URK: Utah Robot Kit - A 3-link Robot Manipulator Prototype

Mohamed Dekhil, Tarek M. Sobh, and Thomas C. Henderson [1]

UUCS-94-006

Department of Computer Science
University of Utah
Salt Lake City, UT 84112 USA

February 17, 1994

Abstract

In designing robot manipulators, the interaction between several modules (S/W, VLSI, CAD, CAM, Robotics, and Control) illustrates an interdisciplinary prototyping environment that includes different types of information that are radically different but combined in a coordinated way. This paper describes the analysis and design of a 3-link robot manipulator prototype as part of a research project for building a prototyping environment for electro-mechanical systems. This prototype robot will be used as an educational tool in robotics and control classes.

[1]This work was supported in part by DARPA grant N00014-91-J-4123, NSF grant CDA 9024721, and a University of Utah Research Committee grant. All opinions, findings, conclusions or recommendations expressed in this document are those of the author and do not necessarily reflect the views of the sponsoring agencies. This report appears as a paper in the proceedings of the 1994 IEEE International Conference on Robotics and Automation.

URK: Utah Robot Kit - A 3-link Robot Manipulator Prototype

Mohamed Dekhil, Tarek M. Sobh, and Thomas C. Henderson *

Department of Computer Science
University of Utah
Salt Lake City, Utah 84112

Abstract

In designing robot manipulators, the interaction between several modules (S/W, VLSI, CAD, CAM, Robotics, and Control) illustrates an interdisciplinary prototyping environment that includes different types of information that are radically different but combined in a coordinated way. This paper describes the analysis and design of a 3-link robot manipulator prototype as part of a research project for building a prototyping environment for electro-mechanical systems. This prototype robot will be used as an educational tool in robotics and control classes.

1 Introduction

Designing and building an electro-mechanical system, such as a robot manipulator, requires a lot of tasks, starting with specifying the tasks and performance requirements, determining the robot configuration and parameters that are most suitable for the required tasks, ordering the parts and assembling the robot, developing the necessary software and hardware components (controller, simulator, monitor), and finally, testing the robot and measuring its performance.

Our goal is to build a framework for the optimal and flexible design of robot manipulators with the required software and hardware systems and modules which are independent of the design parameters, so that it can be used for different configurations and varying parameters. Figure 1 Shows a schematic view of the prototyping environment with its sub-systems and the interface.

A prototype three-link robot manipulator was built to determine the required sub-systems and interfaces to build the prototyping environment, and to provide hands-on experience for the real problems and difficulties that we would like to address and solve using this environment.

*This work was supported in part by DARPA grant N00014-91-J-4123, NSF grant CDA 9024721, and a University of Utah Research Committee grant. All opinions, findings, conclusions or recommendations expressed in this document are those of the author and do not necessarily reflect the views of the sponsoring agencies.

2 Background

2.1 Robot Modules and Parameters

Controlling and simulating a robot is a process that involves a large number of mathematical equations. To be able to deal with the required computation, it is better to divide into modules, where each module accomplishes a certain task. The most important modules, as described in [2] are: kinematics, inverse kinematics, dynamics, trajectory generation, and feedback control.

The behavior of the robot is affected by the choice of the robot parameters such as, degrees of freedom, joint allocation, link lengths, link masses, joint friction, feedback gains, and motor parameters. An optimal design system to select the optimal values for those parameters according to some performance requirements will be designed as part of the prototyping environment, more details of which can be found in [8].

