2001-12

The Naval Postgraduate School's Small Robotics Technology initiative initial platform integration and testing

Chicoine, Andrew G.

Monterey, California. Naval Postgraduate School

NAVAL POSTGRADUATE SCHOOL
Monterey, California

THESIS

THE NAVAL POSTGRADUATE SCHOOL'S SMALL ROBOTICS TECHNOLOGY INITIATIVE: INITIAL PLATFORM INTEGRATION AND TESTING

by

Andrew G. Chicoine

December 2001

Thesis Advisor:	Richard M. Harkins
Second Reader:	Thomas J. Hofler

Approved for public release, Distribution is unlimited

THIS PAGE INTENTIONALLY LEFT BLANK

REPORT DOCUMENTATION PAGE			Form Approved OMB No. 0704-0188
Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instruction, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188) Washington DC 20503.			
1. AGENCY USE ONLY *(Leave blank)*	**2. REPORT DATE** December 2001	**3. REPORT TYPE AND DATES COVERED** Master's Thesis	
4. TITLE AND SUBTITLE: Title (Mix case letters) The Naval Postgraduate School's Small Robotics Technology Initiative: Initial Platform Integration and Testing			**5. FUNDING NUMBERS**
6. AUTHOR(S) Chicoine, Andrew G.			
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Postgraduate School Monterey, CA 93943-5000			**8. PERFORMING ORGANIZATION REPORT NUMBER**
9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES) N/A			**10. SPONSORING / MONITORING AGENCY REPORT NUMBER**
11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.			
12a. DISTRIBUTION / AVAILABILITY STATEMENT Approved for public release, Distribution is unlimited			**12b. DISTRIBUTION CODE**

13. ABSTRACT ***(maximum 200 words)***

The Naval Postgraduate School's Small Robotic Technology (SMART) Initiative is an ongoing research effort within the Combat Systems Science and Technology Curriculum that engages in forward-looking applications of small robotic technology for military employment. The goal of the program is to develop a multipurpose robotic platform that is capable of hosting varied sensor packages for military research. This thesis successfully modified a Foster Miller Lemming tracked vehicle: Payload volume was increased to allow for ease of systems testing and access while incorporating a method for deploying varied sensor modules on a towed sled. Original micro-controller hardware has been replaced with a COTS system that allowed for simplified interfacing with a Honeywell digital compass and a Motorola G.P.S. card. Communications with the Robot were provided through the Internet via a modem. A control interface for use on a personal computer was implemented by creating a JAVA application; the control interface has also been converted to a JAVA applet that the Robot is capable of downloading to a user via a web interface. Follow on research will fully integrate the Robot with a variety of sensor packages including a synthetic array seismic sonar, visual and infrared devices and chemical detection devices.

14. SUBJECT TERMS Robotics, Autonomous, Micro-Controller, Embedded Processor, Differential GPS, Digital Compass, C, JAVA, HTML, Web interface, Man-portable.			**15. NUMBER OF PAGES**
			16. PRICE CODE
17. SECURITY CLASSIFICATION OF REPORT Unclassified	**18. SECURITY CLASSIFICATION OF THIS PAGE** Unclassified	**19. SECURITY CLASSIFICATION OF ABSTRACT** Unclassified	**20. LIMITATION OF ABSTRACT** UL

NSN 7540-01-280-5500 Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. 239-18

THIS PAGE INTENTIONALLY LEFT BLANK

Approved for public release, Distribution is unlimited

THE NAVAL POSTGRADUATE SCHOOL'S SMALL ROBOTICS TECHNOLOGY INITIATIVE: INITIAL PLATFORM INTEGRATION AND TESTING

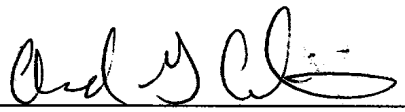
Andrew G Chicoine
Lieutenant, United States Navy
B.S., University of Idaho, 1995

Submitted in partial fulfillment of the
requirements for the degree of

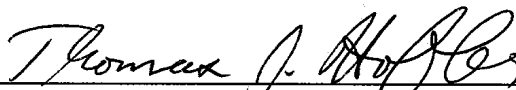
MASTER OF SCIENCE IN APPLIED PHYSICS

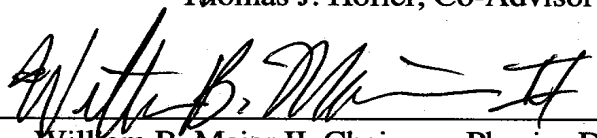
from the

NAVAL POSTGRADUATE SCHOOL
DECEMBER 2001

Author: 
Andrew G. Chicoine

Approved by: 
Richard M. Harkins, Thesis Advisor


Thomas J. Hofler, Co-Advisor


William B. Maier II, Chairman Physics Department

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

The Naval Postgraduate School's Small Robotic Technology (SMART) Initiative is an ongoing research effort within the Combat Systems Science and Technology Curriculum that engages in forward-looking applications of small robotic technology for military employment. The goal of the program is to develop a multipurpose robotic platform that is capable of hosting varied sensor packages for military research. This thesis successfully modified a Foster Miller Lemming tracked vehicle: Payload volume was increased to allow for ease of systems testing and access while incorporating a method for deploying varied sensor modules on a towed sled. Original micro-controller hardware has been replaced with a COTS system that allowed for simplified interfacing with a Honeywell digital compass and a Motorola G.P.S. card. Communications with the Robot were provided through the Internet via a modem. A control interface for use on a personal computer was implemented by creating a JAVA application; the control interface has also been converted to a JAVA applet that the Robot is capable of downloading to a user via a web interface. Follow on research will fully integrate the Robot with a variety of sensor packages including a synthetic array seismic sonar, visual and infrared devices and chemical detection devices.

THIS PAGE INTENTIONALLY LEFT BLANK

TABLE OF CONTENTS

LIST OF FIGURES

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF TABLES

THIS PAGE INTENTIONALLY LEFT BLANK

I. INTRODUCTION

A. WHY ROBOTICS?

Since the time of Robby the Robot in "Forbidden Planet", man has dreamed of using the mastery of technology to assist him with the toils of daily life. Since robots don't require food or water, since they don't complain, and because they are inexpensive and expendable, they seem to be a good fit for the routine, stealthy, environmentally hazardous (chemical, biological and radioactive) and dangerous tasks usually delegated to humans.

Today, robots are used throughout the spectrum of military conflict. Modern examples include de-mining operations in Kosovo and Surveillance flights over Afghanistan. Projections for future uses include urban surveillance, scout missions from the sea, and chemical-biological hazard area detection.

1. Force Multiplication/Casualty Reduction

Robots can be used to greatly increase the combat effectiveness of the individual soldier. For example, a robotic sensing platform will allow the soldier to observe the battle-space with senses he does not naturally have. This could include infrared, ultra-violet and enhanced acoustics sensors. Additionally, the expendability of the robot will allow the soldier to expand the location of his senses to places where personal surveillance would be too dangerous. The result is the ability to maintain battlefield superiority while placing fewer and fewer forces in harms way.

2. Military Operations Other Than War (MOOTW) Missions:

a. Surveillance

The ability of a robot to conduct covert surveillance will perform a major role in the reduction of casualties in war. Although it is doubtful that a robot would eliminate the need for manned surveillance of prospective points of an invasion, robotic surveys could help to identify and eliminate candidate locations before lives are lost in efforts to survey heavily mined or defended locations. As shown in figure 1, a sea launched crawler could covertly swim ashore using a variety of sensors to detect the presence of buried mines, spot hidden bunkers and then relay information to nearby

launch platforms. The conceived surveillance mission could either continue until the robot runs out of power or returns to the launch platform.

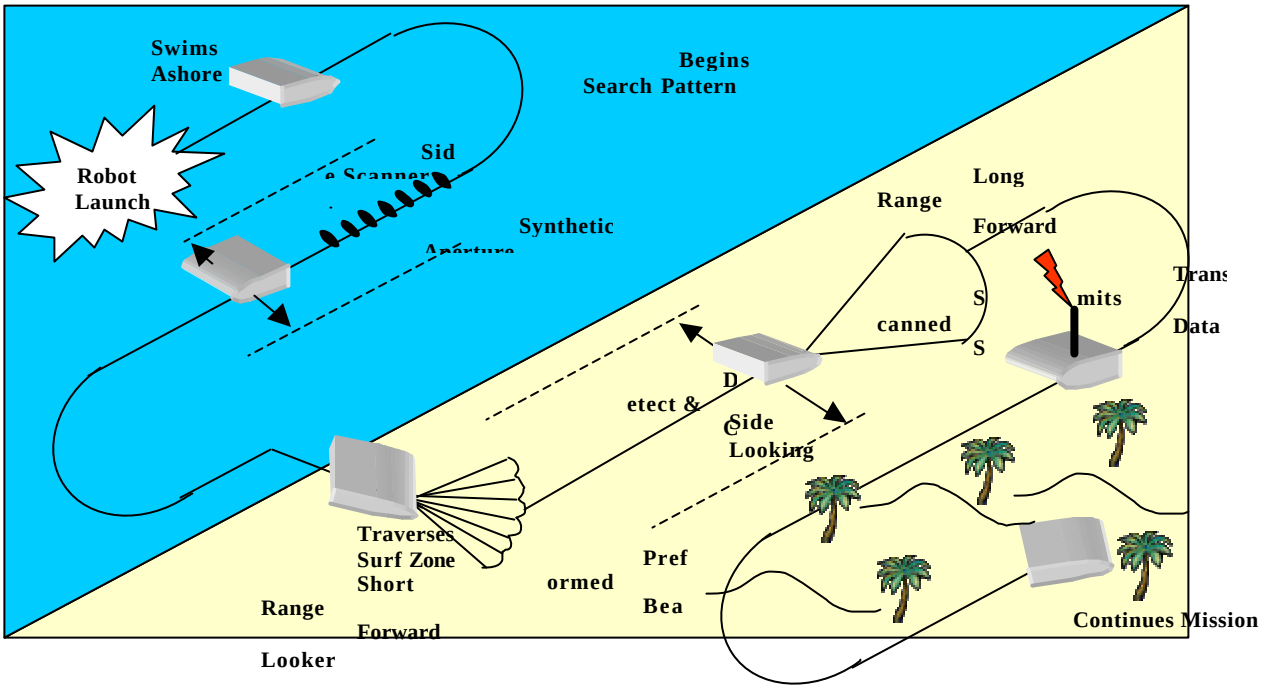


Figure 1. Conceptual Autonomous Robotic Mine Reconnaissance Mission. [From Ref. 25]

b. Mine detection and removal/Explosives Ordinance Disposal.

The utility of a robotic platform is clearly obvious in the tasks of explosive ordinance disposal and mine detection. The Talon robot built by the Foster-Miller Company is used in the diffusing of mines and unexploded ordinance. The robot uses a two-way RF link providing video and operator manipulability of the control arm. During operations in Bosnia the Talon received extensive use and received generally high marks.

Mine detection schemes for use with robotic platforms vary from the use of simple metal detectors for the detection of buried metallic mines to the current NPS project of deploying a towed Seismo-Acoustic sonar behind a robotic platform.

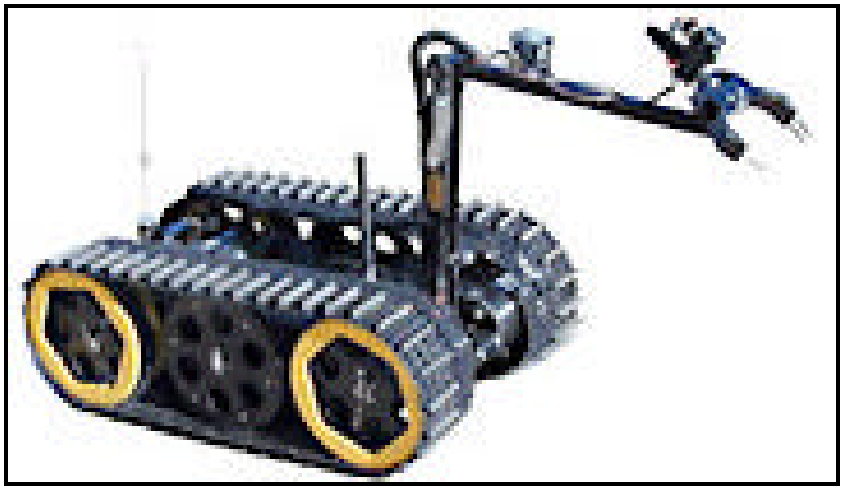


Figure 2. Talon (Foster-Miller) [From Ref. 9]

B. SMART VISION

- Create an ongoing research effort within the CSS&T Curriculum that engages in a forward looking application of Small Robotic Technology for Military Employment
- Develop a Multipurpose Robotic Platform capable of hosting varied sensor packages.

The specific goals accomplished during this thesis research included:

- The successful operational demonstration of a lemming tracked vehicle in manual and autonomous modes.
- The unique integration of diverse COTS components into a single operational platform.
- The successful programming and integration of a GPS and digital compass navigation system.
- The smooth interface between the microprocessor and the platform motors using a self designed PWM motor control circuit and COTS electronic speed control units.
- The establishment of wireless networking for communications using TCP/IP socket connections.
- The creation of a user friendly GUI using a JAVA application and the subsequent conversion of this GUI to a downloadable JAVA applet.

1. Platform

a. Current

The SMART platform is a modified Foster-Miller Lemming. The computing, sensing and communication elements have been removed and replaced with configurable and programmable COTS equipment. An electronics enclosure was added to increase the available area for components. It also acts as a base designed to interface with modular sensor units. While the current platform operates at only a walking pace and has a large vibration and noise signature due to its metallic gearing, future iterations of the platform will include motor and gearing options to enhance performance as well as belt drives to reduce inherent vibrations and noise.

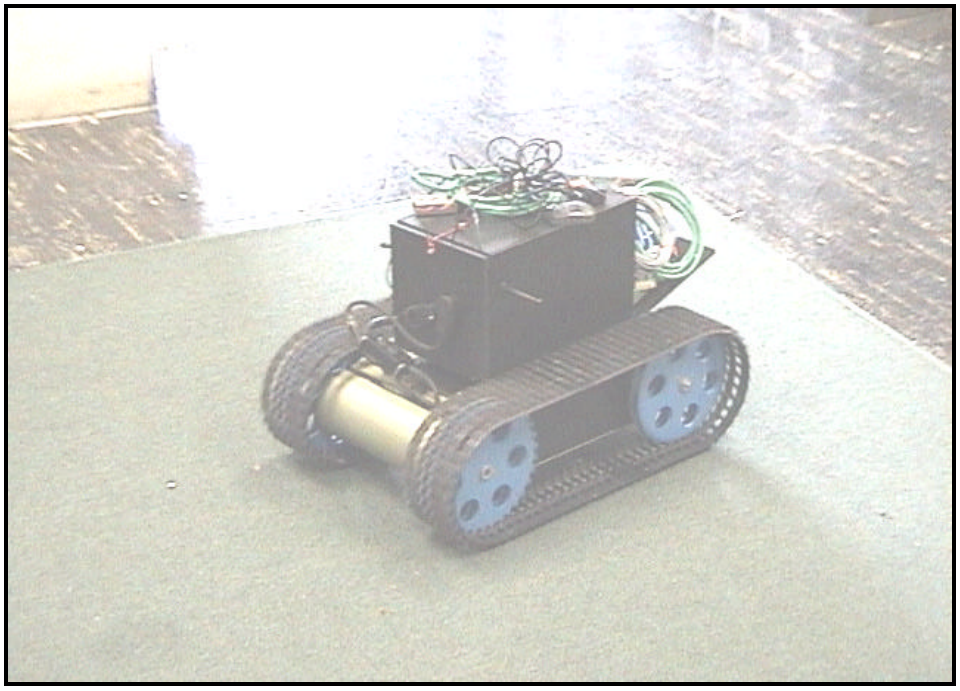


Figure 3. Bender in a developmental stage

b. Communication

Communication with the SMART robot is handled via an internet interface using sockets and IP addressing. A client JAVA application establishes sockets

for use in the transmittance and receipt of information. Future plans for a WEB interface will eventually allow remote operators to control the platform from a secure location via a standard web browser.

c. Basic Operation

The heart of the SMART platform is the "ALLTASK.C" program, which is compiled to the BL2000 microprocessor. The program communicates with the client JAVA control application via a standard wireless Ethernet link. Rs-232 serial channels on the BL2000 are used to communicate with the onboard GPS and Digital Compass. Both devices provide position and heading up-dates once per second.

Movement commands, which include speed, course and GPS waypoint positions, are initiated via the client Java application and are sent to the robot as Ethernet packets over a wireless connection. The BL2000 microprocessor interprets these commands and, in turn, drives the plant (motors) via a Pulse Width Modulator (PWM).

Remaining digital I/O and serial ports on the BL2000 are reserved for future expansion of the basic platform and communications with future sensor arrays. (See figure 4.)

