

Concurrent Design of a Three-link Manipulator Prototype

Tarek M. Sobh, Mohamed Dekhil, Thomas C. Henderson,
Anil Sabbavarapu and Raul Mihali

Department of Computer Science and Engineering
University of Bridgeport
Bridgeport, CT 06601, USA

and

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

Abstract

The paper presents an affordable and comprehensive robotic model of critical aid to any engineering school involved in teaching robotics. We present the stages of designing a three-link robot manipulator prototype that was built as part of a research project for establishing a prototyping environment for robot manipulators. Building this robot helped determine the required subsystems and interfaces for building the prototyping environment, and provided hands-on experience for real problems and difficulties that are addressed and solved using this environment. The robot is now successfully used as an educational tool in robotics and control classes.¹

Keywords: Robotics, Prototyping, Control, Design.

¹This 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.

1 Introduction

Teaching robotics in most engineering schools lacks practicality. Usually students spend most of the time studying the theoretical background and the mathematics, and possibly writing some simulation programs. In many cases, they do not get the chance to apply and practice what they have learned using real robots. This is due to the fact that most of the robots available in the market are either too advanced, complicated, and expensive (e.g., specialized industrial robots), or toy-like robots which are trivial and do not give the required level of depth or functionality needed to demonstrate the main concepts of robot design and control. One of our goals in this project, was to build a robot that is inexpensive, flexible, easy to use and connect to any workstation or PC, and at the same time is capable of demonstrating some of the design and control concepts. We also tried to keep the cost as low as possible, thus making it available to any engineering school or industrial organization. While a great part of the theoretical concepts or practical approaches considered in the design stages of the presented manipulator are nowadays well known and even obsolete, they define the skeleton of any proficient robotics teaching environment.

We consider the main contribution of this work is building URK (Utah Robot Kit) which is a three-link robot prototype that has a small size and reasonable weight, which is convenient for a small lab or a class room. URK can be connected to any workstation or PC through the standard serial port with an RS232 cable, and can be controlled using a software controller with a graphical user interface. This software applies a simple PID control law for each link which does not require knowledge of the robot parameters. It can be also used on any electro-mechanical system that can be controlled by a physical PID controller. The interface enables the user to change any of the control parameters and to monitor the behavior of the system with on-line graphs and a 3-D view of the robot showing the current position of the robot.

The paper starts with a brief background of robot design and modules and the related work in this area. A detailed description of designing and building URK is presented in Section 3. The communication between the robot and the workstation is discussed in detail in Section 4. Section 5 shows some results of testing and running URK. Finally, Section 6 includes our conclusions.

2 Background and Related Work

Controlling and simulating a robot is a process that involves a large number of mathematical equations. To deal with the magnitude of computations, it is advisable to divide them into modules. Each module accomplishes a certain task. The most important modules, as described in [2], are kinematics, inverse kinematics, dynamics, trajectory generation, and linear feedback control.

2.1 Robot Modules

There has been a lot of research to automate kinematic and inverse kinematic calculations. A software package called SRAST (Symbolic Robot Arm Solution Tool) that symbolically solves the forward and inverse kinematics for n -degree of freedom manipulators has been developed by Herrera-Bendezu, Mu, and Cain [5]. The input to this package is the Denavit-Hartenberg parameters, and the output is the direct and inverse kinematics solutions. Another method of finding symbolic solutions for the inverse kinematics problem was proposed in [12]. Kelmar and Khosla proposed a method for automatic generation of forward and inverse kinematics for a reconfigurable manipulator system [7].

Dynamics is the study of the forces required to cause the motion. There are some parallel algorithms to calculate the dynamics of a manipulator. Several approaches have been suggested in [8, 9, 11] based on a multiprocessor controller, and pipelined architectures to speed up the calculations.

Linear feedback control 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 constitutes 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 the main concern. Figure 1 shows a block diagram for the controller, and the role of each of the robot modules in the system.

