A Case Study of Rotating Sonar Sensor Application in Unmanned Automated Guided Vehicle

Pravin Chandak, Ming Cao and Ernest L. Hall
University of Cincinnati
Center for Robotics
University of Cincinnati
Cincinnati, OH 45221-0072

ABSTRACT

A single rotating sonar element is used with a restricted angle of sweep to obtain readings to develop a range map for the unobstructed path of an autonomous guided vehicle (AGV). A Polaroid ultrasound transducer element is mounted on a micromotor with an encoder feedback. The motion of this motor is controlled using a Galil DMC 1000 motion control board. The encoder is interfaced with the DMC 1000 board using an intermediate IMC 1100 break-out board. By adjusting the parameters of the Polaroid element, it is possible to obtain range readings at known angles with respect to the center of the robot. The readings are mapped to obtain a range map of the unobstructed path in front of the robot. The idea can be extended to a 360 degree mapping by changing the assembly level programming on the Galil Motion control board. Such a system would be compact and reliable over a range of environments and AGV applications.

1. INTRODUCTION

The purpose of this paper is to study the implementation of rotating sonar and methods for compensating a rotating sonar sensor motor. These include analytical modeling, simulation and experimental approaches. The system analytical model is described in Section 2. The simulation approach is described in Section 3. The experimental approach is described in Section 4. Finally conclusions are given in Section 5.

The motion control system of a mobile robot or unmanned Automated Guided Vehicle (AGV) helps it maneuver to negotiate curves and drive around obstacles. This study has been carried out on the 'Bearcat' mobile robot designed by the Center for Robotics at the University of Cincinnati. The Bearcat uses a combination of obstacle avoidance system and line following vision system to decide its path. The obstacle avoidance uses sonar sensors to detect the obstacles. Data about distance of obstacle is transferred to the processor and it decides the path of the vehicle to avoid them while staying inside the track.

Earlier four stationary sonars were mounted on the front of the vehicle for this purpose. We replaced them with one sonar mounted on a motor which can be rotated to different angles so as to provide readings in different directions. A rotating sonar can not only detect more obstacles but can also help us in mapping there positions more accurately thus providing for better path planning. Also it can be used to follow another vehicle which is not always possible using stationary sonars. Also the number of stops and angle sizes can be modified providing more options.

Selecting the right parameters for the compensator portion of a controller is the most challenging step for in this motion control system design. Also, for rotating sensors, the motion control is critical for positioning the sensor at a given location while a measurement is taken.