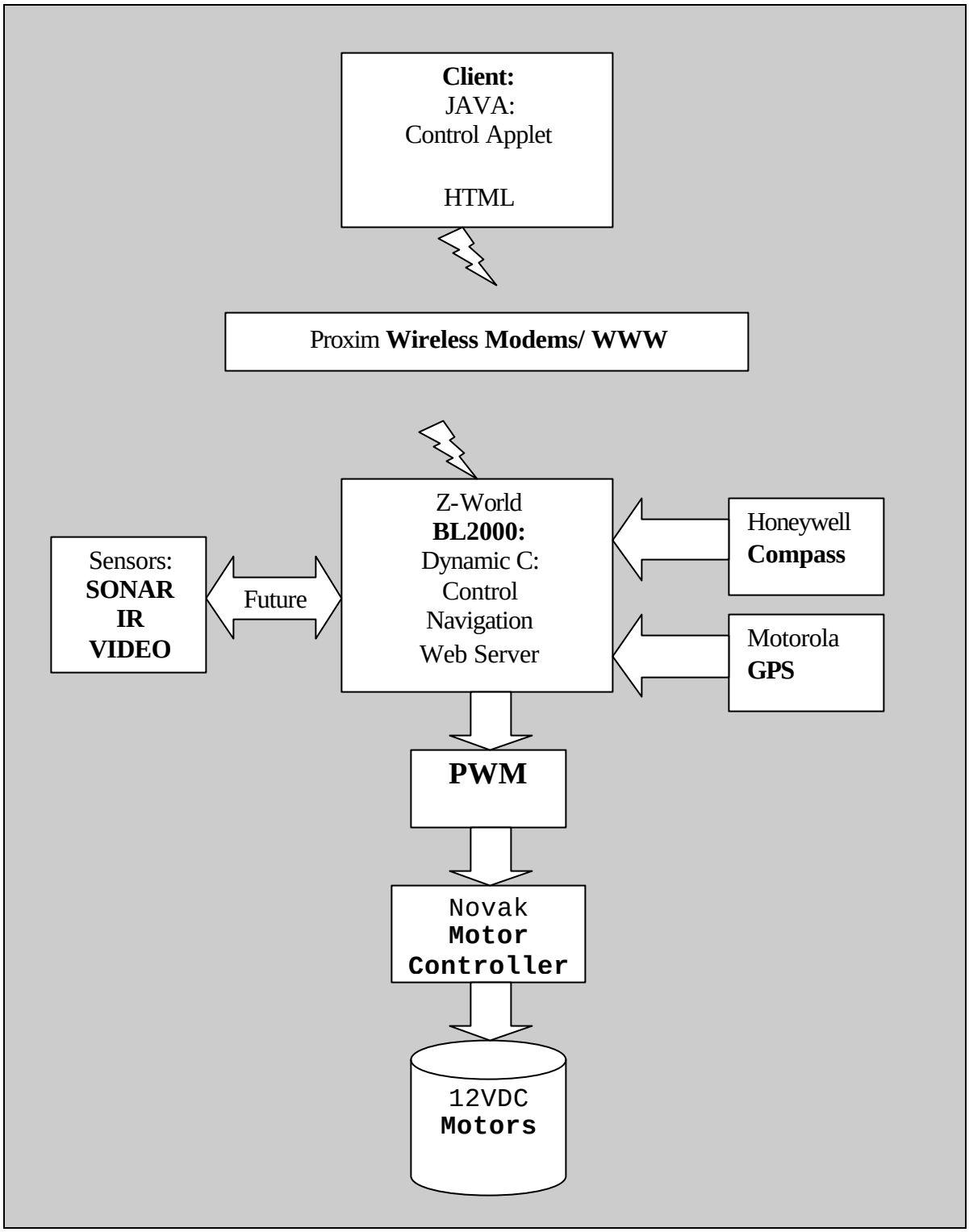


Figure 4. Block Diagram

d. Sensors:

The current sensor project for the SMART robot is the Seismo-Acoustic sonar under development by a group in the CSS&T curriculum under the supervision of Dr Thomas Muir. This sonar generates and exploits Rayleigh wave phenomena, which propagates outward from the signal source and remains near the surface. Sophisticated signal analysis and filtering techniques make this sonar a good fit for the detection of buried mines in the surf-zone.

The towed platform (Figure 5.) is the transmission portion of the Seismo-Acoustic sonar. A microprocessor creates the desired output waveform that is amplified by the acoustic amplifier on the forward end of the towed unit. The amplified signal is then fed into a series of acoustic "thumpers" which are located in the aluminum cylinder at the rear of the towed unit. The cylinder was chosen as a vehicle for the thumpers because it allows the maximum surface contact area (and therefore maximum energy transfer) with the ground while allowing a platform that the robot would be able to tow. Seismometers will either be placed onboard the thumper unit or deployed by a separate platform.

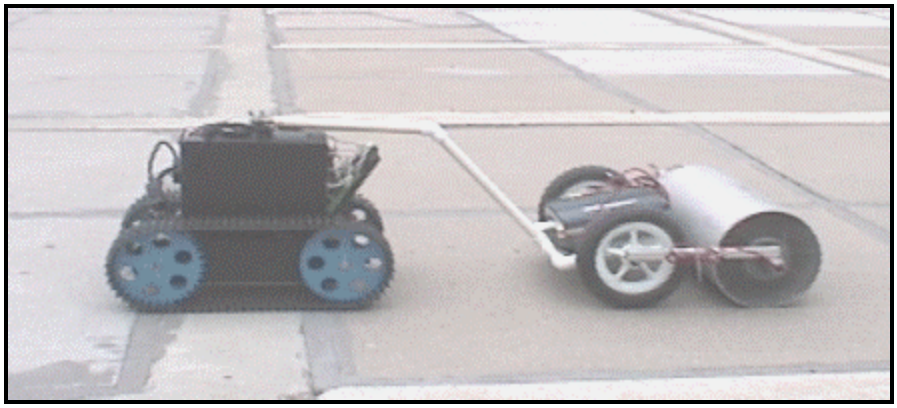


Figure 5. Seismic Sonar

THIS PAGE INTENTIONALLY LEFT BLANK

II. OPERATIONAL SPECIFICATIONS

A. COMMUNICATIONS RANGE

An operational test of the communications range of the SMART robot showed satisfactory results when compared to the advertised range of the Ethernet modems employed. In an operational test at the Navy beach in Monterey CA generated a line of sight communications range of over 200m. When the robot was driven over a hill beyond the line of sight communications was lost at a range of 150m. Further tests at the NPS softball field showed a line of sight communications range of over 100m and additional testing will be completed to determine the actual maximum range.

B. ENDURANCE

Run time operational tests were performed to determine the endurance of the SMART platform with its current battery configuration. The communications system and the microprocessor are powered by a single 12-volt 3.0 AH battery and the endurance in operational tests was three to four hours. A 6-volt 1.3AH battery powers the compass and GPS cards and the endurance for the power supply was tested to be over 4 hours. The motors for the SMART robot are powered by a 12-volt battery that will last 2-4 hours depending on the terrain and operational tempo. Field operational tests were limited by the battery endurance of the laptop computer, which limited field operations to approximately 2-3 hours.

C. HEADING ACCURACY

As a check of the heading accuracy of the compass guidance functionality of the SMART platform two tests were completed. The first test was a swing-check of the compass conducted with the robot under static conditions in a controlled environment. A reference line of 000° magnetic was placed on the classroom floor and compass rose points were drawn in reference to this mark. The direction of magnetic north was determined with a generic magnetic compass. The robot was then aligned with the points and the readout from the onboard magnetic compass was read with the motors off and operating in all combinations. Results are specified in Table 1, and reflect an accuracy of ±7°.

Actual Heading	Stop	Forward	Reverse	Left	Reverse
000	001	001	001	001	001
045	050	050	050	050	050
090	098	098	098	098	098
135	142	142	142	142	142
180	180	179	179	179	179
225	219	219	219	219	219
270	263	263	263	263	263
315	310	310	310	310	310

Table 1. Compass Swing Check.

Additional tests of heading accuracy were conducted in the field at the Navy beach in Monterey CA and at the softball field on NPS. Accounting for initial corrections made by the platform as it found its desired headings, all tests showed an operational heading accuracy of ±10°.

D. GPS ACCURACY

Multiple test runs were conducted at the softball field at NPS with encouraging results. The GPS card on the SMART platform was operating without differential antennae installed and in that configuration; the rated accuracy of the card is only 100m. During a series of tests heading towards a common waypoint from various distances and directions the robot was able to accurately navigate to within 15m of the desired location. Given that the robot was programmed to stop navigating when it was within 5m of the waypoint and the accuracy of the card was 100m a repeatable accuracy of 15m was acceptable. Future iterations of the SMART platform will employ a differential antennae and increased accuracy is expected.

III. HARDWARE

A. PLATFORM

1. Foster-Miller Lemming

The Foster-Miller Corporation has been building the Lemming robot platform for many years and the basic design plan is to "develop light, sturdy, compact robots that can be adapted for ordnance removal, reconnaissance, communication, relays, sensing and security"[Ref 5]. The NPS SMART robot is an older generation Lemming received from Coastal Systems Station (CSS) Naval research facility in Panama City, Florida. Upon receipt at NPS the existing control hardware was removed and the focus of this thesis is the integration of the new COTS component based control system.

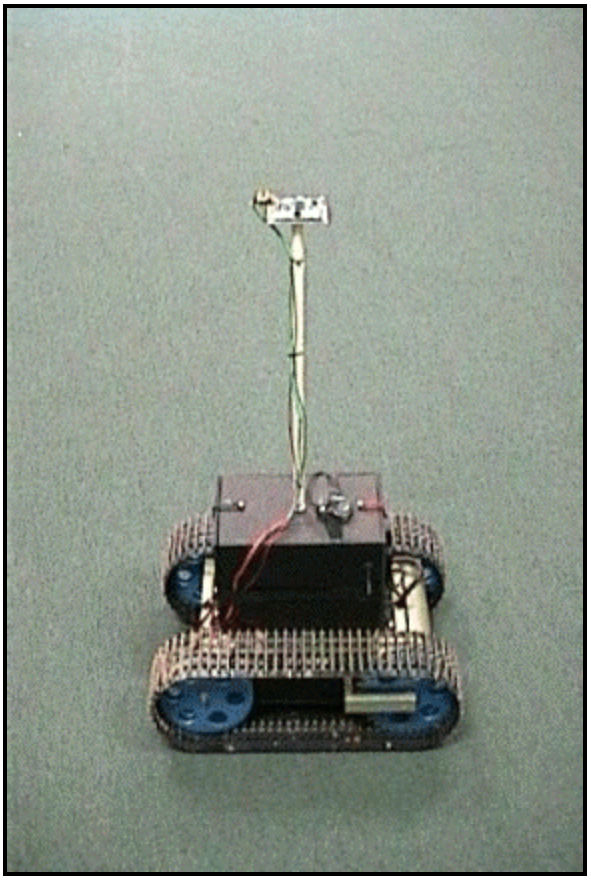


Figure 6. Bender

2. Electronics Enclosures

a. The Platform

The platform enclosure was built to allow for ease of access during the initial program development and hardware integration phase. The platform is constructed of clear plastic that is approximately 0.325in thick. The platform places an 8.75in wide, 13.5in long platform 3.5in above the base allowing for two large placement areas that could be visible during all aspects of testing.

b. The Box

The box electronics enclosure was constructed for use as splash-proof protective enclosure, which would also act as a base for the tow hitch used with the Seismo-acoustic sensor trailer. The box is 6.5in tall, 6in wide and 10.25in long, providing sufficient space for all electronics. Electronics are mounted to the floor or walls of the box by mounting screws (if provided) or Velcro. A mast was added the top of the tow hitch as a mounting location for the magnetic compass, the mast places the compass 27in above the main base of the Lemming and is high enough to eliminate magnetic interference from the motors and the GPS antenna which has a built in magnet used for mounting on automobile or marine applications.

3. Measured Figures

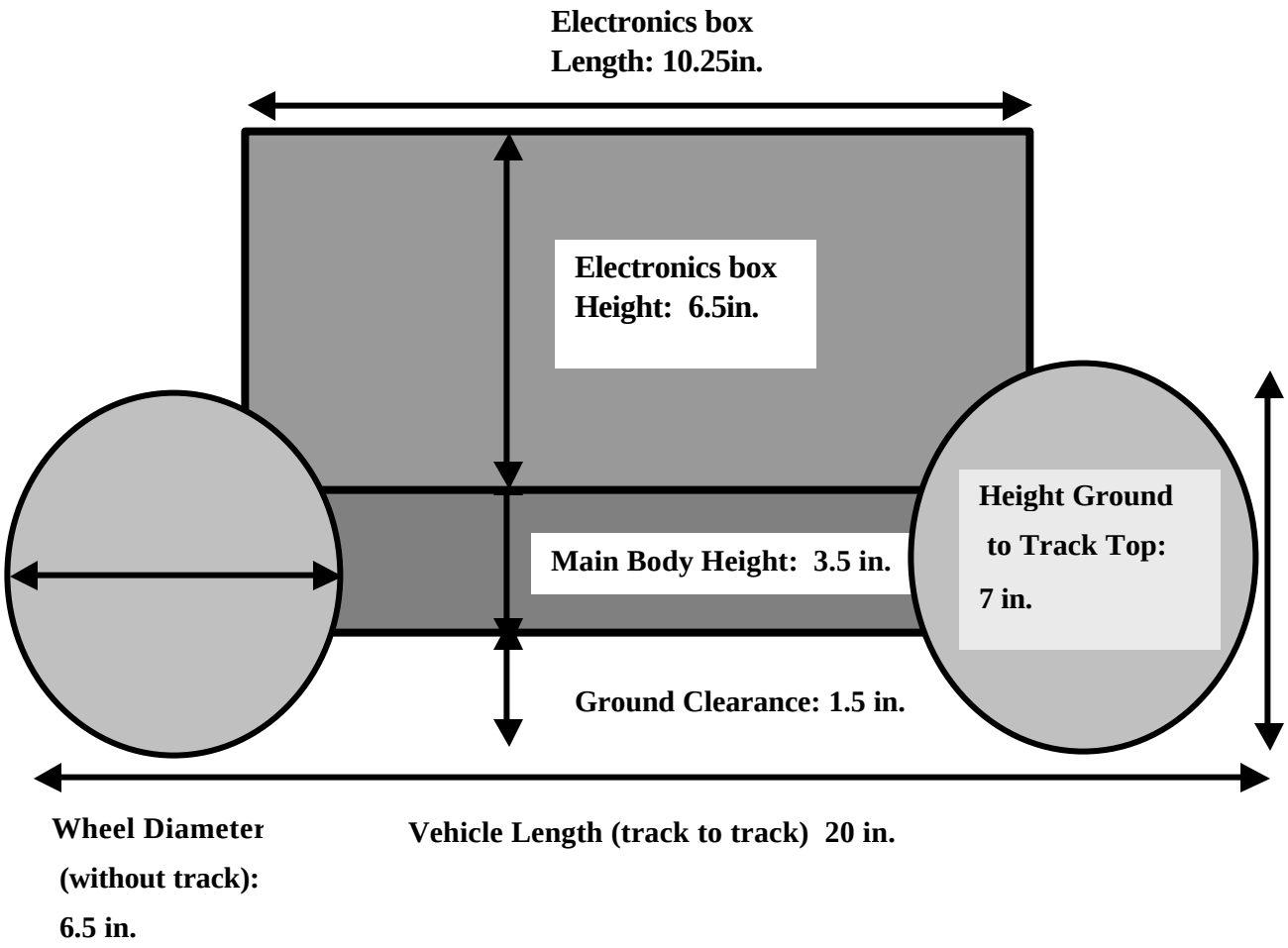


Figure 7. SMART Platform side view (± 0.25 in.).

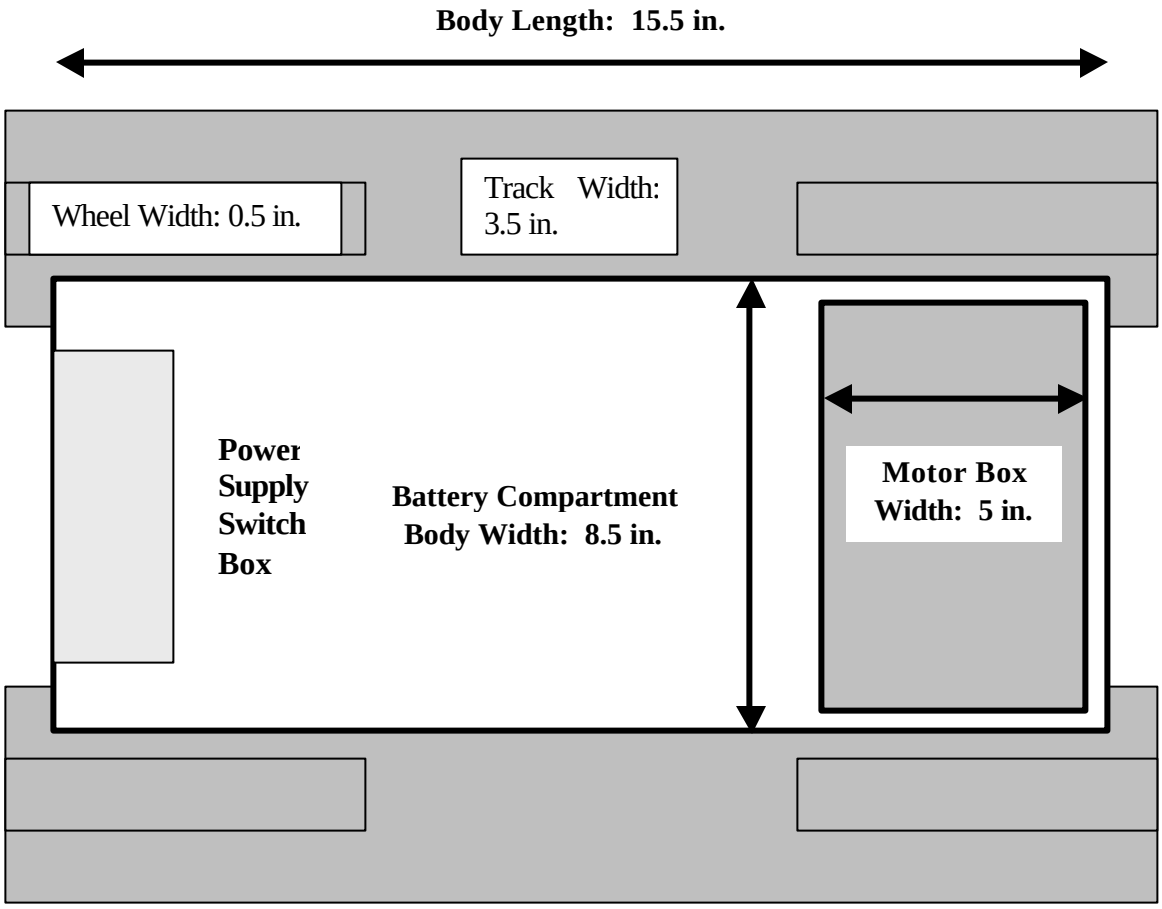


Figure 8. SMART Platform Main Body Top View (± 0.25 in.).

B. MICROPROCESSOR

1. Z-WORLD BL2000

The BL2000 microprocessor is a many-functioned extremely capable microprocessor clearly designed with both the student and serious computer user in mind. Superior documentation and excellent customer support made the integration of Internet communications, serving a web page, multiple digital inputs; analog outputs and a cooperative multitasking control program a manageable task. The BL2000 can be powered with between nine and forty volts and in this application shares a battery with the communications modem. The development kit for the microprocessor contains a sturdy plastic protective case, which is used as a mounting device.

