Implementation of Conventional and Neural Controllers Using Position and Velocity Feedback

Senior Project Proposal

By:
Christopher Spevacek
and
Manfred Meissner

Advisor:
Dr. Gary Dempsey

Table Of Contents

Overview:
Our project objective is to design and evaluate different controllers for a small robot arm-motor platform from Quanser Consulting. Our main effort will be to work in a software environment on a 200 MHz or higher Pentium Computer to develop the controllers and signal processing algorithms in C language. An internal A/D and D/A converter card is connected to the external plant as shown in Fig. 1. The plant consists of an amplifier, DC motor assembly, external gear train, external load, and potentiometer for the position sensor (also from Quanser Consulting). The feedback signal will be passed through an antialiasing filter, to the A/D converter, into the computer. The feedback voltage signal is proportional to the position of the robotic arm. The arm position, output position on Fig. 1, is the primary output of the system, although arm velocity will also be of interest.

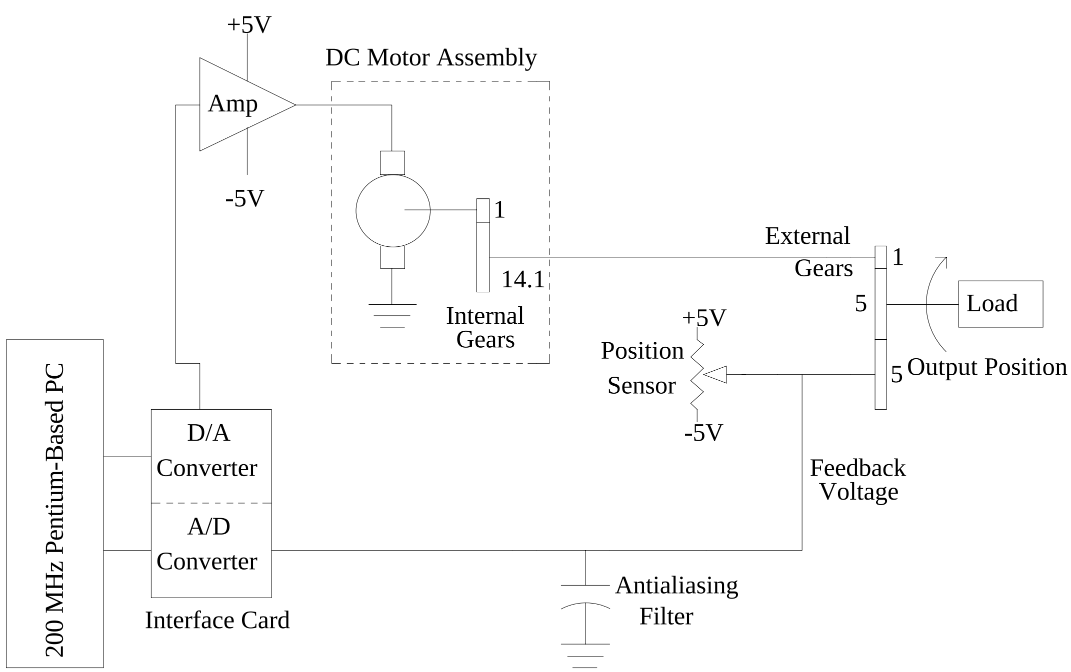


Fig. 1. Low Level System Block Diagram

Previous Work:
Quanser Consulting provided software to use with to robot arm, A/D, and D/A converters. This software provided the user interface and real-time graphing options. Other work completed was performed by Dr. Dempsey on different velocity algorithms and the structure of the neural network architecture.

High Level Block Diagram:
The high-level system block diagram in Fig. 2 shows the main parts of the project. The robotic arm system is expanded in Fig. 1. It consists of mechanical devices and the circuitry to convert the input and output of the A/D and D/A converters to the proper analog signals. The D/A converter converts the PC signals to analog signals that will perform actions on the robot arm. The position sensor, where a velocity feedback loop will be added later, sends the position of the robot arm through an A/D converter back to the PC. The Pentium-based PC will perform all the calculations, update display information, and generate the command signal. If the optional joystick is present in the system, the command signal will be generated by the user's interactions with the joystick. The programs will be written in C and used under windows. All the mechanical components are ready to use and Quanser Consulting supplied some PC programs.