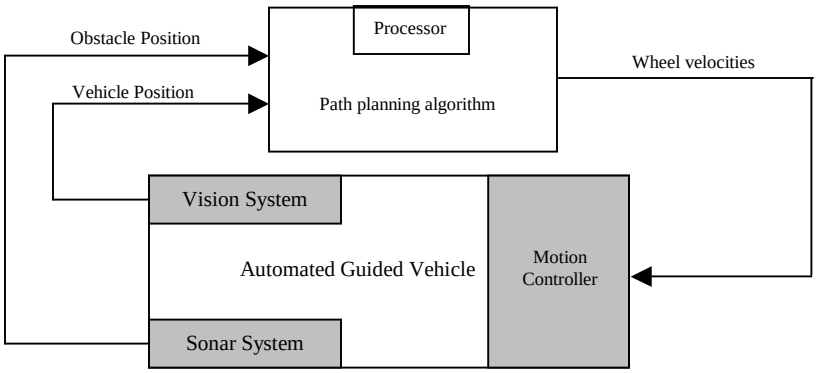


Figure-1. Block diagram of motion control system

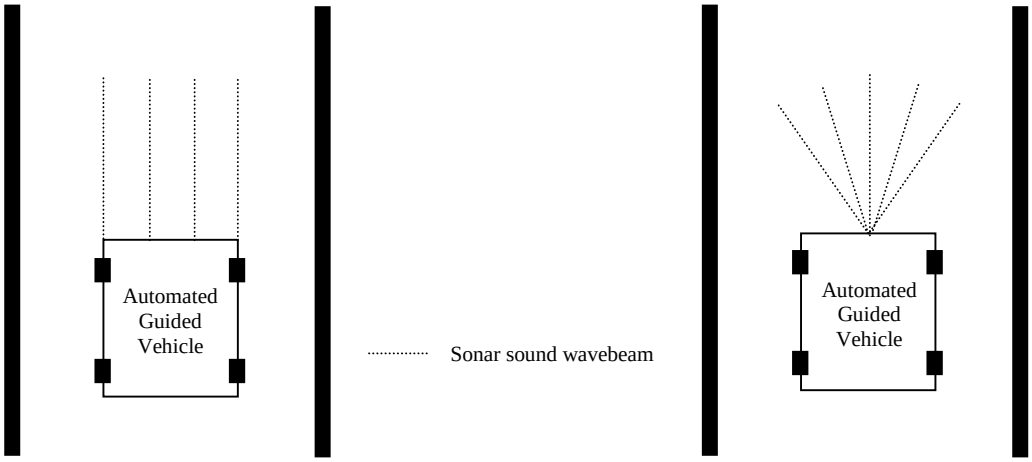


Figure-2. Figures showing range of stationary sonar and rotating sonar

System Modeling

The position-controlled system comprises a position servomotor with (Electrocraft Brush type DC motor) an Encoder, a PID controller (Galil DMC 1030 motion control board) and an amplifier (Galil MSA 12-80).

The sonar system is driven by an Electrocraft brush-type DC servomotor. An encoder provides position feedback for the system. The drive motor is operated in current loops mode using a Galil MSA 12-80 amplifiers. The main controller card is the Galil DMC 1030 motion control board and is controlled through a computer.

Sonar motor and tuning requirements

The current algorithm requires the sonar to be positioned at five equiangular points with the center being exactly in front of the robot. So starting initially at the center we rotate the motor by 15deg, beginning twice in one direction, then four times in the opposite direction and again reverse four (thus forming a cycle of 0,15,30,15,0,-15,-30,-15,0,15,30,15). Also the movement should be rapid as the robot moves forward at quite a high speed and should foresee obstacles as early as possible. At each point the motor stops for a very short time period for the sensor to give accurate readings.

Thus following characteristics are required of the sonar motion

a) Fast response – the sonar should be rotated to the desired position at the fastest speed.
b) Least error – the stops should be accurate without overshoot or undershoot as the errors will become cumulative considering the cycle, and even a small error would render the obstacle avoidance program useless.

Considering the small load - the only load mounted on the sonar motor is the sonar, which is less than a quarter of pound and taking into account the above features, we select Electro-Craft 0260-06-018 servomotor. The encoder of this motor is 2000 counts/revolution.

2. ANALYTICAL MODEL

Motion systems architecture and descriptions

Amplifier mode settings - there are generally 3 modes of configurations for each amplifier, the current mode(torque mode), voltage mode and velocity mode. The current mode will provide large torque while the voltage mode will provide fast speed. During the field test, the robot will always have to concur ramps, high friction grassland, etc. So the two wheel motors were configured as current mode. The only load of the sonar motor is the quarter-pound sonar head. So, the load for the sonar amplifier would be small. Voltage mode would be selected to support a prompt response.