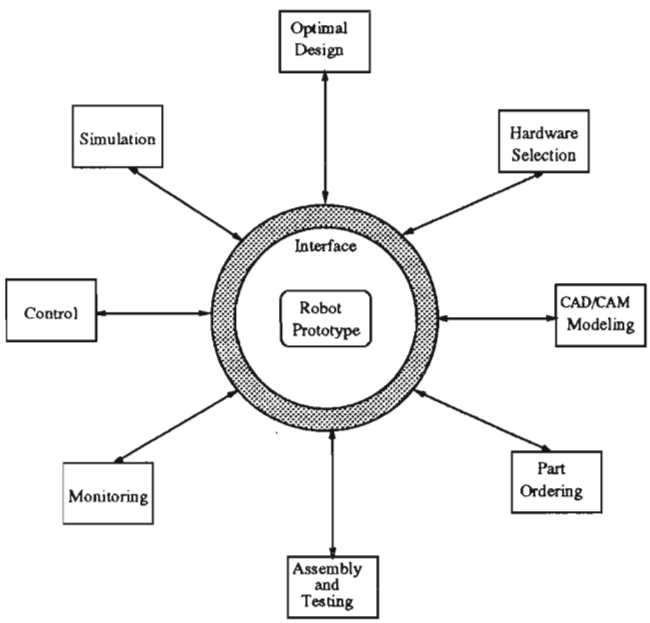

Figure 1: Schematic View for the Robot Prototyping Environment.

2.2 Robot Dynamics

Dynamics is the study of the forces required to cause the motion. There are two problems related to the dynamics of a manipulator: *controlling* the manipulator (inverse dynamics), and *simulating* the motion of the manipulator (forward dynamics). The dynamics module is the most time consuming among the manipulator's modules. That is because of the tremendous amount of calculation involved in the dynamics equations. This fact makes the dynamics module a good candidate for hardware implementation, to enhance the performance of the control and/or the simulation system.

There are some parallel algorithms to calculate the dynamics of a manipulator. Several approaches have been suggested in [5, 6, 7] based on a multiprocessor controller, and pipelined architectures to speed up the calculations.

2.3 Trajectory Generation

This module computes a trajectory in multidimensional space which describes the motion of the manipulator. There are several strategies to calculate trajectory points which generate a smooth motion for the manipulator. One of the simplest methods is using *cubic polynomials*.

2.4 Linear Feedback Control

A linear feedback control system is used in most control systems for positioning and trajectory tracking. There are sensors at each joint to measure the joint angle and velocity, and there is an actuator at each joint to apply a torque on the neighboring link. The readings from the sensors will constitute the feedback of the control system. By choosing appropriate gains we can control the behavior of the output function representing the actual trajectory generated. Minimizing the error between the desired and actual trajectories is our main concern. Figure 2 shows a block diagram for the controller, and the role of each of the robot modules in the system.

2.5 Local PD feedback Control vs Robot Dynamic Equations

Most of the feedback algorithms used in current control systems are implementations of a proportional plus derivative (PD) control. In industrial robots, a local PD feedback control law is applied at each joint independently. The advantages of using a PD controller are the following:

- Very simple to implement.
- No need to identify the robot parameters.
- Suitable for real-time control since it has very few computations compared to the complicated nonlinear dynamic equations.
- The behavior of the system can be controlled by changing the feedback gains.

On the other hand, there are some disadvantages of using a PD controller instead of the dynamic equations such as:

- It needs high update rate to achieve reasonable accuracy.
- To simulate the robot manipulator behavior the dynamic equations should be used.
- There is always trade-off between static accuracy and the overall system stability.
- Using local PD feedback law at each joint independently does not consider the couplings of dynamics between robot links.

Some ideas have been suggested to enhance the usability of the local PD feedback law for trajectory tracking. One idea is to add a lag-lead compensator using frequency response analysis [1]. Another method is to build an inner loop stabilizing controller using a multivariable PD controller, and an outer loop tracking controller using a multi-variable PID controller [9]. In general, using a local PD feedback controller with high update rates can give an acceptable accuracy for trajectory tracking applications. It was proved that using a linear PD feedback law is useful for positioning and trajectory tracking [3].

3 Prototyping a 3-Link Robot

As part of this research project, a three-link robot manipulator was designed along with its controller and simulator. The main concern was trying to make the installation and wiring of this robot as simple as possible so that it is easy to connect to any workstation or PC through a standard serial port using an RS232 cable. The following sections describes the stages and activities accomplished during the design process.