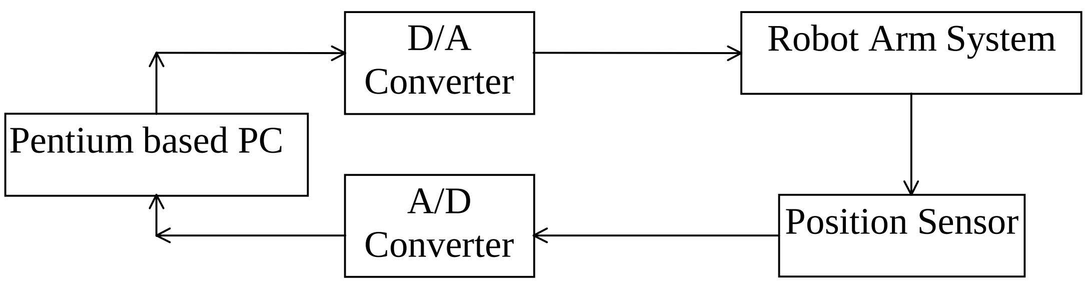


Fig. 2. High Level Block Diagram

Modes of Operation:
Two modes of operation will be incorporated into the product. The first mode is an option of connecting a joystick to one A/D channel and will allow the user to control (command) the robot arm position. An external digital input can be used to signal the software that a joystick is present. The other mode (default) will use an internal software command signal to control robot arm position. The user via keyboard will be able to change the set point (desired final position) and the slope of the command signal (velocity).

Control Block Diagram:
The signals that are important in this system are described below. They are separated into three categories: (1) inputs, (2) outputs, and (3) internal signals to the computer that are important for controller evaluation and testing, they are shown in Fig 3.

(1) Input signals to computer:
The letters in brackets refer to the letters in Fig. 3:

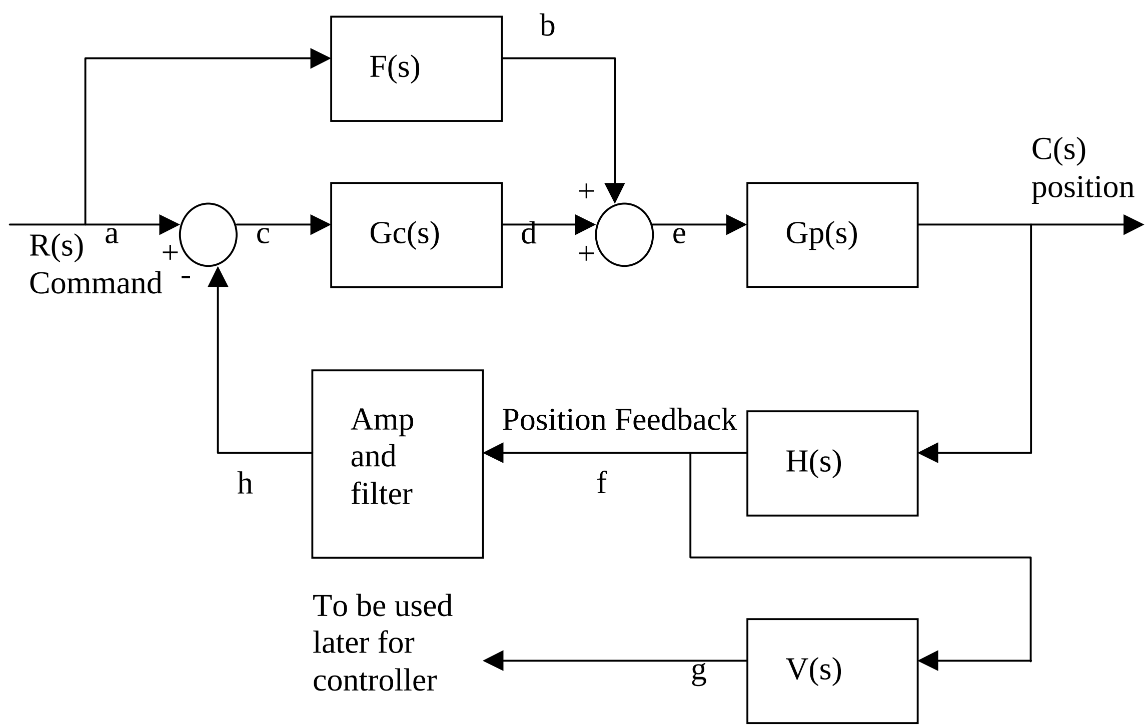
Signal	**Description**
Voltage from position sensor (proportional to position)[f]	Used to help create the new control signal
Voltage from joystick	Used to create the command signal[a]
Digital input	Used to signal if joystick is present

