Manipulator, Mounting and Integration, and Communication Brick Collection

CAMILLA LJUNGSTRÖM

KTH Electrical Engineering

Masters' Degree Project
Stockholm, Sweden September 2008

XR-EE-RT 2008:020

Abstract

The aim of this project is to add manipulation functionality to one of the NASA's Mars rovers through the integration of a robotic arm. The functionality is needed to enable retrieval of deployed communication bricks from the Martian surface. For this, a simple hook end-effector is designed and tested using a controller implemented on a computer.

The manipulator is a five degree of freedom robotic arm with revolute joints. It is a nonlinear MIMO system but can be approximated to be five highly decoupled linear SISO systems. The manipulator is controlled in joint space, where each joint follows a trajectory. The trajectory is generated as four sequential sub-trajectories, where the manipulator comes to rest at each transition point.

We assume that the Martian surface is free of obstacles and that the manipulator can work in any position on the workspace. We also assume that the position of the hole on the communication brick is known.

For controlling the end-effector in joint space, a trajectory-following PD-controller is constructed.

When testing the controller the first four joints (waist, shoulder, elbow and twist) follow the desired trajectory well. The last joint, wrist, follows the trajectory well for the first period of time and then start to oscillate around the path curve. The controller was tuned by decreasing K_{p} value for the wrist joint. The reason for deciding to decrease K_{p} was to slow the system down and therefore gain an increased stability. Tuning resulted in a better behaving wrist joint controller which follows the desired trajectory in a satisfactory way. Oscillations in wrist have disappeared and the controller works as intended.

We designed and manufactured an end-effector that was within the IRG price range. We established baseline communication with the manipulator, developed a standalone control software and implemented a software for collection of communication bricks. This controller was working properly and fulfilled the goal set at the beginning of the thesis to design a communication brick collecting manipulator.

Acknowledgements

I would like to thank everyone who supported me during this thesis work:

- Vytas Sunspiral, for having me as an intern, escorting me on and off base every day for two months and letting me be a part of the arm lab. All support on my work and effort on finding segmentation faults in my C++ code.
- Terry Fong, for arranging my internship at NASA Ames and spending a lot of time getting my badge and computer account.
- IRG, I would like to thank everyone at IRG for their support. Special thanks to Vinh for guiding me through C++ and CLARAty, and Susan for helping with hardware issues.
- CMU and CMU West Coast Campus, for handling my visa and for making my internship at NASA possible.
- Robert Platt, for answering my questions about manipulation and being a good source on model and control.
- Daniel, Thomas and Stefan, for funny times in the arm lab and delicious coffee during the breaks.
- Eric, for introducing me to Terry and delicious yoghurt evenings in Palo Alto.
- My family, for supporting my studies at KTH and thesis in California.
- Joakim, for being by my side and supporting my work.

Acronyms

CLARAty	Coupled Layered Architecture for Robotic Autonomy
CORBA	Common Object Requesting Broker Architecture
DC-motor	Direct current motor
DH parameter	Denavit-Hartenberg paratmeter
DOF	degrees of freedom
FIDO	Field Integrated Design and Operations
IRG	NASA Ames Intelligent Robotics Group
JPL	NASA Jet Propulsion Laboratory, California Institute of Technology
MER	Mars Exploration Rover
MIMO	Multiple-Input Multiple-Output
NASA	National Aeronautics and Space Administration
PD	Proportional-Derivative
SISO	Single-Input Single-Output

Symbols

Model

b	[lb$_f$-in/(deg/s)]	Load shaft viscous damping coefficient
b_m	[lb$_f$-in/(deg/s)]	Motor shaft viscous damping coefficient
i_a	[A]	Armature current
I	[lb$_f$-in-s^2]	Lumped load polar inertia
I_m	[lb$_f$-in-s^2]	Lumped motor polar inertia
k_a	[-]	Amplifier gain
k_e	[V/(deg/s)]	Back emf constant
k_m	[lb$_f$-in/A]	Torque constant
l_a	[H]	Armature inductance
η	[-]	Gear ratio
r_a	[Ω]	Armature resistance
τ	[lb$_f$-in]	Load torque
τ_m	[lb$_f$-in]	Generated motor torque
v_a	[V]	Armature voltage
v_b	[V]	Back emf
$\dot{\theta}$	[deg/s]	Load shaft velocity
$\dot{\theta}_m$	[deg/s]	Motor shaft velocity
ω_n	[Hz]	Natural frequency
ω_{res}	[Hz]	Vibration frequency

Control

e	[deg]	Position error
$\dot{e}$	[deg/s]	Velocity error
θ	[deg]	Position, output from the system
$\dot{\theta}$	[deg/s]	Velocity, output from the system
θ_d	[deg]	Position, desired by trajectory
$\dot{\theta}_d$	[deg/s]	Velocity, desired by trajectory

Contents

List of Figures

Chapter 1

Introduction

The overall objective of this thesis project was to work towards adding manipulator functionality to a K10 mobile robot. Research was performed at NASA Ames Research Center Intelligent Robotics Group (IRG), Moffett Field, California and Carnegie Mellon University (CMU). In this report the robotic arm will be referred to as a manipulator, while the hand attached at the end of the manipulator will be referred to as an end-effector.

1.1 Background

In 2003 the Mars Exploration Rover (MER) mission was launched with the twin rovers Spirit and Opportunity (Figure 1.1), landing on Mars in January 2004. MER is a part of NASA's Mars Exploration Program, a long-term effort of robotic exploration of Mars, see [NASA Mars Rovers]. The science objective for MER is to answer the question of how past water activity on Mars influenced the environment over time. While there is no liquid water on the surface today, record of past water activity can be found in rocks, minerals and geological landforms, particularly in those that can only form in the presence of water. That is why the rovers are specially equipped with tools to study a diverse collection of rocks and soils that may hold clues to past water activity. The twin rovers each have a five degrees of freedom (DOF) manipulator mounted in front, underneath the solar panel platform. The manipulator has four different sophisticated tools for the different tasks the rovers are supposed to perform.

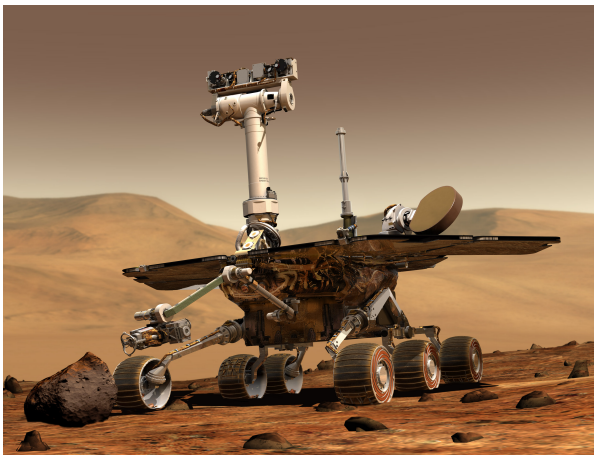

Figure 1.1. One of the MER rovers (Courtesy NASA).

The twin rovers have been researching the Martian surface since 2004 and are still active, far exceeding their original three-month prime mission, and will possibly keep on running through 2009. The four year old rovers show some signs of aging, but they are still in good health and capable of conducting great science. Opportunity has had minor problems with the manipulator since late 2005 and during spring 2008 the problem increased. Sometimes the waist motor (the motor which turns the end-effector sideways) stalls and is impossible to move. This joint is mainly used for stowing and unstowing the arm, and if the joint stalls in the stowed position the arm will be unusable. To prevent this from happening, the manipulator is kept unstowed during the night and only stowed away when the rover is driving.

Currently NASA and IRG are doing research on the next generation of autonomous rovers. The technology developed at IRG might be used on future rovers sent to Mars. IRG has two rovers used for research purposes, K10 black and K10 red. To maintain a communication link between rover and operators, a wireless network system from SPAWAR has been integrated [Pezeshkian et al. 2007]. The idea is a system where communication bricks are automatically deployed whenever the wireless signal strength drops below a threshold. This system can be used for maintaining a wireless link when driving around hills to avoid going out of range from base camp. Whenever the rovers move on and leave a spot the communication bricks need to be collected and taken back to the base camp. This is where the manipulator comes in with its hook-shaped end-effector. The rover approaches the communication brick and the manipulator picks it up. The manipulator moves to a carry position and the rover travels back to the base camp where the communication brick will be collected by an astronaut who will fold the antenna back into the communication brick and lock it by closing the two flaps.

The K9 arm is a 5-DOF replica of the Field Integrated Design and Operations (FIDO) rover's micro-instrument 4-DOF robotic arm based upon documentation obtained from NASA Jet Propulsion Laboratory (JPL). In order to allow the instrument to reach any point in the workspace, an additional twisting joint was added just beyond the elbow joint. The arm, with a Camera HAndlens MicroscoPe (CHAMP) attached at the end, is presently mounted on the K9 rover and will be moved to one of IRG's K10 mobile robots. CHAMP will share the tool palatte with an end-effector developed for the purpose of collecting communication bricks.

1.2 Problem Definition

The objective of this thesis was to work towards adding manipulator functionality to the K10 rover through integration of a robotic arm. Required tasks include:

- Forward and inverse kinematic control of a 5-DOF robotic arm
- End-effector design
- Arm integration design on K10
- System integration
- Control algorithm design for collection of communication brick

The existing K9 arm has five highly geared MicroMo harmonic drive motors controlled through a stack of JR-Kerr Pic-Servo boards. Each motor is instrumented with both a motor shaft encoder and a potentiometer for relative and absolute position sensing. One part of the project will be to extract the most recent arm code from the K9 software repository and make it standalone for use on the K10 platform. K9 is a rover, similar to the Mars rovers Spirit and Opportunity, designed by IRG. K9 was the rover on which IRG did all their testing until the group started using the k10

rovers. The software has dependencies to Coupled Layered Architecture for Robotic Autonomy (CLARAty), which has quite a few differences between K9 and K10 versions. CLARAty is a reusable robotic software framework created and updated collaboratively by four institutions: JPL, NASA Ames, CMU and University of Minnesota. One of the big differences between the two generations of rovers is that the K10 software is built so that different parts of the rover code base can be used standalone and linked together using CORBA. This is a standard that enables software components written in different computer languages and running on multiple computers to work together.