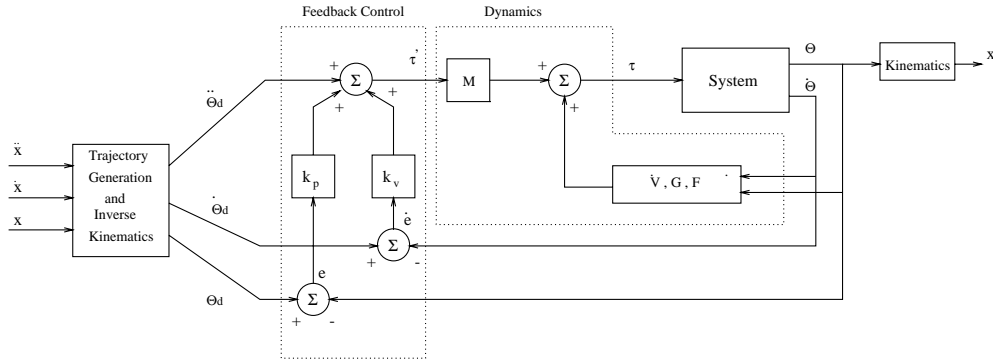


Figure 1: Block diagram of the Controller of a Robot Manipulator.

2.2 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:

- It is very simple to implement.
- There is no need to identify the robot parameters.
- It is suitable for real-time control since it includes few computations compared to the complicated non-linear dynamic equations.
- The behavior of the system can be controlled by changing the feedback gains.

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

- 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 multi-variable PD controller, and an outer loop tracking controller using a multi-variable PID controller [13]. In general, using a local PD feedback controller with high update rates can result in acceptable accuracy for trajectory tracking applications. It was proved that using a linear PD feedback law is useful for positioning and trajectory tracking [6].

3 Prototyping a 3-Link Robot

3.1 Analysis Stage

This project was initiated by studying a set of robot configurations and analyzing the type and amount of calculations 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 three-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 determine the specifications of the robot for performing various tasks, it also helped us decide which parts (algorithms) should be hardwired to achieve specific mechanical performances, and also how to supply the control signals efficiently and at what rates.

3.2 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. The type of input and the user interface should be determined at this stage. We should know the parameters of the robot, such as: link lengths, masses, inertia tensors, distances between joints, the configuration of the robot, and the type of each link (revolute or prismatic). To implement a modular and flexible design, variable parameters are used that can be fed to the system at run-time, so that the controller can be used for different configurations without encoding any changes.

Three different configurations have been chosen for development and study. The first configuration is revolute-revolute-prismatic with the prismatic link in the same plane as the first and second links. The second configuration is also revolute-revolute-prismatic with the prismatic link perpendicular to the plane of the first and second links. The last configuration is three revolute joints (see Figure 2).

The kinematics and the dynamics of the three models have been generated using tools in the department (*genkin* and *gendyn*) that are supplied with the configuration of the manipulator and generate the corresponding kinematics and dynamics for that manipulator. For the trajectory generation, The cubic polynomials method, described in the trajectory generation section, was used. This method is easy to implement and does not require many computations. It generates a cubic function that describes the motion from a starting point to a goal point in a certain time. Thus, this module will provide the desired trajectory to be followed, and this trajectory will serve as the input to the control module.

The error in position and velocity is calculated using the readings of the actual position and velocity from the sensors at each joint. Our control module simulated a PID controller to minimize that error. The error depends on several factors such as the frequency of update, the frequency of reading from the sensors, and the desired trajectory.

3.3 Simulation

A simulation program has been implemented to study the performance of each manipulator and the effect of varying the update frequency on the system. It helps in finding approximate

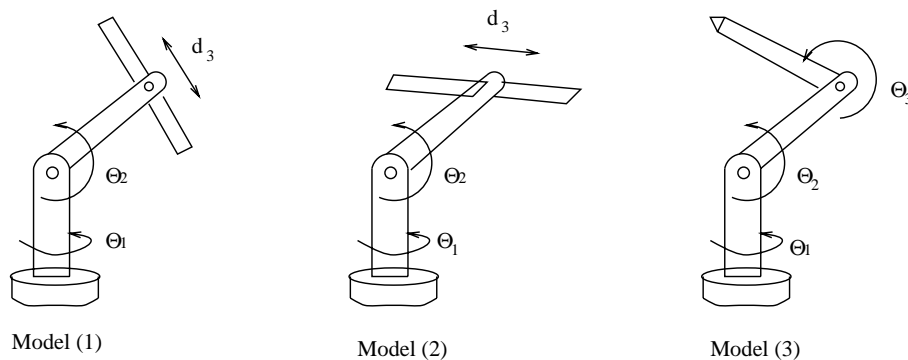


Figure 2: Three different configurations of the robot manipulator.

