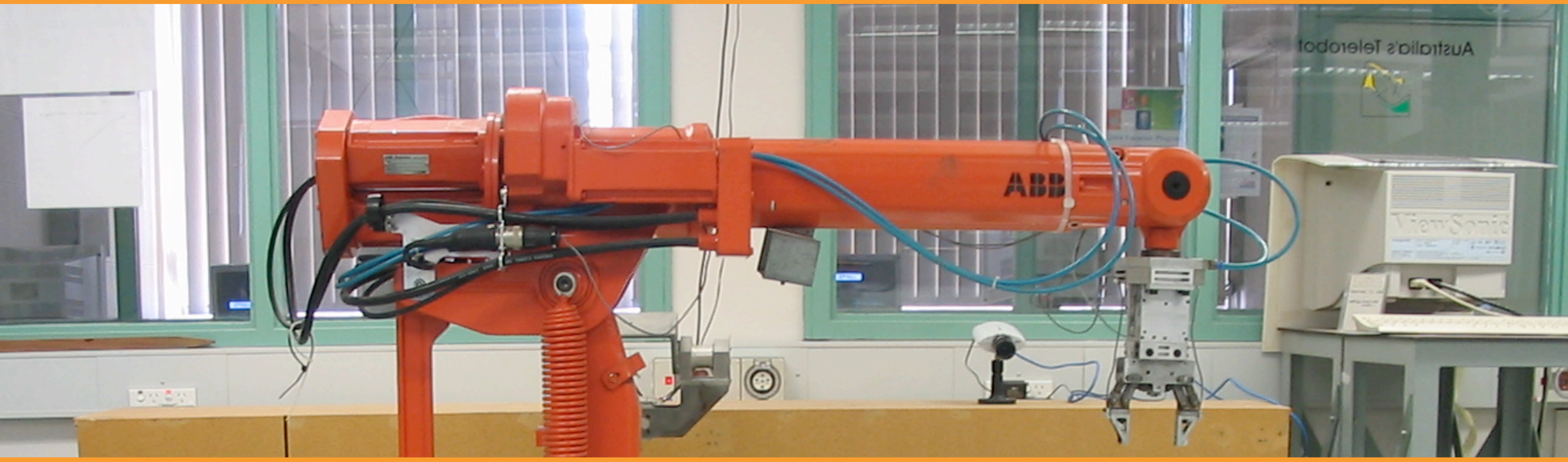
The University of Western Australia

Faculty of Engineering, Computing and Mathematics



Using telerobotics for remote kinematics laboratories

Final Year Project, 2004

Samuel Rae

Supervisors: A/Prof. James Trevelyan
A/Prof. Karol Miller (Interim)

Samuel Rae
PO Box 609
DONNYBROOK WA 6239

October 31st, 2004

The Dean
Faculty of Engineering, Computing and Mathematics
The University of Western Australia
35 Stirling Highway
CRAWLEY WA 6009

Dear Sir,

Please accept this thesis entitled **'Using telerobotics for remote kinematics laboratories'** as part of the requirement for the degree of Bachelor of Engineering (Mechatronics) with Honours.

Yours sincerely,

Samuel Rae

But because of his great love for us,
God, who is rich in mercy, made us
alive with Christ even when we
were dead in transgressions—
it is by grace you have
been saved.

Ephesians 2:4–5

Abstract

For the first time, the University of Western Australia Telerobot can be used for remote laboratories. These laboratories will give flexibility to the teaching curriculum in the the School of Mechanical Engineering, and will allow students to learn in their own time and at their own pace. Several interesting problems were solved to achieve control of the robot's joints. Additionally, the forward kinematics was solved, leading to a simulation of the robot. This thesis shows the techniques used and the problems solved to allow the robot to be used for remote kinematics laboratories.

Acknowledgements

There are a number of people I would like to thank for their help with this thesis:

A/Prof. James Trevelyan, my supervisor, for your encouragement when things were going well, your words of wisdom when things looked down, and your guidance when I was lacking direction. It's been great to work with you.

A/Prof. Karol Miller my interim supervisor, for getting me started at the beginning of the year when James was away, and for your wise advice and good humour. Thanks also for the chance to demonstrate laboratories and for your openness to the idea of making a remote laboratory for your unit.

Sabbia Tilli, for putting up with my constant badgering about things I could have found out for myself, and for being a friendly face in the lab.

Angus Stuart, from tech support, for your excellent service. Talking to final years from other schools, you must be the most efficient and responsive tech support department in the faculty.

Andrew Guzzomi, for being a great friend for the last seven years. Thanks for all the good times and for proofreading my thesis many times over. I hope our friendship remains strong into the future.

Daniel Bond, for the 2 a.m. visits down the hallway, great friendship and the chance to put my mini cricket bat to good use.

Matt Harding, for your Godly mateship, encouragement and support. Thanks for your wisdom when I needed it, and for the late-night hymn-singing in the lab.

Everyone at St. George's College and the UWA Christian Union for making my time at uni much more fun and worthwhile than just a degree.

Mum & Dad, for being more generous and encouraging to me during my education than I ever could have asked for. Thanks for your love, support and continual encouragement in everything I do. This thesis is for you.

Contents

List of Figures

List of Tables

Glossary

Where terms listed in the glossary are mentioned for the first time, they appear in a margin note.

Glossary term

ABB

Asea Brown Boveri, a Swedish automation company with a robotics division. Makers of the IRB 1400 robot. Website: http://www.abb.com/

ActiveX

A Microsoft technology that specifies rules to govern how different programs interact. ActiveX is used to provide the communications link between the S4 Controller and the Hardware Master.

Adept

Large multinational automation systems manufacturer. Website: http://www.adept.com

Asea

Robotics company that later became ABB.

BCF

Base Coordinate Frame. The coordinate frame at the base of the robot.

Block Diagram

The ‘code’ of the LabVIEW VI. It shows all the wiring of inputs and outputs.

CGI

Common Gateway Interface. A set of rules that define how a web server communicates with another piece of software on the same computer. CGI scripts are often used on the internet to do some task on the server and give a remote user some information.

End-effector

The end part of a robot. This could be a gripper (as in the case of the Telerobot), or any number of tools such as arc-welding rods.

Forward kinematics

Calculating the position of the end-effector given the joint angles of the mehcanism.

Front Panel

The part of the LabVIEW program that the user interacts with. It can contain many different sorts of inputs and displays.

Hardware Master

The local computer in the same physical room as the laboratory equipment that constantly monitors and controls the state of the equipment.

Inverse kinematics

Calculating the joint angles of the mechanism given the end-effector position and orientation.

IRB 1400

A model of robot made by ABB. This is what the UWA Telerobot is.

Java

A full-featured, modern, widely-used programming language developed by Sun Microsystems. Designed to be able to run one program on multiple different platforms (Mac, Linux, Windows). Often used in remote laboratory contexts because of its multi-platform ability and the capacity to be run in a web browser over the internet. Website: http://www.sun.com/java/

Kinematics

Equations that describe how mechanisms move. In robotics, how the joint angles and end-effector position are related.

LabVIEW

A powerful graphical dataflow programming language created by National Instruments. It is particularly useful for data acquisition and analysis. The Telelabs system is built with LabVIEW. Website: http://www.ni.com/labview/

LOL

Labs-On-Line, the software that powers the Telelabs remote laboratory system

NUWAR

New University of Western Australia Robot, a parallel pick-and-place robot created by A/Prof. Miller, School of Mechanical Engineering, The University of Western Australia

Object

In software engineering, an object represents a set of related pieces of information. For example, a *robtarget* object has pieces of information to specify the *x*, *y* and *z* coordinates and the orientation.

RAPID

ABB programming language used to program the S4 Controller.

Remote Client

A remote interface to the laboratory equipment that allows a user to use the equipment remotely. It communicates with the LOL server, sending commands and receiving feedback.

Remote laboratory

A laboratory using real, physical laboratory equipment controlled from a remote location, often through the internet.

Repeatability

The ability of a robot to move back to a point within a certain error

RobComm

The Windows software from ABB that provides the ActiveX routines to allow communication between the S4 Controller and the Hardware Master.

S4 Controller

The ABB proprietary, low-down robot controller. This is what figures out how fast to move the motors and when to stop. The controller runs ABB's operating system, S4.

Singularity

A position in which the Jacobian matrix of a robot (which defines the relationship between the end-effector position and the joint angles) becomes singular (that is, it has no inverse). This corresponds to the robot losing a degree of freedom (it cannot move in a certain direction).

SPICE

Widely-used circuit simulation software. Created by the University of California in Berkeley. Website: http://bwrc.eecs.berkeley.edu/Classes/IcBook/SPICE/

SubVI

A LabVIEW VI that has an input and an output, and is run by another VI. One 'high-up' VI may call many 'lower-down' VIs, each of which does a small task.

Teach pendant

The manual joystick and control box used to manually move ('jog') the robot.

Telerobotics

Controlling robots remotely.

TypeDef

A type of LabVIEW VI that defines what a certain type of variable contains.

UCF

User Coordinate Frame. A coordinate frame that specifies the position of a fixture near the Telerobot. In the current setup, this is on the corner of the worktable.

VI

Virtual Instrument. This is a program in LabVIEW, with a Front Panel containing inputs and displays, and a Block Diagram, showing the wiring and code.

WCF

World Coordinate Frame. A robot coordinate frame defining the reference to the floor.

CHAPTER 1

Introduction

ENGINEERING students can benefit from remote laboratories. Recent literature has shown that students learn just as effectively using real laboratory equipment operated remotely as they do using the equipment in the laboratory room. The University of Western Australia Telerobot has now been online for ten years, but has not yet been used remotely for teaching purposes. The aim of this project was to modify the Telerobot software to allow kinematics laboratories to be performed remotely.

1.1 Thesis overview

This thesis can be broken down as follows:

- Chapters 2–5 provide the background to the project, some research into how remote laboratories are currently being used, and show how this project fits in to the current teaching programme at The University of Western Australia
- Chapters 6–8 detail the main three technical developments completed in the project
- Chapter 9 summarises the results from the project
- Chapter 10 provides some conclusions and outlines some areas of possible further work

CHAPTER 2

Background

THE work completed for this project is a significant step in a series of developments involving the University of Western Australia Telerobot. It is also part of a larger project known as Telelabs. This section gives a brief background of the Telerobot, the Telelabs system and the software used to power them.

2.1 Some definitions

Before there can be any discussion of the background to the project, the reader will need to know simple definitions of some technical terms:

Remote laboratories:	Laboratories using real, physical laboratory equipment controlled from a remote location, often through the internet.
Telerobotics:	Controlling robots remotely.
Kinematics:	Equations that describe how mechanisms move. In robotics, how the joint angles and end-effector position are related.

A full list of technical terms used in this thesis can be found in the Glossary. Where terms listed in the glossary are mentioned for the first time, they appear in a margin note.

Glossary term

2.2 The Telerobot

The UWA Telerobot is located in the School of Mechanical Engineering at The University of Western Australia. It has been online[1] for ten years, and was the first robot of its kind in the world to be operated through the internet. It has been used by many students and third-parties to move building blocks on its work table (Figure 2.1). The Telerobot has also been used in laboratory sessions, but only using manual control, not remote control.

[1]http://telerobot.mech.uwa.edu.au/

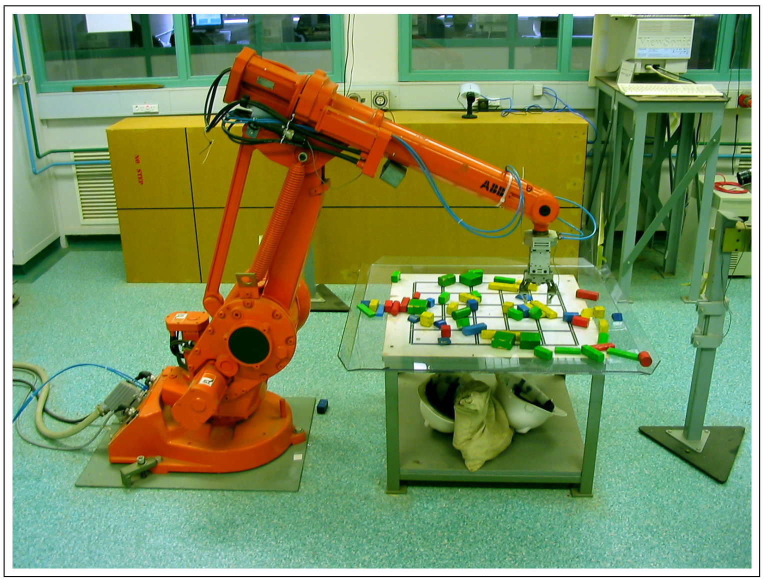


Figure 2.1: The University of Western Australia Telerobot

ABB

IRB 1400

End-effector

Repeatability

The Telerobot is a commercial ABB[2] IRB 1400 robot. It is floor-mounted and has six rotary joints actuated by AC electrical servo motors. It has a small footing (620*mm* × 450*mm*) and a large workspace (Figure 2.2), making it ideally suited to applications that have restricted floor space yet require a large working range. It has a top speed of $2.1m/s$ (moving the end-effector along a straight line), and has a repeatability of 0.05mm. This type of robot would typically be used in industry for tasks such as welding and materials handling [1]. The Telerobot's current end-effector is a gripper actuated by compressed-air, which is used to manipulate the building blocks.

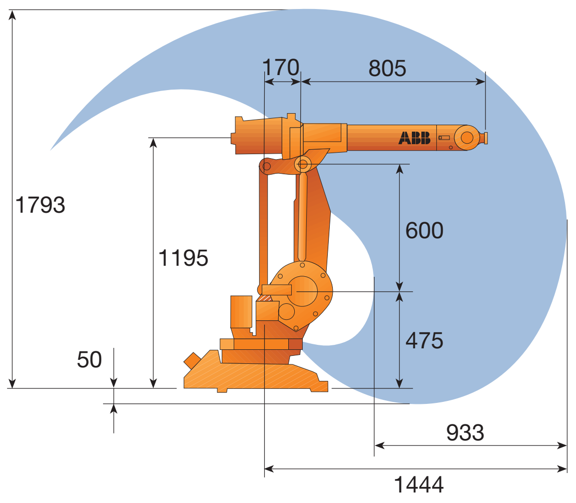


Figure 2.2: Workspace of the IRB 1400 robot (measurements in *mm*) [1]

[2]Asea Brown Boveri (http://www.abb.com/robotics)

2.2.1 A brief history of the Telerobot

The original Telerobot was an Asea IRB-6 Robot, which was initially put online by Ken Talyor [2] in 1994. The web-control was powered by a set of CGI scripts. The IRB-6 robot was replaced with the current ABB IRB 1400 robot in 1996. Various improvements were made to the system software, and further cameras were added to give the user more views of the work area.

Asea

CGI

In 1998, Dalton [3] rewrote the Telerobot code in Java[3]. The new software proved very capable and popular. After Dalton left UWA in 2001, the Telerobot code was no longer maintainable. In 2003 Babbage [4] rewrote the Telerobot code in LabVIEW and, with the help of A/Prof. Trevelyan, integrated it into the Telelabs system.

Java

LabVIEW

There have been other more recent developments on the Telerobot. For example, Palmer, Goh, Walker, and Chew [5] created an Augmented Reality interface to manipulate blocks on the table just by dragging them around on a web-cam image. This is not currently integrated into the main Telerobot software.

For a more detailed history of the Telerobot, see Babbage [4].

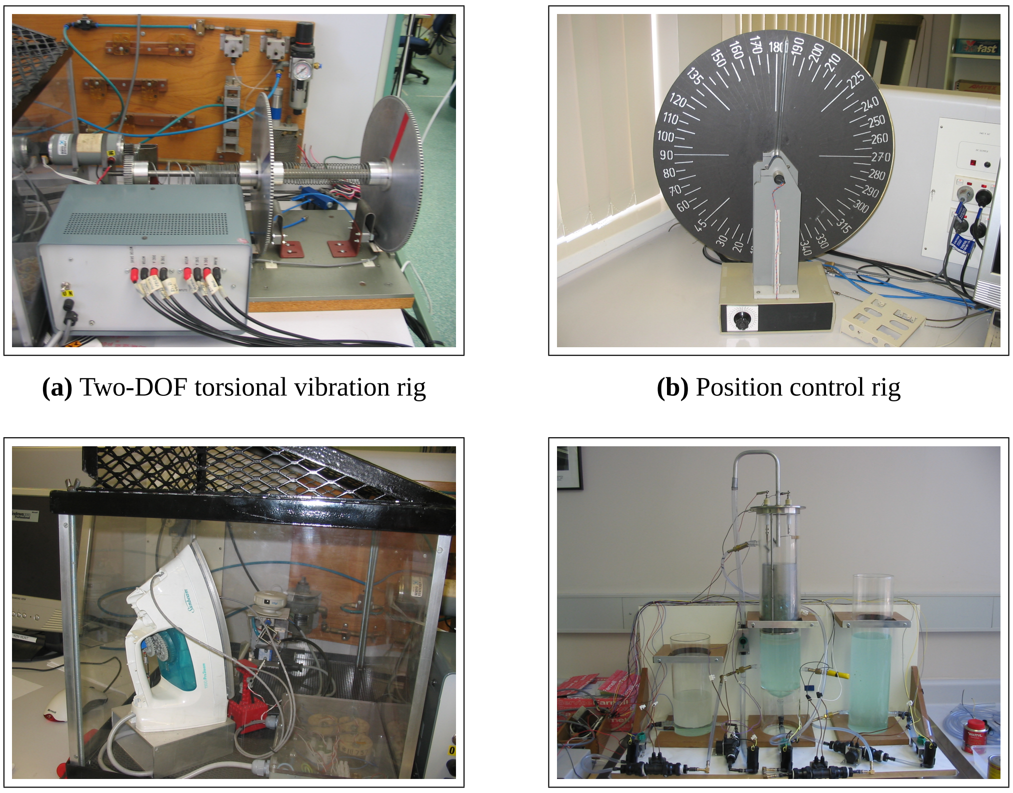
2.3 Telelabs

The Telelabs system, created by ICON Technologies[4] and UWA Mechatronics, is a platform that facilitates the control of laboratory equipment remotely [6]. Any type of experimental equipment may be integrated into the system. The system itself does not know how to control any particular rig, but rather provides all the communications between the hardware and the remote user, and handles all the accounts and administration. Equipment currently available with the Telelabs system includes:

- Two-Degree-Of-Freedom (DOF) torsional vibration rig (Figure 2.3a)
- Position control rig (rotational) (Figure 2.3b)
- Electric iron—temperature control (Figure 2.3c)
- A liquid-to-liquid contactor—simulating part of the production process of Liquid Natural Gas. To be added end-2004. (Figure 2.3d)
- The Telerobot

[3]Java™ by Sun Microsystems (http://www.sun.com/java/)

[4]http://www.icon-tech.com.au/



(a) Two-DOF torsional vibration rig

(b) Position control rig

(c) Electric iron

(d) Liquid-to-Liquid Contacter

Figure 2.3: Equipment currently integrated into Telelabs

2.3.1 Telelabs control scheme

Hardware Master

Each piece of laboratory equipment is controlled by a software controller written in LabVIEW, which runs on a nearby computer in the laboratory room. This is called the "Hardware Master". The Hardware Master controls all the actual inputs sent to and outputs received from the equipment, and monitors the equipment so that it is always safe (for example, monitoring the iron temperature to ensure it is within safe limits).

LOL

Remote Client

The Hardware Master communicates with the Labs-On-Line (LOL) server, which accepts connections from "Remote Clients" and handles all the communication between the Remote Client and the Hardware Master. This is a form of "supervisory control", which allows a user to control the equipment remotely while the local controller ensures tasks are performed within reasonable limits. The Hardware Master has built-in control and limit schemes that may override a user's commands to ensure the safety of the equipment.

Figure 2.4 shows an example of a Remote Client interface. It shows the graphs of temperature and other variables on the electric iron. Client interfaces may also have streaming (real-time) video of the rig if it involves physical movements.

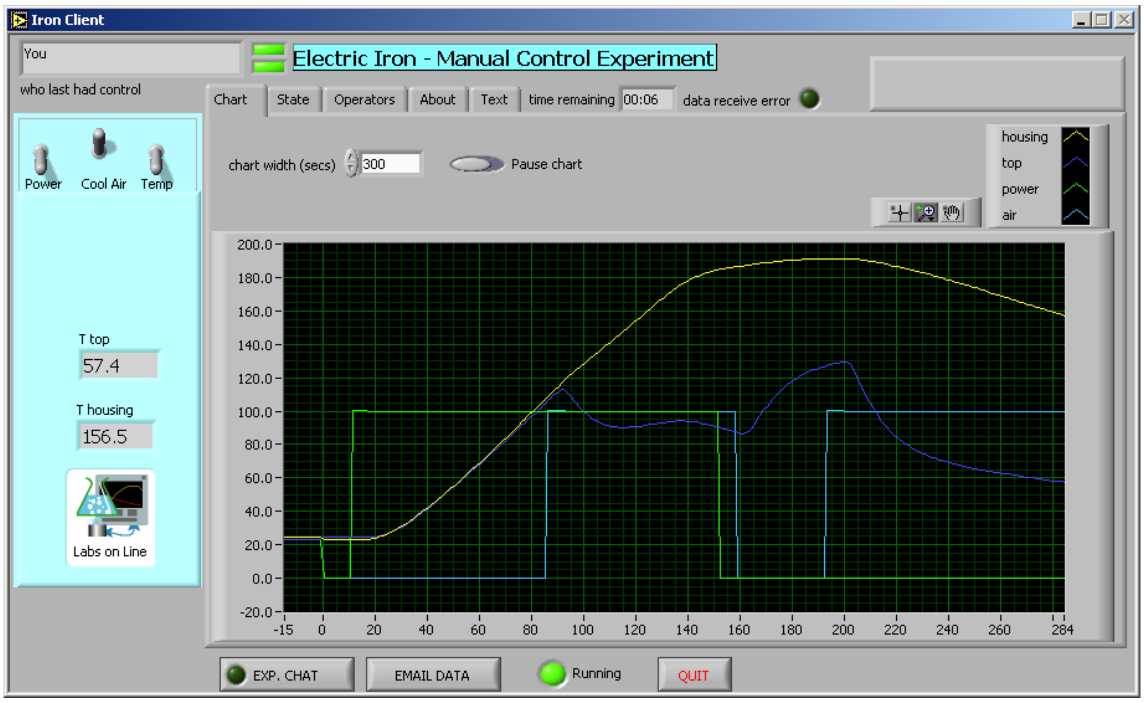


Figure 2.4: Remote Client Front Panel for the electric iron, showing two temperature curves (yellow and dark blue) and power (green) and cool air (light blue) step functions

2.4 LabVIEW

The Telelabs and Telerobot software is made using the LabVIEW environment. LabVIEW (Laboratory Virtual Instruments Engineering Workbench) is a graphical data-flow programming language created by National Instruments[5]. It is very powerful and easy to use, and is particularly useful for data acquisition and data processing applications. A LabVIEW program is called a Virtual Instrument (VI), and has two parts: the *Front Panel* and the *Block Diagram*:

VI

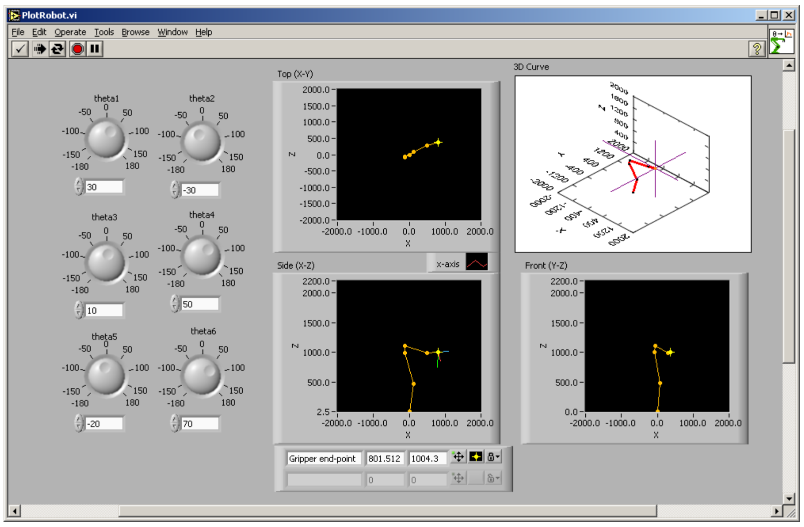
Front Panel

Block Diagram

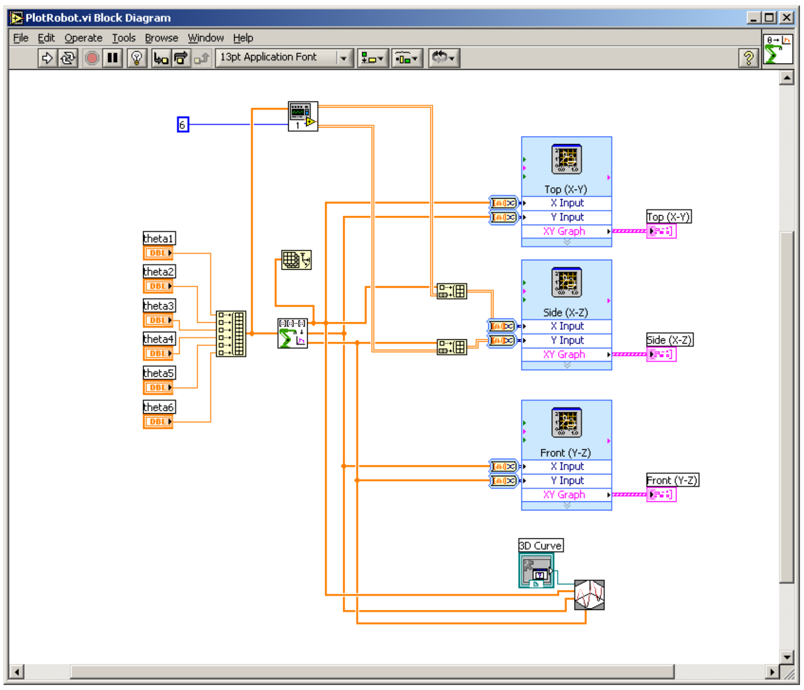
- The Front Panel shows a virtual instrument panel with various buttons, knobs, sliders and indicators. This is how the user interacts with the program (Figure 2.5a).
- The Block Diagram is the graphical 'code', with the wires showing the flow of data (Figure 2.5b).

LabVIEW was initially chosen as the software for Telelabs because of its attractive price and licencing structure, but has since proved to be an excellent platform on which to build the system.

[5]http://www.ni.com/labview/



(a)



(b)

Figure 2.5: PlotRobot.vi Front Panel (a) and Block Diagram (b)

CHAPTER 3

Current developments in the field

THERE is a growing interest from the academic community in the use of online methods for teaching, from primary school to University level. These methods can range from online discussion forums to simulations and remotely controlled laboratories. This section provides an overview of the current work in the fields of online learning, particularly remote laboratories and simulation, and also gives a brief summary of the various current uses for LabVIEW, the software used in this project.