BL200 Features
Operating Language: Dynamic C
22.1 MHz
128K static RAM and 256K flash memory
28 Digital I/O pins
9 12 bit A/D converters
2 12 bit D/A converters
10Base-T Ethernet Port
4 Serial Ports
Real time clock
E-mail and Web server capabilities

Table 2. BL2000 Features

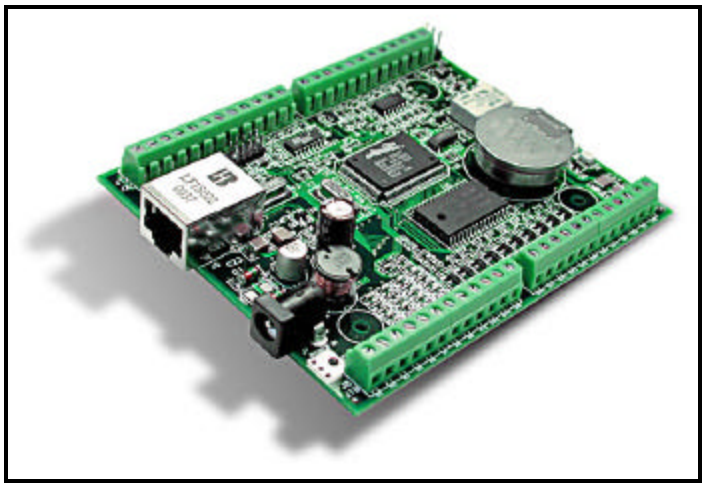


Figure 9. BL2000 (Z-World) [From Ref. 8]

C. G.P.S

1. Motorola M12

The Motorola M12 GPS card is designed for use in either automotive or marine applications. The card utilizes a 6VDC input and provides a standard serial port output. A powered antenna is provided with a built-in magnetic base for ease of mounting on automotive or marine applications. The accuracy of the GPS card is 100m without a differential antenna but with a differential antenna an accuracy of 1-5m can be expected. The output from the card is fully programmable and can easily be changed to meet the users needs, the output string will always contain unneeded data about the status of the satellites tracked and the signal strength, this unneeded data is filtered out in the software and this process is covered in more detail later.

The string is formatted as follows:

@@Eq,mm,dd,yy,ss,**dd,mm.mmmm,n,dd,mm.mmmm,w**,shhhhh.h,sss.s,hhh.h, m,t, dd.d,nn,rrrr,aa,CCC<CR><LF>.

- @@Eq >> Header
- mm,dd,yy,ss >> Time stamp
- dd,mm.mmmm >> Latitude

- n >> North or South
- dd,mm.mmmm >> Longitude
- w >> East or West
- shhhhh.h >> Height
- sss.s >> Velocity
- hhh.h >> Heading
- m,t,dd.d,rrrr,aa >> Receiver, Signal and Satellite Status
- CCC >> Checksum

The needed data from the GPS card is just the location portion of the data string that is highlighted in bold above. Additional data from the GPS card includes in order, a leading time stamp, height, velocity, heading, receiver status and a final checksum. The output rate from the card can be set to once per second as a maximum and down to once every 255 seconds. In our application it is set to update once per second.

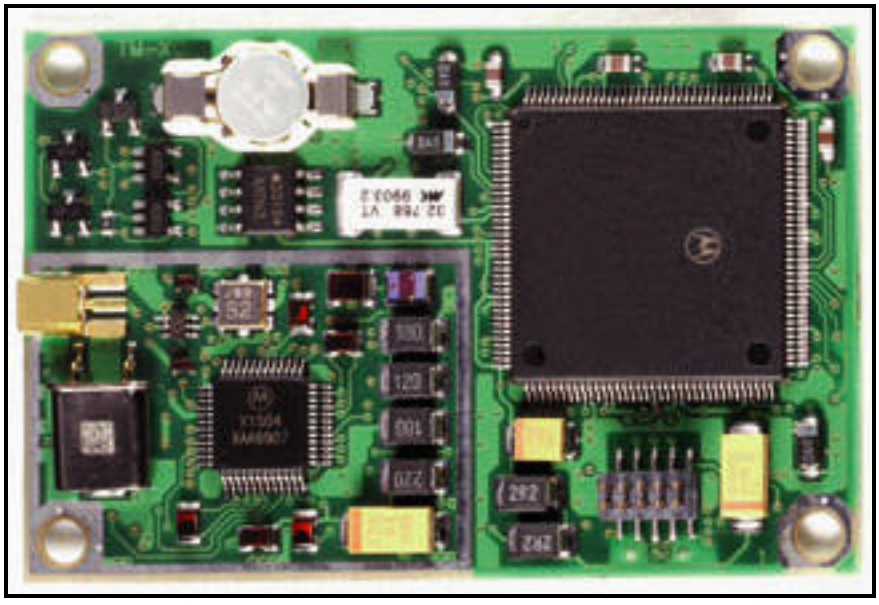


Figure 10. Motorola M12 [From Ref. 11]

D. DIGITAL COMPASS

1. Honeywell HMR3000

The Honeywell HMR3000 is a small single card electronic compass unit capable of providing both heading and pitch information. The sensor on the compass is a magneto resistive sensor, which utilizes the anisotropic magneto resistance of the ferrous material of the sensor. The sensor material is placed in a basic wheat stone bridge configuration and the heading is calculated. The card utilizes a 6VDC input and provides a standard serial output to the user. The output is programmable to provide multiple signals including pitch, roll, heading, or any combination of these required in various formats. The output signal can be updated from six to three hundred times per minute and in our application is set to one update per second.

Upon initial receipt the digital compass is programmed to provide a standardized output for marine applications. The standard is from the National Marine Electronics Association (NMEA) and is known as NMEA 0183. This standard programmed output is a string, containing heading, pitch and roll information.

The string is formatted as follows:

$PTNPHTR,HHH.H,N,P.P,N,R.R,N*2C.

- $PTNPHR >> Header
- HHH.H >> Heading
- P.P >> Pitch
- R.R >> Roll
- N >> Mode, N=normal

The string is started with a standard header and is followed by the heading in degrees and tenths of degrees (HHH.H), the next character (N) indicates normal operation, followed by the pitch in degrees (P.P) finally followed by roll in degrees (R.R). Utilizing the provided interface program a message can be sent to the compass modifying its output. For our application, that string was parsed for the heading (HHH.H) ASCII information only.

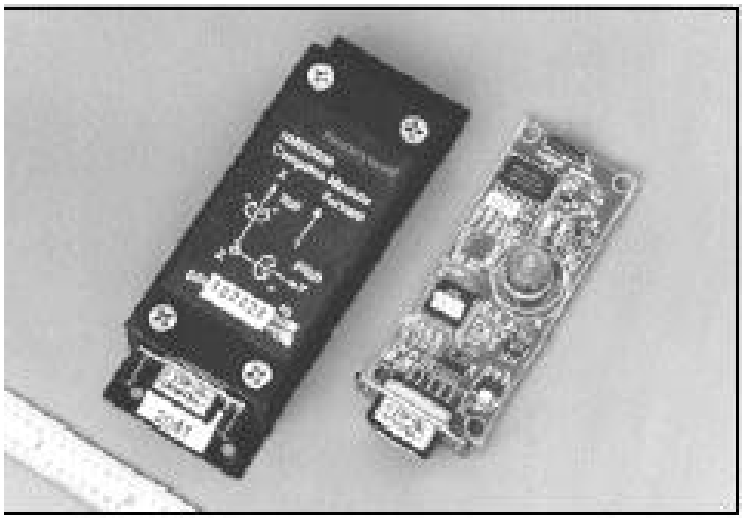


Figure 11. Honeywell HMR3000 [From Ref. 10]

E. COMMUNICATIONS

1. Proxim RangeLAN2 7920

The Proxim RangeLAN2 7920 is a COTS Ethernet LAN module. The unit is powered by 12 volts and has a current draw of less than one amp when transmitting. The 7920 is self-configuring and will auto-negotiate network parameters upon power up .The 7920 is capable of being either a master or a slave station to either another 7920 or a designated master station. The self-determining master or slave option allows the system to either work within the range of a LAN or set up its own LAN if being operated in the field.

Communications between RangeLAN2 products is conducted in the 2.4GHz range utilizes frequency hopping and spread spectrum technology. The speed of data transfer is 1.6Mbps. The 7920 have a published range of 1000 feet and experiments with the system showed a range of 150m in a beach environment beyond line of sight. Line of sight maximum range in other environments has not yet been determined.

Figure 12. Proxim RangeLAN2 7920

F. MOTOR CONTROL

1. Pulse Width Modulation

A pulse width modulation circuit is the standard for control of motors in the robotics field.

a. Circuit

The PWM control circuit designed for the SMART platform is constructed through the use of basic 555 timer chips. A basic PWM circuit can be found on the data sheet for National Semiconductors LM555 chip. The specific circuit designed for the SMART robot (Figure 13) uses a 555 timer to establish a base frequency. The base frequency and a control voltage provided from the analog outputs from the BL2000 are then fed into a 556 timer chip which performs the pulse width modulation function for the left and right motors.

b. Calculations

For the circuit in Figure 13 where R_A=10 kΩ, R_B=1 kΩ and C=1 uF. The frequency of the 555-timer output is determined by the charge and discharge time of the capacitor attached to the trigger and threshold inputs of the chip. The period (T) is determined by:

$$(1) \quad T = 1/f = 0.693(R_A + 2R_B)C$$

The frequency (f) is given by:

$$(2) \quad T = 1/f$$

For the pulse width modulation part of the circuit R=100 kΩ, C=0.047 uF V_{cc}=6V and Vin is determined by the output from the BL2000 digital to analog converters. The positive pulse width is given by:

$$(3) \quad T = -\ln(1 - V_{in}/V_{cc})RC$$

	Calculated	Left PWM	Right PWM
Frequency	120.25 Hz	118Hz	118Hz
Stop Vin	1.5V	1.56V	1.56V
Stop Pulse Width	1.421msec	1.41msec	1.41msec
Reverse Vin	1.0V	1.04V	1.04V
Rev Pulse Width	0.898msec	0.920msec	0.920msec
Forward Vin	2.0V	2.05V	2.05V
Fwd Pulse Width	2.009msec	2.05msec	2.05msec

Table 3. PWM Specifications

c. Schematic

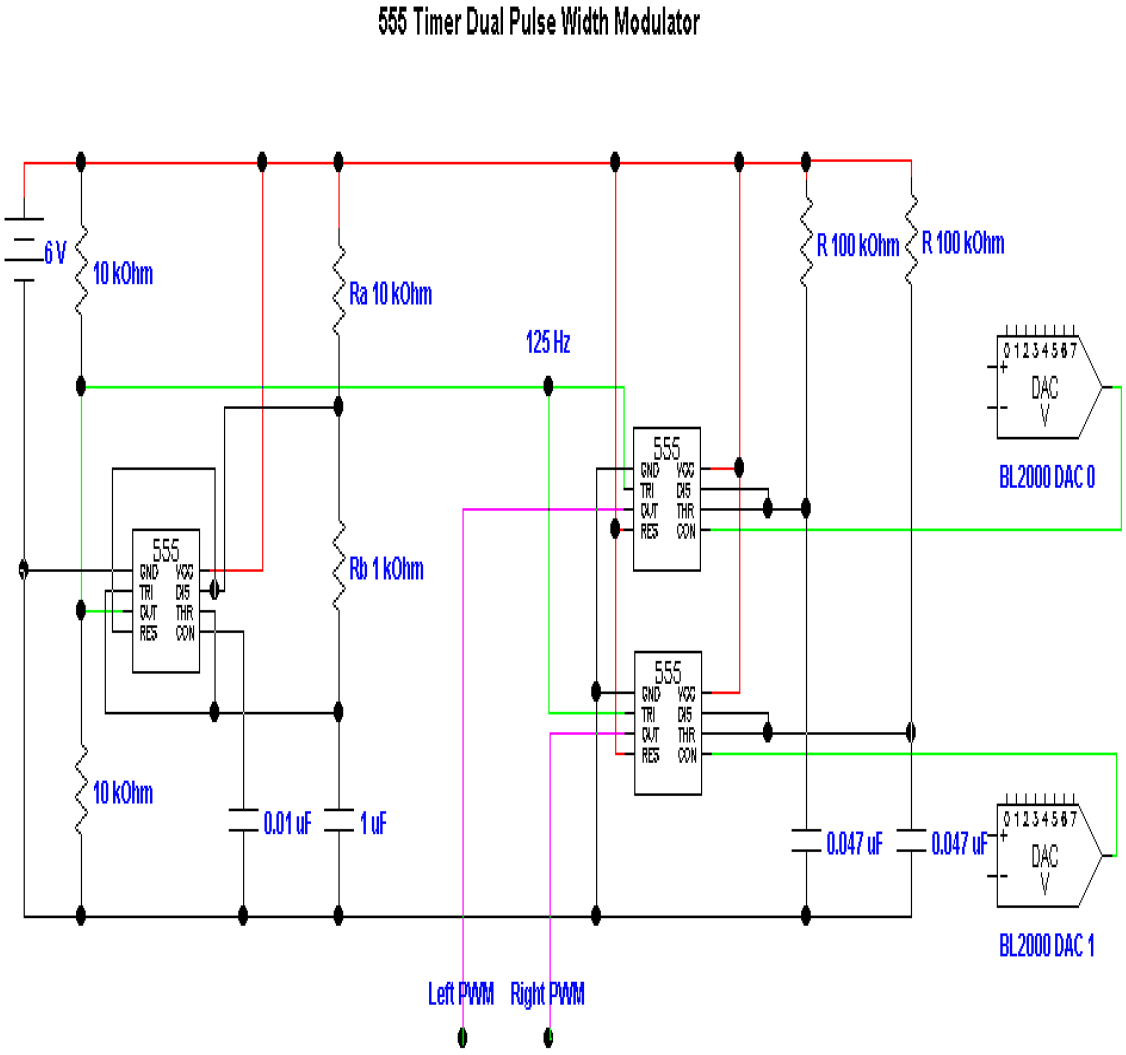


Figure 13. Pulse Width Modulator Schematic

2. Motor Controllers

a. Novak Super Rooster

The Novak Super Rooster is a COTS Electronic Speed Control (ESC) used for competition radio control car racing. The ESC is capable of handling a twelve-volt power supply and can safely route 320 amps in the forward and 160 amps in the reverse direction while providing an impedance of only a few milliohms. With one touch

programming the ESC is quickly calibrated to the input signals provided and the calibration data is placed in permanent flash memory until the user reprograms the ESC. The ESC interprets the PWM supplied to it from the PWM circuit and converts the signal to the appropriate desired power supply to the motors.

Figure 14. Novak Super Rooster [From Ref. 13]

THIS PAGE INTENTIONALLY LEFT BLANK

IV. SOFTWARE

A. OPERATIONAL THEORY

The control mechanism for the SMART robotic platform depends upon a smooth interaction between two programs and the various threads running within these programs. The control interface for the SMART robot is a JAVA created GUI with sliders, pushbuttons and multiple interactive text fields. The program that actually controls the robot is a cooperative multitasking program in Dynamic C that interacts with both the JAVA application and the robot's hardware package.

Communication between the two programs is handled with standard TCP/IP sockets, (see figure 17). When a socket is established between two programs the programs are able to communicate by sending packets of information over the Internet. The JAVA and Dynamic C programs interact to establish five different sockets to perform various tasks.

- The Motorsock socket is established to allow for direct control of the motors in a manual mode, this socket is established upon power up and is persistent until the robot is secured.
- The Headsock and GPSsock sockets are established to facilitate the sending of data from the compass and GPS cards, these sockets are established when needed and are not persistent.
- The DheadLocsock is established to allow the user to send a requested heading for movement or a GPS waypoint to drive to; this socket is also not persistent.
- The Stopnavsocket is established to allow the user to manually stop the robot's navigation process when desired, the socket is non persistent and is only established momentarily to reset values which will stop the navigation costate.

The use of both persistent and non-persistent sockets was based on programmer's preference and a desire for direct control of communication links that were being opened.