3.1 Analysis Stage

We started this project with the study of a set of robot configurations and analyzing the type and amount of calculation involved in each of the robot controller modules (kinematics, inverse kinematics, dynamics, trajectory planning, feed-back control, and simulation). this phase was accomplished by working through a generic example for a 3-link robot to compute symbolically the kinematics, inverse kinematics, dynamics, and trajectory planning; these were linked to a generic motor model and its control algorithm. This study enabled us to specify the parameters of the robot for performing various tasks, it also helped us determine which parts (algorithms) should be hardwired to achieve specific mechanical performances, and also how to supply the control signals efficiently and at what rates.

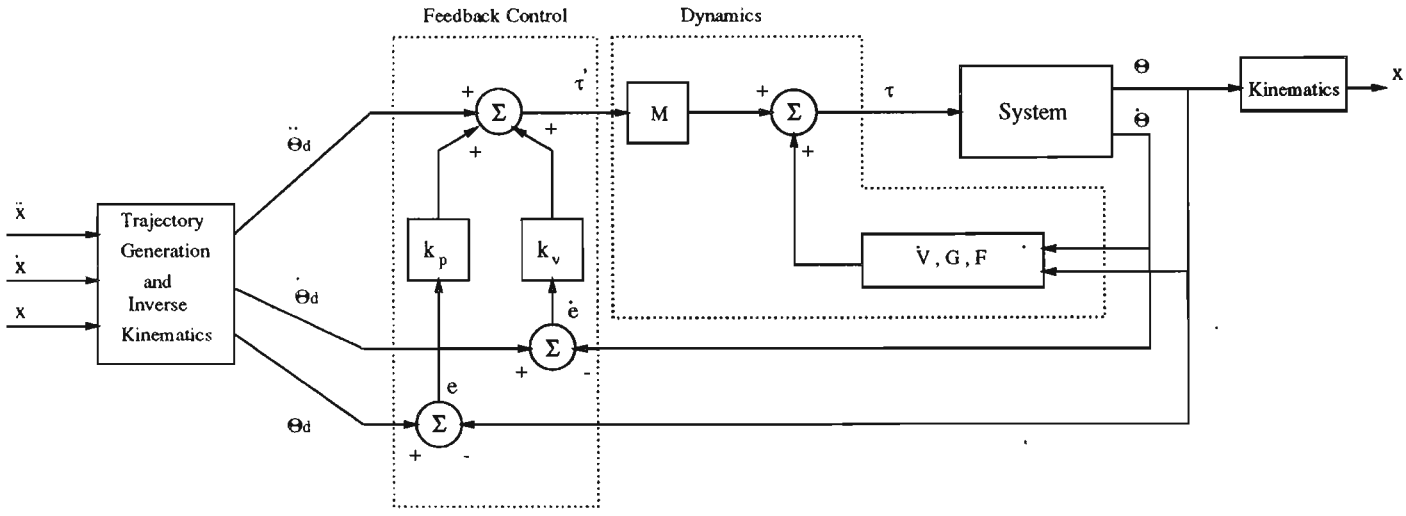

Figure 2: Block diagram of the Controller of a Robot Manipulator.

3.2 One Link Manipulator

Controlling a one-link robot in a real-time manner is not too difficult, but on the other hand it is not a trivial task. This is the basis of controlling multi-link manipulators, and it gives an indication of the type of problems and difficulties that might arise in a larger environment. The idea here is to establish a complete model for controlling and simulating a one-link robot, starting from the analysis and design, through the simulation and error analysis.

A motor controlled by PID controller was used with an analog I/O card, named PC-30D, connected to a Hewlett Packard PC. This card has sixteen 12-bit A/D input channels, two 12-bit D/A output channels. There are also the card interface drivers with a Quick BASIC program that uses the card drivers to control the DC motor.