(2) Output signals from computer:
The letters in brackets refer to the letters in Fig. 3:

Signal	**Description**
Control (Actuating) Signal[e]	Used to drive the robot arm
Calculated velocity from robot arm[g]	Used to help create the new control signal

(3) Internal signals (in software, each will be displayed on computer monitor):
The letters in brackets refer to the letters in Fig. 3:

Signal	**Description**
Command Signal[a]	Desired signal of robot arm position
Feed-Forward Signal[b]	The signal output from the feed-forward compensator
Error Signal[c]	The difference between the desired position and the calculated actual position
Conventional Controller Signal[d]	The signal output from the PID-type controller
Control (Actuating) Signal[e]	The signal created by the feed forward signal and the PID controller signal
Filtered and amplified position sensor output signal[h]	The filtered and amplified position signal
Calculated velocity signal[g]	The signal that is calculated by using the filtered position output signal



F(s) is the feed-forward compensator implemented in software
Gc(s) is the PID-type controller implemented in software
Gp(s) is the plant this is part of the hardware
H(s) is the position sensor this is part of the hardware
Amp is an amplifier part of the hardware
V(s) is an algorithm to be determined for velocity feedback
C(s) robot arm position output
R(s) command generated in software

Fig .3. Control Block Diagram

Software Flowchart:
In Fig. 4, the software flowchart is shown. In the table below the blocks are listed and described.

Blocks	**Description**
Initialization	Interrupts will be setup to generate a 200Hz sampling (5ms).
Interrupt Service routine and Performing of PID controller	Send signals from the calculated values of the main program and interrupt service routine to the robot arm and to the monitor at user specified times.
Main Program	Will calculate values for the display, check the keyboard, read the joystick, and generate the command signal.
Keyboard	Gives the user a choice of which signals are to be displayed. Type of controller can also be selected.
Display	Will show all internal and external signals chosen by the user.
Joystick Check	There is the possibility that the robot arm can be driven by a joystick or by PC commands. The connection will be checked and if yes the joystick position will be read and saved for the interrupt service routine. If not, the PC will generate the command signal.
Generate Command Signal	This option occurs when the joystick is not connected. The PC will calculate the command signal and save its value for the interrupt service routine.