Table 1: Number of calculations involved in the dynamics module.

	Additions	Multiplications	Divisions
Model 1	89	271	13
Model 2	85	307	0
Model 3	195	576	22

ranges for the required torque and/or voltage, and determine the maximum velocity to derive the necessary type of sensors and A/D conversion. To derive the benchmarks, as described in the next section, we did not use a graphical interface for the simulator, since the drawing routines are time consuming, and thus give misleading figures for the speed.

In this simulator the user can select the length of the simulation, and the update frequency. The third model was used for testing and benchmarking because its dynamics are the most difficult and time consuming compared to the other two models. Table 1 shows the number of calculations in the dynamics module for each model.

3.4 PID Controller Simulator

As mentioned in Section 2.2, 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 in avoiding the complex (and almost impossible) task of determining the robot parameters for our three-link prototype robot. One of the most complicated parameters is the inertia tensor matrix for each link, especially when the links are nonuniform and have complicated shapes.

This simulator has a user friendly interface that enables the user to change any of the feedback coefficients and the forward gains on-line. It can also read a pre-defined position trajectory for the robot to follow. It also serves as a monitoring system that provides several graphs and reports. The system is implemented using a graphical user interface development

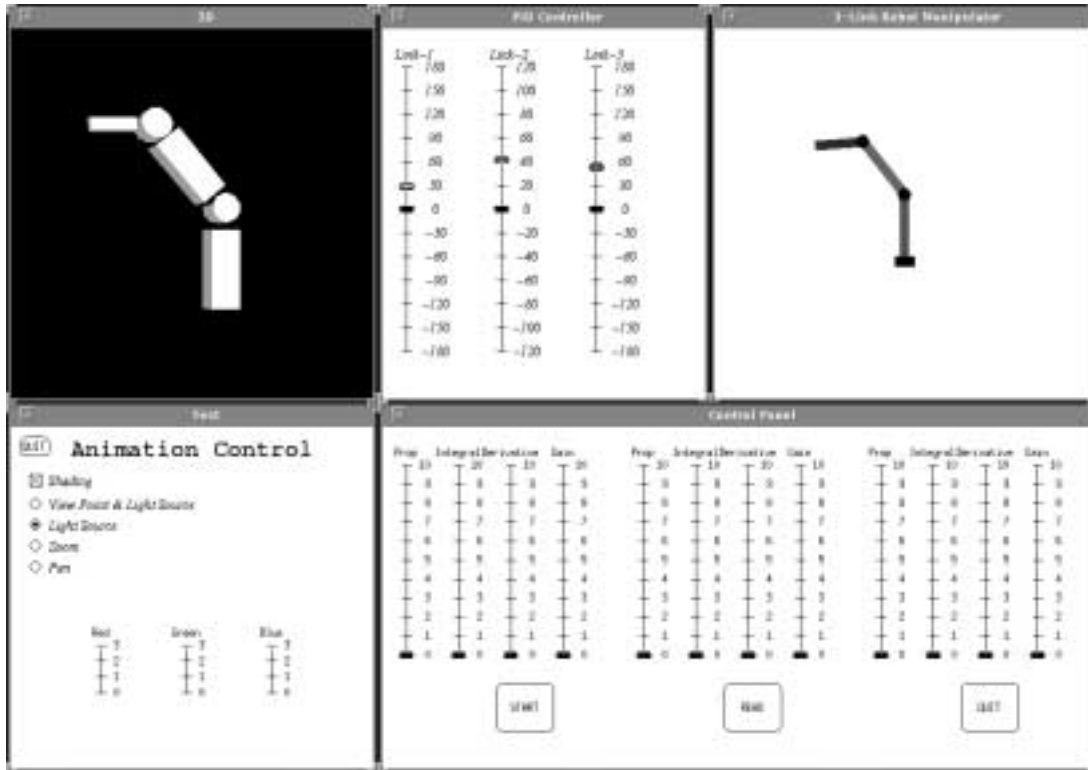


Figure 3: The interface window for the PID controller simulator.

kit called GDI.² Figure 3 shows the interface window of that simulator.