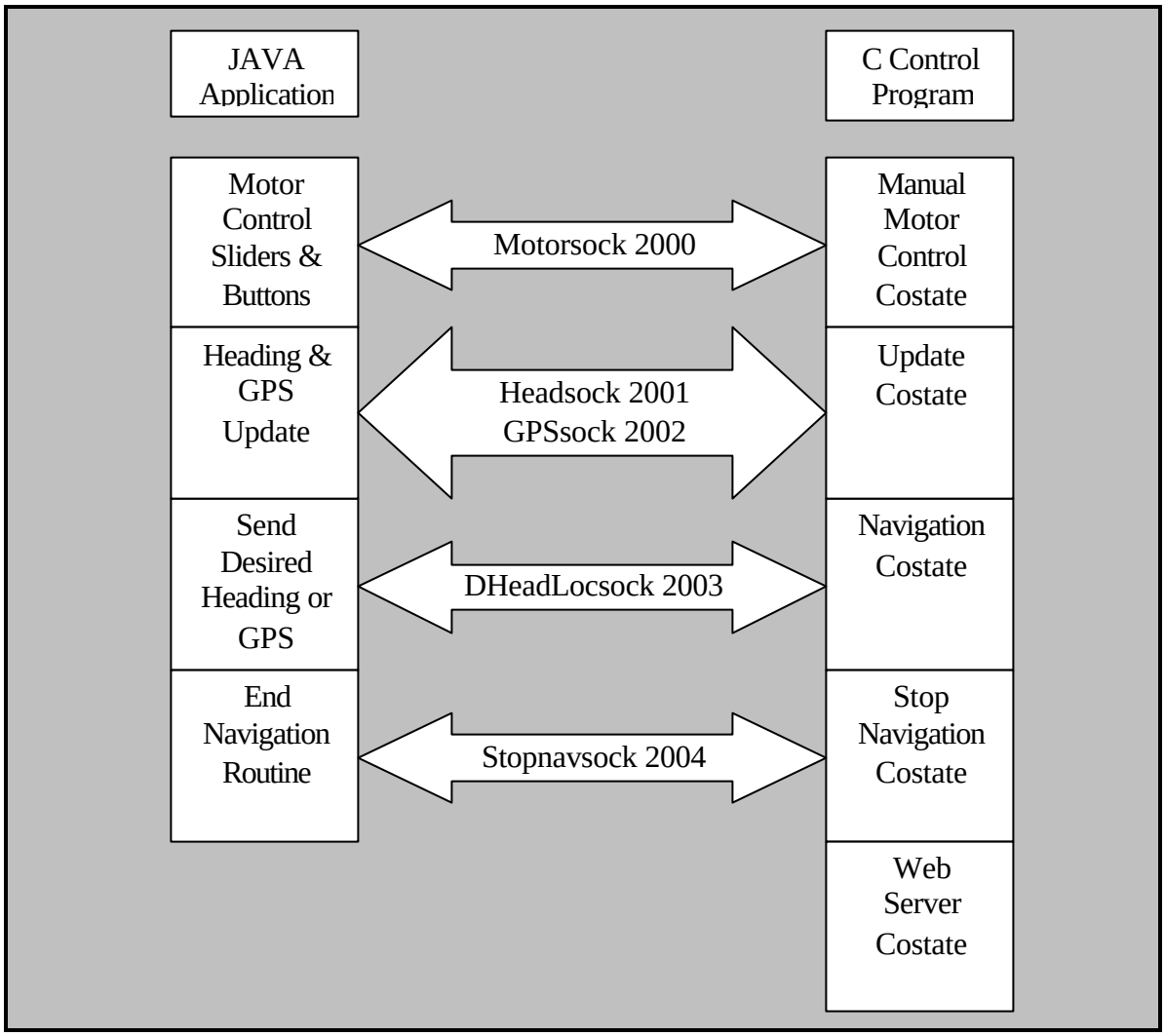


Figure 15. Program Interaction and SOCKETS

B. CONTROL PROGRAM

The control program (Alltask.c) for the SMART robot is written in Dynamic C and controls all tasks of the robot in a cooperative multitasking environment (figure 18). The program has five costates that perform the basic functions of the robot: manual motor control, GPS and compass update, navigation, stop navigation and web page server. These costates interact with the JAVA control application and with other functions within the Alltask.c program to govern operation of the robot.

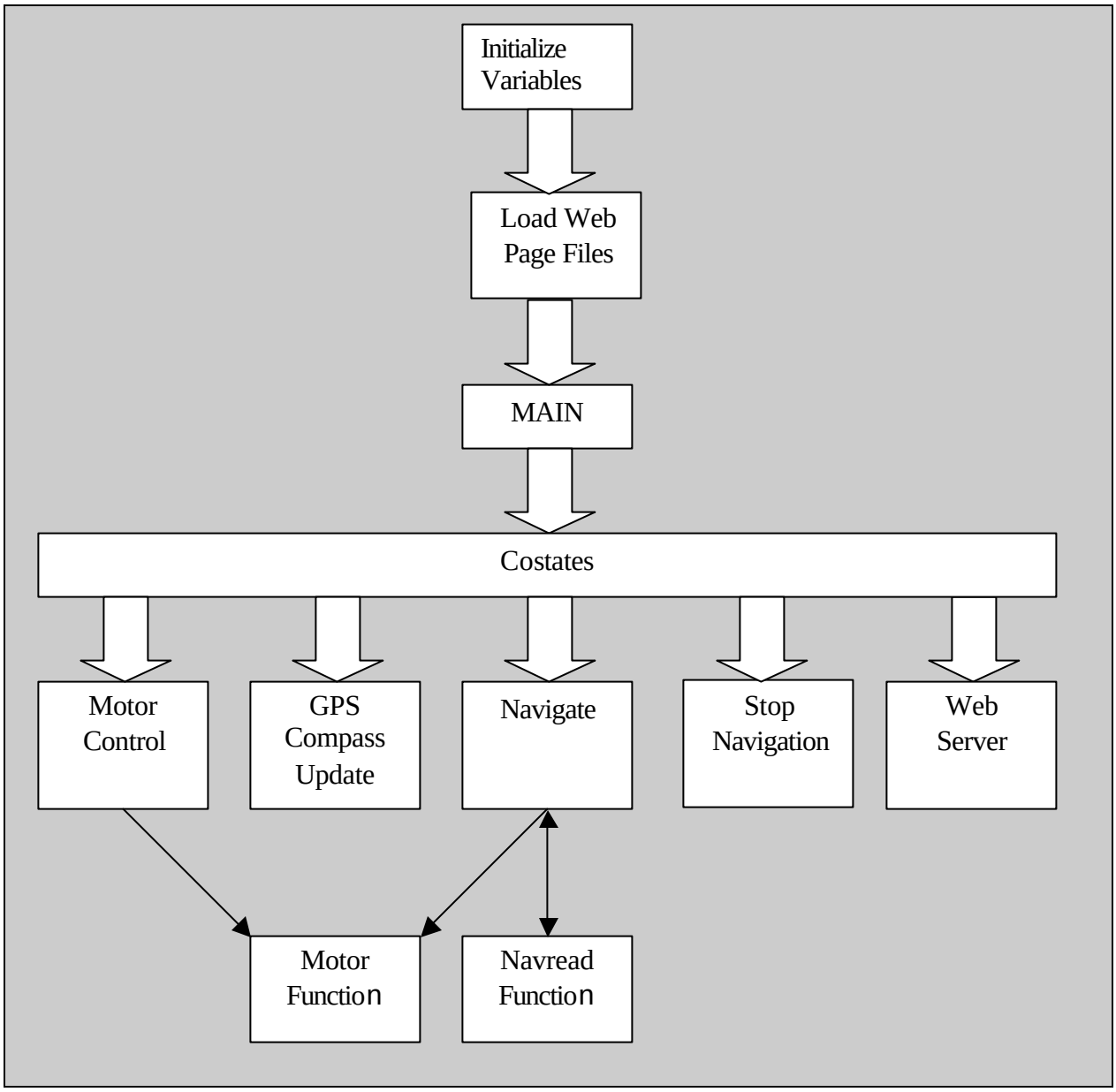


Figure 16. BL2000 Control Program

1. Motor Control

The motor control function of the Alltask.c program is used during manual control operations of the robot. A socket is established with the JAVA control application and a packet is received with a data stream that the motor control costate interprets. The JAVA application sends a comma delimited data stream, the first token is a placeholder, the second is the motor designator, and the third is the speed and direction

control variable. Using a string tokenizer function, the program converts the value of the speed and direction variable to motor speed variable, the program then calls the motor control function which places the appropriate output value on the BL2000 analog outputs which is then converted to a pulse width by the PWM circuit for feeding to the motor controllers.

2. GPS/Compass Update

The GPS/Compass co-statement queries the GPS and compass cards and then forwards the received data to the JAVA application for display. The GPS and compass cards provide a standard ASCII output that is routed to the "serial a" and "serial b" ports on the BL2000. The costate waits until a socket is established before performing any reading functions, which allows the function to be "sleeping" until called upon by the user via the JAVA application. The function firsts flushes the buffers for the serial ports and waits for the latest input from both the GPS and compass. The input strings are then copied to a buffer variable and using the puts command sends that string to the JAVA application and closes it's sockets and begins to wait for another input.

The GPS data string is long and, as with the motor control string, needs to be tokenized to parse out needed data from the string.

The string is formatted as follows:

@@Eq,mm,dd,yy,ss,**dd,mm.mmmm,n,dd,mm.mmmm,w**,shhhhh.h,sss.s,hhh.h,
m,t, dd.d,nn,rrrr,aa,CCC<CR><LF>.

- @@Eq >> Header
- mm,dd,yy,ss >> Time stamp
- dd,mm.mmmm >> Latitude
- n >> North or South
- dd,mm.mmmm >> Longitude
- w >> East or West
- shhhhh.h >> Height

- sss.s >> Velocity
- hhh.h >> Heading
- m,t,dd.d,rrrr,aa >> Receiver, Signal and Satellite Status
- CCC >> Checksum

This data string is comma delineated and can therefore be separated into its separate components using the string tokenizer function in Dynamic C. The needed data from the GPS card is just the heading portion of the data string that is highlighted in bold above. Additional data from the GPS card includes in order, a leading time stamp, height, velocity, heading, receiver status and a final checksum. The data string is 96 bytes long and this length is used in the software as a method to determine if a valid string has been received.

3. Navigation

The navigation costate like the GPS/Compass costate waits for an input from the user before coming out of the sleep state. Upon receipt of a Desired Heading or Location string (DHeadLoc) from the JAVA application the navigation costate tokenizes the DHeadLoc string and determines its size and the value of its components. The navigation costate then calls the navread function which depending on the size of the DHeadLoc string will either conduct a GPS location calculation or a heading calculation determining a Course To Steer (CTS) and a difference between the CTS and the current heading. The difference value is then returned to the navigation costate which depending upon the values of the difference will turn towards the desired heading. When the desired location is reached, the robot will automatically stop and signal that it has completed the navigation costate.

a. Feedback Error Control

The principle of control for a robotic system is straightforward in that we will employ mechanisms to minimize some error function, figure 17. If we assume that our goal is to turn the robot to a desired angle q_d from a current angle q, then the error E would be given by:

$$(4)\quad E = q_d - q$$

Now if we apply a drive signal to the plant that is proportional to E, then we can say that our feedback mechanism invokes proportional error control in our system. The equation of motion for such motion is given by:

$$(5)\ T_m = J_{eq}\ddot{q} + F_{eq}\dot{q} \quad \text{where } \dot{q} = \frac{dq}{dt} \quad \text{and } \ddot{q} = \frac{d^2q}{dt^2}$$

T_m is the motor torque, $J_{eq} = J_l + J_m$, J_l is the inertia of the load reflected through the gears and J_m is the inertia of the motors, F_{eq} is friction and q is angle. We can also say that:

$$(6)\ T_m = K_m I$$

where I is the motor current and K_m is a constant. Equating (4) and (5) gives:

$$(7)\ K_m I = J\ddot{q} + F\dot{q}$$

If we invoke proportional control, we can assume that we will apply a current that is proportional to the error: $I \propto E$ or we can say using equation (3) that $I = k(q_d - q)$, where k is a constant. Therefore:

$$(8)\ k_e(q_d - q) = J\ddot{q} + F\dot{q}$$

Here we have absorbed the constants $k_m k$ into k_e and rewritten J for J_{eq} and F for F_{eq} to simplify. Now if we assume that $q_d = 0$, then our equation of motion is:

$$(9)\ J\ddot{q} + F\dot{q} + k_e q = 0$$

If we assume a solution $q = e^{qt}$ and let $k_e = k$ then (8) reduces to:

$$(10)\ Jq^2 + Fq + k = 0$$

Solving for q gives:

$$(11)\ q = \frac{-F \pm \sqrt{F^2 - 4Jk}}{2k}$$

We see that if:

- $\sqrt{F^2 - 4Jk}$ > 0 (the system is over-damped)
- $\sqrt{F^2 - 4Jk}$ = 0 (the system is critically damped)
- $\sqrt{F^2 - 4Jk}$ < 0 (the system is under-damped)

Our goal was to provide critically damped control to the turn of the robot.

The navread function in the navigation costatement determines the error signal applied in the proportional control algorithm. The error is multiplied by a constant K and converted to a signal to the plant, which causes motors to spin and the platform to turn. Assuming that J and F are constant we manipulate k to get our system response critically damped. We found that this was more difficult than imagined because, as it turns out, k is a function of time $k(t)$ for our application.

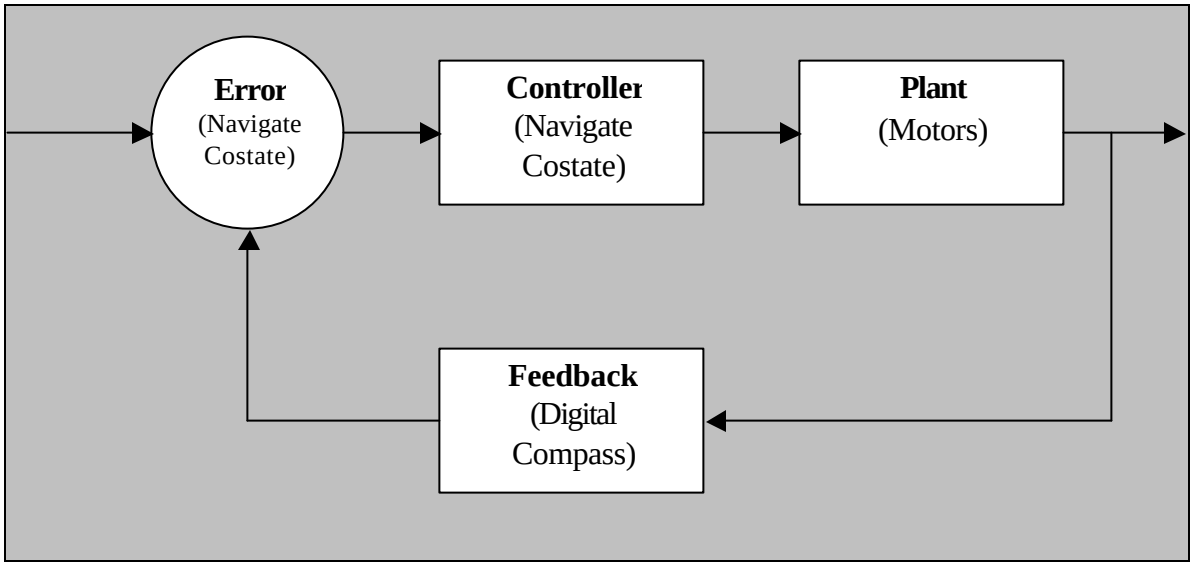


Figure 17. Proportional Controller

A combination of proportional, derivative and integrative (PID) would be the ideal method of feedback to use for a navigational system such as this. Hardware issues and the limitations of the platform made PID control difficult. Due to the power limitations of the Lemming chassis, a full power turn is required to ensure that the robot will turn on any terrain, a feedback system, which would slow the turn as the robot approached the desired heading, was therefore impractical. The time required to

complete a sample of the actual heading was also a factor since the compass was only providing output data at a rate of one string per second and testing showed that there was an indeterminate yet not insignificant compass delay while the output signal caught up with the actual heading.

b. Where am I?

In the navread function the robots desired and actual locations are compared and a desired heading is determined. The GPS inputs are converted to a grid location by a simple multiplication of the degrees and minutes by a constant to determine the distance north of the equator and by a constant and a scaling factor to determine the distance west of the prime meridian. Once the desired and actual locations are known trigonometry and Pythagoreans theorem are employed to determine the desired heading to steer and the distance to the desired location. The data is then fed back to the navigation costate which occasionally re-queries the navread function for an update on new headings and distance.

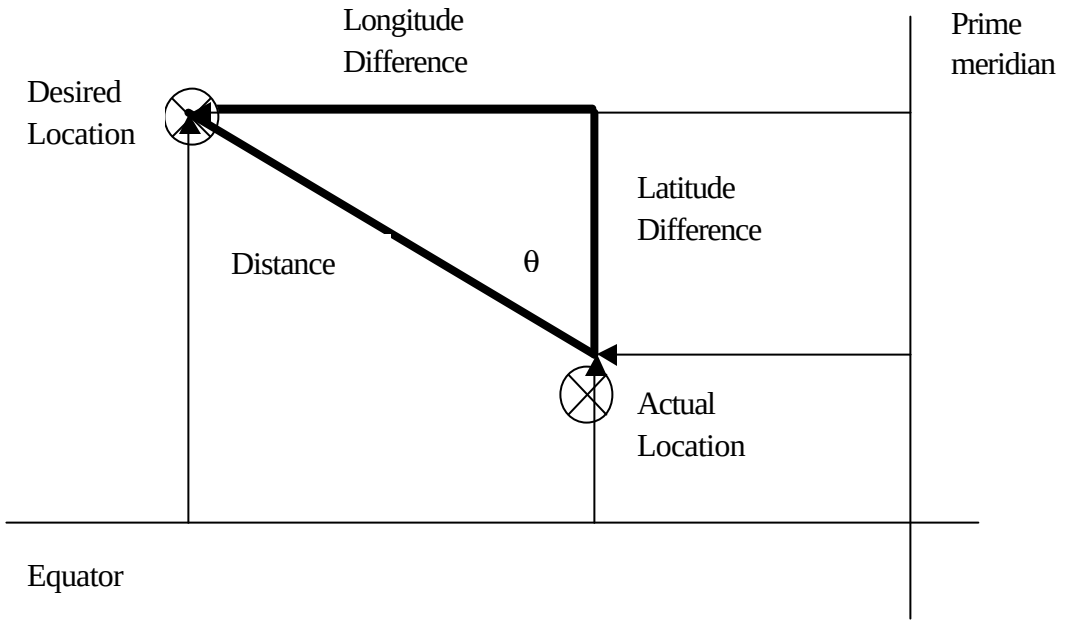


Figure 18. Distance and Course to Steer Calculations

The actual calculations to determine the location of the robot and the robot's desired location are based on converting a GPS location to an equivalent grid point. The latitude in meters is determined by:

$$(11) \quad LATm = ((LATDEG*60) + LATMIN)*1852.$$

The Longitude in meters is:

$$(12) \quad LONGm = -((LONGDEG*60) + LONGMIN)*1852*\cos(.628).$$

The cosine is used to correct for the difference in the sizes of a degree of latitude and longitude at the location of NPS. To make the program fully functional worldwide, the cosine correction must utilize the actual location of the robot and the software must be further modified to recognize east longitudes and south latitudes.

C. CONTROL INTERFACE

A goal of the SMART project was to construct a web based GUI that would both easily interact with the robot control program, form a user friendly interface, and be relatively easy to construct using available technologies and accessible instructor expertise as needed. Various programming methods were investigated including CGI scripts, PERL scripts and JAVA, based on the advice of NPS staff members from the computer science department it was decided to use JAVA.

1. JAVA Application

The JAVA application created is a multi-layered window utilizing sliders, pushbuttons and interactive text-boxes as a method of controlling and interacting with the SMART platform. JAVA automatically handles the multitasking issue, the user sets up "action event listeners" which will allow a thread to sleep until the button is pushed or the slider is moved. Similar to the process of programming in Dynamic C the first actions in the initialization of a JAVA program is to call needed library functions and then to initialize variables. Once this initial setup is complete, the JAVA program can set up the control interface.