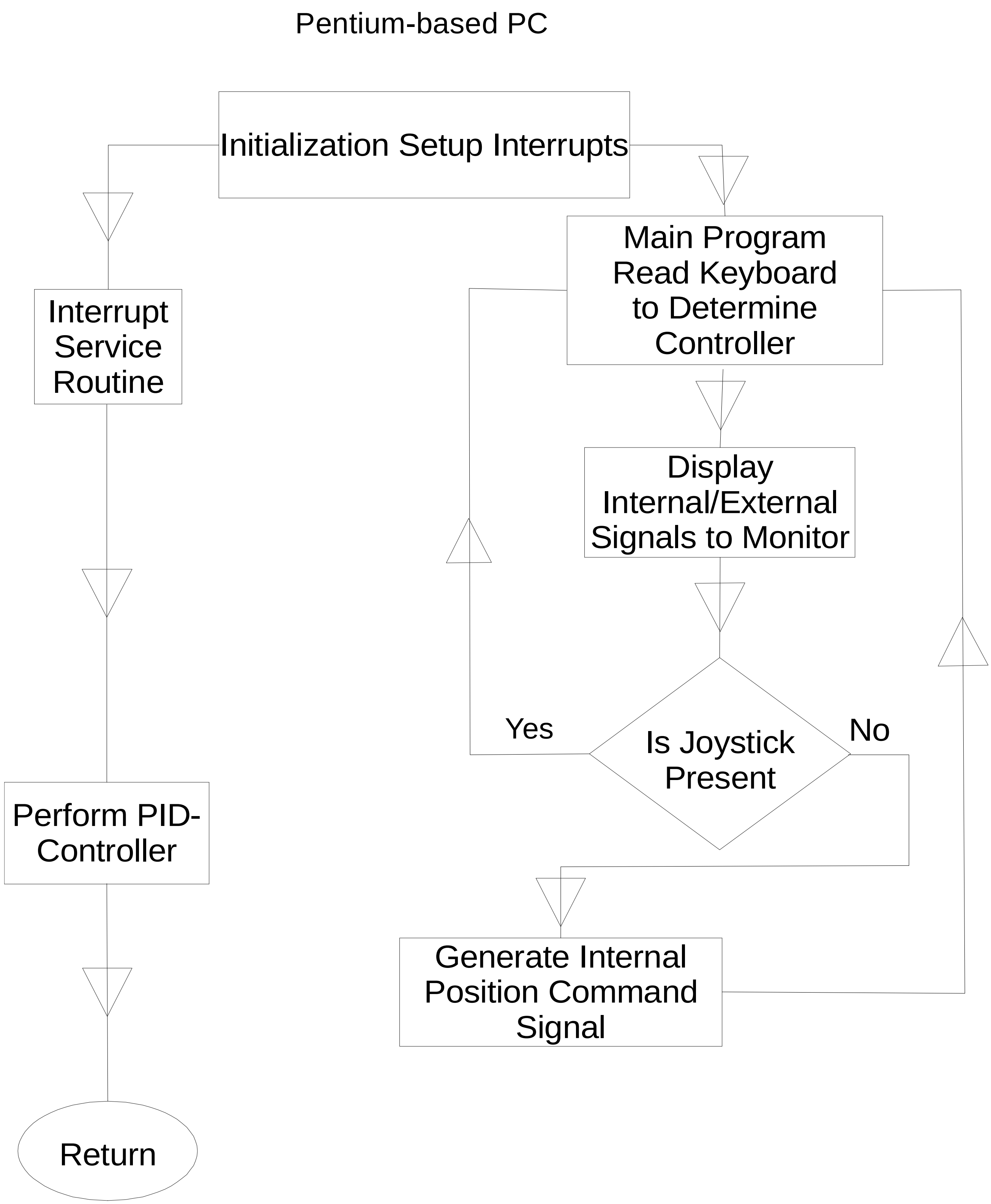


Fig. 4. Software Flowchart

Literature Search:
For our project it was more suitable to perform a literature search on the velocity algorithms that are suitable for our controller method. A tachometer will not be considered for this system because of cost and the low velocities associated with this small robot arm system. Instead we will calculate velocity from a position sensor. Two types of position sensors will be investigated. The analog position sensor consists of a potentiometer connected to a dual ± 5 volt supply. The potentiometer mechanical arm is connected to the external robot arm via a 1:1 gear. Therefore the potentiometer's arm voltage is proportional to position. For example, zero volts would correspond to a robot arm angle of zero degrees. The digital position sensor is a rotary encoder. The advantages of the digital sensor are improvement in reliability, the output frequency is proportional to velocity, and the output is independent of power supply changes. The disadvantage is a loss in position resolution. Both sensors will be used in our controller designs and compared with the different velocity calculation algorithms.

The first velocity calculation algorithm that will be implemented will be based on Tustin's method [1]. Essentially we will design an analog differentiator (phase lead network) and convert it to a digital filter using Tustin's method. Tustin's method is also called the bilinear transformation method. The second method will use a polynomial curve fit algorithm to approximate velocity from current and past position readings. This algorithm is currently being researched by our advisor, Dr. Dempsey. There are numerous velocity calculation algorithms that would require extensive design and implementation time. One method uses a neural network approach to improve the output velocity information by reducing the level of noise [2]. This neural method would require more time than is allowed for our total senior project. The two approaches that we will design and implement will be tried with both types of position sensors and should provide valuable information to Dr. Dempsey for his 2000 senior project group.

Standards:
The Controller software will be implemented in C-code and installed on an IBM compatible computer. The standards for the project are shown in the table below.