An end-effector will be designed and created which will be able to collect objects. This end-effector will be designed as simple as possible – either a simple shovel, or a flat board which will slide under a communication brick and pick it up by lifting the arm. Another set of tasks will be to work on low level motor control of the arm, and to establish both forward and inverse kinematic control of the arm so that it can be commanded to arbitrary positions within its workspace. Finally, the whole system will be physically mounted on a K10 robot and will be integrated with the robot's electrical and data systems (the arm communicates via a standard RS-232 serial cable). Once fully integrated, the system will be demonstrated by collecting the device via commands generated by the K10 rover in IRG's outdoor Marscape test facility.

1.3 Objectives

The following objectives have been put forth for the project. Objectives 5 and 6 are optional and will only be implemented if time permits.

1. Establish baseline communication with arm
2. Develop standalone control software for the arm
3. Design an end-effector
4. Write software for controlling manipulator with end-effector attached
5. Integrate arm software with K10 rover code base
6. Mount arm on K10 rover

1.4 Previous Work

Many universities and corporate robotics labs perform research with manipulators and in the field of manipulation. Robotic arms are used to manipulate objects in many ways including grasping, modification, and destruction of objects. Industrial robotic manipulators are used for welding, painting, ironing, assembly, pick and place, packaging and palletizing, product inspection, and testing. Often robot arms have replaceable end-effectors. Each end-effector allows them to perform a small range of tasks.

Many manipulators have six joints, corresponding to the six DOF needed to obtain arbitrary positions and orientations of the end-effector in three-dimensional space. Arms like the PUMA 560 have six revolute joints. In such an arm, the joints may be grouped into two sets of three joints each. The first set may be used to place the end-effector at an arbitrary position within the three-dimensional workspace. The last set may be used to obtain an arbitrary orientation of the end-effector at that position. This set is called the wrist mechanism. Industrial examples of a three revolute (RRR) manipulator are the PUMA and the Cincinnati-Milacron T3 735

manipulators. There are also five DOF manipulators used in industry today, with three joints for position and two joints for orientation.

According to [Craig. 2002] the industrial manipulators today use a PD or PID controller where the servo portion of the controller is ignored. For most manipulators it sufficient to decompose the system into a set of linear SISO systems.

1.5 Summary of contributions

Our contributions at IRG during this project were many different tasks all connected to the manipulator project in some way. Our project time can be divided into two equally big parts. The first part was spent getting the old arm code working without any influences from CLARAty, making the arm move while still mounted to K9, and getting the arm software working with the rest of K9's code. Neither of these approaches worked because the arm code depended on CLARAty at too many levels, and CLARAty had changed too much since it was last used with K9. During the first part of the project we also derived the Denavit-Hartenberg (DH) parameters for the arm, for later use when working on the kinematics for the arm. Another task was to investigate if it would be possible to use an object-oriented framework for the development of control programs for robotic manipulators, called Operational Software Components for Advanced Robotics (OSCAR), developed by University of Texas at Austin. OSCAR was interesting to use because it is an easy way to derive kinematics for the arm and could be used to replace the currently not working inverse kinematics. The main drawback with OSCAR was that it is developed to be run on a Windows computer. Although we managed to get it to work in Linux, there were too many issues involved in making it work: the version IRG had was not up to date (difficult to update due to downloading issues), and people in IRG did not want to depend on software from an outside robotics lab due past experiences. Therefore we decided not to use OSCAR when developing code for the arm.

The second part of the project began by removing the arm from K9 and mounting it on a test platform. The electronics were modified so the arm could be connected to K10. At this time, we also started to develop standalone code for the arm, without dependencies on K9 rover code as before. When the arm was usable we started designing a simple end-effector. After the arm had been moved to the test platform we were able to test run the mechanics of the arm. It turned out there were a few issues that occurred when trying to move the arm. The first issue we noticed was the waist motor stalling or not moving at all sometimes. This happened when the arm was more than $\pm$ 90 degrees from zero position. The fault turned out to be the motor cable to the waist motor being partly cut. This problem manifested itself when the cable kept on touching the potentiometer gear. When turning the waist to positions far from zero position the cut in the cable would be dragged apart, while at an angle close to zero position the threads in the cut cable would be kept together and signals could go through to the motor. The broken cable was fixed by cutting of the broken part and attaching a new connector at the end. Another issue that occurred was that the waist motor would work but there was no movement on the actual link. This was a recurring problem during the project and we spent many hours trying to figure out what was wrong. When the waist motor was taken apart we found that the motor shaft was disconnected from the harmonic drive. This was fixed by applying Loc-Tite adhesive to the shaft to the drive. This solution added mechanical play of approximately 10 degrees in the waist. By now the project was in the final stage and we did not have time to look further into this issue. therefore we disassembled IRG's 3-DOF replica of the arm (missing the twist and wrist joint). From the new arm we retrieved an exact copy of the faulty motor block in the arm. With the new motor block mounted, the waist functioned properly, but an offset of 6 degrees was introduced. Because of a lack of time, we did not re-calibrate the zero position, but

just took this offset into consideration when reading and writing joint angles.

1.6 Thesis Outline

We will start by describing the system architecture in Chapter 2. Hardware used in this project will be introduced, except the manipulator itself which will be described later. We will start by introducing the K9 rover and then continue with the recent version of K10. Finally the communication brick that is the manipulation focus for this project will be presented. We will also talk about the software architecture used.

In Chapter 3 the manipulator will be described in detail with all components making the arm; the actuators and sensors. Finally we will give a description of the end-effector developed for the collection purpose.

In Chapter 4 the theory behind generation of a path for the end-effector to follow will be described. In this chapter we will also talk about the reason for choosing the trajectory generator we use and how we implement and use the path.

Control strategy used in this project is discussed in Chapter 5. We also derive an simplified model of the manipulator where each joint is considered to be a SISO system and each system controlled separately.

In Chapter 6 the experimental results will be reported.

Finally, in Chapter 7 the project will be summarized and possible future work will be discussed.

Chapter 2

System Architecture

In this chapter we will introduce the rovers linked to this project. Starting with the K9 rover on which the manipulator used to be mounted and moving on the the K10 rover where the arm will be mounted in the future. We will continue with describing the communication brick being the object for picking up. Finally we will talk about the software used and how we implemented the control.

2.1 K9 Rover

K9 rover is a Mars Exploration Rover (MER) prototype, built upon a FIDO-style JPL base. It was developed as a robotic test platform, a test bed for engineers to integrate and demonstrate new robotic technologies. K9 is a six-wheeled, solar-powered autonomous rover weighing 65 kg and measuring 1.6 m high. The rover carries a variety of instruments on board including a compass, an inertial measurement unit, and three pairs of monochromatic cameras used for navigation and instrument placement. It also carries a pair of high-resolution color stereo cameras and an arm mounted focusable microscopic camera (CHAMP). CHAMP is used to take samples of targets such as rocks, to determine mineralogy, to obtain microscopic images and to facilitate other operations needed to understand the planet's geology and search for evidence of past or present life.

2.2 K10 Rover

K10 is K9's successor and is the rover currently used for autonomous research within IRG. It is a 4-wheel drive rover with 4-wheel steering. K10's dimensions are width 0.9 m, length 1.0 m and height 1.3 m with sensor mast. It weighs 100 kg including 25 kg payload. The maximum speed the rover can reach in a 10 deg slope is 0.9 m/s. IRG has two K10 rovers; red and black, which are instrumented with dGPS, stereo cameras, compass and 2D laser scanner. The big difference between K9 and K10 is that K10 is manufactured mostly from commercial of-the-shelf parts. Its on-board computational resources consist of a dual-core 2 GHz laptop with 2 GB of RAM running Red Hat Enterprise Linux 5.

2.3 ADCR Relay Brick

The ADCR Relay Brick (henceforth referred to as communication brick) is a communication node which provides extended communications range, [Pezeshkian et al. 2007]. A few seconds after being launched it will automatically open up and self-right, extending the antenna. The

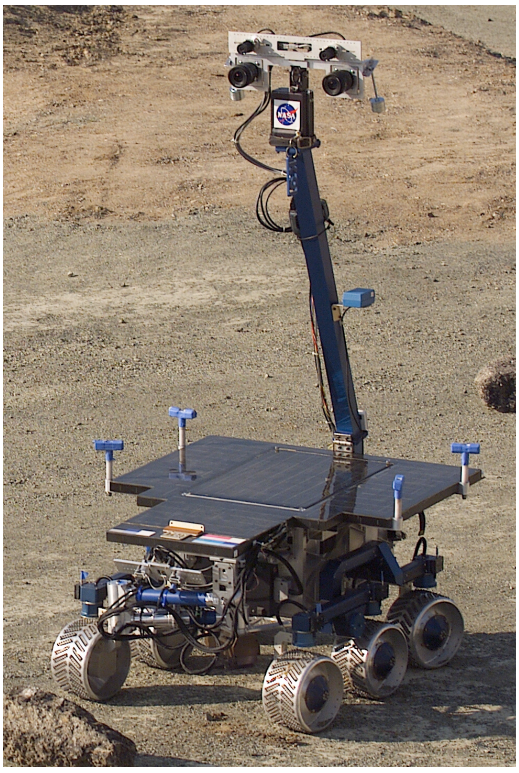

Figure 2.1. K9 rover (Courtesy NASA Ames).