3.1 Online learning

One area of online learning is using the web for online assessment and revision exercises. This type of system is currently used at The University of Western Australia in Electrical & Electronic Engineering[1] and the University has developed its own software for Mechanical Engineering[2] and Mathematics[3] courses.

Other universities have also implemented and tested online homework systems. Bonham, Deardorff, and Beichner [7] found that Physics students who completed assessed exercises online performed just as well as those who completed traditional paper-based exercises.

Similar results were found using online classrooms instead of traditional lectures. Kekkonen-Moneta and Moneta [8] at the Hong Kong University of Science and Technology found that students taught from online resources were as capable as, and in some cases outperformed, students taught from traditional lectures. The Faculty of Arts at UWA uses an online lecture system, allowing students to listen to lectures in the computer labs, at home or anywhere else, including the University's country campus[4].

[1]Mallard (http://www.cen.uiuc.edu/Mallard/)
[2]Jellyfish (http://www.mech.uwa.edu.au/login.html)
[3]Calmæth (http://CalMaeth.maths.uwa.edu.au/)
[4]iLectures (http://ilectures.uwa.edu.au/)

In some cases, online learning has been taken to extremes. The Hong Kong CyberU is a new online wing of the Hong Kong Polytechnic University that has most of its classes taken online. Some classes are still held at the University campus, giving a “blended” approach to learning. Cheung [9] looked at Postgraduate students undertaking an MSc in e-Commerce at the CyberU, and found that there was little difference between those who studied the program traditionally and those who studied online.

One of the benefits of online learning is that students can work at any time and place [10]. Wilkins and Barrett [11] report on the creation of an online database of construction sites to replace site visits, which are difficult to timetable, and are highly dependent on current construction sites and access permission. Goldberg et al. [12] similarly wanted to retain the benefits of field trips without the practical difficulties of group travel. However, while online facilities may offer greater convenience, one would begin to think that they may defeat the entire purpose of gaining experience on real sites.

3.2 Remote laboratories

In the last few years, several universities have experimented with remotely operated laboratory equipment. Motivations behind producing these remote laboratories include the possibility to improve distance education, timetabling issues, accessibility and the cost of running laboratories. Li, Wang, Lai, and Wu [13] also cite the scarcity of equipment and supervisors as a reason for pursuing remote laboratories.

From an educational point of view, research shows that remote laboratories are as good as several other forms of teaching. Wagner and Tuttas [14] and Esche, Chassapis, Nazalewicz, and Hromin [15] found that students using remote laboratories learnt just as much as those using traditional local laboratories.

Remotely operated laboratories are either created exclusively for one particular piece of equipment, or are platforms to which any number of different pieces of equipment can be added. The following sections provide current examples of both types of system.

3.2.1 Multi-Laboratory systems

Multi-Laboratory systems are platforms upon which any number of remote laboratories can be built. They are not equipment-specific, but rather supply a standard architecture upon which specific laboratories can be built. This is the type of system the UWA Telerobot is integrated into. Some of the current contributors to these systems are:

- Telelabs: The University of Western Australia [17]
- Cyberlab: Norwegian University of Science and Technology [16]
- ACT: The University of Siena [18]
- R-Lab: Eastern Mediterranean University [19]
- DIESEL: The University of Ulster [20]
- ReMLab: Politecnico di Milano [21]
- Lab-on-web: UniK—Center for Technology, Norway, Norwegian University of Science and Technology & Rennselaer Polytechnic Institute [22]
- WAVES: University of Arizona [23]

3.2.2 Single-Laboratory systems

Single-Laboratory systems are developed as the need arises for a single piece of laboratory equipment. There are numerous online laboratories in electronic engineering that observe the behaviour of circuits, including basic electric circuits [24], integrated circuits [25] and semiconductors [26].

Many online laboratories also involve control systems. These systems include common DC motor control [27, 28] and position control [29], and many other systems, from controlling water levels in tanks [30] to controlling temperature and toy helicopters [31].

These remote laboratories often make equipment more accessible. In some cases, they may offer students access to equipment that was previously unavailable. One interesting example is allowing school students to operate a radio telescope [32], obviously a rare piece of equipment.

3.2.3 Robotics

Remote control of robots is clearly of importance for this project. There have been several recent experiments in controlling and programming robots remotely, similar to the University of Western Australia Telerobot. Marin, Sanz, Nebot, and Esteller [33] have set up two educational and two industrial robots for remote use, and Lowe and Cambrell [34] have made available an Adept robot and another 6-axis robot. In a slightly different application, Skrzypczynski [35] has enabled network-access to mobile robots, allowing students to manipulate the robots via a web-browser interface.

Adept

3.2.4 Issues with remote laboratories

Laboratory demonstrators
One problem with remote laboratories is the obvious absence of a laboratory demonstrator. This means students have little assistance or immediate feedback when performing the laboratory. Research at the School of Mechanical Engineering at the University of Western Australia has shown that students value highly good demonstrators, especially how well the demonstrators are informed, how friendly they are, and their proficiency with the English language [36].

Motuk, Erkmen, and Erkmen [37] propose an intelligent coaching program to help students while they are performing the laboratory. Li et al. [13] likewise designed an autonomous coaching program to help students when they are in “bad need of teachers’ guidance”.

Cost effectiveness
Despite some papers citing lower costs as a motivation for creating online laboratories [21, 15], there appears to have been no detailed study of the cost-effectiveness of remote laboratories compared to traditional laboratories. It is unclear whether the costs of developing and maintaining remote laboratories are lower than the costs of running traditional laboratory classes in physical laboratories. Indeed, instead of spending time and money creating coaching programs as mentioned previously, it may be more cost-effective and helpful for students if the university were to simply pay a well-informed and friendly laboratory demonstrator. These factors need to be weighed against the cost of purchasing and maintaining more physical equipment.

3.3 Simulation

Simulation is an alternative to traditional and remote laboratories. Simulations are simply mathematical models of real systems. They can be used to enhance visualisation of systems, make lectures more exciting, and allow students to explore projects at their own pace [38].

Simulation has many uses. It can be used to learn about particular systems, from cellular biology [39] to microcontrollers [40]. It can also be used for technical analysis of systems. For example, the circuit simulation software SPICE[5] is very widely used in Electrical Engineering. Simulation can also be used for scenarios such as large management projects, which are otherwise too difficult to create [41, 42]. While much of the current simulation activity seems to be in electrical engineering applications such as circuits [43, 44], it is also used in mechanical [45] and mechatronic [46] engineering.

SPICE

Simulation is used side-by-side with Telerobotic laboratories: Chong et al. [47] use a local simulation to compensate for time delay; Safaric, Hedrih, Klobucar, and Sorgo [48] use simulation to enhance the visualisation of the robot and its workspace; Kuc, Jackson, and Kuc [49] and Marin, Sanz, and Pobil [50] allow students to practise with a simulation before sending their instructions to the real robot.

Compared to real laboratories, Foss et al. [16] argue that while simulations can be useful in engineering education, there are aspects of physical systems that simulations cannot replicate, including the obvious connection to the real world and the level of complexity in their behaviour. Corradini et al. [31] cite the same disadvantage with simulations, saying “the richness of physical realities is inevitably omitted”. Furthermore, it is expensive and time-consuming to create accurate simulations [19].

3.4 Current use of LabVIEW

As mentioned in section 2.4, LabVIEW is a powerful and flexible graphical development environment used for creating measurement and control applications. It is particularly useful for data acquisition and control. LabVIEW is becoming more popular in both academia and industry for rapidly developing control applications. Some examples of simulations developed in LabVIEW include boilers [51], dynamic light scattering [52]

[5]Created by the University of California at Berkeley (http://bwrc.eecs.berkeley.edu/Classes/IcBook/SPICE/)

and inter-chip connections in electronics [53].

Universities have used LabVIEW to control local and remote laboratories. Local applications include fault detection in hydraulics [54] and position control [55]. It has been used to create remote laboratories for testing integrated circuits [25] and for controlling telescopes [32], DC motors [27, 28] and even underwater vehicles [56]. Strandman et al. [22] have also created a remote laboratory framework with LabVIEW as one of its main components.

CHAPTER 4

Experiences with laboratories

LABORATORY classes are taken by every engineering student at UWA. Having a knowledge of students' experiences with laboratory classes provides some understanding of how an online laboratory should be set up and what problems it can fix. This chapter looks at some of the author's personal experiences of laboratory classes.

4.1 Personal Experiences

This section is written from a personal point of view, explaining some of the motivations behind this project and outlining the various changes made to the laboratory session involved. While focusing on the Telerobot laboratory in the third-year unit Mechanisms & Multibody Systems 319 (MMS 319), the conclusions extend to other laboratories.

4.1.1 Telerobot laboratory

As a third-year student I studied MMS 319 in 2002. The unit contained several laboratories involving unique equipment. Of these laboratories, one was on the New University of Western Australia Robot (NUWAR, a very fast parallel robot developed by A/Prof. Karol Miller), the second was on the UWA Telerobot and the third was on gyroscopes.

NUWAR

The Telerobot laboratory aimed to teach us how a real robot behaved, and to demonstrate a practical application of the kinematics theory shown in lectures. The robot was controlled entirely from the teach pendant (the manual joystick). Control of the robot was demonstrated by the laboratory supervisor, and we were each allowed a short turn controlling the robot. During the laboratory we performed forward and inverse kinematics calculations and checked our solutions by actually moving the robot.

Teach pendant

As the equipment was quite unique, the laboratory groups were large. The size of the groups and the following factors meant the laboratory was of little assistance in gaining an understanding of the robot:

1. The laboratory sheet was difficult to follow
2. The laboratory usually proceeded at the pace of the fastest student
3. Some time was spent waiting idly for others to complete parts of the laboratory

Laboratory sheet

On a purely aesthetic level, the laboratory handout (Appendix A) was difficult to read. The pages contained large blocks of unbroken text, with a mix of diagrams at the back of the handout. It was difficult to follow exactly what the sheet required, as there were no obvious steps to work through. The tasks were bulleted, but at a glance the points were almost indiscernible from the rest of the text.

Pace

Students were, as always, keen to complete the laboratory as quickly as possible. This meant that as soon as one student achieved a result, the remaining students ceased to try, as they could get the result off the first student later (and leave the laboratory sooner). The result of this was that the pace of the laboratory was always that of the fastest person or sub-group of keen students. Hence, everyone except the fastest student (or fastest few) were left behind.

The majority of the group did not finish the calculations and did not have time to understand the context or material of the laboratory. From observations during my demonstrations of the laboratory in 2004, the group dynamics also begin to collapse when the group breaks down into smaller groups of faster students explaining concepts to slower students. The laboratory ceases to be a valuable team learning exercise and easily becomes frustrating for both the students and the demonstrator.

Waiting

Ironically, even though most of the laboratory moved at a fast pace, there were times when the majority of the group had to wait for a student to finish a certain section. This was the case when students were given the opportunity to operate the robot. Most benefits were gained in operating the robot for yourself, and I was only able to learn a small amount from observing a classmate operate the robot. Most of this time spent waiting was thus unproductive.

CHAPTER 5

Telerobot laboratory

THIS project builds remote use of the robot into the existing traditional laboratory mentioned in Chapter 4. A technical understanding of this laboratory is thus needed to follow the direction of this project. Also, as established in Chapter 4, the laboratory had various difficulties. The laboratory handout was thus rewritten to make it easier to follow, and more helpful for the students. This chapter outlines the aims of the previous (traditional) laboratory and the changes made to it.

5.1 Laboratory context & aims

The Telerobot laboratory is part of the third-year unit Mechanisms & Multibody Systems 319. This unit takes students through the dynamics and kinematics of mechanisms such as planar linkages and gyroscopes. The laboratory session using the Telerobot was designed to give students some hands-on experience with a real industrial robot, to manually manipulate it, and to practise some simple forward and inverse kinematics .

Forward kinematics

Inverse kinematics

For the forward kinematics exercise, the robot was set to its zero position, and joints 2, 3 and 5 were moved to arbitrary angles. This effectively made the robot a 3-link manipulator in the x-z plane (Figure 5.1). Once the robot had finished moving, the joint angles and end-effector position were recorded. For the laboratory write-up, the students performed the forward kinematics calculations, and compared their calculated values for the end-effector position to the actual position obtained during the laboratory.

The inverse kinematics exercise was made even easier. Only angles θ_2 and θ_3 were changed (θ_5 was not used), creating a two-link manipulator in the x-z plane. Using the teach-pendant, the students moved the end-effector of the robot straight along the z-axis. The end-effector x and z coordinates and angles θ_2 and θ_3 were recorded for two positions. The students then performed the inverse kinematics calculations using the end-effector positions and compared their theoretical angles with those found in the laboratory.

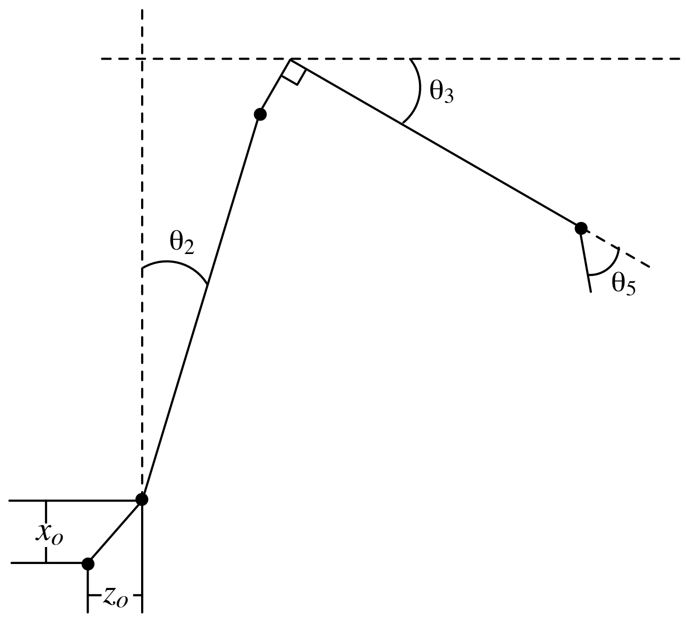


Figure 5.1: Kinematic diagram for the robot for the MMS 319 forward kinematics exercise

The kinematics part of the laboratory is well suited to remote operation, considering that it is primarily concerned with retrieving data and checking theoretical values against the actual rig. The idea was obviously viable according to one student in a laboratory class in Semester 2, 2004, who was overheard as saying “they should put this lab online”. Making the laboratory available online allows students to perform more calculations of their own and check them against the robot, and lets students do the kinematics part of the laboratory whenever they want.

5.2 Laboratory handout

The laboratory handout for the MMS 319 ABB Industrial Robot laboratory was rewritten to make it easier to follow and to correct some errors contained in the previous handout. The revised handout can be found in Appendix B. The changes made fall under the following headings.

5.2.1 Readability

As mentioned in Section 4.1.1, the previous laboratory handout was difficult to read because it was essentially one big block of text with occasional bullet points. It was difficult to see the individual tasks and what actually had to be performed for the laboratory. The new handout was rewritten to be more spaced out, to have obvious headings, and to provide places for students to write as the laboratory progressed. This made the laboratory handout much easier to follow, and meant the students would have a complete record of what the laboratory required and the results they obtained in one document.

5.2.2 Coordinate frames

One section of the laboratory involves the students finding the centre of the User Coordinate Frame (UCF) in terms of the Base Coordinate Frame (BCF). The UCF is located on a corner of the robot's work table. The previous laboratory handout mistakenly called the UCF the World Coordinate Frame (WCF). In fact, the WCF is coincident with the BCF because the robot is attached directly to the floor (the floor is the 'world'). Also the previous laboratory handout did not take into account that the *x* and *y* axes of the UCF point in opposite directions to the BCF (Figure 5.2).

UCF

BCF

WCF

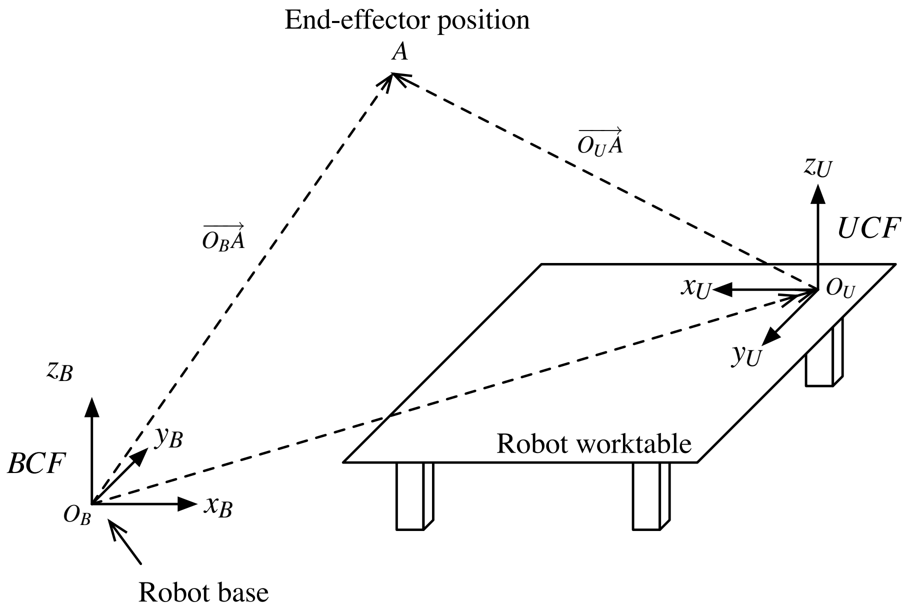


Figure 5.2: Base and User Coordinate Frames (BCF & UCF)

5.2.3 Robot diagrams

The dimensions of the robot were required to perform the kinematics calculations. The previous laboratory handout had several diagrams of the robot, some of which were superfluous. All but one of the diagrams were omitted. Further details were added to the remaining diagram to show all the lengths of the robot in its zero position in the x-z plane. The required lengths were calculated from the other diagrams. This saves several sheets of unnecessary diagrams.

5.2.4 Calculations

The kinematics section of the laboratory requires students to perform the forward and inverse kinematics exercises sometime after the laboratory in order to add them to the laboratory report. The wording of the previous laboratory handout made it seem like the calculations were meant to be performed during the laboratory session, although the intention was for students to work through the calculations at a later date. The wording of the handout was changed to make this clearer.

5.2.5 Inverse kinematics

For the inverse kinematics section of the laboratory, the students move the end-effector straight up the z-axis. The intention is that moving the gripper straight up the z-axis should change only θ_2 and θ_3. The previous laboratory handout advised to do this by first setting θ_1, θ_4, θ_5 and θ_6 to zero, and then move the gripper straight up the z-axis. However, when this is done the robot controller keeps the gripper at the same orientation (relative to fixed global axes), resulting in a change in θ_5 (Figure 5.3). The solution to this was to move both θ_4 and θ_5 to 90°. This moves the gripper out of the x-z plane (Figure 5.4), eliminating any gripper length in the x-z plane, and therefore removing any reference to θ_5. The end point is taken as being straight through the end of link 3 (the horizontal link when in the zero-position).

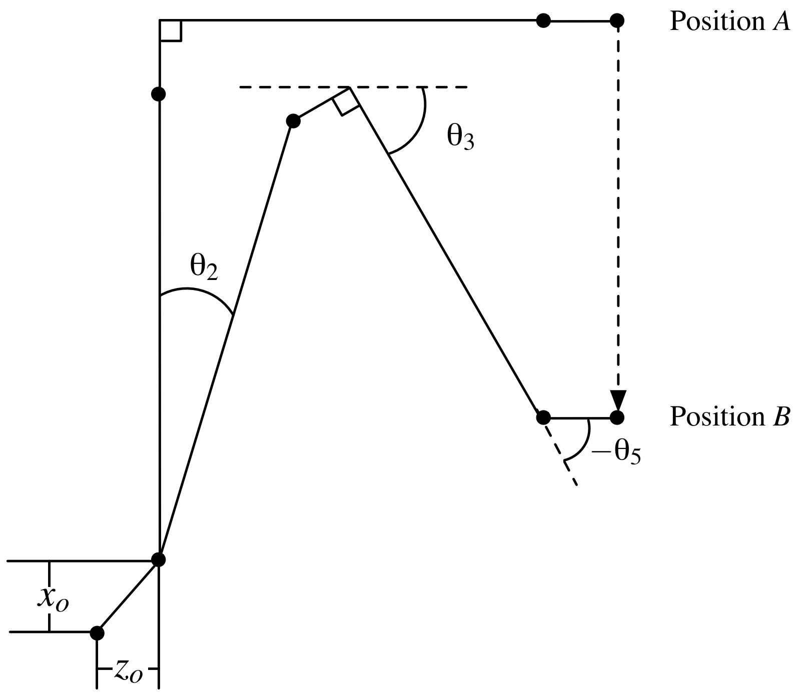


Figure 5.3: Kinematic diagram showing θ_5 changing in the original inverse kinematics exercise

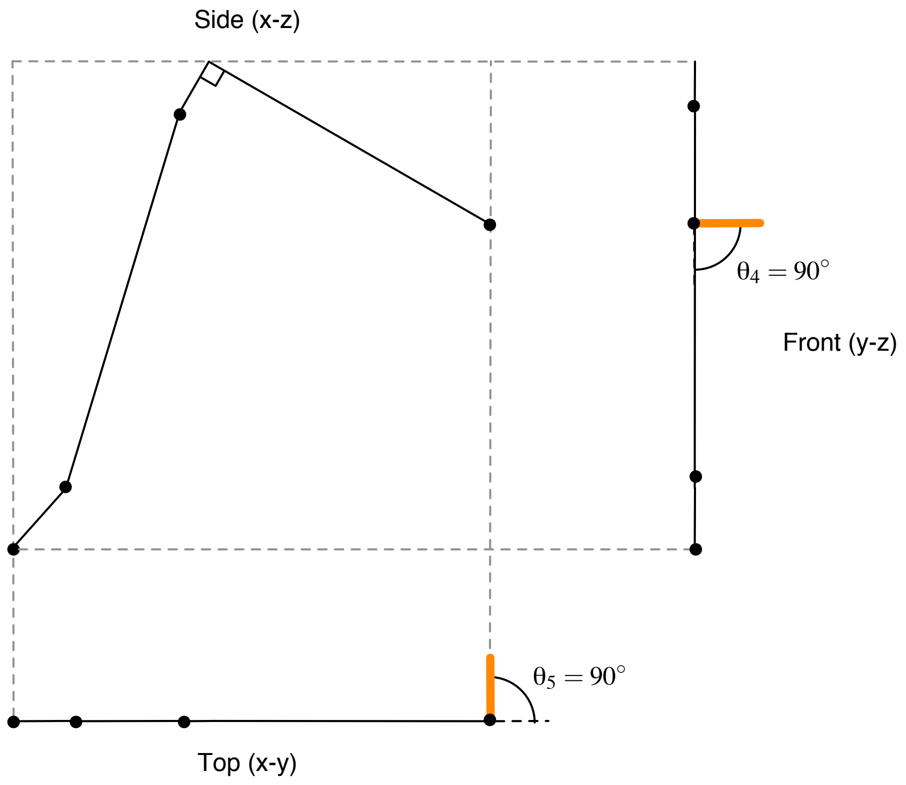


Figure 5.4: Setting θ_4 and θ_5 to 90° points the gripper (highlighted) out of the x-z plane

CHAPTER 6

Remote Joint Control

THE ability to move the Telerobot to a set of joint angles is essential for an online kinematics laboratory. Students must be able to move the robot to a specific configuration to check their forward kinematics solutions. Likewise, it is essential to be able to read the current joint angle values from the robot. This is important to see that the robot actually ends up where it was expected to go, and also to check inverse kinematics solutions.

Moving the robot to a set of joint angles also allows the robot configuration to be definitely specified. Currently, if the robot is instructed to move its end-effector to a position and orientation (in cartesian coordinates), there may be more than one configuration it can be in to get there. That is, there are multiple solutions. There is currently no way to guarantee the robot will use the intended configuration to reach the end point. By specifying a set of joint angles, the configuration of the robot links can be absolutely stated.

Moving the robot to a set of joint angles was previously thought to be impossible, but has now been achieved by using a dummy data structure. This chapter explains how sending and receiving joint angle data was done.

6.1 Communicating with the robot controller

S4 controller

Moving the robot to a set of joint angles requires the Hardware Master and the S4 controller (the ABB proprietary robot controller) to communicate with each other and send data back and forth. The Hardware Master packages a variable and sends it to the S4 controller, overwriting the variable in the S4 program's memory. The S4 controller then executes a program to move the robot, using the variables in the instructions.

ActiveX

Object

RobComm

ActiveX is used to package the objects and send them to the S4 controller. ABB supplies a suite of ActiveX routines in their RobComm software to provide an interface between LabVIEW and the S4 controller. This is how the position was sent to the robot previously (Figure 6.1).

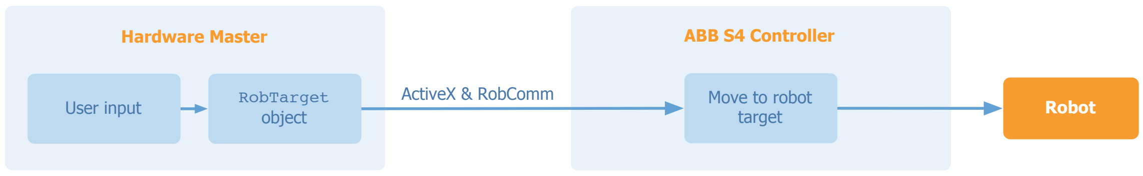


Figure 6.1: Changing robot position

6.2 Difficulties with ActiveX

For sending the position data, the position was packaged into a *robtarget* object, which is what the S4 position command requires, and sent to the S4 controller (Figure 6.1). However, to move the robot to a set of joint angles the required object is a *jointtarget*. Upon investigating the different ActiveX routines, it was found that the RobComm software does not provide a routine to package a *jointtarget* object. This made it impossible to send joint data to the robot in the same simple way the position data was sent.

In addition to this, there was little information within the Telerobot documentation on how to package an S4 variable in LabVIEW using ActiveX. There was some information about what the subVIs did in general, but very little about how ActiveX was integrated or how a future developer could use it to perform a new task. As a result, it was extremely difficult to create a subVI to write any kind of variable to an S4 program.

SubVI

6.3 Writing joint angles using a dummy data structure

To send joint data to the robot, a different method was created. Instead of attempting to transport a *jointtarget* data structure to the S4 controller, the joint angle information was put into a *robtarget* data structure (the one previously used by Babbage [4] to send position and orientation data) and then extracted on the S4 controller.

Babbage's subVIs were altered to create subVIs to package the angle information into a *robtarget* object. The angle information was then sent to the S4 controller, where a new program copied each element from the *robtarget* object into a *jointtarget* object (Figure 6.2). The *robtarget* object is simply being used as a 'dummy' data structure to transport the information from LabVIEW to the S4 controller (Figure 6.3).