One of the problems we faced in this process is to establish the transfer function between the torque and the voltage. We used the motor parameters to form this function by making some simplifications, since some of the motor parameters have non-linear components which makes it too difficult to make an exact model. The transfer function is in the form:

$$v(t) = \frac{T_m(t)}{K_T} R + \frac{\dot{T_m}(t)}{K_T} L + K_E \dot{\theta},$$

where,

$v(t)$ is the voltage at time t.
$T_m(t)$ is the torque at time t.
K_T is the torque from the motor.
$L = 13.4 \times 10^{-3} H$.
$R = 4.96\Omega$.
$K_T = 20.8 oz.in.sec^2$.
$K_E = 0.147 v/rad/sec$.

several input sequences have been used for the desired trajectories and the actual positions and velocities are measured using a potentiometer and a tachometer. The results were fed to a graphical simulation program which displays the movement of the link, the desired and actual positions, the desired and actual velocity, and the error in position and velocity. The interface window is shown in Figure 4.

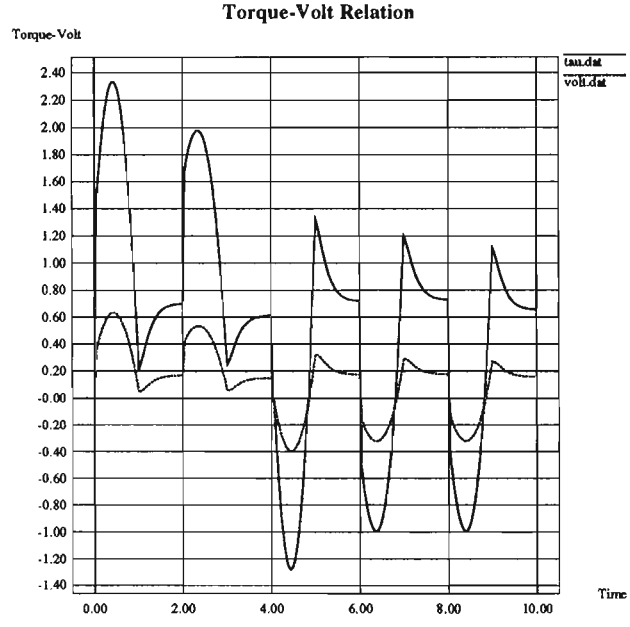

Figure 3: The Relation Between the Torque and the Voltage.

In general, This experiment gave us an indication of the feasibility of our project, and good practical insight. It also helped us determine some of the technical problems that we might face in building and controlling the three-link robot. More details about this experiment

can be found in [8].

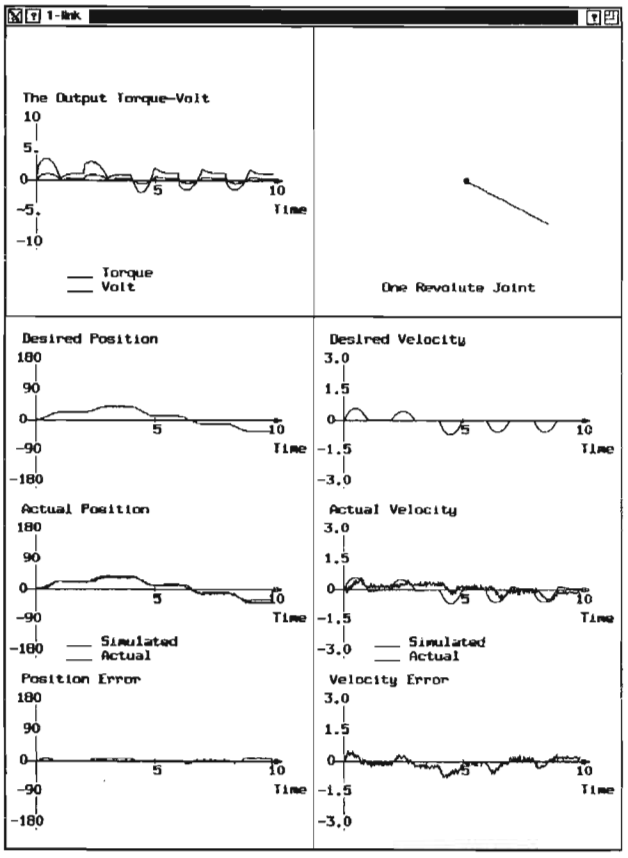

Figure 4: The Output Window of the Simulation Program.