Figure 2.2. K10 rover (Courtesy NASA Ames).

flaps and antenna mast are spring-loaded. The communication bricks were designed to allow the antenna to be collapsed back into the bay. This is done by folding the antenna, then folding the inner flap over the antenna and finally by closing the outer flap. The communication brick is box-shaped with size: width 82 mm, length 190 mm, height 66 mm and with the antenna mast folded out a height of 518 mm, see Figure 2.3. It weighs 0.5 kg which is far below the 2 kg designed payload for the arm.

As shown in Figure 2.3, the brick has a hole on the lower (to the left in figure) flap with dimension 25 mm and 40 mm. A basic approach strategy is to pick up the communication relay via the hole by using the simplest possible end-effector: a hook. Since this is a first pass approach to this problem, a simple hook made from a metal bar and bent into a hook-shape is adequate.

Figure 2.3. ADCR brick relay.

2.4 Software

2.4.1 CLARAty

CLARAty is a reusable robotic software framework (see [Nesnas, 2007, Nesnas, 2006]) developed collaboratively by four institutions: Jet Propulsion Laboratory, NASA Ames Research Center, Carnegie Mellon University and University of Minnesota. It stands for Coupled-Layer Architecture for Robotic Autonomy and was designed for improving the modularity of system

software. CLARAty defines interfaces for common robotic functionality and integrates multiple implementations of any given functionality.

2.4.2 Implementation

This project was implemented in C++ on a standalone Red Hat Enterprise Linux 5 computer. Implementation of the robotic arm uses the CLARAty architecture.

Chapter 3

Manipulator

In this project we will use the K9 arm (see Figure 3.1) and work towards integrating it with the K10 rover. This manipulator is a modified replica of an manipulator mounted on JPL's FIDO rover. The manipulator has five degrees of freedom (DOF). Since it is missing the sixth DOF it is not possible to reach all positions and orientations in the workspace by just moving the arm. To be able to reach them all, it might be possible to use the rover's 3-DOF's. For controlling the manipulator several Pic-Servo and BB boards are used (see hardware setup in Figure 3.4), one for each DOF. For position sensing, each joint has an incremental shaft encoder (see Section 3.2.2) and a potentiometer (see Section 3.2.1). The potentiometers are used for initialization of the joint positions while the encoders are used continuously after that and return a relative position to the initialized (start) position. This means that the potentiometer value is read only once and the encoder value every time after that.

A manipulator is composed of a set of joints separated in space by the arm links, where the arm base is called link 0 (in lower right corner in Figure 3.2) and the last link is the end-effector. The motion occurs in the joint while the link is of a fixed construction. Thus the links maintain a fixed relationship between the joints. Each joint represent one DOF. Sometimes a joint has n DOF, though it can be modeled as n joints of 1-DOF connected with *n-1* links of zero length. It is important to be able to specify the locations of the links with respect to each other. We associate each link i with a coordinate frame (x_i, y_i, z_i) fixed to that link. This is done using the Denavit-Hartenberg (DH) representation [Craig. 2002]. In order to deal with the complex geometry of a manipulator, we will affix frames to each link and then describe the relationship between each frame. The frame attached to link 0 is called the base frame.

There are two different sorts of robot joints, revolute and prismatic. The revolute joint allows rotary motions about an axis of rotation, while the prismatic joint allows an extension or telescopic motion. The joint axis of a revolute joint is the axis around which the rotation occurs. The right-handed screw rule yields the following: the curled fingers of the right hand indicate the direction of rotation while the thumb indicates the direction of the axis of rotation.

3.0.3 Hardware description

The manipulator in question has five revolute joints connected to each other through links. The joints are called waist, shoulder, elbow, twist and wrist. The waist is the joint connecting the arm base to the first link, see Figure 3.2. Arm measurements are stated in Table 3.1. Each joint is only 1-DOF.

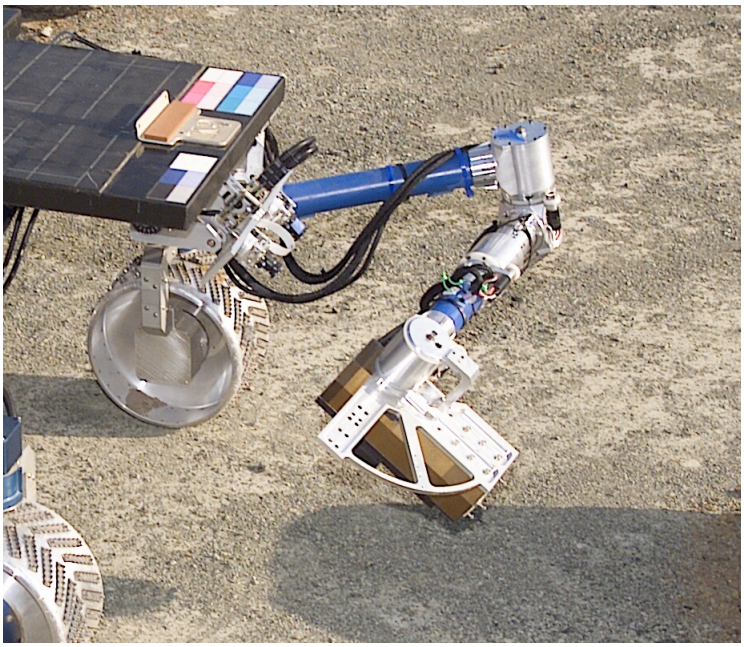

Figure 3.1. K9 Arm mounted on K9 rover (courtesy NASA Ames IRG).

Arm mass	2.35 kg
Designed for payload	2 kg
Total extended length (shoulder to end of end-effector)	0.7936 m
Distance arm mount to elbow	0.3302 m
Distance elbow to twist	0.140676 m
Distance elbow to wrist	0.311 m
Distance wrist to end-effector	0.1524 m
Max lift from shoulder	4.86 kg

Table 3.1. Arm measurements.

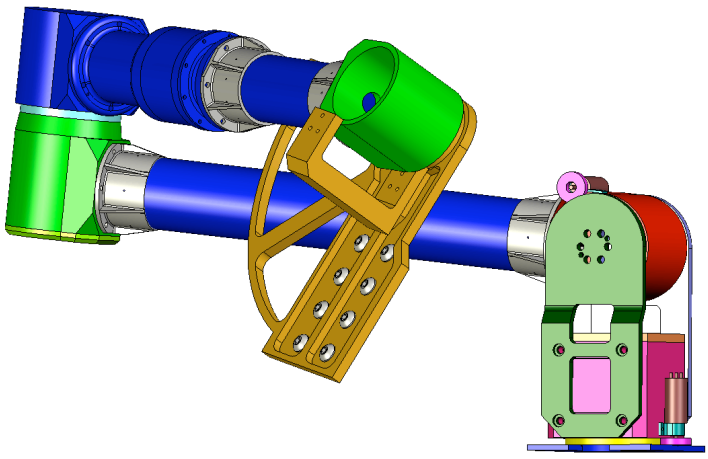

Figure 3.2. CAD drawing of the arm (courtesy NASA Ames IRG).

3.1 Actuator

The joint design uses a *MicroMo* motor with an integral encoder (encoder attached to the motor shaft) and a *MicroMo* planetary gearhead. The output stage is a harmonic drive: a three piece elliptical spline system. A harmonic drive is an extremely precise, zero backlash speed reduction system. It is light weight, compact, and has considerable torque for its size. The input stage is a wave generator. The circular spline is always attached to the housing. This housing also has bearings for the input stage. The output of the joint is always connected to the flexspline. A harmonic drive is an input/output gear reduction mechanism. Very high reduction ratios are possible in a small volume. Some of the joints have a gear pair between the gear head and the harmonic, this is either bevel or spur. For the relevant data sheet see [MicroMo Motor 1219G].

3.2 Sensors

In this section the two different position sensors will be described.

3.2.1 Potentiometer

The potentiometer is a type of bridge circuit that provides position information by varying resistance according to position. By applying a voltage across the bridge and measuring the output voltage in between, position can be deduced. When initializing the manipulator, the potentiometer value is read 100 times and a mean value is calculated and returned as the initialized start joint position.

3.2.2 Encoder

Each joint has one MicroMo magnetic encoder, described in [MicroMo Magnetic Encoder]. Magnetic encoders are, compared to optical ones, highly reliable and have high performance [Winter]. Their sensors detect variations in magnetic field embedded in the rotor. It does not produce errors due to contamination such as oil, dirt or water, because those contaminants do not affect the magnetic field. For position sensing the encoder uses a solid state Hall sensor and a low inertia magnetic disc. The disc turns with the shaft being measured, with alternating north and south poles around the circumference of the disc. The Hall effect sensor works by detecting a change in voltage caused by magnetic deflection of electrons. It provides a quadrature output, meaning two channels with 90 ° phase shift, illustrated in Figure 3.3. The encoder values are more precise than the values read from the potentiometer but do not provide absolute position information.

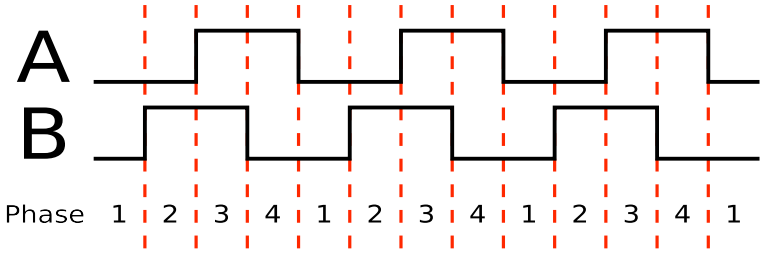

Figure 3.3. Two square waves in quadrature (Courtesy Wikipedia).

3.3 Communication Boards

In this section the Pic-Servo board and a BB board used for controlling the manipulator will be described.