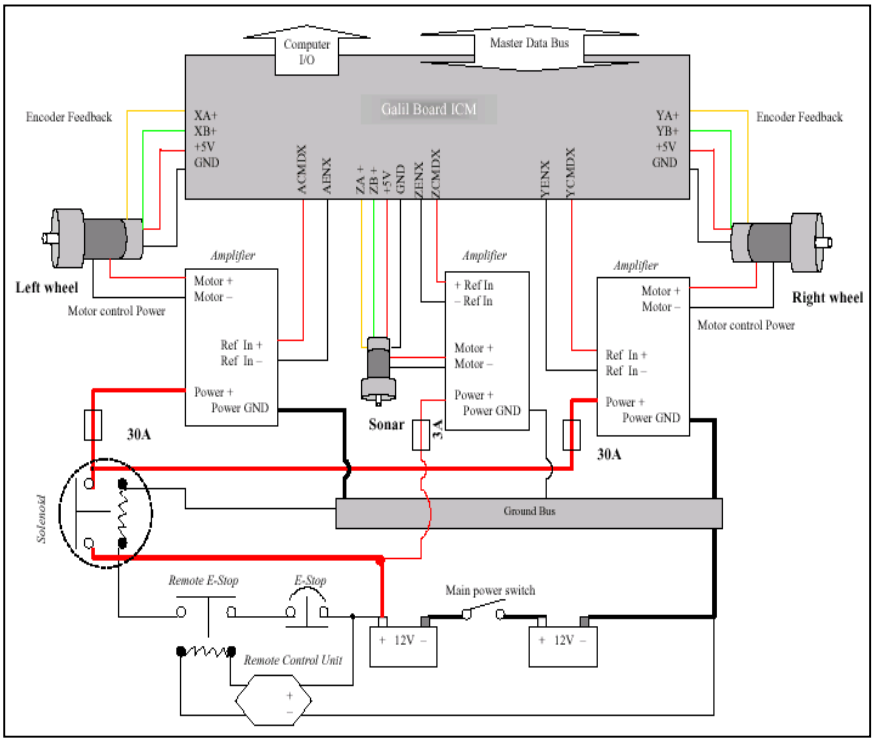


Figure-3. BEARCAT III Motion Control Systems Architecture

Control board and amplifier configuration

The basic configuration between the motor, amplifier and the Galil controller is shown in Fig-3. The Galil motion controller plugs into the PC bus and accepts several high level, ASCII commands. Analog motor command of ±10 V range signal for driving servo amplifiers; 16-bit resolution or. The Galil board is the main interface to transfer data to the computer and also the commands to the amplifier. It gets motor status information via encoder and then reports the data to the computer. The computer command is first sent to the Galil board and then transferred into analog control signal, which is between +/- 10 V via a 16 bit D/A converter (resolution 0003 Volts). This signal is a reference voltage which is applied to the amplifier reference in +/- pins to control the motor direction and speed. Depending on the amplifier mode, the control signal controls current, voltage or speed. The two wheel motor amplifiers is powered by two 24V DC power (2 batteries in serial connection). These amplifiers get control signals from Reference in + and Reference in -, then output voltage/current power to the motor accordingly. The variation of the reference in error will determine the strength of the power output to the motor.

The encoder attached to the motor reads and gives negative feedback information of motor motion status. The negative feedback causes a difference between the command signal and feedback signal. This difference is called the error signal. The amplifier compares the feedback signal to the command signal to produce the required output to the load by continually reducing the error signal to zero. The encoder gets +5V DC power directly from the Galil board. The encoder signal is transferred to the Galil board via pin *A+ and pin *B+. The motor amplifiers get control signals from the Galil controller pin ACMD* and pin AEN* (* means X or Y or Z, depend on the axis). For safety consideration, the motor amplifier is protected by a 30A fuse and the sonar amplifier by a 3A fuse.

Amplifier tuning

There are 4 potentiometers to be adjusted built into the amplifier, loop gain, current limit, reference gain and offset - the loop gain adjustment in voltage & velocity modes; voltage to current scaling factor adjustment in current mode. The current limit adjusts both continuous and peak current limit by maintaining their ratio (50%); the reference gain adjusts the ratio between input signal and output variables (voltage, current, velocity);
The offset/test is used to adjust any imbalance in the input signal or in the amplifier. Before operation, these potentiometers need to be tuned to gain an optimum performance.
The tuning procedures are outlined as following:

1. Initialization
   Apply zero speed command to the amplifier. Reference inputs should be grounded. Turn all the potentiometer to minimum settings (counterclockwise).
2. Offset adjustments
   Put the offset switch in the on position. Trim the offset potentiometer for minimum amplifier output current by observing motor drift. Turn test offset switch off.
3. Loop gain adjustments
   Turn clockwise the loop gain potentiometer until the motor shaft oscillates, then back one turn.
4. Reference gain adjustment
   Reference gain determines the ratio between input signal and output variables (voltage, current, velocity). Turn this potentiometer clockwise until the required output is obtained for a given input signal.
5. Current limit adjustment
   It is important to set the current limit so that the instantaneous motor current does not exceed the specified motor peak current rating. The maximum current output of the Advanced Motion Controls 25A8 is +/-25 A, far less than the motor current limit 60A, this potentiometer could be set as its maximum value (14 clockwise turn).

An encoder is mounted on each drive motor shaft. The encoder feedback positions signals, which can be differentiated to provide the velocity feedback into the controller. The AGV moving direction is achieved by differentiating speed of the drive wheel. These are drive wheels whose speeds can be varied according to the change in the direction of the track being followed.

Sonar motor amplifier

Since the sonar rotation is a light load. A Miniature Brush type Servo Amplifier, MSA-12-80, was selected. It is low-cost, easy to use amplifier for driving brush type servo motors at high switching frequencies. The amplifier utilizes power MOSFETs and surface mount technology to produce high power in a small package. The MSA-12-80 accepts a +/- 10V range input signal directly from Galil control board, or it can be configured as a stand-alone drive. An unregulated DC power supply is required to drive the MSA-12-80.