In order to make this dummy data structure method possible, new code had to be added to both the LabVIEW program and the S4 program.

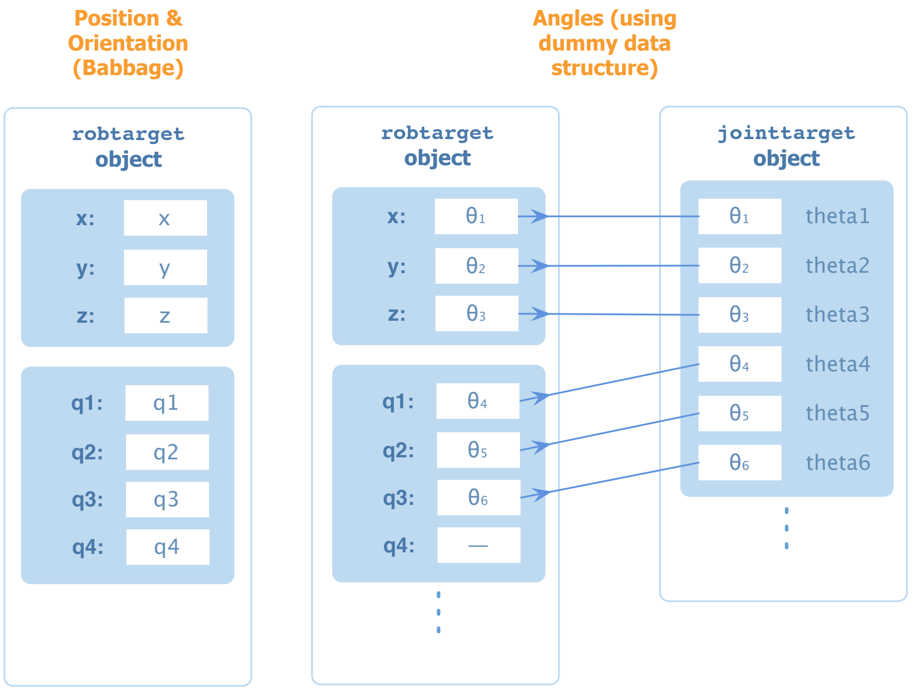


Figure 6.2: Using a dummy data structure to package joint angles

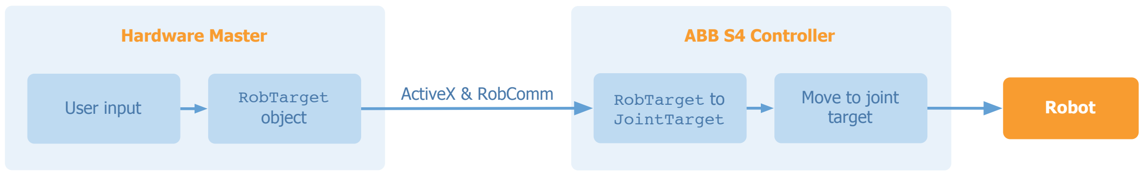


Figure 6.3: Changing the joint angles by using a dummy data structure

6.4 Reading the current joint angles

Reading the current joint angles back from the robot was done in a similar way to writing the angles. The joint angles were retrieved in the S4 program, put into individual *num* variables, and read one-by-one by the Hardware Master (Figure 6.4). The *robtarget* object was not used, because the function to read the current robot position and orientation generates the *robtarget* object automatically—it cannot be made to return specific values.

6.4.1 Problems reading the joint angles

There were two similar, but unrelated, problems in reading back the joint angles. One problem affected all moves; the other affected only long moves of the robot.

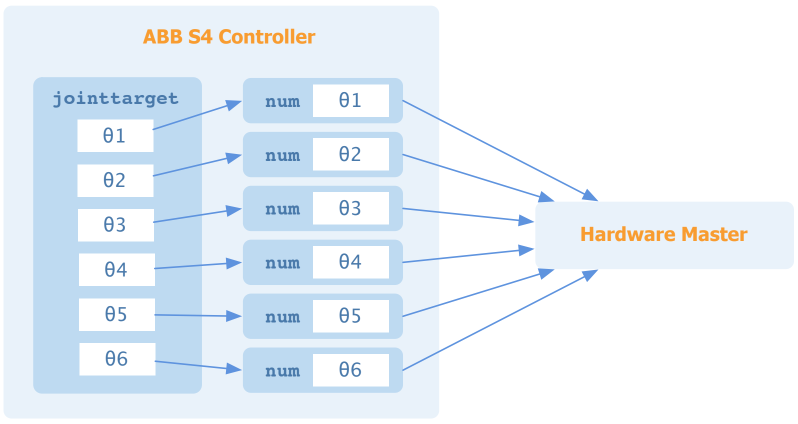


Figure 6.4: Reading the current joint angles

Timing problem affecting all moves

The S4 controller returned values of the joint angles to the Hardware Master about half a second before the robot had finished moving. This obviously meant that the position reported on the software interface was not the position the robot finally reached. The problem may have been because the instruction in the program to move the robot finishes slightly before the robot stops physically moving, and thus the program continues straight on before the robot has actually ceased moving (Figure 6.5).

This problem was solved by inserting a small time-delay between the robot movement instruction and the instructions to send the joint angles back to the Hardware Master.

Timing problem affecting only long moves

On long moves, the angles were returned to the Hardware Master several seconds before the end of the move. This was initially thought to be the same problem as before, simply exacerbated by a longer move. However, it was found that the problem actually resulted from an assumption in the Hardware Master code that any move would take no longer that seven seconds (Figure 6.6). This assumption was inserted in the original code because the software occasionally could not detect that the robot had finished moving. The time-limit patched this problem so that the robot would be available to move after seven seconds even if the Hardware Master had not been picked up that it had finished moving.

The delay of seven seconds was sensible for small moves around the table, which was all the original code allowed, but was too limiting for joint-control, where complicated moves from one extreme of the workspace to another could take a larger amount time. To fix this problem, the time limit was increased to 20 seconds.

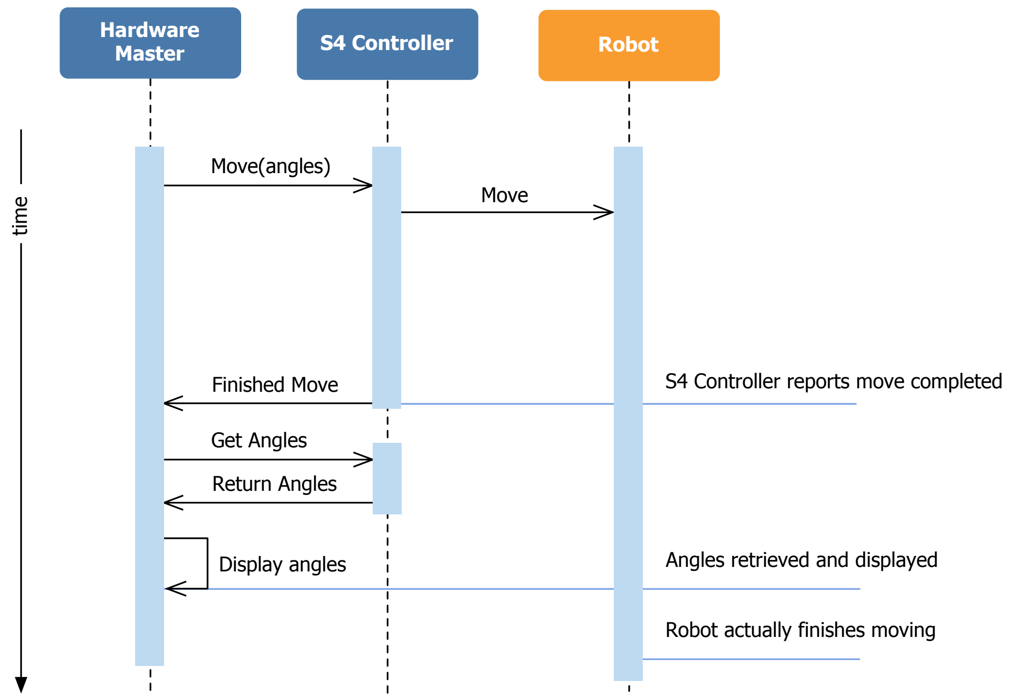


Figure 6.5: Sequence diagram showing the first timing problem, which affected all moves

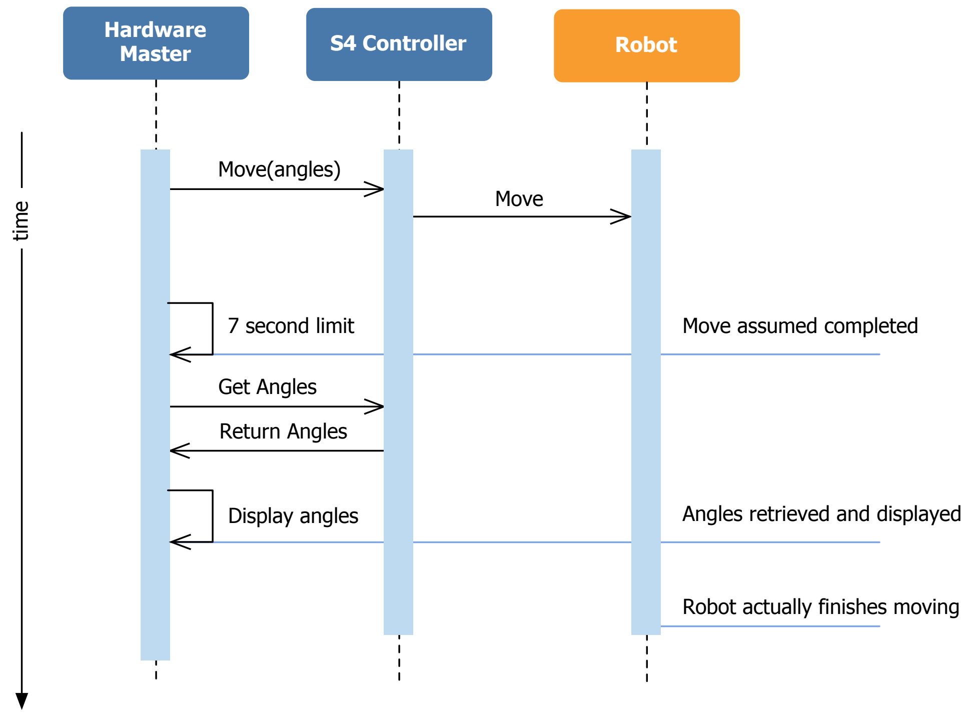


Figure 6.6: Sequence diagram showing the second timing problem, which affected long moves only

6.5 Changes to the LabVIEW program

Various modifications were made to the Hardware Master to allow joint angles to be sent to the robot. This section gives an overview of the changes made.

6.5.1 Joint controls

Controls were added to the Front Panel to allow a user to input the angles. In addition to this, a “Configuration Position” button was added to the Front Panel (Figure 2.4). This moves the robot to a configuration of angle values $(0,0,0,0,90,0)$, corresponding to the gripper being directly above the table, pointing downward (Figure 6.7). This is a ‘safe’ position to which the robot can be returned to should it get into a strange configuration. This configuration was chosen over the ‘zero-position’ (or ‘calibration-position’) at (0,0,0,0,0,0) because the robot has a singularity at that position, making it difficult to move. If the robot is in the Configuration Position it can be moved easily to the table or another configuration.

Singularity

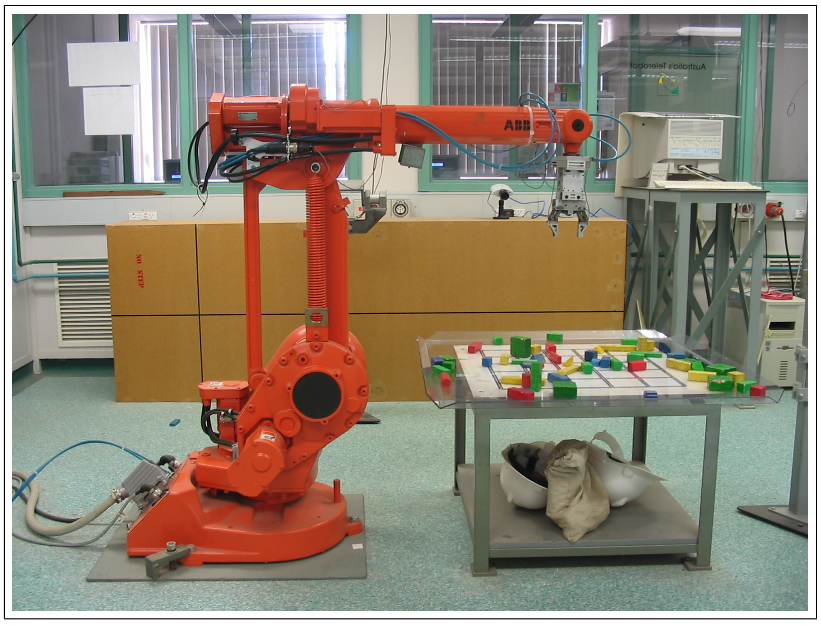


Figure 6.7: The Configuration Position

6.5.2 Control cluster

Redesign

The control cluster is an array containing all the moves awaiting execution by the Telerobot. The array is added to when a user sends a move command from either the Hardware Master or the Remote Client. Each element of the array originally contained variables for the position, orientation and gripper state. This had to be changed to allow joint angles to be sent as well as the position and orientation.

As any change to the control cluster would break all the old Remote Clients, the number of times the control cluster had to be changed needed to be minimised. It was therefore optimal to make all the required changes to the control cluster at once. Also, after consideration of what would be required in the future, allowances were made for expansion of the software.

Instead of limiting the types of movement by having unique variables for the position and orientation and the joint angles, a generic "move" array was used. This array contains either the joint angle values or the position and orientation, depending on the user's command. Having a generic "move" variable allows new types of moves in the future to still use the same structure. The code is thus quite extensible. In addition to this, variables for speed and coordinate frame were added. Newell [57] implemented these changes, as well as his own proposals, which included the priority and version number.

Definition

TypeDef

The Control cluster TypeDef, which defines what the Control cluster looks like, was defined as an array of an assortment of variables—there was no definition for a single move. This was changed so that the Control cluster was made up of an array of "Instructions", where an instruction was a single move, and had its own TypeDef. This made it much easier to deal with single movement instructions, instead of the entire array.

Precision

Newell's revised control cluster used low-precision *integers* (INTs) to store the movement data. Integers (as the name suggests) can only be integer values. This low level of precision was deemed insufficient for accurate control of the robot, so the "move" array was changed to the higher precision *double* (DBL), which allows decimal values. This allows a user to specify the joint angles to 0.01° (a greater number of decimal places can be used, but the user is limited to 0.01° increments by the Front Panel).

6.5.3 Packaging the joint angles

In order to send the joint information to the S4 Controller, the joint angles were packaged in a *robtarget* object. This was done by making a copy of Babbage's *S4Robtarget.vi* and wiring the joint angles in instead of the position and orientation.

6.5.4 Modularisation

Extensive modifications were made to the Telerobot LabVIEW code in order to allow control of the robot by both joint angles and position and orientation. A new variable (*mode*) was created to specify the type of movement. This had the initial options of "Cartesian" and "Joints", which were later changed to the more specific names of "Table" (for cartesian movements around the table) and "MMS" (for joint-control for MMS 319 students).

The Hardware Master and Remote Client Block Diagrams were modified to check the mode of movement and to place the appropriate data (joint angles or position and orientation) in the control cluster. Figure 6.8 shows the general structure of the subVI *RunHardware.vi*, which handles the movement of the robot. Figures 6.9a and 6.9b show a section of *RunHardware.vi* packaging and sending "Table" and "MMS" movement, respectively.

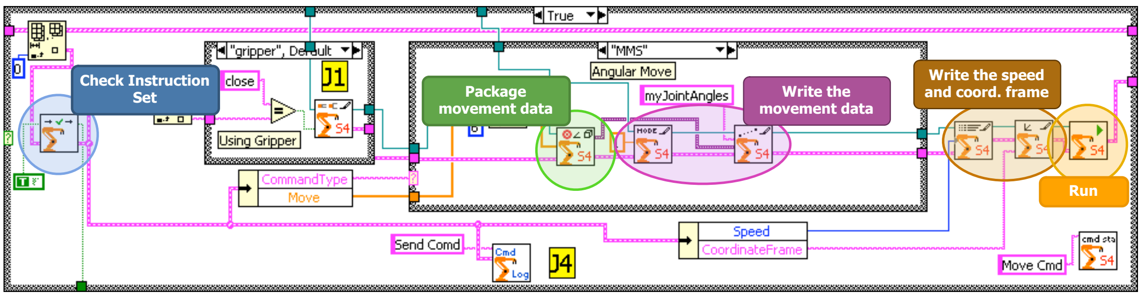
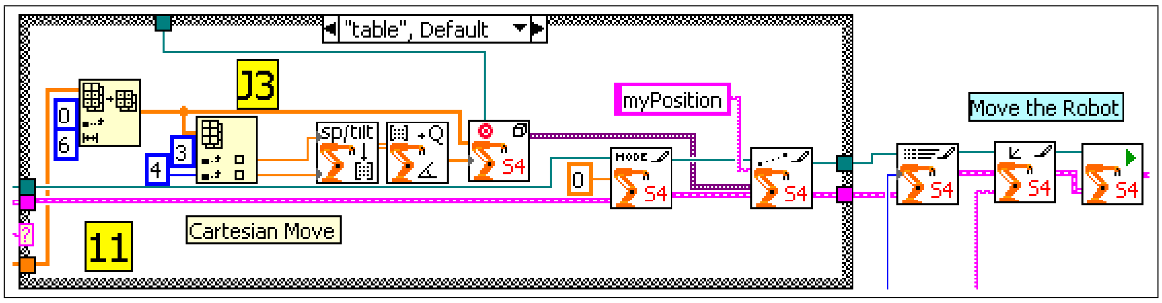


Figure 6.8: *RunHardware.vi* Block Diagram showing Instruction checking and data packaging and writing

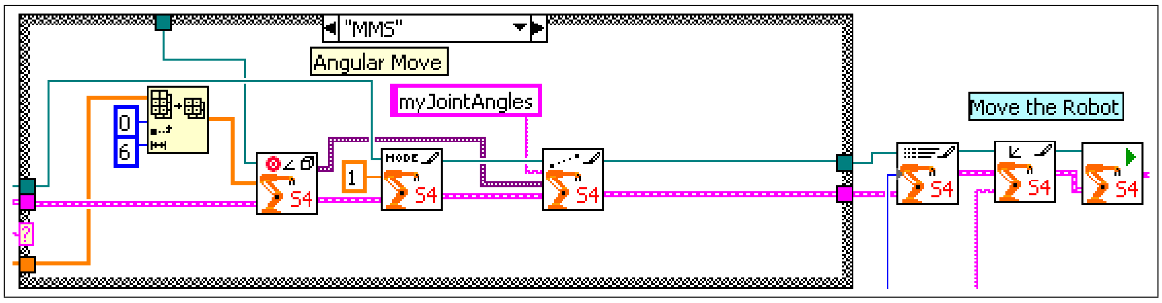
6.6 Changes to the S4 program

A new program had to be written for the S4 controller in order for it to know what to do with the angles. This required learning the RAPID programming language (used to program the robot controller), and modifying the movement program *MOVE.PRG*. A new

RAPID



(a) Position and orientation information



(b) Joint angle information

Figure 6.9: A section of *RunHardware.vi* packaging and sending different types of information to the robot

program, *MOVEII.PRG*, was created to determine the mode of movement and execute the appropriate movement instruction.

6.6.1 Program structure

The S4 program is contained in one file with multiple subrountines. The main program, *MOVEII*, is called by the Hardware Master. It then looks at the value of the variable *mode* and executes either the *jointsMV* or *tableMV* subroutine. An alternative structure would have been to have two completely separate programs that are called individually by the Hardware Master depending on the movement mode. The advantage of the subroutine structure is that it allows common code to be put before and after the movement-specific subroutines, and means only one file needs to be modified when something such as a variable name needs to be changed (Figure 6.10).

A full listing of the code for the original *MOVE.PRG* program and the new *MOVEII.PRG* can be found in Appendices C and D, respectively.

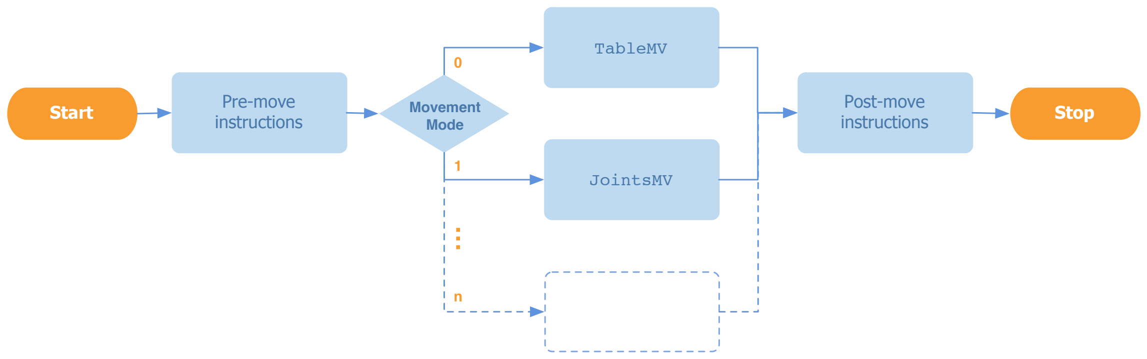


Figure 6.10: Structure of the new S4 program *MOVEII.PRG*

Keeping track of the mode

The *MOVEII.PRG* program keeps track of the current and previous movement modes. If the mode changes (for example, from "table" to "MMS"), the program first moves the robot through the Configuration Position. This acts as a safety precaution, taking the robot back to a default safe position before entering another mode and executing the move. This prevents potentially dangerous moves (that is, moves that could make the robot collide with the table, see Chapter 8) when changing from one type of movement to another.

6.7 Extending control to further variables

The method of using a dummy data structure can be used to transfer any type of data to the S4 controller. If an ActiveX routine exists to package the required data, a dummy data structure will, of course, not be required.

A quick inspection of the code for *MOVEII.PRG* in Appendix D reveals few fixed values, with the majority of the movement parameters set up to be easily changed from the Hardware Master. To demonstrate the simplicity of changing these parameters, the Hardware Master was set up to modify a further two variables: the speed and the coordinate frame. The maximum linear speed of the end-effector can be specified and is written to the S4 program by *S4WriteSpeed*. The coordinate frame defines the origin and orientation of the coordinate frame relative to which the position of the robot will be measured. The coordinate frame is written straight after the speed by *S4WriteFrame* (Figure 6.8). Changing other variables should likewise be relatively easy, requiring few modifications to the S4 program, and simply replicating some of the LabVIEW subVIs.

A 'How-to' document was created to show how to use ActiveX in the Telerobot software to modify S4 variables. This can be found in Appendix F.

CHAPTER 7

Robot kinematics

KINEMATICS is used to determine where the gripper will end up given the input joint angle values (*forward kinematics*), or to determine the values of the joint angles needed to give a particular gripper position and orientation (*inverse kinematics*). The kinematics of the Telerobot were solved so that a simulation of the robot could be created. A local simulation allows students to check their commands quickly and safely before they move the actual robot. This chapter explains the kinematics of the Telerobot and how it was implemented in software.

7.1 Homogeneous transformations

A Homogeneous Transformations (HT) is a way of describing a coordinate frame. It is a 4×4 matrix (square, so several transformations can be multiplied together) with a 3×3 rotation matrix in the upper left, and *x*, *y* and *z* displacements in the fourth column (Equation 7.1).

$$T = \begin{bmatrix} r_{11} & r_{12} & r_{13} & x \\ r_{21} & r_{22} & r_{23} & y \\ r_{31} & r_{32} & r_{33} & z \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{7.1}$$

Any number of HTs can be multiplied together to give the overall transformation from the initial coordinate frame to the final coordinate frame (Figure 7.1). For a robot, the HTs describing the transformations between each of the joints can be multiplied together to give the resulting end-effector position relative to the base of the robot (the base coordinate frame). A diagram of the Telerobot with all its coordinate frames can be seen in Figure 7.2.

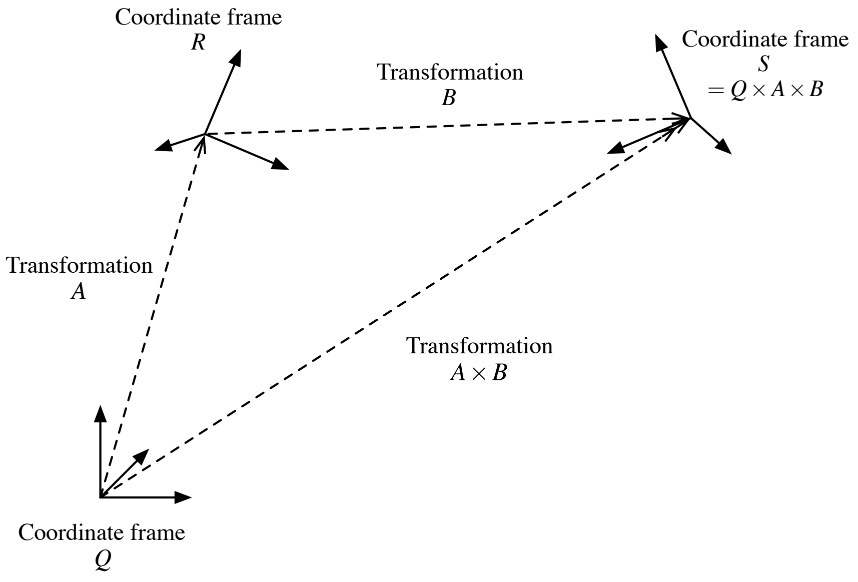


Figure 7.1: Multiplication of Homogeneous Transformations is associative. That is, the transformations can be multiplied together to give the overall transformation. The overall transformation here is $S = (Q \times A) \times B = Q \times (A \times B)$.