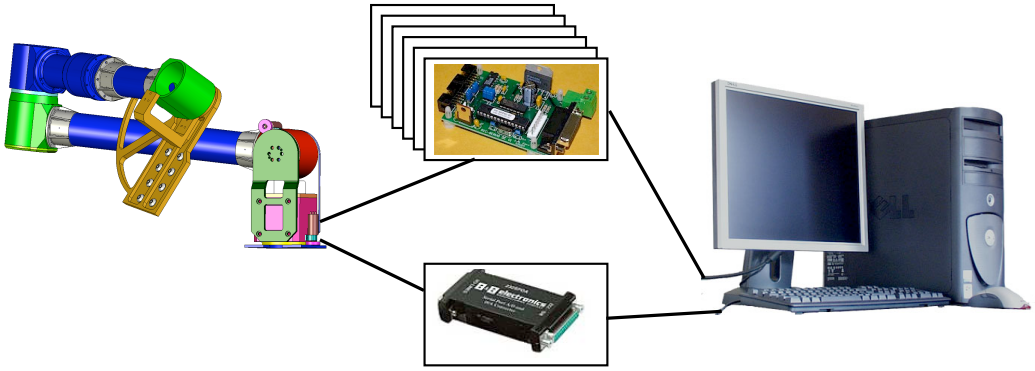

Figure 3.4. Arm hardware setup.

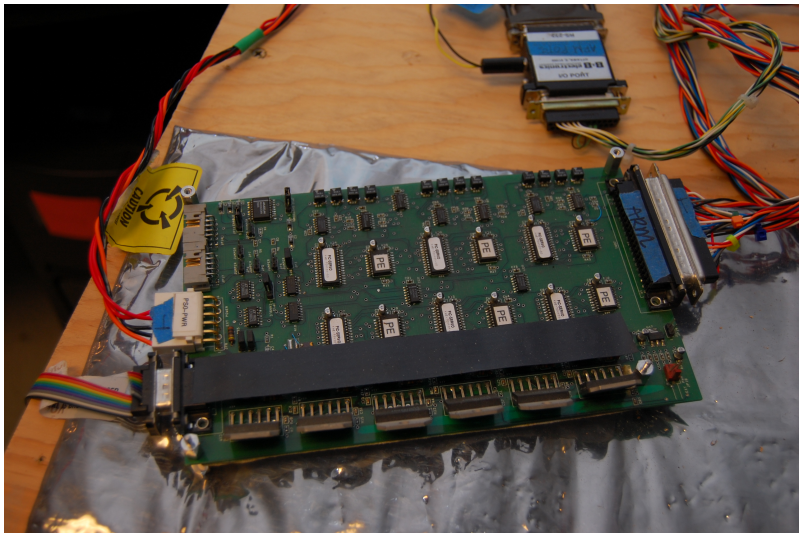

Figure 3.5. The two communication boards used; Pic-Servo board in the middle and BB board in the upper right corner.

3.3.1 Pic-Servo Board

The *J.R.Kerr* Pic-Servo motion control board is a complete motor servo control system used for DC-motors with incremental encoders feedback. The Pic-Servo board is connected to two power sources; 5 V and 12 V. Where the motors are run on 12 V and the logic, which process the encoder readings, use 5 V. Most applications use the Pic-Servo's serial interface to send motion control commands to the manipulator. The Pic-Servo boards has five different operating modes covering a variety of servo control applications including: PWM (torque) output mode, velocity mode, trapezoidal mode, coordinated motion control (CMC) mode and step & direction mode. In this project we will use the PWM mode.

PWM mode

A PWM signal is a square wave of varying duty cycle where a PWM value of 255 corresponds to 100% and a value of 0 corresponds to 0%. The PWM value is derived as follows: if you are using a 12 V motor powered by 12 V (as in this case), the PWM value, or output limit would be set to $\frac{255}{12/12} = 255$. In PWM mode, which is the lowest layer of control, the user can specify the torque output signal sent directly to the amplifier. When sending a square wave to the motor, what actually happens is that the motor is turned on and off in conjunction with to the signal.

3.3.2 BB Board

The BB board is an A/D converter, converting analog signals to digital values and then communicating them to the serial port and vice versa. It is used for converting the potentiometer values read from the joints. The BB board used for the manipulator is from *B&B electronics* and is powered by 12 V.

3.4 End-Effector

An end-effector is the part on a human body that correspond to a hand. In robotics it is desirable to manipulate objects in an environment using the end-effector in the same way humans would use their hands. Usually a robot has one end-effector per manipulation action. It is analogous to a human using an appropriate tool for the task.

In Section 2.3 we discussed the pick-up strategy for collection of the communication bricks. We will use the hole on the communication brick's lower flap. This is a the simplest possible method to pick up the brick and we decided to use it as a first solution. Constraints placed upon this project made it desirable to construct an inexpensive and simple end-effector. These facts combined with the hole placed on a spring-loaded flap on the brick resulted in a hook end-effector design. For this, a metal bar with width of 12.7 mm (measuring half of the hole width) was bent into hook shape and shortened to the appropriate length. The hook was mounted on the tool palette, attached to the wrist joint. The tool palette has the advantage that different tools can be mounted on the arm at the same time, i.e. in the future it might be interesting to mount the end-effector together with e.g. the CHAMP.

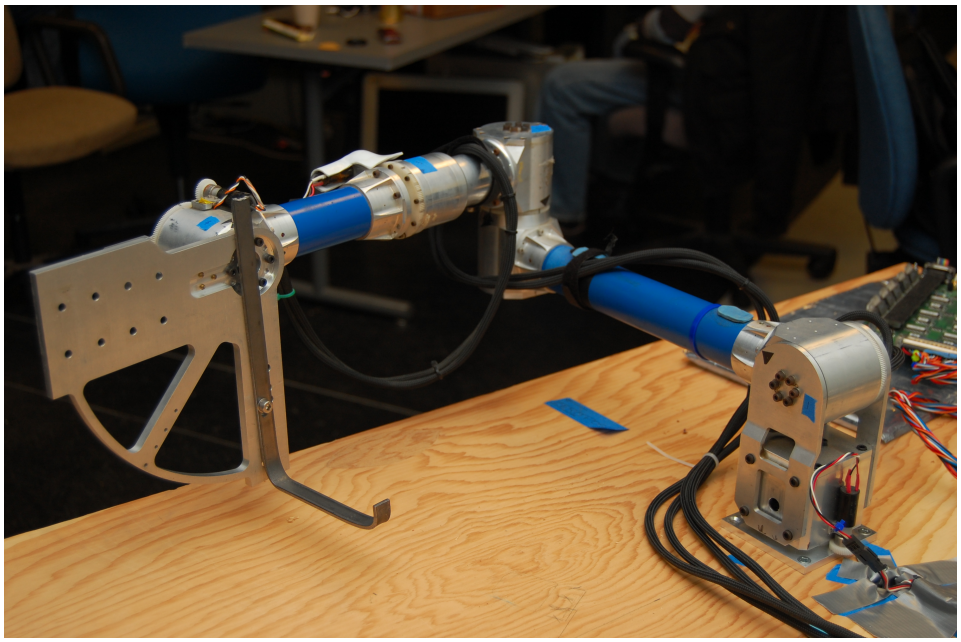

Figure 3.6. End-Effector.

Chapter 4

Trajectory Generation

This chapter introduces trajectory generation and the strategy used for calculating paths for the manipulator to follow. There are many different strategies used for trajectory generation. We will discuss one basic approach that was used as a first solution.

4.1 Introduction

Trajectory generation, or path planning, is a term used in robotics for computing a trajectory that describes the desired motion of a manipulator in multidimensional space. Though motion planning is simple for humans, it is one of the most challenging tasks in robotics. The problem is to create an algorithm that would be able to find its way around a room with obstacles, perhaps accomplishing some tasks on the way. The goal of trajectory generation is to find the best path from point A to point B given an environment that may contain obstacles. Algorithms are used in a large variety of fields, such as autonomous mobile robotics, video games and robotic surgery. The word trajectory refers to a time history of position, velocity and acceleration for each degree of freedom. The system user specifies the desired goal position and orientation of the end-effector, leaving for the system to decide on the exact shape of the path to get there, including computation of the duration and the velocity profile.

We were considering motions of the manipulator as motions of the tool frame, {T}, relative to the station frame, {S}, [Craig. 2002]. This is exactly the same way a system user would think of it when seeing a manipulator move according to a trajectory. The basic problem is to move the manipulator from initial position to a desired final position. Remember that we actually want to move the tool frame from its current value, $\{T_{initial}\}$, to a desired final value, $\{T_{final}\}$. The motion plan encapsulates both movement in position and movement in orientation of the tool relative to the station. Sometimes it is interesting to specify the motion in more detail, e.g. by defining a sequence of desired via points (intermediate points) between initial and final position. When the tool follows a more constrained trajectory, it must pass through a set of positions and orientations as described by the via points. The generated path including initial and final points plus all via points are called path points. Here, the word point refer to a frame that specify both the position and orientation of the tool relative to the station.

The motion of the manipulator is desired to be smooth. The reason for this is that rough motion will increase wear on the mechanics and cause unnecessary vibration. To avoid this, we need to put some constraints on the spatial and temporal qualities of the path between the via points. [Craig. 2002] defines a smooth function that is continuous and has a continuous first derivative.

There are two different kinds of trajectories; one specified in joint space and another specified in cartesian space. The word joint space refers to the fact that the end-effector position is defined

with joint angles (one angle per degree of freedom), while in cartesian space it is defined with position (x,y,z) and orientation (pitch, roll, jaw). Remember that transformations between joint space and cartesian space is performed through forward kinematics, and the reverse, going from cartesian to joint space, with inverse kinematics. The calculations involved in inverse kinematics are more difficult than forward kinematics, and depending on the setup of the manipulator it might be impossible to find a solution. The inverse kinematics in the original K9 arm code was not working for an arbitrarily chosen position and orientation in cartesian space, therefore this project will only consider a method in joint space, because of a lack of time deriving the inverse kinematics for the setup and it is usually the easiest to compute and requires fewer resources.

