS-485 communications scheme successfully provided the necessary communications in the arm to effectively control the robotic manipulator.

10.1.3 Collision Prevention

The Separating Axis Theory collision prevention algorithms successfully prevented the arm from colliding with other robot elements. Modelling the robot elements as simplistic polygons did not represent the bounds of the components completely accurately although this only proved to create a larger cushion area. The high speed SAT algorithms, together with the simplistic polygon models, provided fast collision detection processing.

10.1.4 Interface and Usability

The positive responses from the test candidates who completed the questionnaires after operating the RATEL robotic arm illustrate the ease of use of the system; an overall score of over 4/5. The inclusion of the lock key functionality proved to be of valuable assistance as shown by the highest average score. The comments given by the users indicate that the sensitivity of the rotation of the arm when extended and the inability to change the speed of the arm were both detrimental to the ease of use of the manipulator.

The ease of use of the system was, however, further established by the time taken to complete, as well as the accuracy of, the task asked of the novice operators. After only three attempts at controlling the arm, the users were able to accurately place the bottle in the desired location in only a matter of minutes. This is an effective indication of the success of the project.

10.2 Recommendations for Future Work

10.2.1 Modify Speed Controllers

As stated above, the minimum motor speed dictated by the Maxon speed controllers negatively affected the motor position control accuracy. Using speed controllers which could output zero motor speed would allow for more accurate position control as well as smoother arm motion. Dedicated position controllers are another option. However, due to the cost of position controllers in comparison to speed controllers together with the fact that, for the purposes of a teleoperated robotic arm, the position control of the manipulator does not need to be accurate to within a single encoder count, this option could be considered an 'overkill'.

10.2.2 Redesign of Sensor Payload

The sensor payload proved to be too heavy for even the replacement motors to control. A redesign of the sensor payload focusing on weight reduction would allow for better position control of the device. At the end of 2012 an undergraduate project focussing on the redesign of the sensor payload body was completed in the RARL lab. Graeme Wiley, equipped with a smaller, lighter thermal camera, designed a lighter sensor payload to surround the tilt arm (shown in Figure 10.1) thus reducing the moment acting upon the tilt motor [81]. Preliminary, informal tests showed that the tilt arm was able to hold the payload at any angle thus successfully resolving this concern.

Figure 10.1: 3D render of the RATEL robot equipped with the new sensor payload developed by Wiley [81].

10.2.3 Pan/tilt and Second Elbow Gearing Systems

Even with the redesigned sensor payload, incorporating self-locking gearing systems in the pan and tilt joints, similar to those used in the other links of the arm, would provide better position control for the sensor payload. The risk of losing control of the tilt arm would then be completely resolved. In a similar way, re-looking at the hypoid gearing system used by Henson [19] in the second elbow joint of the arm could provide better position control of the final link of the arm. Henson, in his thesis, provides a recommendation to lengthen the aluminium spacer between the worm and wheel to reduce backlash in the system. If this fails to provide a better response, opting for a duplex worm gearbox, as used in the lower three joints of the arm, would provide superior position control response.

10.2.4 Motor Speed

In all the joints of the arm, the speed settings on the motor speed controllers (set by DIP switches) were set to a minimum. In addition to this, the speed value sent to the motor controllers from the PD position control algorithms were capped to half of the maximum value. The reasonable operating speeds of the motors in the arm were therefore only a fraction of their possible speeds. Higher reduction motor gearboxes could therefore be used to lower the

output shaft speed and increase the motor position accuracy in the arm. Alternatively, lower power motors which produce a lower speed with acceptable torque could be investigated.

10.2.5 Include Speed Selection Option on the Graphical User Interface

Due to the comments from the users, altering the rotation speed of the arm, depending on the extension of the manipulator, would make the arm easier to use. This functionality coincides with allowing the user to select a motion speed for the arm. Creating an additional speed variable to send to the slave boards, dictating a speed of motion (fast, medium and slow), would further improve the ease of use of the system.