The control interface is a multi-layered window that has 2 sub-panels; the first panel is the manual control panel (figure 20), which has motor control sliders for control

of the left, right, and both motors. Manual push buttons allow for quicker access to full power left and right turns as well as full forward, full reverse and all motors stop. The navigation control panel (figure 21) contains non-editable text boxes for display of actual location and heading data, editable text boxes for input of commands, push buttons for entry of commands to either update the location/heading boxes or to commence or stop the navigation subroutine. Additional pushbuttons not used at this time are available for use in ordering the robot to conduct pre-programmed patterns that can be utilized for conducting mine search or surveillance patterns.

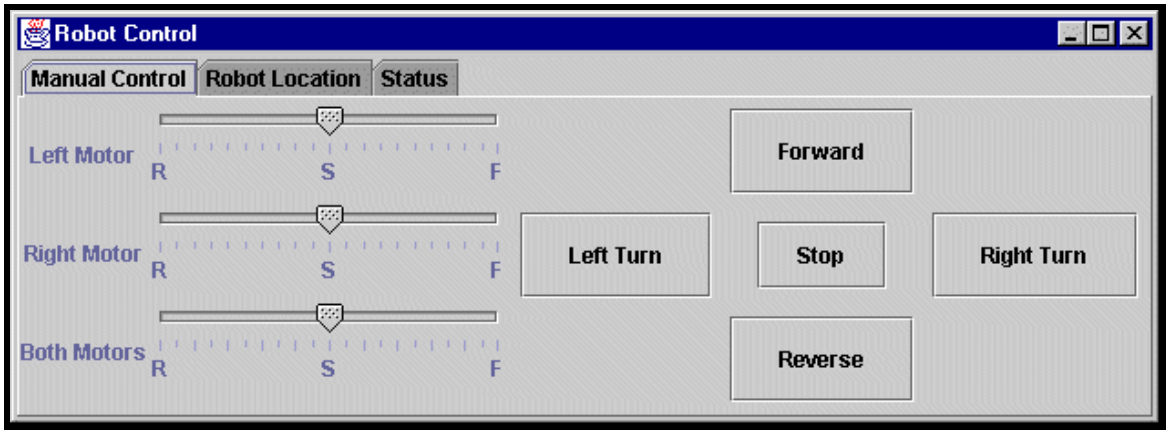


Figure 19. Manual Control Panel

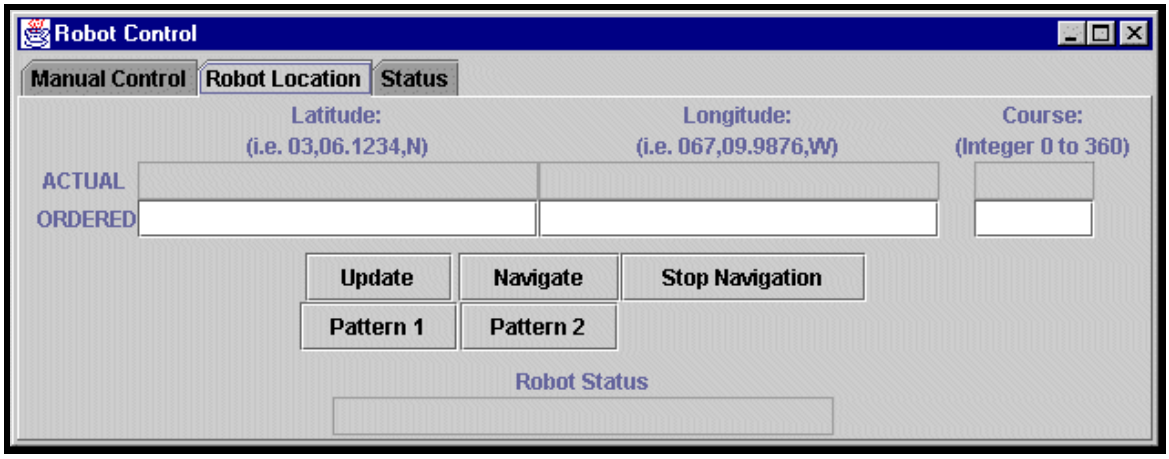


Figure 20. Navigation Control Panel

D. WEB INTERFACE

The BL2000 microprocessor is capable of serving a web page and this ability has the potential to greatly increase the operational ability of the SMART platform. One major feature of JAVA programs is the ability for the JAVA application to be converted to a JAVA applet that can be downloaded by a web server. A downloadable JAVA control applet would enable a remote user from any location in the world (with the proper access codes) the ability to control the SMART platform whether the operator had the control application loaded on a computer or not. When the additional capability of streaming video and increased communications abilities are added to the SMART platform, an operator could theoretically control the SMART platform from any location in the world. This increased capability would allow for real time command, control and access to sensor data as it was gathered at the highest levels of the chain of command making the SMART platform a valuable piece of the network-centric war.

1. HTML Pages

The present web of the SMART platform is rudimentary and is meant for a proof of concept only. A home page which will eventually be password protected provide credit to NPS and the CSS&T curriculum and acts as an access point to the following pages. Subsequent pages credit people who have worked on the SMART initiative and another provides links to hardware used during the projects. The last page downloads the JAVA control applet and will be further password protected.

THIS PAGE INTENTIONALLY LEFT BLANK

IV. CONCLUSIONS/SMART FUTURE

Autonomous and remotely controlled robotic systems will certainly have an increased role in future conflicts. The SMART vision was to create an ongoing research effort within the CSS&T Curriculum that engages in a forward-looking application of small robotic technology for military employment that goal has been reached with the completion of this second thesis of the SMART initiative. The specific goals accomplished during this thesis research included:

- The successful operational demonstration of a lemming tracked vehicle in manual and autonomous modes.
- The unique integration of diverse COTS components into a single operational platform.
- The successful programming and integration of a GPS and digital compass navigation system.
- The smooth interface between the microprocessor and the platform motors using a self designed PWM motor control circuit and COTS electronic speed control units.
- The establishment of wireless networking for communications using TCP/IP socket connections.
- The creation of a user friendly GUI using a JAVA application and the subsequent conversion of this GUI to a downloadable JAVA applet.

With the completion of this initial research, the SMART platform waits for the completion of further sensor applications so that integration with these new platforms can be made.

The future of the SMART initiative has many options; ongoing research within the CSS&T curriculum will certainly generate many new and exciting sensor platform options. The current Seismo-acoustic sonar sensor will be fully integrated with the SMART platform within a year and other projects include IR and chemical agent sensors. The capabilities of the SMART platform will also continue to increase, initial research has already been conducted into a video system and future plans include the design and production of a complete platform within the CSS&T curriculum accounting for all expected future requirements. The next generation SMART platform will employ advanced battery and power usage techniques as well as a modified track layout to enable

increased maneuverability. The addition of a second platform will enable more advanced research in the areas of network centric warfare and cooperative engagements.

The SMART initiative in the CSS&T curriculum at NPS has the potential to contribute greatly to the robotic capability of our expeditionary and covert forces. The addition of the SMART initiative strongly compliments ongoing NPS research into unmanned airborne vehicles, unmanned underwater vehicles and space-based systems. As missions such as surveillance, mine sweeping and chemical detection become increasingly dangerous, continued research in the area of autonomous and remotely controlled sensing platforms will be a cornerstone in our preparation for future conflict.