3.3 Sensors and Actuators Interface

This part is concerned with the communication between the manipulator and the workstation used to control it. Basically, we have three D/A lines that transmit the required torque (or voltage) from the workstation to the three manipulator's actuators, and we have six A/D lines that transmit the sensors readings at each joint to the workstation (three for the position, and three for the velocity). These readings are used in the controller for feedback information.

A microcontroller system (the MC68HC11EVBU - Universal Evaluation Board) was used. It has a microcontroller, an 8-channel A/D, an RS-232 compatible terminal I/O port, and some wire wrap area for additional circuitry like the D/A unit. The MC68HC11E9 is a high-speed, low-power chip with a multiplexed bus capable of running at up to 3 MHz. Its fully static design allows it to operate at very low frequencies.

3.4 Speed Considerations

There are several factors which affect the desired speed (frequency of calculations). Some of these factors are:

- Input frequency
- Sensing frequency
- Noise frequency
- Machine speed
- Maximum error allowed

Khosla performed some experiments to study the effect of changing the control sampling rate on the performance of the manipulator behavior [4] and showed that increasing the update rate decreases the error.

3.5 Controller Design

The first step in the design of a controller for a robot manipulator is to solve for its kinematics, inverse kinematics, dynamics, and the feedback control equation that will be used. Also the type of input and the user interface should be determined at this stage.

For trajectory generation, we used the cubic polynomials method which generates a cubic function that describes the motion from a starting point to a goal point in a certain time. The error in position and velocity is calculated using the readings from the sensors. Our control module simulates a PD controller to minimize that error. The error depends on several factors such as: frequency of update, frequency of sensors reading, and the desired trajectory (for example, if we want to move with a big angle in a very small time interval, the error will be large).

3.6 PID controller Simulator

As mentioned earlier, a simple linear feedback control law can be used to control the robot manipulator for positioning and trajectory tracking. For this purpose, a PID controller simulator was developed to enable testing and analyzing the robot behavior using this control strategy. Using this control scheme helps us avoid the complex (and almost impossible) task of determining the robot parameters for our 3-link prototype robot. One of the most complicated parameters is the inertia tensor matrix for each link, especially when the links are non-uniform and has complicated shape.

This simulator has a user friendly interface that enables the user to make on-line changes to any of the feedback coefficients and the forward gains. It can also read a pre-defined position trajectory for the robot to follow. It will also serve as a monitoring system that provides several graphs and reports. Figure 5 shows the interface window of that simulator.