Device	**Requirement**
Computer	486 or higher processor IBM compatible Math co-processor
DOS	Version 6.2
Windows	Version 3.11 or higher
1 Bus Expansion	For A/D and D/A converter card

Specifications:

Inputs

1. Step Function: The step function to run the robot arm system is shown in Fig. 5.

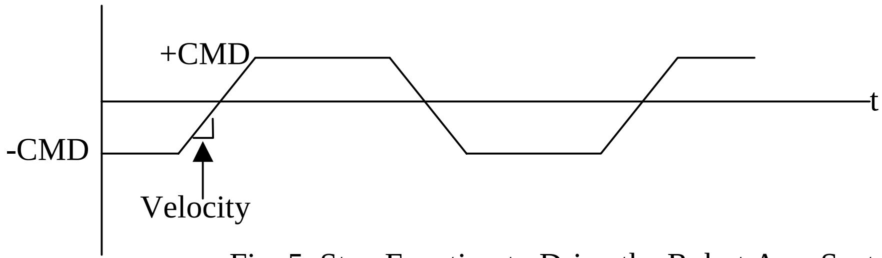


Fig. 5. Step Function to Drive the Robot Arm System

   A. |CMD| ≤ 90°
   B. $Velocity_{max}$ = 45°/sec

Outputs

1. Position
   A. Percent Overshoot(%O.S.) = 5%
   B. Time to First Peak(tp) = 3s
   C. Magnitude of Peak in Frequency Domain(Mp) = 1.32dB
   D. Frequency of Peak(ωp) = 170mHz
   E. Bandwidth Closed Loop(BW_{cl}) = 290mHz
   F. Phase Margin(PM) = 50°
   G. Gain Margin(GM) = 6dB
   H. Steady State Error(ess) < 2°

2. Velocity
   A. Tracking Error < 2°

User Interface:

The user interface for the robot arm system is a computer monitor and a keyboard. The preliminary monitor display is shown in Fig. 6. The keyboard will allow the user to change the different options of the robot arm system. Also, a help screen will be added to add in the keystrokes to change the options.

User controlled setting are shown in the table below.

Options	**Description**
Motor On/Off	Turns motor on or off
Kp gain	Set the gain of the system
Command Set Point	The final robot arm angle
Ramp Velocity	Has to be less than 45°/sec
Step Frequency	The frequency of the step function
Plotting Variable	The variable that is shown on the plot
Controller Type	Type of controller to be used

Interface display shows the following:

Display	Description
Real Time	Time the program has been running
Cycles	How many times the systems has been ran
Actual Position	The position of the arm

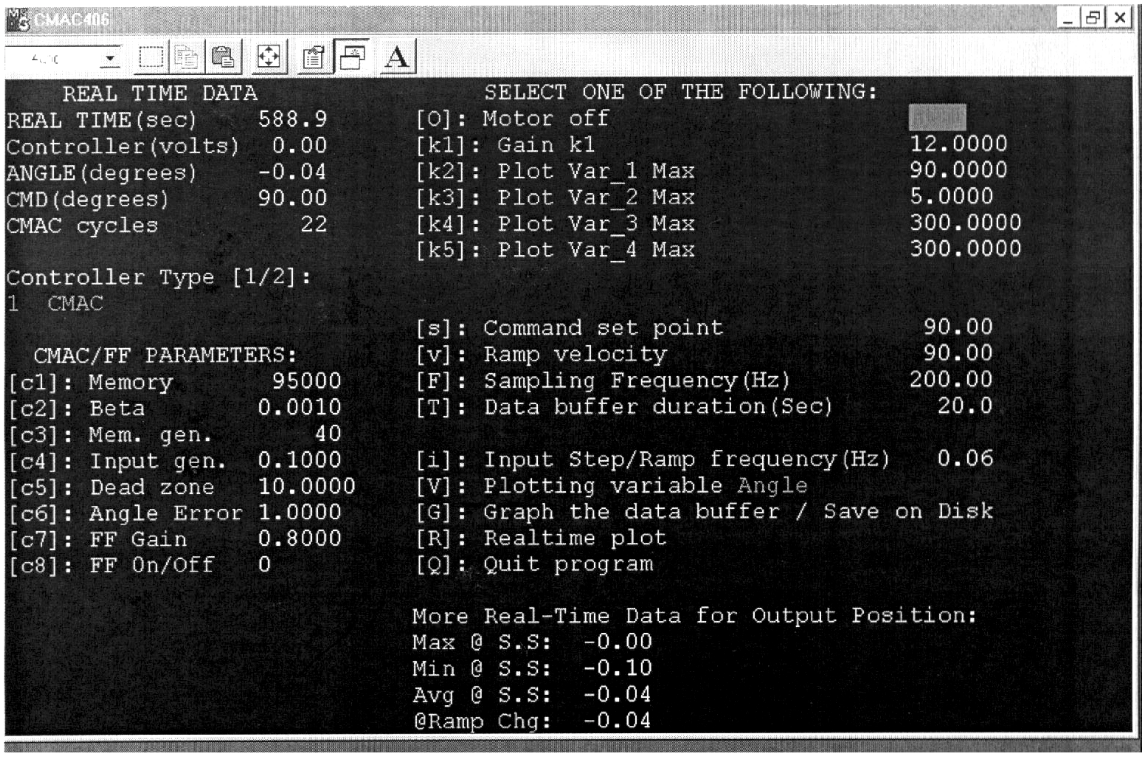


Fig. 6. User Interface

Final Testing:

To test the final system a command signal will be entered. With this command signal the different types of controllers will be used and the final position signal of the robot arm will be shown. With the more advanced type of controllers the final angle will be closer to the desired angle. Then the neural network will be shown using velocity feedback. With this controller the system will learn to go the desired angle. Also, to help show the system works the real-time graphs will be used. To prove that the system is working correctly we will compare the experimental position curve versus time with a curve generated from a simulation environment(MATLAB).

Preliminary Lab Work:
To get some experience in designing and testing a conventional controller we designed and tested a P controller. The following block diagram shows the P-Controller and the plant, where the plant is given as Equ. 1.

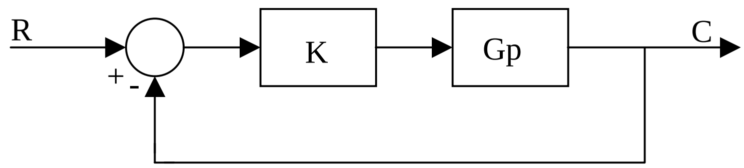


Fig. 7. P Controller

$$Gp = \frac{1}{s(s+1)} \qquad (1)$$

Since we deal with an excact 2nd order system we used the following design equations were used. With percent overshoot zeta can be found by using the equation shown in Equ. 2. After that using equation shown in Equ. 3. Wn can be found. Using Wn the gain value, k, can be found, this equation is shown in Equ. 4.

$$\%OS = e^{-\zeta\pi / \sqrt{1-\zeta^2}} \qquad (2)$$

$$w_n = 1/2\xi \qquad (3)$$

$$w_n = \sqrt{k} \qquad (4)$$

MatLab 5.3 was used to draw the root locus and find the gain needed for a desired overshoot. Then an m-file was created to evaluate the times for settling, first peak, rising, and to find the bandwidth and magnitude of the peak in frequency domain for the closed loop system. We compared the simulation and experimental results. The table shows the simulation results.

Percent Overshoot(%O.S)	Gain(k)	Settling Time(Ts)	Time to First Peak(Tp)
0%	0.25	11.7sec	4
5%	0.525	8.4sec	6.0sec
25%	1.534	6.78sec	2.76sec

Next step was to design a circuit from the block diagram; this circuit is shown in Fig. 8.