4.2 Theory

A joint space trajectory is generated from desired start and end angles, and is expressed as a set of functions that describe the angle of a joint given the time since start of motion. In joint space, each path point is a set of joint angles. The desired joint angle function for a particular joint does not depend on functions for the other joints. A smooth function is found for each of the N joints, which pass through the via points and end at the goal position. The time required for each segment is the same for each joint so that all joints will reach the via point at the same time. We will discuss a cubic spline where only initial and final points are defined.

4.2.1 Cubic Spline

A cubic spline is a third order polynomial. The initial and final positions are known in the form of a set of joint angles. From these positions we calculate a function for each joint whose value at t_0 is the initial position of the joint and at t_f is the desired final position of that joint. In the process of making a smooth motion, four constraints on the function are applied. Two constraints come from the selection of desired initial and final position:

$$\begin{aligned} \theta(0) &= \theta_0 \\ \theta(t_f) &= \theta_f \end{aligned} \tag{4.1}$$

A continuous velocity function is desirable, as with the other two constraints, which in this case means that the initial and final velocity are zero:

$$\begin{aligned} \dot{\theta}(0) &= 0 \\ \dot{\theta}(t_f) &= 0 \end{aligned} \tag{4.2}$$

These constraints specify a unique cubic spline with the form

$$\theta(t) = a_0 + a_1 t + a_2 t^2 + a_3 t^3 \tag{4.3}$$

From Equation (4.3) joint velocity and acceleration are derived

$$\begin{aligned} \dot{\theta}(t) &= a_1 + 2a_2 t + 3a_3 t^2 \\ \ddot{\theta}(t) &= 2a_2 + 6a_3 t \end{aligned} \tag{4.4}$$

Combining Equations (4.3) and (4.4) with the four constraints in Equations (4.1) and (4.2) (this step can be found in [Craig. 2002]) and solving these equations for a_i, we obtain

$$\begin{aligned} a_0 &= \theta_0 \\ a_1 &= 0 \\ a_2 &= \frac{3}{t_f^2}(\theta_f - \theta_0) \\ a_3 &= -\frac{2}{t_f^3}(\theta_f - \theta_0) \end{aligned} \tag{4.5}$$

Using Equation (4.5) in Equations (4.3) and (4.4) yields the cubic spline that connects the initial position with the desired final position.

4.3 Implementation

A first trajectory generation approach was to generate three separate cubic splines, connecting start position with final position, see Figure 4.1. At every transition point between the splines the manipulator comes to a stop. Velocity is set to zero at the transition points because at these points the end-effector is really close to the ground (about 10 mm) and the manipulator can easily hit and break if control of the arm is not working properly.

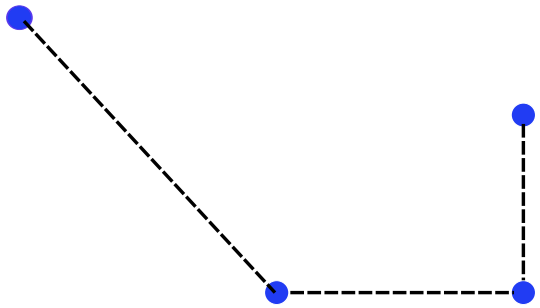

Figure 4.1. Trajectory with three segments (in 2D).

In Figure 4.2 the trajectory with position and velocity for waist joint has been illustrated, where each via point has been marked with a red dotted line. The first 5 seconds are used for the manipulator to move from start position to the first via point. At this point the manipulator stops, and then continues using the next 5 seconds to move to the second via point. Now the tip of the hook is straight underneath the hole on the communication brick's flap. The next step is to maneuver the manipulator upwards so the end-effector hooks onto the hole. The end-effector has reached the final position. The last trajectory brings the brick to a safe carrying position. In this position the antenna is sticking straight out from the arm, a safe distance from the ground and the rover, so neither part can be hit and damaged. When the manipulator has reached carrying position, the rover will start moving towards the base camp and hand the communication brick over to an astronaut. Since the rover is moving in rough terrain the choice of carrying position needs to be carefully selected.

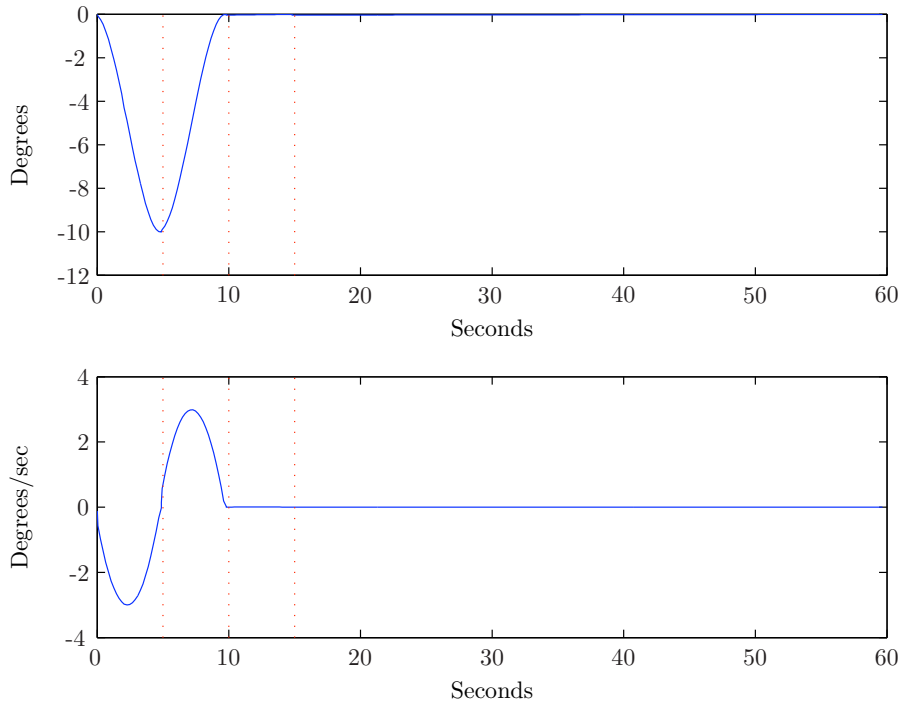

Figure 4.2. Trajectory with position and velocity for waist joint, where each transition point has been marked with a red dotted line.

Chapter 5

Control Strategy

In this chapter we will introduce the control strategy used when moving the manipulator. We will also discuss the model of the manipulator leading to the control algorithm used for moving the arm from point to point, using a basic PD controller.

5.1 Introduction

A manipulator is a multi-input, multi-output (MIMO) control system. It is desirable to take the simplest approach and construct a control system by treating each joint as a separate system to be controlled. In this type of control each joint of the manipulator is controlled as a single-input, single-output (SISO) system. The N-DOF manipulator will be considered as N independent SISO control systems. This control approach is an approximate method and is considered to be good enough for many manipulation purposes since the joints are not independent but rather highly decoupled. The SISO model approximation works well for manipulation not involving very fast motion, especially in robots with large gear reduction between the actuators and the links. The manipulator used during this project is rather slow and has a large gear reduction and therefore it is possible to use this approximation. A manipulator is a nonlinear control system, but can be approximated with a linear model. This is often a reasonable approximation since the large gear reduction decreases the nonlinear part of the model. [Craig. 2002, Spong et al. 2006, Lewis et al. 2004, Franklin et al. 2002, Kurfess. 2005, Mittal and Nagrath. 2003]

The manipulator has at each joint, an actuator to apply torque on the neighboring link and also sensors (both a potentiometer and an encoder) to measure the joint angle. We want the manipulator joints to follow desired position trajectories commanding the actuators in terms of torque. This is done with a control system for each joint that compute actuator commands that will realize this desired motion. An overview of this control system is shown in Figure 5.1, where position, velocity and acceleration of θ_d are input to the system. From measured system output θ, the velocity is calculated, and both are used for calculating the error in position and velocity, while comparing the trajectory θ_d with the measured θ, we obtain two error functions, one in position, $e = \theta_d - \theta$, and one in velocity, $\dot{e} = \dot{\theta}_d - \dot{\theta}$. This is fed into the controller, which calculates the torque needed to move the joint according to the desired trajectory. As mentioned before, the manipulator used in this project has slow motion joints and therefore there is no need to model dynamic coupling between joints since it is not cost effective. [Platt. 2007]

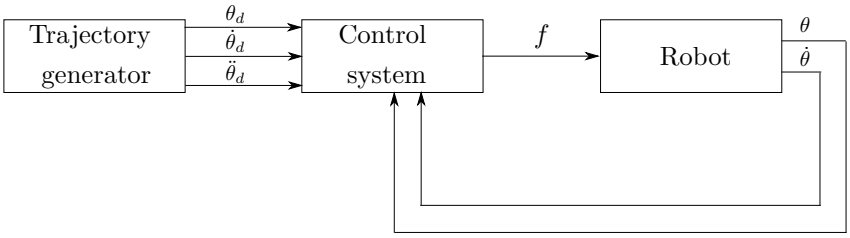

Figure 5.1. Overview of Control System.

5.2 Model of a Single Joint

A few assumptions were made when developing a simplified model of each manipulator joint. These assumptions led to a second-order linear system. Each assumption will be presented further down in this section as they occur in the model process. The modeling of a manipulator joint has been divided into three parts; motor-armature inductance, effective inertia and unmodeled flexibility.

5.2.1 Model Components

Actuator

The actuator used is a direct current (DC) motor. A DC-motor has two parts; a fixed stator and a movable rotor. The stator consists of a housing, bearings and magnets, which establish a magnetic field across the stator. The torque causing the rotor to rotate is the motor torque constant which relates armature current to the output torque as

$$\tau_m = k_m i_a \tag{5.1}$$

where τ_m is the motor torque, k_m the physical constant and i_a the armature current.