3.5 Building the Robot

The assembly process of the mechanical and electrical parts was done in the Advanced Manufacturing Lab (AML) with the help of Mircea Cormos and Prof. Stanford Meek. In this design the last link is movable, so that different robot configurations can be used (see Figure 4).

There are three 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.

This robot can be controlled using analog control by interfacing it with an analog PID controller. Digital control can also be used by interfacing the robot with either a workstation

²GDI was developed in the department of Computer Science, University of Utah, under supervision of Prof. Beat Brüderlin.



Figure 4: The physical three-link robot manipulator.

(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 5 shows an overall view of the different interfaces and platforms that can control the robot. A summary of this design can be found in [3, 4].

4 Robot-computer Interface

The sensor and actuator interface is an essential part of the project. It is concerned with the communication between the manipulator and the computer used to control it. A resident program on the SUN can send out voltage values that will drive the motors in a desired direction (forward or backward), and read values from sensors placed on each link that correspond to the position of that link. It was obvious that we would need A/D's to convert the values coming from the motors to digital so that they can be sent to the workstation (where the control program resides), D/As to convert the values sent by the program to the actual analog voltage, and an RS-232 communication to the workstation to send these digital data to and from the workstation. We need some control of sampling, sending, and receiving data outside the workstation.

For this purpose, we used an MC68HC11 MCU device which is an advanced single-chip

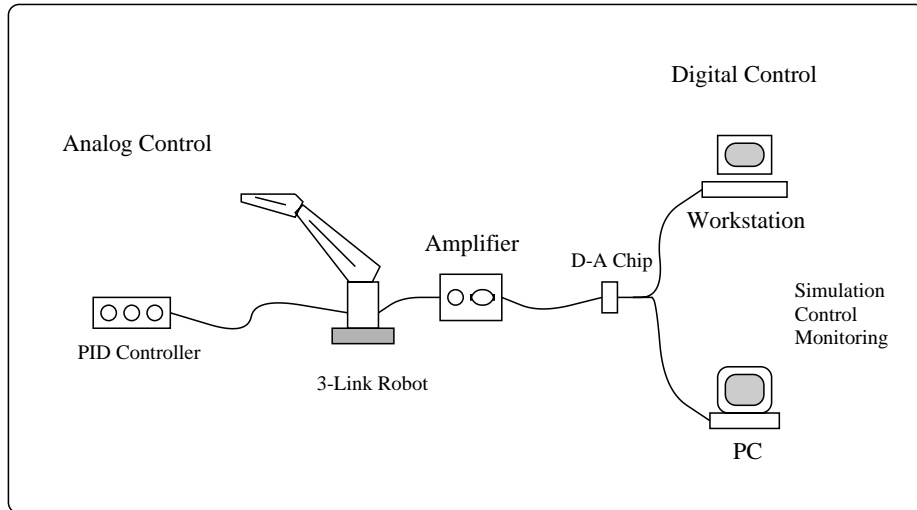


Figure 5: Controlling the robot using different schemes.

MCU (Micro Control Unit) with on-chip memory and peripheral functions. The EVBU comes with a monitor/debugging program called BUFFALO (Bit User Fast Friendly Aid to Logical Operations), which is contained in the MCU ROM. More details about this chip can be found in [10].

4.1 Analog to Digital Converter

The A/D system is an 8-channel, 8-bit, multiplexed-input converter. It does not require external sample-and-hold circuit because of the type of charge redistribution technique used. A/D converter timing can be synchronized to the system clock, or to an internal RC oscillator. The A/D converter system consists of four functional blocks: multiplexer, analog converter, digital control, and result storage.

The A/D converter operations are performed in sequences of four conversions each. A conversion sequence can repeat continuously or stop after one iteration. The conversion complete flag (CCF) is set after the fourth conversion in a sequence to show the availability of data in the result registers.