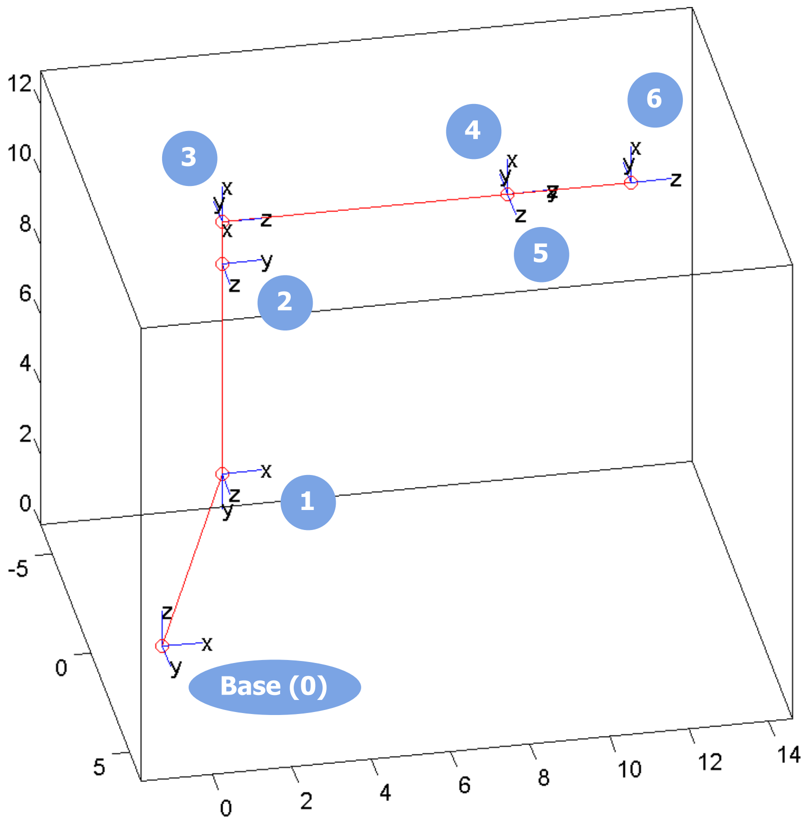


Figure 7.2: Coordinate frames of the joints of the Telerobot

7.2 Denavit-Hartenburg transformations

Denavit-Hartenburg (DH) transformations are a method of finding the HTs between the coordinate frames of each joint on a robot[1]. They use four simple transformations to get from one coordinate frame to the next:

1. a rotation about the z-axis—the axis of the joint—of θ_i (*joint angle*)
2. a translation along the z-axis of d_i (*offset*)
3. a translation along the x-axis of a_i (*link length*)
4. a rotation about the x-axis of α_i (*twist*)

The transformations are multiplied in order from left to right (Equation 7.2). This equation can be written several ways. Appendix G contains a discussion of the different conventions used in the literature.

$$T_{i-1}^{i} = Rot(z_i, \theta_i)\ Trans(0,0,d_i)\ Trans(a_i,0,0)\ Rot(x_i,\alpha_i) \tag{7.2}$$

These rotations and translations are simple HTs. Their forms are shown in Equations 7.3 to 7.5.

$$rot(x,\theta) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos(\theta) & -\sin(\theta) & 0 \\ 0 & \sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{7.3}$$

$$rot(z,\theta) = \begin{bmatrix} \cos(\theta) & -\sin(\theta) & 0 & 0 \\ \sin(\theta) & \cos(\theta) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{7.4}$$

$$trans(x,y,z) = \begin{bmatrix} 1 & 0 & 0 & x \\ 0 & 1 & 0 & y \\ 0 & 0 & 1 & z \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{7.5}$$

[1]For a detailed explanation of DH transformations see Craig [58, Chapter 3]

7.2.1 DH parameters

θ_i, d_i, a_i and α_i are known as the DH parameters. The transformation between any two consecutive joints on a robot can be described by a DH transformation using these parameters. Therefore, it is possible to create a table containing all the DH parameters for each transformation between links on a robot.

The DH parameters for the Telerobot were calculated, and are given in Table 7.1. The values of θ_2 and θ_3 are complicated because θ_2 is measured from the vertical, and θ_3 is measured from the horizontal, not from the last link as is the convention.

Table 7.1: Denavit-Hartenburg parameters for the Telerobot

Joint	**Angle** (θ)	**Offset** (d)	**Length** (a)	**Twist** (α)
1	θ_1	475	150	-90
2	$\theta_2 - 90$	0	600	0
3	$\theta_3 - \theta_2$	0	120	-90
4	θ_4	720	0	90
5	θ_5	0	0	-90
6	θ_6	310	0	0

7.3 Forward kinematics

By solving the forward kinematics it is possible to calculate where the end-effector will end up given the joint angles. Implementing this in software effectively provides a simulation of the robot.

Given the joint angles, the position and orientation of the coordinate frame at a joint on the robot can be found by multiplying the DH transformations as given in Equation 7.6.

$$ {}^{0}_{n}T = {}^{0}_{1}T \; {}^{1}_{2}T \; \cdots {}^{n-1}_{n}T \tag{7.6}$$

Using the DH parameters from Table 7.1, it is possible to find all the transformations for the Telerobot. These are listed in Appendix H. To find the position and orientation of the end-effector from the base, ${}^{0}_{6}T$, the DH transformations are multiplied as given in Equation 7.7.

$$ {}^{0}_{6}T = {}^{0}_{1}T \; {}^{1}_{2}T \; {}^{2}_{3}T \; {}^{3}_{4}T \; {}^{4}_{5}T \; {}^{5}_{6}T \tag{7.7}$$

${}^0_6T$ gives the total transformation from the base coordinate frame of the robot to the end-effector coordinate frame. That is, it describes the orientation and position of the end-effector relative to the base. This is the solution to the forward kinematics problem.

7.4 Inverse kinematics

Due to time constraints, the inverse kinematics was not fully completed. A module was created to solve the wrist angles using inverse Euler angles (see Craig [58, section 2.8]), but this gave incorrect values, and was not tested thoroughly. Once inverse Euler angles have been used to solve for θ_4–θ_6, angles θ_1–θ_3 can be found by simple geometry.

7.5 Implementation

This section gives an overview of the code used to implement the solution to the forward kinematics problem in LabVIEW.

7.5.1 Forward Kinematics

The implementation of the forward kinematics in LabVIEW follows the procedure outlined above to create the DH transformations and then multiply them together to give the end-effector position. Figure 7.3 shows the Block Diagram of the top-level forward kinematics VI, which sends the DH parameters and the angles to a subVI that multiplies the transformations together.

The transformations are created by a lower-down subVI (Figure 7.4). This calls smaller subVIs to create the HT matrices for the *rotz*, *rotx* and *trans* transformations mentioned previously.

Additional subVIs have been created to perform other functions not directly needed for the kinematics, but useful for general matrix manipulation.

7.5.2 Plotting

Once the forward kinematics is solved, the robot is plotted from three orthogonal views and a three-dimensional view (Figure 7.5). This plotting VI is intended as a preview of functionality that should be integrated into the Remote Clients.

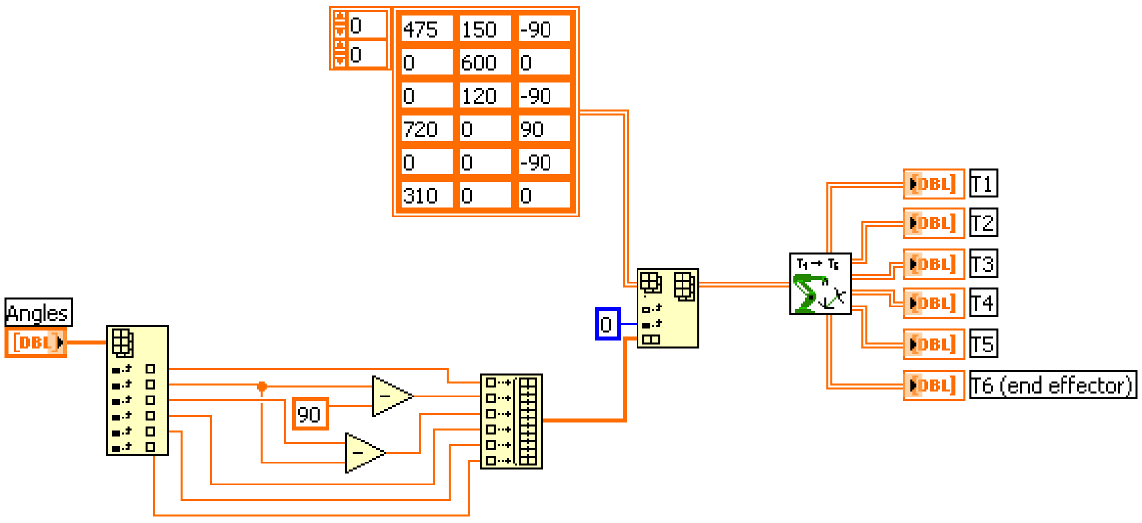


Figure 7.3: *TelerobotForwardKinematics.vi* Block Diagram showing the input of the DH parameters and using subVIs to solve for the HT at each joint

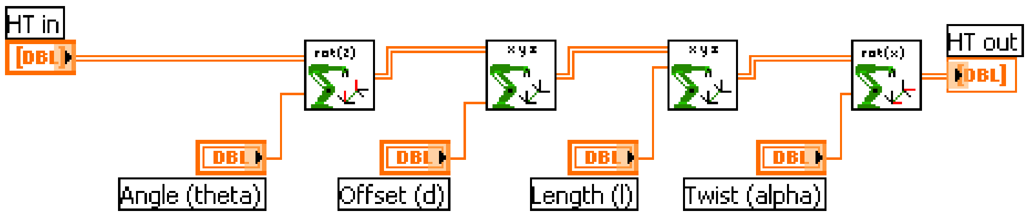
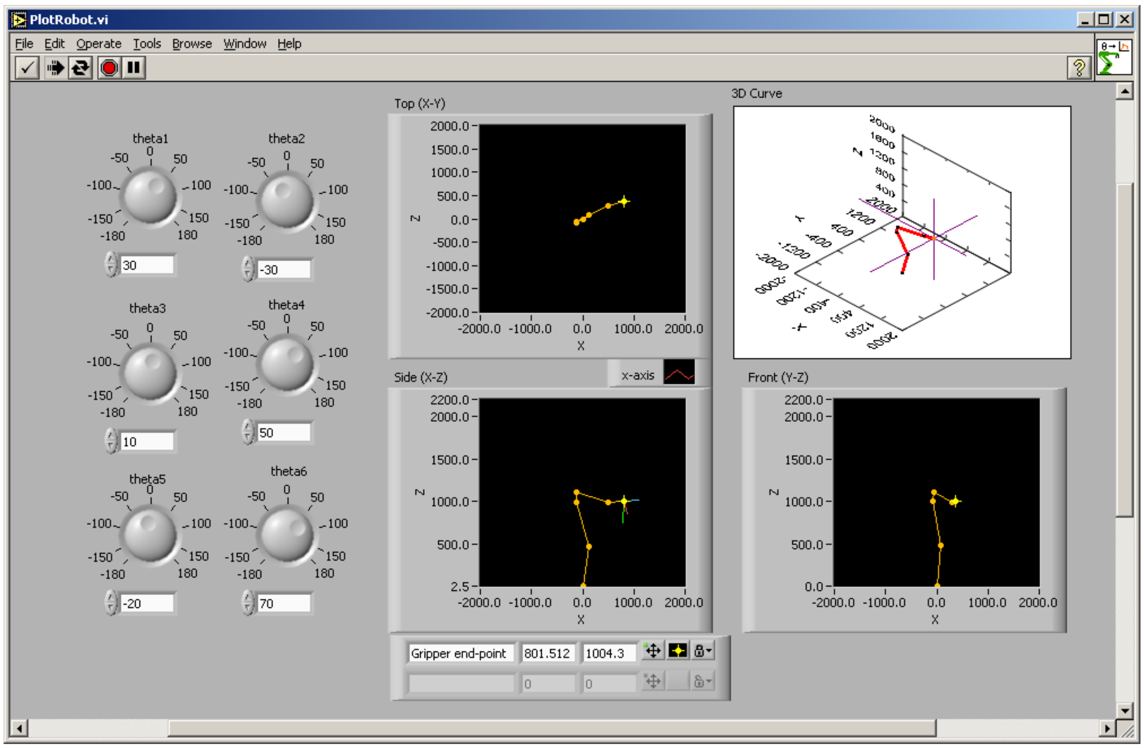


Figure 7.4: *DHTransform.vi* Block Diagram showing the multiplication of the four simple transformations that comprise a DH transformation



theta1	theta2	theta3	theta4	theta5	theta6
30	-30	10	50	-20	70

Gripper end-point	801.512	1004.3
	0	0

Figure 7.5: *PlotRobot.vi* Front Panel showing angle inputs and graphical output from the forward kinematics

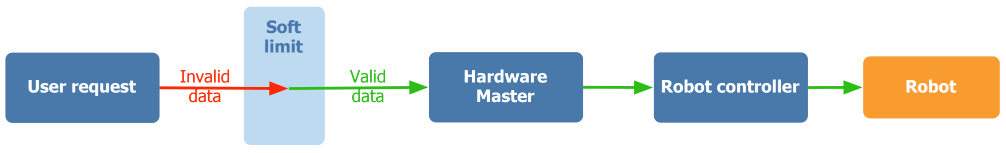
The plotting modules take the HTs describing the links and extract the x,y and z information to be used in the graphs. They also construct vectors to plot the coordinate frame at each joint. Plotting the links and coordinate frames requires some complicated LabVIEW wiring, as an inspection of the Block Diagrams will reveal.

By solving the forward kinematics and plotting the output, there is now a simulation of the Telerobot. This can be used to allow users to check their movements before sending them to the robot, or to implement advanced features such as obstacle avoidance.

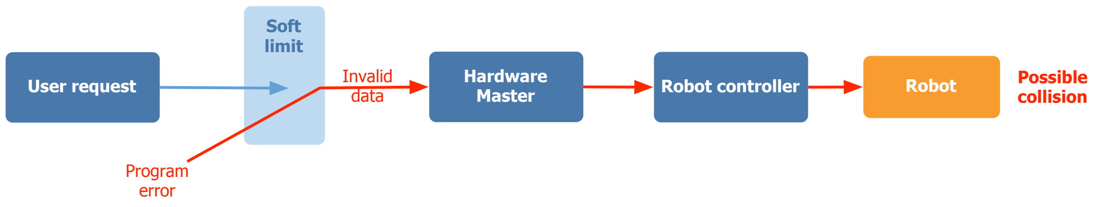
CHAPTER 8

Value checking

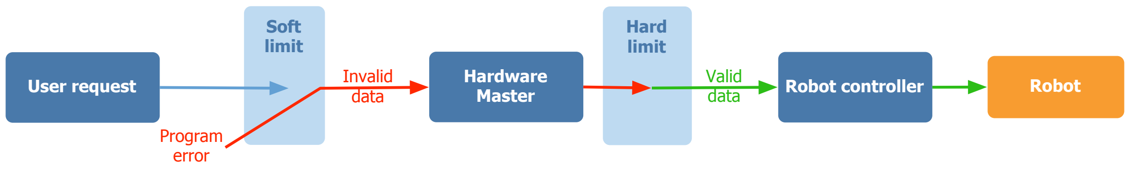
PREVIOUSLY, any values sent to the robot (position and orientation, joint angles or any other data) were not checked to see if they lay between safe limits. The only limits were on the sliders used to change the values on the Front Panel (Figure 9.1). These are what will be referred to as 'soft' limits; that is, they limit the the range of values a user can choose, but if the value is changed some other way (for example, by a bug in the program), the limits cannot do anything about it. This kind of 'soft' checking is shown in Figure 8.1a.



(a) Soft limit



(b) Soft limit with error



(c) Hard limit

Figure 8.1: Value checking

A problem arises when an error in the program (or a malicious user) sends an out-of-range value to the Hardware Master. This will go undetected by the soft checks, because the value was not selected using the interface. This bad value then is sent to the Hardware Master and on to the robot, which will execute the command without checking. This can result in dangerous movements, possibly causing the robot to collide with something and be damaged (Figure 8.1b). This scenario occurred earlier in the year when a mistake in Babbage's programming caused a large value for 'tilt' (a variable specifying the angle to rotate the gripper about a certain axis) to be sent to the robot, causing the robot to collide with and break the table (Figure 8.2).

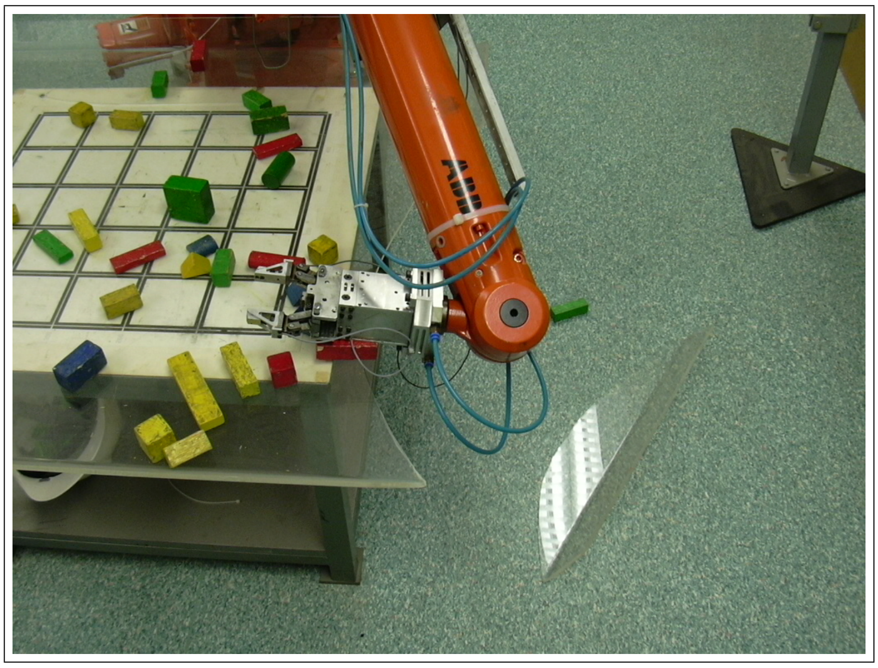


Figure 8.2: The robot and broken table resulting from a bug in the programming

To prevent similar problems in the future, value checking in the Hardware Master was implemented. These modules check all values sent to the robot by either a Remote Client or by the Hardware Master itself. The values are checked to see whether they are within a certain allowable range and, if not, are coerced into that range. Limits have been put in place for both the 'MMS' and 'table' modes, and should be created for any future movement modes. Creation of additional checking VIs has been made easy due to the modular nature of the code.

8.1 Implementation

Before a movement instruction is executed in *RunHardware.vi*, the instruction is passed through the subVI *CheckInstructionSet.vi* (Figure 6.8, blue). This subVI checks the movement mode and passes the instruction to the correct subVI to check the values (Figure 8.3). The values are checked against the limits for that mode, corrected if needs be, and the corrected instruction is output ready for execution (Figure 8.4).

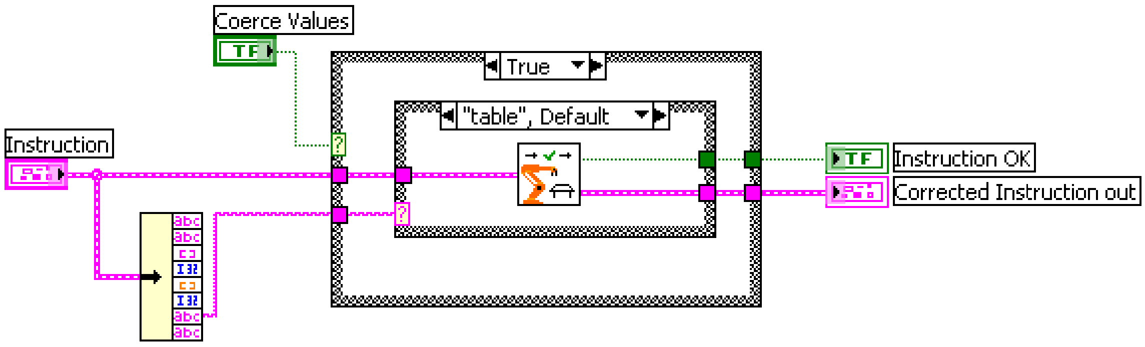


Figure 8.3: *CheckInstructionSet.vi* Block Diagram showing "table" move mode case

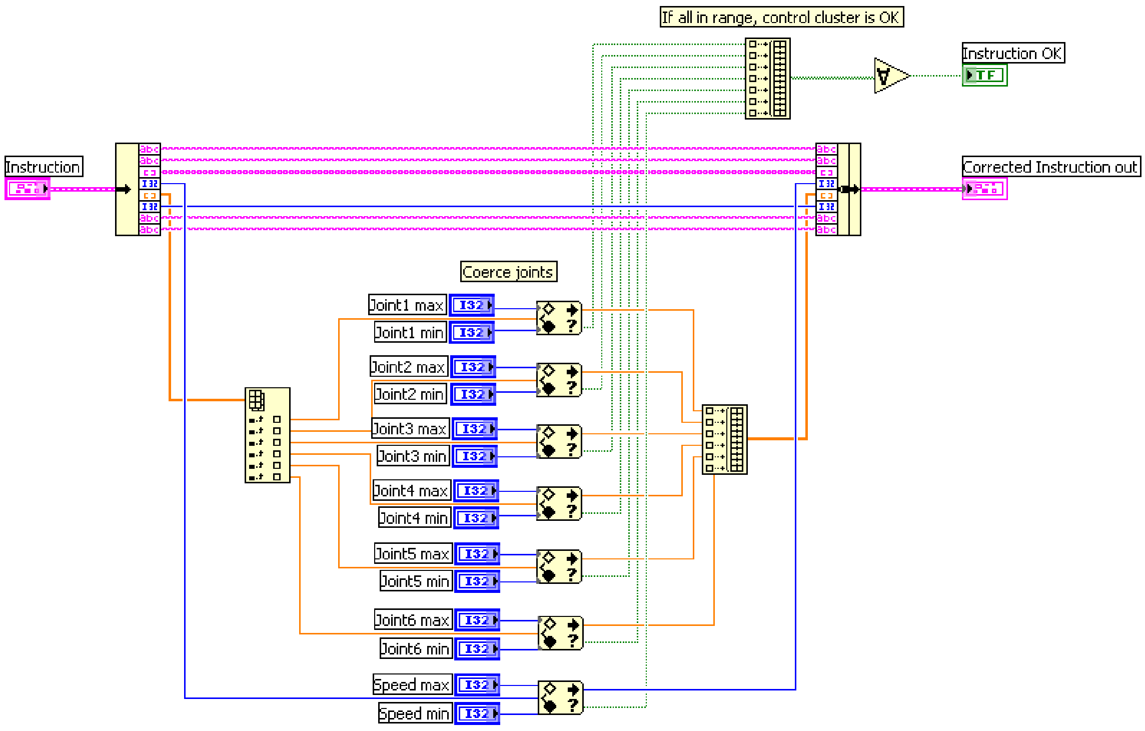


Figure 8.4: *MMSCheck.vi* coerces the joint angle value into specified ranges for the "MMS" movement mode

8.1.1 Limit values

Limit values were chosen for the different movement modes. The physical limits of the joints and the imposed limits for the "MMS" mode are shown in Table 8.1. The limits were chosen to make it impossible for a user to collide with anything in the workspace.

Table 8.1: Physical limits of joint angles and imposed limits in the 'MMS' movement mode

	Physical limits		Imposed limits	
θ_i	Lower	Upper	Lower	Upper
θ_1	-170	170	-90	45
θ_2	-70	70	-65	20
θ_3	$-65-\theta_2$	$70+\theta_2$	-90	0
θ_4	-150	150	-90	90
θ_5	-114.6	114.6	-90	90
θ_6	-300	300	-180	180

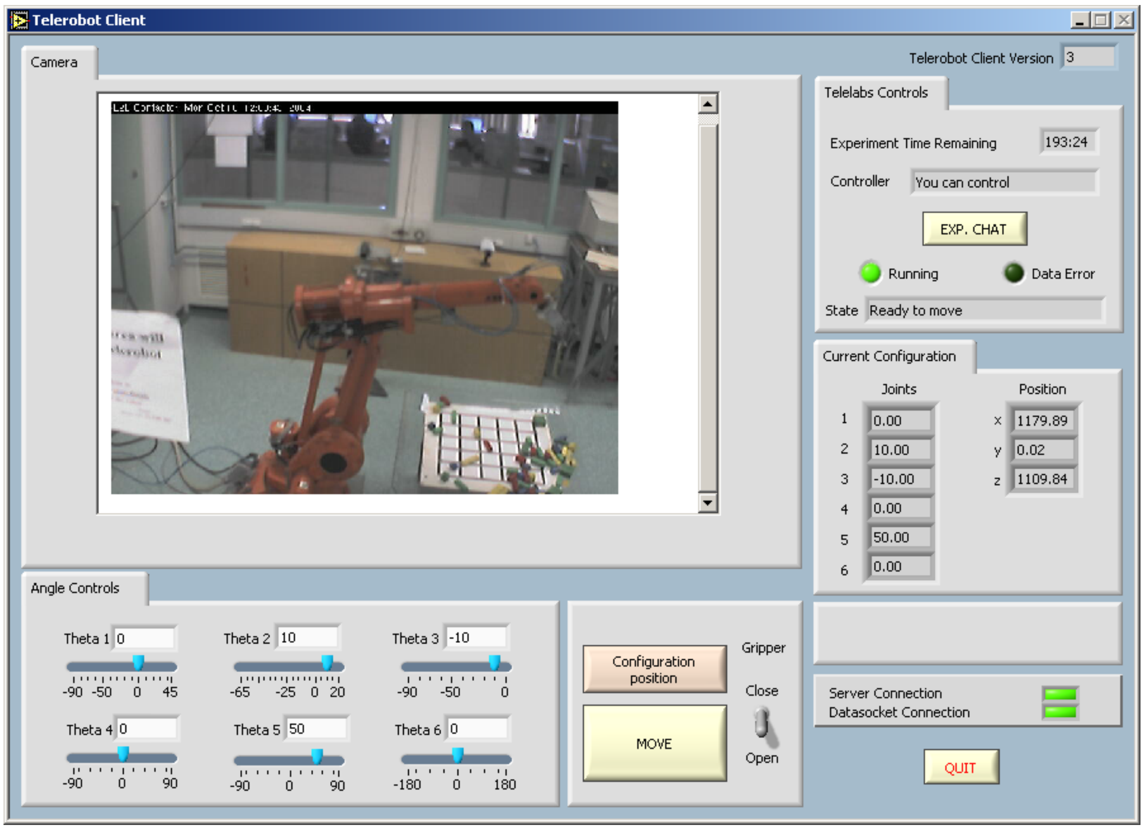
Table 8.2: Imposed limits in 'table' mode (from Babbage [4])

	Limits	
Variable	Lower	Upper
X	-50	550
Y	-50	550
Z	7	450
Spin	-90	90
Tilt	-45	45

CHAPTER 9

Results

As a result of the developments made from this project, the Telerobot has now been used in undergraduate teaching programmes at The University of Western Australia. A Remote Client (Figure 9.1) was built in conjunction with James Newell [57], and is now available for use by students. Although the software was not sufficiently tested to allow it to be relied upon for assessed remote laboratories, it was used in the physical laboratory classes, and also for a revision exercise for the students.