Whenever a motor is rotating, a voltage, v_a is developed across the armature. It will tend to oppose the current flow in the conductor. A second motor constant, the back emf constant, describes the voltage generated for a given rotational velocity:

$$v_b = k_e \dot{\theta}_m \tag{5.2}$$

where v_b is the back emf, $\dot{\theta}_m$ the angular velocity of the rotor, and k_e the proportionality constant.

The motor circuit, where the components are; the voltage source, v_a, the inductance, l_a, the resistance, r_a, and the generated back emf, v_b. The circuit can be described with the following first-order differential equation.

$$l_a \dot{i}_a + r_a i_a = v_a - k_e \dot{\theta}_m \tag{5.3}$$

For simplifying this equation we introduce the first assumption, where the inductance is neglected. This means we will assume that the actuator acts as a pure torque source that we command directly using a scale factor.

Effective Inertia

The rotor is connected through a gear reduction to an inertial load, where index m indicate that we are talking about the motor side of the mechanics. The motor torque, τ_m, is the torque applied to the rotor given by Equation (5.1) as a function of the current i_a flowing in the circuit.

The gear ratio, η, causes increased torque seen on the load and a decreased speed of the load, given by

$$\begin{aligned} \tau &= \eta\tau_m \\ \dot{\theta} &= \frac{1}{\eta}\dot{\theta}_m \end{aligned} \tag{5.4}$$

where $\eta > 1$. The equation describing the mechanical system in terms of torque at the rotor is

$$\tau_m = \left(I_m + \frac{I}{\eta^2}\right)\ddot{\theta}_m + \left(b_m + \frac{b}{\eta^2}\right)\dot{\theta}_m \tag{5.5}$$

where I_m and I are the inertias of the motor and the load, and b_m and b are damping coefficients for the rotor and load bearings. Equation (5.5) can be rewritten as

$$\tau = \left(I + \eta^2 I_m\right)\ddot{\theta} + \left(b + \eta^2 b_m\right)\dot{\theta} \tag{5.6}$$

where $I + \eta^2 I_m$ is the effective inertia at the load side and $b + \eta^2 b_m$ is the effective damping. The inertia of a joint varies with configuration and load. However, for a highly geared joint, i.e. $n >> 1$, the inertia of the rotor is an extensive portion of the combined effective inertia and the variations of inertia due to configuration and load are small. This leads to the second assumption because the effective inertia only consists of the rotor inertia, which is a constant.

Link and Joint Flexibility

The last assumption used is that gearing, shaft, bearings and the driven link are not flexible. If the system is sufficiently stiff, the natural frequencies of these unmodeled resonances are very high and can be neglected. Since we have chosen not to model flexibilities in the system, we must be careful not to excite these resonances. [Craig. 2002] states that if the lowest structural resonance is ω_{res}, then we must limit the natural frequency with

$$\omega_n \leq \frac{1}{2}\omega_{res} \tag{5.7}$$

This relationship provides some guidance on how to choose gains in a controller. Increasing gains leads to faster response and lower steady-state error.

5.2.2 Model in Simulink

Equations (5.1) - (5.5) can be presented in a block diagram. Deriving from the block diagram a transfer function:

$$\frac{\theta}{v_{ref}} = \frac{1}{\eta s}\frac{k_m}{(l_a s + r_a)(I_m s + b_m) + k_b k_m} \tag{5.8}$$

where $v_{ref} = k_a v_a$.

5.3 Control System

When choosing a control algorithm for controlling the manipulator, the system was approximated to be a linear system where a linear control method can be implemented. According to [Craig. 2002], it is desirable to design a controller where it has been partitioned into a model-based portion and a servo portion. The reason for this is that the system's parameters (m, b and

k) only appear in the model-based portion, while the servo portion is independent from these parameters. A controller with trajectory following is illustrated in Figure 5.2.

$$f' = \ddot{x}_d + k_v\dot{e} + k_p e \tag{5.9}$$

Each joint was considered to be a separate control system and controlled with a basic PD controller. For design we need to simplify the system by making the three assumptions introduced earlier in this chapter; the motor inductance l_a can be neglected, modeling the effective inertia as a constant and structural flexibilities are neglected. With these assumptions, a partitioned controller can be derived as

$$\begin{aligned} \alpha &= I_{max} + \eta^2 I_m \\ \beta &= (b + \eta^2 b_m)\dot{\theta} \\ \tau' &= \ddot{\theta}_d + k_v\dot{e} + k_p e \end{aligned} \tag{5.10}$$

where the gains are chosen as following.

$$\begin{aligned} k_p &= \omega_n^2 = \frac{1}{4}\omega_{res}^2 \\ k_v &= 2\sqrt{k_p} \end{aligned} \tag{5.11}$$

This controller has been illustrated in Figure 5.2. Results from when using this controller will be presented in the next chapter.

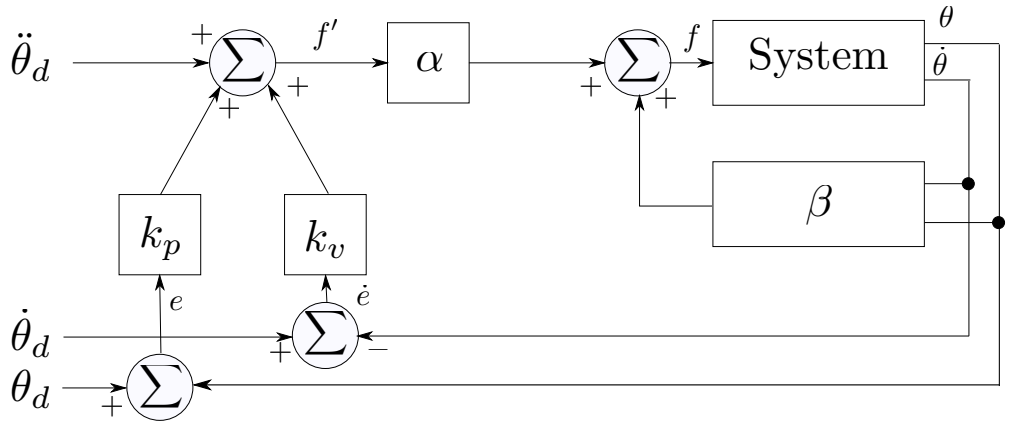

Figure 5.2. A trajectory following controller.

Chapter 6

Results

6.1 Test bed

During testing the manipulator is mounted on a wooden plate with the waist rotating around the axis perpendicular to the wooden plate. The manipulator is setup with the same hardware that will be used when it is mounted on the rover, except for the computer which is different but also runs Linux Red Hat 5.

6.2 Test setup

The manipulator is moved according to the pre-calculated trajectory. The accuracy of the movement compared to the trajectory is defined in the following section. The trajectory studied does not include lifting the arm to carrying position, so the studied range in effect spans 15 seconds.

6.3 Performance metric

For performance metric, we use average absolute error.

$$\sum |\theta_d - \theta| \tag{6.1}$$

This is calculated for each joint separately, summarizing position errors at every reading point from start position to lift position. It is desirable to get all five residual errors as small as possible and also have the joints move smoothly. The velocity (derivative of the position) is calculated. Note that a modification in control parameters for one joint will change slightly how the other joints behave, i.e. the residual error for all joints will change. We do not measure smoothness, but rather look for information in illustrated results.

6.4 Parameter choice

Nowadays industrial robots according to [Craig. 2002] only have the model-based portion implemented, this means $\alpha = I$ and $\beta = 0$ in Equation (5.10). This will be our first approach to a control strategy. Starting off with parameter values chosen by using the rule-of-thumb values mentioned in 5.3. Those parameter values were tuned to improve the fitting of the position curve to the trajectory curve. When tuning, the interesting parameters to look at are the residual value and the smoothness of the position curve. The tuning is performed using residual error

as described in Section 6.3. The smoothness criteria is used to reject certain configurations, that cause oscillations or jerkiness.

6.5 Data

Our first choice of controller values (the rule-of-thumb controller) produces the result illustrated in Figure 6.1 where joints are drawn in order from waist in position one and down to wrist in position five. In Figure 6.1 the reached position curve is drawn with a continuous blue line while the desired trajectory curve is drawn with a red dotted line. When looking at the position error, illustrated in Figure 6.2, we can see that the first four joints (waist, shoulder, elbow and twist) follow the desired trajectory well. It is also clear from looking at the residual error in Table 6.1. The last joint, wrist, follows the trajectory well for the first period of time and then start to oscillate around the path curve. Figure 6.2 shows oscillations starting at 20 seconds. The result of the control indicated this was a good starting point for improving the control of the arm, and with some tuning it should be possible to decrease the residual error and increase the smoothness and accuracy of the curve.

Joint	K_p	K_v	Residual error
waist	5102	$2\sqrt{K_\mathrm{p}}$	0.2451
shoulder	5102	$2\sqrt{K_\mathrm{p}}$	0.5584
elbow	5102	$2\sqrt{K_\mathrm{p}}$	0.6390
twist	3086	$2\sqrt{K_\mathrm{p}}$	0.7531
wrist	5102	$2\sqrt{K_\mathrm{p}}$	0.0343

Table 6.1. K_p and K_v chosen by rule-of-thumb and its corresponding residual error.

An improvement to make on this controller was to decrease the oscillation appearing on the wrist curve at 20 seconds. When tuning the controller the proportional value K_p was modified manually. A change in value also modified the value of K_v since the derivative gain is proportional to K_p, see Equation (5.11).

When tuning, remember the impact the different parts of the PD controller has on the system. An increased proportional value ($\mathrm{K_p}$) yields an increasing speed on the system and also decreased stability. The proportional part uses the current error and controls the system using this. An increased derivative value ($\mathrm{K_p}$) improves the stability through decreasing the speed of the system. The derivative part of the controller predicts future errors and controls the system using these errors.