10.2.6 Use of C Code on the LM3S8962

Although potentially more difficult to implement, the use of C code on the LM3S8962 would allow for the availability of serial interrupts. This would improve the speed and stability of the RS-485 communications as there would be no need to implement custom delays in the code. LabVIEW Embedded for Arm Microcontrollers also makes use of the RTX kernel which places a significant overhead on the processing power of the LM3S8962. Using C code would remove this overhead and therefore speed up the overall processing speed of the motor control algorithms. It was discovered that coding C example code onto the LM3S8962 lowered the current usage of the boards. This provides an added benefit of using C code.

10.2.7 Current Monitoring Collision Detection

Current monitoring boards developed by Rieger [18] (see Figure 10.2 and Figure 10.3) were used to provide current draw information of the individual motors in the arm. By monitoring the current draw of each motor, collisions with external objects (not elements of the RATEL robot) can be detected. Analysing spikes in the current draw of each motor can give indications of when motors are experiencing abnormal amounts of resistance. This detection process, together with procedures to move the arm away from a detected collision, would provide superior safety for the arm.

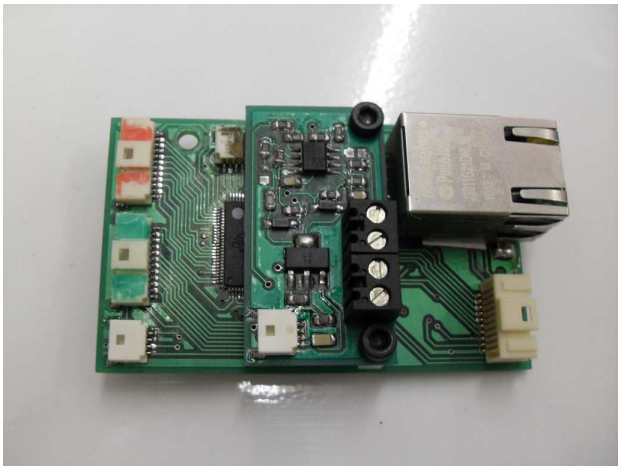

Figure 10.2: Rieger's motor current monitoring board mounted on the LM3S8962 motor control board [18].