Modeling the Amplifier can be configured in three modes namely, voltage loop, current loop and the velocity loop. The transfer function relating the input voltage V to the motor position P depends upon the configuration mode of the system.
a. Voltage Loop
In this loop, the amplifier acts as a voltage source to the motor. The gain of the amplifier will be $K_{v.}$ And the transfer function of the motor with respect to the voltage will be

$$\frac{P}{V} = \frac{K_v}{[K_t s(s\tau_m + 1)(s\tau_e + 1)]}$$

where ,

$$\tau_m = \frac{RJ}{K_t^2}\ (s) \quad \textit{and} \qquad \tau_e = \frac{L}{R}(s)$$

The motor parameters and the units are:
K_t : Torque constant (Nm/A)
R : Armature resistance
J : Combined Inertia of the motor and load (kg-m^2)
L : Armature Inductance

b. Current Loop
In this mode the amplifier acts as a current source for the motor. The corresponding transfer function will be as follows
Where,
$K_{a-=}$ Amplifier gain
K_t and J are as defined earlier

$$\frac{P}{V} = \frac{K_a K_t}{Js^2}$$

c. Velocity Loop
In the velocity loop, a tachometer feedback to the amplifier is incorporated. The transfer function is now the ratio of the Laplace transform of the angular velocity to the voltage input. This is given by

$$\frac{\omega}{V} = \frac{\frac{K_a K_t}{J_s}}{1 + \frac{K_a K_t K_g}{J_s}} = \frac{1}{[K_g(s\tau_1 + 1)}$$

where,

$$\tau_1 = \frac{J}{K_a K_t K_g} \quad \textit{and therefore}$$

$$\frac{P}{V} = \frac{1}{[K_g s(s\tau_1 + 1)]}$$

The Encoder
The encoder is an integral part of the servomotor and has two signals A and B, which are in quadrature and 90 degrees out of phase. Due to the quadrature relationship, the resolution of the encoder is increased to 4N quadrature counts/rev. N is the number of pulses generated by the encoder per revolution.
The model of the encoder can be represented by a gain of

$$K_f = \frac{4N}{2\pi}\ [counts/rad]$$

The Controller
The controller in the Galil DMC 1030 board has three elements, namely the Digital-to-Analog Converter (DAC), the Digital Filter and the Zero Order Hold (ZOH).

a. Digital-to-Analog Converter (DAC)
The Digital-to-Analog Converter (DAC) converts a 14-bit number to an analog voltage. The input range of numbers is 16384 and the output voltage is $\pm$10V
For the DMC 1030, the DAC gain is given by K_d=0.0012 [V/count]

b. Digital Filter
The digital filter has a discrete system transfer function given by

$$D(z) = \frac{K(z-A)}{z + \frac{Cz}{z-1}}$$

The filter parameters are K, A and C. These are selected by commands KP, KI and KD, where KP, KI and KD are respectively the Proportional, Integral and Derivative gains of the PID controller. The two sets of parameters for the DMC 1030 are related according to the equations,

$$K = K_p + K_d$$
$$A = \frac{K_d}{(K_p + K_d)}$$
$$C = \frac{K_i}{8}$$

c. Zero Order Hold (ZOH)
The ZOH represents the effect of the sampling process, where the motor command is updated once per sampling period. The effect of the ZOH can be modeled by the transfer function,

$$H(s) = \frac{1}{\left(1 + s\frac{T}{2}\right)}$$

In most applications, H(s) can be approximated as 1.
Having modeled the system, we now have to obtain the transfer functions with the actual system parameters.

System Analysis
The system transfer functions are determined by computing transfer functions of the various components.

- Motor and the Amplifier
  The system is operated in a current loop and hence the transfer function of the motor-amplifier is given by
- Encoder
  The encoder on the DC motor has a resolution of 500 lines per revolution. Since this is in quadrature, the position

$$\frac{P}{V} = \frac{K_a K_t}{Js^2}$$

  resolution is given by 4*500=2000 counts per revolution.
  The encoder can be represented by a gain of

$$K_f = \frac{4 \times N}{2\pi} = \frac{2000}{2\pi} = 318$$

- DAC
  From the Galil manual, the gain of the DAC on the DMC 1030 is represented as
  K_d = 0.0012 V/count
- OH
  The ZOH transfer function is given by

$$H(s) = \frac{1}{1 + s\frac{T}{2}}$$

  Where, T is the sampling time. The sampling time in this case is 0.001s. Hence the transfer function of the ZOH is:

$$H(s) = \frac{2000}{s + 2000}$$

Example of System Compensation Objective :
The analytical system design is aimed at closing the loop at a crossover frequency ω. This crossover frequency is required to be greater than 200 rad/sec. An existing system is taken as a reference and the crossover frequency of that system is used since the two are similar.