Telerobot Client Version: 3
Experiment Time Remaining: 193:24
Controller: You can control
🟢 Running ● Data Error
State: Ready to move

Joints		Position	
1	0.00	x	1179.89
2	10.00	y	0.02
3	-10.00	z	1109.84
4	0.00		
5	50.00		
6	0.00		

Theta 1: 0
Theta 2: 10
Theta 3: -10
Theta 4: 0
Theta 5: 50
Theta 6: 0

Server Connection 🟩
Datasocket Connection 🟩

Figure 9.1: Remote Client interface (adapted from original by Newell [57])

9.1 Laboratory classes

Computer joint-angle control of the robot was used in Semester 2, 2004, in the physical laboratory classes for Mechanisms & Multibody Systems 319 at UWA. The new features have reduced the duration of laboratory from over 2 hours to 1.5 hours, and has made the laboratory tasks simpler for students to perform.

The computer interface was used for ‘simple’ tasks such as setting the robot back to its zero position (a task that previously took several minutes to do manually, and to a poor degree of accuracy). The computer control was also used for entering joint angles for the forward kinematics section, allowing the students to choose easy values to work with rather than the more complicated values read from the teach-pendant (which often had decimal values). Also, when controlling the robot manually with the joystick, it can be difficult to change one angle without changing others. Entering angles on the computer interface effectively eliminated error from the accidental movement of the joystick in more than one direction at once.

In addition to using the computer control interface in the laboratory next to the robot, some students have used it remotely to complete their laboratory when they lost their laboratory sheets, or recorded erroneous readings. Obtaining new readings was very quick and easy, required no supervision, and allowed students to gather as much data as they wanted.

9.2 Online exercise

An exercise has now been put online to demonstrate the new capabilities of the Telerobot, and to provide students with some revision practise (Appendix E). The exercise is a simple calculation involving the Jacobian of a two-bar mechanism (which the Telerobot can behave as). The students perform the calculation themselves, then log-on to the Telerobot and check their solution by actually moving the robot.

CHAPTER 10

Conclusions & Further work

FOR the first time in its ten-year history, the UWA Telerobot can be used for remote laboratories at The University of Western Australia. The work completed in this project opens the door for many new and exciting possibilities for the Telerobot, and provides a firm foundation upon which many high-powered features can be built. Results so far have been encouraging; the new developments have improved traditional laboratory classes and have provided students with alternative ways to complete their work.

Next year it is expected that the new features will be used to provide a fully-online laboratory for third-year Mechanical and Mechatronics Engineering students. This will allow students to work in their own time, at their own pace, and will provide staff with a new and convenient resource to use. This project improves accuracy and accessibility for future laboratories, and provides a solid basis for further work.

10.1 Further work

This project has created many new possibilities for further work. This section outlines some of the possibilities for future students.

10.1.1 Problems to be corrected

Joint-limit feedback

Currently the user is not alerted when a joint goes beyond its limits. This usually occurs when certain values of θ_2 and θ_3 send the robot past the limit of its 'elbow'. When this happens, the robot stops, and the current joint angles and position are returned. The joint angles returned will obviously not be the same as those entered, because the robot could not get there. It would be helpful for the user to have an indication that the robot could not complete their requested move, and that they need to enter some different angles.

10.1.2 Advanced features

The Telerobot can now be simulated and controlled by joint angles, which opens up new possibilities for advanced features.

Inverse kinematics

The inverse kinematics was started during this project, but not finished due to time constraints. Finishing the inverse kinematics would give the Telerobot a complete set of kinematics modules. As mentioned previously, a subVI was created to solve for the wrist joints using inverse Euler angles. However, this did not give the correct values, and was not tested thoroughly. More thought will need to go into how the Telerobot wrist actually works, and if the inverse Euler angles technique is the best option. Once the wrist angles are found, the remaining angles can be solved using simple trigonometry.

Curve-following

Once the inverse kinematics is solved, the robot could be made to follow curves using Jacobian Motion Rate Control (JMRC). To complete this task effectively, it would be necessary to investigate the effect of any lag from the Hardware Master to the robot (as JMRC relies on small, fast instructions). Another option would be to implement the JMRC code in RAPID, and allow parameters to be changed from the Hardware Master.

Obstacle Avoidance

From the robot simulation, it is now possible to accurately predict where the robot will move when given a set of joint angles. This could be used to provide obstacle avoidance features by ruling out moves that would make the robot collide with a fixture. Currently, the range of angles available to the remote user is very limited to make sure there are no collisions. Obstacle avoidance would allow a far greater range of movement, whilst still protecting the robot. For use with cartesian moves, the inverse kinematics also needs to be solved.

10.1.3 Remote laboratory

Simulation

The simulation using the forward kinematics needs to be integrated to the Remote Client. Thought should be put into how it should be used (for example, as a practise tool or as a stand-alone simulation), and how best to integrate it with the Remote Client.

Additional movements

In order to fully replicate the traditional laboratory, the user should be able to move the robot straight along cartesian axes. This would allow the user to put the robot in any position, then choose to move it along a certain axis. This is different to the table-based control, which specifies a position: the user should not specify a position, but rather a move of a certain length along an axis. This is used in the inverse kinematics part of the laboratory, and would be a useful feature in general. The development would simply require reading the current position from the robot and sending it a new position that was translated along a certain axis.

References

[1] ABB Robotics. *IRB 1400 Industrial Robot*, 2002. URL `http://www.abb.com/robotics`.

[2] K Taylor and J Trevelyan. Australia's telerobot on the web. In *26th International Symposium of Industrial Robotics*, pages 39–44, Singapore, 1995.

[3] Barnaby Dalton. *Techniques for Web Telerobotics*. PhD thesis, School of Mechanical Engineering, University of Western Australia, 2001.

[4] A. Babbage. Software system for controlling web-based laboratory robots. Honours dissertation, School of Mechanical Engineering, The University of Western Australia, 2003.

[5] Rowe Palmer, Vincent Goh, Joel Walker, and Jee Loong Chew. Augmented reality control of the telerobot. Project report for Mechatronics Design 310, School of Mechanical Engineering, The University of Western Australia, 2003.

[6] James Trevelyan and Alexander C. Le Dain. Telelabs specifications. Technical report, 2003.

[7] Scott W. Bonham, Duane L. Deardorff, and Robert J. Beichner. Comparison of student performance using web and paper-based homework in college-level physics. *Journal of Research in Science Teaching*, 40(10):1050–1071, 2003. ISSN 0022-4308.

[8] S. Kekkonen-Moneta and G. B. Moneta. E-learning in Hong Kong: comparing learning outcomes in online multimedia and lecture versions of an introductory computing course. *British Journal of Educational Technology*, 33(4):423–33, Sept. 2002. ISSN 0007-1013.

[9] R.C.T. Cheung. Innovative teaching through the cyber university. In *Advances in Web-Based Learning, First International Conference, ICWL 2002. Proceedings, 17-19 Aug. 2002*, pages 287–99, Dept. of Comput., Hong Kong Polytech. Univ., China, 2002. Springer-Verlag. ISBN 3 540 44041 0.

[10] B.L. Kurtz, D. Parks, and E. Nicholson. Effective internet education: strategies and tools. In *Proceedings of Conference on Frontiers in Education, 6-9 Nov. 2002*, volume vol.2, pages 2–14, Appalachian State Univ., Boone, NC, USA;, 2002. IEEE. ISBN 0 7803 7444 4.

[11] B. Wilkins and J. Barrett. The virtual construction site: a web-based teaching/learning environment in construction technology. *Automation in Construction*, 10(1):169–79, Nov 2000. ISSN 0926-5805.

[12] K. Goldberg, D. Song, Y. Khor, D. Pescovitz, A. Levandowski, J. Himmelstein, J. Shih, A. Ho, E. Paulos, and J. Donath. Collaborative online teleoperation with spatial dynamic voting and a human ’tele-actor’. In *2002 IEEE International Conference on Robotics and Automation, May 11-15 2002*, volume 2, pages 1179–1184, UC Berkeley, Berkeley, CA, United States, 2002. Institute of Electrical and Electronics Engineers Inc.

[13] L. Li, F.-Y. Wang, G. Lai, and F. Wu. Online autonomous guidance system for remote experiments in control engineering. In *System Security and Assurance, Oct 5-8 2003*, volume 3, pages 2444–2449, Systems and Industrial Eng. Dept., University of Arizona, Tucson, AZ, United States, 2003. Institute of Electrical and Electronics Engineers Inc.

[14] B. Wagner and J. Tuttas. Team learning in an online lab. In *31st Annual Frontiers in Education Conference. Impact on Engineering and Science Education. Conference Proceedings, 10-13 Oct. 2001*, volume vol.1, pages 18–22. IEEE, 2001. ISBN 0 7803 6669 7.

[15] S.K. Esche, C. Chassapis, J.W. Nazalewicz, and D.J. Hromin. A scalable system architecture for remote experimentation. *Frontiers in Education, 2002. FIE 2002. 32nd Annual*, 1:T2E–1–T2E–6 vol.1, 2002. ISSN 0190-5848.

[16] Bjarne A. Foss, Tor I. Eikaas, and Morten Hovd. Merging physical experiments back into the learning arena. In *2000 American Control Conference, Jun 28-Jun 30 2000*, volume 4, pages 2944–2948. Institute of Electrical and Electronics Engineers Inc., Piscataway, NJ, USA, 2000.

[17] James Trevelyan. Lessons learned from 10 years experience with remote laboratories. In *Progress Through Partnership—International Conference on Engineering Education and Research*, 2004.

[18] M. Casini, D. Prattichizzo, and A. Vicino. The automatic control telelab. *Control Systems Magazine, IEEE*, 24(3):36–44, 2004. ISSN 0272-1708.

[19] D.Z. Deniz, A. Bulancak, and G. Ozcan. A novel approach to remote laboratories. *Frontiers in Education, 2003. FIE 2003. 33rd Annual*, 1:T3E–8–T3E–12 Vol.1, 2003. ISSN 0190-5848.

[20] M.J. Callaghan, J. Harkin, G. Prasad, T.M. McGinnity, and L.P. Maguire. Integrated architecture for remote experimentation. *Systems, Man and Cybernetics, 2003. IEEE International Conference on*, 5:4822–4827, 2003. ISSN 1062-922X.

[21] A. Ferrero, S. Salicone, C. Bonora, and M. Parmigiani. Remlab: a java-based remote, didactic measurement laboratory. *Instrumentation and Measurement, IEEE Transactions on*, 52(3):710–715, 2003. ISSN 0018-9456.

[22] J.O. Strandman, R. Berntzen, T.A. Fjeldly, T. Ytterdal, and M.S. Shur. Lab-on-web: performing device characterization via internet using modern web technology. In *Proceedings of the Fourth IEEE International Caracas Conference on Devices, Circuits and Systems, 17-19 April 2002*, pages 022–1. IEEE, 2002. ISBN 0 7803 7380 4.

[23] Mo Fu, Christopher Yeo, Yuetong Lin, and Fei-Yue Wang. WAVES: Towards real time laboratory experiments in cyberspace. In *2001 IEEE International Conference on Systems, Man and Cybernetics, Oct 7-10 2001*, volume 5, pages 3470–3474. Institute of Electrical and Electronics Engineers Inc., 2001.

[24] I. Gustavsson. Remote laboratory experiments in electrical engineering education. In *Proceedings of the Fourth IEEE International Caracas Conference on Devices, Circuits and Systems, 17-19 April 2002*, pages 025–1, Dept. of Telecommun. & Signal Process., Blekinge Inst. of Technol., Ronneby, Sweden, 2002. IEEE. ISBN 0 7803 7380 4.

[25] E.C. Chung and A.H. Titus. Development of a remotely accessible integrated circuit test facility based on telepresence. In *17th IEEE Instrumentation and Measurement Technology Conference, 1-4 May 2000*, volume vol.3, pages 1591–5, Dept. of Electr. Eng., Rochester Inst. of Technol., NY, USA, 2000. IEEE. ISBN 0 7803 5890 2.

[26] Hong Shen, M.S. Shur, T.A. Fjeldly, and K. Smith. Low-cost modules for remote engineering education: performing laboratory experiments over the internet. *Frontiers in Education Conference, 2000. FIE 2000. 30th Annual*, 1:T1D/7 vol.1, 2000.

[27] K.K. Tan, T.H. Lee, and F.M. Leu. Development of a distant laboratory using labview. *International Journal of Engineering Education*, 16(3):273–82, 2000. ISSN 0949-149X.

[28] Kin Yeung and Jie Huang. Development of a remote-access laboratory: A dc motor control experiment. *Computers in Industry*, 52(3):305–311, 2003. ISSN 0166-3615.

[29] R.M. Parkin, C.A. Czarnecki, R. Safaric, and D.W. Calkin. A pid servo control system experiment conducted remotely via internet. *Mechatronics*, 12(6):833–43, Jul 2002. ISSN 0957-4158.

[30] A. Bauchspiess, B. Guimaraes, and H.L. Gosmann. Remote experimentation on a three coupled water reservoirs. *Industrial Electronics, 2003. ISIE '03. 2003 IEEE International Symposium on*, 1:572–577 vol. 1, 2003.

[31] M.L. Corradini, G. Ippoliti, T. Leo, and S. Longhi. An internet based laboratory for control education. In *Proceedings of 40th Conference on Decision and Control, 4-7 Dec. 2001*, volume vol.3, pages 2833–8, Dipt. di Ingegneria dell'Innovazione, Lecce Univ., Italy;, 2001. IEEE. ISBN 0 7803 7061 9.

[32] D. Hinkson, C. Marshall, and S. Robinson. Design and development of a user interface to remotely control a radio telescope using virtual instruments. In *Proceedings IEEE SoutheastCon 2002, 5-7 April 2002*, pages 279–82, South Carolina State Univ., Orangeburg, SC, USA, 2002. IEEE. ISBN 0 7803 7252 2. Also available on CD-ROM in PDF format.

[33] R. Marin, P.J. Sanz, P. Nebot, and R. Esteller. Multirobot internet-based architecture for telemanipulation: experimental validation. In *SMC '03 Conference Proceedings. 2003 IEEE International Conference on Systems, Man and Cybernetics, 5-8 Oct. 2003*, volume vol.4, pages 3565–70. IEEE, 2003. ISBN 0 7803 7952 7.

[34] G. Lowe and A. Cambrell. Web system for control of mechatronic devices. In *ICARV 2002: The Seventh International Conference on Control, Automation, Robotics and Vision, 2-5 Dec. 2002*, volume vol.3, pages 1464–9. Nanyang Technological Univ, 2002. ISBN 981 04 8364 3.

[35] P. Skrzypczynski. A Java based system for navigation and tele-operation of a mobile robot. In *First IFAC-Conference on Telematics Applications in Automation and Robotics TA 2001, 24-26 July 2001*, pages 309–14, Dept. of Control, Robotics, & Comput. Sci., Tech. Univ. Poznan, Poland, 2001. Elsevier Sci. ISBN 0 08 043856 3.

[36] Alicia Webb. Design of a thermofluids pump laboratory. Honours dissertation, School of Mechanical Engineering, The University of Western Australia, 2003.

[37] H.E. Motuk, A.M. Erkmen, and I. Erkmen. Student performance evaluation in web based access to robot supported laboratories. In *2003 IEEE International Conference on Robotics and Automation, Sep 14-19 2003*, volume 3, pages 4408–4413, Middle East Technical University, Department of Electrical Engineering, 06531 Ankara, Turkey, 2003. Institute of Electrical and Electronics Engineers Inc.

[38] R. Morgan and K.O. Jones. The use of simulation software to enhance student understanding. In *IEE International Symposium Engineering Education: Innovations in Teaching, Learning and Assessment, 4-5 Jan. 2001*, volume 2, pages 33–1, Sch. of Eng., Liverpool John Moores Univ., UK, 2001. IEE.

[39] Semahat S. Demir. Icell: An interactive web resource for simulation-based teaching and learning in electrophysiology training. In *A New Beginning for Human Health: Proceedings of the 25th Annual International Conference of the IEEE Engineering in Medicine and Biology Society, Sep 17-21 2003*, volume 4, pages 3501–3504, Joint Biomedical Engineering Program, Univ. of Memphis/Univ. of Tennessee, 330 Engineering Technology Building, Memphis, TN 38152-3210, United States, 2003. Institute of Electrical and Electronics Engineers Inc.

[40] Alfredo del Rio and Juan Jose Rodriguez Andina. UVI51: A simulation tool for teaching/learning the 8051 microcontroller. In *30th Annual Frontiers in Education Conference—Building on a Century of Progress in Engineering Education, Oct 18-Oct 21 2000*, volume 2, pages 4–11–4–16, Univ of Vigo, Vigo, Spain, 2000. Institute of Electrical and Electronics Engineers Inc., Piscataway, NJ, USA.

[41] Saad Al-Jibouri and Michael Mawdesley. A simulation game for teaching project control in construction. In *Proceedings of the 10th ISPE International Conference on Concurrent Engineers: Research and Applications, Enhanced Interporable Systems, Jul 26-30 2003*, pages 1229–1234, Construction Process Management, University of Twente, Enschede, Netherlands, 2003. A.A. Balkema Publishers. ISBN 905809622X.

[42] M. Brian Blake. A student-enacted simulation approach to software engineering education. *IEEE Transactions on Education*, 46(1):124–132, 2003. ISSN 0018-9359.

[43] C. Fernandez, O. Garcia, J.A. Cobos, and J. Uceda. Self-learning laboratory set-up for teaching power electronics combining simulations and measurements. In *2002 IEEE 33rd Annual Power Electronics Specialists Conference (PESC), Jun 23-27 2002*, volume 2, pages 449–454, UPM, DIE, 28006 Madrid, Spain, 2002. Institute of Electrical and Electronics Engineers Inc.

[44] A. Luchetta, S. Manetti, and A. Reatti. SAPWIN - a symbolic simulator as a support in electrical engineering education. *IEEE Transactions on Education*, 44(2):9 pp., May 2001. ISSN 0018-9359.

[45] M. Hoorfar, H. Najjaran, and W.L. Cleghorn. Simulation and animation of mechanical systems to enhance student learning. *Computers in Education Journal*, 13(1): 39–44, Jan 2003. ISSN 1069-3769.

[46] T. Kenjo, T. Kikuchi, and M. Kubo. Developing educational software for mechatronics simulation. *IEEE Transactions on Education*, 44(2):29 pp., May 2001. ISSN 0018-9359.

[47] N.Y. Chong, T. Kotoku, K. Ohba, K. Komoriya, F. Ozaki, H. Hashimoto, J. Oaki, K. Maeda, N. Matsuhira, and K. Tanie. Development of a multi-telerobot system for remote collaboration. In *2000 IEEE/RSJ International Conference on Intelligent Robots and Systems, Oct 31-Nov 5 2000*, volume 2, pages 1002–1007. Institute of Electrical and Electronics Engineers Inc., 2000.

[48] R. Safaric, I. Hedrih, R. Klobucar, and B. Sorgo. Remote controlled robot arm. *Industrial Technology, 2003 IEEE International Conference on*, 2:1202–1207 Vol.2, 2003.

[49] Roman Kuc, Edward W. Jackson, and Alexander Kuc. Teaching introductory autonomous robotics with JavaScript simulations and actual robots. *IEEE Transactions on Education*, 47(1):74–82, 2004. ISSN 0018-9359.

[50] Raul Marin, Pedro J. Sanz, and Angel P. Del Pobil. The UJI online robot: An education and training experience. *Autonomous Robots*, 15(3):283–297, 2003. ISSN 0929-5593.

[51] P. Da Silva and G. Knabe. Labhouse: System simulation and emulation within boiler development. *Building Services Engineering Research and Technology*, 24(4):281–287, 2004. ISSN 0143-6244.

[52] Jin Shen, Gang Zheng, Guo-Qiang Sun, and Song-Lin Zhuang. Computer simulation of nanometer particle dynamic light scattering by labview. *Guangdian Gongcheng/Opto-Electronic Engineering*, 29(SUPPL):43–45, 2002. ISSN 1003-501X.

[53] R. Bockstaele and R. Baets. Simulation of interchip interconnections based on resonant cavity leds, plastic optical fibres and cmos interface circuits. In *2000 Digest of the LEOS Summer Topical Meetings. Electronic-Enhanced Optics. Optical Sensing in Semiconductor Manufacturing. Electro-Optics in Space. Broadband Optical Networks, 24-28 July 2000*, pages 39–40, Dept. of Inf. Technol., Ghent Univ., Belgium, 2000. IEEE. ISBN 0 7803 6252 7.

[54] Alfred C.H. Tan, Patrick S.K. Chua, and G.H. Lim. Fault diagnosis of water hydraulic actuators under some simulated faults. *Journal of Materials Processing Technology*, 138(1-3):123–130, 2003. ISSN 0924-0136.

[55] Kelvin R. Aaron, Noreen L. Foster, Dannielle P. Hazel, and A. M. Hasanul Basher. Closed-loop position control system using labview. In *IEEE SoutheastCon 2002, Apr 5-7 2002*, pages 283–286, South Carolina State University, Orangeburg, SC 29117, United States, 2002. Institute of Electrical and Electronics Engineers Inc.

[56] Xunzhang Wang, G.G.L. Seet, M.W.S. Lau, E. Low, and K.C. Tan. Exploiting force feedback in pilot training and control of an underwater robotics vehicle: an implementation in labview. In *OCEANS 2000 MTS/IEEE Conference and Exhibition. Conference Proceedings, 11-14 Sept. 2000*, volume vol.3, pages 2037–42. IEEE, 2000. ISBN 0 7803 6551 8.

[57] James Newell. Implementing a remote laboratory for the Telerobot. Honours dissertation, School of Mechanical Engineering, The University of Western Australia, 2004.

[58] John J. Craig. *Introduction to Robotics: Mechanics and Control*. Pearson Education, 2005.

[59] Henry W. Stone. *Kinematic Modeling, Identification, and Control of Robotic Manipulators*. Kluwer Academic Publishers, 1987.

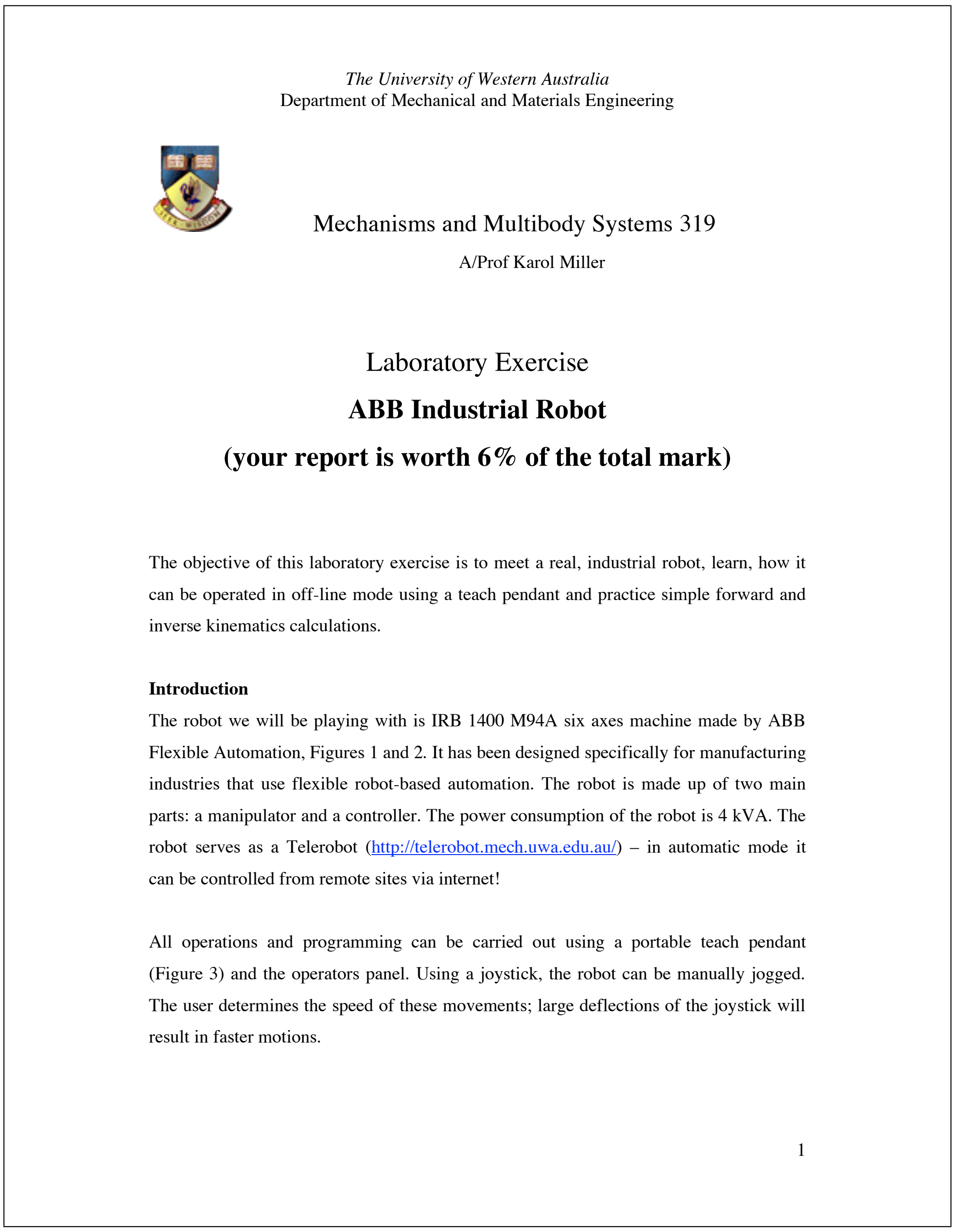
[60] A J Koivo. *Control of robotic manipulators*. John Wiley & Sons, Inc, 1989.

[61] Richard D. Klafter, Thomas A. Chmielewski, and Michael Negin. *Robotic Engineering: An Integrated Approach*. Prentice Hall, 1989.

Appendices

APPENDIX A

Original MMS 319 Lab sheet (Pre-2004)



The University of Western Australia
Department of Mechanical and Materials Engineering

Mechanisms and Multibody Systems 319

A/Prof Karol Miller

Laboratory Exercise

ABB Industrial Robot

(your report is worth 6% of the total mark)

The objective of this laboratory exercise is to meet a real, industrial robot, learn, how it can be operated in off-line mode using a teach pendant and practice simple forward and inverse kinematics calculations.

Introduction

The robot we will be playing with is IRB 1400 M94A six axes machine made by ABB Flexible Automation, Figures 1 and 2. It has been designed specifically for manufacturing industries that use flexible robot-based automation. The robot is made up of two main parts: a manipulator and a controller. The power consumption of the robot is 4 kVA. The robot serves as a Telerobot (http://telerobot.mech.uwa.edu.au/) – in automatic mode it can be controlled from remote sites via internet!

All operations and programming can be carried out using a portable teach pendant (Figure 3) and the operators panel. Using a joystick, the robot can be manually jogged. The user determines the speed of these movements; large deflections of the joystick will result in faster motions.

1

The University of Western Australia
Department of Mechanical and Materials Engineering

Even though the manipulator weights 225 kg it can move payloads of only up to 5kg. This is common to serial manipulators, which have to carry not only the payload but also motors, gears and other components required to move each joint.