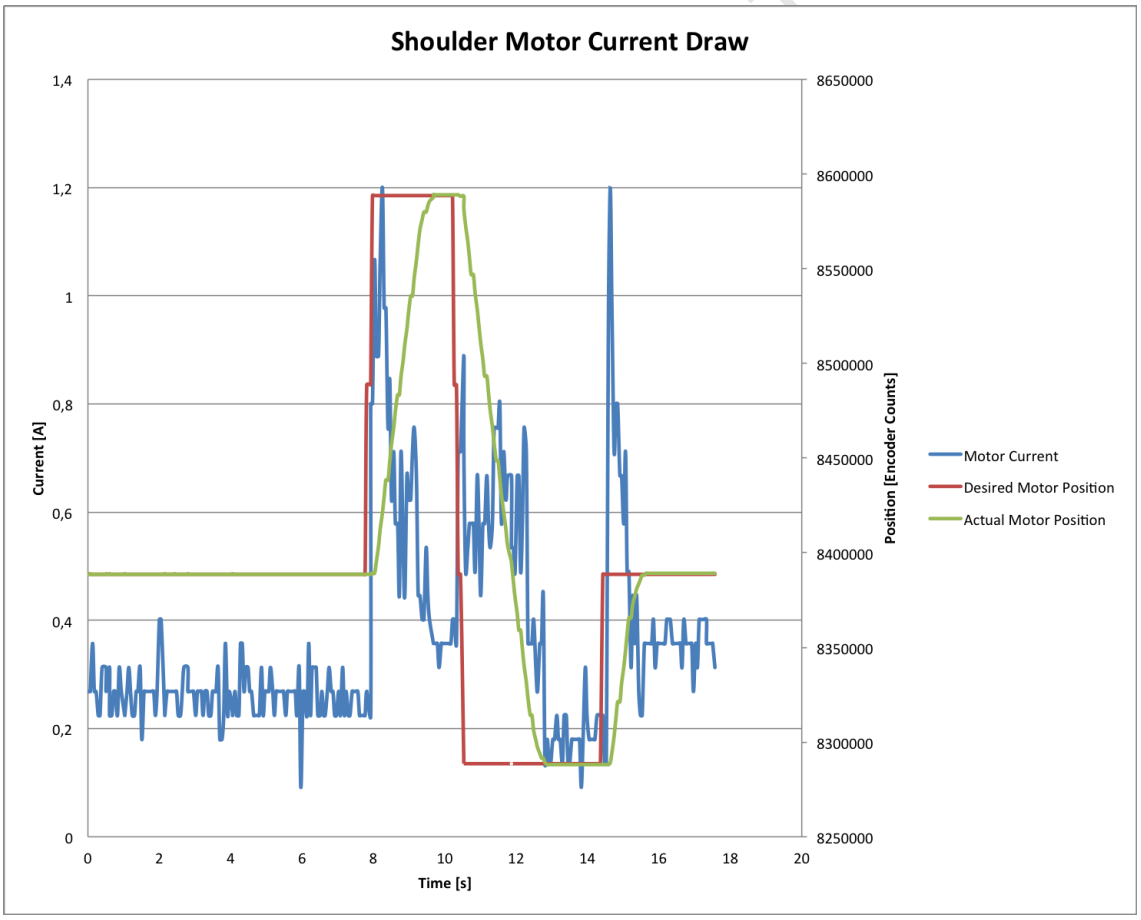

Figure 10.3: Plot of the shoulder motor current in comparison to the desired and actual motor positions.

10.2.8 Latching of FPGA communications data

Even though the process was discarded, latching the encoder count data at the start of a communications transfer between the FPGA and Coldfire would prevent possible overflow errors and incorrect data transmissions.

List of References

[1] R. Murphy, "Marsupial and shape-shifting robots for urban search and rescue." *IEEE Intelligent Systems*, vol. 15, no. 2, p. 14, 2000.

[2] C. Glover, B. Russell, A. White, M. Miller, and A. Stoytchev, "An effective and intuitive control interface for remote robot teleoperation with complete haptic feedback," in *Proceedings of the 2009 Emerging Technologies Conference (ETC), Ames, IA, USA*, 2009.

[3] J. Cui, S. Tosunoglu, R. Roberts, C. Moore, and D. Repperger, "A Review of Teleoperation System Control," in *Proceedings of the Florida Conference on Recent Advances in Robotics, FCRAR, Boca Raton, Florida, USA*, 2003.

[4] M. Baker, R. Casey, B. Keyes, and H. Yanco, "Improved interfaces for human-robot interaction in urban search and rescue," in *Systems, Man and Cybernetics, 2004 IEEE International Conference on*, vol. 3. IEEE, 2004, pp. 2960–2965.

[5] R. Murphy, "Human-Robot Interaction in Rescue Robotics," *IEEE Transactions on Systems, Man and Cybernetics, Part C (Applications and Reviews)*, vol. 34, no. 2, pp. 138–153, May 2004.

[6] P. Dvorak and M. Obe, "After Nuclear Milestone, a Long Road," Dec. 2011. [Online]. Available: http://online.wsj.com/article/SB10001424052970204336104577096281099680526.html

[7] Zimbio, "Quince, the Fukushima robot," 2012. [Online]. Available: http://www.zimbio.com/Japan+Earthquake+2011/articles/7fhOKBteLZZ/Quince+the+Fukushima+robot

[8] 3D Connexion, "3D Connexion SpaceNavigator," 2012. [Online]. Available: http://www.3dconnexion.com/products/spacenavigator.html

[9] A. Jacoff, E. Messina, B. Weiss, S. Tadokoro, and Y. Nakagawa, "Test arenas and performance metrics for urban search and rescue robots," in *Intelligent Robots and Systems, 2003.(IROS 2003). Proceedings. 2003 IEEE/RSJ International Conference on*, vol. 4. IEEE, 2003, pp. 3396–3403.