Tuning was performed by decreasing K_p value for the wrist joint, as stated in Table 6.2. The reason for deciding to decrease K_p and also K_v was to slow the system down and therefore gain an increased stability. Tuning resulted in a better behaving wrist joint controller, see Figure 6.3, which follows the desired trajectory in a satisfactory way. As shown in Figure 6.4 oscillations in wrist have disappeared and the controller works as intended.

6.6 Trajectory

In Chapter 4 when generating a trajectory we talk about separating the path into three segments. The result of this decision was a safe trajectory for the manipulator to follow. At the closest distance to the surroundings the manipulator was only 5 mm away from hitting the wooden plate on which the communication brick was placed.

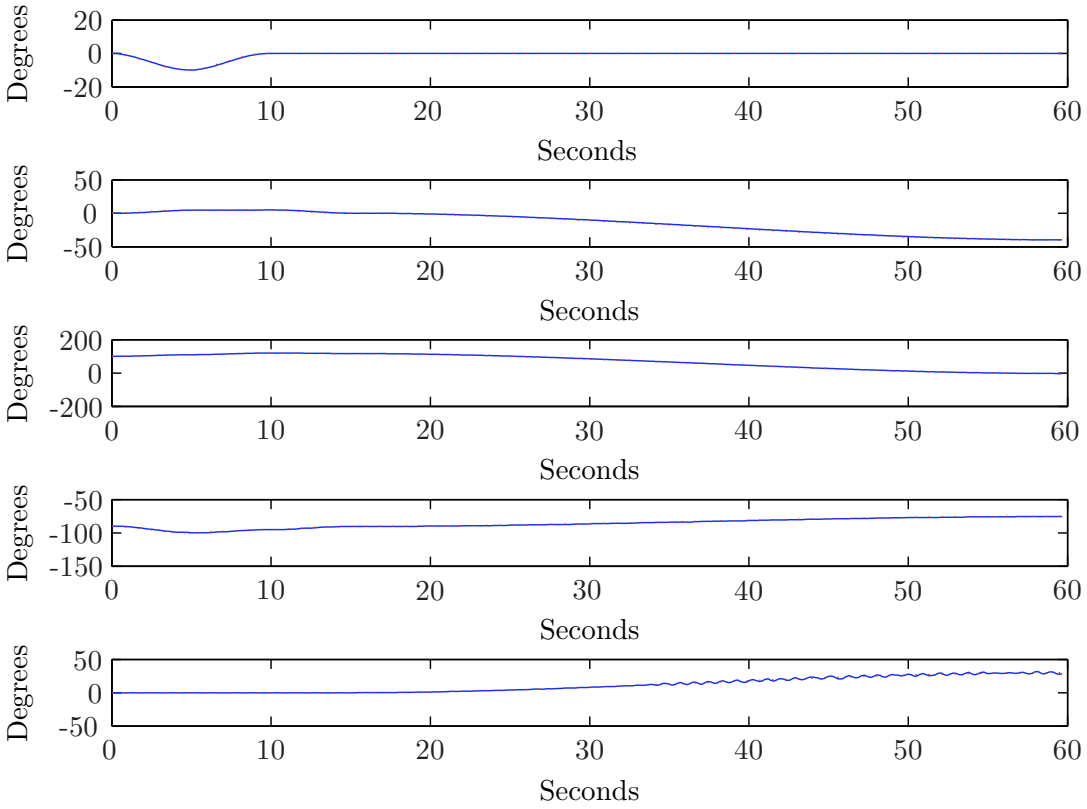

Figure 6.1. Reached position trajectory (blue line) and planned path (red dotted line) for rule-of-thumb controller.

6.7 Hardware Result

The main focus of this project was to implement a working controller, but the fulfillment of this goal depends on the hardware used. As mentioned before the manipulator have slow motion joints. Therefore the approximated model mapped onto a PD controller was good enough since the limitations in the system were in the hardware and not the controller.

During the project we had a couple of issues with faulty hardware. One issue was the waist motor stalling or not moving. The fault turned out to be the motor cable being partly cut and was fixed by removing the broken part of the cable and mounting a new connector. Another issue was the waist motor would run but no visible movement on the corresponding link. The reason turned out to be the motor shaft was disconnected from the harmonic drive. This was fixed by applying Loc-Tite adhesive to the shaft. The last issue with the arm was a 10 degrees play in the waist joint. Since this happened during the last weeks of the project we decided to swop the motor with the waist motor from a 3-DOF replica. With the new joint the manipulator was working properly and the testing could be finished off.

6.8 Summary of Experimental Results

The controller with rule of thumb tuning performs well except the wrist joint, which oscillates. The controller was tuned by decreasing K_p value for the wrist joint. The reason for deciding to decrease K_p was to slow the system down and therefore gain an increased stability. Tuning resulted in a better behaving wrist joint controller which follows the desired trajectory in a

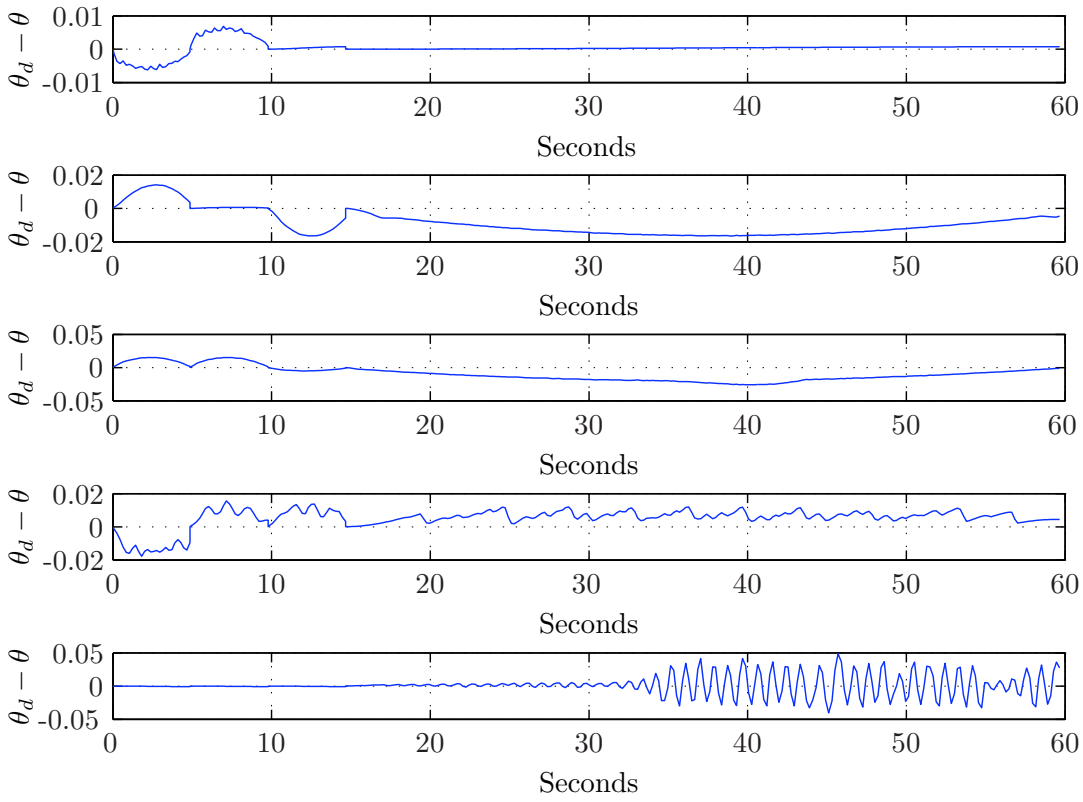

Figure 6.2. Position error for each time sample, for rule-of-thumb controller. At 20 seconds wrist joint starts oscillating.

Joint	K_p	K_v	Residual error
waist	5102	$2\sqrt{K_\mathrm{p}}$	0.2284
shoulder	5102	$2\sqrt{K_\mathrm{p}}$	0.5367
elbow	5102	$2\sqrt{K_\mathrm{p}}$	0.6122
twist	3086	$2\sqrt{K_\mathrm{p}}$	0.7306
wrist	1020	$2\sqrt{K_\mathrm{p}}$	0.0202

Table 6.2. K_p and K_v chosen by rule-of-thumb and its corresponding residual error.

satisfactory way. Oscillations in wrist have disappeared and the controller works as intended.

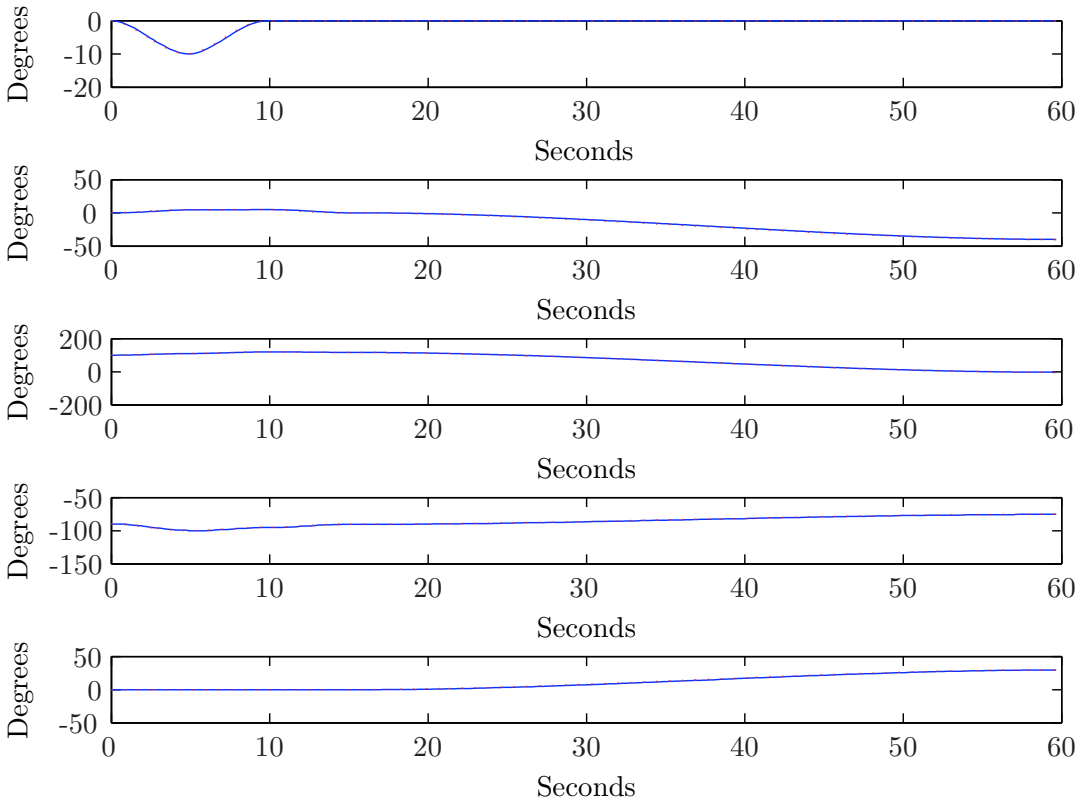

Figure 6.3. Reached position trajectory (blue line) and planned path (red dotted line) for tuned controller.