The following are the parameters of the system:
The design objective is set at obtaining a phase margin of 45 degrees.

1	Time Constant of the motor	K_t	2.98 lb-in/amp	0.3375N-m/amp
2	Moment of Inertia of the system	J	2.2e02 lb-in^2 (approx.)	2.54e04 Kg-m^2 (approx)
3	Motor resistance	R	0.42 ohms	
4	Amplifier Gain in current loop	K_a	1.2 amps/volt	
5	Encoder gain	K_f	318 counts/rev	

Using above equations we select the filter function of the form G(s)=P+sD;
such that at crossover frequency of 200, it would have a magnitude of 66 and a phase of 50 degrees.

$$|G(j200)| = |P + (j200D)| = 66$$

and

$$Arg[G(j200)] = \tan -1\left[\frac{200D}{P}\right] = 50^{\circ}$$

Solving the equations, we get,
P=42;
D=0.25
The filter transfer function is given by G(s)=0.25s+42.

3. SIMULATION AND OPTIMIZATION

The basic model of the control system was setup-using MATLAB. For this purpose the SIMULINK toolbox was used. This toolbox allows the modeling of the various systems on the control system. Optimization was done using the OPTIMIZATION toolbox.
The objective of the model was to attain a stable control system. One of the main requirements was that the phase margin should be less than 45 degrees and a gain margin more than 10 decibels with the percentage overshoot not exceeding 20%. The Galil DMC 1000 controller has a proportional integral and derivative controller to provide the necessary compensation. The simulation involves three steps – i.) a Matlab file that has the model of the transfer function; ii.) a second Matlab source file converts the digital gains to analog gains; iii.)A Simulink graphics model which takes the analog values of the gains and simulates the system step response.

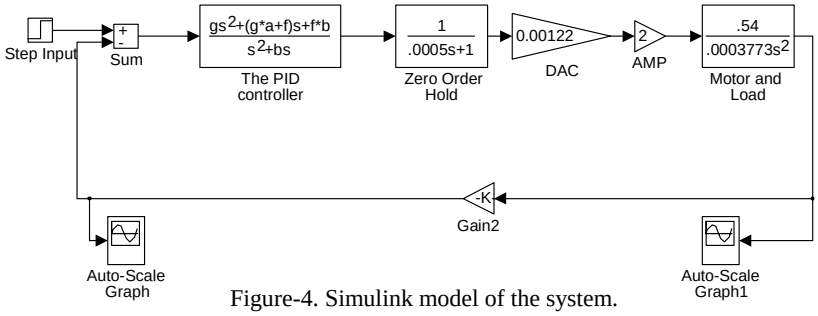


Figure-4. Simulink model of the system.

The model consists of a step input signal fed to a summation block. The constant values to be used in the PID block are calculated using a separate M-file. These values calculate the analog gains for the various digital gains. The calculated analog values are stored in the Matlab kernel and, are read automatically when the model file is run. Analog values in the PID controller adjust the input signal and feed it to the Zero order hold. The zero order hold holds the input level until the next input is given in order to smoothen the input wave. Sampling time for this is modeled in the block. The signal is then fed into the DAC (digital to analog converter) and then to the amplifier. The amplifier gain is set at 2 but could be changed on the actual device to suit the amplification needed. The amplified signal is fed to the load which in our case is the overall system including the motor and drive train. The motors have encoders that give the position feed back signal, which is fed to a summation block for correction. From the simulation it was found that the phase margin was within tolerable limits and the overshoot was less than 15%. These are observed in the magnitude and phase of the Bode plots.

Optimizing the control system to obtain PID values

The control parameters in the model above are optimized using a solver. This is a non-linear model to be optimized. This would enable the system to track a unit step input to the system with minimum error. MATLAB's optimization toolbox is used to solve the non-linear problem. The way this problem is tackled is in two steps. 1. The first method minimizes the error between the output and the input signal using *'lsqnonlin'* function 2. The second method uses the *'fminimax'* function. In this case, rather than minimizing the error between the output and input the function minimizes the maximum value of the output at any time. The variables are the parameters of the PID controller. If the error needs to be minimized only at one time, it would be a single objective function. But, we need to minimize the error for all time steps so it becomes a multiobjective function.