[10] The Robocup Federation, "Robocup," 2011. [Online]. Available: www.robocup.org

[11] C. Schlenoff and E. Messina, "A robot ontology for urban search and rescue," *Proceedings of the 2005 ACM workshop on Research in knowledge representation for autonomous systems - KRAS '05*, pp. 27–34, 2005.

[12] E. Messina, A. Jacoff, J. Scholtz, C. Schlenoff, H. Huang, A. Lytle, and J. Blitch, "Statement of requirements for urban search and rescue robot performance standards," *NIST Draft Report*, pp. 1–91, 2005.

[13] E. Dreyer, "Development of an Unmanned Ground Vehicle Platform," University of Cape Town, Tech. Rep., 2012.

[14] D. Lwabona, "The Completion of a Urban Search and Rescue Robot Platform," University of Cape Town, Tech. Rep., 2013.

[15] C. Sharpe, "The Design and Development of a Sensor Payload for an Urban Search and Rescue Robot," University of Cape Town, Tech. Rep., 2013.

[16] R. Whittemore, "Machine Vision for Urban Search and Rescue/Mining Robots," University of Cape Town, Tech. Rep., 2013.

[17] J. Kent, "SLAM for Urban Search and Rescue." University of Cape Town, Tech. Rep., 2013.

[18] M. Rieger, "Development of a Rescue Robot End-Effector," University of Cape Town, Tech. Rep., 2013.

[19] P. Henson, "Development of a 6 Degree-of-Freedom Manipulator Arm for Use on an Urban Search and Rescue Robot," University of Cape Town, Tech. Rep., 2012.

[20] RoboCup Technical Committee, "RoboCup Rescue Rules 2011," 2011. [Online]. Available: http://www.nist.gov/el/isd/upload/Robocup_Rules_2011.pdf

[21] R. Sheh and H. Komsuoglu, "MANIPULATION for SSRR," in *The IEEE-RAS Safety Security and Rescue Robotics Summer School 2012*, no. september, 2012.

[22] K. Lonji, "Mobile Robot Teleoperation Using Enhanced Video," Ph.D. dissertation, McGill University, 1996.

[23] P. Hokayem and M. Spong, "Bilateral teleoperation: An historical survey," *Automatica*, vol. 42, no. 12, pp. 2035–2057, Dec. 2006.

[24] A. Tunwannarux and S. Tunwannarux, "Design of a 5-Joint Mechanical Arm with User-Friendly Control Program," in *Proceedings of World Academy of Science, Engineering and Technology*, vol. 21, 2007, pp. 43–48.

[25] A. Milstein, M. Mcgill, C. Sammut, R. Salleh, R. Farid, J. V. Miro, G. Dissanayake, and M. Norouzi, "RoboCupRescue 2010 - Robot League Team Team CASualty (Australia)," Tech. Rep., 2010.

[26] M. W. Kadous, R. K.-M. Sheh, and C. Sammut, "Effective user interface design for rescue robotics," *Proceeding of the 1st ACM SIGCHI/SIGART conference on Human-robot interaction - HRI '06*, p. 250, 2006.

[27] N. Enayati, G. Mahaseni, A. Tamjidi, M. Fallahinejad, A. Mobarhani, S. Malektjati, and F. Najafi, "RoboCupRescue 2010 - Robot League Team PARS (Iran)," Tech. Rep., 2010.

[28] J. Drury, J. Scholtz, and H. Yanco, "Awareness in human-robot interactions," in *Systems, Man and Cybernetics, 2003. IEEE International Conference on*, vol. 1, no. October. IEEE, 2003, pp. 912–918.

[29] H. Filippi, "Wireless teleoperation of robotic arms," Lulea University of Technology, Tech. Rep., 2009.

[30] D. King, "SPACE SERVICING: PAST, PRESENT AND FUTURE," in *Proceedings of the 6 th International Symposium on Artificial Intelligence and Robotics & Automation in Space: i-SAIRAS 2001, Canadian Space Agency, St-Hubert, Quebec, Canada*, no. 6, 2001, pp. 1–8.