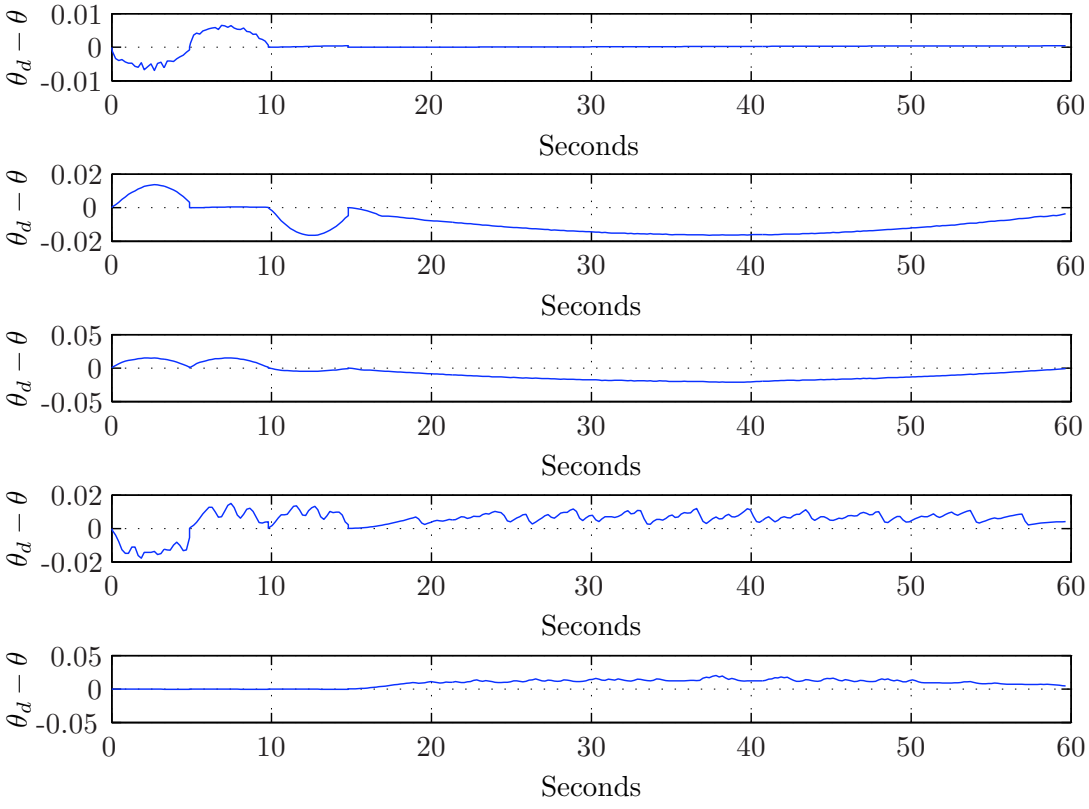

Figure 6.4. Position error for each time sample, for tuned controller. Oscillation on wrist joint at 20 seconds is gone.

Chapter 7

Summary

This chapter summarizes achievements in this project, and concludes the thesis.

7.1 Achievements

During the project we have achieved to setup the arm hardware to be in a state ready for experiments. This included both mounting the manipulator to a wooden plate, connecting cables and soldering connectors onto cables. The manipulator broke down a couple of times when driving the arm and performing testing. We fixed the issues as they occurred. It included repairing and changing broken cables, applying Loc-Tite to the disconnected motor shaft and harmonic drive and switching a faulty waist joint with a working one. We revived a subset of manipulator driver code from the K9 code base, and added functionality needed to carry out the investigations. Provided insight into the process of making the manipulator code independent of CLARAty framework. The time needed to get the manipulator code independent from CLARAty was approximated to take too much time from the actual thesis, and therefore a small amount of work towards getting arm code CLARAty independent was put in.

We planned a trajectory from point to point in joint space, using a smooth spline path. The path was separated into three segments, where the manipulator came to rest at each transition point. We implemented a path generation/tool use strategy that successfully picks up a communication brick in a known position. The nonlinear MIMO system was approximated with five linear SISO systems. This is a valid approximation for this manipulator since it is instrumented with slow joints with high gear reduction. We applied a linear model of a single joint, simplified to map onto a PD controller. We implemented the controller and tuned it looking at the stability and residual error.

In summary, the arm and its software are in a state where others can build upon it with further experiments using the experimental platform, and it achieves the goal of picking up a communication brick in a known position.

7.2 Conclusion

During the project we have worked towards fulfilling the objectives (stated in Section 1.3) put forth in the beginning of the project. We will go through the objectives and discuss the outcome to see if we fulfilled them all.

The first objective said that we should establish baseline communication with the manipulator. This was done so we could move the manipulator a desired amount of ticks using a software for the Pic-Servo board and BB board.

The next goal was to develop a standalone control software. We implemented a software using the CLARAty code base. This software is working and the result is discussed in Chapter 6.

The third objective was to design a simplified end-effector. The solution involving the bending of a metal bar was simple and low in price, $6 which was within the IRG price range for this project.

Finally the last obligatory objective was to write a controller for collection of the communication bricks with the end-effector attached to the end of the manipulator. With these four objectives fulfilled we have a working controller for the purpose of this project.

The two last objectives, which were optional, were not considered due to lack of time.

The next chapter suggests directions for future work.

Chapter 8

Future Work

There are a couple of different focuses for future work on this project.

A more advanced end effector needs to be designed, possibly together with a better grip on the communication brick. Such an end effector would need to securely hold the node, even while the rover is traversing terrain.

Obstacle avoidance and other considerations need to enter into the calculation of arm trajectories.

The rover needs to find the communication brick in the first place, by means of cameras or other sensors.

There are more advanced controllers that might be beneficial, though the current algorithm does meet the performance goals in this regard. Examples include complex state-space controllers modeling the entire arm, and control in cartesian space around the target tool position.

Replacing the software dependencies to CLARAty with an IRG written code base is a possibility. This is interesting because CLARAty is not up to date and therefore IRG would like to stop using it. K10 is built with less CLARAty then its predecessor the K9 rover and it would make sense if this was also the case with the manipulator.

Complete integration with the K10 code base would open up for further real world tests and development.

Bibliography

[Nesnas, 2007] Nesnas, I. (2007). Claraty: A collaborative software for advancing robotic technologies. In NASA Science and Technology Conference.

[Nesnas, 2006] Nesnas, I. A. D. (2006). CLARAty : improving software reliability for robotic space applications.

[Craig. 2002] Craig, J. 2005, *Introduction to Robotics: Mechanics and Control*, third edition, Pearson Prentice Hall, Upper Saddle River, New Jersey.

[Franklin et al. 2002] Franklin, G., Powell, J., Emami-Naemi, A. 2002, *Feedback Control of Dynamic Systems*, fourth edition, Pearson Prentice Hall, Upper Saddle River, New Jersey.

[Kurfess. 2005] Kurfess, T. 2005, *Robotics and automation handbook*. CRC Press, Boca Raton, Florida.

[Lewis et al. 2004] Lewis, F., Dawson, D., Abdallah, C. 2004, *Robot Manipulator Control: Theory and Practice*, second edition, Marcel Dekker, Inc, New York.

[Mittal and Nagrath. 2003] Mittal, R., Nagrath, I. 2003, *Robots and Control*, Tata McGraw-Hill.

[NASA Mars Rovers] NASA webpage about Mars Rovers. <http://marsrovers.jpl.nasa.gov/home/index.htm

[Pezeshkian et al. 2007] Pezeshkian, N., Nguyen, H., Burmeister, A. 2007, Unmanned ground vehicle readio relay deployment system for non-line-of-sight operations, 13^{th} *IASTED International Conference on Robotics & Applications*, August 29-31, Würzburg, Germany.

[Platt. 2007] Platt, R. 2007, *Introduction to Robotics*, Lecture notes, Rice University, US. http://www.owlnet.rice.edu/∼mech498/.

[Spong et al. 2006] Spong, M., Hutchinson, S., Vidyasagar, M. 2006, *Robot modeling and control*, John Wiley & Sons, Inc, United States of America.

[Winter] Winter, B. Lower Cost Magnetic Encoders Solve Encoder Problems for Motor Manufacturers, Avatron Manufacturing, Inc. <http://www.avtron.com/pdf/encoders/mag-encoders-solve-prob.pdf>.

Data sheet

[MicroMo Motor 1219G] MicroMo Electronics, Motor series 1219G. <http://www.micromo.com/uploadpk/1219_G_MME.PDF>.

[MicroMo Magnetic Encoder] MicroMo Electronics, Magnetic encoder. <http://www.micromo.com/uploadpk/HEM_MME.PDF>.