The lsqnonlin method

The routine lsqnonlin is used to perform a least square fit on the tracking of the output.

E_i = Input – Output

The objective would be to minimize the mean squared error , i.e.

Minimize

$$\overline{E} = \frac{1}{N}\sum_{i=1}^{N} E_i^2$$

The routine lsqnonlin is used to perform a least squares fit on the tracking of the output. This is defined via a function in a separate M-file called 'lsq.m'. It defines the error signal. The function 'lsq' runs the simulation. Shown below is the file

Step 1: File lsq.m

```
function F = lsq(pid, a1, a2)
Kp = pid(1); % Move variables into model parameters
Ki = pid(2);
Kd = pid(3);
% choose solver and set model workspace to this function
opt = simset('solver', 'ode5', 'SrcWorkspace', 'Current');
[tout, xout, yout] = sim('optsim', [0 100], opt);
F = yout-1; % compute error signal
```

The simulink model file is specified in another M-file called the 'optroutine'. The model is the one shown above.

Step 2: File optroutine.m

```
kolmodel % loads the model
pid0 = [1 10 12] % setting initial values
a1 = 0; a2 = 0.0005;
options = optimset('LargeScale', 'off', 'Display', 'iter', 'TolX', 0.001, 'TolFun', 0.001);
pid = lsqnonlin('lsq', pid0, [], [], options, a1, a2)
% put variables back into the base workspace
Kp = pid(1); Ki = pid(2); Kd = pid(3);
```

First the variables Kp, Kd, Ki are all defined and initialized before calling 'lsq'. When the 'optroutine' file is run the model is loaded and function 'lsq' does the optimization based on the initial values of specified for the parameters and, constraints. A solver called simset is chosen and the simulation is run.

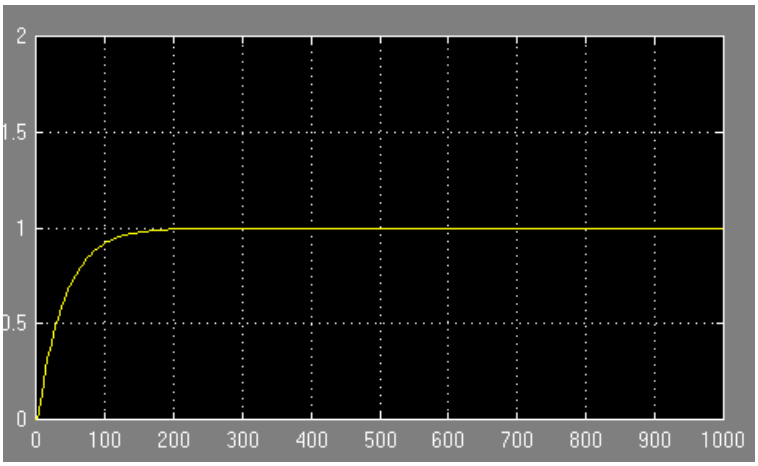
Finally the optimization gives the solution for the Proportional, Integral, and Derivative (Kp, Ki, Kd) gains of the controller after 55 function evaluations. The result from Matlab is shown below.

pid0 = 1 5 10

Iteration	Func-count	Residual	Step-size	Directional derivative	Lambda
1	3	4.76719e+012	1	-4.91e+012	
2	10	4.70723e+012	0.375	-1.51e+011	1.32032e+012
3	17	4.65431e+012	0.253	-2.33e+011	8.77204e+011
4	24	4.58281e+012	0.293	-8.38e+010	1.68981e+012
5	31	4.56367e+012	0.0868	8.32e+010	2.02202e+012
6	39	4.53102e+012	0.119	-1.63e+011	2.60576e+012
7	55	4.53102e+012	3.28e-007	-3.27e+011	1.07078e+018

Optimization terminated successfully:
Search direction less than tolX
pid = 0.7234 4.9399 10.0233

The final optimized parameter values are Kp = 0.7235; Ki = 4.9399; Kd = 10.0233
Now, these optimized values are put into the PID block and the simulation is run. The step response obtained is shown below.



Time (seconds)

Figure-5. Step response using the optimized values.

4. WSDK METHOD

The Windows Servo Design Kit (WSDK) software helps us in the configuration, tuning, testing, and analysis of the Galil Motion Controllers. WSDK has a number of functions that simplify and aid in setting up and evaluating a complete servo system. These include tuning the servo system, monitoring and recording data using the storage scopes and evaluating system performance.