[31] Japan Aerospace Exploration Agency, "Shuttle Robot Arm," 2000. [Online]. Available: http://iss.jaxa.jp/iss/3a/orb_rms_e.html

[32] K. D. Edwards, "Design, Construction and Testing of a Wheelchair-Mounted Robotic Arm," Ph.D. dissertation, University of South Florida, 2005.

[33] D. Ryu, J.-B. Song, C. Cho, S. Kang, and M. Kim, "Development of a six DOF haptic master for teleoperation of a mobile manipulator," *Mechatronics*, vol. 20, no. 2, pp. 181–191, Mar. 2010.

[34] G. Sukhatme and G. Kim, "Haptic control of a mobile robot: a user study," in *IEEE/RSJ International Conference on Intelligent Robots and System*, no. 2. Ieee, 2002, pp. 2867–2874.

[35] C. Wagner, N. Stylopoulos, and R. Howe, "The role of force feedback in surgery: analysis of blunt dissection," in *Proceedings 10th Symposium on Haptic Interfaces for Virtual Environment and Teleoperator Systems. HAPTICS 2002.* IEEE Comput. Soc, 2002, pp. 68–74.

[36] SensAble Technologies Inc., "Haptic Devices," 2011. [Online]. Available: http://www.sensable.com/products-haptic-devices.htm

[37] T. Massie, "Initial haptic explorations with the phantom: Virtual touch through point interaction," Ph.D. dissertation, 1996.

[38] S. Wall, "A high bandwidth interface for haptic human computer interaction," *Mechatronics*, vol. 11, pp. 371–387, 2001.

[39] J. R. Rogers, "Low-cost teleoperable robotic arm," *Mechatronics*, vol. 19, no. 5, pp. 774–779, Aug. 2009.

[40] D. Kostic, B. de Jager, M. Steinbuch, and R. Hensen, "Modeling and identification for high-performance robot control: An RRR-robotic arm case study," *Control Systems Technology, IEEE Transactions on*, vol. 12, no. 6, pp. 904–919, 2004.

[41] D. Bertram, J. Kuffner, R. Dillmann, and T. Asfour, "An integrated approach to inverse kinematics and path planning for redundant manipulators," in *Robotics and Automation, 2006. ICRA 2006. Proceedings 2006 IEEE International Conference on.* IEEE, 2006, pp. 1874–1879.

[42] R. Edlinger, A. Pölzleithner, and M. Zauner, "Mechanical Design and System Architecture of a Tracked Vehicle Robot for Urban Search and Rescue Operations," *rrt.fh-wels.at*, pp. 1–12.

[43] P. Chang and H. Park, "Development of a robotic arm for handicapped people: a task-oriented design approach," *Autonomous robots*, vol. 15, no. 1, pp. 81–92, 2003.

[44] G. Tevatia and S. Schaal, "Inverse kinematics for humanoid robots," in *Robotics and Automation, 2000. Proceedings. ICRA'00. IEEE International Conference on*, vol. 1, no. Icra. IEEE, 2000, pp. 294–299.

[45] Z. Mao and T. C. Hsia, "Obstacle avoidance inverse kinematics solution of redundant robots by neural networks," *Robotica*, vol. 15, no. 1, pp. 3–10, Jan. 1997.

[46] B. Siciliano and O. Khatib, *Springer Handbook of Robotics.* Springer, 2008.

[47] F. Pierrot, E. Dombre, E. Dégoulange, L. Urbain, P. Caron, S. Boudet, J. Gariépy, and J. Mégnien, "Hippocrate: A safe robot arm for medical applications with force feedback," *Medical Image Analysis*, vol. 3, no. 3, pp. 285–300, 1999.

[48] M. C. Lin, C. Hill, J. D. Cohen, and M. Ponamgi, "I-COLLIDE: An interactive and exact collision detection system for large-scale environments," in *Proceedings of the 1995 symposium on Interactive 3D graphics*, 1995, pp. 189–197.

[49] C. A. Shaffer and G. M. Herb, "A real-time robot arm collision avoidance system," *IEEE Transactions on Robotics and Automation*, vol. 8, no. 2, pp. 149–160, 1992.