Figures 4 and 5 provide information on robot dimensions. The floor mounted robot differs from the suspended one only in the length of the second link (450mm for suspended robot, Figure 5). However the length of the second link of the floor mounted robot can be easily worked out by noting that the total height of the floor mounted version is 1310 mm (Fig. 4) and that of suspended one – 1160 mm (Fig. 5). So that the length of link two R2=(1310-1160)+450=600 [mm]. The length of a gripper attached to the robot is 225 mm.
Figure 6 shows robot's workspace

In order to program and execute robot motion the user has to use appropriate coordinate systems. Figure 7 shows six coordinate systems used by ABB robot controller:
The world coordinate system defines the reference to the floor
The base coordinate system is referenced to the base mounting surface of the robot
The tool coordinate system specifies the tool's centre point and orientation
The user coordinate system specifies the position of a fixture
The object coordinate system specifies how a workpiece is positioned in a fixture

In off-line mode the robot can be moved in a variety of ways using the teach pendant (Figure 3). Each axis (actuated joint) can be moved separately. Alternatively, the tip of the robot can be moved along x, y or z axis of the "world" coordinate frame. It is also possible to rotate the robot around "world" coordinate frame axes having the tip of the robot as a fixed point of the motion.

Laboratory work

❖ What conventions are used to measure joint angles and which direction of rotation is positive?

The University of Western Australia
Department of Mechanical and Materials Engineering

To answer this question you need to investigate the motion of the robot in joint space. Start with moving all joints to their respective zero positions. Record the x,y,z coordinates of the robot tip – you will need this information later. Beginning with joint one, move each motorised joint separately by about 20 degrees in both directions. Read joint angles from the teach pendant. This will allow you to find out, which direction of rotation is positive. Pay special attention to whether the motion in one joint affects other joint angles. Record carefully with respect to which axes the angles are measured and positive directions of rotations. You will need this information to conduct kinematics calculations.

❖ The position of the tip is given in "world" coordinate frame. Investigate how the axes are oriented and where the origin of this frame is.

To answer this question you need to investigate the motion of the robot in task space. The end effector can be moved in straight lines along axes of world coordinate frame. This mode can be used to verify what the axes' directions are. We expect the "world" coordinate frame to have axes x and y in horizontal plane and axis z vertical. Moreover, we expect axis x to correspond to angle in the first joint equal to zero degrees. To confirm this set all joint angles to zero and move joint #2. If this results in no change of y coordinate you can safely confirm your expectation. The directions of axes y and z can be confirmed by visual inspection by moving a tip along those axes.
At this stage we are not sure where the centre of the "world" coordinate frame is. To investigate that set all joint angles to zero and read the position of the tip. Knowing the dimensions of the robot (Figures 4, 5) the centre of the "world" coordinate system can be easily calculated.

❖ Direct kinematics of three-link planar robot.

By setting joint angles 1,4 and 6 to zero we end up with three link planar manipulator working in x-z plane. Collect measurements of robot tip positions and corresponding joint 2,3 and 5 angles. Using forward kinematics equations for a planar robot (x,y,z as functions of theta2,theta3, theta4) and measured joint angles compute tip positions. Compare them to the ones read from the teaching pendant. Repeat the procedure for three points.

The University of Western Australia
Department of Mechanical and Materials Engineering

- ❖ Inverse kinematics of two-link planar robot.

By setting joint angles 1,4,5 and 6 to zero we end up with two-link planar manipulator working in x-z plane. Perform a straight line motion along z axis. Record x,z coordinates and theta2, theta3 angles of start and end point of this trajectory. Using inverse kinematics equations and robot dimensions compute angles theta2 and theta3 for these two points. Compare the results to what you have recorded from the teach pendant.

- ❖ It is also possible to rotate the robot around x, y or z axis having robot's tip as a fixed point of the motion. Perform these rotations and pay attention to quite complicated motions of the robot.

Report:

One report per group is required. High standard of submission is expected.

Your report should contain the following information:

- kinematic diagram of the robot with indication in which directions joint angles are measured and with respect to which axes
- calculation of the position of the centre of the "world" coordinate frame and the resulting diagram of the robot together with the "world" coordinate frame
- equations for forward kinematics for three-link, planar manipulator, calculation of tip position from known joint angles and the comparison to the values recorded during the exercise
- equations for inverse kinematics for two-link planar manipulator, calculation of joint angles from known tip positions (beginning and end of a vertical trajectory) and the comparison to the values recorded during the exercise
- conclusions

Reference:

ABB Robotics Products, Product Specification IRB 1400, Issue M94A

J J Craig, "Introduction to Robotics, Mechanics and Control", Addison Wesley, 1989.

4

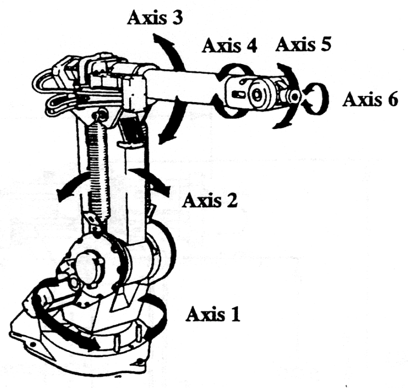


Figure 1 The IRB 1400 manipulator has 6-axes.

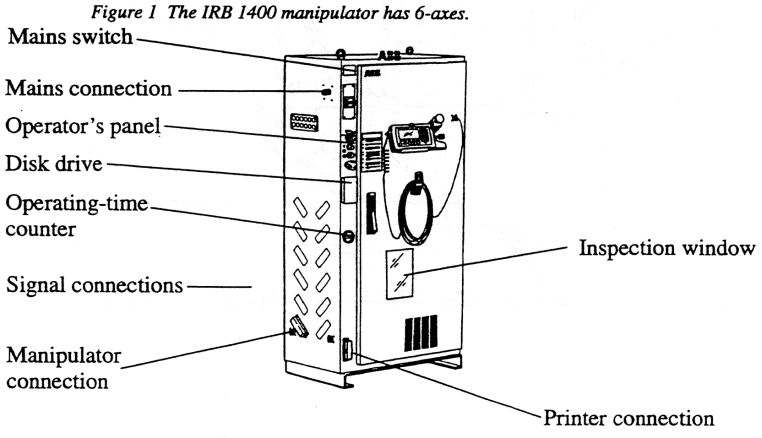


Figure 2 The controller is specifically designed to control robots, which means that optimal performance and functionality is achieved.

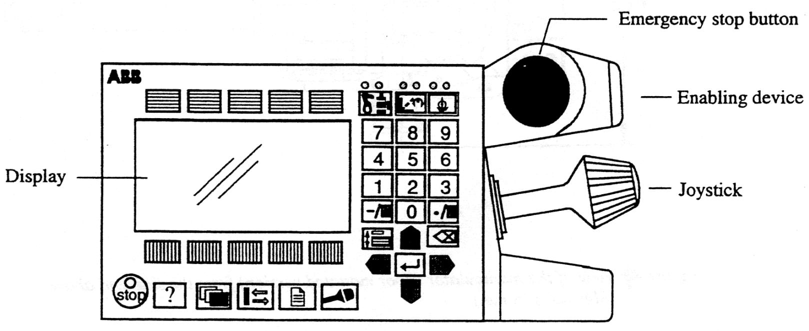


Figure 3 The teach pendant is equipped with a large display, which displays prompts, information, error messages and other information in plain English.

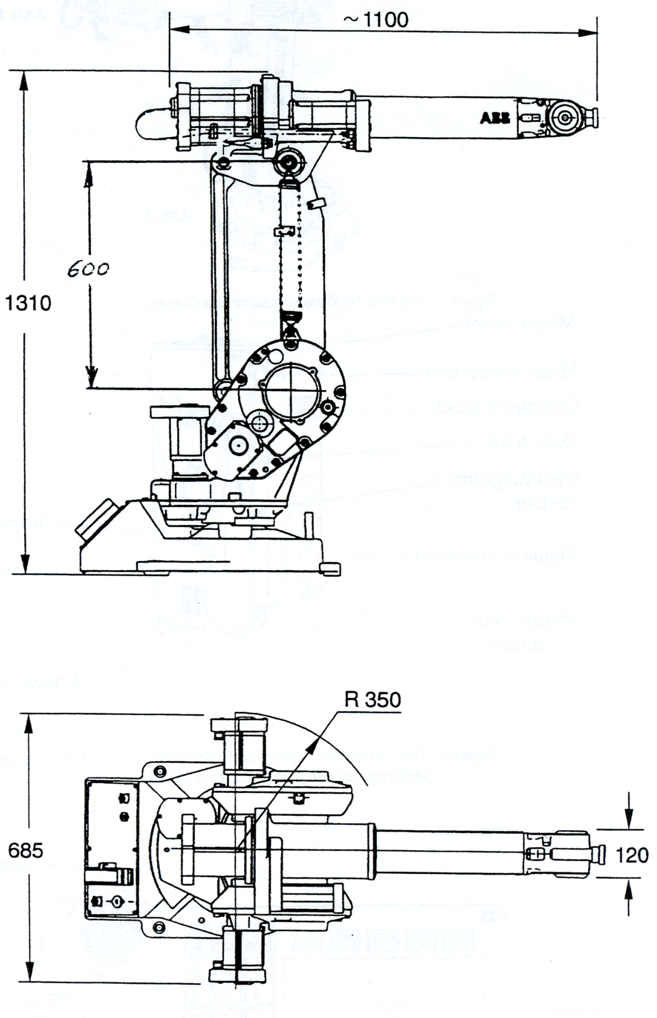


Figure 4 View of the manipulator (floor mounted version) from the side and above (Measures in mm)

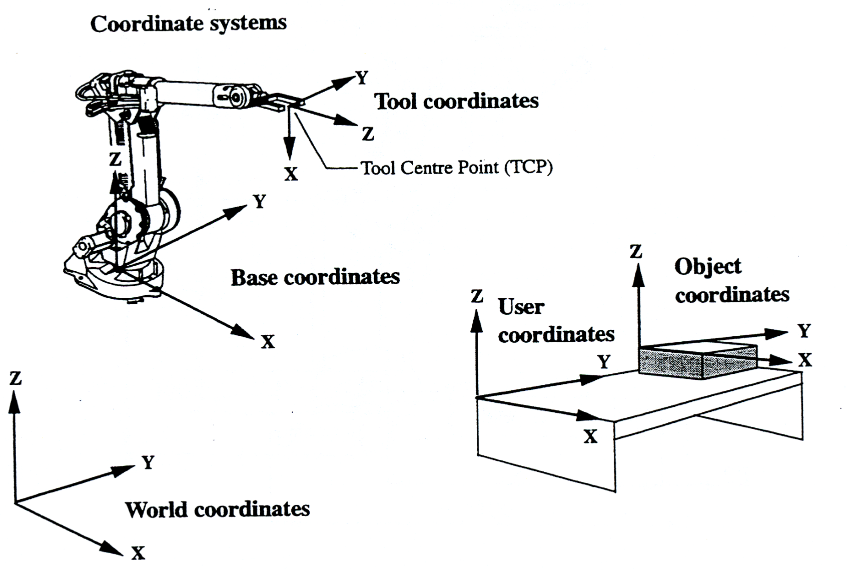


Figure 7 The coordinate systems, used to make jogging and off-line programming easier.

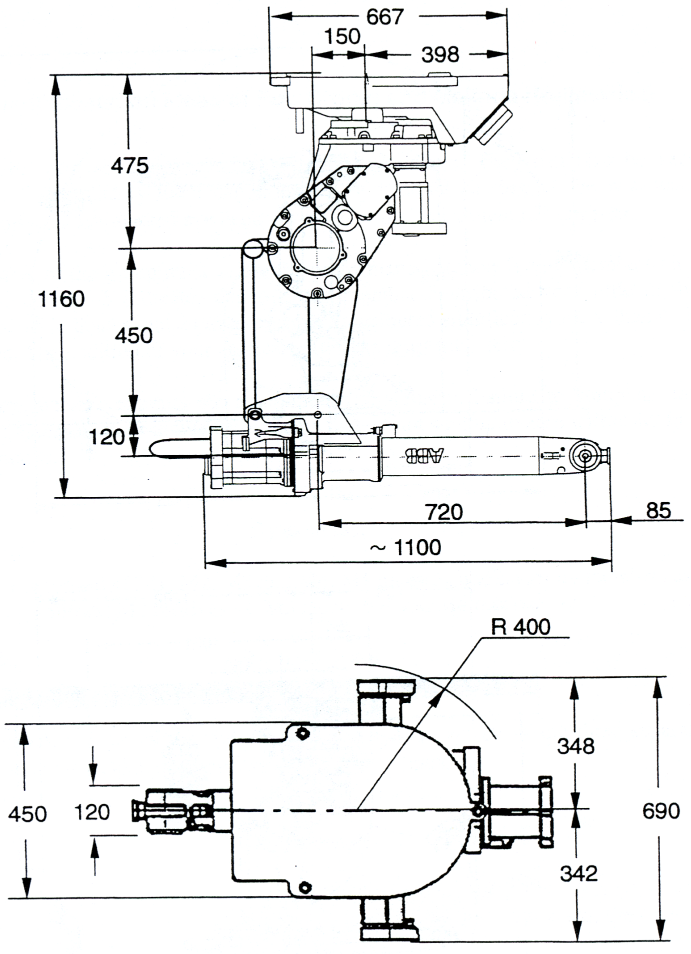


Figure 5 View of the manipulator (inverted mounted version) from the side and above (Measures in mm

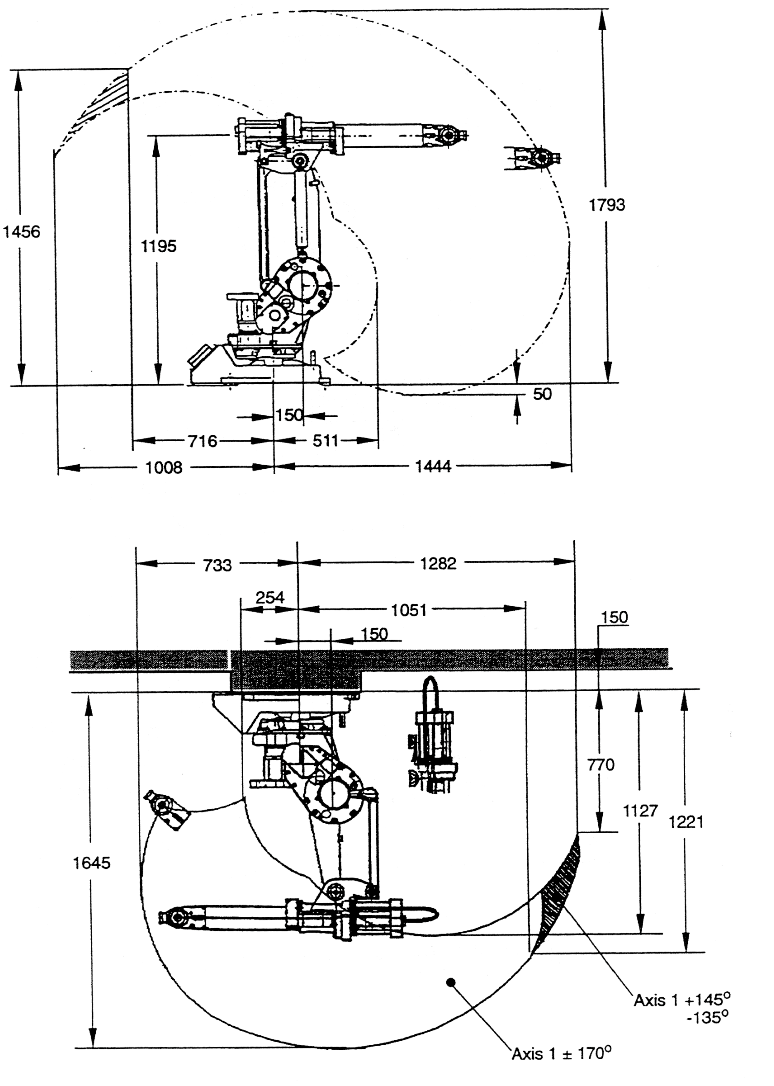
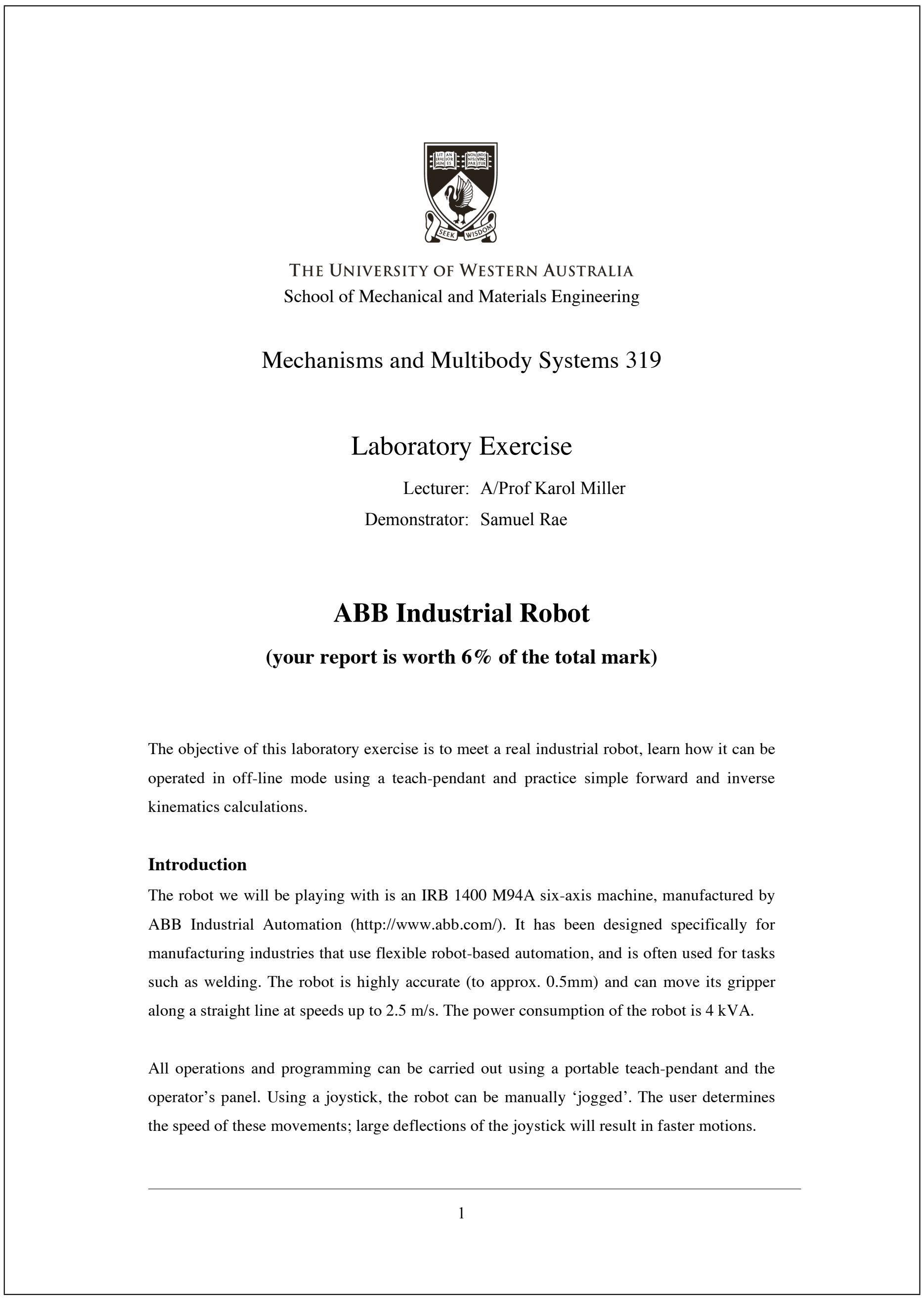


Figure 6 Working space of IRB 1400

APPENDIX B

New MMS 319 Laboratory Sheet (2004)



THE UNIVERSITY OF WESTERN AUSTRALIA
School of Mechanical and Materials Engineering

Mechanisms and Multibody Systems 319

Laboratory Exercise

Lecturer: A/Prof Karol Miller
Demonstrator: Samuel Rae

ABB Industrial Robot

(your report is worth 6% of the total mark)

The objective of this laboratory exercise is to meet a real industrial robot, learn how it can be operated in off-line mode using a teach-pendant and practice simple forward and inverse kinematics calculations.

Introduction

The robot we will be playing with is an IRB 1400 M94A six-axis machine, manufactured by ABB Industrial Automation (http://www.abb.com/). It has been designed specifically for manufacturing industries that use flexible robot-based automation, and is often used for tasks such as welding. The robot is highly accurate (to approx. 0.5mm) and can move its gripper along a straight line at speeds up to 2.5 m/s. The power consumption of the robot is 4 kVA.

All operations and programming can be carried out using a portable teach-pendant and the operator's panel. Using a joystick, the robot can be manually 'jogged'. The user determines the speed of these movements; large deflections of the joystick will result in faster motions.

1

Even though the manipulator weighs 225 kg it can move payloads of only up to 5kg. This is common to serial manipulators, which have to carry not only the payload but also motors, gears and other components required to move each joint.

The robot serves as a telerobot (http://telerobot.mech.uwa.edu.au/), and can be controlled over the internet. It has been used extensively by people around the world, and was the first robot of its kind to be controlled over the internet.

The Teach-Pendant

The device used to program and control the robot is called the Teach-Pendant (Figure 1).

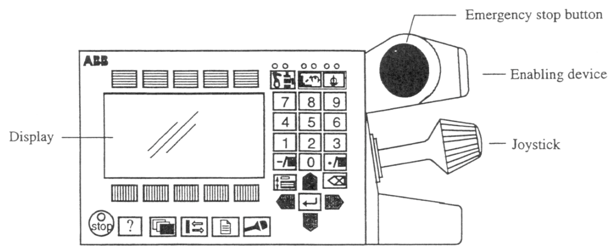


Figure 1—The teach-pendant

The robot can be moved by the teach-pendant in a number of ways. The gripper can be translated linearly along the axes of a coordinate frame or reoriented (ie. rotated) about the axes of a coordinate frame, and the joints (axes 1–6) can be moved individually (Figure 2).

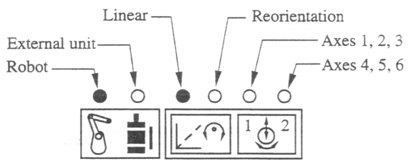


Figure 2—Teach-pendant movement modes

The robot is made to move in these modes by the joystick on the teach-pendant. The joystick can be moved in three ways: vertically, horizontally and rotated about its axis (Figure 3).

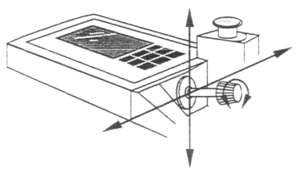


Figure 3—Motions of the teach-pendant joystick

Dimensions

Figure 4 shows the dimensions you will require for this lab. The length of the gripper attached to the robot is 225 mm.

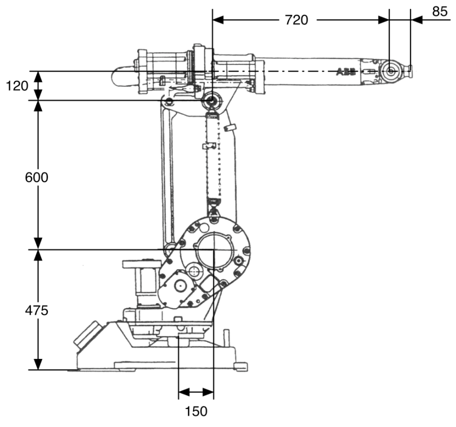


Figure 4—Robot with dimensions

Coordinate Frames

In order to program and execute robot motion the user has to use appropriate coordinate frames. Figure 5 shows the five coordinate frames used by ABB robot controller:

1. The **Base Coordinate Frame** is the reference from the surface on which the robot base is mounted
2. The **World Coordinate Frame** defines the reference to the floor (in our case this is coincident with the Base Coordinate Frame)
3. The **Tool Coordinate Frame** specifies the tool's centre point and orientation
4. The **User Coordinate Frame** specifies the position of a fixture
5. The **Object Coordinate Frame** specifies how a workpiece is positioned on a fixture

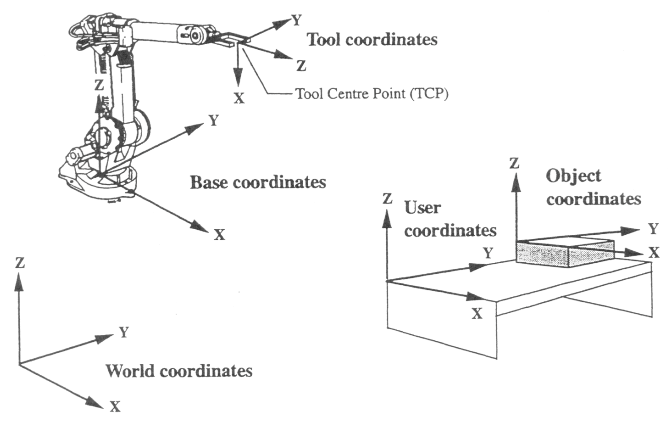
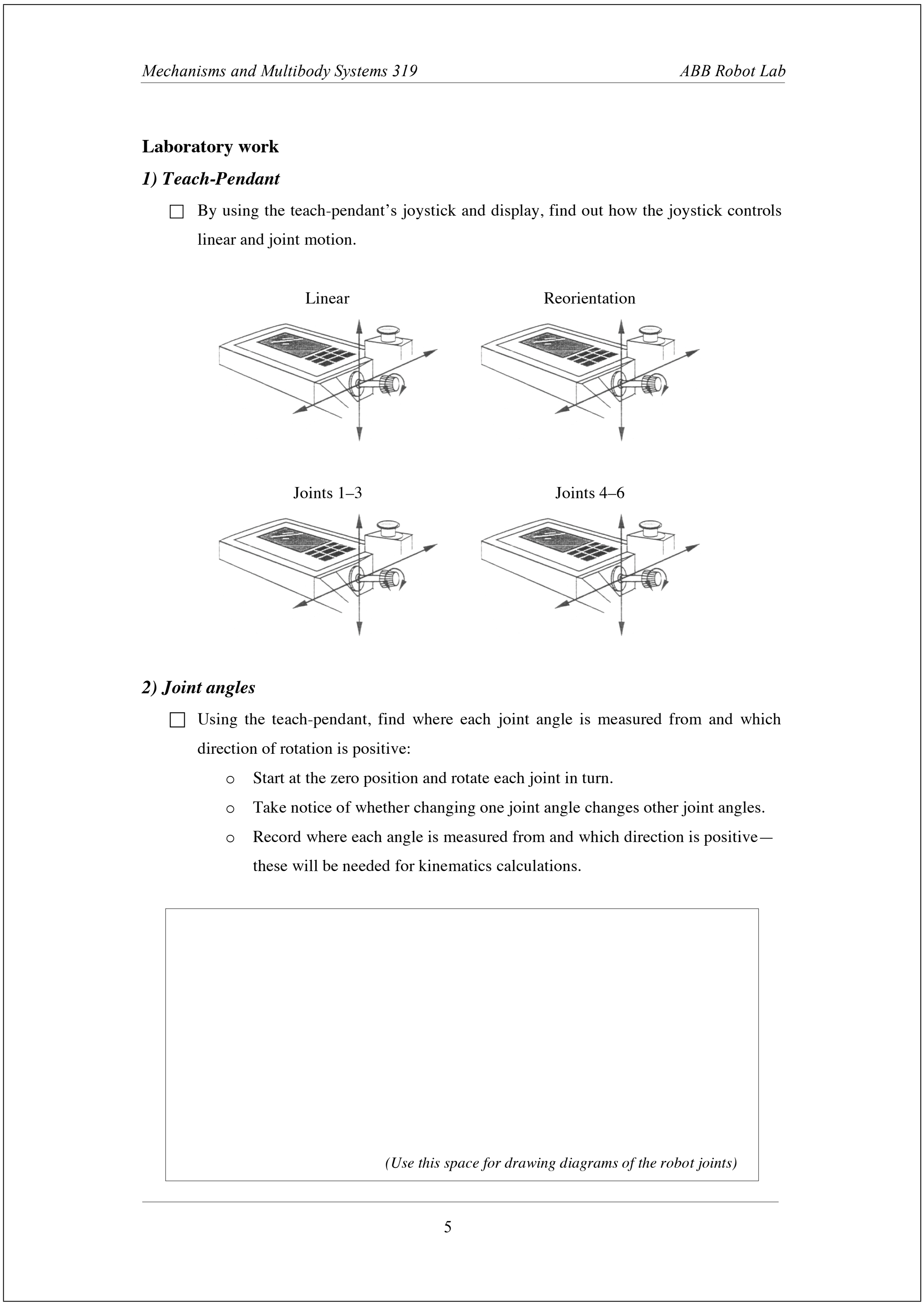


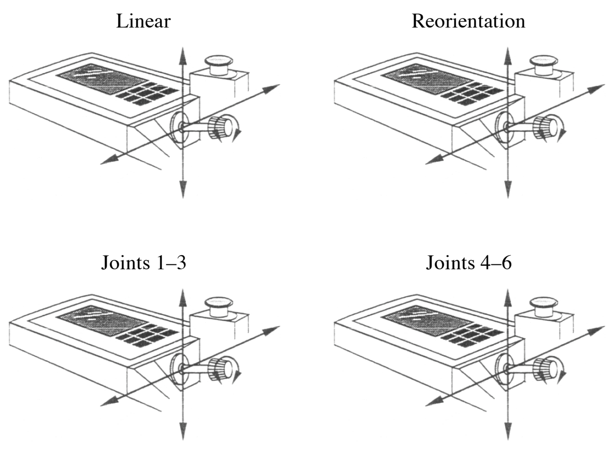
Figure 5—Robot coordinate frames



Laboratory work

1) Teach-Pendant

☐ By using the teach-pendant's joystick and display, find out how the joystick controls linear and joint motion.