As stated, when operating a Galil Motion Controller with servo motors, the system parameters for each axis must be established to provide optimal servo operation. The main parameters to be determined for optimal performance KP, KD, and KI can be determined manually or automatically through the tuning section of the WSDK software.

Setup & Running WSDK

To communicate with the Galil controller we need to first set it up. This involves registering it in the Windows registry. We need to setup the appropriate parameters for the controller. We have used a DMC-1000. We select an I/O port address (default 1000) and an interrupt (IRQ). We can use WSDK with multiple controllers at the same time. Communication with the controller can be done using the terminal to send commands.

Tuning Methods

There are 4 built-in tuning methods available with the WSDK - Auto Crossover Frequency, Crossover Frequency, General Tuning, Conservative Tuning, along with an option for manual tuning. A brief description of the four tuning methods provided by the Galil kit are as follows -

General Tuning

This routine tunes the system by increasing each gain until instability occurs, then decreases the gain. The WSDK finds the best KP for a given value of KD. Once the best KP is found, KD is increased and WSDK finds the best KD for that KP value. This process is repeated until KD cannot be increased further without causing instability in the system. WSDK then decreases KP and KD to stable values. It then determines the highest value of KI which does not cause instability.

Automatic Crossover Frequency

This method attempts to determine the PID parameters that correspond to the crossover frequency of the system. This routine is used if the system's bandwidth is unknown.

Crossover Frequency

The Crossover Frequency routine allows defining a crossover frequency (rad/sec), and then attempts to tune the axis to provide the best system response at that frequency. The value of the crossover frequency should be set to the bandwidth of the system for best results. [We can choose a frequency between 5 and 2000 rad/sec.] Lower frequencies tend to over damp the system, resulting in a slower and more sluggish response. The higher frequencies produce faster system response, but because it tends to under damp it may cause the system to overshoot.

Conservative Tuning

The Conservative Compensation routine tries to find a gain setting that will not cause "buzzing" or overdrive the system. As the name suggests, the PID parameters are chosen to avoid any instability in the system but still provide a responsive system. The resulting parameters are very conservative relative to the ones from the other tuning methods.
This tuning method has two-parts. The first part is a "static" test and the second part is a "dynamic" test. In the static test, the gain is increased and the system is excited by a small pulse until the system becomes unstable. Then the gain value is reduced and the results are displayed. In the dynamic test, the gain is increased while the system is in motion. Once the system becomes unstable, the motion is stopped and the gain value is reduced.

Point to Point Tuning

This tuning method allows us to define "settling". We specify the error limit and time for which the system must stay within those limits. The program then iterates through the PID parameters within the specified range to determine which combination results in the fastest settling time. The larger the error limit and the shorter the time the system must stay within those limits, the faster the settling time.
To reduce the number of iterations we can use one of the auto-tuning methods to estimate the range of PID values and, then, use this method to fine tune them.

About the Manual Tuning

The Manual Tuning method provides a means of testing the responsiveness of your system. This method allows to choose the gain and damping of your system. Unlike the other tuning methods, this is not an automatic routine. All tuning parameters may be adjusted with sliders or by entering a value. The changes are in real-time which results in instant feedback regarding the effect of the parameters on the system's stability. We can test the system after any change.

Running the Tuning Methods

To run any method, we select a tuning method and the axis to be tuned. We also select to show a step response based on the current PID filter settings to provide a visual indication of how well the system is tuned. We can change most of the parameters involved in the tests. The Auto-Crossover Frequency and Crossover Frequency tuning methods send pulses or offsets to the diagnosed axis. The values of the Pulse Magnitude (Volts) and the Pulse Duration (msec) can be varied. For General Tuning Parameters the size of the step pulse may need to be adjusted. For point to point tuning the distance, speed, acceleration, and deceleration may be changed by entering a new value or using the slider. Other limits may be imposed on this tuning method. Entering the value or using the slider can define a maximum voltage contribution by the KI parameter. The range of PID parameters that the method iterates through and the increments at which it will iterate by may be designated here. Also, the amount of time the program will wait for the system to settle before aborting the test can be entered. We can change any of the following items: Dwell Time, Step Size, Acceleration, Deceleration, and Maximum Speed, as well as some display options. You can either type in new values or use the sliders to change the values. When you are done, click the OK button to return to the previous screen. If you wish to save these values permanently, click the Save button. If you wish to restore the values to their defaults, click the Defaults button. If you do not want to keep the changes you made, click the Cancel button.