APPENDIX A. CONTROL PROGRAM IN DYNAMIC C

```
/* Alltask.c
   Andrew G. Chicoine
   Novemb er 2001

This program is written for the Z-World BL2000.

 This program multitasks the functions of the SMART robot:

  1. Manual control.
  2. GPS location and Compass heading updates.
  3. Automatic GPS Navigation.
  4. Stop Navigation Routine.
  5. Serve Web page.

**************************************************/

//VARIABLE DEFINITIONS FOR I/O AND INTERNET CONNECTIONS

#define SERIAL_PORT_SPEED 115200

//Some network stuff

#define MY_IP_ADDRESS       "131.120.101.81"
#define MY_NETMASK          "255.255.240.0"
#define MY_GATEWAY          "131.120.96.1"
#define MY_NAMESERVER       "131.120.254.58"
#define MAX_SOCKETS 6

//MAPPING AND LIBRARY FILES TO USE

#memmap xmem
#use "dcrtcp.lib"
#use "vserial.lib"
#use "http.lib"
```

```
//IMPORT WEB PAGE

#ximport "thesis/html/Robot1.html"    Robot1_html
#ximport "thesis/html/Robot2.html"    Robot2_html
#ximport "thesis/html/Robot3.html"    Robot3_html
#ximport "thesis/html/Robot4.html"    Robot4_html
#ximport "thesis/html/Flag.gif"    Flag_gif
#ximport "thesis/html/nps.gif"    nps_gif
#ximport "thesis/html/robot.gif"    robot_gif

const HttpType http_types[] =
{
  { ".html", "text/html", NULL},
  { ".gif", "image/gif", NULL},
//  { ".jpg", "image/jpg", NULL}
};

const HttpSpec http_flashspec[] =
{
  { HTTPSPEC_FILE,  "/",             Robot1_html,    NULL, 0, NULL, NULL},
  { HTTPSPEC_FILE,  "/Robot1.html",   Robot1_html,    NULL, 0, NULL, NULL},
  { HTTPSPEC_FILE,  "/Robot2.html",   Robot2_html,    NULL, 0, NULL, NULL},
  { HTTPSPEC_FILE,  "/Robot3.html",   Robot3_html,    NULL, 0, NULL, NULL},
  { HTTPSPEC_FILE,  "/Robot4.html",   Robot4_html,    NULL, 0, NULL, NULL},
  { HTTPSPEC_FILE,  "/Flag.gif",   Flag_gif,   NULL, 0, NULL, NULL},
  { HTTPSPEC_FILE,  "/nps.gif",   nps_gif,   NULL, 0, NULL, NULL},
  { HTTPSPEC_FILE,  "/robot.gif",   robot_gif,   NULL, 0, NULL, NULL},
};

//DEFINITIONS FOR ROBOT CONTROL

#define PORT0 2000
#define PORT1 2001
#define PORT2 2002
#define PORT3 2003
#define PORT4 2004
```

```
#define PORT5 2005

tcp_Socket Motorsock;
tcp_Socket Headsock;
tcp_Socket GPSsock;
tcp_Socket DHeadLocsock;
tcp_Socket StopNavsock;
tcp_Socket Statussock;

#define BINBUFSIZE  63
#define BOUTBUFSIZE 63
#define CINBUFSIZE  1023
#define COUTBUFSIZE 1023
#define TIMEOUT 40UL   // will time out 20 milliseconds after receiving any
                       // character unless MAXSIZE characters are received
#define MAXSIZE  128
#define MAXSIZE2 160

char HeadIn[MAXSIZE];
char GPSIn[MAXSIZE];
char HeadOut[MAXSIZE2];
char GPSOut[MAXSIZE2];
int x;
char HeadBuffer[2048];
char GPSBuffer[2048];
char DHeadLocBuffer[127];
char StopNavBuffer[31];
char DHeadLoc[127];
char StopNav[127];

//VARIABLE DEFINITIONS FOR NAVIGATION

float AVGCSE;     /*   Average heading to minimize hunting  */
float CTS;        /*   Course to steer  computed from waypoint and current location  */
float ABSDIFF;    /*   Absolute difference between CTS and AVGCSE  */
float DIST;       /*   Distance btween current location and WP  */
float SPD;        /*   SPEED  */
```

```
float DIFF;          /*  Difference between CTS and AVGCSE  */
long i;              /*  counter for delay          */
int j;               /*  counter for nav loop*/
int n;               /*  subscript for variables  */
int WPHEAD;
double WPLATDEG;     /*  Waypoint position degrees of Latitude    */
double WPLATMIN;     /*  Waypoint position minutes of Latitude    */
double WPLONGDEG;    /*  Waypoint position degrees of Longitude   */
double WPLONGMIN;    /*  Waypoint position degrees of Longitude   */
double HDGMAG;       /*  Compass heading*/
double HDGTRU;       /*  Combination of HDGMAG + VAR + DEV*/
const double DEV = 4;    /*  Deviation for current location*/
const double VAR = 0;    /*  Variation due to motors*/
double SOG;          /*  Speed over ground of robot*/
double Speed;        /*  Dummy variable for SOG*/
double COG;          /*  GPS course over ground of robot*/
longword ip;
int status;
int count;
float TurnTime;      /*Proportional control for major turns*/

//MOTOR CONTROL VARIABLES

char MotorBuffer[512];
char DMotor[31];

const int channel0 = 0;
const int channel1 = 1;

float Motor_Speed;
float LMotor_Speed;
float RMotor_Speed;
int Motor_Desig;
```

```
main()
{

//STRING PARSE VARIABLES

char *DHeadLoc0,*DHeadLoc1,*DHeadLoc2,*DHeadLoc3,*DHeadLoc4,*DHeadLoc5,
*DHeadLoc6,*DHeadLoc7;
char *DMotor0,*DMotor1,*DMotor2,*DMotor3,*DMotor4;
char p[2];
        p[0] = ',';
        p[1] = 0;
         /*Tokenizer*/
//Open both serial ports
        serBopen(9600);
        serCopen(9600);

//Flush serial port B & C input buffer
        serBrdFlush();
        serCrdFlush();

        sock_init();
        brdInit();
        http_init();
        tcp_reserveport(80);

//Motor Socket

        tcp_listen(&Motorsock,PORT0,0,0,NULL,0);
        sock_wait_established(&Motorsock,0,NULL,&status);
        sock_mode(&Motorsock,TCP_MODE_ASCII);

while(1)
{
costate{

// This costate manages motor control in the manual mode.
```

```
// Recieve data and feedback.
//                              printf("1");
        sock_tick(&Motorsock,&status);
                if (sock_bytesready(&Motorsock)>=0){
                        x = sock_gets(&Motorsock,MotorBuffer,512);
//                      printf("%d\n", x);
                                if(x>3){
                        sock_tick(&Motorsock,&status);
                        sprintf(DMotor, "\"%s\"\n",MotorBuffer);
                        sprintf(MotorBuffer, "\"%s\"\n\n",NULL);
//                      printf("%s\n", DMotor);
                        sock_tick(&Motorsock,&status);
//                      waitfor(DelayMs(50));
                        sock_puts(&Motorsock,MotorBuffer);
//                      waitfor(DelayMs(50));
                        sock_tick(&Motorsock,&status);

// Control motors

                        DMotor0 = strtok(DMotor, p);
                        DMotor1 = strtok(NULL, p);
                                Motor_Desig = atoi(DMotor1);
//                              printf("Motor_Desig = %d\nn", Motor_Desig);
                        sock_tick(&Motorsock,&status);
                        DMotor2 = strtok(NULL, p);
                                Motor_Speed = atof(DMotor2);
//                              printf("Speed... Volts out = %f\n\n", Motor_Speed);
                        sock_tick(&Motorsock,&status);
                        DMotor3 = strtok(NULL, p);

                        if (Motor_Desig == 0) /*LEFT*/
                                LMotor_Speed = Motor_Speed/100;

                        if (Motor_Desig == 1) /*RIGHT*/
                                RMotor_Speed = Motor_Speed/100;

                        if (Motor_Desig == 2){ /*BOTH*/
```

```
                                        LMotor_Speed = Motor_Speed/100;
                                        RMotor_Speed = Motor_Speed/100;}

                                if (Motor_Desig == 3){ /*STOP*/
                                        LMotor_Speed = 1.5;
                                        RMotor_Speed = 1.5;}

                                Motor();
                                        }
                                }
                        }//End Costate

costate{

// This costate updates the GPS location and Heading in the Java application.
//                                      printf("2");

//Flush serial port B & C input buffer

        serBrdFlush();
        serCrdFlush();

//Head Socket

        tcp_listen(&Headsock,PORT1,0,0,NULL,0);
        waitfor(sock_established(&Headsock));
        sock_mode(&Headsock,TCP_MODE_ASCII);

//B Wait for the Heading message
        while ((n = serBread(HeadIn, MAXSIZE-1, TIMEOUT)) == 0) ;
                HeadIn[n] = 0;

//Copy Heading to buffer and send
                sprintf(HeadOut, "%s",HeadIn);
                sprintf(HeadBuffer, "\"%s\"\n\n",HeadOut);
                sock_puts(&Headsock,HeadBuffer);
                sock_close(&Headsock);
```

```
                    // printf("\nHead done\n");

//GPS Socket

                    tcp_listen(&GPSsock,PORT2,0,0,NULL,0);
                    sock_wait_established(&GPSsock,0,NULL,&status);
                    sock_mode(&GPSsock,TCP_MODE_ASCII);

//C Wait for the GPS message
                    while ((n = serCread(GPSIn, MAXSIZE-1, TIMEOUT)) == 0) ;
                              GPSIn[n] = 0;

//Copy messages to out variables and send
                    sprintf(GPSOut, "%s",GPSIn);
                    sprintf(GPSBuffer, "\"%s\"\n\n",GPSOut);
                    sock_puts(&GPSsock,GPSBuffer);
                    sock_close(&GPSsock);
                    // printf("\nGPS done\n");

                    } //End costate

costate{

// This costate manages navigation.
// printf("3");
//DHeadLoc Socket

                    tcp_listen(&DHeadLocsock,PORT3,0,0,NULL,0);
                    waitfor(sock_established(&DHeadLocsock));
                    sock_mode(&DHeadLocsock,TCP_MODE_ASCII);
                    sock_wait_input(&DHeadLocsock,0,NULL,&status);
                    x = sock_gets(&DHeadLocsock,DHeadLocBuffer,512);
                    sprintf(DHeadLoc, "\"%s\"\n\n",DHeadLocBuffer);
                    // printf("%s\n", DHeadLoc);
                    sock_close(&DHeadLocsock);

                    // printf("\nDHeadLoc done\n");
```

```
	sock_err:
		switch(status) {
			case 1: /* foreign host closed */
				// printf("User closed session\n");
				break;

			case -1: /* timeout */
				// printf("\nConnection timed out\n");
				break;}

//Check if desired GPS location or heading has been entered

	if(strlen(DHeadLoc)>8){

/*********************************************************************/
/*Move towards a given GPS Location or Heading.*/
/*********************************************************************/

	//printf("\nCommencing Navigation loop\n");

//Convert DHeadLoc String to Way Point Variables

	//printf("%s\n",DHeadLoc);
	x = strlen(DHeadLoc);
	//printf("Stringlength = %f\n", x);
	DHeadLoc0 = strtok(DHeadLoc, p);
	DHeadLoc1 = strtok(NULL, p);
	WPHEAD = atoi(DHeadLoc1);
	printf("WPHEAD = %d\n", WPHEAD);
	DHeadLoc2 = strtok(NULL, p);
	WPLATDEG = atof(DHeadLoc2);
	printf("WPLat Degrees = %f\n", WPLATDEG);
	DHeadLoc3 = strtok(NULL, p);
	WPLATMIN = atof(DHeadLoc3);
	printf("WPLat Minutes = %f\n", WPLATMIN);
	DHeadLoc4 = strtok(NULL, p);
```

```
        DHeadLoc5 = strtok(NULL, p);
        WPLONGDEG = atof(DHeadLoc5);
        printf("WPLong Degrees = %f\n", WPLONGDEG);
        DHeadLoc6 = strtok(NULL, p);
        WPLONGMIN = atof(DHeadLoc6);
        printf("WPLong Minutes = %f\n", WPLONGMIN);
        DHeadLoc7 = strtok(NULL, p);

DIST = 25;

//Main Navigation Loop

                while(DIST>20){

                // printf("3");

/*Initial read.*/
                navread();
                yield;
/*Check for major course correction and turn.*/

                        while (ABSDIFF >= 15 && (360-ABSDIFF) >= 15 && DIST >20){
                                navread();
                                if (ABSDIFF <= 180) {
                                        if (DIFF <= 0){printf("1");   /*Right turn*/
                                                LMotor_Speed = 2;
                                                RMotor_Speed = 1;}
                                        else{/*Left turn*/
                                                LMotor_Speed = 1;
                                                RMotor_Speed = 2;}
                                                }
                                else{
                                        if (DIFF > 0) {       /*Right turn*/
                                                DIFF= 360-ABSDIFF;
                                                LMotor_Speed = 2;
                                                RMotor_Speed = 1;}
                                        else{/*Left turn*/
```

```
                    DIFF = 360-ABSDIFF;
                    LMotor_Speed = 1;
                    RMotor_Speed = 2;}
                    }

                Motor();

/*Determine time for turn depending on error   PROPORTIONAL CONTROL*/

            if (ABSDIFF <= 180)
                TurnTime = ABSDIFF/180;
            else
                TurnTime = (360 - ABSDIFF)/180;
                TurnTime = TurnTime*250000;
                    for (i = 1; i <= TurnTime; i++);
            /*Motors Stop*/
            LMotor_Speed = 1.5;
            RMotor_Speed = 1.5;

                Motor();
                yield;
                }

/*Midcourse correction.*/

        if (ABSDIFF >= 5 && (360-ABSDIFF) >=5){
            if (ABSDIFF <= 180) {
                if (DIFF <= 0){     /*Right turn*/
                    LMotor_Speed = 2;
                    RMotor_Speed = 1;}

                else{/*Left turn*/
                    LMotor_Speed = 1;
                    RMotor_Speed = 2;}
                    }
            else{
                if (DIFF > 0) {/*Right turn*/
```

```
                    DIFF= 360-ABSDIFF;
                    LMotor_Speed = 2;
                    RMotor_Speed = 1;}
                else{/*Left turn*/
                    DIFF = 360-ABSDIFF;
                    LMotor_Speed = 1;
                    RMotor_Speed = 2;}
                    }
                Motor();
                }
                for (i = 1; i <= 25000; i++);

                /*Forward movement.*/

        if (DIST > 20){/*Move forward*/
            LMotor_Speed = 2;
            RMotor_Speed = 2;
            Motor();
            for (i = 1; i <= 150000; i++);}
        else{/*Motors Stop*/
            LMotor_Speed = 1.5;
            RMotor_Speed = 1.5;
            Motor();}
            yield;
        }

//EXIT FROM LOOP DANCE

        /*Motors Stop*/
        LMotor_Speed = 1.5;
        RMotor_Speed = 1.5;
        Motor();
        for (i = 1; i <= 50000; i++);
        /*Right Turn*/
        LMotor_Speed = 2;
        RMotor_Speed = 1;
        Motor();
```

```
                        for (i = 1; i <= 50000; i++);
                        /*Left Turn*/
                        LMotor_Speed = 1;
                        RMotor_Speed = 2;
                        Motor();
                        for (i = 1; i <= 50000; i++);
                        /*Motors Stop*/
                        LMotor_Speed = 1.5;
                        RMotor_Speed = 1.5;
                        Motor();
                        for (i = 1; i <= 50000; i++);
                        }

else;{

/*********************************************************************/
/*Only update GPS & Compass data.*/
/*********************************************************************/
}
//          printf("\nLoop done\n");

                        }//End Navigation Costate

costate{ /*This costate stops the navigation loop*/

//StopNav Socket
//printf("4");

                tcp_listen(&StopNavsock,PORT4,0,0,NULL,0);
                waitfor(sock_established(&StopNavsock));
                sock_mode(&StopNavsock,TCP_MODE_ASCII);
                sock_close(&StopNavsock);
                DIST = 0;
                        }//End Costate

 costate{ /*This costate operates the web page */
 //                     printf("5");
```

```
    http_handler();
        }//End Costate

            }//End While
        }//End Main

/*********************************************************************/
/*Function to read in Navigation data and convert.*/
/*********************************************************************/

navread(){



double LATDEG;          /*  Current position  degrees of Latitude      */
double LATMIN;          /*  Current position  minutes of Latitude      */
double LONGDEG;         /*  Current position  degrees of Longitude     */
double LONGMIN;         /*  Current position  minutes of Longitude     */
double LATm;            /*  Converted Latitude into meters from equator        */
double LONGm;           /*  Converted Longitude into meters from meridian      */
double WPLATm;          /*  Converted Latitude into meters from equator*/
double WPLONGm;         /*  Converted Longitude into meters from meridian*/
double NUMSAT;          /*  Number of satalites*/
double ALT;             /*  Altitude in meters*/
double DOP;             /*  Quality of GPS signal*/
double DIFFLATm;        /*    Difference in latitude between the current position and WP
(meters)*/
double DIFFLONGm;       /*    Difference in longitude between the current position and WP
(meters)*/
const double pi = 3.14; /*  Pi     */

//STRING PARSE VARIABLES

        char *GPSIn0,*GPSIn1,*GPSIn2,*GPSIn3,*GPSIn4,*GPSIn5,*GPSIn6;
        char *GPSIn7,*GPSIn8,*GPSIn9,*GPSIn10,*GPSIn11,*GPSIn12,*GPSIn13;
        char *GPSIn14,*GPSIn15,*GPSIn16,*GPSIn17,*GPSIn18,*GPSIn19,*GPSIn20;
        char *GPSIn21,*GPSIn22;
        char GPS7[10],GPS8[10],GPS10[10],GPS11[10];
```

```
        char q[2];
                q[0] = ',';
                q[1] = 0;
                 /*Tokenizer*/

        if(x>30){

//Flush serial port B & C input buffer

        serBrdFlush();
        serCrdFlush();

        tcp_tick(&StopNavsock);

//Wait for the GPS message from the GPS card, tokenize and convert to integers

        while ((n = serCread(GPSIn, MAXSIZE-1, TIMEOUT)) == 0) ;
        GPSIn[n] = 0;

                if(strlen(GPSIn)>92){
                // printf("%s\n",GPSIn);

                        GPSIn0 = strtok(GPSIn, q);
                        GPSIn1 = strtok(NULL, q);
                        GPSIn2 = strtok(NULL, q);
                        GPSIn3 = strtok(NULL, q);
                        GPSIn4 = strtok(NULL, q);
                        GPSIn5 = strtok(NULL, q);
                        GPSIn6 = strtok(NULL, q);
                        GPSIn7 = strtok(NULL, q);
                        // printf("GPSIn7 = %s\n",GPSIn7);
                        LATDEG = atof(GPSIn7);
                        // printf("Lat Degrees = %f\n", LATDEG);
                        GPSIn8 = strtok(NULL, q);
                        // printf("GPSIn8 = %s\n",GPSIn8);
                        LATMIN = atof(GPSIn8);
                        // printf("Lat Minutes = %f\n", LATMIN);
```

```
                    GPSIn9 = strtok(NULL, q);
                    GPSIn10 = strtok(NULL, q);
                    // printf("GPSIn10 = %s\n",GPSIn10);
                    LONGDEG = atof(GPSIn10);
                    // printf("Long Degrees = %f\n", LONGDEG);
                    GPSIn11 = strtok(NULL, q);
                    // printf("GPSIn11 = %s\n",GPSIn11);
                    LONGMIN = atof(GPSIn11);
                    // printf("Long Minutes = %f\n", LONGMIN);
                    GPSIn12 = strtok(NULL, q);
                    GPSIn13 = strtok(NULL, q);
                    GPSIn14 = strtok(NULL, q);
                    GPSIn15 = strtok(NULL, q);
                    GPSIn16 = strtok(NULL, q);
                    GPSIn17 = strtok(NULL, q);
                    GPSIn18 = strtok(NULL, q);
                    GPSIn19 = strtok(NULL, q);
                    GPSIn20 = strtok(NULL, q);
                    GPSIn21 = strtok(NULL, q);
                    GPSIn22 = strtok(NULL, q);
          }

//scanf("%f",&SOG);
//scanf("%f",&COG);

//Flush serial port B & C input buffer

                    serBrdFlush();
                    serCrdFlush();

                    tcp_tick(&StopNavsock);
//Wait for the Heading message from compass card and convert to an integer

                    while ((n = serBread(HeadIn, MAXSIZE-1, TIMEOUT)) == 0) ;
                    HeadIn[n] = 0;
                    HDGMAG = atoi(HeadIn);
```

```
        tcp_tick(&StopNavsock);
/*Convert degree based LAT/LONG to meters.*/

LATm = ((LATDEG* 60)+LATMIN)*1852;
LONGm = -((LONGDEG*60)+LONGMIN)*1852*cos (.628 );
// printf("longm = %f\n",LONGm);
// printf("latm = %f\n",LATm);

/*Convert Waypoint LAT/LONG to meters.*/

WPLATm = ((WPLATDEG*60)+WPLATMIN)*1852;
WPLONGm = -((WPLONGDEG*60)+WPLONGMIN)*1852*cos (.628);
// printf("wplongm = %f\n",WPLONGm);
// printf("wplatm = %f\n",WPLATm);

/*Determine hdg and dist to WP.*/

DIFFLONGm = WPLONGm - LONGm;
DIFFLATm = WPLATm - LATm;
// printf("difflongm = %f\n",DIFFLONGm);
// printf("difflatm = %f\n",DIFFLATm);

DIST = sqrt (DIFFLONGm*DIFFLONGm + DIFFLATm*DIFFLATm);
// printf("dist = %f\n",DIST);

//DIVIDE BY ZERO PREVENTION

        if(DIFFLONGm == 0)
                DIFFLONGm = 0.001;
        if(DIFFLATm == 0)
                DIFFLATm = 0.001;

//Conditional statements to determine heading depending upon quadrant

if(DIFFLONGm>=0 && DIFFLATm>=0)
        {CTS = 180/pi * atan (DIFFLONGm/DIFFLATm);}
```

```
else if(DIFFLONGm>=0 && DIFFLATm<=0)
        {CTS = 90 - 180/pi * atan (DIFFLATm/DIFFLONGm);}

else if(DIFFLONGm<=0 && DIFFLATm>=0)
        {CTS = 360 + 180/pi * atan (DIFFLONGm/DIFFLATm);}

else
        {CTS = 180 + 180/pi * atan (DIFFLATm/DIFFLONGm);}

 printf("cts = %f\n",CTS);

/*Apply magnetic dev and var corrections and determine velocity and time average hdg.*/

HDGTRU = HDGMAG + DEV + VAR;
if (SOG > 1) Speed = SOG;

        else    Speed = 0;
if (fabs(AVGCSE - HDGTRU) >= 20) AVGCSE = HDGTRU;
AVGCSE = (AVGCSE + HDGTRU + Speed*COG)/(2+Speed);
ABSDIFF = fabs(AVGCSE-CTS);
DIFF =(AVGCSE-CTS);

//for (i = 1; i <= 100000; i++);

/*Test output.*/

 printf("hdgtru = %f\n",HDGTRU);
// printf("AVGCSE = %f\n",AVGCSE);
 printf("absdiff = %f\n",ABSDIFF);
 printf("diff = %f\n",DIFF);
}

        else{

/*************************************************************************/
/*Function to read in Heading data and convert.*/
/*************************************************************************/
```

```
//Flush serial port B & C input buffer

        serBrdFlush();
        serCrdFlush();


//Wait for the Heading message from compass card and convert to an integer

        while ((n = serBread(HeadIn, MAXSIZE-1, TIMEOUT)) == 0) ;
        HeadIn[n] = 0;
        HDGMAG = atoi(HeadIn);
        CTS = WPHEAD;
         printf("cts = %f\n",CTS);
/*Apply magnetic dev and var corrections and determine velocity and time average hdg.*/

        HDGTRU = HDGMAG + DEV + VAR;
        if (SOG > 1) Speed = SOG;
        else        Speed = 0;
        if (fabs(AVGCSE - HDGTRU) >= 20) AVGCSE = HDGTRU;
        AVGCSE = (AVGCSE + HDGTRU + Speed*COG)/(2+Speed);
        ABSDIFF = fabs(AVGCSE-CTS);
        DIFF =(AVGCSE-CTS);
        }
}
```

```
/*********************************************************************/
/*Function to Control Motors.*/
/*********************************************************************/

Motor(){

        /*      Analog channel 0 = Left Motor
                Analog channel 1 = Right Motor */

                        anaOutVolts(channel0, LMotor_Speed);
                        anaOutVolts(channel1, RMotor_Speed);
                }
```