2) Joint angles

☐ Using the teach-pendant, find where each joint angle is measured from and which direction of rotation is positive:

- Start at the zero position and rotate each joint in turn.
- Take notice of whether changing one joint angle changes other joint angles.
- Record where each angle is measured from and which direction is positive—these will be needed for kinematics calculations.

(Use this space for drawing diagrams of the robot joints)

3) Coordinate frames

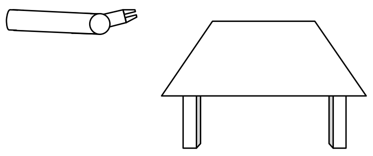
By default the teach-pendant gives the position of the robot in the User Coordinate Frame (UCF). We want to how this frame is positioned and oriented relative to the Base Coordinate Frame (BCF). The BCF is at floor-level with the z-axis along the axis of joint 1, and the x-axis pointing straight forwards.

☐ Find the origin of the UCF and the directions of its axes relative to the BCF:
- o Put the robot into its zero position.
- o Record the position of the tip in UCF (from the teach-pendant).

	X	Y	Z
Tip position in UCF			

- o Calculate the position of the tip in BCF (from the dimensions of the robot). *(Report)*
- o You should now be able to calculate the position of the origin of the UCF in terms of the BCF. *(Report)*

☐ Find the direction of the axes of the UCF (use the teach-pendant joystick and display).



4) Direct kinematics of a three-link planar robot

Direct (or forward) kinematics involves finding the position of the gripper (or end-effector) of the robot given the joint angles.

☐ Collect joint angle and position data:
- o Set joint angles 1, 4 and 6 to zero *(by doing this we end up with three-link planar manipulator working in the x-z plane).*
- o Move joints 2, 3 and 5 to arbitrary angles, and record the angles and the robot tip position. Do this for three points.

θ_2	θ_3	θ_5	X	Z

☐ Using forward kinematics equations for a planar robot (x, y and z as functions of θ_2, θ_3 and θ_4) and the measured joint angles, calculate the theoretical tip positions. *(Report)*

☐ Compare the calculated positions to those read from the teach-pendant. *(Report)*

5) Inverse kinematics of a two-link planar robot

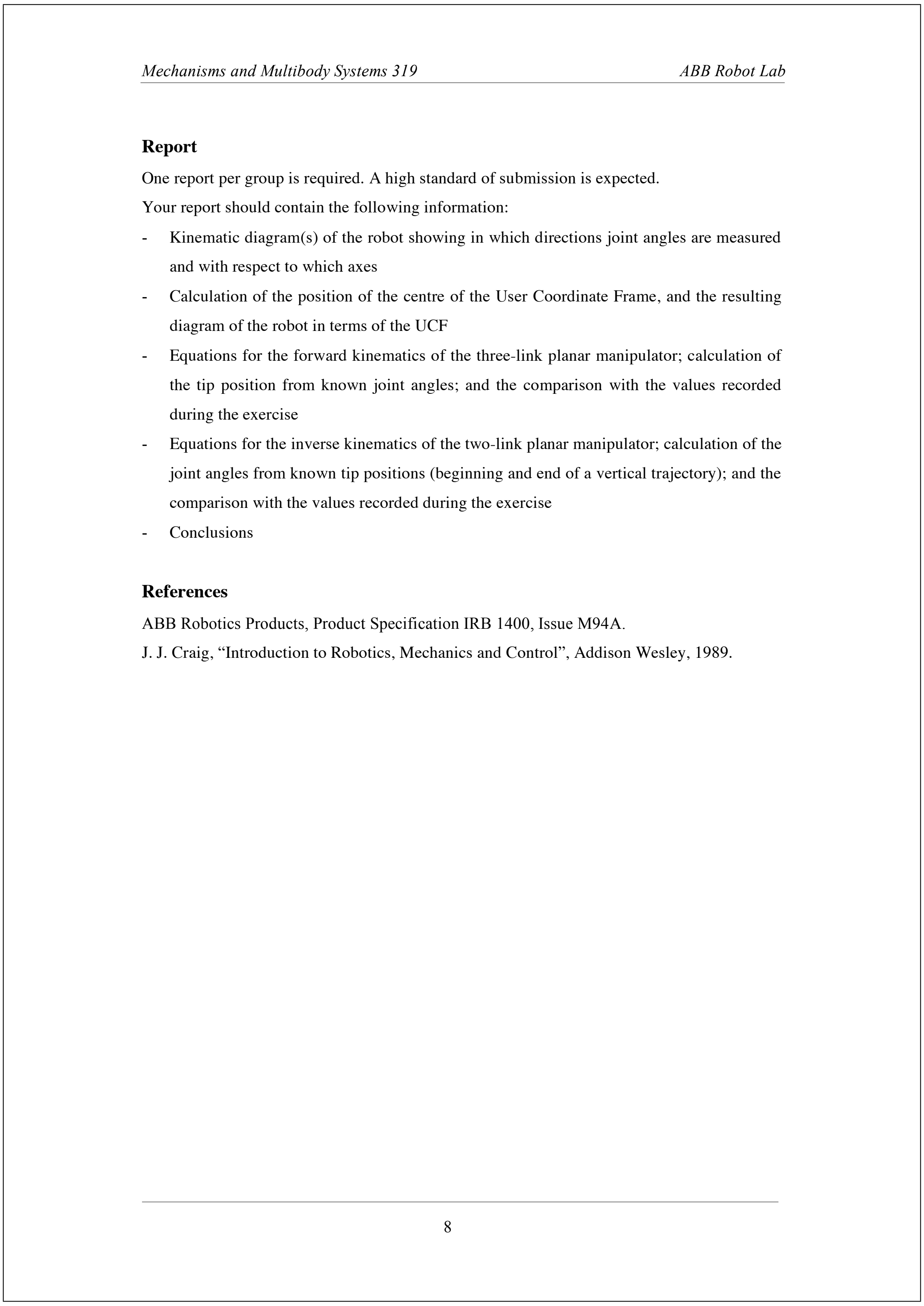
Inverse kinematics involves finding the joints angles required to achieve a given end-effector position.

☐ Collect joint angle and position data:
- Set joint angles 1, 4, 5 and 6 to zero *(by doing this we end up with two-link planar manipulator working in the x-z plane).*
- Using the teach-pendant, move the gripper straight up the z-axis. Record the x- and z-coordinates and the angles θ_2 and θ_3 at the start and end points of this trajectory.

θ_2	θ_3	X	Z

☐ Using inverse kinematics equations and the robot dimensions calculate the theoretical angles θ_2 and θ_3 for these two points. *(Report)*

☐ Compare the calculated angles to those read from the teach-pendant. *(Report)*



Report

One report per group is required. A high standard of submission is expected.

Your report should contain the following information:

- Kinematic diagram(s) of the robot showing in which directions joint angles are measured and with respect to which axes
- Calculation of the position of the centre of the User Coordinate Frame, and the resulting diagram of the robot in terms of the UCF
- Equations for the forward kinematics of the three-link planar manipulator; calculation of the tip position from known joint angles; and the comparison with the values recorded during the exercise
- Equations for the inverse kinematics of the two-link planar manipulator; calculation of the joint angles from known tip positions (beginning and end of a vertical trajectory); and the comparison with the values recorded during the exercise
- Conclusions

References

ABB Robotics Products, Product Specification IRB 1400, Issue M94A.

J. J. Craig, "Introduction to Robotics, Mechanics and Control", Addison Wesley, 1989.

APPENDIX C

Original S4 movement program (MOVE.PRG)

Begin MOVE.PRG

```
%%%
  VERSION:1
  LANGUAGE:ENGLISH
%%%

MODULE MOVE
  CONST speeddata v160:=[150,150,150,1000];
  CONST speeddata vslowrotate:=[150,40,150,1000];
  CONST dionum OldGrip:=0;
  PERS speeddata vsearch:=[150,40,150,1000];
  PERS num dorapid:=0;
  PERS num usedigtool:=0;
  PERS dionum grippos:=0;
  PERS robtarget p10:=[[1092.27,249.549,503.473],[0.0683157,0.286426,-0.881994,-0.367939],[0,0,
  VAR pos currpos;
  VAR robtarget waypoint;
  VAR robtarget cpoint;
  VAR jointtarget joints;

  PROC main()
    ConfL\Off;
    joints:=CJointT();
    IF joints.robax.rax_6>180 THEN
      joints.robax.rax_6:=0;
      MoveAbsJ joints,v160,z60,tool1\WObj:=table;
    ENDIF
    IF joints.robax.rax_6<-180 THEN
      joints.robax.rax_6:=0;
      MoveAbsJ joints,v160,z60,tool1\WObj:=table;
    ENDIF
    !try to get arm up off the table and get joint 5 out of trouble
    IF joints.robax.rax_5>110 THEN
      joints.robax.rax_2:=joints.robax.rax_2-5;
      joints.robax.rax_3:=joints.robax.rax_3-5;
      joints.robax.rax_5:=110;
      MoveAbsJ joints,v160,z200,tool1\WObj:=table;
      joints.robax.rax_5:=90;
      MoveAbsJ joints,v160,z60,tool1\WObj:=table;
    ENDIF
    IF grippos<>DOutput(Gripper) THEN
      SetDO Gripper,grippos;
      WaitTime 0.7;
    ENDIF
    currpos:=CPos();
      IF dorapid=1 THEN
        IF currpos.z<p10.trans.z THEN
          waypoint:=Offs(CRobT(),0,0,p10.trans.z-currpos.z);
          MoveL waypoint,vslowrotate,z10,tool1\WObj:=table;
          MoveL p10,vslowrotate,fine,tool1\WObj:=table;
        ELSE
          waypoint:=Offs(p10,0,0,currpos.z-p10.trans.z);
          MoveL waypoint,vslowrotate,z10,tool1\WObj:=table;
          MoveL p10,vslowrotate,fine,tool1\WObj:=table;
        ENDIF
      ELSE
        MoveL p10,v160,fine,tool1\WObj:=table;
      ENDIF
    ConfL\On;
  ENDPROC
ENDMODULE
```

End MOVE.PRG

APPENDIX D

New S4 movement program (MOVEII.PRG)

Begin MOVEII.PRG

```
%%%
  VERSION:1
  LANGUAGE:ENGLISH
%%%

MODULE MOVEII
  CONST speeddata v160:=[150,150,150,1000];
  VAR jointtarget currentAngles;
  VAR jointtarget joints;
  VAR jointtarget myJointAnglesJT;
  PERS num dorapid:=0;
  PERS num usedigtool:=0;
  CONST dionum OldGrip:=0;
  PERS dionum grippos:=1;
  VAR pos currpos;
  VAR robtarget waypoint;
  VAR robtarget cpoint;
  VAR num compare;

  PERS num prevMode:=0;

  CONST speeddata vslowrotate:=[150,40,150,1000];
  PERS speeddata vsearch:=[150,40,150,1000];

  PERS jointtarget configPos:=[[0,0,0,0,90,0],[0,0,0,0,0,0]];

  PERS robtarget myPosition:=[[480,0,7],[0,0.942641,-0.333807,0],[0,0,0,0],[0,0,0,0,0,0]];
  PERS robtarget myJointAngles:=[[0,0,0],[0,0,0,0],[0,0,0,0],[0,0,0,0,0,0]];
  PERS num mySpeed:=160;
PERS speeddata mySpeedData:=[160,500,5000,1000];
PERS zonedata myZone:=[FALSE,60,90,90,9,90,9];
PERS tooldata myTool:=[TRUE,[[0,0,255],[1,0,0,0]],[1.5,[0,0,110],[1,0,0,0],0,0,0]];
PERS wobjdata myCoordFrame:= [FALSE,TRUE,"",[[1153.96,244.808,507.182],[0.020587,-0.004764,-0.001071,0.9
PERS string myFrame:="";
PERS num myMode:=0;

PERS num joint1:=0;
PERS num joint2:=0;
PERS num joint3:=0;
PERS num joint4:=0;
PERS num joint5:=0;
PERS num joint6:=0;

PROC tableMV()
    !If joint angle 6 is greater than +-180, set to 0
    joints:=CJointT();
    IF joints.robax.rax_6>180 THEN
      joints.robax.rax_6:=0;
      MoveAbsJ joints,mySpeedData,myZone,myTool\WObj:=table;
    ENDIF
    IF joints.robax.rax_6<-180 THEN
      joints.robax.rax_6:=0;
      MoveAbsJ joints,mySpeedData,myZone,myTool\WObj:=table;
    ENDIF

    !Try to get arm up off the table and get joint 5 out of trouble
    IF joints.robax.rax_5>95 THEN
      joints.robax.rax_2:=joints.robax.rax_2-5;
      joints.robax.rax_3:=joints.robax.rax_3-5;
      joints.robax.rax_5:=95;
      MoveAbsJ joints,mySpeedData,z200,myTool\WObj:=table;
      joints.robax.rax_5:=110;
      MoveAbsJ joints,mySpeedData,myZone,myTool\WObj:=table;
    ENDIF

    IF grippos<>DOutput(Gripper) THEN
      SetDO Gripper,grippos;
      WaitTime 0.7;
    ENDIF

    currpos:=CPos();
    IF dorapid=1 THEN
```

```
      IF currpos.z<myPosition.trans.z THEN
        waypoint:=Offs(CRobT(),0,0,myPosition.trans.z-currpos.z);
        MoveL waypoint,vslowrotate,z10,myTool\WObj:=table;
        MoveL myPosition,vslowrotate,fine,myTool\WObj:=myCoordFrame;
      ELSE
        waypoint:=Offs(myPosition,0,0,currpos.z-myPosition.trans.z);
        MoveL waypoint,vslowrotate,z10,myTool\WObj:=table;
        MoveL myPosition,vslowrotate,fine,myTool\WObj:=myCoordFrame;
      ENDIF
    ELSE
      MoveL myPosition,mySpeedData,fine,myTool\WObj:=myCoordFrame;
    ENDIF

    RETURN;
ENDPROC


PROC jointsMV()
        IF grippos<>DOutput(Gripper) THEN
          SetDO Gripper,grippos;
          WaitTime 0.7;
  ENDIF

  !Put joint angles into JointTarget data structure
      myJointAnglesJT.robax.rax_1:= myJointAngles.trans.x;
      myJointAnglesJT.robax.rax_2:= myJointAngles.trans.y;
      myJointAnglesJT.robax.rax_3:= myJointAngles.trans.z;
      myJointAnglesJT.robax.rax_4:= myJointAngles.rot.q1;
      myJointAnglesJT.robax.rax_5:= myJointAngles.rot.q2;
      myJointAnglesJT.robax.rax_6:= myJointAngles.rot.q3;

      MoveAbsJ myJointAnglesJT,mySpeedData,myZone,myTool\WObj:=myCoordFrame;
      RETURN;
ENDPROC


PROC main()
  ConfL\Off;

  !Set required coordinate frame
  IF myFrame="base" THEN
          myCoordFrame:=WObj0;
  ELSE
```

```
            myCoordFrame:=table;
    ENDIF

    !Set tool to be the gripper. This could be changed in later versions.
    myTool:=tool1;

    !Put linear speed value into SpeedData object
    mySpeedData.v_tcp:=mySpeed;

    !Check if the movement mode has changed.
    !If so, take the robot through the configuration position
    !before executing the move.
    compare:=prevMode-myMode;
    IF compare<>0 THEN
      MoveAbsJ configPos,v160,myZone,myTool\WObj:=myCoordFrame;
    ENDIF

    !Determine which mode to move in
    IF myMode=1 THEN
          jointsMV;
    ELSE
          tableMV;
    ENDIF

    !Keep track of the previous mode
    prevMode:=myMode;

    WaitTime 1.5;

    !Put current angles into NUMs
    currentAngles := CJointT();

    joint1 := currentAngles.robax.rax_1;
    joint2 := currentAngles.robax.rax_2;
    joint3 := currentAngles.robax.rax_3;
    joint4 := currentAngles.robax.rax_4;
    joint5 := currentAngles.robax.rax_5;
    joint6 := currentAngles.robax.rax_6;

    ConfL\On;
  ENDPROC
ENDMODULE
```

End MOVEII.PRG

APPENDIX E

MMS 319 Remote Exercise

MMS 319 Telerobot Remote Exercise

2nd Semester, 2004

The aim of this task is to give you some practise at solving the kinematics of a simple two-bar linkage, and to test the remote control of the Telerobot. This will be good planar kinematics revision.

The Telerobot's links 2 & 3 can be represented as a two-bar mechanism:

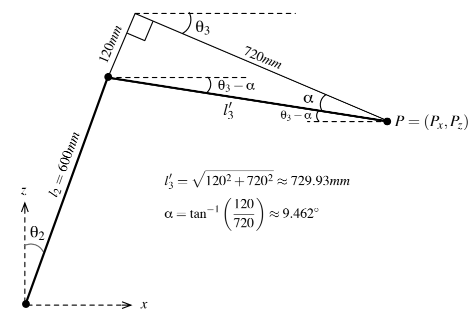


Figure 1: Side view of the Telerobot links 2 & 3

A change in the value of a joint angle causes the end-effector to move in an arc about that joint.

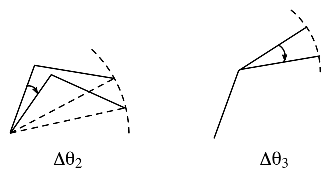


Figure 2: Changes in θ_2 and θ_3 cause the end-effector to move in an arc

Prepared by: Samuel Rae 1

For a small change, this arc can be approximated by a straight line. Thus, for small changes in the angles ($\Delta\theta_2$ and $\Delta\theta_3$) we can approximate the change in the end-effector position ($\Delta P_x, \Delta P_y$). We can do this by solving:

$$\frac{\delta\Phi}{\delta q}\Delta q = 0$$

where $\frac{\delta\Phi}{\delta q}$ is the Jacobian, and Δq is a small change in the inputs. For this two-bar mechanism, $q = \begin{bmatrix} \theta_2 \\ \theta_3 \\ P_x \\ P_z \end{bmatrix}$. (See your lecture notes for more information).

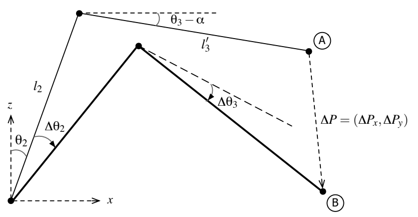


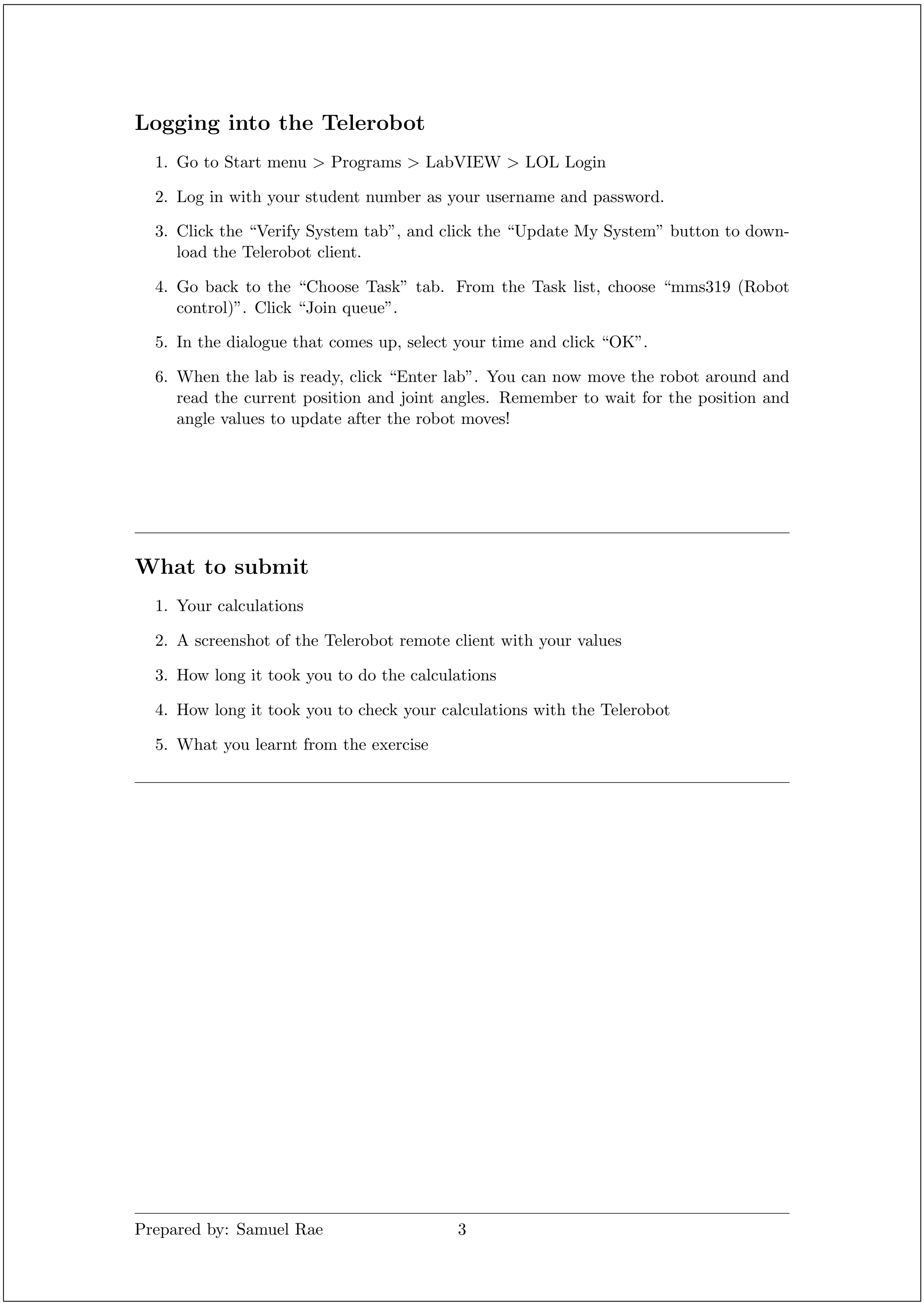
Figure 3: Small angle changes $\Delta\theta_2$ and $\Delta\theta_3$

Procedure

1. Write down the vector closure equation for the linkage. Split this into equations for the x and z directions.
2. Write down Φ and q for the mechanism.

$$\Phi = \begin{bmatrix} \textit{x-direction components} \\ \textit{z-direction components} \end{bmatrix} = \tilde{0}$$

3. Find $\frac{\delta\Phi}{\delta q}$, the derivative of Φ with respect to q (in this case it should be a 2×4 matrix).
4. Using $\Delta q = \begin{bmatrix} \Delta\theta_2 & \Delta\theta_3 & \Delta P_x & \Delta P_z \end{bmatrix}$, write out the matrix $\frac{\delta\Phi}{\delta q}\Delta q = 0$ (in this case it is a 2×1 matrix). Then write the two rows in $\Delta P_x = \ldots$ and $\Delta P_y = \ldots$ form.
5. Choose arbitrary values for θ_2, θ_3, $\Delta\theta_2$ and $\Delta\theta_3$ ($\Delta\theta_2$ and $\Delta\theta_3$ should be $< 3°$), and calculate ΔP_x and ΔP_y. (Make sure you convert $\Delta\theta_2$ and $\Delta\theta_3$ to radians)
6. Log into the Telerobot (see instructions below).
7. Set the angles to (0, θ_2, θ_3, 90, 90, 0), move the robot and record the x and z coordinates.
8. Set the angles to (0, $\theta_2 + \Delta\theta_2$, $\theta_3 + \Delta\theta_3$, 90, 90, 0), move the robot and compare the position to what you calculated before.
9. Choose some larger values for $\Delta\theta_2$ and $\Delta\theta_3$ and find out at what values the approximation starts to break down.



Logging into the Telerobot

1. Go to Start menu > Programs > LabVIEW > LOL Login
2. Log in with your student number as your username and password.
3. Click the "Verify System tab", and click the "Update My System" button to download the Telerobot client.
4. Go back to the "Choose Task" tab. From the Task list, choose "mms319 (Robot control)". Click "Join queue".
5. In the dialogue that comes up, select your time and click "OK".
6. When the lab is ready, click "Enter lab". You can now move the robot around and read the current position and joint angles. Remember to wait for the position and angle values to update after the robot moves!