4.2 Digital to Analog Converter

For the D/A conversion, we used an 8-Bit microprocessor compatible, double buffered DAC0830. The DAC0830 is an advanced CMOS 8-bit multiplying DAC designed to interface directly with most of the popular microprocessors. The circuit uses CMOS current switches and control logic to achieve low power consumption and low output leakage current errors. Double buffering allows these DACs to output a voltage corresponding to one digital word while holding the next digital word. The DAC can be used in different modes of operation.

5 Testing and Results

5.1 Simulator for three-link Robot

This simulator was used to give some rough estimates about the required design parameters such as link lengths, link masses, update rate, feedback gains, etc. It is also used in the benchmarking described earlier. Figure 6 shows the simulated behavior of a three-link robot. It shows the desired and actual position and velocity for each link and the error for each of them. It also shows a line drawing for the robot from two different view points.

This simulator uses an approximate dynamic model for the robot, and it allows any of the design parameters to be changed. For example, the effect of changing the update rate on the position error is shown in Figure 7. From this figure, it is clear that increasing the update rate decreases the position error.

5.2 Software PID Controller

A software controller was implemented for the three-link robot. This controller uses a simple local PID control algorithm, and simulates three PID controllers; one for each link. Several experiments and tests have been conducted using this software to examine the effects of changing some of the control parameters on the performance of the robot.

The control parameters that can be changed in this program are:

- forward gain (k_g)
- proportional gain (k_p)