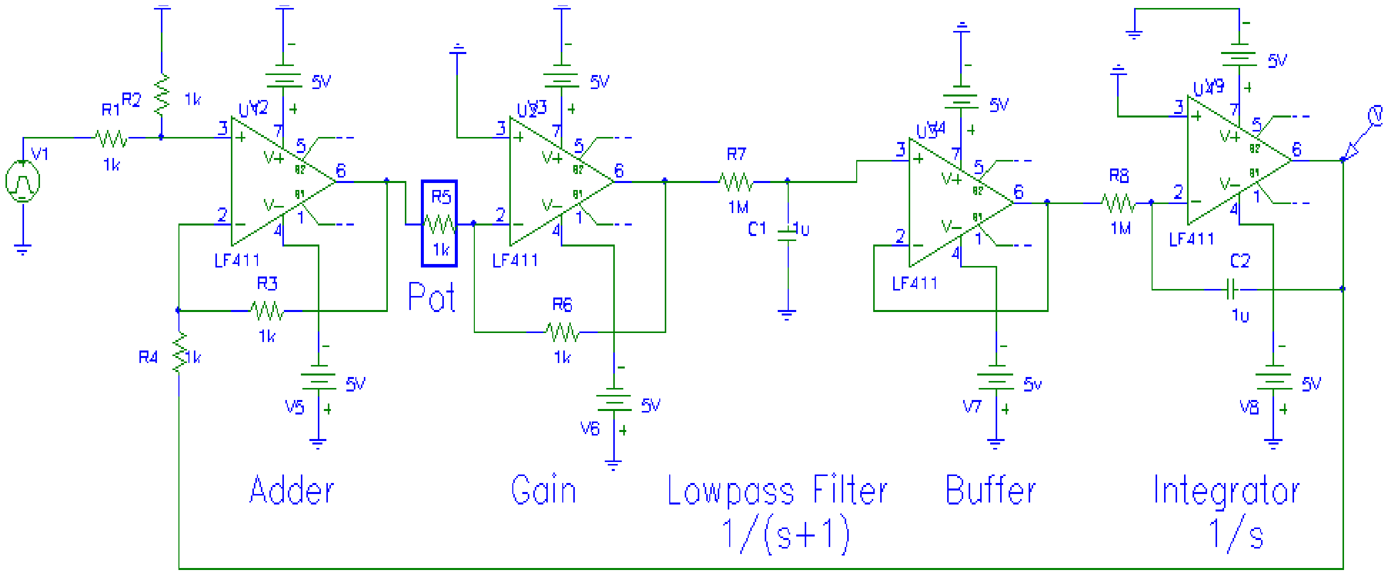


Fig. 8. Circuit Diagram

The calculation for circuitry was done is this manner: 1µF Capacitors for integrator and filter and 1kΩ resistors for gain and summer were chosen. For the integrator and filter, the resistor value was calculated to be 1MΩ to make the gain of one for the integrator and the filter. The table below shows the experimental results.

Percent Overshoot	Resistance of Potentiometer	Calculated Gain
0%	0%	0.25
5%	5%	0.525
25%	652Ω	1.534

Shown in Fig 9. is the output of the circuit shown in Fig. 8. at 25% overshoot.

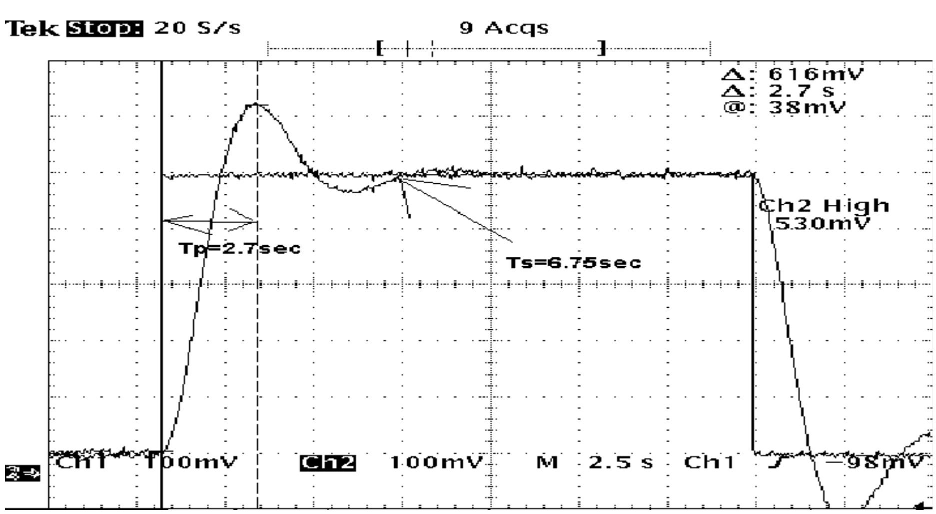


Fig. 9. Output for the 25% Overshoot

The measurement was obtained by choosing a square wave input from at least double the settling time. Then the potentiometer was adjusted to get the desired percent overshoot and results obtained from the scope. For the frequency domain we swept a sine wave starting from 1mHz to get the bandwidth, -3dB down point, and magnitude of the peak.

Comparison between calculation and experimental for 25% overshoot is shown in the table below

Calculation	Measurement
Ts=6.78sec	Ts=6.75sec
Tp=2.76sec	Tp=2.7sec
BW=290mHz	BW=270mHz
fpeak=175mHz	Fpeak=161.8mHz

The comparison shows that the values are pretty close to the calculations. The minor errors resulted from breadboard capacities and in standardized capacitor and resistor values.