[50] E. Cheung and V. Lumelsky, "Motion Planning for a Whole-Sensitive Robot Arm Manipulator," in *IEEE International Conference on Robotics and Automation*, 1990, pp. 344–349.

[51] A. Gregory, M. C. Lin, S. Gottschalk, and R. Taylor, "Fast and Accurate Collision Detection for Haptic Interaction Using a Three Degree-of-Freedom Force-Feedback Device," *Computational Geometry*, vol. 15, no. 1, pp. 69–89, 2000.

[52] S. Gottschalk, M. C. Lin, D. Manocha, and C. Hill, "OBBTree: A Hierarchical Structure for Rapid Interference Detection," in *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, 1996, pp. 171–180.

[53] G. V. D. Bergen, "Proximity Queries and Penetration Depth Computation on 3D Game Objects," in *Game developers conference*, 2001.

[54] T. Akenine-Möller, "Fast 3D Triangle-Box Overlap Testing," *Journal of Graphics Tools*, vol. 6, no. 1, pp. 29–33, 2001.

[55] S. Redon, A. Kheddar, and S. Coquillart, "Fast Continuous Collision Detection between Rigid Bodies," in *Computer graphics forum*, vol. 21, no. 3, 2003.

[56] K. Uschin, S. Inteam, T. Benjawilaikul, and N. Sae-eaw, "RoboCup Rescue 2010 - Robot League Team iRAP _ PRO (Thailand)," Tech. Rep., 2010.

[57] ESA CSIP, "7 Function Hydraulic Manipulator Arm," 2009. [Online]. Available: http://www.csip.co.uk/

[58] M. Shimizu, "RoboCupRescue 2010 - Robot League Team C-Rescue (Japan)," Tech. Rep., 2010.

[59] H. Wang, X. Wang, R. Burman, W. Han, and W. Xian, "RoboCupRescue 2010 - Robot League Team eeeBot (Singapore)," Tech. Rep., 2010.

[60] M. Ravandi, S. Sadeghnejad, A. Sheikhjafari, and M. Nabi, "RoboCupRescue 2010 - Robot League Team Pasargad (Iran)," Tech. Rep., 2010.

[61] M. Hillman, K. Hagan, and S. Hagan, "The Weston wheelchair mounted assistive robot-the design story," *Robotica*, vol. 20, no. 02, pp. 125–132, Apr. 2002.

[62] I. Kelly, O. Holland, and C. Melhuish, "Slugbot: A robotic predator in the natural world," in *Proceedings of the 5th International Symposium on Artificial Life and Robotics*, 2000.

[63] M. Lauria, Y. Piguet, and R. Siegwart, "Octopus-an autonomous wheeled climbing robot," in *Proceedings of the Fifth International Conference on Climbing and Walking Robots*, 2002.

[64] K. Tindell, A. Burns, and A. Wellings, "Calculating controller area network (CAN) message response times," *Control Engineering Practice*, vol. 3, no. 8, pp. 1163–1169, 1995.

[65] J. Wang, Y. Tan, Y. Xu, X. Xu, J. Rong, and C. Jiang, "RoboCupRescue 2010 - Robot League Team SEU-RedSun (China)," Tech. Rep., 2010.

[66] T. Asfour, K. Berns, and R. Dillmann, "The humanoid robot ARMAR: Design and control," in *The 1st IEEE-RAS International Conference on Humanoid Robots (HUMANOIDS 2000)*. Citeseer, 2000, pp. 7–8.

[67] M. Cavusoglu, D. Feygin, and F. Tendick, "A critical study of the mechanical and electrical properties of the phantom haptic interface and improvements for highperformance control," *Presence: Teleoperators & Virtual Environments*, vol. 11, no. 6, pp. 555–568, 2002.

[68] C. Glover, B. Russell, A. White, and M. Miller, "An effective and intuitive control interface for remote robot teleoperation with complete haptic feedback," *Proceedings of the*, 2009. [Online]. Available: http://www.ece.iastate.edu/ alexs/papers/ETC_2009/ETC_2009.pdf

[69] M. Williamson, "Robot arm control exploiting natural dynamics," Ph.D. dissertation, 1999.