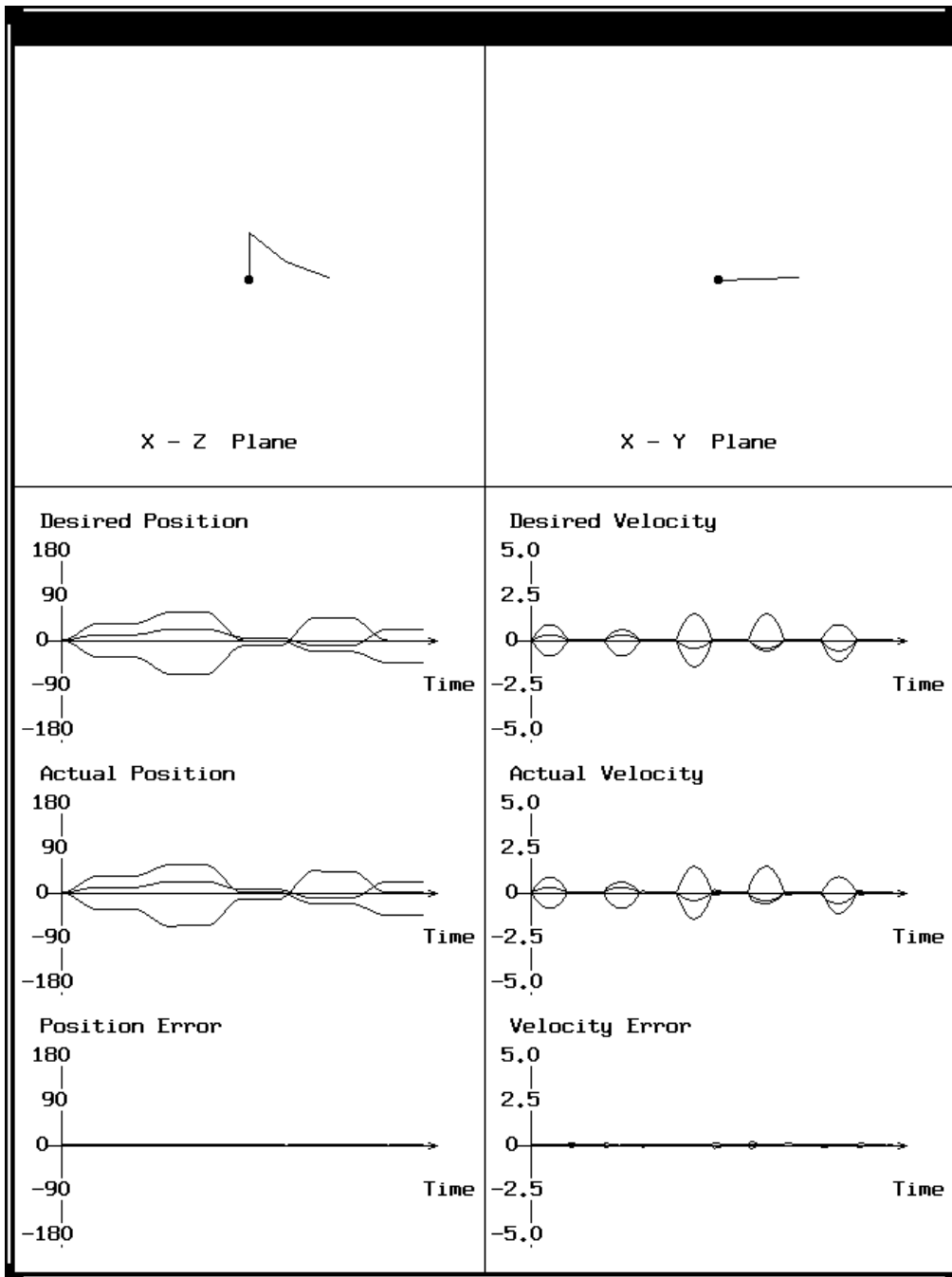


Figure 6: The output window of the simulator for the three-link robot.

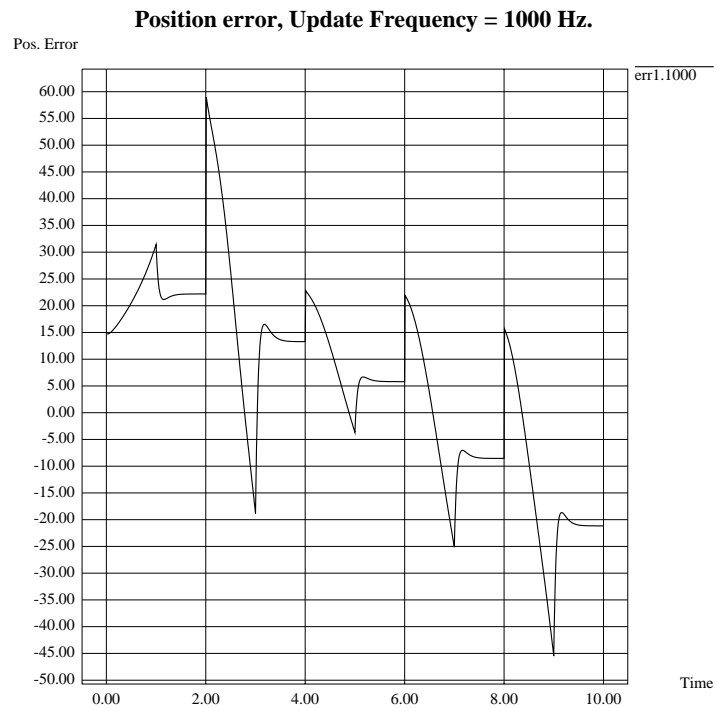
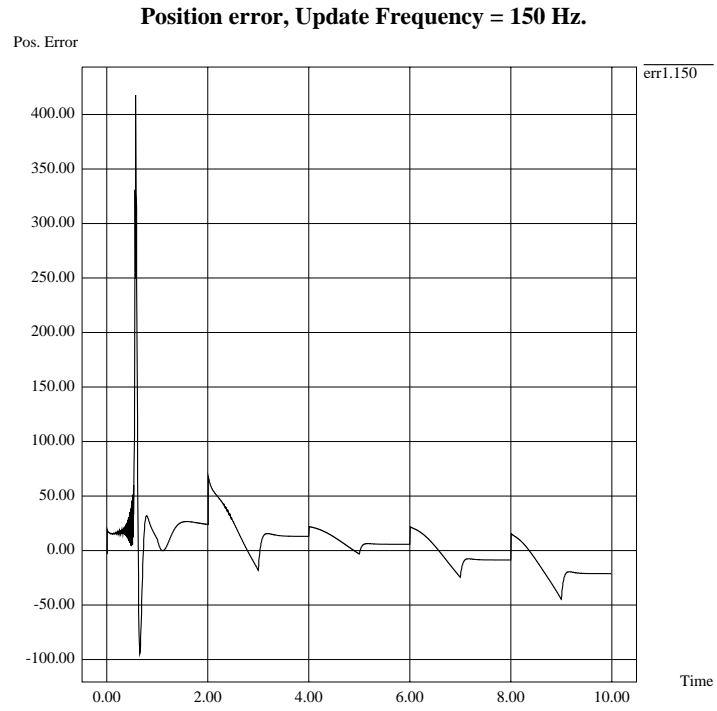