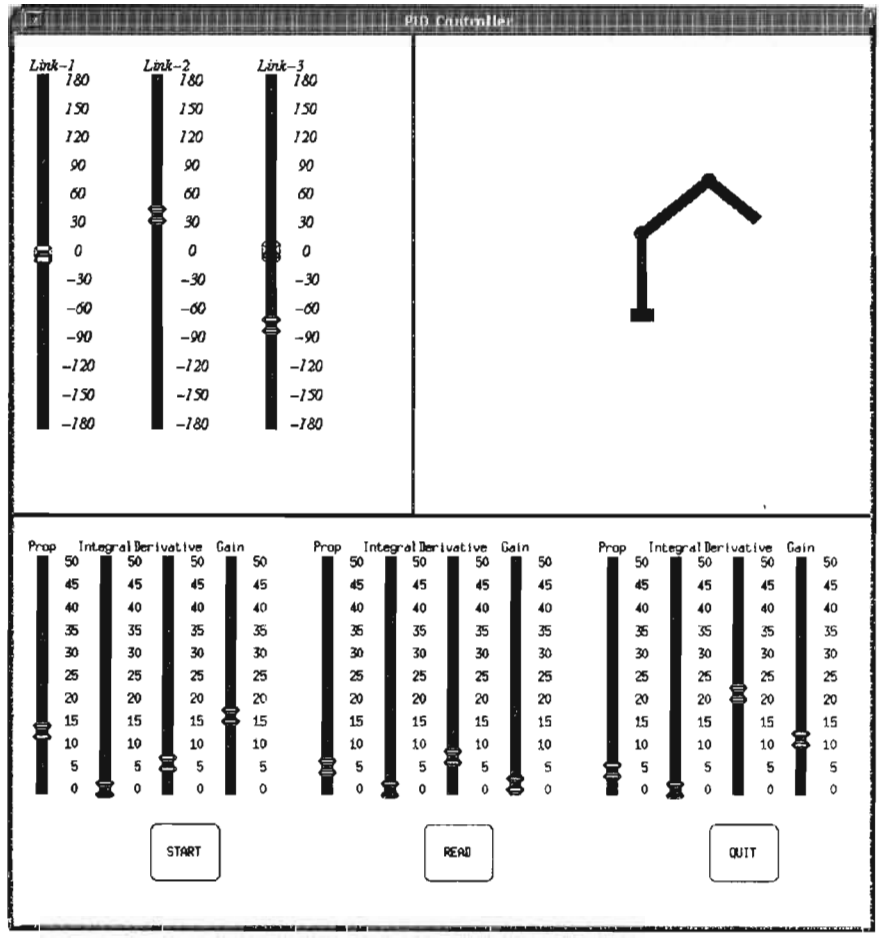

Figure 5: The Interface Window for the PID Controller Simulator.

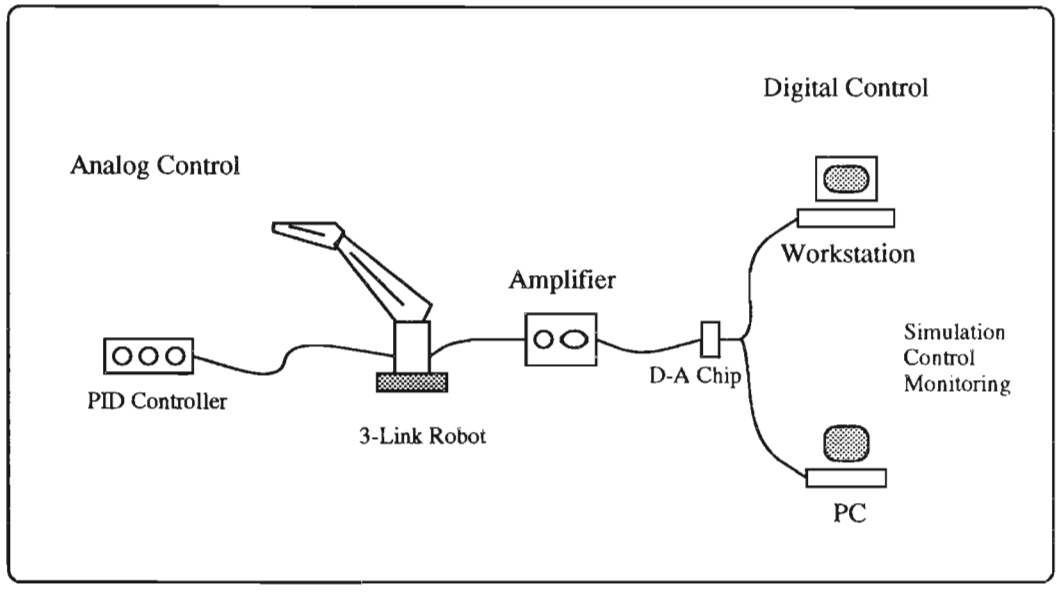

Figure 7: Controlling the robot using different schemes

3.7 Building the Robot

The design and manufacture of the robot is completed and the required parts (motors, actuators, and sensors) are now assembled. The manufacture and assembly process of the links, mechanical, and electrical parts were done at the Advanced Manufacturing Laboratory (AML) in our computer science department. The last link will be either revolute or prismatic, so that the robot can be set in different configurations.