[70] G. Marani, S. K. Choi, and J. Yuh, "Underwater autonomous manipulation for intervention missions AUVs," *Ocean Engineering*, vol. 36, no. 1, pp. 15–23, Jan. 2009.

[71] P. Arcara, "Control schemes for teleoperation with time delay: A comparative study," *Robotics and Autonomous Systems*, vol. 38, pp. 49–64, 2002.

[72] M. Zhihong and M. Palaniswami, "A robust tracking control scheme for rigid robotic manipulators with uncertain dynamics," *Computers & electrical engineering*, vol. 21, no. 3, pp. 211–220, 1995.

[73] C. Y. Su and T. P. Leung, "A Sliding Mode Controller with Bound Estimation for Robot Manipulators," *IEEE Transactions on Robotics*, vol. 9, no. 2, pp. 208–214, 1993.

[74] Y. Guo and P. Y. Woo, "An Adaptive Fuzzy Sliding Mode Controller for Robotic Manipulators," *IEEE TRANSACTIONS ON SYSTEMS, MAN, AND CYBERNETICS*, vol. 33, no. 2, pp. 149–159, 2003.

[75] Freescale, "MCF51JM128 ColdFire Â® Integrated Microcontroller Reference Manual," Tech. Rep., 2009.

[76] T. Asfour, K. Berns, and J. Schelling, "Programming of manipulation tasks of the humanoid robot ARMAR," in *The 9th International Conference on Advanced Robotics*, no. October, 1999, pp. 107–112.

[77] R. Whittemore, "Byte Stuffing Encoding Algorithm," 2012.

[78] Texas Instruments, "RS-422 and RS-485 Standards Overview and System Configurations," Tech. Rep., 2010.

[79] N. Clark, "3D mouse support using classes and events," 2009. [Online]. Available: http://www.mathworks.com/matlabcentral/fileexchange/22124-3d-mouse-support-using-classes-and-events

[80] G. Heimbach, "Text to Speech using LabVIEW and Microsoft Speech SDK," 2008. [Online]. Available: https://decibel.ni.com/content/docs/DOC-2263

[81] G. Wiley, "Development of a New Sensor Payload for Rescue Robot," University of Cape Town, Tech. Rep., 2012.