Figure 7: The effect of changing the update rate on the position error.

- differential gain (k_v)
- integral gain (k_i)
- input trajectory
- update rate

In these experiments, the program was executed on a Sun SPARCStation-10, and the A/D chip was connected to the serial port of the workstation. One problem we encountered with this workstation is the slow protocol for reading the sensor data, since it waits for an I/O buffer to be filled before it returns control to the program. We tried to change the buffer size or the time-out value that is used, but we had no success in that. This problem causes the update rate to be very low (about 30 times per second), and this affects the positional accuracy of the robot. We were able to solve this problem on an HP-700 machine, and we reached an update rate of 120 times per second which was good enough for our robot. Figure 8 shows the desired and actual position for different test cases using different feedback gains.

6 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. The robot proves to be a tool with a distinct didactic and affordable character. Students that have taken robotics courses relying on the prototype and its design process (as contoured in the paper) consider it essential, report major improvements in perceiving the field of robotics and significant enthusiasm in exploring its endless possibilities. More detailed information pertaining to the robot (design, cost, application, software and hardware) can be found at http://www.cs.utah.edu/vision/robotics_kit.html.

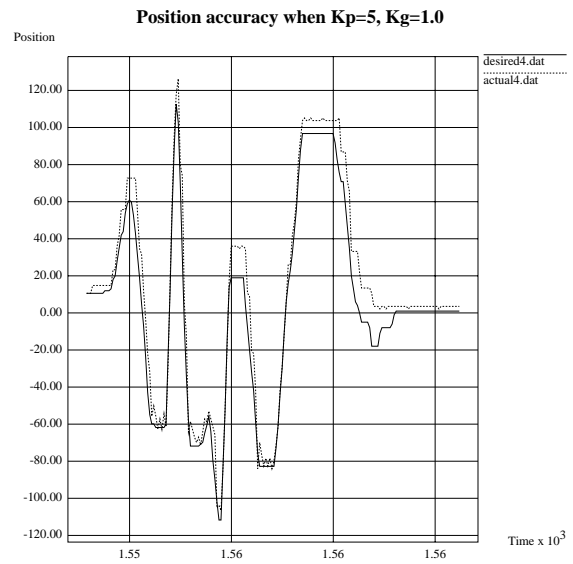
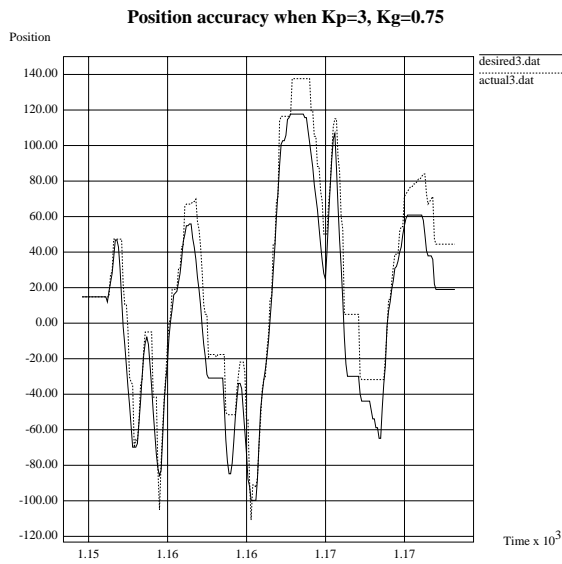
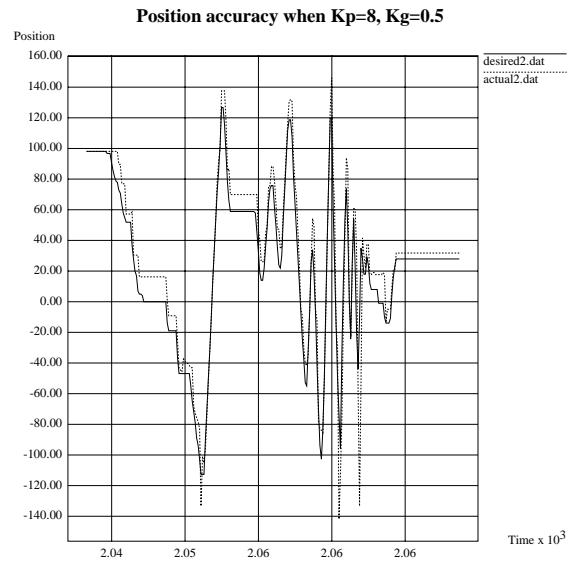
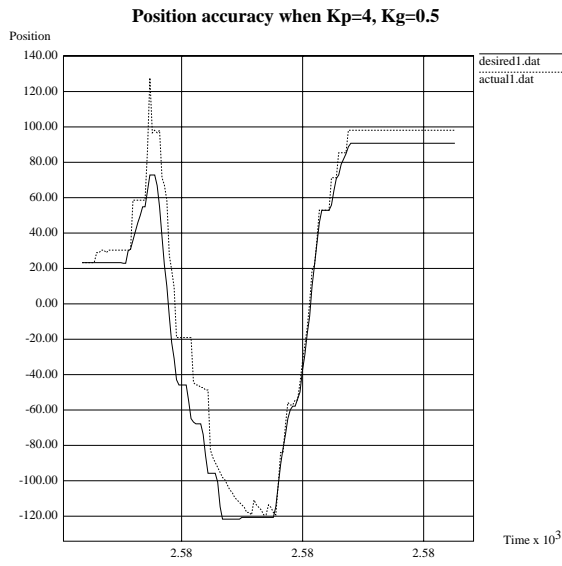