There are three different motors to drive the three links, and six sensors (three for position and three for velocity), to read the current position and velocity for each link to be used in the feedback control loop.

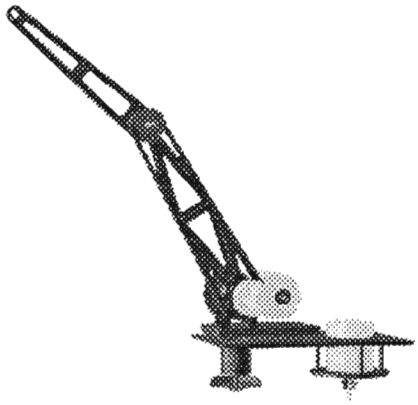

Figure 6: The Physical Three-Link Robot manipulator

This robot can be controlled using analog control by interfacing it with an analog PID controller, and monitoring its behavior an oscilloscope. Digital control is used by interfacing the robot with either a workstation (SUN, HP, etc.) or a PC via the standard RS232. This requires an A/D and D/A chip to be connected to the workstation (or the PC) and an amplifier that provides enough power to drive the motors. Figure 7 shows an overall view of the different interfaces and platforms that can control that robot.

4 Conclusion

A prototype 3-link robot manipulator was built to determine the required components for a flexible prototyping environment for electro-mechanical systems in general, and for robot manipulators in particular. A local linear PD feedback law was used for controlling the robot for positioning and trajectory tracking. A graphical user interface was implemented for controlling and simulating the robot. This robot is intended to be an educational tool, therefore it was designed in such a way that makes it very easy to install and manipulate. The design process of this robot helped us determine the necessary components for building a prototyping environment for electro-mechanical systems.

5 Acknowledgements

We would like to express our thanks to Mircea Cormos, Sanford Meek, Anil Sabbavarapu, and Beat Brüderlin for helping out make this robot come to life.

References

[1] CHEN, Y. Frequency response of discrete-time robot systems - limitations of pd controllers and improvements by lag-lead compensation. In *IEEE Int. Conf. Robotics and Automation* (1987), pp. 464–472.

[2] CRAIG, J. *Introduction To Robotics.* Addison-Wesley, 1989.

[3] KAWAMURA, S., MIYAZAKI, F., AND ARIMOTO, S. Is a local linear pd feedback control law effictive for trajectory tracking of robot motion? In *IEEE Int. Conf. Robotics and Automation* (1988), pp. 1335–1340.

[4] KHOSLA, P. K. Choosing sampling rates for robot control. In *IEEE Int. Conf. Robotics and Automation* (1987), pp. 169–174.

[5] LATHROP, R. H. Parallelism in manipulator dynamics. *Int. J. Robotics Research 4*, 2 (1985), 80–102.

[6] LEE, C. S. G., AND CHANG, P. R. Efficient parallel algorithms for robot forward dynamics computation. In *IEEE Int. Conf. Robotics and Automation* (1987), pp. 654–659.

[7] NIGAM, R., AND LEE, C. S. G. A multiprocessor-based controller for mechanical manipulators. *IEEE Journal of Robotics and Automation 1*, 4 (1985), 173–182.

[8] SOBH, T. M., DEKHIL, M., AND HENDERSON, T. C. Prototyping a robot manipulator and controller. Tech. Rep. UUCS-93-013, Univ. of Utah, June 1993.

[9] TAROKH, M., AND SERAJI, H. A control scheme for trajectory tracking of robot manipulators. In *IEEE Int. Conf. Robotics and Automation* (1988), pp. 1192–1197.