APPENDIX B. CONTROL INTERFACE IN JAVA

```
//RobotControl.java
//Andrew G. Chicoine
//November 2001

//This program is the GUI for the SMART robot.
//1. Slider and pushbutton control.
//2. Receive GPS & Compass updates.
//3. Send GPS & heading commands.



//import javax.media.*;
//import com.sun.media.ui.*;
//import javax.media.protocol.*;
//import javax.media.protocol.DataSource;
import javax.swing.*;
import javax.swing.event.*;
import java.awt.*;
import java.awt.event.*;
import java.net.*;
import java.io.*;
import java.util.Vector;
import java.util.*;

public class RobotControl extends JFrame {

   public    static final int REVERSE = 100;
   public    static final int FORWARD = 200;
   public    static final int STOP = 150;

   private   JPanel mainPanel = new JPanel(new BorderLayout());

   protected JTabbedPane tabs = new JTabbedPane();
   protected DefaultListModel model = new DefaultListModel();
   protected JList list = new JList(model);
   protected JSlider leftMotor;
   protected JSlider rightMotor;
   protected JSlider bothMotor;
   protected Socket sock;
   protected BufferedReader cmdIn;
   protected OutputStream cmdOut;

/************************************/

   private JTextField lat, longi, head, Dcourse, DLat, DLong, RobotStatus;
   private int numcourse, numDcourse, numDLatDeg, numDLatMin, numDLongDeg, numDLongMin;
   private Button Update;
   private Button StopNav;
   private Button Go;
   String incourse = new String();
   String inlat = new String();
   String inlongi = new String();
```

```
  String inHead = new String();
  String inGPS = new String();
  String StrDHead = new String();
  String StrDLat = new String();
  String StrDLong = new String();
  String StrDHeadLoc = new String();
  String nohit = "No GPS Update Available...";
  String HO = "Turning to desired heading";
  String NAV = "Commencing Navigation Routine";
  String NA = "No action performed";
  String LT = "Turning Left...";
  String RT = "Turning Right...";
  String TC = "Turn Complete...";

//NETWORK VARIABLES FOR HEADING AND GPS

  private Socket Headsock;
  private Socket GPSsock;
  private Socket DHeadLocsock;
  private Socket StopNavsock;

  private BufferedReader HeadIn;
  private OutputStream HeadOut;

  private BufferedReader GPSIn;
  private OutputStream GPSOut;

  private BufferedReader DHeadLocIn;
  private OutputStream DHeadLocOut;

  private BufferedReader StopNavIn;
  private OutputStream StopNavOut;

  private static final String ip = "131.120.101.81";


/*************************************************************************/

    public RobotControl(String ip) {
     super("Robot Control");
     setDefaultCloseOperation(JFrame.DO_NOTHING_ON_CLOSE);
     addWindowListener(
       new WindowAdapter() {
         public void windowClosing(WindowEvent ev) {
           terminate();
         }
       });

     setLocation(200,350);

/*************************************************************************/

/* This section sets up the network socket and port for the client/server
  relationship */

//NETWORKING for CLIENT/SERVER
```

```
      try {
       sock = new Socket(ip, 2000);
       cmdIn = new BufferedReader(new InputStreamReader(sock.getInputStream()));
       cmdOut = sock.getOutputStream();
      } catch (Exception ex) {
       System.err.println("Could not connect to " + ip + ".\nQuitting.");
       System.exit(0);
      }


/*************************************************************************/
  // GENERAL LAYOUT (Panel Assignments)

          GridBagLayout gbl = new GridBagLayout();
     GridBagConstraints gbc = new GridBagConstraints();

   //MANUAL CONTROL TAB
      JPanel main = new JPanel(new BorderLayout());
      //Sliders
      JPanel controls = new JPanel(gbl);
      //Buttons
      JPanel butt = new JPanel(gbl);

       GridBagLayout gblay = new GridBagLayout();
       GridBagConstraints gbcon = new GridBagConstraints();


   //AUTO-MODE TAB
      JPanel autonomous = new JPanel(new BorderLayout());
      //Lat, Long, & Heading
      JPanel q = new JPanel(gblay);
      //Robot Status
      JPanel p = new JPanel(gblay);
      //Navigation GPS
      JPanel s = new JPanel(gblay);

/*************************************************************************/

/* This is the motor controller section.  Control sliders are created for
   the Left and Right Motor Speeds.  Control Buttons are created for
   movement and direction. */

/****************** CONTROL SLIDERS ********************/

//LEFT MOTOR SPEED ONLY
     leftMotor = new JSlider(JSlider.HORIZONTAL, REVERSE, FORWARD, STOP);
     leftMotor.addChangeListener(
       new ChangeListener() {
         public void stateChanged(ChangeEvent ev) {
           int val = leftMotor.getValue();
           try {
/* First variable x is a place holder, second variable is the motor designator:
   L=0,R=1,B=2,S=3, val is the speed variable. */

             cmdOut.write(("x,0," + val + ",\n").getBytes());
```

```
                                for (int i = 0; i<100000;i++){}
                    model.addElement(cmdIn.readLine());
                  } catch (Exception ex) {
                  }
                }
              });
            leftMotor.setMajorTickSpacing(50);
            leftMotor.setMinorTickSpacing(5);
            leftMotor.setSnapToTicks(true);
            leftMotor.setPaintTicks(true);

           //Create the label table
            Hashtable labelTable = new Hashtable();
            labelTable.put( new Integer( REVERSE ), new JLabel("R") );
            labelTable.put( new Integer(  STOP   ), new JLabel("S") );
            labelTable.put( new Integer( FORWARD ), new JLabel("F") );

            leftMotor.setLabelTable( labelTable );
            leftMotor.setPaintLabels(true);
            leftMotor.setBorder(BorderFactory.createEmptyBorder(0,0,10,0));
               gbc.gridx = 0;
               gbc.gridwidth = 2;
               gbc.gridy = 0;
               gbc.gridheight = 2;
            JLabel label = new JLabel("Left Motor");
               gbl.setConstraints(label, gbc);
            controls.add(label);
               gbc.gridx = 2;
               gbc.gridy = 0;
               gbc.gridy = 0;
               gbl.setConstraints(leftMotor, gbc);
            controls.add(leftMotor);


        //RIGHT MOTOR SPEED ONLY
            rightMotor = new JSlider(JSlider.HORIZONTAL, REVERSE, FORWARD, STOP);
            rightMotor.addChangeListener(
              new ChangeListener() {
                public void stateChanged(ChangeEvent ev) {
                  int val = rightMotor.getValue();
                  try {
                    cmdOut.write(("x,1," + val + ",\n").getBytes());
                                for (int i = 0; i<100000;i++){}
                    model.addElement(cmdIn.readLine());
                  } catch (Exception ex) {
                  }
                }
              });
            rightMotor.setMajorTickSpacing(50);
            rightMotor.setMinorTickSpacing(5);
            rightMotor.setSnapToTicks(true);
            rightMotor.setPaintTicks(true);

           //Create the label table
            rightMotor.setLabelTable( labelTable );
            rightMotor.setPaintLabels(true);
            rightMotor.setBorder(BorderFactory.createEmptyBorder(0,0,10,0));
```

```
        gbc.gridx = 0;
        gbc.gridwidth = 2;
        gbc.gridy = 2;
        gbc.gridheight = 2;
      label = new JLabel("Right Motor");
        gbl.setConstraints(label, gbc);
      controls.add(label);
        gbc.gridx = 2;
        gbc.gridy = 2;
        gbl.setConstraints(rightMotor, gbc);
      controls.add(rightMotor);


//BOTH MOTOR SPEEDS
    bothMotor = new JSlider(JSlider.HORIZONTAL, REVERSE, FORWARD, STOP);
    bothMotor.addChangeListener(
      new ChangeListener() {
        public void stateChanged(ChangeEvent ev) {
          int val = bothMotor.getValue();
          leftMotor.setValue(val);
          rightMotor.setValue(val);
          try {
            cmdOut.write(("x,2," + val + ",\n").getBytes());
                              for (int i = 0; i<100000;i++){}
            model.addElement(cmdIn.readLine());
          } catch (Exception ex) {
          }
        }
      });
    bothMotor.setMajorTickSpacing(10);
    bothMotor.setMinorTickSpacing(5);
    bothMotor.setSnapToTicks(true);
    bothMotor.setPaintTicks(true);

//Create the label table
    bothMotor.setLabelTable( labelTable );
    bothMotor.setPaintLabels(true);
    bothMotor.setBorder(BorderFactory.createEmptyBorder(0,0,10,0));
        gbc.gridx = 0;
        gbc.gridwidth = 2;
        gbc.gridy = 4;
        gbc.gridheight = 2;
      label = new JLabel("Both Motors");
        gbl.setConstraints(label, gbc);
      controls.add(label);
        gbc.gridx = 2;
        gbc.gridy = 4;
        gbl.setConstraints(bothMotor, gbc);
      controls.add(bothMotor);

      main.add(controls, BorderLayout.WEST);


/********************* CONTROL BUTTONS ***********************/

//STOP BUTTON
```

```
    JButton stop = new JButton("Stop");
    stop.addActionListener(
      new ActionListener() {
        public void actionPerformed(ActionEvent ev) {
          try {
            cmdOut.write("x,3,\n".getBytes());
            model.addElement(cmdIn.readLine());
          } catch (Exception ex) {
          }
          leftMotor.setValue(STOP);
          rightMotor.setValue(STOP);
          bothMotor.setValue(STOP);
        }
      });
    gbc.gridx = 3;
    gbc.gridy = 3;
    gbc.gridwidth = 2;
    gbc.gridheight = 2;
    gbc.ipadx = 10;
    gbc.ipady = 10;
    gbc.insets = new Insets(10,0,0,0);
    gbl.setConstraints(stop, gbc);
    butt.add(stop);


//FORWARD BUTTON
    JButton forward = new JButton("Forward");
    forward.addActionListener(
      new ActionListener() {
        public void actionPerformed(ActionEvent ev) {
                    try {
            cmdOut.write("x,2,200,\n".getBytes());
            model.addElement(cmdIn.readLine());
          } catch (Exception ex) {
          }
          leftMotor.setValue(STOP+50);
          rightMotor.setValue(STOP+50);
          bothMotor.setValue(STOP+50);
        }
      });
    gbc.gridx = 3;
    gbc.gridy = 1;
    gbc.gridwidth = 2;
    gbc.gridheight = 1;
    gbc.ipadx = 20;
    gbc.ipady = 20;
    gbc.insets = new Insets(5,5,5,5);
    gbl.setConstraints(forward, gbc);
    butt.add(forward);


//LEFT TURN BUTTON
    JButton leftturn = new JButton("Left Turn");
    leftturn.addActionListener(
      new ActionListener() {
        public void actionPerformed(ActionEvent ev) {
```

```
            try {
//            cmdOut.write("x,3,\n".getBytes());
              model.addElement(cmdIn.readLine());
            } catch (Exception ex) {
            }
            leftMotor.setValue(STOP-50);
            rightMotor.setValue(STOP+50);
          }
        });
      gbc.gridx = 1;
      gbc.gridy = 3;
      gbc.gridheight = 4;
      gbc.ipadx = 20;
      gbc.ipady = 20;
      gbl.setConstraints(leftturn, gbc);
      butt.add(leftturn);


  //RIGHT TURN BUTTON
      JButton rightturn = new JButton("Right Turn");
      rightturn.addActionListener(
        new ActionListener() {
          public void actionPerformed(ActionEvent ev) {
            try {
//            cmdOut.write("x,3,\n".getBytes());
              model.addElement(cmdIn.readLine());
            } catch (Exception ex) {
            }
            leftMotor.setValue(STOP+50);
            rightMotor.setValue(STOP-50);
          }
        });
      gbc.gridx = 5;
      gbc.gridy = 3;
      gbc.gridheight = 4;
      gbc.ipadx = 20;
      gbc.ipady = 20;
      gbl.setConstraints(rightturn, gbc);
      butt.add(rightturn);


  //REVERSE BUTTON
      JButton backward = new JButton("Reverse");
      backward.addActionListener(
        new ActionListener() {
          public void actionPerformed(ActionEvent ev) {
            try {
//            cmdOut.write("x,3,\n".getBytes());
              model.addElement(cmdIn.readLine());
            } catch (Exception ex) {
            }
            leftMotor.setValue(REVERSE);
            rightMotor.setValue(REVERSE);
            bothMotor.setValue(REVERSE);
          }
        });
```

```
    gbc.gridx = 3;
    gbc.gridy = 7;
    gbc.gridwidth = 2;
    gbc.gridheight = 4;
    gbc.insets = new Insets(5,5,5,5);
    gbc.ipadx = 20;
    gbc.ipady = 20;
    gbl.setConstraints(backward, gbc);
    butt.add(backward);

    main.add(butt, BorderLayout.CENTER);


/*****************************************************************************************
***/

/*    SETUP UPDATEABLE DISPLAY OF HEADING, LATITUDE & LONGITUDE
        DESIRED COURSE, LATITUDE & LONGITUDE
        PUSHBUTTONS TO COMMENCE NAV/UPDATE ACTION AND STOP NAV    */

/************************** LABELS ***************************/

        JLabel label1 = new JLabel("ACTUAL ");
                gbcon.gridx = 0;
                gbcon.gridy = 3;
                gblay.setConstraints(label1, gbcon);
                q.add(label1);

        JLabel label2 = new JLabel("ORDERED");
                gbcon.gridx = 0;
                gbcon.gridy = 4;
                gblay.setConstraints(label2, gbcon);
                q.add(label2);

        JLabel label3 = new JLabel("Latitude:");
                gbcon.gridx = 1;
                gbcon.gridy = 0;
                gblay.setConstraints(label3, gbcon);
                q.add(label3);

        JLabel label4 = new JLabel("(i.e. 03,06.1234,N)");
                gbcon.gridx = 1;
                gbcon.gridy = 1;
                gblay.setConstraints(label4, gbcon);
                q.add(label4);

        JLabel label5 = new JLabel("Longitude:");
                gbcon.gridx = 2;
                gbcon.gridy = 0;
                gblay.setConstraints(label5, gbcon);
                q.add(label5);

        JLabel label6 = new JLabel("(i.e. 067,09.9876,W)");
                gbcon.gridx = 2;
                gbcon.gridy = 1;
                gblay.setConstraints(label6, gbcon);
```

```
                q.add(label6);

        JLabel label7 = new JLabel("   Course:");
                gbcon.gridx = 3;
                gbcon.gridy = 0;
                gblay.setConstraints(label7, gbcon);
                q.add(label7);

        JLabel label8 = new JLabel("   (Integer 0 to 360)");
                gbcon.gridx = 3;
                gbcon.gridy = 1;
                gblay.setConstraints(label8, gbcon);
                q.add(label8);

/***************** ACTUAL LAT, LONG & HEADING ******************/

        lat = new JTextField(20);
        gbcon.gridx = 1;
        gbcon.gridy = 3;
        gblay.setConstraints(lat,gbcon);
        q.add(lat);
                lat.setEditable(false);

        longi = new JTextField(20);
        gbcon.gridx = 2;
        gbcon.gridy = 3;
        gblay.setConstraints(longi,gbcon);
        q.add(longi);
                longi.setEditable(false);

        head = new JTextField(6);
        gbcon.gridx = 3;
        gbcon.gridy = 3;
        gblay.setConstraints(head,gbcon);
        q.add(head);
                head.setEditable(false);


/***************** DESIRED LAT, LONG & HEADING ******************/


        DLat = new JTextField(20);
        gbcon.gridx = 1;
        gbcon.gridy = 4;
        gblay.setConstraints(DLat,gbcon);
        q.add(DLat);
                DLat.setEditable(true);

        DLong = new JTextField(20);
                gbcon.gridx = 2;
        gbcon.gridy = 4;
        gblay.setConstraints(DLong,gbcon);
        q.add(DLong);
                DLong.setEditable(true);

        Dcourse = new JTextField(6);
```

```
            gbcon.gridx = 3;
            gbcon.gridy = 4;
            gblay.setConstraints(Dcourse,gbcon);
            q.add(Dcourse);
                        Dcourse.setEditable(true);

            autonomous.add(q, BorderLayout.NORTH);

/************************ STATUS ******************************/
            JLabel label9 = new JLabel(" Robot Status ");
                        gbcon.gridx = 0;
                        gbcon.gridy = 0;
                        gblay.setConstraints(label9, gbcon);
                        p.add(label9);

            RobotStatus = new JTextField(25);
            gbcon.gridx = 0;
            gbcon.gridy = 1;
                        gblay.setConstraints(RobotStatus,gbcon);
            p.add(RobotStatus);
                        RobotStatus.setEditable(false);

            autonomous.add(p, BorderLayout.SOUTH);

/******************* BUTTONS ****************************************/

      JButton Update = new JButton(" Update ");
                        gbcon.gridx = 0;
            gbcon.gridy = 0;
            gblay.setConstraints(RobotStatus,gbcon);
            s.add(Update);

      JButton Go = new JButton(" Navigate ");
                        gbcon.gridx = 1;
            gbcon.gridy = 0;
            gblay.setConstraints(RobotStatus,gbcon);
            s.add(Go);

            JButton StopNav = new JButton("  Stop Navigation  ");
                        gbcon.gridx = 0;
            gbcon.gridy = 3;
                        gblay.setConstraints(RobotStatus,gbcon);
            s.add(StopNav);


Update.addActionListener(
       new ActionListener() {
         public void actionPerformed(ActionEvent ev) {


//GET NEW HEADING, LATITUDE AND LONGITUDE
//NETWORKING for CLIENT/SERVER

     try {
        Headsock = new Socket("131.120.101.81", 2001);
        HeadIn = new BufferedReader(new InputStreamReader(Headsock.getInputStream()));
```

```
            HeadOut = Headsock.getOutputStream();
      }  catch (Exception ex) {
        System.err.println("Could not connect to " + "131.120.101.81" + ".          Quitting.");
       System.exit(0);
      }
          try {
                              inHead = (HeadIn.readLine());
                        } catch (Exception ex) {}

//UPDATE COURSE TEXTBOX

                        inHead = inHead.substring(1,4);
                        head.setText(inHead);

//NETWORKING for CLIENT/SERVER

     try {
        GPSsock = new Socket("131.120.101.81", 2002);
        GPSIn = new BufferedReader(new InputStreamReader(GPSsock.getInputStream()));
            GPSOut = GPSsock.getOutputStream();
      }  catch (Exception ex) {
        System.err.println("Could not connect to " + "131.120.101.81" + ".          Quitting.");
       System.exit(0);
      }
          try {
                              inGPS = (GPSIn.readLine());
                        } catch (Exception ex) {}



// CHECK FOR A GOOD GPS HIT THEN UPDATE TEXTBOXES

                                    int GPSlength = inGPS.length();
                                          if(GPSlength > 92){
                                                inlat = inGPS.substring(24,36);
                                                inlongi = inGPS.substring(37,50);

                                                lat.setText(inlat);
                                                longi.setText(inlongi);
                                                }
                                          else{
                                                lat.setText(nohit);
                                                longi.setText(nohit);

                                                }

                              }
          }
            );



Go.addActionListener(
       new ActionListener() {
         public void actionPerformed(ActionEvent ev) {
```

```
//RETRIEVE DESIRED COURSE or GPS POSITION

StrDHead = Dcourse.getText();
StrDLat = DLat.getText();
StrDLong = DLong.getText();
StrDHeadLoc = "x," + " " + StrDHead + "," + StrDLat + "," + StrDLong;

//UPDATE ROBOT STATUS WINDOW

if(StrDHeadLoc.length()<5)
            RobotStatus.setText(NA);
else
 if(StrDHeadLoc.length()<8)
            RobotStatus.setText(HO);
 else
            RobotStatus.setText(NAV);



//SEND DHeadLoc DATA

    try {
        DHeadLocsock = new Socket("131.120.101.81", 2003);
        DHeadLocIn = new BufferedReader(new InputStreamReader(DHeadLocsock.getInputStream()));
        DHeadLocOut = DHeadLocsock.getOutputStream();
      }  catch (Exception ex) {
        System.err.println("Could not connect to " + "131.120.101.81" + ".          Quitting.");
       System.exit(0);
      }
           try {
             DHeadLocOut.write((StrDHeadLoc + "\n").getBytes());
                    } catch (Exception ex) {}

                                    }
          }
            );


    StopNav.addActionListener(
      new ActionListener() {
        public void actionPerformed(ActionEvent ev) {

//SEND STOP SIGNAL
//SETUP INITIAL COMMS ON STOP PORT
    try {
        StopNavsock = new Socket("131.120.101.81", 2004);
        StopNavIn = new BufferedReader(new InputStreamReader(StopNavsock.getInputStream()));
        StopNavOut = StopNavsock.getOutputStream();
      }  catch (Exception ex) {
        System.err.println("Could not connect to " + "131.120.101.81" + ".          Quitting.");
       System.exit(0);
      }
}
```

```
}
);



/*************************************************************************/

/*This section creates the Buttons which will allow the user
  to choose between pre-programmed patterns. */

//PATTERN 1
    JButton pattern1 = new JButton("Pattern 1");
    pattern1.addActionListener(
      new ActionListener() {
        public void actionPerformed(ActionEvent ev) {

          doPattern1();

        }
      });
    gbcon.gridx = 0;
    gbcon.gridy = 1;
    gblay.setConstraints(pattern1, gbcon);
    s.add(pattern1);

//PATTERN 2
    JButton pattern2 = new JButton("Pattern 2");
    pattern2.addActionListener(
      new ActionListener() {
        public void actionPerformed(ActionEvent ev) {

                    doPattern2();

        }
      });
    gbcon.gridx = 1;
    gbcon.gridy = 1;
    gblay.setConstraints(pattern2, gbcon);
    s.add(pattern2);

    autonomous.add(s, BorderLayout.CENTER);



/*************************************************************************/
/*
    VideoStream vid = new VideoStream("file:SE3015.mpg");
    main.add(vid, BorderLayout.SOUTH);
*/
/*************************************************************************/

/* Tabs are created to display three windows:  Control window, Auto window
  and Status window */
```

```
//TABS SUBROUTINE
    mainPanel.add(tabs, BorderLayout.CENTER);

    tabs.addTab("Manual Control", main);
//    tabs.addTab("Control", new JTextArea());

    tabs.addTab("Robot Location", autonomous);

    tabs.addTab("Status", new JScrollPane(list));

    setContentPane(mainPanel);
    pack();

  }

/************** PATTERN 1 (SOME RANDOM TEST PATTERN ) ****************************/

/* This function sends the commands to move through pattern1 */

        public void doPattern1()
         {

                leftMotor.setValue(FORWARD);
                rightMotor.setValue(FORWARD);

                try {
                // 1000 milliseconds == one second
        Thread.sleep(3000);
        } catch (InterruptedException e){}

                leftMotor.setValue(FORWARD);
                rightMotor.setValue(REVERSE);

                try {
        Thread.sleep(3000);
        } catch (InterruptedException e){}

        leftMotor.setValue(STOP);
                rightMotor.setValue(FORWARD);

                try {
        Thread.sleep(3000);
        } catch (InterruptedException e){}

                leftMotor.setValue(REVERSE);
                rightMotor.setValue(STOP);

                try {
        Thread.sleep(3000);
        } catch (InterruptedException e){}

        leftMotor.setValue(FORWARD);
                rightMotor.setValue(FORWARD);

                try {
        Thread.sleep(3000);
```

```
            } catch (InterruptedException e){}

                        leftMotor.setValue(FORWARD);
                        rightMotor.setValue(REVERSE);

                        try {
            Thread.sleep(3000);
            } catch (InterruptedException e){}

            leftMotor.setValue(STOP);
                        rightMotor.setValue(STOP);

            }
/**************PATTERN 2 (TBD PATTERN )*************************/

/* This function sends the commands to move through pattern2 */

            public void doPattern2() {
            }
/*************************************************************************/


/*This section handles exceptions to the program  */

  public void terminate(){
    try {
       sock.close();
       }
      catch (Exception ex) {}
    System.exit(0);
  }


  public static void main(String args[]){
    if (args.length != 1) {
      System.err.println("usage: java RobotControl <ip address>");
      System.exit(0);
                                    }
          RobotControl url = new RobotControl(args[0]);
     url.setVisible(true);

  }


}
```