C-Code Development:
To get some experience in writing C-Code and designing digital controllers we started designing a digital filter with a 20Hz cutoff frequency. We used Tustin's method, which is also known as the bilinear Z-transformation. The sampling time, T, was chosen to be 0.005sec to make sure the calculations are done before the next input arrives.

$$S = \frac{2}{T}\frac{z-1}{z+1} \qquad (5)$$

The filter transfer function we were looking for was:

$$G(s) = \frac{1}{1+\frac{s}{20*2\pi}} \qquad (6)$$

After calculations the C-Code line for the filter looked like:
Output = 0.239*Input+0.239*Past Input+0.522*Past Output
Where the Past Input & Past Output had to be set to zero to start calculations of the output.

For the measurement we used a 1V sine wave@20Hz to see the 3dB point. The output is shown in Fig.10.

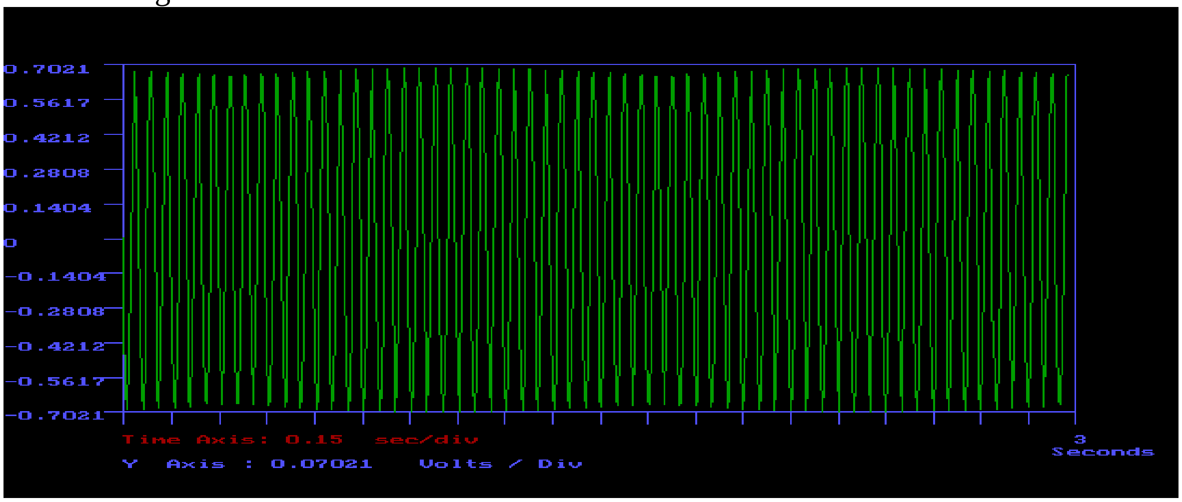


Fig. 10. Output of Filter Using C Program

As it can be seen the output did not look like a sine wave. So the next step was to improve the real time display. An Auto Scaling on the Y-Axis was given and the problem was that it always takes the maximum value from the last screen to calculate the next Y-Axis maximum. This results in a plot off the screen. The improvement was made so that it now takes the maximum value of all screens and adjusts the y-axis only upwards

Equipment List:
Hardware:
Gateway 500MHz IBM compatible Pentium III PC
Quansar Robot Arm System
Quansar A/D & D/A converter card
Quansar Amplifier

Software:
Borland 4.5 C-Compiler
MatLab 5.3
WinCom V2.0 Simulation Program

Schedule:
Since we have only 12 weeks available until the presentation at the Student Expo at the end of, April the schedule turned out to be:

Subproject	Persons	Time(weeks)
System Identification	Chris, Manfred, and Dr. Dempsey	3
Menu	Chris and Manfred	1
P-Controller Design and Testing	Chris	1
Investigate and Implement Neural Network with P-Controller	Manfred and Dr. Dempsey	1
Velocity Algorithm	Chris	1
Two Loop Design With Neural Networks	Manfred and Dr. Dempsey	1
Redesign with Rotary Encoder	Chris and Manfred	1
Feed-Forward Control and Implementation in Neural Networks	Chris, Manfred, and Dr. Dempsey	1
Digital Control Analysis	Chris and Manfred	2

There is additional work to be done:
Presentation at Student Expo
Conference Report
Presentation Board

References:

[1] Charles L. Phillips and Royce D. Harbor. Feedback Control Systems, 4th edition, Prentice-Hall Inc., 1999.

[2] Olli Vainio. "Adaptive Derivative Estimation for Delay-Constrained Acceleration Measurement", *IEEE Transactions on Industrial Electronics*, Vol. 46, No. 5, October 1999.