WSDK also provides storage scopes to record the actions of your controller and display them on the screen. It is also possible to display a derivative of the data you have recorded.

System Evaluation

WSDK provides a section on system evaluation which allows us to check the response characteristics of the system. This can be done in different ways. The first method relies on evaluating the step response which is displayed as part of the tuning procedure. The second method requires that the user write a program, run it, and evaluate the system by comparing the actual profile to the commanded profile. Another way would be to directly use the System Evaluation section of WSDK There are three tests that can be performed by the System Evaluation Section of WSDK.

Absolute Stability Test

The Absolute Stability test saturates the system to determine if it is conditionally unstable. The system is subjected to a pulse at the maximum output voltage of the controller (10V). If it is not unstable, it will remain stable unless under extreme or unusual conditions. The test displays the actual position and the requested torque on separate scopes. Typically, a well-tuned system will generate a position profile that consists of a single overshoot on the trailing edge. Excessive ringing on the trailing edge is a sign of instability and reveals that the PID parameters are not optimal. If the position profile shows no overshoot, the system is over damped; in which case, it is not tuned for the maximum bandwidth. The requested torque graph allows the user to determine whether the impulse size needs to be adjusted for the test. If the torque goes to the maximum of 10 Volts, the controller is saturated and will not reflect proper test results. If the torque is less that 1 Volt, the impulse response may not detect the instability within the system.

Frequency Response on Open Loop and Frequency Response on Closed Loop

The system is subjected to a range of open loop/ closed loop frequencies in order for the user to analyze the response of the system. The actual position and positional error are plotted in the upper scope. The lower scope displays the system gain as a function of frequency. This data can aid the user in determining whether further tuning or refining is necessary.

Step Response Test

This test displays the actual position so the user can determine whether further tuning or correction to the system is necessary.

We use both the frequency response tests along with the step response test results to aid us in manually tuning the motor as per our requirements.

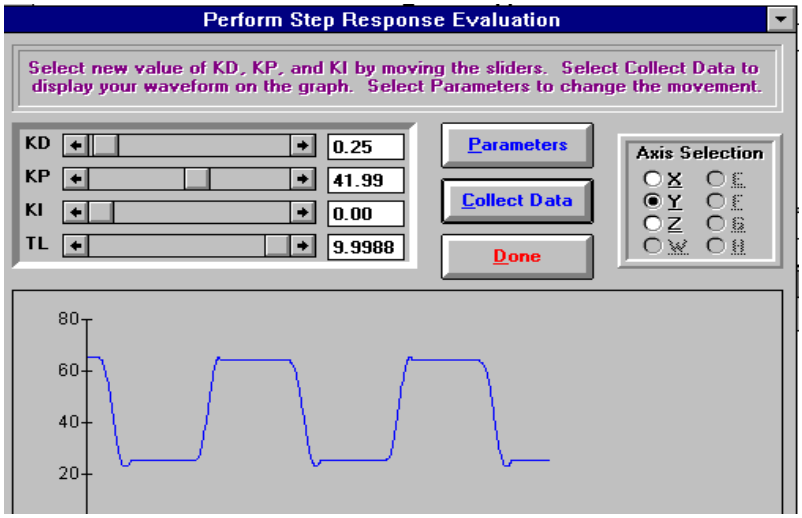


Figure 6. Step Response Plot from the Galil servo design kit.

5. CONCLUSION

The new rotating sonar system is certainly an improvement over the older stationary system in terms of better obstacle avoidance as well as lesser use of resources. This is supported by the distance traveled by the robot in this years contest compared to earlier showings. All the tuning methods have been used variedly over a period of time on the motors driving the wheels as well as to tune the sonar motor. The values from WSDK tuning were used in this years contest and provided good performance. It is difficult to incorporate the load factor on motor in the simulation and optimization methods whereas WSDK carries it out in real environment. That's because it's difficult to model precisely each component such as time delay. On the other hand simulation method offers more independence with the variables.

6. REFERENCES

1. Ernest L. Hall, Krishnamohan Kola and Ming Cao, "Fundamentals of Digital Motion Control," Handbook of Industrial Automation, Marcel Dekker, New York, 2000, pp. 157-175.
2. Ming Cao, "Development of Motion Control Systems for an Autonomous Vehicle," MS Thesis, University of Cincinnati, 1999.
3. Sathish K. Shanmugasundaram, "Control System Design for an Autonomous Mobile Robot, MS Thesis, University of Cincinnati, 2000, pp. 63-73.