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX C. WEB PAGE IN HTML

A. PAGE 1

```
<html>
<head>
<meta http-equiv="Content-Language" content="en-us">
<meta http-equiv="Content-Type" content="text/html; charset=windows-1252">
<meta name="GENERATOR" content="Microsoft FrontPage 4.0">
<meta name="ProgId" content="FrontPage.Editor.Document">
<title>New Page 1</title>
</head>
<body>
<p align="center"><img border="0" src="Flag.gif" width="100" height="55"></p>
<p align="center"><img border="0" src="NPSeal.gif" width="505" height="111" align="left"></p>
<p align="center"><img border="0" src="Herrmann.jpg" width="325" height="183" align="left"></p>
<p align="center"> </p>
<p align="center"> </p>
<p align="center"> </p>
<p align="center"> </p>
<p align="center"> </p>
<p align="center"> </p>
<p align="center"> </p>
<p align="center"> </p>
<p align="center"> </p>
<p align="center"> 

<Center>
<table border=10 cellpadding=0 cellspacing=0>
<tr><th nowrap>
<img align=center src="comsys.gif">
<img border=1 align=center src="physicscard.gif" alt="Our business card"
width=270 height=150 >
<img align=center src="nps.gif"></th></tr>
<p align="center"> </p>
```

```
</table>

</center>

<p align="center">&amp;nbsp;</p>

<p align="center"><b><span style="mso-bidi-font-size: 10.0pt; mso-fareast-font-family: Times New Roman; mso-ansi-language: EN-US; mso-fareast-language: EN-US; mso-bidi-language: AR-SA"><font face="Arial Black" size="5" color="#FF0000">Small

Robotic Technology (SMART) Initiative</font></span></b></p>

<p align="center"><img border="0" src="robot.jpg" width="640" height="317"></p>

<p align="center"><b><font size="5" face="Arial Black" color="#0000FF"><a href="http://131.120.101.81/Robot2.html">ENTER</a></font></b></p>

<p align="center">&amp;nbsp;</p>

<p align="center">&amp;nbsp;</p>


</body>
```

B. PAGE 2

```
<html>

<head>

<meta http-equiv="Content-Type" content="text/html; charset=windows-1252">

<meta name="GENERATOR" content="Microsoft FrontPage 4.0">

<meta name="ProgId" content="FrontPage.Editor.Document">

<title>New Page 1</title>

</head>

<body>

<p align="center"><font size="5" face="Arial Black" color="#0000FF"><b>NAVAL

POSTGRADUATE SCHOOL</b></font></p>

<p align="center"><font size="5" face="Arial Black"><b>PHYSICS
DEPARTMENT</b></font></p>

<p align="center"><b><span style="mso-bidi-font-size: 10.0pt; mso-fareast-font-family: Times
New Roman; mso-ansi-language: EN-US; mso-fareast-language: EN-US; mso-bidi-language: AR-
SA"><font face="Arial Black" size="5" color="#FF0000">Small

Robotic Technology (SMART) Initiative</font></span></b></p>

<hr>

<p align="left"><span style="font-size:12.0pt;mso-bidi-font-size:10.0pt;

font-family:&amp;quot;Times New Roman&amp;quot;;mso-fareast-font-family:&amp;quot;Times New
Roman&amp;quot;;

mso-ansi-language:EN-US;mso-fareast-language:EN-US;mso-bidi-language:AR-SA">The

Naval Postgraduate School’s Small Robotic Technology (SMART) Initiative is an

ongoing research effort within the Combat Systems Science and Technology

Curriculum that engages in forward-looking applications of small robotic

technology for military employment.<span style="mso-spacerun: yes">&amp;nbsp; </span>The

immediate goal of which is to develop a multipurpose robotic platform that is

capable of hosting varied sensor packages for military research.<span style="mso-spacerun:

yes">&amp;nbsp;</span></span></p>

<hr>

<p align="center">&amp;nbsp;</p>

<p align="center"><a href="http://131.120.101.81/Robot3.html">TEAM MEMBERS</a></p>

<p align="center"><a href="http://131.120.101.81/Robot.html">ROBOT OPERATIONS</a></p>

<p align="center"><a href="http://131.120.101.81/Robot4.html">LINKS</a></p>

<p align="center">&amp;nbsp;</p>

<p align="center">&amp;nbsp;</p>
```

```
</body>
</html>
```

C. PAGE 3

```
<html>

<head>

<meta http-equiv="Content-Type" content="text/html; charset=windows-1252">

<meta name="GENERATOR" content="Microsoft FrontPage 4.0">

<meta name="ProgId" content="FrontPage.Editor.Document">

<title>New Page 1</title>

</head>


<body>


<p align="center"><font size="5" face="Arial Black" color="#0000FF"><b>NAVAL

POSTGRADUATE SCHOOL</b></font></p>

<p align="center"><font size="5" face="Arial Black"><b>PHYSICS DEPARTMENT</b></font></p>

<p align="center"><b><span style="mso-bidi-font-size: 10.0pt; mso-fareast-font-family: Times New Roman; mso-ansi-language: EN-US; mso-fareast-language: EN-US; mso-bidi-language: AR-SA"><font face="Arial Black" size="5" color="#FF0000">Small

Robotic Technology (SMART) Initiative</font></span></b></p>

<p align="center"><font face="Arial Black" size="5"><b>Development Team</b></font></p>

<hr>

<p align="left"><font face="Arial Black" size="5" color="#FF0000">Faculty:  

</font><font face="Courier" size="5" color="#FF0000"> </font><font face="Times New Roman" size="5">Professor

Richard Harkins</font></p>

<p align="left">   <font face="Times New Roman" size="5">                       

Professor Thomas Hofler</font></p>

<p align="left">   <font face="Times New Roman" size="5">                       

LCDR Chris Eagle (USN)</font></p>

<hr>

<p align="left"><b><font color="#0000FF" face="Arial Black" size="5">Thesis

Students:   </font></b><font color="#0000FF" face="Times New Roman" size="5">

</font><font face="Times New Roman" size="5"><font color="#0000FF">  </font>Capt.

Todd Ferry (USMC)</font></p>
```

```
<p align="left">   <font face="Times New Roman" size="5">                                              

LT Andrew G. Chicoine (USN)</font></p>

<hr>

<p align="left"><font face="Arial Black" size="5" color="#FF0000"><b>SE3015

Robotic Technology Class (Summer 2001):</b></font></p>

<p align="left"><font face="Times New Roman" size="5">LT Andrew G. Chicoine (USN)

- </font><font face="Times New Roman" size="4">SOFTWARE/HARDWARE INTEGRATION</font></p>

<p align="left"><font face="Times New Roman" size="5">LT Allan Feliciano (USN) -

</font><font face="Times New Roman" size="4">JAVA  </font><font face="Times New Roman" size="5"> 

</font></p>

<p align="left"><font face="Times New Roman" size="5">Maj. Don Carrier (USMC) - </font><font face="Times New Roman" size="4">NAVIGATION

SOFTWARE</font></p>

<p align="left"><font face="Times New Roman" size="5">LT Derek Adametz (USN) - </font><font face="Times New Roman" size="4">HARDWARE</font></p>

<p align="left"><font face="Times New Roman" size="5">LT Jon Bartee (USN) - </font><font face="Times New Roman" size="4">NAVIGATION

SOFTWARE</font></p>

<p align="left"><font face="Times New Roman" size="5">LT Scott McClelland (USN)

- </font><font face="Times New Roman" size="4">HARDWARE/SEISMIC SONAR PLATFORM

DEVELOPMENT</font></p>

<p align="left"><font face="Times New Roman" size="5">LT Marty Wallace (USN) - </font><font face="Times New Roman" size="4">HTML/JAVA</font></p>

<p align="left"><font face="Times New Roman" size="5">LT Toby Lemerande (USN) - </font><font face="Times New Roman" size="4">HARDWARE</font></p>

<p align="left"><font face="Times New Roman" size="5">Capt. Tina Olson (USAF) - </font><font face="Times New Roman" size="4">HARDWARE</font></p>

<p align="left"> </p>

<p align="left"> </p>

<p align="left"> </p>

<p align="left"> </p>

<p align="left"> </p>

<p align="left"> </p>

<p align="left"> </p>
```

```
</body>

</html>
```

D. PAGE 4

```
<html>

<head>

<meta http-equiv="Content-Type" content="text/html; charset=windows-1252">

<meta name="GENERATOR" content="Microsoft FrontPage 4.0">

<meta name="ProgId" content="FrontPage.Editor.Document">

<title>NAVAL POSTGRADUATE SCHOOL</title>

</head>

<body>

<p align="center"> </p>

<p align="center"><font size="5" face="Arial Black" color="#0000FF"><b>NAVAL

POSTGRADUATE SCHOOL</b></font></p>

<p align="center"><font size="5" face="Arial Black"><b>PHYSICS DEPARTMENT</b></font></p>

<p align="center"><b><span style="mso-bidi-font-size: 10.0pt; mso-fareast-font-family: Times New Roman; mso-ansi-language: EN-US; mso-fareast-language: EN-US; mso-bidi-language: AR-SA"><font face="Arial Black" size="5" color="#FF0000">Small

Robotic Technology (SMART) Initiative</font></span></b></p>

<p align="center"><b><span style="mso-bidi-font-size: 10.0pt; mso-fareast-font-family: Times New Roman; mso-ansi-language: EN-US; mso-fareast-language: EN-US; mso-bidi-language: AR-SA"><font face="Arial Black" size="5">LINKS</font></span></b></p>

<hr>

<p align="center"> </p>

<p align="center"><b><font face="Arial Black" size="5"><span style="mso-bidi-font-size: 10.0pt; mso-fareast-font-family: Times New Roman; mso-ansi-language: EN-US; mso-fareast-language: EN-US; mso-bidi-language: AR-SA"><a href="http://www.foster-miller.com/index.htm">PLATFORM</a></span></font></b></p>

<p align="center"><b><font face="Arial Black" size="5"><span style="mso-bidi-font-size: 10.0pt; mso-fareast-font-family: Times New Roman; mso-ansi-language: EN-US; mso-fareast-language: EN-US; mso-bidi-language: AR-SA"><a href="http://www.zworld.com/products/bl2000/index.html">MICROPROCESSOR</a></span></font></b></p>

<p align="center"><b><font face="Arial Black" size="5"><span style="mso-bidi-font-size: 10.0pt; mso-fareast-font-family: Times New Roman; mso-ansi-language: EN-US; mso-fareast-language: EN-US; mso-bidi-language: AR-SA"><a href="http://www.motorola.com/ies/GPS/products/prodm12.html">G.P.S.</a></span></font></b></p>

<p align="center"><b><font face="Arial Black" size="5"><span style="mso-bidi-font-size: 10.0pt; mso-fareast-font-family: Times New Roman; mso-ansi-language: EN-US; mso-fareast-language: EN-US; mso-bidi-language: AR-SA"><a href="http://www.ssec.honeywell.com/magnetic/description/desc_3000.html">COMPASS</a></span></font></b></p>
```

```
<p align="center"><b><font face="Arial Black" size="5"><span style="mso-bidi-font-size:
10.0pt; mso-fareast-font-family: Times New Roman; mso-ansi-language: EN-US; mso-fareast-language:
EN-US; mso-bidi-language: AR-SA"><a href="http://www.nps.navy.mil/">NPS
HOME</a></span></font></b></p>
<p align="center"> </p>
<p align="center"> </p>

</body>

</html>
```

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF REFERENCES

1. Z-World, BL2000 User's Manual, Z-World Inc. 2001

2. Z-World, Dynamic C Premier User's Manual, Z-World Inc. 2001

3. Z-World, An Introduction to TCP/IP, Z-World Inc. 2001

4. Z-World, Dynamic C TCP/IP User's Manual, Z-World Inc. 2001

5. Aitken, P., Jones, B.L., Sams Teach Yourself C in 21 Days, Sams Publishing, 1997.

6. Liang, Y.D., Introduction to JAVA Programming, Que Education and Training, 1999.

7. Overland, B., Morrison, M., JAVA 2 In Plain English, M&T Books, 2001.

8. Z-World Products BL2000, [http://www.zworld.com/products/bl2000/index.html] Nov. 2001.

9. Foster-Miller Mine/Ordnance Robots, [http://www.foster-miller.com/] Nov. 2001.

10. Honeywell Magnetic Sensors Data Sheets, [http://www.ssec.Honeywell.com/magnetic/datasheets.html] Nov. 2001.

11. Motorola Global Positioning Systems-Products, [http://www.Motorola.com/ies/gps/products/prodm12.html] Nov. 2001.

12. RangeLan2 Ethernet Adapter, [http://www.proxim.com/products/all/rangelan2/7920/index.html] Nov. 2001.

13. Novak Super Rooster, [http://www.teamnovak.com/products/esc/super_r/super_ro.htm] Nov. 2001.

14. Ferry, Todd W. *The NPS Small Robotic Technology Initiative, Man-Portable Robots for Low Intensity Conflict*. Master's Thesis, Naval Postgraduate School, Monterey California, June 2000.

15. Space and Naval Warfare Systems Center, San Diego, Advanced Systems Division D37, [http://www.spawar.navy.mil/] May 2001.

16. Coastal Systems Station Panama City, [http://www.ncsc.navy.mil/] May 2001.

17. Mesa Associates, Inc., “Matilda Robotic Platform” [http://www.mesai.com/] May 2001.

18. iRobot Corporation, “ISR&D Programs: Urban Robot” [http://www.irobot.com/rd/] may 2001.

19. IT related Definitions, whatis?com, [http://whatis.techtarget.com/] June 2001.

20. Storey, N. Electronics, A Systems Approach, 2nd Ed., Addison Wesley Longman Ltd. 1998.

21. Trimble Information Services, Inc. “All About GPS”, [http://www.trimble.com/gps/] May 2001.

22. Dana, Peter H., “Global Positioning Overview”, [http://www.colorado.edu/geography/], The Geographer's Craft Project, Department of Geography, The University of Colorado at Boulder, 1994.

23. Coastal Systems Station, Panama City Florida, “VSW/SZ MCM Program: Surf Zone Robotics”, Power Point Slide Presentation, 2000.

24. Sheetz, K.E., *Advancements in Buried Mine Detection Using Seismic Sonar*, Master’s Thesis, Naval Postgraduate School, Monterey California, December 2000.

25. Muir, T. Smith, E., Wilson, S., “Seismo-Acoustic Sonar for Buried Object Detection, Applied Research Laboratories,” paper presented at the Proceedings Mine Warfare Symposium, Naval Postgraduate School, Monterey, California 1996

26. Borenstein, J., Everett, H.R., and Feng, L., *Where am I? Sensors and Methods for Mobile Robot Positioning*. University of Michigan, 1996.

27. Bradley, Mitchell, “Computer Networking- Introduction to Sockets” [http://compnetworking.about.com/compute/compnetworking/library/] Apr 2001.

28. Sun Microsystems, Inc. “The Java Tutorial, All about Sockets” [http://java.sun.com/docs/books/tutorial/networking/sockets/] May 2001.

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center
   8725 John J. Kingman Road, Suite 0944
   Ft. Belvoir, VA 22060-6218

2. Dudley Knox Library
   Naval Postgraduate School
   411 Dyer Road
   Monterey, CA 93943-5101

3. Richard M. Harkins, Code PH/HR
   Department of Physics
   Naval Postgraduate School
   Monterey, CA 93943-5101

4. Thomas J. Hofler, Code PH/HF
   Department of Physics
   Naval Postgraduate School
   Monterey, CA 93943-5101

5. Chairman, Code PH
   Department of Physics
   Naval Postgraduate School
   Monterey, CA 93943-5101

6. Curricular Officer, Code 34
   Engineering and Technology
   Naval Postgraduate School
   Monterey, CA 93943-5107

7. Capt. Todd W. Ferry 1
   Weapons and Systems Engineering Dept.
   United States Naval Academy
   105 Maryland Ave.
   Annapolis, MD 21402

8. Mr. Robin T. Laird (PL-SS) 1
   SPAWARSYSCEN D371
   53406 Woodward Rd. Rm. 108
   San Diego CA 92152-7383

9. Mr. Chuck Bernstein 1
Coastal Systems Station R23
6703 W. Hwy 98
Panama City, FL 32407-7001

10. Mr. Xiaoping Yun, Code EC/YX 1
Department of Electrical and Computer Engineering
Naval Postgraduate School
Monterey, CA 93943

11. LT Andrew G. Chicoine
Surface Warfare Officers School Command
446 Cushing Rd
Newport RI 02841