Figure 8: Desired and actual position for several test cases.

Acknowledgments

We would like to express our thanks to Mircea Corman, Prof. Sanford Meek, and Prof. Beat Brüderlin for helping 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] DEKHIL, M., SOBH, T. M., AND HENDERSON, T. C. URK: Utah Robot Kit - a 3-link robot manipulator prototype. In *IEEE Int. Conf. Robotics and Automation* (May 1994).
- [4] DEKHIL, M., SOBH, T. M., HENDERSON, T. C., AND MECKLENBURG, R. Robotic prototyping environment (progress report). Tech. Rep. UUCS-94-004, University of Utah, Feb. 1994.
- [5] HERRERA-BENDEZU, L. G., MU, E., AND CAIN, J. T. Symbolic computation of robot manipulator kinematics. In *IEEE Int. Conf. Robotics and Automation* (1988), pp. 993–998.
- [6] KAWAMURA, S., MIYAZAKI, F., AND ARIMOTO, S. Is a local linear pd feedback control law effective for trajectory tracking of robot motion? In *IEEE Int. Conf. Robotics and Automation* (1988), pp. 1335–1340.
- [7] KELMAR, L., AND KHOSLA, P. K. Automatic generation of forward and inverse kinematics for a reconfigurable manipulator system. *Journal of Robotic Systems* 7, 4 (1990), pp. 599–619.
- [8] LATHROP, R. H. Parallelism in manipulator dynamics. *Int. J. Robotics Research* 4, 2 (1985), pp. 80–102.

- [9] 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.
- [10] MOTOROLA INC. *MC68HC11E9 HCMOS Microcontroller Unit*, 1991.
- [11] NIGAM, R., AND LEE, C. S. G. A multiprocessor-based controller for mechanical manipulators. *IEEE Journal of Robotics and Automation* 1, 4 (1985), pp. 173–182.
- [12] RIESELER, H., AND WAHL, F. M. Fast symbolic computation of the inverse kinematics of robots. In *IEEE Int. Conf. Robotics and Automation* (1990), pp. 462–467.
- [13] 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.