Appendix A

Wiring Diagrams

A.1 Manipulator Wiring Diagram

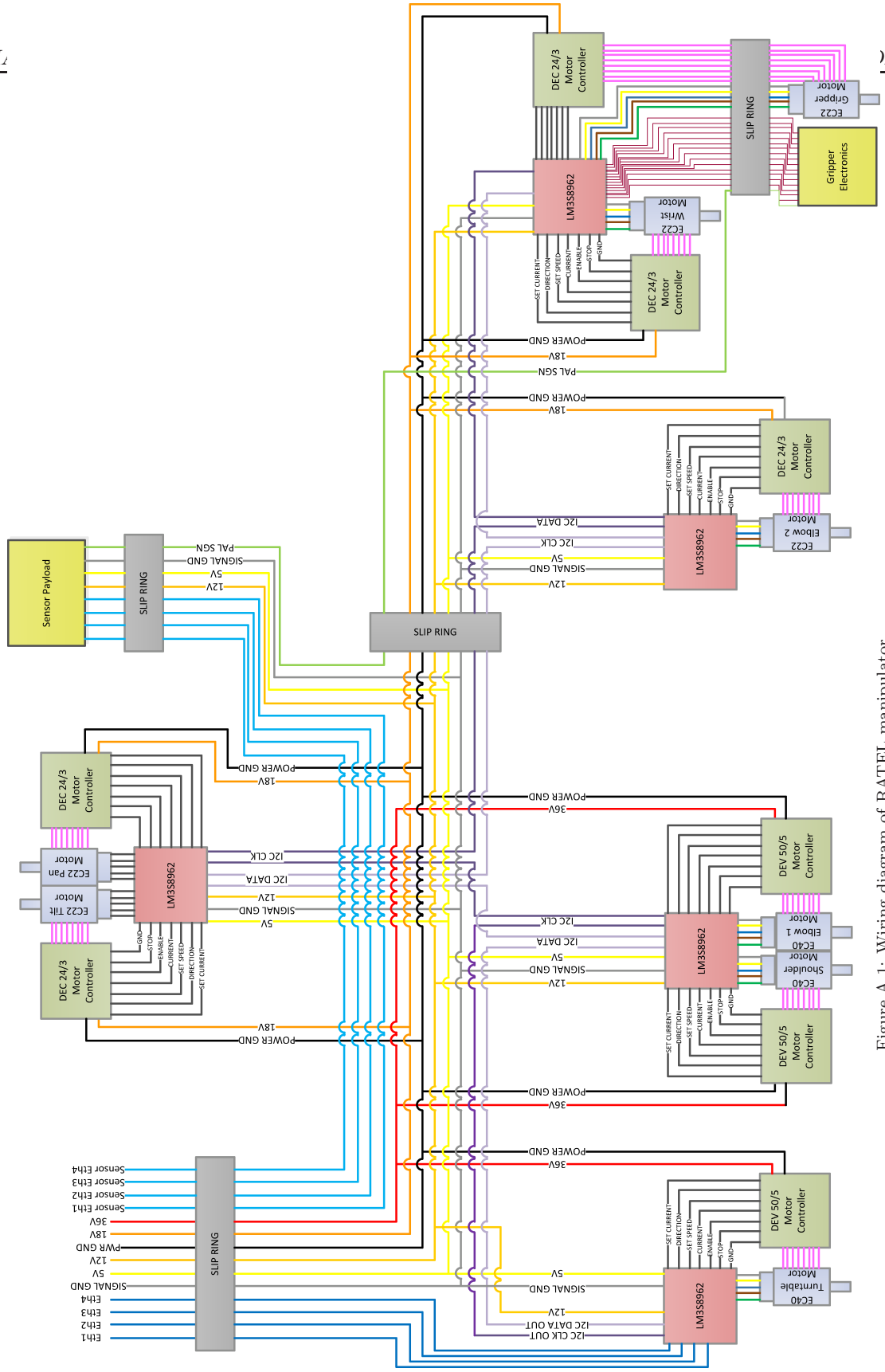

Figure A.1: Wiring diagram of RATEL manipulator.

A.2 Intermediate Connection Board/Slip Ring Wiring Breakdown

Connection Board #		Slip Ring #	Number of wires	Wire Colour		
1	TX	1	1	White		
2	RX	2	1	Grey		
3	TX	3	1	White		
4	RX	4	1	Grey		
5	ETH1	5	1	Blue		
6	ETH2	6	1	Blue/White		
7	ETH3	7	1	Green		
8	ETH4	8	1	Green/White		
9	5V	9--10	2	Pink		
10	12V	11--13	3	Red		
11	GND (36V, 18V)	14--17	4	Black		
12	GND (Signal)	18--22	5	Brown		
13	18V	23--26	4	Yellow		
14	36V	27--30	4	Orange		

Table A.1: Intermediate Connection Board/Slip Ring Wiring Breakdown

Appendix B

User Questionnaire

RATEL Manipulator Arm Useability Questionnaire

User Number:

Time to complete task:

Distance from target:

GUI:

Please rate the user interface according to the following categories. 1 being inadequate, 5 being very effective.

	1	2	3	4	5
Representation of Arm:	☐	☐	☐	☐	☐
Collision Warnings:	☐	☐	☐	☐	☐

Additional Comments:

Control:

Please rate the control of the arm according to the following categories. 1 being inadequate, 5 being very effective.

	1	2	3	4	5
3D Mouse:	☐	☐	☐	☐	☐
Lock Key Functionality:	☐	☐	☐	☐	☐
Intuitiveness:	☐	☐	☐	☐	☐
Overall Ease of Use:	☐	☐	☐	☐	☐

Additional Comments:

Figure B.1: User questionnaire completed by novice operators after using the arm.

Appendix C

Assessment of Ethics in Research Projects Form