What to submit

1. Your calculations
2. A screenshot of the Telerobot remote client with your values
3. How long it took you to do the calculations
4. How long it took you to check your calculations with the Telerobot
5. What you learnt from the exercise

Prepared by: Samuel Rae 3

APPENDIX F

How-to: Using ActiveX objects within the Telerobot software

How-to: Using ActiveX objects within the Telerobot software

Samuel Rae

September 4, 2004

This document shows the process to go through when using an ActiveX method in the Telerobot software.

1 Classes & methods

Before looking at the specifics of the Telerobot software, some background is needed on ActiveX classes and methods. The following steps can be done in a blank VI to get familiar with ActiveX in LabVIEW.

To do anything with ActiveX in LabVIEW, you first need a reference to an ActiveX control. This is created by going to *Controls Palette>Refnum>Automation Refnum* (figure 1). This creates a control reference without a class.

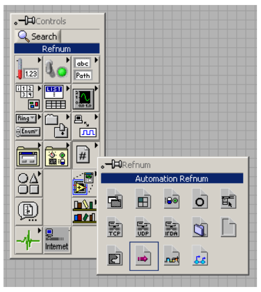


Figure 1: Placing an Automation Refnum (ActiveX object reference)

To select a class, right-click on the control and go to *Select AvtiveX Class>Browse. . . .* You will be presented with a dialog showing all the available ActiveX controls. To see the controls that allow interaction with the robot, click the "Type Library" menu and select "ABB RobComm OLE Custom Control module Version 1.0" (figure 2).

1

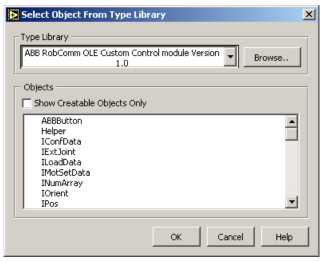


Figure 2: Selecting the ActiveX object

You must now select an object from the bottom list. The names are mainly based on the types of variables in the RAPID language used on the S4 controller. Work out which one you need, then select the appropriate object. For many tasks, the "Helper" object will have an appropriate method.

Once you have selected a class, go to the block diagram of the VI. Now that you have the object reference, you need to get it to do something. Right-click to get the controls palette, and navigate to *Application Control>Invoke node*. An Invoke node is used for getting ActiveX to do something, while a Property node (from the same palette) is used for reading some property of the object—usually you will want to use an Invoke node. Wire the ActiveX object reference to the top left terminal of the Invoke node. You will notice that the Invoke node takes on the last part of the name of the object.

You can now select the method you want the invoke node to perform. Right-click the Invoke node and navigate to "Methods". This will show all the methods that can be invoked for that object. Select the appropriate method (if you don't know what you want, look at the names and figure out what you need before you start hacking away).

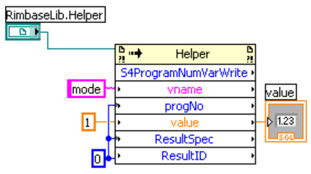


Figure 3: An Invoke node to write a `num` variable

The invoke node will now have all its inputs and outputs. You will need to wire all the required terminals (figure 3). If you don't know what the value for an input should be, set it to zero. The important inputs are usually the name of the thing you want to change, and the new value for that

2

thing. In the case of reading a variable from the robot, you will need to input the name of the variable to read, some zeros for the numerical arguments, and an empty container of the type of the variable (eg. and empty string, or a 0 integer) for ActiveX to put the retrieved value in.

2 Telerobot code

Now that you know how classes and methods work, we can get stuck into the specific Telerobot code. This section will go through what you need to do to insert a new ActiveX method.

2.1 S4ActiveXControls.vi

As mentioned before, all ActiveX commands require a reference to an ActiveX control in order to function. These references MUST be on the front panel of the main VI (Telerobot Hardware Master) to work (though they can be made invisible). To save the code looking like spaghetti, the references to the four ActiveX controls currently in use are stored in a subVI called `S4ActiveXControls` (figure 4). The references can be extracted at any location just by placing the subVI and running the appropriate wires out from it. The subVI itself is essentially just a loop with shift-registers in it, effectively making the controls global variables (figure 5).

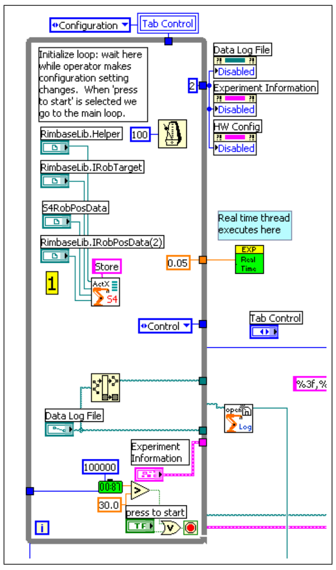


Figure 4: Initialising the ActiveX controls

3

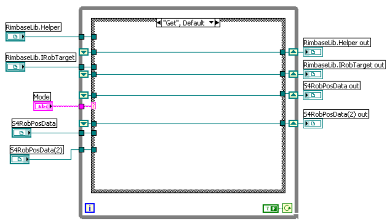


Figure 5: The `S4ActiveXControls` subVI

3 Procedure for adding a new ActiveX method

1. To use a new ActiveX method, first see if it can be invoked from any of the references in the `S4ActiveXControls` subVI.

   (a) If it can, then you need not modify `S4ActiveXControls`.

   (b) If not, you will need to create a new reference of the appropriate type on the front panel, and add the connectors and wiring to the `S4ActiveXControls` subVI.

2. The subVI RunHardware has all the code for controlling the robot. Open it and go to the case “Start up” (figure 6). In this case the ActiveX controls are retrieved and put into shift registers.

   (a) If you made no modifications to `S4ActiveXControls`, you do not need to change anything here.

   (b) If you created a new object reference in `S4ActiveXControls`, you will need to get it out of the `S4ActiveXControls` subVI here and wire it to a new shift register.

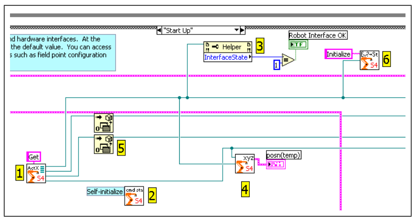


Figure 6: RunHardware in “Start up” case

3. Go to the case “Default”, where the code to move the robot is. You will see that the dark green wires of the ActiveX controls branch off into various sections of the code to perform

4

different functions. For your function, simply run a wire off the appropriate reference to your subVI to perform whatever function you want.

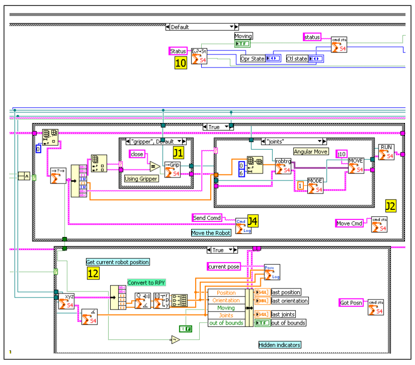


Figure 7: RunHardware in "Default" case

4 Done!

Assuming that you've done all that properly, and selected the right class and method, that should be all there is to it. Make sure you wire up the error cluster to pass through any modules you make, so that you can see where things go wrong.

APPENDIX G

Denavit-Hartenburg transformations: conventions

G.1 Introduction

The term Denavit-Hartenburg can refer to a number of similar transformations. Different textbooks use different conventions. This appendix explains some of the differences, and why the convention used in this thesis was chosen.

G.2 Transformation order

As shown in Chapter 7, Denavit-Hartenburg transformations are a method of describing robot links using four parameters (θ, d, a, and α) and the equation:

$$A^i_{i-1} = Rot(z_{i-1}, \theta_i)\ Trans(0,0,d_i)\ Trans(a_i,0,0)\ Rot(x_i,\alpha_i) \tag{G.1}$$

For simplicity's sake, Equation G.1 is written in this appendix as:

$$A = Rot(z,\theta)\ Trans(0,0,d)\ Trans(a,0,0)\ Rot(x,\alpha) \tag{G.2}$$

where A is the transformation from the coordinate frame at the base of a link to the frame at the base of the next link.

Equation G.2 could be written in a number of ways. Firstly, it is obvious that the z-axis does not change in the first two transforms, so they are commutative:

$$Rot(z,\theta)\ Trans(0,0,d) = Trans(0,0,d)\ Rot(z,\theta)$$

The same is true of the x-axis in the last two transforms, so:

$$Trans(a,0,0)\ Rot(x,\alpha) = Rot(x,\alpha)\ Trans(a,0,0)$$

Thus the full Denavit-Hartenburg equation could be written as any of the following:

$$A = Rot(z,\theta)\ Trans(0,0,d)\ Trans(a,0,0)\ Rot(x,\alpha) \tag{G.3}$$

$$A = Rot(z,\theta)\ Trans(0,0,d)\ Rot(x,\alpha)\ Trans(a,0,0) \tag{G.4}$$

$$A = Trans(0,0,d)\ Rot(z,\theta)\ Rot(x,\alpha)\ Trans(a,0,0) \tag{G.5}$$

$$A = Trans(0,0,d)\ Rot(z,\theta)\ Trans(a,0,0)\ Rot(x,\alpha) \tag{G.6}$$

Stone [59] uses the form in Equation G.3, Koivo [60] uses that in Equation G.5, Klafter, Chmielewski, and Negin [61] use that in G.3 but multiplies right-to-left, Craig [58] uses that in Equation G.6 but multiples right-to-left also.

I chose the form in Equation G.3 because I believe it is the most intuitive and easiest to follow when thinking of a real robot.

G.3 Attaching axes to joints

You may in certain textbooks see the z-axis rotation part of Equation G.1 written as:

$$rot(z_i,\theta_i) \tag{G.7}$$

or

$$rot(z_{i-1},\theta_i) \tag{G.8}$$

The reason for this discrepancy is that the books use different conventions to attach coordinate frames to links. Craig, for instance, attaches the ith frame to the ith joint, while Stone attaches the $i-1$th coordinate frame to the ith joint. Thus depending on which convention you use, you would use Equation G.7 or G.8 respectively.

The choice of which convention to use is fairly arbitrary, and does not affect the DH parameters. I chose to use Stone's (Equation G.7) because I was familiar with it.

G.4 A and T matrices

In some textbooks the Equation G.1 may begin with $A^i_{i-1} = \ldots$, while others read $T^i_{i-1} = \ldots$ and still others read ${}_{i-1}{}^iA = \ldots$ or some similar variation.

The generally accepted convention seems to be that the matrix formed by Equation G.1 is an 'A' matrix (ie. A^i_{i-1}). The matrix T refers only to the coordinate frame describing the position and orientation of the robot end-effector (final link) relative to the coordinate frame at the base of the robot. That is,

$$T^n_0 = A^n_0 = A^1_0 \ldots A^n_{n-1} \tag{G.9}$$

where n is the number of joints.

It could be argued (as Craig seems to do), that because T^n_0 refers to coordinate frame n in terms of coordinate frame 0, any matrix T^b_a refers to coordinate frame b in terms of coordinate frame a, thus Equation G.9 could be rewritten as

$$T^n_0 = T^1_0 \ldots T^n_{n-1} \tag{G.10}$$

Koivo, however, shows that the matrix T^n_0, while equal to A^n_0, is derived from an entirely different method that works only for the final coordinate frame, and that a general matrix T^b_a does not exist. This would suggest that Equation G.10 is not strictly true.

This thesis uses the generally accepted form of Equation G.9.

G.4.1 Subscripts and superscripts

The different books place the subscripts and superscripts of the A and T matrices at different places, eg:

$$A^i_{i-1} \qquad A^{i-1}_i \qquad A^i_{i-1} \qquad {}_{i-1}{}^iA \qquad A_{i-1,i} \qquad A_i \qquad \ldots$$

Although there are many different forms, they are usually self-explanatory.

G.5 Expanding the matrices

When looking at specific examples of real robots, some textbooks fully expand the matrices given by Equation G.1, eg:

$$A^3_2 = \begin{bmatrix} cos\theta_3 & -sin\theta_3 & 0 & 0 \\ sin\theta_3 & cos\theta_3 & 0 & 0 \\ 0 & 0 & 1 & l_2 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Once this has been done, it is no longer necessary to do all the rotations and translations of G.1 for every link—the matrices can be found by simply substituting in the DH parameters for that link. However, while this may be a quick method to use manually, it is quite pointless in a program such as the LabVIEW controller for the telerobot. In a program it is much easier to create sub-programs to do each of the translations and rotations (eg. *rotz*, *trans* and *rotx*) and then simply multiply them together. This is a trivial operation for a computer, and allows a much better understanding for the programmer, and also gives additional expandability and portability (eg. to other robots).

APPENDIX H

Robot transformation matrices

H.1 The matrices

$$
{}^0_1T = \begin{bmatrix} \cos\theta_1 & 0 & -\sin\theta_1 & 150\cos\theta_1 \\ \sin\theta_1 & 0 & \cos\theta_1 & 150\sin\theta_1 \\ 0 & -1 & 0 & 475 \\ 0 & 0 & 0 & 1 \end{bmatrix}
$$

$$
{}^1_2T = \begin{bmatrix} \cos(90-\theta_2) & \sin(90-\theta_2) & 0 & 600\cos(90-\theta_2) \\ -\sin(90-\theta_2) & \cos(90-\theta_2) & 0 & -600\sin(90-\theta_2) \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}
$$

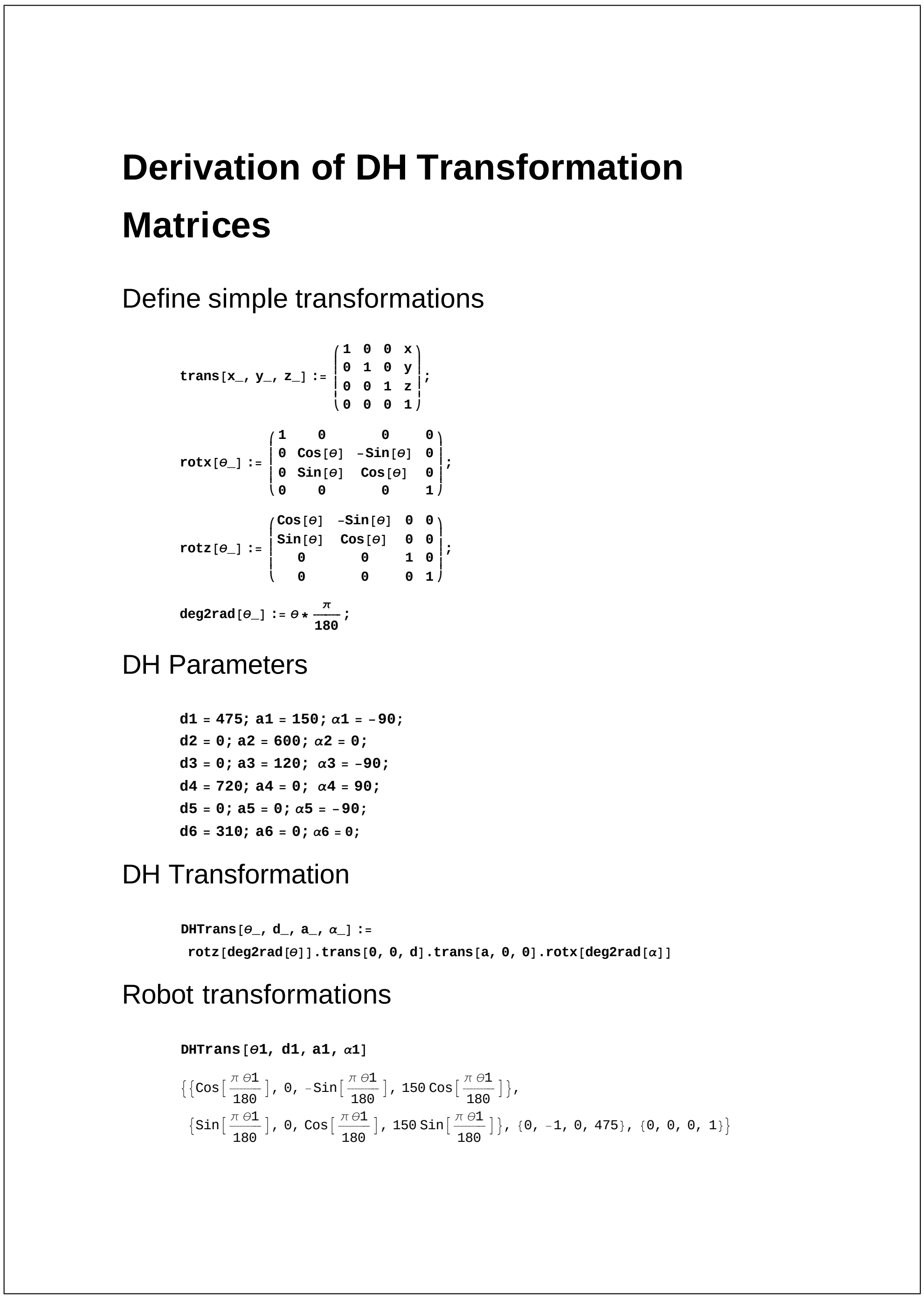
$$
{}^2_3T = \begin{bmatrix} \cos(\theta_2-\theta_3) & 0 & \sin(\theta_2-\theta_3) & 120\cos(\theta_2-\theta_3) \\ -\sin(\theta_2-\theta_3) & 0 & \cos(\theta_2-\theta_3) & -120\sin(\theta_2-\theta_3) \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}
$$

$$
{}^3_4T = \begin{bmatrix} \cos\theta_4 & 0 & \sin\theta_4 & 0 \\ \sin\theta_4 & 0 & -\cos\theta_4 & 0 \\ 0 & 1 & 0 & 720 \\ 0 & 0 & 0 & 1 \end{bmatrix}
$$

$$
{}^4_5T = \begin{bmatrix} \cos\theta_5 & 0 & -\sin\theta_5 & 0 \\ \sin\theta_5 & 0 & \cos\theta_5 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}
$$

$$
{}^5_6T = \begin{bmatrix} \cos\theta_6 & -\sin\theta_6 & 0 & 0 \\ \sin\theta_6 & \cos\theta_6 & 0 & 0 \\ 0 & 0 & 1 & 310 \\ 0 & 0 & 0 & 1 \end{bmatrix}
$$

H.2 Mathematica code



Derivation of DH Transformation Matrices

Define simple transformations

```
                    ⎛1  0  0  x⎞
trans[x_, y_, z_] := ⎜0  1  0  y⎟;
                    ⎜0  0  1  z⎟
                    ⎝0  0  0  1⎠

            ⎛1     0        0      0⎞
rotx[θ_] := ⎜0  Cos[θ]  -Sin[θ]   0⎟;
            ⎜0  Sin[θ]   Cos[θ]   0⎟
            ⎝0     0        0      1⎠

            ⎛Cos[θ]  -Sin[θ]  0  0⎞
rotz[θ_] := ⎜Sin[θ]   Cos[θ]  0  0⎟;
            ⎜  0        0     1  0⎟
            ⎝  0        0     0  1⎠

deg2rad[θ_] := θ * π/180;
```

DH Parameters

```
d1 = 475; a1 = 150; α1 = -90;
d2 = 0; a2 = 600; α2 = 0;
d3 = 0; a3 = 120; α3 = -90;
d4 = 720; a4 = 0; α4 = 90;
d5 = 0; a5 = 0; α5 = -90;
d6 = 310; a6 = 0; α6 = 0;
```

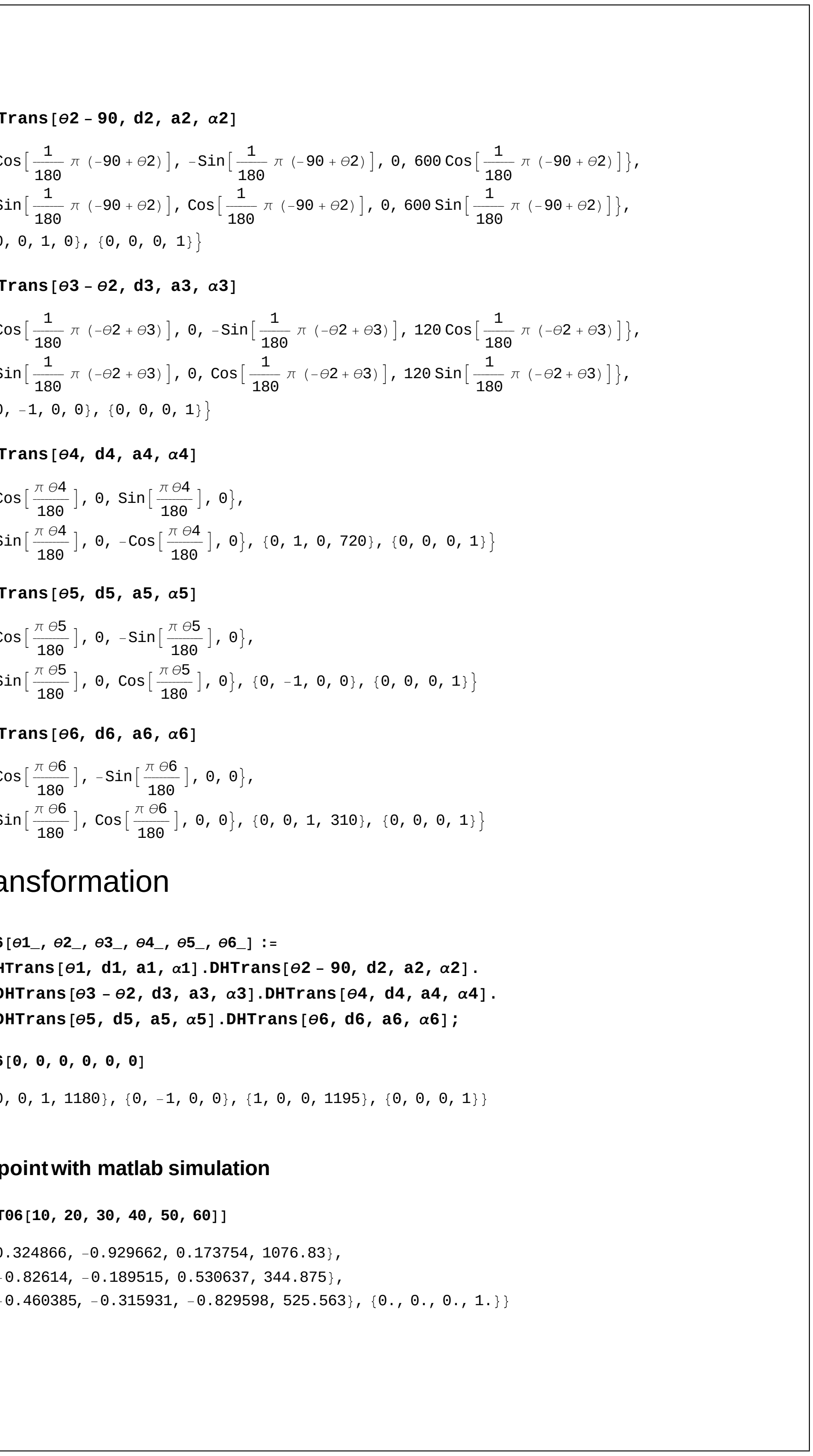
DH Transformation

```
DHTrans[θ_, d_, a_, α_] :=
 rotz[deg2rad[θ]].trans[0, 0, d].trans[a, 0, 0].rotx[deg2rad[α]]
```

Robot transformations

```
DHTrans[θ1, d1, a1, α1]
```

$$\left\{\left\{\mathrm{Cos}\left[\frac{\pi\,\theta 1}{180}\right], 0, -\mathrm{Sin}\left[\frac{\pi\,\theta 1}{180}\right], 150\,\mathrm{Cos}\left[\frac{\pi\,\theta 1}{180}\right]\right\}, \left\{\mathrm{Sin}\left[\frac{\pi\,\theta 1}{180}\right], 0, \mathrm{Cos}\left[\frac{\pi\,\theta 1}{180}\right], 150\,\mathrm{Sin}\left[\frac{\pi\,\theta 1}{180}\right]\right\}, \{0, -1, 0, 475\}, \{0, 0, 0, 1\}\right\}$$



```
Trans[θ2 - 90, d2, a2, α2]

Cos[1/180 π (-90 + θ2)], -Sin[1/180 π (-90 + θ2)], 0, 600 Cos[1/180 π (-90 + θ2)]},
Sin[1/180 π (-90 + θ2)], Cos[1/180 π (-90 + θ2)], 0, 600 Sin[1/180 π (-90 + θ2)]},
), 0, 1, 0}, {0, 0, 0, 1}}

Trans[θ3 - θ2, d3, a3, α3]

Cos[1/180 π (-θ2 + θ3)], 0, -Sin[1/180 π (-θ2 + θ3)], 120 Cos[1/180 π (-θ2 + θ3)]},
Sin[1/180 π (-θ2 + θ3)], 0, Cos[1/180 π (-θ2 + θ3)], 120 Sin[1/180 π (-θ2 + θ3)]},
), -1, 0, 0}, {0, 0, 0, 1}}

Trans[θ4, d4, a4, α4]

Cos[π θ4/180], 0, Sin[π θ4/180], 0},
Sin[π θ4/180], 0, -Cos[π θ4/180], 0}, {0, 1, 0, 720}, {0, 0, 0, 1}}

Trans[θ5, d5, a5, α5]

Cos[π θ5/180], 0, -Sin[π θ5/180], 0},
Sin[π θ5/180], 0, Cos[π θ5/180], 0}, {0, -1, 0, 0}, {0, 0, 0, 1}}

Trans[θ6, d6, a6, α6]

Cos[π θ6/180], -Sin[π θ6/180], 0, 0},
Sin[π θ6/180], Cos[π θ6/180], 0, 0}, {0, 0, 1, 310}, {0, 0, 0, 1}}
```

ansformation

```
5[θ1_, θ2_, θ3_, θ4_, θ5_, θ6_] :=
HTrans[θ1, d1, a1, α1].DHTrans[θ2 - 90, d2, a2, α2].
DHTrans[θ3 - θ2, d3, a3, α3].DHTrans[θ4, d4, a4, α4].
DHTrans[θ5, d5, a5, α5].DHTrans[θ6, d6, a6, α6];

5[0, 0, 0, 0, 0, 0]

), 0, 1, 1180}, {0, -1, 0, 0}, {1, 0, 0, 1195}, {0, 0, 0, 1}}
```

point with matlab simulation

```
T06[10, 20, 30, 40, 50, 60]]

).324866, -0.929662, 0.173754, 1076.83},
-0.82614, -0.189515, 0.530637, 344.875},
-0.460385, -0.315931, -0.829598, 525.563}, {0., 0., 0., 1.}}
```